ഹയർ സെക്കന്ററി കോഴ്സ്

XII

കമ്പ്യൂട്ടർ സയൻസ്

ഭാഗം - I

കേരളസർക്കാർ

പൊതുവിദ്യാഭ്യാസ വകുപ്പ്

സംസ്ഥാന വിദ്യാഭ്യാസ ഗവേഷണ പരിശീലന സമിതി (SCERT); കേരളം

2019

ദേശീയഗാനം

ജനഗണമന അധിനായക ജയഹേ
ഭാരത ഭാഗ്യവിധാതാ,
പഞ്ചാബസിന്ധു ഗുജറാത്ത മറാഠാ
ദ്രാവിഡ ഉത്കല ബംഗാ,
വിന്ധ്യഹിമാചല യമുനാഗംഗാ,
ഉച്ഛല ജലധിതരംഗാ,
തവശുഭനാമേ ജാഗേ,
തവശുഭ ആശിഷ മാഗേ,
ഗാഹേ തവ ജയ ഗാഥാ
ജനഗണമംഗലദായക ജയഹേ
ഭാരത ഭാഗ്യവിധാതാ
ജയഹേ, ജയഹേ, ജയഹേ,
ജയ ജയ ജയ ജയഹേ!

പ്രതിജ്ഞ

ഇന്ത്യ എന്റെ രാജ്യമാണ്. എല്ലാ ഇന്ത്യക്കാരും എന്റെ സഹോദരീ സഹോദരന്മാരാണ്.

ഞാൻ എന്റെ രാജ്യത്തെ സ്നേഹിക്കുന്നു; സമ്പൂർണവും വൈവിധ്യപൂർണവുമായ അതിന്റെ പാരമ്പര്യത്തിൽ ഞാൻ അഭിമാനം കൊള്ളുന്നു.

ഞാൻ എന്റെ മാതാപിതാക്കളെയും ഗുരുക്കന്മാരെയും മുതിർന്നവരെയും ബഹുമാനിക്കും.

ഞാൻ എന്റെ രാജ്യത്തിന്റെയും എന്റെ നാട്ടുകാരുടെയും ക്ഷേമത്തിനും ഐശ്വര്യത്തിനും വേണ്ടി പ്രയത്നിക്കും.

Prepared by:
State Council of Educational Research and Training (SCERT)
Poojappura, Thiruvananthapuram 695012, Kerala
Website : www.scertkerala.gov.in *e-mail* : scertkerala@gmail.com
Phone : 0471 - 2341883, Fax : 0471 - 2341869
Typesetting and Layout : SCERT
© Department of Education, Government of Kerala

ആമുഖം

ഏതു വിജ്ഞാനവും മാതൃഭാഷയിൽ പഠിക്കാനും പ്രകാശനം ചെയ്യാനും സാധിക്കും. അതിനുള്ള അവസരം പഠിതാക്കൾക്ക് ഒരുക്കേണ്ടത്, ഏതൊരു പഠന സമ്പ്രദായത്തിന്റെയും അനിവാര്യതയാണ്. അതിന്റെ തുടക്കമെന്ന നിലയ്ക്കാണ് ഹയർസെക്കന്ററി തലത്തിൽ ഭാഷേതര വിഷയങ്ങളിലെ പാഠപുസ്തകങ്ങൾ മലയാളത്തിൽ പ്രസിദ്ധീകരിക്കുന്നത്.

മാതൃഭാഷയിലൂടെയുള്ള വിദ്യാഭ്യാസം, ജ്ഞാനസമ്പാദനത്തിനുള്ള സുഗമമാർഗം എന്നതിനോടൊപ്പം സാംസ്കാരികതനിമയുടെ തിരിച്ചറിയൽ കൂടിയാണ്. അതുകൊണ്ടാണ് വികസിതരാജ്യങ്ങൾ മാതൃഭാഷയെ മുഖ്യബോധന മാധ്യമമായി സ്വീകരിച്ചിരിക്കുന്നത്. ഇന്ത്യയിലാകട്ടെ, ദേശീയതലത്തിലുള്ള പ്രധാന പരീക്ഷകളെല്ലാം പ്രാദേശിക ഭാഷകളിൽക്കൂടി നടത്തുന്നതിനുള്ള സംവിധാനവും ഉണ്ടായി വരികയാണ്. ഈയൊരു സാഹചര്യത്തിൽ നമ്മുടെ കുട്ടികളും മാതൃഭാഷയുടെ ശക്തിസൗന്ദര്യങ്ങൾ തിരിച്ചറിഞ്ഞ് വിവിധ വിഷയങ്ങളിൽ ജ്ഞാനനിർമിതിയിൽ ഏർപ്പെടേണ്ടതുണ്ട്. അതിന് അവരെ സജ്ജരാക്കുകയാണ് ഈ പാഠപുസ്തകങ്ങളുടെ മുഖ്യലക്ഷ്യം.

പരിഭാഷപ്പെടുത്തിയ പുസ്തകങ്ങളിൽ അതത് വിഷയങ്ങളിലെ സാങ്കേതിക പദങ്ങൾ പരമാവധി മലയാളത്തിലാക്കിയിട്ടുണ്ട്. നമ്മുടെ ഭാഷയിൽ ചിരപരിചിതമായ ഇംഗ്ലീഷ് പദങ്ങളെ അതേപടി സ്വീകരിച്ചിട്ടുമുണ്ട്. വിവർത്തനത്തിന് തീർത്തും വഴങ്ങാത്ത പദങ്ങളെ അതേരീതിയിൽ തന്നെ ഉപയോഗിച്ചിരിക്കുന്നു. മാതൃഭാഷയിൽ പഠിക്കുന്നവർക്ക് ആശയഗ്രഹണം സുഗമമാക്കുന്ന വിധത്തിലാണ് പാഠപുസ്തകരചന നടത്തിയിരിക്കുന്നത്. അതോടൊപ്പം മലയാളഭാഷയുടെ വളർച്ചയ്ക്കും ഈ പ്രവർത്തനം സഹായകമാകുമെന്ന് കരുതുന്നു.

പാഠപുസ്തകവിവർത്തന രംഗത്ത് നമ്മുടെ രാജ്യത്ത് നടന്ന വലിയൊരു കാൽവെപ്പാണ് ഇത്. പ്രഥമ സംരംഭമെന്നനിലയിൽ പല പരിമിതികളും പരിഭാഷയിൽ വന്നിട്ടുണ്ടാകാം. ക്ലാസ്മുറിയിൽ പ്രയോഗത്തിൽ വരുമ്പോഴാണ് അവയെല്ലാം കൂടുതൽ ബോധ്യപ്പെടുക. തുടർന്ന് വരുന്ന ഘട്ടങ്ങളിൽ അവയൊക്കെ പരിഹരിക്കുന്നതിന് എല്ലാ അഭ്യുദയകാംക്ഷികളിൽ നിന്നും വിശിഷ്യ അധ്യാപകർ, വിദ്യാർത്ഥികൾ എന്നിവരിൽ നിന്നും അഭിപ്രായങ്ങളും നിർദേശങ്ങളും പ്രതീക്ഷിക്കുന്നു.

ഡോ. ജെ. പ്രസാദ്

ഡയറക്ടർ,

എസ്.സി.ഇ.ആർ.ടി. കേരളം

പാഠപുസ്തക നിർമാണ സമിതി

ശ്രീ. ജോയ് ജോൺ
എച്ച്.എസ്സ്.എസ്സ്.ടി.,
സെന്റ് ജോസഫ് എച്ച്.എസ്സ്.എസ്സ്. തിരുവനന്തപുരം

ശ്രീ. വിനോദ് വി.
എച്ച്.എസ്സ്.എസ്സ്.ടി., എൻ.എസ്സ്.എസ്സ്
എച്ച്.എസ്സ്.എസ്സ്.,പ്രാകുളം, കൊല്ലം.

ശ്രീ. എ.എസ്.ഇസ്മൈൽ
എച്ച്.എസ്സ്.എസ്സ്.ടി., പി.ജെ.എം.എസ്
ജി.എച്ച്.എസ്സ്.എസ്സ്, കണ്ടശ്ശാൻകടവ്, തൃശ്ശൂർ

ശ്രീ. റോയ് ജോൺ
എച്ച്.എസ്സ്.എസ്സ്.ടി., സെന്റ് അലോഷ്യസ്
എച്ച്.എസ്സ്.എസ്സ്., ഇൽതുരുത്ത്, തൃശ്ശൂർ

ശ്രീ. പ്രശാന്ത് പി.എം
എച്ച്.എസ്സ്.എസ്സ്.ടി.,
സെന്റ് ജോസഫ് ബോയ്സ് എച്ച്.എസ്സ്.എസ്സ്.,
കോഴിക്കോട്

ശ്രീ. സുനിൽ കാര്യത്താൻ
എച്ച്.എസ്സ്.എസ്സ്.ടി., ഗവ. ബ്രിനൻ എൻ.എസ്സ്.എസ്സ്,
തലശ്ശേരി

ശ്രീ. രാജ്മോഹൻ സി.
എച്ച്.എസ്സ്.എസ്സ്.ടി., നവമുകുന്ദ എച്ച്.എസ്.എസ്.
തിരുനാവായ, മലപ്പുറം

ശ്രീ. ടി. മുഹമ്മദ് സലീം
എച്ച്.എസ്സ്.എസ്സ്.ടി., ഓറിയന്റൽ എച്ച്.എസ്സ്.എസ്സ്.,
തിരുവങ്ങാടി, മലപ്പുറം.

വിദഗ്ധർ

ഡോ. ലജീഷ് വി.എൽ
അസ്സി. പ്രൊഫ. ഡിപ്പാർട്ട്മെന്റ് ഓഫ്
കമ്പ്യൂട്ടർ സയൻസ്, കോഴിക്കോട് സർവകലാശാല

ശ്രീ. മധു വി.ടി.
ഡയറക്ടർ, കമ്പ്യൂട്ടർ സെന്റർ, സർവകലാശാല
കോഴിക്കോട്.

ഡോ. സുഷീൽ കുമാർ
അസ്സി. പ്രൊഫ. ഡിപ്പാർട്ട്മെന്റ് ഓഫ് ഇംഗ്ലീഷ്,
ഡി.ബി.കോളജ്, ശാസ്താംകോട്ട

ഡോ. വിനായകുമാരൻ നായർ എൻ.
അസ്സി. കമാൻഡന്റ്, ഹൈ-ടെക് സെൽ, പോലീസ്
ഹെഡ് ക്വോർട്ടേർസ്, തിരുവനന്തപുരം

ഡോ. മധു എസ്സ്. നായർ
അസ്സി. പ്രൊഫ. ഡിപ്പാർട്ട്മെന്റ് ഓഫ്
കമ്പ്യൂട്ടർ സയൻസ്, കേരള സർവകലാശാല

ഡോ. ബിനു പി.ചാക്കോ
അസോ. പ്രൊഫ. ഡിപ്പാർട്ട്മെന്റ് ഓഫ്
കമ്പ്യൂട്ടർ സയൻസ് , പ്രജ്യോതി നികേതൻ കോളജ്,
പുതുക്കാട്, തൃശ്ശൂർ

ഡോ. ദീപ എൽ.സി.
അസ്സി. പ്രൊഫ. ഡിപ്പാർട്ട്മെന്റ് ഓഫ് ഇംഗ്ലീഷ്,
ഗവ.വിമൻസ് കോളജ്, തിരുവനന്തപുരം

ഡോ. കബീർ.വി
അസ്സി. പ്രൊഫ. & ഹെഡ്, ഡിപ്പാർട്ട്മെന്റ് ഓഫ്
കമ്പ്യൂട്ടർ സയൻസ്, ഫറൂക്ക് കോളേജ്, കോഴിക്കോട്

ആർട്ടിസ്റ്റ്

സുധീർ വൈ **വിനീത് വി**

അക്കാദമിക് കോർഡിനേറ്റർ

ഡോ. മീന എസ്.
അസ്സി. പ്രൊഫ., എസ്.സി.ഇ.ആർ.ടി

പാഠപുസ്തക പരിഭാഷ സമിതി
(മലയാളം)

ഡോ. ബിനു പി ചാക്കോ
അസോസിയേറ്റ് പ്രൊഫസർ, ഡിപ്പാർട്ട്മെന്റ് ഓഫ് കമ്പ്യൂട്ടർ സയൻസ്
പ്രജ്യോതി നികേതൻ കോളേജ്, പുതുക്കാട്

ഡോ. ഗ്ലാഡ്സ്റ്റൺ എസ്.രാജ്
അസിസ്റ്റന്റ് പ്രൊഫസർ & ഹെഡ്, ഡിപ്പാർട്ട്മെന്റ് ഓഫ് കമ്പ്യൂട്ടർ സയൻസ്
ഗവ. കോളേജ്, നെടുമങ്ങാട്

ഡോ. എസ്.ഗിരീഷ്കുമാർ
അസിസ്റ്റന്റ് പ്രൊഫസർ, മലയാള വിഭാഗം
ശ്രീ.കേരളവർമ്മ കോളേജ്, തൃശൂർ

ഡോ. പ്രിയ ആർ
അസിസ്റ്റന്റ് പ്രൊഫസർ & ഹെഡ്, ഡിപ്പാർട്ട്മെന്റ് ഓഫ് കമ്പ്യൂട്ടർ സയൻസ്
ഗവ. കോളേജ് കാര്യവട്ടം, തിരുവനന്തപുരം

ആദർശ്.വി.കെ
സീനിയർ മാനേജർ (ടെക്നിക്കൽ), യൂണിയൻ ബാങ്ക് ഓഫ് ഇന്ത്യ

ശ്രീ. അരുൺ പി മാത്യു
എച്ച്.എസ്.എസ്.ടി, എം.ടി.എം.എച്ച്.എസ്.എസ്
പത്തനംതിട്ട

ശ്രീ. ഹരി കെ
എച്ച്.എസ്.എസ്.ടി, ഗവ. വി.& എച്ച്.എസ്.എസ്.
കടപ്പുറം, ചാവക്കാട്, തൃശൂർ

ശ്രീ. മുഹമ്മദ് സർഫറാസ്
എച്ച്.എസ്.എസ്.ടി, സി.എച്ച്.എസ്.എസ്. നെല്ലിപുഴ

ശ്രീ. ഷിനിൽ.പി.പി
എച്ച്.എസ്.എസ്.ടി, പാലോറ എച്ച്.എസ്.എസ്
ഉള്ളിശേരി, കോഴിക്കോട്

ശ്രീ. സുനിൽകുമാർ
എച്ച്.എസ്.എസ്.ടി, ഗവ. എച്ച്.എസ്.എസ്.
തേമ്പാംമൂട്, തിരുവനന്തപുരം

ശ്രീ. വിനയചന്ദ്രൻ സി
എച്ച്.എസ്.എസ്.ടി, വി.ജി.എച്ച്.എസ്.എസ്.
അംബികോദയം എച്ച്.എസ്.എസ്
നെടിയവിള, കൊല്ലം

ശ്രീ. ഷിജു ആർ.
എച്ച്.എസ്.എസ്.ടി.,
ജി.എച്ച്.എസ്.എസ്. വടകര, പുത്തൂർ

അക്കാദമിക് കോർഡിനേറ്റർ

ശ്രീമതി റിയാന അൻസാരി
റിസർച്ച് ഓഫീസർ, എസ്.സി.ഇ.ആർ.ടി

ഉള്ളടക്കം

1.	സ്ട്രക്ചറും പോയിന്ററും	9
2.	ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ്ങിന്റെ തത്വങ്ങൾ	48
3.	ഡാറ്റസ്ട്രക്ചറുകളും പ്രവർത്തനങ്ങളും	71
4.	വെബ് സാങ്കേതികവിദ്യ	101
5.	HTML ഉപയോഗിച്ചുള്ള വെബ്‌രൂപകല്പന	159
6.	ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ചുള്ള ക്ലൈന്റ് സൈഡ് സ്ക്രിപ്റ്റിങ്	206
7.	വെബ് ഹോസ്റ്റിങ്	254

C5T9D2

പാഠപുസ്തകത്തിൽ ഉപയോഗിച്ചിരിക്കുന്ന സൂചനകൾ

നമുക്കു ചെയ്യാം

നിങ്ങളുടെ പുരോഗതി അറിയുക

ഇൻഫർമേഷൻ ബോക്സ്

നമുക്കു പരിശീലിക്കാം

നമുക്കു സംഗ്രഹിക്കാം

J1S8A4

1 സ്ട്രക്ചറും പോയിന്ററും

പ്രധാന പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പഠനം പൂർത്തിയാകുന്നതോടെ പഠിതാവ് ആർജിക്കേണ്ട പഠനനേട്ടങ്ങൾ

- ഉപയോക്താവിനാൽ നിർവചിക്കുന്ന (user defined) ഡാറ്റ ഇനത്തിന്റെ ആവശ്യകത, സ്ട്രക്ചർ ഉപയോഗിച്ചു വിവിധ ഇനം ഡാറ്റ ഏകോപിപ്പിക്കുന്നതിന്റെ സാധ്യത എന്നിവ തിരിച്ചറിയുക.
- സ്ട്രക്ചർ ഡാറ്റ ഇനങ്ങൾ നിർമിച്ച് അതിലെ അംഗങ്ങളെ സൂചിപ്പിക്കുക.
- അടിസ്ഥാന ഡാറ്റകളെയും ഏകോപിത ഡാറ്റകളെയും പ്രതിനിധീകരിക്കുന്നതിൽ നെസ്റ്റഡ് സ്ട്രക്ചറിന്റെ ഉപയോഗം മനസിലാക്കുക.
- ദൈനംദിന ജീവിതത്തിലെ പ്രശ്നങ്ങൾ പരിഹരിക്കാൻ സ്ട്രക്ചർ ഉപയോഗിച്ചുള്ള C++ പ്രോഗ്രാമുകൾ നിർമിക്കുക.
- പോയിന്റർ എന്ന ആശയവും, &, * എന്നീ ഓപ്പറേറ്ററുകളുടെ ഉപയോഗവും വിവരിക്കുക.
- രണ്ട് തരം മെമ്മറി നീക്കിവയ്ക്കലുകൾ താരതമ്യം ചെയ്യുക. ഡൈനാമിക് ഓപ്പറേറ്ററുകളായ `new`, `delete` എന്നീ ഓപ്പറേറ്ററുകളുടെ ഉപയോഗം മനസ്സിലാക്കുക.
- പോയിന്ററുകൾക്കു മേലുള്ള പ്രവർത്തനങ്ങൾ ഉദാഹരണസഹിതം വിവരിക്കുകയും അതിന്റെ ഔട്ട്പുട്ട് നിർവചിക്കുകയും ചെയ്യുക.
- പോയിന്ററും അറെയും തമ്മിലുള്ള ബന്ധം സമർഥിക്കുക.
- സ്ട്രിങ്ങുകൾ കൈകാര്യം ചെയ്യുന്നതിൽ പോയിന്ററിന്റെ ഉപയോഗം അറിയുക.
- സ്വയം സൂചിത (self referential) സ്ട്രക്ചറിന്റെ ആശയം വിവരിക്കുക.

വിവിധ പ്രശ്നങ്ങൾ നിർധാരണം ചെയ്യാനുള്ള C++ പ്രോഗ്രാമുകൾ പരിചയപ്പെട്ടിട്ടുണ്ടല്ലോ. മിക്കവാറും എല്ലാ പ്രശ്നങ്ങളിലും ഡാറ്റ പ്രോസസിങ് ആവശ്യമായി വരാറുണ്ട്. അടിസ്ഥാന ഡാറ്റ വിഭാഗത്തിലുള്ള പൂർണസംഖ്യകൾ (integers), ദശാംശ സംഖ്യകൾ (fractional numbers) കാരക്ടറുകൾ, സ്ട്രിംഗുകൾ എന്നിവയെക്കുറിച്ച് നാം കഴിഞ്ഞ വർഷം മനസ്സിലാക്കി. ഈ ഡാറ്റകളെ സൂചിപ്പിക്കുന്നതിന് വേരിയബിളുകൾ ഉപയോഗിക്കുകയും ഇത്തരം വേരിയബിളുകളെ C++ ലെ അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ ഉപയോഗിച്ച് സ്ഥാപിക്കുകയും ചെയ്തു. എല്ലാ ഡാറ്റയും അടിസ്ഥാന വിഭാഗത്തിൽ ഉൾപ്പെടുന്നില്ല എന്നു നമുക്കറിയാം. ചിലത് വിവിധതരം അടിസ്ഥാന ഡാറ്റ വിഭാഗങ്ങളുടെ ഏകോപനത്തിലൂടെയാണ് നിർമിക്കപ്പെടുന്നത്. എല്ലാത്തരം ഡാറ്റകളെയും ഉൾക്കൊള്ളുന്ന തരത്തിൽ ഡാറ്റ ഇനങ്ങളെ നിർണയിക്കാൻ ഒരു പ്രോഗ്രാമിങ് ഭാഷയ്ക്കും സാധിക്കില്ല. അതുകൊണ്ടുതന്നെ പ്രോഗ്രാമിങ് ഭാഷകളിൽ ഉപയോക്താവിന്റെ ആവശ്യമനുസരിച്ചു പുതിയ ഡാറ്റ ഇനങ്ങൾ നിർമിക്കാനുള്ള സൗകര്യം അനുവദിച്ചിട്ടുണ്ട്. ഈ അധ്യായത്തിൽ നമ്മൾ അത്തരത്തിലുള്ള ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനം (user-defined data type) ആയ സ്ട്രക്ചർ

(structure) എന്ന ഡാറ്റ ഇനത്തെക്കുറിച്ച് ചർച്ച ചെയ്യുന്നു. മറ്റൊരുതരം വേരിയബിളായ പോയിന്ററിനെ (pointer) കുറിച്ചും ഈ അധ്യായത്തിൽ പരിചയപ്പെടാം. പോയിന്റർ എന്ന ആശയം C, C++ എന്നീ ഭാഷകളിലെ പ്രത്യേകതയാണ്. ഇത് മെമ്മറി സ്ഥാനങ്ങളെ നേരിട്ട് അതിന്റെ വിലാസത്തിലൂടെ (Address) പ്രാപ്യമാക്കുകയും അതുവഴി പ്രോഗ്രാം നടപ്പിലാക്കൽ (Execution) വേഗത്തിലാക്കുകയും ചെയ്യുന്നു. ഈ ആശയം നന്നായി മനസ്സിലാക്കിയാൽ നമുക്ക് സിസ്റ്റം ലെവൽ (system level) പ്രോഗ്രാമുകളും ഡാറ്റാ സ്ട്രക്ചർ ആപ്ലിക്കേഷനുകളും എളുപ്പത്തിൽ തയാറാക്കാം.

കഴിഞ്ഞ വർഷം നിങ്ങൾ GNU കമ്പയിലർ കളക്ഷൻ (GCC) ഉൾക്കൊള്ളുന്ന ജിനി IDE യോ ടർബോ C++ IDE യോ C++ പ്രോഗ്രാം നിർമിക്കാൻ ഉപയോഗിച്ചിരിക്കാം. ഈ അധ്യായത്തിൽ C++ പ്രോഗ്രാമുകൾ പൂർണമായും GCC ഉപയോഗിച്ച് ചെയ്യാവുന്ന തരത്തിലാണ് അവതരിപ്പിച്ചിട്ടുള്ളത്.

1.1 സ്ട്രക്ചറുകൾ (Structures)

വിദ്യാർഥികൾ, തൊഴിലാളികൾ, ഉദ്യോഗസ്ഥർ തുടങ്ങിയവർ അവരവരുടെ സ്ഥാപനങ്ങൾ നൽകുന്ന തിരിച്ചറിയൽ കാർഡ് (Identity card) ധരിക്കാറുണ്ട്. ചിത്രം 1.1 കാണിക്കുന്നത് ഒരു കുട്ടിയുടെ തിരിച്ചറിയൽ കാർഡ് ആണ്. പട്ടിക 1.1 ലെ ഒന്നാമത്തെ കോളത്തിൽ കാർഡിൽ പ്രിന്റ് ചെയ്തിരിക്കുന്ന ചില ഡാറ്റകളാണ് കാണിച്ചിരിക്കുന്നത്. പട്ടികയുടെ രണ്ടാമത്തെ കോളം ഈ ഡാറ്റകൾക്കനുയോജ്യമായ C++ ഡാറ്റ ഇനങ്ങൾ ഉപയോഗിച്ച് പൂരിപ്പിക്കുക.

ചിത്രം 1.1: കുട്ടിയുടെ തിരിച്ചറിയൽ കാർഡ്

ഡാറ്റ	C++ ഡാറ്റ ഇനം
12345 Sneha S. Raj 20/02/1997 O+ve Snehanilayam, Gandhi Nagar, Chemmanavattom, Pin 685 531	

പട്ടിക 1.1: ഡാറ്റയും C ഡാറ്റാ ടൈപ്പുകളും

നിങ്ങൾ അഡ്മിഷൻ നമ്പറിന് `short` അല്ലെങ്കിൽ `int` ഡാറ്റ ഇനങ്ങളും പേര് (Sneha S. Raj), രക്തഗ്രൂപ്പ് (blood group, (O +ve)), വിലാസം (അഡ്രസ്സ്) തുടങ്ങിയവയ്ക്ക് `char` അറെയും ഉപയോഗിച്ചിട്ടുണ്ടാവും. ചിലപ്പോൾ നിങ്ങൾക്ക് ജനന തീയതിക്കും (Date of birth) വിലാസത്തിനും ഏറ്റവും യോജിച്ച ഡാറ്റ ഇനം കണ്ടെത്താൻ കഴിഞ്ഞില്ല എന്നു വന്നേക്കാം. നമുക്ക് 20/02/1997 എന്ന തീയതി പരിഗണിക്കാം. ഇത് ഒരു സംയോജിത (compound) ഡാറ്റ ഇനമാണെന്നു എന്ന് വിശകലനം ചെയ്താൽ മനസ്സിലാവും. ഇതിൽ ദിവസസംഖ്യ (20), മാസസംഖ്യ (02), വർഷസംഖ്യ (1997) എന്നിവ

കൂടിച്ചേർന്നിരിക്കുന്നു. മാസസംഖ്യക്കു പകരം മാസത്തിന്റെ പേര് ഉപയോഗിക്കാറുണ്ട്. ഇതുപോലെ വിലാസം എന്നത്, വീട്ടുനമ്പർ, വീട്ടുപേര്, സ്ഥലം, ജില്ല, സംസ്ഥാനം, പിൻകോഡ് എന്നിവയുടെ ഒരു സംയോജിത രൂപമാണ്. ഈ കാർഡിലെ മുഴുവൻ വിവരങ്ങളെയും വേണമെങ്കിൽ ഒരു വലിയ ഏകോപിത ഡാറ്റ ഇനമായി (grouped datatype) പരിഗണിക്കാം. ഇത്തരം ഏകോപിത ഡാറ്റ ഇനത്തിനെ പ്രതിനിധാനം ചെയ്യാൻ ഉപയോക്താവ് നിർവചിക്കുന്ന ഡാറ്റ ഇനത്തെ ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനം (user defined data type) എന്നു വിളിക്കുന്നു. C++ ൽ ഇത്തരം പുതിയ ഡാറ്റ ഇനങ്ങളെ നിർവചിക്കാനുള്ള സൗകര്യമുണ്ട്.

സ്ട്രക്ചർ: - യുക്തിപരമായി പരസ്പരം ബന്ധപ്പെട്ടു നിൽക്കുന്ന ഒരുകൂട്ടം ഡാറ്റ ഇനങ്ങളെ പ്രതിനിധാനം ചെയ്യാൻ C++ ൽ ഉപയോഗിക്കുന്ന പൊതു പേരോടുകൂടിയ ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനമാണ് സ്ട്രക്ചർ. ഈ ഡാറ്റകൾ ചിലപ്പോൾ വ്യത്യസ്തങ്ങളായ ഡാറ്റ ഇനങ്ങളിലുള്ളവ ആയിരിക്കും. 11-ാം ക്ലാസിൽ നമ്മൾ പഠിച്ച 'അറേ' ഒരേ തരത്തിലുള്ള ഡാറ്റയുടെ ഏകോപിതരൂപമാണ്. പക്ഷേ ഒരു പേരിലൂടെ അറിയപ്പെടുന്ന പല തരത്തിലുള്ള ഡാറ്റയുടെ ഏകോപിത രൂപമാണ് സ്ട്രക്ചർ. C++ ൽ സ്ട്രക്ചർ നിർവചിക്കുന്നതും ഉപയോഗിക്കുന്നതും നമുക്ക് ചർച്ച ചെയ്യാം.

1.1.1 സ്ട്രക്ചർ നിർവചനം (Structure definition)

കമ്പ്യൂട്ടർ പ്രോഗ്രാമിലൂടെ പ്രശ്നങ്ങൾ നിർധാരണം ചെയ്യുമ്പോൾ മേൽ വിവരിച്ച പ്രകാരം ഏകോപിത ഡാറ്റ ഇനങ്ങൾ ഉപയോഗിക്കേണ്ടിവരും. ആ സമയത്ത് ഏകോപിത ഡാറ്റ ഇനത്തിന് കൃത്യമായ പേര് നൽകുന്നതോടൊപ്പം, അതിൽ അടങ്ങിയിരിക്കുന്ന ഡാറ്റ ഇനങ്ങൾ കണ്ടെത്തി നിർവചിക്കേണ്ടതുണ്ട്. താഴെ കൊടുത്തിരിക്കുന്ന വാക്യഘടന ഉപയോഗിച്ച് സ്ട്രക്ചറിനെ നിർവചിക്കാം:

```
struct structure_tag
{
    data_type variable1;
    data_type variable2;
    ...................;
    ...................;
    data_type variableN;
};
```

മുകളിൽ ചേർത്ത വാക്യഘടനയിൽ `struct` എന്നത് സ്ട്രക്ചർ നിർവചിക്കാനുള്ള കീവേഡാണ്. `structure_tag or (structure name)` സ്ട്രക്ചറിന്റെ പേര് നിർണയിക്കുന്ന ഐഡന്റിഫയർ ആണ്. `variable1, variable2, ..., variableN` എന്നിവ സ്ട്രക്ചറിലെ അടിസ്ഥാന ഡാറ്റ ഘടകങ്ങളുടെ പേര് നിർണയിക്കുന്ന ഐഡന്റിഫയറുകളാണ്. സ്ട്രക്ചറിന്റെ പേര് സൂചിപ്പിക്കാൻ ഉപയോഗിക്കുന്ന ഐഡന്റിഫയർ ആയ `structure_tag or structure name` ഒരു പുതിയ ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനമാണ്. ഈ ഡാറ്റ ഇനത്തെയും മറ്റേതൊരു ഡാറ്റ ഇനത്തിനും പോലെ നിശ്ചിത

മെമ്മറി ആവശ്യമാണ്. ഇത് പുതിയ വേരിയബിളുകൾ പ്രഖ്യാപിക്കാൻ ചെയ്യാൻ ഉപയോഗിക്കുന്നു. ഈ ഡാറ്റ ഇനത്തിലുള്ള വേരിയബിളുകൾ ഫങ്ഷൻ ആർഗ്യുമെന്റായോ, ഫങ്ഷനിൽനിന്നും റിട്ടേൺ ചെയ്യുന്ന വേരിയബിളായോ ഉപയോഗിക്കാം. സ്ട്രക്ചർ നിർവചനത്തിൽ ബ്രായ്ക്കറ്റിനുള്ളിൽ ({}) കാണുന്ന വേരിയബിളുകളെ സ്ട്രക്ചറിന്റെ മെമ്പർ വേരിയബിളുകൾ എന്നു പറയുന്നു. ഇത്തരം മെമ്പർ വേരിയബിളിന്റെ ഡാറ്റ ഇനം അടിസ്ഥാന ഡാറ്റ ഇനമോ ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനമോ ആവാം. മെമ്പർ വേരിയബിളുകളുടെ വലുപ്പങ്ങളുടെ തുകയാണ് സ്ട്രക്ചർ വേരിയബിളിന്റെ ആകെ വലുപ്പം.

ഇനി നമുക്ക് തിരിച്ചറിയൽ കാർഡിലെ ജനനതീയതിയായ 20/02/1997 പ്രതിനിധീകരിക്കുന്ന ഒരു സ്ട്രകചർ തയാറാക്കി നോക്കാം. ഈ തീയതിയിൽ മൂന്ന് പൂർണ സംഖ്യകൾ ഉണ്ട്. ആദ്യത്തേത് ദിവസസംഖ്യയും രണ്ടാമത്തേത് മാസസംഖ്യയും മൂന്നാമത്തേത് വർഷസംഖ്യയുമാണ്. ചുവടെചേർത്ത രീതിയിൽ നമുക്ക് ഒരു സ്ട്രക്ചർ നിർവചിക്കാം.

```
struct date
{
    int dd;
    int mm;
    int yy;
};
```

ഇവിടെ date എന്നത് സ്ട്രക്ചർ ടാഗിനെ (സ്ട്രക്ച്ചറിന്റെ പേരിനെ) സൂചിപ്പിക്കുന്നു. dd, mm, yy എന്നീ int ഇനത്തിലുള്ള വേരിയബിളുകൾ യഥാക്രമം ദിവസം, മാസം, വർഷം എന്നിവയെ സൂചിപ്പിക്കുന്നു. ഇവ സ്ട്രക്ചറിലെ അംഗങ്ങളാണ്. മാസസംഖ്യയ്ക്ക് പകരം മാസത്തിന്റെ പേര് ചേർക്കണമെങ്കിൽ സ്ട്രക്ചർ നിർവചനത്തിൽ താഴെ കൊടുത്ത രീതിയിൽ മാറ്റം വരുത്തണം.

```
struct strdate
{
    int day;
    char month[10];     // മാസത്തിന്റെ പേര് സ്ട്രിങ്ങാണ്
    int year;
};
```

പ്രോഗ്രാമുകൾ നിർമിക്കുന്ന സമയത്ത് ചില ഡാറ്റകൾ പരസ്പരബന്ധം പുലർത്തുന്നതായി കാണാം. ഈ ഡാറ്റകളെ ഏകോപിപ്പിച്ച് കുറച്ചുകൂടി ഒതുക്കമുള്ള രൂപത്തിലേക്ക് മാറ്റാൻ സ്ട്രക്ചറുകൾ ഉപയോഗിക്കാം. ഉദാഹരണത്തിന് അഡ്മിഷൻ നമ്പർ, പേര്, ഗ്രൂപ്പ്, ഫീസ് തുടങ്ങിയ വിവരങ്ങൾ കുട്ടിയുമായി ബന്ധപ്പെട്ടിരിക്കുന്നു. താഴെ കൊടുത്ത രീതിയിൽ ഒരു സ്ട്രക്ചർ നിർമിക്കാം.

```
struct student
{
    int adm_no;
```

```
        char name[20];
        char group[10];
        float fee;
    };
```

നമുക്കു ചെയ്യാം

ഇനി നിങ്ങൾ സ്വന്തമായി സ്ട്രക്ചർ നിർവചിക്കാൻ ശ്രമിക്കൂ. 'address'ഉം 'blood group' ഉം ആകട്ടെ പുതിയ സ്ട്രക്ചറുകൾ. ബ്ലഡ് ഗ്രൂപ്പിൽ, ഗ്രൂപ്പിന്റെ പേരും, rh വിലയും ഉണ്ടായിരിക്കണം.

തൊഴിലാളിയുടെ വിവരങ്ങളിൽ employee code, name, gender, designation, salary എന്നിവയൊക്കെ പ്രധാനമാണെന്നു നമുക്കറിയാമല്ലോ. ഇവ ഉൾപ്പെടുത്തി അനുയോജ്യമായ ഒരു സ്ട്രക്ചർ നിർവചിക്കുക.

ഇതുവരെ നമ്മൾ ചർച്ച ചെയ്തത് സ്ട്രക്ചർ നിർവചിക്കുന്ന രീതിയാണ്. ഇനി സ്ട്രക്ചർ വേരിയബിളുകൾ എങ്ങനെ പ്രഖ്യാപിക്കാമെന്നും അതിൽ ഡാറ്റ എങ്ങനെ ശേഖരിക്കാമെന്നും മനസ്സിലാക്കാം.

1.1.2 വേരിയബിൾ പ്രഖ്യാപനവും (Declaration) മെമ്മറി നീക്കി വയ്ക്കലും (Allocation)

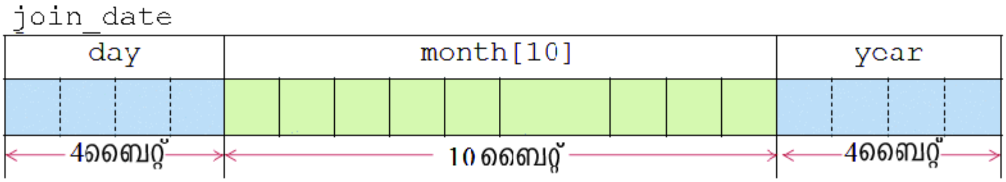
അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളെ പോലെ തന്നെ സ്ട്രക്ചർ ഡാറ്റ ഇനത്തിനും വേരിയബിളുകൾ ഉണ്ടെങ്കിൽ മാത്രമേ ഡാറ്റ ശേഖരിക്കാൻ കഴിയൂ. താഴെ കൊടുത്തിരിക്കുന്ന മാതൃകയിലാണ് വേരിയബിളുകൾ പ്രഖ്യാപിക്കുന്നത്.

```
struct structure_tag var1, var2, ..., varN;
                        OR
structure_tag var1, var2, ..., varN;
```

ഈ വാക്യഘടനയിൽ structure_tag എന്നത് സ്ട്രക്ചറിന്റെ പേരും var1, var2, ..., varN എന്നിവ സ്ട്രക്ചർ വേരിയബിളുകളുമാണ്. നമുക്ക് date, strdate എന്നീ സ്ട്രക്ചറുകളിൽ വേരിയബിളുകൾ പ്രഖ്യാപിക്കാം.

```
date dob, today;       OR    struct date dob, today;
strdate adm_date, join_date;
```

വേരിയബിളിന്റെ പ്രഖ്യാപന പ്രസ്താവന അതിന്റെ വലുപ്പം അനുസരിച്ച് മെമ്മറി നീക്കി വയ്ക്കുന്നു എന്ന് നമുക്കറിയാം. ഒരു സ്ട്രക്ചർ വേരിയബിളിന്റെ വലുപ്പം എന്തായിരിക്കും? സ്ട്രക്ചർ ഒരു ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനമായതിനാൽ അതിന്റെ വലുപ്പം, നിർവചനം അനുസരിച്ചായിരിക്കും. മേൽ സൂചിപ്പിച്ച വേരിയബിൾ പ്രഖ്യാപനത്തിൽ, സ്ട്രക്ച്ചർ date വേരിയബിളുകളായ dob യും today യും 12 byte വീതം വലുപ്പമുള്ളവയായിരിക്കും (മൂന്ന് int ഡാറ്റ ഇനത്തിലുള്ള അംഗങ്ങൾ, GCC യിൽ int ന്റെ വലുപ്പം 4 byte ആയതിനാൽ 3 x 4 = 12 bytes). സ്ട്രക്ചർ strdate ലെ വേരിയബിളായ join_date ന്റെ മെമ്മറി ഘടന ചിത്രത്തിൽ ചേർക്കുന്നു.



ചിത്രം 1.2: സ്ട്രക്ചർ വേരിയബിളിന്റെ മെമ്മറി നീക്കിവയ്ക്കൽ

GCC യിൽ int ടൈപ്പിന്റെ വലുപ്പം 4 ബൈറ്റും Turbo IDE യിൽ 2 ബൈറ്റും ആണ്. ചിത്രം 1.2 ൽ day, year എന്നീ അംഗങ്ങൾ 4 ബൈറ്റും വീതം ഉപയോഗിക്കുന്നു. GCC ഉപയോഗിക്കുന്നതിനാൽ ചിലപ്പോൾ നമ്മൾക്ക് ഇത്രയും മെമ്മറി ആവശ്യമില്ല എങ്കിൽ short ഡാറ്റാ ഇനം ഉപയോഗിക്കുകയാണ് നല്ലത്.

join_date എന്ന വേരിയബിളിൽ day, month, year എന്നീ മൂന്ന് അംഗങ്ങളാണ് ഉള്ളത്. ഇവ യഥാക്രമം 4 ബൈറ്റ്, 10 ബൈറ്റ്, 4 ബൈറ്റ് മെമ്മറി ഉപയോഗിക്കുന്നു. അങ്ങനെ ഈ വേരിയബിളിന്റെ ആകെ മെമ്മറി ഉപയോഗം 18 ബൈറ്റ് ആണ്.

നമുക്കു ചെയ്യാം

ഇനി നമ്മൾ മുമ്പു നിർവചിച്ച സ്ട്രക്ചർ **student** ന്റെ മെമ്മറി വലുപ്പം കണ്ടുപിടിക്കുക. കൂടാതെ **student** സ്ട്രക്ചറിൽ വേരിയബിൾ പ്രഖ്യാപിക്കുക. ഈ വേരിയബിളിന്റെ മെമ്മറി നീക്കി വയ്ക്കുന്നതിന്റെ ചിത്രം വരയ്ക്കുക.

താഴെ കൊടുത്തിരിക്കുന്ന രീതിയിൽ ഒരു സ്ട്രക്ചർ വേരിയബിളിനെ അതിന്റെ നിർവചനത്തിന്റെ കൂടെത്തന്നെ പ്രഖ്യാപിക്കാം.

```
struct complex
{
    short real;
    short imaginary;
}c1, c2;
```

complex എന്നു പേരുള്ള സ്ട്രക്ചർ കോംപ്ലക്സ് സംഖ്യയെ പ്രതിനിധാനം ചെയ്യുന്നു. c1 ഉം c2 ഉം ഈ സ്ട്രക്ചർ തരത്തിലുള്ള രണ്ടു വേരിയബിളുകളാണ്. സ്ട്രക്ചർ നിർവചനത്തിന്റെ കൂടെ വേരിയബിൾ പ്രഖ്യാപിക്കുമ്പോൾ വേണമെങ്കിൽ സ്ട്രക്ചർ ടാഗ് ഒഴിവാക്കുകയും ചെയ്യാം. ചുവടെ കൊടുത്ത ഉദാഹരണം നോക്കുക.

```
struct
{
    int a, b, c;
}eqn_1, eqn_2;
```

ഈ നിർവചനത്തിനു ഒരു പരിമിതിയുണ്ട്. സ്ട്രക്ചർ ടാഗ് ഇല്ലാത്തതിനാൽ വീണ്ടും പുതിയ വേരിയബിളുകൾ നിർമിക്കാൻ ഈ നിർവചനം പര്യാപ്തമല്ല. സ്ട്രക്ചറിലെ അംഗങ്ങൾ ഒരേ തരമാണെങ്കിൽ ഒറ്റവരി പ്രഖ്യാപനമേ ആവശ്യമുള്ളൂ.

വേരിയബിളിന് പ്രാരംഭ വിലനൽകൽ (Veriable Initialisation)

വേരിയബിളുകൾ പ്രഖ്യാപിക്കുമ്പോൾത്തന്നെ അതിലേക്ക് ചില വിലകൾ നൽകാവുന്നതാണ്. ഇത് സ്ട്രക്ചർ വേരിയബിളിന്റെ കാര്യത്തിലും ശരിയാണ്. സ്ട്രക്ചർ വേരിയബിളിന് താഴെ കൊടുത്ത രീതിയിൽ പ്രാരംഭവില നൽകാം.

```
structure_tag variable={value1, value2,..., valueN};
```

ഉദാഹരണമായി student സ്ട്രക്ചറിന് താഴെ കൊടുത്ത രീതിയിൽ പ്രാരംഭവില നൽകാം.

```
student s={3452, "Vaishakh", "Science", 270.00};
```

സ്ട്രക്ചർ അംഗങ്ങളായ വേരിയബിളുകൾക്ക് നിർവചനത്തിൽ നൽകിയിരിക്കുന്ന അതേ ക്രമത്തിലാണ് ബ്രായ്ക്കറ്റിൽ കൊടുത്ത ഡാറ്റകൾ നൽകിയിരിക്കുന്നത്. അതുകൊണ്ടു തന്നെ ഡാറ്റയുടെ ക്രമത്തിലും തരത്തിലും ശ്രദ്ധിക്കേണ്ടതാണ്. മുകളിൽ കൊടുത്ത C++ വാചകം 38 ബൈറ്റ്സ് മെമ്മറി s എന്ന വേരിയബിളിനു വേണ്ടി ഉപയോഗിക്കുന്നു. 3452, "Vaishakh", "Science", 270.00 എന്നീ ഡാറ്റ adm_no, name, group, fee എന്നീ വേരിയബിളുകൾക്ക് യഥാക്രമം നൽകപ്പെടുന്നു (Assign ചെയ്യപ്പെടുന്നു).

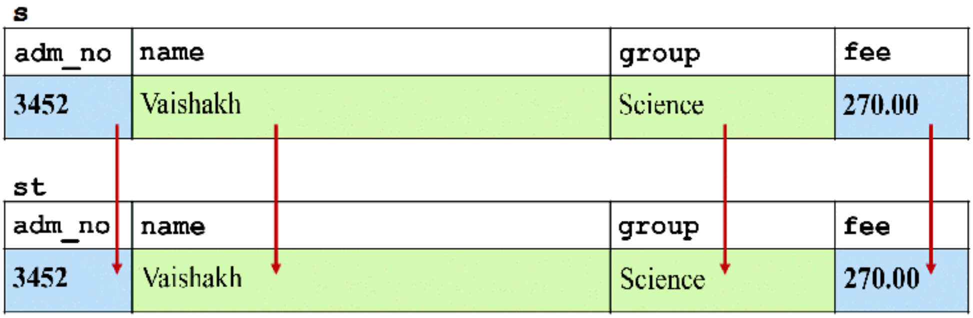
എല്ലാ അംഗങ്ങൾക്കും വിലകൾ നൽകിയില്ലെങ്കിൽ ആദ്യം മുതലുള്ള അംഗങ്ങൾക്ക് ക്രമത്തിൽ വിലകൾ നൽകുകയും, അവസാനം ബാക്കി വരുന്ന അംഗങ്ങൾക്ക് സംഖ്യയാണെങ്കിൽ 0 (zero) യും, സ്ട്രിംഗ് ആണെങ്കിൽ '\0' (നൾ ക്യാരക്ടറും) നൽകുന്നു.

ഒരു സ്ട്രക്ചർ വേരിയബിളിലേക്ക് മറ്റൊരു സ്ട്രക്ചർ വേരിയബിളിന്റെ ഡാറ്റ നൽകാൻ കഴിയും. അങ്ങനെ ചെയ്യണമെങ്കിൽ രണ്ട് വേരിയബിളുകളും ഒരേ സ്ട്രക്ചർ ഇനത്തിലുള്ളവയായിരിക്കണം. ഉദാഹരണം താഴെ ചേർക്കുന്നു.

```
student st = s;
```

ഈ വാചകം വേരിയബിൾ st ലേക്ക് വേരിയബിൾ 's' ന്റെ ഡാറ്റ ശേഖരിക്കുന്നു.

ഇങ്ങനെ വില നൽകിയിരിക്കുന്നത് ചിത്രം 1.3 ൽ കാണിച്ചിരിക്കുന്നു.



ചിത്രം 1.3: സ്ട്രക്ചറിലേക്ക് വില നൽകുന്നു

സ്ട്രക്ചർ നിർവചിക്കുമ്പോൾ അതിനകത്തുള്ള അംഗങ്ങളായ വേരിയബിളുകൾക്ക് പ്രാരംഭവില നൽകാൻ പാടില്ല. കാരണം, സ്ട്രക്ചർ നിർവചനം നടക്കുമ്പോൾ മെമ്മറി നീക്കിവയ്ക്കപ്പെടുന്നില്ല. അതായത്, സ്ട്രക്ച്ചർ നിർവചനത്തിനെ ഒരു വീടിന്റെ ബ്ലൂപ്രിന്റ് (പ്ലാൻ) ആയി കണക്കാക്കാം. പ്ലാനിൽ മുറിയുടെ പേരും രൂപവും കാണാമെങ്കിലും മുറികളിൽ ഒന്നും ശേഖരിക്കാൻ കഴിയില്ല. കാരണം, അവയ്ക്ക് യഥാർഥ സ്ഥലം ഇല്ല എന്നതുതന്നെ. ഒരു വീടിന്റെ ആകെ സ്ഥലം എന്നത് എല്ലാ മുറികളുടേയും സ്ഥലത്തിന്റെ ആകെത്തുകയാണ്. അതുപോലെ സ്ട്രക്ചർ വേരിയബിളിന്റെ മെമ്മറി വലുപ്പം എന്നത് അതിന്റെ അംഗങ്ങളുടെ വലുപ്പത്തിന്റെ തുകയാണ്. ഈ പ്ലാൻകൊണ്ട് ഒരേ രീതിയിലുള്ള കുറേ വീടുകൾ നിർമിക്കാം. വീടുകൾക്ക് വ്യത്യസ്തമായ പേരും സ്ഥലവും ഉണ്ടാകാം. അതുപോലെ സ്ട്രക്ചർ നിർവചനം എന്നത് ബ്ലൂപ്രിന്റും സ്ട്രക്ചർ വേരിയബിൾ എന്നത് അതിന്റെ പൂർത്തീകരണവുമാണ്. ഒരു ബ്ലൂപ്രിന്റ് ഉപയോഗിച്ച് അനേകം വീടുകൾ നിർമിക്കുന്നതുപോലെ സ്ട്രക്ചർ നിർവചനം ഉപയോഗിച്ച് വ്യത്യസ്തമായ പേരിൽ അനേകം വേരിയബിളുകൾ പ്രഖ്യാപിക്കാം. ഓരോന്നിനും വ്യത്യസ്തമായ മെമ്മറി സ്ഥലവും ഉണ്ടായിരിക്കും.

1.1.3 സ്ട്രക്ചർ അംഗങ്ങളെ ഉപയോഗിക്കൽ (Accessing elements of structure)

അറെ (array) എന്നത് ഡാറ്റയുടെ കൂട്ടമാണ്. ഒരു അറെയിലെ അംഗങ്ങളെ സബ്സ്ക്രിപ്റ്റുപയോഗിച്ച് നമുക്ക് ശേഖരിക്കാനും എടുക്കാനും കഴിയും. അതുപോലെ സ്ട്രക്ചർ അംഗങ്ങളെ ഉപയോഗിക്കാൻ വേണ്ടി പിരീഡ് ചിഹ്നം അഥവാ ഡോട്ട് ഓപ്പറേറ്റർ (.) ഉപയോഗിക്കുന്നു. അതിന്റെ വാക്യഘടന ചുവടെ കൊടുത്തിരിക്കുന്നു.

```
structure_variable.element_name
```

പ്രോഗ്രാമുകളിൽ സ്ട്രക്ചർ ഡാറ്റ ഉപയോഗിക്കണമെങ്കിൽ മുകളിൽ കൊടുത്തപോലെ സ്ട്രക്ചർ അംഗങ്ങളെ സൂചിപ്പിക്കണം. ചില ഉദാഹരണങ്ങൾ ചുവടെ കൊടുത്തിരിക്കുന്നു.

```
today.dd = 10;
strcpy(adm_date.month, "June");
cin >> s1.adm_no;
cout << c1.real + c2.real;
```

എന്നാൽ c1+c2 എന്ന പ്രയോഗം സാധ്യമല്ല. കാരണം '+' എന്ന ഓപ്പറേറ്റർ സംഖ്യകളുടെ കൂടെ മാത്രമേ ഉപയോഗിക്കാൻ പാടുള്ളൂ.

ഇനി നമുക്ക് സ്ട്രക്ചർ വേരിയബിളിലെ വിലനൽകൽ (Assignment) പ്രവർത്തനവുമായ ബന്ധപ്പെട്ട രസകരമായ ചില കാര്യങ്ങൾ മനസ്സിലാക്കാം.

താഴെ കൊടുത്തിരിക്കുന്നതുപോലെ രണ്ട് സക്ട്രചറുകൾ നിർവചിച്ചിരിക്കുന്നു.

```
struct test_1
{
    int a;
    float b;
}t1={3, 2.5};
```

```
struct test_2
{
    int a;
    float b;
}t2;
```

മുകളിൽ കൊടുത്ത രണ്ട് സ്ട്രക്ചറുകളിലെയും അംഗങ്ങൾ ഒരേ എണ്ണവും ഒരേ തരവും, ഒരേ പേരുള്ളവയുമാണ്. test_1 ലെ വേരിയബിളായ t1 ലെ അംഗങ്ങളായ a യ്ക്കും b യ്ക്കും 3, 2.5 എന്നീ പ്രാരംഭവിലകൾ യഥാക്രമം നൽകിയിട്ടുണ്ട്. പക്ഷെ എന്നാൽ t2=t1; എന്ന വിലനൽകൽ പ്രസ്താവന തെറ്റാണ്. കാരണം t1 ഉം t2 ഉം രണ്ട് തരത്തിലുള്ള സ്ട്രക്ചർ വേരിയബിളുകളാണ്. പക്ഷെ നമ്മൾക്ക് t1 സ്ട്രക്ചറിലെ വില t2 സ്ട്രക്ചറിൽ നൽകണമെങ്കിൽ ചുവടെ ചേർത്ത രീതി ഉപയോഗിക്കാം.

```
t2.a = t1.a;        t2.b = t1.b;
```

ഇത് സാധ്യമാവാൻ കാരണം, int ടൈപ്പിലുള്ള വേരിയബിളുകളാണ് ഇവിടെ വില നൽകൽ പ്രസ്താവനയിൽ (Assignment statement) ഉപയോഗിച്ചിരിക്കുന്നത്.

ഇനി നമുക്ക് ഇതുവരെ പഠിച്ച കാര്യങ്ങൾ ഉപയോഗിച്ചുള്ള ഒരു പ്രോഗ്രാം പരിചയപ്പെടാം. ഇവിടെ student സ്ട്രക്ചറിൽ രജിസ്റ്റർ നമ്പർ, പേര്, കുട്ടിയുടെ CE മാർക്ക് (തുടർ മൂല്യനിർണയം), PE മാർക്ക് (പ്രാക്ടിക്കൽ മൂല്യനിർണയം), TE (ടേം മൂല്യനിർണയം) എന്നിവ അംഗങ്ങളായി ചേർത്തിട്ടുണ്ട്. സ്ട്രക്ചറിലെ ഈ അംഗങ്ങളുടെ വിലകൾ ഇൻപുട്ടായി നൽകിയാൽ പ്രോഗ്രാം തുടർച്ചയും സമഗ്രവുമായ മൂല്യനിർണയത്തിന്റെ ഭാഗമായുള്ള ആകെ മാർക്ക് കണ്ടുപിടിച്ച് എല്ലാ വിവരങ്ങളും പ്രദർശിപ്പിക്കും.

Program 1.1: കുട്ടിയുടെ ആകെ മാർക്ക് കാണാൻ

```
#include <iostream>
#include <cstdio>       //gets() ഫങ്ഷൻ ഉപയോഗിക്കാൻ
using namespace std;
struct student   //സ്ട്രക്ചർ നിർവചനം തുടങ്ങുന്നു
{
    int reg_no; //  രജിസ്റ്റർ നമ്പർ 32767 ലും കൂടാം, അതുകൊണ്ട് int ഉപ
                    യോഗിക്കുന്നു.
    char name[20];
    short ce;//int 4 ബൈറ്റ് ഉപയോഗിക്കും എന്നാൽ ce സ്കോറിന് ചെറിയ
                    സംഖ്യ മതി
    short pe;
    short te;
};                     //സ്ട്രക്ചർ നിർവചനത്തിന്റെ അവസാനം
int main()
{
    student s;          //സ്ട്രക്ചർ വേരിയബിളിന്റെ പ്രഖ്യാപനം
    int tot_score;
    cout<<"Enter register number: ";
    cin>>s.reg_no;
    fflush(stdin);      //കീബോർഡ് ബഫർ ശൂന്യമാക്കാൻ
    cout<<"Enter name: ";
    gets(s.name);
```

```
    cout<<"Enter scores in CE, PE and TE: ";
    cin>>s.ce>>s.pe>>s.te;
    tot_score=s.ce+s.pe+s.te;
    cout<<"\nRegister Number: "<<s.reg_no;
    cout<<"\nName of Student: "<<s.name;
    cout<<"\nCE Score: "<<s.ce<<"\tPE Score: "<<s.pe
     <<"\tTE Score: "<<s.te;
    cout<<"\nTotal Score     : "<<tot_score;
    return 0;
}
```

പ്രോഗ്രാം 1.1 ന്റെ ഔട്ട്പുട്ട് താഴെ ചേർക്കുന്നു.:

ഔട്ട്പുട്ട്:

```
Enter register number: 23545
Enter name: Deepika Vijay
Enter scores in CE, PE and TE: 19  38  54

Register Number: 23545
Name of Student: Deepika Vijay
CE Score: 19 PE Score: 38    TE Score: 54
Total Score     : 111
```

പ്രോഗ്രാം 1.1 ൽ സ്ട്രക്ചർ നിർവചിച്ചിരിക്കുന്നത് `main()` ഫങ്ഷന് പുറത്താണ്. ഇതുവേണമെങ്കിൽ `main()` ഫങ്ഷന് അകത്തും ചെയ്യാം. നിർവചനത്തിന്റെ സ്ഥാനം സ്ട്രക്ചറിന്റെ പരിധിയും (scope) ആയുസ്സും (life) നിർണയിക്കുന്നു. നമ്മൾ 11-ാം ക്ലാസിലെ 10-ാം അധ്യായത്തിൽ പഠിച്ച ലോക്കൽ, ഗ്ലോബൽ ഡാറ്റയുടെ സാധ്യതകൾ അതായത് സ്ട്രക്ചർ നിർവചനം `main()` ന്റെ ഉള്ളിലായാൽ ആ സ്ട്രക്ചറിന് `main()` ന്റെ ഉള്ളിൽ മാത്രമേ വേരിയബിളുകൾ നിർമിച്ച് ഉപയോഗിക്കാൻ കഴിയൂ. എന്നാൽ സ്ട്രക്ചർ നിർവചനം `main()` ന് പുറത്താണെങ്കിൽ ആ സ്ട്രക്ചറിനെ ഏത് ഫങ്ഷനിലും വേരിയബിളുകൾ നിർമിക്കാൻ ഉപയോഗിക്കാം.

പ്രോഗ്രാം 1.1 `fflush()` എന്ന ഒരു ഫങ്ഷൻ `gets()` ഉപയോഗിക്കുന്നതിന് മുമ്പായി ഉപയോഗിച്ചിട്ടുണ്ട്. ഇതിനു കാരണം പ്രോഗ്രാമിൽ നമ്മൾ `gets()` ന് ശേഷം മറ്റൊരു ഇൻപുട്ട് നൽകുകയാണെങ്കിൽ, തൊട്ടുമുമ്പ് `gets()` ലെ സ്ട്രിംഗിന്റെ റീഡിങ് അവസാനിപ്പിക്കാൻ നാം നൽകിയ `'\n'` ക്യാരക്ടർ (ന്യൂലൈൻ) ഇൻപുട്ട് ബഫറിൽ തന്നെ തങ്ങിനിൽക്കും. ഇതായിരിക്കും അടുത്ത ഇൻപുട്ട് ഡാറ്റയായി സ്വീകരിക്കപ്പെടുക. ഈ പ്രശ്നം ഒഴിവാക്കാൻ `fflush()` ഫങ്ഷനിലൂടെ ഇൻപുട്ട് ബഫർ ശൂന്യമാക്കിയ ശേഷം മാത്രമെ അടുത്ത ഇൻപുട്ട് നൽകാൻ പാടുള്ളൂ.

പ്രോഗ്രാം 1.1 ൽ ഒരു സ്ട്രക്ചർ വേരിയബിൾ മാത്രമേ ഉപയോഗിച്ചിട്ടുള്ളൂ. അതിനാൽ പ്രോഗ്രാമിൽ ഒരു കുട്ടിയുടെ മാർക്ക് വിവരങ്ങൾ മാത്രമെ കൈകാര്യം ചെയ്യാൻ കഴിയൂ. കൂടുതൽ കുട്ടികളുടെ വിവരങ്ങൾ കൈകാര്യം ചെയ്യാൻ നമ്മൾ സ്ട്രക്ചർ വേരിയബിളിന്റെ അറെ (array) ഉപയോഗിക്കേണ്ടതായി വരും. നമുക്ക് മറ്റൊരു ഉദാഹരണത്തിലൂടെ സ്ട്രക്ചറുകളുടെ അറെ പരിചയപ്പെടാം.

പ്രോഗ്രാം 1.2 ൽ ഒരുകൂട്ടം വിൽപ്പനക്കാരുടെ വിവരങ്ങൾ സ്വീകരിക്കുന്നു. സെയിൽസ്മാൻ കോഡ്, പേര്, 12 മാസത്തെ സെയിൽസ് ഡാറ്റ എന്നിവയാണ് സ്ട്രക്ചറിലെ അടിസ്ഥാന വിവരങ്ങൾ. ഈ പ്രോഗ്രാം നൽകിയ വിവരങ്ങളുടെ കൂടെ, ഓരോ വിൽപ്പനക്കാരന്റെയും ശരാശരി വിൽപ്പന എത്രയാണെന്നും പ്രദർശിപ്പിക്കുന്നു. ഈ പ്രോഗ്രാമിൽ ഉപയോഗിക്കുന്ന സ്ട്രക്ചറിൽ ഒരു ഫ്ളോട്ടിംഗ് പോയിന്റ് അറെ ഉപയോഗിച്ചിട്ടുണ്ട്.

Program 1.2 : വിൽപ്പനക്കാരന്റെ ശരാശരി വിൽപന കാണാൻ

```
#include <iostream>
#include <cstdio>
#include <iomanip>    //setw() എന്ന ഫങ്ഷൻ ഉപയോഗിക്കാൻ
using namespace std;
struct sales_data
{
    int code;
    char name[15];
    float amt[12]; //12 മാസത്തെ വിൽപ്പന സംഖ്യ സൂക്ഷിക്കാൻ ഉള്ള അറെ
    float avg;
};
int main()
{
    sales_data s[20]; //സ്ട്രക്ചറിന്റെ അറെ
    short n,i,j; //short മെമ്മറി ഉപയോഗം കുറയ്ക്കുന്നു
    float sum;
    cout<<"Enter the number of salesmen: ";
    cin>>n;
    for(i=0; i<n; i++)
    {
        cout<<"Enter details of Salesman "<<i+1;
        cout<<"\nSalesman Code: ";
        cin>>s[i].code;
        fflush(stdin);
        cout<<"Name: ";
        gets(s[i].name);
        cout<<"Amount of sales in 12 months: ";
        for(sum=0,j=0; j<12; j++)
```

```
        {
            cin>>s[i].amt[j];
            sum=sum+s[i].amt[j];
        }
        s[i].avg=sum/12;
    }
    cout<<"\t\tDetails of Sales\n";
    cout<<"Code\t\tName\t\tAverage Sales\n";
    for(i=0;i<n;i++)
    {
        cout<<setw(4)<<s[i].code<<setw(15)<<s[i].name;
        for (j=0;j<12;j++)
            cout<<setw(4)<<s[i].amt[j];
        cout<<s[i].avg<<'\n';
    }
    return 0;
}
```

നിങ്ങൾ ഈ പ്രോഗ്രാം ലാബിൽ ചെയ്തു നോക്കി അതിന്റെ ഔട്ട്പുട്ട് നിരീക്ഷിക്കുക. ഈ പ്രോഗ്രാമിൽ സ്ട്രക്ചർ അംഗമായി ഫ്ളോട്ടിംഗ് പോയിന്റ് അറെ വേരിയബിൾ ഉപയോഗിച്ചിട്ടുണ്ട്. `int` ഡാറ്റടൈപ്പ് 4 ബൈറ്റ് ഉപയോഗിക്കുന്നതിനാൽ `n`, `i` and `j` എന്നീ വേരിയബിളുകൾ `short` ഡാറ്റ ഇനം ഉപയോഗിച്ചാണ് ഈ പ്രോഗ്രാമിൽ പ്രഖ്യാപിച്ചിരിക്കുന്നത്.

1.1.4 നെസ്റ്റഡ് സ്ട്രക്ചർ (Nested structure)

ഒരു സ്ട്രക്ചറിന്റെ അംഗം മറ്റൊരു സ്ട്രക്ചർ വേരിയബിൾ ആവാം. ഇങ്ങനെയുള്ള സ്ട്രക്ചറിനെ നെസ്റ്റഡ് സ്ട്രക്ചർ എന്നു വിളിക്കുന്നു. ഈ ആശയം വളരെ ശക്തമായ ഡാറ്റാ സ്ട്രക്ചറുകൾ നിർമിക്കാൻ സഹായകമാണ്. നമ്മൾക്ക് പ്രവേശന തീയതി (Admission date) കൂടി `student` സ്ട്രക്ചറിൽ ഉൾപ്പെടുത്തണമെങ്കിൽ, പട്ടിക 1.2 ൽ നൽകിയിരിക്കുന്ന നിർവചനങ്ങളിൽ (നിർവചനം A, നിർവചനം B) ഏതെങ്കിലും ഒന്ന് ഉപയോഗിക്കാം.

നിർവചനം A

```
struct date
{
    short day;
    short month;
    short year;
};
struct student
{
    int adm_no;
    char name[20];
    date dt_adm;
    float fee;
};
```

നിർവചനം B

```
struct student
{
    int adm_no;
    char name[20];
    struct date
    {
        short day;
        short month;
        short year;
    } dt_adm;
    float fee;
};
```

പട്ടിക 1.2 : രണ്ട് തരം നെസ്റ്റിംഗുകൾ

പട്ടിക 1.2 ലെ നിർവചനം A യിൽ രണ്ട് സ്ട്രക്ചറുകളും വെവ്വേറെ നിർവചിച്ചിട്ടുണ്ട്. student എന്ന സ്ട്രക്ചർ dt_adm എന്ന സ്ട്രക്ചർ വേരിയബിളിനെ അംഗമാക്കിയിട്ടുണ്ട്. ഇത് date തരത്തിലുള്ള സ്ട്രക്ചർ വേരിയബിളാണ്.

അകത്തുള്ള സ്ട്രക്ചർ നെസ്റ്റിംഗ് നടത്തുന്നതിന് മുമ്പ് നിർവചിക്കണം എന്നത് ഇവിടെ നമ്മൾ ഉറപ്പ് വരുത്തേണ്ടതാണ്. എന്നാൽ ഡെഫനിഷൻ - B യിൽ സ്ട്രക്ചർ date തന്നെ സ്ട്രക്ചർ studentന് ഉള്ളിൽ നിർവചിച്ചിരിക്കുന്നു. ഈ തരത്തിൽ ഉപയോഗിച്ചാൽ സ്ട്രക്ചർ date ന്റെ പരിധി സ്ട്രക്ചർ studentന് ഉള്ളിൽ മാത്രമായി ചുരുങ്ങും. അതിനാൽ പുറത്തെവിടെയും date സ്ട്രക്ചർ ഇനത്തിൽ വേരിയബിളുകൾ പ്രഖ്യാപിക്കാൻ സാധ്യമല്ല. വേണമെങ്കിൽ അകത്തുള്ള സ്ട്രക്ചർ നിർവചനത്തിലെ ടാഗിന്റെ പേര് ഒഴിവാക്കാം. ഒരു നെസ്റ്റഡ് സ്ട്രക്ചർ വേരിയബിളിന് പ്രാരംഭവില നൽകുന്നത് ചുവടെ ചേർക്കുന്നു.

```
student s = {4325, "Vishal", {10, 11, 1997}, 575};
cout<<s.adm_no<<s.name;
cout<<s.dt_adm.day<<"/"<<s.dt_adm.month<<"/"<<s.dt_adm.year;
```

ഇന്നർ സ്ട്രക്ചർ എലിമെന്റുകളെ ഉപയോഗിക്കാനുള്ള രീതി ചുവടെ ചേർക്കുന്നു.

```
outer_structure_varaiable.inner_structure_variable.element
```

നമുക്കു ചെയ്യാം

എംപ്ലോയികോഡ്, പേര്, പ്രവേശനതീയതി, ജോലി, ശമ്പളം എന്നീ വിവരങ്ങൾ അടങ്ങിയ employee എന്ന സ്ട്രക്ചർ നിർമിക്കുക. employee ടൈപ്പിൽ ഉള്ള വേരിയബിളിന്റെ മെമ്മറി ഘടനവരച്ച് അതിന്റെ വലുപ്പം കണ്ടു പിടിക്കുക.

അറേയും സ്ട്രക്ചറും തമ്മിലുള്ള താരതമ്യം

ഒരു പേരിലൂടെ അനേകം ഡാറ്റയെ സൂചിപ്പിക്കാനുപയോഗിക്കുന്ന ഡാറ്റാ ഇനങ്ങളാണ് സ്ട്രക്ചറും, അറേയും. എന്നാൽ അവ ചില കാര്യങ്ങളിൽ വ്യത്യസ്തമാണ്. അവയെക്കുറിച്ചുള്ള താരതമ്യം ചുവടെ പട്ടികയിൽ നൽകിയിരിക്കുന്നു.

അറേ	സ്ട്രക്ചർ
1. ഇത് ഒരു രൂപീകൃത ഡാറ്റ ഇനമാണ് (Derived data type)	1. ഇത് ഒരു ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനമാണ്.
2. ഒരേ ഇനം ഡാറ്റയുടെ കൂട്ടമാണ്.	2. വിവിധ ഇനം ഡാറ്റകളുടെ കൂട്ടമാണ്.
3. അറേയിലെ അംഗങ്ങളെ സൂചിപ്പിക്കുന്നത് സബ്സ്ക്രിറ്റ് ഉപയോഗിച്ചാണ്.	3. സ്ട്രക്ചറിലെ അംഗങ്ങളെ സൂചിപ്പിക്കുന്നത് ഡോട്ട് ഓപ്പറേറ്റർ (.) ഉപയോഗിച്ചാണ്.
4. അറേയിലെ ഒരു അംഗം മറ്റൊരു അറേയെ പ്രതിനിധാനം ചെയ്യുമ്പോൾ മൾട്ടി ഡയമൻഷണൽ അറേ രൂപീകരിക്കപ്പെടുന്നു.	4. സ്ട്രക്ചറിലെ ഒരു അംഗം മറ്റൊരു സ്ട്രക്ചറിനെ പ്രതിനിധാനം ചെയ്യുന്നു എങ്കിൽ അത് നെസ്റ്റഡ് സ്ട്രക്ചറിനെ രൂപീകരിക്കുന്നു.
5. സ്ട്രക്ചറിന്റെ അറേ നിർമിക്കൽ സാധ്യമാണ്.	5. സ്ട്രക്ചറിലെ അംഗമായി അറേ ഉപയോഗിക്കാം.

പട്ടിക 1.3: അറേയും സ്ട്രക്ചറും തമ്മിലുള്ള താരതമ്യം

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. സ്ട്രക്ചർ എന്നാൽ എന്ത്?
2. സ്ട്രക്ചർ വിവിധതരം ഡാറ്റകളെ ഒരു യൂണിറ്റായി ഏകോപിപ്പിക്കുന്നു - ശരി/തെറ്റ്.
3. സ്ട്രക്ചർ അംഗത്തെ ഉപയോഗിക്കുന്നതിന് താഴെ കൊടുത്തവയിൽ ഏതാണ് ശരി.
   a. `struct.element`
   b. `structure_tag.element`
   c. `structure_variable.element`
   d. `structure_tag.structure_variable`
4. നെസ്റ്റഡ് സ്ട്രക്ചർ എന്നാൽ എന്ത്? ഉദാഹരണമെഴുതുക.
5. അറേയ്ക്ക് സബ്സ്ക്രിപ്റ്റ് ആണെങ്കിൽ സ്ട്രക്ചറിന് __________ ആണ്.

1.2 പോയിന്ററുകൾ (Pointers)

നമുക്ക് 'കമ്പ്യൂട്ടിങ്ങിലെ മുന്നേറ്റങ്ങൾ' എന്ന വിഷയത്തിൽ ഒരു ഉപന്യാസം തയാറാക്കണമെന്ന് കരുതുക. നമുക്ക് അനുയോജ്യമായ പുസ്തകങ്ങൾ നാം പരിശോധിക്കേണ്ടതായി വരും. തീർച്ചയായും നമ്മൾ ലൈബ്രറിയിൽ പരതി നോക്കും. നമ്മൾക്ക് പുസ്തകം കണ്ടു പിടിക്കാൻ കഴിഞ്ഞില്ലെങ്കിൽ ലൈബ്രേറിയനോ, കമ്പ്യൂട്ടർ സയൻസ് അധ്യാപകനോ

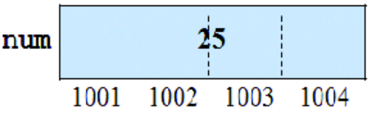
നമ്മളെ ഇക്കാര്യത്തിൽ സഹായിക്കും. ഇവിടെ നമുക്ക് ലൈബ്രേറിയന്റെ/അധ്യാപകന്റെ പങ്കെന്താണെന്ന് പരിശോധിക്കാം. അദ്ദേഹം എല്ലായ്പ്പോഴും ഒരു റഫറൻസാണ്. ലൈബ്രറിയിലെവിടേയോ ഉള്ള പുസ്തകം കണ്ടെത്തി നമുക്ക് തരാൻ അദ്ദേഹം സഹായിക്കുന്നു.

ചിത്രം 1.4: റഫറൻസിന്റെ ഉദാഹരണം

മുകളിൽ കൊടുത്ത ഉദാഹരണത്തിലെ ലൈബ്രേറിയന്റെ/അധ്യാപകന്റെ പങ്കാണ് പോയിന്റർ വേരിയബിളിനും ഉള്ളത്. പോയിന്റർ കമ്പ്യൂട്ടർ മെമ്മറിയിലേക്ക് ഉള്ള ഒരു റഫറൻസ് ആണ്. താഴെ കൊടുത്ത വാചകം പരിഗണിക്കുക. ചിത്രം 1.4 : റഫറൻസിന്റെ ഉദാഹരണം.

```
int num=25;
```

ഇത് ഒരു പ്രാരംഭവിലനൽകൽ പ്രസ്താവനയാണെന്ന് നമുക്കറിയാം. ഇതിൽ num എന്ന വേരിയബിൾ പ്രഖ്യാപിക്കുന്നതോടൊപ്പം അതിലേക്ക് 25 എന്ന സംഖ്യ ശേഖരിക്കുകയും ചെയ്യുന്നു. ഇവിടെയുള്ള മെമ്മറി നീക്കിവയ്ക്കലിന്റെ ചിത്രം 1.5 ൽ കാണിച്ചിരിക്കുന്നു. ചിത്രത്തിൽ നിന്നും നമുക്ക് വേരിയബിളിന്റെ മൂന്ന് സ്വഭാവ ഘടകങ്ങൾ മനസ്സിലാക്കാം - പേര്, വിലാസം (Address), ഉള്ളടക്കം (Content) എന്നിവയാണ് ആ ഘടകങ്ങൾ. ഇവിടെ വേരിയബിളിന്റെ പേര് num, അതിന്റെ ഉള്ളടക്കം 25. num ഒരു int ഇനത്തിലുള്ള വേരിയബിളാണ്. ഇതിനു വേണ്ടി 4 ബൈറ്റ് മെമ്മറിയാണ് ഉപയോഗിക്കുന്നത് (GCCയിൽ). RAM ലെ ഓരോ സെല്ലിലും 1 ബൈറ്റ് മെമ്മറിയാണ് ഉപയോഗിക്കുന്നത്. ഓരോ സെല്ലിനും വ്യത്യസ്തമായ വിലാസം ഉണ്ട്. ഒന്നിലധികം സെല്ലുകൾ ചേർന്ന് ഒരു സ്റ്റോറേജ് ലൊക്കേഷൻ ആയി ഉപയോഗിക്കുമ്പോൾ അതിനെ മെമ്മറി വേഡ് എന്നു പറയുന്നു. 1001, 1002, 1003, 1004 എന്നീ മെമ്മറി സ്ഥാനങ്ങൾ (Locations) വേരിയബിൾ ഉപയോഗിക്കുന്നു. ഇതിൽ തുടക്കത്തിലെ സെല്ലിന്റെ വിലാസമായ (ബേസ് അഡ്രസ്സ്) 1001 ആണ് നമ്മൾ num എന്ന വേരിയബിളിന്റെ വിലാസമായി പരിഗണിക്കുന്നത്. 11-ാം ക്ലാസ്സിൽ നമ്മൾ പഠിച്ച ഒരു കാര്യം ഓർമയുണ്ടല്ലോ? ഒരു വേരിയബിളിന് L വില, R വില എന്നിങ്ങനെ രണ്ട് വിലകൾ ഉണ്ട്. ഇതിൽ L വില വിലാസത്തെയും, R വില ഉള്ളടക്കത്തേയും പ്രതിനിധീകരിക്കുന്നു. ചിത്രം 1.5 ൽ num എന്ന വേരിയബിളിനു L വില 1001 ഉം R വില 25 ഉം ആണ്.



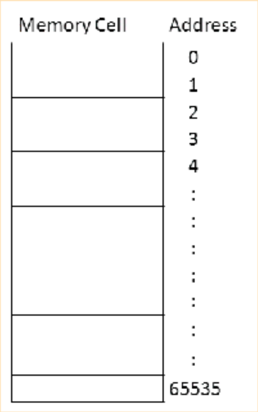
ചിത്രം 1.5: മെമ്മറി നീക്കിവയ്ക്കൽ

നമുക്ക് ഒരു വേരിയബിളിന്റെ വിലാസം (L-value) മറ്റൊരു വേരിയബിളിൽ ശേഖരിക്കണം എന്ന് കരുതുക. ഇതിനുപയോഗിക്കുന്ന വേരിയബിളിനെ പോയിന്റർ വേരിയബിൾ എന്നു പറയുന്നു. ഇപ്രകാരം നമുക്ക് പോയിന്ററിനെ (Pointer) മെമ്മറി വിലാസം സൂക്ഷിക്കുന്ന ഒരു വേരിയബിളായി നിർവചിക്കാം. പോയിന്റർ ഒരു അടിസ്ഥാന ആശയമാണ്. കാരണം ഇത് മെമ്മറി വിലാസത്തിനെ അതിന്റെ യഥാർഥ രൂപത്തിൽ (Atomic) ഉൾക്കൊള്ളുന്നു. അതുകൊണ്ട് പോയിന്റർ എന്നാൽ മെമ്മറി സ്ഥാനത്തേക്ക്-അതിന്റെ ഉള്ളടക്കത്തിലേക്ക് - ചൂണ്ടുന്ന ഒരു വേരിയബിളായി നമുക്ക് പറയാം.

നമ്മൾ ഒരു വേരിയബിളിന്റെ മെമ്മറി വിലാസം പ്രദർശിപ്പിച്ചു നോക്കിയാൽ ഹെക്സാഡെസിമൽ സംഖ്യകളായ 0x7ffe69878784, 0x7ffe69878785 തുടങ്ങിയവയായിരിക്കും ലഭിക്കുക. ഇതിനുകാരണം ഇക്കാലത്ത് മെമ്മറി വിലാസം സൂചിപ്പിക്കുന്നത് ഹെക്സാ ഡെസിമൽ നമ്പറുകളിലാണ് എന്നതാണ്. എന്നാൽ ഈ പുസ്തകത്തിൽ സൗകര്യത്തിനായി മെമ്മറി വിലാസങ്ങൾ 1001, 1002 തുടങ്ങിയ ഡെസിമൽ പൂർണസംഖ്യകളുപയോഗിച്ച് രേഖപ്പെടുത്തിയിരിക്കുന്നു (ചിത്രം 1.5 ശ്രദ്ധിക്കുക).

1964 ലെ പോയിന്ററിന്റെ കണ്ടുപിടുത്തം അറിയപ്പെടുന്നത് ഹെറോൾഡ് ലോസൻ (ജനിച്ചത് 1937) എന്ന സോഫ്റ്റ് വെയർ എഞ്ചിനീയറുടെ (ഇദ്ദേഹം ഒരു സിസ്റ്റം എഞ്ചിനീയറും കംപ്യൂട്ടർ ആർക്കിടെക്റ്റുമായിരുന്നു) പേരിലാണ്. 2000 ത്തിൽ, ലോസൻ ഈ കണ്ടുപിടുത്തത്തിന്റെ പേരിൽ IEEE യുടെ കംപ്യൂട്ടർ പയനിയർ അവാർഡ് നേടുകയുണ്ടായി.

പ്രോഗ്രാം നിർദേശങ്ങളും, വേരിയബിളിന്റെ വിലകളും ശേഖരിക്കാനാണ് കംപ്യൂട്ടർ അതിന്റെ മെമ്മറി ഉപയോഗിക്കുന്നത് എന്ന് നിങ്ങൾക്കറിയാം. മെമ്മറി എന്നത് ചിത്രം 1.6 കാണിച്ചതുപോലെ സ്റ്റോറേജ് സെല്ലുകളുടെ തുടർച്ചയായ ഒരു ശേഖരമാണ്. സാധാരണമായി ഓരോ സെല്ലിനും ഒരു ബൈറ്റ് വലുപ്പമാണ് ഉള്ളത്. ഓരോ സെല്ലിനും അതിന്റേതായ മെമ്മറി വിലാസം ഉണ്ടായിരിക്കും. വിലാസങ്ങൾ പൂജ്യത്തിൽ തുടങ്ങി തുടർച്ചയായി നമ്പർ ചെയ്തിരിക്കും. അവസാനത്തെ മെമ്മറിസെല്ലിന്റെ വിലാസം മെമ്മറിയുടെ വലുപ്പത്തെ ആശ്രയിച്ചിരിക്കും. 64 K (64 x 1024 = 65536 ബൈറ്റ്) മെമ്മറി വലുപ്പം ഉള്ള ഒരു കംപ്യൂട്ടറിന്റെ മെമ്മറിയുടെ അവസാനത്തെ വിലാസം 65,535 ആയിരിക്കും.



ചിത്രം 1.6: മെമ്മറിയുടെ ഘടന

നമ്മൾ ഒരു വേരിയബിളിനെ പ്രഖ്യാപിക്കുമ്പോൾ ആ വേരിയബിളിനു വേണ്ടി മെമ്മറിയിലെവിടെയോ ഒരു സ്ഥലം സൃഷ്ടിക്കപ്പെടുന്നു. ഈ സ്ഥലത്താണ് അതിന്റെ R വില (ഉള്ളടക്കം) ശേഖരിക്കപ്പെടുന്നത്. ഓരോ വേരിയബിളിനും അതിന്റേതായ വിലാസം ഉണ്ടായിരിക്കും. ഇക്കാലത്ത് RAMന്റെ വലുപ്പം GBയിലും, മെമ്മറി സ്ഥാനത്തിന്റെ വിലാസം ഹെക്സാഡെസിമൽ നമ്പറിലും ആണ് രേഖപ്പെടുത്തുന്നത്. ഇതിനുകാരണം ഡെസിമൽ നമ്പർസിസ്റ്റത്തെ അപേക്ഷിച്ച് ഹെക്സാഡെസിമൽ നമ്പർ സിസ്റ്റത്തിന് കുറഞ്ഞ അക്കങ്ങൾ ഉപയോഗിച്ച് വലിയ സംഖ്യകൾ രേഖപ്പെടുത്താൻ കഴിയും എന്നതാണ്.

1.2.1 പോയിന്റർ വേരിയബിളിന്റെ പ്രഖ്യാപനം (Declaration of pointer variable)

പോയിന്റർ ഒരു രൂപീകൃത ഡാറ്റ ഇനമാണ് (Derived data type). അതുകൊണ്ടുതന്നെ പ്രോഗ്രാമിൽ ഉപയോഗിക്കുന്നതിന് മുമ്പ് പോയിന്റർ വേരിയബിളുകളെ പ്രഖ്യാപിക്കേണ്ടതായിട്ടുണ്ട്. താഴെ കൊടുത്ത വാക്യഘടനയിലാണ് പോയിന്റർ വേരിയബിൾ പ്രഖ്യാപിക്കുന്നത്.

```
data_type * variable;
```

ഇവിടെ ഡാറ്റ ഇനം എന്നത് അടിസ്ഥാന ഡാറ്റ ഇനമോ, ഉപയോക്തൃ നിർവചിത ഡാറ്റ

ഇനമോ ആവാം. variable എന്നാൽ ഐഡന്റിഫയർ ആണ്. ഡാറ്റ ഇനത്തിനും വേരിയബിളിനും ഇടയിലുള്ള asterisk (*) ചിഹ്നം ശ്രദ്ധിക്കുക. ഇത് പോയിന്റർ വേരിയബിളിന്റെ പ്രഖ്യാപനത്തിലെ പ്രത്യേകതയാണ്. ചുവടെ ചേർത്തവ പോയിന്റർ വേരിയബിൾ പ്രഖ്യാപനത്തിന്റെ ഉദാഹരണങ്ങളാണ്.

```
int *ptr1;
float *ptr2;
struct student *ptr3;
```

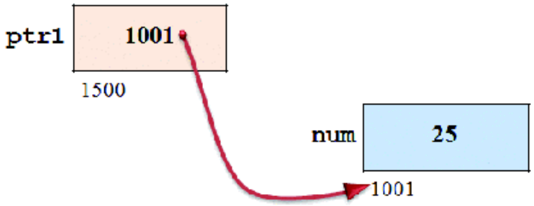
സാധാരണ പോലെ പോയിന്റർ വേരിയബിളുകൾക്കും മെമ്മറി നീക്കിവയ്ക്കൽ നടക്കുന്നു. പോയിന്ററിന് ആവശ്യമായ മെമ്മറി സ്ഥാനത്തിന്റെ വലുപ്പം ഡാറ്റാതരത്തിന് അനുസരിച്ചാണെന്നു നിങ്ങൾ വിചാരിക്കുന്നുണ്ടോ? നമുക്കറിയാം മെമ്മറി വിലാസത്തിന്റെ തരം അൺസൈൻഡ് ഇന്റിജർ ആണ്. ഇതിനർഥം എല്ലാ പോയിന്ററും അൺസൈൻഡ് ഇന്റിജർ ഇനത്തിൽ പ്രഖ്യാപിക്കണം എന്നല്ല. പിന്നെ എന്താണ് പോയിന്ററിന്റെ ഡാറ്റ ഇനം തീരുമാനിക്കുന്നതിലെ മാനദണ്ഡം? ഒരു പോയിന്റർ വേരിയബിളിന്റെ ഡാറ്റ ഇനം അത് പോയിന്റ് ചെയ്യുന്ന ഡാറ്റയുടെ ഇനം തന്നെയാണ്. മുകളിൽ കൊടുത്ത ഉദാഹരണത്തിൽ ptr1 എന്ന വേരിയബിളിൽ ഒരു ഇന്റിജർ മെമ്മറി സ്ഥാനത്തിന്റെ വിലാസമാണ് സൂക്ഷിക്കപ്പെടുന്നത്. അതുപോലെ ptr2 ഒരു ഫ്ളോട്ടിങ് പോയിന്റ് ലൊക്കേഷന്റെയും ptr3 student ഇനത്തിലുള്ള സ്ട്രക്ചറിന്റെയും വിലാസം ഉൾക്കൊള്ളുന്നു. കൂടുതൽ മെമ്മറി സ്ഥാനങ്ങൾ ഉള്ള ഡാറ്റ ഇനങ്ങളെ പോയിന്റ് ചെയ്യുമ്പോൾ അതിന്റെ തുടക്കത്തിലെ സ്ഥാനത്തിന്റെ വിലാസമാണ് (Base Address) പോയിന്റർ വേരിയബിളിൽ ശേഖരിക്കപ്പെടുന്നത്. അപ്പോൾ ഒരു പോയിന്റർ വേരിയബിളിന്റെ മെമ്മറി വലുപ്പം എന്തായിരിക്കും? ഈ വലുപ്പം കമ്പ്യൂട്ടറിലെ വിലാസ സമ്പ്രദായത്തിനനുസരിച്ചായിരിക്കും (Addressing scheme). സാധാരണയായി C++ ൽ ഒരു പോയിന്റർ വേരിയബിളിന്റെ മെമ്മറി വലുപ്പം 2 മുതൽ 4 ബൈറ്റ് വരെ ആയിരിക്കും. ഒരു പ്രോഗ്രാമറെ സംബന്ധിച്ചിടത്തോളം പ്രശ്നങ്ങൾ നിർധാരണം ചെയ്യുമ്പോൾ പോയിന്ററിന്റെ വലുപ്പം അറിയണമെന്ന് നിർബന്ധമില്ല.

1.2.2 &, * എന്നീ ഓപ്പറേറ്ററുകൾ (The Operators & and *)

ഒരിക്കൽ ഒരു പോയിന്റർ വേരിയബിൾ പ്രഖ്യാപിച്ചാൽ അതേ ഡാറ്റ ഇനത്തിലുള്ള മെമ്മറി ലൊക്കേഷന്റെ വിലാസം മാത്രമേ അതിൽ ശേഖരിക്കാൻ കഴിയൂ. സാധാരണ C++ ൽ ഒരു വേരിയബിളിന്റെ പേര് നൽകിയാൽ അതിന്റെ R വില മാത്രമേ സൂചിപ്പിക്കപ്പെടുന്നുള്ളൂ. അപ്പോൾ എങ്ങനെയാണ് L വില അഥവാ വിലാസം ലഭിക്കുന്നത്? C++ ൽ ഇതിനുവേണ്ടി അഡ്രസ്സ് ഓപ്പറേറ്റർ (&) ഉപയോഗിക്കുന്നു. ഉദാഹരണമായി num ഒരു ഇന്റിജർ വേരിയബിളാണെങ്കിൽ അതിന്റെ വിലാസം താഴെ കൊടുത്ത രീതിയിൽ `ptr1` എന്ന പോയിന്ററിൽ ശേഖരിക്കാം.

```
ptr1 = &num;
```

പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുന്ന സമയത്ത് ഈ വാചകം `ptr1` എന്ന പോയിന്റർ വേരിയബിളിനും num എന്ന മെമ്മറി സ്ഥാനത്തിനും ഇടയിൽ ഒരു ബന്ധം സ്ഥാപിക്കുന്നു. ഇത് ചിത്രം 1.7 ൽ കാണിച്ചിരിക്കുന്നു.



ചിത്രം 1.7 പോയിന്ററും അത് ചൂണ്ടുന്ന ലൊക്കേഷനും

പോയിന്റർ ഒരു റഫറൻസ് ആണെന്നു നാം മനസ്സിലാക്കി. അതുകൊണ്ട് മെമ്മറിയിൽ എവിടെ ഡാറ്റ ശേഖരിച്ചാലും അത് സൂചിപ്പിക്കാൻ (refer) പോയിന്റർ വേരിയബിളിനു കഴിയും. പോയിന്റർ വേരിയബിളിന്റെ ഡീറഫറൻസിലൂടെ നമുക്ക് ഡാറ്റ ലഭ്യമാവും. ഇതിനുവേണ്ടി ഇൻഡയറക്ഷൻ ഓപ്പറേറ്റർ (indirection operator) അഥവാ (*) ഉപയോഗിക്കുന്നു. ഇതിനെ ഡീറഫറൻസ് ഓപ്പറേറ്റർ (dereference operator) എന്നും പറയുന്നു. താഴെ കൊടുത്ത C++ പ്രസ്താവന ptrl എന്ന പോയിന്റർ വേരിയബിൾ ചൂണ്ടുന്ന ലൊക്കേഷനിലെ വില പ്രദർശിപ്പിക്കുന്നു.

```
cout << *ptr1;
```

ഈ പ്രസ്താവന `cout << num;` എന്ന പ്രസ്താവനയ്ക്ക് തുല്യമാണെന്ന് വ്യക്തമാണല്ലോ.

* ഓപ്പറേറ്റർ ഒരു പോയിന്റർ ലൊക്കേഷനിലെ വില തിരിച്ച് തരുന്നതു കൊണ്ട് 'വാല്യു അറ്റ് ഓപ്പറേറ്റർ' എന്നും അറിയപ്പെടുന്നു.

& ഓപ്പറേറ്ററും * ഓപ്പറേറ്ററും യൂനറി ഓപ്പറേറ്ററുകളുമാണ്. എല്ലാ വേരിയബിളിനും മെമ്മറി സ്ഥാനവും വിലാസവും ഉള്ളതിനാൽ & ഓപ്പറേറ്റർ എല്ലാ വേരിയബിളിന്റെയും കൂടെ ഉപയോഗിക്കാം. എന്നാൽ * ഓപ്പറേറ്റർ പോയിന്റർ വേരിയബിളിന്റെ കൂടെ മാത്രമേ ഉപയോഗിക്കാൻ കഴിയുകയുള്ളൂ.

ചിത്രം 1.7 ൽ കൊടുത്ത വേരിയബിൾ പരിശോധിക്കൂ. ചുവടെ കൊടുത്ത ഉദാഹരണങ്ങൾ &, * ഓപ്പറേറ്ററുകളുടെ ഉപയോഗം കാണിക്കുന്നു.

```
cout<< &num; // 1001 num ന്റെ വിലാസം പ്രദർശിപ്പിക്കുന്നു
cout<< ptr1; // 1001 ptr1 ന്റെ ഉള്ളടക്കം പ്രദർശിപ്പിക്കുന്നു
cout<< num;  // 25 num ന്റെ ഉള്ളടക്കം പ്രദർശിപ്പിക്കുന്നു
cout<< *ptr1;/* 25 ptr1 പോയിന്റ് ചെയ്യുന്ന സ്ഥാനത്തെ വില പ്രദർശിപ്പി
ക്കുന്നു
cout<< &ptr1;// 1500 (ptr1 ന്റെ വിലാസം പ്രദർശിപ്പിക്കുന്നു.
cout<< *num; // Error!! num എന്നത് പോയിന്റർ വേരിയബിളല്ല.
```

അവസാനത്തെ പ്രസ്താവന തെറ്റാണ്. കാരണം num ഒരു പോയിന്റർ വേരിയബിളല്ല. അതിലുള്ള 25 എന്ന വില ഒരു മെമ്മറിവിലാസവും അല്ല. * ഓപ്പറേറ്റർ പോയിന്റർ വേരിയബിളിനോടു കൂടി മാത്രമേ ഉപയോഗിക്കാൻ പാടുള്ളൂ.

1.3 മെമ്മറി നീക്കിവയ്ക്കലിന്റെ രീതികൾ (Methods of memory allocation)

വേരിയബിൾ പ്രഖ്യാപന പ്രസ്താവനയിലൂടെയാണ് മെമ്മറി നീക്കിവയ്ക്കൽ തുടങ്ങുന്നതെന്നു എന്ന് നമുക്കറിയാം. പ്രോഗ്രാം റാമിൽ ലോഡ് ചെയ്യുമ്പോൾ ആവശ്യമുള്ള മെമ്മറിയും നീക്കിവയ്ക്കുന്നു. മെമ്മറി നീക്കിവയ്ക്കലിനു ശേഷമാണ് പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കൽ ആരംഭിക്കുന്നത്. മെമ്മറി നീക്കി വയ്ക്കുന്നതിന്റെ അളവ് പ്രോഗ്രാമിൽ ഉപയോഗിച്ചിരിക്കുന്ന വേരിയബിളുകളുടെ എണ്ണത്തെയും ഇനത്തെയും അനുസരിച്ചാണ്. ഈ അളവ് സ്ഥിരമാണ് (static). ഇത് ഒരിക്കലും പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുന്ന സമയത്ത്

കൂടുകയോ കുറയുകയോ ഇല്ല. ഇങ്ങനെ പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുന്നതിനു മുമ്പ് നടക്കുന്ന മെമ്മറി നീക്കിവയ്ക്കലിനെ സ്റ്റാറ്റിക് മെമ്മറി നീക്കിവയ്ക്കൽ (Static Memory Allocation) എന്ന് പറയുന്നു. ഈ നീക്കിവയ്ക്കൽ നടക്കുന്നത് പ്രോഗ്രാമിലെ വേരിയബിൾ പ്രഖ്യാപന പ്രസ്താവനയ്ക്കനുസരിച്ചാണ്. ഇനി മറ്റൊരുതരം മെമ്മറി നീക്കിവയ്ക്കൽ ആയ ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കൽ (Dynamic Memory Allocation) പരിചയപ്പെടാം. ഇവിടെ മെമ്മറി നീക്കിവയ്ക്കൽ നടക്കുന്നത് പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോഴാണ്. ഈ നീക്കിവയ്ക്കൽ നടക്കുന്നത് new എന്ന ഓപ്പറേറ്റർ മുഖേനയാണ്. ഇങ്ങനെ നീക്കിവച്ച് മെമ്മറി സ്വതന്ത്രമാക്കാൻ (Deallocate) delete എന്ന ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു.

1.3.1 ഡൈനാമിക് ഓപ്പറേറ്ററുകൾ - new, delete

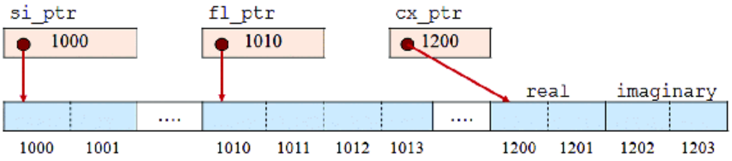
C++ ലെ new എന്ന കീവേഡ് ഒരു ഓപ്പറേറ്റർ ആണ്. ഇത് ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കലിനുവേണ്ടി ഉപയോഗിക്കുന്നു. ഈ നീക്കിവയ്ക്കൽ നടക്കുന്നത് പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുന്ന സമയത്താണ് (റൺ ടൈമിലാണ്). ഇത് ഒരു യൂനറി ഓപ്പറേറ്റർ ആണ്. അടിസ്ഥാന ഡാറ്റ ഇനത്തിലെയോ ഉപയോക്തൃനിർവചിത ഡാറ്റ ഇനത്തിലേയോ ഒരു ഓപ്പറേറ്ററാണ് ഇതിൽ ഉപയോഗിക്കുന്നത്. new എന്ന ഓപ്പറേറ്ററും ഈ ഓപ്പറേറ്ററും കൂടിച്ചേർന്ന് ഒരു പദപ്രയോഗം (Expression) ഉണ്ടാക്കുകയും ആ പദപ്രയോഗം ഒരു വില തിരിച്ച് നൽകുകയും ചെയ്യുന്നു. ഒരു മെമ്മറി സ്ഥാനത്തിന്റെ വിലാസമാണ് ആ വില. മെമ്മറി സ്ഥാനത്തിന്റെ വലുപ്പം ഓപ്പറേറ്ററിന്റെ ഡാറ്റ ഇനത്തിനനുസരിച്ചായിരിക്കും. ഡൈനാമിക് നീക്കിവയ്ക്കലിന് താഴെ കൊടുത്ത വാക്യഘടന ഉപയോഗിക്കുന്നു.

```
pointer_variable = new data_type;
```

പോയിന്റർ വേരിയബിൾ ഉപയോഗിക്കുന്നത് new ഓപ്പറേറ്റർ തിരിച്ച് തരുന്ന മെമ്മറി വിലാസം സൂക്ഷിക്കാൻ വേണ്ടിയാണ്. അതുകൊണ്ട് new ഓപ്പറേറ്ററിന് ശേഷം നൽകിയ ഡാറ്റ ഇനത്തിൽ തന്നെ പോയിന്റർ വേരിയബിൾ നേരത്തേ പ്രഖ്യാപിച്ചിരിക്കണം.

```
short * si_ptr;
float * fl_ptr;
struct complex * cx_ptr;
si_ptr = new short;
fl_ptr = new float;
cx_ptr = new complex;
```

ഇവയുടെ മെമ്മറി നീക്കിവയ്ക്കൽ ചിത്രം 1.8 ൽ കാണിച്ചിരിക്കുന്നു.



ചിത്രം 1.8: ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കുന്നതിന്റെ

ചിത്രം 1.8 ൽ കാണിച്ചിരിക്കുന്നത് ശ്രദ്ധിക്കുക. ഇവിടെ 2 ബൈറ്റ് മെമ്മറി സ്ഥാനം short ഇനത്തിലുള്ള ഡാറ്റയ്ക്ക് നീക്കിവയ്ക്കുന്നു. ഇതിന്റെ വിലാസം 1000 ആണ്, ഇതുപോലെ 1010 എന്ന വിലാസത്തിൽ 4 ബൈറ്റ് മെമ്മറി float ഇനത്തിലുള്ള ഡാറ്റയ്ക്ക്

നീക്കിവയ്ക്കുന്നു. ഇത് സൂക്ഷിക്കുന്നത് fl_ptr എന്ന പോയിന്റർ വേരിയബിളിലാണ്. മുമ്പ് നാം പഠിച്ച complex എന്ന സ്ട്രക്ചർ 2 short തരത്തിലുള്ള അംഗങ്ങളെ ഉൾക്കൊള്ളുന്നു. അതുകൊണ്ട് 1200 എന്ന വിലാസത്തിൽ തുടങ്ങുന്ന 4 ബൈറ്റ് മെമ്മറി ലൊക്കേഷൻ cx_ptr എന്ന complex സ്ട്രക്ചർ ഇനത്തിലുള്ള പോയിന്റർ വേരിയബിൾ സൂചിപ്പിക്കുന്നു. (short real നും short imaginary യ്ക്കും 2 ബൈറ്റ് വീതം ആകെ 4 ബൈറ്റ്). ഡൈനാമിക് മെമ്മറി അലോക്കേഷൻ നടത്തിയ സ്ഥലങ്ങൾ സാധാരണ വേരിയബിളുകൾക്ക് സൂചിപ്പിക്കാൻ കഴിയില്ല. ഇത് സൂചിപ്പിക്കാൻ പോയിന്റ് വേരിയബിൾ തന്നെ ഉപയോഗിക്കണം. താഴെ കൊടുത്ത ഉദാഹരണം ശ്രദ്ധിക്കുക.

```
*si_ptr = 247;
cin >> *fl_ptr;
```

നമുക്ക് സ്ട്രക്ചർ പോയിന്റർ വേരിയബിളായ cx_ptr ഉണ്ട്. ഇത് സൂചിപ്പിക്കുന്ന ഡാറ്റ മുകളിൽ കൊടുത്ത രീതിയിൽ ഉപയോഗിക്കാൻ കഴിയില്ല. ഇതിന്റെ ഉപയോഗരീതി ഈ അധ്യായത്തിൽത്തന്നെ നമുക്ക് പിന്നീട് മനസ്സിലാക്കാം.

നമുക്ക് സ്റ്റാറ്റിക് മെമ്മറി നീക്കിവയ്ക്കൽ വേരിയബിളുകൾക്ക് പ്രാരംഭവില നൽകുന്നതു പോലെ ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കലിലും വേരിയബിളുകൾക്ക് പ്രാരംഭവില നൽകാം. ഇതിന് ചുവടെ ചേർത്ത വാക്യഘടന ഉപയോഗിക്കുന്നു.

```
pointer_variable = new data_type(value);
```

ചുവടെ കൊടുത്തവ ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കലിന്റെ കൂടെയുള്ള പ്രാരംഭവില നൽകലിന് ഉദാഹരണങ്ങളാണ്.

```
si_ptr = new short(0);
fl_ptr = new float(3.14);
```

cx_ptr ന്റെ കാര്യത്തിൽ ഈ തരത്തിലുള്ള പ്രാരംഭവില നൽകൽ സാധ്യമല്ല.

ഒരിക്കൽ നമ്മൾ new ഓപ്പറേറ്റർ ഉപയോഗിച്ച് ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കൽ നടത്തിയാൽ അത് സ്വതന്ത്രമാക്കേണ്ടതായിട്ടുണ്ട് അഥവാ ഡീ അലോക്കേറ്റ് ചെയ്യേണ്ട തായിട്ടുണ്ട്. C++ ൽ ഇതിനുവേണ്ടി delete ഓപ്പറേറ്ററിനെ ഉപയോഗിക്കുന്നു. സ്റ്റാറ്റിക് മെമ്മറി നീക്കിവയ്ക്കലിന്റെ കാര്യത്തിൽ വേരിയബിളിന്റെ പരിധിയും, നിലനിൽക്കുന്ന സമയവും നോക്കി ഓപ്പറേറ്റിംഗ് സിസ്റ്റം തന്നെ മെമ്മറി നീക്കിവയ്ക്കലും, സ്വതന്ത്രമാക്കലും നടത്തും. എന്നാൽ ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കലുമായി ബന്ധപ്പെട്ട മെമ്മറി സ്വതന്ത്രമാക്കാൻ പ്രോഗ്രാമിൽ തന്നെ delete ഓപ്പറേറ്റർ ഉപയോഗിച്ചുള്ള നിർദേശങ്ങൾ നൽകേണ്ടതാണ്. delete ഓപ്പറേറ്ററിന്റെ ഉപയോഗം താഴെ നൽകിയിരിക്കുന്നു.

```
delete pointer_variable;
```

ചുവടെ ഉദാഹരണങ്ങൾ ചേർക്കുന്നു.

```
delete si_ptr;
delete fl_ptr, cx_ptr;
```

1.3.2 മെമ്മറി ലീക്ക് (Memory leak)

new ഓപ്പറേറ്റർ ഉപയോഗിച്ച് നീക്കി വച്ച മെമ്മറി ബ്ളോക്ക് delete ഓപ്പറേറ്റർ ഉപയോഗിച്ച് സ്വതന്ത്രമാക്കുന്നില്ല എങ്കിൽ അതിനെ ഓർഫൻറ് ബ്ലോക്ക് എന്ന് പറയുന്നു. ഇത് ഉപയോഗിക്കാതെ ബാക്കി വരുന്ന മെമ്മറി ബ്ലോക്ക് ആണ്. പക്ഷെ ഇത് വീണ്ടും

ഡാറ്റ സൂക്ഷിക്കാൻവേണ്ടി നീക്കിവയ്ക്കാൻ കഴിയില്ല. പ്രോഗ്രാമിന്റെ ഓരോ പ്രവർത്തിപ്പിക്കലിലും ഇത്തരത്തിലുള്ള ബ്ലോക്കുകൾ സൃഷ്ടിക്കപ്പെടുകയും മെമ്മറിയുടെ ഉപയോഗിക്കത്തക്ക ഭാഗം തുടർച്ചയായി കുറയുകയും ചെയ്യും. ഇങ്ങനെ മെമ്മറി നഷ്ടപ്പെടുന്നതിനെ മെമ്മറി ലീക്ക് എന്ന് പറയുന്നു.

താഴെ പറയുന്നവയാണ് മെമ്മറി ലീക്കിന്റെ കാരണങ്ങൾ.

- ഡൈനാമിക്ക് നീക്കിവയ്ക്കൽ നടത്തിയ മെമ്മറി (new ഓപ്പറേറ്റർ ഉപയോഗിച്ച്) സ്വതന്ത്രമാക്കാൻ മറന്നു പോകുന്നത്.
- പ്രോഗ്രാമിങ്ങിലെ ലോജിക്കൽ പിഴവ് മൂലം 'delete' നിർദേശം പ്രവർത്തിപ്പിക്കാത്തത്.
- നിലവിൽ ഒരു മെമ്മറി ബ്ലോക്കിനെ പോയിന്റ് ചെയ്യുന്ന പോയിന്ററിലേയ്ക്ക് new ഓപ്പറേറ്ററിലൂടെ ഒരു പുതിയ മെമ്മറി വിലാസം ശേഖരിക്കുന്നത്.

new ഓപ്പറേറ്റർ ഉപയോഗിച്ച് നടത്തിയ മെമ്മറി നീക്കിവയ്ക്കൽ delete ഓപ്പറേറ്റർ ഉപയോഗിച്ച് സ്വതന്ത്രമാക്കുക എന്നതാണ് മെമ്മറി ലീക്കിനുള്ള പരിഹാരം. ഡൈനാമിക് നീക്കിവയ്ക്കലിന്റെ കാര്യത്തിൽ മാത്രമാണ് മെമ്മറി ലീക്ക് ഉണ്ടാകുന്നത്. സ്റ്റാറ്റിക് നീക്കിവയ്ക്കലിൽ മെമ്മറി നീക്കിവയ്ക്കലും സ്വതന്ത്രമാക്കലും ഓപ്പറേറ്റിംഗ് സിസ്റ്റം നേരിട്ട് നടപ്പിലാക്കുന്നു. ഇതിന് പ്രത്യേക നിർദേശത്തിന്റെ ആവശ്യം ഇല്ല. അതുകൊണ്ട് സ്റ്റാറ്റിക് മെമ്മറി നീക്കിവയ്ക്കലിൽ മെമ്മറി ലീക്കിനുള്ള സാധ്യതയില്ല.

നമുക്കു ചെയ്യാം

ഇപ്പോൾ നമുക്ക് സ്റ്റാറ്റിക് മെമ്മറി നീക്കിവെക്കലും ഡൈനാമിക് മെമ്മറി നീക്കിവെക്കലും താരതമ്യം ചെയ്യാം. പട്ടിക 1.4 ൽ ഈ താരതമ്യം കൊടുത്തിരിക്കുന്നു. ചില ഭാഗങ്ങൾ നിങ്ങൾക്ക് പൂരിപ്പിക്കാൻ വേണ്ടി വിട്ടിരിക്കുന്നു.

സ്റ്റാറ്റിക് മെമ്മറി നീക്കിവയ്ക്കൽ	ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കൽ
i. പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുന്നതിന് മുമ്പ് നടക്കുന്നു.	
ii.	new ഓപ്പറേറ്റർ ആവശ്യമുണ്ട്.
iii.	പോയിന്റർ നിർബന്ധമാണ്.
iv. ഡാറ്റയെ വേരിയബിളുകൾ ഉപയോഗിച്ച് സൂചിപ്പിക്കുന്നു.	
v. ഡീ-നീക്കിവയ്ക്കലിന് വേണ്ടി പ്രത്യേകം സ്റ്റേറ്റ്മെന്റ് ആവശ്യമില്ല.	

പട്ടിക 1.4: സ്റ്റാറ്റിക് നീക്കിവയ്ക്കലും ഡൈനാമിക് നീക്കിവയ്ക്കലും തമ്മിലുള്ള താരതമ്യം

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. പോയിന്റർ എന്നാൽ എന്ത്?
2. ഒരു പോയിന്ററിന്റെ ഡാറ്റ ഇനം നിർണയിക്കുന്നതിന്റെ മാനദണ്ഡമെന്ത്?
3. `mks` ഒരു ഇന്റിജർ വേരിയബിളാണെങ്കിൽ, ഈ വേരിയബിളിന്റെ വിലാസം ഒരു പോയിന്റർ വേരിയബിളിൽ ശേഖരിക്കാനുള്ള C++ വാചകം എഴുതുക.
4. `ptr` ഒരു പോയിന്റർ വേരിയബിൾ ആണ്. ഇതുപയോഗിച്ച് ഒരു ഇന്റിജറിന് വേണ്ടി മെമ്മറി നീക്കി വച്ച് അതിൽ 12 എന്ന സംഖ്യ പ്രാരംഭവിലയായ് ചേർക്കുന്നതിനുള്ള C++ വാചകം എഴുതുക.
5. ചുവടെ ചേർത്ത വാചകങ്ങൾ പരിഗണിക്കുക: `int *p, a=5; p=&a; cout<<*p+a;` ഇതിന്റെ ഔട്ട്പുട്ട് എന്താണ്?

1.4 പോയിന്ററിലെ പ്രവർത്തനങ്ങൾ (Operations on pointers)

പോയിന്ററുമായി ബന്ധപ്പെട്ട് ഇൻഡയറക്ഷൻ (*) ഓപ്പറേറ്റർ, അഡ്രസ്സ് ഓഫ് (&) ഓപ്പറേറ്റർ എന്നിവ നമ്മൾ ചർച്ച ചെയ്തു കഴിഞ്ഞല്ലോ. 11-ാം ക്ലാസ്സിൽ അരിത്തമാറ്റിക്, റിലേഷണൽ, ലോജിക്കൽ എന്നീ ഓപ്പറേറ്ററുകളും പരിചയപ്പെട്ടു. ഈ ഭാഗത്ത് പോയിന്റർ വേരിയബിളിൽ ഉപയോഗിക്കുന്ന പ്രവർത്തനങ്ങളാണ് നാം പരിചയപ്പെടാൻ പോകുന്നത്.

1.4.1 പോയിന്ററിലെ അരിത്തമാറ്റിക് പ്രവർത്തനങ്ങൾ (Arithmetic operation on pointers)

മെമ്മറി വിലാസം എന്നത് ഒരു സംഖ്യയാണെന്ന് നമുക്ക് അറിയാം. അതിനാൽ ചില അരിത്തമാറ്റിക് പ്രവർത്തനങ്ങളും നമുക്ക് പോയിന്റർ വേരിയബിളിൽ പ്രയോഗിക്കാം. ഭാഗം 1.3.1 ൽ ഉപയോഗിച്ചു `si_ptr`, `fl_ptr` എന്നീ പോയിന്റർ വേരിയബിളുകൾ പരിഗണിക്കുക. (ചിത്രം 1.8 കാണുക). ഇനി ചുവടെ ചേർത്ത പ്രസ്താവനകൾ പരിശോധിക്കുക.

```
cout << si_ptr + 1;
cout << fl_ptr + 1;
```

എന്തായിരിക്കും ഔട്ട്പുട്ട്? 1001 ഉം 1011 ആണ് ഔട്ട്പുട്ട് എന്ന് നിങ്ങൾ കരുതുന്നുണ്ടോ.

പോയിന്ററിനോട് 1 കൂട്ടുന്നത് `int` തരത്തിലുള്ള വേരിയബിളിനോടോ `float` തരത്തിലുള്ള വേരിയബിളിനോടോ 1 കൂട്ടുന്നത് പോലെ അല്ല. നമ്മൾ ഒരു `short int` പോയിന്ററിനോട് 1 കൂട്ടുമ്പോൾ, ആ പ്രയോഗം തൊട്ടടുത്ത മെമ്മറി സ്ഥാനത്തിന്റെ വിലാസമായി മാറുന്നു. അതായത് 1000, 1001 എന്നീ ലൊക്കേഷനുകൾ `short int` വേരിയബിൾ ഉപയോഗിക്കുന്ന ലൊക്കേഷനുകളാണ്. 1002 എന്നത് അടുത്ത `short int` ന്റെ ബേസ് അഡ്രസ്സായിരിക്കും. അതായത് നമ്മൾ `short int` പോയിന്റർ വേരിയബിളിനോട് ഒന്ന് കൂട്ടുമ്പോൾ `short int` ന്റെ യഥാർഥ വലുപ്പം (2) ആണ് മെമ്മറി അഡ്രസ്സിനോട് കൂട്ടുന്നത്. ഇതുപോലെ ഒരു ഫ്ളോട്ട് പോയിന്റർ വേരിയബിളിനോട് 1 കൂട്ടുമ്പോൾ അതിന്റെ വലുപ്പമായ 4 ആണ് വിലാസത്തോട് കൂട്ടുന്നത്. അതുകൊണ്ട് `fl_ptr+1`

എന്ന പദപ്രയോഗം നൽകുന്നത് 1014 എന്ന വിലാസമാണ്. അതുകൊണ്ട് si_ptr+4 എന്ന പദപ്രയോഗം 1008 (1000+4×2) എന്ന വിലാസം നൽകുമെന്ന് വ്യക്തമാണ്. ഇതുപോലെ fl_ptr+3 നൽകുന്നത് 1022 (1010+3×4) എന്ന വിലാസമാണ്. ഇതുപോലെ പോയിന്റർ വേരിയബിളുകളിൽ വ്യവകലനവും (Subtraction) നടത്താം. മറ്റ് അരിത്തമാറ്റിക് ഓപ്പറേഷനുകൾ ഒന്നും തന്നെ പോയിന്റർ വേരിയബിളിനു മുകളിൽ പ്രവർത്തിക്കുകയില്ല.

നമുക്കു ചെയ്യാം

ചുവടെ കൊടുത്തിരിക്കുന്ന അരിത്തമാറ്റിക് പദപ്രയോഗങ്ങൾ നൽകുന്ന വിലകൾ കണ്ടുപിടിക്കുക.

a) si_ptr + 10 b) fl_ptr + 7

c) si_ptr - 5 d) fl_ptr - 10

ഇത്തരത്തിലുള്ള ഓപ്പറേഷനുകൾ പ്രായോഗികമായി ചിലപ്പോൾ തെറ്റായേക്കാം. കാരണം ഈ പ്രവർത്തനത്തിലൂടെ സൂചിപ്പിക്കുന്ന മെമ്മറി സ്ഥാനങ്ങൾ ഉപയോഗിക്കുവാനുള്ള അവകാശപ്രകാരം ഉപയോഗിക്കാൻ അനുവാദമില്ലാത്തവയായിരിക്കും.

ചുവടെ ചേർത്തിരിക്കുന്ന ഉദാഹരണങ്ങൾ പോയിന്റർ ഓപ്പറേഷനുകൾ വിശദീകരിക്കുന്നു.

```
int *ptr1, *ptr2;        // രണ്ട് പോയിന്റർ വേരിയബിളുകളുടെ പ്രഖ്യാപനം
ptr1 = new int(5);       /* ഡൈനാമിക് മെമ്മറി അലോക്കേഷൻ (വിലാസം 1000
                            ആണെന്നിരിക്കട്ടെ), 5 എന്ന സംഖ്യ പ്രാരംഭവിലയായ്
                            ചേർന്നിരിക്കുന്നു. 5*/
ptr2 = ptr1 + 1;         /* ptr2 തൊട്ടടുത്ത ഇന്റിജർ സ്ഥാനമായ 1004 ലേക്ക്
                            പോയിന്റ് ചെയ്യും */
++ptr2;                  // ptr2 = ptr2 + 1 എന്ന ഓപ്പറേഷന് തുല്യം
cout<< ptr1;             // 1000 പ്രദർശിപ്പിക്കുന്നു
cout<< *ptr1;            // 5 പ്രദർശിപ്പിക്കുന്നു
cout<< ptr2;             // 1004 പ്രദർശിപ്പിക്കുന്നു
cin>> *ptr2;             /* ഒരു ഇന്റിജർ റീഡ് ചെയ്ത് (12 ആണെന്നിരിക്കട്ടെ)
                            ഇതിനെ 1004 എന്ന ലൊക്കേഷനിൽ ശേഖരിക്കുന്നു */
cout<< *ptr1 + 1;        // 6 (5 + 1) പ്രദർശിപ്പിക്കുന്നു
cout<< *(ptr1 + 2);      // 12 പ്രദർശിപ്പിക്കുന്നു, (1004 ലെ വില)
ptr1--;                  // ptr1 = ptr1 - 1 എന്ന പ്രസ്താവനയ്ക്ക്
                            തുല്യം
```

ഇനി നമുക്ക് പോയിന്ററിന്റെ പ്രവർത്തനങ്ങൾ വിവരിക്കുന്ന ഒരു പ്രോഗ്രാം എഴുതാം. പ്രോഗ്രാം 1.3 ഒരു കൂട്ടം കുട്ടികളുടെ ശരാശരി ഉയരം കാണാനുള്ളതാണ്.

Program 1.3: കുട്ടികളുടെ ശരാശരി ഉയരം കാണാൻ

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int *ht_ptr, n, s=0;
    float avg_ht;
    ht_ptr = new int;       //ഡൈനാമിക് മെമ്മറി അലോക്കേഷൻ
    cout<<"Enter the number of students: ";
    cin>>n;
    for (int i=0; i<n; i++)
    {
        cout<<"Enter the height of student "<<i+1<<" - ";
        cin>>*(ht_ptr+i);//to get the address of the next location
        s = s + *(ht_ptr+i);
    }
    avg_ht = (float)s/n;
    cout<<"Average height of students in the class = "<<avg_ht;
    return 0;
}
```

പ്രോഗ്രാം 1.3 ൽ ഡൈനാമിക് നീക്കിവയ്ക്കലിലൂടെ ഒരു ഇന്റിജർ മെമ്മറിസ്ഥാനം നിർമിച്ച് അതിന്റെ വിലാസം ht_ptr എന്ന പോയിന്റർ വേരിയബിളിൽ ശേഖരിക്കുന്നു. ലൂപ്പിന്റെ ബോഡി ആദ്യത്തെ പ്രാവശ്യം പ്രവർത്തിക്കുമ്പോൾ 0 ആണ് വിലാസത്തോട് കൂട്ടുന്നത്. ഇത് പ്രത്യേകിച്ച് മാറ്റമൊന്നും വരുത്തുന്നില്ല. ഇൻപുട്ട് ഡാറ്റ മെമ്മറി സ്ഥാനത്ത് സൂക്ഷിക്കുന്നു. ലൂപ്പ് ബോഡിയുടെ അടുത്ത പ്രവർത്തനത്തിൽ വിലാസത്തോട് 1 കൂട്ടുന്നു. അങ്ങനെ അടുത്ത ഇന്റിജർ സ്ഥാനത്തിൽ രണ്ടാമത്തെ ഡാറ്റ ശേഖരിക്കപ്പെടുന്നു. ഈ പ്രവൃത്തി n തവണ ആവർത്തിക്കപ്പെടുന്നു. അങ്ങനെ n തവണ കുട്ടികളുടെ ഡാറ്റയും തുടർച്ചയായ ഇന്റിജർ സ്ഥാനങ്ങളിൽ ശേഖരിക്കപ്പെടുന്നു. ഓരോ ഡാറ്റയും നൽകുന്ന സമയത്തുതന്നെ അതിന്റെ തുകയും കണക്കാക്കുന്നു. അവസാനം ശരാശരി ഉയരം കണക്കാക്കി പ്രദർശിപ്പിക്കുന്നു. കൃത്യമായ ശരാശരി ഉയരം ലഭിക്കുവാൻ തുകയെ ബാഹ്യ ഇനം മാറ്റലിലൂടെ (Explicit type conversion) ഫ്ളോട്ടാക്കി മാറ്റിയിരിക്കുന്നു. പ്രോഗ്രാമിന്റെ ഔട്ട്പുട്ട് ചുവടെ ചേർക്കുന്നു.

```
Enter the number of students: 5
Enter the height of student 1 - 170
Enter the height of student 2 - 169
Enter the height of student 3 - 175
Enter the height of student 4 - 165
Enter the height of student 5 - 177
Average height of students in the class = 171.199997
```

പ്രോഗ്രാം 1.3 മറ്റൊരു കാര്യം കൂടി വിവരിക്കുന്നു. ഒരേതരം ഡാറ്റയുടെ കൂട്ടത്തെ അനായാസമായി പോയിന്ററിലൂടെ കൈകാര്യം ചെയ്യാം. നമുക്കറിയാം 11-ാം ക്ലാസിൽ ഇതേ ആവശ്യത്തിനു വേണ്ടി നമ്മൾ അറെ ഉപയോഗിച്ചു. പക്ഷെ അറെ പ്രഖ്യാപനത്തോടൊപ്പം തന്നെ അറെയുടെ വലുപ്പവും നൽകിയിരിക്കണം. ഇത് മെമ്മറി അനാവശ്യമായി

നഷ്ടപ്പെടുത്തുകയും ആവശ്യത്തിന് മെമ്മറി ലഭിക്കാതിരിക്കാൻ കാരണമാവുകയും ചെയ്യുന്നു. എന്നാൽ പോയിന്റർ വേരിയബിളുപയോഗിച്ച് ഡൈനാമിക് അലോക്കേഷൻ നടത്തുമ്പോൾ അറെയുടെ വലുപ്പം കൃത്യം കുട്ടികളുടെ എണ്ണത്തിനനുസരിച്ചായിരിക്കും. ഇവിടെ മെമ്മറി പാഴാക്കപ്പെടുന്നില്ല.

പക്ഷെ ഈ തരത്തിലുള്ള മെമ്മറി ഉപയോഗത്തിന് ചില പ്രശ്നങ്ങളുണ്ട്. പ്രോഗ്രാം 1.3 എപ്പോഴും `n` ന്റെ എല്ലാ വിലകൾക്കും പ്രവർത്തിക്കണം എന്നില്ല. GCC, `avg_ht` എന്ന വേരിയബിളിന്റെ ഒരു ഔട്ട്പുട്ടും ചിലപ്പോൾ നൽകില്ല. ചില അപ്രതീക്ഷിത വിലകൾ പ്രദർശിപ്പിക്കാനും സാധ്യതയുണ്ട്. ഇതിനു കാരണം `ht_ptr` ന് പ്രാരംഭവിലയായി നൽകുന്നത് ഏതെങ്കിലും ഒരു മെമ്മറി സ്ഥാനത്തിന്റെ വിലാസമാണ്. നമ്മൾ ഉപയോഗിക്കുന്ന പോയിന്റർ അരിത്തമാറ്റിക് ഓപ്പറേഷനും നേരത്തെ പറഞ്ഞപോലെ അനുവാദമില്ലാത്ത സ്ഥാനങ്ങൾ ഉപയോഗിക്കുന്നതിന് കാരണമായേക്കാം. ഇതെല്ലാം പ്രോഗ്രാമിന്റെ അപ്രതീക്ഷിതമായ അവസാനിപ്പിക്കലിന് കാരണമാവും. പലപ്പോഴും മെമ്മറിയിൽ സൂക്ഷിച്ച ഡാറ്റയും നഷ്ടപ്പെട്ടു പോകും, അങ്ങനെ നമുക്ക് കൃത്യമായ ഔട്ട്പുട്ടും ലഭിക്കുകയില്ല. ഈ പ്രശ്നങ്ങൾ പരിഹരിക്കാൻ നമുക്ക് ഭാഗം 1.5 ലെ ഡൈനാമിക് അറെകൾ ഉപയോഗപ്പെടുത്താം.

1.4.2 പോയിന്ററിലെ റിലേഷണൽ പ്രവർത്തനങ്ങൾ (Relational operations on pointers)

നാം പഠിച്ച ആറ് റിലേഷണൽ ഓപ്പറേറ്ററുകളിൽ == (Equality-തുല്യം), != (Non equality-തുല്യമല്ല) എന്നീ ഓപ്പറേറ്ററുകൾ മാത്രമേ പോയിന്റർ വേരിയബിളുകളിൽ ഉപയോഗിക്കാറുള്ളൂ. മെമ്മറി വിലാസം എന്നാൽ ഒരു മെമ്മറി സ്ഥാനം തിരിച്ചറിയാനുള്ള സവിശേഷ സംഖ്യയാണ്. അതിനാൽ ഇതിനു മുകളിൽ മറ്റ് റിലേഷനിൽ ഓപ്പറേറ്ററുകൾ ഉപയോഗിക്കുന്നതിൽ അർഥമില്ല. `p`, `q` എന്നിവ രണ്ട് പോയിന്ററുകൾ ആണെങ്കിൽ അവയിൽ തുല്യമായതോ/വ്യത്യസ്തമായതോ ആയി മെമ്മറി സ്ഥാനങ്ങളുടെ വിലാസങ്ങളാണ് ഉണ്ടാവുക. ഇത് പരിശോധിക്കാൻ `p==q` അല്ലെങ്കിൽ `p!=q` എന്നീ പദപ്രയോഗങ്ങൾ ഉപയോഗിക്കാം.

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. C++ ലെ ഡൈനാമിക് മെമ്മറി നീക്കിവയ്ക്കൽ ഓപ്പറേറ്റർ ആണ് __________.
2. ചുവടെ കൊടുത്ത പ്രസ്താവന പ്രവർത്തിക്കുമ്പോൾ എന്ത് സംഭവിക്കുന്നു?

   `int *b = new int(10);`
3. ഓർഫൻസ് മെമ്മറി ബ്ലോക്ക് എന്നാലെന്ത്?
4. `p` ഒരു ഇന്റിജർ പോയിന്ററാണെന്നിരിക്കട്ടെ. ചുവടെ കൊടുത്തവയിൽ ഏതൊക്കെയാണ് അസാധുവായത്?

a. `cout<<&p;`	b. `p=p*5;`	c. `p>0`
d. `p++;`	e. `p=1500;`	f. `cout<<*p * 2;`

1.5 പോയിന്ററും അറെയും (Pointer and array)

ഒരു പേരിൽ ഒരേ തരത്തിലുള്ള ഒന്നിലധികം ഡാറ്റകളെ ശേഖരിക്കാൻ അറെയ്ക്ക് കഴിയുമെന്ന് നാം പഠിച്ച് കഴിഞ്ഞല്ലോ. തുടർച്ചയായ മെമ്മറി സ്ഥാനങ്ങളിൽ ഡാറ്റ ശേഖരിക്കുന്നതിനു അറെ ഉപയോഗിക്കാം. ചിത്രം 1.9 കാണിക്കുന്നത് `ar[10]` എന്ന `int` ടൈപ്പ് അറെയുടെ ചിത്രമാണ്. ഇതിൽ 10 സംഖ്യകളുണ്ട്.

	ar
1000	34
1004	12
1008	8
1012	18
1016	24
1020	38
1024	43
1028	14
1032	7
1036	19

ചിത്രം 1.9: ar എന്ന അറെയുടെ മെമ്മറി നീക്കിവയ്ക്കൽ

ഈ അറെ 1000 എന്ന മെമ്മറി ലൊക്കേഷനിൽ തുടങ്ങുന്നതായി സങ്കൽപ്പിക്കുക. ഇതിനകത്തെ ഓരോ സ്ഥാനവും 4 ബൈറ്റ് വീതം ആണ് (GCC അനുസരിച്ച്). ഈ അറെയിലെ ഏത് അംഗത്തെയും അറെയുടെ പേരും സബ്സ്ക്രിപ്റ്റും ഉപയോഗിച്ച് സൂചിപ്പിക്കാം. ഉദാഹരണം `ar[0]` എന്നത് 34 എന്ന സംഖ്യ നൽകുന്നു. `ar[1]` 12 ഉം `ar[9]` 19 ഉം തിരിച്ചു തരുന്നു.

നമുക്കു ചെയ്യാം

ar എന്ന അറെയിലെ 10 സംഖ്യകളും പ്രദർശിപ്പിക്കുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക.

എങ്ങനെയാണ് ഈ അറെയുടെ ഒന്നാമത്തെ സ്ഥാനത്തിന്റെ വിലാസം ഒരു പോയിന്ററിൽ ശേഖരിക്കുന്നത്?

`ptr` എന്നത് ഒരു ഇന്റിജർ പോയിന്റർ ആണെങ്കിൽ `ar` എന്ന അറെയുടെ ആദ്യത്തെ സ്ഥാനത്തിന്റെ വിലാസം താഴെ കൊടുക്കുന്ന രീതിയിൽ പോയിന്ററിൽ ശേഖരിക്കാം.

```
    ptr = &ar[0];
```

ഇനി നമുക്ക് ചുവടെ ചേർത്ത വാചകങ്ങളുടെ ഔട്ട്പുട്ട് പരിശോധിക്കാം.

```
cout<<ptr;       //1000 പ്രദർശിപ്പിക്കുന്നു, ar[0] യുടെ വിലാസം
cout<<*ptr;      //34 പ്രദർശിപ്പിക്കുന്നു, ar[0] യുടെ വില
cout<<(ptr+1);   //1004 പ്രദർശിപ്പിക്കുന്നു, ar[1] ന്റെ വിലാസം
cout<<*(ptr+1);  //12 പ്രദർശിപ്പിക്കുന്നു, ar[1] ന്റെ വില
cout<<(ptr+9);   //1036 പ്രദർശിപ്പിക്കുന്നു, ar[9] ന്റെ വിലാസം
cout<<*(ptr+9);  //19 പ്രദർശിപ്പിക്കുന്നു, ar[9] ന്റെ വില
```

`cout<<ar;` എന്ന പ്രസ്താവനയുടെ ഔട്ട്പുട്ട് നിങ്ങൾക്ക് കണ്ടെത്താൻ കഴിയുമോ?

ഔട്ട് പുട്ട് 1000 ആണ്. ഇത് അറെയുടെ ആദ്യത്തെ സ്ഥാനത്തിന്റെ വിലാസമാണ്. ഈ വിലാസത്തെ അറെയുടെ അടിസ്ഥാന വിലാസം എന്നു പറയുന്നു. വിലാസം ശേഖരിക്കുന്ന വേരിയബിൾ ഒരു പോയിന്ററാണെന്നു നമുക്കറിയാം. അതുകൊണ്ട് അറെയുടെ പേരായ `ar` ഉം ഒരു പോയിന്ററാണ്. ആയതിനാൽ താഴെ കൊടുത്ത വാചകങ്ങൾ ശരിയാണ്.

```
cout<<ar;      //1000 പ്രദർശിപ്പിക്കുന്നു, ar[0] യുടെ വിലാസം
ptr=ar;        //ptr=&ar[0] എന്നതിന് തുല്യം;
cout<<*ar;     //34 പ്രദർശിപ്പിക്കുന്നു, cout<<ar[0]; എന്നതിന് തുല്യം
```

```
cout<<(ar+1); //1004 പ്രദർശിപ്പിക്കുന്നു, ar[1]; ന്റെ വിലാസം
cout<<*(ar+1);  //34 പ്രദർശിപ്പിക്കുന്നു, cout<<ar[1]; എന്നതിന് തുല്യം
```

ചുവടെ കൊടുത്ത C++ വാചകങ്ങൾ ഈ അറെയുടെ എല്ലാ അംഗങ്ങളെയും പ്രദർശിപ്പിക്കുന്നു.

```
    for (int i=0; i<10; i++)
        cout<<*(ar+i)<<'\t';
```

സാധാരണ പോയിന്ററും അറെയുടെ പേരും തമ്മിൽ ചില വ്യത്യാസങ്ങളുണ്ട്. ptr++; എന്ന വാചകം ശരിയാണ് ഇത് ptr=ptr+1; എന്നതിന് തുല്യമാണ്. ഈ വാചകം പ്രവർത്തിപ്പിച്ചു കഴിഞ്ഞാൽ ptr, ar[1] ന്റെ മെമ്മറി സ്ഥാനത്തേക്ക് പോയിന്റ് ചെയ്യും. അതായത് ptr ൽ ar[1] ന്റെ വിലാസം ആണ് ശേഖരിക്കപ്പെടുന്നത്. ptr++ന് പകരമായി ar++ എന്ന് ഉപയോഗിക്കാൻ കഴിയില്ല. കാരണം അറെയുടെ പേര് എപ്പോഴും അറെയുടെ ബേസ് അഡ്രസ്സ് മാത്രം ഉൾക്കൊള്ളുന്നു. ഇത് മാറ്റാൻ സാധ്യമല്ല.

ഡൈനാമിക് അറെ

C++ ൽ അറെ ഒരേ ഇനത്തിലുള്ള ഡാറ്റ കൈകാര്യം ചെയ്യാൻ ഉപയോഗിക്കുന്നു. പക്ഷെ ഡാറ്റയുടെ എണ്ണം മുൻകൂട്ടി അറിയില്ല എങ്കിൽ അറെ പ്രഖ്യാപിക്കാൻ പ്രയാസം നേരിടുന്നു. ഒരു ജില്ലയിലെ എല്ലാ ഹയർ സെക്കണ്ടറി സ്കൂളുകളിലെയും വിജയ ശതമാനം ശേഖരിക്കുന്നതിന് ഒരു അറെ പ്രഖ്യാപിക്കുന്നത് എങ്ങനെയാണ്? float pass[n]; float pass[]; എന്നീ രണ്ട് പ്രഖ്യാപനങ്ങളും C++ ൽ തെറ്റാണ്. അറെയുടെ വലുപ്പം ഒരു ഇന്റിജർ സ്ഥിര വിലയായിരിക്കണം. അങ്ങനെ പ്രഖ്യാപിക്കുമ്പോൾ മെമ്മറി പാഴാക്കപ്പെടുവാനോ തികയാതെ വരുവാനോ കാരണമാകാം. ജില്ലയിലെ സ്കൂളുകളുടെ എണ്ണം പ്രോഗ്രാം തയാറാക്കുമ്പോൾ അറിയില്ല എങ്കിൽ, ഇൻപുട്ടിനനുസരിച്ച് മെമ്മറി നീക്കി വയ്ക്കുന്ന ഒരു അറെ ഉപയോഗിച്ചാൽ മതിയാകും. ഇവിടെയാണ് ഡൈനാമിക് അറെയുടെ ആവശ്യം ഉണ്ടാകുന്നത്. ഡൈനാമിക് അറെ പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോഴാണ് നിർമിക്കപ്പെടുന്നത്. ഇതിനുവേണ്ടി *ഡൈനാമിക് മെമ്മറി* അലോക്കേഷൻ ഓപ്പറേറ്ററായ new ഉപയോഗിക്കുന്നു. ഇതിന്റെ വാക്യഘടന ചുവടെ ചേർക്കുന്നു.

```
    pointer = new data_type[size];
```

ഇവിടെ size ഒരു സ്ഥിരവില (constant), വേരിയബിൾ, ഇന്റിജർ എക്സ്പ്രഷൻ ഇവയിൽ ഏതെങ്കിലും ഒന്നായാൽ മതി.

പ്രോഗ്രാം 1.4, ഡൈനാമിക് അറെയുടെ ആശയം വിവരിക്കുന്നു. ഈ പ്രോഗ്രാമിൽ വിജയശതമാനം ശേഖരിക്കുന്നത് ഉപയോക്താവ് റൺ ടൈമിൽ നൽകുന്ന സ്കൂളിന്റെ എണ്ണമനുസരിച്ചാണ്.

Program 1.4: ഏറ്റവും കൂടിയ വിജയ ശതമാനം കാണാൻ

```
#include <iostream>
using namespace std;
int main()
{
    float *pass, max;
```

```
    int i, n;
    cout<<"Enter the number of schools: ";
    cin>>n;    //സ്കൂളിന്റെ എണ്ണം ഇൻപുട്ട് ചെയ്യാൻ
    pass = new float[n];//n അംഗങ്ങളുള്ള ഡൈനാമിക് അറെയുടെ പ്രഖ്യാ
    പനം
    for (i=0; i<n; i++)
    {
        cout<<"Percent of pass by school "<<i+1<<": ";
        cin>>pass[i];  //സബ്സ്ക്രിപ്റ്റഡ് വേരിയബിളിന്റെ ഉപയോഗം
    }
    max=pass[0];
    for (i=1; i<n; i++)
        if (pass[i]>max) max = *(pass+i);
    /* അംഗങ്ങളെ സബ്സ്ക്രിപ്റ്റ് ഉപയോഗിച്ചും പോയിന്റർ ഓപ്പറേഷനിലൂടെയും ഉപ
    യോഗിക്കുന്നു. */
    cout<<"Highest percent is "<<max;
    return 0;
}
```

Output:

```
Enter the number of schools: 5
Percent of pass by school 1: 75.6
Percent of pass by school 2: 66.5
Percent of pass by school 3: 89.3
Percent of pass by school 4: 71
Percent of pass by school 5: 70.6
Highest percent is 89.3
```

പ്രോഗ്രാം 1.4 ൽ ഡൈനാമിക് അറെ ഉപയോഗിച്ചാണ് ഡാറ്റ ശേഖരിക്കുന്നത്. മെമ്മറി നീക്കിവയ്ക്കൽ നടക്കുന്നത് റൺ ടൈമിൽ മാത്രമാണ്. 4 ബൈറ്റ് വീതമുള്ള 5 സ്ഥാനങ്ങൾ `pass` എന്ന അറെയിൽ റൺ ടൈമിൽ തയാറാക്കുന്നു. ഈ അറെയിലെ അംഗങ്ങളെ പോയിന്ററോ സബ്സ്ക്രിപ്റ്റോ ഉപയോഗിച്ച് എടുക്കാൻ കഴിയും.

നമുക്കു ചെയ്യാം

പ്രോഗ്രാം 1.4 തയാറാക്കുന്ന സമയത്ത് ജില്ലയിലെ സ്കൂളുകളുടെ എണ്ണം കൃത്യമായി പ്രോഗ്രാമർക്ക് അറിയില്ല. ഇവിടെ സാധാരണ അറെ ഉപയോഗിക്കുകയാണെങ്കിൽ ഒരു കൂടിയ സംഖ്യ വലുപ്പമായി നൽകി അറെ പ്രഖ്യാപിക്കേണ്ടതായി വരുകയും ഇത് ശേഷിക്കുന്ന മെമ്മറി പാഴാക്കാൻ കാരണമാവുകയും ചെയ്യും. എന്നാൽ ഇവിടെ ഡൈനാമിക് അറെ ഉപയോഗിച്ചാൽ ഉപയോക്താവ് നൽകിയ കൃത്യമായ എണ്ണത്തിൽ അറെ നിർമിക്കുന്നതിനാൽ മെമ്മറി പാഴാക്കപ്പെടുന്നില്ല.

നമുക്കു ചെയ്യാം

താഴെയുള്ള രണ്ട് പ്രസ്താവനകൾ തമ്മിലുള്ള വ്യത്യാസം എഴുതുക:

```
int *ptr = new int(10);
int *ptr = new int[10];
```

1.6 പോയിന്ററും സ്ട്രിങ്ങും (Pointer and string)

സ്ട്രിങ്ങിനെ കാരക്ടർ അറെ വഴി ശേഖരിക്കാമെന്നും അറെയുടെ പേരിനെ സ്ട്രിങ് വേരിയബിളായി കണക്കാക്കാമെന്നും നമ്മൾ 11-ാം ക്ലാസിൽ പഠിച്ചു. അറെയുടെ പേരിൽ അതിന്റെ ബേസ് അഡ്രസ്സ് അടങ്ങിയിരിക്കുന്നു എന്നും നമ്മൾ മനസ്സിലാക്കി. അതു കൊണ്ട് അറെയെ ഒരു പോയിന്ററായി കണക്കാക്കാം. ഇനി നമുക്ക് ഈ ആശയങ്ങളെ യോജിപ്പിച്ചുകൊണ്ട് ഒരു സ്ട്രിങ്ങിനെ പോയിന്ററിലൂടെ എങ്ങനെ ഉപയോഗിക്കാമെന്നും അതിന് മറ്റ് പോയിന്ററുമായി എന്ത് വ്യത്യാസമുണ്ടെന്നും ചർച്ച ചെയ്യാം. താഴെ കൊടുത്ത വാചകങ്ങൾ പരിശോധിക്കുക.

```
char str[20];      //ക്യാരക്ടർ അറെ പ്രഖ്യാപനം
char *sp;          //ക്യാരക്ടർ പോയിന്റർ പ്രഖ്യാപനം
cin>>str;          //"Program" എന്ന സ്ട്രിങ്ങിനെ ഇൻപുട്ട് ചെയ്യുന്നു എന്ന് കരുതുക
cout<<str;         //"Program" എന്ന സ്ട്രിങ്ങിനെ പ്രദർശിപ്പിക്കാൻ
sp=str;   //str എന്ന സ്ട്രിങ്ങിന്റെ ഉള്ളടക്കം, പോയിന്റർ sp യിലേക്ക് കോപ്പി ചെയ്യുന്നു.
cout<<sp; //"Program" എന്ന സ്ട്രിങ് പ്രദർശിപ്പിക്കുന്നു
cout<<&str[0];     //"Program" എന്ന സ്ട്രിങ് പ്രദർശിപ്പിക്കുന്നു
cout<<sp+1;        //"rogram" എന്ന സ്ട്രിങ് പ്രദർശിപ്പിക്കുന്നു
cout<<&str+1;      //"rogram" എന്ന സ്ട്രിങ് പ്രദർശിപ്പിക്കുന്നു
cout<<str[0];      //'P' എന്ന ക്യാരക്ടർ പ്രദർശിപ്പിക്കുന്നു
cout<<*sp;         //'P' എന്ന ക്യാരക്ടർ പ്രദർശിപ്പിക്കുന്നു
cout<<&str;  //str അറെയുടെ ബേസ് അഡ്രസ്സ് പ്രദർശിപ്പിക്കുന്നു
cout<<&sp;   //sp എന്ന പോയിന്ററിന്റെ വിലാസം പ്രദർശിപ്പിക്കുന്നു
```

ഒരു അറെയിൽ ശേഖരിച്ചിരിക്കുന്ന സ്ട്രിങ്, വില നൽകൽ ഓപ്പറേറ്റർ (=) ഉപയോഗിച്ച് മറ്റൊരു കാരക്ടർ അറെയിലേക്ക് കോപ്പി ചെയ്യാൻ സാധിക്കില്ല. (ഇതിനു വേണ്ടി നമ്മൾ `strcpy()` എന്ന ഫംങ്ഷൻ ഉപയോഗിക്കുന്നു) എന്നാൽ ക്യാരക്ടർ അറെയ്ക്ക് പകരം ക്യാരക്ടർ പോയിന്റുകൾ ഉപയോഗിച്ചാൽ ഇത് സാധ്യമാകും. മുകളിലത്തെ ഉദാഹരണങ്ങളിൽ `sp=str; cout<<sp;` എന്നീ വാചകങ്ങൾ ഈ വസ്തുത സ്ഥിരീകരിക്കുന്നു. ഇതിനർഥം കാരക്ടർ പോയിന്ററിന് ഒരു സ്ട്രിങ്ങിനെ ശേഖരിക്കാൻ കഴിയും എന്നാണ്. ഈ പോയിന്ററിനെ ഒരു സ്ട്രിങ് വേരിയബിൾ ആയി കണക്കാക്കാം.

മറ്റൊരു രസകരമായ കാര്യം `cout<<&str[0];` എന്ന വാചകം മുഴുവൻ സ്ട്രിങ്ങിനേയും പ്രദർശിപ്പിക്കുന്നു എന്നതാണ്. ഇതിന്റെ അർഥം നമ്മൾ ഒരു സ്ട്രിങ്ങിന്റെ ബേസ് അഡ്രസ്സ് പ്രദർശിപ്പിക്കാൻ ശ്രമിച്ചാൽ ആ സ്ട്രിങ് മുഴുവനായും പ്രദർശിപ്പിക്കപ്പെടും. ഇതുപോലെ ഒരു `int` അറെയുടേയോ `float` അറെയുടേയോ ഒന്നാമത്തെ സ്ഥാനത്തിന്റെ വിലാസം പ്രദർശിപ്പിക്കാൻ ശ്രമിച്ചാൽ അതിന്റെ ബേസ് അഡ്രസ്സായിരിക്കും പ്രദർശിപ്പിക്കപ്പെടുന്നത്.

ഉദാഹരണത്തിന് `int a[3] = {8, 9, 10};`

```
cout<<&a[0];
```

എന്നീ പ്രസ്താവനകൾ പ്രവർത്തിപ്പിക്കുമ്പോൾ a എന്ന അറെയുടെ ബേസ് അഡ്രസ്സ് ആയിരിക്കും പ്രദർശിപ്പിക്കപ്പെടുന്നത്.

കാരക്ടർ പോയിന്ററിന്റെ നേട്ടങ്ങൾ

സ്ട്രിങ്ങിനെ സ്റ്റോർ ചെയ്യാൻ കാരക്ടർ പോയിന്റർ ഉപയോഗിക്കുമ്പോൾ ചുവടെ കൊടുത്ത നേട്ടങ്ങൾ ഉണ്ടാകുന്നു.

- വലുപ്പം പ്രഖ്യാപന പ്രസ്താവനയിൽ രേഖപ്പെടുത്താത്തതുകൊണ്ട്, എത്ര വലുപ്പമുള്ള സ്ട്രിങ്ങും ശേഖരിക്കാം. മെമ്മറി പാഴാക്കപ്പെടുകയോ തികയാതിരിക്കുകയോ ചെയ്യുന്നില്ല. എന്നാൽ ഇത് പ്രാരംഭവില നൽകുമ്പോൾതന്നെ ചെയ്തിരിക്കണം. (e.g., `char *str = "Program";`)
- വിലനൽകൽ ഓപ്പറേറ്റർ (=) ഉപയോഗിച്ച് സ്ട്രിങ് കോപ്പി ചെയ്യാം.
- സ്ട്രിങ്ങിലെ ഏത് കാരക്ടറും പോയിന്റർ ഉപയോഗിച്ച് എളുപ്പം പുറത്തെടുക്കാം.
- കാരക്ടർ അറെകളെ കുറഞ്ഞ മെമ്മറി സ്ഥലം ഉപയോഗിച്ച് കൈകാര്യം ചെയ്യാം.

സ്ട്രിങ്ങുകളുടെ അറെ

നമുക്ക് ഒരാഴ്ചയിലെ ദിവസങ്ങളുടെ പേരുകൾ ഒരു അറെയിൽ ശേഖരിക്കണമെന്നു കരുതുക. ഒരു കാരക്ടർ അറെക്ക് അല്ലെങ്കിൽ കാരക്ടർ പോയിന്ററിന് ഒരു പേര് മാത്രമേ ശേഖരിക്കാൻ കഴിയൂ. നമുക്ക് സൂചിപ്പിക്കേണ്ടത് സ്ട്രിങ്ങുകളുടെ ഒരു കൂട്ടമാണ്. ("Sunday", "Monday", ..., "Saturday") തീർച്ചയായും ഇതിനു വേണ്ടി നമ്മൾ ഉപയോഗിക്കുന്നത് കാരക്ടർ അറെകളുടെ ഒരു അറെയാണ്; (`char` തരത്തിലുള്ള ഒരു 2D അറെ) അല്ലെങ്കിൽ കാരക്ടർ പോയിന്ററുകളുടെ ഒരു അറെയാണ്. കാരക്ടർ പോയിന്ററുകളുടെ ഒരു അറെ പ്രഖ്യാപിക്കുന്നതിന്റെ രീതി താഴെ കൊടുത്തിരിക്കുന്നു.

```
char *week[7];
```

ഈ അറെക്ക് പരമാവധി 7 സ്ട്രിങ്ങുകളെ ഉൾകൊള്ളാനുള്ള കഴിവുണ്ട്. അതിൽ ഓരോ സ്ട്രിങ്ങിലും പരിധിയില്ലാതെ കാരക്ടറുകൾ ഉൾപ്പെടുത്താവുന്നതാണ്. പോയിന്റർ അറെക്ക് പ്രാരംഭവില നൽകുന്ന വിധം താഴെ ചേർക്കുന്നു.

```
char *week[7]={"Sunday", "Monday", "Tuesday", "Wednesday",
               "Thursday", "Friday", "Saturday"};
```

ചിത്രം 1.10 ൽ മെമ്മറി നീക്കിവയ്ക്കലിന്റെ കൃത്യമായ വിവരണം കൊടുത്തിരിക്കുന്നു. ഷേഡ് ചെയ്തിരിക്കുന്ന ഭാഗമാണ് നീക്കിവയ്ക്കപ്പെട്ടിരിക്കുന്നത്.

നമുക്കു ചെയ്യാം

ക്ലാസ് 11 ൽ പഠിച്ചിരുന്ന സോർട്ടിംഗ് ടെക്നിക് ഉപയോഗിച്ച് പേരുകൾ സോർട്ട് ചെയ്യാനുള്ള ഒരു C++ പ്രോഗ്രാം എഴുതുക. കാരക്ടർ പോയിന്റർ ഉപയോഗിക്കുന്നതിനാൽ സ്ട്രിങ്ങിനെ വില നൽകൽ ഓപ്പറേറ്റർ (=) ഉപയോഗിച്ച് കോപ്പി ചെയ്യാം. കമ്പ്യൂട്ടർ ലാബിൽ വച്ച് നിങ്ങൾ എഴുതിയ പ്രോഗ്രാം ശരിയാണോ എന്ന് പരിശോധിക്കുക.

Week

S	u	n	d	a	y	\0				
M	o	n	d	a	y	\0				
T	u	e	s	d	a	y	\0			
W	e	d	n	e	s	d	a	y	\0	
T	h	u	r	s	d	a	y	\0		
F	r	i	d	a	y	\0				
S	a	t	u	r	d	a	y	\0		

ചിത്രം 1.10: സ്ട്രിങ്ങുകളുടെ മെമ്മറി നീക്കിവയ്ക്കൽ

സ്ട്രിങ്ങുകളെ കൈകാര്യം ചെയ്യുന്നതിന് പോയിന്റർ അറെയ്ക്ക് പകരം സ്ട്രിങ്ങുകളുടെ അറെ ഉപയോഗിക്കാം. സ്ട്രിങ്ങുകളുടെ അറെ തയാറാക്കാൻ 2D കാരക്ടർ അറെ ഉപയോഗിക്കുന്നു. പ്രഖ്യാപന പ്രസ്താവന താഴെ കൊടുത്തിരിക്കുന്നു.

```
char name[10][20];
```

ഈ അറെയ്ക്ക് 10 പേരുകൾ ഉൾക്കൊള്ളാനുള്ള കഴിവുണ്ട്. ഓരോ പേരിലും മാക്സിമം 10 ക്യാരക്ടർ വരെ അനുവദിച്ചിരിക്കുന്നു. ഒരു ബൈറ്റ് നൾ ക്യാരക്ടറിനു ('\0') വേണ്ടി റിസർവ് ചെയ്യപ്പെട്ടിരിക്കുന്നു. ഓരോ സ്ട്രിങ്ങും `name[i]` എന്ന എക്സ്പ്രഷനിലൂടെ റഫർ സൂചിപ്പിക്കുന്നു. ഇതിൽ സബ്സ്ക്രിപ്റ്റ് `i` യ്ക്ക് 0 മുതൽ 9 വരെയുള്ള വിലകൾ സ്വീകരിക്കാം. ഇവിടെ `strcpy()` ഫംഗ്ഷൻ ഉപയോഗിച്ച് മാത്രമേ സ്ട്രിങ്ങുകൾ കോപ്പി ചെയ്യാൻ കഴിയൂ.

ചിത്രം 1.10 ൽ കാണിച്ചിരിക്കുന്ന `week` എന്ന അറെയെ `char` തരത്തിലുള്ള ഒരു 2D അറെ ഉപയോഗിച്ച് പ്രഖ്യാപിക്കുന്ന രീതി ചുവടെ കൊടുത്തിരിക്കുന്നു.

```
char week[7][11]={"Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday"};
```

എന്താണ് ഈ അറെയും മുമ്പ് സൂചിപ്പിച്ച ക്യാരക്ടർ പോയിന്റർ അറെയും ഉപയോഗിക്കുന്നതിലുള്ള വ്യത്യാസം? ക്യാരക്ടർ പോയിന്റർ അറെ ഉപയോഗിക്കുമ്പോൾ അറെയുടെ നീക്കി വയ്ക്കാത്ത ഭാഗങ്ങൾ (ചിത്രം 1.10 ൽ ഷേഡ് ചെയ്യാത്ത ഭാഗങ്ങൾ) മറ്റ് മെമ്മറി നീക്കി വയ്ക്കലിനുവേണ്ടി ഉപയോഗപ്പെടുത്താം. എന്നാൽ 2D ക്യാരക്ടർ അറെ ഉപയോഗിക്കുമ്പോൾ ഈ ഭാഗങ്ങൾ നീക്കിവയ്ക്കാൻ കഴിയാതെ പാഴാക്കപ്പെടുന്നു.

താഴെ കൊടുത്ത വാചകം സ്ട്രിങ്ങുകളെ ഉപയോഗിക്കുന്ന രീതി വിവരിക്കുന്നു.

```
for (i=0; i<7; i++)
    cout<<week[i];
```

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. ഡൈനാമിക് അറെ എന്നാൽ എന്ത്?
2. ഒരു അറെയിലെ ഒന്നാമത്തെ ലൊക്കേഷന്റെ വിലാസത്തെ ________ എന്ന് പറയുന്നു.
3. arr ഒരു ഇന്റിജർ അറെയാണെങ്കിൽ താഴെ കൊടുത്തവയിൽ തെറ്റേത്?

   a. `cout<<arr;` b. `arr++;` c. `cout<<*(arr+1);`

   d. `cin>>arr;` e. `arr=1500;` f. `cout<<*arr * 2;`
4. 10 പുസ്തകങ്ങളുടെ പേരുകൾ പോയിന്റർ ഉപയോഗിച്ച് സൂചിപ്പിക്കാനുള്ള C++ പ്രഖ്യാപന പ്രസ്താവന എഴുതുക.
5. ഒരു പോയിന്റർ പ്രഖ്യാപിച്ച് അതിൽ നിങ്ങളുടെ പേര് പ്രാരംഭവിലയായി ചേർക്കുക.

1.7 പോയിന്ററും സ്ട്രക്ചറും (Pointer and structure)

ഈ അധ്യായത്തിന്റെ തുടക്കത്തിൽ നമ്മൾ സ്ട്രക്ചർ ഡാറ്റ ഇനത്തെക്കുറിച്ചും അതിന്റെ ഉപയോഗത്തെക്കുറിച്ചും ചർച്ച ചെയ്തു. ഈ ഭാഗത്ത് നമ്മൾ സ്ട്രക്ചർ വേരിയബിളിനെ പോയിന്ററിന്റെ സഹായത്താൽ എങ്ങനെ ഉപയോഗിക്കാമെന്ന് പഠിക്കുന്നു. ഒരു employee സ്ട്രക്ചർ ചുവടെ ചേർക്കുന്നു.

```
struct employee
{
    int ecode;
    char ename[15];
    float salary;
};
```

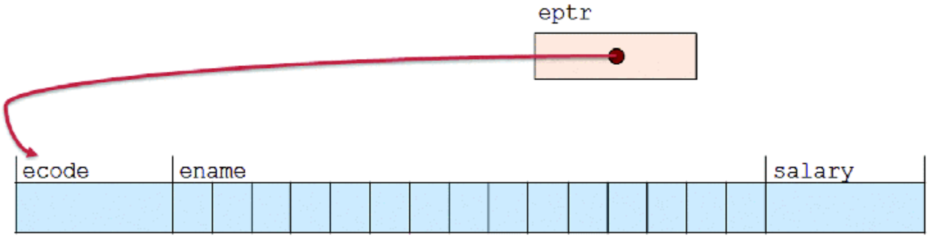
ഇനി ചുവടെ കൊടുത്തിരിക്കുന്ന പ്രഖ്യാപന പ്രസ്താവന പരിശോധിക്കാം.

```
employee *eptr;
```

ഇവിടെ `eptr` എന്ന പോയിന്ററിന് ഒരു `employee` സ്ട്രക്ചറിന്റെ അഡ്രസ്സ് ഉൾക്കൊള്ളാൻ കഴിയും. ഇനി താഴെ കൊടുത്ത പ്രസ്താവന ശ്രദ്ധിക്കുക.

```
eptr = new employee;
```

ഈ വാചകം 23 ബൈറ്റ് മെമ്മറി നീക്കിവയ്ക്കുകയും ചെയ്യുകയും അതിന്റെ ബേസ് അഡ്രസ്സ് `eptr` എന്ന പോയിന്ററിൽ ശേഖരിക്കുകയും ചെയ്യും. ചിത്രം 1.11 ൽ ഇക്കാര്യം വിവരിക്കുന്നു.



ചിത്രം 1.11: employee ടൈപ്പ് ഡാറ്റയുടെ ഡൈനാമിക് അലോക്കേഷൻ

സ്ട്രക്ചർ അംഗങ്ങളെ താഴെ കൊടുത്ത വിധമാണ് ഉപയോഗിക്കുന്നത് എന്ന് നമുക്കറിയാം.

```
structure_variable.element_name
```

ecode, ename, salary എന്നീ അംഗങ്ങളെ ഉപയോഗിക്കാൻ നമുക്ക് ഇവിടെ സ്ട്രക്ചർ വേരിയബിൾ ഇല്ല. eptr എന്ന പോയിന്ററിലൂടെയാണ് ഇവയെ ഉപയോഗിക്കേണ്ടത്. ഇതിന്റെ വാക്യഘടന ചുവടെ ചേർക്കുന്നു.

```
structure_pointer->element_name
```

ശ്രദ്ധിക്കുക, സ്ട്രക്ചർ പോയിന്ററും അംഗങ്ങളും '->' ഓപ്പറേറ്ററിലൂടെ ബന്ധിപ്പിക്കപ്പെട്ടിരിക്കുന്നു. ഈ ഓപ്പറേറ്റർ '-' നു പുറകെ '>' ചിഹ്നം കൂടി ഉപയോഗിച്ചാണ് എഴുതുന്നത്. ചുവടെ കൊടുത്ത പ്രസ്താവനകൾ ചിത്രം 1.11 ലെ സ്ട്രക്ചർ അംഗങ്ങളെ ഉപയോഗിക്കുന്നത് എങ്ങനെയെന്ന് ശ്രദ്ധിക്കുക.

```
eptr->ecode = 657346;      //employee code ന് വില നൽകുന്നു
gets(eptr->ename);         //employee യുടെ പേര് ഇൻപുട്ട് ചെയ്യുന്നു
cin>> eptr->salaray;       //employee യും സാലറി ഇൻപുട്ട് ചെയ്യുന്നു
cout<< eptr->salary * 0.12; //സാലറിയുടെ 12% പ്രദർശിപ്പിക്കുന്നു
```

നമുക്കു ചെയ്യാം

1.3.1 എന്ന ഭാഗത്തിൽ നമ്മൾ cx_ptr എന്ന പോയിന്റർ വേരിയബിൾ complex ടൈപ്പ് സ്ട്രക്ചറിൽ ഉപയോഗിച്ചു. ഒരു complex നമ്പർ ഇൻപുട്ട് ചെയ്ത് അതിനെ യഥാർഥ രൂപത്തിൽ പ്രദർശിപ്പിക്കാനുള്ള C++ പ്രസ്താവനകൾ സ്ട്രക്ചർ പോയിന്റർ ഉപയോഗിച്ച് എഴുതുക.

നമുക്ക് ഒരു പുതിയ അംഗത്തെ കൂട്ടിച്ചേർത്തുകൊണ്ട് employee സ്ട്രക്ചറിനെ പരിഷ്കരിക്കാം.

```
struct employee
{
    int ecode;
    char ename[15];
    float salary;
    int *ip;
};
```

തീർച്ചയായും ip എന്ന അംഗം ഒരു ഇന്റിജർ പോയിന്ററാണ്. ഇതിന് ഒരു ഇന്റിജർ സ്ഥാനത്തിന്റെ വിലാസം സൂക്ഷിക്കാം. ചുവടെ കൊടുത്ത പ്രസ്താവനകൾ പോയിന്റർ ip യുടെ ഉപയോഗം വിവരിക്കുന്നു.

```
eptr->ip = new int(5); /* ഡൈനാമിക് നീക്കിവയ്ക്കൽ നടത്തി ip യെ
ഒരു ഇന്റിജർ ലൊക്കേഷനുമായി ബന്ധിപ്പിച്ച് അതിൽ 5 ശേഖരിക്കുന്നു */ .
cout << *(eptr->ip);    // 5 എന്ന വില പ്രദർശിപ്പിക്കുന്നു
int n = eptr->*ip+1;    // 5 നോട് 1 കൂട്ടി n ൽ ശേഖരിക്കുന്നു
```

നോക്കുക ip പോയിന്റ് ചെയ്യുന്ന വില രണ്ട് രീതിയിൽ പുറത്തെടുക്കാം. *(eptr->ip) അല്ലെങ്കിൽ eptr->*ip. ഒരു സ്ട്രക്ചറിന് ഏത് ഇനത്തിലുള്ള പോയിന്ററിനെയും അംഗങ്ങളാക്കാം. വേണമെങ്കിൽ ഒരു സ്ട്രക്ചറിന്റെ തരത്തിൽ തന്നെയുള്ള പോയിന്ററിനെ അതിന് അംഗമാക്കാം.

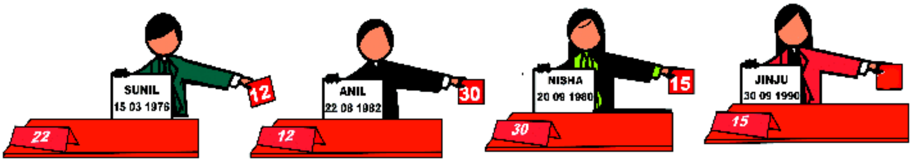
```
struct employee
{
    int ecode;
    char ename[15];
    float salary;
    employee *ep;
};
```

ep എന്ന അംഗം employee ഇനത്തിലുള്ള പോയിന്റർ ആണ്.

ഇപ്പോൾ സ്ട്രക്ചർ employee യെ സ്വയം സൂചിത സ്ട്രക്ചർ (Self referential structure) എന്ന് വിളിക്കുന്നു. ഇതിനെക്കുറിച്ച് നമുക്കു വിശദമായി ചർച്ച ചെയ്യാം.

സ്വയം സൂചിത സ്ട്രക്ചർ (Self referential structure)

സ്വയം സൂചിത സ്ട്രക്ചർ ഒരു പ്രത്യേക തരം സ്ട്രക്ചറാണ്. ഇതിൽ ഇതേ സ്ട്രക്ചർ ഇനത്തിലുള്ള ഒരു പോയിന്റർ വേരിയബിൾ ഉണ്ടായിരിക്കും. ഈ പോയിന്റിന് ഇതേ പോലുള്ള മറ്റൊരു സ്ട്രക്ചറിലേക്ക് പോയിന്റ് ചെയ്യാം. ഇങ്ങനെ നിർമിക്കുന്ന കണ്ണി എത്ര വേണമെങ്കിലും ദീർഘിപ്പിക്കാം. ചിത്രം 1.12 ഈ ആശയം വിവരിക്കുന്നു.



ചിത്രം 1.12: ഒരു employee സ്ട്രക്ചർ മറ്റൊന്നിനെ പോയിന്റ് ചെയ്യുന്നു

"Sunil" എന്ന എംപ്ലോയി "Anil" എന്ന എംപ്ലോയിയെ പോയിന്റ് ചെയ്യുന്നു. അയാളുടെ ടേബിൾ നമ്പർ 12 ആണ്. "Anil" പോയിന്റ് ചെയ്യുന്നത് "Nisha" എന്ന എംപ്ലോയിയെയാണ് ഇത് തുടരുന്നു.

C, C++ ഭാഷകളിൽ ലിങ്ക്ഡ്ലിസ്റ്റ്, ട്രീ തുടങ്ങിയ ഡയനാമിക് ഡാറ്റാ സ്ട്രക്ചറുകളുടെ നിർമാണത്തിന് സഹായകമായ ഒരു പ്രധാന ടൂൾ ആണ് സ്വയം സൂചിത സ്ട്രക്ചർ. പ്രോഗ്രാമിന്റെ പ്രവർത്തന സമയത്ത് നീക്കി വയ്ക്കുന്ന മെമ്മറി സ്ഥാനങ്ങളിലെ ഡാറ്റയുടെ കൂട്ടത്തെ ഡൈനാമിക് ഡാറ്റാ സ്ട്രക്ചർ എന്നു വിളിക്കുന്നു. ലിങ്ക്ഡ് ലിസ്റ്റ് ഡാറ്റസ്ട്രക്ചറിനെക്കുറിച്ച് കൂടുതലായി അധ്യായം 3 ൽ ചർച്ച ചെയ്യുന്നു.

നമുക്കു സംഗ്രഹിക്കാം

C++ ലെ നൂതനമായ ഡാറ്റ ഇനങ്ങളെ കുറിച്ച് നാം ചർച്ച് ചെയ്തു. ഏകോപിത ഡാറ്റയെ ഒരു പേരിൽ പ്രതിനിധാനം ചെയ്യാൻ സ്ട്രക്ചർ ഡാറ്റ ഇനം ഉപയോഗിച്ചു. സ്ട്രക്ചർ അംഗങ്ങളെ ഡോട്ട് ഓപ്പറേറ്ററി (.) ലൂടെ ഉപയോഗിക്കുന്നത് നാം ചർച്ച ചെയ്തു. പോയിന്റർ പ്രത്യേക ഡാറ്റ ഇനം ആണെന്ന് മനസ്സിലാക്കി. പോയിന്ററുമായി ബന്ധപ്പെട്ട ഓപ്പറേഷനുകൾ പദപ്രയോഗത്തിന്റെ സഹായത്തോടെ വിവരിച്ചു. ഡൈനാമിക് മെമ്മറി അലോക്കേഷൻ എന്ന ആശയവും, ആവശ്യമായ ഓപ്പറേറ്ററുകളും നേട്ടങ്ങളും ചർച്ച ചെയ്തു. അറെയും പോയിന്ററും തമ്മിലുള്ള ബന്ധം വിവരിച്ചു. സ്ട്രിങ് ഡാറ്റ പോയിന്ററിലൂടെ ഉപയോഗിക്കുന്നതും പോയിന്ററിനെയും സ്ട്രക്ചറിനെയും ബന്ധിപ്പിക്കുന്ന വിധവും മനസ്സിലാക്കി. ഈ അധ്യായത്തിൽ പഠിച്ച പല കാര്യങ്ങളും അധ്യായം - 3 പഠിക്കുന്നതിനുള്ള അടിസ്ഥാനമായി ഉപയോഗിക്കാം.

നമുക്കു പരിശീലിക്കാം

1. ടെലിഫോൺ വരിക്കാരന്റെ വിവരങ്ങൾ അടങ്ങുന്ന ഒരു സ്ട്രക്ചർ നിർവചിക്കുക. അതിൽ പേര്, ടെലിഫോൺ നമ്പർ എന്നീ അംഗങ്ങൾ ഉണ്ടായിരിക്കണം. ഇതുപയോഗിച്ച് പേര് നൽകിയാൽ നമ്പർ തരുന്നതും, അല്ലെങ്കിൽ നമ്പർ നൽകിയാൽ പേര് തരുന്നതുമായ ഒരു മെനു നിയന്ത്രിതമായ C++ പ്രോഗ്രാം തയാറാക്കുക.
2. ഒരു ബാങ്കിലെ ഉപഭോക്താവിന്റെ വിവരങ്ങൾ അടങ്ങുന്ന സ്ട്രക്ചർ നിർവചിക്കുക. അക്കൗണ്ട് നമ്പർ, പേര്, അക്കൗണ്ട് ഓപ്പണിംഗ് തീയതി, ബാലൻസ് എന്നീ വിവരങ്ങൾ ഉണ്ടായിരിക്കണം. ഇതുപയോഗിച്ച് നിക്ഷേപം, പിൻവലിക്കൽ, ഫലം കാണുക എന്നീ പ്രവൃത്തികൾ നടത്താനാവശ്യമായ ഒരു മെനു നിയന്ത്രിത പ്രോഗ്രാം തയാറാക്കുക. നിക്ഷേപവും പിൻവലിക്കലും നടക്കുമ്പോൾ ബാലൻസ് സംഖ്യയിൽ മാറ്റം വരണം. അക്കൗണ്ടിൽ മിനിമം ബാലൻസായി Rs. 1000/- ഉണ്ടായിരിക്കണം (അക്കൗണ്ട് ആരംഭിച്ച തീയതി നെസ്റ്റെഡ് സ്ട്രക്ചറായി ഉൾപ്പെടുത്താം).
3. വിദ്യാർഥികൾക്ക് കമ്പ്യൂട്ടർ സയൻസിൽ കിട്ടിയ TE സ്കോർ അവരോഹണ ക്രമത്തിൽ ക്രമീകരിക്കുന്നതിനു പോയിന്റർ ഉപയോഗിച്ച് ഒരു C++ പ്രോഗ്രാം എഴുതുക.
4. ക്യാരക്ടർ പോയിന്റർ ഉപയോഗിച്ച് ഇൻപുട്ട് ചെയ്ത ഒരു സ്ട്രിങ് പാലിൻഡ്രോം ആണോ അല്ലയോ എന്ന് പരിശോധിക്കാനുള്ള ഒരു C++ പ്രോഗ്രാം തയാറാക്കുക.
5. പോയിന്റർ ഉപയോഗിച്ച്, നിങ്ങളുടെ ക്ലാസ്സിലെ എല്ലാ കുട്ടികളുടെയും പേരുകൾ ഇൻപുട്ട് ചെയ്യുക. ഇതിൽനിന്നും ഒരു റോൾ ലിസ്റ്റ് നിർമിക്കുക. ഈ ലിസ്റ്റിൽ പേരുകൾ അക്ഷരമാല ക്രമത്തിലും റോൾ നമ്പർ 1, 2, 3 ... എന്ന ക്രമത്തിലും വരണം.
6. രജിസ്റ്റർ നമ്പർ, പേര്, ആറ് വിഷയത്തിലെ CE മാർക്കുകൾ എന്നീ വിവരങ്ങൾ അടങ്ങുന്ന ഒരു student സ്ട്രക്ചർ നിർവചിക്കുക. സ്ട്രക്ചർ പോയിന്റർ ഉപയോഗിച്ച് ഈ വിവരങ്ങൾ ഇൻപുട്ട് ചെയ്യാനും രജിസ്റ്റർ നമ്പർ, പേര്, സി.ഇ. മാർക്കുകളുടെ തുക എന്നിവ പ്രദർശിപ്പിക്കാനും വേണ്ട പ്രോഗ്രാം എഴുതുക.

നമുക്കു വിലയിരുത്താം

1. C++ ലെ അറെയും സ്ട്രക്ചറും താരതമ്യം ചെയ്യുക.

2. താഴെ കൊടുത്തിരിക്കുന്ന സ്ട്രക്ചർ നിർവചനത്തിലെ തെറ്റുകൾ കണ്ടെത്തി അതിനുള്ള കാരണമെഴുതുക.

```
struct
{
    int roll, age;
    float fee=1000;
};
```

3. താഴെ കൊടുത്തിരിക്കുന്ന സ്ട്രക്ചർ നിർവചനം വായിച്ച് ചോദ്യങ്ങൾക്ക് ഉത്തരം എഴുതുക.

```
struct book
{
    int book_no;
    char bk_name[20];
    struct
    {
        short dd;
        short mm;
        short yy;
    }dt_of_purchase;
    float price;
};
```

   a. ഈ സ്ട്രക്ചറിലെ വിവരങ്ങൾ സൂചിപ്പിക്കാൻ book ടൈപ്പിൽ ഒരു വേരിയബിൾ പ്രഖ്യാപിക്കാനുള്ള C++ പ്രസ്താവന എഴുതുക. വേരിയബിളിന് എത്ര മെമ്മറി ആവശ്യമുണ്ട്. ന്യായീകരിക്കുക.

   b. നിങ്ങളുടെ കമ്പ്യൂട്ടർ സയൻസ് പാഠപുസ്തകത്തിലെ വിവരങ്ങൾ ഉപയോഗിച്ച് ഈ സ്ട്രക്ചറിന് പ്രാരംഭവില നൽകാനുള്ള C++ പ്രസ്താവന എഴുതുക.

   c. book ന്റെ വിവരങ്ങൾ പ്രദർശിപ്പിക്കാൻ ആവശ്യമായ C++ പ്രസ്താവനകൾ എഴുതുക.

   d. "അകത്തുള്ള സ്ട്രക്ചർ ടാഗിന്റെ അഭാവം ഒരു തെറ്റും വരുത്തുന്നില്ല" - ഈ വാചകം ശരിയോ തെറ്റോ എന്ന് കാരണസഹിതം സമർഥിക്കുക.

4. "സ്ട്രക്ചർ ഒരു ഉപഭോക്തൃനിർവചിത ഡാറ്റ ഇനമാണ്" - ഉദാഹരണ സഹിതം സമർഥിക്കുക.

5. താഴെ കൊടുത്തിരിക്കുന്ന വാചകങ്ങൾ വായിക്കുക:

i. C++ ൽ സ്ട്രക്ചർ നിർവചിക്കുമ്പോൾ ടാഗ് ഒഴിവാക്കാം.

ii. ഒരു സ്ട്രക്ചർ വേരിയബിളിലെ ഡാറ്റ മറ്റൊരു സ്ട്രക്ചർ വേരിയബിളിലേക്ക് കോപ്പി ചെയ്യണമെങ്കിൽ രണ്ട് സ്ട്രക്ച്ചർ വേരിയബിളുകളും ഒരേ ടാഗിൽ നിർവചിച്ചിരിക്കണം.

iii. സ്ട്രക്ചർ അംഗങ്ങളെ structure_name.element എന്ന രീതിയിലാണ് സൂചിപ്പിക്കുന്നത്.

iv. ഒരു സ്ട്രക്ചറിന് മറ്റൊരു സ്ട്രക്ചറിനെ ഉൾക്കൊള്ളാം.

ഇനി താഴെ കൊടുത്തവയിൽ ശരിയായത് തിരഞ്ഞെടുക്കുക.

a. (i) ഉം (ii) ഉം വാചകങ്ങൾ ശരിയാണ്

b. (ii) ഉം (iv) ഉം വാചകങ്ങൾ ശരിയാണ്

c. (i) ഉം, (ii) ഉം, (iv) ഉം വാചകങ്ങൾ ശരിയാണ്

d. എല്ലാ വാചകങ്ങളും ശരിയാണ്

6. ചുവടെ ചേർത്തിരിക്കുന്ന C++ പ്രസ്താവനകൾ വായിക്കുക.

```
int * p, a=5;
p=&a;
```

a. p എന്ന വേരിയബിളിന്റെ പ്രത്യേകതയെന്ത്?

b. രണ്ടാമത്തെ വാചകം പ്രവർത്തിപ്പിച്ചതിനുശേഷം p യുടെ ഉള്ളടക്കം എന്തായിരിക്കും.

c. `*p+1, *(p+1)` എന്നീ പദപ്രയോഗങ്ങൾ എങ്ങനെ വ്യത്യാസപ്പെട്ടിരിക്കുന്നു?

7. ചുവടെ കൊടുത്തിരിക്കുന്ന C++ പ്രോഗ്രാം ശകലത്തിലെ തെറ്റ് കണ്ടുപിടിച്ച് കാരണമെഴുതുക.

```
int *p,*q, a=5;
float b=2;
p=&a;
q=&b;
cout<<p<<*p<<*a;
if (p<q)cout<<p;
   cout<<*p * a;
```

8. ഒരു പ്രോഗ്രാം തയാറാക്കുമ്പോൾ ഡൈനാമിക് അലോക്കേഷൻ രീതി ഉപയോഗിച്ചു. എന്നാൽ delete ഓപ്പറേറ്റർ ഉപയോഗിച്ചുള്ള വാചകം ആ പ്രോഗ്രാമിൽ ഇല്ലായിരുന്നു. ഇത് സൃഷ്ടിക്കുന്ന പ്രശ്നം വിവരിക്കുക.

9. താഴെ കൊടുത്തിരിക്കുന്ന C++ പ്രസ്താവനകൾ വായിച്ച് ചോദ്യങ്ങൾക്ക് ഉത്തരമെഴുതുക.

```
int mark[] = {34, 12, 25, 56, 38};
int *p = mark;
```

a. p യുടെ വിലയെന്ത്

b. `*p + *(ar+2)` എന്ന പദപ്രയോഗത്തിന്റെ വില എന്താണ്?

c. `ar++;` എന്ന വാചകം തെറ്റാണ് - എന്തുകൊണ്ട്? ഇത് `p++;` എന്ന വാചകത്തിൽ നിന്നും എങ്ങനെ വ്യത്യാസപ്പെട്ടിരിക്കുന്നു.

10. താഴെ കൊടുത്തിരിക്കുന്ന പ്രോഗ്രാം ശകലത്തിന്റെ പ്രവർത്തനം വിവരിച്ച് ഔട്ട്പുട്ട് കണ്ടെത്തുക.

```
char *str = "Tobacco Kills";
for (int i=0; str[i]!='\0'; i++)
    if (i>8)
        *(str+i) = toupper(*(str+i));
cout<<str;
```

11. താഴെ കൊടുത്ത C++ പ്രസ്താവനകൾ ശ്രദ്ധിക്കുക.

```
int ar[] = {14, 29, 32, 63, 30};
```

ചുവടെ ചേർത്തവയിൽ ഒരു വാചകം 32 എന്ന അംഗത്തെ നൽകുന്നില്ല. ഏതാണത്?

a. `ar[2]` b. `ar[*ar%3]` c. `*ar+2` d. `*(ar+2)`

12. `new`, `delete` എന്നീ ഓപ്പറേറ്ററുകളുടെ പ്രവർത്തനങ്ങൾ ഉദാഹരണ സഹിതം വിവരിക്കുക.

13. മെമ്മറി ലീക്ക് എന്നാലെന്ത്? എന്താണ് ഇതിന് കാരണം? ഈ സാഹചര്യം എങ്ങനെ ഒഴിവാക്കാം?

14. ചുവടെ കൊടുത്തിരിക്കുന്ന പ്രസ്താവനകൾ താരതമ്യം ചെയ്യുക.

```
int a=5;
int *a=new int(5);
```

15. ചുവടെ കൊടുത്തിരിക്കുന്ന സ്ട്രക്ചർ നിർവചനം വായിച്ച് ചോദ്യങ്ങൾക്ക് ഉത്തരമെഴുതുക.

```
struct sample
{
    int num;
    char *str;
}   *sptr;
```

a. `sample` ഡാറ്റാ തരത്തിലെ ഒരു ലൊക്കേഷനിൽ ഡൈനാമിക് അലോക്കേഷൻ നടത്തി അഡ്രസ്സ് sptr ൽ ശേഖരിക്കാൻ വേണ്ട C++ പ്രസ്താവന എഴുതുക.

b. `sptr` പോയിന്റ് ചെയ്യുന്ന മെമ്മറി സ്ഥാനത്തേക്ക് ഡാറ്റ ഇൻപുട്ട് ചെയ്യാനുള്ള C++ പ്രസ്താവനകൾ എഴുതുക.

c. ഈ സ്ട്രക്ചറിനെ സ്വയം സൂചിത സ്ട്രക്ചറായി നവീകരിക്കുക.

D2P5S8

2 ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ്ങിന്റെ തത്വങ്ങൾ

പ്രധാന പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പഠനം പൂർത്തിയാകുന്നതോടെ പഠിതാവ് ആർജിക്കേണ്ട പഠനനേട്ടങ്ങൾ

- വിവിധ പ്രോഗ്രാമിങ് മാതൃകകൾ താരതമ്യം ചെയ്യാൻ സാധിക്കുന്നു.
- പ്രൊസിജർ ഓറിയന്റഡ് പ്രോഗ്രാമിങ് മാതൃകയുടെ വിവിധ സവിശേഷതകൾ സൂചിപ്പിക്കാൻ സാധിക്കുന്നു.
- ഒബ്ജക്ട് ഓറിയന്റഡ് പ്രോഗ്രാമിങ് മാതൃകയുടെ ഗുണങ്ങൾ സൂചിപ്പിക്കാൻ കഴിയുന്നു.
- ഡാറ്റ അബ്സ്ട്രാക്ഷൻ, ഡാറ്റ എൻക്യാപ്സുലേഷൻ എന്നീ ആശയങ്ങൾ ഉദാഹരണസഹിതം വിശദീകരിക്കാൻ സാധിക്കുന്നു.
- ഇൻഹെറിറ്റൻസ്, പോളിമോർഫിസം എന്നിവ യഥാർഥ ജീവിത ഉദാഹരണങ്ങൾ ഉപയോഗിച്ച് വിശദീകരിക്കാൻ സാധിക്കുന്നു.

കമ്പ്യൂട്ടറുകളെ കൂടുതൽ ഉപയോഗപ്രദമാക്കുന്നതിനായി നമ്മൾ പ്രോഗ്രാമുകൾ (Software) വികസിപ്പിക്കുന്നു.അതിനായി വിവിധ പ്രോഗ്രാമിങ് ഭാഷകൾ നമ്മൾ ഉപയോഗിക്കുന്നു. പ്രോഗ്രാമിന്റെ വലിപ്പം കൂടുന്തോറും അത് നിർമിക്കുവാനുള്ള ബുദ്ധിമുട്ടും കൂടുന്നു. ഈ ബുദ്ധിമുട്ട് ഒഴിവാക്കുന്നതിനായി IDE, ഡീബഗ്ഗർ, കംപൈലർ എന്നിങ്ങനെ വിവിധ ഉപാധികൾ നമുക്ക് ഉപയോഗിക്കാം. കൂടാതെ വിവിധ നിർമ്മാണ സമീപന രീതികളായ മോഡുലാർ, സ്ട്രക്ച്ചർഡ്, പ്രൊസീജറൽ, ഒബ്ജക്റ്റ് ഓറിയന്റഡ് എന്നിവ സോഫ്റ്റ്വെയർ നിർമ്മാണത്തിന് നമുക്ക് പിന്തുടരാവുന്നതാണ്. ഈ ഉപാധികളും സമീപനങ്ങളും സോഫ്റ്റ്വെയർ നിർമ്മാണ സമയത്തെ സോഫ്റ്റ്വെയർ ഉൽപ്പന്നങ്ങളുടെ പരിപാലനം (Maintainabilitiy), പുനരുപയോഗം (Reusabilitiy), വഹനീയത, (Portabilitiy), സുരക്ഷിതത്വം, സമഗ്രത (Integritiy), ഉപയോക്തസൗഹൃദം മുതലായ പ്രശ്നങ്ങളെ അഭിമുഖീകരിക്കാൻ സഹായിക്കുന്നു.

മുൻ അധ്യായത്തിൽ വിവിധ പ്രോഗ്രാമിങ് ആശയങ്ങളെപറ്റിയും C++ ഭാഷ ഉപയോഗിച്ച് പ്രശ്നപരിഹാരത്തിനായി പ്രോഗ്രാമുകൾ നിർമ്മിക്കുന്നതിനെപ്പറ്റിയും നമ്മൾ പഠിച്ചതാണ്. ഇൻപുട്ടുകളെ പ്രോസസ്സ് ചെയ്ത് ഔട്ട്പുട്ട് ലഭ്യമാകുക എന്നതായിരുന്നു അവിടെയെല്ലാം നമ്മുടെ ലക്ഷ്യം. തയാറാക്കിയ പ്രോഗ്രാമുകൾ ചെറുതും ലളിതവും ആയിരുന്നതിനാൽ കൈക്കൊണ്ട സമീപനങ്ങളെ പറ്റിയോ ഡാറ്റയുടെ സുരക്ഷിതത്വത്തെ പറ്റിയോ നമ്മൾ ഒരിക്കലും ചിന്തിച്ചിരുന്നില്ല.

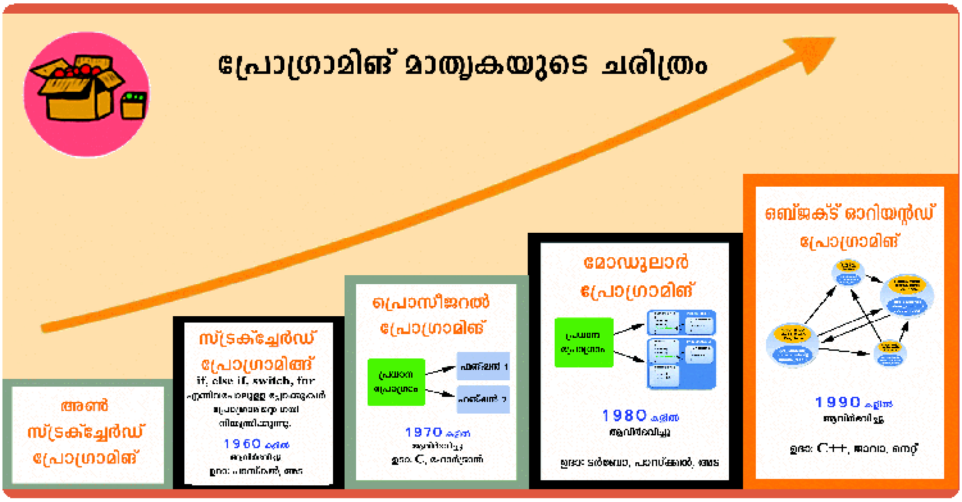
എന്നാലും പ്രശ്നപരിഹാരത്തിനായി പ്രോഗ്രാമുകൾ തയ്യാറാക്കുമ്പോൾ അറിഞ്ഞോ അറിയാതെയോ നാം ഒരു സമീപനം പിന്തുടർന്നു കൊണ്ടിരുന്നു. പ്രോഗ്രാം വികസിപ്പിക്കുവാൻ ഏറ്റവും ഉചിതമായ സമീപനം തിരഞ്ഞെടുക്കുന്നതിന് നമ്മെ പ്രാപ്തരാക്കുന്നതിനായി ഈ അധ്യായത്തിൽ, ഇതുവരെ നാം ഉപയോഗിച്ച പ്രോഗ്രാമിങ് സമീപനത്തെപ്പറ്റി ചർച്ച ചെയ്യുകയും പുതിയ സമീപനം അവതരിപ്പിക്കുകയും ചെയ്യുന്നു.

2.1 പ്രോഗ്രാമിങ് മാതൃക (Programming paradigm)

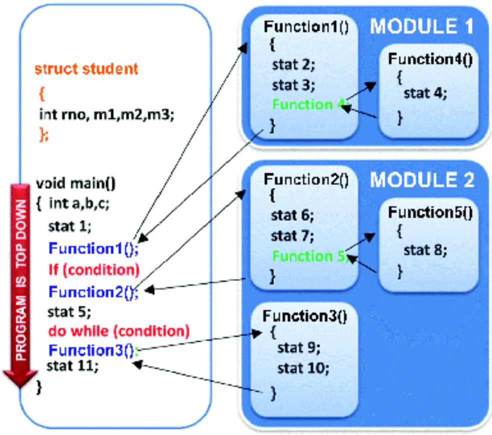
പ്രോഗ്രാമിങ് മാതൃക എന്നാൽ ഒരു പ്രോഗ്രാം എങ്ങനെ ചിട്ടപ്പെടുത്തിയിരിക്കുന്നു എന്നതിനെ സൂചിപ്പിക്കുന്നതാണ്. പ്രോഗ്രാം വളരെ ചെറുതാണെങ്കിൽ അത് ചിട്ടപ്പെടുത്താൻ പ്രത്യേക തത്വമൊന്നും പാലിക്കേണ്ട കാര്യമില്ല. എന്നാൽ പ്രോഗ്രാം വലുതാകുംതോറും അതിന്റെ സങ്കീർണത കുറയ്ക്കുവാനും അതിനെ പരിപാലിക്കുന്നതിനും ചില മുൻകരുതലുകൾ എടുക്കേണ്ടി വരുന്നു.

ചില മാതൃകകൾ നടപടിക്രമങ്ങൾക്കു കൂടുതൽ പ്രാധാന്യം കൊടുക്കുമ്പോൾ മറ്റു ചിലത് ഡാറ്റയ്ക്ക് കൂടുതൽ പ്രാധാന്യം കൊടുക്കുന്നു. വിവിധ പ്രോഗ്രാമിങ് ഭാഷകളുടെ കാര്യക്ഷമതയും ശൈലിയുമെല്ലാം അവ സ്വീകരിക്കുന്ന പ്രോഗ്രാം മാതൃകകളാണ് നിർവചിക്കുന്നത്. മോഡുലാർ പ്രോഗ്രാമിങ്, ടോപ്-ഡൗൺപ്രോഗ്രാമിങ്, ബോട്ടം-അപ്പ് പ്രോഗ്രാമിങ്, സ്ട്രക്ച്ചർഡ് പ്രോഗ്രാമിങ് മുതലായ രീതികൾ നേരത്തെ പരീക്ഷിക്കപ്പെട്ടവയാണ്. ഇവ ഓരോന്നും പ്രോഗ്രാമിങ്ങിന്റെ സങ്കീർണത കുറയ്ക്കുവാനും വിശ്വസ്തവും പരിപാലന യോഗ്യവുമായ പ്രോഗ്രാമുകൾ നിർമ്മിക്കുവാനും വേണ്ടിയാണ് ഉപയോഗിച്ചത്.

ഒരു പ്രോഗ്രാമിങ് മാതൃക മാത്രം പിന്തുടരുന്ന രീതിയിലാണ് ചില പ്രോഗ്രാമിങ് ഭാഷകൾ രൂപകൽപ്പന ചെയ്തിരിക്കുന്നത്. എന്നാൽ മറ്റു ചിലത് ഒന്നിലധികം മാതൃകകൾ പിന്തുടരുന്നു. C++ ഒന്നിലധികം പ്രോഗ്രാമിങ് മാതൃകകൾ പ്രയോജനപ്പെടുത്തുന്ന ഭാഷയാണ്. C++ ഉപയോഗിച്ച് ഏറ്റവും പ്രധാനപ്പെട്ട രണ്ടു മാതൃകകളായ പ്രോസിജറൽ മാതൃകയും (Procedural paradigm) ഒബ്ജക്റ്റ് ഓറിയന്റഡ് മാതൃകയും (Object Oriented Paradigm) നമുക്ക് പ്രാവർത്തികമാക്കാവുന്നതാണ്. ഇവ ഓരോന്നിനെയും പറ്റി നമുക്ക് വിശദമായി ചർച്ച ചെയ്യാം.



2.1.1 പ്രോസിജർ ഓറിയന്റഡ് പ്രോഗ്രാമിങ് മാതൃക (Procedure-Oriented Programming paradigm)



ചിത്രം 2.1 പ്രോസിജറൽ മാതൃക

പ്രോസിജർ ഓറിയന്റഡ് പ്രോഗ്രാമിങ് എന്നത് ഒരു പ്രോഗ്രാം നിർമ്മിക്കാൻ ആവശ്യമായ, നല്ല രീതിയിൽ ക്രമീകരിച്ച ഘട്ടങ്ങളുടെ ശ്രേണിയെയും പ്രവർത്തനങ്ങളെയും സൂചിപ്പിക്കുന്നു. ഒരു പ്രവൃത്തിയെ അല്ലെങ്കിൽ പ്രോഗ്രാം പൂർത്തീകരിക്കുന്നതിനായുള്ള ചിട്ടയായ ക്രമീകരണങ്ങളോട് കൂടിയ പ്രസ്താവനകളും നടപടികളും നിർദേശങ്ങളും ഇവയിൽ അടങ്ങിയിരിക്കും. ഈ നിർദേശങ്ങൾ ഇൻപുട്ട് സ്വീകരിക്കാനുള്ളതോ, ഗണിതപരമോ യുക്തിപരമോ ആയ ക്രിയകൾ നിർവഹിക്കാനുള്ളതോ അതുമല്ലെങ്കിൽ പരിണിതഫലം പ്രദർശിപ്പിക്കാനുള്ളതോ ആവാം. ഈ സമീപനത്തിൽ കാര്യങ്ങൾ ചെയ്യുന്നതിനാണ് ഊന്നൽ നൽകിയിരിക്കുന്നത്. ഈ മാതൃകയിൽ പ്രോഗ്രാം സങ്കീർണവും വലുപ്പം കൂടിയതുമാകുമ്പോൾ നിർദേശങ്ങളുടെ പട്ടികയെ വിഭജിച്ചു നിർദേശങ്ങളെ ഫങ്ഷനുകളായി രൂപപ്പെടുത്തുന്നു. പ്രോഗ്രാമിലെ മറ്റു ഫങ്ഷനുകളുമായുള്ള സമ്പർക്ക മാർഗത്തെയും അതിന്റെ ആവശ്യകതയെയും ഫങ്ഷൻ വ്യക്തമായി നിർവചിക്കുന്നു. അതുവഴി പ്രോഗ്രാമിന്റെ സങ്കീർണത കുറക്കാൻ ഫങ്ഷൻ സഹായിക്കുന്നു. സങ്കീർണത വീണ്ടും കുറയ്ക്കുന്നതിനായി പൊതുവായ ക്രിയകളുമായി ബന്ധപ്പെട്ട ഫങ്ഷനുകളെ മോഡ്യൂളുകളായി രൂപപ്പെടുത്തുന്നു. ഒരു വലിയ പ്രോഗ്രാമിനെ അഞ്ചു വ്യത്യസ്ത ഫങ്ഷനുകളായി വിഭജിക്കുകയും പൊതുവായ പ്രവർത്തനങ്ങൾ ഉൾപ്പെടുന്ന ഫങ്ഷനുകളെ കൂട്ടിച്ചേർത്തു രണ്ട് മൊഡ്യൂളുകളാക്കുകയും ചെയ്യുന്ന പ്രോസിജർ ഓറിയന്റഡ് മാതൃകയാണ് ചിത്രം 2.1-ൽ കാണിച്ചിരിക്കുന്നത്.

C, Pascal, FORTRAN, BASIC മുതലായവ പ്രോസിജർ ഓറിയന്റഡ് പ്രോഗ്രാമിങ് ഭാഷകളാണ്. C++ പ്രോഗ്രാമിങ് പഠന പ്രക്രിയയിൽ നമ്മൾ ഇതുവരെ പിൻതുടർന്നത് പ്രോസിജർ ഓറിയന്റഡ് മാതൃകയാണ്. പ്രോസിജർ ഓറിയന്റഡ് പ്രോഗ്രാമിങ് ഭാഷകളെ ടോപ് ഡൗൺ പ്രോഗ്രാമിങ് ഭാഷകൾ എന്നും വിളിക്കുന്നു.

പക്ഷെ പ്രോഗ്രാമിന്റെ വലുപ്പം കൂടുകയും സങ്കീർണത വർധിക്കുകയും ചെയ്യുന്നതോടെ അതിന്റെ പരിമിതികൾ വെളിപ്പെട്ടു തുടങ്ങുന്നു. ഈ സമീപനം എത്ര നല്ല രീതിയിൽ പ്രാവർത്തികമാക്കിയാലും വലിയ പ്രോഗ്രാമുകൾ സങ്കീർണമായിത്തന്നെ തുടരുന്നു. പ്രോസിജർ ഓറിയന്റഡ് ഭാഷകളുടെ സങ്കീർണത വർധിക്കുവാനുള്ള പ്രധാന കാരണങ്ങൾ താഴെ പറയുന്നവയാണ്.

a. ഡാറ്റയുടെ പ്രാധാന്യം കുറച്ചു കാണുന്നു.

b. പുതിയ ഡാറ്റ അംഗത്തെ കൂട്ടിച്ചേർക്കുമ്പോൾ എല്ലാ ഫങ്ഷനുകൾക്കോ അല്ലെങ്കിൽ ചില ഫങ്ഷനുകൾക്കോ മാറ്റം വേണ്ടി വരുന്നു.

c. പുതിയ ഡാറ്റ ഇനങ്ങൾ നിർമിക്കുക പ്രയാസകരമാണ്.

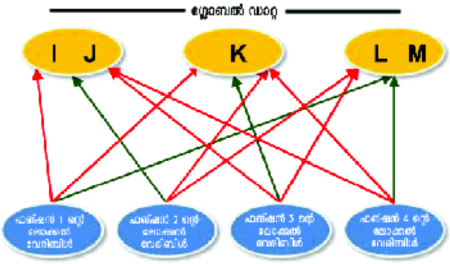
d. യഥാർഥ ജീവിത മാതൃകയ്ക്കു യോജിച്ചതല്ല.

നമുക്ക് ഇത് ഓരോന്നിനെക്കുറിച്ചും ഇവിടെ ചർച്ച ചെയ്യാം.

a. ഡാറ്റയുടെ പ്രാധാന്യം കുറച്ചു കാണുന്നു

പ്രോസിജർ ഓറിയന്റഡ് ഭാഷകൾ, കാര്യങ്ങൾ ചെയ്യുന്നതിനാണ് പ്രാമുഖ്യം നൽകുന്നത്. ഡാറ്റയ്ക്ക് പ്രാധാന്യം കുറവാണ്. ഒരു ഉദാഹരണത്തിന്റെ സഹായത്തോടെ ഇത് വിശദീ കരിക്കാം. സ്കൂളിലെ പ്രവർത്തനങ്ങൾ യന്ത്രവൽകരിക്കുന്നതിനായി ഒരു സോഫ്റ്റ് വെയർ വികസിപ്പിക്കുന്നു എന്ന് കരുതുക. ഒരു പുതിയ വിദ്യാർഥിയെ ഉൾപ്പെടുത്തുക, ഒരു വിദ്യാർഥിയെ നീക്കം ചെയ്യുക, ഫീ ശേഖരണ വിവരങ്ങൾ രേഖപ്പെടുത്തുക മുതലാ യവയാകാം ഇതിൽ ഉൾപ്പെടുന്ന പ്രവർത്തനങ്ങൾ. സോഫ്റ്റ്‌വെയറിൽ ഈ പ്രവർത്ത നങ്ങൾക്കായി പ്രത്യേക ഫങ്ഷനുകൾ ഉപയോഗിച്ചിരിക്കുന്നതായി എന്ന് അനുമാനിക്കുക. വിദ്യാർഥികളുടെ വിവരങ്ങൾ അടങ്ങിയ സ്ട്രക്ചറുകളുടെ (സ്റ്റുഡൻറ്) അറേ ഉപയോഗിച്ച് ഈ ഫംഗ്ഷനുകളുടെ പ്രവർത്തനത്തിനായുള്ള ഡാറ്റ സംഭരിക്കാവുന്നതാണ്. ഈ ഡാറ്റ എല്ലാ ഫംഗ്ഷനുകൾക്കും ലഭ്യമാക്കണമെങ്കിൽ അറേയെ ഗ്ലോബൽ ആയി നിർവചി ക്കേണ്ടി വരും (ഒന്നാം വർഷ പാഠപുസ്തകത്തിന്റെ ഭാഗം 10.5 നോക്കുക). ഇപ്പോൾ ഈ ഫംഗ്ഷനുകളെ കൂടാതെ പ്രോഗ്രാമിലുള്ള മറ്റ് ഏതു ഫങ്ഷനുകൾക്കു വേണമെങ്കിലും ഡാറ്റ ഉപയോഗിക്കാവുന്ന അവസ്ഥയായി. തന്മൂലം അറിഞ്ഞോ അറിയാതെയോ ഡാറ്റ യ്ക്ക് മാറ്റം സംഭവിക്കാനുള്ള കളമൊരുങ്ങുന്നു. ഇത് തടയുവാൻ നമുക്ക് യാതൊരു തരത്തിലുമുള്ള നിയന്ത്രണവും ഏർപ്പെടുത്താൻ സാധിക്കില്ല.

അടുത്ത ദിവസം സമർപ്പിക്കേണ്ട അസൈൻമെന്റ് തീൻ മേശപ്പുറത്തു വെച്ചിട്ടു പോകുന്നത് പോലെയാണിത്. ഈ അസൈൻമെന്റ് നാശമാകാനുള്ള സാധ്യത ഇവിടെ വളരെ കൂടു തലാണ്. ചെറിയ കുട്ടികളുണ്ടെങ്കിൽ അവർ അത് കീറുകയോ അല്ലെങ്കിൽ അതിൽ ചിത്ര ങ്ങൾ വരയ്ക്കുകയോ ചെയ്യാം. അതുമല്ലെങ്കിൽ മേശപ്പുറത്തു വച്ചിരിക്കുന്ന ചായ അബദ്ധ ത്തിൽ അതിന്റെ മേൽ മറിഞ്ഞു വീഴാം. ആർക്കു വേണമെങ്കിലും എടുക്കാവുന്ന രീതിയിൽ അസൈൻമെന്റ് വെക്കുന്നത് കൊണ്ടാണ് ഇത് സംഭവിക്കുന്നത്.



ചിത്രം 2.2: ഡാറ്റ ഉപയോഗപ്പെടുത്തുന്ന ഫങ്ഷനുകൾ

ഒരു പ്രോസീജറൽ ഓറിയന്റഡ് പ്രോഗ്രാമിങ് മാതൃ കയിലെ ലോക്കൽ വേരിയബിളുകൾ, ഗ്ലോബൽ വേരിയബിളുകൾ, ഫംങ്ഷനുകൾ മുതലായവയു ടെ ക്രമീകരണമാണ് ചിത്രം 2.2-ൽ കാണിച്ചിരി ക്കുന്നത്. ഡാറ്റ ആവശ്യമായ ഫങ്ഷനുകളെ പച്ച വരകൾ സൂചിപ്പിക്കുന്നു. ഡാറ്റയുടെ അനധികൃ തമായ ഉപയോഗത്തെ ചുവപ്പു വരകൾ സൂചിപ്പി ക്കുന്നു.

b. പുതിയ ഡാറ്റ അംഗത്തെ കൂട്ടിച്ചേർക്കുമ്പോൾ എല്ലാ ഫങ്ഷനുകൾക്കോ അല്ലെ ങ്കിൽ ചില ഫങ്ഷനുകൾക്കോ മാറ്റം വേണ്ടിവരുന്നു

പല ഫങ്ഷനുകളും ഗ്ലോബൽ ഡാറ്റ ഉപയോഗിക്കുന്നതിനാൽ ഡാറ്റ എങ്ങനെ സംഭരി ച്ചിരിക്കുന്നു എന്നത് ഏറെ പ്രാധാന്യമുള്ളതാണ്. ഡാറ്റ ഉപയോഗിക്കുന്ന ഫങ്ഷനുകളെ പരിഷ്കരിക്കാതെ ഡാറ്റയുടെ ക്രമീകരണത്തിൽ മാറ്റം വരുത്തുക സാധ്യമല്ല. നമ്മൾ പുതിയ ഡാറ്റ അംഗത്തെ കൂട്ടി ചേർക്കുമ്പോൾ ആ ഡാറ്റ ഉപയോഗിക്കുന്ന എല്ലാ ഫങ്ഷനുകളെയും പരിഷ്കരിക്കേണ്ടതുണ്ട്. എന്നാൽ മാത്രമേ ഈ ഫങ്ഷനുകൾക്കു പുതിയ ഡാറ്റ അംഗം ലഭ്യമാകൂ. ഇത്തരത്തിൽ ഒരു ഡാറ്റയുമായി ബന്ധപ്പെട്ട എല്ലാ

ഫങ്ഷനുകളെയും കണ്ടെത്തുക എന്നത് വളരെ ശ്രമകരമായ കാര്യമാണ് എന്ന് മാത്രമല്ല പുതിയ ഡാറ്റ ലഭ്യമാകത്തക്ക രീതിയിൽ അവയെ കൃത്യമായി പരിഷ്ക്കരിക്കുക എന്നത് അതിലും ബുദ്ധിമുട്ടേറിയതാണ്.

നമ്മുടെ സ്കൂൾ സോഫ്റ്റ്‌വെയറിലെ സ്റ്റുഡൻറ് സ്ട്രക്ച്ചറിലേക്കു 'age' എന്നൊരു പുതിയ ഇനം കൂട്ടിച്ചേർക്കണം എന്നിരിക്കട്ടെ. സോഫ്റ്റ്‌വെയറിന്റെ സുഗമമായ പ്രവർത്തനത്തിന് സ്റ്റുഡൻറ് ഡാറ്റ ഉപയോഗിക്കുന്ന എല്ലാ ഫങ്ഷനുകളെയും കണ്ടെത്തി അവയിൽ ആവശ്യമായ മാറ്റം വരുത്തൽ അനിവാര്യമാണ്.

c. പുതിയ ഡാറ്റ ഇനം നിർമിക്കുക പ്രയാസകരമാണ്

ഇൻറ്റിജർ, ഫ്ളോട്ട്, ക്യാരക്ടർ മുതലായ ഡാറ്റ ഇനങ്ങൾ ആണ് കമ്പ്യൂട്ടർ ഭാഷകളിൽ സാധാരണയായി ഉപയോഗിക്കാറുള്ളത്. ചില പ്രോഗ്രാമിങ് ഭാഷകൾ അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളെ കൂടാതെ മറ്റു ഡാറ്റ ഇനങ്ങൾ നിർമിക്കാൻ അനുവാദം നൽകിയിരിക്കുന്നു. അതിനർഥം അവ വിപുലീകരിക്കാവുന്നവയാണ് എന്നാണ്. അടിസ്ഥാന രൂപകൽപ്പനകളിൽ മാറ്റം വരുത്തുകയോ അല്ലെങ്കിൽ പ്രോഗ്രാം കോഡിൽ കാര്യമായ മാറ്റം വരുത്താതെ പ്രോഗ്രാമിന്റെ കഴിവിലും കാര്യക്ഷമതയിലും ഗണ്യമായ വർധന വരുത്തുകയോ ചെയ്യാനുള്ള കഴിവിനെയാണ് വിപുലീകരണ സാധ്യത എന്ന് പറയുന്നത്. ഈ സവിശേഷത പ്രോഗ്രാമുകളുടെ സങ്കീർണത കുറയ്ക്കാൻ സഹായിക്കുന്നതിനോടൊപ്പം അതിന്റെ കഴിവുകൾ വർധിപ്പിക്കുകയും ചെയ്യുന്നു. പ്രോസിജറൽ പ്രോഗ്രാമുകൾ വിപുലീകരണ സാധ്യതയില്ലാത്തവയാണ്.

d. യഥാർഥ ജീവിത മാതൃകയ്ക്കു യോജിച്ചതല്ല

പ്രോസിജറൽ പ്രോഗ്രാമിങ് മാതൃകയിൽ ഉപയോഗിക്കുന്ന ഫങ്ഷനുകളെയും പ്രോഗ്രാമുകളെയും ഒറ്റ ഘടകമായി പരിഗണിക്കുകയില്ല. മറിച്ചു അവ ഓരോന്നും സ്വതന്ത്രമായി നിലകൊള്ളുന്നു. അതിനാൽ ഈ മാതൃകയിൽ ഉപയോഗിക്കുന്ന ഡാറ്റയും ഫങ്ഷനുകളും ഒരു യഥാർഥ ജീവിത മാതൃകയായി പറയാനാവില്ല.

ഉദാഹരണത്തിന് നമ്മുടെ സ്കൂൾ സോഫ്റ്റ്‌വെയറിൽ നമ്മൾ വിദ്യാർഥികളുടെ ഡാറ്റയും അവയെ കൈകാര്യം ചെയ്യുന്ന ഫങ്ഷനുകളും ഉപയോഗിക്കുന്നു. അതുപോലെ അധ്യാപകരുടെ ഡാറ്റയും അവ കൈകാര്യം ചെയ്യുന്ന ഫങ്ഷനുകളും ഉണ്ടായിരിക്കും. പ്രോസിജറൽ മാതൃകയാണെങ്കിൽ വിദ്യാർഥികളുടെ ഡാറ്റയും അവയുടെ ഫങ്ഷനുകളും ഒറ്റ ഘടകമാക്കാനും അതുപോലെ അധ്യാപകരുടെ ഡാറ്റയും ഫങ്ഷനുകളും മറ്റൊരു ഘടകമാക്കി മാറ്റാനും സാധിക്കില്ല.

നിത്യജീവിതത്തിലെ ഓരോ വസ്തുവും പരിശോധിച്ചാൽ, അവയെല്ലാം അവയുടെ സവിശേഷതകളും സ്വഭാവഗുണങ്ങളും ഒത്തു ചേർന്ന ഒറ്റ ഘടകമാണെന്ന് നമുക്ക് കാണാം. ഉദാഹരണത്തിന് മനുഷ്യനെ പരിഗണിച്ചാൽ, പേര്, പൗരത്വം, ലിംഗം മുതലായവ സവിശേഷതകളും സംസാരം, ചിരി മുതലായവ സ്വഭാവ ഗുണങ്ങളും ആവാം. പ്രോസിജറൽ മാതൃകയിൽ, സ്വഭാവ ഗുണങ്ങളെ ഫങ്ഷനുകളായും സവിശേഷതകളെ ഡാറ്റയായും സൂചിപ്പിക്കാമെങ്കിലും അവയെ ഒറ്റ ഘടകമാക്കി മാറാൻ സാധിക്കാത്തതിനാൽ, ഈ മാതൃക യഥാർഥ ജീവിത മാതൃകയുമായി സാമ്യമില്ലാത്തതാണ്.

അടുത്തതായി ഒബ്ജക്റ്റ് ഓറിയന്റഡ് മാതൃകയെപ്പറ്റി ചർച്ച ചെയ്യുകയും അത് പ്രോസിജറൽ മാതൃകയുടെ പരിമിതികൾ എങ്ങനെ മറികടക്കുന്നു എന്ന് കാണുകയും ചെയ്യാം.

നിങ്ങളുടെ പുരോഗതി അറിയുക

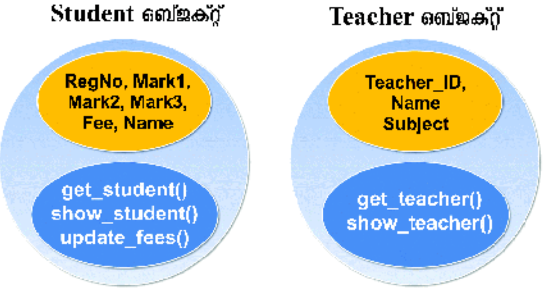
1. താഴെ പറയുന്ന മൂന്നു പ്രസ്താവനകൾ ശരിയോ തെറ്റോ എന്ന് പറയുക:
   a. ഗ്ലോബൽ വേരിയബിളുകൾ ഒന്നിലധികം ഫങ്ഷനുകളിൽ ഉപയോഗിക്കാൻ സാധിക്കില്ല.
   b. യഥാർഥ ലോകവുമായി വളരെ സാമ്യമുള്ളതാണ് പ്രോസിജറൽ മാതൃക.
   c. പ്രോസിജറൽ മാതൃകയിൽ ഡാറ്റയും ഫങ്ഷനുകളും ഒറ്റ ഘടകമായി കൂട്ടിച്ചേർക്കാൻ സാധിക്കില്ല.
2. താഴെ പറയുന്നവയിൽ നിന്ന് പ്രോസിജറൽ ഭാഷകൾ തിരഞ്ഞെടുക്കുക.

   C, C++, ഫോർട്രാൻ, ജാവ, പാസ്കൽ

2.1.2 ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ് മാതൃക (Object-Oriented Programming (OOP) paradigm)

ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ് മാതൃക, പ്രോസിജറൽ മാതൃകയിലെ പരിമിതികൾ പരിഹരിച്ചു കൊണ്ട് ഡാറ്റയെയും അവയുടെ മേൽ പ്രവർത്തിക്കുന്ന ഫങ്ഷനുകളെയും കൂട്ടിച്ചേർത്ത് ഒറ്റ ഘടകമാക്കി മാറ്റുന്നു. ഈ ഘടകത്തെ ഒബ്ജക്റ്റ് (object) എന്ന് വിളിക്കുന്നു.

ഉദാഹരണത്തിന് സ്കൂൾ സോഫ്റ്റ്‌വെയറിൽ OOP നടപ്പിലാക്കിയാൽ വിദ്യാർഥികളുടെ ഡാറ്റയും അവ ഉപയോഗപ്പെടുത്തുന്ന ഫങ്ഷനുകളും കൂട്ടിയോജിപ്പിച്ചു സ്റ്റുഡൻറ് (Student) എന്ന ഒബ്ജക്റ്റ് നമുക്ക് നിർമിക്കാം. അതുപോലെ അധ്യാപകരുടെ ഡാറ്റയും ഫങ്ഷനുകളും കൂട്ടിച്ചേർത്തു ടീച്ചർ (Teacher) എന്നൊരു ഒബ്ജക്റ്റും നിർമിക്കാം. ഒരു ഒബ്ജക്ടിലുള്ള ഫങ്ഷനുകൾക്ക് അനുവാദം കൂടാതെ മറ്റൊരു ഒബ്ജക്ടിലെ ഡാറ്റയെ ഉപയോഗിക്കുവാൻ സാധിക്കുകയില്ല.



ചിത്രം 2.3 ഡാറ്റകളും ഫങ്ഷനുകളും അടങ്ങുന്ന ഒബ്ജക്റ്റുകൾ

OOP ഉപയോഗിക്കുന്നത് കൊണ്ടുള്ള മേന്മകൾ :

a. സ്പഷ്ടമായ ഘടകങ്ങൾ അടങ്ങിയ ഒരു പ്രോഗ്രാമിങ് ഘടന OOP പ്രദാനം ചെയ്യുന്നു.

b. അബ്സ്ട്രാക്ട് ഡാറ്റ ഇനങ്ങൾ നിർവചിക്കാൻ ഇത് വളരെ നല്ലതാണ്.

c. എങ്ങനെയാണ് പ്രയോഗത്തിൽ വരുത്തിയിരിക്കുന്നത് എന്നുള്ള വിവരങ്ങൾ മറ്റു ഘടകങ്ങളിൽനിന്നും മറച്ചു വയ്ക്കാനും വ്യക്തമായ ഒരു സമ്പർക്കമുഖം പ്രദാനം ചെയ്യാനും ഇത് സഹായിക്കുന്നു.

d. നിലവിലുള്ളവയെ ബാധിക്കാത്ത വിധത്തിൽ പുതിയ ഒബ്ജക്റ്റുകൾ നിർമിക്കാവുന്നതിനാൽ നിലവിലുള്ള കോഡിനെ മാറ്റാനും പരിപാലിക്കാനും മറ്റും എളുപ്പമാണ്.

e. യഥാർഥ ജീവിത സാഹചര്യങ്ങളെ പ്രയോഗത്തിൽ വരുത്താൻ സഹായിക്കുന്നു.

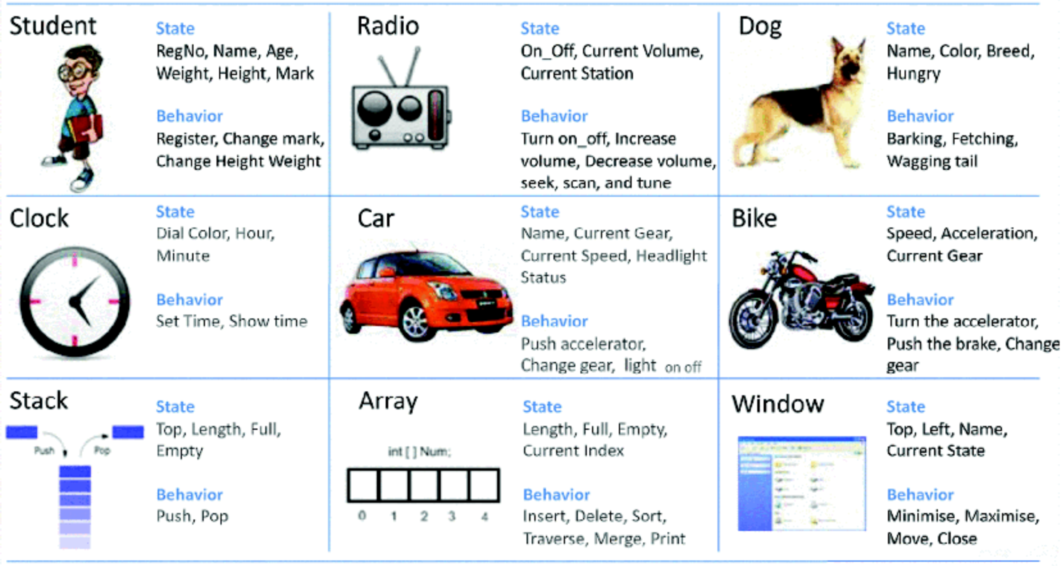
f. ഓപ്പറേറ്ററുകൾക്കു പുതിയ ഡാറ്റ ഇനങ്ങളും പുതിയ പ്രവർത്തനങ്ങളും നിർവചിക്കാൻ സഹായിക്കുന്നു.

2.2 OOP ന്റെ അടിസ്ഥാന തത്വങ്ങൾ (Basic concepts of OOP)

ഒബ്ജക്റ്റുകൾ, ക്ലാസ്സുകൾ, ഡാറ്റ അബ്സ്ട്രാക്ഷൻ (Data Abstraction), ഡാറ്റ എൻക്യാപ്സുലേഷൻ (Data Encapsulation), മോഡുലാരിറ്റി (Modulartiy), ഇൻഹെറിറ്റൻസ് (Inheritance), പോളിമോർഫിസം (Polymorphism) മുതലായ ആശയങ്ങൾ നൽകുന്നതിലൂടെ ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ് സോഫ്റ്റ്‌വെയർ വികസനവും പരിപാലനവും ലളിതമാക്കുന്നു. ഈ ആശയങ്ങൾ നമുക്ക് വിശദമായി ചർച്ച ചെയ്യാം.

2.2.1 ഒബ്ജക്റ്റുകൾ (Objects)

നമുക്ക് ചുറ്റുമുള്ള ഏതൊരു വസ്തുവിനെയും ഒബ്ജക്റ്റായി പരിഗണിക്കാം. എല്ലാ ഒബ്ജക്റ്റുകൾക്കും ഗുണവിശേഷങ്ങളും (ഡാറ്റ/അംഗം/അവസ്ഥ) പ്രവർത്തനരീതികളും (ഫങ്ഷനുകൾ/മെത്തേഡുകൾ) ഉണ്ടായിരിക്കും. ചിത്രം 2.4-ൽ ഒബ്ജക്റ്റുകളും അവയുടെ ഗുണവിശേഷങ്ങളും രീതികളും ഉദാഹരണസഹിതം പട്ടികപ്പെടുത്തിയിരിക്കുന്നു.



ചിത്രം 2.4: നിത്യ ജീവിതത്തിലെ ഒബ്ജക്റ്റുകൾ അവയുടെ സവിശേഷതകളോടും (അവസ്ഥ) രീതികളും (സ്വഭാവ ഗുണങ്ങളോടും) കൂടി

നമുക്കു ചെയ്യാം

നിങ്ങൾക്ക് ചുറ്റുമുള്ള വസ്തുക്കൾ നിരീക്ഷിക്കുകയും അവ ഓരോന്നിന്റെയും സവിശേഷതകളും സ്വഭാവഗുണങ്ങളും തിരിച്ചറിയുകയും ചെയ്യുക. നിങ്ങളുടെ കണ്ടെത്തലുകൾ താഴെ പറയുന്ന പട്ടികയിൽ എഴുതുക:

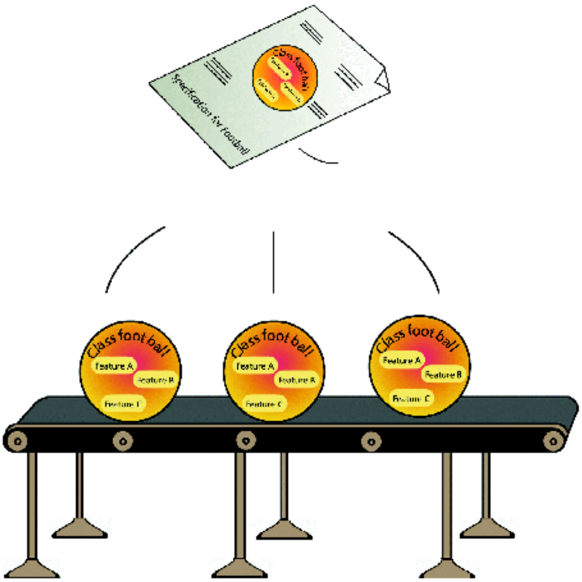
വസ്തുവിന്റെ പേര്	സവിശേഷതകൾ	സ്വഭാവ ഗുണങ്ങൾ

പ്രോഗ്രാമിങ് ഉപയോഗിച്ച് പ്രശ്ന പരിഹാരത്തിനായി OOP പ്രയോജനപ്പെടുത്തുകയാണെങ്കിൽ, പ്രശ്നത്തിനെ ഫങ്ഷനുകളായി വിഭജിക്കുന്നതിനു പകരം ഒബ്ജക്റ്റുകളായി വിഭജിക്കുന്നതായി പരിഗണിക്കുക. ഫങ്ഷനുകളായി പരിഗണിക്കാതെ ഒബ്ജക്റ്റുകളായി പരിഗണിക്കുമ്പോൾ പ്രോഗ്രാമിന്റെ രൂപകൽപ്പന ലളിതമായി മാറുന്നു. എന്തുകൊണ്ടെന്നാൽ പ്രോഗ്രാമുകളിലെ ഒബ്ജക്റ്റുകളും യഥാർഥ ജീവിതത്തിലെ ഒബ്ജക്റ്റുകളും തമ്മിൽ വളരെ സാമ്യമുണ്ട്.

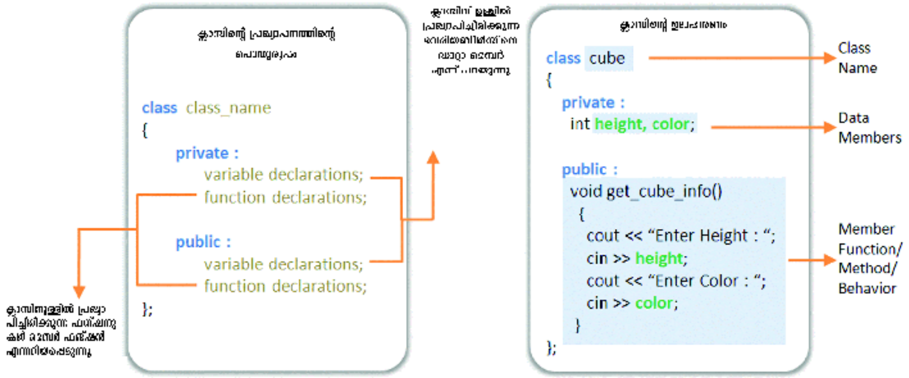
ഡാറ്റയും അവയുടെ മേൽ പ്രവർത്തിക്കുന്ന ഫങ്ഷനുകളും സംയോജിപ്പിച്ച് ഒറ്റ ഘടകമാക്കിയാണ് OOP ൽ ഒബ്ജക്റ്റ് ലഭ്യമാക്കുന്നത്. സംയോജനത്തിനു ശേഷം ഒബ്ജക്റ്റിലുള്ള ഫങ്ഷനുകളെ മെമ്പർ ഫങ്ഷൻ എന്നും ഡാറ്റയെ മെമ്പർ (മെമ്പർ ഡാറ്റ) എന്നും വിളിക്കുന്നു.

2.2.2 ക്ലാസുകൾ (Classes)

ഒരു ഒബ്ജക്റ്റിനെപ്പറ്റിയുള്ള എല്ലാം നിർണയിക്കുന്ന ക്ലാസ് ഉപയോഗിച്ചാണ് അതിനെ നിർവചിക്കുന്നത്. ഒരു പ്രത്യേക തരത്തിൽപ്പെട്ട എല്ലാ വസ്തുക്കൾക്കുമുള്ള പൊതുവായ ഗുണങ്ങളെ നിർവചിക്കുന്ന മാതൃകയാണ് ക്ലാസ്. ഈ നിർദേശങ്ങളിൽ ഡാറ്റയെ കുറിച്ചും അവയുടെ മേൽ പ്രവർത്തിക്കുന്ന ഫങ്ഷനുകളെ കുറിച്ചും വിവരങ്ങൾ അടങ്ങിയിരിക്കും. ഒരു ക്ലാസ്സിന്റെ ഒബ്ജക്റ്റിനെ ആ ക്ലാസ്സിന്റെ ഇൻസ്റ്റൻസ് എന്ന് വിളിക്കുന്നു. മാത്രമല്ല ഒരു ക്ലാസ്സിൽ നിന്നും എത്ര ഒബ്ജക്റ്റ് വേണമെങ്കിലും നിർമിക്കാം. (ചിത്രം 2.5 കാണുക).

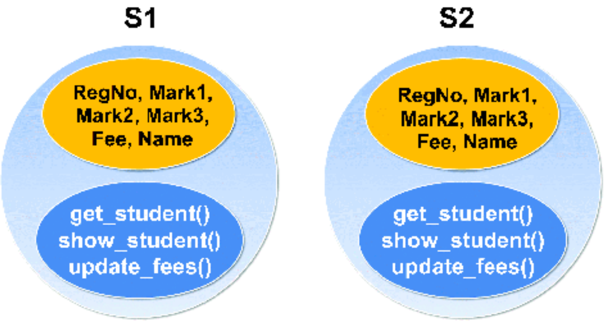


ചിത്രം 2.5: ക്ലാസുകളും അതിന്റെ ഒബ്ജക്റ്റുകളും



ചിത്രം 2.6: ഒരു ക്ലാസ്സിനെ ഡിക്ലയർ ചെയ്യുവാനുള്ള പൊതുവായ രീതി ഉദാഹരണസഹിതം

ഒരു ക്ലാസ്സിന്റെ നിർവചനവും ഉപയോഗവും സ്ട്രക്ചറിന് സമാനമാണ്. ഒരു സ്ട്രക്ച്ചറിൽ ഡാറ്റയെ കുറിച്ചുള്ള വിവരങ്ങളാണുള്ളതെങ്കിൽ ക്ലാസ്സിൽ ഡാറ്റയെയും അവയുടെ മേൽ പ്രവർത്തിക്കുന്ന ഫങ്ഷനുകളെയും പറ്റിയുള്ള വിവരങ്ങളാണ് അടങ്ങിയിരിക്കുന്നത് (ചിത്രം 2.6 കാണുക). 'struct' എന്ന കീവേർഡ് ഉപയോഗിച്ചാണ് സ്ട്രക്ചർ നിർമിക്കുന്നതെങ്കിൽ 'class' എന്ന കീവേർഡ് ഉപയോഗിച്ചാണ് ക്ലാസ് നിർമിക്കുന്നത്.



ചിത്രം 2.7: സ്റ്റുഡന്റ് ക്ലാസ്സിന്റെ ഒബ്ജക്റ്റുകൾ

സ്ട്രക്ചർ ഒരു വാല്യൂ ടൈപ്പും ക്ലാസ് ഒരു റഫറൻസ് ടൈപ്പുമാണ് അതായത് സ്ട്രക്ചർ വേരിയബിളുകൾ നിർമിക്കുമ്പോൾ അവ സ്ട്രക്ചറിലെ യഥാർഥ ഡാറ്റയെ സൂചിപ്പിക്കുന്നു. എന്നാൽ ക്ലാസ് ടൈപ്പ് ഒബ്ജക്ടുകൾ നിർമിക്കുമ്പോൾ അവ ഒബ്ജക്ടിന്റെ മെമ്മറിയെ പ്രതിനിധീകരിക്കുകയാണ് (റഫർ ചെയ്യുകയാണ്) ഉണ്ടാവുക.

`Student` എന്ന ക്ലാസ് ഉപയോഗിച്ച് 'S1', 'S2' എന്നീ പേരുകളുള്ള രണ്ടു ഒബ്ജക്റ്റുകൾ നിർമിക്കാനുള്ള പ്രസ്താവന താഴെ കൊടുത്തിരിക്കുന്നു. (ചിത്രം 2.7 ൽ ഉള്ളതുപോലെ).

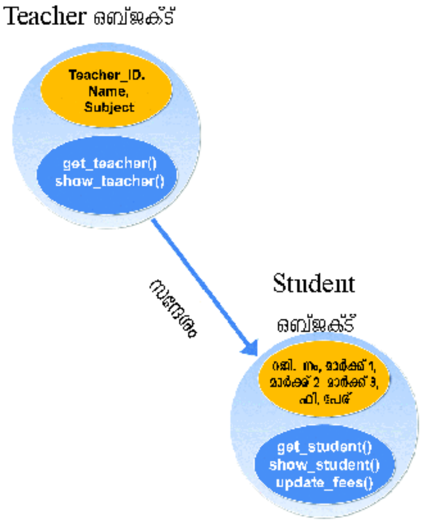
```
Student S1,S2;
```

താഴെ പറയുന്ന പ്രസ്താവന ഉപയോഗിച്ച് ഒരു ക്ലാസ്സിൽനിന്നും ഒബ്ജക്റ്റുകളെ നിർമിക്കാം.

Cube S; അല്ലെങ്കിൽ **Cube *C; C = new Cube;**

ഇവിടെ `Cube` എന്നത് ക്ലാസിന്റെ പേരും, `C` എന്നത് ഒബ്ജക്റ്റിന്റെ പേരും ആണ്.

രണ്ടു ഒബ്ജക്റ്റുകൾ തമ്മിൽ ആശയവിനിമയം നടത്തുന്നത് സന്ദേശം (മെസ്സേജ്) കൈമാറിക്കൊണ്ടാണ്. നിത്യ ജീവിതത്തിൽ ആളുകൾ പരസ്പരം സന്ദേശങ്ങൾ കൈമാറുന്നതിന് സമാനമാണിത്. യഥാർഥ ജീവിതത്തെ അനുകരിക്കുന്ന തരത്തിലുള്ള സംവിധാനം നിർമിക്കുവാൻ ഇതു സഹായിക്കുന്നു. ഒരു ഒബ്ജക്റ്റിന്റെ മെമ്പർ ഫങ്ഷനെ മറ്റൊരു ഒബ്ജക്റ്റ് വിളിക്കുന്നതിനെ മെസ്സേജ് പാസിംഗ് എന്ന് പറയുന്നു. മെസ്സേജ് പാസിംഗ് പ്രക്രിയയിൽ ഒബ്ജക്ടിന്റെ പേര്, മെമ്പർ ഫങ്ഷന്റെ പേര്, അയയ്ക്കുവാനുള്ള വിവരം എന്നിവ സൂചിപ്പിക്കുന്നു.



ചിത്രം 2.8: ഒബ്ജക്റ്റുകൾ സന്ദേശം കൈമാറുന്നു

ഉദാഹരണത്തിന് നമ്മുടെ സ്കൂൾ സോഫ്റ്റ് വെയറിൽ `Teacher` ഒബ്ജക്റ്റിന് ഒരു വിദ്യാർഥിയുടെ ഫീസ് പരിഷ്കരിക്കണമെങ്കിൽ `Student` ഒബ്ജക്ടിലേക്കു ഡാറ്റ ആവശ്യപ്പെട്ടു കൊണ്ട് ഒരു സന്ദേശം അയക്കുവാൻ `update_fee()` എന്ന മെമ്പർ ഫങ്ഷനെ വിളിക്കുന്നു (ചിത്രം 2.8 കാണുക).

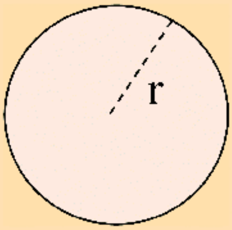
```
S1.update_fees("Rahul", 1000);
```

ഇവിടെ `S1` എന്നത് സ്റ്റുഡന്റിന്റെ ഒബ്ജക്റ്റ് ആണ്. സന്ദേശം ലഭിക്കുന്ന മുറയ്ക്ക് `Student` ഒബ്ജെക്റ്റിനു `Teacher` ഒബ്ജക്റ്റ് നൽകുന്ന ഡാറ്റ ഉപയോഗിച്ച് `Student` ന്റെ ഫീസ് പരിഷ്കരിക്കുന്നു.

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. OOP എന്നാൽ __________ആണ്.
2. OOP ൽ ഒരു ഒബ്ജെക്റ്റിനുള്ള ബ്ലൂ പ്രിന്റിനെ __________വിളിക്കുന്നു.
3. ക്ലാസ്സുമായി ബന്ധപ്പെട്ട ഫങ്ഷനുകളെ __________ എന്ന് വിളിക്കുന്നു.
4. ക്ലാസ്സിന്റെ അകത്തു നിർവചിക്കുന്ന വേരിയബിളുകളെ __________ എന്ന് പറയുന്നു.
5. ക്ലാസ്സും സ്ട്രക്ചറും തമ്മിലുള്ള വ്യത്യാസം എന്ത്?

താഴെ പറയുന്ന പ്രോഗ്രാം വൃത്തത്തെ ഒബ്ജക്റ്റായി ഉപയോഗിക്കുന്നു 'Circle' എന്ന ക്ലാസ്സ് ആവശ്യമായ ഏക ഡാറ്റയായ radius പ്രസ്താവിക്കുകയും radius നെ സ്വീകരിക്കാനും വിസ്തീർണം പ്രദർശിപ്പിക്കാനുമുള്ള മെമ്പർ ഫംഗ്ഷനുകളെ പ്രസ്താവിക്കുകയും ചെയ്യുന്നു.




```
using namespace std;
#include<ciostream>
class Circle
 {
   private:
      float r;
   public:
   void get_radius()
   {
     cout << "Enter Radius :";
     cin >> r;
   }
   void display_area()
   {
    cout<< "Area:"<< 3.14*r*r;
   }
   };
   int main()
   {
   Circle C1;
   C1.get_radius();  //സന്ദേശം അയയ്ക്കുന്നു
   C1.display_area();  //സന്ദേശം അയയ്ക്കുന്നു
   }
```

Member (മെമ്പർ) — float r;

Member functions (മെമ്പർ ഫങ്ഷൻ) — get_radius(), display_area()

Class Declaration (ക്ലാസ് നിർവചനം) — class Circle { ... };

Class (ക്ലാസ്) — Circle

Object (ഒബ്ജക്റ്റ്) — C1

മെയിൻ പ്രോഗ്രാം (Main program) — int main() { ... }

Output :

Enter Radius : 2.0

Area: 12.56

2.2.3 ഡാറ്റ അബ്സ്ട്രാക്ഷൻ (Data Abstraction)

ആവശ്യമായ വിവരങ്ങൾ മാത്രം പുറം ലോകത്തിനു വെളിപ്പെടുത്തുകയും ബാക്കി വിവരങ്ങൾ പിന്നണിയിൽ മറച്ചു വെക്കുകയും ചെയ്യുന്നതിനെയാണ് ഡാറ്റ അബ്സ്ട്രാക്ഷൻ എന്നത് കൊണ്ട് ഉദ്ദേശിക്കുന്നത്.

നിത്യ ജീവിതത്തിലെ ഒരു ഉദാഹരണമായ ടിവിയുടെ പ്രവർത്തന രീതി പരിഗണിക്കുക. സ്വിച്ച് ഓൺ ചെയ്യുക പവർ ഓഫ് ചെയ്യുക, ചാനൽ മാറ്റുക, ശബ്ദം നിയന്ത്രിക്കുക, VCR പ്ലേയർ, DVD മുതലായ ഉപകരണങ്ങൾ കൂട്ടിച്ചേർക്കുക തുടങ്ങിയ കാര്യങ്ങൾ നമുക്ക് ചെയ്യാവുന്നതാണ്. എന്നിരുന്നാലും നമുക്കതിന്റെ ആന്തരിക പ്രവർത്തനങ്ങളെക്കുറിച്ച് അറിവില്ല. ടിവിയിലേക്ക് എങ്ങനെ സിഗ്നലുകൾ ലഭിക്കുന്നു, അതെങ്ങനെ പരിവർത്തനം ചെയ്യപ്പെടുന്നു, എങ്ങനെയത് സ്ക്രീനിൽ പ്രദർശിപ്പിക്കുന്നു എന്നിവയെക്കുറിച്ചു നമുക്കു അറിയില്ല.

അപ്പോൾ ടെലിവിഷൻ അതിന്റെ ആന്തരികമായ പ്രവർത്തനതലത്തെയും ബാഹ്യമായ സമ്പർക്കമുഖത്തെയും വേർതിരിക്കുന്നതായി ഇതിൽനിന്നും നമുക്ക് മനസ്സിലാക്കാം. ആന്തരിക പ്രവർത്തനത്തെക്കുറിച്ചുള്ള പരിജ്ഞാനം കൂടാതെതന്നെ പവർ ബട്ടൺ, ചാനൽ മാറ്റുവാനുള്ള ബട്ടൺ, ശബ്ദം നിയന്ത്രിക്കാനുള്ള ബട്ടൺ മുതലായ നമുക്ക് ഉപയോഗിക്കാവുന്നതാണ്.

ഇതു പോലെ C++ ക്ലാസുകളും ഡാറ്റ അബ്സ്ട്രാക്ഷൻ പ്രദാനം ചെയ്യുന്നു. ഒരു ഒബ്ജക്റ്റിന്റെ പ്രവർത്തനത്തെ ഉപയോഗപ്പെടുത്തുവാനും അതിലെ ഡാറ്റ ലഭ്യമാക്കാനും പുറം ലോകത്തിനു ഉപയോഗിക്കുവാനും സാധിക്കുന്ന പൊതുവായ മെത്തേഡുകൾ അവ പ്രദാനം ചെയ്യുന്നു. ക്ലാസിന്റെ ആന്തരിക ഘടനയെ കുറിച്ച് അറിയാതെ തന്നെ ഈ മെത്തേഡുകൾ ഉപയോഗപ്പെടുത്തി ഒബ്ജക്റ്റുകളെ കൈകാര്യം ചെയ്യാവുന്നതാണ്.

ഉദാഹരണമായി, വിദ്യാർഥികളുടെ വിശദാംശങ്ങൾ പ്രദർശിപ്പിക്കുന്നതിനായി `Student` ഒബ്ജക്റ്റിനോടുകൂടി `show_student()` എന്ന മെമ്പർ ഫങ്ഷനെ വിളിക്കാവുന്നതാണ്. ഏതു സമയത്തു വേണമെങ്കിലും നമുക്ക് `show_student()`ന്റെ പ്രവർത്തന രീതി മാറ്റാവുന്നതാണ്. എന്നിരുന്നാലും ഫങ്ഷന്റെ പേരിന് മാറ്റം വരാത്തിടത്തോളം കാലം ഫങ്ഷനെ വിളിക്കുന്നതിന് ഒരു മാറ്റവും വരുന്നില്ല.

ഡാറ്റ അബ്സ്ട്രാക്ഷന്റെ രണ്ടു പ്രധാനപ്പെട്ട പ്രയോജനങ്ങൾ:

- ഉപയോക്തൃ തലത്തിൽ അപ്രതീക്ഷിതമായി വരാവുന്ന തെറ്റുകളിൽനിന്നും ക്ലാസിന്റെ ആന്തരിക ഘടനയെ സംരക്ഷിക്കുന്നു. ഈ തെറ്റുകൾ ഒബ്ജക്റ്റിന്റെ അവസ്ഥയെ പ്രതികൂലമായി ബാധിക്കുന്നത് ഒഴിവാക്കുവാൻ സാധിക്കുന്നു.
- ആവശ്യകതയനുസരിച്ചു ഉപയോക്ത തലത്തിലുള്ള കോഡിന് മാറ്റം വരുത്താതെ തന്നെ ക്ലാസിന്റെ പ്രവർത്തനത്തിനു മാറ്റം വരുത്താവുന്നതാണ്. കാലാന്തരത്തിലെ ആവശ്യകത അനുസരിച്ച് ക്ലാസിന്റെ നടപ്പിലാക്കലിൽ മാറ്റം വരുത്തുന്നതിന്, ക്ലാസിനുള്ളിലെ പ്രവർത്തന നിർദേശങ്ങൾ മാറ്റം വരുത്തേണ്ടതായി വരുന്നില്ല.

2.2.4 ഡാറ്റ എൻക്യാപ്സുലേഷൻ (Data Encapsulation)

ഫങ്ഷനുകൾ, ഡാറ്റ എന്നീ രണ്ടു അടിസ്ഥാന ഘടകങ്ങൾ അടങ്ങിയതാണ് C ++ -ലെ എല്ലാ പ്രോഗ്രാമുകളും. ഡാറ്റയെയും അവയുടെ മേൽ പ്രവർത്തിക്കുന്ന ഫങ്ഷനുകളെയും പരസ്പരം ബന്ധിപ്പിക്കുകയും അവയെ ബാഹ്യമായ ഇടപെടലുകളിൽനിന്നും ദുരുപയോഗത്തിൽനിന്നും സംരക്ഷിക്കുകയും ചെയ്യുന്ന OOP തത്വമാണ് ഡാറ്റ എൻക്യാപ്സുലേഷൻ.

ക്ലാസ് എന്ന് പറയുന്ന ഉപയോക്തൃ നിർവചിതമായ ഡാറ്റ ടൈപ്പ് ഉപയോഗിച്ചാണ് C++ ഡാറ്റ എൻക്യാപ്സുലേഷൻ പ്രാവർത്തികമാക്കുന്നത്. `private`, `protected`, `public` എന്നീ മൂന്നുതരം അംഗങ്ങൾ ഒരു ക്ലാസിൽ ഉണ്ടാകാം (ചിത്രം 2.6 കാണുക). ക്ലാസിൽ നിർവചിച്ചിരിക്കുന്ന എല്ലാ അംഗങ്ങളും സ്വാഭാവികമായി `private` ആയിരിക്കും. `private` ആയി നിർവചിക്കപ്പെട്ട അംഗങ്ങൾ ക്ലാസിനു പുറത്തു ദൃശ്യമാവുകയില്ല. `protected` ആയി നിർവചിക്കപ്പെട്ട അംഗങ്ങൾ അടിസ്ഥാന ക്ലാസ്സിന് പുറമെ രൂപീകൃത ക്ലാസിലും ഉപയോഗിക്കാം (ഭാഗം 2.2.6 ൽ വിശദീകരിച്ചിരിക്കുന്നു). പക്ഷേ അവ ക്ലാസിനു പുറത്തു ദൃശ്യമാകുകയില്ല.

`Student` ക്ലാസിലെ അംഗങ്ങളായ `Regno`, `Name`, `Mark1`, `Mark2`, `Mark3`, `Fee` എന്നിവ `private` ആയിട്ടാണ് നിർവചിച്ചിരിക്കുന്നത്. അതായത് `Student` ക്ലാസിലെ മെമ്പർ ഫങ്ഷനുകൾക്കു മാത്രമേ അവയെ ഉപയോഗിക്കാൻ സാധിക്കൂ, പ്രോഗ്രാമിലെ മറ്റു ഭാഗങ്ങളിൽ ഇവ ലഭ്യമാകുകയില്ല. ഇവിടെ ഡാറ്റ മറയ്ക്കുക വഴി എൻക്യാപ്സുലേഷൻ സാധ്യമാക്കുകയും ചെയ്യുന്നു.

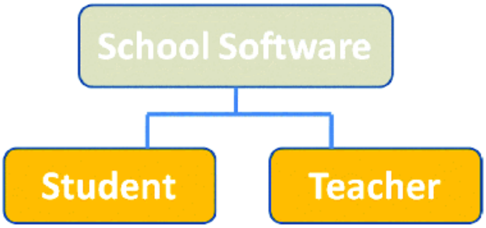
ക്ലാസിലെ അംഗങ്ങൾ പ്രോഗ്രാമിലെ മറ്റു ഭാഗങ്ങളിലേക്ക് ലഭ്യമാകണമെങ്കിൽ അവയെ `public` എന്ന കീവേർഡ് ഉപയോഗിച്ച് നിർവചിക്കണം. `public` എന്ന വാക്കിനു ശേഷം നിർവചിക്കപ്പെടുന്ന എല്ലാ ഡാറ്റയും ഫങ്ഷനുകളും പ്രോഗ്രാമിന്റെ മറ്റു ഭാഗങ്ങൾക്കു ലഭ്യമാകുന്നതാണ്. ഉദാഹരണത്തിന് `Student` എന്ന ക്ലാസിലെ അംഗങ്ങൾ പ്രോഗ്രാമിന്റെ മറ്റു ഭാഗങ്ങൾക്കു ലഭ്യമാകണമെങ്കിൽ `public` ആയിട്ട് അവയെ പ്രസ്താവിക്കണം.

2.2.5 മോഡുലാരിറ്റി (Modularity)

ഒരു പ്രശ്നപരിഹാരത്തിനായി കമ്പ്യൂട്ടർ പ്രോഗ്രാം എഴുതുമ്പോൾ അതിനെ ഉപപ്രോഗ്രാമുകളായി വിഭജിച്ച് അവ ഓരോന്നിനും പ്രതിവിധി (ഫലം) കണ്ടെത്തുന്നു. സമ്പർക്കമുഖവും (interface), പ്രവർത്തനതലവും (implementation), വിവരണങ്ങളും (specification) അടങ്ങിയ വ്യത്യസ്ത ഘടകങ്ങളായിരിക്കും ഓരോ ഉപപ്രോഗ്രാമുകളുടെയും പരിഹാരമായി ലഭിക്കുക.

മോഡുലാരിറ്റി എന്ന ആശയം ഉപയോഗിച്ചാണ് തന്നിരിക്കുന്ന പ്രശ്നത്തെ സ്വതന്ത്രമായ ഘടകങ്ങളായി വിഭജിക്കുന്നത്. ഇത്തരത്തിലുള്ള ഓരോ മൊഡ്യൂളും മറ്റു മൊഡ്യൂളുകളിൽ നിന്ന് സ്വതന്ത്രമായി എഴുതപ്പെടുന്നവയായിരിക്കും. പിന്നീട് ഈ മൊഡ്യൂളുകൾ കോർത്തിണക്കി പൂർണമായ ഒരു സോഫ്റ്റ്വെയർ നിർമിക്കുന്നു. ഇത്തരം മൊഡ്യൂളുകൾ ആശയവിനിമയം ചെയ്യുന്നത് പരസ്പരം സന്ദേശങ്ങൾ അയക്കുന്നതിലൂടെ (മെസ്സേജ് പാസിംഗ്) ആണ്.

ഫങ്ഷനുകൾ ഉപയോഗിച്ച് മോഡുലാരിറ്റി പ്രാവർത്തികമാക്കുന്നത് എങ്ങനെയാണെന്നു നമ്മൾ മുമ്പ് പഠിച്ചിട്ടുണ്ട് (പതിനൊന്നാം തരത്തിലെ പാഠപുസ്തത്തിൽ അധ്യായം 10 ഫങ്ഷനുകൾ കാണുക). മോഡുലാരിറ്റി പ്രവർത്തികമാക്കിയിരിക്കുന്നത് ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ്ങിൽ ക്ലാസിന്റെ സഹായത്തോടുകൂടിയാണ്. ഉദാഹരണത്തിന് നമ്മുടെ സ്കൂൾ സോഫ്റ്റ്വെയറിൽ ക്ലാസ് എന്ന ആശയത്തിന്റെ സഹായത്തോടെ വിദ്യാർഥികളുമായി ബന്ധപ്പെട്ടതും അധ്യാപകരുമായി ബന്ധപ്പെട്ടതുമായ കാര്യങ്ങൾ പ്രത്യേകം മൊഡ്യൂളുകളായി തരം തിരിച്ചിരിക്കുന്നു (ചിത്രം 2.9 കാണുക).

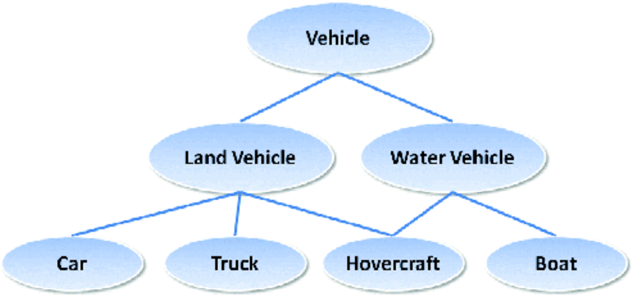


ചിത്രം 2.9: മോഡ്യൂലാരിറ്റി

2.2.6 ഇൻഹെറിറ്റൻസ് (Inheritance)

ഒരു ക്ലാസിലെ ഒബ്ജക്റ്റുകൾ മറ്റൊരു ക്ലാസിലെ സവിശേഷതകളും പ്രവർത്തനങ്ങളും ആർജിക്കുന്ന പ്രക്രിയയാണ് ഇൻഹെറിറ്റൻസ് എന്ന് പറയുന്നത്. ക്രമപ്രകാരമുള്ള തരം തിരിക്കൽ (Hierarchial Classification), പുനരുപയോഗം (Reusabiltiy) എന്നീ തത്വങ്ങളെ ഇൻഹെറിറ്റൻസ് പിന്തുടരുന്നു.

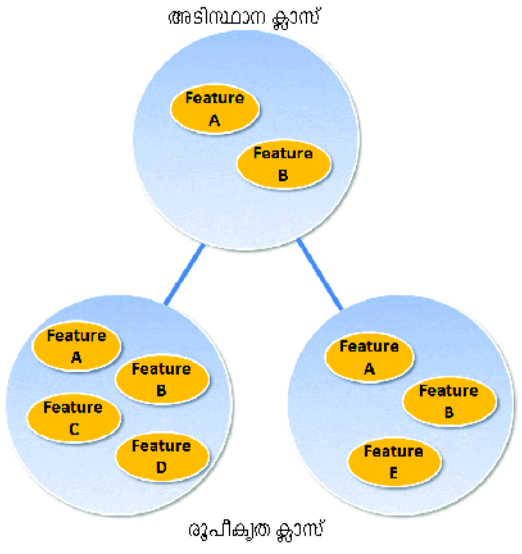
നിത്യജീവിതത്തിലെ ഒരു ഉദാഹരണത്തിലൂടെ നമുക്കീ സാഹചര്യം വിവരിക്കാം. ചിത്രം 2.10 ൽ ഭൂതല വാഹനത്തിനും (land vehicle) ജലവാഹനത്തിനും (water vehicle) വാഹനത്തിന്റെ (Vehicle) സവിശേഷതകൾ (അതായത് ഡാറ്റ അംഗങ്ങളും അംഗങ്ങളായ ഫങ്ഷനുകളും) കൈവരുന്നു. കാറിനും ട്രക്കിനും ഭൂതല വാഹനത്തിന്റെ സവിശേഷതകൾ ലഭ്യമാകുന്നു (അതായത് കാർ/ട്രക്ക് = വാഹനം + ഭൂതല വാഹനം). അത് പോലെ ബോട്ടിന് ജല വാഹനത്തിന്റെ സവിഷേതകൾ (അതായത് ബോട്ട്= വാഹനം+ജല വാഹനം) ലഭിക്കുന്നു. ജലത്തിലും ഭൂതലത്തിലും സഞ്ചരിക്കുന്ന ഒരു ഹോവർ ക്രാഫ്റ്റിനു (hovercraft) ഭൂതല വാഹനത്തിന്റെയും ജല വാഹനത്തിന്റെയും സവിശേഷതകൾ (അതായത് ഹോവർ ക്രാഫ്റ്റ്= വാഹനം+ഭൂതല വാഹനം+ജല വാഹനം) ലഭിക്കുന്നു. അതുപോലെ കാർ ഹാച്ച്ബാക്ക് (Hatchback), സെഡാൻ (Sedan), എസ് യു വി (SUV) മുതലായവയായി വീണ്ടും തരം തിരിക്കാവുന്നതാണ്. അവ ഓരോന്നും വാഹനം, ഭൂതല വാഹനം, കാർ എന്നിവയുടെ സവിശേഷതകൾ ആർജിക്കുന്നു, അതേ സമയം തന്നെ അവയുടേതായിട്ടുള്ള സവിശേഷതകൾ നിലനിർത്തപ്പെടുകയും ചെയ്യുന്നു. ഇത്തരത്തിലുള്ള ക്രമീകരണം ഏതു തലത്തിലേക്കും വ്യാപിപ്പിക്കാവുന്നതാണ്.



ചിത്രം 2.10: യഥാർത്ഥ ജീവിതത്തിലെ ഇൻഹെറിറ്റൻസ്

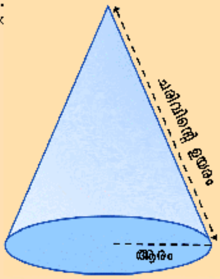
ഈ ഉദാഹരണത്തിൽ അഥവാ ഭൂതല വാഹനത്തിന്റെയും ജല വാഹനത്തിന്റെയും പൊതുവായ സവിശേഷതകളും പ്രവർത്തനങ്ങളും വേർപിരിച്ച് വാഹനം എന്ന ക്ലാസ്സിൽ തന്നെ വച്ചിരുന്നില്ലെങ്കിൽ ആ രണ്ടു ക്ലാസ്സുകളിലും അവ ആവർത്തിക്കേണ്ടി വന്നേനെ. ഇത് പ്രോഗ്രാമിങ്ങിന്റെ വലിപ്പവും കോഡ് ചെയ്യാനും തെറ്റ് കണ്ടെത്തി തിരുത്താനുമുള്ള സമയവും വർധിപ്പിക്കുന്നു. തന്നിരിക്കുന്ന ചാർട്ടിന്റെ തുടക്കം മുതൽ അവസാനം വരെ നോക്കിയാൽ പ്രോഗ്രാമിന്റെ സങ്കീർണത എത്രത്തോളം കുറയ്ക്കുവാൻ ഇൻഹെറിറ്റൻസ് കൊണ്ട് സാധിക്കുന്നു എന്ന് നമുക്ക് കാണാം.

ക്ലാസ് നിർമിക്കപ്പെടുകയും തെറ്റ് തിരുത്തപ്പെടുകയും ചെയ്തു കഴിഞ്ഞാൽ ആവശ്യമെങ്കിൽ മറ്റു പ്രോഗ്രാമുകൾക്ക് അതിനെ വിതരണം ചെയ്യാവുന്നതാണ്. ഇതിനെ പുനരുപയോഗം (റീയൂസബിലിറ്റി) എന്ന് വിളിക്കുന്നു. OOP ൽ ഇൻഹെറിറ്റൻസ് തത്വം പുനരുപയോഗം എന്ന ആശയത്തെ ഒന്നുകൂടി വിപുലീകരിക്കുന്നു. പുനരുപയോഗത്തിലൂടെ നിലവിലുള്ള ക്ലാസിൽ മാറ്റം വരുത്താതെ തന്നെ കൂടുതൽ വിശേഷഗുണങ്ങൾ നമുക്ക് കൂട്ടി ചേർക്കാവുന്നതാണ്. നിലവിലുള്ള ക്ലാസിൽനിന്നും പുതിയ ഒരു ക്ലാസിനെ ഉത്പാദിപ്പിച്ചിട്ടാണ് ഇത് സാധ്യമാകുന്നത്. പുതിയ ക്ലാസിന് അതിന്റേതായ സവിശേഷതകൾ കൈവരുന്നതിനോടൊപ്പം നിലവിലുള്ള ക്ലാസിന്റെ സവിശേഷതകൾ പൈതൃകമായി ആർജിക്കുകയും ചെയ്യുന്നു. നിലവിലുള്ള ക്ലാസിനെ അടിസ്ഥാന ക്ലാസ് എന്നും പുതിയ ക്ലാസിനെ രൂപീകൃത ക്ലാസ് എന്നും വിളിക്കുന്നു. രൂപീകൃത ക്ലാസിനു രണ്ടിന്റെയും സംയുക്തമായ സവിശേഷതകൾ ഉണ്ടായിരിക്കും. നിലവിലുള്ള ക്ലാസിൽനിന്നും എത്ര രൂപീകൃത ക്ലാസ് വേണമെങ്കിലും ഉൽപാദിപ്പിക്കാം. ചിത്രം 2.11 -ൽ പുതിയ ക്ലാസിന്റെ ഉൽപത്തി കാണിച്ചിരിക്കുന്നു.



ചിത്രം 2.11: OOP ലെ ഇൻഹെറിറ്റൻസ്

താഴെ പറയുന്ന പ്രോഗ്രാം ഇൻഹെറിറ്റൻസ് വഴി 'Cone' ക്ലാസ്സിലെ ഒബ്ജക്ടിൽ നിന്നും 'Circle' ക്ലാസിലെ ഒരു ഒബ്ജക്ടിനെ ഉൽപ്പാദിപ്പിക്കുന്നു. ആരം (r) നേരത്തെ തന്നെ 'Circle', ക്ലാസിൽ ഡിക്ലയർ ചെയ്തിരിക്കുന്നതിനാൽ രൂപീകൃത ക്ലാസായ 'Cone' ൽ സ്ലാൻറ് ഹൈറ്റ് (s). മാത്രം ഡിക്ലയർ ചെയ്താൽ മതിയാകും. ഡാറ്റ സ്വീകരിക്കാനും വിസ്തീർണം പ്രദർശിപ്പിക്കാനും രണ്ടു മെമ്പർ ഫങ്ഷനുകൾ ഉപയോഗിച്ചിരിക്കുന്നു. 'Circle' ലെ ഡാറ്റ അംഗങ്ങൾ 'protected' ആയി ഡിക്ലയർ ചെയ്തിരിക്കുന്നതിനാൽ, 'Cone' ക്ലാസിനു അവ ലഭ്യമാകു;




```
#include <iostream>
using namespace std;
class Circle
{ protected:
     float r;
  public:
     void get_radius(){
        cout << "Enter Radius :";
        cin >> r;
     }
     void display_area(){
       cout<< "Area:"<< 3.14*r*r;
     }
};
class Cone : public Circle
{ private:
     float s;
  public:
     void get_cone_data() {
        get_radius();
        cout << "Enter slant height:";
        cin >> s;
     }
     void display_cone_area(){
     cout << "Area :" << 3.14*r*(s+r);
     }
};
int main()     {
     Cone C1;
     C1.get_cone_data(); //Sending message
     C1.display_cone_area(); //Sending message
}
```

അടിസ്ഥാന ക്ലാസ് (Base Class)

പുതിയ മെമ്പർ (New member)

പുതിയ മെമ്പർ ഫങ്ഷനുകൾ (New member functions)

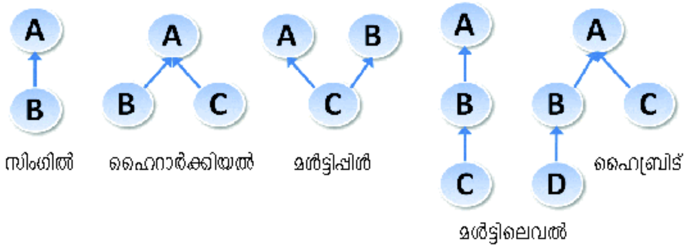
രൂപീകൃത ക്ലാസ് (Derived class)

മെയിൻ ഫങ്ഷൻ (Main function)

Output :

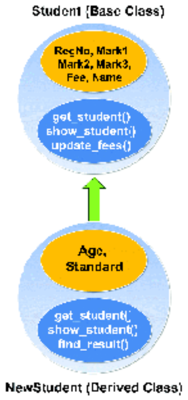
```
Enter Radius : 2.0
Enter slant height: 5.0
Area :43.96
```

സിംഗിൾ ഇൻഹെറിറ്റൻസ്, മൾട്ടിപ്പിൾ ഇൻഹെറിറ്റൻസ്, ഹൈറാർക്കിയൽ ഇൻ ഹെറിറ്റൻസ്, മൾട്ടി ലെവൽ ഇൻഹെറിറ്റൻസ്, ഹൈ ബ്രിഡ് ഇൻഹെറിറ്റൻസ് എന്നിവയാണ് വിവിധ തര ത്തിലുള്ള ഇൻഹെറിറ്റൻസു കൾ (ചിത്രം 2.12 കാണുക).



ചിത്രം 2.12: വിവിധ തരത്തിലുള്ള ഇൻഹെറിറ്റൻസ്

സ്കൂൾ സോഫ്റ്റ്‌വെയറിൽ ഇൻഹെറിറ്റൻസ് നടപ്പിലാക്കുന്നത് എങ്ങനെയാണെന്ന് നോക്കാം. നിലവിലുള്ള ഡാറ്റയും ഫങ്ഷനും കൂടാതെ Student ക്ലാസിലേക്ക് വയസ്സ്, പഠിക്കുന്ന ക്ലാസ് എന്നീ ഡാറ്റയും പരീക്ഷാഫലം കണ്ടു പിടിക്കാനുള്ള ഫങ്ഷനും കൂട്ടിച്ചേർക്കണം എന്ന് കരുതുക. Student ക്ലാസിനു മാറ്റം വരുത്താതെ NewStudent എന്ന ഒരു പുതിയ ക്ലാസിനെ നമ്മൾ Student ക്ലാസിൽ നിന്നും ഉൽപ്പാദിപ്പിക്കുന്നു. Student ക്ലാസ് അതേപടി തുടരുന്നു. ഇവിടെ Student ക്ലാസ് അടിസ്ഥാന ക്ലാസും, NewStudent ക്ലാസ് രൂപീകൃത ക്ലാസും ആണ് (ചിത്രം 2.13 കാണുക).



ചിത്രം 2.13: ഇൻഹെറിറ്റൻസിന്റെ ഉദാഹരണം

രൂപീകൃത ക്ലാസിനെ പ്രഖ്യാപിക്കാനുള്ള ഘടന താഴെ പറയുന്നത് പോലെയാണ്.

```
class derived_class: AccessSpecifier
base_class
{
 //മെമ്പറുകളുടെയും മെമ്പർ ഫങ്ഷനുകളുടെയും പ്രഖ്യാപനം
(declaration of members and memberfunctions)
 };
```

ഇവിടെ derived_class എന്നത് രൂപീകൃത ക്ലാസിനെയും base_class എന്നത് ഏതു ക്ലാസിൽ നിന്നാണോ രൂപീകൃത ക്ലാസിനെ ഉൽപാദിപ്പിക്കുന്നത് ആ ക്ലാസിനെയുമാണ് സൂചിപ്പിക്കുന്നത്. AccessSpecifier എന്നത് public, protected, private ഇവയിൽ ഏതെങ്കിലും ആവാം. ഇത് അടിസ്ഥാന ക്ലാസിൽ നിന്നും പൈതൃകമായി ലഭിച്ച അംഗങ്ങളുടെ ലഭ്യതയെ സൂചിപ്പിക്കുന്നു.

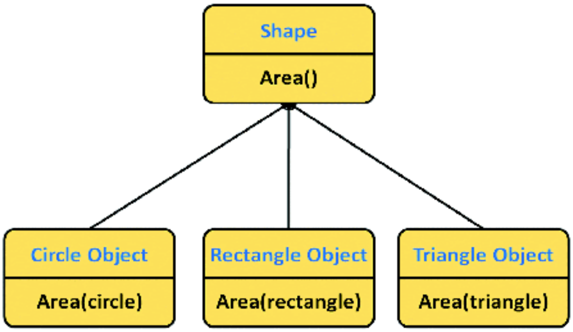
2.2.7 പോളിമോർഫിസം (Polymorphism)

'പോളി' എന്നാൽ ഒന്നിലധികം എന്നും 'മോർഫ്' എന്നാൽ ആകൃതികൾ എന്നുമാണ് അർഥം. വിവിധ രൂപത്തിൽ പ്രകടിപ്പിക്കാനുള്ള കഴിവിനെ പോളിമോർഫിസം എന്ന് നിർവചിക്കാം. ചിത്രം 2.14 -ൽ ഇത് വിശദീകരിച്ചിരിക്കുന്നു. ഇവിടെ 'Now Speak' എന്ന ഒരേ നിർദേശമാണ് വിവിധ ജീവികൾക്ക് നൽകുന്നത്, പക്ഷെ അവ ഓരോന്നും വ്യത്യസ്ത രീതികളിലാണ് ഈ നിർദേശത്തോട് പ്രതികരിക്കുന്നത്.

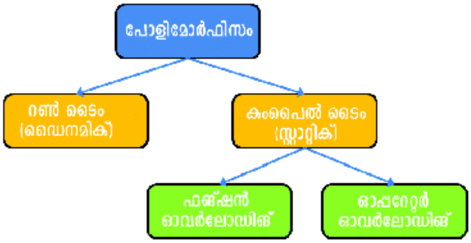
ഒബ്ജക്റ്റുകളുടെ ഡാറ്റ ഇനം അല്ലെങ്കിൽ ക്ലാസ് അനുസരിച്ച് അവയെ വ്യത്യസ്തമായി പ്രോസസ്സ് ചെയ്യാനുള്ള പ്രോഗ്രാമിങ് ഭാഷയുടെ കഴിവിനെയാണ് ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ്ങിൽ പോളിമോർഫിസം എന്ന് പറയുന്നത്. ഒന്നുകൂടി വ്യക്തമായി പറഞ്ഞാൽ രൂപീകൃത ക്ലാസിന്റെ രീതികളെ പുനർ നിർവചിക്കാനുള്ള കഴിവാണ് പോളിമോർഫിസം.

ചിത്രം 2.14: പോളിമോർഫിസത്തിന്റെ വിശദീകരണം

ഉദാഹരണത്തിന് `Shape` എന്ന ബേസ് ക്ലാസിൽ നിന്നും ഉല്പാദിപ്പിച്ചിരിക്കുന്ന `circle`, `rectangle`, `triangle` മുതലായ രൂപീകൃത ക്ലാസുകൾക്ക് `area` എന്ന പേരിൽ വിവിധ മെത്തേഡുകൾ നിർവചിക്കുവാനുള്ള സൗകര്യം പോളിമോർഫിസം പ്രോഗ്രാമർക്കു നൽകുന്നു (ചിത്രം 2.15 കാണുക). ഒബ്ജക്റ്റ് ഏതായാലും `area` എന്ന മെത്തേഡ് പ്രയോഗിക്കുമ്പോൾ കൃത്യമായ ഫലം ലഭിക്കുന്നു.



ചിത്രം 2.15: പോളിമോർഫിസത്തിന്റെ ഉദാഹരണം



ചിത്രം 2.16: പോളിമോർഫിസത്തിന്റെ വർഗീകരണം

പോളിമോർഫിസം രണ്ടു തരത്തിലുണ്ട് (ചിത്രം 2.16 കാണുക)

a. കംപൈയിൽ ടൈം പോളിമോർഫിസം (ഏർലി ബൈൻഡിങ്/സ്റ്റാറ്റിക്) (early binding/static)

b. റൺ ടൈം പോളിമോർഫിസം (ലേറ്റ് ബൈൻഡിങ്/ഡൈനാമിക്) (late binding/dynamic)

a. കംപൈയിൽ ടൈം പോളിമോർഫിസം/ഏർലി ബൈൻഡിങ് (Compile time polymorphism)

കംപൈൽ സമയത്തു തന്നെ ഫങ്ഷൻ വിളിക്കുന്നതിനെ ഫങ്ഷൻ നിർവചനവുമായി ബന്ധിപ്പിക്കാനുള്ള കംപൈലറിന്റെ കഴിവിനെയാണ് കംപൈയിൽ ടൈം പോളിമോർഫിസം സൂചിപ്പിക്കുന്നത്. ഇതിന്റെ കീഴിൽ വരുന്നതാണ് ഫങ്ഷൻ ഓവർലോഡിങ്, ഓപ്പറേറ്റർ ഓവർലോഡിങ് എന്നിവ.

ഫങ്ഷൻ ഓവർലോഡിങ് (Function Overloading): ഒരേ പേരും വ്യത്യസ്തങ്ങളായ സിഗ്നേച്ചറുകളും (ഫങ്ഷൻ ആർഗ്യുമെന്റുകളുടെ എണ്ണവും തരവും) ഉള്ള ഫങ്ഷനുകൾ വ്യത്യസ്തങ്ങളായി പ്രവർത്തിക്കുന്നു. ഉദാഹരണത്തിന് `area(int)` എന്നത് ഒരു സമചതുരത്തിന്റെ വിസ്തീർണം കണ്ടു പിടിക്കാൻ ഉപയോഗിക്കുന്നു. അതേസമയം `area(int, int)` എന്നത് ഒരു ചതുരത്തിന്റെ വിസ്തീർണം കണ്ടുപിടിക്കാൻ ഉപയോഗിക്കുന്നു. ഇപ്രകാരം `area()` എന്ന ഒരേ ഫങ്ഷൻ വ്യത്യസ്ത സിഗ്നേച്ചറുകൾക്കനുസരിച്ച് രണ്ടു രീതിയിൽ പ്രവർത്തിക്കുന്നു. ഒന്നിൽ കൂടുതൽ ഫങ്ഷനുകൾ ഒരേ പേരും വ്യത്യസ്ത സിഗ്നേച്ചറുകളോടും കൂടി നിർവചിക്കുന്നതിനെ ഫങ്ഷൻ ഓവർലോഡിങ് എന്ന് പറയുന്നു.

ഓപ്പറേറ്റർ ഓവർലോഡിങ് (Operator overloading): നിലവിലുള്ള C++ ഓപ്പറേറ്ററുകൾക്ക് (+, -, =, * മുതലായവ) പുതിയ നിർവചനം നൽകുന്ന ആശയമാണ് ഓപ്പറേറ്റർ ഓവർലോഡിങ്. ലഭ്യമാകുന്ന ഓപെറന്റുകൾക്ക് അനുസരിച്ച് ഒരു ക്ലാസിലെ വ്യത്യസ്ത ഒബ്ജക്റ്റുകളുടെ മേൽ വ്യത്യസ്തമായി പ്രവർത്തിക്കാൻ സാധാരണ ഓപ്പറേറ്ററുകളെ പ്രാപ്തരാക്കുന്ന പ്രക്രിയയാണിത്. ഒരു ഓപ്പറേറ്ററിനെ ഓവർലോഡ് ചെയ്യണമെങ്കിൽ ഓവർലോഡ് ചെയ്യുന്ന ഓപ്പറേറ്ററിനായി ഒരു മെമ്പർ ഫങ്ഷൻ നിർവചിക്കേണ്ടതാണ്.

ഉദാഹരണത്തിന് C++ ലെ + (പ്ലസ്) എന്ന ഓപ്പറേറ്റർ നിലവിൽ തന്നെ ഓവർലോഡ് ചെയ്യപ്പെട്ടതാണ്. ഇതിന് പൂർണ സംഖ്യകളെയും (`4 + 5`) അസറിര ദശാംശ സംഖ്യകളെയും കൂട്ടുവാൻ (`3.14 + 2.6`) സാധിക്കുന്നു. ആവശ്യമെങ്കിൽ രണ്ടു ഒബ്ജക്റ്റുകൾ തമ്മിലുള്ള സങ്കലനത്തിനും ഇതേ ഓപ്പറേറ്റർ ഉപയോഗിക്കാം (അതിനുള്ള കോഡ് ക്ലാസിൽ ചേർക്കണം). ഉദാഹരണത്തിന്, `T1 = T2 + T3`. ഇവിടെ `T1, T2, T3` എന്നത് '`time`' എന്ന ക്ലാസ്സിന്റെ ഒബ്ജക്റ്റുകളാണ് . '+' എന്ന ഓപ്പറേറ്റർ ഉപയോഗിച്ച് HH:MM:SS മാതൃകയിലുള്ള രണ്ടു സമയങ്ങളുടെ തുക കണ്ടുപിടിക്കാവുന്നതാണ്.

b. റൺ ടൈം പോളിമോർഫിസം/ലേറ്റ് ബൈൻഡിങ് (Run time polymorphism)

റൺ ടൈമിൽ ഫങ്ഷൻ ഉപയോഗിക്കുമ്പോൾ ഫങ്ഷൻ നിർവചനവുമായി ബന്ധിപ്പിക്കുന്നതിനെ റൺ ടൈം പോളിമോർഫിസം എന്ന് പറയുന്നു. ഇൻഹെറിറ്റൻസ്, പോയിന്ററുകൾ എന്നീ ആശയങ്ങളാണ് ഇതിനായി ഉപയോഗിക്കുന്നത്.

താഴെ പറയുന്ന പ്രോഗ്രാം ഫംഗ്ഷൻ ഓവർലോഡിങ് ഉപയോഗിച്ച് സമചതുരത്തിന്റെയും ദീർഘചതുരത്തിന്റെയും വിസ്തീർണം കണ്ടുപിടിക്കുന്നു. സമചതുരത്തിന്റെ വിസ്തീർണം കണ്ടെത്താൻ വേണ്ടിയും ദീർഘചതുരത്തിന്റെ വിസ്തീർണം കണ്ടെത്താൻ വേണ്ടിയും രണ്ടു ഫംഗ്ഷനുകൾ ഇത് നിർവചിക്കുന്നു.അവ രണ്ടിനും **'area'** എന്ന പൊതുവായ പേര് നൽകിയിട്ടുണ്ടെങ്കിലും അവ രണ്ടിലേക്കും നൽകുന്ന വിലകൾ വ്യത്യസ്തമാണ്.

ഫങ്ഷൻ പേരുകൾ ഒരുപോലെയാണ് (Function names are same)

```
#include<ciostream>
using namespace std;
int area(int s){ //സമചതുരത്തിന്റെ വിസ്തീർണം കണ്ടുപിടിക്കാൻ
return s * s;
}
int area(int s1, int s2){ //ചതുരത്തിന്റെ വിസ്തീർണം
return (s1 * s2);           //കണ്ടുപിടിക്കാൻ
}
int main()
{
cout << "Area of Square:" << area(5); << endl;
cout << "Area of Rectangle:"<< area (7,2);
}
```

സിഗ്നേച്ചറുകൾ വ്യത്യസ്തമാണ് (Signatures are different)

Output:

```
Area of Square: 25
Area of Rectangle: 14
```

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. ഡാറ്റയെയും ഫങ്ഷനുകളെയും ഒറ്റ ഘടകമാക്കി മാറ്റുന്നതിനെ ___________ എന്ന് വിളിക്കുന്നു.
2. ഡാറ്റയിലേക്കുള്ള അനുമതി നിയന്ത്രിക്കുന്ന സവിശേഷത ______ _____ എന്നറിയപ്പെടുന്നു.
3. ഒബ്ജക്റ്റുകൾ സാധാരണയായി ________ ബൈൻഡിങ് ഉപയോഗിക്കുന്നു.
4. C++ ___________, ___________ ബൈൻഡിങ് ഉപയോഗിക്കുന്നു
5. ഏർലി ബൈൻഡിങ്ങിനെ ___________ എന്ന് വിളിക്കുന്നു.
6. ലേറ്റ് ബൈൻഡിങ്ങിനെ ___________ എന്ന് വിളിക്കുന്നു.
7. വിവിധ തരത്തിലുള്ള ഇൻഹെറിറ്റൻസ് ഏതൊക്കെയാണ്?

നമുക്കു സംഗ്രഹിക്കാം

സോഫ്റ്റ്‌വെയർ കമ്പ്യൂട്ടറിനെ ഉപയോഗപ്രദമായ ഒരു ഉപകരണമാക്കി മാറ്റുന്നതിനാൽ, അതിന്റെ നിർമ്മാണത്തിനും പരിപാലനത്തിലും പ്രത്യേക പരിഗണന ആവശ്യമാണ്. മറ്റൊരു തരത്തിൽ പറഞ്ഞാൽ സോഫ്റ്റ്‌വെയർ വികസനം ഫലപ്രദമാക്കാനും പരിപാലനത്തിനുള്ള ചെലവ് കുറക്കാനും പല തരത്തിലുള്ള മാതൃകകൾ പരീക്ഷിക്കപ്പെട്ടിട്ടുണ്ട്. സ്ട്രക്ചർഡ് മാതൃക, പ്രോസിജറൽ ഓറിയന്റഡ് മാതൃക, ഒബ്ജക്റ്റ് ഓറിയന്റഡ് മാതൃക (OOP) മുതലായവ അവയ്ക്ക് ഉദാഹരണമാണ്. നൂതനവും പ്രചാരത്തിലുള്ളതുമായ പ്രോഗ്രാമിങ് ഭാഷകൾ OOP മാതൃക പിന്തുടരുന്നു. പരസ്പരം ആശയവിനിമയം നടത്തുന്ന വസ്തുക്കളുടെ (ഒബ്ജക്റ്റുകളുടെ) സഹായത്തോടെയാണ് OOP പ്രാവർത്തികമാക്കിയിരിക്കുന്നത്. ഇവിടെ ഡാറ്റക്ക് കൂടുതൽ പ്രാധാന്യം നൽകിയിരിക്കുന്നു. ആധികാരികമല്ലാത്ത ഡാറ്റയുടെ ഉപയോഗത്തിൽനിന്നും സംരക്ഷണം ലഭിക്കാൻ വിവിധ ആക്സസ് സ്പെസിഫയറുകൾ ഫങ്ഷനുകളുടെ കൂടെ ഉൾപ്പെടുത്തുകയും ചെയ്യുന്നു. പുനരുപയോഗം മെച്ചപ്പെടുത്തുന്നതിനോടൊപ്പം കോഡ് വിപുലീകരണത്തിനും ഫലപ്രദമായ ഘടനാ ക്രമീകരണത്തിനും സ്റ്റാറ്റിക്, ഡൈനാമിക് എന്നീ പോളിമോർഫിസം പ്രാവർത്തികമാക്കാനും OOP സഹായിക്കുന്നു.

നമുക്കു വിലയിരുത്താം

1. അധികാരികത ഇല്ലാത്ത ഫങ്ഷനുകളിൽനിന്നും ഡാറ്റയെ സംരക്ഷിക്കുന്നതാണ് ________.
   a. പോളിമോർഫിസം   b. എൻക്യാപ്സുലേഷൻ
   c. ഡാറ്റ അബ്സ്ട്രാക്ഷൻ   d. ഇൻഹെറിറ്റൻസ്
2. അടിസ്ഥാന ക്ലാസിനെ ________ എന്നും വിളിക്കുന്നു.
   a. ചൈൽഡ് ക്ലാസ്സ്   b. സബ് ക്ലാസ്സ്   c. ഡിറൈവ്ഡ് ക്ലാസ്   d. പേരന്റ് ക്ലാസ്
3. താഴെ പറയുന്നവയിൽ ഇൻഹെറിറ്റൻസ് അല്ലാത്തത് ഏത്?
   a. ഹൈബ്രിഡ്   b. മൾട്ടിപ്പിൾ   c. മൾട്ടിലെവൽ   d. മൾട്ടിക്ലാസ്സ്
4. സബ് ക്ലാസിനു തുല്യമാണ്
   a. രൂപീകൃത ക്ലാസ്   b. സൂപ്പർ ക്ലാസ്
   c. അടിസ്ഥാന ക്ലാസ്   d. ഇതൊന്നും അല്ല
5. തനതായുള്ള ആക്സസ് സ്പെസിഫയർ ആണ്
   a. പബ്ലിക്   b. പ്രൈവറ്റ്   c. പ്രൊട്ടക്റ്റഡ്   d. ഇവ ഒന്നുമല്ല
6. താഴെ പറയുന്നവയിൽ OOP ആശയം അല്ലാത്തതേത്
   a. ഓവർലോഡിങ്   b. പ്രോസിജറൽ ഓറിയന്റഡ് പ്രോഗ്രാമിങ്
   c. ഡാറ്റ അബ്സ്ട്രാക്ഷൻ   d. ഇൻഹെറിറ്റൻസ്

7. ഒരു ഡാറ്റക്ക് അല്ലെങ്കിൽ മെസ്സേജിന് ഒന്നിലധികം രൂപത്തിൽ പ്രോസസ്സ് ചെയ്യപ്പെടാനുള്ള കഴിവാണ്.

   a. പോളിമോർഫിസം b. എൻക്യാപ്സുലേഷൻ

   c. ഡാറ്റ ഹൈഡിങ് d. ഇൻഹെറിറ്റൻസ്

8. C++ ഒരു ____________ ഭാഷയാണ്.

   a. ഒബ്ജക്ട് b. നോൺ പ്രൊസിജറൽ

   c. ഒബ്ജക്റ്റ് ഓറിയന്റഡ് d. പ്രൊസിജറൽ

9. താഴെ പറയുന്നവയിൽ OOP -ന്റെ പ്രത്യേകത അല്ലാത്തതേത്?

   a. ഡാറ്റയെക്കാൾ നടപടികൾക്ക് പ്രാധാന്യം നൽകുന്നു

   b. യഥാർഥ ലോകത്തിന്റെ മാതൃകയിലുള്ളത്

   c. ഡാറ്റയെയും ബന്ധപ്പെട്ട ഫങ്ക്ഷനുകളെയും ഒറ്റ ഘടകമായി കൂട്ടിയിണക്കുന്നു.

   d. ഇവ ഒന്നുമല്ല

10. താഴെ പറയുന്നവയിൽ OOP -നെ കുറിച്ച് ശരിയായതേത്?

   a. ഡാറ്റ അബ്സ്ട്രാക്ഷനെ പിന്തുണക്കുന്നു

   b. പോളിമോർഫിസത്തെ പിന്തുണക്കുന്നു

   c. ഘടനാപരമായ പ്രോഗ്രാമിങ്ങിനെ പിന്തുണക്കുന്നു

   d. ഇവയെയെല്ലാം പിന്തുണക്കുന്നു

11. പ്രോഗ്രാമിങ് മാതൃക എന്നാലെന്ത്? പ്രോഗ്രാമിങ് മാതൃകകളുടെ പേരെഴുതുക.

12. പ്രോസിജറൽ പ്രോഗ്രാമിങ് സമീപനത്തിന്റെ പരിമിതികൾ എന്തെല്ലാം?

13. ഒബ്ജക്റ്റ് ഓറിയന്റഡ് പ്രോഗ്രാമിങ് മാതൃക എന്നാലെന്ത്? OOP ന്റെ അടിസ്ഥാന തത്വങ്ങൾ പട്ടികപ്പെടുത്തുക.

14. C++ ൽ എങ്ങനെയാണു OOP പ്രയോഗിച്ചിരിക്കുന്നത്?

15. എൻക്യാപ്സുലേഷൻ എന്നാലെന്ത്?

16. ക്ലാസും ഒബ്ജക്റ്റും തമ്മിലുള്ള വ്യത്യാസം കണ്ടെത്തുക.

17. ബേസ് ക്ലാസ്, സബ് ക്ലാസ് എന്നാലെന്ത്? ബേസ് ക്ലാസും സബ് ക്ലാസും തമ്മിലുള്ള ബന്ധം എന്താണ്?

18. ഡാറ്റ അബ്സ്ട്രാക്ഷൻ എന്ന ആശയം വിവരിക്കുക. ഒരു ഉദാഹരണം നൽകുക.

19. ഇൻഹെറിറ്റൻസിനെ പറ്റി ഒരു ലഘു വിവരണം എഴുതുക.

20. ഒരു കാർ പ്രവർത്തിപ്പിക്കുന്നതിനായി സ്റ്റിയറിംഗ്, ബ്രേക്ക്, ആക്സിലറേറ്റർ മുതലായവ നമ്മൾ ഉപയോഗിക്കുന്നു. ഇവ ഉപയോഗിക്കുമ്പോൾ ആന്തരികമായി എന്താണ് സംഭവിക്കുന്നത് എന്നു നമുക്കറിയേണ്ടതില്ല. ഏതെങ്കിലും OOP ആശയവുമായി നിങ്ങൾക്കിതിനെ ബന്ധപ്പെടുത്താമോ? വിശദീകരിക്കുക.

21. ഇൻഹെറിറ്റൻസ് എന്നാലെന്ത്? പുനരുപയോഗത്തിന് ഇത് എങ്ങനെ സഹായകമാകുന്നു?

22. പോളിമോർഫിസം എന്നാലെന്ത്? ഇത് വിവരിക്കുവാൻ ഒരു ഉദാഹരണം നൽകുക.

23. OOP എന്ന ആശയം ഉദാഹരണ സഹിതം വിവരിക്കുക.

24. സ്വിച്ചോടു കൂടിയ ഒരു പ്ലഗ് പോയിൻറ് പരിഗണിക്കുക. സ്വിച്ചിന്റെ പ്രവർത്തനം എങ്ങനെയായിരിക്കും എന്നത്, ഏതു സാഹചര്യത്തിലാണത് ഉപയോഗിക്കുന്നത് എന്നതിന്റെയും പ്ലഗ് പോയിന്റിൽ എന്താണ് ഘടിപ്പിച്ചിരിക്കുന്നത് എന്നതിന്റെയും അടിസ്ഥാനത്തിയായിരിക്കും. ഏതെങ്കിലും OOP ആശയവുമായി നിങ്ങൾക്കിതിനെ ബന്ധപ്പെടുത്താമോ? വിശദീകരിക്കുക.

25. 'Horse', 'Fish', 'Bird' എന്നീ ഉപക്ലാസുകളെ ഉൽപാദിപ്പിച്ചിരിക്കുന്ന 'LivingBeings' എന്നൊരു അടിസ്ഥാന ക്ലാസ് പരിഗണിക്കുക. പരാമർശിച്ചിരിക്കുന്ന എല്ലാ ക്ലാസുകളും ഇൻഹെറിറ്റ് ചെയ്യുന്ന 'Move' എന്നൊരു ഫങ്ഷൻ LivingBeings ക്ലാസിലുണ്ടെന്ന് അനുമാനിക്കുക. Horse ക്ലാസിന്റെ ഒബ്ജക്ട് 'Move' ഫങ്ഷനെ വിളിക്കുമ്പോൾ, കുതിക്കുന്നു എന്നായിരിക്കാം സ്ക്രീനിൽ പ്രദർശിപ്പിക്കുന്നത്. അതേസമയം Fish ക്ലാസിന്റെ ഒബ്ജക്ട് അതേ ഫങ്ഷനെ വിളിക്കുമ്പോൾ നീന്തുന്നു എന്നായിരിക്കാം സ്ക്രീനിൽ പ്രദർശിപ്പിക്കുക. Bird ന്റെ ഒബ്ജക്ടിന്റെ കാര്യത്തിൽ പറക്കുന്നു എന്നായിരിക്കാം. ഏതെങ്കിലും OOP ആശയവുമായി നിങ്ങൾക്കിതിനെ ബന്ധപ്പെടുത്താമോ? വിശദീകരിക്കുക.

Z9Z2T4

3 ഡാറ്റ സ്ട്രക്ചറുകളും പ്രവർത്തനങ്ങളും

പ്രധാന പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പഠനം പൂർത്തിയാകുന്നതോടെ പഠിതാവ് ആർജിക്കേണ്ട പഠനനേട്ടങ്ങൾ

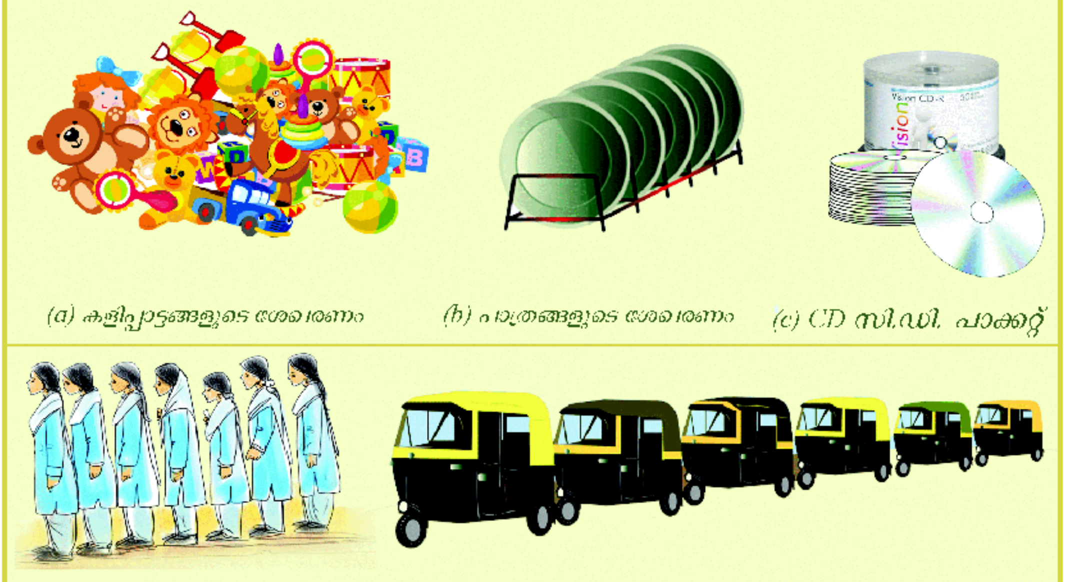
- ഡാറ്റ സ്ട്രക്ചറുകളുടെ ആശയം ഉദാഹരണ സഹിതം വിവരിക്കുവാൻ സാധിക്കുന്നു.
- ഡാറ്റ സ്ട്രക്ചറുകളെ വിവിധ മാനദണ്ഡങ്ങൾക്കനുസരിച്ചു തരംതിരിക്കാൻ സാധിക്കുന്നു.
- ഡാറ്റ സ്ട്രക്ചറുകളിലുള്ള വിവിധ പ്രവർത്തനങ്ങൾ ഏതൊക്കെയെന്നും മനസ്സിലാക്കാനും അവയെക്കുറിച്ച് വിശദീകരിക്കുവാനും സാധിക്കുന്നു.
- സ്റ്റാക്ക് ഡാറ്റ സ്ട്രക്ചറിന്റെ ഘടന ഉദാഹരണസഹിതം വിശദീകരിക്കാൻ സാധിക്കുന്നു.
- സ്റ്റാക്കിലെ പുഷ്, പോപ് പ്രവർത്തനങ്ങൾക്കുള്ള അൽഗോരിതം വികസിപ്പിക്കാൻ സാധിക്കുന്നു.
- ക്യൂ സ്ട്രക്ചറിന്റെ ഘടന ഉദാഹരണ സഹിതം വിശദീകരിക്കാൻ സാധിക്കുന്നു.
- രേഖീയമായ ക്യൂവിൽ ഉൾപ്പെടുത്താനും, നീക്കം ചെയ്യൽ പ്രവർത്തനങ്ങൾക്കുള്ള അൽഗോരിതം വികസിപ്പിക്കാനും സാധിക്കുന്നു.
- വൃത്താകൃതിയിലുള്ള ക്യൂവിനു രേഖീയമായ ക്യൂവിനു മേലുള്ള നേട്ടങ്ങൾ വിശദീകരിക്കാൻ സാധിക്കുന്നു.
- ലിങ്ക്ഡ് ലിസ്റ്റ് ഡാറ്റ സ്ട്രക്ചറുകളുടെ ആശയം വിശദീകരിക്കാനും അറെകളുടെയും മറ്റു സ്ഥിര ഡാറ്റ സ്ട്രക്ചറുകളുടെയും മേൽ അവക്കുള്ള നേട്ടങ്ങൾ വിവരിക്കാനും സാധിക്കുന്നു.

കമ്പ്യൂട്ടറുകൾ ഉപയോഗിച്ച് പ്രശ്നം പരിഹരിക്കുമ്പോൾ മിക്ക സന്ദർഭങ്ങളിലും ഡാറ്റ പ്രോസസ് ചെയ്യേണ്ടി വരാം. ഈ ഡാറ്റയുടെ ഇനം അടിസ്ഥാനപരമായതോ സംയോജിതമായതോ (ഒരു കൂട്ടം) ആകാം. ഇത്തരം ഡാറ്റയെ പരാമർശിക്കുന്നതിനു വേണ്ടി വേരിയബിളുകൾ ആവശ്യമാണെന്ന് നമുക്കറിയാവുന്നതാണ്. C, C++, ജാവ മുതലായ ഭാഷകളിലെ പ്രോഗ്രാമിൽ വേരിയബിളുകൾ ഉപയോഗിക്കുന്നതിനു മുമ്പ് അവയുടെ പ്രഖ്യാപനം വേണമെന്ന് ഊന്നിപ്പറയുന്നു. അടിസ്ഥാനപരമായ ഡാറ്റകൾക്ക് വേണ്ടിയുള്ള വേരിയബിളുകൾ `int`, `char`, `float`, `double` മുതലായ അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളോ അവയുടെ ടൈപ്പ് മോഡിഫൈറുകളോ ഉപയോഗിച്ചാണ് പ്രഖ്യാപിക്കുന്നതെന്നു നമ്മൾ പഠിച്ചതാണ്. കൂട്ടമായിട്ടുള്ള ഡാറ്റയെ സൂചിപ്പിക്കാൻ അറെകളും സ്ട്രക്ചറുകളും ഉപയോഗിക്കുന്നു എന്നും നാം കണ്ടതാണ്. ഒരേ തരത്തിലുള്ള ഡാറ്റയുടെ ശേഖരത്തെ അറെ എന്നും വ്യത്യസ്ത തരത്തിലുള്ള ഡാറ്റയുടെ ശേഖരത്തെ സ്ട്രക്ചർ എന്നും പറയുന്നു.

വിവിധ തത്വങ്ങളുടെയും മാനദണ്ഡങ്ങളുടെയും അടിസ്ഥാനത്തിൽ പ്രോഗ്രാമിങ് ഭാഷകളിൽ ഡാറ്റയെ വ്യത്യസ്ത തരങ്ങളായി ചിട്ടപ്പെടുത്താൻ ഉപയോഗിക്കുന്ന സംവിധാനങ്ങളെയാണ് ഈ അധ്യായത്തിൽ അവതരിപ്പിച്ചിരിക്കുന്നത്. ഒരു തരത്തിൽ ഉൾക്കൊള്ളാവുന്ന ഡാറ്റയുടെ അളവും അതിന്റെ മുകളിൽ പ്രാവർത്തികമാക്കാവുന്ന പ്രവർത്തനങ്ങളും, ഡാറ്റ ഗണങ്ങൾ ചിട്ടപ്പെടുത്തുവാൻ ഉപയോഗിക്കുന്ന തത്വങ്ങളുടെ അടിസ്ഥാനത്തിൽ വ്യത്യസ്തമായതായിരിക്കും.

3.1 ഡാറ്റ സ്ട്രക്ചർ (Data Structure)

ചിത്രം 3.1 ൽ ചില ഗ്രൂപ്പുകൾ കാണിച്ചിരിക്കുന്നു. ഓരോ ഗ്രൂപ്പിലും ക്രമീകരണത്തിനായി പലതരത്തിലുള്ള തന്ത്രങ്ങൾ ഉപയോഗിച്ചിരിക്കുന്നു. ഇതിലെ ഓരോ ചിത്രവും ഓരോ വസ്തുക്കളുടെ ശേഖരമാണ്. ഓരോ ഗ്രൂപ്പിലും വസ്തുക്കളെ ക്രമീകരിക്കാൻ ഉപയോഗിച്ചിരിക്കുന്ന തത്വം അല്ലെങ്കിൽ മാതൃക ഏതാണെന്നു പറയാമോ?



(a) കളിപ്പാട്ടങ്ങളുടെ ശേഖരണം (b) പാത്രങ്ങളുടെ ശേഖരണം (c) CD സി.ഡി. പാക്കറ്റ്

(d) ക്യൂ

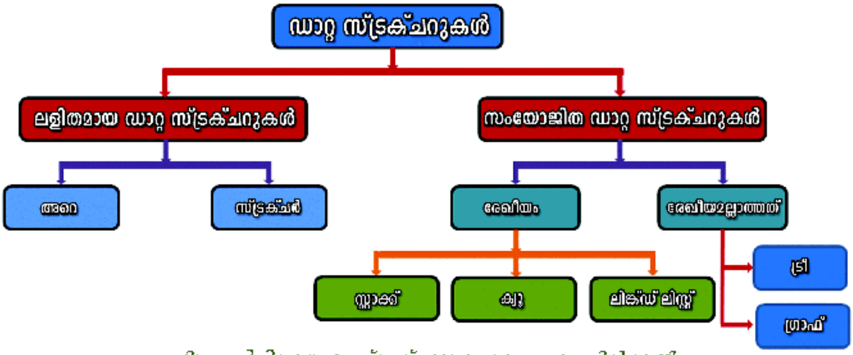
ചിത്രം 3.1: വിവിധ തരത്തിലുള്ള ശേഖരങ്ങൾ

കളിപ്പാട്ടങ്ങളുടെ ഒരു ശേഖരമാണ് ചിത്രം 3.1(a) ൽ കാണുന്നത്. പ്രത്യേകിച്ച് ഒരു ക്രമമോ ക്രമീകരണമോ ഇല്ലാതെ കൂട്ടിയിട്ടിരിക്കുകയാണ് ഈ കളിപ്പാട്ടങ്ങൾ. ചിത്രം 3.1(b) ൽ കാണിച്ചിരിക്കുന്നത് ഒരു തട്ടിലെ ഒരു കൂട്ടം പാത്രങ്ങളാണ്. ഒന്നിന് പുറകിൽ മറ്റൊന്നായിട്ടാണ് പാത്രങ്ങൾ വച്ചിരിക്കുന്നത്. തട്ടിൽ സൂക്ഷിക്കാവുന്ന പാത്രങ്ങളുടെ എണ്ണത്തിന് ഒരു പരിമിതിയുണ്ട്. സ്ഥലമുണ്ടെങ്കിൽ തട്ടിൽ എവിടെ വേണമെങ്കിലും പുതിയ പാത്രം വയ്ക്കാവുന്നതും ഏതു പാത്രം വേണമെങ്കിലും തിരികെ എടുക്കാവുന്നതുമാണ്. സി ഡി പായ്ക്കറ്റിലെ ഒരു കൂട്ടം സി ഡികൾ ആണ് ചിത്രം 3.1(c) ൽ കാണിച്ചിരിക്കുന്നത്. ഈ ശേഖരത്തിലും ഡിസ്കുകളുടെ എണ്ണത്തിന് ഒരു പരിധിയുണ്ട്. ഒരു പുതിയ സി.ഡി. ശേഖരത്തിന്റെ മുകളിൽ മാത്രമേ ചേർക്കാൻ സാധ്യമാകുകയുള്ളൂ. അതുപോലെ മുകളിലുള്ള സി.ഡി. യെ മാത്രമെ ശേഖരത്തിൽ നിന്നും ഒഴിവാക്കുവാനും സാധിക്കുകയുള്ളൂ. പുറകുവശത്തു കൂടി മാത്രം അടുത്ത വ്യക്തിക്ക് (അല്ലെങ്കിൽ അടുത്ത ഓട്ടോറിക്ഷക്ക്) വരിയിൽ ചേരാൻ കഴിയുന്ന തരത്തിലുള്ള ഒരു വരി ആണ് ചിത്രം 3.1(d) ൽ സൂചിപ്പിക്കുന്നത്. വരിയുടെ മുൻവശത്തു കൂടി മാത്രമെ ഒരു വ്യക്തിക്ക് (അല്ലെങ്കിൽ ഓട്ടോറിക്ഷയ്ക്ക്) വരിയിൽ നിന്ന് പുറത്തേക്കു പോകുവാൻ സാധിക്കുകയുള്ളൂ. ഈ വരിയിൽ വ്യക്തികളുടെ എണ്ണത്തിന് പരിധി ഉണ്ടാകണമെന്നില്ല. എന്നാൽ ചില സാഹചര്യങ്ങളിൽ വരിയുടെ വലുപ്പത്തിന് പരിധിയുണ്ടാകാം.

ചിത്രം 3.1 ലെ (b), (c), (d) എന്നീ ചിത്രങ്ങളിലെ ശേഖരങ്ങളുമായി സാമ്യമുള്ളതാണ് ഡാറ്റാസ്ട്രക്ചറുകളുടെ തത്വം. പതിനൊന്നാം തരത്തിൽ നമ്മൾ പഠിച്ച അറെയുമായി സാമ്യമുള്ളതാണ് ചിത്രം 3.1(b)ൽ കാണിച്ചിരിക്കുന്ന ശേഖരം. അതുകൊണ്ട് അറെ ഒരു ഡാറ്റ സ്ട്രക്ചർ ആണെന്ന് നമുക്ക് പറയാവുന്നതാണ്. ഒറ്റ ഘടകമായി പ്രോസസ് ചെയ്യാവുന്ന യുക്തിപരമായി ഒരേപോലെ ഉള്ളതും അല്ലെങ്കിൽ വ്യത്യസ്തമായതും ആയ ഡാറ്റയെ പ്രത്യേക രീതിയിൽ ക്രമീകരിക്കാനുള്ള മാർഗത്തെയാണ് കമ്പ്യൂട്ടർ ശാസ്ത്രത്തിൽ ഡാറ്റ സ്ട്രക്ചർ എന്ന് പറയുന്നത്. ഉപയോക്താവിന് വിവിധ തരത്തിലുള്ള ഡാറ്റയെ കൂട്ടിയോജിപ്പിക്കാൻ മാത്രമല്ല അവയെ ഒറ്റ ഘടകമായി കണക്കാക്കി പ്രവർത്തിക്കാനും ഡാറ്റസ്ട്രക്ചർ സഹായിക്കുന്നു.

3.1.1 ഡാറ്റ സ്ട്രക്ചറുകളുടെ തരം തിരിക്കൽ (Classification of data structures)

ഡാറ്റ സ്ട്രക്ചറുകളെ പൊതുവെ ലളിതമായ ഡാറ്റ സ്ട്രക്ചറുകളെന്നും സംയോജിത ഡാറ്റ സ്ട്രക്ചറുകളെന്നും തരംതിരിക്കാം. ഇതിന്റെ വിശദമായ തരംതിരിവ് ചിത്രം 3.2 ൽ കാണിച്ചിരിക്കുന്നു.



ചിത്രം. 3.2: ഡാറ്റ സ്ട്രക്ചറുകളുടെ തരംതിരിക്കൽ

അറേ, സ്ട്രക്ചർ തുടങ്ങിയ ലളിതമായ ഡാറ്റ സ്ട്രക്ചറുകൾ നമുക്ക് പരിചിതമാണ്. ഡാറ്റയുടെ ശേഖരത്തെ സൂചിപ്പിക്കുന്നതിനായി C++ പ്രോഗ്രാമുകളിൽ നമ്മൾ അവ ഉപയോഗിച്ചതുമാണ്. ഇത്തരം ലളിതമായ ഡാറ്റ സ്ട്രക്ചറുകളെ വിവിധ തരത്തിൽ കൂട്ടിയോജിപ്പിച്ച് സംയോജിത ഡാറ്റ സ്ട്രക്ചറുകൾ രൂപപ്പെടുത്തുന്നു. ചിത്രം 3.2 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ സംയോജിത ഡാറ്റ സ്ട്രക്ചറുകളെ രേഖീയമായതും (Linear) രേഖീയമല്ലാത്തതും (Non-Linear) എന്ന് വീണ്ടും തരംതിരിച്ചിരിക്കുന്നു. ഒരു ഡാറ്റ സ്ട്രക്ചറിലെ ഡാറ്റ അംഗങ്ങൾ അനുവർത്തനമായി ക്രമീകരിക്കപ്പെടുകയാണെങ്കിൽ ആ ഡാറ്റ സ്ട്രക്ചറിനെ രേഖീയമായത് എന്ന് പറയുന്നു. രേഖീയമായ ഡാറ്റ സ്ട്രക്ചറിന്റെ അംഗങ്ങളെ പ്രതിനിധീകരിക്കാൻ തുടർച്ചയായ മെമ്മറി ലൊക്കേഷനുകൾ ഉപയോഗിക്കുന്നു. ഒരു ഡാറ്റ സ്ട്രക്ചറിലെ അംഗങ്ങൾ മെമ്മറിയിൽ അനുവർത്തന ക്രമത്തിലല്ലെങ്കിൽ ആ ഡാറ്റ സ്ട്രക്ചറിനെ ഒരു രേഖീയമല്ലാത്ത ഡാറ്റ സ്ട്രക്ചർ എന്ന് പറയുന്നു. ഇത്തരം ഡാറ്റ സ്ട്രക്ചറുകളിൽ ക്രമത്തിലല്ലാതെയുള്ള മെമ്മറി ലൊക്കേഷനുകളിലാണ് ഡാറ്റ അംഗങ്ങൾ സംഭരിക്കപ്പെടുക. മാത്രമല്ല അവ ഉപയോഗിക്കപ്പെടുന്നത് ക്രമത്തിലായിരിക്കണമെന്നില്ല. രേഖീയമല്ലാത്ത ഡാറ്റ സ്ട്രക്ചറുകൾ വളരെ സങ്കീർണമായതിനാൽ ഉപരിപഠന സമയത്ത് അവയെപ്പറ്റി വിശദമായി പഠിക്കുന്നതാണ്. രേഖീയമായ ഡാറ്റ സ്ട്രക്ചറുകളായ സ്റ്റാക്ക് (Stack), ക്യൂ (Queue), ലിങ്ക്ഡ് ലിസ്റ്റ് (Linked List) എന്നിവയെപ്പറ്റി വിശദമായി ഈ പാഠഭാഗത്തിൽ അവതരിപ്പിച്ചിരിക്കുന്നു.

ഡാറ്റ സ്ട്രക്ചറുകൾ ഡാറ്റയുടെ ശേഖരത്തെ സൂചിപ്പിക്കുന്നതിനാൽ അവയ്ക്ക് കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയുമായി വളരെ അടുത്ത ബന്ധമാണുള്ളത്. എന്തെന്നാൽ ഡാറ്റയെ സംഭരിക്കാനുള്ള സ്ഥലമാണ് മെമ്മറി. മെമ്മറി പ്രാഥമികമോ ദ്വിതീയമോ ആകാം. മെമ്മറി നീക്കിവയ്പ്പിനനുസരിച്ച് ഡാറ്റ സ്ട്രക്ചറുകളെ സ്ഥിര (static) ഡാറ്റ സ്ട്രക്ചർ എന്നും പരിവർത്തനാത്മക (dynamic) ഡാറ്റ സ്ട്രക്ചർ എന്നും രണ്ടായി തരംതിരിക്കാം. സ്റ്റാറ്റിക് ഡാറ്റ സ്ട്രക്ചറുകൾ പ്രാഥമിക മെമ്മറിയുമായി മാത്രം ബന്ധപ്പെട്ടിരിക്കുന്നു. പ്രോഗ്രാമിന്റെ പ്രവർത്തനത്തിനു മുമ്പായി ആവശ്യമായ മെമ്മറി നീക്കിവയ്ക്കുകയും പ്രവർത്തന ഘട്ടത്തിലുടനീളം അതിനു മാറ്റംവരാതെ നിലകൊള്ളുകയും ചെയ്യുന്നു. അതായത് ഡാറ്റ സ്ട്രക്ചർ രൂപകൽപ്പന ചെയ്യുമ്പോൾത്തന്നെ അതിന്റെ വലുപ്പം നിർണയിക്കുന്നതുകൊണ്ട് പിന്നീട് അത് മാറ്റാൻ സാധിക്കുന്നതല്ല. അറേകൾ ഉപയോഗിച്ച് രൂപകൽപന ചെയ്തിരിക്കുന്നതും പ്രാവർത്തികമാക്കിയിരിക്കുന്നതും ആയ ഡാറ്റ സ്ട്രക്ചറുകൾ സ്ഥിര സ്വഭാവമുള്ളവയായിരിക്കും. പക്ഷെ പരിവർത്തനാത്മക ഡാറ്റ സ്ട്രക്ചറുകൾക്ക് പ്രവർത്തന ഘട്ടത്തിലാണ് മെമ്മറി നിർണയിക്കപ്പെടുക. ലിങ്ക്ഡ് ലിസ്റ്റുകൾ ഉപയോഗിച്ച് പ്രാവർത്തികമാക്കിയിരിക്കുന്ന ഡാറ്റ സ്ട്രക്ചറുകൾ പരിവർത്തനാത്മകമായിരിക്കും. പരിവർത്തനാത്മക ഡാറ്റ സ്ട്രക്ചറിൽ ശേഖരത്തിന്റെ വലുപ്പം മുൻകൂട്ടി പ്രസ്താവിച്ചിരിക്കുകയില്ല, പകരം ഉപയോക്താവിന്റെ ഇഷ്ടാനുസരണം പ്രവർത്തന വേളയിൽ വളരുകയും ചുരുങ്ങുകയും ചെയ്തു കൊണ്ടിരിക്കും. ഡാറ്റ സംഭരണത്തിനായി ദ്വിതീയ മെമ്മറി നമ്മൾ ഉപയോഗിക്കുമ്പോൾ അത് ഫയലുകളുടെ രൂപത്തിലായിരിക്കും. ഡാറ്റ കൂട്ടിച്ചേർക്കുന്നതിനനുസരിച്ചു ഫയലിന്റെ വലുപ്പം കൂടുകയും ഡാറ്റ നീക്കം ചെയ്യുന്നതിനനുസരിച്ച് വലുപ്പം കുറയുകയും ചെയ്യുന്നു. അതിനാൽ ഫയലുകളെ പരിവർത്തനാത്മക ഡാറ്റ സ്ട്രക്ചറുകൾ എന്ന് പറയുന്നു.

3.1.2 ഡാറ്റ സ്ട്രക്ചറുകളുടെ മേലുള്ള പ്രവർത്തനങ്ങൾ (Operations on data structures)

ഡാറ്റ സ്ട്രക്ചറുകൾ ഉപയോഗിച്ച് പ്രതിനിധീകരിച്ചിരിക്കുന്ന ഡാറ്റ ചില പ്രവർത്തനങ്ങളുടെ സഹായത്തോടെയാണ് പ്രോസസ് ചെയ്യപ്പെടുന്നത്. വാസ്തവത്തിൽ ചില പ്രത്യേക പ്രവർത്തനങ്ങൾ എത്ര തവണ പ്രവർത്തിക്കണം എന്നതിനനുസരിച്ചാണ് ഡാറ്റ സ്ട്രക്ചർ തിരഞ്ഞെടുക്കുന്നത്. കടന്നു പോകുക, തിരയുക, ഉൾപ്പെടുത്തുക, നീക്കം ചെയ്യുക, ക്രമപ്പെടുത്തുക, ലയിപ്പിക്കുക മുതലായവയാണ് ഡാറ്റ സ്ട്രക്ചറുകളുടെ മേൽ ചെയ്യാവുന്ന പ്രവർത്തനങ്ങൾ. ഇവയെ നമുക്ക് പരിചയപ്പെടാം.

a. കടന്നുപോകുക (Traversing)

ഡാറ്റ സ്ട്രക്ചറിലെ ഓരോ അംഗത്തെയും സന്ദർശിക്കുന്ന പ്രവർത്തനമാണ് കടന്നുപോകൽ. ആദ്യത്തെ അംഗത്തിൽ തുടങ്ങി അവസാനത്തെ അംഗം വരെ സഞ്ചാരം തുടരുന്നു. സന്ദർശിച്ച അംഗത്തെ എങ്ങനെ പ്രോസസ് ചെയ്യണമെന്നുള്ളത് പ്രശ്നത്തിന്റെ ആവശ്യകതയ്ക്കനുസരിച്ചായിരിക്കും. ഒരു അറേയിലുള്ള എല്ലാ അംഗങ്ങളും തിരിച്ചെടുക്കുന്നത് കടന്നുപോകുന്ന പ്രവർത്തനത്തിന് ഒരു ഉദാഹരണമാണ് (പതിനൊന്നാം തരത്തിലെ കമ്പ്യൂട്ടർ സയൻസ് പുസ്തകത്തിലെ അധ്യായം 8 നോക്കുക).

b. തിരയൽ (Searching)

ഒരു ഡാറ്റ സ്ട്രക്ചറിലെ ഒരു പ്രത്യേകഅംഗത്തിന്റെ സ്ഥാനം കണ്ടുപിടിക്കാനുള്ള പ്രക്രിയയെയാണ് തിരയൽ എന്നത് കൊണ്ടുദ്ദേശിക്കുന്നത്. ഒന്നോ അതിലധികമോ വ്യവസ്ഥ

കൾ നിറവേറ്റുന്ന എല്ലാ അംഗങ്ങളുടെയും സ്ഥാനം കണ്ടുപിടിക്കാനുള്ള പ്രക്രിയയെയും തിരയൽ എന്നുപറയാം. മറ്റൊരർഥത്തിൽ തിരയലെന്നാൽ ഡാറ്റ സ്ട്രക്ച്ചറിൽ സംഭരിച്ചു വച്ചിരിക്കുന്ന വിലകൾ കണ്ടെത്തി ഉപയോഗിക്കുക എന്നാണ്. പതിനൊന്നാം തരത്തിൽ തിരയലിനുള്ള രണ്ടു മാർഗങ്ങൾ നമ്മൾ പഠിച്ചതാണ്.

c. ഉൾപ്പെടുത്തൽ (Inserting)

ഡാറ്റ സ്ട്രക്ച്ചറിൽ ഒരു പ്രത്യേക സ്ഥാനത്തേക്ക് പുതിയ അംഗത്തെ ചേർക്കുന്ന പ്രക്രിയയാണ് ഉൾപ്പെടുത്തൽ എന്ന് പറയുന്നത്. ചില സാഹചര്യങ്ങളിൽ, പ്രത്യേകിച്ച് ഡാറ്റ സ്ട്രക്ച്ചറിലെ ഡാറ്റ അംഗങ്ങൾ ക്രമത്തിലാണെങ്കിൽ, പുതിയ ഡാറ്റ ഉൾപ്പെടുത്തേണ്ട സ്ഥാനം ആദ്യം കണ്ടെത്തുകയും അതിനു ശേഷം ഡാറ്റ ഉൾപ്പെടുത്തുകയും ചെയ്യുന്നു.

d. നീക്കം ചെയ്യൽ (Deleting)

ഡാറ്റ സ്ട്രക്ച്ചറിൽ നിന്നും ഒരു അംഗത്തെ ഒഴിവാക്കുന്ന പ്രക്രിയയെയാണ് നീക്കം ചെയ്യൽ എന്നു പറയുന്നത്. നീക്കം ചെയ്യുന്നതിനായി ഡാറ്റ അംഗത്തിന്റെ സ്ഥാനമോ നീക്കം ചെയ്യേണ്ട അംഗത്തെ തന്നെയോ പരാമർശിക്കണം.

e. ക്രമപ്പെടുത്തൽ (Sorting)

ബബിൾ സോർട്ട്, സെലക്ഷൻ സോർട്ട് എന്നീ രണ്ടു രീതികൾ ഉപയോഗിച്ച് ഒരു അറെയെ ക്രമപ്പെടുത്തുന്നത് നമ്മൾ നേരത്തെ പരിചയപ്പെട്ടതാണ്. അംഗങ്ങളെ ഒരു പ്രത്യേക ക്രമത്തിൽ അതായത് ആരോഹണ ക്രമത്തിലോ അവരോഹണ ക്രമത്തിലോ അടുക്കി വയ്ക്കുന്ന രീതിയെയാണ് ക്രമപ്പെടുത്തൽ എന്ന് പറയുന്നത്. ഡാറ്റ സ്ട്രക്ച്ചറിലെ ഡാറ്റാ അംഗങ്ങളെ ക്രമപ്പെടുത്തുന്നത് തിരയൽ വേഗത്തിലാക്കാൻ സഹായിക്കുന്നു.

f. ലയിപ്പിക്കൽ (Merging)

ക്രമപ്പെടുത്തിയിരിക്കുന്ന രണ്ടു ഡാറ്റ സ്ട്രക്ച്ചറുകളുടെ അംഗങ്ങളെ കൂട്ടിയോജിപ്പിച്ച് പുതിയ ഒരു ഡാറ്റ സ്ട്രക്ച്ചർ രൂപീകരിക്കുന്ന പ്രക്രിയയെ ലയിപ്പിക്കൽ എന്ന് പറയുന്നു. ലയിപ്പിക്കലിന്റെ ഏറ്റവും ലളിതമായ രൂപം, അംഗങ്ങളില്ലാത്ത ഒരു ഡാറ്റ സ്ട്രക്ച്ചറിലേക്കു മറ്റു രണ്ടു ഡാറ്റ സ്ട്രക്ച്ചറുകളെ കൂട്ടിച്ചേർക്കുന്നതാണ്. അറെ ഉപയോഗിച്ചു ചെയ്യുകയാണെങ്കിൽ ആദ്യം ഒരു അറെയിലെ അംഗങ്ങൾ, അംഗങ്ങളില്ലാത്ത മൂന്നാമത്തെ അറെയിലേക്ക് പകർത്തുകയും, ശേഷം രണ്ടാമത്തെ അറെയിലെ അംഗങ്ങൾ മൂന്നാമത്തെ അറെയിലേക്ക് കൂട്ടിച്ചേർക്കുകയും ചെയ്യുന്നു.

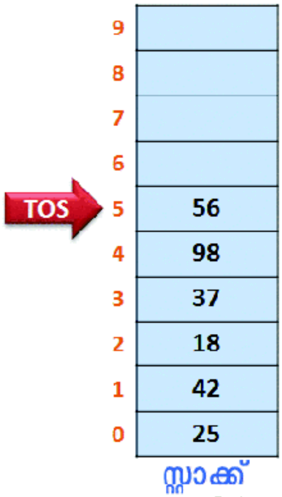
തിരയൽ, ക്രമപ്പെടുത്തൽ, ലയിപ്പിക്കൽ എന്നീ മൂന്ന് പ്രവർത്തനങ്ങളാണ് സംഭരണ ഉപകരണങ്ങളിൽ നിന്നും ഡാറ്റ വീണ്ടെടുക്കുന്ന പ്രക്രിയ എളുപ്പവും വേഗത്തിലും കാര്യക്ഷമവുമാക്കുന്നത്. അറെ എന്ന ഡാറ്റ സ്ട്രക്ച്ചറിനെക്കുറിച്ചും അതിന്മേൽ പ്രാവർത്തികമാക്കാവുന്ന മേൽപറഞ്ഞ പ്രവർത്തനങ്ങളെ പറ്റിയും നമ്മൾ പതിനൊന്നാം തരത്തിൽ പഠിച്ചതാണ്. മാത്രമല്ല സ്ട്രക്ച്ചറുകളെയും അവയുടെ അംഗങ്ങളുടെ മേലുള്ള പ്രവർത്തനങ്ങളെയും കുറിച്ച് ഈ പുസ്തകത്തിന്റെ ഒന്നാം അധ്യായത്തിൽ നാം ചർച്ച ചെയ്തിരുന്നു. ഇനി നമുക്ക് സംയോജിത രേഖീയ ഡാറ്റ സ്ട്രക്ച്ചറുകളായ സ്റ്റാക്ക്, ക്യൂ, ലിങ്ക്ഡ് ലിസ്റ്റ് മുതലായവയെപ്പറ്റി ചർച്ച ചെയ്യാം.

3.2 സ്റ്റാക്ക് (Stack)

ചിത്രം 3.1(c) ഒരിക്കൽക്കൂടി ശ്രദ്ധിക്കുക. താഴെ കൊടുത്തിരിക്കുന്ന ചിത്രം 3.3 ൽ കാണിച്ചിരിക്കുന്ന ശേഖരവും നോക്കുക. ഈ രണ്ടു വിഭാഗത്തിലും കൊടുത്തിരിക്കുന്ന ഇനങ്ങളുടെ ഘടന ഒന്നുതന്നെയാണ്.

ചിത്രം 3.3: നിത്യജീവിതത്തിലെ സ്റ്റാക്കിന് ഉദാഹരണങ്ങൾ

ഒരു ഇനത്തിന്റെ മുകളിൽ മറ്റൊന്ന് ചേർത്തിട്ടാണ് ഇവിടെ ശേഖരം രൂപീകരിച്ചിരിക്കുന്നത്. മറ്റൊരു തരത്തിൽ പറഞ്ഞാൽ ഇനങ്ങൾ ഏറ്റവും മുകളിലാണ് ചേർക്കുന്നത്. അതുപോലെ ഏറ്റവും അവസാനം ചേർത്ത ഇനം മാത്രമേ നമുക്ക് നീക്കം ചെയ്യാൻ സാധിക്കൂ. ഈ ക്രമീകരണ തത്വത്തെ ലാസ്റ്റ് ഇൻ ഫസ്റ്റ് ഔട്ട് (LIFO) എന്ന് പറയുന്നു. LIFO തത്വം പിന്തുടരുന്ന ഡാറ്റ സ്ട്രക്ചറിനെ സ്റ്റാക് എന്നു പറയുന്നു. LIFO എന്ന് പറയുന്ന അറ്റത്തു നിന്ന് മാത്രം ഇനങ്ങൾ കൂട്ടിച്ചേർക്കുകയും നീക്കംചെയ്യുകയും ചെയ്യുന്ന ക്രമപ്പെടുത്തിയ പട്ടികയാണ് ഇത്.



ചിത്രം 3.4: പൂർണ്ണസംഖ്യകളുടെ സ്റ്റാക്ക്

സ്റ്റാക്ക് എന്നത് യുക്തിപരമായ ഒരു ആശയമാണ്. സ്റ്റാക്ക് നിർമിക്കാൻ പ്രോഗ്രാമിങ് ഭാഷകളിൽ ഒരു പ്രത്യേക സംവിധാനം ഇല്ല. അറെ ഉപയോഗിച്ച് നമുക്ക് സ്റ്റാക്ക് പ്രാവർത്തികമാക്കാവുന്നതാണ്. അത്തരത്തിലുള്ള ഒരു സ്റ്റാക്കിന് അറെയുടെ എല്ലാ ഗുണങ്ങളും ഉണ്ടായിരിക്കുന്നതാണ്. ഇതിന്റെ വലുപ്പം മുമ്പേ നിർണയിക്കപ്പെട്ടിരിക്കുന്നതിനാൽ ഇത് സ്ഥിരമായിരിക്കുന്നതാണ്. അറെ ഉപയോഗിച്ച് പ്രാവർത്തികമാക്കിയിരിക്കുന്ന ഒരു സ്റ്റാക്ക് ആണ് ചിത്രം 3.4 ൽ കാണിച്ചിരിക്കുന്നത്. പരമാവധി 10 പൂർണസംഖ്യകളെ ഇതിൽ ഉൾപ്പെടുത്താനാകും. ചിത്രം 3.4 പ്രകാരം നിലവിൽ സ്റ്റാക്കിൽ ആറ് അംഗങ്ങളാണുള്ളത്. അതുപോലെ അഞ്ചാം സ്ഥാനത്തുള്ള 56 എന്നതാണ് ഈ സ്റ്റാക്കിലെ അവസാനത്തെ അംഗം. അങ്ങനെയെങ്കിൽ **TOS** ന്റെ മൂല്യം 5 ആയിരിക്കും. സ്റ്റാക്കിന്റെ അവസാനത്തെ അംഗത്തെ സൂചിപ്പിക്കുന്നതിനു വേണ്ടി സ്റ്റാക്കിന്റെ പേരിന് **StackName[TOS]** എന്ന പദപ്രയോഗം ഉപയോഗിക്കുന്നു. ഇവിടെ **STACK[TOS]** എന്നത് 56 ആയിരിക്കും. എന്തെന്നാൽ സ്റ്റാക് നിർമിക്കാൻ നമ്മൾ അറെയാണ് ഉപയോഗിച്ചിരിക്കുന്നത്. അറെയിൽ ആദ്യത്തെ അംഗത്തെ സൂചിപ്പിക്കാൻ 0 എന്ന സൂചികയാണ് ഉപയോഗിക്കുന്നത്. ഇവിടെ **STACK[0]** എന്നത് 25 ആകുന്നു.

3.2.1 സ്റ്റാക്ക് പ്രാവർത്തികമാക്കൽ (Implementation of stack)

അറെ ഉപയോഗിച്ച് സ്റ്റാക്ക് പ്രാവർത്തികമാക്കാം എന്ന് നേരത്തെ പരാമർശിച്ചിരുന്നു. അങ്ങനെയെങ്കിൽ ഒരു സ്റ്റാക്ക് ഉപയോഗിച്ച് പ്രതിനിധീകരിക്കാവുന്ന അംഗങ്ങളുടെ എണ്ണത്തിന് പരിധിയുണ്ട്. മാത്രമല്ല അത് അറെയുടെ വലുപ്പത്തിനനുസരിച്ചായിരിക്കും. തുടക്കത്തിൽ സ്റ്റാക്ക് ശൂന്യമാണ് എന്നതു സൂചിപ്പിക്കുന്നതിനായി, TOS ന്റെ വില 1 ആയി നൽകുന്നു. അറെയുടെ സൂചികയുടെ ഏറ്റവും ഉയർന്ന വില എത്തുന്നത് വരെ, സ്റ്റാക്കിലേക്ക് ഓരോ അംഗവും കൂട്ടിച്ചേർക്കപ്പെടുമ്പോൾ TOS ന്റെ വില ഒന്നു വച്ച് കൂടുന്നു. N അംഗങ്ങളുള്ള Top എന്ന അറെ ഉപയോഗിച്ച് സ്റ്റാക്ക് പ്രവർത്തികമാക്കുകയാണെങ്കിൽ, TOS ന്റെ വില 0 മുതൽ (N-1) വരെ മാറി വരാം. അതുപോലെ സ്റ്റാക്കിലെ അംഗങ്ങളെ സൂചിപ്പിക്കാൻ STACK[0], STACK[1], STACK[2], ..., STACK[N-1] മുതലായ പദങ്ങൾ ഉപയോഗിക്കുന്നു.

3.2.2 സ്റ്റാക്കിലെ പ്രവർത്തനങ്ങൾ (Operations on stack)

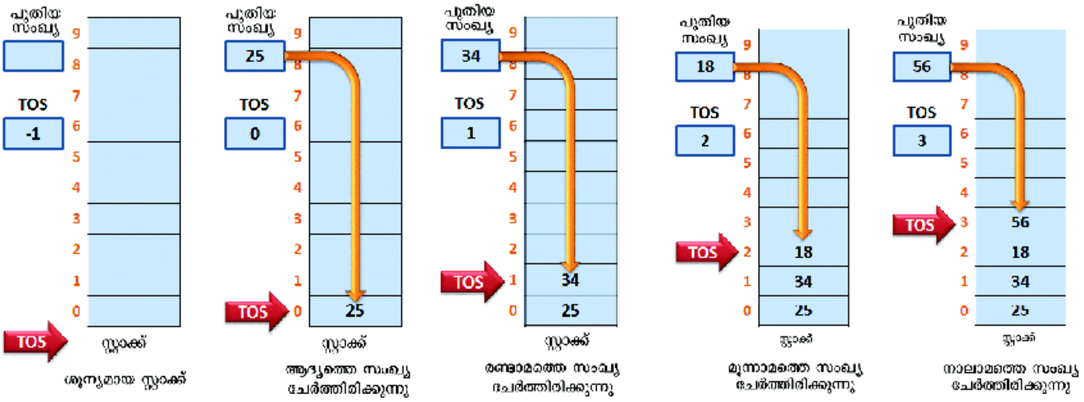
സ്റ്റാക്ക് ഡാറ്റ സ്ട്രക്ചറുകൾ അറെ ഉപയോഗിച്ചാണ് പ്രാവർത്തികമാക്കിയിരിക്കുന്നതെങ്കിലും അറെക്കു ബാധകമായിട്ടുള്ള എല്ലാ പ്രവർത്തനങ്ങളും സ്റ്റാക്കിൽ അതേ രൂപത്തിൽ ഉപയോഗിക്കുന്നില്ല. ഉദാഹരണത്തിന് അറെയിൽ ഏതു സ്ഥാനത്തു വേണമെങ്കിലും ഉൾപ്പെടുത്തൽ, നീക്കം ചെയ്യൽ എന്നീ പ്രവർത്തനങ്ങൾ ചെയ്യാവുന്നതാണ്. എന്നാൽ സ്റ്റാക്കിൽ മുകളിൽ നിന്ന് മാത്രമേ ഈ പ്രവർത്തനങ്ങൾ ചെയ്യാൻ സാധിക്കൂ. സ്റ്റാക്കിൽ നിർവഹിക്കുന്ന ഉൾപ്പെടുത്തൽ, നീക്കം ചെയ്യൽ പ്രവർത്തനങ്ങളെ യഥാക്രമം പുഷ് (***push***) എന്നും പോപ് (***pop***) എന്നും പറയുന്നു. ഈ പ്രവർത്തനങ്ങൾ എങ്ങനെയാണ് ചെയ്യുന്നതെന്നു നമുക്കു നോക്കാം.

a. പുഷ് പ്രവർത്തനം (Push operation)

സ്റ്റാക്കിലേക്കു ഒരു പുതിയ അംഗത്തെ ഉൾപ്പെടുത്തുന്ന പ്രക്രിയയെയാണ് പുഷ് പ്രവർത്തനം എന്ന് പറയുന്നത്. സ്റ്റാക്കിന്റെ നിർമാണം എന്ന് പറയുന്നത് പുഷ് പ്രവർത്തനത്തിന്റെ ആവർത്തിച്ചുള്ള പ്രയോഗം കൊണ്ടാണ് സാധ്യമാകുന്നത്.

നമുക്കു ചെയ്യാം

പുഷ് പ്രവർത്തനം നടപ്പിലാക്കുമ്പോഴുള്ള സ്റ്റാക്കിന്റെ സ്ഥിതിയാണ് ചിത്രം 3.5 ൽ കാണിച്ചിരിക്കുന്നത്. സ്റ്റാക്ക് പ്രാവർത്തികമാക്കുന്നതിനായി ഒരു അറെ നിർമിച്ചതായും TOS-1 ആയും നമുക്ക് അനുമാനിക്കാം. ചിത്രം നിരീക്ഷിച്ചു പ്രവർത്തനം നിർവഹിക്കാനുള്ള നടപടിക്രമങ്ങൾ എഴുതുക.



ചിത്രം 3.5: തുടർച്ചയായ പുഷ് പ്രവർത്തനത്തിന് ശേഷമുള്ള സ്റ്റാക്കിന്റെ അവസ്ഥ

ചിത്രം 3.5 തുടർച്ചയായ പുഷ് പ്രവർത്തനത്തിന് ശേഷമുള്ള സ്റ്റാക്കിന്റെ അവസ്ഥ സ്റ്റാക്കിന് മേലുള്ള പുഷ് പ്രവർത്തനത്തിനു വേണ്ടി താഴെ പറയുന്ന ഘട്ടങ്ങൾ നിർവചിക്കാൻ കഴിയുന്നുണ്ടോ എന്ന് പരിശോധിക്കുക.

ഘട്ടം 1: സ്റ്റാക്കിലേക്കു ഉൾപ്പെടുത്തുവാനുള്ള വില ഒരു വേരിയബിളിലേക്കു സ്വീകരിക്കുക.

ഘട്ടം 2: Top ന്റെ വില ഒന്ന് വച്ച് കൂട്ടുക.

ഘട്ടം 3: TOS ന്റെ സ്ഥാനത്തെ വില സംഭരിക്കുക.

സ്റ്റാക്കിന്റെ ശൂന്യമായ സ്ഥലം ഉണ്ടെങ്കിൽ മാത്രമേ പുതിയ അംഗങ്ങളെ ഉൾപ്പെടുത്താനുള്ള മുകളിൽ പറഞ്ഞിരിക്കുന്ന ഘട്ടങ്ങളുടെ പ്രവർത്തനങ്ങൾ സാധ്യമാവുകയുള്ളൂ. ചിത്രം 3.5 ൽ കാണിച്ചിരിക്കുന്ന സ്റ്റാക്കിൽ TOS ന്റെ വില 9 ആയിക്കഴിഞ്ഞാൽ പുതിയ അംഗത്തെ ഉൾപ്പെടുത്താൻ കഴിയില്ല. സ്റ്റാക്ക് നിറയുകയും പുതിയ ഒരു അംഗത്തെ ഉൾപ്പെടുത്താൻ നാം ശ്രമിക്കുകയും ചെയ്താൽ ഉടലെടുക്കുന്ന സാഹചര്യത്തെ (***stack overflow***) എന്ന് പറയുന്നു. സ്റ്റാക്കിലെ പുഷ് പ്രവർത്തനത്തിനുള്ള അൽഗോരിതം നമുക്ക് നോക്കാം.

സ്റ്റാക്കിന് മേൽ പുഷ് പ്രവർത്തനം ചെയ്യാനുള്ള അൽഗോരിതം

N പരമാവധി വലുപ്പമുള്ള STACK[N] എന്ന അറെ പരിഗണിക്കുക. സ്റ്റാക്കിന്റെ മുകൾഭാഗത്തിന്റെ സ്ഥാനം സൂചിപ്പിക്കുന്നതിനായി TOS എന്ന വേരിയബിളും ഉണ്ട്. VAL എന്ന വേരിയബിളിൽ ഡാറ്റ സൂക്ഷിച്ചിരിക്കുന്നു. VAL-നെ സ്റ്റാക്കിലേക്ക് ചേർക്കണം. എന്ന നിർദേശത്തിനും അവസാനിപ്പിക്കുന്ന നിർദേശത്തിനും ഇടയിലായി പുഷ് പ്രവർത്തനത്തിനാവശ്യമായ ഘട്ടങ്ങൾ നൽകിയിരിക്കുന്നു.

തുടങ്ങുക

1: അഥവാ (TOS < N-1) ആണെങ്കിൽ //സ്റ്റാക്ക് ശൂന്യമാണോ എന്ന് പരിശോധിക്കുന്നു

2: TOS = TOS + 1

3: STACK[TOS] = VAL

4: അല്ല എങ്കിൽ

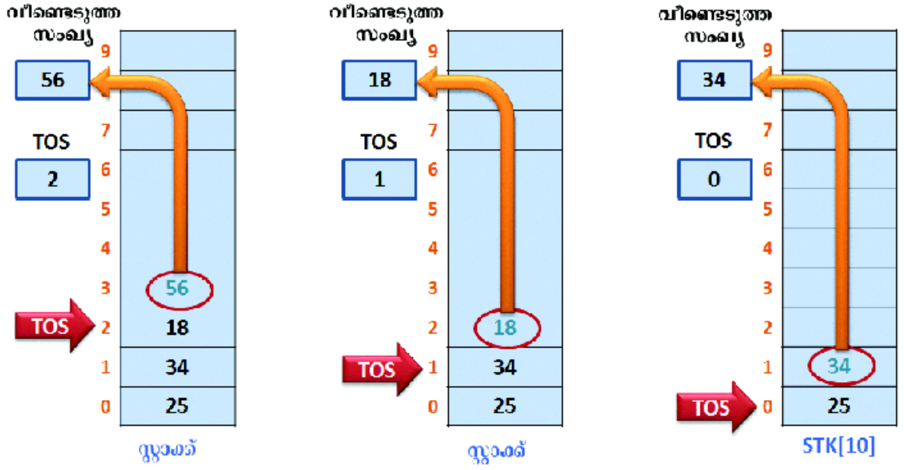
5: 'സ്റ്റാക്ക് ഓവർ ഫ്ളോ' എന്ന് പ്രിൻറ് ചെയ്യുക
6: പുറത്തേക്കു പോവുക
അവസാനിപ്പിക്കുക

a. പോപ് പ്രവർത്തനം (Pop operation)

ഒരു സ്റ്റാക്കിന്റെ ഏറ്റവും മുകളിലുള്ള അംഗത്തെ നീക്കം ചെയ്യുന്ന പ്രക്രിയയെയാണ് പോപ് പ്രവർത്തനം എന്ന് പറയുന്നത്. ഓരോ പോപ് പ്രവർത്തനത്തിന് ശേഷവും TOS ന്റെ വില ഒന്ന് വച്ച് കുറയുന്നു.

നമുക്കു ചെയ്യാം

ഒരു സ്റ്റാക്കിന്റെ ഏറ്റവും മുകളിലുള്ള അംഗത്തെ നീക്കം ചെയ്യാനുള്ള ഘട്ടങ്ങൾ നിർവചിക്കാൻ നമുക്ക് ശ്രമിക്കാം. ഇതിനായി ചിത്രം 3.6 തന്നിരിക്കുന്നു. ചിത്രം നിരീക്ഷിച്ചു പോപ് ഓപ്പറേഷന് വേണ്ടിയുള്ള ഘട്ടങ്ങൾ കണ്ടുപിടിച്ചു എഴുതുക.



അംഗത്തെ തിരിച്ചെടുത്തശേഷം TOS 2 ആയി മാറുന്നു | അംഗത്തെ തിരിച്ചെടുത്തശേഷം TOS 1 ആയി മാറുന്നു | അംഗത്തെ തിരിച്ചെടുത്തശേഷം TOS 0 ആയി മാറുന്നു

ചിത്രം 3.6: തുടർച്ചയായ പോപ് പ്രവർത്തനത്തിന് ശേഷമുള്ള സ്റ്റാക്കിന്റെ അവസ്ഥ

സ്റ്റാക്കിനു മേലുള്ള പോപ് പ്രവർത്തനത്തിനുവേണ്ടി താഴെ പറയുന്ന ഘട്ടങ്ങൾ നിർമിക്കാൻ കഴിയുന്നുണ്ടോ എന്ന് പരിശോധിക്കുക

ഘട്ടം 1: TOS ന്റെ സ്ഥാനത്തുള്ള അംഗത്തിന്റെ വില ഒരു വേരിയബിളിൽ സംഭരിക്കുക.
ഘട്ടം 2: TOS ന്റെ വില ഒന്ന് കുറയ്ക്കുക.

സ്റ്റാക്കിൽ അംഗങ്ങളുണ്ടെങ്കിൽ മുകളിൽ പറഞ്ഞ രണ്ടു ഘട്ടങ്ങൾ പ്രവർത്തിക്കുന്നു. അറെ ഉപയോഗിച്ചുള്ള സ്റ്റാക്കിൽ, പോപ് പ്രവർത്തനം നടത്തുമ്പോൾ യഥാർഥത്തിൽ അംഗങ്ങളെ നീക്കം ചെയ്യുന്നില്ല. മറിച്ച് TOS ന്റെ വില കുറച്ചുകൊണ്ട് അത്തരം അംഗങ്ങളെ ഉപയോഗിക്കുന്നതിൽ നിന്നും തടസ്സപ്പെടുത്തുന്നു. ഏറ്റവും അവസാനത്തെ അംഗം നീക്കം ചെയ്യപ്പെടുന്നതുവരെ ചിത്രം 3.6 ൽ കൊടുത്തിരിക്കുന്ന സ്റ്റാക്കിൽ പോപ് പ്രവർത്തനം ചെയ്യാവുന്നതാണ്. ഏറ്റവും അവസാനത്തെ അംഗം നീക്കം ചെയ്യപ്പെട്ടു കഴിഞ്ഞാൽ TOS

ന്റെ വില -1 ആയി മാറുന്നു. ഇപ്പോൾ സ്റ്റാക്ക് ശൂന്യമാണ്. ഇങ്ങനെ ശൂന്യമായ സ്റ്റാക്കിൽ നിന്നും ഒരു അംഗത്തെ നീക്കം ചെയ്യാൻ ശ്രമിക്കുകയാണെങ്കിൽ ഉടലെടുക്കുന്ന പ്രതിസന്ധിയെ സ്റ്റാക്ക് അണ്ടർ ഫ്ളോ (***stack underflow***) എന്ന് പറയുന്നു. ഇനി നമുക്ക് സ്റ്റാക്കിന് മേലുള്ള പോപ് പ്രവർത്തനത്തിന്റെ അൽഗോരിതം എഴുതി നോക്കാം.

സ്റ്റാക്കിന് മേലുള്ള പോപ് പ്രവർത്തനത്തിനുള്ള അൽഗോരിതം

പരമാവധി N അംഗങ്ങളെ സംഭരിക്കാവുന്ന ഒരു സ്റ്റാക്കിനെ പ്രാവർത്തികമാക്കാൻ വേണ്ടി STACK[N] എന്ന ഒരു അറെ പരിഗണിക്കുക. സ്റ്റാക്കിലെ ഏറ്റവും മുകളിലെ സ്ഥാനത്തിന്റെ വിവരം സൂക്ഷിക്കുന്നതിനായി TOS എന്നൊരു വേരിയബിൾ ഉപയോഗിക്കുന്നു. സ്റ്റാക്കിൽ നിന്നും നീക്കം ചെയ്യുന്ന അംഗങ്ങളെ സൂക്ഷിക്കുന്നതിനായി VAL എന്ന വേരിയബിൾ ഉപയോഗിക്കുന്നു. പോപ് പ്രവർത്തനത്തിനുള്ള നിർദേശങ്ങൾ തുടങ്ങുക, അവസാനിക്കുക എന്നിവയ്ക്കിടയിൽ കൊടുത്തിരിക്കുന്നു.

തുടങ്ങുക

1: അഥവാ (TOS > -1) ആണെങ്കിൽ //സ്റ്റാക്ക് ശൂന്യമാണോ എന്ന് പരിശോധിക്കുന്നു (അണ്ടർ ഫ്ളോ)

2: VAL = STACK[TOS]

3: TOS = TOS - 1

4: അല്ല എങ്കിൽ

5: 'സ്റ്റാക് അണ്ടർ ഫ്ളോ' എന്ന് പ്രിന്റ് ചെയ്യുക

6: പുറത്തേക്കു പോകുക

അവസാനിപ്പിക്കുക

സ്റ്റാക്ക് പ്രവർത്തങ്ങൾക്ക് വേണ്ടിയുള്ള C++ ഫങ്ഷനുകൾ

tos, n എന്നിവ ഗ്ലോബൽ വേരിയബിൾ ആണെന്ന് കരുതുക

പുഷ് പ്രവർത്തനം

```
void push(int stack[],int val)
{
    if (tos < n-1)
    {
        tos++;
        stack[tos]=val;
    }
    else
        cout<<"Overflow";
}
```

പോപ്പ് പ്രവർത്തനം

```
int pop(int stack[])
{
    int val;
    if (tos > -1)
    {
        val=stack[tos];
        tos--;
    }
    else
        cout<<"Underflow";
    return val;
}
```

TOS, N എന്നീ വേരിയബിളുകൾ ഗ്ലോബൽ വേരിയബിളുകളായി പരിഗണിച്ചിരിക്കുന്നു.

സ്റ്റാക്കിന്റെ ഉപയോഗം

സ്റ്റാക്കുകൾ LIFO തത്വത്തെ പിന്തുടരുന്നതിനാൽ സ്ട്രിങ്ങിനെ തിരിച്ചെഴുതുവാനും, പോളിഷ് സ്ട്രിങ് നിർമിക്കാനും മറ്റും ഉപയോഗിക്കുന്നു. സ്ട്രിങ്ങിനെ തിരിച്ചെഴുതുക എന്ന് പറഞ്ഞാൽ തന്നിരിക്കുന്ന സ്ട്രിങ്ങിലെ അക്ഷരങ്ങളെ അവസാനത്തേതിൽ നിന്നും ആദ്യത്തേതിലേക്ക് എന്ന ക്രമത്തിൽ ഒരു സ്ട്രിങ് രൂപപ്പെടുത്തുക എന്നാണ്. ഉദാഹരണത്തിന് "SAD" എന്ന സ്ട്രിങ്ങിനെ തിരിച്ചെഴുതുകയാണെങ്കിൽ "DAS" എന്ന് ലഭിക്കുന്നു. പോളിഷ് സ്ട്രിങ് എന്നാൽ ഓപ്പറന്റുകൾക്കു മുമ്പോ ശേഷമോ ഓപ്പറേറ്ററുകൾ വരുന്ന തരത്തിലുള്ള ഒരു ഗണിത പ്രയോഗശൈലിയാണ്. ഉദാഹരണത്തിന് A+B എന്നത് AB+ എന്നോ +AB എന്നോ ആക്കി മാറ്റാവുന്നതാണ്. A+B എന്നതിനെ ഇൻഫിക്സ് (infix) പ്രയോഗശൈലി എന്ന് പറയുന്നു. AB+, +AB എന്നിവ യഥാക്രമം പോസ്റ്റ്ഫിക്സ് (postfix) എന്നും പ്രീഫിക്സ് (prefix) പ്രയോഗശൈലി എന്നും അറിയപ്പെടുന്നു. ഇവ ALU വിന്റെ പ്രവർത്തനത്തിന് ആവശ്യമാണ്. ഇൻഫിക്സിനെ മറ്റു രണ്ടു രൂപത്തിലേക്ക് വിവർത്തനം ചെയ്യാൻ പുഷ്, പോപ്പ് പ്രവർത്തനങ്ങളാണ് കമ്പ്യൂട്ടർ ഉപയോഗിക്കുന്നത്. മാത്രമല്ല ഈ പ്രയോഗശൈലികളെ വിലയിരുത്താനും ALU സ്റ്റാക്കുകളെ ആശ്രയിക്കുന്നു.

3.3 ക്യൂ (Queue)

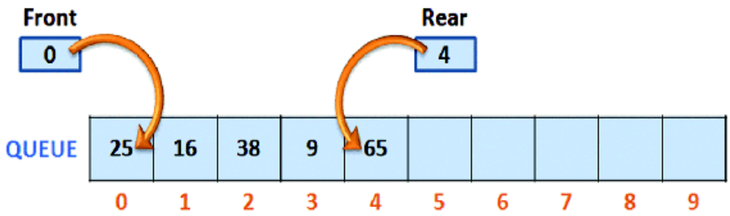
വരികൾ നമുക്ക് സുപരിചിതങ്ങളാണ്. പല സാഹചര്യങ്ങളിലും നമ്മൾ വരികളുടെ ഭാഗമാകാറുമുണ്ട്. ചിത്രം 3.7 ൽ പോളിങ് സ്റ്റേഷനിലെ വരിയാണ് സൂചിപ്പിച്ചിരിക്കുന്നത്. ഇവിടെ വരിയുടെ മുമ്പിലുള്ള ആദ്യത്തെ വോട്ടർ വോട്ട് ചെയ്യുന്നു. പുതിയ വ്യക്തി വരിയിൽ ചേരുമ്പോൾ ഏറ്റവും പുറകിലായിട്ടാണ് വന്നു നിൽക്കുക. ഈ ഉദാഹരണത്തിൽ നിന്നും വ്യക്തമാകുന്നത്, ആദ്യം വരിയിൽ നിൽക്കുന്ന വ്യക്തിയാണ് വരിയിൽ നിന്നും ആദ്യം പുറത്തു പോവുക. ഈ തരത്തിൽ ഒരു കൂട്ടം ഇനങ്ങളെ ക്രമീകരിക്കുന്നതിനെ ഫസ്റ്റ് ഇൻ ഫസ്റ്റ് ഔട്ട് (FIFO) തത്വം എന്ന് പറയുന്നു. അതിനാൽ FIFO തത്വത്തിന്റെ അടിസ്ഥാനത്തിൽ പ്രവർത്തിക്കുന്ന ഡാറ്റ സ്ട്രക്ചറിനെ ക്യൂ (Queue) എന്ന് വിളിക്കുന്നു. ചിത്രത്തിൽ കണ്ടതുപോലെ ഒരു ക്യൂവിനു രണ്ട് അറ്റങ്ങൾ ഉണ്ട് : മുൻഭാഗവും (Front) പിൻഭാഗവും (Rear). ക്യൂവിൽ പുതിയ ഡാറ്റ അംഗങ്ങളെ ഉൾപ്പെടുത്തുന്നത് പിൻഭാഗത്തും നീക്കം ചെയ്യുന്നത് മുൻഭാഗത്തും ആയിരിക്കും. സ്റ്റാക്കിൽ പറഞ്ഞത് പോലെ ക്യൂവും ഒരു യുക്തിപരമായ ആശയം മാത്രമാണ്. ഒരു അറെ ഉപയോഗിച്ചാണ്ക്യൂ പ്രാവർത്തികമാക്കുന്നതെങ്കിൽ അത്തരത്തിലുള്ള ക്യൂ (സ്റ്റാറ്റിക്) സ്വഭാവമുള്ളതായിരിക്കും.

ചിത്രം 3.7: പോളിങ് സ്റ്റേഷനിലെ വരി

3.3.1 ക്യൂവിനെ പ്രവർത്തികമാക്കൽ (Implementation of Queue)

ഒരു അറെ ഉപയോഗിച്ച് ക്യൂ പ്രായോഗികമാക്കുകയാണെങ്കിൽ, അതിൽ ഉൾക്കൊള്ളിക്കാവുന്ന അംഗങ്ങൾക്ക് ഒരു പരിധിയുണ്ട്. Queue[10] എന്ന അറെ ഉപയോഗിച്ച് നടപ്പിലാക്കിയിരിക്കുന്ന ഒരു ക്യൂ ആണ് ചിത്രം 3.8 ൽ കാണുന്നത്. ഇതിന് പരമാവധി

10 അംഗങ്ങളെ മാത്രമേ ഉൾക്കൊള്ളാനാകൂ. ഈ ചിത്രത്തിൽ പ്രകാരം 0 മുതൽ 4 വരെ സ്ഥാനങ്ങളിലായി അഞ്ച് അംഗങ്ങളാണ് ക്യൂവിലുള്ളത്. അതായത് QUEUE 0 വും Rear 4 ഉം ആയിരിക്കും. ക്യൂവിലെ ആദ്യത്തെ അംഗത്തെ QUEUE[Front] എന്നും അവസാനത്തെ അംഗത്തെ QUEUE[Rear] എന്നും സൂചിപ്പിക്കാവുന്നതാണ്.



ചിത്രം 3.8: അറെ ഉപയോഗിച്ച് പ്രാവർത്തികമാക്കിയ ക്യൂ

അറെയുടെ അവസാന സൂചികയായ 9 ആണ് ക്യൂവിന് അനുവദനീയമായിട്ടുള്ള ഏറ്റവും ഉയർന്ന വില. തുടക്കത്തിൽ മുൻഭാഗത്തിന്റെയും (Rear) പിൻഭാഗത്തിന്റെയും (Front) വില -1 ആയിരിക്കും. ഇത് ക്യൂ ശൂന്യമാണെന്നതിനെ സൂചിപ്പിക്കുന്നു. ക്യൂ വിലേക്കു ആദ്യത്തെ അംഗത്തെ ഉൾപ്പെടുത്തുമ്പോൾ ഈ രണ്ടു വിലകളും 0 ആയി മാറുന്നു. തുടർന്ന് ഓരോ പുതിയ അംഗത്തെ ഉൾപ്പെടുത്തുമ്പോഴും Rear ന്റെ വില ഒന്ന് വച്ചു കൂടുന്നു. ഏറ്റവും ഉയർന്ന സൂചിക (ഇവിടെ 9) എത്തുന്നത് വരെ ഇത് തുടർന്ന് പോകാവുന്നതാണ്. അതുപോലെ ഓരോ അംഗത്തെ നീക്കം ചെയ്യുമ്പോഴും Front ന്റെ വില ഒന്നുവച്ച് കൂടുന്നതാണ്. Front ന്റെ വില Rear നെക്കാൾ കൂടുതലാകുന്നതു വരെ ഇത് തുടരാവുന്നതാണ്.

3.3.2 ക്യൂവിലെ പ്രവർത്തനങ്ങൾ (Operations on queue)

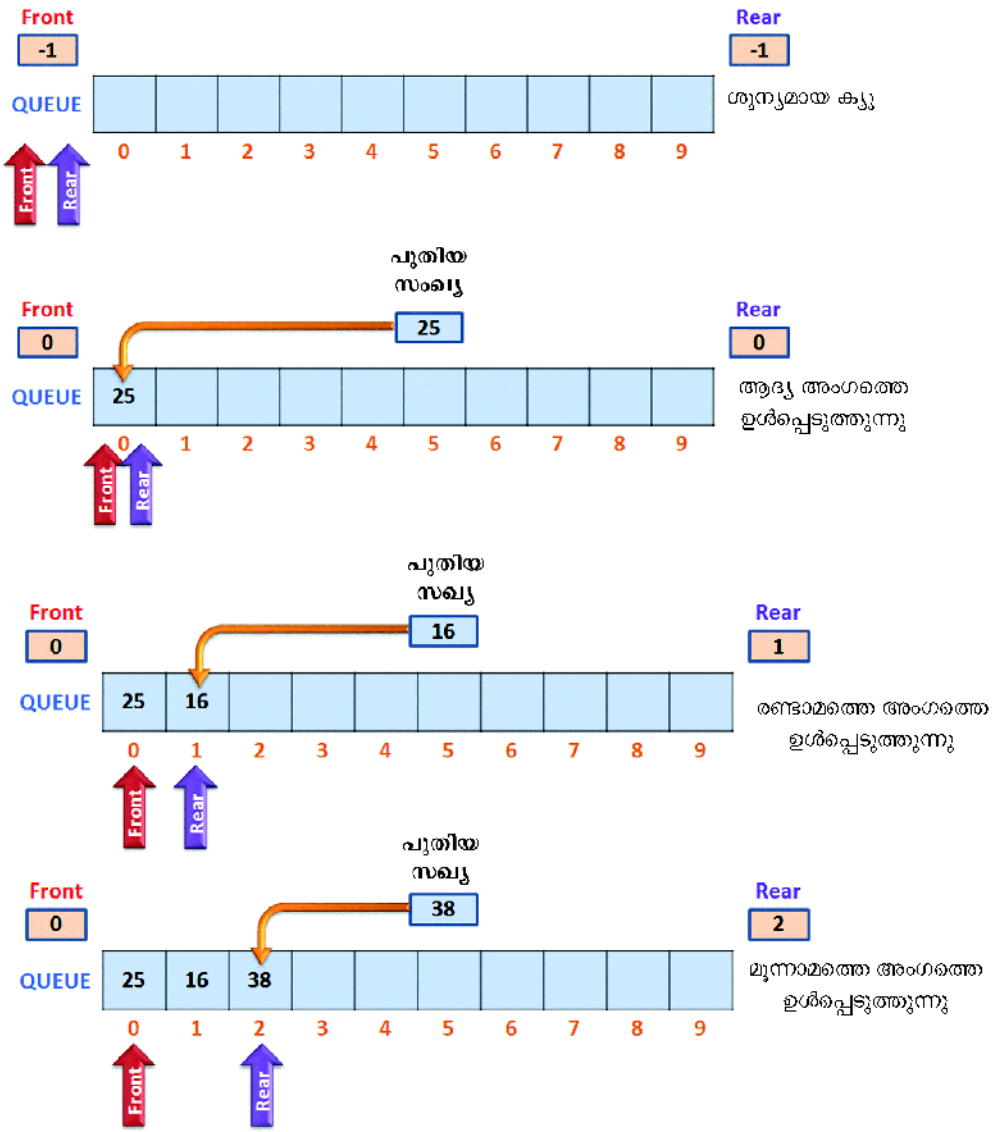
സ്റ്റാക്കിലേതു പോലെ ക്യൂവിലും ഉൾപ്പെടുത്തൽ, നീക്കം ചെയ്യൽ പ്രവർത്തനങ്ങൾക്കു ചില നിയന്ത്രണങ്ങളുണ്ട്. ഒരു സാധാരണ അറെയിൽ ഉൾപ്പെടുത്തൽ, നീക്കം ചെയ്യൽ എന്നീ പ്രവർത്തനങ്ങൾ ഏതു സ്ഥാനത്തു വേണമെങ്കിലും ചെയ്യാവുന്നതാണ്. എന്നാൽ ഒരു സ്റ്റാക്കിൽ ഇത് ഏറ്റവും മുകളിൽ നിന്ന് മാത്രമായിരിക്കും. അതുപോലെ ക്യൂവിൽ ഉൾപ്പെടുത്തൽ, നീക്കം ചെയ്യൽ പ്രവർത്തനങ്ങൾ രണ്ട് അറ്റത്തു മാത്രമായിരിക്കും നടക്കുക.

a. ഉൾപ്പെടുത്തൽ പ്രവർത്തനം (Insertion operation)

ക്യൂവിന്റെ പിൻഭാഗത്തു പുതിയ അംഗത്തെ കൂട്ടിച്ചേർക്കുന്ന പ്രക്രിയയെയാണ് ഉൾപ്പെടുത്തൽ എന്ന് പറയുന്നത്. അടുത്ത മെമ്മറി സ്ഥാനത്തെ സൂചിപ്പിക്കാൻ Rear ന്റെ വില ആദ്യം കൂട്ടുകയും പിന്നീട് ആ സ്ഥാനത്ത് അംഗത്തെ ഉൾപ്പെടുത്തുകയും ചെയ്യുന്നു. ഉൾപ്പെടുത്തൽ പ്രവർത്തനം തുടർച്ചയായി ചെയ്യുമ്പോഴാണ് ഒരു ക്യൂ രൂപപ്പെടുന്നത്.

നമുക്കു ചെയ്യാം

ഉൾപ്പെടുത്തൽ പ്രവർത്തനത്തിന് വിധേയമാകുന്ന ഒരു ക്യൂവിന്റെ അവസ്ഥ ചിത്രം 3.9 ൽ കാണിച്ചിരിക്കുന്നു.
ഇതിനായി ഒരു അറെ നിർമ്മിക്കുകയും, Front, Rear എന്നിവയ്ക്ക് -1 എന്ന വില നൽകുകയും ചെയ്യുന്നു ചിത്രം നിരീക്ഷിച്ചു പ്രവർത്തനം ചെയ്യുവാനുള്ള നടപടിക്രമം എഴുതുക.



ചിത്രം 3.9: തുടർച്ചയായ ഉൾപ്പെടുത്തൽ പ്രവർത്തനം നടത്തിയതിന് ശേഷമുള്ള ക്യൂവിന്റെ അവസ്ഥ

ഉൾപ്പെടുത്തൽ പ്രവർത്തനത്തിന് വേണ്ടി താഴെ പറയുന്ന ഘട്ടങ്ങൾ നിർവചിക്കുവാൻ കഴിയുമോ എന്ന് പരിശോധിക്കുക.

ഘട്ടം 1: ക്യൂവിലേക്കു ഉൾപ്പെടുത്തുവാനുള്ള വില ഒരു വേരിയബിളിലേക്കു സ്വീകരിക്കുക

ഘട്ടം 2: Rear ന്റെ വില ഒന്ന് കൂട്ടുക.

ഘട്ടം 3: Rear ന്റെ സ്ഥാനത്ത് വില സംഭരിക്കുക.

ശൂന്യമായ ഒരു ക്യൂവിലേക്കു ആദ്യമായി ഉൾപ്പെടുത്തൽ പ്രവർത്തനം ചെയ്യുമ്പോൾ Front, Rear എന്നിവയുടെ വില ഒന്നുവച്ചു കൂട്ടുന്നു. അതായത് Rear ഉം Front ഉം 0 ആയി മാറുന്നു. അതിനു ശേഷം വരുന്ന ഉൾപ്പെടുത്തലുകൾക്കെല്ലാം Rear മാത്രം കൂട്ടുന്നു. Rear 9 ആകുന്നതു വരെ (അറേ യുടെ അവസാനത്തെ സൂചിക) ഇത് തുടരുന്നതാണ്. അതിനു ശേഷം വീണ്ടും ഉൾപ്പെടുത്താൻ പ്രവർത്തനത്തിനു ശ്രമിക്കുകയാണെങ്കിൽ സ്റ്റാക്കിലേതു പോലെ 'ക്യൂ ഓവർ ഫ്ളോ' (*queue overflow*) സംഭവിക്കുന്നു. ക്യൂവിൽ ഉൾപ്പെടുത്തൽ പ്രവർത്തനം നടത്തുവാനുള്ള അൽഗോരിതം നമുക്ക് എഴുതി നോക്കാം.

ക്യൂവിൽ ഉൾപ്പെടുത്തൽ പ്രവർത്തനം ചെയ്യുവാനുള്ള അൽഗോരിതം

ക്യൂ നടപ്പിലാക്കുന്നതിന് വേണ്ടി, N വലുപ്പമുള്ള QUEUE[N] എന്ന ഒരു അറേ പരിഗണിക്കുക. ക്യൂവിന്റെ മുൻഭാഗത്തെയും പിൻഭാഗത്തെയും സൂചിപ്പിക്കുന്നതിനായി FRONT, REAR എന്നീ രണ്ടു വേരിയബിളുകൾ ഉപയോഗിക്കുന്നു. ക്യൂവിലേക്കു ഉൾപ്പെടുത്താനുള്ള ഡാറ്റ VAL എന്ന വേരിയബിളിൽ സംഭരിച്ചു വയ്ക്കുന്നു. തുടങ്ങുക, അവസാനിപ്പിക്കുക എന്നീ നിർദേശങ്ങൾക്കിടയിൽ കൊടുത്തിരിക്കുന്ന ഘട്ടങ്ങൾ ഉൾപ്പെടുത്തൽ പ്രവർത്തനത്തെ വിശദീകരിക്കുന്നു.

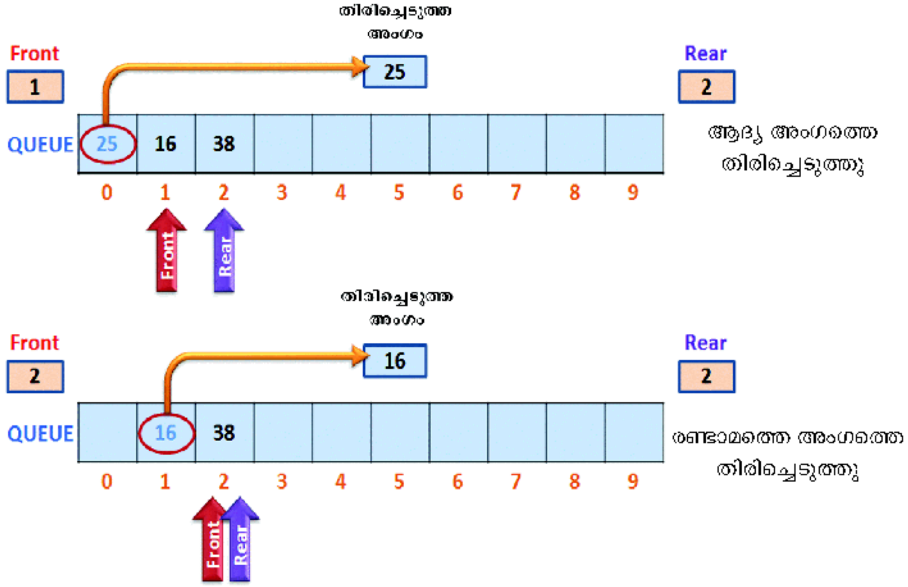
```
തുടങ്ങുക
    1:  അഥവാ (REAR == −1) ആണെങ്കിൽ //സ്ഥലത്തിന്റെ ലഭ്യത പരിശോധിക്കുന്നു
    2:        FRONT = REAR = 0 'ക്യൂ ഓവർ ഫ്ളോ' പ്രിന്റ് ചെയ്യുക
    3:        Q[REAR] = VAL പുറത്തേക്കു പോകുക
    4:  അല്ലെങ്കിൽ അഥവാ (REAR < N-1) ആണെങ്കിൽ // സ്ഥലത്തിന്റെ ലഭ്യത പരി
                                                 ശോധിക്കുന്നു.
    5:        REAR = REAR + 1
    6:        Q[REAR] = VAL
    7:  അല്ലെങ്കിൽ
    8:        Print "Queue Overflow "
    9:  പുറത്തേക്കു പോകുക
        അവസാനിപ്പിക്കുക
```

b. നീക്കം ചെയ്യൽ പ്രവർത്തനം (Deletion operation)

ക്യൂ വിന്റെ മുൻഭാഗത്തു നിന്നും അംഗത്തെ ഒഴിവാക്കുന്ന പ്രക്രിയയെയാണ് നീക്കം ചെയ്യൽ എന്ന് പറയുന്നത്. നീക്കം ചെയ്യലിന് ശേഷം Front ന്റെ വില ഒന്നുവച്ചു കൂട്ടുന്നു. അറെ ഉപയോഗിച്ചുള്ള ക്യൂവിൽ നീക്കം ചെയ്യൽ പ്രവർത്തനം നടത്തുമ്പോൾ യഥാർഥത്തിൽ അംഗങ്ങളെ നീക്കം ചെയ്യുന്നില്ല. മറിച്ച് Front ന്റെ വില കൂട്ടിക്കൊണ്ട് അത്തരം അംഗങ്ങളെ ഉപയോഗിക്കുന്നതിൽ നിന്നും തടസ്സപ്പെടുത്തുന്നു.

നമുക്കു ചെയ്യാം

അറെ ഉപയോഗിച്ച് നടപ്പിലാക്കിയിരിക്കുന്ന ഒരു ക്യൂ പരിഗണിക്കുക. ചിത്രം 3.10 ൽ നീക്കം ചെയ്യൽ പ്രവർത്തനത്തിന് വിധേയമാകുന്ന ക്യൂവിന്റെ അവസ്ഥകൾ കാണിച്ചിരിക്കുന്നു. ഈ പ്രവർത്തനത്തിനുള്ള വിവിധ ഘട്ടങ്ങൾ നിർവചിക്കാൻ നമുക്കു ശ്രമിക്കാം.



ചിത്രം 3.10: തുടർച്ചയായ നീക്കം ചെയ്യൽ പ്രവർത്തനം നടത്തിയതിന് ശേഷമുള്ള ക്യൂവിന്റെ അവസ്ഥ

നീക്കം ചെയ്യൽ പ്രവർത്തനത്തിന് താഴെ പറയുന്ന ഘട്ടങ്ങൾ നമുക്ക് നിർവചിക്കാം.

Step 1: Front ന്റെ സ്ഥാനത്തുള്ള അംഗത്തെ ഒരു വേരിയബിളിലേക്കു സ്വീകരിക്കുക.

Step 2: Front ന്റെ വില ഒന്ന് കൂട്ടുക.

ചിത്രം 3.10 പ്രകാരം ക്യൂവിന്റെ മുൻഭാഗത്തേക്ക് അംഗങ്ങളുടെ സ്ഥാനമാറ്റം നടക്കുന്നില്ല. അതായത് നിത്യജീവിതത്തിലെപ്പോലെയല്ല ഇവിടുത്തെ ക്യൂവിന്റെ സങ്കൽപ്പം. ക്യൂ ഡാറ്റ സ്ട്രക്ചറിൽ നീക്കം ചെയ്യൽ പ്രവർത്തത്തിൽ അംഗങ്ങളുടെ സ്ഥാനമാറ്റം നടത്തുന്നതിന് പകരം Front ന്റെ വില കൂട്ടുകയാണ് ചെയ്യുന്നത്. ക്യൂവിൽ അംഗങ്ങൾ ഉള്ളിടത്തോളം കാലം നീക്കം ചെയ്യൽ പ്രവർത്തനത്തിന് മേൽനിർവചിച്ച രണ്ടു ഘട്ടങ്ങൾ മതിയാകുന്നതാണ്. രണ്ടാമത്തെ അംഗത്തിന്റെ നീക്കം ചെയ്യലിന് ശേഷമുള്ള ക്യൂവിന്റെ അവസ്ഥ നോക്കുക. Front ന്റെയും Rear ന്റെയും വില ഒരേ അംഗത്തെ സൂചിപ്പിക്കുന്നു. അതായത് രണ്ട് എന്ന അറയുടെ സൂചിക. ക്യൂവിൽ ഒരു നീക്കം ചെയ്യൽ പ്രവർത്തനം കൂടി നടത്തുന്നതായി അനുമാനിക്കുക. നടപടി ക്രമം പ്രകാരം Front ന്റെ വില 3 ആയി മാറുന്നു. ഇത് Rear ന്റെ വിലയേക്കാൾ കൂടുതലാണ്. നമുക്കറിയാം ഒരു ക്യൂവിൽ അത് ഉചിതമല്ല. മാത്രമല്ല ക്യൂ ഇപ്പോൾ ശൂന്യമാണെന്നു നമുക്ക് കാണാവുന്നതാണ്. ക്യൂ ശൂന്യമായാൽ Front ന്റെയും Rear ന്റെയും വില-1 ആകണമെന്ന് നേരത്തെ പരാമർശിച്ചതാണ്. അപ്പോൾ അവസാനത്തെ അംഗത്തെ നീക്കം ചെയ്താൽ Front ന്റെയും Rear ന്റെയും വില -1 ആക്കാനുള്ള ഘട്ടം കൂടി അൽഗോരിതത്തിൽ ഉൾപ്പെടുത്തേണ്ടതാണ്. ഈ അവസ്ഥയിൽ വീണ്ടും നീക്കം ചെയ്യൽ അനുവദനീയമല്ല. ശൂന്യമായ ക്യൂ വിൽ നിന്നും നീക്കം ചെയ്യാൻ ശ്രമിക്കുമ്പോളുണ്ടാകുന്ന അവസ്ഥയെ 'ക്യൂ അണ്ടർഫ്ളോ' (***queue underflow***) എന്ന് പറയുന്നു. ഇനി നമുക്ക് ക്യൂവിൽ നിന്നും നീക്കം ചെയ്യൽ പ്രവർത്തനം നടത്താനുള്ള അൽഗോരിതം തയാറാക്കാം.

ഒരു ക്യൂവിൽ നിന്ന് നീക്കം ചെയ്യൽ പ്രവർത്തനം നടത്താനുള്ള അൽഗോരിതം

ക്യൂ നടപ്പിലാക്കാൻ വേണ്ടി പരമാവധി N അംഗങ്ങളുള്ള QUEUE[N] എന്ന ഒരു അറെ പരിഗണിക്കുക. FRONT, REAR എന്നീ വേരിയബിളുകൾ ക്യൂവിന്റെ മുൻഭാഗത്തെയും പിൻഭാഗത്തെയും സൂചിപ്പിക്കാൻ ഉപയോഗിക്കുന്നു. ക്യൂവിൽ നിന്നും നീക്കം ചെയ്യുന്ന അംഗത്തെ സംഭരിക്കുന്നതിനായി VAL എന്ന വേരിയബിൾ ഉപയോഗിക്കുന്നു. നീക്കം ചെയ്യൽ പ്രവർത്തനത്തിനുള്ള ഘട്ടങ്ങൾ തുടങ്ങുക, അവസാനിപ്പിക്കുക എന്നീ നിർദേശങ്ങൾക്കിടയിൽ നൽകിയിരിക്കുന്നു.

തുടങ്ങുക

1: അഥവാ (FRONT > −1) ആണെങ്കിൽ // ശൂന്യമായ അവസ്ഥ പരിശോധിക്കുന്നു
2: VAL = Q[FRONT]
3: FRONT = FRONT + 1
4: അല്ലെങ്കിൽ
5: 'ക്യൂ ഓവർ ഫ്ളോ' എന്നു പ്രിൻറ് ചെയ്യുക
6: പുറത്തേക്കു പോകുക
7: അഥവാ (FRONT > REAR) ആണെങ്കിൽ // അവസാനത്തെ അംഗത്തിന്റെ നീക്കം ചെയ്യൽ പരിശോധിക്കുന്നു
8: FRONT = REAR = -1
9: പുറത്തേക്കു പോകുക

അവസാനിപ്പിക്കുക

ക്യൂ പ്രവർത്തങ്ങൾക്ക് വേണ്ടിയുള്ള C++ ഫങ്ഷനുകൾ

n, front, rear എന്നിവ ഗ്ലോബൽ വേരിയബിൾ ആണെന്ന് കരുതുക

ക്യൂ വിൽ ഉൾപ്പെടുത്തൽ പ്രവർത്തനം

```
void ins_q(int queue[],int val)
{
    if (rear == -1)
    {
        front=0;
        rear=0;
        q[rear]=val;
    {
    else (if rear < n-1)
    {
        rear++;
        q[rear]=val;
    }
    else
        cout<<"Overflow";
}
```

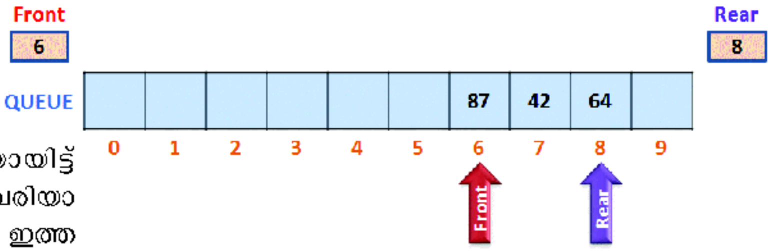
ക്യൂവിൽ നീക്കം ചെയ്യൽ പ്രവർത്തനം

```
int del_q(int queue[])
{
    int val;
    if (front > -1)
    {
        val=q[front];
        front++;
    }
    else
        cout<<"Underflow";
    if (front > rear)
    {
        front= -1;
        rear= -1
    }
    return val;
}
```

ക്യൂവിന്റെ ഉപയോഗം

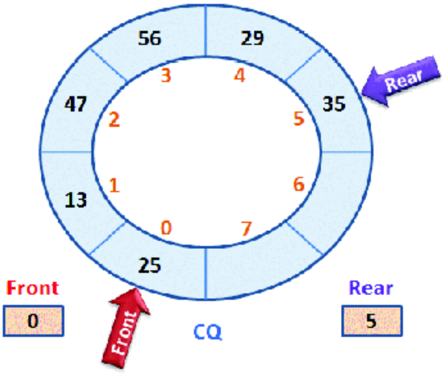
ക്യൂ അധികവും കമ്പ്യൂട്ടർ സയൻസിൽ ജോബ് ആസൂത്രണം ചെയ്യുവാൻ വേണ്ടിയാണ് ഉപയോഗിക്കുന്നത്. മെമ്മറി, പ്രോസസ്സർ, ഫയലുകൾ മുതലായവ ആസൂത്രണം ചെയ്യുന്നതിനാണ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റം ക്യൂ ഉപയോഗിക്കുന്നത്. ഇതിന് ഉദാഹരണമാണ് പ്രിൻറ് ക്യൂ. പ്രോസെസ്സറിനെ അപേക്ഷിച്ചു പ്രിന്ററിന് വേഗത കുറവായതിനാൽ പ്രിൻറ് ചെയ്യുവാനുള്ള ജോലികളെല്ലാം പ്രിൻറ് ബഫറിൽ നിക്ഷേപിക്കുന്നു. പ്രിന്റ് ബഫർ FIFO തത്വം പിന്തുടരുന്നതിനാൽ അതിനെ ഒരു ക്യൂ ആയി കണക്കാക്കാം.

3.3.3 വൃത്താകൃതിയിലുള്ള ക്യൂ (Circular queue)



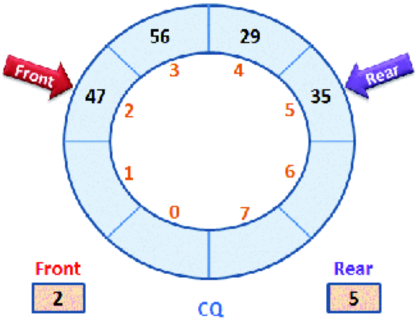
ചിത്രം 3.11: തുടർച്ചയായ 6 അംഗങ്ങളെ നീക്കം ചെയ്തതിനു ശേഷം 3 അംഗങ്ങളുള്ള ക്യൂ

നമ്മൾ ഇതുവരെ ചർച്ച ചെയ്ത ക്യൂവിനെ രേഖീയ ക്യൂ എന്നാണ് പറയുക. ഇതിലെ അംഗങ്ങൾ ഒരു നേർരേഖയായിട്ട് അല്ലെങ്കിൽ ഒരു നേർവരിയായിട്ടാണ് കാണപ്പെടുക. ഇത്തരം ക്യൂകളുടെ രണ്ടറ്റങ്ങൾ തമ്മിൽ ഒരിക്കലും കൂട്ടിമുട്ടുകയില്ല. രേഖീയ ക്യൂവിൽ ഒരു ന്യൂനതയുണ്ട്. ചിത്രം 3.11 ൽ കൊടുത്തിരിക്കുന്ന, 6 അംഗങ്ങളെ നീക്കം ചെയ്യൽ പ്രവർത്തനങ്ങൾക്കു വിധേയമായ ക്യൂ പരിഗണിക്കുക. നിലവിൽ അതിൽ മൂന്ന് അംഗങ്ങൾ മാത്രമേയുള്ളൂ. സ്വാഭാവികമായിട്ടും Front ന്റെ വില 6 ഉം Rear ന്റെ വില 8 ഉം ആയിരിക്കും.



ചിത്രം 3.12: വൃത്താകൃതിയിലുള്ള ക്യൂ

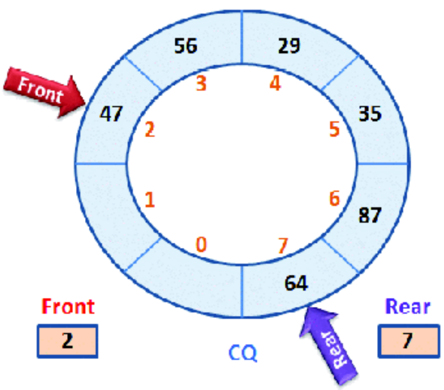
ആദ്യത്തെ 6 സ്ഥാനങ്ങൾ ശൂന്യമാണെങ്കിലും ഉൾപ്പെടുത്തൽ അൽഗോരിതപ്രകാരം നമുക്ക് ഒരു അംഗത്തെ മാത്രമേ കൂട്ടിച്ചേർക്കാൻ കഴിയൂ. അവസാനത്തെ സ്ഥാനത്ത് ഒരു അംഗം മാത്രമേ ഉള്ളൂ എന്ന് കരുതുക. അങ്ങനെയെങ്കിൽ പുതിയതായി ഒരു അംഗത്തെ കൂട്ടിച്ചേർക്കുവാൻ ശ്രമിക്കുകയാണെങ്കിൽ 'ക്യൂ ഓവർ ഫ്ളോ' (Queue Overflow) സംഭവിക്കുന്നു. രേഖീയ ക്യൂവിന്റെ ഈ പരിമിതി വൃത്താകൃതി ക്യൂ ഉപയോഗിച്ച് പരിഹരിക്കാവുന്നതാണ്. ചിത്രം 3.12 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ രണ്ടറ്റങ്ങളും കൂട്ടി മുട്ടുന്ന ഒരു ക്യൂ ആണിത്.



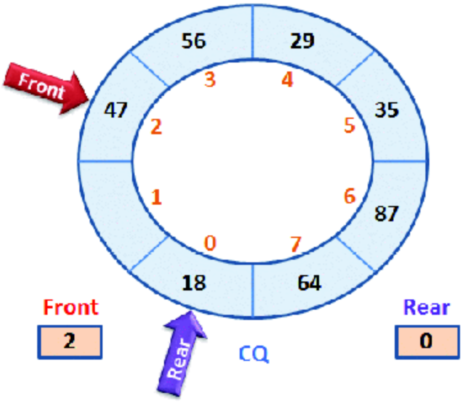
ചിത്രം 3.13(a): സംഖ്യ നീക്കം ചെയ്തതിനു ശേഷമുള്ള വൃത്താകാര ക്യൂ

ചിത്രം 3.13 ൽ രണ്ട് അംഗങ്ങളെ നീക്കംചെയ്യൽ പ്രവർത്തനങ്ങൾ നടത്തി എന്ന് അനുമാനിക്കുക. അപ്പോൾ Front ന്റെ വില 2 ഉം ചിത്രം 3.13(a) ൽ കാണിച്ചിരിക്കുന്നതുപോലെ 4 ശൂന്യമായ സ്ഥാനങ്ങളും സ്ഥലങ്ങളും നമുക്ക് ലഭിക്കുന്നു. ഈ സ്ഥാനങ്ങളുടെ സൂചികകളുടെ

വില യഥാക്രമം 6, 7, 0, 1 എന്നിങ്ങനെയായിരിക്കും. നമ്മൾ വീണ്ടും രണ്ട് അംഗങ്ങളെ കൂട്ടിച്ചേർക്കുകയാണെങ്കിൽ Rear ന്റെ വില 7 ആയി മാറുന്നു. ചിത്രം 3.13(b) ൽ കാണിച്ചിരിക്കുന്നതു പോലെ 0,1 എന്നീ സൂചികകളുടെ സ്ഥാനം ഇപ്പോഴും ശൂന്യമായിട്ടാണുള്ളത്. അതിനാൽ വീണ്ടും ഉൾപ്പെടുത്തൽ പ്രവർത്തനം ഇവിടെ നിർവഹിക്കാവുന്നതാണ്. ഇത്തവണ Rear ന്റെ വില 0 ആയി നിശ്ചയിക്കുകയും അതിനു ശേഷം ഉൾപ്പെടുത്തൽ പ്രവർത്തനം നിർവഹിക്കുകയും ചെയ്യാവുന്നതാണ്. ക്യൂവിന്റെ ഈ അവസ്ഥയെ ചിത്രം 3.13(c) ൽ കാണിച്ചിരിക്കുന്നു.



ചിത്രം 3.13 (b): Rear ന് അതിന്റെ ഉയർന്ന വില



ചിത്രം 3.13(c): Rear ന് അതിന്റെ താഴ്ന്ന വില

നമുക്ക് വിലയിരുത്താം

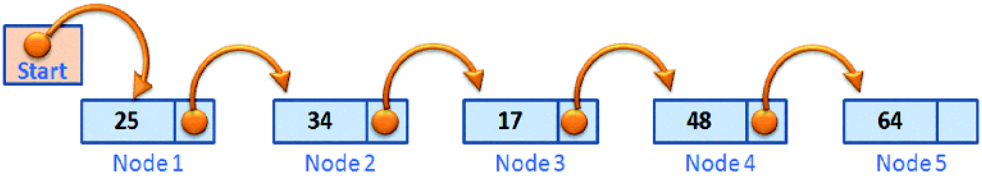
1. ഡാറ്റ സ്ട്രക്ചർ എന്നാലെന്ത്?
2. ഡാറ്റ ക്രമീകരണത്തിനായി സ്റ്റാക്കിൽ ഉപയോഗിക്കുന്ന തത്വം ____________ ആകുന്നു.
3. FIFO എന്ന തത്വം ഉപയോഗപ്പെടുത്തുന്ന ഡാറ്റാസ്ട്രക്ചറിന്റെ പേരെഴുതുക.
4. അറെയിലെ അണ്ടർഫ്ളോ (underflow) എന്നാലെന്ത്?
5. സ്റ്റാക്കിന്റെ ഏത് അംഗമാണ് നീക്കം ചെയ്യാൻ സാധിക്കുക (ആദ്യത്തേത് അല്ലെങ്കിൽ അവസാനത്തേത്)?

3.4 ലിങ്ക്ഡ് ലിസ്റ്റ് (Linked list)

അംഗങ്ങളുടെ എണ്ണത്തിന് പരിമിതിയില്ലാത്ത ഡാറ്റ അംഗങ്ങളുടെ ശേഖരത്തെയാണ് ലിങ്ക്ഡ് ലിസ്റ്റ് എന്ന് പറയുന്നത്. അറേകൾ ഉപയോഗിച്ച് പ്രാവർത്തികമാക്കുന്ന യുക്തിപരമായ ആശയങ്ങൾ മാത്രമാണ് കഴിഞ്ഞ ഭാഗങ്ങളിൽ വിശദീകരിച്ച സ്റ്റാക്ക്, ക്യൂ മുതലായ ഡാറ്റ സ്ട്രക്ചറുകൾ. അതിനാൽ ഇവ സ്ഥിര ഡാറ്റ സ്ട്രക്ചറുകളാണ്. എന്നാൽ ലിങ്ക്ഡ് ലിസ്റ്റ് ഒരു പരിവർത്തനാത്മക ഡാറ്റ സ്ട്രക്ചറാണ്. പുതിയ ഡാറ്റ ഇനങ്ങൾ കൂട്ടിച്ചേർക്കുമ്പോൾ ഇത് വളരുകയും നീക്കം ചെയ്യുമ്പോൾ ചുരുങ്ങുകയും ചെയ്യുന്നു. ലിങ്ക്ഡ് ലിസ്റ്റിന് മുഴുവൻ അംഗങ്ങൾക്കും ആവശ്യമായ മെമ്മറി ആദ്യമേ അനുവദിക്കുകയില്ല.

ലിസ്റ്റിലേക്ക് ഒരു പുതിയ അംഗത്തെ ഉൾപ്പെടുത്തുന്നതിന് തൊട്ടു മുമ്പാണ് ആ അംഗത്തിന് ആവശ്യമായ മെമ്മറി അനുവദിക്കുന്നത്. അതുകൊണ്ടാണ് ഇതിനെ (ഡൈനാമിക്) ഡാറ്റ സ്ട്രക്ചറുകളായി കരുതുന്നത്. അറെ അധിഷ്ഠിത ഡാറ്റ സ്ട്രക്ചറുകളുമായി ലിങ്ക്ഡ് ലിസ്റ്റിനുള്ള മറ്റൊരു വ്യത്യാസം ലിങ്ക്ഡ് ലിസ്റ്റിലെ അംഗങ്ങൾ മെമ്മറിയിൽ പല ഭാഗത്തായിട്ടാണ് സംഭരിക്കപ്പെടുന്നത്. എന്നാൽ അവ പോയിന്ററുകളുടെ സഹായത്തോടെ പരസ്പരം ബന്ധിപ്പിച്ചിരിക്കുന്നു. നമ്മൾ അധ്യായം ഒന്നിൽ പഠിച്ചത് പ്രകാരം പോയിന്റർ എന്നാൽ മെമ്മറി സ്ഥാനങ്ങളുടെ വിലാസം സംഭരിച്ചു വച്ചിരിക്കുന്ന വേരിയബിളാണ്. അപ്പോൾ ഒരുകാര്യം വ്യക്തമാണ് ലിങ്ക്ഡ് ലിസ്റ്റിലെ ഒരു അംഗത്തിന് ഡാറ്റയും അഡ്രസ്സും ഉണ്ടായിരിക്കും. ലിങ്ക്ഡ് ലിസ്റ്റിലെ ഒരു അംഗത്തെ നോഡ് (Node) എന്ന് വിളിക്കുന്നു. നോഡിൽ അടങ്ങിയിരിക്കുന്ന അഡ്രസ്സിനെ ലിങ്ക് (Link) എന്ന് പറയുന്നു.

ഒരു ഡാറ്റയും ലിങ്കും (ലിസ്റ്റിലെ അടുത്ത നോഡിലേക്കുള്ള പോയിന്റർ) അടങ്ങിയ നോഡുകളുടെ ശേഖരമാണ് ലിങ്ക്ഡ് ലിസ്റ്റ്. അതായത് ലിസ്റ്റിലെ ആദ്യത്തെ നോഡിൽ ആദ്യത്തെ ഡാറ്റ അംഗവും രണ്ടാമത്തെ നോഡിന്റെ വിലാസവും അടങ്ങിയിരിക്കുന്നു. രണ്ടാമത്തെ നോഡിൽ രണ്ടാമത്തെ ഡാറ്റ അംഗവും മൂന്നാമത്തെ നോഡിന്റെ വിലാസവും അടങ്ങിയിരിക്കുന്നു. ഇപ്രകാരം ലിസ്റ്റ് തുടരുന്നു. അവസാനത്തെ നോഡിൽ അവസാനത്തെ ഡാറ്റയും ഒരു നൾ പോയിന്ററും (Null Pointer) അടങ്ങിയിരിക്കുന്നു. എവിടെയും പോയിൻറ് ചെയ്യാത്ത പോയിന്ററാണ് Null Pointer. അങ്ങനെയെങ്കിൽ ആദ്യത്തെ നോഡിന്റെ വിലാസം എവിടെയാണുണ്ടാകുക? ആദ്യത്തെ നോഡിന്റെ വിലാസം അടങ്ങിയ ഒരു പ്രത്യേക പോയിന്റർ എല്ലാ ലിങ്ക്ഡ് ലിസ്റ്റിലും ഉണ്ടായിരിക്കും. ഇതിനെ സ്റ്റാർട്ട് (Start) അല്ലെങ്കിൽ ഹെഡർ (Header) എന്നു പറയുന്നു. അഞ്ച് അംഗങ്ങൾ അടങ്ങിയ ഒരു ലിങ്ക്ഡ് ലിസ്റ്റിനെ ചിത്രം 3.14 ൽ പ്രദർശിപ്പിച്ചിരിക്കുന്നു.



ചിത്രം 3.14: നോഡുകളുള്ള ലിങ്ക്ഡ് ലിസ്റ്റ്

ഡാറ്റയായി ഒരു സംഖ്യയും പോയിന്റർ ആയി അടുത്ത നോഡിലേക്കു ചൂണ്ടിക്കാണിക്കുന്ന ലിങ്കും അടങ്ങിയ നോഡുകൾ ആണ് ചിത്രത്തിൽ കാണിച്ചിരിക്കുന്നത്. എല്ലാ നോഡുകളുടെയും വലുപ്പം സമമായിരിക്കും. അതായത് ഓരോ നോഡിനു വേണ്ടിയും മാറ്റി വയ്ക്കുന്ന മെമ്മറി സമമായിരിക്കും.

3.4.1 ലിങ്ക്ഡ് ലിസ്റ്റ് പ്രാവർത്തികമാക്കൽ (Implementation of linked list)

ഓരോ നോഡിനും മെമ്മറി അനുവദിക്കുന്ന നോഡുകളുടെ ശേഖരമാണ് ലിങ്ക്ഡ് ലിസ്റ്റ് എന്ന് നമ്മൾ നേരത്തെ കണ്ടു കഴിഞ്ഞു. നോഡിനു വേണ്ടിയുള്ള മെമ്മറിയിൽ ഏറ്റവും കുറഞ്ഞത് രണ്ടു തരത്തിൽപ്പെട്ട ഡാറ്റയാണ് അടങ്ങിയിരിക്കുന്നത്, അതിൽ ഒന്ന് അംഗത്തിന്റെ യഥാർഥ ഡാറ്റയും രണ്ടാമത്തേത് അടുത്ത നോഡിലേക്കുള്ള പോയിന്ററും ആയിരിക്കും. വിവിധ തരത്തിൽപ്പെട്ട ഡാറ്റകൾ അടങ്ങിയ ഉപയോക്തൃ നിർവചിതമായ

ഡാറ്റ ഇനമാണ് സ്ട്രക്ചർ എന്നത് നാം അധ്യായം ഒന്നിൽ കണ്ടതാണ്. സ്ട്രക്ചറിലെ അംഗം ഒരു പോയിന്റർ ആകാം എന്ന് മാത്രമല്ല അതേ സ്ട്രക്ചറിലേക്കു സൂചിപ്പിക്കുന്ന പോയിന്ററുമാകാം. അത്തരം സ്ട്രക്ചറുകളെ സ്വയം സൂചിതസ്ട്രക്ചറുകൾ (Self Referential Sructure) എന്ന് പറയുന്നു. അതായത് സ്വയം സൂചിത സ്ട്രക്ചറുകളുടെ സഹായത്തോടെയാണ് ലിങ്ക്ഡ് ലിസ്റ്റുകൾ നിർമ്മിക്കുന്നത്. താഴെ പറയുന്ന സ്വയം സൂചിത സ്ട്രക്ചർ ഉപയോഗിച്ച് ചിത്രം 3.14 ൽ കാണിച്ചിരിക്കുന്ന ലിങ്ക്ഡ് ലിസ്റ്റിന്റെ നോഡുകൾ രൂപകൽപ്പന ചെയ്തിരിക്കുന്നു.

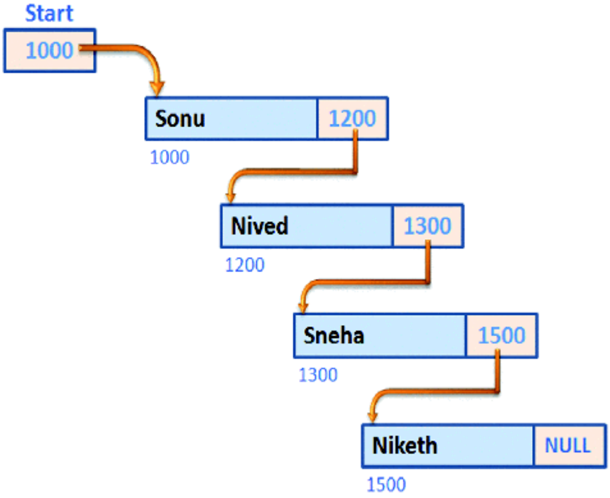
```
struct Node
{
    int data;
    Node *link;
};
```

സ്ട്രക്ചറിന്റെ പേര് Node ആണെന്നും സ്ട്രക്ചറിലെ രണ്ടാമത്തെ അംഗം അതേ സ്ട്രക്ചറിന്റെ തരത്തിലുള്ള ഒരു പോയിന്റർ ആണെന്നും നമുക്ക് കാണാം. താഴെ പറയുന്ന പ്രസ്താവന ഉപയോഗിച്ച് ആദ്യത്തെ നോഡിനെ സൂചിപ്പിക്കുന്ന `Start` എന്ന പ്രത്യേക പോയിന്റർ നിർമിക്കാവുന്നതാണ്.

```
struct Node * Start;        or        Node *Start;
```

ചിത്രം 3.15 ൽ സ്ട്രിങ്ങുകളുടെ ഒരു ലിങ്ക്ഡ് ലിസ്റ്റ് ആണ് കാണിച്ചിരിക്കുന്നത്. ഇതിലെ നോഡുകളുടെ ഡാറ്റ സ്ട്രിങ്ങുകൾ ഉപയോഗിച്ചും ലിങ്കുകൾ മറ്റു നോഡുകളുടെ വിലാസം ഉപയോഗിച്ചും നിറച്ചിരിക്കും. ശ്രദ്ധിക്കേണ്ട കാര്യം ഇവിടെ വിലാസങ്ങൾ എന്നത് അനുമാനിച്ചിരിക്കുന്ന ചില സംഖ്യകൾ ആണ്. ഈ നോഡുകളെ താഴെ പറയുന്ന സ്ട്രക്ചർ ഉപയോഗിച്ച് പ്രതിനിധീകരിക്കാവുന്നതാണ്.

```
struct Node
{
    char data[10];
    Node *link;
};
```



ചിത്രം 3.15: ലിങ്ക്ഡ് ലിസ്റ്റ് പ്രാവർത്തികമാക്കൽ

പ്രവർത്തന സമയത്ത് ആദ്യത്തെ നോഡിന് 1000 എന്ന വിലാസമുള്ള മെമ്മറി സ്ഥാനം നീക്കിവയ്ക്കുന്നതായി അനുമാനിക്കുക. അതിനാൽ `Start` പോയിന്ററിന്റെ വില 1000 ആയിരിക്കും. ഒന്നാമത്തെ നോഡിന്റെ ഡാറ്റ ഭാഗത്തു "Sonu" എന്ന സ്ട്രിങ് സംഭരിച്ചിരിക്കുന്നു. രണ്ടാമത്തെ നോഡിന് 1200 എന്ന മെമ്മറി വിലാസമാണ് നൽകിയിരിക്കുന്നത്. അതിന്റെ ഡാറ്റ ഭാഗത്തു "Nived" എന്ന വില സംഭരിച്ചിരിക്കുന്നു. രണ്ടാമത്തെ നോഡ്

ആയതിനാൽ ആദ്യത്തെ നോഡിന്റെ ലിങ്ക് ഭാഗത്ത് ഈ നോഡിന്റെ അഡ്രസ് സംഭരിച്ചിരിക്കുന്നു. ചിത്രത്തിൽ കാണിച്ചിരിക്കുന്നതു പോലെ അവസാനത്തെ നോഡിൽ ലിങ്കിന്റെ സ്ഥാനത്ത് NULL പോയിന്റർ ആയിരിക്കും ഉണ്ടായിരിക്കുക.

3.4.2 ലിങ്ക്ഡ് ലിസ്റ്റിലെ പ്രവർത്തനങ്ങൾ (Operations on linked list)

ഭാഗം 3.1.2 ൽ പ്രതിപാദിച്ചിരിക്കുന്ന എല്ലാ പ്രവർത്തനങ്ങളും ഉപാധികളില്ലാതെ ലിങ്ക്ഡ് ലിസ്റ്റിൽ ചെയ്യുവാൻ സാധിക്കുന്നതാണ്. പക്ഷേ നിർമാണം, കടന്നുപോകൽ, ഉൾപ്പെടുത്തൽ, നീക്കം ചെയ്യൽ എന്നീ പ്രവർത്തനങ്ങളെപ്പറ്റി മാത്രമേ നമ്മൾ ഇവിടെ ചർച്ച ചെയ്യുന്നുള്ളൂ. ഉപരിപഠന സമയത്തു ബാക്കി പ്രവർത്തനങ്ങളെപ്പറ്റി നിങ്ങൾ പഠിക്കുന്നതാണ്.

a. ലിങ്ക്ഡ് ലിസ്റ്റിന്റെ നിർമാണം (Creation of linked list)

തുടക്കത്തിൽ നമുക്ക് സ്വയം സൂചകമായി ഒരു സ്ട്രക്ചർ നിർവചിക്കേണ്ടതുണ്ട്. Start അല്ലെങ്കിൽ Header എന്ന ഒരു പോയിന്റർ വേരിയബിളിനെ നിർവചിച്ച് നൾ (NULL) എന്ന വില നൽകുന്നു. ഇനി ആവശ്യകതയനുസരിച്ച് നോഡുകൾക്കുള്ള മെമ്മറി അസ്ഥിരമായി നീക്കിവച്ചുകൊണ്ട് ലിങ്ക്ഡ് ലിസ്റ്റ് നിർമിക്കാം.

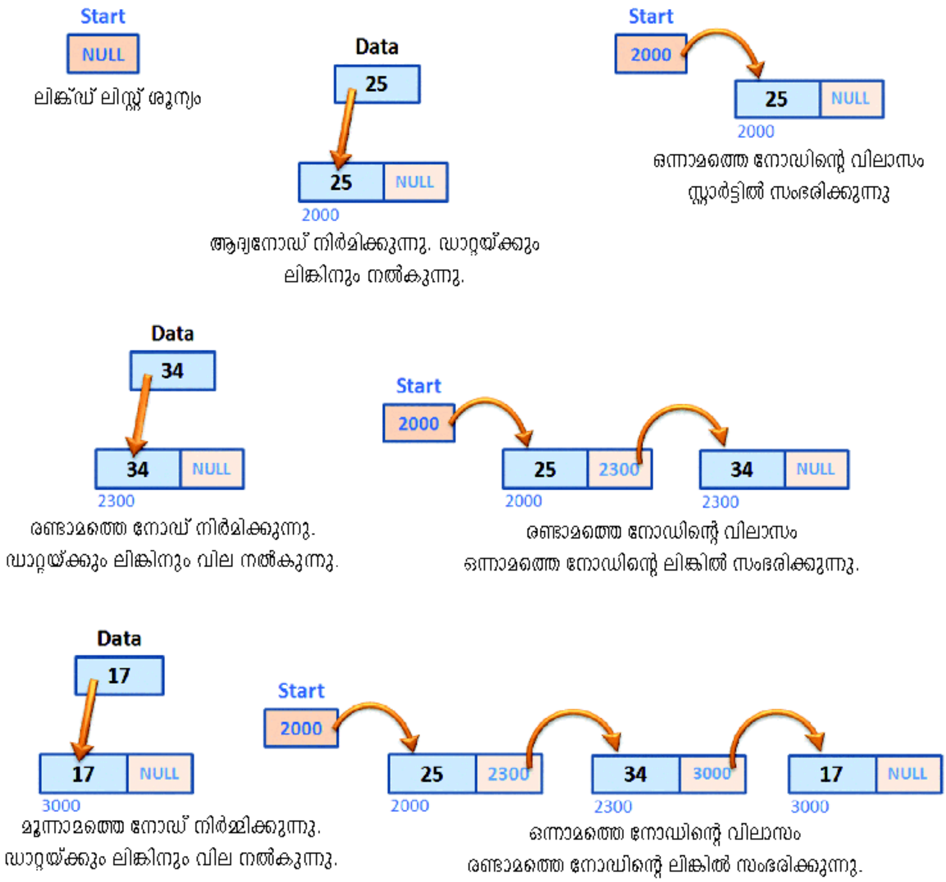
താഴെ പറയുന്ന ഘട്ടങ്ങൾ വികസിപ്പിക്കുവാൻ സാധിക്കുമോ എന്ന് പരിശോധിക്കുക.

ഘട്ടം 1: ഒരു നോഡ് നിർമിച്ച് അതിന്റെ മെമ്മറി വിലാസം ലഭ്യമാക്കുക.

ഘട്ടം 2: ഡാറ്റയും NULL എന്ന വിലയും നോഡിൽ സംഭരിക്കുക.

നമുക്കു ചെയ്യാം

ലിങ്ക്ഡ് ലിസ്റ്റ് നിർമാണ സമയത്തെ അവസ്ഥയാണ് ചിത്രം 3.16 ൽ കാണിച്ചിരിക്കുന്നത്. Node എന്ന പേരുള്ള ഒരു സ്വയം സൂചിത സ്ട്രക്ചർ നിർമിച്ചതായും Start എന്ന NULL തരത്തിൽപ്പെട്ട പോയിന്ററിനു Node എന്ന വില നൽകിയതായും നമുക്ക് അനുമാനിക്കാം. ഇനി താഴെ പറയുന്ന ചിത്രങ്ങൾ നിരീക്ഷിച്ച് ലിങ്ക്ഡ് ലിസ്റ്റ് നിർമാണത്തിനാവശ്യമായ നടപടിക്രമം തയാറാക്കുക.



ചിത്രം 3.16: ലിങ്ക്ഡ് ലിസ്റ്റ് നിർമാണത്തിന്റെ തുടർച്ചയായ പ്രവർത്തനങ്ങൾ

ഘട്ടം 3: അത് ആദ്യത്തെ നോഡ് ആണെങ്കിൽ അതിന്റെ വിലാസം node ൽ സംഭരിക്കുക.

ഘട്ടം 4: അത് ആദ്യത്തെ നോഡ് അല്ലെങ്കിൽ അതിന്റെ വിലാസം തൊട്ടു മുമ്പത്തെ നോഡിന്റെ ലിങ്കിൽ സംഭരിക്കുക.

ഘട്ടം 5: ഉപയോക്താവിന്റെ ആവശ്യാനുസരണം ഘട്ടം 1 മുതൽ 4 വരെ ആവർത്തിക്കുക.

യഥാർഥത്തിൽ ലിങ്ക്ഡ് ലിസ്റ്റിന്റെ നിർമാണം എന്നത് ലിങ്ക്ഡ് ലിസ്റ്റിന്റെ അവസാന ഭാഗത്ത് ആവർത്തിച്ചു നടത്തുന്ന ഉൾപ്പെടുത്തൽ പ്രവർത്തനം ആണ്. രണ്ടാമത്തെ നോഡ് മുതൽ ഉൾപ്പെടുത്തൽ പ്രവർത്തനം ചെയ്യണമെങ്കിൽ നിലവിലുള്ള ലിസ്റ്റിലെ അവസാനത്തെ അംഗത്തിന്റെ വിലാസം ലഭിക്കുന്നതിനാവശ്യമായ കടന്നുപോകൽ പ്രവർത്തനം നടത്തണം. അതിനാൽ കടന്നുപോകൽ പ്രവർത്തനത്തെപ്പറ്റി നമുക്ക് വിശദമായി ചർച്ച ചെയ്യാം.

b. ലിങ്ക്ഡ് ലിസ്റ്റിലൂടെ കടന്നുപോകുന്നു (Traversing a linked list)

നമുക്കറിയാം ഒരു ഡാറ്റ സ്ട്രക്ചറിൽ കടന്നുപോകൽ എന്നാൽ അതിലുള്ള എല്ലാ അംഗങ്ങളെയും സന്ദർശിക്കലാണ്. ലിങ്ക്ഡ് ലിസ്റ്റുകളുടെ കാര്യത്തിൽ യാത്ര ചെയ്യൽ ആരംഭിക്കുന്നത് ആദ്യത്തെ നോഡ് മുതലാണ്. ആദ്യത്തെ നോഡിന്റെ വിലാസം *arrow* എന്ന

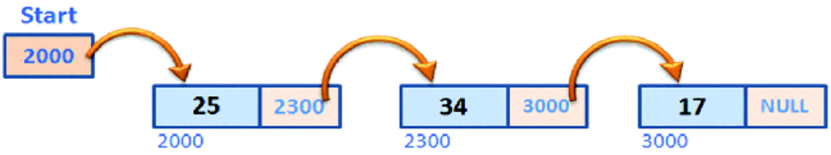
പോയിന്ററിൽ നിന്നും ലഭിക്കുന്നു. ഇതുപയോഗിച്ച് ആരോ (->) ഓപ്പറേറ്ററിന്റെ സഹായത്തോടെ നമുക്ക് ഡാറ്റ ഭാഗത്തെ ലഭ്യമാക്കാം. ശേഷം രണ്ടാമത്തെ നോഡിന്റെ വിലാസം അടങ്ങിയ ആദ്യത്തെ നോഡിന്റെ ലിങ്ക് ഭാഗത്തെ ഉപയോഗപ്പെടുത്തുന്നു. ഈ വിലാസം ഉപയോഗിച്ച് രണ്ടാമത്തെ നോഡിലെ ഡാറ്റയും ലിങ്കും ലഭ്യമാക്കുന്നു. നോഡിന്റെ ലിങ്കിൽ NULL ലഭിക്കുന്നത് വരെ ഈ പ്രക്രിയ തുടരുന്നു.

താഴെ പറയുന്ന ഘട്ടങ്ങൾ നിങ്ങൾക്ക് നിർവചിക്കാൻ കഴിയുന്നുണ്ടോ എന്ന് പരിശോധിക്കുക.

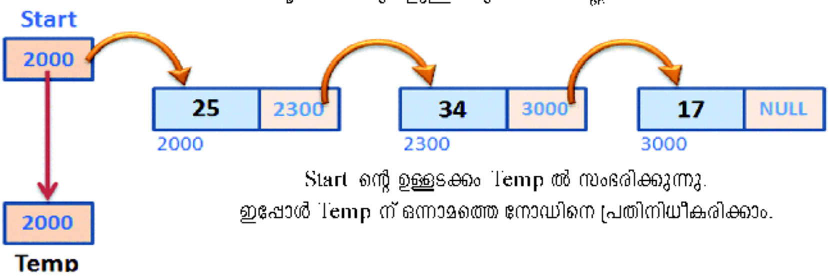
ഘട്ടം 1: ആദ്യത്തെ നോഡിന്റെ വിലാസം Start ൽ നിന്നും Temp ലേക്ക് സംഭരിക്കുക.

നമുക്ക് ചെയ്യാം

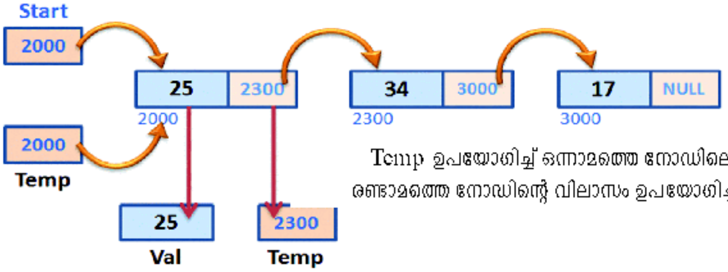
ചിത്രം 3.17 ൽ നൽകിയിരിക്കുന്ന ലിങ്ക്ഡ് ലിസ്റ്റ് നിരീക്ഷിച്ച് യാത്ര ചെയ്യൽ പ്രവർത്തനത്തിന് വേണ്ട ഘട്ടങ്ങൾ നമുക്ക് നിർവചിക്കാം. Node തരത്തിൽപ്പെട്ട ഒരു പോയിന്റർ ആണ് TEMP എന്നും VAL എന്നത് നോഡിൽ നിന്നുമുള്ള ഡാറ്റ സംഭരിക്കാനാവശ്യമായ ഒരു വേരിയബിൾ ആണെന്നും കരുതുക.



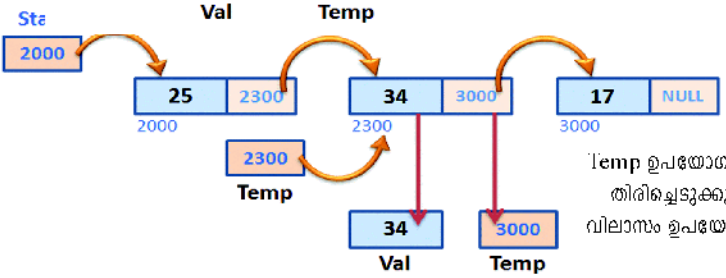
മൂന്ന് നോഡുകളുള്ള ഒരു ലിങ്ക്ഡ് ലിസ്റ്റ്



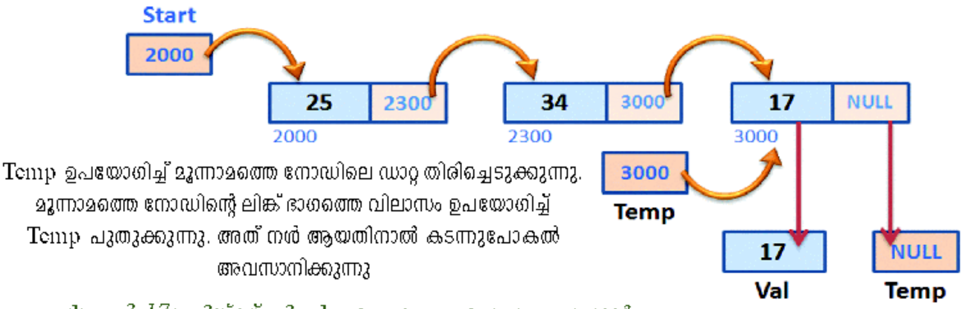
Start ന്റെ ഉള്ളടക്കം Temp ൽ സംഭരിക്കുന്നു.
ഇപ്പോൾ Temp ന് ഒന്നാമത്തെ നോഡിനെ പ്രതിനിധീകരിക്കാം.



Temp ഉപയോഗിച്ച് ഒന്നാമത്തെ നോഡിലെ ഡാറ്റ തിരിച്ചെടുക്കുന്നു.
രണ്ടാമത്തെ നോഡിന്റെ വിലാസം ഉപയോഗിച്ച് Temp നെ പുതുക്കുന്നു.



Temp ഉപയോഗിച്ച് രണ്ടാമത്തെ നോഡിലെ ഡാറ്റ തിരിച്ചെടുക്കുന്നു. മൂന്നാമത്തെ നോഡിന്റെ വിലാസം ഉപയോഗിച്ച് Temp നെ പുതുക്കുന്നു.



ചിത്രം 3.17: ലിങ്ക്ഡ് ലിസ്റ്റിലൂടെ കടന്നുപോകുന്ന പ്രവർത്തനം

ഘട്ടം 2: Temp ലെ വിലാസം ഉപയോഗിച്ച് ആദ്യത്തെ നോഡിന്റെ ഡാറ്റ Val ലിൽ സംഭരിക്കുക.

ഘട്ടം 3: ഈ നോഡിന്റെ ലിങ്ക് ഭാഗത്തെ വിലാസം (അതായത് അടുത്ത നോഡിന്റെ വിലാസം) Temp ലേക്ക് സ്വീകരിക്കുക.

ഘട്ടം 4: Temp ലെ വിലാസം NULL അല്ലെങ്കിൽ ഘട്ടം 2 ലേക്ക് പോകുക അല്ലെങ്കിൽ അവസാനിപ്പിക്കുക.

Start ന്റെ വില NULL അല്ലാത്ത സാഹചര്യത്തിൽ ഒരു ലിങ്ക്ഡ് ലിസ്റ്റിനെ നിർമിക്കാൻ മേൽപ്പറഞ്ഞ ഘട്ടങ്ങൾ ഉപയോഗിക്കുന്നു. ഇത്തരം സാഹചര്യത്തിൽ അവസാനത്തെ നോഡിന്റെ വിലാസം കണ്ടുപിടിക്കേണ്ടതുണ്ട്. എന്തിനെന്നാൽ പുതിയ നോഡിന്റെ വിലാസം അതിന്റെ ലിങ്ക് ഭാഗത്താണ് സംഭരിക്കേണ്ടത്. Start ൽ നിന്നും ആദ്യത്തെ നോഡിന്റെ വിലാസം സ്വീകരിച്ചുകൊണ്ടാണ് യാത്രചെയ്യൽ പ്രവർത്തനം ആരംഭിക്കുന്നത്. ഈ വിലാസം താൽക്കാലിക പോയിന്റർ വേരിയബിളിലേക്ക് പകർത്തുകയും (ചിത്രത്തിൽ Temp) പിന്നീട് Temp ചൂണ്ടുന്ന നോഡിലെ ലിങ്ക് ഭാഗത്തുള്ള വിലാസം പകർത്തിയെടുത്തു. താൽക്കാലിക പോയിൻറ്റർ വേരിയബിൾ പുതുക്കപ്പെടുകയും ചെയ്യുന്നു. Temp ചൂണ്ടുന്ന നോഡിന്റെ ലിങ്ക് ഭാഗത്തു നൾ വില ലഭിക്കുന്നത് വരെ ഈ സന്ദർശനം തുടർന്ന് കൊണ്ടേയിരിക്കുന്നു.

c. ലിങ്ക്ഡ് ലിസ്റ്റിലെ ഉൾപ്പെടുത്തൽ പ്രവർത്തനം (Insertion in a linked list)

ലിങ്ക്ഡ് ലിസ്റ്റിൽ ഒരു അംഗത്തെ ഉൾപ്പെടുത്തൽ എന്നാൽ ആ അംഗം ഉൾപ്പെടുന്ന നോഡിനെ നിശ്ചിതസ്ഥാനത്തു സ്ഥാപിക്കുന്ന പ്രക്രിയയാണ്. അറെയിലേതു പോലെ ലിങ്ക്ഡ് ലിസ്റ്റിൽ നോഡിനെ എവിടെ വേണമെങ്കിലും ഉൾപ്പെടുത്താവുന്നതാണ്. തുടക്കത്തിലോ, അവസാനത്തിലോ അല്ലെങ്കിൽ രണ്ടു നോഡുകൾക്കിടയിലോ ആകാം.

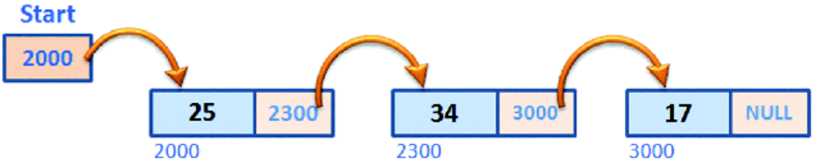
ഉൾപ്പെടുത്തൽ പ്രവർത്തനത്തിന് താഴെ പറയുന്ന ഘട്ടങ്ങൾ നിർവചിക്കാം:

ഘട്ടം 1: നോഡ് നിർമിച്ച് അതിന്റെ വിലാസം Temp ൽ സംഭരിക്കുന്നു.

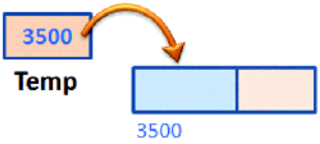
ഘട്ടം 2: Temp ഉപയോഗിച്ച് ഈ നോഡിന്റെ ഡാറ്റാഭാഗവും ലിങ്ക് ഭാഗവും സംഭരിക്കുക.

നമുക്കു ചെയ്യാം

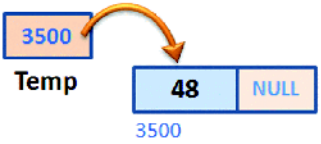
ആരംഭത്തിൽ മൂന്ന് നോഡുകളുള്ള ഒരു ലിങ്ക്ഡ് ലിസ്റ്റിന്റെ മൂന്നാം സ്ഥാനത്തേക്കു നോഡിനെ ഉൾപ്പെടുത്താനുള്ള പ്രവർത്തനങ്ങളാണ് ചിത്രം 3.18ൽ കാണിച്ചിരിക്കുന്നത്. ചിത്രം നിരീക്ഷിച്ചു ലിങ്ക്ഡ് ലിസ്റ്റിലേക്ക് ഒരു നോഡിനെ ഉൾപ്പെടുത്താനുള്ള പ്രവർത്തനത്തിന്റെ ഘട്ടങ്ങൾ നമുക്ക് നിർവചിക്കാം. Node തരത്തിൽപ്പെട്ട പോയിന്ററുകളാണ് Temp, PreNode, PostNode എന്നിവ എന്നും POS നോഡിനെ ഉൾപ്പെടുത്തേണ്ട സ്ഥാനം സംഭരിച്ചിരിക്കുന്ന വേരിയബിൾ ആണെന്നും നമുക്ക് അനുമാനിക്കാം.



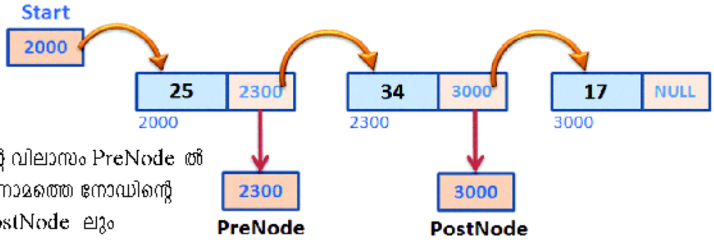
മൂന്ന് നോഡുള്ള ഒരു ലിങ്ക്ഡ് ലിസ്റ്റ്



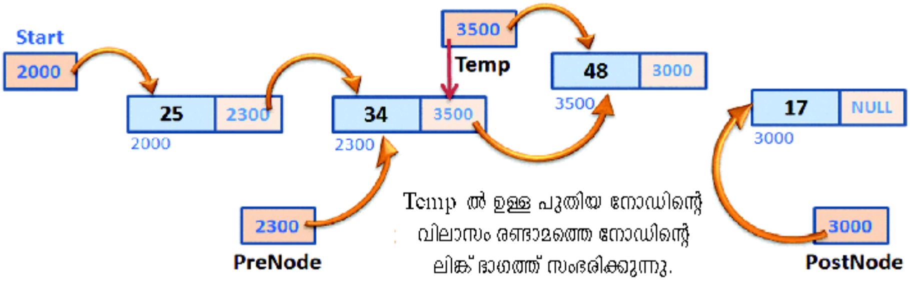
പുതിയ നോഡ് നിർമിച്ച് വിലാസം Tempൽ സംഭരിക്കുന്നു



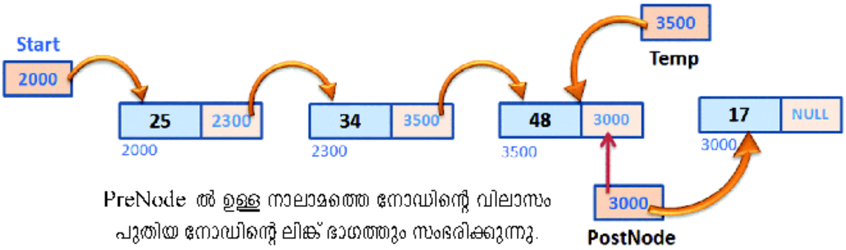
പുതിയ നോഡിന്റെ ഡാറ്റ ഭാഗത്തിനും
ലിങ്ക് ഭാഗത്തിനും വില നൽകുന്നു



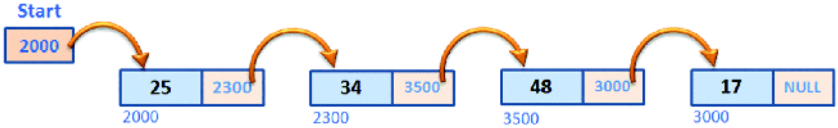
രണ്ടാമത്തെ നോഡിന്റെ വിലാസം PreNode ൽ സംഭരിക്കുന്നു. മൂന്നാമത്തെ നോഡിന്റെ വിലാസം PostNode ലും



Temp ൽ ഉള്ള പുതിയ നോഡിന്റെ
വിലാസം രണ്ടാമത്തെ നോഡിന്റെ
ലിങ്ക് ഭാഗത്ത് സംഭരിക്കുന്നു.



PreNode ൽ ഉള്ള നാലാമത്തെ നോഡിന്റെ വിലാസം
പുതിയ നോഡിന്റെ ലിങ്ക് ഭാഗത്തും സംഭരിക്കുന്നു.



മൂന്നാമത്തെ സ്ഥാനത്ത് പുതിയ നോഡ് ഉൾക്കൊള്ളിച്ച ശേഷം ഉള്ള അവസ്ഥ

ചിത്രം 3.18: ലിങ്ക്ഡ് ലിസ്റ്റിലെ ഉൾപ്പെടുത്തൽ പ്രവർത്തനം

ഘട്ടം 3: യാത്ര ചെയ്യൽ പ്രവർത്തനത്തിന്റെ സഹായത്തോടെ (POS-1), (POS+1) എന്നീ സ്ഥാനങ്ങളിലുള്ള നോഡുകളുടെ വിലാസം യഥാക്രമം PreNode, PostNode എന്നീ പോയിന്ററുകളിലേക്കു സ്വീകരിക്കുക.

ഘട്ടം 4: Temp (POS-1) സ്ഥാനത്തുള്ള നോഡ്) ന്റെ link ഭാഗത്തിലേക്കു PreNode (പുതിയ നോഡ്) ന്റെ വിലാസം സംഭരിക്കുക.

ഘട്ടം 5: Temp (പുതിയ നോഡ്) ന്റെ link ഭാഗത്തിലേക്കു PostNode (POS+1) സ്ഥാന ത്തുള്ള നോഡ്) ന്റെ വിലാസം സംഭരിക്കുക.

d. ലിങ്ക്ഡ് ലിസ്റ്റിൽ നിന്നും നീക്കം ചെയ്യൽ (Deletion from a linked list)

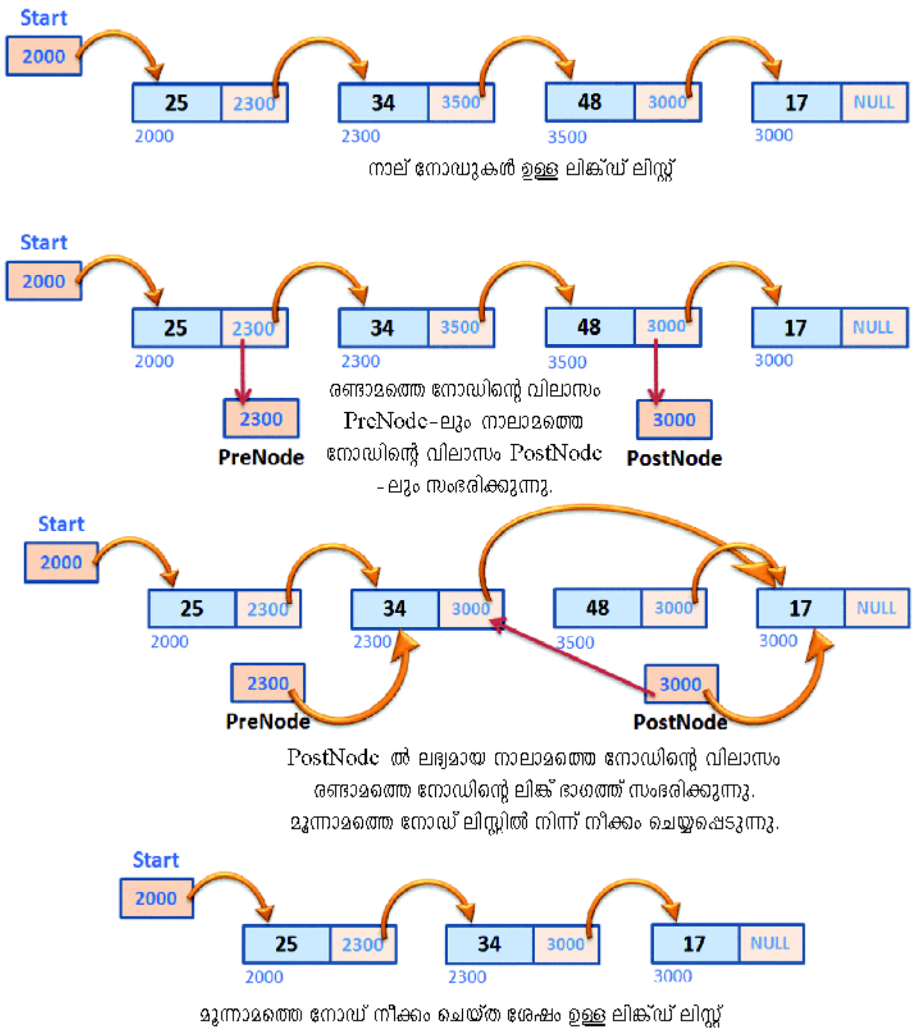
ലിങ്ക്ഡ് ലിസ്റ്റിലുള്ള ഒരു അംഗത്തെ നീക്കം ചെയ്യൽ എന്നാൽ ലിസ്റ്റിൽ നിന്നും ഒരു നോഡിനെ ഒഴിവാക്കുന്ന പ്രക്രിയയാണ്. നീക്കം ചെയ്യാനുള്ള നോഡിന്റെ സ്ഥാനം നൽകിയിട്ടുണ്ടായിരിക്കും. അതിനു പകരം ഡാറ്റയാണ് നൽകിയിരിക്കുന്നതെങ്കിൽ, ആ ഡാറ്റ അടങ്ങിയ നോഡ് തിരഞ്ഞു കണ്ടുപിടിക്കുകയും അതിന്റെ സ്ഥാനം കണ്ടെത്തേ ണ്ടതുമാണ്. അതിനുശേഷം നീക്കം ചെയ്യൽ പ്രവർത്തനത്തിനുള്ള നടപടിക്രമങ്ങൾ നടപ്പി ലാക്കേണ്ടതാണ്. ലിസ്റ്റിലെ ഏതു നോഡിനെയും നീക്കം ചെയ്യുന്നതിന് സാധ്യമാണ്. ഒന്നാമത്തെ നോഡാണ് നീക്കം ചെയ്യേണ്ടതെങ്കിൽ ഒന്നാമത്തെ നോഡിലെ link ലെ വിലാസം START ലേക്ക് സംഭരിക്കണം. അതുപോലെ അവസാനത്തെ നോഡാണ് നീക്കം ചെയ്യേണ്ടതെങ്കിൽ അവസാനത്തേതിന്റെ തൊട്ടു മുമ്പുള്ള നോഡിലെ link ന്റെ വില NULL ആയി സംഭരിക്കണം. ഒരു പ്രത്യേക സ്ഥാനത്തുള്ള നോഡിനെ നീക്കം ചെയ്യുന്ന തിന് ആവശ്യമായ നടപടിക്രമങ്ങൾ നമുക്ക് ചർച്ച ചെയ്യാം.

നീക്കം ചെയ്യൽ പ്രവർത്തനത്തെ നിർവചിക്കാൻ താഴെ പറയുന്ന നടപടികൾ നിങ്ങളെ സഹായിക്കുന്നു.

ഘട്ടം 1: കടന്നുപോകൽ പ്രവർത്തനത്തിന്റെ സഹായത്തോടെ (POS-1), (POS+1) എന്നീ സ്ഥാനങ്ങളിലുള്ള നോഡുകളുടെ വിലാസം യഥാക്രമം PreNode, PostNode എന്നീ പോയിന്ററുകളിലേക്കു സ്വീകരിക്കുക.

നമുക്കു ചെയ്യാം

തുടക്കത്തിൽ നാല് നോഡുകളുള്ള ലിങ്ക്ഡ് ലിസ്റ്റിൽ നിന്നും മൂന്നാ മത്തെ നോഡിനെ നീക്കം ചെയ്യാനുള്ള നടപടികളാണ് ചിത്രം 3.19ൽ പ്രതിപാദിച്ചിരിക്കുന്നത്. ചിത്രം നിരീക്ഷിച്ച് ലിങ്ക്ഡ് ലിസ്റ്റിൽ നിന്നും ഒരു നോഡിനെ നീക്കം ചെയ്യാനുള്ള പ്രവർത്തനത്തിന്റെ ഘട്ടങ്ങൾ നമുക്ക് നിർവചിക്കാം. Node തരത്തിൽപ്പെട്ട പോയിന്ററുകളാണ് PreNode, PostNode എന്നിവ എന്നും POS നോഡിനെ നീക്കം ചെ യ്യേണ്ട സ്ഥാനം സംഭരിച്ചിരിക്കുന്ന വേരിയബിൾ ആണെന്നും നമുക്ക് അനുമാനിക്കാം.



ചിത്രം 3.19: ലിങ്ക്ഡ് ലിസ്റ്റിലെ നീക്കം ചെയ്യൽ പ്രവർത്തനം

ഘട്ടം 2: PostNode ((POS+1) സ്ഥാനത്തുള്ള നോഡ്) ന്റെ link ഭാഗത്തിലേക്കു PreNode ((POS-1) സ്ഥാനത്തുള്ള നോഡിന്റെ വിലാസം സംഭരിക്കുക.

ഘട്ടം 3: POS സ്ഥാനത്തുള്ള നോഡിനെ മെമ്മറിയിൽ നിന്നും ഒഴിവാക്കുക.

ചിത്രം 3.19 ൽ സൂചിപ്പിച്ചിരിക്കുന്ന ആദ്യത്തെ രണ്ടു ഘട്ടങ്ങൾ പാലിച്ച് മൂന്നാമത്തെ നോഡിനെ രണ്ടാമത്തേതിൽ നിന്നും വേർപ്പെടുത്തിയാലും, മൂന്നാമത്തെ നോഡ് നാലാമത്തേതിലേക്കു ചൂണ്ടിക്കൊണ്ട് മെമ്മറിയിൽത്തന്നെ അവശേഷിക്കുന്നു. അതിനാൽ പ്രോഗ്രാമിങ് ഭാഷകളിലെ മെമ്മറി സ്വതന്ത്രമാക്കൽ സംവിധാനങ്ങൾ ഉപയോഗപ്പെടുത്തി ആ നോഡിനെ നീക്കം ചെയ്യേണ്ടതാണ്. അതുപോലെ ലിങ്ക്ഡ് ലിസ്റ്റുകളിൽ പ്രവർത്തനങ്ങൾ നടത്തുമ്പോൾ പ്രവർത്തനത്തിന് ശേഷം TEMP, PreNode, PostNode മുതലായ താൽക്കാലിക പോയിന്ററുകളും സ്വതന്ത്രമാക്കപ്പെടേണ്ടവയാണ്.

പുരോഗതി ഇതുവരെ

1. ഡൈനാമിക് ഡാറ്റാസ്ട്രക്ചറിന് ഒരു ഉദാഹരണം പറയുക.
2. ലിങ്ക്ഡ് ലിസ്റ്റ് എന്നാലെന്ത്?
3. ഒരു ലിങ്ക്ഡ് ലിസ്റ്റിന്റെ നോഡിൽ ______ ഉം ______ ഉം അടങ്ങിയിരിക്കും.
4. ഒരു ലിങ്ക്ഡ് ലിസ്റ്റിലെ നോഡിനെ നിർവചിക്കാൻ ഉപയോഗിക്കുന്ന പ്രോഗ്രാമിങ് ഭാഷയിലെ സംവിധാനം ഏതാണ്?
5. ഒരു ലിങ്ക്ഡ് ലിസ്റ്റിൽ Start അല്ലെങ്കിൽ Header ന്റെ ഉള്ളടക്കം എന്തായിരിക്കും?

സ്റ്റാക്കും ക്യൂ വും ലിങ്ക്ഡ് ലിസ്റ്റുകൾ ഉപയോഗിച്ചും നിർമിക്കാവുന്നതാണ്. അവയെ ഡൈനാമിക് സ്റ്റാക് ഡൈനാമിക് ക്യൂ എന്ന് വിളിക്കുന്നു. അടുത്ത നോഡിലേക്കു മാത്രം ചൂണ്ടുന്നതാകയാൽ നമ്മൾ ചർച്ച ചെയ്ത ലിങ്ക്ഡ് ലിസ്റ്റുകളെ സിംഗിളി ലിങ്ക്ഡ് ലിസ്റ്റ് എന്ന് വിളിക്കുന്നു. എന്നാൽ അടുത്ത നോഡിലേക്കും തൊട്ടു മുമ്പുള്ള നോഡിലേക്കും ചൂണ്ടുന്ന ഡബിളി ലിങ്ക്ഡ് ലിസ്റ്റും ഉണ്ട്. സ്വയം സൂചിതസ്ട്രക്ചറിൽ രണ്ടു പോയിന്ററുകൾ ഉപയോഗിച്ചാണ് ഇത് സാധ്യമാക്കുന്നത്. ട്രീ (TREE) പോലുള്ള സങ്കീർണമായ ഡാറ്റ സ്ട്രക്ചറുകൾ ഡബിളി ലിങ്ക്ഡ് ലിസ്റ്റുകൾ ഉപയോഗിച്ചാണ് നിർമ്മിക്കുന്നത്.

നമുക്ക് സംഗ്രഹിക്കാം

ഡാറ്റസ്ട്രക്ചറുകൾ എന്ന ആശയത്തെയും അവയുടെ മേൽ നടപ്പിലാക്കാവുന്ന പ്രവർത്തനങ്ങളെയും നമ്മൾ പരിചയപ്പെട്ടു. ഏതു തരത്തിലുള്ള ഡാറ്റയെയും പ്രതിനിധീകരിക്കാൻ അനുയോജ്യമായ വ്യത്യസ്തങ്ങളായ ഡാറ്റ സ്ട്രക്ചറുകളുണ്ട്. ഇവയിൽ ഉപയോഗിക്കുന്ന പ്രവർത്തനങ്ങളും അംഗങ്ങളെ ഗണങ്ങളാക്കാൻ ഉപയോഗിക്കുന്ന തത്വത്തിനനുസരിച്ചു വ്യത്യാസപ്പെട്ടിരിക്കും. ചില ഡാറ്റാ സ്ട്രക്ചറുകൾ യുക്തിപരമായ ആശയങ്ങൾ മാത്രമാണെങ്കിലും അവയുടെ നടപ്പിലാക്കലിനെക്കുറിച്ചും നമ്മൾ ചർച്ച ചെയ്തു. അറെയും ലിങ്ക്ഡ് ലിസ്റ്റും തമ്മിലുള്ള വ്യത്യാസങ്ങളെപ്പറ്റിയും ചർച്ചയിൽ പരാമർശിച്ചിരുന്നു. കമ്പ്യൂട്ടർ സയൻസിന്റെ ഉപരിപഠനത്തിനു ഈ അധ്യായത്തിൽ പറഞ്ഞ ആശയങ്ങൾ വളരെ വേണ്ടപ്പെട്ടവയാണ്.

നമുക്ക് പരിശോധിക്കാം

1. താഴെ പറയുന്ന പ്രസ്താവനകൾ വായിക്കുക:
   (i) ഒറ്റ ഘടകമായി പ്രോസസ് ചെയ്യപ്പെടുന്ന ഡാറ്റയുടെ ശേഖരം.
   (ii) അറെ ഉപയോഗിച്ചാണ് എല്ലാ ഡാറ്റ സ്ട്രക്ചറുകളും പ്രാവർത്തികമാക്കിയിരിക്കുന്നത്.
   (iii) സ്റ്റാക്ക്, ക്യൂ എന്നിവ യുക്തിസഹമായ ആശയങ്ങളും അറെ, ലിങ്ക്ഡ് ലിസ്റ്റ് എന്നിവ ഉപയോഗിച്ച് പ്രാവർത്തികമാക്കിയവയുമാണ്.
   (iv) സ്റ്റാറ്റിക് ഡാറ്റ സ്ട്രക്ചറുകളുടെ കാര്യത്തിൽ ഓവർഫ്ളോ സംഭവിക്കുന്നു.

   മുകളിൽ കൊടുത്തിരിക്കുന്ന പ്രസ്താവനകളിൽ ഏതാണ് ശരിയായിട്ടുള്ളത്? ഏറ്റവും ഉചിതമായത് തിരഞ്ഞെടുക്കുക.

   a. പ്രസ്താവനകൾ (i), (iii) മാത്രം
   b. പ്രസ്താവനകൾ (i), (ii), (iii) മാത്രം
   c. പ്രസ്താവനകൾ (i), (iii), (iv) മാത്രം
   d. പ്രസ്താവനകൾ (i), (ii), (iv) മാത്രം

2. ഡാറ്റ സ്ട്രക്ചറുകൾ സ്റ്റാറ്റിക്കോ ഡൈനാമിക്കോ ആവാം.
   a. സ്റ്റാറ്റിക് ഡാറ്റ സ്ട്രക്ചറുകൾക്കു രണ്ട് ഉദാഹരണം നൽകുക.
   b. സ്റ്റാറ്റിക് ഡാറ്റ സ്ട്രക്ചറുകൾ ഓവർ ഫ്ളോ സാഹചര്യം നേരിടേണ്ടി വരാം. എന്തുകൊണ്ട്?
   c. ലിങ്ക്ഡ് ലിസ്റ്റ് ഒരു ഡൈനാമിക് ഡാറ്റ സ്ട്രക്ചർ ആണ്. ഈ പ്രസ്താവനയെ ന്യായീകരിക്കുക.

3. സ്റ്റാക്കിലേക്കു ഒരു അംഗത്തെ ഉൾപ്പെടുത്താനുള്ള അൽഗോരിതം എഴുതുക.

4. പുഷ്, പോപ് പ്രവർത്തനങ്ങൾ എന്നാലെന്ത്?

5. സ്റ്റാക്കിൽ നിന്ന് ഒരു അംഗത്തെ നീക്കം ചെയ്യാനുള്ള അൽഗോരിതം എഴുതുക.

6. രേഖീയക്യൂവിൽ ഉൾപ്പെടുത്തൽ, നീക്കംചെയ്യൽ പ്രവർത്തനങ്ങൾ പ്രാവർത്തികമാക്കാനുള്ള അൽഗോരിതം എഴുതുക.

7. രേഖീയക്യൂവിന്റെ പരിമിതികൾ വൃത്താകൃതി ക്യൂ എങ്ങനെ മറികടക്കുന്നു.

8. ഡാറ്റ സ്ട്രക്ചറുകളിൽ നടപ്പാക്കാവുന്ന ചില പ്രവർത്തനങ്ങൾ താഴെ പറയുന്നു

   i. ഉൾപ്പെടുത്തൽ ii. നീക്കംചെയ്യൽ iii. തിരയൽ iv. ക്രമപ്പെടുത്തൽ

   a. ഇവയിൽ ഏതു പ്രവർത്തനത്തിനാണ് അണ്ടർ ഫ്ളോ സാഹചര്യം നേരിടേണ്ടി വരിക.
   b. ഈ സാഹചര്യം ഉചിതമായ ഡാറ്റ സ്ട്രക്ചറിന്റെ പശ്ചാത്തലത്തിൽ വിവരിക്കുക.

9. ചേരുംപടി ചേർക്കുക

A	B	C
a. അറെ	i. സ്റ്റാർട്ട്	1. വ്യത്യസ്ത ഭാഗങ്ങളിൽ ഉൾപ്പെടുത്തലും നീക്കം ചെയ്യലും സംഭവിക്കുന്നു.
b. സ്റ്റാക്ക്	ii. സൂചിക	2. ഒരേ ഭാഗത്തു തന്നെ ഉൾപ്പെടുത്തലും നീക്കം ചെയ്യലും സംഭവിക്കുന്നു
c. ക്യൂ	iii. റിയർ	3. സ്വയം സൂചിത സ്ട്രക്ചറുകൾ ഉപയോഗപ്പെടുത്തുന്നു.
d. ലിങ്ക്ഡ് ലിസ്റ്റ്	iv. ടോപ്പ്	4. അംഗത്തിന്റെ സ്ഥാനം ഉപയോഗിച്ചിട്ടാണ് അതിനെ ആക്സസ് ചെയ്യുന്നത്

10. അറെ കേന്ദ്രീകൃത ഡാറ്റ സ്ട്രക്ചറുകളുടേതുപോലെ ലിങ്ക്ഡ് ലിസ്റ്റുകൾക്ക് ഓവർഫ്ളോ സാഹചര്യം നേരിടേണ്ടി വരാത്തത് എന്തുകൊണ്ടെന്നു വിശദീകരിക്കുക.

X4U1I7

4 വെബ് സാങ്കേതികവിദ്യ

പ്രധാന പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പഠനം പൂർത്തിയാകുന്നതോടെ പഠിതാവ് ആർജിക്കേണ്ട പഠനനേട്ടങ്ങൾ

- സുരക്ഷിത ആശയവിനിമയം വെബ്പേജുകളിലൂടെ സാധ്യമാകുന്നതെങ്ങനെ എന്ന് വിശദീകരിക്കുന്നു.
- വെബ്സെർവറിന്റെ ഉപയോഗവും വെബ്ഹോസ്റ്റിങ്ങിനെക്കുറിച്ചുള്ള അറിവും വിവരിക്കുന്നു.
- സ്റ്റാറ്റിക്, ഡൈനാമിക് എന്നീ വെബ് പേജുകൾ താരതമ്യം ചെയ്യുന്നു.
- പ്രോഗ്രാമിങ് ഭാഷകളും സ്ക്രിപ്റ്റുകളും തമ്മിലുള്ള വ്യത്യാസം തിരിച്ചറിയുന്നു.
- വിവിധ സ്ക്രിപ്റ്റിംഗ് ഭാഷകൾ താരതമ്യം ചെയ്യുന്നു.
- കാസ്കേഡിംഗ് സ്റ്റൈൽ ഷീറ്റിന്റെ ആവശ്യകത വിശദീകരിക്കുന്നു.
- ലളിതമായ HTML എലമെന്റുകൾ ഉപയോഗിച്ചുള്ള വെബ്പേജ് നിർമാണം മനസ്സിലാക്കുന്നു.
- അടിസ്ഥാന HTML ടാഗുകളും അവയുടെ പ്രധാന ആട്രിബ്യൂട്ടുകളും പട്ടികപ്പെടുത്തുന്നു.
- HTML ടാഗുകളെ തരം തിരിക്കുന്നു.
- HTML പേജുകൾ ആകർഷകമാക്കുന്നതിന് ഫോർമാറ്റിംഗ് ടാഗുകൾ HTML ഡോക്യുമെന്റുകളിൽ ഉപയോഗിക്കുന്നത് എങ്ങനെയെന്ന് മനസ്സിലാക്കുന്നു.
- ഫോർമാറ്റിംഗ് ടാഗുകൾ തമ്മിലുള്ള സാമ്യങ്ങളും വ്യത്യാസങ്ങളും തിരിച്ചറിയുന്നു.
- <PRE>, <DIV> എന്നീ ടാഗുകളുടെ ഉപയോഗം നിരീക്ഷിക്കുന്നു.
- ഒരു ഡോക്യുമെന്റിലെ ഉള്ളടക്കമോ മറ്റ് ഒബ്ജക്ടുകളോ (വസ്തുക്കളോ) ചലിപ്പിക്കുന്നതിനുള്ള ടാഗ് തിരിച്ചറിയുന്നു.
- <FONT> ടാഗ് ഉപയോഗിച്ച് ടെക്സ്റ്റ് ഉള്ളടക്കം ഫലപ്രദമായി രൂപപ്പെടുത്തുന്നു.
- HTML ഡോക്യുമെന്റുകളിൽ കമന്റുകൾ ഉപയോഗിക്കുന്നു.
- <IMG> ടാഗ് ഉപയോഗിച്ച് HTML ഡോക്യുമെന്റുകളിൽ ചിത്രങ്ങൾ ഉൾക്കൊള്ളിക്കുന്നു.

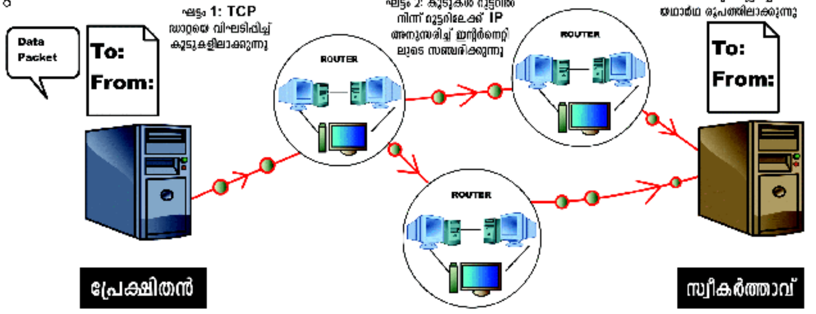
ഇന്റർനെറ്റ് യുഗത്തിലാണ് നാമെല്ലാം ജീവിക്കുന്നത്. വിവരങ്ങൾ ആവശ്യമായി വരുമ്പോൾ നാം ആദ്യം തിരയുന്നത് ഇന്റർനെറ്റിലായിരിക്കും. 11-ാം ക്ലാസിലെ പരീക്ഷാഫലം വെബ്സൈറ്റിൽ നിങ്ങൾ തിരഞ്ഞ് അറിഞ്ഞിട്ടുണ്ട്. ഓൺ ലൈൻ അപേക്ഷ സമർപ്പണം, വെബ്സൈറ്റിലുള്ള കാര്യങ്ങൾ പരിശോധിക്കൽ, സിനിമ കാണൽ, ധനവ്യാപാരസ്ഥാപനങ്ങളിലെ ക്രയവിക്രയം, ഓൺലൈൻ ക്രയവിക്രയത്തിലൂടെ സാധനങ്ങൾ വാങ്ങൽ തുടങ്ങി വിവിധ ആവശ്യങ്ങൾക്കായി ധാരാളം വെബ്സൈറ്റുകളെ നാം ആശ്രയിക്കാറുണ്ട്. വെബ്പേജുകളുടെ ഒരു കൂട്ടമാണ് വെബ് സൈറ്റ് എന്ന് നമുക്കറിയാം. ചിഹ്നങ്ങൾ (അക്ഷരങ്ങളോ അല്ലാത്തവയോ ആയത്), ചിത്രങ്ങൾ, ഗ്രാഫുകൾ, ചലനാത്മകചിത്രങ്ങൾ, ശബ്ദ വീഡിയോകൾ തുടങ്ങിയവ വെബ് പേജുകളിലുണ്ടാവാം. സ്ഥാപനങ്ങൾ, ഉൽപന്നങ്ങൾ, സേവനങ്ങൾ തുടങ്ങിയവയെക്കുറിച്ചുള്ള വിവരം പ്രദാനം ചെയ്യുന്നതിനാണ് ധാരാളം പേജുകളുള്ള ഒരു വെബ്സൈറ്റ് രൂപകൽപന ചെയ്യുന്നത്. ഈ വെബ്സൈറ്റ് ഇന്റർനെറ്റിൽ നിന്നും നമുക്ക് എങ്ങനെ ലഭിക്കുന്നു? ഇന്റർനെറ്റുമായി ബന്ധിപ്പിച്ചിട്ടുള്ള വെബ് സെർവറുകളിൽ ഈ പേജുകൾ മറ്റുള്ളവർക്ക് ലഭിക്കത്തക്കവിധം സംഭരിച്ച് വച്ചിരിക്കുന്നു. ഈ അധ്യായത്തിൽ ഇന്റർനെറ്റ് വഴിയുള്ള ആശയവിനിമയവും വെബ് സെർവറുകൾക്ക് അതിലുള്ള പ്രാധാന്യവും അവലോകനം ചെയ്യുന്നു. വെബ് സൈറ്റുകൾ വികസിപ്പിക്കുന്നതിനാവശ്യമായ വിവിധ

ടൂളുകളെക്കുറിച്ചും, സാങ്കേതികതകളെക്കുറിച്ചും ഇവിടെ പ്രതിപാദിക്കുന്നുണ്ട്. പരിവർത്തനാത്മക (dynamic) വെബ് പേജുകളുടെ ആശയവും അവ പരിവർത്തനാത്മകം ആക്കുന്നതിന് ഉപയോഗിക്കുന്ന സ്ക്രിപ്റ്റിംഗ് ഭാഷകളുടെ അവലോകനവും ഇവിടെ ചർച്ചചെയ്യുന്നു. ഹൈപ്പർ ടെക്സ്റ്റ് മാർക്കപ്പ് ഭാഷ (HTML) ഉപയോഗിച്ചാണ് വെബ് പേജുകൾ നിർമിക്കുന്നത്. ഇന്റർനെറ്റിന്റെ ഭാഷയായി HTML അറിയപ്പെടുന്നു. ഒരു വെബ് പേജിന്റെ ഉള്ളടക്കം ബ്രൗസർ വിൻഡോയിൽ എങ്ങനെ പ്രദർശിപ്പിക്കണമെന്നാണ് HTML ബ്രൗസറിനോട് പറയുന്നത്. HTML ഉപയോഗിച്ചുള്ള വെബ് പേജ് നിർമിതിയുടെ അടിസ്ഥാന തത്വങ്ങൾ ഈ അധ്യായത്തിൽ നമുക്ക് പരിചയപ്പെടാം.

4.1 വെബ്ബിലൂടെയുള്ള ആശയ വിനിമയം (Communication on the web)

വെബ് ബ്രൗസർ (ക്ലയന്റ്) ഇന്റർനെറ്റിലുള്ള ഒരു വെബ് സൈറ്റ് പരിശോധിക്കുമ്പോൾ എന്തെല്ലാം കാര്യങ്ങൾ നടക്കുന്നുവെന്ന് 11-ാം ക്ലാസ്സിലെ അധ്യായം 12 -ൽ നാം പഠിച്ചതാണ്. അനുബന്ധ ഇന്റർനെറ്റ് പ്രോട്ടോകോൾ (IP) മേൽവിലാസത്തിനായി URL (യൂണിഫോം റിസോഴ്സ് ലൊക്കേറ്റർ)-നെ DNS (ഡൊമൈൻ നെയിം സിസ്റ്റം) സെർവറിലേയ്ക്ക് അയയ്ക്കുകയും തുടർന്ന് ലഭിക്കുന്ന IP മേൽവിലാസം ഉപയോഗിച്ച് ബ്രൗസർ സെർവറുമായി ബന്ധം സ്ഥാപിക്കുകയും ചെയ്യുന്നു. സെർവർ കമ്പ്യൂട്ടറിലുള്ള സെർവർ സോഫ്റ്റ്‌വെയർ ഈ ആവശ്യം പ്രോസസ് ചെയ്ത് പ്രദർശിപ്പിക്കേണ്ട വെബ് പേജിനെ ക്ലയന്റിന് അയച്ച് കൊടുക്കുന്നു. ക്ലയന്റ് ബ്രൗസർ വെബ് പേജ് ക്രമമായി കാണിക്കുന്നു.

വെബ് ഉപയോഗിച്ചുള്ള ആശയവിനിമയത്തിന്, കമ്പ്യൂട്ടറുകൾ അല്ലെങ്കിൽ ഉപകരണങ്ങൾ പരസ്പരം തിരിച്ചറിയേണ്ടതുണ്ട്. ട്രാൻസ്മിഷൻ കൺട്രോൾ പ്രോട്ടോകോൾ/ ഇന്റർനെറ്റ് പ്രോട്ടോകോൾ (TCP/IP) എന്ന ഒറ്റ പ്രോട്ടോകോൾ തന്നെ എല്ലാ ഉപകരണങ്ങളും അനുസരിക്കുന്നത് കൊണ്ടാണ് ഇത് സാധ്യമാകുന്നത്. TCP/IP പ്രോട്ടോകോളുകളെക്കുറിച്ചും അവയുടെ പ്രവർത്തനത്തെക്കുറിച്ചും 11-ാം ക്ലാസിലെ കമ്പ്യൂട്ടർ ശൃംഖലകൾ (കമ്പ്യൂട്ടർ നെറ്റ്‌വർക്ക്സ്) എന്ന അധ്യായത്തിൽ നാം വിശദമായി ചർച്ച ചെയ്തിട്ടുണ്ട്. അയയ്ക്കേണ്ട ഡാറ്റയെ, TCP/IP പ്രോട്ടോകോൾ വിഘടിപ്പിച്ച് ചെറിയ കൂടുകളിലാക്കി സ്വീകർത്താവിന്റെ കമ്പ്യൂട്ടർ വിലാസത്തോടൊപ്പം ചേർക്കുന്നു. ഇന്റർനെറ്റ് പ്രോട്ടോകോളിന്റെ സഹായത്താൽ, റൂട്ടറുകൾ (വഴികാട്ടുന്ന ഉപകരണങ്ങൾ) ഈ ഡാറ്റ ഉൾക്കൊള്ളുന്ന കൂടുകളെ അവ എത്തേണ്ട കമ്പ്യൂട്ടറിലെത്തിക്കുന്നു. അയക്കുന്നയാൾ (പ്രേഷിതൻ) ഡാറ്റയുടെ ഒരു കൂട് സ്വീകർത്താവിൽ എത്തിക്കുന്നതെങ്ങനെ എന്ന് ചിത്രം 4.1 ൽ കാണിച്ചിരിക്കുന്നു.



ചിത്രം 4.1: ഡാറ്റ ഉൾക്കൊള്ളുന്ന കൂടുകൾ പ്രേഷിതനിൽ നിന്ന് സ്വീകർത്താവിലേക്ക് വഴികാട്ടി അയക്കുന്നു

വെബ്സൈറ്റ് പരിശോധിക്കൽ, ഇ-മെയിൽ അയക്കൽ തുടങ്ങി പലതരത്തിലുള്ള ആശയവിനിമയങ്ങൾ ഇന്റർനെറ്റിൽ നടക്കുന്നു. വെബ്സൈറ്റുകൾക്ക് വേണ്ടിയുള്ള തിരയൽ HTTP (ഹൈപ്പർ ടെക്സ്റ്റ് ട്രാൻസ്മിഷൻ പ്രോട്ടോകോൾ) ഉപയോഗിച്ചും ഇ-മെയിൽ വഴിയുള്ള ആശയവിനിമയം SMTP (സിമ്പിൾ മെയിൽ ട്രാൻസ്ഫർ പ്രോട്ടോകോൾ) ഉപയോഗിച്ചുമാണ് നടക്കുന്നതെന്ന് നാം പഠിച്ചിട്ടുണ്ട്. താഴെ തലത്തിലുള്ള ഇന്റർനെറ്റ് പ്രോട്ടോകോളിന്റെ ഏറ്റവും മുകളിലായാണ് ഈ രണ്ട് പ്രോട്ടോകോളുകളും പ്രവർത്തിക്കുന്നത്. ഇന്റർനെറ്റ് വഴിയുള്ള ആശയവിനിമയം ലളിതമാക്കുന്നത് ഇന്റർനെറ്റ് പ്രോട്ടോകോളാണ്. ഈ പ്രോട്ടോകോൾ തന്നെ ഉപയോഗിക്കുന്നതുകൊണ്ട് ധാരാളം നേട്ടങ്ങളുണ്ട്. വിവിധതരം ഡാറ്റ കൈകാര്യം ചെയ്യുന്നതിനായി റൂട്ടറുകൾ പ്രത്യേകം പ്രോഗ്രാം ചെയ്യേണ്ടിവരുന്നില്ല. കൊണ്ടുപോകേണ്ട ഡാറ്റ ശ്രദ്ധിക്കാതെ അവ ഉൾക്കൊള്ളുന്ന കൂടുകൾ എത്തിക്കേണ്ട മേൽവിലാസത്തെക്കുറിച്ച് മാത്രം ശ്രദ്ധിച്ചാൽ മതിയാകും. കൂടുകളിലെ ഡാറ്റ ഉൾക്കൊള്ളുന്ന ഭാഗം സുതാര്യമായതിനാൽ, പുതിയ പ്രോട്ടോകോളുകൾ രൂപപ്പെടുത്തി ഇന്റർനെറ്റിൽ ഉപയോഗിക്കുവാനുള്ള സ്വാതന്ത്ര്യം നമുക്ക് ലഭിക്കുന്നു. TCP/IP യുടെ ഈ സുതാര്യതയും വഴക്കവുമാണ്, സന്ദേശങ്ങൾ കൈകാര്യം ചെയ്യുന്നതിനുള്ള സമൂഹമാധ്യമ സൈറ്റുകൾ, ശബ്ദം, വീഡിയോ തുടങ്ങിയവ ഉള്ളടക്കങ്ങളായി വരുന്ന സൈറ്റുകൾ, സുരക്ഷിതപണമിടപാടുകൾക്കായി ധനകാര്യസ്ഥാപനങ്ങൾ ഉപയോഗിക്കുന്ന സൈറ്റുകൾ തുടങ്ങിയവയ്ക്കായി പ്രോട്ടോകോളുകൾ നിർമിക്കുന്നതിന് സഹായിച്ചത്. ഇന്റർനെറ്റിന്റെ സാമ്പത്തിക വിജയത്തിന് കാരണം ഇതായിരുന്നു.

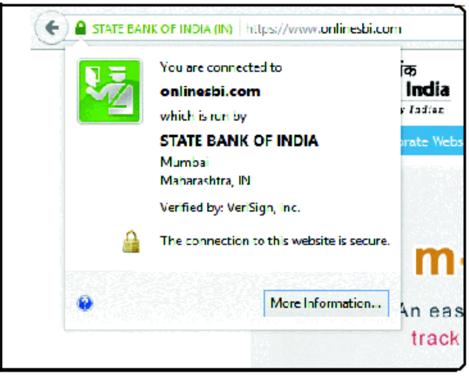
വെബ് വഴിയുള്ള ആശയവിനിമയം ക്ലയന്റ് ബ്രൗസറിൽ നിന്ന് വെബ് സെർവറിലേക്കെന്നും, വെബ് സെർവറിൽനിന്ന് വെബ് സെർവറിലേക്കെന്നും, രണ്ടായി തരംതിരിച്ചിരിക്കുന്നു. ഇന്റർനെറ്റ് വഴിയുള്ള ആശയ വിനിമയത്തിന് ആധികാരികതയും സുരക്ഷിതത്വവും അത്യാവശ്യമാണ്. വെബിലുള്ള ഒരു കമ്പ്യൂട്ടർ/സെർവർ യഥാർഥ അവകാശിയായ കമ്പ്യൂട്ടറാണെന്ന് ഉറപ്പ് വരുത്തുന്ന പ്രക്രിയയാണ് ആധികാരികത ഉറപ്പിക്കൽ (authentication). ഇന്റർനെറ്റ് വഴിയുള്ള ആശയവിനിമയത്തിന് സുരക്ഷിതത്വം ലഭിക്കുന്നതിനാൽ, കമ്പ്യൂട്ടറിലെ വിവരങ്ങൾ ചോർത്തുന്നവർക്ക് (ഹാക്കേഴ്സിന്) സന്ദേശങ്ങൾ പിടിച്ചെടുക്കുവാനോ, രൂപമാറ്റം വരുത്തുവാനോ സാധിക്കുകയില്ല.

4.1.1 ക്ലയന്റിൽ നിന്നും വെബ് സെർവറിലേക്കുള്ള ആശയവിനിമയം (Client to web server communication)

ക്ലയന്റിൽ നിന്ന് വെബ് സെർവറിലേക്കുള്ള ആശയവിനിമയത്തിന് സാധാരണഗതിയിൽ ആധികാരികത ഉറപ്പിക്കേണ്ടതില്ല. എന്നാൽ, ധനകാര്യസ്ഥാപനങ്ങളുടെ ഉപയോഗത്തിനോ, ഇ-മെയിൽ സേവനത്തിനോ ആണെങ്കിൽ യൂസർ നാമവും (യൂസർ നെയിം) പാസ്‌വേഡും സെർവറിലേക്ക് അയക്കേണ്ടതുണ്ട്. സുരക്ഷിതത്വകാരണങ്ങളാൽ, ഈ വിവരങ്ങൾ സാധാരണ ചിഹ്നങ്ങളായി/അക്ഷരങ്ങളായി സെർവറിലേയ്ക്ക് അയയ്ക്കാറില്ല. അഥവാ അങ്ങനെ ചെയ്താൽ നുഴഞ്ഞ് കയറ്റക്കാർക്ക് (ഹാക്കേഴ്സിന്) യൂസർ നാമവും പാസ്‌വേഡും മോഷ്ടിക്കാനാകും. അത്തരം സാഹചര്യങ്ങളിൽ യൂസർ നാമ

വും പാസ്‌വേഡും രൂപമാറ്റം വരുത്തി (എൻക്രിപ്റ്റ് ചെയ്ത്) സെർവറിലേക്ക് അയക്കാനായി നാം HTTPS (ഹൈപ്പർ ടെക്സ്റ്റ് ട്രാൻസ്ഫർ പ്രോട്ടോകോൾ സെക്യൂർ) എന്ന സാങ്കേതികവിദ്യ ഉപയോഗിക്കുന്നു. ഇന്റർനെറ്റിലുള്ള കമ്പ്യൂട്ടറുകൾ തമ്മിൽ ഇത്തരത്തിൽ ബന്ധം സ്ഥാപിക്കാൻ ഉതകുന്ന സുരക്ഷിത സാങ്കേതികവിദ്യയായ SSL (സെക്യൂർഡ് സോക്കറ്റ് ലെയർ) ഉപയോഗിച്ച് ആണ് HTTPS പ്രവർത്തിക്കുന്നത്. HTTP യ്ക്ക് സുരക്ഷിതത്വം പ്രദാനം ചെയ്യുന്നത് HTTPS ആണ്. SSL പ്രോട്ടോകോൾ സ്വകാര്യത ഉറപ്പ് വരുത്തുന്നതോടൊപ്പം മറ്റൊരു വെബ്സൈറ്റിനും ഉപയോക്താവിന്റെ ലോഗിൻ അക്കൗണ്ടിൽ ആൾമാറാട്ടം നടത്താനോ അയച്ച വിവരം മാറ്റം വരുത്താനോ സാധിക്കാത്തവിധം സുരക്ഷ ഉറപ്പ് വരുത്തുകയും ചെയ്യുന്നു.

ചിത്രം 4.2(a) : ക്ലയന്റ്-സെർവർ ആധികാരികത ഉറപ്പിക്കൽ പ്രവർത്തനം

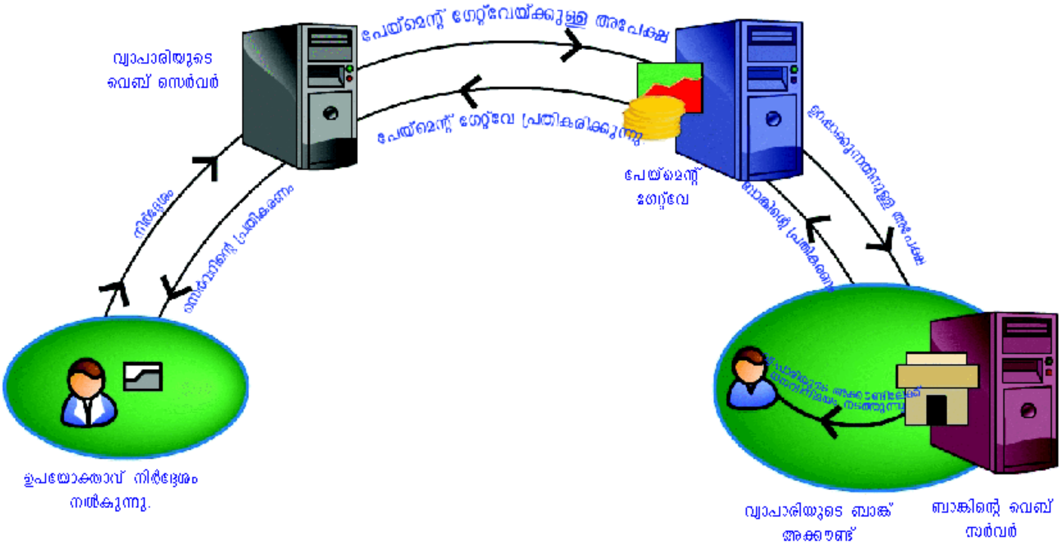


ചിത്രം 4.2(b): SBI വെബ് സൈറ്റിന്റെ ആധികാരികത ഉറപ്പിക്കൽ

ബ്രൗസർ ഒരു സുരക്ഷിത പേജ് ആവശ്യപ്പെടുമ്പോൾ, സെർവർ ആദ്യം അതിന്റെ SSL യോഗ്യതാപത്രം തിരിച്ച് നൽക്കുന്നു. തുടർന്ന് ഈ സാക്ഷ്യപത്രം ബന്ധപ്പെട്ട സാക്ഷ്യപത്ര അധികാരികളുമായി ഒത്തുനോക്കി സാധുത ഉറപ്പ് വരുത്തുന്നു. സാക്ഷ്യപത്ര അധികാരിക്കു നൽകിയ യോഗ്യതാപത്രം സാധുത ഉള്ളതാണെങ്കിൽ, അധികാരി സാക്ഷ്യപ്പെടുത്തുന്നു. ഒരു പ്രത്യേക വെബ്സൈറ്റ് അത് അവകാശപ്പെടുന്ന സ്ഥാപനത്തിന്റേത് തന്നെ ആണെന്ന് ഉറപ്പിക്കലാണ് ഇത്. വെരിസൈൻ ഇൻകോർപ്പറേഷൻ ഒരു സാക്ഷ്യപത്ര അധികാരിയാണ്. ഒരു SSL സാക്ഷ്യപത്രം സാധുത ഉള്ളതായാൽ, ബ്രൗസർ രൂപമാറ്റം വരുത്തിയുള്ള (എൻക്രിപ്റ്റ് ചെയ്ത) ആശയ വിനിമയത്തിന് വേണ്ടി ഒരു നിശ്ചിത സമയം തുടങ്ങിവയ്ക്കുന്നു. ഈ പ്രവർത്തനം ചിത്രം 4.2 (a)-യിൽ ചിത്രീകരിച്ചിരിക്കുന്നു. ഇന്ത്യയിൽ ധനകാര്യസ്ഥാപനങ്ങളുമായുള്ള ഇടപാടുകൾക്ക് ഉപയോഗിക്കുന്ന വെബ്സൈറ്റുകൾ, ക്ലയന്റുകളിൽ നിന്നും രഹസ്യവിവരങ്ങൾ സ്വീകരിക്കുവാൻ HTTPS ഉപയോഗിക്കണമെന്ന് വിവരസാങ്കേതികവിദ്യ നിയമം നിഷ്കർഷിച്ചിട്ടുണ്ട്. നിങ്ങൾ അഡ്രസ് ബാറിലുള്ള ലോക്കിൽ ക്ലിക്ക് ചെയ്താൽ ചിത്രം 4.2 (b)-യിൽ തന്നിട്ടുള്ള സാക്ഷ്യപത്രം കാണാവുന്നതാണ്.

4.1.2 വെബ് സെർവറിൽ നിന്നും വെബ് സെർവറിലേക്കുള്ള ആശയ വിനിമയം (Web server to web server communication)

ചില വെബ് ആപ്ലിക്കേഷനുകളുടെ പ്രവർത്തനത്തിന് വെബ് സെർവറിൽനിന്ന് വെബ് സെർവറിലേക്കുള്ള ആശയവിനിമയം ആവശ്യമാണ്. ഉദാഹരണമായി, ഇന്റർനെറ്റ് വഴി സാധനങ്ങൾ വാങ്ങാൻ ഉപയോഗിക്കുന്ന ഒരു വെബ്സൈറ്റ് സൂക്ഷിച്ചിരിക്കുന്ന വെബ് സെർവറിന് (ഇന്റർനെറ്റിലെ വ്യാപാരിക്ക്/വിൽപ്പനക്കാരന്), ധനകാര്യസ്ഥാപനത്തിന്റെ വെബ് സെർവറിലേക്കും അതുപോലെ തിരിച്ചും രഹസ്യ വിവരങ്ങൾ അയയ്ക്കേണ്ടി വരുന്നു. അത്തരം സന്ദർഭങ്ങളിൽ വ്യാപാരിയുടെയും ധനകാര്യ സ്ഥാപനത്തിന്റെയും വെബ്സർവറുകളുടെ ആധികാരികത ഉറപ്പിക്കേണ്ടതുണ്ട്. സ്വീകരിക്കപ്പെട്ട ഡാറ്റ യഥാർഥ സെർവറുകളിൽ നിന്നുള്ളവയാണോ എന്നുറപ്പിക്കാൻ ഡിജിറ്റൽ സാക്ഷ്യപത്ര



ചിത്രം 4.3 : ഒരറ്റം മുതൽ മറ്റേ അറ്റം വരെയുള്ള ഓൺലൈൻ വ്യവഹാരം

ങ്ങൾ സഹായിക്കുന്നു. ഒരിക്കൽ സെർവറുകളുടെ ആധികാരികത ഉറപ്പിച്ചു കഴിഞ്ഞാൽ, അവ രൂപമാറ്റം വരുത്തിയ (എൻക്രിപ്റ്റ്) ഡാറ്റ ഉപയോഗിച്ച് ആശയവിനിമയം നടത്തുന്നു. ഇന്റർനെറ്റ് വഴി പണം അടയ്ക്കുകയോ/കൈമാറുകയോ ചെയ്യുമ്പോൾ, വ്യാപാരിയുടെ സെർവറിനും ധനകാര്യസ്ഥാപനത്തിന്റെ സെർവറിനും ഇടയ്ക്ക് ഒരു പേയ്മെന്റ് ഗേറ്റ്‌വേ സെർവർ പ്രവർത്തിച്ച് രൂപമാറ്റം (എൻക്രിപ്റ്റ് ചെയ്ത) വരുത്തി ധനവിനിമയം നടത്തുന്നു. ഈ പ്രവർത്തനം ചിത്രം 4.3-ൽ കാണിച്ചിരിക്കുന്നു.

4.2 വെബ് സെർവർ സാങ്കേതികവിദ്യകൾ (Web server technologies)

ഹയർസെക്കന്ററി വിദ്യാഭ്യാസ ഡയറക്ടറേറ്റിന്റെ (DHSE-ഡയറക്ടറേറ്റ് ഓഫ് ഹയർ സെക്കന്ററി എഡ്യൂക്കേഷൻ) ഔദ്യോഗിക വെബ്സൈറ്റായ www.dhsekerala.gov.in സന്ദർശിക്കുമ്പോൾ സംഭവിക്കുന്നത് എന്തൊക്കെയെന്ന് നമുക്ക് നോക്കാം. ആദ്യമായി

വെബ് സൈറ്റിന്റെ ഹോം പേജ് (ഒരു വെബ് സൈറ്റ് പരിശോധിക്കുമ്പോൾ കാണിക്കുന്ന ആദ്യത്തെ പേജ്), DHSE യുടെ സെർവറിൽ നിന്നും നമ്മുടെ (ക്ലയന്റിന്റെ) കമ്പ്യൂട്ടറിലേക്ക് കൈമാറ്റം ചെയ്യുന്നു. സാധാരണയായി, വെബ് പേജിന് അഭ്യർഥന നടത്തുന്ന ഒരു കമ്പ്യൂട്ടറോ മൊബൈൽ ഉപകരണമോ ആയിരിക്കും ക്ലയന്റ്. വെബ് സെർവറിൽ ശേഖരിച്ചു വച്ചിരിക്കുന്ന വെബ് സൈറ്റിലെ പേജുകൾ, ക്ലയന്റിന്റെ അപേക്ഷയ്ക്കനുസരിച്ച്, സെർവർ സോഫ്റ്റ്‌വെയർ വിട്ടു നൽകുന്നു. വെബ് സെർവറിന്റെ സവിശേഷതകളെക്കുറിച്ചും അവ ക്രമീകരിക്കുവാൻ ഉപയോഗിക്കുന്ന വസ്തുക്കളെക്കുറിച്ചും നമുക്ക് ചർച്ച ചെയ്യാം.

4.2.1 വെബ് സെർവർ (Web server)

വെബ്സൈറ്റുകൾ സംഭരിച്ചിരിക്കുന്ന സെർവർ കമ്പ്യൂട്ടറുകളെ സൂചിപ്പിക്കുന്നതിനാണ് പലപ്പോഴും വെബ് സെർവർ എന്ന പദം ഉപയോഗിക്കുന്നത്. ഒരു സെർവർ കമ്പ്യൂട്ടറിനെ വെബ്സെർവറാക്കാനായി അതിൽ സ്ഥാപിച്ചിട്ടുള്ള വെബ് സെർവർ സോഫ്റ്റ്‌വെയറിനെയും ഈ പദം ഉപയോഗിച്ച് സൂചിപ്പിക്കാറുണ്ട്. വെബ്സെർവർ കമ്പ്യൂട്ടറിന്റെയും വെബ് സെർവർ സോഫ്റ്റ്‌വെയറിന്റെയും സവിശേഷതകൾ എന്തൊക്കെയാണെന്ന് നമുക്ക് നോക്കാം.

ചിത്രം 4.4 : ഒരു ഡാറ്റാ സെന്റർ

ഇന്റർനെറ്റ് ഉപയോക്താക്കൾക്ക് ഇ-മെയിൽ, ബ്ലോഗ് തുടങ്ങിയ സേവനങ്ങളോ വെബ് പേജുകളോ വിട്ട് നൽകാൻ വെബ്സെർവർ സഹായിക്കുന്നു. WWW, ഇ-മെയിൽ തുടങ്ങിയ സേവനങ്ങൾ ഇന്റർനെറ്റിലൂടെ നൽകുന്നതിനു വേണ്ടി ഒരു സെർവർ ഓപ്പറേറ്റിംഗ് സിസ്റ്റവും സെർവർ സോഫ്റ്റ്‌വെയറും പ്രവർത്തനക്ഷമമാക്കിയിട്ടുള്ള ഒരു സെർവർ കമ്പ്യൂട്ടറാണ് വെബ്സെർവർ. ഇത് വളരെ കാര്യക്ഷമതയുള്ളതും എപ്പോഴും പ്രവർത്തിക്കുന്നതും ഉയർന്ന ബാന്റ് വിഡ്ത്തോടുകൂടിയ ഇന്റർനെറ്റ് ബന്ധമുള്ളതും ആയിരിക്കും. ലോകമെമ്പാടുമുള്ള ഇന്റർനെറ്റ് ഉപയോക്താക്കൾക്ക് ഇതിലുള്ള വെബ് സൈറ്റുകളും സേവനങ്ങളും ഉപയോഗിക്കാനാകും. ഒരു വെബ് സെർവറിൽ ഒന്നോ അതിലധികമോ പ്രൊസസ്സറുകളും, ഉയർന്ന വേഗതയുള്ള റാമും, വർധിച്ച കാര്യക്ഷമതയുള്ള ഹാർഡ് ഡിസ്കും, ദ്രുതഗതിയിലുള്ള ആശയവിനിമയത്തിന് ഉപകരിക്കുന്ന ഈഥർനെറ്റ് കാർഡുമൊക്കെയുണ്ടാകും. വേഗതയുള്ള ഇന്റർനെറ്റ് ബന്ധം ഉറപ്പാക്കാനും, ആവശ്യാനുസരണം വൈദ്യുതി എത്തിക്കുന്നതിനും മറ്റുമായി ഒരു വെബ് സെർവർ സാധാരണ ഗതിയിൽ ഡാറ്റ സെന്ററുകളിലാണ് സ്ഥാപിക്കാറുള്ളത്. സ്ഥാപനങ്ങളുടെ സെർവറുകളും അനുബന്ധ കമ്പ്യൂട്ടറുകളും സ്ഥാപിക്കുന്നതിന് മാത്രമായി വിനിയോഗിച്ചിട്ടുള്ള ഭൗതികസ്ഥലങ്ങളാണ് ഡാറ്റസെന്ററുകൾ. ക്ലയന്റിന് വേണ്ടി, നിയുക്ത പ്രവൃത്തിയുമായി ബന്ധപ്പെട്ട നിർണായക ഡാറ്റ സംഭരിക്കുന്നതിനും, പ്രോസസ്സ് ചെയ്യുന്നതിനും, പ്രയോജനപ്പെടു

ത്തുന്നതിനും വേണ്ടിയാണ് ഡാറ്റസെന്ററുകൾ ഉപയോഗിക്കുന്നത്. വൈദ്യുതവിതരണത്തിന് തടസം വരാതിരിക്കാനുള്ള സംവിധാനം, ശീതീകരണ സംവിധാനം, വേഗമേറിയ നെറ്റ്‌വർക്ക്, സുരക്ഷിതത്വക്രമീകരണങ്ങൾ തുടങ്ങിയ സൗകര്യങ്ങൾ ഡാറ്റസെന്ററിന് ആവശ്യമാണ്. ഇത്തരത്തിലുള്ള ഒരു ഡാറ്റസെന്റർ ചിത്രം 4.4-ൽ കാണിച്ചിരിക്കുന്നു. ചിത്രം 4.5-ൽ കാണുന്ന തരത്തിലുള്ള എളുപ്പത്തിൽ പരിപാലിക്കാവുന്നതും വാതായനങ്ങളോട് കൂടിയതുമായ പ്രത്യേകതരം ചട്ടക്കൂടുകളിൽ ഉറപ്പിച്ച ധാരാളം സെർവറുകൾ ഡാറ്റ സെന്ററുകളിലുണ്ടാവും.

ചിത്രം 4.5 : പ്രത്യേകതരം ചട്ടക്കൂടുകളിലുറപ്പിച്ചിട്ടുള്ള സെർവറുകൾ

വിവിധ ലിനക്സ് പതിപ്പുകൾ (റെഡ്ഹാറ്റ്, ഓപ്പൺസൂസ്, ഡിബിയൻ, ഉബുണ്ടു തുടങ്ങിയവ), മൈക്രോ സോഫ്റ്റ് വിൻഡോസ് സെർവർ, ഫ്രീ ബി.എസ്.ഡി, ഒറാക്കിൾ സൊളാരീസ് എന്നിവ പ്രചാരത്തിലുള്ള സെർവർ ഓപ്പറേറ്റിംഗ് സിസ്റ്റങ്ങളാണ്. ഒരു സെർവർ സ്ഥാപിച്ച് അതിൽ ഒരു വെബ് സെർവർ സോഫ്റ്റ്‌വെയർ ഉൾക്കൊള്ളിച്ച് ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന് അനുയോജ്യമായ രീതിയിൽ അതിനെ രൂപപ്പെടുത്തിയെടുക്കേണ്ടതുണ്ട്. സംഭരിച്ച് വച്ചിട്ടുള്ള ഫയലുകൾ ഉപയോക്താക്കൾക്ക് സമയബന്ധിതമായി നൽകുന്നതിനു വേണ്ടി, ഹൈപ്പർ ടെക്സ്റ്റ് ട്രാൻസ്ഫർ പ്രോട്ടോകോളിന്റെ സഹായത്താൽ ക്ലയന്റ്-സെർവർ മോഡലിൽ പ്രവൃത്തിക്കുന്ന ഒരു പ്രോഗ്രാമാണ് വെബ് സെർവർ സോഫ്റ്റ്‌വെയർ. ഉപയോക്താക്കൾ നൽകുന്ന അപേക്ഷയ്ക്കനുസരിച്ച്, വെബ് സെർവർ സോഫ്റ്റ്‌വെയറിന്റെ സഹായത്തോടെ ഈ ഫയലുകൾ വെബ് പേജുകളാക്കി ക്ലൈയന്റിന് അയയ്ക്കുന്നു. തുടർന്ന് ക്ലയന്റ് ബ്രൗസറിൽ ഇവ കാണുന്നു. അപ്പാഷെ സെർവർ, മൈക്രോസോഫ്റ്റ് ഇന്റർനെറ്റ് ഇൻഫർമേഷൻ സർവീസസ്, ഗൂഗിൾ വെബ്സെർവർ (GWS), എൻജിൻ-എക്സ് തുടങ്ങിയവ പ്രചാരത്തിലുള്ള വെബ്സെർവർ പാക്കേജുകളാണ്.

സോഫ്റ്റ്‌വെയർ സ്ഥാപിച്ച് പ്രവർത്തന യോഗ്യമാക്കിയശേഷം അതോടൊപ്പം സോഫ്റ്റ്‌വെയർ പാക്കേജുകളായ FTP (ഫയൽ ട്രാൻസ്ഫർ പ്രോട്ടോകോൾ), ഇ-മെയിൽ, DNS, ഡാറ്റ ബേസ് തുടങ്ങിയവ കൂടി വെബ്സെർവറിൽ ഉൾപ്പെടുത്താവുന്നതാണ്. വെബ് സൈറ്റിന്റെ സുരക്ഷിതത്വത്തിന് രൂപംകൊടുക്കുക, ഒന്നിലധികം വെബ് സൈറ്റുകളുടെ നിർമാണം സാധ്യമാക്കുക എന്നിങ്ങനെയുള്ള വ്യവസ്ഥകൾ വെബ് സെർവർ സോഫ്റ്റ്‌വെയർ തന്നെ കൈകാര്യം ചെയ്യും.

4.2.2 സോഫ്റ്റ്‌വെയർ പോർട്ടുകൾ (Software ports)

കമ്പ്യൂട്ടർ സിസ്റ്റത്തിലെ ഹാർഡ്‌വെയർ പോർട്ടുകളെക്കുറിച്ച് 11-ാം ക്ലാസിൽ നാം പഠിച്ചിട്ടുണ്ട്. പുറമേ നിന്നുള്ള ഉപകരണങ്ങൾ കമ്പ്യൂട്ടറുമായി ബന്ധിപ്പിക്കുവാൻ ഹാർഡ്‌വെ

യർ പോർട്ടുകൾ ഉപയോഗിക്കുന്നു. ഈ പോർട്ടുകൾ ഉപയോഗിച്ചാണ് ഇത്തരം ഉപകരണങ്ങൾ കമ്പ്യൂട്ടറുമായി ആശയവിനിമയം നടത്തുന്നത്. ഉദാഹരണമായി VGA പോർട്ട് മോണിട്ടറുമായി ബന്ധിപ്പിക്കുന്നതിനും, PS/2 പോർട്ട് കീബോർഡ്/മൗസ് എന്നിവയുമായി ബന്ധിപ്പിക്കുന്നതിനും ഉപയോഗിക്കുന്നു. അതുപോലെ, ഒരു സോഫ്റ്റ്‌വെയർ പോർട്ട് ഉപയോഗിച്ചുകൊണ്ട് ക്ലയന്റ് കമ്പ്യൂട്ടറിനെ സെർവറുമായി ബന്ധിപ്പിച്ച്, അതിലെ സേവനങ്ങളായ HTTP, FTP, SMTP തുടങ്ങിയവ ഉപയോഗിക്കാനാകും. പോർട്ടുകളെ തമ്മിൽ തിരിച്ചറിയുന്നതിനായി അവയ്ക്ക് അതിവിശിഷ്ടമായ സംഖ്യാനാമം നൽകുന്നു. ഇ-മെയിൽ, ഫയൽ കൈമാറ്റം തുടങ്ങിയ ഒരേ സെർവർ കമ്പ്യൂട്ടറിൽ തന്നെ നടക്കുന്ന വ്യത്യസ്ത സേവനങ്ങളെ തിരിച്ചറിയുക എന്നതാണ് സോഫ്റ്റ്‌വെയർ പോർട്ടിന്റെ ഉദ്ദേശ്യം. വ്യത്യസ്ത പോർട്ട് സംഖ്യാനാമങ്ങളുപയോഗിച്ച്, സെർവറിൽ ലഭ്യമായ ഓരോ സേവനവും തുടങ്ങാനും, ഉപയോഗിക്കാനും/എടുക്കാനും സാധിക്കുന്നു. പോർട്ട് സംഖ്യാനാമം എന്നത് ഒരു 16-ബിറ്റ് സംഖ്യയാണ്. അത് കമ്പ്യൂട്ടറിന്റെ IP മേൽവിലാസത്തോടൊപ്പം (URL-ന് ഒപ്പം) ചേർത്തുപയോഗിച്ച് സെർവറിൽ ലഭ്യമായ ഒരു പ്രത്യേക സേവനവുമായി ആശയവിനിമയം സാധ്യമാക്കുന്നു. ഒരു വെബ്സൈറ്റിലുള്ള സേവനങ്ങൾ താഴെ തന്നിരിക്കുന്ന വിധം ഉപയോഗിക്കാൻ സാധിക്കും.

http://google.co.in:80

ഇവിടെ http എന്നത് പ്രോട്ടോകോളും, google.co.in എന്നത് ഡൊമൈൻ നാമവും 80 എന്നത് പോർട്ടിന്റെ സംഖ്യാനാമവുമാണ്. പട്ടിക 4.1-ൽ അറിയപ്പെടുന്ന ചില പോർട്ട് സംഖ്യാനാമങ്ങളും അനുബന്ധ സേവനങ്ങളും കൊടുത്തിരിക്കുന്നു.

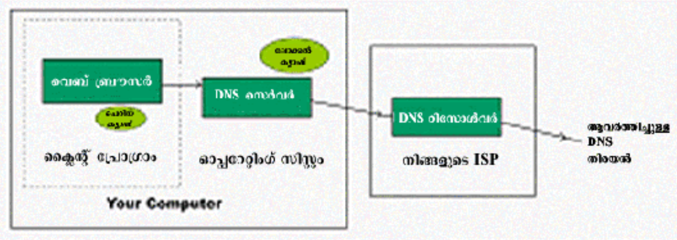
തനത് പോർട്ട് സംഖ്യ	സേവനം
20 & 21	ഫയൽ കൈമാറുന്നതിനുള്ള ഫയൽ ട്രാൻസ്ഫർ പ്രോട്ടോകോൾ (FTP)
22	സെക്യൂർ ഷെൽ (SSH)
25	ലഘു സന്ദേശങ്ങൾ കൈമാറുന്നതിനുള്ള സിമ്പിൾ മെയിൽ ട്രാൻസ്ഫർ പ്രോട്ടോകോൾ (SMTP)
53	ഡൊമൈൻ നെയിം സിസ്റ്റം (DNS)
80	ഹൈപ്പർ ടെക്സ്റ്റ് ട്രാൻസ്ഫർ പ്രോട്ടോകോൾ (HTTP)
110	പോസ്റ്റ് ഓഫീസ് പ്രോട്ടോകോൾ (POP3)
443	HTTP സെക്യൂർ (HTTPS)

പട്ടിക 4.1 : പോർട്ടുകളും അവയുടെ സേവനങ്ങളും

4.2.3 DNS സെർവറുകൾ (DNS servers)

ചിത്രം 4.6 : ലോകമെമ്പാടുമുള്ള റൂട്ട് സെർവറുകൾ

11-ാം ക്ലാസിൽ കമ്പ്യൂട്ടർ ശൃംഖലകൾ എന്ന അധ്യായത്തിൽ ഡൊമൈൻ നാമകരണ രീതിയെക്കുറിച്ച് നാം പഠിച്ചിട്ടുണ്ട്. ഡൊമൈൻ നാമങ്ങളുടെ ഡാറ്റബേസും അനുബന്ധ IP മേൽവിലാസങ്ങളും ഉള്ള ഒരു പ്രത്യേക നെറ്റ്‌വർക്കിങ് സോഫ്റ്റ്‌വെയർ ഓരോ DNS സെർവറിലും പ്രവൃത്തിക്കുന്നുണ്ടാവും. ഡൊമൈൻ നാമകരണ സമ്പ്രദായം ഒരു DNS സെർവറിൽ പ്രവർത്തിക്കുകയും ക്ലയന്റ് കമ്പ്യൂട്ടർ നൽകുന്ന അപേക്ഷയ്ക്കനുസരിച്ച് ഡൊമൈൻ നാമത്തിന്റെ IP മേൽവിലാസം തിരിച്ച് നൽകുകയും ചെയ്യുന്നു.



ചിത്രം 4.7 : DNS തിരയൽ

ഒരു ഡൊമൈൻ നാമകരണ സമ്പ്രദായത്തിൽ ധാരാളം DNS സെർവറുകളുണ്ടായിരിക്കും. ഇവ സ്ഥാനക്രമമനുസരിച്ച് ക്രമീകരിച്ചിരിക്കും. .com, .org തുടങ്ങിയ ഉയർന്നതും മുകൾ തട്ടിലുള്ളതുമായ ഡൊമൈനുകളോ, രാജ്യങ്ങളെ സൂചിപ്പിക്കുന്ന ഡൊമൈൻ നാമങ്ങളായ .in അല്ലെങ്കിൽ .uk തുടങ്ങിയവയോ ഉൾപ്പെടുന്ന 13 റൂട്ട് സെർവറുകളായിരിക്കും ഏറ്റവും മുകളിലുള്ളത്. ചിത്രം 4.6-ൽ കാണിച്ചിരിക്കുന്നത് പോലെ ലോകമെമ്പാടുമായി പലസ്ഥലങ്ങളിൽ റൂട്ട് സെർവറുകളുടെ ധാരാളം പകർപ്പുകൾ സ്ഥാപിച്ചിട്ടുണ്ടാകും. മറ്റെല്ലാ DNS സെർവറുകളും, സ്ഥാനക്രമത്തിന്റെ താഴെത്തലങ്ങളിൽ പ്രതിഷ്ഠിച്ചിരിക്കും. DNS എങ്ങനെയാണ് ഡൊമൈൻ നാമത്തിന്റെ IP മേൽവിലാസം തിരഞ്ഞെടുക്കുന്നതെന്ന് നമുക്ക് നോക്കാം.

കേരളത്തിലെ പോലീസ് വകുപ്പിന്റെ വെബ് സൈറ്റ് നാം സന്ദർശിക്കുന്നു എന്ന് കരുതുക. www.keralapolice.org എന്ന പേരിലുള്ള പോലീസ് വകുപ്പിന്റെ ഡൊമൈൻ നാമം നമുക്ക് ബ്രൗസറിൽ ടൈപ്പ് ചെയ്യാം. DNS എങ്ങനെ IP മേൽവിലാസം കണ്ടെത്തുന്നു. എന്ന് താഴെ തന്നിട്ടുള്ള ഘട്ടങ്ങൾ വിശദീകരിക്കുന്നു. ചിത്രം 4.7 കാണുക.

1. എല്ലാ ബ്രൗസറുകളും അതിന്റെ ക്യാഷ് മെമ്മറിയിൽ സമീപകാലത്ത് സന്ദർശിച്ച IP മേൽവിലാസങ്ങൾ സൂക്ഷിക്കുന്നുണ്ടാകും. അതിനാൽ, ബ്രൗസർ ആദ്യം പ്രാദേശിക മെമ്മറി (മിനി ക്യാഷ്) പരിശോധിച്ച് IP മേൽവിലാസമുണ്ടോ എന്ന് ഉറപ്പ് വരുത്തുന്നു. ഉണ്ടെങ്കിൽ അതുപയോഗിക്കുന്നു.

2. ബ്രൗസറിന്റെ ക്യാഷ് മെമ്മറിയിൽ IP മേൽവിലാസം ഇല്ലെങ്കിൽ അത് ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന്റെ പ്രാദേശിക ക്യാഷ് മെമ്മറി പരിശോധിക്കുന്നു.

3. അവിടെയുമില്ലെങ്കിൽ ISP-യുടെ പ്രാദേശിക DNS സെർവറിൽ പരിശോധിക്കുന്നു.

4. ISP യുടെ DNS സെർവറിലും ഡൊമൈൻ നാമമില്ല എങ്കിൽ ISP യുടെ DNS സെർവർ, റൂട്ട് സെർവറിൽ തുടങ്ങി ആവർത്തിച്ചുള്ള തിരച്ചിലിലൂടെ അത് കണ്ടെത്തുന്നു.
5. ISP-യുടെ DNS സെർവർ, IP മേൽവിലാസം ബ്രൗസറിന് തിരിച്ച് നൽകുന്നു.
6. www.keralapolice.gov യുടെ IP മേൽവിലാസം ഉപയോഗിച്ചുകൊണ്ട് ബ്രൗസർ, വെബ് സെർവറുമായി ബന്ധപ്പെടുകയും, ബ്രൗസർ വിൻഡോയിൽ വെബ് പേജ് കാണിക്കുകയും ചെയ്യുന്നു. IP മേൽവിലാസം കിട്ടിയില്ലായെങ്കിൽ 'സെർവർ കണ്ടെത്തിയില്ല' എന്ന സന്ദേശം ബ്രൗസർ വിൻഡോയിലെത്തിക്കുന്നു.

ഡൊമൈൻ നാമങ്ങളുടെയും അനുബന്ധ മേൽവിലാസങ്ങളുടെയും മൊത്തമായുള്ള ഡാറ്റാബേസ് ലോകമെമ്പാടുമുള്ള 13 റൂട്ട് സെർവറുകളാണ് കൈകാര്യം ചെയ്യുന്നത്. ഓരോ റൂട്ട് സെർവറും പലരാജ്യങ്ങളിൽ സ്ഥാപിച്ചിട്ടുള്ള സെർവറുകളുടെ ഒരു ശൃംഖലയിലാണുള്ളത്. അവയ്ക്ക് A, B, C, D, E, F, G, H, I, J, K, L, M എന്നിങ്ങനെ നാമകരണം നൽകിയിരിക്കുന്നു. ICANN, NASA (നാഷണൽ ഏറോനോട്ടിക്സ് ആന്റ് സ്പേസ് അഡ്മിനിസ്ട്രേഷൻ), മേരിലാന്റ് സർവകലാശാല, വെരിസൈൻ ഇൻകോർപ്പറേഷൻ തുടങ്ങിയ സ്ഥാപനങ്ങളാണ് ഈ സെർവറുകളെ പരിപാലിക്കുന്നത്. റൂട്ട് സെർവറുകളെ പരിപാലിക്കുന്ന സ്ഥാപനങ്ങൾ ചേർന്നാണ് ICANN (ഇന്റർനെറ്റ് കോർപ്പറേഷൻ ഫോർ അസൈൻഡ് നെയിംസ് ആന്റ് നമ്പേഴ്സ്) റൂട്ട് സെർവർ സമ്പ്രദായ ഉപദേശക സമിതി ഉണ്ടായിരിക്കുന്നത്. ഇവരാണ് ഇന്റർനെറ്റിലെ റൂട്ട് സെർവർ സമ്പ്രദായത്തിന്റെ പ്രവർത്തനം, കാര്യനിർവഹണം, സുരക്ഷിതത്വം, സമ്പൂർണത ഉറപ്പാക്കൽ തുടങ്ങിയവയുമായി ബന്ധപ്പെട്ട ഉപദേശങ്ങൾ ICANN ന് നൽകുന്നത്. ഇന്ത്യയിൽ NIXI (നാഷണൽ ഇന്റർനെറ്റ് എക്സ്ചേഞ്ച് ഓഫ് ഇന്ത്യ) യാണ് മുംബൈ (I റൂട്ട്), ഡൽഹി (K റൂട്ട്), ചെന്നൈ (F റൂട്ട്) എന്നിവിടങ്ങളിൽ റൂട്ട് സെർവറുകൾ നിർമിച്ച് നൽകിയിട്ടുള്ളത്.

നൂറ് കണക്കിന് കമ്പ്യൂട്ടറുകളും ഉപകരണങ്ങളും ഇന്റർനെറ്റുമായി ബന്ധിപ്പിച്ചിട്ടുള്ള വിദ്യാഭ്യാസസ്ഥാപനങ്ങൾ, സർക്കാർ വകുപ്പുകൾ, സോഫ്റ്റ്‌വെയർ നിർമാണശാലകൾ തുടങ്ങിയവർ അവരുടെ ഇൻട്രാനെറ്റിൽ സ്വന്തം DNS സെർവർ സ്ഥാപിച്ചിട്ടുണ്ടാകും. ഉപയോക്താക്കൾ സ്ഥിരമായി ഉപയോഗിക്കുന്ന ഡൊമൈൻനാമങ്ങളും അവയുടെ IP മേൽവിലാസങ്ങളും ഉൾപ്പെടുന്ന ഒരു പട്ടിക ഈ പ്രാദേശിക DNS സെർവറിലുണ്ടാകും. പുതിയ ഡൊമൈൻ നാമങ്ങൾ ഉൾപ്പെടുത്തിക്കൊണ്ട് ഈ പട്ടിക സമയാസമയം നവീകരിക്കാറുമുണ്ട്. ഇന്റർനെറ്റിലെ കമ്പ്യൂട്ടർ ഒരു വെബ്സൈറ്റ് ഉപയോഗിക്കാൻ ശ്രമിക്കുമ്പോൾ പ്രസ്തുത വെബ് സൈറ്റിന്റെ ഡൊമൈൻ നാമവുമായി ബന്ധപ്പെട്ട IP മേൽവിലാസത്തിനായി ആദ്യം പ്രാദേശിക DNS സെർവറിൽ അന്വേഷിച്ച് കണ്ടെത്തുന്നു. ഇത് സ്ഥാപനത്തിന്റെ ഇന്റർനെറ്റ് വേഗത വർധിപ്പിക്കുന്നു. ഈ രീതിയിൽ IP മേൽവിലാസം കണ്ടെത്താനായില്ലായെങ്കിൽ ISP യുടെ DNS സെർവറിൽ അന്വേഷിക്കുന്നു.

നിലവിലുള്ള നമ്മുടെ DNS ദാതാവിന് പകരം ഉപയോഗിക്കാവുന്ന സ്വതന്ത്ര DNS സേവനമാണ് ഗൂഗിളിന്റെ പൊതു DNS. ഇതിന്റെ IP മേൽവിലാസങ്ങളാണ് 8.8.8.8, 8.8.4.4 എന്നിവ. ഈ പൊതു DNS ന്റെ IP മേൽവിലാസങ്ങളിൽ ഏതിലേക്കെങ്കിലും നമ്മുടെ കമ്പ്യൂട്ടറിനെ നയിക്കുന്നരീതിയിൽ നമ്മുടെ ശൃംഖലയെ ക്രമീകരിക്കാവുന്നതാണ്.

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. ഇന്റർനെറ്റ് പ്രോട്ടോകോളിന്റെ (IP) ഉയർന്ന തലത്തിൽ പ്രവർത്തിക്കുന്ന ഒരു പ്രോട്ടോകോളിന്റെ പേരെഴുതുക.
2. HTTPS എന്നതിന്റെ പൂർണ്ണ രൂപം എഴുതുക.
3. ഡാറ്റ സെന്ററുകളിൽ വെബ്സെർവറുകൾ സ്ഥാപിക്കുന്നതിന്റെ നേട്ടങ്ങൾ എന്തെല്ലാം?
4. ശരിയോ തെറ്റോ എന്ന് പ്രസ്താവിക്കുക.
   a. ക്ലയന്റ്-സെർവർ മാതൃക അനുസരിച്ചാണ് വെബ് സെർവർ സോഫ്റ്റ്‌വെയർ പ്രവർത്തിക്കുന്നത്.
   b. സെർവർ ഓപ്പറേറ്റിംഗ് സിസ്റ്റവും വെബ് സെർവർ സോഫ്റ്റ്‌വെയറും ചേർന്നതാണ് വെബ് സെർവർ.
5. ഒരു സോഫ്റ്റ്‌വെയർ പോർട്ട് സംഖ്യാനാമസമ്പ്രദായത്തിലെ ബിറ്റുകളുടെ എണ്ണം _______ ആണ്.

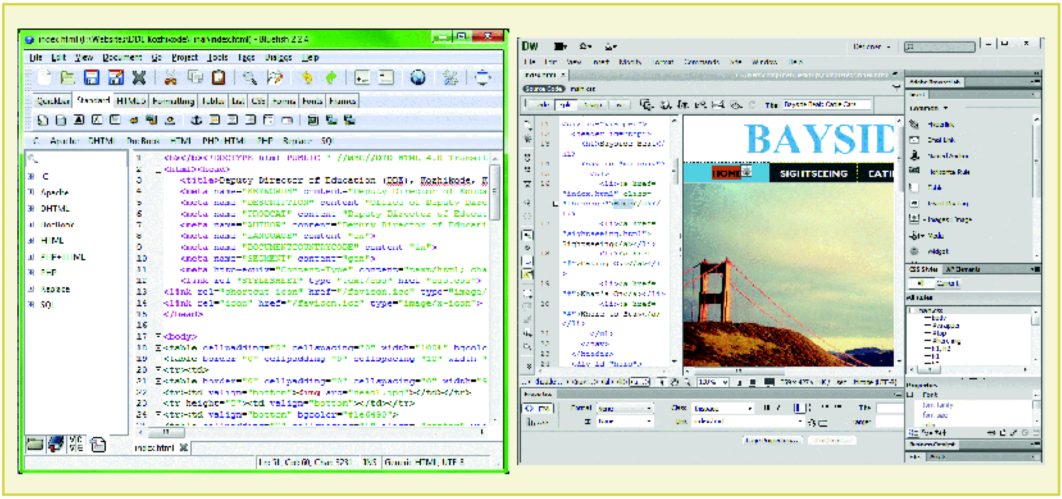
   a. 8 b. 16 c. 32 d. 64
6. ഡൊമൈൻ നാമകരണസമ്പ്രദായം ഡൊമൈൻ നാമത്തിന്റെ _____________ തിരികെ നൽകുന്നു.

4.3 വെബ് രൂപകൽപന ചെയ്യാം (Web designing)

വെബ് സൈറ്റ് നിർമാണത്തിന്റെ ആദ്യഘട്ടം വെബ് പേജുകളുടെ ആസൂത്രണമാണ്. നമ്മുടെ വിദ്യാലയത്തിനായി ഒരു വെബ്സൈറ്റ് വികസിപ്പിക്കുന്നു എന്ന് കരുതുക. വെബ് പേജുകളും അവ തമ്മിലുള്ള ബന്ധങ്ങളും തീരുമാനിച്ച് കഴിഞ്ഞാൽ അവയുടെ രൂപകൽപ്പനയിലേക്ക് കടക്കാം. ഇതിനായി ഏതെങ്കിലും ഒരു ടെക്സ്റ്റ് എഡിറ്റർ ഉപയോഗിക്കാവുന്നതാണ്. ഒരു ഹോം പേജും, സ്കൂളിൽ ലഭ്യമാകുന്ന കോഴ്സുകൾ, സൗകര്യങ്ങൾ, ബന്ധപ്പെടേണ്ട മേൽവിലാസം തുടങ്ങിയവ ദൃശ്യമാക്കുന്ന മറ്റൊരു പേജും നിർമിച്ച് അവ തമ്മിൽ മെനു ഉപയോഗിച്ച് ബന്ധപ്പെടുത്തുന്ന രീതിയിൽ മനോഹരമായ ഒരു വെബ് സൈറ്റ് നമുക്ക് നിർമിക്കാം.

വെബ് രൂപകൽപ്പന ചെയ്യുന്നതിന് നിരവധി സോഫ്റ്റ്‌വെയറുകൾ ലഭ്യമാണ്. വേഡ് പ്രൊസസർ ഉപയോഗിച്ച് അനായാസമായി ഒരു ഡോക്യുമെന്റ് തയ്യാറാക്കുന്നത് പോലെ, വെബ് നിർമാണ ടൂളുകളിൽ ലഭ്യമായിട്ടുള്ള സങ്കേതങ്ങളുടെ സഹായത്തോടെ വെബ് പേജുകൾ നിർമിക്കാവുന്നതാണ്. വെബ് പേജുകൾ രൂപകൽപന ചെയ്യുന്നതിനും അവയെ പരസ്പരം ബന്ധിപ്പിച്ച് കൊണ്ട് വെബ്സൈറ്റ് രൂപീകരിക്കുന്നതിനുമുള്ള സംവിധാനങ്ങൾ ഈ സോഫ്റ്റ്‌വെയറുകൾ നൽകുന്നു. FTP പ്രോട്ടോകോളുകൾ ഉപയോഗിച്ച് ഫയലുകൾ സെർവറുകളിലേക്ക് കൈമാറ്റം ചെയ്യുന്നതിനുള്ള സ്വാഭാവിക സൗകര്യം ഇത്തരം സോഫ്റ്റ്‌വെയറുകളിൽ ഉണ്ടാകും. ബ്ലൂഫിഷ്, ബൂട്ട് സ്ട്രാപ്പ്, അഡോബ് ഡ്രീംവീവർ, മൈക്രോസോഫ്റ്റ് എക്സ്പ്രഷൻ വെബ് തുടങ്ങിയവ വെബ് നിർമാണത്തിന് ഉപയോഗിക്കുന്ന പ്രമുഖ സോഫ്റ്റ്‌വെയറുകളാണ്, ചിത്രം 4.8-ൽ വെബ് നിർമാണത്തിന് ഉപയോഗിക്കുന്ന

പ്രശസ്തമായ ചില സോഫ്റ്റ്വെയറുകളുടെ IDE (ഇന്റഗ്രേറ്റഡ് ഡവലപ്മെന്റ് എൻവ യോൺമെന്റ്) കാണിച്ചിരിക്കുന്നു.



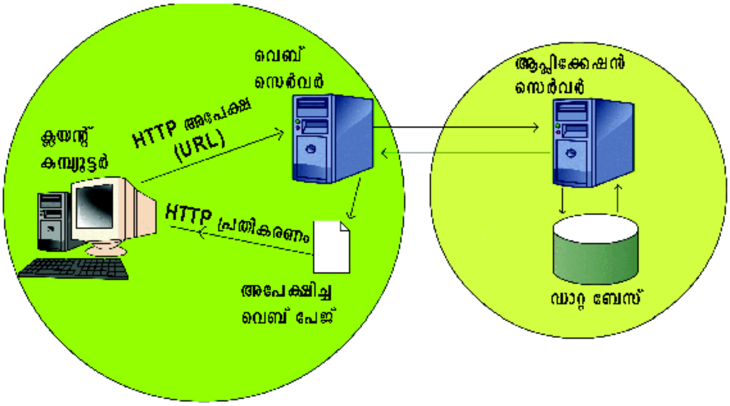
ബ്ലൂഫിഷ്   ഡ്രീം വീവർ

ചിത്രം 4.8 : വെബ് രൂപകൽപനയ്ക്കുപയോഗിക്കുന്ന സോഫ്റ്റ്വെയറുകളുടെ IDE

വെബ് സൈറ്റുകളുടെ നിർമാണത്തെക്കുറിച്ച് ഹൈസ്കൂൾ ക്ലാസുകളിൽ നിങ്ങൾ പഠിച്ചിട്ടുണ്ട്. HTML-ൽ ഉള്ള ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും ഉപയോഗിച്ച് വെബ് പേജുകൾ രൂപകൽപ്പന ചെയ്യാം.

4.4 സ്റ്റാറ്റിക്, ഡൈനാമിക് വെബ് പേജുകൾ (Static and dynamic web pages)

ചെറുകിട വ്യവസായങ്ങൾ, വിദ്യാലയങ്ങൾ തുടങ്ങിയവയുടെ വെബ് സൈറ്റുകൾ എപ്പോൾ സന്ദർശിച്ചാലും, പേജുകളിലെ ഉള്ളടക്കം സ്ഥിരമായി (സ്റ്റാറ്റിക് ആയി) നിൽക്കുന്നത് ശ്രദ്ധിച്ചു കാണുമല്ലോ? ഇത്തരം വെബ് സൈറ്റുകളെ സ്റ്റാറ്റിക് വെബ്സൈറ്റുകൾ എന്ന് വിളിക്കുന്നു. എസ്.എസ്.എൽ.സി/ഹയർസെക്കന്ററി പരീക്ഷ എഴുതിയ വിദ്യാർഥികൾ തങ്ങളുടെ രജിസ്റ്റർ നമ്പർ നൽകുമ്പോൾ, അവരവരുടെ പരീക്ഷാഫലം ദൃശ്യമാകുംവിധം ഉള്ളടക്കം മാറിവരുന്ന വെബ് സൈറ്റുകളും നിങ്ങൾ കണ്ടിട്ടുണ്ടാകും. ഇവ ഡൈനാമിക് പേജുകൾ എന്നറിയപ്പെടുന്നു. പ്രോഗ്രാം കോഡിൽ മാറ്റം വരുത്താത്തിടത്തോളം ഉള്ളടക്കം സ്ഥിരമായിത്തന്നെ നിലനിൽക്കുന്ന പേജുകളാണ് സ്റ്റാറ്റിക് വെബ് പേജുകൾ. ആദ്യകാലത്ത് HTML മാത്രം ഉപയോഗിച്ചാണ് ഈ സ്റ്റാറ്റിക് വെബ് പേജുകൾ നിർമിച്ചിരുന്നത്. പിന്നീട് ജാവ സ്ക്രിപ്റ്റ്, വിബി സ്ക്രിപ്റ്റ് തുടങ്ങിയ സ്ക്രിപ്റ്റിംഗ് ഭാഷകളുടെ വരവോടെ വെബ് പേജുകൾക്ക് ചലനാത്മകത കൈവന്നു. വെബ് പേജിന്റെ ചില ഭാഗങ്ങളിൽ മൗസ് പോയിന്റർ വയ്ക്കുമ്പോൾ ആ ഭാഗത്ത് നിറവ്യത്യാസവും രൂപമാറ്റവും സംഭവിക്കുന്നതും, ചിത്രങ്ങൾ ഒന്നിന് പിറകേ ഒന്നായി ക്രമീകരിക്കപ്പെടുന്നതുമെല്ലാം സ്ക്രിപ്റ്റിംഗ് ഭാഷ ഉപയോഗിക്കുന്നതു കൊണ്ടാണ്. ഇത്തരം പ്രത്യേകതകളുള്ള വെബ് പേജുകളും സ്റ്റാറ്റിക് വെബ് പേജുകളായിട്ടാണ് കണക്കാക്കിയിട്ടുള്ളത്.



സ്റ്റാറ്റിക് വെബ് പേജിനായുള്ള നിർദേശം   ഡൈനാമിക് വെബ് പേജിനായുള്ള നിർദേശം

ചിത്രം 4.9 : സ്റ്റാറ്റിക്, ഡൈനാമിക് എന്നീ വെബ് പേജുകൾക്കായുള്ള നിർദേശങ്ങൾ

ഓരോ തവണ വെബ് പേജ് എടുക്കുമ്പോഴും സെർവർ ഭാഗത്ത് പേജിലുള്ള കോഡ് അത് നിർമിക്കുന്നു എങ്കിൽ അത്തരം പേജിനെ ഡൈനാമിക് വെബ് പേജ് എന്ന് വിളിക്കുന്നു. വെബ് സെർവറിൽ സ്ഥാപിച്ചിട്ടുള്ള ആപ്ലിക്കേഷൻ പ്രോഗ്രാം ഉപയോഗിച്ച് ഡൈനാമിക് പേജുകൾ പ്രവർത്തിപ്പിക്കുന്നു. വെബ് പേജിലെ സ്ക്രിപ്റ്റ്, വെബ് സെർവറിൽ പ്രവർത്തിക്കുമ്പോൾ ഉണ്ടാകുന്ന HTML പേജുകൾ ക്ലയന്റ് ബ്രൗസറിലേക്കയക്കപ്പെടുന്നു. മിക്കവാറും സന്ദർഭങ്ങളിൽ ഇത്തരം വെബ് പേജുകൾ നിർമിക്കാൻ ഡാറ്റബേസിലുള്ള ഡാറ്റയാണ് ഉപയോഗിക്കുന്നത്. SSLC/HSE പരീക്ഷാഫലങ്ങൾ കാണിക്കുന്ന വെബ് സൈറ്റുകൾ, റോഡ്, റെയിൽ, വ്യോമ യാത്രകൾക്കുള്ള ടിക്കറ്റ് മുൻകൂട്ടി ഉറപ്പിക്കാനാകുന്ന സൈറ്റുകൾ, പണമിടപാടുകൾക്കുള്ള വെബ് സൈറ്റുകൾ തുടങ്ങിയവയുടെ പേജുകളിൽ പരിവർത്തനാത്മക ഉള്ളടക്കമുണ്ടാവുകയും അവ ഡൈനാമിക് പേജുകളായി കണക്കാക്കുകയും ചെയ്യുന്നു. PHP, ASP, JSP തുടങ്ങിയവ ഉപയോഗിച്ച് ഡൈനാമിക് വെബ് പേജുകൾ നിർമിക്കുന്നു. സ്റ്റാറ്റിക്, ഡൈനാമിക് എന്നീ വെബ് പേജുകൾ തമ്മിലുള്ള ഒരു താരതമ്യം പട്ടിക 4.2-ൽ തന്നിട്ടുണ്ട്.

സ്റ്റാറ്റിക് വെബ് പേജുകൾ	ഡൈനാമിക് വെബ്പേജുകൾ
വെബ് പേജിന്റെ ഉള്ളടക്കവും രൂപരേഖയും സ്ഥിരമായിരിക്കും.	ഉള്ളടക്കവും രൂപരേഖയും പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോൾ വ്യത്യാസപ്പെടുന്നു.
സ്റ്റാറ്റിക് വെബ് പേജുകൾ ഒരിക്കലും ഡാറ്റാബേസ് ഉപയോഗിക്കുന്നില്ല.	ഡാറ്റാബേസ് ഉപയോഗിച്ച് ക്വറികളുടെ സഹായത്താൽ പരിവർത്തനാത്മക ഉള്ളടക്കം സൃഷ്ടിക്കുന്നു.
സ്റ്റാറ്റിക് വെബ് പേജുകൾ ബ്രൗസറിൽ നേരിട്ട് പ്രവർത്തിക്കുന്നതിനാൽ സെർവർ ഭാഗത്തുള്ള ആപ്ലിക്കേഷൻ പ്രോഗ്രാമുകളുടെ ആവശ്യമില്ല.	ഡൈനാമിക് വെബ് പേജുകൾ സെർവർ ഭാഗത്തുള്ള ആപ്ലിക്കേഷൻ പ്രോഗ്രാമുകളുടെ സഹായത്താൽ പ്രവർത്തിച്ച് ഫലങ്ങൾ പ്രദർശിപ്പിക്കുന്നു.
സ്റ്റാറ്റിക് വെബ് പേജുകൾ എളുപ്പത്തിൽ നിർമിക്കാനാകും	ഡൈനാമിക് വെബ് പേജുകളുടെ നിർമാണത്തിന് പ്രോഗ്രാമിങ് വൈദഗ്ധ്യം ആവശ്യമാണ്.

പട്ടിക 4.2: സ്റ്റാറ്റിക്, ഡൈനാമിക് എന്നീ വെബ് പേജുകളുടെ താരതമ്യം

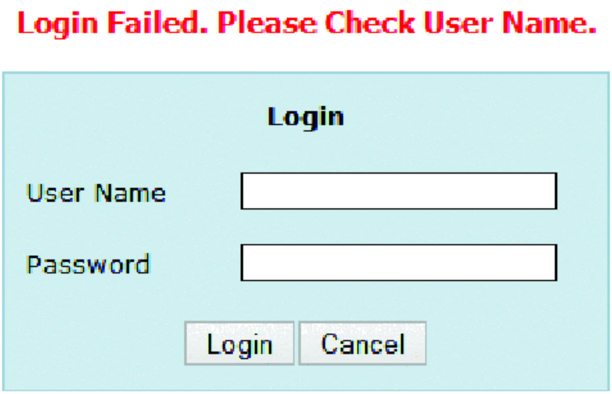
4.5 സ്ക്രിപ്റ്റുകൾ (Scripts)

HTML പേജുകളിൽ ഉൾക്കൊള്ളിക്കുന്ന പ്രോഗ്രാം കോഡുകളാണ് സ്ക്രിപ്റ്റുകൾ. നോട്ട് പാഡ് പോലുള്ള ഒരു ടെക്സ്റ്റ് എഡിറ്റർ ഉപയോഗിച്ചാണ് അവ തയാറാക്കുന്നത്. ഡൈനാമിക് വെബ് പേജുകൾ നിർമിക്കുന്നതിനായി ജാവ സ്ക്രിപ്റ്റ്, വി.ബി. സ്ക്രിപ്റ്റ്, പി.എച്ച്.പി, പേൾ തുടങ്ങിയ സ്ക്രിപ്റ്റിങ് ഭാഷകൾ ഉപയോഗിക്കുന്നു. ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന്റെ സഹായത്തോടെ കമ്പ്യൂട്ടർ ഹാർഡ്‌വെയറിനെക്കൊണ്ട് നിർദേശങ്ങൾ നടപ്പിലാക്കിക്കുന്നവയാണ് സാമ്പ്രദായിക പ്രോഗ്രാമിങ് ഭാഷകൾ. എന്നാൽ വെബ് സെർവറുകളാൽ/വെബ് ബ്രൗസറുകളാൽ വ്യാഖ്യാനിക്കപ്പെടുന്നവയാണ് സ്ക്രിപ്റ്റിംഗ് ഭാഷകൾ. ഇന്ന്, ഭൂരിഭാഗം ഏക കമ്പ്യൂട്ടർ അധിഷ്ഠിത പ്രോഗ്രാമുകളും വെബ് അധിഷ്ഠിത പ്രോഗ്രാമുകളാൽ പുനസ്ഥാപിക്കപ്പെട്ടിട്ടുണ്ട്. ധനകാര്യസ്ഥാപനങ്ങളിലും മറ്റും ഉപയോഗിച്ചിരുന്ന സോഫ്റ്റ്‌വെയറുകൾ അതാത് ശാഖകളിൽത്തന്നെ സ്ഥാപിച്ചിരുന്നവയാണ്. എന്നാൽ ഇന്ന് ഈ സ്ഥാപനങ്ങൾ തങ്ങളുടെ സോഫ്റ്റ്‌വെയർ വെബ് സെർവറിൽ ലഭ്യമാക്കുകയും അത് ഇന്റർനെറ്റിന്റെ സഹായത്താൽ ജീവനക്കാർ ഉപയോഗിക്കുകയും ചെയ്യുന്നു. ധനവിനിമയ സമ്പ്രദായത്തിന് ആവശ്യമായ സോഫ്റ്റ്‌വെയറുകൾ, ഹയർസെക്കന്ററി പ്രവേശനത്തിനുള്ള സോഫ്റ്റ്‌വെയർ തുടങ്ങിയ വെബ് അധിഷ്ഠിത സോഫ്റ്റ്‌വെയറുകളുടെ നിർമാണത്തിന് സ്ക്രിപ്റ്റിങ് ഭാഷകൾ ഉപയോഗിക്കുന്നു. ഒരു HTML പേജിൽ **<SCRIPT>**, **</SCRIPT>** എന്നീ ടാഗുകൾക്ക് ഉള്ളിലായാണ് സ്ക്രിപ്റ്റ് എഴുതുന്നത്. ഒരു HTML ഡോക്യുമെന്റിൽ സ്ക്രിപ്റ്റ് ഉൾപ്പെടുത്തുന്നതിനോ, സൂചിപ്പിക്കുന്നതിനോ ആണ് `<SCRIPT>` ടാഗ് ഉപയോഗിക്കുന്നത്. ഇതിന് **Type**, **Src** എന്നീ ആട്രിബ്യൂട്ടുകളുണ്ട്. സ്ക്രിപ്റ്റ് ടാഗിന് ഉള്ളിലുള്ള സ്ക്രിപ്റ്റിങ് ഭാഷ തിരിച്ചറിയാനായാണ് `Type` ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്. സ്ക്രിപ്റ്റ് കോഡ് ഉൾക്കൊള്ളുന്ന ഫയലിന്റെ പേര് നൽകാനാണ് SRC ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്. HTML ടാഗുകളെയും ആട്രിബ്യൂട്ടുകളെയും കുറിച്ച് നമുക്ക് തുടർന്നുള്ള ഭാഗങ്ങളിൽ മനസിലാക്കാം.

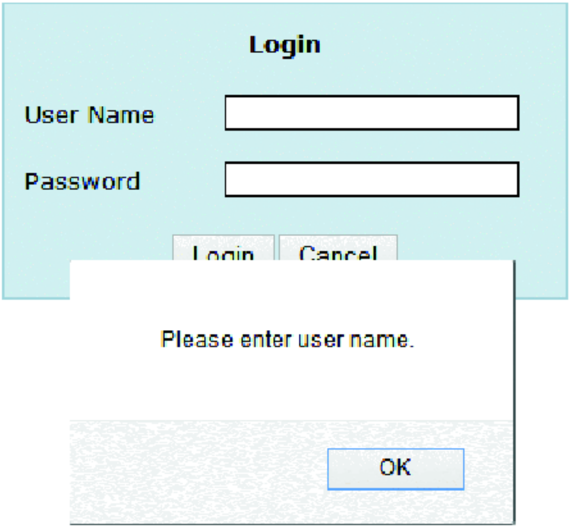
ഒരു HTML കോഡിനുള്ളിൽ ജാവ സ്ക്രിപ്റ്റ് ഉൾപ്പെടുത്താൻ `<SCRIPT Type="text/javascript">` ഉപയോഗിക്കുന്നു.

4.5.1 വിവിധതരം സ്ക്രിപ്റ്റിങ് ഭാഷകൾ (Types of scripting languages)

ഇന്റർനെറ്റിലെ ഒരു വെബ്സൈറ്റിന്റെ ലോഗിൻ പേജ് പരിഗണിക്കുക (സാധാരണയായി ഉപയോക്താവ് യൂസർ നാമവും, പാസ്‌വേഡും നൽകാനുപയോഗിക്കുന്ന പേജാണിത്). ഉപയോക്താവ് യൂസർ നാമവും പാസ്‌വേഡും നൽകിയശേഷം ലോഗിൻ ബട്ടണിൽ ക്ലിക്ക് ചെയ്യും. എന്നാൽ ഇവ നൽകാതെ ലോഗിൻ ബട്ടണിൽ ക്ലിക്ക് ചെയ്താൽ എന്ത് സംഭവിക്കും? സ്വാഭാവികമായും ഇവ നൽകിയിട്ടില്ലെന്ന് കമ്പ്യൂട്ടർ ഉപയോക്താവിനോട് പറയും. രണ്ട് കമ്പ്യൂട്ടറുകളാണ് ഈ വെബ് പേജ് നിയന്ത്രിക്കുന്നത് വെബ് പേജ് പ്രദർശിപ്പിക്കുന്ന ക്ലയന്റ് കമ്പ്യൂട്ടറും വെബ് പേജ് നൽകുന്ന സെർവർ കമ്പ്യൂട്ടറും. ഉപയോക്താവ് യൂസർ നാമം നൽകിയിട്ടുണ്ടോ എന്ന് എവിടെയാണ് പരിശോധിക്കുന്നത്. ക്ലയന്റ് കമ്പ്യൂട്ടറിലോ അതോ സെർവർ കമ്പ്യൂട്ടറിലോ? ധാരാളം ആളുകൾ ഒരേ വെബ്സൈറ്റ് സന്ദർശിക്കുന്നതിനാലും സെർവർ തന്നെ ഇത്തരം അപേക്ഷകളെല്ലാം കൈകാര്യം ചെയ്യേ



ചിത്രം 4.10 : സെർവറിന്റെ പ്രതികരണം



ചിത്രം 4.11 : ക്ലയന്റ് ബ്രൗസറിന്റെ പ്രതികരണം

ണ്ടതിനാലും ക്ലയന്റ് കമ്പ്യൂട്ടറിനേക്കാൾ വളരെയധികം തിരക്കിലായിരിക്കും സെർവർ കമ്പ്യൂട്ടർ. ആയതിനാൽ ക്ലയന്റ് ഭാഗത്ത് ചെയ്യാവുന്ന പ്രവൃത്തികളെല്ലാം അവിടെത്തന്നെ ചെയ്യുന്നതാണ് ഉചിതം. അങ്ങനെ സെർവറിന്റെ ജോലിഭാരം കുറയ്ക്കാനാകും. ഈ പരിശോധന സെർവറിലാണ് നടക്കുന്നതെങ്കിൽ, ഉപയോക്താവ് നൽകിയ അപേക്ഷ ക്ലയന്റിൽ നിന്നും ഇന്റർനെറ്റിലൂടെ സെർവറിൽ അയയ്ക്കേണ്ടിവരും. ചിലപ്പോഴെങ്കിലും സെർവറിലെത്താൻ ഇതിന് ഇന്റർനെറ്റിലൂടെ വളരെ ദൂരം സഞ്ചരിക്കേണ്ടിയും വരും. വളരെ അധികം ക്ലയന്റുകളിൽ നിന്ന് ലഭിക്കുന്ന ഇത്തരം അപേക്ഷകൾ വന്ന ക്രമത്തിൽ സെർവർ ശേഖരിക്കുന്നു. സെർവർ പ്രൊസസ്സ് ചെയ്യുന്നതുവരെ അയച്ച അപേക്ഷ നിരയിൽ കാത്ത് നിൽക്കുന്നു. ക്ലയന്റ് ഏതെങ്കിലും യൂസർ നാമമോ പാസ്‌വേഡോ നൽകിയിട്ടുണ്ടോയെന്ന് ക്രമം അനുസരിച്ച് സെർവർ പരിശോധിക്കുന്നു. നൽകിയിട്ടില്ലെങ്കിൽ, ചിത്രം 4.10-ൽ കാണിച്ചിട്ടുള്ളതുപോലെ ഉചിതമായ ഒരു സന്ദേശം സെർവർ ക്ലയന്റിലേക്ക് അയയ്ക്കുന്നു. ഈ സന്ദേശവും ഇന്റർനെറ്റിലൂടെ വളരെ ദൂരം സഞ്ചരിച്ച് ക്ലയന്റിലെത്തുന്നു. ചുരുക്കത്തിൽ യൂസർ നാമം നൽകാതെ ഉപയോക്താവ് സബ്മിറ്റ് ബട്ടണിൽ ക്ലിക്ക് ചെയ്താൽ യൂസർ നാമം നൽകിയിട്ടില്ല എന്ന സന്ദേശം ക്ലയന്റ് കമ്പ്യൂട്ടറിൽ ലഭിക്കുന്നതുവരെ അയാൾ കാത്തിരിക്കേണ്ടിവരുന്നു. കൂടാതെ ക്ലയന്റിൽ നിന്ന് സെർവറിലേക്കും തിരിച്ചും ഡാറ്റ സഞ്ചരിക്കുന്നതിനാൽ ശൃംഖലയുടെ ഗതാഗത തിരക്ക് വർധിക്കുകയും ചെയ്യുന്നു.

ഈ പരിശോധന ക്ലയന്റ് ഭാഗത്തു തന്നെ നടക്കുകയാണെങ്കിൽ, ഉപയോക്താവ് സബ്മിറ്റ് ബട്ടൺ അമർത്തുമ്പോൾ, യൂസർ നാമമോ, പാസ്‌വേഡോ നൽകിയിട്ടുണ്ടോ എന്ന് ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റ് കോഡിന് തന്നെ പരിശോധിക്കാവുന്നതാണ്. അല്ലാത്തപക്ഷം ഉചിതമായ ഒരു സന്ദേശം ദൃശ്യമാക്കാവുന്നതുമാണ്. ഈ പ്രക്രിയയിൽ, ഡാറ്റ ഇന്റർനെറ്റിലൂടെ സെർവറിലേക്ക് സഞ്ചരിക്കുകയോ, ലളിതമായ ഈ ദൗത്യത്തിന് സെർവറിനെ

ശല്യപ്പെടുത്തുകയോ ചെയ്യുന്നില്ല. ഉപയോക്താവ് സബ്മിറ്റ് ബട്ടൺ അമർത്തുമ്പോൾ യൂസർനാമവും/പാസ്‌വേഡും നൽകിയിട്ടില്ല എന്ന സന്ദേശം നിമിഷനേരം കൊണ്ട് ചിത്രം 4.11-ൽ കാണുന്നത് പോലെ ഉപയോക്താവിനു ലഭിക്കും. ഇതിനായി ശൃംഖലയിലെ വിഭവങ്ങൾ ഉപയോഗിക്കേണ്ടിവരുന്നുമില്ല.

ഇനി നമുക്ക് മറ്റൊരു സാഹചര്യം പരിചയപ്പെടാം. ഒരു ഉപയോക്താവ് തെറ്റായ യൂസർ നാമവും പാസ്‌വേഡും നൽകിയെന്ന് കരുതുക. യൂസർ നാമവും പാസ്‌വേഡും നൽകി യിട്ടുണ്ടോ എന്ന് ക്ലയന്റ് കമ്പ്യൂട്ടറിന് പരിശോധിക്കുവാൻ കഴിഞ്ഞേക്കാം. എന്നാൽ ഇത് ശരിയാണോ എന്ന് പരിശോധിക്കുവാൻ കഴിയില്ല. യൂസർ നാമങ്ങളുടെയും അനുബന്ധ പാസ്‌വേഡുകളുടേയും വിശദാംശങ്ങൾ സെർവർ കമ്പ്യൂട്ടറിലാണ് സൂക്ഷിക്കുന്നത് എന്ന താണ് ഇതിന് കാരണം. ഇവയുടെ പരിശോധന സെർവർ ഭാഗത്ത് മാത്രമേ ചെയ്യാൻ സാധിക്കുകയുള്ളൂ. ഡാറ്റയുടെ സാധുത ഉറപ്പ് വരുത്തുന്നതിന് സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റ് ഉപയോഗിക്കുന്ന സന്ദർഭമാണിത്. സെർവറിലുള്ള വിവരങ്ങൾ ഉപയോഗപ്പെടുത്തിക്കൊണ്ട് ഡാറ്റയുടെ സാധുത ഉറപ്പ് വരുത്തേണ്ടിവന്നാൽ അത് സെർവർ ഭാഗത്ത് തന്നെ ചെയ്യേ ണ്ടതുണ്ട്.

സ്ക്രിപ്റ്റിങ് ഭാഷകളെ ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ എന്നും സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ എന്നും തരംതിരിച്ചിരിക്കുന്നു എന്ന് ഇതിൽ നിന്നും വ്യക്തമാണല്ലോ? ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് ഉപയോഗിക്കുന്നത് ക്ലയന്റ് ഭാഗത്ത് ചെയ്യേണ്ട ജോലി ബ്രൗസറിൽ പ്രവർത്തിപ്പിച്ച് ചെയ്യിക്കുന്നതിനാണ്. സെർവറിൽ പ്രവർത്തിക്കുന്ന സ്ക്രിപ്റ്റുകളെ സെർവർ ഭാഗത്തുള്ള സ്ക്രിപ്റ്റുകൾ എന്നു പറയുന്നു. സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ പ്രവർത്തിക്കുമ്പോൾ ലഭിക്കുന്ന ഔട്ട്പുട്ട് HTML പേജുകളുടെ രൂപത്തിൽ ഉള്ളതും ക്ലയന്റിലേക്ക് അയയ്ക്കാവുന്നതും ആയിരിക്കും.

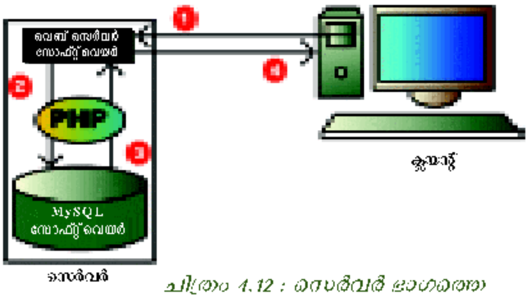
A. ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് (Client side scripting)

ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ്ങിൽ ഡാറ്റയുടെ സാധുത ഉറപ്പാക്കുന്ന സ്ക്രിപ്റ്റ് കോഡ് HTML കോഡിനൊപ്പം ബ്രൗസറിലേക്ക് ഡൗൺലോഡ് ചെയ്യപ്പെടുന്നു. ഡാറ്റ അസാധു ആണെങ്കിൽ ഉചിതമായ സന്ദേശം ദൃശ്യമാകുന്നു. ശരിയായ സാധുത ഉറപ്പാക്കിയശേഷം മാത്രമേ ഡാറ്റയെ വെബ് സെർവറിലേക്ക് അയയ്ക്കുകയുള്ളൂ.

സ്ക്രിപ്റ്റ്, ക്ലയന്റ് ബ്രൗസറിൽ പ്രവർത്തിക്കുന്നതിനാൽ അത് ഉപയോക്താക്കൾക്ക് ഉടൻ തന്നെ പ്രതികരണം നൽകുന്നു. വെബ് ആപ്ലിക്കേഷൻ പ്രവർത്തിക്കുമ്പോൾ, ചില ചുമ തലകൾ ക്ലയന്റ് ബ്രൗസറുമായി പങ്കുവയ്ക്കുവാൻ ഇത്തരം സ്ക്രിപ്റ്റിങ്ങിലൂടെ സാധി ക്കുന്നു. സ്ക്രിപ്റ്റുകളെ പിന്തുണയ്ക്കാത്ത ബ്രൗസറുകൾ ഉള്ളതാണ് ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ്ങിന്റെ പരിമിതി. ചില അവസരങ്ങളിൽ ബ്രൗസറിൽ സ്ക്രിപ്റ്റുകൾ പ്രവർത്തി ക്കുന്നതിനുള്ള അനുമതി ഉപഭോക്താക്കൾ നിഷേധിച്ചിട്ടുമുണ്ടാകും. അത്തരം സാഹച ര്യങ്ങളിൽ ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റ് പ്രവർത്തിക്കുകയില്ല. ജാവാസ്ക്രിപ്റ്റും വി.ബി. സ്ക്രിപ്റ്റും പ്രമുഖ സ്ക്രിപ്റ്റിങ് സാങ്കേതിക വിദ്യകളാണ്. ഡാറ്റ സെർവറിലേക്ക് അയയ്ക്കുന്നതിനു മുൻപ്, സാധുത പരിശോധനയും ലളിതമായ കണക്കുകൂട്ടലുകളും ചെയ്യുന്നതിനാണ് ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് സാധാരണയായി ഉപയോഗിക്കുന്നത്.

B. സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് (Server side scripting)

ഡൈനാമിക് വെബ് പേജുകളെക്കുറിച്ച് കഴിഞ്ഞ ഭാഗത്ത് നാം ചർച്ച ചെയ്തിട്ടുള്ളതാണ്. ഇത്തരം പേജുകൾ ഉണ്ടാക്കാൻ സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് ഉപയോഗിക്കുന്നു. സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിങ്ങിനുള്ള മറ്റൊരു ഉദാഹരണം നോക്കാം. SSLC പരീക്ഷാഫലം പ്രസിദ്ധീകരിക്കുന്ന വെബ് സൈറ്റ് പരിഗണിക്കുക. നാം ഒരു കുട്ടിയുടെ രജിസ്റ്റർ നമ്പർ നൽകുമ്പോൾ, വെബ് സൈറ്റിൽ ആ കുട്ടിയുടെ മാർക്ക് ലിസ്റ്റ് കാണാം. ഓരോ കുട്ടിയുടെ കാര്യവും ഇതുപോലെ ആയിരിക്കും. SSLC പരീക്ഷയെഴുതിയ ലക്ഷക്കണക്കിന് കുട്ടികൾ ഓരോരുത്തർക്കും വേണ്ടി ഓരോ വെബ് പേജ് തയാറാക്കുകയെന്നത് പ്രായോഗികമല്ല. അങ്ങനെയെങ്കിൽ ഇത് സാധിക്കുന്നതെങ്ങനെ? ലക്ഷക്കണക്കിന് കുട്ടികളുടെ പരീക്ഷാഫലം വെബ് സെർവറിലുള്ള ഒരു ഡാറ്റാബേസിൽ സംഭരിക്കുന്നു. ഉപയോക്താവ് രജിസ്റ്റർ നമ്പർ നൽകുമ്പോൾ ആ വിദ്യാർഥിയുടെ പരീക്ഷാഫലം ഡാറ്റബേസിൽ നിന്നും എടുക്കാൻ സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ ഉപയോഗിക്കുന്നു. പിന്നീട് ഈ പരീക്ഷാഫലം ഉപയോഗിച്ച് സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ ഒരു വെബ് പേജ് നിർമിക്കുന്നു. തുടർന്ന് ഈ വെബ് പേജ് ക്ലയന്റ് ബ്രൗസറിലേയ്ക്ക് അയക്കുകയും ബ്രൗസർ അത് പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്നു. ഇത്തരത്തിൽ സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റ്, SSLC പരീക്ഷയെഴുതിയ ഓരോ കുട്ടിക്കുമായി ഓരോ വെബ് പേജ് നിർമിക്കുന്നു. വെബ് അടിസ്ഥാനമാക്കിയുള്ള ആപ്ലിക്കേഷനുകളുടെ ത്വരിത ഗതിയിലുള്ള വളർച്ച സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകളുടെ ഉപയോഗം കൂട്ടിയിട്ടുണ്ട്.



ചിത്രം 4.12 : സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകളുടെ പ്രവർത്തനം

ഉപയോക്താവ് നിർദേശിക്കുന്നതും, സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ ഉൾപ്പെട്ടിട്ടുള്ള വെബ് പേജ് സെർവറിൽ പ്രവർത്തിക്കുകയും, തൽഫലമായി ലഭിക്കുന്ന HTML കോഡ് ക്ലയന്റ് ബ്രൗസറിലേക്ക് അയയ്ക്കുകയും ചെയ്യുന്ന സാങ്കേതികവിദ്യയാണ് സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിങ്. സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ ഉള്ളടക്കം മാറിക്കൊണ്ടിരിക്കുന്ന വെബ് പേജുകൾ നിർമിക്കുന്നു. സ്ക്രിപ്റ്റുകൾ സെർവറിൽ പ്രവർത്തിക്കുന്നതിനാൽ ക്ലയന്റ് കമ്പ്യൂട്ടറിലുള്ള ബ്രൗസറിന്റെയോ ഓപ്പറേറ്റിങ് സിസ്റ്റത്തിന്റെയോ പതിപ്പും തരവും ഈ പ്രവർത്തനത്തെ ബാധിക്കുന്നില്ല. സെർവറിലുള്ള വിഭവങ്ങൾ ഉപയോഗിച്ചുകൊണ്ടാണ് ഇത് പ്രവർത്തിക്കുന്നത്. സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റ് എഴുതുന്നതിന് ഉപയോഗിക്കുന്ന പ്രമുഖമായ സ്ക്രിപ്റ്റിങ് ഭാഷകളാണ് PERL, PHP, ASP, JSP തുടങ്ങിയവ.

സ്ക്രിപ്റ്റിങ് ഭാഷകളുടെ തരംതിരിക്കലിനെക്കുറിച്ചുള്ള ഒരു താരതമ്യപഠനം പട്ടിക 4.3-ൽ തന്നിട്ടുണ്ട്.

ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ്	സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിങ്
സ്ക്രിപ്റ്റ് ക്ലയന്റ് ബ്രൗസറിലേയ്ക്ക് പകർത്തുന്നു.	സ്ക്രിപ്റ്റ് വെബ് സെർവറിൽ നിലനിർത്തുന്നു.
സ്ക്രിപ്റ്റ് ക്ലയന്റ് ബ്രൗസറിൽ പ്രവർത്തിക്കുന്നു.	സ്ക്രിപ്റ്റ് വെബ് സെർവറിൽ പ്രവർത്തിക്കുമ്പോൾ ലഭിക്കുന്ന വെബ് പേജ് ക്ലയന്റ് ബ്രൗസറിലേക്കയയ്ക്കുന്നു.
ക്ലയന്റ് ഭാഗത്തെ ഡാറ്റയുടെ സാധുത പരിശോധനയ്ക്കാണ് പ്രധാനമായും സ്ക്രിപ്റ്റ് ഉപയോഗിക്കുന്നത്.	ഡാറ്റാബേസുമായി ബന്ധപ്പെടുന്നതിനും വെബ് സെർവറിൽ നിന്നും ഡാറ്റ ബ്രൗസറിലേയ്ക്ക് അയക്കുന്നതിനുമാണ് സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ ഉപയോഗിക്കുന്നത്.
ഉപയോക്താവിന് ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിന്റെ പ്രവർത്തനം തടയാൻ സാധിക്കും.	സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റുകളുടെ പ്രവർത്തനം ഉപയോക്താവിന് തടയുവാൻ സാധിക്കില്ല.
ബ്രൗസറിന്റെ തരവും പതിപ്പും ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിന്റെ പ്രവർത്തനത്തെ ബാധിക്കുന്നു.	വെബ് ബ്രൗസറിന്റെ സവിശേഷതകൾ സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിന്റെ പ്രവർത്തനത്തെ ബാധിക്കുന്നില്ല.

പട്ടിക 4.3: ക്ലയന്റ് ഭാഗത്തെയും സെർവർ ഭാഗത്തെയും സ്ക്രിപ്റ്റുകൾ തമ്മിലുള്ള താരതമ്യം

ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ ഡാറ്റയുടെ സാധുത ഉപയോക്താവിന്റെ ബ്രൗസറിൽ വച്ചുതന്നെ പരിശോധിക്കുന്നതിനാണ് ഉപയോഗിക്കുന്നത്. മാത്രമല്ല അത് സെർവറിന്റെ ജോലി ഭാരവും ശൃംഖലയിലെ ഗതാഗതക്കുരുക്കും കുറയ്ക്കുകയും ചെയ്യുന്നു. ആയതിനാൽ, സാധുത പരിശോധിക്കുന്നതിനായുള്ള സ്ക്രിപ്റ്റുകൾ പൂർണമായും ക്ലയന്റ് ഭാഗത്തേക്ക് മാറ്റിയിട്ടുണ്ട്. അതുകൊണ്ട്, സെർവറിലേക്ക് അയയ്ക്കുന്ന ഡാറ്റ പൂർണമായും തെറ്റുകളില്ലാത്തതും നേരിട്ട് ഡാറ്റാബേസിൽ സംഭരിക്കാവുന്നതുമാണ്. എന്നാൽ ക്ലയന്റ് ബ്രൗസർ, സ്ക്രിപ്റ്റുകൾ മനസ്സിലാക്കാതിരിക്കുകയോ, ഉപയോക്താവ് സുരക്ഷിത കാരണങ്ങളാൽ സ്ക്രിപ്റ്റിന്റെ ബ്രൗസറിലുള്ള പ്രവർത്തനം തടയുകയോ ചെയ്താൽ, സാധുത പരിശോധന നടത്താതെ ഡാറ്റ സെർവറിലേക്ക് അയയ്ക്കപ്പെടും. ഇക്കാരണത്താൽ അസാധു ആയ ഡാറ്റ ഡാറ്റബേസിൽ സംഭരിക്കപ്പെടും. ഡാറ്റയുടെ സാധുത സെർവർ ഭാഗത്ത് കൂടി ഉറപ്പാക്കാനായാൽ ഡാറ്റാബേസിൽ സംഭരിക്കുന്ന ഡാറ്റയുടെ സാധുത സംരക്ഷിക്കപ്പെടും.

4.5.2 സ്ക്രിപ്റ്റിങ് ഭാഷകൾ (Scripting languages)

രണ്ടു തരം സ്ക്രിപ്റ്റിങ് ഭാഷകൾ നാം പരിചയപ്പെട്ടു കഴിഞ്ഞു. പ്രചാരത്തിലുള്ള ചില സ്ക്രിപ്റ്റിങ് ഭാഷകളുടെ സവിശേഷതകൾ നമുക്ക് ചർച്ച ചെയ്യാം.

A. ജാവ സ്ക്രിപ്റ്റ് (Java Script)

ഉപയോക്താവുമായി സംവദിക്കുന്ന വെബ് പേജുകളുടെ നിർമാണത്തിന് ഉപയോഗിക്കുന്ന സ്ക്രിപ്റ്റിങ് ഭാഷയാണ് ജാവാസ്ക്രിപ്റ്റ്. നെറ്റ്സ്കേപ്പ് കമ്മ്യൂണിക്കേഷൻ കോർപ്പറേഷനിൽ പ്രവർത്തിച്ചിരുന്ന കാലത്ത് ബ്രണ്ടൻ ഇച്ച് (Brenden Eich - ചിത്രം 4.13) നിർമിച്ചതാണ് ജാവ സ്ക്രിപ്റ്റ്. ക്ലയന്റ് ഭാഗത്തെ വ്യാഖ്യാനിക്കാവുന്ന ഭാഷയാണ് ജാവ സ്ക്രിപ്റ്റ് . ജാവ സ്ക്രിപ്റ്റ് കോഡ് എഴുതുവാൻ ജിനി IDE യോ നോട്ട്പാഡോ പോലുള്ള ടെക്സ്റ്റ് എഡിറ്ററുകൾ ഉപയോഗിക്കാവുന്നതാണ്. ക്ലയന്റ് ഭാഗത്തെ ഒരു പ്രമുഖ സ്ക്രിപ്റ്റിങ് ടൂൾ ആയ ഇത് മിക്കവാറും ബ്രൗസറുകളിൽ പ്രവർത്തിക്കുന്നു. ജാവസ്ക്രിപ്റ്റ് HTML

ചിത്രം 4.13 ബ്രെണ്ടൻ ഇച്ച് (1961 -)

കോഡിൽ ഉൾപ്പെടുത്തിയോ പ്രത്യേക ഫയലിലെഴുതി HTML ഫയലിനോട് ലിങ്ക് ചെയ്തോ ഉപയോഗിക്കാം. പ്രത്യേക ഫയലിൽ എഴുതിയ ജാവ സ്ക്രിപ്റ്റ് .js എക്സ്റ്റൻഷനോടുകൂടി സൂക്ഷിക്കണം. ജാവാസ്ക്രിപ്റ്റ് ഫയലുകളെ തിരിച്ചറിയാൻ ഇത് സഹായിക്കുന്നു. ക്ലയന്റ് ഭാഗത്തെ ഫോമുകളിലെ ഡാറ്റയുടെ സാധുത ഉറപ്പിക്കാൻ ഉപയോഗിക്കുന്ന ടൂൾ എന്ന നിലയിൽ ജാവാ സ്ക്രിപ്റ്റ് പ്രശസ്തമാണ്. ലളിതമായ കണക്ക് കൂട്ടലുകളും ചലിക്കുന്ന കാർട്ടൂൺ ചിത്രങ്ങളും വെബ് പേജുകളിൽ ഉൾപ്പെടുത്തുന്നതിന് ഇത് ഉപയോഗിക്കുന്നു. അധ്യായം 6-ൽ ജാവ സ്ക്രിപ്റ്റിനെക്കുറിച്ച് വിശദമായി ചർച്ച ചെയ്യാം.

ജാവ സ്ക്രിപ്റ്റിന്റെ പ്രചാരം ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റുകൾ തയാറാക്കുന്ന പ്രവർത്തനത്തിന്റെ പുരോഗതിക്ക് കാരണമായി. ഒന്നാംവർഷ ഹയർസെക്കന്ററി പ്രവേശനത്തിന് അപേക്ഷ സമർപ്പിക്കുമ്പോൾ, SSLC രജിസ്റ്റർ നമ്പർ നൽകിയാലുടൻ, നിങ്ങളുടെ പേരും, ജനനത്തീയതിയും, മറ്റ് വിവരങ്ങളും താഴെയുള്ള ടെക്സ്റ്റ് ബോക്സുകളിൽ പ്രത്യക്ഷപ്പെടുന്നത് ശ്രദ്ധിച്ചിരിക്കുമല്ലോ. ബ്രൗസറിൽ പുനർപ്രവേശിക്കാതെയാണ് ഇത് ചെയ്യുന്നത്. വെബ് പേജ് പുനർനിർമിക്കാതെ (റിഫ്രഷ് ചെയ്യാതെ) സെർവറിൽ നിന്ന് ഡാറ്റ എടുത്ത് ടെക്സ്റ്റ് ബോക്സുകളിൽ നിറയ്ക്കുന്നു. അജാക്സ് (Ajax) എന്ന സാങ്കേതികവിദ്യയാണ് ഇവിടെ ഉപയോഗിക്കുന്നത്. ബ്രൗസറുകളുടെ വിനിമയശേഷി വർധിപ്പിക്കുന്നതിനാണ് അജാക്സ് ഉപയോഗിക്കുന്നത്. അജാക്സ് എന്നാൽ അസിങ്ക്രോണസ് ജാവ സ്ക്രിപ്റ്റ്, എക്സ്റ്റൻസിബിൾ മാർക്കപ്പ് ലാംഗേജ് (XML) എന്നിവ ചേർന്നതാണ്. ഉപയോക്താക്കൾക്ക് പുതിയ ടാഗുകൾ നിർമിക്കാൻ ഉതകുന്ന മാർക്കപ്പ് ഭാഷയാണ് XML. അജാക്സ് വെബ്സൈറ്റിൽ പ്രവർത്തനക്ഷമമായ ശേഷം പരിവർത്തനാത്മക ഉള്ളടക്കം പ്രദർശിപ്പിക്കാനായി വെബ് പേജ് പൂർണമായും വീണ്ടും കൊണ്ടുവരേണ്ടതില്ല. ക്ലയന്റും സെർവറും തമ്മിൽ പിന്നണിയിൽ ഡാറ്റ കൈമാറ്റം ചെയ്തുകൊണ്ട് വെബ് പേജിനെ നവീകരിക്കാൻ അജാക്സ് സഹായിക്കുന്നു. ബ്രൗസറിന് ജാവ സ്ക്രിപ്റ്റ് കൈകാര്യം ചെയ്യാനാകാതെ വരുകയോ, ബ്രൗസറിലുള്ള ജാവ സ്ക്രിപ്റ്റിന്റെ പ്രവർത്തനം ഉപയോക്താവ് തടയുകയോ ചെയ്താൽ അജാക്സ് ആപ്ലിക്കേഷൻ പ്രവർത്തിക്കുകയില്ല.

B. വിബി സ്ക്രിപ്റ്റ് (VB Script)

വിഷ്വൽ ബേസിക് എന്ന പ്രോഗ്രാമിങ് ഭാഷ അടിസ്ഥാനമാക്കി മൈക്രോസോഫ്റ്റ് കോർപ്പറേഷൻ വികസിപ്പിച്ച സ്ക്രിപ്റ്റിങ് ഭാഷയാണ് വിബി സ്ക്രിപ്റ്റ്. മൈക്രോസോഫ്റ്റ് ഇന്റർനെറ്റ് എക്സ്പ്ലൊററിനായി ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് ഭാഷയായും മൈക്രോസോഫ്റ്റ് ഇന്റർനെറ്റ് ഇൻഫർമേഷൻ സർവീസസിനായി സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് ഭാഷയായും വി.ബി. സ്ക്രിപ്റ്റ് ഉപയോഗിക്കുന്നു. നിർഭാഗ്യവശാൽ, മറ്റ് ബ്രൗസറുകൾക്ക് വിബി സ്ക്രിപ്റ്റ് കോഡ് കൃത്യമായി മനസിലാക്കാനോ പ്രദർശിപ്പിക്കാനോ സാധിക്കില്ല. അതുകൊണ്ടു തന്നെ, ക്ലയന്റ് ഭാഗത്തെ പ്രചാരം കുറഞ്ഞ ഒരു സ്ക്രിപ്റ്റിങ് ടൂൾ ആണ് ഇത്. സെർവർ ഭാഗത്ത് വിൻഡോസ് ഓപ്പറേറ്റിങ് സിസ്റ്റത്തിന് പ്രചാരമേറെയുള്ളതിനാൽ സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റെഴുതുന്നതിന് വിബി സ്ക്രിപ്റ്റ് കൂടുതലായി ഉപയോഗിക്കുന്നു.

.NET ഫ്രെയിംവർക്കിന്റെ – ഉപയോഗയോഗ്യമായ പ്രോഗ്രാം കോഡിന്റെ ഒരു ശേഖരം, ആവിർഭാവത്തോടെ വി.ബി. സ്ക്രിപ്റ്റ് .NET ഫ്രെയിംവർക്കിലെ ASP.NET ന്റെ ഭാഗമായി മാറി.

C. PHP

ചിത്രം 4.14 റാസ്മസ് ലെർഡോർഫ് (1968 -)

PHP എന്നത് ഹൈപ്പർ ടെക്സ്റ്റ് പ്രീപ്രൊസസറിനെ സൂചിപ്പിക്കുന്നു. വെബ് വികസനത്തിന് അനുയോജ്യമായതും, HTML- ൽ ഉൾക്കൊള്ളിക്കാവുന്നതുമായ ഇത് പൊതു ആവശ്യങ്ങൾക്കായുള്ള ഒരു ഓപ്പൺ സോഴ്സ് സ്ക്രിപ്റ്റിങ് ഭാഷയാണ്. സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റ് എഴുതുന്നതിനുള്ള ടൂൾ ആയ ഇതിന്റെ കോഡ് ജാവ, C, PERL എന്നിവയ്ക്ക് സമാനമാണ്. വളരെ എളുപ്പത്തിൽ പരിവർത്തനാത്മക വെബ് പേജുകൾ നിർമിക്കാൻ ഇത് ഉപയോഗിക്കാം. 1994-ൽ റാസ്മസ് ലെർഡോർഫ് (Rasmus Lerdorf-ചിത്രം 4.14) ആണ് PHP വികസിപ്പിച്ചത്, എങ്കിലും PHP സമൂഹമാണ് തുടർന്ന് അത് പരിപോഷിപ്പിച്ച് കൊണ്ടിരിക്കുന്നത്. PHP കോഡുള്ള വെബ് പേജുകളുടെ ഫയൽനാമത്തിന്റെ എക്സ്റ്റൻഷൻ .php എന്നായിരിക്കും. ഉപയോക്താവ് PHP വെബ് പേജിനായി അപേക്ഷിക്കുമ്പോൾ, വെബ് സെർവർ HTML കോഡിൽ ഉൾപ്പെടുത്തിയിട്ടുള്ള PHP കോഡ് വ്യാഖ്യാനിക്കുകയും പ്രവർത്തിപ്പിക്കുകയും ചെയ്യുന്നു. PHP കോഡ് വെബ് സെർവറിൽ പ്രവർത്തിക്കണമെങ്കിൽ ഒരു PHP ഇന്റർപ്രട്ടർ വെബ് സെർവറിൽ ഉണ്ടായിരിക്കണം. വെബ് സെർവറിൽ PHP കോഡ് പ്രവർത്തിക്കുമ്പോൾ നിർമിക്കപ്പെടുന്ന HTML പേജ് ക്ലയന്റ് ബ്രൗസറിലേക്ക് അയയ്ക്കുന്നു. ഡാറ്റാബേസ് പ്രോഗ്രാമിങ്ങിന് സഹായിക്കുന്നു എന്നതാണ് PHP യുടെ പ്രധാനപ്പെട്ട സവിശേഷത. PHP യോടൊപ്പം സാധാരണ ഉപയോഗിക്കുന്ന ഡാറ്റാബേസാണ് MYSQL. എല്ലാ ഓപ്പറേറ്റിങ് സിസ്റ്റങ്ങൾക്കും PHP ഇന്റർപ്രട്ടർ ലഭ്യമാണ്. സൗജന്യമായി ഇന്റർനെറ്റിൽ നിന്നും പകർത്താവുന്ന LAMP (Linux, Apache, MYSQL, PHP) സെർവർ സോഫ്റ്റ്‌വെയറാണ് ലിനക്സ് പ്ലാറ്റ്ഫോമുകളിൽ ഉപയോഗിക്കുന്നത്. ഇവിടെ, ലിനക്സ് എന്നത് സെർവർ ഓപ്പറേറ്റിങ് സിസ്റ്റവും, അപ്പാഷെ എന്നത് വെബ് സെർവറും, MYSQL എന്നത് ഡാറ്റാബേസും, PHP എന്നത് സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിംഗ് ഭാഷയും ആയി പ്രവർത്തിക്കുന്നു. ഇന്റർനെറ്റിൽ നിന്നും സൗജന്യമായി പകർത്താവുന്ന WAMP (WINDOWS, Apache, MYSQL, PHP) സെർവർ സോഫ്റ്റ്‌വെയറാണ് വിൻഡോസ് ഓപ്പറേറ്റിങ് സിസ്റ്റം ഉപയോഗിക്കുന്നത്. 10-ാമത്തെ അധ്യായത്തിൽ PHP-യെക്കുറിച്ച് കൂടുതലായി ചർച്ച ചെയ്യാം.

D. ആക്റ്റീവ് സെർവർ പേജുകൾ (Active Server Pages)

പരസ്പരവിനിമയം നടത്തുന്ന വെബ് ആപ്ലിക്കേഷനുകൾ നിർമിക്കുന്നതിനും പ്രവൃത്തിപ്പിക്കുന്നതിനും വേണ്ടി സെർവർ ഭാഗത്ത് ഉപയോഗിക്കുന്ന സ്ക്രിപ്റ്റിങ് സംവിധാനമാണ് മൈക്രോസോഫ്റ്റിന്റെ ആക്റ്റീവ് സെർവർ പേജുകൾ (ASP). ASP -യിൽ HTML-ഉം, സ്ക്രിപ്റ്റിങ് ഭാഷയിലെ കോഡുണ്ടായിരിക്കും. സ്ക്രിപ്റ്റിങ് ഭാഷ വിബി സ്ക്രിപ്റ്റോ ജാവ സ്ക്രിപ്റ്റോ ആയിരിക്കും. ASP ഫയലുകളുടെ എക്സ്റ്റൻഷൻ .asp എന്നായിരിക്കും. മൈക്രോ സോഫ്റ്റിന്റെ സെർവർ സോഫ്റ്റ്‌വെയറായ ഇന്റർനെറ്റ് ഇൻഫർമേഷൻ

സർവീസസിൽ (IIS-ൽ) തനതായി നിർമിക്കപ്പെട്ടിട്ടുള്ള സവിശേഷത ഉപയോഗിച്ച് ഈ ഫയലുകൾ കംപൈൽ ചെയ്യുന്നു. ASP ഫയലുകൾ വിൻഡോസ് ഓപ്പറേറ്റിങ് സിസ്റ്റത്തിൽ മാത്രം പ്രവർത്തിക്കുന്നവയാണ്. സെർവറിൽ എക്സിക്യൂട്ട് ചെയ്തശേഷം ലഭിക്കുന്ന HTML വെബ് പേജ് ക്ലയന്റ് ഭാഗത്തെ ബ്രൗസറിലേക്കയക്കുന്നു. വിവിധ തരത്തിലുള്ള ഡാറ്റബേസുകളെ പിന്തുണയ്ക്കുന്ന, വളരെ പ്രമുഖമായ ഒരു സ്ക്രിപ്റ്റിങ് ഭാഷയാണിത്. ASP.NET -ന്റെ വരവോടെ, ASP -യുടെ തുടർപതിപ്പുകൾ പുറത്തിറക്കുന്നത് മൈക്രോസോഫ്റ്റ് നിർത്തലാക്കി. കോഡുകളുടെ ചുരുക്കെഴുത്തും, വ്യത്യസ്തങ്ങളായ ബട്ടണുകൾ, ടെക്സ്റ്റ് ബോക്സുകൾ തുടങ്ങിയവയുടെ ലഭ്യതയും വെബ് ആപ്ലിക്കേഷനുകൾ നിർമിക്കാനായി ASP.NET-ലുള്ള സവിശേഷതകളാണ്.

E. ജാവ സെർവർ പേജുകൾ (Java Server Pages)

പരിവർത്തനാത്മക വെബ് പേജ് നിർമിക്കുന്നതിനുള്ള എളുപ്പവും വേഗമേറിയതുമായ സാങ്കേതിക വിദ്യ JSP പ്രദാനം ചെയ്യുന്നു. 1999-ൽ സൺ മൈക്രോസിസ്റ്റംസ് നിർമിച്ച സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് ഭാഷയാണ് ഇത്. PHP-ക്ക് സമാനമാണ് JSP, എങ്കിലും പ്രോഗ്രാമിങ് ഭാഷയായി ജാവയാണ് ഉപയോഗിക്കുന്നത്. JSP ഫയലുകളുടെ ഫയൽ എക്സ്റ്റൻഷൻ .jsp എന്നാണ്. JSP പ്രവർത്തിപ്പിക്കുന്നതിന് അപ്പാഷെ ടോം ക്യാറ്റ് വെബ് സെർവർ ആവശ്യമാണ്. JSP കോഡിലുള്ള HTML കോഡും ജാവകോഡും വെബ് സെർവറിൽ പ്രവർത്തിക്കുന്നതിന്റെ (എക്സിക്യൂട്ട് ചെയ്ത്) ഫലമായി ലഭിക്കുന്ന HTML കോഡിനെ ബ്രൗസറിലേയ്ക്ക് അയക്കുന്നു. വൻതോതിലുള്ള വെബ് അധിഷ്ഠിത സോഫ്റ്റ്‌വെയറുകൾ നിർമിക്കുന്നതിനും പ്രവർത്തിപ്പിക്കുന്നതിനും ഉപയോഗിക്കുന്ന ജാവ 2 പ്ലാറ്റ്ഫോം എന്റർപ്രൈസ് എഡിഷന്റെ (J2EE) അവിഭാജ്യ ഘടകമാണ് JSP.

4.6 കാസ്കേഡിങ് സ്റ്റൈൽ ഷീറ്റ് (Cascading Style Sheet)

HTML ഭാഷയിൽ എഴുതിയിട്ടുള്ള ഒരു ലിഖിതത്തിന്റെ ഘടന വിവരിക്കാൻ ഉപയോഗിക്കുന്ന സ്റ്റൈൽ ഷീറ്റ് ഭാഷയാണ് കാസ്കേഡിങ് സ്റ്റൈൽ ഷീറ്റ് (CSS). ഒരു വെബ് പേജിലുള്ള ചിഹ്നങ്ങളുടെ നിറം, അക്ഷരങ്ങളുടെ രീതി, ഖണ്ഡികകൾക്ക് ഇടയിലുള്ള സ്ഥലം, നിരകളുടെ വലുപ്പം പ്രദർശിപ്പിക്കുന്ന രീതി, അരികുകളും അവയുടെ നിറങ്ങളും, പശ്ചാത്തല ചിത്രങ്ങൾ/നിറങ്ങൾ എങ്ങനെയാണ് ഉപയോഗിക്കുന്നത് തുടങ്ങിയ വിവിധതരത്തിലുള്ള പ്രതീതികൾ നിയന്ത്രിക്കുവാൻ CSS ഉപയോഗിക്കുന്നു. HTML ഉള്ളടക്കത്തെ അതിന്റെ രൂപഭംഗിയിൽ നിന്നും വേർതിരിച്ച് നിർത്താൻ CSS നമ്മെ അനുവദിക്കുന്നു. മൂന്ന് വ്യത്യസ്ത രീതികളിൽ CSS നമുക്ക് നടപ്പിലാക്കാം - ഇൻലൈൻ (ഒപ്പംതന്നെ), എംബഡഡ് (ഉൾക്കൊള്ളിച്ച്), ലിങ്ക്ഡ് (ബന്ധിപ്പിച്ച്) എന്നിവയാണവ.

- ഒപ്പമെഴുതുന്ന രീതിയിൽ, വെബ് പേജിന്റെ ബോഡി ഭാഗത്ത്, സ്റ്റൈൽ ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് കൊണ്ട്, CSS ശൈലി ഓരോ ടാഗിനും പ്രത്യേകമായി പ്രയോഗിക്കാവുന്നതാണ്.
- ഉൾക്കൊള്ളിക്കുന്ന CSS കോഡുകൾ, വെബ്പേജിന്റെ ഹെഡ് ഭാഗത്ത് സ്ഥാപിക്കുന്നു.
- CSS കോഡ് മാത്രമുൾക്കൊള്ളുന്ന ഒരു ബാഹ്യ ഫയലിനെ വെബ് പേജുമായി ബന്ധിപ്പിച്ച് കൊണ്ടാണ് CSS നിർവഹണം ലിങ്ക്ഡ് രീതിയിൽ നടപ്പിലാക്കുന്നത്.

എല്ലാ പേജുകൾക്കും ഒരേ കോഡ് പുനരുപയോഗിക്കാമെന്നതാണ് CSS -ന്റെ നേട്ടം. ഒരു വെബ്സൈറ്റിനുവേണ്ടിയുള്ള CSS സ്റ്റൈൽ ബാഹ്യഫയലായി ലിങ്ക്ചെയ്താണ് നടപ്പിലാക്കുന്നതെങ്കിൽ, ഒരു സ്റ്റൈൽ വ്യത്യാസപ്പെടുത്തുന്നതിലൂടെ, വെബ് സൈറ്റിലെ എല്ലാ പേജുകളിലെയും ടാഗുകളുടെ അവതരണ രീതി വ്യത്യാസപ്പെടുത്താനാകും. CSS സ്റ്റൈൽ പൊതുവായി ഒരു സ്ഥലത്ത് എഴുതുന്നതിനാൽ HTML-ഉം, CSS വേർതിരിച്ച് നിർത്താൻ സാധിക്കുകയും അങ്ങനെ പരിപാലനം എളുപ്പമാവുകയും ചെയ്യുന്നു. കൂടാതെ വെബ് പേജുകളിലുള്ള ടാഗുകളുടെ സ്റ്റൈൽ വിവരണവുമായി നല്ല രീതിയിൽ ബന്ധപ്പെടുത്തി ക്രമീകരിച്ചിരിക്കുന്നതുകൊണ്ട് എളുപ്പത്തിൽ അത് മനസ്സിലാക്കാനാവുന്നു. ഇത് വെബ് പേജുകളുടെ ബൈറ്റുകളിലുള്ള വലുപ്പം കുറയ്ക്കുകയും അങ്ങനെ വേഗത്തിൽ വെബ് പേജുകൾ ബ്രൗസറിൽ ദൃശ്യമാക്കുകയും ചെയ്യുന്നു.

വ്യത്യസ്ത സ്ക്രീൻ വലുപ്പമുള്ള ഉപകരണങ്ങളായ ഡെസ്ക്ടോപ്പ് മോണിട്ടറുകൾ, ടാബ് കമ്പ്യൂട്ടറുകൾ, മൊബൈൽ ഫോണുകൾ തുടങ്ങിയവയിൽ വെബ് പേജുകൾ പ്രദർശിപ്പിക്കാൻ അനുയോജ്യമാക്കുന്നത് CSS ആണ്. ചലനാത്മചിത്രങ്ങൾ, ആനിമേഷനുകൾ, പരസ്പരസംവേദനം എന്നിവ കൊണ്ടുവരാനാകുന്ന HTML-ന്റെ അടുത്ത പതിപ്പായ HTML5-ൽ ഉപയോഗിച്ചിരിക്കുന്നത് ജാവ സ്ക്രിപ്റ്റിനോടൊപ്പമുള്ള CSS ആണ്.

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. കോഡ് വ്യത്യാസപ്പെടുത്തുന്നത് വരെ ഉള്ളടക്കം മാറാതെ നിൽക്കുന്ന വെബ് പേജുകളെ ________ എന്ന് വിളിക്കുന്നു.
2. ഡൈനാമിക് വെബ് പേജുകൾ നിർമിക്കുന്നതിന് ഉപയോഗിക്കുന്ന രണ്ട് സാങ്കേതിക വിദ്യകളുടെ പേരെഴുതുക.
3. സ്ക്രിപ്റ്റ് ഉൾക്കൊള്ളിക്കാൻ ഉപയോഗിക്കുന്ന ടാഗാണ് ________.
4. ക്ലയന്റ് ഭാഗത്ത് സ്ക്രിപ്റ്റ് എഴുതുന്നതിന്റെ ഏതെങ്കിലും ഒരു ഉപയോഗമെഴുതുക.
5. ഒരു ജാവ സ്ക്രിപ്റ്റ് ഫയലിന്റെ എക്സ്റ്റൻഷൻ ________ ആണ്.
6. അജാക്സ് ഉപയോഗിക്കുന്നത് കൊണ്ടുള്ള നേട്ടങ്ങൾ എന്തൊക്കെയാണ്?
7. താഴെതന്നിട്ടുള്ള സ്ക്രിപ്റ്റിങ് ഭാഷകളെ ക്ലയന്റ് ഭാഗത്തേതെന്നും സെർവർ ഭാഗത്തേതെന്നുമായി തരംതിരിക്കുക.

   ജാവ സ്ക്രിപ്റ്റ്, PHP, ASP, വി.ബി. സ്ക്രിപ്റ്റ്
8. .asp ഫയലുകൾ ________ എന്ന വെബ് സെർവർ സോഫ്റ്റ്വെയർ ഉപയോഗിച്ചാണ് കംപൈൽ ചെയ്യുന്നത്.
9. CSS പ്രാവർത്തികമാക്കുന്നതിനുള്ള വ്യത്യസ്ത മാർഗങ്ങൾ എഴുതുക.

4.7 HTML ഡോക്യുമെന്റുകളുടെ അടിസ്ഥാന ആശയം (Basic concepts of HTML documents)

വെബ് പേജുകളുടെ നിർമാണത്തിന് വ്യാപകമായി ഉപയോഗിക്കുന്ന ഭാഷയാണ് HTML. ഓരോ വെബ് പേജും യഥാർഥത്തിൽ ഒരു HTML ഫയലാണ്. ഹൈപ്പർ ടെക്സ്റ്റ് ഡോക്യുമെന്റുകൾ നിർമിക്കുന്നതിനുള്ള ഒരു കൂട്ടം നിർദേശങ്ങൾ അടങ്ങിയ ലളിതമായ വാചകങ്ങളാണ് (ടെക്സ്റ്റുകളാണ്) ഓരോ HTML ഫയലും. ഈ നിർദേശങ്ങൾ HTML ടാഗുകൾ എന്ന് അറിയപ്പെടുന്നു. ഈ ടാഗുകൾ ഉപയോഗിക്കുമ്പോൾ നിർദേശങ്ങൾ കൂടുതൽ വ്യക്തതയുള്ളതാക്കാനായി ചില കീ-വേഡുകൾ അവയോടൊപ്പം ചേർക്കുന്നു. ഈ വാക്കുകൾ ആട്രിബ്യൂട്ടുകൾ എന്നറിയപ്പെടുന്നു. ഒരു വെബ് പേജിലെ ഉള്ളടക്കം ബ്രൗസറിൽ എങ്ങനെ പ്രദർശിപ്പിക്കണമെന്ന് തീരുമാനിക്കുന്നത് HTML ഡോക്യുമെന്റിലുള്ള ടാഗുകളും ആട്രിബ്യൂട്ടുകളും ഒരുമിച്ച് പ്രവർത്തിച്ചുകൊണ്ടാണ്. HTML ടാഗുകളെയും അവയുടെ ആട്രിബ്യൂട്ടുകളെയും കുറിച്ചുള്ള പഠനമാണ് യഥാർഥത്തിൽ HTML പഠനം. ടാഗുകളുടെയും ആട്രിബ്യൂട്ടുകളുടെയും വിശദാംശങ്ങളിലേക്ക് കടക്കുന്നതിന് മുമ്പ് ഒരു HTML ഡോക്യുമെന്റിന്റെ അടിസ്ഥാന ഘടന നമുക്ക് പരിശോധിക്കാം.

4.7.1 ഒരു HTML ഡോക്യുമെന്റിന്റെ അടിസ്ഥാനഘടന (Basic structure of an HTML document)

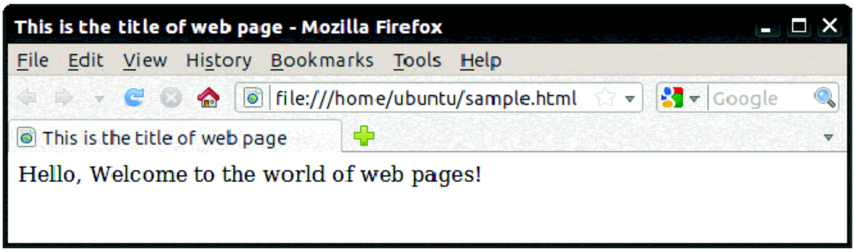
ഉദാഹരണം 4.1-ൽ ഒരു HTML ഡോക്യുമെന്റിന്റെ അടിസ്ഥാനഘടന കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.1: HTML ഡോക്യുമെന്റിന്റെ മാതൃക

```
<HTML>
   <HEAD>
     <TITLE> This is the title of web page </TITLE>
   </HEAD>
   <BODY>
     Hello, Welcome to the world of web pages!
   </BODY>
</HTML>
```

ഒരു ജോഡി ആംഗിൾ ബ്രാക്കറ്റുകൾക്കുള്ളിലായി (<, >) വലിയ അക്ഷരങ്ങളിലുള്ള ചില വാക്കുകൾ നിങ്ങൾക്ക് കാണാം. ഇവ HTML ടാഗുകളാണ്. HTML ടാഗുകൾ വലിയ അക്ഷരങ്ങളിലെഴുതണമെന്ന് നിർബന്ധമില്ല. HTML-ൽ വലിയ അക്ഷരവും ചെറിയ അക്ഷരവും തമ്മിൽ വ്യത്യാസമില്ല (കേയ്സ് സെൻസിറ്റീവ് അല്ല). നമുക്ക് വലിയ അക്ഷരത്തിലോ, ചെറിയ അക്ഷരത്തിലോ അവ കൂട്ടികലർത്തിയോ ഉപയോഗിക്കാവുന്നതാണ്. മറ്റ് വാക്കുകളിൽനിന്നും അല്ലെങ്കിൽ ചിഹ്നങ്ങളിൽനിന്നും വേർതിരിച്ചറിയാനായി, HTML ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളുടെ ആദ്യക്ഷരവും വലിയ അക്ഷരങ്ങളായും മറ്റുള്ളവ ചെറിയ അക്ഷരങ്ങളായുമാണ് സൂചിപ്പിക്കുന്നത്.

ഉദാഹരണം 4.1-ൽ കാണിച്ചിരിക്കുന്നത് പോലെ, എല്ലാ HTML പേജുകളും തുടങ്ങുന്നത് **<HTML>** ടാഗിലും അവസാനിക്കുന്നത് **</HTML>** ടാഗിലുമായിരിക്കും. ഒരു HTML ലിഖിതത്തെ ഹെഡ് ഭാഗമെന്നും ബോഡി ഭാഗമെന്നും രണ്ടായി തിരിച്ചിരിക്കുന്നു. ഹെഡ് വിശദീകരിക്കുന്നതിന് **<HEAD>** ടാഗുപയോഗിക്കുന്നു. വെബ് പേജിന്റെ ശീർഷകം ഉൾപ്പെടെ ലിഖിതത്തെക്കുറിച്ചുള്ള വിവരങ്ങൾ ഹെഡ് ഭാഗത്തുണ്ടാകും. ബ്രൗസർ വിൻഡോയിലെ ടൈറ്റിൽ ബാറിൽ കാണിക്കുന്ന വെബ് പേജിന്റെ ശീർഷകം നിർണയിക്കുന്നത് **<TITLE>** ടാഗ് ഉപയോഗിച്ചുകൊണ്ടാണ്. ബോഡി ഭാഗം നിർവചിക്കുന്നതിന് **<BODY>** ടാഗ് ഉപയോഗിക്കുന്നു. ബോഡി ഭാഗത്തുള്ളത് വെബ് പേജിന്റെ ഉള്ളടക്കമായിരിക്കും. മുകളിൽ തന്നിട്ടുള്ള HTML ലിഖിതം ബ്രൗസർ വിൻഡോയിൽ കണ്ടാൽ ചിത്രം 4.15-ൽ കാണുന്നത് പോലുണ്ടാകും.



ചിത്രം 4.15 : വെബ് ബ്രൗസർ ഉപയോഗിച്ച് തുറന്ന ഒരു വെബ് പേജ്

വിവിധ HTML പതിപ്പുകൾ: 1991 ന്റെ അവസാനം ടിം ബെർണേഴ്സ് ലീ (Tim Berners Lee) ആണ് HTML നിർമ്മിച്ചത്. എന്നാൽ, നിലവാരമുള്ള ആദ്യത്തെ HTML പതിപ്പായ HTML 2.0 പ്രസിദ്ധീകരിച്ചത് 1995-ലാണ്. പ്രമുഖ HTML പതിപ്പായ HTML 4.01 പ്രസിദ്ധീകരിച്ചത് 1999-കളുടെ അവസാനമാണ്. 2012-ലാണ് ഏറ്റവും പുതിയ പതിപ്പായ HTML 5 പുറത്തിറക്കിയത്. ഇത് കൂടുതൽ കൂടുതൽ ദൃശ്യ-ശ്രാവ്യ സൗകര്യങ്ങൾ ഉൾപ്പെടുത്തിക്കൊണ്ട് ഇപ്പോഴും പരിഷ്കരിച്ചുകൊണ്ടിരിക്കുന്നു.

4.7.2 HTML ഡോക്യുമെന്റിലെ ടാഗുകൾ (Tags in HTML document)

മുമ്പ് സൂചിപ്പിച്ചതു പോലെ, വെബ് പേജുകളുടെ ഉള്ളടക്കം ഏതു രീതിയിൽ ക്രമീകരിക്കണമെന്നും പ്രദർശിപ്പിക്കണമെന്നും ബ്രൗസറിനോട് പറയുന്ന നിർദേശങ്ങളാണ് HTML ടാഗുകൾ. എല്ലാ ടാഗുകൾക്കും <, > എന്നീ ആങ്കിൾ ബ്രായ്ക്കറ്റുകളിലടക്കം ചെയ്ത ടാഗ് നാമമുണ്ടായിരിക്കും.

4.7.3 കണ്ടെയ്നർ ടാഗുകളും ശൂന്യ ടാഗുകളും (Container tags and empty tags)

ഭൂരിഭാഗം ടാഗുകളും ജോഡികളായാണ് ഉപയോഗിക്കുന്നത്. പ്രാരംഭ (ഓപ്പണിംഗ്) ടാഗും അവസാനിപ്പിക്കുന്ന (ക്ലോസിങ്) ടാഗും ഉണ്ടാകും. ഉദാഹരണമായി, <HTML> എന്നത് തുടങ്ങുന്ന ടാഗും </HTML> എന്നത് അവസാനിപ്പിക്കുന്ന ടാഗുമാണ്. അവസാനിപ്പിക്കുന്ന ടാഗിനും, തുടങ്ങുന്ന ടാഗിനും ഒരേ പേരാണുള്ളത്. എന്നാൽ ആദ്യത്തെ ആങ്കിൾ

ബ്രാക്കറ്റിന് ശേഷം ഒരു ഫോർവേഡ് സ്ലാഷ് (/) ചിഹ്നം അവസാനിപ്പിക്കുന്ന ടാഗിന് അധികമായുണ്ടായിരിക്കും. തുടങ്ങുന്ന ടാഗും അവസാനിപ്പിക്കുന്ന ടാഗും ആവശ്യമുള്ള ടാഗുകളാണ് കണ്ടെയ്നർ ടാഗുകൾ. ഒരു സെക്ഷനിലേക്ക് ഉപയോഗിക്കാവുന്നവയാണ് കണ്ടെയ്നർ ടാഗുകൾ. സെക്ഷന്റെ തുടക്കത്തിൽ ഓപ്പണിംഗ് ടാഗും അവസാനം ക്ലോസിങ് ടാഗും നൽകുന്നു. ഉദാഹരണമായി, ഒരു HTML ലിഖിതത്തിന്റെ തുടക്കവും ഒടുക്കവും ക്രമപ്പെടുത്തുന്ന ടാഗ് ജോഡികളാണ് <HTML>, </HTML> എന്നിവ.

ചില ടാഗുകൾ ഈ രീതിയിലല്ല ഉപയോഗിക്കുന്നത്. അത്തരം ടാഗുകൾക്ക് ക്ലോസിങ് ടാഗ് കാണുകയില്ല. അവ ശൂന്യ ടാഗുകൾ അഥവാ എംപ്റ്റി ടാഗുകൾ എന്നറിയപ്പെടുന്നു. ഈ ടാഗുകൾ ഒരു സെക്ഷനിലേക്കുള്ളവയല്ല.
, <HR>, <IMG> തുടങ്ങിയവ ശൂന്യ ടാഗുകൾക്ക് ഉദാഹരണങ്ങളാണ്. ഈ അധ്യായത്തിലെ തുടർന്നുള്ള ഭാഗങ്ങളിൽ ഇത്തരം ടാഗുകൾ നമുക്ക് കാണാനാകും.

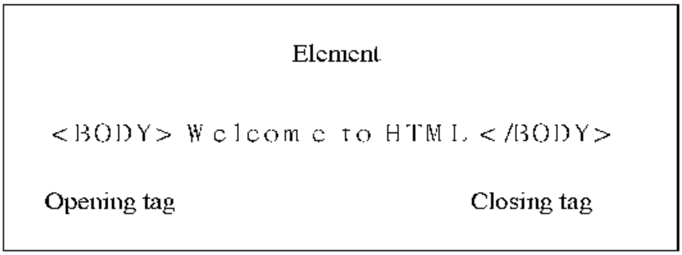
4.7.4 ടാഗുകളുടെ ആട്രിബ്യൂട്ടുകൾ (Attributes of tags)

നിറം, അളവ്, സ്ഥാനം, വിന്യാസം, അല്ലെങ്കിൽ മറ്റ് ആകാരങ്ങൾ തുടങ്ങിയ അധികവിവരങ്ങൾ വെബ് ബ്രൗസറിന് നൽകാനായി ചില ഘടകങ്ങൾ ഓപ്പണിങ് ടാഗിനുള്ളിൽ നൽകുന്നത് പതിവാണ്. ഈ ഘടകങ്ങളെ ആട്രിബ്യൂട്ടുകൾ എന്ന് വിളിക്കുന്നു. ഭൂരിഭാഗം ആട്രിബ്യൂട്ടുകൾക്കും ഒരു വില നൽകേണ്ടത് ആവശ്യമാണ്. HTML-ൽ ഏക ഉദ്ധരണികൾക്കുള്ളിലോ ജോഡികളായ ഉദ്ധരണികൾക്കുള്ളിലോ ആണ് വില നൽകുന്നത് (അതായത്, ആട്രിബ്യൂട്ട്='വില' അല്ലെങ്കിൽ ആട്രിബ്യൂട്ട് ="വില"). ആവശ്യാനുസരണം ഉപയോഗിക്കാവുന്ന ഒരു കൂട്ടം ആട്രിബ്യൂട്ടുകൾ ഓരോ ടാഗിനോടൊപ്പവും ഉണ്ടായിരിക്കും. ഒരു ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുകയാണെങ്കിൽ, സാധാരണയായി ടാഗ് നാമത്തിനുശേഷം ഒരു സ്പെയ്സ് കഴിഞ്ഞാകും അത് നൽകുന്നത്. ഒന്നിലധികം ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിക്കുമ്പോൾ, അവയുടെ ക്രമത്തിന് പ്രാധാന്യം നൽകേണ്ടതില്ല.

ഉദാഹരണമായി, ഒരു വെബ് പേജിലെ പശ്ചാത്തലനിറം മഞ്ഞയാക്കി മാറ്റാൻ `<BODY Bgcolor = "Yellow">`. ഇവിടെ Bgcolor എന്നത് <BODY> ടാഗിന്റെ ആട്രിബ്യൂട്ടും Yellow എന്നത് അതിന്റെ വിലയുമാണ്. ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും തുടർന്ന് വരുന്ന ഭാഗങ്ങളിൽ നാം ചർച്ച ചെയ്യുന്നതാണ്.

4.7.5 HTML എലമെന്റുകൾ (HTML Elements)

ഒരു ജോഡി ടാഗുകളും അവയ്ക്കിടയിൽ ഉൾക്കൊള്ളിച്ചിട്ടുള്ള ഉള്ളടക്കവും ചേർന്നതാണ് എലമെന്റ്. ചിത്രം 4.16-ൽ കാണിച്ചിരിക്കുന്നത് പോലെ, ബോഡി എലമെന്റിൽ ഓപ്പണിങ് ടാഗായ <BODY> യും ക്ലോസിങ് ടാഗായ </BODY> യും അവയ്ക്കിടയിലുള്ള ഉള്ളടക്കവും ഉണ്ടാകും.



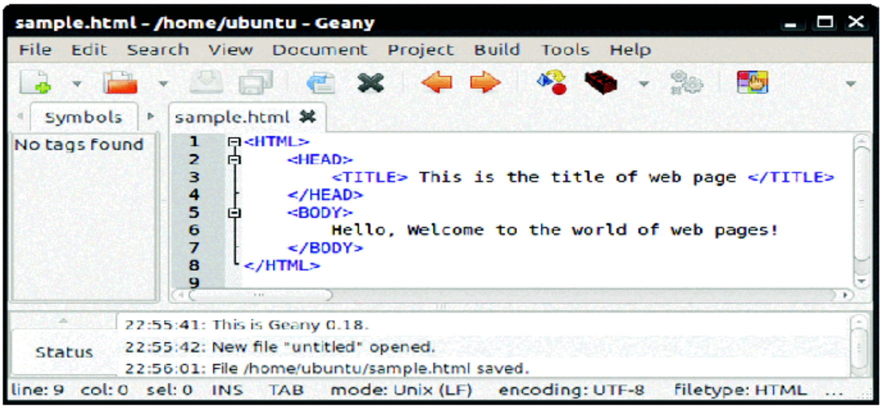
ചിത്രം 4.16 : ഒരു HTML ഘടകം

ഒരു HTML ലിഖിതത്തിന്റെ അടിസ്ഥാനഘടനയിൽ നാല് ജോഡി ടാഗുകളുണ്ടാകും. അവയാണ് :

```
<HTML>     </HTML>
<HEAD>     </HEAD>
<TITLE>    </TITLE>
<BODY>     </BODY>
```

4.8 ഒരു HTML ഡോക്യുമെന്റ് നിർമിക്കുന്നു (Creating an HTML document)

ഉദാഹരണം 4.17-ൽ തന്നിട്ടുള്ള HTML കോഡ് ഉപയോഗിച്ച് ലളിതമായ ഒരു വെബ് പേജ് നമുക്ക് നിർമിക്കാം. ജീനി (Geany), ജിഎഡിറ്റ് (Gedit), ടെക്സ്റ്റ്പാഡ് (TextPad), നോട്ട്പാഡ് (NotePad), നോട്ട്പാഡ്++ (NotePad++) തുടങ്ങിയ ടെക്സ്റ്റ് എഡിറ്ററുകളിൽ ഏതെങ്കിലും ഉപയോഗിച്ച് HTML ലിഖിതം നിർമിക്കാം. ഒരു ഫയൽ നാമവും എക്സ്റ്റെൻഷൻ ആയി .htm അല്ലെങ്കിൽ .html എന്നും നൽകി ഫയൽ സേവ് ചെയ്യണം (ഉദാഹരണമായി, sample.html). ജീനി എഡിറ്ററിൽ Sample.html എന്ന പേരിൽ സേവ് ചെയ്തിട്ടുള്ള HTML കോഡ് നമുക്ക് ചിത്രം 4.17-ൽ കാണാം.



ചിത്രം 4.17 : ജീനി എഡിറ്ററിലുള്ള HTML കോഡ്

ഒരു HTML ഡോക്യുമെന്റ് ബ്രൗസറിലൂടെ കാണുന്നു (Viewing an HTML document in a Browser)

HTML ലിഖിതം തയാറാക്കിക്കഴിഞ്ഞാൽ, അത് ബ്രൗസർ ഉപയോഗിച്ച് കാണാവുന്നതാണ്. മോസില്ല ഫയർഫോക്സ് (Mozilla Firefox), ഗൂഗിൾ ക്രോം (Google Chrome), ഇന്റർനെറ്റ് എക്സ്പ്ലോറർ (Internet Explorer), നെറ്റ്സ്കേപ്പ് നാവിഗേറ്റർ (Netscape Navigator) തുടങ്ങിയ ധാരാളം ബ്രൗസറുകളുണ്ട്. മോസില്ല ഫയർഫോക്സിൽ വെബ് പേജ് തുറന്നപ്പോൾ കിട്ടിയ സ്ക്രീൻ ചിത്രങ്ങളാണ് ഈ പുസ്തകത്തിൽ തന്നിട്ടുള്ളത്. ജീനിയുടെ ടൂൾബാറിലുള്ള Execute (എക്സിക്യൂട്ട്) ബട്ടൺ ക്ലിക്ക് ചെയ്യുമ്പോഴും, HTML കോഡിന്റെ ഔട്ട്പുട്ട് ലഭിക്കും.

4.9 അനിവാര്യമായ HTML ടാഗുകൾ (Essential HTML tags)

ഒരു വെബ് പേജ് നിർമിക്കുന്നതിന് അവശ്യം വേണ്ട ടാഗുകളെക്കുറിച്ച് നമുക്ക് മനസ്സിലാക്കാം. ടാഗുകൾ, അവയുടെ ഉപയോഗം, ബന്ധപ്പെട്ടുകിടക്കുന്ന ആട്രിബ്യൂട്ടുകളും അവയുടെ വിലകളും, ബ്രൗസർ വിൻഡോയിൽ അവയുടെ രൂപം തുടങ്ങിയവ ഇവിടെ വിശദീകരിക്കുന്നു.

4.9.1 <HTML>- ഒരു HTML പേജ് തുടങ്ങുന്നു (<HTML> - Starting an HTML page)

HTML ഡോക്യുമെന്റ് പൂർണമായി ഒരു ജോഡി <HTML>, </HTML> എന്നീ ടാഗ് ജോഡികൾക്കുള്ളിൽ വലയം ചെയ്തിരിക്കുന്നു. ഒരു HTML ഡോക്യുമെന്റ് തിരിച്ചറിയാനാണ് <HTML> ടാഗ് ഉപയോഗിക്കുന്നത്. പൊതുവായിപ്പറഞ്ഞാൽ ഒരു HTML പേജിൽ <HTML> ആദ്യ ടാഗും </HTML> അവസാന ടാഗുമായിരിക്കും. വെബ് പേജിലെ മറ്റെല്ലാം ഈ രണ്ട് ടാഗുകൾക്ക് ഇടയിലായിരിക്കും. അതായത് ഹെഡ് ഭാഗവും ബോഡി ഭാഗവും <HTML>, </HTML> എന്നീ ടാഗുകൾക്കിടയിലായിരിക്കും ഉൾക്കൊള്ളിച്ചിരിക്കുന്നത്. ഇത് ഒരു ജോഡി കണ്ടെയ്നർ ടാഗുകളാണ്. **Dir**, **Lang** എന്നിവയാണ് <HTML> ടാഗിന്റെ പ്രധാന ആട്രിബ്യൂട്ടുകൾ.

Dir

വെബ് പേജിൽ പ്രദർശിപ്പിക്കുന്ന (കാണിക്കുന്ന) അക്ഷരങ്ങളുടെ ദിശ സൂചിപ്പിക്കുന്നതിനാണ് <HTML> ടാഗിലെ Dir ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്. **ltr** (ലെഫ്റ്റ് ടു റൈറ്റ് - ഇടത്ത് നിന്ന് വലത്തേക്ക്) **rtl** (റൈറ്റ് ടു ലെഫ്റ്റ് – വലത്ത് നിന്ന് ഇടത്തേക്ക്) എന്നീ വിലകളാണ് ഈ ആട്രിബ്യൂട്ടിനുള്ളത്. ഈ ആട്രിബ്യൂട്ടിന്റെ തനതുവില വില ltr ആയിരിക്കും. ഉള്ളടക്കം അവതരിപ്പിക്കുന്നതിന് ഹീബ്രു, അറബിക്, ചൈനീസ് തുടങ്ങിയ ഭാഷകൾ ഉപയോഗിക്കുമ്പോൾ rtl എന്ന വില നൽകണം. ഉദാഹരണമായി <HTML Dir = "rtl"> എന്ന കോഡ് ഡോക്യുമെന്റ് വലത് വശത്ത് നിന്നും ഇടത് വശത്തേക്ക് വായിക്കണമെന്ന് സൂചിപ്പിക്കുന്നു.

Sl. No.	Code	Language
1	En	English
2	Fr	French
3	De	German
4	It	Italian
5	El	Greak
6	Es	Spanish
7	Ar	Arabic
8	Ja	Japanese
9	Hi	Hindi
10	Ru	Russian

പട്ടിക 4.4: പ്രചാരത്തിലുള്ള ചില ഭാഷകളും അവയുടെ കോഡുകളും

Lang

ലിഖിതത്തിനുള്ളിൽ പൊതുവായി ഉപയോഗിക്കുന്ന ഭാഷയെ സൂചിപ്പിക്കുന്നതിന് <HTML> ടാഗിന്റെ **Lang** ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. “en” എന്നത് ഇംഗ്ലീഷ് ഭാഷയേയും “it” എന്നത് ഇറ്റാലിയൻ ഭാഷയേയും സൂചിപ്പിക്കുന്നു.

ഉദാഹരണമായി <HTML Lang = "ar"> എന്ന കോഡ് അറബിക് ഭാഷയിലാണ് HTML ഡോക്യുമെന്റെഴുതിയിട്ടുള്ളതെന്ന് സൂചിപ്പിക്കുന്നു. പട്ടിക 4.3-ൽ Lang ആട്രിബ്യൂട്ടിന്റെ വിലയായി ഉപയോഗിക്കാവുന്ന ചില ഭാഷാകോഡുകൾ കൊടുത്തിട്ടുണ്ട്.

4.9.2 <HEAD> - ഹെഡ് ഭാഗം നിർമിക്കുന്നു (<HEAD> - Creating head)

ശീർഷകം, ഉപയോഗിച്ചിട്ടുള്ള സ്ക്രിപ്റ്റുകൾ, ശൈലി നിർവചനങ്ങൾ തുടങ്ങിയ ഡോക്യു മെന്റിനെ കുറിച്ചുള്ള വിവരങ്ങൾ HTML ലിഖിതത്തിന്റെ ഹെഡ്ഭാഗത്ത് ഉൾപ്പെടുത്തുന്നു. ഹെഡ് ഭാഗം പ്രസ്താവിക്കുന്ന ടാഗ് ജോഡികളാണ് <HEAD>, </HEAD> എന്നിവ. ഇതും ഒരു കണ്ടെയ്നർ ടാഗ് ജോഡിയാണ്.

4.9.3 <TITLE> - ശീർഷകം നിർമിക്കുന്നു (<TITLE> - Creating a title)

വെബ് ബ്രൗസറിന്റെ ടൈറ്റിൽ ബാറിൽ പ്രത്യക്ഷപ്പെടുന്ന വെബ് പേജിന്റെ ശീർഷകം അടങ്ങിയിട്ടുള്ള ഒരു കണ്ടെയ്നർ ടാഗാണിത്. ശീർഷകം ഉപയോഗിച്ചാണ് സെർച്ച് എൻജിൻ പേജ് തിരിച്ചറിയുന്നത്. <HEAD>, </HEAD> എന്നീ ടാഗ് ജോഡികൾക്കുള്ളിൽ <TITLE>, </TITLE> എന്നീ ടാഗ് ജോഡികൾ ഉപയോഗിച്ച് വെബ് പേജിന് ശീർഷകം നൽകുന്നു.

4.9.4 <BODY>-വെബ് പേജിന് ഒരു ബോഡി നിർമിക്കുന്നു (<BODY> - Creating a body)

ലിഖിതത്തിന്റെ ബോഡി ഭാഗം വിവരിക്കുന്നതിന് ഉപയോഗിക്കുന്ന ബോഡി ടാഗ് ജോഡി കളാണ് <BODY>, </BODY> എന്നിവ. ബ്രൗസർ വിൻഡോയിൽ പ്രദർശിപ്പിക്കേണ്ട ഉള്ളടക്കം ഈ ഭാഗത്ത് ഉൾക്കൊള്ളിക്കുന്നു. ഡോക്യുമെന്റിന്റെ ഉള്ളടക്കം നിർവചി ക്കുന്ന മറ്റെല്ലാ ടാഗുകളും ബോഡി ഭാഗത്താണ് ഉൾപ്പെടുത്തുന്നത്. ഈ ടാഗുകളെക്കു റിച്ച് വിശദീകരിക്കുന്നതിന് മുമ്പ്, <BODY> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകൾ ഏതെല്ലാമെന്ന് നോക്കാം.

Background

വെബ് പേജിന് ഒരു പശ്ചാത്തലചിത്രം നൽകുന്നത് ഈ ആട്രിബ്യൂട്ടാണ്. <BODY> ടാഗിന്റെ ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് പേജിനെ കൂടുതൽ ആകർഷകമാക്കുന്നു. അതിന്റെ പൊതു ഘടനയാണ്:

```
<BODY Background = "URL of the picture">
```

ആകാശത്തിന്റെ ചിത്രം ഒരു വെബ് പേജിന് പശ്ചാത്തലമായി നൽകുന്നതിനുള്ള HTML കോഡ് ഉദാഹരണം 4.2-ൽ കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.2: ഒരു വെബ് പേജിന് പശ്ചാത്തലമായി ചിത്രം നൽകുന്നു

```
<HTML>
<HEAD>
   <TITLE> Background Image </TITLE>
</HEAD >
<BODY Background = "Sky.jpg">
   Hello, Welcome to the world of Web Pages!........
</BODY>
</HTML>
```

ഉദാഹരണം 4.1-ൽ ഉള്ള HTML കോഡിലെ <BODY> ടാഗിന് ഒരു Background ആട്രിബ്യൂട്ടും അതിന് വിലയായി "Sky.jpg" എന്നും നൽകിക്കൊണ്ട് <BODY Background= "Sky.jpg"> ആയി വ്യത്യാസപ്പെടുത്തിയിരിക്കുന്നു. വെബ് പേജ് തുറക്കുന്നതിന് മുമ്പ്, ചിത്രത്തിന്റെ ഫയൽ ഇപ്പോൾ ഉപയോഗിച്ചുകൊണ്ടിരിക്കുന്ന ഫോൾഡറിൽ/ഡയറക്ടറിയിൽ ഉൾപ്പെടുത്തേണ്ടതാണ്. ചിത്രം 4.18-ൽ കാണിച്ചിരിക്കുന്നത് പോലെ വെബ് പേജ് പ്രദർശിപ്പിക്കാവുന്നതാണ്.

ചിത്രം 4.18 : ഒരു പശ്ചാത്തല ചിത്രം

Bgcolor

ഡോക്യുമെന്റിന്റെ ബോഡി ഭാഗത്തെ പശ്ചാത്തലത്തിന് ഒരു നിറം നിർദേശിക്കുന്ന ആട്രിബ്യൂട്ടാണിത്. ഉദാഹരണമായി, <BODY Bgcolor = "grey"> എന്ന കോഡ് പശ്ചാത്തലത്തിന് ചാരനിറം നൽകുന്നു.

Bgcolor ആട്രിബ്യൂട്ടിന് രണ്ട് രീതിയിൽ വില നൽകാനാകും.

- ***Color_name*** - പശ്ചാത്തലത്തിന് നിറനാമം നിർദ്ദേശിക്കുന്നു ("red", "grey" തുടങ്ങിയവ പോലെ).
- ***Hex_number*** - പശ്ചാത്തലത്തിന് ഹെക്സാഡെസിമൽ കോഡിലുള്ള നിറം നിർദേശിക്കുന്നു. ("#ff6080", "#303030" തുടങ്ങിയവ പോലെ). ഓരോ പതിനാറക്ക കോഡും ഹാഷ് ചിഹ്നത്തിലാണ് ആരംഭിക്കുന്നത്.

Colour	Colour Name	Colour HEX
	Black	#000000
	Red	#FF0000
	Green	#00FF00
	Blue	#0000FF
	Yellow	#FFFF00
	Aqua	#00FFFF
	Grey	#C0C0C0
	White	#FFFFFF

പട്ടിക 4.5 : നിറങ്ങളുടെയും, അവയുടെ പേരിന്റെയും, 16 അക്ക കോഡ് വിലയുടെയും പട്ടിക

ആറക്ക സംഖ്യയും അക്ഷരങ്ങളും ചേർന്ന് നിറങ്ങൾക്ക് RGB (ചുവപ്പ്, പച്ച, നീല) വിലകൾ നൽകി പ്രദർശിപ്പിക്കുന്നു. ആറ് അക്കങ്ങളിൽ ആദ്യ രണ്ടെണ്ണം ചുവപ്പിന്റെ അളവിനെയും, അടുത്ത രണ്ട് അക്കങ്ങൾ പച്ചയുടെ തോതിനെയും, അവസാനത്തെ രണ്ടക്കങ്ങൾ നീലയുടെ അളവിനെയും 00 മുതൽ FF വരെയുള്ള വിലകൾ കൊണ്ട് സൂചിപ്പിക്കുന്നു. ഉദാഹരണമായി, #000000 എന്നത് കറുപ്പും, #FF0000 എന്നത് തിളക്കമുള്ള ചുവപ്പും, #00FF00 എന്നത് തിളങ്ങുന്ന പച്ചയും, #FFFFFF എന്നത് വെള്ളയുമായിരിക്കും (മൂന്ന് നിറങ്ങളും പൂർണ

മായും ഉൾക്കൊള്ളിച്ചിരിക്കുന്നു). ഹെക്സാഡെസിമൽ കോഡിലെ വിവിധ നിറക്കൂട്ടുകൾ നമുക്ക് പരീക്ഷിക്കാവുന്നതാണ്. ചില നിറങ്ങളും, അവയുടെ പേരുകളും, പതിനാറക്ക കോഡ് വിലകളും പട്ടിക 4.5-ൽ കാണിച്ചിരിക്കുന്നു.

Text

പേജിന്റെ ഉള്ളടക്കത്തിലുള്ള വാക്യങ്ങളുടെ നിറം സൂചിപ്പിക്കുന്ന ആട്രിബ്യൂട്ടാണ് ഇത്. തനത് രൂപത്തിൽ വെള്ള/ചാര നിറമുള്ള പശ്ചാത്തലവും വാക്യങ്ങൾക്ക് കറുപ്പ് നിറവുമായി ബ്രൗസർ കാണിക്കുന്നു. Bgcolor ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് പശ്ചാത്തലത്തിന്റെ നിറം എങ്ങനെ വ്യത്യാസപ്പെടുത്താമെന്ന് നാം ചർച്ച ചെയ്ത് കഴിഞ്ഞു. അതുപോലെ, Text ആട്രിബ്യൂട്ടുപയോഗിച്ചുകൊണ്ട് വാക്യങ്ങളുടെ നിറം വ്യത്യാസപ്പെടുത്താനാകും. ഉദാഹരണമായി, `<BODY Text = "yellow">` എന്നത് മഞ്ഞ നിറത്തിലുള്ള വാക്യത്തെ പ്രദർശിപ്പിക്കുന്നു. Bgcolor-ന് വില നൽകിയത് പോലെ നിറത്തിന്റെ പേരോ പതിനാറക്ക കോഡോ ഉപയോഗിച്ച് Text ആട്രിബ്യൂട്ടിന് വില നൽകാവുന്നതാണ്. ഉദാഹരണമായി Text = "Blue" അല്ലെങ്കിൽ Text = "#00FFDD" തുടങ്ങിയവ.

Link, Alink and Vlink

ക്ലിക്ക് ചെയ്യുമ്പോൾ, ഒരു ഡോക്യുമെന്റിൽ നിന്ന് മറ്റൊന്നിലേക്കോ, അല്ലെങ്കിൽ അതേ ഡോക്യുമെന്റിന്റെ വേറൊരു ഭാഗത്തേക്കോ നീങ്ങാൻ സഹായിക്കുന്ന ഒരു ഘടകം/ഒരു വാക്യം/ഒരു ചിത്രമാണ് ഹൈപ്പർലിങ്ക്. ഒരു ലിഖിതത്തെ മൊത്തമായോ അല്ലെങ്കിൽ ഡോക്യുമെന്റിലുള്ള പ്രത്യേക ഭാഗത്തെയോ ആണ് ഹൈപ്പർ ലിങ്ക് സൂചിപ്പിക്കുന്നത്. ഹൈപ്പർ ലിങ്കിനെക്കുറിച്ച് വിശദമായി പിന്നീട് നമുക്ക് ഈ അധ്യായത്തിൽ ചർച്ച ചെയ്യാം.

Link: കാണുന്നയാൾ സന്ദർശിക്കാത്ത ഹൈപ്പർ ലിങ്കിന്റെ നിറം സൂചിപ്പിക്കുന്നതിന് ഉപയോഗിക്കുന്ന ആട്രിബ്യൂട്ടാണ് ഇത്. Link ആട്രിബ്യൂട്ടിന്റെ തനത് നിറം നീലയായിരിക്കും.

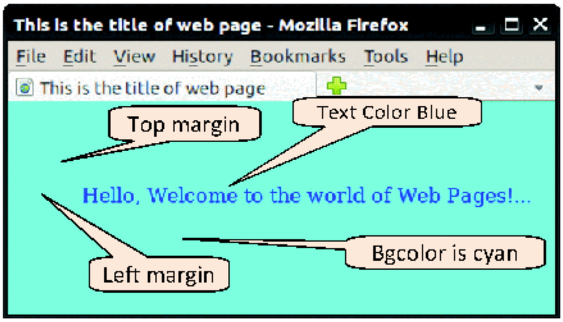
Alink: സജീവമായ ഹൈപ്പർ ലിങ്കിന്റെ നിറമാണ് ഇത് സൂചിപ്പിക്കുന്നത്. മൗസ് അതിനുമേൽ ക്ലിക്ക് ചെയ്യുന്ന നിമിഷം മാത്രമാണ് ആ ലിങ്ക് സജീവമാകുന്നത്. ആയതിനാൽ, തിരഞ്ഞെടുക്കുമ്പോൾ ഹൈപ്പർ ലിങ്കിന്റെ നിറം Alink ന്റെ വില സൂചിപ്പിക്കും. ഇവിടെ ഹൈപ്പർ ലിങ്കിന്റെ തനത് നിറം പച്ചയായിരിക്കും.

Vlink: കാണുന്നയാൾ സന്ദർശിച്ച ഹൈപ്പർ ലിങ്കിന്റെ നിറം സൂചിപ്പിക്കാനാണ് ഇത് നൽകുന്നത്. Vlink ന്റെ തനത് നിറം ധൂമനിറം (പർപ്പിൾ) ആയിരിക്കും.

Leftmargin, Topmargin

പേജിന്റെ അരികിൽ നിന്നുള്ള ശൂന്യസ്ഥലത്തെയാണ് മാർജിൻ സൂചിപ്പിക്കുന്നത്. ബ്രൗസർ ജാലകത്തിൽ ഡോക്യുമെന്റിന്റെ ഇടത് ഭാഗത്ത് കുറച്ച് ശൂന്യപ്രദേശം ഒഴിച്ചിടാനായി Leftmargin ആട്രിബ്യൂട്ടും മുകളറ്റത്ത് കുറച്ച് ശൂന്യ സ്ഥലം നൽകാനായി Topmargin ആട്രിബ്യൂട്ടും ഉപയോഗിക്കുന്നു. വിലനൽകുന്നത് പിക്സൽ പോയിന്റുകളുടെ എണ്ണമായാണ്.

ഉദാഹരണമായി, <BODY Leftmargin = "60" Topmargin = "70"> എന്നത്, പ്രദർശിപ്പിക്കേണ്ട വാചകം വെബ് പേജിന്റെ ഇടത്തേ അറ്റത്ത് നിന്ന് 60 പിക്സൽ പോയിന്റ് അകലെയും, മുകളറ്റത്ത് നിന്ന് 70 പിക്സൽ പോയിന്റ് അകലെയും ആണെന്ന് കാണിക്കുന്നു. <BODY> ടാഗിന്റെ Bgcolor, Text, Top margin, Leftmargin എന്നീ ആട്രിബ്യൂട്ടുകൾ ഉദാഹരണം 4.3-ലെ കോഡിലൂടെ വിശദീകരിച്ചിരിക്കുന്നു. ചിത്രം 4.19-ൽ അനുബന്ധ വെബ് പേജ് കാണാവുന്നതാണ്.



ചിത്രം 4.19 : BODY ടാഗിനോടൊപ്പം ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിച്ചിരിക്കുന്നു

ഉദാഹരണം 4.3: ഒരു വെബ് പേജിന്റെ പശ്ചാത്തലത്തിന് നിറം കൊടുക്കുന്നു

```
<HTML>
<HEAD>
     <TITLE> This is the title of web page </TITLE>
</HEAD>
<BODY Bgcolor= "cyan" Text= " Blue"
      Topmargin= "70" Leftmargin= "60">
      Hello, Welcome to the world of Web Pages!........
</BODY>
</HTML>
```

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. HTML പ്രതിനിധീകരിക്കുന്നത് ________ നെയാണ്.
2. എന്താണ് കണ്ടെയ്നർ ടാഗ്?
3. അവസാനിപ്പിക്കുന്ന ടാഗ് ഇല്ലാതെ തുടക്ക ടാഗ് മാത്രം വേണ്ട ഇനം ടാഗാണ് _______.
4. ശരിയോ തെറ്റോ എന്ന് പ്രസ്താവിക്കുക.
   a. ടാഗുകൾ കേയ്സ് സെൻസിറ്റീവ് ആണ്.
   b. <BODY> ടാഗിന്റെ ഒരു ആട്രിബ്യൂട്ടാണ് Bgcolor.
   c. ഒരു എംപ്റ്റി ടാഗാണ് <TITLE>.
   d. <HEAD> ടാഗിന്റെ ഒരു ആട്രിബ്യൂട്ടാണ് Dir.
5. <HTML> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകളുടെ പേരെഴുതുക.
6. ഒരു ടാഗിൽ ആട്രിബ്യൂട്ടുകളുടെ ഉപയോഗമെന്താണ്?
7. <BODY> ടാഗിന്റെ വിവിധ ആട്രിബ്യൂട്ടുകൾ എഴുതുക.

നമുക്കു ചെയ്യാം

നിങ്ങളുടെ വിദ്യാലയത്തിന്റെ പേരും അതോടൊപ്പം പശ്ചാത്തലത്തിൽ വിദ്യാലയത്തിന്റെ ചിത്രവും പ്രദർശിപ്പിക്കുന്ന ഒരു HTML ലിഖിതം നിർമിക്കുക. തുടർന്ന് വാക്യത്തിന്റെ നിറവും പശ്ചാത്തലവും വ്യത്യാസപ്പെടുത്തി പേജ് പരിഷ്കരിക്കുക.

4.10 പൊതുവായ ചില ടാഗുകൾ (Some common tags)

HTML ഡോക്യുമെന്റിനു വേണ്ട അടിസ്ഥാന ടാഗുകളെയും അവയുടെ ആട്രിബ്യൂട്ടുകളെയും കുറിച്ച് നാം ചർച്ച ചെയ്തു കഴിഞ്ഞു. വെബ് പേജിലെ ഉള്ളടക്കം കൂടുതൽ ആകർഷകമാക്കാൻ ഉപയോഗിക്കുന്ന മറ്റു ടാഗുകളുമുണ്ട്. ഇക്കൂട്ടത്തിൽ, HTML ലിഖിതത്തിന്റെ ബോഡി ഭാഗത്തെ ഉള്ളടക്കത്തിലുള്ള വാക്യങ്ങളുടെ ഘടന രൂപപ്പെടുത്താൻ ഉപയോഗിക്കുന്ന ടാഗുകളെ ഫോർമാറ്റിങ് ടാഗുകൾ എന്ന് വിളിക്കുന്നു. ബോഡി ഭാഗത്തെ ഉള്ളടക്കത്തിന്റെ രൂപഘടനയ്ക്ക് ആവശ്യമായ ടാഗുകളിൽ ചിലത് നമുക്ക് പരിചയപ്പെടാം.

4.10.1 <H1>, <H2>, <H3>, <H4>, <H5>, <H6> - ശീർഷക ടാഗുകൾ (Heading tags)

ഒരു ഖണ്ഡികയുടെ തുടക്കത്തിൽ അത് എന്താണെന്ന് വിശദീകരിക്കുന്നതിനായി കൊടുക്കുന്ന ഒരു വാക്കോ, വാക്യമോ, വാചകമോ ആണ് ശീർഷകം. ബോഡി ഭാഗത്തെ സാധാരണ വാക്യങ്ങളെക്കാൾ വലുതോ കടുപ്പിച്ചതോ ആയ സവിശേഷതയുള്ള അക്ഷരങ്ങൾ കൊണ്ടാണ് ശീർഷകം പ്രദർശിപ്പിക്കുന്നത്. **<H1>** മുതൽ **<H6>** വരെ ആറ് തലങ്ങളിലുള്ള ശീർഷകങ്ങൾ HTML-ൽ ഉണ്ട്. ഇവിടെ **<H1>** ഏറ്റവും വലുതും **<H6>** ഏറ്റവും ചെറുമായ ശീർഷകങ്ങളാണ് നിർമിക്കുന്നത്. ശീർഷകം പ്രദർശിപ്പിക്കുന്നതിന് മുന്നിലും പിന്നിലുമായി ബ്രൗസർ ഓരോവരി ചേർക്കുന്നു. ഈ ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ട് **Align** ആണ്. അതിന് നൽകാവുന്ന വിലകൾ താഴെ തന്നിരിക്കുന്നു.

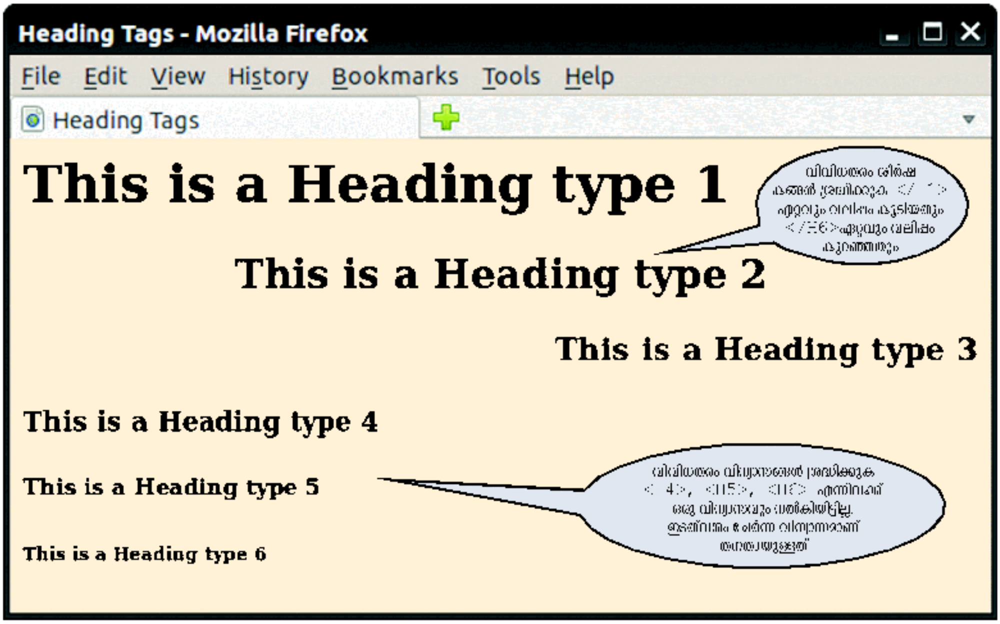
Left : വാക്യം ഇടത് അരികിനോട് (മാർജിനുമായി) പൊരുത്തപ്പെട്ട് നിൽക്കും.
Right : വാക്യം വലത് അരികിനോട് (മാർജിനുമായി) പൊരുത്തപ്പെട്ട് നിൽക്കും.
Center : വാക്യം പേജിന്റെ മാധ്യഭാഗത്ത് പ്രത്യക്ഷപ്പെടും.

വിവിധങ്ങളായ ശീർഷകങ്ങളും, പൊരുത്തപ്പെടലുകളും ഉദാഹരണം 4.4 - ൽ കാണിച്ചിരിക്കുന്നതോടൊപ്പം അനുബന്ധ വെബ് പേജ് ചിത്രം 4.20-ൽ പ്രദർശിപ്പിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.4: വിവിധ ശീർഷക ശൈലികൾ വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> Heading Tags </TITLE>
</HEAD >
<BODY Bgcolor= "#FFEFD5">
   <H1 Align= "left"> This is a Heading type 1 </H1>
   <H2 Align= "center"> This is a Heading type 2 </H2>
     <H3 Align= "right"> This is a Heading type 3 </H3>
     <H4> This is a Heading type 4 </H4>
```

```
    <H5> This is a Heading type 5 </H5>
    <H6> This is a Heading type 6 </H6>
</BODY>
</HTML>
```



ചിത്രം 4.20 : വ്യത്യസ്ത ശീർഷക ടാഗുകളോടൊപ്പമുള്ള വിവിധ പൊരുത്തപ്പെടലുകൾ

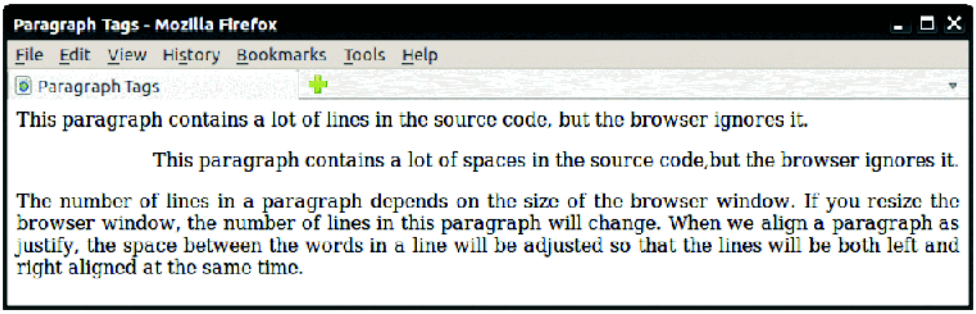
4.10.2 <P> ടാഗ് – ഖണ്ഡികകൾ നിർമിക്കുന്നു (Creating Paragraphs)

<BODY> ടാഗിലെ വാക്യങ്ങളെ വിവിധ ഖണ്ഡികകളിൽ ആക്കുവാൻ നമ്മെ സഹായിക്കുന്നത് **<P>** ടാഗാണ്. ഇത് ഒരു പുതിയ ഖണ്ഡികയെ സൂചിപ്പിക്കുകയും ഒരു ശൂന്യവരി ഖണ്ഡികയ്ക്ക് മുന്നിൽ ചേർക്കാൻ ബ്രൗസറിനോട് നിർദേശിക്കുകയും ചെയ്യുന്നു. മറ്റ് ഏതൊരു വേർഡ് പ്രൊസ്സസറിലെയും ഖണ്ഡികകൾ പോലെ തന്നെയാണ് HTML-ലെ ഖണ്ഡികയും. ഒരു ഖണ്ഡിക <P> ടാഗിൽ തുടങ്ങി </P> ടാഗിൽ അവസാനിക്കുന്നു. `Left`, `right`, `center`, `justify` എന്നീ വിലകൾ നൽകി ഖണ്ഡികയിലെ വാക്യങ്ങളുടെ വിന്യാസം ഉറപ്പിക്കുന്നത് **`Align`** ആട്രിബ്യൂട്ട് ഉപയോഗിച്ചു കൊണ്ടാണ്. വ്യത്യസ്ത വിന്യാസം നൽകി ഖണ്ഡികകൾ എങ്ങനെ രൂപപ്പെടുത്തുന്നുവെന്ന് ഉദാഹരണം 4.5-ലുള്ള കോഡിൽ തന്നിരിക്കുന്നു. അങ്ങനെ ലഭിക്കുന്ന വെബ് പേജ് ചിത്രം 4.21-ൽ കാണിച്ചിരിക്കുന്നു

ഉദാഹരണം 4.5: വിവിധ വിന്യാസത്തോട് കൂടിയ ഖണ്ഡികകൾ രൂപകൽപന ചെയ്യുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
    <TITLE> Paragraph Tags </TITLE>
```

```
</HEAD>
<BODY>
      <P>
      This paragraph contains
      a lot of lines in the source code,
      but the browser ignores it.
      </P>
      <P Align= "right">
      This paragraph contains a lot of spaces
      in the source code,but the browser ignores it.
      </P>
      <P Align= "justify">
      The number of lines in a paragraph depends on
      the size of the browser window.
      If you resize the browser window, the number of lines
      in this paragraph will change.
      When we align a paragraph as justify, the space between
      the words in a line will be adjusted
      so that the lines will be both left and right aligned
      at the same time.
    </P>
</BODY>
</HTML>
```



ചിത്രം 4.21 : Align ആട്രിബ്യൂട്ട് ഉൾക്കൊള്ളിച്ചിട്ടുള്ള <P> ടാഗിന്റെ ഉപയോഗം

ചിത്രം 4.5-ലെ സോഴ്സ് കോഡിൽ അധിക ശൂന്യസ്ഥലങ്ങളോട് കൂടിയ മൂന്ന് വരികൾ ഒന്നാം ഖണ്ഡികയിലും, രണ്ട് വരികൾ രണ്ടാം ഖണ്ഡികയിലും ഉൾക്കൊള്ളിച്ചിട്ടുള്ളത് നമുക്ക് കാണാനാകും. എന്നാൽ ചിത്രം 4.21-ൽ കാണിച്ചിട്ടുള്ള ഈ കോഡിന്റെ വെബ് പേജിൽ ആദ്യത്തെ രണ്ട് ഖണ്ഡികകൾക്ക് ഓരോ വരിയാണുള്ളത്. അതായത്, പേജ് പ്രദർശിപ്പിക്കുമ്പോൾ, അധികമുള്ള വരികളും ശൂന്യസ്ഥലങ്ങളും ബ്രൗസർ നീക്കം ചെയ്യുന്നു. രണ്ടാം ഖണ്ഡിക വലതുവശത്ത് പൊരുത്തപ്പെട്ട് നിൽക്കുന്നതും മൂന്നാം ഖണ്ഡിക ഇരുവശങ്ങളിലും പൊരുത്തപ്പെട്ട് നിൽക്കുന്നതും ശ്രദ്ധിക്കുക. ഒന്നിലധികം

ശൂന്യസ്ഥലങ്ങളും വരികളും ഒരൊറ്റ ശൂന്യസ്ഥലമായാണ് കരുതുന്നത്. അതുകൊണ്ട്, HTML കോഡിൽ അധികമായി ശൂന്യസ്ഥലങ്ങളോ, വരികളോ ചേർക്കുന്നതുകൊണ്ട് ഔട്ട്പുട്ട് വ്യത്യാസപ്പെടുത്താനാകില്ല. എന്നാൽ ഇത്
 ടാഗ് ഉപയോഗിച്ച് സാധ്യമാകും. ഉള്ളടക്ക വാക്യങ്ങളിൽ
 ടാഗ് ഉപയോഗിച്ചുകൊണ്ട് അധികവരി ചേർക്കാം. വലുതോ ചെറുതോ ആയ സ്ക്രീനുകളിലും, വലുപ്പം വ്യത്യാസപ്പെടുത്തിയ വിൻഡോകളിലും ലഭിക്കുന്ന വെബ് പേജുകൾ, ചിത്രം 4.21-ൽ തന്നിട്ടുള്ളതിന് സമാനമായിരിക്കുകയില്ല എന്നത് ശ്രദ്ധിക്കുക.

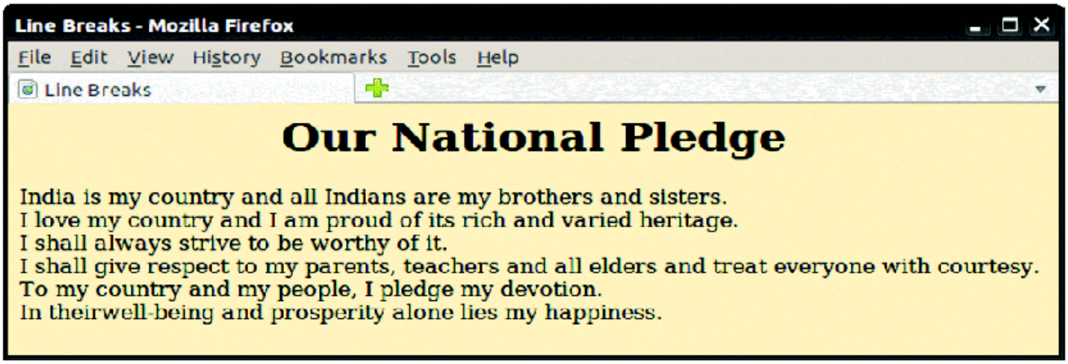
4.10.3
 ടാഗ് - വരി വിഘടിപ്പിക്കുന്നു (Inserting line break)

ഒരു വെബ് പേജിലെ വരി മാറ്റിയെഴുതുക എന്നതാണ്
 ടാഗിന്റെ ഉദ്ദേശം. നിലവിലുള്ള വരിയിലെ വാക്യം മുറിച്ച് അടുത്ത വരിയുടെ തുടക്കത്തിൽ നിന്നാരംഭിക്കാനാണ്
 ടാഗുപയോഗിക്കുന്നത്. ഇതൊരു എംറ്റി ടാഗാണ്.

ഉദാഹരണം 4.6-ൽ തന്നിട്ടുള്ള HTML കോഡ് നമ്മുടെ രാജ്യത്തിന്റെ പ്രതിജ്ഞ പ്രദർശിപ്പിക്കുന്നു. ചിത്രം 4.22-ൽ കാണിച്ചിട്ടുള്ള പരിണത ഫലമായി ലഭിക്കുന്ന പേജിൽ
 ടാഗിന്റെ ഉപയോഗവും
, <P> എന്നീ ടാഗുകൾ തമ്മിലുള്ള വ്യത്യാസവും നമുക്ക് കാണാം.

ഉദാഹരണം 4.6: ദേശീയ പ്രതിജ്ഞയിൽ വരികൾ മുറിക്കുന്നത് (Line break) കാണിക്കുവാൻ വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> Line Breaks </TITLE>
</HEAD >
<BODY Bgcolor = "#FFEFC5">
   <H1 Align = "center"> Our National Pledge </H1>
   <P>India is my country and all Indians
     are my brothers and sisters.<BR>
     I love my country and I am proud of its
     rich and varied heritage.<BR>
     I shall always strive to be worthy of it.<BR>
     I shall give respect to my parents, teachers and all
     elders and treat everyone with courtesy.<BR>
     To my country and my people, I pledge my
     devotion. <BR>In theirwell-being and prosperity alone
     lies my happiness.
   </P>
</BODY>
</HTML>
```



ചിത്രം 4.22 :
 ടാഗിന്റെ ഉപയോഗം കാണിക്കുന്ന ഔട്ട്പുട്ട്

നമുക്കു ചെയ്യാം

*ഉദാഹരണം 4.6, ചിത്രം 4.24 എന്നിവ നൽകുന്ന സൂചനയ്ക്ക് അനുസൃതമായി, <P> ടാഗും
 ടാഗും തമ്മിൽ വേർതിരിക്കുന്ന ആശയങ്ങൾകൊണ്ട് പട്ടിക 4.5 പൂരിപ്പിക്കുക.*

<P> ടാഗ്	 ടാഗ്
	നിലവിലുള്ള വരി ഖണ്ഡിക്കുകയും പുതിയ വരിയിൽ തുടരുകയും ചെയ്യുന്നു.
കണ്ടെയ്നർ ടാഗ്	

പട്ടിക 4.6 : <P> ടാഗും
 ടാഗും

4.10.4 <HR> ടാഗ് - തിരശ്ചീനമായ വര സൃഷ്ടിക്കുന്നു (Creating horizontal line)

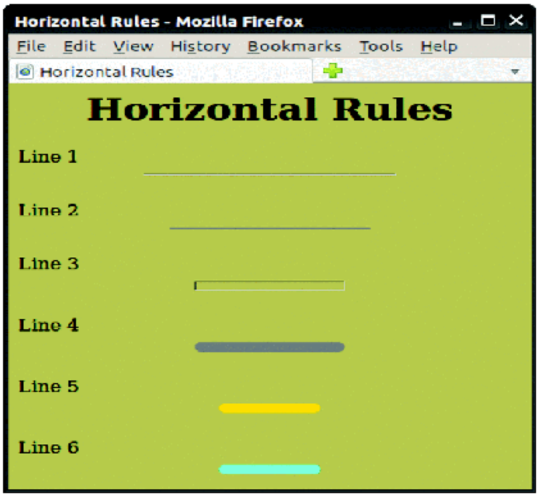
ബ്രൗസർ വിൻഡോയുടെ വീതിക്ക് കുറുകേ തിരശ്ചീനമായ ഒരു വര നിർമിക്കുന്നത് **<HR>** ടാഗാണ്. അതിന്റെ ആട്രിബ്യൂട്ടുകളായ **Size, Width** എന്നിവ ഉപയോഗിച്ച് നമുക്ക് വരയുടെ **വീതിയും കനവും** വ്യത്യാസപ്പെടുത്താം. കനത്തിന് പിക്സലുകളായും വീതിക്ക് പിക്സലുകളായോ മൊത്തം വീതിയുടെ ശതമാനമായോ വില നൽകുന്നു. <HR> ടാഗിന്റെ മറ്റ് രണ്ട് ആട്രിബ്യൂട്ടുകളാണ് Noshade, Color എന്നിവ. Noshade ആട്രിബ്യൂട്ടിന് പ്രത്യേക വില കാണുകയില്ല. വരയ്ക്ക് നിറം കൊടുക്കുന്നത് Color ആട്രിബ്യൂട്ടാണ്. left, center, right എന്നിങ്ങനെ വരയുടെ വിന്യാസത്തിന് ഉപയോഗിക്കുന്ന Align ആട്രിബ്യൂട്ടാണ് മറ്റൊന്ന്. <HR> ഒരു എംപ്റ്റി ടാഗാണ് എന്ന് ഓർക്കുക.

<HR> ടാഗിന്റെയും അതിന്റെ ആട്രിബ്യൂട്ടുകളുടെയും പ്രവർത്തനം വ്യക്തമാക്കുന്ന കോഡ് ഉദാഹരണം 4.7-ൽ തന്നിട്ടുണ്ട്. അതിന്റെ ഔട്ട്പുട്ട് ചിത്രം 4.23-ൽ പ്രദർശിപ്പിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.7: വിവിധ തരത്തിലുള്ള രേഖകൾ വരയ്ക്കുന്നതിന്

```
<HTML>
<HEAD>
```

```
   <TITLE> Horizontal Rules </TITLE>
</HEAD >
<BODY Bgcolor= "#BDB76B">
<H1 Align= "center"> Horizontal Rules </H1>
Line 1<HR Width= "50%" Align= "center"> <BR>
Line 2<HR Width= "40%" Align= "center" Noshade> <BR>
Line 3<HR Size= "10" Width= "30%" Align= "center"> <BR>
Line 4<HR Size= "10" Width= "30%" Align= "center" Noshade><BR>
Line 5<HR Size= "10" Width= "20%" Align= "center" Noshade
          Color= "gold"> <BR>
Line 6<HR Size="10" Width="20%" Align="center" Color="Aqua">
</BODY>
</HTML>
```



ചിത്രം 4.23 : പലതരത്തിലുള്ള തിരശ്ചീനമായ രേഖകൾ

ഈ HTML ലിഖിതവും അതിന്റെ ഔട്ട്പുട്ടും പരിശോധിച്ചശേഷം, വരകളുടെ ആകൃതികളിലുള്ള സമാനതകളും വ്യത്യാസങ്ങളും നിങ്ങൾക്ക് തിരിച്ചറിയാനാകുമോ? ഇവിടെ, ആദ്യത്തെ രേഖ Size, Noshade എന്നീ ആട്രിബ്യൂട്ടുകളില്ലാതെയും, രണ്ടാമത്തെ രേഖ Size ആട്രിബ്യൂട്ടില്ലാതെയുമാണ് വരച്ചിരിക്കുന്നത്. രേഖ മൂന്നിനും, നാലിനും വലുപ്പം 10 ആണ്. എന്നാൽ മൂന്നാമത്തേതിന് Noshade ആട്രിബ്യൂട്ട് ഇല്ല. അവസാനത്തെ രണ്ട് രേഖകൾക്ക് Color ആട്രിബ്യൂട്ട് അധികമായുണ്ട്. ഓരോ അവസ്ഥയിലുമുള്ള ഈ രേഖകളുടെ ആകൃതികൾ ശ്രദ്ധിക്കുക. നിങ്ങളുടെ കമ്പ്യൂട്ടർ ലാബിലെ പ്രവർത്തനത്തിൽ സാധ്യമായ എല്ലാവിലകളും വ്യത്യസ്തങ്ങളായ ആട്രിബ്യൂട്ടുകൾക്ക് നൽകിക്കൊണ്ട് നിങ്ങൾക്ക് കോഡ് മാറ്റാവുന്നതും വെബ് പേജിൽ അപ്പോഴുണ്ടാകുന്ന മാറ്റം ദർശിക്കാവുന്നതുമാണ്.

4.10.5 <CENTER> ടാഗ് – ഉള്ളടക്കം മധ്യഭാഗത്താക്കുന്നു (Centering the content)

വെബ് പേജിലെ ഉള്ളടക്കം തിരശ്ചീനമായി മധ്യഭാഗത്തേക്ക് കൊണ്ടു വരുന്നത് **<CENTER>** ടാഗാണ്. ഉള്ളടക്കം സാധാരണയായി വാക്യങ്ങൾ, ചിത്രങ്ങൾ, പട്ടികകൾ തുടങ്ങിയവയായിരിക്കും. ഇത് ഒരു കണ്ടെയ്നർ ടാഗ് ആയതുകൊണ്ട് <CENTER>, </CENTER> എന്നീ ടാഗ് ജോഡികൾക്ക് ഇടയ്ക്ക് നൽകുന്ന ഉള്ളടക്കം ബ്രൗസർ ജാലകത്തിന്റെ മധ്യഭാഗത്ത് നിർത്തുന്നു. ഈ ടാഗിന് ആട്രിബ്യൂട്ടുകൾ ഇല്ല.

4.10.6 വാക്യങ്ങൾക്ക് രൂപഘടന നൽകുന്ന ടാഗുകൾ (Text formating tags)

വേഡ് പ്രോസസിംഗ് പ്രോഗ്രാമുകൾ ചെയ്യുന്നതു പോലെ, നമുക്ക് വെബ് പേജിലെ വാക്യങ്ങൾക്ക് രൂപഘടന നൽകാനാകും. വാക്യങ്ങളുടെ പ്രാധാന്യത്തെ സൂചിപ്പിക്കുന്നതിന് ഉപയോഗിക്കുന്ന സവിശേഷതകളാണ് ബോൾഡ്, ഇറ്റാലിക്, അണ്ടർലൈൻ തുടങ്ങിയവ. ഇതിനായുള്ള HTML ടാഗുകളെക്കുറിച്ച് നമുക്ക് ചർച്ച ചെയ്യാം.

<B> - ടെക്സ്റ്റ് ബോൾഡ് ആക്കുന്നു

ഈ ടാഗ് ടെക്സ്റ്റിന്റെ ശൈലി ബോൾഡ് ആക്കുന്നു. ഉള്ളടക്കത്തിലെ വാക്യങ്ങളുടെ ബാഹ്യരൂപം കൂടുതൽ കനത്തിൽ പ്രദർശിപ്പിക്കുന്നത് <B> ടാഗാണ്.

<I> - ടെക്സ്റ്റ് ഇറ്റാലിക്സിലാക്കുന്നു

ഇത് വാക്യങ്ങൾക്ക് ചരിവ് നൽകി രൂപപ്പെടുത്തുന്നു. <I> ടാഗിനും </I> ടാഗിനുമിടയ്ക്കുള്ള ഉള്ളടക്കത്തിന് ചരിവ് നൽകുന്നു.

<U> - ടെക്സ്റ്റിന് അടിവര ഇടുന്നു

വെബ് പേജിലെ വാക്യത്തിന് അടിവര നൽകാനാണ് <U> ടാഗുപയോഗിക്കുന്നത്. <U> ടാഗിനും </U> ടാഗിനുമിടയ്ക്ക് ഉൾക്കൊള്ളിച്ചിട്ടുള്ള ഉള്ളടക്കം അടിവര ഉള്ളതായി മാറുന്നു. രൂപഘടന നൽകുന്ന ടാഗുകളായ <U>, <B>, <I> എന്നിവ ഒന്നിച്ച് ഉപയോഗിക്കാനാകും എന്നതിനാൽ, ഉള്ളടക്കം കനം കൂടിയതും, ചരിഞ്ഞതും, അടിവരയുള്ളതുമാകും.

<S>, <STRIKE> - ടെക്സ്റ്റിന് കുറുകെ വരയ്ക്കുന്നു

<S>, <STRIKE> എന്നീ ടാഗുകൾ ഉപയോഗിക്കുന്നത് ഒരേ പ്രക്രിയയ്ക്കാണ്. വാക്യങ്ങൾ കുറുകെ വെട്ടിയ രൂപത്തിൽ പ്രദർശിപ്പിക്കുന്നു. ഉദാഹരണമായി, കുറുകേ വെട്ടിയ വാക്യമാണ് ~~Thank you all~~.

<BIG> - ടെക്സ്റ്റ് വലുപ്പം ഉള്ളതാക്കുന്നു

സാധാരണ വലുപ്പത്തിലുള്ള ടെക്സ്റ്റിനേക്കാൾ വലുപ്പം ഉള്ളടക്കത്തിന് നൽകാനാണ് <BIG> ടാഗ് ഉപയോഗിക്കുന്നത്. ഡോക്യുമെന്റിലെ വാക്കുകൾക്ക് അല്ലെങ്കിൽ വരികൾക്ക് പ്രത്യേക പ്രാധാന്യം നൽകാനാണ് ഇത് ഉപയോഗിക്കുന്നത്. സാധാരണ ഗതിയിൽ <BIG> ടാഗിലെ അക്ഷരങ്ങളുടെ വലുപ്പം നിലവിലുള്ള വലുപ്പത്തേക്കാൾ ഒരുപടി കൂടുതലായിരിക്കും.

<SMALL>- ടെക്സ്റ്റ് ചെറുതാക്കുന്നു

ടെക്സ്റ്റുകളുടെ നിലവിലുള്ള വലുപ്പത്തേക്കാൾ ചെറുതാക്കാനാണ് <SMALL> ടാഗ് ഉപയോഗിക്കുന്നത്. സാധാരണയായി, <SMALL> ടാഗിലെ അക്ഷരങ്ങളുടെ വലുപ്പം, നിലവിലുള്ള അക്ഷരങ്ങളുടെ വലുപ്പത്തേക്കാൾ ഒന്ന് കുറവായിരിക്കും.

<STRONG> - ടെക്സ്റ്റ് കൂടുതൽ കനമുള്ളതാക്കുന്നു

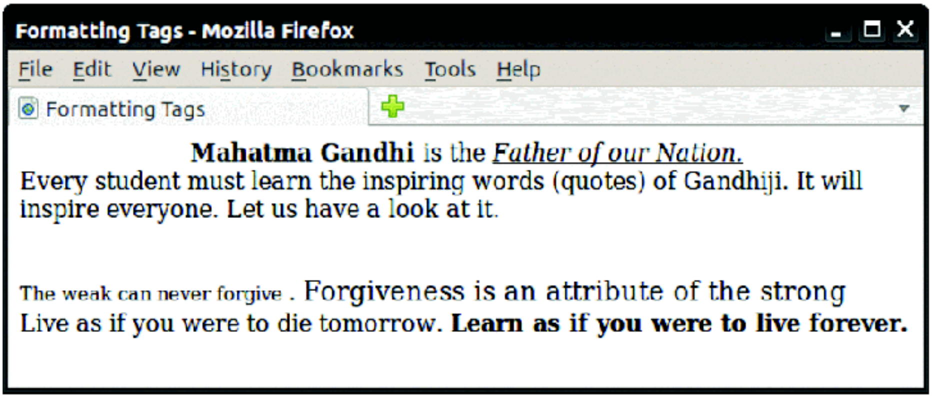
ഒരു ഉപവാക്യ ടാഗാണ് <STRONG> ഇത് പ്രാധാന്യമുള്ള വാക്യത്തെ നിർവചിക്കുന്നു. <STRONG> വാക്യം സാധാരണയായി കൂടുതൽ കനമുള്ളതാക്കിയ ബാഹ്യാകൃതിയോട്

കൂടിയാണ് ആവിഷ്കരിക്കുന്നത്. <B> ടാഗിനെപ്പോലെയാണിത്. വാക്യരൂപത്തിലുള്ള ഉള്ളടക്കത്തിലെ ഒരു ഉപവാക്യത്തിന് കൂടുതൽ പ്രാധാന്യം നൽകാനാണ് <STRONG> ടാഗ് ഉപയോഗിക്കുന്നത്.

ഉദാഹരണം 4.8-ൽ തന്നിട്ടുള്ള കോഡ് ഈ ടാഗുകളുടെ പ്രായോഗികത വിവരിക്കുന്നു. സന്ദർഭോചിതമായി, നമുക്ക് മഹാത്മാഗാന്ധിയുടെ വാക്കുകൾ ഉദ്ധരിക്കാം. ഈ കോഡ് ആവിഷ്കരിക്കുന്നതിന്റെ ഫലമായുണ്ടാകുന്ന വെബ് പേജ് ചിത്രം 4.24-ൽ കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.8: വാക്യങ്ങൾ രൂപപ്പെടുത്തുന്ന ടാഗുകളെക്കുറിച്ച് വിവരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> Formatting Tags </TITLE>
</HEAD>
<BODY>
   <P>
   <CENTER><B>Mahatma Gandhi </B>is the <I> <U> Father of
              our Nation.</U> </I> </CENTER>
   Every student must learn the inspiring words (quotes)
   of Gandhiji. It will inspire everyone. Let us have a
   look at it.
   </P><BR>
   <SMALL> The weak can never forgive </SMALL>.
   <BIG> Forgiveness is an attribute of the strong</BIG>
   <BR> Live as if you were to die tomorrow.
   <STRONG> Learn as if you were to live forever.</STRONG>
</BODY>
</HTML>
```



ചിത്രം 4.24 : വാക്യഘടന രൂപപ്പെടുത്തുന്ന ടാഗുകളെക്കുറിച്ചുള്ള വിവരണം

വാക്യങ്ങൾക്ക് ചില പ്രത്യേക രൂപം നൽകുന്ന ടാഗുകൾ നമുക്ക് പരിചയപ്പെടാം.

<EM> - വാക്യങ്ങൾക്ക് പ്രത്യേക പ്രാധാന്യം നൽകുന്നു

വാക്യങ്ങൾക്ക് പ്രത്യേക പ്രാധാന്യം നൽകാൻ ഉപയോഗിക്കുന്ന ടാഗാണ് <EM>. പ്രയോഗത്തിൽ വരുമ്പോൾ, ആ ഭാഗം സാധാരണ ഇറ്റാലിക്സിലായിരിക്കും ആവിഷ്കരിക്കുന്നത്. <I> ടാഗുപയോഗിക്കുമ്പോഴുള്ള ഗുണം തന്നെയാകും <EM> ടാഗും തരുന്നത്.

<SUB>, <SUP> എന്നീ ടാഗുകൾ - സബ് സ്ക്രിപ്റ്റും സൂപ്പർ സ്ക്രിപ്റ്റും നിർമ്മിക്കുന്നു

ജലം, സൽഫ്യൂരിക് അമ്ലം തുടങ്ങിയവയുടെ രാസനാമം വാക്യം ഹൈസ്കൂൾ ക്ലാസ്സുകളിൽ നാം പഠിച്ചിട്ടുള്ളതാണ്. അവ യഥാക്രമം H_2O, H_2SO_4 എന്നിങ്ങനെ ആയിരിക്കും. HTML-ൽ അത്തരം പ്രതീകങ്ങൾ നമുക്ക് എങ്ങനെ ചിത്രീകരിക്കാനാകും? അക്കങ്ങൾ സബ്സ്ക്രിപ്റ്റ് രൂപത്തിലാണ് എഴുതിയിട്ടുള്ളതെന്ന് നമുക്ക് കാണാം. ഒരു വെബ് പേജിൽ സബ്സ്ക്രിപ്റ്റ് നിർമിക്കാനാണ് <SUB> ടാഗുപയോഗിക്കുന്നത്. `H<SUB>2</SUB>O` എന്ന കോഡുപയോഗിച്ച് കൊണ്ട് H_2O എന്ന വാക്യം നമുക്ക് പ്രദർശിപ്പിക്കാനാകും.

അതുപോലെ, $(a+b)^2 = a^2 + 2ab + b^2$ പോലുള്ള ബീജഗണിത ആവിഷ്കാര രീതിയിലെ സൂപ്പർ സ്ക്രിപ്റ്റുകൾ <SUP> ടാഗ് ഉപയോഗിച്ച് നമുക്ക് സൂചിപ്പിക്കാനാകും. ഈ സൂത്രവാക്യം നമുക്ക് ഇങ്ങനെ എഴുതാം.

```
(a+b)<SUP>2</SUP> = a<SUP>2</SUP> + 2ab + b<SUP>2</SUP>
```

<BLOCKQUOTE>, <Q> എന്നീ ടാഗുകൾ - ഒരു ഉദ്ധരണി പ്രദർശിപ്പിക്കുന്നു

<BLOCKQUOTE> ടാഗ് അവയ്ക്കിടയിൽ ഉൾക്കൊള്ളിച്ചിട്ടുള്ള ഉള്ളടക്കം മാറ്റി പ്രദർശിപ്പിക്കാൻ ഉപയോഗിക്കുന്നു. വാക്യങ്ങളെ ഇരട്ട ഉദ്ധരണികൾക്കുള്ളിൽ മാറ്റി പ്രദർശിപ്പിക്കുന്നത് സൂചിപ്പിക്കാനാണ് <Q> ടാഗ് ഉപയോഗിക്കുന്നത്. ഖണ്ഡിക മുറിക്കാതെയുള്ള ചുരുങ്ങിയ ഉദ്ധരണികൾക്ക് വേണ്ടിയുള്ളതാണ് ഈ ടാഗ്. എന്നാൽ <BLOCKQUOTE> വലിയ ഉദ്ധരണികൾക്ക് ഉപയോഗിക്കാൻ വേണ്ടിയുള്ളതാണ്.

ഉദാഹരണം 4.9-ൽ മുകളിൽ സൂചിപ്പിച്ചിട്ടുള്ള ടാഗുകളുടെ വിശദീകരണമുണ്ട്. ലോക പരിസ്ഥിതി ദിനം ആഘോഷിക്കുന്നത് ജൂൺ 5-ന് ആണ് എന്നത് നമുക്ക് അനുസ്മരിക്കാം. ഈ ദിവസത്തേക്കുള്ള ചിന്തകൾ നാം സൃഷ്ടിക്കുകയും ഇത്തരം ടാഗുകളുപയോഗിച്ച് ഒരു വെബ് പേജ് നിർമിക്കുകയും ചെയ്യാം. ചിത്രം 4.25-ൽ അനുബന്ധമായ വെബ് പേജ് കാണിച്ചിട്ടുണ്ട്.

ഉദാഹരണം 4.9: <SUP>, <BLOCKQUOTE>, <Q> എന്നീ ടാഗുകൾ വിശദീകരിക്കുന്നതിന്

```
<HTML>
<HEAD>
```

```
    <TITLE> BlockQuote and Q tags  </TITLE>
</HEAD>
<BODY Bgcolor= "#98FB98" Text= "#008000">
    Every year we celebrate World Environment Day on 5<SUP>
    th</SUP> June. Let us have a message to all on this
    occassion.
    <BLOCKQUOTE> <B>June 5<SUP>th</SUP> is World Environment
      Day. </B>
      Mother nature too needs care and protection. Show her
      your care by caring for her trees. Love trees and love
      nature. And work for a greener environment because
      generations have to come... The future depends on us...
    </BLOCKQUOTE>
    <Q>Keep your world clean and green. Save trees, Save the
       environment!!
    </Q>
</BODY>
</HTML>
```

ആദ്യ ഖണ്ഡിക ആരംഭിക്കുന്നത് ബ്രൗസർ ജാലകത്തിന്റെ ഇടത് മൂലയിൽ നിന്നാണെന്ന് ചിത്രം 4.25-ൽ നിന്നും നമുക്ക് മനസിലാക്കാം. അതിൽ സൂപ്പർ സ്ക്രിപ്റ്റ് പ്രയോഗിച്ചിട്ടുണ്ട്. <BLOCKQUOTE> ടാഗ് ഉപയോഗിച്ചതിനാൽ രണ്ടാമത്തെ ഖണ്ഡിക മാറ്റി പ്രദർശിപ്പിച്ചു കൊണ്ടാണ് തുടങ്ങിയിരിക്കുന്നത്. അവസാനമായി, മൂന്നാമത്തെ ഖണ്ഡിക ഇരട്ട ഉദ്ധരണികൾക്ക് ഉള്ളിലാക്കിയിരിക്കുന്നു. അതിന്റെ ഉള്ളടക്കം <Q>, </Q> ടാഗ് ജോഡികൾക്ക് ഉള്ളിൽ ഉൾപ്പെടുത്തിയിരിക്കുന്നു. ഈ ഖണ്ഡികകളുടെ വിന്യാസം കൂടി നിരീക്ഷിക്കുക. <BLOCK QUOTE> ടാഗിൽ ഒന്നിലധികം ഖണ്ഡികകൾ ഉൾപ്പെടുത്തിയിരിക്കുന്നത് ശ്രദ്ധിക്കുക.

ചിത്രം 4.25 : ഉദാഹരണം 4.9 ലെ കോഡിന് അനുബന്ധമായ വെബ് പേജ്

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. വാക്യങ്ങൾക്ക് ഘടന നൽകുന്ന ചില ടാഗുകളുടെ പേരെഴുതുക.
2. <HR> ടാഗിന്റെ വ്യത്യസ്ത ആട്രിബ്യൂട്ടുകളുടെ നാമാവലി തയാറാക്കുക.
3. എത്ര തലങ്ങളിലുള്ള ശീർഷക ടാഗുകൾ HTML -ൽ ലഭ്യമാണ്?
4. x^3+y^3 എന്ന് പ്രദർശിപ്പിക്കുന്നതിനുള്ള HTML കോഡ് ശകലം എഴുതുക.
5. ശരിയോ തെറ്റോ എന്ന് പ്രസ്താവിക്കുക.
   a.
 ടാഗ് ഒരു എംപ്റ്റി ടാഗാണ്.
   b. HTML ഡോക്യുമെന്റിൽ <EM> ടാഗിനും <I> ടാഗിനും ഒരേ ഉപയോഗമാണുള്ളത്.
   c. <U> ടാഗിന്റെയും <I> ടാഗിന്റെയും ഒരുമിച്ചുള്ള ഉപയോഗം അനുവദനീയമല്ല.
6. <STRONG> ടാഗിന്റെ ഉപയോഗമെന്താണ്?
7. <STRONG> ടാഗിന്റേതിന് സമാനമായ കൃത്യം നിർവഹിക്കുന്ന ടാഗ് ഏതാണ്?
8. താഴെതന്നിട്ടുള്ളവയിൽ നിന്നും ഒറ്റപ്പെട്ടത് പുറത്തെടുക്കുക.
   a. HTML b. ALIGN c. HEAD d. CENTER

4.10.7 <PRE> - പൂർവ രൂപീകൃത ടെക്സ്റ്റുകൾ പ്രദർശിപ്പിക്കുന്നു (Displaying Preformatted text)

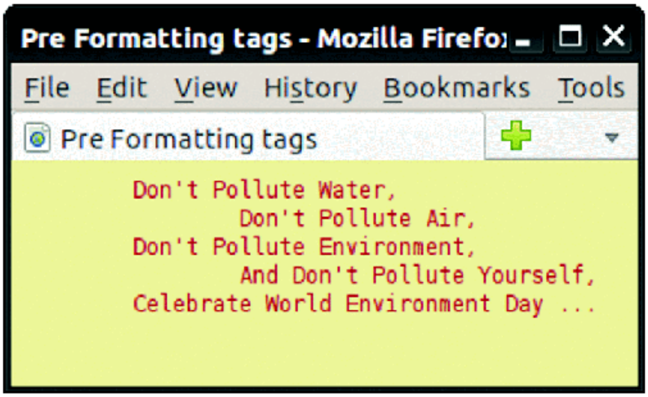
ടെക്സ്റ്റ് എഡിറ്ററിൽ നൽകിയ ഉള്ളടക്കം നമുക്ക് അതേ രൂപത്തിൽ തന്നെ പ്രദർശിപ്പിക്കണം എന്നിരിക്കട്ടെ. ഈ ആവശ്യം **<PRE>** ടാഗിന് നിർവഹിക്കാനാകും. സാധാരണഗതിയിൽ വൈറ്റ് സ്പേസുകൾ, ന്യൂ ലൈൻ ക്യാരക്ടറുകൾ, ടാബ് സ്പെയ്സുകൾ തുടങ്ങിയവയ്ക്ക് ബ്രൗസർ പരിധി നിശ്ചയിച്ചിട്ടുണ്ട്. അതുകൊണ്ട്, ബ്രൗസറിന്റെ സ്വയം പ്രേരിത ഘടനാരൂപീകരണ സ്വഭാവം <PRE> ടാഗുപയോഗിച്ച് നമുക്ക് പ്രവർത്തന രഹിതമാക്കാനാകും. ഈ ടാഗ്, അതിലുൾപ്പെടുത്തിയിട്ടുള്ള വാക്യത്തിന്റെ ഘടന രൂപപ്പെടുത്തിക്കഴിഞ്ഞതാണ് എന്ന് ബ്രൗസറിനോട് പറയുകയും, ആയതിനാൽ വീണ്ടും ഘടന പുനർനിർണയിക്കേണ്ടതില്ല എന്ന് ഓർമിപ്പിക്കുകയും ചെയ്യുന്നു. അതായത്, വാക്യം അതേപടി തന്നെ പ്രദർശിപ്പിക്കുവാൻ ബ്രൗസറിനോട് പറയുന്നു.

ഉദാഹരണം 4.10, ചിത്രം 4.26 എന്നിവ ഈ ടാഗിനെക്കുറിച്ച് ഒരു ധാരണ നമുക്ക് നൽകുന്നു. ലോക പരിസ്ഥിതിദിനത്തോടനുബന്ധിച്ച് കുറച്ച് വാക്യങ്ങളുള്ള ഒരു വെബ് പേജ് നമുക്ക് നിർമിക്കാം.

ഉദാഹരണം 4.10: <PRE> ടാഗ് വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> Pre Formatting tags </TITLE>
</HEAD>
<BODY Bgcolor = "#eee8aa" Text = "#b22222">
   <PRE>
     Don't Pollute Water,
            Don't Pollute Air,
     Don't Pollute Environment,
            And Don't Pollute Yourself,
     Celebrate World Environment Day ...
   </PRE>
</BODY>
</HTML>
```

<PRE>, </PRE> എന്നീ ടാഗുകൾക്ക് ഇടയിലുള്ളതെന്തും HTML ലിഖിതത്തിലുള്ളത് പോലെ തന്നെ പ്രദർശിപ്പിക്കും എന്ന് ചിത്രം 4.26-ൽ കാണിച്ചിട്ടുള്ള വെബ് പേജിൽ നിന്നും മനസിലാക്കാം.



ചിത്രം 4.26 : <PRE> ടാഗിനെക്കുറിച്ചുള്ള വിശദീകരണം

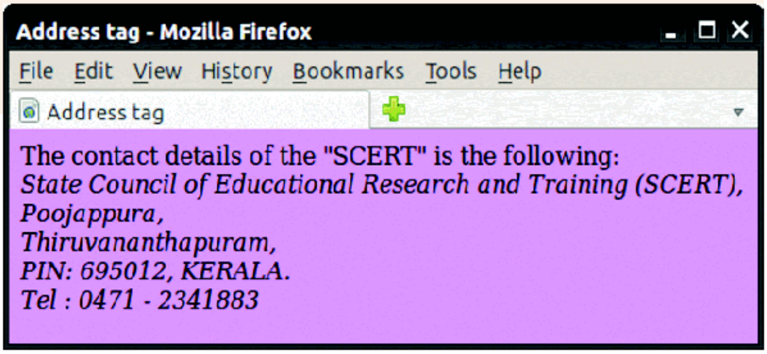
4.10.8 <ADDRESS> - മേൽവിലാസം പ്രദർശിപ്പിക്കുന്നു (Displaying the address)

ഒരു പ്രമാണത്തിന്റെ അല്ലെങ്കിൽ പ്രബന്ധത്തിന്റെ രചയിതാവിനെ/ഉടമസ്ഥനെ ബന്ധപ്പെടാനുള്ള വിവരം പോലുള്ളവ നിർവചിക്കുവാനുള്ള ടാഗാണ് **<ADDRESS>** ടാഗ്. പേര്, ടെലഫോൺ നമ്പർ, പിൻ നമ്പർ, ഇ-മെയിൽ മേൽവിലാസങ്ങൾ തുടങ്ങിയവയാണ് ഈ ടാഗിന്റെ ഉള്ളടക്കമായി ഉൾക്കൊള്ളിക്കുന്നത്. വാക്യങ്ങളെ ഇറ്റാലിക്സ് രീതിയിലാണ് മിക്കവാറും ബ്രൗസറുകൾ പ്രദർശിപ്പിക്കുന്നത്.

ഉദാഹരണം 4.11-ൽ തന്നിട്ടുള്ള കോഡ് <ADDRESS> ടാഗിനെക്കുറിച്ച് വിശദീകരിക്കുന്നു. പേജിന്റെ രൂപം ചിത്രം 4.27-ൽ കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.11: <ADDRESS> ടാഗ് വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> Address tag </TITLE>
</HEAD>
   <BODY Bgcolor= "#DDA0DD">
     The contact details of the "SCERT" is the following:
   <ADDRESS>
     State Council of Educational Research and Training
     (SCERT),<BR>
     Poojappura,<BR>
     Thiruvananthapuram,<BR>
     PIN: 695012, KERALA.<BR>
     Tel : 0471 - 2341883
   </ADDRESS>
</BODY>
</HTML>
```



ചിത്രം 4.27 : <ADDRESS> ടാഗ് ഉൾപ്പെടുത്തിയിട്ടുള്ള ഒരു വെബ് പേജ്

ബന്ധപ്പെടാനുള്ള വിവരത്തിന്റെ ഭാഗമെന്ന വരുമ്പോൾ, <ADDRESS> ടാഗ് സാധാരണ ഗതിയിൽ തപാൽ മേൽവിലാസം സൂചിപ്പിക്കാനാണ് ഉപയോഗിക്കുന്നത്. എന്നിരുന്നാലും, തനത് ശൈലിയിൽ <EM>, <I> എന്നിവ പ്രദർശിപ്പിക്കുന്ന ശൈലിക്ക് സമാനമായാണ് <ADDRESS> ടാഗും വാക്യം പ്രദർശിപ്പിക്കുന്നത്. ആയതിനാൽ, ബന്ധപ്പെടാനുള്ള വിവരം കൈകാര്യം ചെയ്യാൻ അനുയോജ്യമായ രീതിയിൽ അത് ഉപയോഗിക്കാവുന്നതാണ്. എഴുത്തുകാരന്റെ പേര്, പകർപ്പവകാശം തുടങ്ങിയ വിവരങ്ങൾ ഉൾക്കൊള്ളുന്ന അടിക്കുറിപ്പിനുള്ളിലാണ് <ADDRESS> ടാഗ് എലമെന്റ് മാതൃകാപരമായി നൽകാറുള്ളത്.

4.10.9 <MARQUEE> - മാർക്ക്യൂ വാക്യം പ്രദർശിപ്പിക്കുന്നു (Displaying text in a scrolling Marquee)

നാം ഇതുവരെ ചർച്ച ചെയ്ത HTML ടാഗുകളെല്ലാം ഉള്ളടക്കങ്ങൾ ബ്രൗസറിൽ പ്രദർശിപ്പിക്കുക മാത്രമാണ് ചെയ്യുന്നത്. എന്നാൽ ഒരു വാക്യശകലം അല്ലെങ്കിൽ ചിത്രം വെബ് പേജിൽ തിരശ്ചീനമായോ ലംബമായോ ഒഴുകി നീങ്ങുന്നതായോ കാണിക്കുന്നതിന്, **<MARQUEE>** ടാഗ് ഉപയോഗിക്കുന്നു.

<MARQUEE> ടാഗിനൊപ്പം ഉപയോഗിക്കുന്ന പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടുകളുടെ നാമാവലി താഴെ തന്നിട്ടുണ്ട്.

- **Height**: പിക്സലായോ ബ്രൗസർ ജാലകത്തിലെ ഉയരത്തിന്റെ ശതമാനമായോ മാർക്യൂവിന്റെ ഉയരം നിശ്ചയിക്കുന്നു.
- **Width**: പിക്സലായോ, ബ്രൗസർ ജാലകത്തിലെ വീതിയുടെ ശതമാനമായോ വീതി സൂചിപ്പിക്കുന്നു.
- **Direction**: ഏത് ദിശയിലാണ് ടെക്സ്റ്റ് ചലിക്കേണ്ടതെന്ന് സൂചിപ്പിക്കുന്നു. `up`, `down`, `left`, `right` എന്നീ വിലകളാണ് ഇതിനുള്ളത്.
- **Behaviour**: മാർക്യൂവിന്റെ സ്ക്രോളിങ് ഏത് വിധത്തിലാകണമെന്ന് സൂചിപ്പിക്കുന്നു. `scroll`, `slide`, `alternate` എന്നീ വിലകളാണ് ഇതിനുള്ളത്.
- **Scrolldelay**: ഓരോ സ്ക്രോളിനും ഇടയ്ക്കുള്ള സമയദൈർഘ്യം സൂചിപ്പിക്കുന്നു. 10, 15 തുടങ്ങിയവ പോലെ സെക്കന്റിലുള്ള വിലകളാണ് ഇതിനുള്ളത്.
- **Scrollamount**: മാർക്യൂ ടെക്സ്റ്റിന്റെ വേഗത സൂചിപ്പിക്കുന്നതിന് ഉപയോഗിക്കുന്നു.
- **Loop**: മാർക്യൂ ടെക്സ്റ്റ് എത്രപ്രാവശ്യം സ്ക്രീനിൽ സ്ക്രോൾ ചെയ്യണമെന്ന് സൂചിപ്പിക്കുന്നു. ഇതിന്റെ തനത് വില `Infinte` ആയിരിക്കും, അങ്ങനെ ആയാൽ മാർക്യൂ അവസാനമില്ലാതെ സ്ക്രോൾ ആകും.
- **Bgcolor**: നിറമോ, നിറത്തെ സൂചിപ്പിക്കുന്ന ഹെക്സാഡെസിമൽ സംഖ്യയോ ഉപയോഗിച്ച് പശ്ചാത്തലനിറം സൂചിപ്പിക്കുന്നു.
- **Hspace**: ഇത് മാർക്യൂവിന് ചുറ്റുമുള്ള തിരശ്ചീനമായ സ്ഥലം സൂചിപ്പിക്കുന്നു. ഇതിന്റെ വില പിക്സൽ ആയോ ശതമാനവില ആയോ നൽകുന്നു.
- **Vspace**: മാർക്യൂവിനു ചുറ്റുമുള്ള ലംബമായ ശൂന്യസ്ഥലത്തെ സൂചിപ്പിക്കുന്നു. ഇതിന്റെ വിലയും പിക്സൽ ആയോ ശതമാനവില ആയോ ആണ് നൽകുന്നത്.

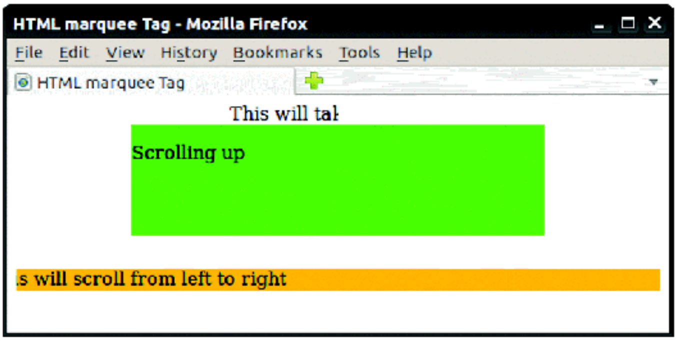
ഉദാഹരണം 4.12-ൽ തന്നിട്ടുള്ള കോഡ് `<MARQUEE>` ടാഗിന്റെ ഉപയോഗത്തേയും ചിത്രം 4.28 അനുബന്ധ വെബ് പേജിനേയും കാണിക്കുന്നു.

ഉദാഹരണം 4.12: <MARQUEE> ടാഗ് വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> HTML marquee Tag </TITLE>
</HEAD>
<BODY>
   <MARQUEE Width= "50%"> This  will take only 50% width of
     Browser Window</MARQUEE>
   <MARQUEE Height= "100" Hspace= "100" Bgcolor= "#44BB22"
          Direction= "up"> Scrolling up </MARQUEE>
   <MARQUEE Height= "20" Vspace= "30" Bgcolor= "#FFBB00"
     Direction= "right"> This will scroll from left to right
   </MARQUEE>
</BODY>
</HTML>
```

വീതി 50% ആയതിനാൽ ആദ്യത്തെ മാർക്യൂ മധ്യഭാഗത്ത് നിന്നും തുടങ്ങി വിൻഡോയുടെ ഇടത് വശത്തേക്ക് സ്ക്രോൾ ചെയ്യുന്നു. രണ്ടാമത്തെ മാർക്യൂ പച്ചനിറത്തോടുകൂടിയതും 100 പിക്സൽ ഉയരം സ്ക്രോൾ ചെയ്യാൻ ഉപയോഗിച്ചിട്ടുള്ളതുമാണ്.

ഇടത് മാർജിനിൽ നിന്നും തിരശ്ചീനമായി 100 പിക്സൽ അകലെയാണ് സ്ക്രോൾ മേഖല ഏരിയ സജ്ജീകരിച്ചിട്ടുള്ളത്. മൂന്നാമത്തെ മാർക്യൂ "#FFBB00" നിറം കൊണ്ട് പശ്ചാത്തലം ഒരുക്കിയിട്ടുള്ളതും മുന്നിലുള്ള മാർക്യൂവിൽ നിന്നും ലംബമായി 30 പിക്സൽ താഴേക്ക് സ്ഥാപിച്ചിട്ടുള്ളതുമാണ്. സ്ക്രോൾ ചെയ്യുന്ന ജാലകം 40 പിക്സൽ ഉയരത്തിലുള്ളതും ഇടത്ത് നിന്ന് വലത്തേക്ക് നീങ്ങുന്നതുമാണ്. വാക്കുകളോ, വാക്യങ്ങളോ മാത്രമല്ല ചിത്രങ്ങളും മാർക്യൂവിന്റെ ഉള്ളടക്കമായി നമുക്ക് ഉപയോഗിക്കാം.



ചിത്രം 4.28 : Marquee ടാഗും അതിന്റെ ആട്രിബ്യൂട്ടുകളും

4.10.10 <DIV> - വാക്യങ്ങളുടെ ഒരു ഭാഗം ചിട്ടപ്പെടുത്തുന്നു (Formatting a block of text)

ഡോക്യുമെന്റിന്റെ ഒരു ഭാഗം അല്ലെങ്കിൽ ഒരു നിര ചിട്ടപ്പെടുത്താനാണ് <DIV> ടാഗ് ഉപയോഗിക്കുന്നത്. HTML ലിഖിതത്തിലുള്ള വലിയ ഭാഗങ്ങൾ ഒരുമിച്ച് ഒരു കൂട്ടമാക്കി ചിട്ടപ്പെടുത്താനും **<DIV>** ടാഗ് കൊണ്ട് സാധിക്കുന്നു. ഖണ്ഡികകൾ, പട്ടികകൾ തുടങ്ങിയവ ഈ ഭാഗത്തുണ്ടാകും. <DIV> ടാഗിന് മുൻപും പിൻപും മിക്കവാറും ബ്രൗസറുകൾ ഒരു വരി (ശൂന്യവരി) ചേർത്തിട്ടുണ്ടാകും. <DIV> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകൾ താഴെ തന്നിട്ടുള്ളവയാണ്.

Align : left, right, center, justify എന്നീ വിലകൾ കൊണ്ട് തിരശ്ചീനമായ വിന്യാസം നിജപ്പെടുത്തുന്നു.

Id : ടാഗിനെ തിരിച്ചറിയാനായി സവിശേഷമായ ഒരു സൂചകം നൽകുന്നു.

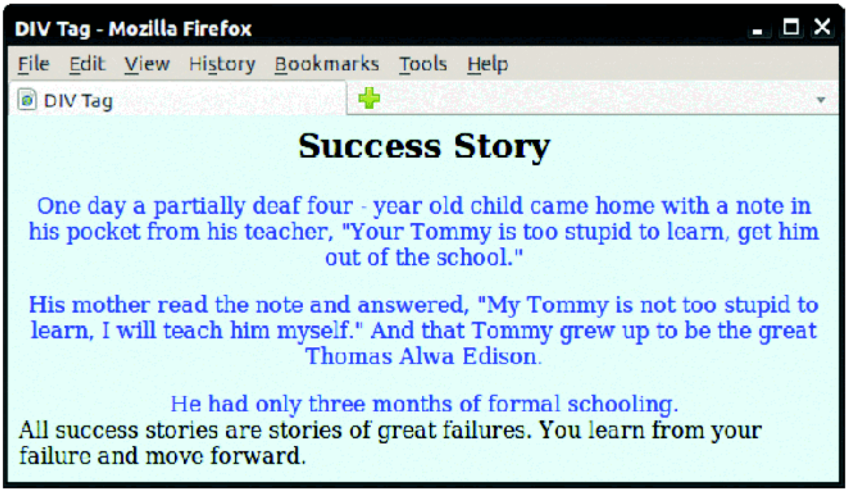
Style : ഉള്ളടക്കത്തിലെ അക്ഷരം, നിറം തുടങ്ങിയവ എങ്ങനെ ആവിഷ്കരിക്കണമെന്ന് നിർദേശിക്കുന്നു.

<DIV> ടാഗിന്റെ Align, Style എന്നീ ആട്രിബ്യൂട്ടുകളുടെ ഉപയോഗം ഉദാഹരണം 4.13-ലും, തൽഫലമായി ലഭിക്കുന്ന വെബ്പേജ് ചിത്രം 4.29-ലും കാണാവുന്നതാണ്.

ഉദാഹരണം 4.13: <DIV> ടാഗ് വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> DIV Tag </TITLE>
</HEAD >
```

```
<BODY Bgcolor= "#ddffff">
   <H2 Align= "center"> Success Story </H2>
   <DIV Align= "Center" Style= "Color:#0000FF"> One day a
   partially deaf four - year old child came home with a
   note in his pocket from his teacher, "Your Tommy is too
   stupid to learn, get him out of the school."
   <P>His mother read the note and answered, "My Tommy is
      not too stupid to learn, I will teach him myself." And
      that Tommy grew up to be the great Thomas Alwa Edison.
   </P> He had only three months of formal schooling.
   </DIV>
   All success stories are stories of great failures.
   You learn from your failure and move forward.
</BODY>
</HTML>
```



ചിത്രം 4.29 : <DIV> ടാഗിന്റെ ഉപയോഗം

4.10.11 <FONT> - ലിപികളുടെ സ്വഭാവസവിശേഷതകൾ നിർദേശിക്കുന്നു (Specifying font characteristics)

<FONT>, </FONT> എന്നീ ടാഗുകൾക്ക് ഇടയിൽ ഉൾക്കൊള്ളിച്ചിട്ടുള്ള വാക്യങ്ങളുടെ ശൈലി, വലുപ്പം, നിറം എന്നിവ വ്യത്യാസപ്പെടുത്തുവാൻ **<FONT>** ടാഗ് നമ്മെ സഹായിക്കുന്നു. ഡോക്യുമെന്റിലെ ഒരു ചെറിയ ഭാഗത്തിന്റെ രൂപഘടന മാറ്റുന്നതിനാണ് ഇത് സാധാരണയായി ഉപയോഗിക്കുന്നത്. <FONT> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകൾ താഴെ തന്നിട്ടുള്ളവയാണ്.

Color : നിറനാമമായോ, ഹെക്സാഡെസിമൽ വിലയായോ ടെക്സ്റ്റിന് നിറം നൽകുന്നു.

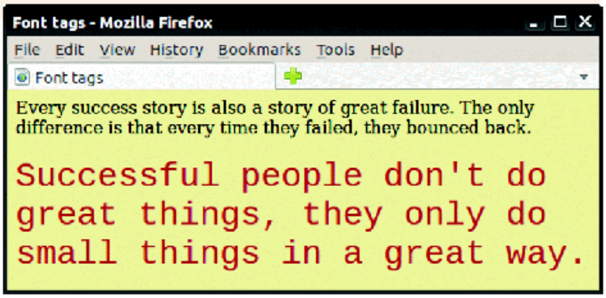
Face : അക്ഷരങ്ങളുടെ ശൈലി സൂചിപ്പിക്കുന്ന ആട്രിബ്യൂട്ടാണ് ഇത്. വില സൂചിപ്പിച്ചിട്ടില്ല എങ്കിൽ ബ്രൗസർ അനുകൂലിക്കുന്ന ശൈലികളിൽ ആദ്യത്തേത് തനത് (Default) ശൈലിയായി ഡോക്യുമെന്റിലെ വാക്യങ്ങൾക്ക് നൽകുന്നു.

Size : അക്ഷരങ്ങളുടെ വലുപ്പം 1 മുതൽ 7 വരെയുള്ള വിലകൾ കൊണ്ട് സൂചിപ്പിക്കാൻ ഉപയോഗിക്കുന്ന ആട്രിബ്യൂട്ടാണ് ഇത്. തനത് വില 3 ആയിരിക്കും.

<FONT> ടാഗിന്റെ ഉപയോഗം ഉദാഹരണം 4.14-ൽ തന്നിട്ടുള്ള കോഡ് വിശദീകരിക്കുകയും അനുബന്ധ വെബ് പേജ് ചിത്രം 4.30 കാണിക്കുകയും ചെയ്യുന്നു.

ഉദാഹരണം 4.14: <FONT> ടാഗ് വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD> <TITLE> Font tags </TITLE> </HEAD>
<BODY Bgcolor= "#eee8aa">
   Every success story is also a story of great failure.
   The only difference is that every time they failed,
   they bounced back. <BR><BR>
   <FONT Size="6" Face="Courier New" Color="#B22222">
   Successful people don&apos;t do great things,
   they only do small things in a great way.
   </FONT>
</BODY>
</HTML>
```



ചിത്രം 4.30 : <FONT> ടാഗിന്റെ ഉപയോഗം

മുകളിൽ തന്നിട്ടുള്ള കോഡിൽ ഒരു ഭാഗം (') നാം പച്ചനിറത്തിൽ രേഖപ്പെടുത്തിയിട്ടുണ്ട്. ഇത് അർഥമാക്കുന്നത് എന്താണ്? **don't** എന്ന വാക്കിലെ ഏക ഉദ്ധരണി ചിഹ്നം കൊണ്ട് വരുന്നതിന് വേണ്ടിയാണിതെന്ന് ലഭിക്കുന്ന വെബ് പേജിൽ നിന്നും നമുക്ക് മനസിലാക്കാനാകും. ഇതുപോലെ മറ്റ് പല ചിഹ്നങ്ങളുമുണ്ട്. തുടർന്ന് വരുന്ന ഭാഗങ്ങളിൽ അവ ചർച്ച ചെയ്യാം.

4.11 സാർഥക ചിഹ്നങ്ങൾക്ക് വേണ്ടി HTML-ൽ ഉപയോഗിക്കുന്ന സവിശേഷ വസ്തുക്കൾ (HTML entities for reserved characters)

<, >, & തുടങ്ങിയ ചിഹ്നങ്ങൾക്ക് പ്രത്യേകമായ അർത്ഥമാണ് HTML-ൽ ഉള്ളത്. മാത്രമല്ല, HTML ഡോക്യുമെന്റിന്റെ ഉള്ളടക്കത്തിലെ വാക്യങ്ങളിൽ ഇവ ഉൾപ്പെടുത്താനും സാധിക്കില്ല. HTML ടാഗുകളോടും ആട്രിബ്യൂട്ടുകളോടും ഒപ്പം കാണപ്പെടുന്ന വിരാമ ചിഹ്നങ്ങളെപ്പോലെ ഈ ചിഹ്നങ്ങളെ ബ്രൗസർ കണക്കാക്കും.

ഉദാഹരണമായി, ആങ്കിൾ ബ്രാക്കറ്റുകളായ <, > എന്നിവ ഉപയോഗിക്കുന്നത് ടാഗുകളെ സൂചിപ്പിക്കാനാണ്. വെബ് പേജിലെ വാക്യങ്ങളിൽ ഈ അടയാളങ്ങൾ നമുക്ക് എപ്പോഴെങ്കിലും പ്രദർശിപ്പിക്കേണ്ടി വന്നാൽ, HTML-ലെ സവിശേഷ വസ്തുക്കൾ നമുക്ക് ഉപയോഗിക്കേണ്ടിവരും. പട്ടിക 4.7-ൽ സാർത്ഥക ചിഹ്നങ്ങളുടെയും അനുബന്ധ HTML വസ്തുക്കളുടെയും നാമാവലി നൽകിയിരിക്കുന്നു.

ഒരു വെബ് പേജിൽ A < B & A > C എന്ന വാക്യം പ്രദർശിപ്പിക്കുന്നതിന്, **`A < B & A > C`** എന്ന് HTML കോഡിൽ നൽകണം.

Character	Entity	Description
		ഖണ്ഡിക്കാനാവാത്ത സ്ഥലം
"	"	ഇരട്ട ഉദ്ധരണി
'	'	ഏക ഉദ്ധരണി
&	&	ആംപർസെന്റ്
<	<	ചെറുത്
>	>	വലുത്
©	©	പകർപ്പവകാശ അടയാളം
TM	™	വ്യാപാര മുദ്ര അടയാളം
®	®	രജിസ്റ്റർ ചെയ്തതിന്റെ അടയാളം

പട്ടിക 4.7 : എന്റിറ്റികളുടെ പട്ടികയും അവയുടെ വിശദീകരണവും

4.12 HTML ഡോക്യുമെന്റിൽ കമന്റുകൾ ഉൾപ്പെടുത്തുന്നു (Adding comments in HTML document)

HTML ഡോക്യുമെന്റിൽ, പ്രത്യേകിച്ച് സങ്കീർണമായ ഡോക്യുമെന്റിൽ കമന്റ് ഉൾപ്പെടുത്തുന്നത് ഒരു നല്ല ശീലമാണ്. കോഡ് മനസ്സിലാക്കാൻ കമന്റുകൾ നമ്മെ സഹായിക്കുന്നു. കൂടാതെ കോഡിന്റെ വായനക്ഷമതയും അത് വർധിപ്പിക്കുന്നു. സോഴ്സ് കോഡിൽ വിവരണങ്ങൾ ഉൾപ്പെടുത്താൻ HTML-ൽ കമന്റ് ടാഗ്കളുണ്ട്. വിവരണങ്ങൾ ബ്രൗസർ ജാലകത്തിൽ കാണിക്കുകയില്ല. HTML വിവരണങ്ങൾ <!-- , --> എന്നീ ടാഗുകൾക്ക് ഇടയ്ക്കാണ് നൽകാറുള്ളത്. അപ്രകാരം നൽകുന്ന ഏത് ഉള്ളടക്കവും കമന്റായി കണക്കാക്കുകയും ബ്രൗസർ അത് പൂർണമായും അവഗണിക്കുകയും ചെയ്യുന്നു. ജിനി എഡിറ്ററിൽ കമന്റുകൾ ചുവപ്പ് നിറത്തിലാണ് പ്രദർശിപ്പിക്കാറുള്ളത്. ഉദാഹരണം 4.15-ലെ കോഡ് കമന്റുകളുടെ ഉപയോഗത്തെക്കുറിച്ച് വിശദീകരിക്കുകയും ചിത്രം 4.31-ലെ വെബ് പേജ് അങ്ങനെ ലഭിക്കുന്ന സന്ദേശം കാണിക്കുകയും ചെയ്യുന്നു.

ഉദാഹരണം 4.15: കമന്റ് ടാഗിന്റെ ഉപയോഗം വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD> <!-- Document Header Starts -->
   <TITLE> Comment Tags </TITLE>
</HEAD>
<BODY Bgcolor= "#D8D8D8">
   <!-- This is a comment -->
```

```
    <p>The paragraph starts here. Comment statements are
    not displayed in the browser window</p>
    <!-- Comments are not displayed in the browser -->
</BODY>
</HTML>
```

ചിത്രം 4.31 : HTML കമന്റുകൾ

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. സാർത്ഥക ചിഹ്നങ്ങൾ എങ്ങനെയാണ് HTML-ൽ സൂചിപ്പിക്കുന്നത്?
2. ________ ടാഗിനോട് ഒപ്പമാണ് `Face` ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്.
3. <FONT> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകളുടെ നാമാവലി തയാറാക്കുക.
4. <PRE> ടാഗിന്റെ ഉപയോഗമെന്താണ്?
5. ഒരു ടെക്സ്റ്റ് സ്ക്രോൾ ചെയ്യാനായി ________ ടാഗ് ഉപയോഗിക്കുന്നു.
6. <MARQUEE> ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടുകൾ ഏതെല്ലാമാണ്?
7. <ADDRESS> ടാഗിന്റെ ഉപയോഗമെന്താണ്?
8. <FONT> ടാഗിലെ അക്ഷരങ്ങളുടെ തനത് വലുപ്പം എത്രയായിരിക്കും?
9. <DIV>ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടുകളുടെ പേരെഴുതുക.

4.13 ചിത്രങ്ങൾ ഉൾക്കൊള്ളിക്കുന്നു (Inserting images)

ഉള്ളടക്ക വിവരണം കൂടുതൽ ആകർഷകവും വിനിമയാത്മകവും ആക്കുന്നത് ചിത്രങ്ങളാണ്. ഇക്കാലത്ത് വെബ്സൈറ്റുകൾ അധികവും ചിത്രങ്ങളാൽ സമ്പന്നമാണ്. വെബ് വികസിപ്പിക്കുന്നതിന് വേണ്ട ധാരാളം സവിശേഷതകളുള്ള പുതിയ HTML പതിപ്പുകളിൽ പോലും ചിത്രങ്ങൾ ഉൾപ്പെടുത്തുന്നതിനുള്ള കോഡ് വളരെ ലളിതമാണ്. HTML പേജുകളിൽ ചിത്രങ്ങൾ ഉൾക്കൊള്ളിക്കുന്നതിന് <IMG> ടാഗ് ഉപയോഗിക്കുന്നു. വളരെ ലളിതമായി ഈ ടാഗ് ഉപയോഗിക്കുന്ന വിധം താഴെ കൊടുത്തിരിക്കുന്നു.

```
<IMG Src = "picture1.jpg">
```

<IMG> ടാഗ് ഒരു എംപ്റ്റി ടാഗാണ്. മാത്രമല്ല, അതിന് ധാരാളം ആട്രിബ്യൂട്ടുകളും ഉണ്ട്. പേജിൽ ഉൾക്കൊള്ളിക്കേണ്ട ചിത്രത്തിന്റെ ഫയൽനാമം സൂചിപ്പിക്കുന്ന Src ആട്രിബ്യൂട്ടാണ് ഏറ്റവും പ്രധാനപ്പെട്ടത്. നമ്മുടെ ആവശ്യത്തിന് അനുസരിച്ച് JPEG, PNG, GIF എന്നീ ചിത്രഫയലുകൾ ഉപയോഗിക്കാനാകുമെങ്കിലും, ശരിയായ ഫയൽ നാമവും അതോടൊപ്പമുള്ള എക്സ്റ്റൻഷനും Src ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് സൂചിപ്പിച്ചിട്ടുണ്ടെന്ന് ഉറപ്പ് വരുത്തണം. ചിത്രഫയൽ ഇപ്പോൾ ഉപയോഗത്തിലുള്ള ഫോൾഡറിൽ ഇല്ലെങ്കിൽ, അത് എവിടെയാണോ ലഭ്യമാകുന്നത് അവിടേക്കുള്ള വഴി (പാത്ത്)/URL നാം സൂചിപ്പിക്കണം.

ചിത്രത്തിന് വേണ്ടി സ്ഥലം ഒരുക്കുന്നു (Setting space for the image)

Width, **Height** എന്നീ ആട്രിബ്യൂട്ടുകൾക്ക് വിലകൾ നൽകിക്കൊണ്ട് വെബ് പേജിൽ ചിത്രത്തിനുവേണ്ടിയുള്ള ഇടം നമുക്ക് ഒരുക്കാം. പിക്സലിന്റെ എണ്ണമായോ അല്ലെങ്കിൽ യഥാർഥ വലുപ്പത്തിന്റെ ശതമാനമായോ വിലകൾ നൽകാവുന്നതാണ്. ഈ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിച്ചിട്ടില്ലെങ്കിൽ ചിത്രത്തിന്റെ യഥാർഥ വലുപ്പത്തിൽ ബ്രൗസർ അത് പ്രദർശിപ്പിക്കും.

ചിത്രങ്ങൾക്ക് ഇടയ്ക്കുള്ള ശൂന്യസ്ഥലങ്ങൾ എങ്ങനെ ക്രമപ്പെടുത്താമെന്ന് നമുക്ക് ചർച്ച ചെയ്യാം. ഒരു ജാലകത്തിൽ ചിത്രങ്ങൾ സ്ഥാപിക്കുമ്പോൾ രണ്ട് രീതിയിലാണ് ശൂന്യസ്ഥലങ്ങൾ അവയ്ക്കിടയിൽ വരാറുള്ളത്-തിരശ്ചീനമായും, ലംബമായും. വെബ് പേജിൽ തിരശ്ചീനമായും ലംബമായും ചിത്രങ്ങൾക്കിടയ്ക്ക് സ്ഥലം നൽകുന്നതിന് HTML-ൽ ലഭ്യമാകുന്ന രണ്ട് ആട്രിബ്യൂട്ടുകളാണ് **Hspace**, **Vspace** എന്നിവ.

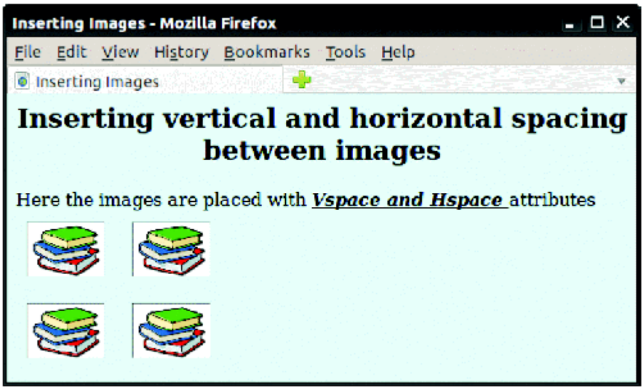
രണ്ട് രീതിയിൽ ചിത്രങ്ങൾക്ക് ഇടയ്ക്ക് ശൂന്യസ്ഥലം നൽകുന്നത് ഉദാഹരണം 4.16-ൽ തന്നിട്ടുള്ള HTML കോഡ് വിശദീകരിക്കുന്നു. അനുബന്ധ വെബ് പേജ് ചിത്രം 4.31-ൽ കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.16: ചിത്രങ്ങൾക്കിടയിൽ പലതരത്തിലുള്ള ശൂന്യസ്ഥലം നൽകുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> Inserting Images </TITLE>
</HEAD>
<BODY Bgcolor="#E0FFFF">
   <H2 Align="center">Inserting vertical and horizontal
                      spacing between images</H2>
   Here the images are placed with <B><I><U> Vspace and
   Hspace </U></I></B> attributes <BR>
   <IMG Src= "book3.jpg" Height= "50" Width= "70"
```

```
        Vspace= "10" Hspace= "10">
    <IMG Src= "book3.jpg" Height= "50" Width= "70"
        Vspace= "10" Hspace= "10"> <BR>
    <IMG Src= "book3.jpg" Height= "50" Width= "70"
        Vspace= "10" Hspace= "10">
    <IMG Src= "book3.jpg" Height= "50" Width= "70"
        Vspace= "10" Hspace= "10"> <BR>
</BODY>
</HTML>
```

നിർദിഷ്ട വീതിയിലും ഉയരത്തിലുമാണ് ചിത്രങ്ങൾ വിന്യസിച്ചിട്ടുള്ളതെന്നും തിരശ്ചീനമായും ലംബമായും നൽകിയ ശൂന്യസ്ഥലം സൂചിപ്പിച്ചിട്ടുള്ളത് പോലെയാണ് ചിത്രങ്ങൾക്കിടയിലെ അകലമെന്നും ചിത്രം 4.32-ൽ കാണാൻ സാധിക്കും.



ചിത്രം 4.32 : നിർദേശിച്ച വലുപ്പങ്ങളിലുള്ളതും, ഇടയ്ക്ക് തിരശ്ചീനവും ലംബവുമായി ശൂന്യസ്ഥലങ്ങൾ ഉൾക്കൊള്ളിച്ചിട്ടുള്ളതുമായ ചിത്രങ്ങൾ

വാക്യങ്ങൾ ആരംഭിക്കുന്ന വരി അടിസ്ഥാനമാക്കി ചിത്രങ്ങൾ വിന്യസിക്കുന്നതിന് ഉപയോഗിക്കുന്ന <IMG> ടാഗിന്റെ Align ആട്രിബ്യൂട്ടിനെക്കുറിച്ച് നമുക്ക് ചർച്ച ചെയ്യാം. ഈ ആട്രിബ്യൂട്ടിന് നൽകാനാകുന്ന വിലകൾ താഴെകൊടുത്തിരിക്കുന്നു.

Bottom : ചിത്രത്തിന്റെ കീഴ്ഭാഗം വാക്യങ്ങൾ ആരംഭിക്കുന്ന വരിയുമായി വിന്യസിക്കുന്നു.

Middle : ചിത്രത്തിന്റെ മധ്യഭാഗം (ലംബമായി) വാക്യങ്ങൾ ആരംഭിക്കുന്ന വരിയുമായി വിന്യസിക്കുന്നു.

Top : വാക്യങ്ങളുടെ മുകൾഭാഗവുമായി ചിത്രം വിന്യസിക്കുന്നു.

Align ആട്രിബ്യൂട്ടിന് വിലകൾ നൽകുമ്പോൾ ഉണ്ടാകുന്ന പ്രതീതി നമുക്ക് കാണാം. ഉദാഹരണം 4.17, ചിത്രം 4.33 എന്നിവ ഇത് വിശദീകരിക്കുന്നു.

ഉദാഹരണം 4.17: ഒരു ചിത്രത്തിന് വ്യത്യസ്ത വിന്യാസങ്ങൾ നൽകുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
    <TITLE> Alignment of Images </TITLE>
</HEAD>
```

```
<BODY Bgcolor= "#E0FFFF">
   <H2 Align= "center">Alignment of Images</H2>
   This Image is <I><U>aligned at the bottom </U></I>
   <IMG Src= "book3.jpg" Height= "40" Width= "50"
        Align= "Bottom"> <BR> <BR>
       This Image is <I><U>aligned at the Middle </U></I>
   <IMG Src= "book3.jpg" Height= "40" Width= "50"
        Align= "Middle"> <BR> <BR>
        This Image is <I><U>aligned at the Top </U></I>
   <IMG Src= "book3.jpg" Height= "40" Width= "50" Align="Top">
</BODY>
</HTML>
```

<IMG> ടാഗിന്റെ Align ആട്രിബ്യൂട്ടിന് മറ്റ് ചില വിലകൾ കൂടിയുണ്ട്. അവ ചിത്രത്തെ ബ്രൗസർ ജാലകത്തിന്റെ ഇടത് ഭാഗത്തേക്കും വലതുഭാഗത്തേക്കും യഥാക്രമം വിന്യസിക്കുന്ന left, right - എന്നിവയാണ്.

ചിത്രം 4.33 : വാക്യവുമായുള്ള ചിത്രത്തിന്റെ വിവിധ പൊരുത്തപ്പെടലുകൾ

ചിത്രം 4.34 : ബ്രൗസർ വിൻഡോയ്ക്ക് ഇടതും വലതുമായി വിന്യസിക്കപ്പെട്ട ചിത്രങ്ങൾ

നമുക്കു ചെയ്യാം

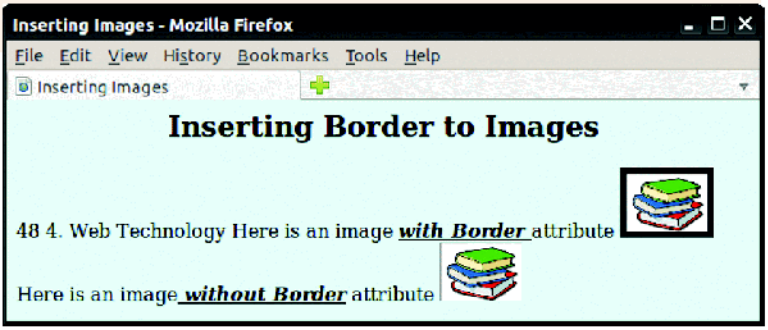
ബ്രൗസർ ജാലകത്തിൽ ചിത്രത്തിന്റെ ഇടത്തോട്ടും വലത്തോട്ടുമുള്ള വിന്യാസം ചിത്രം 4.34 -ൽ കാണിച്ചിട്ടുള്ളത് നിരീക്ഷിക്കുക. ഈ പേജിന് വേണ്ട HTML ലിഖിതം തയാറാക്കുക.

ഒരു ചിത്രത്തിന് ബോർഡർ നൽകുന്നു (Setting border around an image)

ഒരു വെബ് പേജിൽ ഉൾപ്പെടുത്തിയിട്ടുള്ള ചിത്രത്തിന് നമുക്ക് ബോർഡർ നൽകണമെന്ന് വിചാരിക്കുക. <IMG> ടാഗിന്റെ Border ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് ഇത് സാധ്യമാക്കാം. ഈ ആട്രിബ്യൂട്ടിന് അനുയോജ്യമായ വില നൽകികൊണ്ട് ബോർഡറിന് കനം നൽകാം. ഉദാഹരണം 4.18-ലെ HTML കോഡും ചിത്രം 4.35-ലെ അനുബന്ധ വെബ് പേജും Border ആട്രിബ്യൂട്ടിന്റെ ഉപയോഗം നമുക്ക് കാണിച്ചു തരുന്നു.

ഉദാഹരണം 4.18: ചിത്രത്തിന് ബോർഡർ നൽകുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD>
   <TITLE> Inserting Images </TITLE>
</HEAD>
<BODY Bgcolor= "#E0FFFF">
   <H2 Align= "center">Inserting Border to Images</H2>
   Here is an image <B><I><U> with Border </U></I></B>
   attribute
   <IMG Src= "book3.jpg" Height= "50" Width= "70" Border="5">
   <BR>Here is an image<B><I><U> without Border</U></I></B>
   attribute
   <IMG Src= "book3.jpg" Height= "50" Width= "70">
</BODY>
</HTML>
```



ചിത്രം 4.35 : ബോർഡറുകൾ ഉള്ളതും ഇല്ലാത്തതുമായ ചിത്രങ്ങൾ

<IMG> ടാഗിന്റെ വിവിധ ആട്രിബ്യൂട്ടുകളെക്കുറിച്ചും അവ ചിത്രത്തിലുണ്ടാക്കുന്ന വ്യതിയാനങ്ങളെക്കുറിച്ചും നാം പഠിച്ച് കഴിഞ്ഞു. Src ആട്രിബ്യൂട്ടിൽ നിർദ്ദേശിച്ച ചിത്രഫയൽ കണ്ടെത്താനായില്ലെങ്കിൽ വെബ് പേജ് കാഴ്ചയിൽ എങ്ങനെയുണ്ടാകും? വെബ് പേജിൽ ചിത്രത്തിനായി നിർദ്ദേശിച്ച സ്ഥലം ശൂന്യമായിരിക്കും. മറ്റ് ചില കാരണങ്ങളാലും ബ്രൗസറിന് ചിത്രം പ്രദർശിപ്പിക്കാനാകാതെ വരാറുണ്ട്. അത്തരം സന്ദർഭങ്ങളിൽ, ചിത്രത്തിനു പകരം ഒരു വാക്യം ബ്രൗസറിൽ നമുക്ക് ദൃശ്യമാക്കാനാകും. ബ്രൗസറിന് ചിത്രം പ്രദർശിപ്പിക്കാനാകാതെ വന്നാൽ പകരം ഒരു വാക്യം സൂചിപ്പിക്കാൻ HTML നൽകുന്ന ആട്രിബ്യൂട്ടാണ് **Alt**. ഉദാഹരണം 4.19-ലെ കോഡ് Alt ആട്രിബ്യൂട്ടിന്റെ ഉപയോഗം വിശദീകരിക്കുന്നു. അങ്ങനെ ലഭിക്കുന്ന വെബ് പേജ് ചിത്രം 4.36-ൽ കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 4.19: <IMG> ടാഗിലെ Alt ആട്രിബ്യൂട്ടിന്റെ ഉപയോഗം വിശദീകരിക്കുന്നതിന് വേണ്ടിയുള്ളത്

```
<HTML>
<HEAD> <TITLE> Inserting Images </TITLE> </HEAD >
<BODY Bgcolor= "#E0FFFF">
   <H2 Align= "center">Inserting Images</H2>
   If the browser cannot display the image, then the text
   entered in the <B>Alt</B> attribute will be displayed.<BR>
```

```
    <IMG Src= "book5.jpg" Height= "20%" Width= "20%"
         Alt= "Image of an opened book"> <BR>
</BODY>
</HTML>
```

ചിത്രം 4.36 : <IMG> ടാഗിലെ Alt ആട്രിബ്യൂട്ടിന്റെ ഉപയോഗം

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. ഒരു HTML ലിഖിതത്തിൽ ചിത്രങ്ങൾ ഉൾക്കൊള്ളിക്കുന്നതിനായി ________ ടാഗ് ഉപയോഗിക്കുന്നു.
2. <IMG> ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടാണ് ________.
3. <IMG> ടാഗിൽ Alt ആട്രിബ്യൂട്ടിന്റെ ഉപയോഗമെന്ത്?
4. ഒരു ചിത്രം ഒരു പ്രത്യേക വലുപ്പത്തിൽ പ്രദർശിപ്പിക്കുന്നതിന് ഉപയോഗിക്കുന്ന ആട്രിബ്യൂട്ടുകളുടെ പേരെഴുതുക.
5. രണ്ട് ചിത്രങ്ങൾക്കിടയിൽ തിരശ്ചീനവും ലംബവുമായ സ്ഥലങ്ങൾ ലഭ്യമാക്കുന്ന ആട്രിബ്യൂട്ടുകൾ ഏതൊക്കെയാണ്?

നമുക്കു സംഗ്രഹിക്കാം

ഇന്റർനെറ്റിന്റെ സഫലത നിർണയിക്കുന്ന ഘടകമാണ് സുരക്ഷിതമായ ആശയവിനിമയം. ഇന്റർനെറ്റിലൂടെയുള്ള വ്യവഹാരങ്ങളുടെ സുരക്ഷിതത്വം നിർവഹിക്കുന്നത് HTTPS, ഡിജിറ്റൽ സാക്ഷ്യപത്രങ്ങൾ എന്നിവ ഉപയോഗിച്ചാണ്. ഡാറ്റ സംഭരിക്കുന്നതിനും കൈമാറ്റം (വിനിമയം) നടത്തുന്നതിനുമായി ഉപയോഗിക്കുന്ന വെബ് സെർവർ, സോഫ്റ്റ്‌വെയർ പോർട്ടുകൾ, DNS സെർവറുകൾ തുടങ്ങിയ അടിസ്ഥാന സൗകര്യങ്ങൾ ഇന്റർനെറ്റിന് ആവശ്യമാണ്. ഒന്നോ അതിലധികമോ വെബ് പേജുകൾ ചേർന്നാണ് വെബ്സൈറ്റ് ഉണ്ടായിരിക്കുന്നത്. HTML കോഡ് എഴുതിയോ അല്ലെങ്കിൽ വെബ് രൂപകൽപന ചെയ്യാനുപയോഗിക്കുന്ന സോഫ്റ്റ്‌വെയറുകളുടെ സഹായത്താലോ നമുക്ക് വെബ് പേജുകൾ നിർമിക്കാം. വെബ് പേജുകൾ പരിവർത്തനാത്മകമെന്നും സ്ഥിരമായതെന്നും തരംതിരിച്ചിട്ടുണ്ട്. സ്ക്രിപ്റ്റുകൾ ഉപയോഗിച്ചുകൊണ്ട് പരിവർത്തനാത്മക പേജുകൾ വികസിപ്പിക്കാം. ക്ലയന്റ് ഭാഗത്തെ സ്ക്രിപ്റ്റിങ് ഭാഷകളായ ജാവ സ്ക്രിപ്റ്റ്, VB സ്ക്രിപ്റ്റ് തുടങ്ങിയവ ഉപയോഗിച്ച് ക്ലയന്റ് ഭാഗത്ത് ഡാറ്റയുടെ സാധുത ഉറപ്പിക്കാനാവും. പരിവർത്തനാത്മക വെബ് പേജുകൾ സെർവറിൽ നിർമിക്കുന്നതിനായി സെർവർ ഭാഗത്തെ സ്ക്രിപ്റ്റു

കളായ PHP, ASP തുടങ്ങിയവ ഉപയോഗിക്കുന്നു. വെബ് സൈറ്റിന് ആകമാനം ഏകീകൃത ശൈലി ലഭ്യമാക്കാൻ കാസ്കേഡിങ് സ്റ്റൈൽ ഷീറ്റ് (CSS) ഉപയോഗിക്കുന്നു. HTML ഭാഷയെയും വെബ് പേജ് രൂപപ്പെടുത്തലിനെയും കുറിച്ചുള്ള അടിസ്ഥാന ആശയങ്ങൾ നാം ചർച്ച ചെയ്തു. വ്യത്യസ്തങ്ങളായ ടാഗുകളെയും അവയുടെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടുകളേയും നാം പരിചയപ്പെട്ടു. ചില ടാഗുകൾ ജോഡികളായും മറ്റ് ചിലത് ജോഡികളല്ലാതെയും ഉപയോഗിക്കുന്നു. വിവിധ ടാഗുകൾ കൊണ്ട് വാക്യങ്ങൾ ഭംഗിയായി വിന്യസിച്ച് നമുക്കൊരു വെബ് പേജ് നിർമിക്കാം. മാർക്യൂകൾ, ചിത്രങ്ങൾ, ശബ്ദങ്ങൾ, ചലിക്കുന്ന ചിത്രങ്ങൾ എന്നിവ ഉൾപ്പെടുത്തി വെബ് പേജുകളുടെ ഭംഗി വർധിപ്പിക്കാം. ഈ അധ്യായത്തിൽ ചർച്ച ചെയ്ത ആശയങ്ങളെക്കുറിച്ച് വ്യക്തമായ ധാരണയും HTML ഡോക്യുമെന്റുകൾ നിർമിക്കുന്നതിന് വേണ്ട പ്രായോഗിക പരിചയവും നമുക്കുള്ളതുകൊണ്ട് തുടർന്ന് വരുന്ന അധ്യായങ്ങളിലെ ആശയങ്ങൾ എളുപ്പത്തിൽ ഗ്രഹിക്കാവുന്നതാണ്. കൂടാതെ, മനോഹരമായ വെബ് സൈറ്റുകളുടെ മാതൃകയുണ്ടാക്കുന്നതിനും, വെബ് ആപ്ലിക്കേഷനുകൾ സ്വയം നിർമിക്കുന്നതിനും നാം പ്രാപ്തരാവുകയും ചെയ്യും.

നമുക്കു പരിശീലിക്കാം

1. താഴെ കൊടുത്തിരിക്കുന്ന സവിശേഷതകളോടുകൂടിയ കേരളത്തിന്റെ ഒരു വെബ് പേജ് നിർമിക്കുന്നതിന് വേണ്ട HTML കോഡ് എഴുതുക.
   - വാക്യങ്ങൾക്ക് രൂപഘടന നൽകുന്ന ടാഗുകളും ആട്രിബ്യൂട്ടുകളും ഉപയോഗിച്ച് ഒരു തലക്കെട്ടും അഞ്ച് വാചകങ്ങളുള്ള ഒരു ഖണ്ഡികയും തയാറാക്കുക.
   - വെബ് പേജിന്റെ പശ്ചാത്തല ചിത്രമായി ഒരു പ്രകൃതി ദൃശ്യം വേണം.
2. ചുവടെ കൊടുത്തിരിക്കുന്ന വിശദാംശങ്ങളും സവിശേഷതകളുമുള്ള നിങ്ങളുടെ സ്കൂൾ വെബ് പേജിന് ആവശ്യമായ HTML കോഡെഴുതുക.
   - പേജിന്റെ പശ്ചാത്തലത്തിന് ഒരു നിറം നൽകുക.
   - സ്കൂളിന്റെ ഒരു ചിത്രം നൽകുക.
3. ചുവടെ കൊടുത്തിരിക്കുന്ന വിശദാംശങ്ങളും സവിശേഷതകളുമുള്ള നിങ്ങളുടെ സ്കൂൾ വെബ് പേജിനാവശ്യമായ HTML കോഡെഴുതുക.
   - വാക്യങ്ങൾ രൂപപ്പെടുത്തുന്ന ടാഗുകളും ആട്രിബ്യൂട്ടുകളും ഉപയോഗിച്ചുകൊണ്ട് ഓരോ ജില്ലയ്ക്കും ഒരു തലക്കെട്ടും തുടർന്ന് മൂന്ന് വാചകങ്ങളുള്ള ഒരു ഖണ്ഡികയും തയാറാക്കുക.
   - സ്കൂളിന്റെ തപാൽ മേൽവിലാസം ഉൾപ്പെടുത്തുക.
   - "പുതിയ അധ്യയന വർഷത്തേക്കുള്ള പ്രവേശനം മെയ് മാസം 10-ാം തീയതി ആരംഭിക്കുന്നു" എന്ന മാർക്യൂ ഉൾപ്പെടുത്തുക.

4. ചുവടെ കൊടുത്തിരിക്കുന്ന സവിശേഷതകൾ ഉൾപ്പെടുത്തി നമ്മുടെ ദേശീയ ഗാനത്തിന്റെ വരികൾ പ്രദർശിപ്പിക്കുന്ന ഒരു വെബ് പേജിന് ആവശ്യമായ HTML കോഡ് എഴുതുക.
   - വ്യത്യസ്ത സവിശേഷതകൾ ഉള്ള ഫോണ്ട് ഉപയോഗിച്ച് ഒരു തലക്കെട്ട് നിർമിക്കണം.
   - നമ്മുടെ ദേശിയ പതാകയുടെ ഒരു ചിത്രം ഉൾപ്പെടുത്തണം.

നമുക്കു വിലയിരുത്താം

1. ഇന്റർനെറ്റിലൂടെയുള്ള ഡാറ്റയുടെ സഞ്ചാരത്തിൽ റൂട്ടറുകളുടെ പങ്ക് എന്ത്?
2. സമൂഹ മാധ്യമ വെബ് സൈറ്റുകൾ ആശയവിനിമയത്തിന് അവരുടെ സ്വന്തം പ്രോട്ടോകോളാണ് വികസിപ്പിച്ചിരിക്കുന്നത്. ഇന്റർനെറ്റിൽ TCP/IP പ്രോട്ടോകോൾ ഉപയോഗിക്കുന്നതു കൊണ്ട് ഇതെങ്ങനെ സാധ്യമാക്കും?
3. ഇ-മെയിൽ അക്കൗണ്ടിലേക്ക് യൂസർ നാമവും പാസ്‌വേഡും ഇന്റർനെറ്റിലൂടെ സുരക്ഷിതമായാണ് അയയ്ക്കേണ്ടത്.
   a. സെർവറിലേക്ക് ഡാറ്റ അയക്കാൻ ഉപയോഗിക്കുന്ന സുരക്ഷിത സാങ്കേതികവിദ്യയുടെ പേരെഴുതുക.
   b. സുരക്ഷിത ഡാറ്റാവിനിമയത്തിന് ഈ സാങ്കേതികവിദ്യ എങ്ങനെ സഹായിക്കുന്നു?
4. ഓൺ-ലൈനിലൂടെയുള്ള വ്യവഹാരത്തിൽ പേമെന്റ് ഗേറ്റ്‌വേയുടെ ധർമമെന്താണ്?
5. ABC എഞ്ചിനിയറിങ് കോളേജിന്റെ കാമ്പസിൽ ഏകദേശം 1000 കമ്പ്യൂട്ടറുകൾ ഇന്റർനെറ്റുമായി ബന്ധിപ്പിച്ചിട്ടുണ്ട്. ഒരു പ്രാദേശിക സെർവർ ഈ സാങ്കേതികവിദ്യാഭ്യാസ സ്ഥാപനത്തിൽ ഉണ്ടാകുന്നതിന്റെ നേട്ടമെന്താണ്?
6. ഒരു വെബ് സെർവർ ഓപ്പറേറ്റിങ് സിസ്റ്റത്തിനും വെബ് സെർവർ പാക്കേജിനും ഉദാഹരണമെഴുതുക.
7. വെബ് സെർവറിൽ സോഫ്റ്റ്‌വെയർ പോർട്ടിന്റെ ഉപയോഗമെന്ത്?
8. HTTP -ക്ക് ഉപയോഗിച്ചിട്ടുള്ള പോർട്ടാണ് __________.
9. നിങ്ങൾ www.prdkerala.org എന്ന വെബ്സൈറ്റ് പരിശോധിക്കുകയാണെന്ന് കരുതുക. DNS എങ്ങനെയാണ് IP മേൽവിലാസം വേർതിരിച്ചെടുക്കുന്നതെന്ന് വിശദീകരിക്കുക.
10. സ്ക്രിപ്റ്റുകൾ എന്നാൽ എന്ത്? സ്ക്രിപ്റ്റ് എഴുതാൻ ഉപയോഗിക്കുന്ന വിവിധ ഭാഷകൾ വിശദീകരിക്കുക.
11. നിങ്ങളുടെ സ്കൂൾ വെബ് സൈറ്റിലെ ഹോം പേജും 11-ാം ക്ലാസിലെ പരീക്ഷാഫലങ്ങൾ കാണിക്കുന്ന വെബ് പേജും പരിഗണിക്കുക.
   a. ഈ രണ്ട് പേജുകളും തമ്മിൽ നിർമാണത്തിലുള്ള വ്യത്യാസങ്ങൾ താരതമ്യം ചെയ്യുക.
   b. ഈ വെബ് പേജുകൾ വികസിപ്പിക്കുന്നതിന് ഉപയോഗിക്കുന്ന സാങ്കേതികവിദ്യകളെക്കുറിച്ച് എഴുതുക.

12. a. ജാവ സ്ക്രിപ്റ്റ് ഫയലിന്റെ ഫയൽ എക്സ്റ്റൻഷൻ ____________ ആണ്.
    b. ഒരു വെബ് പേജിൽ ജാവ സ്ക്രിപ്റ്റിന്റെ പ്രധാനപ്പെട്ട രണ്ട് ഉപയോഗങ്ങളെഴുതുക.
13. എന്താണ് അജാക്സ്? അതിന്റെ ഉപയോഗമെന്ത്?
14. നിങ്ങളുടെ ക്ലാസിലെ കുട്ടികൾക്ക് ഓരോ പരീക്ഷയ്ക്കും ലഭിച്ച മാർക്ക് പ്രദർശിപ്പിക്കുന്നതിനായി രവി ഒരു വെബ് പേജ് നിർമിക്കാൻ ആഗ്രഹിക്കുന്നു.
    a. അത് നിർവഹിക്കുന്നതിന് ഒരു സാങ്കേതിക വിദ്യ നിർദേശിക്കുക.
    b. നിങ്ങളുടെ നിർദേശം ന്യായീകരിക്കുക.
15. MYSQL-ലെ ഡാറ്റബേസ് ഉപയോഗിച്ചുകൊണ്ട് PHP-യിൽ മനോജ് ഒരു വെബ് സൈറ്റ് വികസിപ്പിക്കാൻ ആലോചിക്കുന്നു. ലിനക്സ് വെബ് ഹോസ്റ്റിംഗ് സംവിധാനമാണ് അയാൾ ഉപയോഗിക്കുന്നത് എങ്കിൽ, അതിന്റെ നിർവ്വഹണത്തിന് ആവശ്യമായവ എന്തെല്ലാമാണ്?
16. “മിക്കവാറും എല്ലാ വെബ് സൈറ്റുകളും വികസിപ്പിക്കുമ്പോൾ CSS ഉപയോഗിക്കാറുണ്ട്. വെബ് സൈറ്റുകളിൽ CSS ഉപയോഗിക്കുന്നത് കൊണ്ടുള്ള നേട്ടങ്ങൾ എന്തൊക്കെയാണ്?
17. HTML വികസിപ്പിച്ചത് ആരാണ്?
18. HTML-ൽ പ്രധാനമായും രണ്ട് ഭാഗങ്ങളാണുള്ളത്. അവയുടെ പേരുകൾ എഴുതുക.
19. നിങ്ങൾ വെബ്സൈറ്റുകൾ വിശകലനം ചെയ്യുകയാണെങ്കിൽ ലിങ്കുകൾ, സന്ദർശിച്ച ലിങ്കുകൾ, പശ്ചാത്തലം തുടങ്ങിയവയ്ക്ക് വിവിധ നിറങ്ങൾ കാണാനാകും. HTML-ൽ ഇത് എങ്ങനെ ചെയ്യുന്നുവെന്ന് ഉദാഹരണസഹിതം വിശദമാക്കുക.
20. HTML-ലെ ജോഡിയായ ടാഗുകളും ജോഡിയല്ലാത്ത ടാഗുകളും ഉദാഹരണസഹിതം താരതമ്യം ചെയ്യുക.
21. `Alink` ആട്രിബ്യൂട്ടിന്റെ തനതായ നിറം ഏതാണ്?
22. `Vlink` ആട്രിബ്യൂട്ടിന്റെ തനതായ നിറം ഏതാണ്?
23. താഴെ തന്നിട്ടുള്ള HTML വാക്കുകളെ തരംതിരിക്കുക.
    `BR, IMG, ALIGN, FONT, FACE`
24. `Noshade` ആട്രിബ്യൂട്ട് ഉള്ള ടാഗിന്റെ പേരെഴുതുക.
25. ഒരു വെബ് പേജിൽ ചിത്രഫയലുകൾ ഉൾക്കൊള്ളിക്കുന്നതിന് ഉപയോഗിക്കുന്ന `<IMG>` ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ട് ഏത്?
26. `<IMG>` ടാഗിൽ `Alt` ആട്രിബ്യൂട്ടിന്റെ ആവശ്യകത പ്രസ്താവിക്കുക.
27. `<IMG>` ടാഗ് ഉപയോഗിക്കുമ്പോൾ ലഭിക്കുന്ന ഒരു ചിത്രത്തിന്റെ തനത് വിന്യാസം ____________ ആകുന്നു.
28. `<FONT>` ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടുകളുടെ നാമാവലി തയാറാക്കുക.

R7B9B7

5 HTML ഉപയോഗിച്ചുള്ള വെബ് രൂപകൽപന

പ്രധാന പഠനനേട്ടങ്ങൾ

പഠനം പൂർത്തിയാകുന്നതോടെ പഠിതാവ് ആർജി ക്കേണ്ട പഠനനേട്ടങ്ങൾ

- വെബ് പേജുകളിലെ വിഭവങ്ങൾ ഫലപ്രദമായി അവതരിപ്പിക്കാൻ വിവിധ ഇനം ലിസ്റ്റുകൾ ഉപയോഗിക്കുന്നു.
- ഹൈപ്പർ ലിങ്കിന്റെ പ്രസക്തി തിരിച്ചറിഞ്ഞ് വിവിധ തരം ലിങ്കിങ്ങിനായി <A> ടാഗ് ഉപയോഗിക്കുന്നു.
- <EMBED> ടാഗ് ഉപയോഗിച്ച് ഓഡിയോ, വീഡിയോ എന്നിവ വെബ് പേജിൽ ഉൾപ്പെടുത്തുന്നു.
- വെബ് പേജിനുള്ളിൽ പശ്ചാത്തലസംഗീതവും വീഡിയോയും ഉൾപ്പെടുത്തുന്നു.
- ടേബിൾ നിർമിക്കാൻ ആവശ്യമായ ടാഗുകളുടെയും അവയുടെ ആട്രിബ്യൂട്ടുകളുടെയും പട്ടിക തയാറാക്കി വിശദീകരിക്കുന്നു.
- വിവിധ സവിശേഷതകളുള്ള ടേബിളിന്റെ രൂപകൽപനയ്ക്കായി <TABLE> ടാഗുമായി ബന്ധപ്പെട്ട ടാഗുകൾ ഉപയോഗിക്കുന്നു.
- വ്യത്യസ്തങ്ങളായ ടേബിൾ രൂപകൽപനക്കായി വേണ്ട ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും മനസിലാക്കുന്നു.
- വെബ് പേജിനുള്ളിൽ ഫ്രെയിമുകളുടെ ആവശ്യകത തിരിച്ചറിയുന്നു.
- ഫ്രെയിം ഉപയോഗിച്ച് ഒരേ ബ്രൗസർ ജാലകത്തിൽ തന്നെ ഒന്നിലധികം വെബ് പേജുകൾ പ്രദർശിപ്പിക്കുന്നതിന് വേണ്ട ടാഗുകൾ ഉപയോഗിക്കുന്നു.
- വെബ് പേജിനുള്ളിൽ ഫോം എന്ന ആശയം തിരിച്ചറിയുന്നു.
- ഒരു ഫോമിന്റെ വിവിധ ഭാഗങ്ങളും അവ നിർമിക്കുന്നതിനാവശ്യമായ ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും പരിചയപ്പെടുന്നു.
- ടേബിളുകൾ, ഫ്രെയിമുകൾ, ഫോം എന്നിവ ഉപയോഗിച്ച് വെബ് പേജുകൾ രൂപകൽപന ചെയ്യുന്നു.

HTML -ലെ അടിസ്ഥാന ടാഗുകൾ കഴിഞ്ഞ അധ്യായത്തിലൂടെ നാം പഠിച്ചു. അത്തരം ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും ഉപയോഗിച്ച് ലളിതമായ വെബ് പേജുകൾ രൂപകൽപന ചെയ്യുന്നതും നാം പഠിച്ചു. എന്നാൽ കൂടുതൽ സൗകര്യങ്ങളും അവശ്യസേവനങ്ങളും നൽകുന്ന മറ്റ് ചില വെബ് സൈറ്റുകളും നമുക്ക് പരിചിതമാണല്ലോ? വിവിധതരം ലിസ്റ്റുകൾ അടങ്ങുന്ന വെബ്സൈറ്റുകളുണ്ട്. എങ്ങനെയാണ് വിവരങ്ങൾ പട്ടികാരൂപത്തിൽ രേഖപ്പെടുത്തുന്നത്? വെബ് പേജുകളുടെ തമ്മിലുള്ള പരസ്പര ബന്ധമാണ് വേൾഡ് വൈഡ് വെബിന്റെ നട്ടെല്ല്. വിവിധതരം ലിങ്കിങ്ങിനെ (Linking) കുറിച്ച് ഈ അധ്യായത്തിൽ നമ്മൾ ചർച്ച ചെയ്യുന്നുണ്ട്. ചില വിവരങ്ങൾ പട്ടികാരൂപത്തിൽ കാണപ്പെടാറുണ്ട്. ചില സമയത്ത് ഒരേ ബ്രൗസർ ജാലകത്തിൽ ഒന്നിലധികം വെബ്പേജുകൾ നമ്മൾ കാണാറുണ്ട്. അതുപോലെതന്നെ പരീക്ഷയുടെ മാർക്ക് ലിസ്റ്റ് ലഭിക്കുവാനായി രജിസ്റ്റർ നമ്പർ നൽകുന്നതിനും സ്കൂൾ, കോളേജ് പ്രവേശനം, സ്കോളർഷിപ്പ് എന്നിവയ്ക്കുള്ള അപേക്ഷകൾ സമർപ്പിക്കുന്നതിനും, വൈദ്യുതി, വെള്ളം എന്നിവയ്ക്കുള്ള ബിൽ അടയ്ക്കുന്നതിനുമുള്ള വെബ്സൈറ്റുകൾ നമുക്ക് പരിചിതങ്ങളാണ്. ഒരേ ബ്രൗസർ ജാലകത്തിൽ ഒന്നിലധികം വെബ് പേജുകൾ ഉൾക്കൊള്ളിക്കാനാകുമോ? ഉൾക്കൊള്ളിക്കാൻ ആകുമെങ്കിൽ അത് എങ്ങനെ? ഉപയോക്താവിൽ നിന്നും ഡാറ്റ സ്വീകരിച്ച് കൂടുതൽ വിവരങ്ങൾ നൽകുന്ന വെബ് പേജുകൾ ഉണ്ടാക്കുന്നതെങ്ങനെ? വെബ് പേജ് നിർമാതാക്കൾക്ക് ഇത്തരത്തിലുള്ള എല്ലാ സൗകര്യങ്ങളും HTML-ൽ ലഭ്യമാണ്. ഈ ചോദ്യങ്ങൾക്കുള്ള ഉത്തരങ്ങൾ ഈ അധ്യായത്തിലൂടെ ചർച്ച ചെയ്യുന്നു.

5.1 HTML-ലെ ലിസ്റ്റുകൾ (Lists in HTML)

വിവരങ്ങൾ അവതരിപ്പിക്കുമ്പോൾ ലിസ്റ്റുകൾ ആശയവിനിമയ സാധ്യത വർധിപ്പിക്കുന്നു. വിവിധതരം ലിസ്റ്റുകളുണ്ട്. അക്കമിട്ട ലിസ്റ്റുകളും ബുള്ളറ്റിട്ട ലിസ്റ്റുകളും നമുക്ക് പരിചിതമാണല്ലോ. ഇത്തരം ലിസ്റ്റുകൾ വെബ്പേജിൽ ഉൾപ്പെടുത്താൻ HTML സൗകര്യം ചെയ്യുന്നു. എല്ലാ ലിസ്റ്റുകളിലും ഒന്നോ അതിലധികമോ ഘടകങ്ങൾ ഉണ്ടായിരിക്കണം. HTML-ലെ ലിസ്റ്റുകളെ ക്രമരഹിത ലിസ്റ്റ്, ക്രമലിസ്റ്റ്, ഡെഫനിഷൻ ലിസ്റ്റ് എന്നിങ്ങനെ മൂന്നായി തരംതിരിച്ചിരിക്കുന്നു.

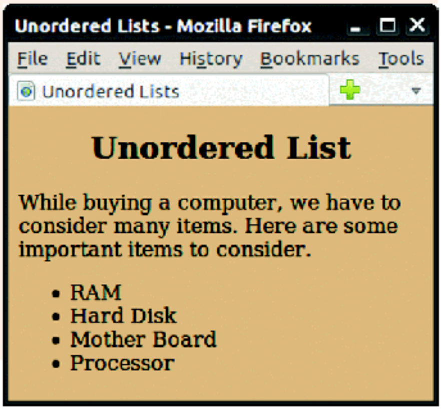
5.1.1 ക്രമരഹിത ലിസ്റ്റുകൾ (Unordered lists)

ക്രമരഹിത ലിസ്റ്റ് അഥവാ ബുള്ളറ്റഡ് ലിസ്റ്റുകളിൽ ഓരോ ഇനത്തിന്റെയും മുന്നിൽ ഒരു ബുള്ളറ്റോ അല്ലെങ്കിൽ മറ്റേതെങ്കിലും ഗ്രാഫിക് ചിഹ്നങ്ങളോ പ്രദർശിപ്പിക്കുന്നു. **<UL>**, **</UL>** എന്നീ ടാഗ് ജോഡികൾ ഉപയോഗിച്ച് ക്രമരഹിത ലിസ്റ്റുകൾ നിർമിക്കുവാൻ നമുക്ക് സാധിക്കും. ലിസ്റ്റിലെ ഒരോ ഇനവും **<LI>**, **</LI>** ടാഗ് ജോഡികൾ ഉപയോഗിച്ച് ചേർക്കപ്പെടുന്നു. ലിസ്റ്റിലെ ഇനങ്ങൾ യാതൊരു ക്രമവും ഇല്ലാതെ പ്രദർശിപ്പിക്കുന്നതിനാണ് ക്രമരഹിത (Unordered) ലിസ്റ്റുകൾ ഉപയോഗിക്കുന്നത്.

ഉദാഹരണം 5.1-ൽ ഒരു കമ്പ്യൂട്ടറിന്റെ ഏതാനും ഹാർഡ്‌വെയർ ഭാഗങ്ങൾ ബുള്ളറ്റഡ് ലിസ്റ്റായി പ്രദർശിപ്പിക്കുന്നതിനുള്ള HTML കോഡ് നൽകുന്നു. ഇതിനനുസൃതമായ വെബ്പേജ് ചിത്രം 5.1 ൽ കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 5.1: ക്രമരഹിത ലിസ്റ്റ് നിർമിക്കാനാവശ്യമായ HTML കോഡ്

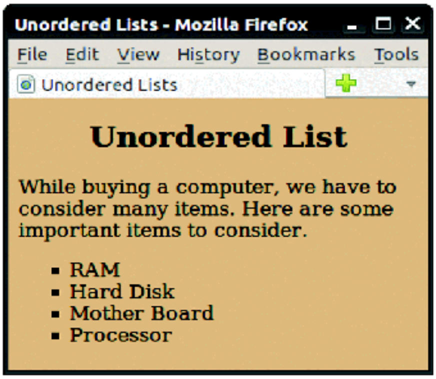
```
<HTML>
<HEAD>
   <TITLE> Unordered Lists </TITLE>
</HEAD>
<BODY Bgcolor= "#DEB887">
   <CENTER> <H2> Unordered List </H2> </CENTER>
   While buying a computer, we have to consider many items.
   Here are some important items
   to consider.
   <UL>
     <LI> RAM </LI>
     <LI> Hard Disk </LI>
     <LI> Mother Board </LI>
     <LI> Processor </LI>
   </UL>
</BODY>
</HTML>
```



ചിത്രം 5.1: ക്രമരഹിത ലിസ്റ്റ് ഉപയോഗിച്ച വെബ് പേജ്

<UL> ടാഗിലെ Type ആട്രിബ്യൂട്ടിന് Disc (തനതുവില), Square, Circle എന്നീ മൂന്ന് വിലകൾ നൽകാവുന്നതാണ്. ഇങ്ങനെ ചെയ്യുമ്പോൾ ലിസ്റ്റിലെ ഓരോ ഇനത്തിന്റെയും മുന്നിൽ അതിനനുസൃതമായ ബുള്ളറ്റ് ചിത്രീകരിക്കുന്നു. താഴെ നൽകിയിട്ടുള്ള HTML കോഡ് ചിത്രം 5.2 ൽ കാണിച്ചിരിക്കുന്ന ലിസ്റ്റ് അടങ്ങിയ വെബ് പേജ് പ്രദർശിപ്പിക്കുന്നു.

```
<UL Type= "Square">
    <LI> RAM </LI>
    <LI> Hard Disk </LI>
    <LI> Mother Board </LI>
    <LI> Processor </LI>
</UL>
```



ചിത്രം 5.2: ബുള്ളറ്റഡ് ക്രമരഹിത ലിസ്റ്റ്

> <UL> ടാഗിന്റെ Type ആട്രിബ്യൂട്ടിന് None എന്ന വില നൽകുകയാണെങ്കിൽ ക്രമരഹിത ലിസ്റ്റിലെ ഓരോ ഇനത്തിന്റെയും മുമ്പിൽ ബുള്ളറ്റുകളൊന്നും പ്രദർശിപ്പിക്കുകയില്ല. മുകളിൽ കൊടുത്തിരിക്കുന്ന HTML കോഡിൽ <UL Type = "None"> എന്ന മാറ്റം വരുത്തി പരിശോധിക്കാം.

5.1.2 ക്രമലിസ്റ്റ് (Ordered lists)

ലിസ്റ്റിലെ ഇനങ്ങളുടെ മുന്നിൽ സംഖ്യകളോ അക്ഷരങ്ങളോ ചേർത്ത് ക്രമമായി പ്രദർശിപ്പിക്കുന്നതിന് ക്രമലിസ്റ്റ് ഉപയോഗിക്കുന്നു. **<OL>, </OL>** എന്നീ ടാഗ് ജോഡികൾ ഉപയോഗിച്ചാണ് ഇത്തരം ലിസ്റ്റ് തയാറാക്കുന്നത്. ക്രമലിസ്റ്റിലെ ഓരോ ഇനവും പ്രദർശിപ്പിക്കുന്നതിന് **<LI>, </LI>** എന്നീ ടാഗ് ജോഡികൾ ഉപയോഗിക്കുന്നു. ഇതിനെ സംഖ്യാനുസൃത ലിസ്റ്റ് എന്നും പറയാറുണ്ട്.

ഉദാഹരണം 5.2 ൽ ക്രമലിസ്റ്റ് നിർമിക്കാനാവശ്യമായ കോഡ് കൊടുത്തിരിക്കുന്നു. ഇതിനനുസൃതമായ വെബ്പേജ് ചിത്രം 5.3 ൽ പ്രദർശിപ്പിക്കുന്നു.

ഉദാഹരണം 5.2: ക്രമലിസ്റ്റ് നിർമിക്കാനാവശ്യമായ HTML കോഡ്

```
<HTML>
<HEAD>
    <TITLE> Ordered Lists </TITLE>
</HEAD>
    <BODY Bgcolor= "#DDA0DD">
    <H2 Align= "center"> Ordered List </H2>
    Consider the memory devices of a computer.
    Then according to the speed of data processing,
```

```
we can arrange the memory devices as follows.
    <OL>
      <LI> Registers </LI>
      <LI> Cache </LI>
      <LI> RAM </LI>
      <LI> Hard Disk </LI>
    </OL>
</BODY>
</HTML>
```

ചിത്രം 5.3 ൽ 1 മുതൽ 4 വരെയുള്ള ക്രമനമ്പർ ഉൾക്കൊള്ളുന്ന ലിസ്റ്റ് പ്രദർശിപ്പിക്കുന്നു. നമുക്ക് മറ്റ് പല സംഖ്യാരീതികളും ഉപയോഗിച്ച് ലിസ്റ്റിലെ ഇനങ്ങൾ പ്രദർശിപ്പിക്കാം. **Type** ആട്രിബ്യൂട്ടിന് ചുവടെ കൊടുത്തിരിക്കുന്ന വിലകൾ നൽകി അനുയോജ്യമായ വിധത്തിൽ ലിസ്റ്റിലെ ഇനങ്ങളുടെ സംഖ്യാരീതിയിൽ നമുക്ക് മാറ്റം വരുത്താം.

ചിത്രം 5.3: ക്രമലിസ്റ്റ് അടങ്ങിയ വെബ് പേജ്

1 തനതായ സംഖ്യാരീതി (1, 2, 3, ...)

A ഇംഗ്ലീഷ് ഭാഷയിലെ വലിയ അക്ഷരങ്ങൾ (A, B, C, ...)

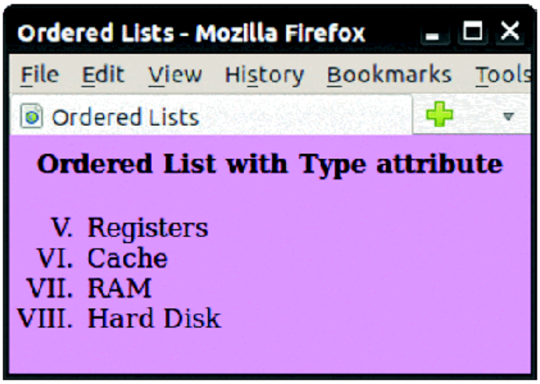
a ഇംഗ്ലീഷ് ഭാഷയിലെ ചെറിയ അക്ഷരങ്ങൾ (a, b, c, ...)

I റോമൻ ലിപിയിലെ വലിയ സംഖ്യകൾ (I, II, III, ...)

i റോമൻ ലിപിയിലെ ചെറിയ സംഖ്യകൾ (i, ii, iii, ...)

ഒരു ക്രമലിസ്റ്റിലെ ഇനങ്ങളുടെ ക്രമനമ്പറിന്റെ തുടക്കം സ്വാഭാവികമായും ആ സംഖ്യാരീതിയിലെ ആദ്യ നമ്പറിലായിരിക്കും. അതായത് ക്രമനമ്പർ തുടങ്ങുന്നത് 1, A, a, I, i എന്നിവയിൽ ഏതെങ്കിലും ഒന്നിലായിരിക്കും. ഇത് നമ്മുടെ ഇഷ്ടാനുസരണം ഏത് സംഖ്യയിലോ അക്ഷരത്തിലോ ആരംഭിക്കാം. ഇതിനായി സ്റ്റാർട്ട് (**Start**) ആട്രിബ്യൂട്ട് ഉപയോഗിക്കാം. ഒരു ക്രമലിസ്റ്റിന്റെ ഇനങ്ങളുടെ ക്രമനമ്പർ 5 ൽ തുടങ്ങണമെങ്കിൽ **<OL Start= "5">** എന്നതിലൂടെ സാധ്യമാകും. അതിനാൽ 5 ൽ തുടങ്ങിയ ക്രമമ്പർ 6, 7, 8, ... എന്നിങ്ങനെ തുടരുന്നതാണ്.

ഏത് സംഖ്യാരീതി ഉപയോഗിക്കുമ്പോഴും **Start** ആട്രിബ്യൂട്ടിന് നൽകുന്ന വില പൂർണസംഖ്യ ആയിരിക്കും. ഉദാഹരണത്തിന് താഴെ കൊടുത്തിരിക്കുന്ന ക്രമലിസ്റ്റിന്റെ തുടക്കനമ്പർ V ആണ്, പിന്നീട് VI, VII, VIII, ... എന്നിങ്ങനെ തുടരുന്നു. ഇതിന്റെ ഔട്ട്പുട്ട് ചിത്രം 5.4-ൽ കാണിച്ചിരിക്കുന്നു.



ചിത്രം 5.4: സംഖ്യാരീതിയും തുടക്കവിലയും വ്യത്യസ്തമായ ക്രമലിസ്റ്റ്

```
    <BODY Bgcolor ="#DDA0DD">
    <H4 Align="center">Ordered List with Type attribute</H4>
    <OL Type= "I" Start= "5">
      <LI> Registers </LI>
      <LI> Cache </LI>
      <LI> RAM </LI>
      <LI> Hard Disk </LI>
    </OL>
</BODY>
```

ഒരു ക്രമലിസ്റ്റിലെ ഇനങ്ങളുടെ ക്രമനമ്പർ അവരോഹണ ക്രമത്തിൽ പ്രദർശിപ്പിക്കാം. ഇതിനായി <OL> ടാഗിന്റെ ആട്രിബ്യൂട്ടായ Reversed ഉപയോഗിക്കാം. ഉദാഹരണം 5.2 ൽ നൽകിയിരിക്കുന്ന HTML കോഡിൽ <OL Reversed> എന്ന മാറ്റം വരുത്തി നമുക്ക് പരിശോധിക്കാം. Reversed ഒരു ബൂളിയൻ ആട്രിബ്യൂട്ട് ആയതിനാൽ ഇതിന് വില നൽകേണ്ടതില്ല.

ഒരു ക്രമലിസ്റ്റിലെ ഇനങ്ങളുടെ ക്രമനമ്പർ നമുക്ക് അനുയോജ്യമായ വില നൽകി പ്രദർശിപ്പിക്കുവാൻ <LI> ടാഗിന്റെ ആട്രിബ്യൂട്ടായ Value ഉപയോഗിക്കാം. മുകളിൽ കൊടുത്തിരിക്കുന്ന HTML കോഡിൽ <LI Value = "10"> RAM </LI> എന്നാണ് നൽകുന്നതെങ്കിൽ, 10 RAM എന്നായിരിക്കും വെബ് പേജിൽ പ്രദർശിപ്പിക്കുക. ഇതേ രീതിയിൽ <OL> ടാഗിനുള്ളിലെ ഓരോ <LI> ഇനങ്ങൾക്കും Value ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് ഓരോരോ വിലകൾ നൽകാം.

5.1.3 ഡെഫനിഷൻ ലിസ്റ്റ് (Definition lists)

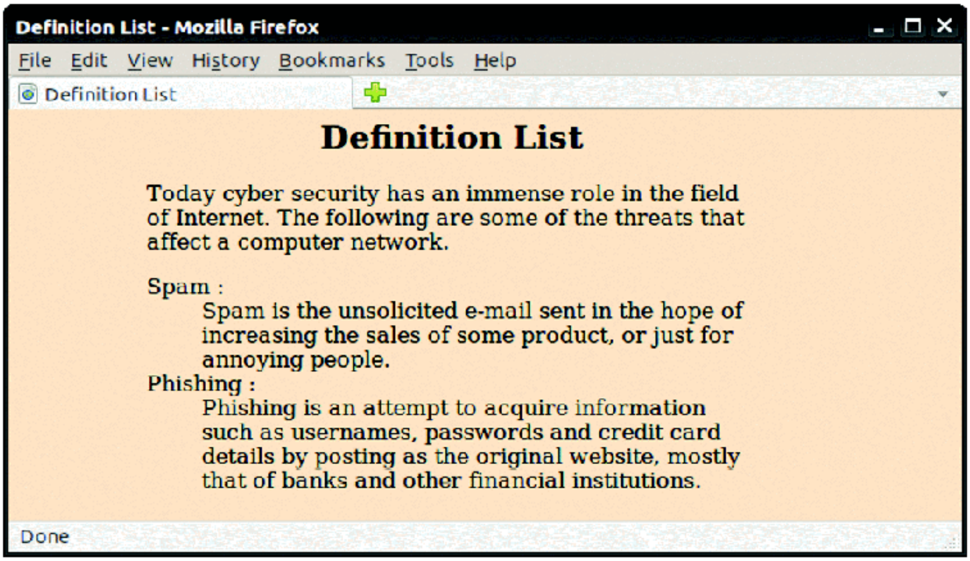
കുറച്ച് പദങ്ങളും അവയുടെ നിർവചനങ്ങളും നൽകുന്നതിന് ഡെഫനിഷൻ ലിസ്റ്റ് ഉപയോഗിക്കുന്നു. തൊട്ടടുത്ത വരിയിൽ പദങ്ങളിൽ നിന്നും തിരശ്ചീനമായി നിശ്ചിത അകലം കഴിഞ്ഞതിന് ശേഷമാണ് നിർവചനങ്ങൾ പ്രദർശിപ്പിക്കുന്നത്. **<DL>**, **</DL>** എന്നീ ടാഗ് ജോഡികൾ ഉപയോഗിച്ച് ഡെഫനിഷൻ ലിസ്റ്റ് നിർമിക്കുവാൻ നമുക്ക് സാധിക്കും. ഇവിടെ ബുള്ളറ്റുകളോ സംഖ്യകളോ ഇനങ്ങൾക്ക് മുൻപിൽ പ്രദർശിപ്പിക്കാറില്ല. **<DT>** ടാഗ് പദങ്ങൾ നിർമിക്കുന്നതിനും **<DD>** ടാഗ് അതിന്റെ നിർവചനങ്ങൾ ചേർക്കുന്നതിനും ഉപയോഗിക്കുന്നു.

ഇന്റർനെറ്റ് സുരക്ഷയുമായി ബന്ധപ്പെട്ട പദങ്ങളും അവയുടെ നിർവചനങ്ങളും പ്രദർശിപ്പിക്കുന്നതിനുള്ള വെബ് പേജിന്റെ HTML കോഡ് ഉദാഹരണം 5.3-ൽ നൽകിയിരിക്കുന്നു. ഇതിന്റെ ഔട്ട് പുട്ട് ചിത്രം 5.5-ൽ കാണാം.

ഉദാഹരണം 5.3: ഡെഫനിഷൻ ലിസ്റ്റ് നിർമാണം

```
<HTML>
<HEAD> <TITLE> Definition List </TITLE> </HEAD >
<BODY Bgcolor= "#FFE4C4" Leftmargin= "100" Rightmargin= "150">
```

```
    <H2 Align= "center"> Definition List </H2>
      Today cyber security has an immense role in the
      field of Internet. The following are some of
      the threats that affect a computer network.
    <DL>
      <DT>Spam :</DT>
      <DD> Spam is the unsolicited e-mail sent in the
           hope of increasing the sales of some product, or
           just for annoying people.</DD>
      <DT>Phishing :</DT>
      <DD> Phishing is an attempt to acquire information
           such as usernames, passwords and credit card details
           by posting as the original website, mostly that
           of banks and other financial institutions. </DD>
    </DL>
</BODY>
</HTML>
```



ചിത്രം 5.5: ഡെഫനിഷൻ ലിസ്റ്റ് ഉൾക്കൊള്ളുന്ന വെബ് പേജ്

ചിത്രം 5.5-ലെ നിർവചനങ്ങളുടെ ഇൻഡന്റേഷനും, ഇടതും വലതും ഭാഗങ്ങളിലുള്ള മാർജിനുകളും ശ്രദ്ധിക്കുക.

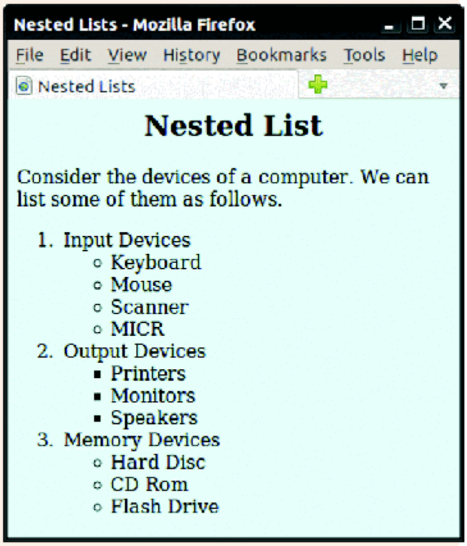
5.1.4 നെസ്റ്റഡ് ലിസ്റ്റുകൾ (Nested lists)

ഒരു ലിസ്റ്റിലെ ഏതെങ്കിലും ഇനത്തിന്റെ താഴെ മറ്റൊരു ലിസ്റ്റ് നൽകുവാനാകും. ഇത്തരം ലിസ്റ്റുകളെ നെസ്റ്റഡ് ലിസ്റ്റ് എന്നറിയപ്പെടുന്നു. ഇത് വിവിധ തരത്തിൽ തയാറാക്കാം.

ഉദാഹരണമായി ഒരു ക്രമലിസ്റ്റിനുള്ളിൽ മറ്റൊരു ക്രമലിസ്റ്റോ, ക്രമരഹിതലിസ്റ്റോ നൽകുവാനാകും. അതുപോലെ ഒരു ക്രമരഹിത ലിസ്റ്റിനുള്ളിൽ മറ്റൊരു ക്രമലിസ്റ്റോ ക്രമരഹിതലിസ്റ്റോ നൽകാനുമാകും. ഉദാഹരണം 5.4 ൽ നൽകുന്ന HTML കോഡ് നെസ്റ്റഡ് ലിസ്റ്റിന്റെ ആശയം വിശദമാക്കുന്നു.

ഉദാഹരണം 5.4: നെസ്റ്റഡ് ലിസ്റ്റ് നിർമാണം

```
<HTML>
<HEAD>
   <TITLE> Nested Lists </TITLE>
</HEAD>
<BODY Bgcolor= "#E0FFFF">
   <H2 Align= "center"> Nested List </H2>
   Consider the devices of a computer.
   We can list some of them as follows.
   <OL>
      <LI> Input Devices </LI>
      <UL>
         <LI>Keyboard</LI>
         <LI>Mouse</LI>
         <LI>Scanner</LI>
         <LI>MICR</LI>
      </UL>
      <LI> Output Devices </LI>
      <UL Type= "Square">
         <LI>Printers</LI>
         <LI>Monitors</LI>
         <LI>Speakers</LI>
      </UL>
      <LI> Memory Devices </LI>
      <UL Type= "Circle">
         <LI>Hard Disc</LI>
         <LI>CD Rom</LI>
         <LI>Flash Drive</LI>
      </UL>
   </OL>
</BODY>
</HTML>
```



ചിത്രം 5.6: നെസ്റ്റഡ് ലിസ്റ്റ് ഉപയോഗിച്ച് ഉപകരണങ്ങളുടെ തരംതിരിവ് പ്രദർശിപ്പിക്കുന്നു

ഉദാഹരണം 5.4 ൽ മൂന്ന് ക്രമരഹിത ലിസ്റ്റുകൾ ഒരു ക്രമലിസ്റ്റിലേക്ക് നെസ്റ്റ് ചെയ്തിരിക്കുന്നു.

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. HTML ലെ വിവിധതരം ലിസ്റ്റുകൾ ഏതൊക്കെയാണ്?
2. നിങ്ങളുടെ ക്ലാസിലെ കുട്ടികളുടെ ലിസ്റ്റ് തയാറാക്കുവാൻ അധ്യാപകൻ ആവശ്യപ്പെട്ടു എന്നു കരുതുക. ഏതുതരം ലിസ്റ്റായിരിക്കും നിങ്ങൾ തിരഞ്ഞെടുക്കുന്നത്? എന്തുകൊണ്ട്?
3. <UL>, <OL> എന്നീ ടാഗുകളുടെ പൊതുവായ ആട്രിബ്യൂട്ടുകൾ എന്തൊക്കെയാണ്?
4. <UL>, <OL> എന്നീ ടാഗുകൾ തമ്മിലുള്ള വ്യത്യാസങ്ങൾ എഴുതുക.
5. ഡെഫനിഷൻ ലിസ്റ്റിൽ ഉപയോഗിക്കുന്ന ടാഗുകളുടെ പേര് എഴുതുക.

5.2 ലിങ്കുകളുടെ നിർമാണം (Creating links)

ഒരു വെബ് പേജിനെ മറ്റൊരു വെബ് പേജുമായോ ഒരു വെബ് പേജിന്റെ ഏതെങ്കിലും ഭാഗത്തെ അതേ പേജിലെ മറ്റൊരു ഭാഗവുമായോ ബന്ധിപ്പിക്കുന്നതിന് സഹായിക്കുന്ന ഘടകമാണ് ഹൈപ്പർലിങ്ക്. ഹൈപ്പർലിങ്കിനെ സാധാരണയായി ലിങ്കുകൾ എന്നാണ് വിളിക്കുന്നത്. HTML-ൽ **<A>** ടാഗിനും **</A>** ടാഗിനും ഇടയിൽ എന്ത് നൽകിയാലും അത് ലിങ്കിന്റെ ഭാഗമാകും. ഉപയോക്താവിന് അവിടെ ക്ലിക്ക് ചെയ്തുകൊണ്ട് ലിങ്ക് ചെയ്തിരിക്കുന്ന ഡോക്യുമെന്റിലേക്ക് എത്താനാകും. <A> ടാഗിന്റെ മുഖ്യ ആട്രിബ്യൂട്ടാണ് ഹൈപ്പർ റഫറൻസ് എന്ന് അറിയപ്പെടുന്ന **Href**. ഇതിന്റെ വിലയായി സാധാരണ URL ആണ് നൽകുന്നത് (ഹൈപ്പർലിങ്ക് ബന്ധിപ്പിക്കേണ്ട വെബ് പേജിന്റെ വിലാസം). ഉദാഹരണത്തിന് താഴെകൊടുത്തിരിക്കുന്ന കോഡ് ഭാഗം ശ്രദ്ധിക്കുക.

```
<A Href= "http://www.dhsekerala.gov.in">Higher Secondary</A>
```

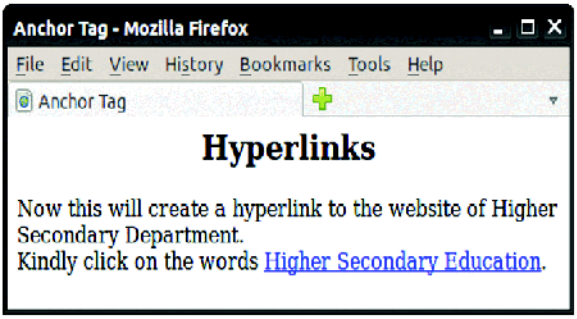
ഇത് "http:// www.dhsekerala.gov.in." എന്ന വെബ്സൈറ്റിനെ ലക്ഷ്യം വയ്ക്കുന്ന ഹൈപ്പർലിങ്ക് സൃഷ്ടിക്കുന്നു. ഉപയോക്താവ് ഈ ലിങ്കിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ ഈ URL-ന്റെ ഹോംപേജ് ബ്രൗസറിൽ ലഭ്യമാകും. <A>, </A> എന്നിവയ്ക്കിടയിൽ നൽകുന്ന വാക്യങ്ങൾ വ്യത്യസ്ത നിറത്തിലും അടിവരയോടുകൂടിയും കാണപ്പെടും.

ഹൈപ്പർലിങ്ക് സൃഷ്ടിക്കുന്ന HTML കോഡ് ഉദാഹരണം 5.5ലും അതിന്റെ ചിത്രം 5.7ലും കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 5.5: ഒരു വെബ് പേജിൽ ഹൈപ്പർലിങ്കിന്റെ നിർമാണം

```
<HTML>
<HEAD>
   <TITLE> Anchor Tag </TITLE>
</HEAD>
<BODY Bgcolor= "#FFFFFF">
   <H2 Align= "center"> Hyperlinks </H2>
   <P>Now this will create a hyperlink to the website of
      Higher Secondary Department.<BR>
```

```
        Kindly click on the words
    <A Href= "http://www.dhsekerala.gov.in">Higher Secondary
        Education</A>.
</BODY>
</HTML>
```



ചിത്രം 5.7: ഹൈപ്പർ ലിങ്ക് ഉൾക്കൊള്ളുന്ന വെബ് പേജ്

ചിത്രം 5.7 ൽ കാണിച്ചിരിക്കുന്നതു പോലെ ഹൈപ്പർ ലിങ്ക് ചെയ്ത വാക്യം (Higher Secondary Education) അടിവരയിട്ട രീതിയിലും വ്യത്യസ്തമായ നിറത്തിലും ആയിരിക്കും.
ഹൈപ്പർ ലിങ്കുകളുടെ ലക്ഷ്യസ്ഥാനത്തിനനുസരിച്ച് അവയെ ഇന്റേണൽ അല്ലെങ്കിൽ എക്സ്റ്റേണൽ എന്ന് തരംതിരിച്ചിരിക്കുന്നു.

5.2.1 ഇന്റേണൽ ലിങ്കിങ് (Internal linking)

ഒരു വെബ് പേജിനെ അതിന്റെ തന്നെ മറ്റൊരു ഭാഗവുമായി ബന്ധിപ്പിക്കുന്നത് ഇന്റേണൽ ലിങ്കിങ് എന്നറിയപ്പെടുന്നു. ഇതിനായി **Name** ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് ലിങ്ക് ചെയ്യേണ്ട ഭാഗം തിരിച്ചറിയുവാൻ പ്രത്യേകം നാമകരണം ചെയ്യേണ്ടതുണ്ട്. ഈ പേര് <A> ടാഗിന്റെ ആട്രിബ്യൂട്ടായ Href-ന്റെ വിലയായി നൽകി ലിങ്കിങ് സാധ്യമാകും.
ഉദാഹരണത്തിന് "Environment Pollution" എന്ന വിഷയത്തെ സംബന്ധിക്കുന്ന ഒരു വെബ് പേജ് നിർമിക്കണം എന്ന് കരുതുക. ഇതിൽ മൂന്ന് ഖണ്ഡികകളിലായി "Introduction", "Air Pollution", "Water Pollution" എന്നിവ നൽകിയിട്ടുണ്ട്. ഈ ഖണ്ഡികകളെ സൂചിപ്പിക്കുന്നതിനായി <A> ടാഗിന്റെ Name ആട്രിബ്യൂട്ടിന്റെ വിലകളായി Introduction, Air, Water എന്നീ പേരുകൾ നൽകിയിരിക്കുന്നു.

```
<A Name= "Introduction"> INTRODUCTION </A>
<A Name= "Air"> Air Pollution </A>
<A Name= "Water"> Water Pollution </A>
```

വെബ് പേജിന്റെ ഈ ഭാഗങ്ങളിലേക്ക് ബന്ധിപ്പിക്കുന്നതിനായി <A> ടാഗിന്റെ Href ആട്രിബ്യൂട്ടിന് യഥാക്രമം #Introduction, #Air, #Water (# ചിഹ്നം നിർബന്ധമാണ്) എന്നീ വിലകൾ നൽകണം. ഇതിനുള്ള HTML കോഡ് ഭാഗം താഴെക്കൊടുക്കുന്നു.

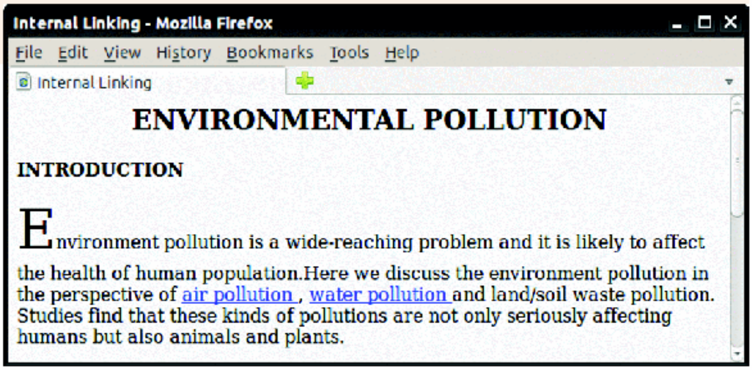
```
<A Href = "#Introduction"> Go to Introduction </A>
<A Href = "#Air"> Air pollution </A>
<A Href = "#Water"> Water pollution </A>
```

ഇന്റേണൽ ലിങ്കിന്റെ ആശയം ഉൾപ്പെടുന്ന വെബ് പേജ് നമുക്ക് നിർമിക്കാം. ഇതിനുള്ള HTML കോഡ് ഉദാഹരണം 5.6 ലും അതിനനുസൃതമായ വെബ് പേജ് ചിത്രം 5.8 ലും നൽകിയിരിക്കുന്നു.

ഉദാഹരണം 5.6: ഇന്റേണൽ ലിങ്ക് ഉൾപ്പെടുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE> Internal Linking </TITLE> </HEAD>
<BODY Bgcolor= "f8f8f8">
   <H2 Align= "center">ENVIRONMENTAL POLLUTION</H2>
   <A Name= "Introduction"><B>INTRODUCTION</B></A>
   <P><FONT Size= "15">E</FONT>nvironment pollution is a
   wide-reaching problem and it is likely to affect the
   health of human population.Here we discuss the environment
   pollution in the perspective of <A HREF= "#Air">air
   pollution </A>,
   <A Href= "#Water"> water pollution </A>and land/soil
   waste pollution. Studies find that these kinds of
   pollutions are not only seriously affecting humans
   but also animals and plants.
   </P>
   <A Name= "Air"><B> Air Pollution</B></A>
   <P>The air we breathe is an essential ingredient for our
   health and wellbeing. Unfortunately polluted air is common
   throughout the world, especially in developed countries.
   </P>
   <A Name= "Water"><B> Water pollution</B></A>
   <P>The water we drink is an essential ingredient for our
   health and wellbeing. Unfortunately polluted water and
   air are common throughout the world.Water pollution is
   caused by the discharge of industrial effluents, sewage
   water and agricultural or household waste.
   </P>
   <A Href= "#Introduction">Go to Introduction </A>
</BODY>
</HTML>
```

ചിത്രം 5.8 ൽ ഈ വെബ് പേജിന്റെ ഒരു ഭാഗം കാണിച്ചിരിക്കുന്നു. ഉദാഹരണം 5.8 ൽ കാണിച്ചിരിക്കുന്ന air pollution, water pollution എന്നീ ലിങ്കുകളിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ

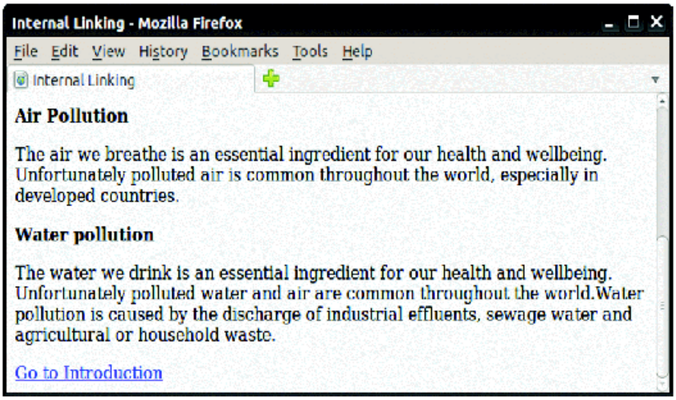


ചിത്രം 5.8: ഇന്റേണൽ ഹൈപ്പർ ലിങ്കുകൾ ഉൾപ്പെടുന്ന വെബ് പേജ്

ചിത്രം 5.9 ൽ കാണിച്ചിരിക്കുന്ന വെബ് പേജിന്റെ അനുയോജ്യമായ ഭാഗം ദൃശ്യമാകും. ഇതുപോലെ ചിത്രം 5.9 ലെ വെബ് പേജിന്റെ താഴെ ഭാഗത്തുള്ള `Go to Introduction` എന്ന ലിങ്കിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ Indroduction എന്ന ഭാഗം ദൃശ്യമാകും.

5.2.2 എക്സ്റ്റേണൽ ലിങ്കിങ് (External linking)

ഒരു വെബ് പേജിനെ മറ്റൊരു വെബ് പേജുമായി ബന്ധിപ്പിക്കുന്ന പ്രവർത്തനത്തെ എക്സ്റ്റേണൽ ലിങ്കിങ് എന്ന് പറയുന്നു. നിലവിൽ ലഭ്യമായ വെബ് പേജിലെ **<A>** ടാഗിന്റെ Href ആട്രിബ്യൂട്ടിന്റെ വിലയായി ബാഹ്യ ഫയലിന്റെ URL നൽകി ഇതു സാധ്യമാക്കാം. ഇതിന്റെ പ്രവർത്തനങ്ങൾ നാം ഈ അധ്യായത്തിലെ ഭാഗം 5.2-ൽ ചർച്ച ചെയ്തതാണ്.



ചിത്രം 5.9: വെബ് പേജിന്റെ ഹൈപ്പർലിങ്ക് ചെയ്യപ്പെട്ട ഭാഗങ്ങൾ

5.2.3 URL എന്ന ആശയം (Concept of URL)

URL-ന്റെ പൂർണ്ണരൂപം Uniform Resource Locator (യൂണിഫോം റിസോഴ്സ് ലൊക്കേറ്റർ) എന്നാണ്. വെബ് അഡ്രസ് എന്നാണിതിന്റെ അർഥം. റിലേറ്റീവ്, അബ്സല്യൂട്ട് എന്നീ രണ്ടു തരം URL കൾ നിലവിലുണ്ട്.

`<A Href= "http://www.scertkerala.gov.in">` എന്നത് ഒരു അബ്സല്യൂട്ട് URL ആണ്. കാരണം ഇവിടെ വെബ് അഡ്രസ് പൂർണമായും നൽകിയിരിക്കുന്നു. എന്നാൽ ലിങ്ക് `<A Href = "image.html">` എന്നാണ് എഴുതിയതെങ്കിൽ അത് റിലേറ്റീവ് ലിങ്ക് ആയി പരിഗണിക്കും. ഇവിടെ നമ്മൾ നൽകിയ "*image.html*" ഒരു റിലേറ്റിവ് URL ആണ്. എന്തുകൊണ്ടെന്നാൽ *image.html* ഉം വെബ് പേജും ഒരേ ഡയറക്ടറിയിൽ ശേഖരിച്ചിരിക്കുന്നു. നിലവിൽ ലഭ്യമായ വെബ് പേജ് */home/kite/HTML/hyperlink.html* എന്നാണ് ശേഖരിച്ചിരിക്കുന്നതെങ്കിൽ `<A Href = "image.html">` എന്ന ഹൈപ്പർ ലിങ്ക് സൂചിപ്പിക്കുന്നത് */home/kite/HTML* എന്ന ഫോൾഡറിലെ *image.html* എന്ന ഫയലിനെ ആയിരിക്കും.

5.2.4 ഗ്രാഫിക്കൽ ഹൈപ്പർ ലിങ്ക് നിർമാണം (Creating graphical hyperlinks)

ടെക്സ്റ്റുകളെ ഹൈപ്പർ ലിങ്കായി ഉപയോഗിക്കുന്ന രീതി നാം മനസിലാക്കിക്കഴിഞ്ഞു. അതുപോലെ ഇമേജുകളും ഹൈപ്പർ ലിങ്കാക്കി ഉപയോഗിക്കാം. അതിനായി **<A>**, **</A>** എന്നീ ടാഗുകളുടെ ഇടയിൽ **<IMG>** ടാഗ് നൽകിയാൽ മതിയാകും. അതിന്റെ HTML കോഡ് ഉദാഹരണം 5.7ൽ കാണിച്ചിരിക്കുന്നു. ഇതു സംബന്ധിച്ച വിവരങ്ങൾ ചിത്രം 5.10 ലും നൽകിയിരിക്കുന്നു.

ഉദാഹരണം 5.7: ഗ്രാഫിക്കൽ ഹൈപ്പർ ലിങ്ക് ഉൾക്കൊള്ളുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE> Graphical Hyperlink </TITLE> </HEAD>
<BODY Bgcolor = "#E0FFFF">
   <H2 Align= "center">Graphical Hyperlink</H2>
   Here is the image with <I>Graphical hyperlink </I>
   <A Href= "https://www.wikipedia.org">
   <IMG Src= "wiki.jpg" Alt= "Image of Wiki"
        Height= "30" Width= "40" Border= "1"> </A>.
        We can click over this image and the home page of
        linked site, wikipedia.org will open.
</BODY>
</HTML>
```

നിങ്ങൾ മൗസ് പോയിന്റർ വിക്കീപീഡിയ ലോഗോയുടെ മുകളിൽ കൊണ്ടുവരുമ്പോൾ അത് ചൂണ്ടു വിരൽ ഉയർത്തിയ കൈ അടയാളമായി മാറുന്നു. ഇത് സൂചിപ്പിക്കുന്നത് ഈ ലോഗോ ഒരു ഗ്രാഫിക്കൽ ഹൈപ്പർ ലിങ്ക് ആണെന്നാണ്. അവിടെ ക്ലിക്ക് ചെയ്യുമ്പോൾ ബ്രൗസറിൽ www.wikipedia.org എന്ന വെബ്സൈറ്റ് ലഭ്യമാകും.

5.2.5 ഇ-മെയിൽ ലിങ്ക് നിർമിക്കുന്ന വിധം (Creating e-mail linking)

ഹൈപ്പർലിങ്ക് പ്രോട്ടോക്കോളായ **mailto** ഉപയോഗിച്ച് നമുക്ക് ഇ-മെയിൽ ഹൈപ്പർലിങ്ക് നിർമിക്കുവാനാകും. ഇത് വിശദമാക്കുന്ന HTML കോഡ് ഉദാഹരണം 5.8 ലും വെബ് പേജിന്റെ ചിത്രം 5.11 ലും നൽകിയിട്ടുണ്ട്.

ചിത്രം 5.10: ഗ്രാഫിക്കൽ ഹൈപ്പർ ലിങ്ക് ഉൾക്കൊള്ളുന്ന വെബ് പേജ്

ഉദാഹരണം 5.8: ഇ-മെയിൽ ലിങ്ക് ഉൾക്കൊള്ളുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE> e-mail Linking </TITLE> </HEAD >
<BODY Bgcolor= "#E0FFFF">
   <H2 Align= "center">e-mail linking</H2>
   Now we can create an <B><I>e-mail hyperlink </I></B> to
   SCERT in the following way. Kindly click on the word
   <A Href= mailto: "scertkerala@gmail.com"> SCERT</A> Kerala.
</BODY>
</HTML>
```

ചിത്രം 5.11- ൽ കാണിച്ചിരിക്കുന്ന വെബ് പേജിൽ SCERT എന്ന ലിങ്ക് നൽകിയിട്ടുണ്ട്. ഇതിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ scertkerala@gmail.com എന്ന് അഡ്രസ് ചെയ്യപ്പെട്ട ഒഴിഞ്ഞ മെസ്സേജ്ബോക്സോടു കൂടിയ ഇ-മെയിൽ പ്രോഗ്രാം തുറക്കുന്നു.

ചിത്രം 5.11: ഇ-മെയിൽ ഹൈപ്പർ ലിങ്ക് ഉൾക്കൊള്ളുന്ന വെബ് പേജ്

ഒരു വെബ് പേജിൽ നിന്ന് ഇ-മെയിൽ ലിങ്ക് നൽകി ഇ-മെയിൽ പ്രോഗ്രാം തുറക്കുമ്പോൾ, ഇ-മെയിൽ പ്രോഗ്രാമിലെ എല്ലാ ഭാഗങ്ങൾക്കും തനതു വിലകൾ നൽകാവുന്നതാണ്. ഉദാഹരണമായി "Thank you sir" എന്ന സന്ദേശം "Director SCERT" എന്ന സബ്ജക്ടോടുകൂടി scertkerala@gmail.com എന്ന വിലാസത്തിലേക്കും, കാർബൺ കോപ്പിയായി (CC) scertkerala1@gmail.com എന്ന വിലാസത്തിലേക്കും, bcc ആയി scertkerala2@gmail.com എന്ന വിലാസത്തിലേക്കും ഒരു ഇ-മെയിൽ ലിങ്ക് വഴി അയക്കണമെങ്കിൽ മുകളിൽ കൊടുത്തിരിക്കുന്ന HTML കോഡിലെ <A> ടാഗ് താഴെ പറയുന്ന വിധം മാറ്റി എഴുതിയാൽ മതിയാകും.

```
<A Href="mailto:scertkerala@gmail.com?
cc=scertkerala1@gmail.com&
bcc=scertkerala2@gmail.com
&samp;subject=Director%20SCERT
&samp;bodt=Thank%20You%20Sir">
send E-mail with cc bcc subject and body </A>
```

<A> ടാഗിലെ ? ചിഹ്നം ആദ്യത്തെ വിഭാഗത്തെയും & മറ്റ് വിഭാഗങ്ങളെയും വേർതിരിക്കാൻ ഉപയോഗിക്കുന്നു.

5.3 സംഗീതം, വീഡിയോ എന്നിവ ഉൾപ്പെടുത്തൽ (Inserting music and video)

ഇപ്പോൾ ലഭ്യമാകുന്ന മിക്കവാറും വെബ് പേജുകളിലും ടെക്സ്റ്റുകൾ, ചിത്രങ്ങൾ എന്നിവയ്ക്ക് പുറമെ സംഗീത, വീഡിയോ ശകലങ്ങളും ഉൾപ്പെടുത്തി സമ്പുഷ്ടമാക്കിയിരിക്കുന്നു. എങ്ങനെയാണ് ഒരു വെബ് പേജിൽ മ്യൂസിക്, വീഡിയോ തുടങ്ങിയവ ഉൾപ്പെടുത്തുന്നത്? മൾട്ടിമീഡിയ വിഭവങ്ങൾ ഉൾപ്പെടുത്താൻ ഇൻലൈൻ, എക്സ്റ്റേണൽ എന്നീ രണ്ടു തരം രീതികൾ ഉപയോഗിക്കുന്നു. ഇൻലൈൻ രീതിയിൽ ഫയലുകളും അതിനാവശ്യമായ വിവരങ്ങളും വെബ് പേജിന്റെ ഭാഗമായാണ് കൈകാര്യം ചെയ്യുന്നത്. ബ്രൗസർ ജാലകം തുറക്കുമ്പോൾത്തന്നെ ഇതിലെ സംഗീതവും വീഡിയോ ദൃശ്യവും ഉപയോക്താവിന് കേൾക്കുവാനും കാണുവാനും സാധിക്കും. എന്നാൽ വെബ് പേജിൽ ലഭ്യമാകുന്ന മൾട്ടിമീഡിയ വിഭവങ്ങൾ ആ കമ്പ്യൂട്ടറിൽ ലഭ്യമായ സോഫ്റ്റ്‌വെയർ ഉപയോഗിച്ച് തുറക്കുന്ന രീതിയാണ് എക്സ്റ്റേണൽ രൂപത്തിൽ അവലംബിച്ചിരിക്കുന്നത്. നമുക്ക് .jpg,

.gif, .avi, .png, .tiff, .mp3, .mp4, തുടങ്ങിയ എക്സ്റ്റൻഷനിൽ കാണപ്പെടുന്ന മൾട്ടിമീഡിയ ഫയലുകൾ ഒരു വെബ് പേജിലേക്ക് ലിങ്ക് ചെയ്യുവാനാകും.

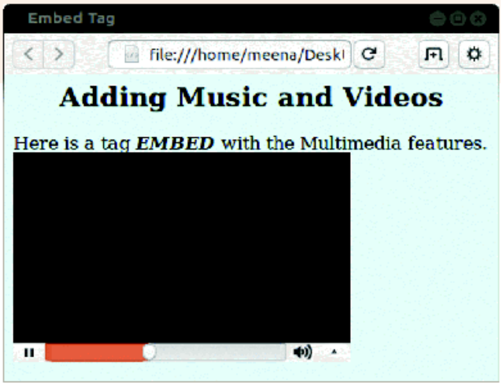
മൾട്ടിമീഡിയ വിഭവങ്ങളായ സംഗീതവും വീഡിയോ ദൃശ്യങ്ങളും ഉൾപ്പെടുത്താൻ HTML ലെ **<EMBED>** എന്ന ടാഗാണ് ഉപയോഗിക്കുന്നത്. വെബ് പേജിൽ ഈ ടാഗ് ഉൾപ്പെടുത്തുമ്പോൾ മൾട്ടിമീഡിയ മാർഗങ്ങളുടെ നിയന്ത്രണത്തിനുവേണ്ട ഉപാധികളും ലഭ്യമാകുന്നു. എന്നാൽ ഈ ടാഗിന്റെ പ്രവർത്തനം പിന്തുണയ്ക്കാത്ത ബ്രൗസറുകളിൽ **<NOEMBED>** എന്ന ടാഗ് ഉപയോഗിച്ച് ബദൽ സംവിധാനം സജ്ജമാക്കാവുന്നതാണ്.

<EMBED> ടാഗിന്റെ പ്രധാന ആട്രിബ്യൂട്ട് **Src** ആണ്. ഇതിന്റെ വിലയായി വെബ് പേജിൽ ഉൾപ്പെടുത്തേണ്ട സംഗീതം/വീഡിയോയുടെ URL നൽകണം. മറ്റ് ആട്രിബ്യൂട്ടുകൾ **Height, Width, Align, Alt,** തുടങ്ങിയവ ആണ്. ഇതിന്റെ വിലകൾ നമുക്ക് പരിചിതമാണല്ലോ? ഇതുകൂടാതെ **Hidden** എന്നൊരു ആട്രിബ്യൂട്ടും ഈ ടാഗിനുണ്ട്. നമ്മൾ വെബ് പേജുകളിൽ ഉൾപ്പെടുത്തുന്ന സംഗീത-വീഡിയോ ശകലങ്ങൾ ഉപയോക്താവിന് കാണണമെങ്കിൽ **Hidden** ആട്രിബ്യൂട്ടിന് തനത് വിലയായ False എന്നും കാണേണ്ട എങ്കിൽ True എന്നും നൽകണം.

ഒരു വെബ് പേജിൽ ഒരു ഓഡിയോ ഉൾപ്പെടുത്തിയതെങ്ങനെയെന്ന് ഉദാഹരണം 5.9 ലും വെബ് പേജിന്റെ ചിത്രം 5.12 ലും നൽകുന്നു.

ഉദാഹരണം 5.9: ഓഡിയോ ലിങ്ക് ഉൾക്കൊള്ളുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD>
     <TITLE> Embed Tag </TITLE>
</HEAD>
<BODY Bgcolor = "#DDFFFF">
   <H2 Align= "center"> Adding Music and Videos </H2>
   Here is a tag <B><I>EMBED </I></B> with the Multimedia
   features.<BR>
   <EMBED Src= "song1.mp3" Width= "300" Height= "60">
   </EMBED>
</BODY>
</HTML>
```



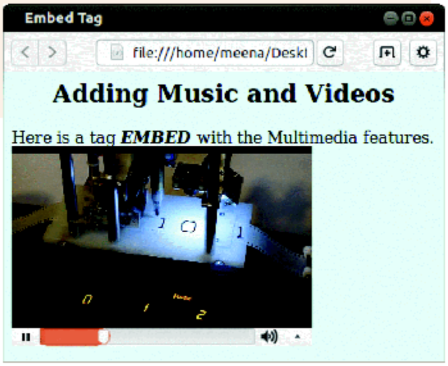
ചിത്രം 5.12: ഓഡിയോ ലിങ്ക് ഉൾക്കൊള്ളുന്ന വെബ് പേജ്

ഉദാഹരണം 5.9 ൽ കൊടുത്തിരിക്കുന്ന HTML പേജ് ബ്രൗസർ വഴി തുറക്കുമ്പോൾ അതിൽ ഉള്ളടക്കം ചെയ്യപ്പെട്ട സംഗീതം കേൾക്കാൻ സാധിക്കുന്നു. ഇതിന്റെ നിയന്ത്രണങ്ങളായ പ്ലേ, പോസ്, ശബ്ദ വ്യതിയാനം എന്നിവയും വെബ് പേജിൽ ലഭ്യമാകും. ഇതുപോലെ വീഡിയോ വെബ് പേജിൽ ലഭ്യമാക്കുന്നതിന് <EMBED> ടാഗും അതിന്റെ ആട്രിബ്യൂട്ടായി **Src** യും ഉപയോഗിക്കാം.

ഉദാഹരണം 5.10 ൽ കൊടുത്തിരിക്കുന്ന HTML കോഡിന് അനുസൃതമായ വെബ് പേജ് (ചിത്രം 5.13) ഈ ലിങ്കിങ് ചിത്രീകരിക്കുന്നു.

ഉദാഹരണം 5.10: ഒരു വെബ് പേജിൽ വീഡിയോ ലഭ്യമാക്കുന്ന HTML കോഡ്

```
<HTML>
<HEAD>
   <TITLE> Embed Tag </TITLE>
</HEAD >
<BODY Bgcolor = "#DDFFFF">
   <H2 Align="center">Adding Music and Videos</H2>
   Here is a tag <B><I>EMBED </I></B>with the Multimedia
   features.
   <EMBED Src= "alan.mp4" Width= "300" Height= "150">
   </EMBED>
   <NOEMBED><IMG Src= "book2.jpg"
        Alt= "Alternative Media">
   </NOEMBED>
</BODY>
</HTML>
```



ചിത്രം 5.13: വീഡിയോ ഉൾക്കൊള്ളുന്ന വെബ് പേജ്

നമ്മൾ ഉദാഹരണം 5.9 ൽ നൽകിയിരിക്കുന്ന HTML പേജ് തുറക്കുമ്പോൾ ഉള്ളടക്കം ചെയ്ത വീഡിയോ പ്ലേ ചെയ്ത് തുടങ്ങും. ഇവിടെയും ശബ്ദം, വീഡിയോ എന്നിവ നിയന്ത്രിക്കാനാവശ്യമായ വോളിയം, പോസ്, ഫുൾ സ്ക്രീൻ മോഡ് എന്നീ കൺട്രോളുകൾ ലഭ്യമാകും. മറ്റ് വെബ് പേജിലുള്ള ഓഡിയോയും വീഡിയോയും നമ്മുടെ വെബ് പേജിലേക്ക് ലിങ്ക് ചെയ്ത് ഉൾപ്പെടുത്താം.

വെബ് പേജിന്റെ പശ്ചാത്തലത്തിൽ ഓഡിയോ ഇൻലൈൻ ആയി മ്യൂസിക് പ്ലേ ചെയ്യുന്നതിന് വേണ്ടി **<BGSOUND>** എന്ന ടാഗ് ഉപയോഗിക്കുന്നു. ഇതിന്റെ ഉദാഹരണം 5.11 ൽ കൊടുത്തിരിക്കുന്നു.

ഉദാഹരണം 5.11: <BGSOUND> ടാഗിന്റെ ഉപയോഗം വ്യക്തമാക്കുന്ന HTML കോഡ്

```
<HTML>
<HEAD> <TITLE> Background Music </TITLE> </HEAD >
   <BODY Bgcolor = "#DDFFFF">
   <H2 Align= "center">Adding Background Music </H2>
   Here is a tag <B><I> BGSOUND </I></B> which helps
   us to play background music in our web page.
   <BGSOUND Src= "Song2.mp3" Loop= "Infinite">
</BODY>
</HTML>
```

ഓഡിയോയുടെ ശബ്ദനിയന്ത്രണത്തിനായി നമുക്ക് <BGSOUND> ടാഗിന്റെ Volume ആട്രിബ്യൂട്ട് ഉപയോഗിക്കാം. HTML പേജിൽ എക്സ്റ്റേണൽ ഫയൽ ഉൾപ്പെടുത്തുന്നതിനായി Src, Href ആട്രിബ്യൂട്ടുകളിൽ അവയുടെ കൃത്യമായ വഴി നൽകിയിരിക്കണം. ഉദാഹരണം 5.10-ൽ ഈ ആട്രിബ്യൂട്ടുകളുടെ വിലകളിൽ മാറ്റം വരുത്തി പരീക്ഷിച്ചു നോക്കാവുന്നതാണ്. Loop ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് മ്യൂസിക് എത്ര തവണ ആവർത്തിക്കണം എന്നുള്ളത് തീരുമാനിക്കാം. ഇതിന്റെ വിലയായി Infinite നൽകിയാൽ വെബ് പേജ് ബ്രൗസറിൽ ലഭ്യമാകുന്നിടത്തോളം തുടർച്ചയായി മ്യൂസിക് പ്ലേ ചെയ്യപ്പെടുന്നതാണ്.

5.4 വെബ് പേജിൽ പട്ടിക ഉൾപ്പെടുത്തുന്ന വിധം (Creating tables in a web page)

ചിലസമയങ്ങളിൽ ധാരാളം വിവരങ്ങൾ നമുക്ക് ചിട്ടയായ രീതിയിൽ പ്രദർശിപ്പിക്കേണ്ടതായിവരും. ഉദാഹരണത്തിന് ഒരു ആശുപത്രിയിൽ 2012-2014 കാലയളവിൽ പുതിയതായി ക്യാൻസർ കണ്ടെത്തിയ രോഗികളിൽ പുകവലി, പാൻ, മദ്യം, ഇതല്ലാതെ മറ്റുകാരണങ്ങളാൽ ക്യാൻസർ ബാധിച്ചവരുടെ എണ്ണം എന്നിവ പട്ടിക രൂപത്തിൽ പട്ടിക 5.1 ൽ നൽകുന്നു.

ഒരു ആശുപത്രിയിൽ 2012 – 2014 കാലയളവിൽ പുതിയതായി ക്യാൻസർ കണ്ടെത്തിയ രോഗികൾ			
Year	2012	2013	2014
Smokers	129	140	143
Pan users	54	56	49
Alcohol users	74	68	77
Other cases	95	93	92

പട്ടിക 5.1: ഒരു ആശുപത്രിയിൽ ക്യാൻസർ കണ്ടെത്തിയ രോഗികൾ

വരിയും നിരയുമായി സെല്ലുകൾ ചിട്ടയായി അടുക്കിവച്ചിരിക്കുന്നതാണ് പട്ടിക എന്നത് നമുക്കെല്ലാം അറിയാം. ഈ അറകളിൽ (cells) ചിത്രങ്ങളോ വാക്യങ്ങളോ മറ്റൊരു ടേബിളോ ഉൾക്കൊള്ളിക്കാം. HTML-ൽ **<TABLE>** എന്ന ടാഗ് ഉപയോഗിച്ചാണ് പട്ടിക നിർമിക്കുന്നത്. ഈ ടാഗിനോടൊപ്പം **<TR>**, **<TH>**, **<TD>** എന്നീ ടാഗുകളും പട്ടിക നിർമിക്കാൻ ഉപയോഗിക്കുന്നു. ഈ ടാഗുകളെക്കുറിച്ച് തുടർന്നുള്ള ഭാഗങ്ങളിൽ ചർച്ച ചെയ്യാം.

5.4.1 <TABLE> ടാഗ്

പട്ടിക ടാഗ് ഒരു കണ്ടെയിനർ ടാഗാണ്. പട്ടിക നിർമാണത്തിനാവശ്യമായ എല്ലാ ഉള്ളടക്കങ്ങളും ടാഗ് ജോഡികളായ **<TABLE>**, **</TABLE>** എന്നിവയുടെ ഇടയിൽ നൽകണം. പട്ടികയുടെ പൊതുവായ രൂപരേഖ തയാറാക്കുന്നതിന് വിവിധ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിക്കുന്നു. ഇവയിൽ മുഖ്യമായവ താഴെ കൊടുത്തിരിക്കുന്നു.

1. **Border**: പട്ടികയ്ക്ക് ചുറ്റുമുള്ള വരകളുടെ കനമാണ് ഈ ആട്രിബ്യൂട്ടിലൂടെ വ്യക്തമാക്കുന്നത്. ഈ ആട്രിബ്യൂട്ടിന് പൂജ്യമല്ലാത്ത വില (പിക്സലായാണ് നൽകേണ്ടത്) നൽകിയാൽ പട്ടികയുടെ ബോർഡർ നിർമിക്കാം. ഇതിന് പൂജ്യം എന്ന വില നൽകുമ്പോൾ ടേബിളിന് ചുറ്റുമുള്ള വരകൾ അപ്രത്യക്ഷമാകും.
2. **Bordercolor**: പട്ടികയുടെ അതിരുകൾക്ക് നിറം നൽകുന്നു.
3. **Align**: ബ്രൗസർ ജാലകത്തിൽ പട്ടികയുടെ സ്ഥാനം നിർണയിക്കുന്നു. ഇതിന്റെ സാധ്യമായ വിലകൾ left (തനതുവില) right, center എന്നിവയാകുന്നു.
4. **Bgcolor**: പട്ടികക്ക് പശ്ചാത്തലനിറം നൽകുന്നതിന് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു.
5. **Background:** പട്ടികയുടെ പശ്ചാത്തലത്തിൽ ഒരു ചിത്രം പ്രദർശിപ്പിക്കുന്നതിന് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. ഇതിനായി ചിത്രം സൂക്ഷിച്ചിരിക്കുന്ന സ്ഥലം (path) Background ആട്രിബ്യൂട്ടിന്റെ വിലയായി നൽകണം. ഉദാഹരണത്തിന് <TABLE Background = "images/flower.gif"> *images* എന്ന ഫോൾഡറിൽ സൂക്ഷിച്ചിരിക്കുന്ന *flower.jpg* എന്ന ചിത്രം ടേബിളിന്റെ ഉള്ളിൽ പ്രദർശിപ്പിക്കുന്നു. <TABLE > ടാഗിന്റെ ആട്രിബ്യൂട്ടുകളായി Bgcolor, Background എന്നിവ രണ്ടും നൽകിയാൽ Background ആട്രിബ്യൂട്ടിന്റെ വിലയായ ചിത്രമായിരിക്കും പ്രദർശിപ്പിക്കുക.
6. **Cellspacing**: പട്ടികയിലെ അറകൾക്ക് (Cell) ഇടയിൽ സ്ഥലം ഉണ്ട്. ഈ സ്ഥലം കൂട്ടുവാനോ കുറയ്ക്കുവാനോ കഴിയും. Cellspacing എന്ന ആട്രിബ്യൂട്ട് അറകൾക്ക് ഇടയിലുള്ള സ്ഥലം നിർണയിക്കുന്നു. ഇതിന്റെ വിലയായി പിക്സലുകളുടെ എണ്ണമാണ് നൽകുന്നത്.
7. **Cellpadding**: പട്ടികക്കുള്ളിലെ അറകളിൽ അടങ്ങിയിരിക്കുന്ന വസ്തുവും അറകളുടെ അതിർത്തിയും തമ്മിലുള്ള സ്ഥാനവ്യത്യാസം നിർണയിക്കുന്നതിനായി ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. പിക്സലിലാണ് ഇതിന്റെ വിലയും നൽകേണ്ടത്.
8. **Width** and **Height**: പട്ടികയുടെ വീതിയും ഉയരവും ഈ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിച്ച് നൽകാം. ഇതിന്റെ വിലകൾ പിക്സലിലോ ബ്രൗസർ ജാലകത്തിന്റെ വീതിയുടെയും ഉയരത്തിന്റെയും ശതമാനമായോ നൽകാം.
9. **Frame**: പട്ടികയുടെ അതിരുകൾ (Border) ഏത് രീതിയിൽ പ്രദർശിപ്പിക്കണം എന്നതിനായി ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കാം. ഈ ആട്രിബ്യൂട്ടിന്റെ വിലകൾ പട്ടികാരൂപത്തിൽ പട്ടിക 5.2 ൽ നൽകുന്നു.

വില	വിവരണം
Void	പട്ടികയുടെ അതിരുകൾ പ്രദർശിപ്പിക്കാതിരിക്കാൻ.
Above	മുകൾഭാഗത്തെ മാത്രം അതിര് പ്രദർശിപ്പിക്കുന്നതിന്.
Below	പട്ടികയുടെ താഴ്ഭാഗത്ത് മാത്രം അതിര് പ്രദർശിപ്പിക്കുന്നതിന്.

Value	Description
Hsides	പട്ടികക്ക് തിരശ്ചീനമായി (മുകളിലും താഴെയും) മാത്രം ബോർഡർ പ്രദർശിപ്പിക്കുന്നതിന്.
lhs or rhs	പട്ടികയുടെ ഇടത് അല്ലെങ്കിൽ വലത് ഭാഗത്ത് മാത്രം ബോർഡർ പ്രദർശിപ്പിക്കുന്നതിന്.
Vsides	പട്ടികക്ക് ലംബമായി ഇടതും വലതും ഭാഗത്ത് മാത്രം ബോർഡർ പ്രദർശിപ്പിക്കുന്നതിന്.
box or border	പട്ടികയുടെ എല്ലാ ഭാഗത്തും ബോർഡർ പ്രദർശിപ്പിക്കുന്നതിന് (തനതു വില)

പട്ടിക 5.2: Frame ആട്രിബ്യൂട്ടിന്റെ വിലകൾ

10. **Rules:** അറകളുടെ ചുറ്റുമുള്ള വരകൾ വരയ്ക്കുന്നതിനായി ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. ഇതിന്റെ വിലകൾ പട്ടിക 5.3-ൽ കാണിച്ചിരിക്കുന്നു.

വില	വിവരണം
none	അറകൾക്ക് (Cell) ചുറ്റും വരകൾ പ്രദർശിപ്പിക്കുന്നില്ല. (border)
cols	നിരകൾക്ക് (Columns) മാത്രം വരകൾ പ്രദർശിപ്പിക്കുന്നതിന്
rows	വരികൾക്ക് (rows) മാത്രം വരകൾ പ്രദർശിപ്പിക്കുന്നതിന്
groups	വരികളുടെ കൂട്ടങ്ങൾക്കും നിരകളുടെ കൂട്ടങ്ങൾക്കും വരകൾ പ്രദർശിപ്പിക്കുന്നതിന്
all	എല്ലാ വരികൾക്കും നിരകൾക്കും ഇടയിൽ വരകൾ പ്രദർശിപ്പിക്കുന്നതിന്

പട്ടിക 5.3:Rules ആട്രിബ്യൂട്ടിന്റെ വിലകൾ

ഇനി <TABLE> ടാഗുമായി ബന്ധപ്പെട്ട മറ്റു ടാഗുകളെക്കുറിച്ച് ചർച്ച ചെയ്യാം.

5.4.2 <TR> ടാഗ്

ഒരു പട്ടിക ടാഗിനുള്ളിൽ വരികൾ (rows) നിർമിക്കുന്നതിനായി <TR> ടാഗ് ഉപയോഗിക്കുന്നു. ഇത് ഒരു കണ്ടെയ്നർ ടാഗ് ആണ്. ഓരോ വരിയും <TR> </TR> ടാഗ് ജോഡികൾക്കിടയിൽ ഉൾപ്പെടുത്തുന്നു. <TR> ടാഗ് എല്ലായ്പ്പോഴും <TABLE> ടാഗിനുള്ളിൽ മാത്രമേ ഉപയോഗിക്കാൻ കഴിയൂ. ഓരോ വരിയിലും ഒരു കൂട്ടം അറകൾ (Cells) അടങ്ങിയിരിക്കുന്നു. ഒരു പട്ടികയിലെ ഏറ്റവും ചെറിയഘടകമാണ് ഒരു അറ (Cell). പട്ടികക്കുള്ളിലെ അറകളെ (Cells) ഹെഡിങ്, ഡാറ്റ എന്നിങ്ങനെ രണ്ടായി തരം തിരിക്കാം. പട്ടിക 5.1 ൽ ചുവന്ന നിറത്തിൽ കാണിച്ചിരിക്കുന്ന വിലകൾ ഹെഡിങ് സെല്ലുകളും, നീല നിറത്തിൽ കാണിച്ചിരിക്കുന്നത് ഡാറ്റാസെല്ലുകളും ആണ്.

5.4.3 <TH> ടാഗ്

ഹെഡിങ് സെല്ലുകളെ (Heading cells) നിർവചിക്കുന്നതിനാണ് **<TH>** ടാഗ് ഉപയോഗിക്കുന്നത്. ഇതും ഒരു കണ്ടയ്നർ ടാഗ് ആണ്. **<TH> </TH>** എന്നീ ടാഗുകൾക്കിടയിലാണ് ഹെഡിങ് നൽകുന്നത്. ഈ ഹെഡിങ് കനം കൂട്ടി (Bold face) സെല്ലിന്റെ മധ്യത്തിൽ (Center) പ്രദർശിപ്പിക്കുന്നു. എല്ലായ്പ്പോഴും <TR> ടാഗിനുള്ളിലാണ് <TH> ടാഗ് നൽകുന്നത്.

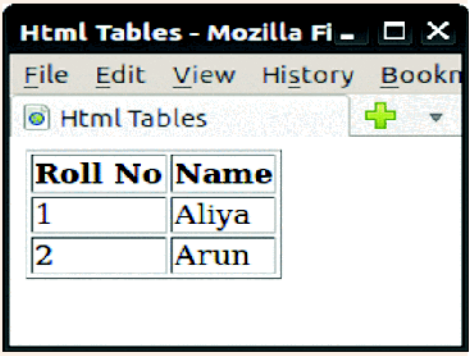
5.4.4 <TD> ടാഗ്

<TH> ടാഗിന് ഏറെക്കുറെ സമാനമായ **<TD>** ടാഗ് ഡാറ്റാസെല്ലുകൾ പ്രദർശിപ്പിക്കുന്നതിന് ഉപയോഗിക്കുന്നു. ഇത് ഒരു കണ്ടയ്നർ ടാഗ് ആണ്. <TD> </TD> ടാഗ്ജോഡികൾക്ക് ഇടയിൽ പ്രദർശിപ്പിക്കേണ്ട ഡാറ്റ നൽകുന്നു. ഈ ടാഗും <TH> ടാഗിനെപോലെ <TR> ടാഗിന് ഉള്ളിലായാണ് നൽകുന്നത്.

ഉദാഹരണം 5.12 നൽകിയിരിക്കുന്ന HTML Code ഒരു പട്ടിക നിർമിക്കാൻ ഉപയോഗിക്കുന്നു. ഇതിന്റെ വെബ് പേജ് ചിത്രം 5.14-ൽ കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 5.12: ഒരു സാധാരണ പട്ടിക ഉൾക്കൊള്ളുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE> Html Tables </TITLE>
</HEAD>
<BODY>
   <TABLE Border="1">
      <TR>
         <TH>Roll No</TH>
         <TH>Name</TH>
      </TR>
      <TR>
         <TD>1</TD>
         <TD>Aliya</TD>
      </TR>
      <TR>
         <TD>2</TD>
         <TD>Arun</TD>
      </TR>
   </TABLE>
</BODY>
</HTML>
```



ചിത്രം 5.14: രണ്ട് നിരകൾ ഉള്ള ഒരു സാധാരണ പട്ടിക ഉൾക്കൊള്ളുന്ന വെബ് പേജ്

Year	2012 - 14
Smokers	412
Pan users	159
Alcohol users	219
Other cases	280

മുകളിൽ പ്രതിപാദിച്ച ഏതാനും ചില ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിച്ച് നമുക്ക് താഴെ പറയുന്ന പട്ടിക നിർമിക്കാം.

ഉദാഹരണം 5.13 - ൽ നൽകിയിരിക്കുന്ന HTML Code ഉപയോഗിച്ച് ചിത്രം 5.15 കാണുന്നതുപോലുള്ള വെബ് പേജ് നിർമിക്കാം.

ഉദാഹരണം 5.13: പട്ടികക്ക് ചുറ്റും ബോർഡറും നിറങ്ങൾ നൽകിയതുമായ പട്ടിക ഉൾക്കൊള്ളുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE> Hospital Table </TITLE> </HEAD>
<BODY>
   <TABLE Border= "3" Bordercolor= "RED" Bgcolor= "#4EB0AF"
          Align= "left" Cellspacing= "16" Cellpadding= "5"
          Width= "50%">
       <TR>
          <TH> Year </TH>
          <TH> 2012-14 </TH>
       </TR>
       <TR>
          <TH> Smokers </TH>
          <TD> 412 </TD>
       </TR>
       <TR>
          <TH> Pan users </TH>
          <TD> 159 </TD>
       </TR>
       <TR>
          <TH> Alcohol users </TH>
          <TD> 219 </TD>
       </TR>
       <TR>
          <TH> Other cases </TH>
          <TD> 280 </TD>
       </TR>
   <TABLE>
<BODY>
<HTML>
```

ചിത്രം 5.15: Cellspacing ഉം Cellpadding ഉം ഉപയോഗിച്ചുള്ള പട്ടിക

ഉദാഹരണം 5.13-ൽ <TR>, </TR> ടാഗ് ജോഡികൾക്കിടയിൽ <TH> ടാഗ് ഉപയോഗിക്കുമ്പോൾ ചിത്രം 5.15 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ പട്ടികയുടെ ആദ്യത്തെ നിര ഹെഡർ കോളത്തിന് തുല്യമായിരിക്കും.

<TR> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകൾ

ഒരു വരിയുടെ സ്വഭാവസവിശേഷതകൾ മാറ്റുന്നതിനായി <TR> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിക്കുന്നു.

1. **Align**: അതാത് വരിയിലെ അറകളിലെ വാക്യങ്ങൾ തിരശ്ചീനമായി സ്ഥാപിക്കുന്നതിന് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. `Left`, `right` or `center` എന്നീ വിലകൾ ഇതിന് നൽകാം. ഈ ആട്രിബ്യൂട്ടിന്റെ തനത് വില ഡാറ്റായെ സംബന്ധിച്ചിടത്തോളം `left` ഉം ഹെഡിങ്ങിന് `center` ഉം ആയിരിക്കും (ചിത്രം 5.15 കാണുക).
2. **Valign**: ഏതെങ്കിലും ഒരു വരിയിലെ (row) അറയിലെ ഉള്ളടക്കം ലംബമായി സ്ഥാപിക്കുന്നതിന് `Valign` ആട്രിബ്യൂട്ട് ഉപയോഗിക്കാം. ഇതിന്റെ വിലയായി `top`, `middle`, `bottom` or `baseline` എന്നിവ നൽകാം. `Baseline` ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുമ്പോൾ അറകളിലെ വാക്യങ്ങളെയും ഉള്ളടക്കത്തെയും അതിന്റെ ബേസ് ലൈനോടുകൂടി സ്ഥാപിക്കുന്നു.
3. **Bgcolor:** ഒരു പ്രത്യേക വരിക്ക് പശ്ചാത്തലനിറം നൽകുന്നതിനായി `Bgcolor` ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. ഒരു വരിയെ പ്രത്യേകമായി എടുത്ത് കാണിക്കുന്നതിന് ഇത് സഹായിക്കുന്നു.

ഉദാഹരണം 5.13-ൽ നൽകിയ HTML കോഡിന്റെ പരിഷ്കരിച്ച കോഡ് ഭാഗം ചുവടെ ചേർക്കുന്നു. ഇത് ചിത്രം 5.16 ൽ കാണുന്നതു പോലെ പട്ടികയിലെ മൂന്നാമത്തെ വരിയുടെ പശ്ചാത്തല നിറത്തിലും തിരശ്ചീനവും ലംബവുമായ വിന്യാസങ്ങളിലും ഭേദഗതി വരുത്തിയിരിക്കുന്നു.

```
<TR Bgcolor= "yellow" Align= "right"
Valign= "middle">
    <TH> Pan users </TH>
    <TD> 159 </TD>
</TR>
```

പരിഷ്കരിച്ച വെബ് പേജ് ചിത്രം 5.16 ൽ കാണിച്ചിരിക്കുന്നു.

ചിത്രം 5.16: Bgcolor, Align, Valign എന്നിവ ഉപയോഗിച്ച് പരിഷ്കരിച്ച വരി ഉൾപ്പെടുന്ന പട്ടിക

<TH>, <TD> ടാഗുകളുടെ ആട്രിബ്യൂട്ടുകൾ

<TH>, <TD> ടാഗുകൾ പട്ടികയിൽ അറകൾ നിർമിക്കുവാൻ ഉപയോഗിക്കുന്നതിനാൽ ഇതിന്റെ മിക്കവാറും ആട്രിബ്യൂട്ടുകൾ ഒരു പോലെയാണ്. ഇതിൽ പ്രധാനപ്പെട്ട ചില ആട്രിബ്യൂട്ടുകൾ നമുക്ക് ചർച്ച ചെയ്യാം.

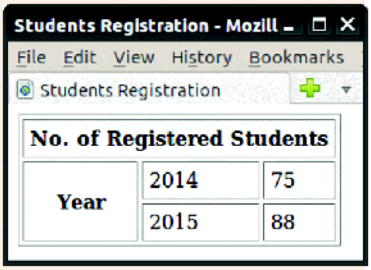
1. **Align**: അറയിലെ ഉള്ളടക്കത്തിന്റെ തിരശ്ചീനമായ വിന്യാസം സൂചിപ്പിക്കുന്നതിന് വേണ്ടിയാണ് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്. ഈ ആട്രിബ്യൂട്ടിന്റെ വിലയായി `left`, `right`, `center` ഇവയിൽ ഏതെങ്കിലും നൽകാം.
2. **Valign**: അറകളിലെ ഉള്ളടക്കത്തിന്റെ ലംബമായ വിന്യാസം സൂചിപ്പിക്കുന്നതിന് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. ഇതിന്റെ വിലയായി `top`, `bottom`, `middle`, `baseline` എന്നിവ നൽകാം.

3. **Bgcolor:** ഏത് അറയ്ക്കും (cell) ഒരു പ്രത്യേക പശ്ചാത്തലനിറം നൽകുന്നതിനാണ് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്. <TABLE>, <TR>, <TD>/<TH> എന്നിവയ്ക്കെല്ലാം Bgcolor ആട്രിബ്യൂട്ടുകൾ നൽകിയാലും <TH>/<TD> ടാഗിന്റെ Bgcolor ആട്രിബ്യൂട്ടിന് നൽകുന്ന വിലയായിരിക്കും അറയുടെ പശ്ചാത്തലനിറമായി പ്രദർശിപ്പിക്കുക.
4. **Colspan:** സാധാരണയായി ഒരു അറ ഒരു നിരയിൽ മാത്രം ഒതുങ്ങുന്നതായിരിക്കും. എന്നാൽ ചില അറകളെ ഒന്നിൽ കൂടുതൽ നിരകളിലേക്ക് ചിലപ്പോൾ വ്യാപിപ്പിക്കേണ്ടതായി വന്നേക്കാം. ഒരു അറ എത്ര നിരകളിലായി വ്യാപിച്ചിരിക്കണം എന്നതനുസരിച്ച് Colspan ആട്രിബ്യൂട്ടിന്റെ വിലനൽകാം. ഉദാഹരണം <TH Colspan="3"> എന്നത് ഒരു അറയെ (Cell) മൂന്ന് നിരകളിലായി വ്യാപിപ്പിക്കുന്നു.
5. **Rowspan:** Colspan-നെ പോലെ തന്നെ എത്ര വരികളിലായി (row) ഒരു അറ (cell) വ്യാപിച്ചിരിക്കുന്നു എന്നതിനാണ് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്. ഉദാഹരണത്തിന് <TD Rowspan="4"> എന്ന് നൽകുമ്പോൾ 4 വരികളിലായി ഒരു അറ വ്യാപിക്കുന്നു.

മുകളിൽ പറഞ്ഞ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിച്ച് ഒരു പട്ടിക നിർമിക്കുന്നതിനുള്ള HTML കോഡ് ഉദാഹരണം 5.14 ൽ നൽകിയിരിക്കുന്നു. വെബ് പേജിന്റെ ചിത്രം 5.17 ൽ നൽകുന്നു.

ഉദാഹരണം 5.14: Colspan, Rowspan എന്നീ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിച്ച് പട്ടിക നിർമിക്കുന്നു

```
<HTML>
<HEAD> <TITLE> Students Registration </TITLE>
</HEAD>
<BODY>
   <TABLE Border= "1" Cellspacing= "3" Cellpadding= "5">
      <TR>
         <TH Colspan= "3"> No. of Registered Students </TH>
      </TR>
      <TR>
         <TH Rowspan= "2"> Year </TH>
         <TD> 2014 </TD> <TD> 75 </TD>
      </TR>
      <TR>
         <TD> 2015 </TD> <TD> 88 </TD>
   </TR>
   </TABLE>
<BODY>
<HTML>
```



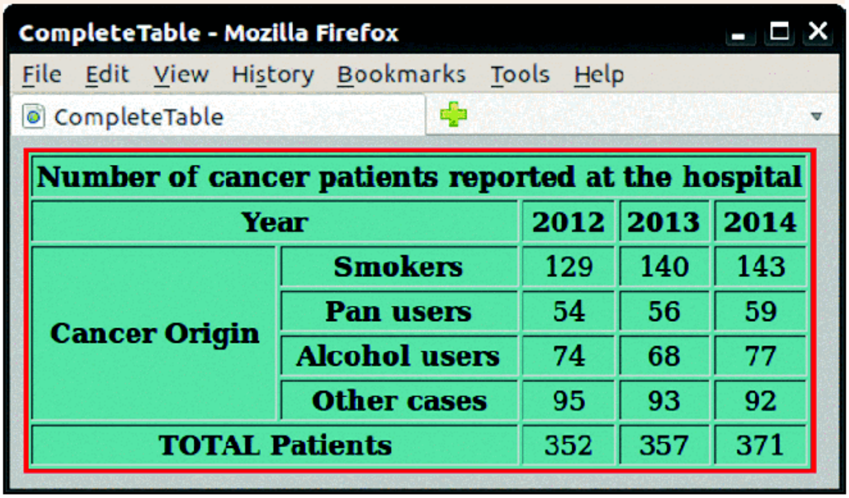
ചിത്രം 5.17: Rowspan, colspan എന്നീ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിച്ച് പട്ടിക നിർമിക്കുന്നു

പട്ടിക 5.1-ൽ കാണിച്ചിരിക്കുന്നതുപോലെ ക്യാൻസർ രോഗികളുടെ വിശദാംശങ്ങൾ ഉൾക്കൊള്ളുന്ന വെബ് പേജ് നിർമിക്കാം. ഇതിന്റെ HTML കോഡ്, ഉദാഹരണം 5.15 ലും ടേബിളിന്റെ ചിത്രം 5.18 ലും നൽകിയിരിക്കുന്നു.

ഉദാഹരണം 5.15: ക്യാൻസർ രോഗികളുടെ വിവരങ്ങൾ അടങ്ങുന്ന വെബ് പേജ്

```
<HTML>
<HEAD> <TITLE> CompleteTable </TITLE> </HEAD>
<BODY Bgcolor= "silver">
   <TABLE Border= "3" Bordercolor= "red" Bgcolor= "#4EB0AF"
         Align= "left" Cellspacing= "2" Cellpadding= "2"
         Width= "50%">
      <TR>
         <TH Colspan= "5"> Number of cancer patients reported
                           at the hospital </TH>
      </TR>
      <TR Align= "center">
         <TH Colspan= "2"> Year </TH>
         <TH> 2012 </TH>
         <TH> 2013 </TH>
         <TH>  2014 </TH>
      </TR>
      <TR Align= "center">
         <TH Rowspan= "4"> Cancer Origin </TH>
         <TH> Smokers </TH>
         <TD> 129 </TD>
         <TD> 140 </TD>
         <TD> 143 </TD>
      </TR>
      <TR Align= "center">
         <TH> Pan users </TH>
         <TD> 54 </TD>
         <TD> 56 </TD>
         <TD> 59 </TD>
      </TR>
      <TR Align= "center">
         <TH> Alcohol users </TH>
         <TD> 74 </TD>
         <TD> 68 </TD>
         <TD> 77 </TD>
      </TR>
      <TR Align= "center">
```

```
          <TH> Other cases </TH>
          <TD> 95 </TD>
          <TD> 93 </TD>
          <TD> 92 </TD>
      </TR>
      <TR Align= "center">
          <TH Colspan= "2"> TOTAL Patients </TH>
          <TD> 352 </TD>
          <TD> 357 </TD>
          <TD> 371 </TD>
      </TR>
  </TABLE>
</BODY>
</HTML>
```



Number of cancer patients reported at the hospital				
Year		2012	2013	2014
Cancer Origin	Smokers	129	140	143
	Pan users	54	56	59
	Alcohol users	74	68	77
	Other cases	95	93	92
TOTAL Patients		352	357	371

ചിത്രം 5.18: Bgcolor, Rowspan, Colspan, Align, Valign എന്നിവ ഉപയോഗിച്ച് ക്യാൻസർ രോഗികളുടെ വിവരങ്ങൾ പട്ടികപ്പെടുത്തിയ വെബ് പേജ്

നമുക്കു ചെയ്യാം

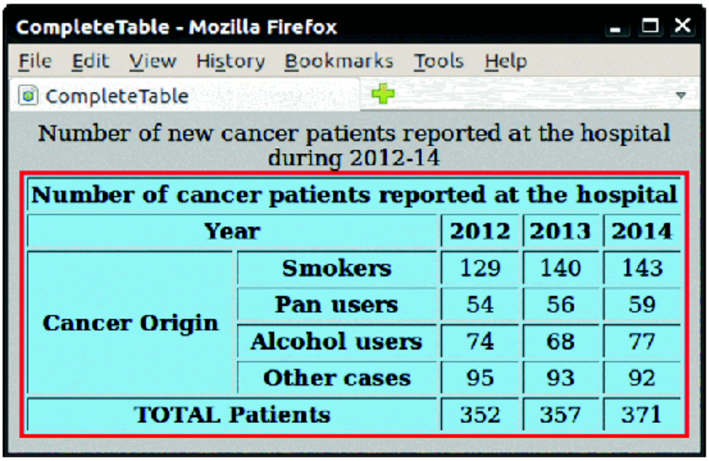
നിങ്ങളുടെ സ്കൂളിലെ വിവിധ രണ്ടാം ഭാഷകൾ പഠിക്കുന്ന കുട്ടികളുടെ എണ്ണം പ്രദർശിപ്പിക്കുന്ന ടേബിൾ നിർമിക്കുക. ഈ ടേബിളിൽ ഹയർ സെക്കന്ററി വിഭാഗത്തിലെ ഓരോ ക്ലാസിലെയും വിവരങ്ങൾ ഉണ്ടായിരിക്കണം.

5.4.5 <CAPTION> ടാഗ് ഉപയോഗിച്ച് ടേബിളിന്റെ ശീർഷകം

<CAPTION> ടാഗ് ഉപയോഗിച്ച് ഒരു പട്ടികയുടെ ശീർഷകം നൽകാം. ഈ ടാഗ് ഉപയോഗിച്ച് ഒരു പട്ടികയെക്കുറിച്ചുള്ള വിവരണം അതിന്റെ ശീർഷകമായി എളുപ്പത്തിൽ നൽകാം. നമുക്ക് ചിത്രം 5.18 ൽ കാണിച്ചിരിക്കുന്ന പട്ടികയ്ക്ക് ഒരു ശീർഷകം നൽകി പരിഷ്കരിക്കണമെന്ന് കരുതുക. ഉദാഹരണം 5.15 ലെ HTML കോഡിലെ **<TR>** ടാഗിന് മുൻപുള്ള ഭാഗത്തിന് താഴെ കൊടുക്കുന്നതുപോലെ മാറ്റം വരുത്താം.

```
<TABLE Border= "3" Bordercolor= "red" Bgcolor= "skyblue"
       Align= "left" Cellspacing= "2" Cellpadding= "2"
       Width= "50%">
<CAPTION>  Number of new cancer patients reported at the
           hospital during 2012-14
</CAPTION>
```

ഇങ്ങനെ ഉണ്ടാക്കിയ പട്ടിക ചിത്രം 5.19 ൽ കാണിച്ചിരിക്കുന്നു.



Number of new cancer patients reported at the hospital during 2012-14

Number of cancer patients reported at the hospital				
Year		2012	2013	2014
Cancer Origin	Smokers	129	140	143
	Pan users	54	56	59
	Alcohol users	74	68	77
	Other cases	95	93	92
TOTAL Patients		352	357	371

ചിത്രം 5.19: ശീർഷകം നൽകി പുതുക്കിയ പട്ടിക

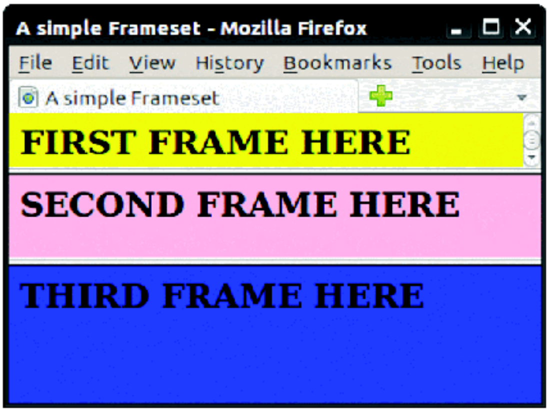
നിങ്ങളുടെ പുരോഗതി അറിയാം

1. <TABLE> ടാഗുമായി ബന്ധപ്പെട്ട രണ്ട് ടാഗുകളുടെ പേരെഴുതുക.
2. താഴെകൊടുത്തിരിക്കുന്നവയിൽ വ്യത്യസ്തമായത് തിരഞ്ഞെടുക്കുക.

   a. TABLE   b. TR   c. TH   d. COLSPAN
3. <TD>, <TH> എന്നീ ടാഗുകളുടെ വ്യത്യാസങ്ങൾ എഴുതുക.
4. ഒരു empty ടാഗാണ് <TABLE> ഈ പ്രസ്താവന ശരിയോ തെറ്റോ എന്ന് എഴുതുക.
5. <TR> ടാഗിന്റെ ഏതെങ്കിലും രണ്ട് ആട്രിബ്യൂട്ടുകൾ എഴുതുക.

5.5 ബ്രൗസർ ജാലകത്തിന്റെ വിഭജനം (Dividing the browser window)

നമുക്ക് ചിലപ്പോൾ ഒരു വെബ് ബ്രൗസർ ജാലകത്തിൽ ഒന്നിൽ കൂടുതൽ വെബ് പേജുകൾ ഉൾപ്പെടുത്തേണ്ടിവരും. ബ്രൗസർ ജാലകത്തെ രണ്ടോ അതിലധികമോ ഭാഗങ്ങളാക്കി തിരിച്ച് ഒരേ സമയം ഓരോ ഭാഗത്തും വ്യത്യസ്തങ്ങളായ വെബ് പേജുകൾ ഉൾക്കൊള്ളിക്കാം. HTML ൽ ബ്രൗസർ ജാലകത്തെ വിവിധ ഭാഗങ്ങളായി വിഭജിക്കുവാൻ **<FRAMESET>** എന്ന ടാഗ് സൗകര്യം നൽകുന്നു. ഇങ്ങനെയുള്ള ഓരോ ഭാഗത്തിലും ഓരോ വെബ് പേജുകൾ ഉൾക്കൊള്ളിക്കാനാവും. ഒരു **<FRAMESET>** ഉപയോ



ചിത്രം 5.20: മൂന്ന് ഫ്രെയിമുകൾ അടങ്ങിയ ഒരു ഫ്രെയിംസെറ്റ്

ഗിച്ച് നിർമിക്കുന്ന ഓരോ വിഭാഗത്തിനേയും ഫ്രെയിം എന്നു വിളിക്കുന്നു. മൂന്ന് ഫ്രെയിമുകൾ ഉൾക്കൊള്ളുന്ന ലളിതമായ ഫ്രെയിം സെറ്റിന്റെ ചിത്രം 5.20-ൽ ചിത്രീകരിച്ചിരിക്കുന്നു. നമുക്ക് **<FRAMESET>**, **<FRAME>** എന്നീ ടാഗുകൾ ഉപയോഗിച്ച് ഒരു ഫ്രെയിം സെറ്റ് നിർമിക്കാം.

5.5.1 <FRAMESET> ടാഗ്

ഒരു കണ്ടയ്നർ ടാഗായ **<FRAMESET>** ബ്രൗസർ ജാലകത്തെ വിവിധ ഫ്രെയിം ഭാഗങ്ങളായി വിഭജിക്കുന്നു. <FRAMESET> </FRAMESET> എന്നീ ടാഗുകളുടെ ഇടയിലാണ് ഫ്രെയിമുകൾ നിർവചിക്കുന്നത്. ഈ ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടുകൾ ചുവടെ ചേർക്കുന്നു.

1. **Cols**: ഈ ആട്രിബ്യൂട്ട് ഫ്രെയിംസെറ്റ് പേജിലെ ലംബമായ ഫ്രെയിമുകളുടെ എണ്ണവും അവയുടെ അളവുകളും നിർണയിക്കുന്നു. ഫ്രെയിമിന്റെ വീതി മൊത്തം വീതിയുടെ ശതമാനത്തിലോ പിക്സലിന്റെ എണ്ണത്തിലോ നൽകാം. ഉദാഹരണമായി <FRAMESET Cols = "30%,500,*"> എന്നത് മൂന്ന് ലംബമായ ഫ്രെയിമുകൾ സൃഷ്ടിക്കുന്നു. ഇതിൽ ആദ്യത്തെ ഫ്രെയിം വിൻഡോയുടെ മൊത്തം വീതിയുടെ 30 ശതമാനവും അടുത്തതിന് 500 പിക്സൽ സ്ഥലവും മൂന്നാമത്തെ ഫ്രെയിമിന്റെ അളവ് * ചിഹ്നത്തിനാൽ രേഖപ്പെടുത്തിയിരിക്കുന്നതിനാൽ ആകെ വീതിയുടെ ബാക്കിവരുന്ന സ്ഥലവും ആയിരിക്കും.
2. **Rows**: ഈ ആട്രിബ്യൂട്ട് Cols നെ പോലെ തിരശ്ചീനമായ ഫ്രെയിമുകളുടെ എണ്ണവും അളവും നിർവചിക്കുന്നു.
3. **Border**: ഈ ആട്രിബ്യൂട്ട് ഫ്രെയിമുകളുടെ അതിരിന്റെ (Border) കനം വ്യക്തമാക്കുന്നു. ഇതിന്റെ വില പിക്സലിന്റെ എണ്ണമായിട്ടാണ് നൽകുന്നത്.
4. **Bordercolor**: ഈ ആട്രിബ്യൂട്ട് ഫ്രെയിമുകളുടെ അതിരിന്റെ (border) നിറം നൽകുവാൻ ഉപയോഗിക്കുന്നു.

5.5.2 <FRAME> ടാഗ്

<FRAMESET> ടാഗിനുള്ളിൽ ഫ്രെയിമുകൾ നിർവചിക്കുന്ന എംപ്റ്റി ടാഗാണിത്. <FRAMESET> ടാഗിന് ഉള്ളിലെ ഓരോ ഭാഗത്തിനും തുല്യമായി **<FRAME>** ടാഗ് നൽകണം. ഫ്രെയിമിനകത്ത് HTML പേജ് പ്രദർശിപ്പിക്കുവാൻ ഉപയോഗിക്കുന്ന **Src** ആട്രിബ്യൂട്ട് എപ്പോഴും <FRAME> ടാഗിനൊപ്പം ഉപയോഗിച്ചിരിക്കണം. <FRAME> ടാഗുമായി ബന്ധപ്പെട്ട പ്രധാന ആട്രിബ്യൂട്ടുകൾ ചുവടെ ചേർക്കുന്നു.

1. **Src**: നമ്മൾ മുമ്പ് ചർച്ചചെയ്തതുപോലെ, ഫ്രെയിമിൽ പ്രദർശിപ്പിക്കേണ്ട വെബ് പേജിന്റെ URL സൂചിപ്പിക്കുന്നതിനാണ് Src ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത്. ഉദാഹരണമായി `<FRAME Src = "school.html">` എന്നത് *school.html* എന്ന പേജിനെ ഫ്രെയിമിൽ പ്രദർശിപ്പിക്കുന്നു.
2. **Scrolling**: ഈ ആട്രിബ്യൂട്ട് ഫ്രെയിമിലെ HTML പേജിൽ ലംബവും തിരശ്ചീനവുമായ സ്ക്രോൾബാറുകൾ പ്രദർശിപ്പിക്കണമോ വേണ്ടയോ എന്ന് തീരുമാനിക്കുന്നു. Scrolling ആട്രിബ്യൂട്ടിന് Yes, No, Auto എന്നീ വിലകൾ നൽകാം. Yes എന്ന വില നൽകുമ്പോൾ സ്ക്രോൾ ബാറുകൾ സ്ഥിരമായി പ്രത്യക്ഷപ്പെടും. No എന്ന വില നൽകുമ്പോൾ സ്ക്രോൾ ബാർ ഒന്നും പ്രത്യക്ഷപ്പെടുന്നില്ല. ഇതിന്റെ തനതുവിലയായ Auto നൽകിയാൽ ഫ്രെയിമിന്റെ ഉള്ളടക്കം അതിന്റെ വലുപ്പത്തെക്കാൾ അധികമാകുമ്പോൾ സ്ക്രോൾബാറുകൾ പ്രത്യക്ഷപ്പെടും.
3. **Noresize**: ഉപയോക്താവ് ഒരു പ്രത്യേക ഫ്രെയിമിന്റെ ബോർഡറുകൾ വലിച്ച് നീട്ടാതിരിക്കാൻ ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. ഈ ആട്രിബ്യൂട്ടിന് വിലകൾ ആവശ്യമില്ല. ഉദാഹരണമായി `<FRAME Src= "school.html" Noresize>`.
4. **Marginwidth and Marginheight**: ഒരു ഫ്രെയിമിലെ ലംബവും തിരശ്ചീനവുമായ മാർജിനുകൾ (margins) പിക്സലിന്റെ (pixels) എണ്ണത്തിൽ നൽകുന്നതിന് യഥാക്രമം Marginwidth, Marginheight എന്നീ ആട്രിബ്യൂട്ടുകൾ ഉപയോഗിക്കുന്നു.
5. **Name**: ഫ്രെയിമിന് പേര് നൽകുന്നതിന് Name ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. തുടർന്ന് വരുന്ന HTML കോഡിൽ ഫ്രെയിമിനെ സൂചിപ്പിക്കുന്നതിനായി ഈ പേര് ഉപയോഗിക്കാം.

ചിത്രം 5.20 ൽ കാണിച്ചിരിക്കുന്ന ഫ്രെയിം സെറ്റ് നിർമിക്കാൻ ചുവടെ ചേർക്കുന്ന HTML കോഡ് ഉപയോഗിക്കാം. മൂന്ന് HTML പേജുകളായ *sampleframe1.html*, *sampleframe2.html*, *sampleframe3.html* എന്നിവ ഇതിന് മുമ്പേ നിർമിച്ചിട്ടുണ്ട് എന്ന് കരുതുക.

```
<HTML>
<HEAD> <TITLE> A simple Frameset </TITLE> </HEAD>
   <FRAMESET Rows= "20%, 30%, 50%">
      <FRAME Src= "sampleframe1.html">
      <FRAME Src= "sampleframe2.html">
      <FRAME Src= "sampleframe3.html">
   </FRAMESET>
</HTML>
```

5.5.3 ഫ്രെയിമിനെ ടാർഗറ്റ് ചെയ്യുന്നു (Targeting frames)

ഒരു ഫ്രെയിംസെറ്റിലുള്ള ഏതെങ്കിലുമൊരു ഫ്രെയിമിലെ പേജിൽ നൽകുന്ന ഹൈപ്പർ ലിങ്കിലൂടെ ബന്ധിപ്പിക്കപ്പെട്ട മറ്റൊരു പേജ് നാം ഉദ്ദേശിക്കുന്ന ഫ്രെയിമിൽ ദൃശ്യമാക്കാൻ സാധിക്കും. നമ്മൾ ഫ്രെയിമിലെ ലിങ്കിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ ബന്ധപ്പെട്ട പേജ് മറ്റൊരു ഫ്രെയിമിൽ തുറക്കും. ഇതിനുവേണ്ടി ആദ്യം നമ്മൾ ഉദ്ദിഷ്ട ഫ്രെയിമിന് Name

ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് പേര് നൽകണം. അതിനുശേഷം ലിങ്കിന്റെ <A> ടാഗിന്റെ Target ആട്രിബ്യൂട്ട് ഉപയോഗിച്ച് പേര് നൽകിയ ഫ്രെയിമിനെ സൂചിപ്പിക്കാം.

നമുക്കു ചെയ്യാം

മൂന്ന് HTML ഫയലുകളായ *bio.html,poem.html,fiction.html* എന്നിവ നിർമിച്ച് അതിൽ ആത്മകഥ, കവിത, നോവൽ എന്നീ ഗണത്തിലുള്ള പുസ്തകങ്ങളുടെ ലിസ്റ്റ് തയാറാക്കുക.

ഇപ്പോൾ രണ്ട് ഫ്രെയിമുകൾ ഉൾപ്പെടുന്ന ഒരു വെബ് പേജ് നിർമിക്കാം. ഇതിൽ ഒന്നാമത്തേത് ഈ ഫയലുകളിലേക്കുള്ള ലിങ്കും രണ്ടാമത്തേത് ലിങ്കുമായി ബന്ധപ്പെട്ട വെബ് പേജ് തുറക്കുന്നതിനും ഉപയോഗിക്കുന്നു. ഒരു ഉപയോക്താവ് ഒന്നാമത്തെ ഫ്രെയിമിൽ സൂചിപ്പിച്ച ഏതെങ്കിലും ഒരു ലിങ്കിൽ ക്ലിക്കു ചെയ്യുമ്പോൾ അതിന് അനുബന്ധമായ ഫയൽ രണ്ടാമത്തെ ഫ്രെയിമിൽ തുറക്കുന്നു.

മുകളിൽ പ്രതിപാദിച്ചിരിക്കുന്ന മൂന്ന് ഫയലുകൾ നിർമിച്ചതിനുശേഷം ഉദാഹരണം 5.16 ൽ കൊടുത്തിരിക്കുന്ന HTML കോഡ് ഉപയോഗിച്ചാൽ Target ആട്രിബ്യൂട്ടിന്റെ പ്രവർത്തനത്തെ കുറിച്ച് മനസിലാക്കാം.

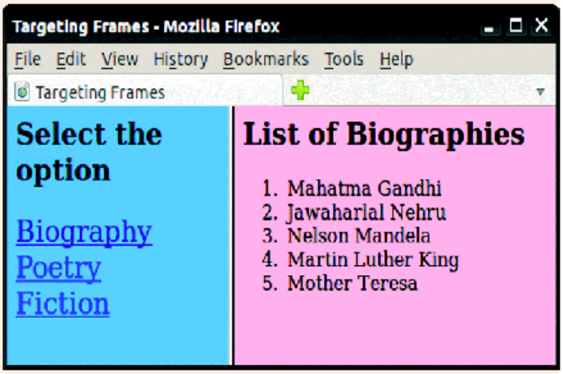
ഉദാഹരണം 5.16: ഫ്രെയിമുകൾ ലക്ഷ്യം വയ്ക്കുന്നതിന്റെ ഉദാഹരണം

താഴെ കൊടുത്തിരിക്കുന്ന കോഡ് ***main.html*** എന്ന ഫയലിൽ സേവ് ചെയ്യുക.

```
<HTML>
<HEAD> <TITLE> Left Frame </TITLE> </HEAD>
<BODY Bgcolor= "#00AFFF" Text= "#282D2F">
    <H2> Select the option </H2>
    <FONT Size= "5">
    <A Href= "bio.html" Target="right_frame">Biography</A><BR>
    <A Href= "poem.html" Target="right_frame">Poetry</A><BR>
    <A Href= "fiction.html" Target="right_frame">Fiction</A>
    </FONT>
</BODY>
</HTML>
```

ഇനി താഴെ കൊടുത്തിരിക്കുന്ന കോഡ് ഒരു ഫയലിൽ സേവ് ചെയ്യാം.

```
<HTML>
<HEAD> <TITLE> Targeting
Frames </TITLE> </HEAD>
<FRAMESET Cols= "200, *">
    <FRAME Src= "main.html"
          Name =
"left_frame">
    <FRAME Name= "right_frame">
</FRAMESET>
</HTML>
```



ചിത്രം 5.21: ഫ്രെയിം ലക്ഷ്യം വയ്ക്കുന്ന വെബ് പേജ്

മുകളിലെ ഡോക്യുമെന്റ് ബ്രൗസറിന് നൽകിയാൽ രണ്ട് ഫ്രെയിമുകൾ ഉൾക്കൊള്ളുന്ന വെബ് പേജ് തുറക്കും. ബ്രൗസർ ജാലകത്തിലെ ആദ്യത്തെ നിരയിൽ *main.html* കാണിക്കുകയും രണ്ടാമത്തെ നിര ഒഴിഞ്ഞുകിടക്കുകയും ചെയ്യും. ആ ഫ്രെയിമിന് വേണ്ടി **Src** ആട്രിബ്യൂട്ട് സൂചിപ്പിച്ചിട്ടില്ല. എന്നാൽ `right_frame` എന്ന പേര് നൽകിയിട്ടുണ്ട് എന്ന കാര്യം ശ്രദ്ധിക്കുക. ഇതിൽ ആദ്യത്തെ ലിങ്കായ Biography തിരഞ്ഞെടുത്താൽ ചിത്രം 5.21 ൽ കാണുന്നതു പോലെ *bio.html* എന്ന ഫയലിന്റെ ഉള്ളടക്കം രണ്ടാമത്തെ ഫ്രെയിമിൽ തുറന്നു വരും. വെബ് പേജിലെ രണ്ടാമത്തെ ഫ്രെയിമിൽ കാണുന്ന *bio.html* എന്ന ഫയലിന്റെ ഉള്ളടക്കം ചിത്രത്തിൽ കാണിച്ചിരിക്കുന്നതുപോലെ ആണെന്ന് കരുതുക. ചിത്രത്തിലെ ലിങ്കുകളുടെ നിറവ്യത്യാസവും ശ്രദ്ധിക്കുക.

main.html എന്ന HTML പേജിൽ മൂന്ന് ജോഡി <A> ടാഗുകൾ മൂന്ന് HTML ഫയലുകളിലേക്ക് ലിങ്ക് ചെയ്യുവാൻ ഉൾപ്പെടുത്തിയിരിക്കുന്നു. ടാഗിന്റെ ആട്രിബ്യൂട്ടായ **Target** ഉപയോഗിച്ച് ലിങ്ക് ചെയ്ത ഫയൽ തുറക്കേണ്ട ഫ്രെയിമിനെ സൂചിപ്പിക്കുന്നു. ഫ്രെയിമിന് പേര് കൊടുക്കേണ്ടതിന്റെ ആവശ്യകത ഇവിടെ സ്പഷ്ടമാകുന്നു.

HTML ന്റെ 5-ാം പതിപ്പ് <FRAMESET> നെ പിന്തുണയ്ക്കുന്നില്ല. ഫ്രെയിംസെറ്റിന്റെ ഉപയോഗത്തിന് എതിരായി ധാരാളം വാദങ്ങൾ ഉയർന്നു വരുന്നു. ഇതിൽ ചിലത് ചുവടെ ചേർക്കുന്നു.

- ഫ്രെയിംസെറ്റിനുള്ളിൽ സാധാരണയായി ബ്രൗസറിന്റെ ബാക്ക്ബട്ടൺ പ്രവർത്തനക്ഷമമല്ല.
- ഒരു ഫ്രെയിം സെറ്റിനുള്ളിൽ ഒരു പ്രത്യേക ഡോക്യുമെന്റ് തുറക്കുവാൻ ബുദ്ധിമുട്ടാണ്.
- ഒരു പ്രത്യേക ഫ്രെയിം റീലോഡ് ചെയ്യാൻ ശ്രമിക്കുമ്പോൾ മൊത്തത്തിലുള്ള ഫ്രെയിം സെറ്റ് റീലോഡ് ചെയ്തേക്കാം. ഇതിന്റെ ഫലമായി ഫ്രെയിമിന്റെ ഉള്ളടക്കങ്ങളെ അതിന്റെ തനതായ ഉറവിടത്തിലേക്ക് റീലോഡ് ചെയ്യും.
- ഫ്രെയിമിലെ ഡോക്യുമെന്റുകളിലൂടെ നാവിഗേറ്റ് ചെയ്യാൻ സർച്ച് എൻജിനുകൾക്ക് ബുദ്ധിമുട്ടാണ്.
- ഒരു ഫ്രെയിംസെറ്റിന്റെ ഉള്ളടക്കം പ്രിന്റ് ചെയ്യുവാൻ ബുദ്ധിമുട്ടാണ്.
- ഒരു ഫ്രെയിം സെറ്റിന്റെ ഉള്ളടക്കം ബുക്ക് മാർക്ക് ചെയ്യുവാൻ ബുദ്ധിമുട്ടാണ്.

5.5.4 ഫ്രെയിംസെറ്റുകളുടെ നെസ്റ്റിങ് (Nesting of framesets)

ചിത്രം 5.20 ലും 5.21 ലും ബ്രൗസർ ജാലകത്തിലെ ഫ്രെയിമുകൾ നമുക്ക് കാണാൻ സാധിക്കും. ആദ്യത്തെ ചിത്രത്തിൽ ബ്രൗസർ ജാലകം തിരശ്ചീനമായ മൂന്ന് ഫ്രെയിമുകളായി തിരിക്കുന്നു. ചിത്രം 5.22-ൽ കാണിച്ചിരിക്കുന്നതുപോലെ ബ്രൗസർ ജാലകത്തെ ലംബമായ രണ്ട് ഫ്രെയിമുകളായി വിഭജിക്കണമെന്ന് കരുതുക. ഇത് കൂട്ടിയിണക്കിയ ഫ്രെയിംസെറ്റുകൾ ഉപയോഗിച്ച് സാധ്യമാകും. ഒരു ഫ്രെയിം

ചിത്രം 5.22: ഫ്രെയിംസെറ്റിന്റെ നെസ്റ്റിങ്

സെറ്റിനുള്ളിൽ മറ്റൊരു ഫ്രെയിം സെറ്റ് കൂട്ടിച്ചേർക്കുന്ന പ്രവർത്തനത്തെ നെസ്റ്റിങ് ഓഫ് ഫ്രെയിംസെറ്റ് എന്നു വിളിക്കുന്നു.

ചിത്രം 5.22-ൽ കാണിച്ചിരിക്കുന്നതുപോലെ ഫ്രെയിമുകൾ കിട്ടാനാവശ്യമായ ഫ്രെയിം സെറ്റ് നിർമിക്കുന്നതിനുള്ള വിവിധ ഘട്ടങ്ങൾ ചുവടെ ചേർക്കുന്നു.

1. രണ്ട് വരികളായി വിഭജിക്കുന്ന ആദ്യത്തെ ഫ്രെയിം സെറ്റ് നിർമിക്കുക.

```
<FRAMESET Rows= "85, *">
    <FRAME Src= "sampleframe1.html">
</FRAMESET>
```

ബ്രൗസർ ജാലകത്തെ തിരശ്ചീനമായ രണ്ട് ഫ്രെയിമുകളാക്കി വിഭജിക്കുകയും ആദ്യത്തെ വരി *sampleframe1.html* എന്നതിന് മാറ്റി വയ്ക്കുകയും ചെയ്യുക.

2. രണ്ടാമത്തെ വരി സ്വതന്ത്രമായി ഇരിക്കുകയാണല്ലോ? ഇപ്പോൾ നമ്മൾ ഇതിനെ ലംബമായ രണ്ട് ഫ്രെയിമുകളാക്കി വിഭജിക്കുന്നു. അതിനായി രണ്ടാമത്തെ <FRAME> ടാഗിന് പകരം പുതിയൊരു <FRAMESET> ടാഗ് തുറക്കുന്നു.

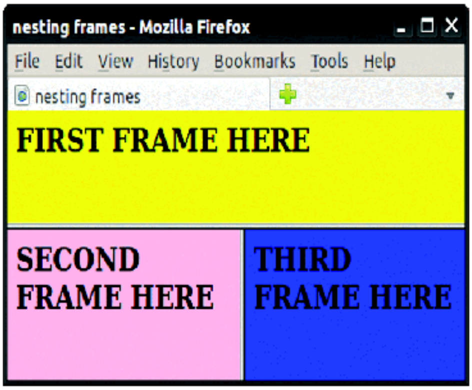
```
<FRAMESET Rows= "85, *">
    <FRAME Src= "sampleframe1.html">
    <FRAMESET Cols= "220, *">
    </FRAMESET>
</FRAMESET>
```

ഈ കോഡ് രണ്ടാമത്തെ വരിയെ രണ്ട് നിരയായി ഭാഗിക്കുന്നു. ഇപ്പോൾ അകത്തെ ഫ്രെയിം സെറ്റിനുള്ളിൽ നമുക്ക് രണ്ട് <FRAME> ടാഗുകൾ കൂട്ടിച്ചേർത്ത് HTML കോഡ് പൂർത്തിയാക്കാം. ഇതിന്റെ ഉദാഹരണം 5.17 ൽ കാണിച്ചിരിക്കുന്നു. ഇതിന്റെ വെബ് പേജിന്റെ ചിത്രം 5.23 ൽ കാണിക്കുന്നു. ഇതിൽ ഉപയോഗിച്ചിരിക്കുന്ന മൂന്ന് HTML പേജുകൾ ചിത്രം 5.17 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ നേരത്തേ തന്നെ നിർമിച്ചിട്ടുണ്ട് എന്ന് കരുതുക.

ഉദാഹരണം 5.17: നെസ്റ്റഡ് ഫ്രെയിം സെറ്റിന്റെ ആശയം വിശദമാക്കുന്നു

```
<HTML>
    <HEAD>
        <TITLE> nesting frames </TITLE>
    </HEAD>
    <FRAMESET Rows= "85, *">
        <FRAME Src= "sampleframe1.html">
        <FRAMESET Cols= "200, *">
            <FRAME Src= "sampleframe2.html">
            <FRAME Src= "sampleframe3.html">
        </FRAMESET>
    </FRAMESET>
</HTML>
```

5.5.5 <NOFRAMES> ടാഗ്



ചിത്രം 5.23: നെസ്റ്റഡ് ഫ്രെയിം സെറ്റ്

പഴയ ചില ബ്രൗസറുകൾ ഫ്രെയിമുകളെ പിന്തുണയ്ക്കുന്നില്ല. ഈ സാഹചര്യം ബ്രൗസറിന് ഉപയോക്താവിനെ അറിയിക്കേണ്ടതുണ്ട്. ഇതിനായി **<NOFRAMES> </NOFRAMES>** ടാഗ് ജോഡികൾ ഉപയോഗിച്ച് ഫ്രെയിംസ് പിന്തുണയ്ക്കാത്ത ബ്രൗസർ ജാലകത്തിൽ ചില വിവരങ്ങൾ പ്രദർശിപ്പിക്കാനാവും. താഴെകൊടുത്തിരിക്കുന്ന കോഡ് <NOFRAMES> ടാഗിന്റെ ഉപയോഗം വ്യക്തമാക്കുന്നു.

```
<HTML>
   <HEAD> <TITLE> A simple
Frameset </TITLE>
   </HEAD>
   <FRAMESET Rows= "20%, 30%, 50%">
      <FRAME Src= "sampleframe1.html">
      <FRAME Src= "sampleframe2.html">
      <FRAME Src= "sampleframe3.html">
   </FRAMESET>
   <NOFRAMES>
      <P> Your browser doesnot support frames.<BR>
          Click <A Href="index.htm">here... </A></P>
   </NOFRAMES>
</HTML>
```

ബ്രൗസർ <FRAMESET> ടാഗ് പിന്തുണയ്ക്കുന്നില്ലെങ്കിൽ, പകരമായി "Click here..." എന്ന സന്ദേശത്തോട് കൂടി *index.html* ലേക്കുള്ള ലിങ്ക് ലഭ്യമാകും.

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. <FRAMESET Rows="100,*"> എന്നത് ബ്രൗസർ ജാലകത്തെ .... ഭാഗങ്ങളാക്കി വിഭജിക്കുന്നു.
2. <FRAME> ടാഗിന്റെ ഏതെങ്കിലും മൂന്ന് ആട്രിബ്യൂട്ടുകൾ എഴുതുക.
3. നെസ്റ്റഡ് ഫ്രെയിംസെറ്റ് എന്നാൽ എന്ത്?
4. <NOFRAME> ടാഗിന്റെ ഉപയോഗം എന്താണ്?
5. ഫ്രെയിംസെറ്റ് പേജിന് <BODY> ഭാഗത്തിന്റെ ആവശ്യകത ഇല്ല. ശരിയോ തെറ്റോ?

5.6 വെബ് പേജുകളിലെ ഫോമുകൾ (Forms in web pages)

ഉപയോക്താവിൽ നിന്ന് ഡാറ്റ ശേഖരിച്ച് അവയിൽ വിവിധ പ്രവർത്തനങ്ങൾ നടത്തുവാൻ HTML ഫോമുകൾ ആവശ്യമാണ്. ഉദാഹരണമായി, കേരളത്തിൽ ഹയർ സെക്കന്ററി വിദ്യാഭ്യാസത്തിനായി XI-ാം ക്ലാസിലെ പ്രവേശനത്തിന് www.hscap.kerala.gov.in എന്ന വെബ്സൈറ്റിൽ നിങ്ങളുടെ പേര്, SSLC രജിസ്റ്റർ നമ്പർ, ഗ്രേഡുകൾ, ആവശ്യമുള്ള വിഷയങ്ങൾ, സ്കൂൾ തുടങ്ങിയ വിവരങ്ങൾ നൽകേണ്ടതുണ്ട്. ഇതിനായി നമുക്ക് HTML ഫോം ഉപയോഗിക്കാം.

ഉപയോക്താവിൽ നിന്ന് വിവരങ്ങൾ ഒരു HTML ഫോമിലൂടെ സ്വീകരിച്ച് ബാക്ക് -എൻഡ് ആപ്ലിക്കേഷനുകളായ കോമൺഗേറ്റ്വേ ഇന്റർഫെസ് (CGI) ആക്ടീവ് സെർവർ പേജുകൾ (ASP), PHP തുടങ്ങിയവയിലേക്ക് ഡാറ്റാപ്രോസസിങ് നടത്തുവാനായി അയയ്ക്കുന്നു. 7-ാമത്തെ അധ്യായത്തിൽ ഈ സാങ്കേതിക വിദ്യയെക്കുറിച്ച് കൂടുതലായി പഠിക്കാം. ഒരു HTML ഫോമിന് രണ്ട് ഘടകങ്ങൾ ഉണ്ട്: **<FORM>** കണ്ടെയ്നറും, അവയ്ക്കുള്ളിൽ സ്ഥാപിക്കാവുന്ന വിവിധ കൺട്രോളുകളും. ടെക്സ്റ്റ് ബോക്സ്, ലിസ്റ്റ്, ടെക്സ്റ്റ് ഏരിയ, ഡ്രോപ്-ഡൗൺ മെനുകൾ, റേഡിയോ ബട്ടണുകൾ, ചെക്ക് ബോക്സ് തുടങ്ങിയവ വിവിധ തരത്തിലുള്ള ഫോം കൺട്രോളുകളാണ്.

5.6.1 <FORM> ടാഗ്

ഫോം നിർമിക്കുന്നതിനുള്ള കണ്ടെയ്നർ ആയി **<FORM>** ടാഗ് പ്രവർത്തിക്കുന്നു. **<FORM>** ൽ ആരംഭിക്കുന്നതും **</FORM>** ൽ അവസാനിക്കുന്നതുമായ ടാഗ് ജോഡി ഉപയോഗിച്ചാണ് ഫോം നിർമിക്കുന്നത്. ഒരു വെബ് ബ്രൗസറിന് ഫോമിൽ കൂടി മാത്രമേ വിവരങ്ങൾ ശേഖരിക്കുവാൻ കഴിയൂ. ശേഖരിച്ച ഡാറ്റ കൈകാര്യം ചെയ്യാൻ ചില ബാക്ക്-എൻഡ് ആപ്ലിക്കേഷനുകൾ ഉപയോഗിക്കണം. ഇതിനായി ഫോം ഹാൻഡ്ലറുകളായ (Form handler) CGI, JavaScript, PHP എന്നിവ ഉപയോഗിക്കാം. ഒരു ഫോമിൽ നിന്ന് ശേഖരിച്ച എല്ലാ ഇൻപുട്ടുകളും ഒരേ ഫോം ഹാൻഡ്ലറിൽ കൂടി മാത്രമേ പ്രോസസ് ചെയ്യാൻ കഴിയൂ. ഒരു ഫോമിൽക്കൂടി അയച്ച ഡാറ്റ നിയന്ത്രിക്കുന്ന വെബ് സർവറിലെ ഒരു പ്രോഗ്രാമാണ് ഫോം ഹാൻഡ്ലർ. `<FORM>` ടാഗിലെ `Action` ആട്രിബ്യൂട്ടിന്റെ വിലയായി ഫോം ഹാൻഡ്ലെറിന്റെ പേര് നൽകുന്നു. സർവർ, അനുബന്ധസാങ്കേതിക വിദ്യകൾ എന്നിവയുടെ ആശയം 7-ാമത്തെ അധ്യായത്തിൽ ചർച്ചചെയ്യാം.

`<FORM>` ടാഗിൽ സാധാരണയായി ഉപയോഗിക്കുന്ന ആട്രിബ്യൂട്ടുകൾ ചുവടെ ചേർക്കുന്നു.

1. **Action**: ഈ ആട്രിബ്യൂട്ടിന്റെ വിലയായി ഡാറ്റ സ്വീകരിക്കുവാനും പ്രോസസ് ചെയ്യുവാനും കഴിയുന്ന ഫോം ഹാൻഡ്ലറിന്റെ URL നൽകുന്നു.
2. **Method**: ഈ ആട്രിബ്യൂട്ടിന്റെ വിലയായി ഡാറ്റ അപ്ലോഡ് (upload) ചെയ്യുന്ന രീതി സൂചിപ്പിക്കുന്നു. സാധാരണയായി `Get` ഉം `Post` ഉം രീതികളാണ് അവലംബിക്കുന്നത്.
3. **Target**: ഈ ആട്രിബ്യൂട്ട് സ്ക്രിപ്റ്റിന്റെ ഫലം പ്രദർശിപ്പിക്കേണ്ട വിൻഡോയോ ഫ്രെയിമോ സൂചിപ്പിക്കുന്നു. ഇതിന്റെ വിലകൾ പട്ടിക 5.4-ൽ നൽകുന്നു `_blank`, `_self`, `_parent`.

വില	വിവരണം
_blank	ലിങ്ക് ചെയ്യപ്പെട്ട ഡോക്യുമെന്റ് പുതിയ ജാലകത്തിൽ തുറക്കുന്നു.
_self	ലിങ്ക് ചെയ്യപ്പെട്ട ഡോക്യുമെന്റ് ലിങ്കുള്ള അതേ ഫ്രെയിമിൽ തുറക്കുന്നു.
_parent	ലിങ്ക് ചെയ്യപ്പെട്ട ഡോക്യുമെന്റ് പേരന്റ് ഫ്രെയിംസെറ്റിൽ തുറക്കുന്നു.
_top	മുഖ്യ ബ്രൗസർ ജാലകത്തിൽ നിലവിലുള്ള ഫ്രെയിമിനെ മാറ്റി ലിങ്ക് ചെയ്യപ്പെട്ട ഡോക്യുമെന്റ് തുറക്കുന്നു.
name	പേര് ഉപയോഗിച്ച് സൂചിപ്പിച്ചിരിക്കുന്ന ജാലകത്തിൽ ലിങ്ക് ചെയ്യപ്പെട്ട ഡോക്യുമെന്റ് തുറക്കുന്നു.

പട്ടിക 5.4: Target ആട്രിബ്യൂട്ടിന്റെ വിലകൾ

ബ്രൗസറിൽ നിന്ന് വെബ്സെർവറിലെ ഫോം ഹാൻഡ്‌ലറിലേക്ക് ഫോം സമർപ്പിക്കുന്നതിന് മുമ്പ് ഉപയോക്താവ് ഫോമിൽ എല്ലാ അവശ്യവിവരങ്ങളും നൽകിയിട്ടുണ്ടോ എന്നും അല്ലെങ്കിൽ സ്വീകരിച്ച വിവരങ്ങൾ നിലനിൽക്കുന്നതാണോ അല്ലയോ എന്നും പരിശോധിക്കുന്നതിനാണ് ജാവാ സ്ക്രിപ്റ്റ് അല്ലെങ്കിൽ മറ്റേതെങ്കിലും ക്ലയന്റ്സൈഡ് സ്ക്രിപ്റ്റിങ് ഭാഷ സാധാരണയായി ഉപയോഗിക്കുന്നത്.

ഫോം കൺട്രോളുകൾ (Form controls)

HTML ഫോമിൽ നിന്ന് ഡാറ്റ ശേഖരിക്കാൻ വിവിധതരം ഫോം കൺട്രോളുകൾ ഉപയോഗിക്കാം. ഇവയിൽ ടെക്സ്റ്റ് ബോക്സ്, പാസ്‌വേർഡ്, ചെക്ക് ബോക്സ്, റേഡിയോ ബട്ടൺ, ടെക്സ്റ്റ് ഏരിയ, സെലക്ട് ബോക്സ്, സബ്മിറ്റ്, റീസെറ്റ് ബട്ടൺ തുടങ്ങിയവ ഉൾപ്പെടുന്നു. ഇതിലെ മിക്ക കൺട്രോളുകളും **<INPUT>** ടാഗ് വഴി ഫോമിൽ ഉൾപ്പെടുത്താം.

5.6.2 <INPUT> ടാഗ്

ഉപയോക്താക്കളിൽ നിന്ന് ഡാറ്റ സ്വീകരിക്കാൻ ഫോമിൽ വിവിധതരം കൺട്രോളുകൾ പ്രദർശിപ്പിക്കും. ഇൻപുട്ടിന്റെ തരം അനുസരിച്ച് വിവിധതരത്തിലുള്ള കൺട്രോളുകളായ ടെക്സ്റ്റ് ബോക്സ്, റേഡിയോ ബട്ടൺ, സബ്മിറ്റ് ബട്ടൺ തുടങ്ങിയവ **<INPUT>** ടാഗ് ഉപയോഗിച്ച് നിർമിക്കാം. ഇതിന്റെ **Type** ആട്രിബ്യൂട്ട് കൺട്രോളിന്റെ തരം തീരുമാനിക്കുന്നു.

<INPUT> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകൾ

1. **Type**: <INPUT> ടാഗ് നിർമിക്കുന്ന കൺട്രോളിന്റെ തരം നിർണയിക്കുന്നത് ഈ ആട്രിബ്യൂട്ടാണ്. ഇതിന്റെ പ്രധാനപ്പെട്ട വിലകൾ പട്ടിക 5.5ൽ കൊടുത്തിരിക്കുന്നു.

വില	വിവരണം
Text	ടെക്സ്റ്റ് ബോക്സ് നിർമിക്കുന്നു.
Password	ടെക്സ്റ്റ് ബോക്സ് തന്നെയാണ് ഇതും നിർമിക്കുന്നത് എന്നാൽ ഇതിൽ നൽകുന്ന അക്ഷരങ്ങൾ എല്ലാം അസ്റ്ററിസ്ക് (*) പോലുള്ള ചിഹ്നങ്ങളാണ് പ്രദർശിപ്പിക്കുക.
Checkbox	അതെ, അല്ല എന്നീ വിലകൾ നൽകുന്നതിന് നിർമിച്ചിരിക്കുന്നു. ഒരു കൂട്ടം വിലകളിൽ ഒന്നോ അതിലധികമോ തിരഞ്ഞെടുക്കാൻ ഉപയോഗിക്കുന്നു.
Radio	ഒരു കൂട്ടം വിലകളിൽ നിന്ന് ഒരു വില തിരഞ്ഞെടുക്കുവാൻ ഇത് ഉപയോഗിക്കുന്നു. Name ആട്രിബ്യൂട്ടിന് ഒരേ വിലയുള്ള ഒന്നിലധികം റേഡിയോ ബട്ടണിൽ നിന്ന് ഒന്നുമാത്രമേ ഒരുസമയത്ത് തിരഞ്ഞെടുക്കുവാൻ കഴിയൂ. ഉപയോക്താവ് സെലക്ഷൻ മാറ്റുമ്പോൾ നേരത്തേ തിരഞ്ഞെടുത്തത് ഡിസെലക്ട് ചെയ്യപ്പെടും.
Reset	ഒരു ഫോമിലെ രേഖപ്പെടുത്തലുകളെല്ലാം ഒഴിവാക്കി ഫോമിനെ പ്രാരംഭഘട്ടത്തിലേതുപോലെ ആക്കുന്നതിന് ഉപയോഗിക്കുന്ന പ്രത്യേക ബട്ടൺ ആണിത്.
Submit	ഫോമിൽ നൽകിയ ഡാറ്റാ സെർവറിലേക്ക് സമർപ്പിക്കുന്നതിനായി ഉപയോഗിക്കുന്ന പ്രത്യേക ബട്ടണാണിത്.
Button	ഫോമിൽ ഒരു ഗ്രാഫിക്കൽ ബട്ടൺ നിർമിക്കുവാൻ ഉപയോഗിക്കുന്നു. ഈ ബട്ടണിൽ ക്ലിക്ക് ചെയ്ത് ഫങ്ഷനുകളെ വിളിക്കാം.

പട്ടിക 5.5: Type ആട്രിബ്യൂട്ടിന്റെ വിവിധ വിലകൾ

2. **Name**: ഇൻപുട്ട് കൺട്രോളുകളുടെ പേര് നൽകുന്നു. ഫോം സമർപ്പിക്കുമ്പോൾ ഡാറ്റായും അത് ഉൾക്കൊള്ളുന്ന കൺട്രോളിന്റെ പേരും സർവറിലേക്ക് അയയ്ക്കുന്നു.
3. **Value**: ഒരു കൺട്രോളിന് തനത് വില നൽകുന്നതിന് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു.
4. **Size**: ഇൻപുട്ട് ടൈപ്പായ ടെക്സ്റ്റ് ബോക്സ്, പാസ്‌വേർഡ് ടെക്സ്റ്റ് എന്നിവയുടെ വലുപ്പം നിശ്ചയിക്കുന്നതിന് ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു ഇൻപുട്ടുകളായ ടെക്സ്റ്റ്, പാസ്‌വേർഡ് എന്നിവയ്ക്ക് മാത്രം ഇത് ബാധകമാകും.
5. **Maxlength**: ടെക്സ്റ്റ് ബോക്സ്, പാസ്‌വേർഡ് എന്നിവയിൽ ഉപയോക്താവിന് ഇൻപുട്ട് ചെയ്യുവാൻ സാധിക്കുന്ന പരമാവധി അക്ഷരങ്ങളുടെ എണ്ണം നിശ്ചയിക്കുവാൻ ഈ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു. ഈ ആട്രിബ്യൂട്ട് ടെക്സ്റ്റ്ബോക്സ്, പാസ്‌വേർഡ് എന്നിവയുടെ കൂടെ മാത്രമേ ഉപയോഗിക്കൂ.

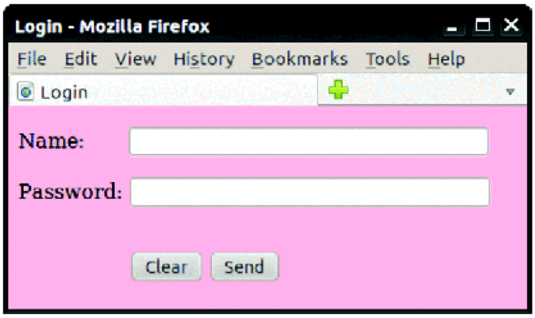
പാസ്‌വേർഡ് ഫീൽഡിലെ ഉള്ളടക്കങ്ങൾ ഉപയോക്താവിന്റെ കാഴ്ചയിൽ നിന്ന് മറയ്ക്കുന്നു. ഇതിൽ നമ്മൾ നൽകുന്ന വിവരങ്ങൾ രഹസ്യകോഡ് ആക്കിമാറ്റുകയോ കൂട്ടിക്കലർത്തുകയോ ചെയ്യുന്നില്ല.

ഒരു HTML ഫോമിൽക്കൂടി പേര്, പാസ്‌വേഡ് എന്നിവ നൽകുന്നതിനുള്ള HTML കോഡ് ഉദാഹരണം 5.18 ലും വെബ് പേജിന്റെ ചിത്രം 5.24 ലും കാണിക്കുന്നു.

ഉദാഹരണം 5.18 : HTML ഫോം നിർമാണം

```
<HTML>
<HEAD> <TITLE> Login </TITLE> </HEAD>
<BODY Bgcolor= "Pink">
   <FORM Action= "login.php" Method= "post">
      <P> Name:    
      <INPUT Type= "text" Name= "firstname" Size= "30"
            Maxlength= "25"> </P>
      <P>Password:
      <INPUT Type= "password" Name= "Psswd" Size= "30"
            Maxlength= "25"> </P> <BR>
                     
       
      <INPUT Type= "reset" Value= "Clear">
      <INPUT Type= "submit" Value= "Send">
   </FORM>
</BODY>
</HTML>
```

ഉദാഹരണം 5.18 ൽ <FORM> ടാഗിലെ Action ആട്രിബ്യൂട്ടിന്റെ വിലയായി *login.php* നൽകുന്നു. ഇതിനർഥം ഫോമിൽക്കൂടി നൽകപ്പെടുന്ന ഡാറ്റ *login.php* എന്ന ഫയലിൽ ശേഖരിച്ചിരിക്കുന്ന ആപ്ലിക്കേഷനിലൂടെ പ്രോസസ് ചെയ്യപ്പെടുന്നു എന്നാണ്. ` ` എന്ന HTML അംഗത്തിന്റെ ഉപയോഗം ശ്രദ്ധിക്കുക. ഇത് അഭാജ്യമായ സ്ഥലം നൽകുവാനാണ് ഉപയോഗിക്കുന്നത്.



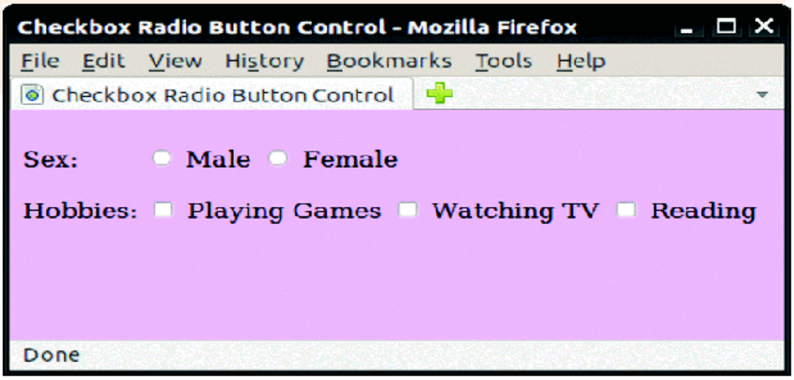
ചിത്രം 5.24: ടെക്സ്റ്റ്/പാസ്‌വേർഡ് എന്നിവ ഉൾപ്പെട്ട HTML ഫോം

ഒരു റേഡിയോ ബട്ടൺ (ഓപ്ഷൻ ബട്ടൺ), ചെക്ക് ബോക്സ് എന്നിവയിലൂടെ ഡാറ്റ ഇൻപുട്ട് ചെയ്യുന്ന HTML ഫോം രൂപകൽപന ചെയ്യാം. ഒന്നിൽ കൂടുതൽ വിലകളിൽ നിന്ന് ഒരു വില മാത്രം തിരഞ്ഞെടുക്കാൻ റേഡിയോ ബട്ടണും, ഒന്നിലധികം വിലകൾ തിരഞ്ഞെടുക്കുവാൻ ചെക്ക് ബോക്സും ഉപയോഗിക്കുന്നു. സാധാരണയായി ഒരു ലിസ്റ്റിലെ ഒന്നോ അതിലധികമോ ഇനങ്ങളെ മാർക്ക് ചെയ്യുന്ന രീതിയിലാണ് ചെക്ക് ബോക്സ് ഉപയോഗിക്കുന്നത്.

ഉദാഹരണം 5.19 ൽ നൽകിയിരിക്കുന്ന HTML കോഡ് ഒരു വ്യക്തിയുടെ ലിംഗഭേദവും വിനോദങ്ങളും യഥാക്രമം റേഡിയോ ബട്ടൺ, ചെക്ക് ബോക്സ് എന്നിവയിലൂടെ സൂചിപ്പിക്കുന്നു. ഇതിനനുസൃതമായ വെബ് പേജ് ചിത്രം 5.25 ൽ കാണിക്കുന്നു.

ഉദാഹരണം 5.19 : റേഡിയോ ബട്ടൺ ചെക്ക് ബോക്സ് എന്നിവ ഉൾപ്പെടുത്തിയ ഒരു HTML ഫോം നിർമാണം

```
<HTML>
<HEAD> <TITLE>Checkbox Radio Button Control</TITLE> </HEAD>
<BODY Bgcolor= "#E9BEE5">
   <FORM> <BR>Sex:       
      <INPUT Type= "radio" Name= "sex" Value= "male"> Male
      <INPUT Type= "radio" Name= "sex" Value= "female"> Female
      <BR> <BR>Hobbies:
      <INPUT Type= "checkbox" Name= "Hobby" Value= "Games">
      Playing Games
      <INPUT Type= "checkbox" Name= "Hobby"
            Value="WatchingTV"> Watching TV
      <INPUT Type= "checkbox" Name= "Hobby" Value= "Reading">
      Reading
   </FORM>
</BODY>
</HTML>
```



ചിത്രം 5.25: ചെക്ക് ബോക്സും റേഡിയോ ബട്ടണും ഉൾപ്പെടുത്തിയ വെബ് പേജ്.

5.6.3 <TEXTAREA> ടാഗ്

ഒരു ഫോമിൽ <INPUT Type = "text"> എന്ന പ്രസ്താവന ഉപയോഗിച്ച് ഒറ്റ വരി വാക്യം നൽകുന്ന രീതി നാം കണ്ടു കഴിഞ്ഞു. (ഉദാഹരണം 5.18 ഉം ചിത്രം 5.24 ഉം) നോക്കുക. എന്നാൽ ചില അവസരങ്ങളിൽ ഒന്നിലധികം വരികളുള്ള വാചകങ്ങൾ ഫോമിൽ നൽകേണ്ടത് ആവശ്യമായി വരും. ഒരു വ്യക്തിയുടെ തപാൽ വിലാസം ഇതിന് ഉദാഹരണമാണ്. ഇതിനായി **<TEXTAREA>** എന്ന കണ്ടെയ്നർ ടാഗ് ഉപയോഗിക്കാം. ടാഗ് ജോഡികൾക്ക് ഇടയിലുള്ള ഒന്നിലധികം വരികളുള്ള വാക്യങ്ങൾ പ്രദർശിപ്പിക്കാ

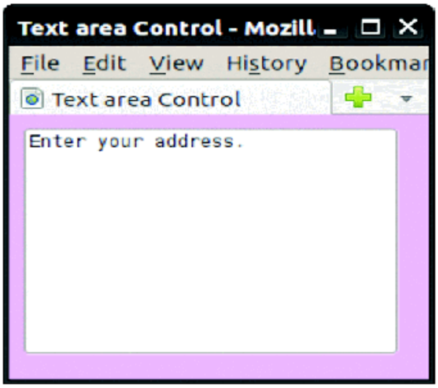
നുള്ള ടെക്സ്റ്റ് ഏരിയായുടെ വിസ്തൃതി നിർണയിക്കുന്നത് ഈ ടാഗിന്റെ ആട്രിബ്യൂട്ടുകളുടെ വിലയെ ആശ്രയിച്ചിരിക്കും. <TEXTAREA> ടാഗിന്റെ പ്രധാന ആട്രിബ്യൂട്ടുകൾ ചുവടെ ചേർക്കുന്നു.

1. **Name:** ഇത് കൺട്രോളിന് പേര് നൽകുന്നു.
2. **Rows:** ഇത് വരികളുടെ എണ്ണം നിർണയിക്കുന്നു.
3. **Cols:** ഇത് നിരകളുടെ എണ്ണം നിർണയിക്കുന്നു. അതായത് ഒരു വരിയിലെ അക്ഷരങ്ങളുടെ എണ്ണം.

താഴെ നൽകിയിരിക്കുന്ന കോഡ് ഭാഗം പരിഗണിച്ച് <TEXTAREA>ടാഗിന്റെ ഉപയോഗം തിരിച്ചറിയുക.

```
<FORM Action= "guestbook.php" Method= "post">
   <TEXTAREA Rows= "10" Cols= "30" Name= "address">
       Enter your address.
   </TEXTAREA>
</FORM>
```

ഈ കോഡിനെ ബ്രൗസറിൽ തുറക്കുമ്പോൾ ലഭ്യമാകുന്ന വെബ് പേജ് ചിത്രം 5.26 ൽ നൽകിയിരിക്കുന്നു.<TEXTAREA>, </TEXTAREA> എന്നീ ടാഗ് ജോഡികൾക്കിടയിൽ നൽകുന്ന വാക്യങ്ങൾ കൺട്രോളിൽ പ്രദർശിപ്പിക്കുന്നു. ഉപയോക്താവ് ഏതെങ്കിലും ഡാറ്റ കൺട്രോളിൽ നൽകുമ്പോൾ ഈ വാക്യത്തിന് പകരമായി അത് പ്രദർശിപ്പിക്കുന്നു.



ചിത്രം 5.26: ടെക്സ്റ്റ് ഏരിയ ഉൾക്കൊള്ളുന്നു.

5.6.4 <SELECT> ടാഗ്

ഉപയോക്താവിന് ഒന്നോ അതിലധികമോ ഓപ്ഷനുകൾ തിരഞ്ഞെടുക്കുവാൻ കഴിയുന്ന കൺട്രോൾ ആണ് സെലക്ട് ബോക്സ്. ഇത് വിവിധ ഓപ്ഷനുകൾ താഴേക്കു വീഴുന്ന ഒരു ലിസ്റ്റ് രൂപത്തിൽ നൽകുന്നു. അതിനാൽ ഇത് ഡ്രോപ്ഡൗൺ ലിസ്റ്റ് ബോക്സ് എന്ന പേരിലും അറിയപ്പെടുന്നു. പരിമിതമായ സ്ഥലത്ത് കൂടുതൽ ഓപ്ഷനുകൾ പ്രദർശിപ്പിക്കുവാൻ സെലക്ട് ബ്ലോക്സ് സഹായിക്കുന്നു. **<SELECT>, </SELECT>** ടാഗിനകത്ത് നൽകുന്ന **<OPTION>** ടാഗ് ഉപയോഗിച്ചാണ് ലിസ്റ്റിൽ പ്രദർശിപ്പിക്കേണ്ട ഇനങ്ങൾ നൽകുന്നത്.

കണ്ടെയ്നർ ടാഗായ <SELECT> ടാഗിന്റെ പ്രധാനപ്പെട്ട ആട്രിബ്യൂട്ടുകൾ ചുവടെ ചേർക്കുന്നു.

1. **Name:** ഈ ആട്രിബ്യൂട്ട് കൺട്രോളിന് പേര് നൽകുന്നു. ഈ പേര് ഇതിലെ ഡാറ്റയോടൊപ്പം സെർവറിലേക്ക് അയയ്ക്കുന്നു. സെർവർ ഈ പേര് ഉപയോഗിച്ച് കൺട്രോളിന്റെ പേരിനൊപ്പമുള്ള വില തിരിച്ചറിയുകയും സ്വീകരിക്കുകയും ചെയ്യുന്നു.

2. **Size**: ഒരു സ്ക്രോളിങ് ലിസ്റ്റ് അവതരിപ്പിക്കുന്നതിന് ഉപയോഗിക്കാം. ഇതിന്റെ വിലയുടെ അടിസ്ഥാനത്തിൽ ഇത് ഒരു ലിസ്റ്റ് ബോക്സാണോ അല്ലെങ്കിൽ ഒരു ഡ്രോപ്ഡൗൺ ലിസ്റ്റ്ബോക്സാണോ എന്ന് തീരുമാനിക്കുന്നു. ഇതിന്റെ വില 1 ആണെങ്കിൽ ഡ്രോപ്ഡൗൺ ലിസ്റ്റ് ബോക്സായിരിക്കും.
3. **Multiple**: ഉപയോക്താവിന് ഒന്നിൽകൂടുതൽ ഇനങ്ങൾ തിരഞ്ഞെടുക്കുവാൻ ഈ ആട്രിബ്യൂട്ട് അനുവദിക്കുന്നു.

ഇനി നമുക്ക് <OPTION> ടാഗിനെക്കുറിച്ച് ചർച്ച ചെയ്യാം. <SELECT> </SELECT> എന്നീ ടാഗ് ജോഡികൾക്കിടയിൽ നൽകുന്ന എംപ്റ്റി ടാഗ് ആണിത്. <OPTION> ടാഗ് സെലക്ട് ബോക്സിനകത്ത് നൽകിയിരിക്കുന്ന ഓപ്ഷനുകൾ ലിസ്റ്റ് ചെയ്യുന്നു. ഈ ടാഗിന്റെ വിവിധ ആട്രിബ്യൂട്ടുകൾ ചുവടെ ചേർക്കുന്നു.

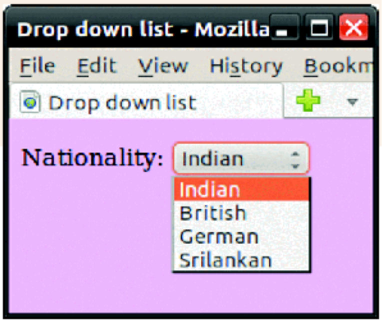
1. **Selected**: തനത് സെലക്ഷനെ സൂചിപ്പിക്കാൻ ഉപയോഗിക്കുന്നു.
2. **Value**: <OPTION> ടാഗിന് അകത്തുള്ള വിലകളിൽ നിന്നും ഉള്ളടക്കത്തേക്കാൾ വിഭിന്നമായ വില നൽകുവാൻ ഇത് ഉപയോഗിക്കുന്നു. ഈ വില ലഭ്യമല്ലെങ്കിൽ ഇതിന്റെ ഉള്ളടക്കം വിലയായി സ്വീകരിക്കും.

ഉദാഹരണം 5.20 ൽ നൽകിയിരിക്കുന്ന HTML കോഡ് ഉപയോഗിച്ച് ഒരു ഡ്രോപ്ഡൗൺ ലിസ്റ്റ് ബോക്സ് ഉൾപ്പെടുന്ന വെബ് പേജ് നിർമിക്കാം. <SELECT> ടാഗിന്റെ Size ആട്രിബ്യൂട്ടിന് ഒന്ന് (1) എന്ന വില ആണെന്നും കൂടാതെ സെലക്ട് ബോക്സിന് 4 ഓപ്ഷനുകൾ ഉണ്ടെന്നും പ്രത്യേകം ശ്രദ്ധിക്കുക.

ഉദാഹരണം 5.20: ഒരു ഡ്രോപ്ഡൗൺ ലിസ്റ്റ് ഉൾപ്പെട്ട HTML ഫോം നിർമാണം

```
<HTML>
<HEAD> <TITLE> Drop down list </TITLE> </HEAD>
<BODY Bgcolor= "#E9BEE5">
   <FORM Action= "guestbook.php" Method= "post">
      <P> Nationality:
      <SELECT Name= "Nationality" Size= "1">
          <OPTION Value= "Indian" selected> Indian
          <OPTION Value= "British"> British
          <OPTION Value= "German"> German
          <OPTION Value= "Srilankan"> Srilankan
      </SELECT>
   </FORM>
</BODY>
</HTML>
```

കോമ്പോ ബട്ടണിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ ലഭ്യമായ ഡ്രോപ്ഡൗൺ ലിസ്റ്റിൽ ചിത്രം 5.27ൽ കാണിക്കുന്നു.



ചിത്രം 5.27: സെലക്ട് ബോക്സ് ഉള്ള ഫോം

നമുക്കു ചെയ്യാം

ഉദാഹരണം 5.20 ലെ പ്രോഗ്രാം കോഡിൽ Size ആട്രിബ്യൂട്ടിനെ വ്യത്യാസപ്പെടുത്തിക്കൊണ്ട് സെലക്ട് ബോക്സിൽ ഉണ്ടാകുന്ന മാറ്റം നിരീക്ഷിക്കുക.

5.6.5 <FIELDSET> ടാഗ് ഉപയോഗിച്ച് ഫോം ഡാറ്റയെ ഒന്നിപ്പിക്കുന്നു (Grouping form data with <FIELDSET>tag)

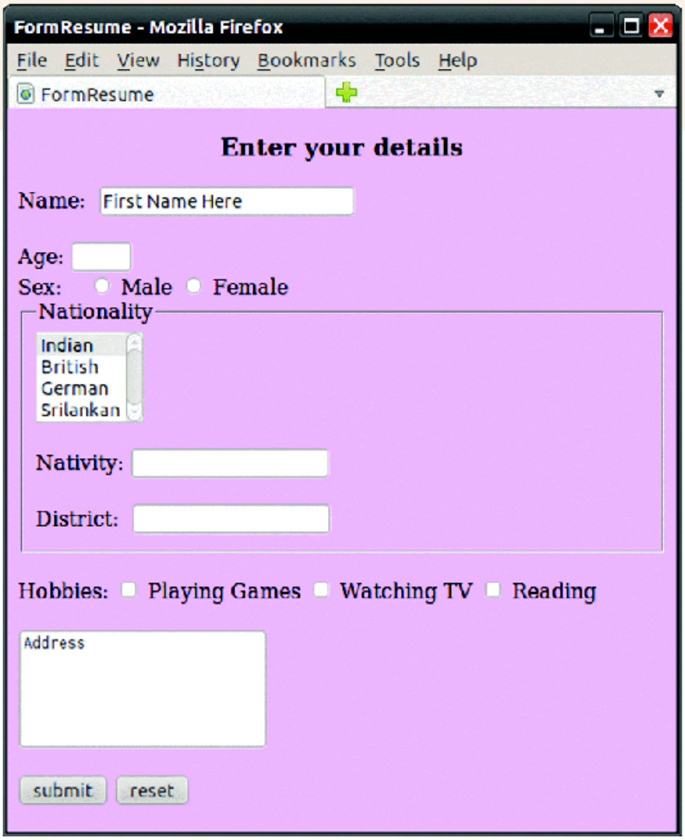
ചിലപ്പോൾ പരസ്പരബന്ധമുള്ള ഡാറ്റ ഉൾക്കൊള്ളുന്ന കൺട്രോളുകളെ ഒരുമിച്ച് ചേർക്കേണ്ടത് ആവശ്യമായി വന്നേക്കാം. ഇതിനായി **<FIELDSET>** ടാഗ് ഉപയോഗിക്കാം. ഈ ടാഗ് ഉപയോഗിച്ച് ഫോമിനകത്ത് പരസ്പര ബന്ധമുള്ള കൺട്രോളുകൾ അടങ്ങിയ ഉപവിഭാഗങ്ങൾ ഉൾപ്പെടുത്തുവാൻ സാധിക്കും. ഓരോ ഉപവിഭാഗത്തെയും തിരിച്ചറിയുവാൻ **<LEGEND>** ടാഗ് ഉപയോഗിക്കാം. <FIELDSET> ഇനങ്ങൾക്ക് ശീർഷകം നൽകുന്നതിന് <LEGEND> ടാഗ് ഉപയോഗിക്കാം. ഇത് ഒരു കണ്ടെയ്നർ ടാഗ് ആണ്.

നമ്മൾ ഇതുവരെ ചർച്ച ചെയ്ത എല്ലാ കൺട്രോളുകളും ഉൾപ്പെടുത്തി പേര്, വയസ്, അഡ്രസ്, വിനോദങ്ങൾ എന്നീ വിവരങ്ങൾ സമർപ്പിക്കുന്നതിനുള്ള HTML കോഡ് ഉദാഹരണം 5.21 ലും അതിനനുസൃതമായ വെബ് പേജ് ചിത്രം 5.28 ലും കാണിച്ചിരിക്കുന്നു.

ഉദാഹരണം 5.21: ഒരു വിദ്യാർഥിയുടെ വിവരങ്ങൾ സമർപ്പിക്കുന്ന HTML കോഡ്

```
<HTML>
<HEAD> <TITLE> FormResume </TITLE> </HEAD>
<BODY Bgcolor= "#E9BEE5">
   <CENTER ><H3>Enter your details</H3></CENTER>
   <FORM Action= "guestbook.php" Method= "get">
      Name: 
      <INPUT Type= "text" Name= "first_name" Size= "20"
            Maxlength= "20" Value= "First Name Here"><BR><BR>
      Age:
      <INPUT Type="text" Name="age" Size="3" Maxlength="3"><BR>
      Sex:    
      <INPUT Type="radio" Name="sex" Value="male"> Male
      <INPUT Type="radio" Name="sex" Value="female"> Female
      <FIELDSET>
         <LEGEND>Nationality</LEGEND>
         <SELECT Name= "Nationality" Size= "4">
            <OPTION Value= "Indian" Selected> Indian
            <OPTION Value= "British"> British
            <OPTION Value= "German"> German
            <OPTION Value= "Srilankan"> Srilankan
         </SELECT><BR><BR>
```

```
            Nativity:
            <INPUT Type= "text" Name="State" Size="15"><BR><BR>
            District: 
            <INPUT Type= "text" Name= "District" Size= "15">
        </FIELDSET><BR>
        Hobbies:
        <INPUT Type= "checkbox" Name= "Hobby" Value= "games">
                Playing Games
        <INPUT Type= "checkbox" Name= "Hobby"
                Value= "WatchingTV"> Watching TV
        <INPUT Type= "checkbox" Name= "Hobby" Value= "Reading">
                Reading<BR><BR>
        <TEXTAREA Rows= "5" Cols= "25" Name= "address">Address
        </TEXTAREA><BR><BR>
        <INPUT Type= "submit" Value= "submit">
        <INPUT Type= "Reset" Value= "reset">
    </FORM>
</BODY>
</HTML>
```



ചിത്രം 5.28: ഒരു വിദ്യാർഥിയുടെ വിവരങ്ങൾ സമർപ്പിക്കുന്നതിനുള്ള HTML ഫോം

5.6.6 ഫോം സമർപ്പണം (Form submission)

നാം ഫോമിലെ submit ബട്ടണിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ ഫോമിൽ നൽകിയ ഡാറ്റ സെർവറിലേക്ക് അയയ്ക്കപ്പെടുന്നു. ഉപയോക്താവിന്റെ (ക്ലയന്റ്) കമ്പ്യൂട്ടറിൽ നിന്നും വരുന്ന വിവരങ്ങൾ വെബ്സെർവർ സ്വീകരിച്ച് അതിൽ ലഭ്യമായ സെർവർസൈഡ് പ്രോഗ്രാം ഉപയോഗിച്ച് പ്രോസസ് ചെയ്യുന്നു.

ഫോം സമർപ്പിക്കുമ്പോൾ ചെയ്യേണ്ട പ്രവർത്തനങ്ങളെ നിർവചിക്കുന്ന ആട്രിബ്യൂട്ടാണ് `Action`. സാധാരണയായി സെർവർസൈഡ് പ്രോഗ്രാമിന്റെ URL ആണ് ഇതിന്റെ വിലയായി നൽകുന്നത്. ഒരു ഫോം സർവറിലേക്ക് അയയ്ക്കുന്നതിനുള്ള പൊതുവായ മാർഗം **submit** ബട്ടണിൽകൂടി ആണ്. സമർപ്പിക്കുന്ന ഫോമിലെ വിവരങ്ങൾ കൈകാര്യം ചെയ്യാൻ *guestbook.php* എന്ന സെർവർ സൈഡ് സ്ക്രിപ്റ്റ് ഉപയോഗിച്ചിരിക്കുന്നതായി ഉദാഹരണം 5.21 ൽ കാണാവുന്നതാണ്.

`<FORM>` ടാഗിലെ `Method` ആട്രിബ്യൂട്ട് ഫോം സമർപ്പിക്കുന്നതിനുള്ള HTTP മാർഗം (`get` or `post`) സൂചിപ്പിക്കുന്നു.

നിങ്ങളുടെ പുരോഗതി അറിയുക

1. വെബ് പേജിലൂടെ ഇൻപുട്ട് ഡാറ്റ നൽകുവാൻ HTML ൽ ________ ആണ് ഉപയോഗിക്കുന്നത്.
2. ഫോമിനകത്ത് ടെക്സ്റ്റ് ഡാറ്റ ഇൻപുട്ട് ചെയ്യുന്നതിന് ഉപയോഗിക്കുന്ന രണ്ട് ടാഗുകളുടെ പേരെഴുതുക.
3. റേഡിയോ ബട്ടൺ, ചെക്ക്ബോക്സ് എന്നിവ തമ്മിലുള്ള വ്യത്യാസങ്ങൾ എഴുതുക.
4. ഫോമിനുള്ളിൽ ഡാറ്റ ഒരുമിപ്പിക്കുവാൻ ഉപയോഗിക്കുന്ന ടാഗ് ഏതാണ്?
5. <FORM> ടാഗിനുള്ളിൽ ഡാറ്റ ഇൻപുട്ട് ചെയ്യുന്നതിന് ഉപയോഗിക്കുന്ന ടാഗിന്റെ പേര് എഴുതുക.

5.7 HTML 5 ന്റെ പൊതു അവലോകനം (Overview of HTMLS)

HTML ന്റെ പുതുക്കിയ പതിപ്പാണ് HTML 5. HTML 4.01 ന് ശേഷം വേൾഡ് വൈഡ് വെബ് കൺസോർഷ്യവും (W3 C) വെബ് ഹൈപ്പർ ടെക്സ്റ്റ് ആപ്ലിക്കേഷൻ ടെക്നോളജി വർക്കിങ് ഗ്രൂപ്പും (WHATWG) കൂട്ടായി വികസിപ്പിച്ചെടുത്തതാണ് HTML5. ഇതിൽ വീഡിയോ പ്ലേ ബാക്ക്, ഡ്രാഗ് ആൻഡ് ഡ്രോപ്പ് തുടങ്ങിയവ കൂട്ടിച്ചേർത്തിട്ടുണ്ട്.

പ്രമുഖ ബ്രൗസറുകളായ ഗൂഗിൾ ക്രോം, മോസില്ല ഫയർഫോക്സ്, ആപ്പിൾ സഫാരി ഓപ്പേറ, ഇന്റർനെറ്റ് എക്സ്പ്ലോറർ എന്നിവയുടെ ഏറ്റവും പുതിയ പതിപ്പുകൾ HTML5 നെ പിന്തുണയ്ക്കുന്നു. ഐ ഫോണുകൾ, ഐപാഡുകൾ, ആൻഡ്രോയിഡ് ഫോണുകൾ തുടങ്ങിയവയിലുള്ള മൊബൈൽ ബ്രൗസറുകളും HTML 5 നെ പിന്തുണയ്ക്കുന്നു. ഇപ്പോഴുള്ള ബ്രൗസറുകളിൽ പഴയതിനെയും പ്രവർത്തിപ്പിക്കാവുന്ന രീതിയിൽ ഉൾക്കൊള്ളിച്ചാണ് HTML 5 രൂപകൽപന ചെയ്തിരിക്കുന്നത്. HTML5 ന്റെ ലോഗോ ചിത്രം 5.29-ൽ നൽകിയിരിക്കുന്നു.

ചിത്രം 5.29: HTML 5 ന്റെ ലോഗോ.

HTML5 ൽ പരിചയപ്പെടുത്തുന്ന ഏതാനും പുതിയ ടാഗുകൾ താഴെ നൽകുന്നു.

1. **<VIDEO>, <AUDIO>:** HTML ഡോക്കുമെന്റുകളിൽ ദൃശ്യ ശ്രാവ്യ മാധ്യമങ്ങൾ വളരെ എളുപ്പം ഉൾക്കൊള്ളിക്കുവാൻ ഈ ടാഗുകൾ സൗകര്യമൊരുക്കുന്നു.

2. **<CANVAS>:** ചിത്രം ഗ്രാഫിക്സ് എന്നിവ വരയ്ക്കുക. ചാർട്ടുകളും ഗ്രാഫുകളും തയാറാക്കുക, നമുക്കാവശ്യമുള്ള രീതിയിൽ ഗ്രാഫിക്സ് മാറ്റിയെടുക്കുക തുടങ്ങിയവയ്ക്കുള്ള ലളിതവും ശക്തവുമായ മാർഗമാണ് ഈ ടാഗ് നൽകുന്നത്.

3. **<HEADER>, <FOOTER>:** ഒരു ഡോക്മെന്റിന്റെയോ സെക്ഷന്റെയോ ഹെഡർ വ്യക്തമാകുവാൻ <HEADER> ടാഗും, ഫൂട്ടർ (അടിക്കുറുപ്പ്) വ്യക്തമാക്കാൻ <FOOTER> ടാഗും ഉപയോഗിക്കുന്നു. ഒരു ഫൂട്ടറിനുള്ളിൽ ഡോക്കുമെന്റിന്റെ രചയിതാവ് പകർപ്പവകാശത്തെക്കുറിച്ചുള്ള വിവരങ്ങൾ, ബന്ധപ്പെടുവാനുള്ള വിലാസം തുടങ്ങിയവ സൂചിപ്പിക്കുന്നു. നമുക്ക് ഒരേ ഡോക്കുമെന്റിൽ തന്നെ ധാരാളം <HEADER>, <FOOTER> ഇനങ്ങൾ ഉൾക്കൊള്ളിക്കാനാകും. ഒരു വെബ് പേജിനുള്ളിൽ ആർട്ടിക്കിളുകളും സെക്ഷനുകളും നിർമിക്കുവാൻ ഈ ടാഗുകൾ ഉപയോഗിക്കുന്നു.

4. **<ARTICLE>, <SECTION>:** ആർട്ടിക്കിളുകൾ സ്വതന്ത്രവും വ്യത്യസ്തവുമായ വാർത്താക്കുറിപ്പ് അല്ലെങ്കിൽ ബ്ലോഗ് പോസ്റ്റ് ആണ്. ഒരു പേജിനകത്ത് വ്യത്യസ്തങ്ങളായ വിഷയമേഖലകളാക്കുകയോ ആർട്ടിക്കിളിനെ പലഭാഗങ്ങളാക്കുകയോ ചെയ്യുന്നതിന് സെക്ഷൻ ഉപയോഗിക്കുന്നു. <ARTICLE>, <SECTION> എന്നീ ടാഗുകൾ ശരിയായ രീതിയിൽ ഉപയോഗിച്ചാൽ അത് സെർച്ച് എൻജിനിൽ ഈ പേജ് ലഭ്യതയ്ക്കുള്ള സാധ്യത വർധിക്കും.

5. **<OUTPUT>:** ഒരു സ്ക്രിപ്റ്റ് ഉപയോഗിച്ച് നടത്തുന്ന കണക്കുകൂട്ടലുകളുടെ ഫലം ലഭ്യമാക്കുന്നതിന് ഈ ടാഗ് ഉപയോഗിക്കുന്നു.

6. **<DETAILS>:** ഹെഡറിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ എക്സ്പാൻഡ് ചെയ്യുന്നതിനും കൊളാപ്സ് ചെയ്യുന്നതിനും വേണ്ട വാക്യങ്ങളുടെ ഭാഗം നിർണയിക്കുവാൻ ഈ ടാഗ് ഉപയോഗിക്കുന്നു.

7. **<FIGURE>** and **<FIGCAPTION>:** ചിത്രങ്ങൾ ഉൾക്കൊള്ളിക്കുവാൻ <FIGURE> ടാഗും, അതിന്റെ ശീർഷകം നൽകാൻ <FIGCAPTION> ടാഗും ഉപയോഗിക്കുന്നു. <FIGCAPTION> ടാഗ് <FIGURE> ടാഗിനുള്ളിലാണ് നൽകുന്നത്.

8. **<PROGRESS>, <METER>:** <PROGRESS>, <METER> എന്നീ ടാഗുകൾ ഒരേ പോലെയാണ്. ഒരു ജോലിയുടെ പുരോഗതിയെ സൂചിപ്പിക്കാൻ <PROGRESS> ടാഗ് ഉപയോഗിക്കുന്നു. ഇത് ഫയൽ അപ്‌ലോഡ് പോലുള്ള പ്രവർത്തനങ്ങളുടെ പുരോഗതി കാണിക്കുന്നതിന് ഉപയോഗിക്കുന്നു. എന്നാൽ <METER> ടാഗ് ഒരു പരിധിയിലുള്ള അളവ് മാത്രം നൽകുന്നതിന് ഉപയോഗിക്കുന്നു (ഉദാഹരണത്തിന് ഹാർഡ് ഡിസ്കിന്റെ ഉപയോഗം നൽകുന്നതിന്).

നമുക്കു സംഗ്രഹിക്കാം

ഈ അധ്യായത്തിലൂടെ HTML ന്റെ ചില മികച്ച സവിശേഷതകൾ നാം കണ്ടു കഴിഞ്ഞു. ഉള്ളടക്കവാക്യങ്ങൾ ആകർഷകമാക്കുന്നതിനായി വിവിധയിനം ലിസ്റ്റുകൾ നാം ചർച്ച ചെയ്തു. ഒരു വെബ് പേജിൽ ഓഡിയോ, വീഡിയോ, ചിത്രങ്ങൾ തുടങ്ങിയവ ഉൾപ്പെ

ടുത്തി അതിന്റെ മനോഹാരിത വർധിപ്പിക്കാം. കൂടാതെ ഒരു വെബ് പേജിൽ ഉൾപ്പെടുത്തുന്ന വിവിധതരം ഹൈപ്പർ ലിങ്കുകളെയും അവയുടെ പ്രാധാന്യത്തെയും കുറിച്ച് നാം മനസിലാക്കി. പല വിവരങ്ങളും വ്യക്തമായും എളുപ്പത്തിലും മനസിലാക്കുവാൻ കഴിയുന്ന വിധത്തിൽ രേഖപ്പെടുത്തുവാൻ പട്ടിക ഉപയോഗിച്ചു. പട്ടിക നിർമാണത്തിനാവശ്യമായ <TABLE>, <TR>, <TH>, <TD> എന്നീ ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും നാം ചർച്ച ചെയ്തു. പട്ടികയിലെ വരികളും നിരകളും ആവശ്യാനുസരണം ഒന്നിച്ച് ചേർക്കുന്ന Rowspan, Colspan എന്നീ ആട്രിബ്യൂട്ടുകളെയും നാം മനസിലാക്കി. ഒരേ സമയം വിവിധ വെബ് പേജുകളെ തുറന്നു കാണുവാൻ ബ്രൗസർ ജാലകത്തെ പല ഭാഗങ്ങളായി വിഭജിക്കാമെന്ന് നാം കണ്ടു. നെസ്റ്റഡ് ഫ്രെയിം സെറ്റ് എന്ന ആശയമുപയോഗിച്ച് ബ്രൗസർ ജാലകത്തെ വിവിധരീതികളിൽ വിഭജിക്കാമെന്നും നാം മനസ്സിലാക്കി. ബ്രൗസർ ജാലകത്തിലെ വിവിധ ഫ്രെയിമുകളെ ബന്ധിപ്പിക്കാൻ Target ആട്രിബ്യൂട്ടും നാം ചർച്ച ചെയ്തു. ഒരു വെബ് ഡോക്യുമെന്റിൽ ഉപയോക്താവിൽ നിന്നും വിലകൾ സ്വീകരിച്ച് സെർവറിലേക്ക് അയയ്ക്കുവാൻവേണ്ട ഫോം നിർമാണം നാം പഠിച്ചു. വിവിധ തരത്തിൽ ഡാറ്റ സ്വീകരിക്കുവാൻ വേണ്ടി ടെക്സ്റ്റ് ബോക്സ്, പാസ്‌വേർഡ്, റേഡിയോ ബട്ടൺ, ടെക്സ്റ്റ് ഏരിയ, സെലക്റ്റ് ബോക്സ് തുടങ്ങിയ വിവിധ കൺട്രോളുകൾ നാം പരിചയപ്പെട്ടു. ക്ലയന്റ് സൈഡ് സ്ക്രിപ്റ്റ്, സർവർസൈഡ് സ്ക്രിപ്റ്റ് എന്നിവയെ കുറിച്ച് ചില കാര്യങ്ങൾ സൂചിപ്പിച്ചു. ഈ വിഷയങ്ങൾ ചർച്ച ചെയ്യുന്ന അടുത്ത അധ്യായങ്ങളിലേക്കുള്ള ചവിട്ടുപടിയായി ഈ അധ്യായത്തെ കരുതാം.

നമുക്കു പരിശീലിക്കാം

1. താഴെ കൊടുത്തിരിക്കുന്ന വിശദാംശങ്ങളും സവിശേഷതകളും ഉൾക്കൊള്ളിച്ച് കേരളത്തിലെ ഏതെങ്കിലും ഒരു ജില്ലയുടെ വെബ് പേജ് തയാറാക്കാനുള്ള HTML കോഡ് എഴുതുക.
   - ടെക്സ്റ്റ് ഫോർമാറ്റിങ് ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും ഉപയോഗിച്ച് ഒരു ശീർഷകവും അതിനു താഴെ മൂന്ന് വാചകത്തിൽ കുറയാതെയുള്ള ഒരു ഖണ്ഡികയിൽ ജില്ലയെക്കുറിച്ചുള്ള വിവരണവും നൽകുക.
   - ജില്ലയിലെ പ്രധാനപ്പെട്ട വിനോദസഞ്ചാര കേന്ദ്രങ്ങളുടെ ലിസ്റ്റ്.
2. നിങ്ങളുടെ സ്കൂളിന്റെ വെബ് പേജ് രൂപകൽപന ചെയ്യാൻ ആവശ്യമായ HTML കോഡ് താഴെപ്പറയുന്ന വിവരങ്ങൾ/സവിശേഷതകൾ ഉൾപ്പെടുത്തി എഴുതുക.
   - ടെക്സ്റ്റ് ഫോർമാറ്റിങ് ടാഗുകളും അവയുടെ ആട്രിബ്യൂട്ടുകളും ഉപയോഗിച്ച് ഒരു ശീർഷകവും അതിനനുബന്ധമായി മൂന്ന് വാചകങ്ങളിൽ കുറയാതെ നിങ്ങളുടെ സ്കൂളിന്റെ പ്രത്യേകതകൾ എടുത്തുപറയുന്ന ഖണ്ഡികയും.
   - നിങ്ങളുടെ സ്കൂളിലെ ഏതെങ്കിലും പഠനത്തോടൊപ്പമുള്ള പ്രവർത്തനങ്ങളുടെ ലിസ്റ്റ്. ഉദാഹരണം : സ്കൗട്ട്, ഗൈഡ്, എൻ.എസ്.എസ്.

3. താഴെപ്പറയുന്ന വിവരങ്ങൾ വ്യക്തമാക്കുന്ന വെബ് പേജിന്റെ HTML കോഡ് എഴുതുക.

Components of a Computer

- **Hardware**
  1. I/O Devices
  2. RAM
  3. Hard Disk & DVD Drive
- **Software**
  1. Operating System
  2. Application Programs

4. കേരളത്തിന്റെ ദേശീയോത്സവമായ ഓണത്തിന്റെ എടുത്തുപറയത്തക്ക പ്രത്യേകതകൾ വർണിക്കുന്ന വെബ് പേജിനുവേണ്ടി താഴെപ്പറയുന്ന സവിശേഷതകൾ ഉൾപ്പെടുത്താനുള്ള HTML കോഡ് എഴുതുക.

- ആകർഷകമായ ഫോണ്ടിലുള്ള ശീർഷകം.
- വള്ളംകളിയുടെ ചിത്രം പേജിന്റെ പശ്ചാത്തലമാക്കുക.
- ഓണത്തോട് അനുബന്ധിച്ച് നടത്തപ്പെടുന്ന പരമ്പരാഗത ആചാരങ്ങളായ അത്തപ്പൂക്കളം, തുമ്പിതുള്ളൽ, വള്ളംകളി, കുമ്മാട്ടിക്കളി, പുലികളി എന്നിവ ഏതെങ്കിലും രണ്ടിലേക്ക് ഇന്റേണൽ ലിങ്ക് നൽകുക.

5. നിങ്ങളുടെ സ്കൂളിന്റെ ഹൈസ്കൂൾ, ഹയർസെക്കന്ററി വിഭാഗങ്ങളുടെ വിവരങ്ങൾ അടങ്ങുന്ന രണ്ട് വെബ് പേജുകൾ നിർമിക്കുക. അടുത്ത വെബ് പേജ് നിർമിച്ച് തിരശ്ചീനമായ രണ്ട് ഭാഗങ്ങളാക്കി തിരിക്കുന്നു. ഒന്നാമത്തെ ഫ്രെയിമിൽ സ്കൂളിനെക്കുറിച്ചുള്ള ആമുഖവും രണ്ട് ലിങ്കുകളുമാണ് (ഹൈസ്കൂൾ, ഹയർസെക്കന്ററി) നൽകേണ്ടത്. ഉപയോക്താവ് ലിങ്കിൽ ക്ലിക്ക് ചെയ്യുമ്പോൾ യഥാക്രമം അവയുടെ വെബ് പേജുകൾ രണ്ടാമത്തെ ഫ്രെയിമിൽ ലഭ്യമാക്കുന്നതിന് വേണ്ട HTML കോഡ് തയാറാക്കുക.

6. ചുവടെ ചേർക്കുന്ന ടേബിളും അതിന് താഴെ കൊടുത്തിരിക്കുന്ന കേരള പോലീസിന്റെ വെബ്സൈറ്റിലേക്കുള്ള എക്സ്റ്റേണൽ ലിങ്ക് എന്നിവ പ്രദർശിപ്പിക്കുന്ന വെബ് പേജിന് ആവശ്യമായ HTML കോഡും തയാറാക്കുക.

Road Accidents in Kerala during 2012 - 2014

Year	Total Number of		
	Cases	Persons Killed	Persons Injured
2012	36174	4286	41915
2013	35215	4258	40346
2014	36282	4049	41096

Data Source: www.keralapolice.org

7. ചുവടെ ചേർക്കുന്ന രീതിയിലുള്ള ഒരു അപേക്ഷ ഫോം തയാറാക്കുന്നതിനാവശ്യമായ HTML കോഡ് എഴുതുക.

APPLICATION FOR THE BEST STUDENT AWARD

Name: [ ] Sex: Male ○ Female ○

Class & Division: [Select ▼]

Total Grade Point in Class XI: [ ]

Average Grade Point in Termly Exams in Class XII: [ ]

Cocurricular Activities:

NCC ☐ NSS ☐ Sports ☐ Arts ☐ Literary ☐

Other Achievements: [ ]

നമുക്കു വിലയിരുത്താം

1. ചുവടെ ചേർക്കുന്നവയിൽ തെറ്റ് കണ്ടുപിടിച്ച് തിരുത്തി എഴുതുക.

a.
```
<UL Type = "i" Start = 3>
```

b.
```
<IMG Src = "Myschool.jpg" Size  = "50" >
```

c.
```
<HTML>
     <HEAD><TITLE><HEAD></TITLE>
     <BODY> this is the body of the HTML
       document</BODY>
</HTML>
```

2. രോഹിത് HTML ഉപയോഗിച്ച് നിർമിച്ച ടേബിളിൽ ബോർഡർ കാണുവാൻ സാധിക്കുന്നില്ല. ഇതിന്റെ കാരണം എന്തായിരിക്കും?

3. ലിങ്ക് ചെയ്ത പേജ് ഒരു ഫ്രെയിമിനുള്ളിൽ പ്രദർശിപ്പിക്കുവാൻ <A> ടാഗിന്റെ __________ ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നു.

4. നിങ്ങളുടെ ആത്മാർഥ സുഹൃത്തുക്കളുടെ ഒരു ലിസ്റ്റ് തയാറാക്കുവാൻ നിങ്ങളുടെ കമ്പ്യൂട്ടർ അധ്യാപകൻ/അധ്യാപിക ആവശ്യപ്പെടുന്നു എന്ന് കരുതുക. നിങ്ങൾ ഏത് ടാഗിനാണ് പരിഗണന നൽകുക. ഇതിനുവേണ്ട HTML കോഡ് ഭാഗം തയാറാക്കുക.

5. <OL> ടാഗിലെ type ആട്രിബ്യൂട്ടിന് നൽകാവുന്ന വിലകളുടെ പേര് എഴുതുക.

6. <UL> ടാഗിന്റെ ആട്രിബ്യൂട്ടുകൾ എഴുതുക.

7. ഇംഗ്ലീഷ് ഭാഷയിലെ വലിയ അക്ഷരങ്ങൾ ഉപയോഗിച്ച് ക്രമനമ്പരോട് കൂടിയ ലിസ്റ്റ് നിങ്ങൾ എങ്ങനെ നിർമിക്കും?
8. ഇംഗ്ലീഷ് ഭാഷയിലെ ചെറിയ അക്ഷരങ്ങൾ ഒരു ലിസ്റ്റിലെ ഇനങ്ങളുടെ ക്രമനമ്പർ ആയി പ്രദർശിപ്പിക്കണം എന്ന് കരുതുക. എങ്ങനെ ഇത് സാധ്യമാക്കാം?
9. ക്രമനമ്പർ 6 മുതൽ ആരംഭിക്കുന്ന ഒരു ലിസ്റ്റ് നമുക്ക് എങ്ങനെ നിർമിക്കാം?
10. താഴെ കൊടുത്തിരിക്കുന്ന വിവരങ്ങൾ ഉൾക്കൊള്ളിച്ചു വെബ് പേജിനു വേണ്ട <HTML>കോഡ് എഴുതുക.

ABC Pvt. Ltd.
Kerala

1. Health Care
2. Baby Products
   - Toys
   - Dress
3. Ladies Wear
   - Kurthas
   - Jeans

11. വരുൺ ഒരു വെബ് പേജ് നിർമിക്കുന്നു. അതിൽ 'sample' എന്ന വാക്ക് ഹൈപ്പർ ലിങ്കായി പ്രദർശിപ്പിച്ച് /home/scert/Desktop-ൽ സംഭരിക്കുന്ന *sample.html* എന്ന ഫയലിലേക്ക് ലിങ്ക് നൽകുവാനുള്ള HTML കോഡ് ഭാഗം എഴുതുക.
12. `Noshade` എന്ന ആട്രിബ്യൂട്ടുള്ള ടാഗിന്റെ പേര് എഴുതുക.
13. സുനിലിന് അയാളുടെ സ്വന്തം ആവശ്യങ്ങൾക്കായി ഒരു ഇ-മെയിൽ ലിങ്ക് ഉൾപ്പെടുന്ന വെബ് പേജ് നിർമിക്കണം. നിങ്ങൾക്ക് ഇ-മെയിൽ ലിങ്കിനുവേണ്ട പ്രോട്ടോകോൾ നിർദ്ദേശിക്കാമോ?
14. സഹീറിന് അയാളുടെ വെബ് പേജിൽ നിന്നും www.gmail.com എന്ന വെബ്സൈറ്റിലേക്ക് ബന്ധിപ്പിക്കണം. ഇതിനാവശ്യമായ ടാഗും അവയുടെ ആട്രിബ്യൂട്ടുകളും എഴുതുക.
15. HTML ൽ ലഭ്യമായ രണ്ട് തരം ഹൈപ്പർ ലിങ്കുകളുടെ സ്വഭാവ സവിശേഷതകൾ പ്രതിപാദിക്കുക.
16. `Cellspacing`, `Cellpadding` എന്നിവയുടെ വ്യത്യാസങ്ങൾ എഴുതുക.
17. `Text`, `Textarea` എന്നീ കൺട്രോളുകളുടെ വ്യത്യാസങ്ങൾ എഴുതുക.
18. `<FORM>` ടാഗിന്റെ പ്രധാന ആട്രിബ്യൂട്ടുകൾ `Action` നും .... ഉം ആകുന്നു.
19. തെറ്റോ ശരിയോ എന്ന് പറയുക.
    a. `<TABLE>` ടാഗിലെ `Align` ആട്രിബ്യൂട്ടിന്റെ തനത് വില `center` ആകുന്നു.
    b. `<FRAME>` ഒരു കണ്ടെയ്നർ ടാഗ് ആകുന്നു.
    c. ഉപയോക്താവ് ഒരു നിശ്ചിത ഫ്രെയിമിലെ ബോർഡറിന്റെ വലുപ്പ വ്യത്യാസം വരുത്തുന്നത് തടയുവാൻ സ്ക്രോളിങ് ഉപയോഗിക്കുന്നു.

20. ഏത് ടാഗാണ് ഒരു ബ്രൗസർ ജാലകത്തെ വിവിധ ഭാഗങ്ങളാക്കി വിഭജിക്കുന്നത്?

21. <FRAME> ടാഗിന്റെ പ്രധാന ആട്രിബ്യൂട്ടുകൾ എഴുതുക.

22. ഒരു വെബ് പേജിനെ ലംബമായ രണ്ട് ഭാഗങ്ങളാക്കി തിരിക്കുന്ന ഫ്രെയിം സെറ്റ് നിർമ്മിക്കുക. നിങ്ങൾക്ക് ഇഷ്ടപ്പെട്ട ഫുട്ബോൾ കളിക്കാരുടെ പേര് ഇടതുഭാഗത്തും, സെലക്ട് ചെയ്യുന്ന കളിക്കാരനെക്കുറിച്ചുള്ള ലഘു വിവരണം വലതുഭാഗത്തും പ്രദർശിപ്പിക്കണം. ഇതിനാവശ്യമായ കോഡ് എഴുതുക.

23. ചുവടെ ചേർക്കുന്ന HTML കോഡ് പരിശോധിച്ചതിന് ശേഷം അതിന്റെ ഔട്ട്പുട്ട് എന്താകും എന്ന് എഴുതുക.

```
<HTML>
<HEAD><TITLE>A simple table</TITLE></HEAD>
<BODY>
<TABLE border="1" Cellspacing= "1" Cellpadding= "10">
  <TR><TD> 1 </TD><TD> 2 </TD><TD> 3 </TD></TR>
  <TR><TD> 4 </TD><TD> 5 </TD><TD> 6 </TD></TR>
  <TR><TD> 7 </TD><TD> 8 </TD><TD> 9 </TD></TR>
</TABLE>
</BODY>
</HTML>
```

24. താഴെ കൊടുത്തിരിക്കുന്ന പട്ടിക പ്രദർശിപ്പിക്കുവാൻ വേണ്ട HTML കോഡ് എഴുതുക.

1	2	3
4	5	6
7		

25. നിങ്ങളുടെ ഇമെയിൽ, വിലാസം, ഫോൺ നമ്പർ, പാസ്‌വേഡ് തുടങ്ങിയവ സ്വീകരിക്കുവാൻ വേണ്ട ഫോം നിർമ്മിക്കുന്നതിനാവശ്യമായ HTML കോഡ് എഴുതുക.

26. നിങ്ങൾക്ക് ഇഷ്ടപ്പെട്ട മൂന്ന് ബൈക്കുകളുടെ പേര് അടങ്ങുന്ന ലിസ്റ്റ് ഒരു ഫ്രെയിമിലും, അതിന്റെ ചിത്രവും, വിവരണവും അടുത്ത ഫ്രെയിമിൽ ലിങ്ക് ചെയ്തും പ്രദർശിപ്പിക്കുന്നതിനുള്ള HTML കോഡ് എഴുതുക.

27. ചുവടെ ചേർക്കുന്ന HTML കോഡ് ഉപയോഗിച്ച് രാജ് നിർമിച്ച വെബ് പേജിലെ പട്ടിക ശരിയായ രൂപത്തിൽ കാണുവാൻ സാധിക്കുന്നില്ല. ഇതിന്റെ കാരണം കണ്ടെത്തി ശരിയായ രീതിയിൽ എഴുതുക.

```
<HTML>
<HEAD><TITLE> My Page </TITLE></HEAD>
<BODY>
<TABLE><TR><TH>Roll No. </TH><TH> Name </TH></TR>
<TR><TD>1 </TD><TD> Huda </TD></TR>
<TR><TD>2 </TD><TD>Bincy</TD></TR>
</TABLE>
</BODY>
</HTML>
```

R9N7R1

6 ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ചുള്ള ക്ലൈന്റ് സൈഡ് സ്ക്രിപ്റ്റിങ്

പ്രധാന പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പഠനം പൂർത്തിയാകുന്നതോടെ പഠിതാവ് ആർജിക്കേണ്ട പഠനനേട്ടങ്ങൾ

- ക്ലൈന്റ് വശത്തെയും സെർവർ വശത്തെയും സ്ക്രിപ്റ്റിംഗ് ഭാഷകളുടെ ഉപയോഗം തിരിച്ചറിയുന്നു.
- ക്ലൈന്റ് സൈസ് സ്ക്രിപ്റ്റിംഗ് ഭാഷകളുടെ ആവശ്യകത വിശദീകരിക്കുന്നു.
- ക്ലൈന്റ് സൈസ് സ്ക്രിപ്റ്റിംഗ് ഭാഷയായ ജാവാസ്ക്രിപ്റ്റിന്റെ പ്രാധാന്യം തിരിച്ചറിയുന്നു.
- വെബ്പേജിൽ ജാവാസ്ക്രിപ്റ്റ് ഫംഗ്ഷൻ ഉപയോഗിക്കുന്നു
- ജാവാസ്ക്രിപ്റ്റിലെ വിവിധ ഡാറ്റ ഇനങ്ങൾ വിശദീകരിക്കുന്നു.
- ജാവാസ്ക്രിപ്റ്റിൽ ശരിയായ വേരിയബിളുകൾ ഉപയോഗിക്കുന്നു.
- ജാവാസ്ക്രിപ്റ്റിൽ അനുയോജ്യമായ കൺട്രോൾ സ്ട്രക്ച്ചറുകൾ ഉപയോഗിക്കുന്നു.
- ജാവാസ്ക്രിപ്റ്റിൽ അനുയോജ്യമായ അന്തർനിർമിത ഫംഗ്ഷനുകൾ ഉപയോഗിക്കുന്നു.
- ജാവാസ്ക്രിപ്റ്റിലെ ഡോക്യുമെന്റ് ഘടകങ്ങൾ ഉപയോഗിക്കുന്ന രീതി വിവരിക്കുന്നു.
- ടെക്സ്റ്റ് ബോക്സുകളിലേയും കോംബോ ബോക്സുകളിലേയും വിലകൾ കൈകാര്യം ചെയ്യുന്നതിനുള്ള ജാവാസ്ക്രിപ്റ്റ് ഫങ്ഷനുകൾ നിർമിക്കുന്നു.

മുൻ അധ്യായങ്ങളിൽ ടെക്സ്റ്റുകളും ഗ്രാഫിക്സുകളും ഉൾപ്പെടുന്ന വിവിധ തരം വെബ് പേജുകൾ നിർമിക്കുവാൻ നാം പഠിച്ചു. ഈ ഇന്റർനെറ്റ് യുഗത്തിൽ നമ്മളിൽ പലരും വിവിധ ആവശ്യങ്ങൾക്കായി അനവധി വെബ്സൈറ്റുകൾ സന്ദർശിക്കാറുണ്ട്. നമുക്ക് പരിചിതമല്ലാത്ത അനേകം സവിശേഷതകൾ ഇത്തരം വെബ് പേജുകളിൽ അടങ്ങിയിരിക്കുന്നു. അതുകൊണ്ടുതന്നെ ഇത്തരം വെബ് പേജുകൾ നിർമിക്കുന്നതിനായി സ്ക്രിപ്റ്റിംഗ് ഭാഷകളിലുള്ള അറിവ് ആവശ്യമാണ്. ക്ലൈന്റ് സൈഡിലും സെർവർ സൈഡിലും വിവിധയിനം സ്ക്രിപ്റ്റിംഗ് ഭാഷകൾ ഉപയോഗിക്കുന്നുണ്ട്. ക്ലൈന്റ് സൈഡ് സ്ക്രിപ്റ്റിംഗ് ഭാഷകളായ ജാവാസ്ക്രിപ്റ്റ്, വിബി സ്ക്രിപ്റ്റ് എന്നിവയിൽ ജാവാസ്ക്രിപ്റ്റാണ് കൂടുതലായി ഉപയോഗിക്കുന്നത്. കാരണം വിബി സ്ക്രിപ്റ്റിനെ അപേക്ഷിച്ച് ജാവാസ്ക്രിപ്റ്റിനെയാണ് കൂടുതൽ ബ്രൗസറുകളും പിന്തുണയ്ക്കുന്നത്. ഇന്റർനെറ്റിൽ അനേകം ആളുകൾ വ്യത്യസ്ത ബ്രൗസറുകൾ ഉപയോഗിക്കുന്നതിനാൽ വെബ് പേജുകൾ കഴിയുന്നത്ര എല്ലാ ബ്രൗസറുകൾക്കും അനുയോജ്യമായ രീതിയിൽ നിർമിക്കേണ്ടതുണ്ട്. ഒരു വെബ് പേജിൽ ജാവാസ്ക്രിപ്റ്റ് എങ്ങനെ ഉപയോഗിക്കാമെന്ന് ഈ അധ്യായത്തിൽ മനസ്സിലാക്കാം. നമുക്ക് C++ പരിചയമുള്ളതിനാൽ ജാവാസ്ക്രിപ്റ്റ് മനസ്സിലാക്കുവാൻ വളരെ എളുപ്പമാണ്. കാരണം C++ ന്റെ അതേ വാക്യഘടനയാണ് ജാവാസ്ക്രിപ്റ്റ് പിന്തുടരുന്നത്.

നെറ്റ്സ്കേപ്പ് ബ്രൗസറിനുവേണ്ടി 'ബ്രെൻഡൻ ഇച്ച്' ആണ് ജാവാസ്ക്രിപ്റ്റ് വികസിപ്പിച്ചത്. ജാവാസ്ക്രിപ്റ്റിന്റെ ആദ്യ നാമം 'മോച്ച' എന്നായിരുന്നു. 1995-ൽ നെറ്റ് സ്കേപ്പ് ബ്രൗസർ പതിപ്പ് 2.0 ത്തിൽ ഉൾപ്പെടുത്തിയപ്പോൾ ഇതിന്റെ പേര് 'ജാവാസ്ക്രിപ്റ്റ്' എന്നാക്കിമാറ്റി. ആദ്യകാലങ്ങളിൽ, നെറ്റ്സ്കേപ്പ് ബ്രൗസർ മാത്രമേ ജാവാസ്ക്രിപ്റ്റിന് പിന്തുണ നൽകിയിരുന്നുള്ളൂ. എന്നാൽ ജാവാ സ്ക്രിപ്റ്റിന്റെ വിപുലമായ പ്രചാരം കാരണം, ഇന്റർനെറ്റ് എക്സ്പ്ലോറർ 1996-ൽ ജാവാസ്ക്രിപ്റ്റിന് പിന്തുണ നൽകി. ഇപ്പോൾ, ലോകത്തിലെ മിക്ക ബ്രൗസറുകളും ജാവാസ്ക്രിപ്റ്റിനെ പിന്തുണയ്ക്കുന്നു.

ബ്രെൻഡൻ ഇച്ച്

6.1 ജാവസ്ക്രിപ്റ്റ് ആരംഭിക്കാം (Getting started with JavaSript)

നാലാമത്തെ അധ്യായമായ വെബ് ടെക്നോളജിയിൽ ക്ലൈന്റ് സൈഡിലും സെർവർ സൈഡിലും ഉള്ള സ്ക്രിപ്റ്റിംഗ് ഭാഷകളുടെ ഉപയോഗം നാം ചർച്ചചെയ്തു. ക്ലൈന്റ് സൈഡിലെ ഡാറ്റയുടെ സാധുത ഉറപ്പുവരുത്തുന്നതിനായി ക്ലൈന്റ് സൈഡ് സ്ക്രിപ്റ്റിംഗ് ഭാഷകൾ ഉപയോഗിക്കുന്നു. ഇത് സെർവറിലെ നെറ്റ് വർക്ക് ട്രാഫിക്കും ജോലി ഭാരവും കുറയ്ക്കുന്നു. സെർവർ സൈഡ് സ്ക്രിപ്റ്റിംഗ് ഭാഷകൾ സെർവറിൽ പ്രവർത്തിക്കുന്നു. ഇതിലൂടെ നിർമിക്കപ്പെട്ട വെബ് പേജ് ക്ലൈന്റ് ബ്രൗസറിലേക്ക് തിരിച്ചയക്കുന്നു. ഡാറ്റാ ബേസിന്റെ രൂപത്തിൽ വലിയ അളവ് ഡാറ്റ സെർവറിൽ സംഭരിക്കുന്നു. അതിനാൽ സെർവർ സൈഡിലെ സ്ക്രിപ്റ്റിംഗ് ഭാഷ വ്യത്യസ്ത ആവശ്യങ്ങൾക്ക് വേണ്ടി ഈ ഡാറ്റബേസുമായി സംവദിക്കേണ്ടതായി വരും. എന്നാൽ ഒരു ക്ലൈൻറ് സൈഡ് സ്ക്രിപ്റ്റിംഗ് ഭാഷ സെർവർ ഡാറ്റബേസുമായി നേരിട്ട് ബന്ധപ്പെടുന്നില്ല. അതുകൊണ്ടുതന്നെ ക്ലൈന്റ് സൈഡിലും സെർവർ സൈഡിലും ഉപയോഗിക്കുന്ന ഭാഷകളും കമാൻഡുകളും വ്യത്യസ്തങ്ങളാണ്.

ഈ അധ്യായത്തിൽ ക്ലൈന്റ് വശം സ്ക്രിപ്റ്റിംഗ് ഭാഷ എന്ന നിലയിൽ നമുക്ക് ജാവാ സ്ക്രിപ്റ്റിന്റെ അടിസ്ഥാന വിവരങ്ങൾ മനസിലാക്കാം. ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ച് HTML പേജിലെ വിവിധ ഭാഗങ്ങളിൽ പ്രോഗ്രാമുകൾ ഉൾപ്പെടുത്താൻ കഴിയും. ഇതിനായി <SCRIPT> ടാഗ് ഉപയോഗിക്കുന്നു.

<SCRIPT> Tag

ഒരു HTML പേജിൽ സ്ക്രിപ്റ്റിംഗ് കോഡ് ഉൾപ്പെടുത്താൻ <SCRIPT> ടാഗ് ഉപയോഗിക്കുന്നു. <SCRIPT> ടാഗിലെ 'language' ആട്രിബ്യൂട്ട് ഉപയോഗിക്കുന്നത് സ്ക്രിപ്റ്റിംഗ് ഭാഷയുടെ പേര് വ്യക്തമാക്കുന്നതിനാണ്. ഇവിടെ ക്ലൈന്റ് സൈഡ് സ്ക്രിപ്റ്റിംഗ് ഭാഷ യായി ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിക്കുന്നതിനുവേണ്ടി <SCRIPT> ടാഗിലെ 'language' ആട്രി ബ്യൂട്ടിന്റെ വില 'JavaScript' എന്ന് നൽകണം.

ഒരു HTML പേജിൽ <SCRIPT> ടാഗ് ഉപയോഗിക്കുന്നത് താഴെ കാണുന്നത് പോലെ യാണ്.

The <SCRIPT> tag can be used in an HTML page as follows.

```
<SCRIPT Language= "JavaScript">
```

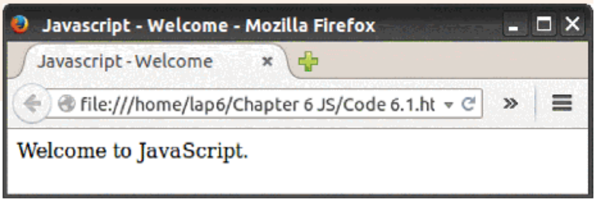
```
......................
......................
</SCRIPT>
```

ജാവാസ്ക്രിപ്റ്റിലെ ഐഡന്റിഫയറുകൾ കേസ് സെൻസിറ്റീവ് ആണ്. ജാവാസ്ക്രിപ്റ്റിൽ സാധാരണയായി ഐഡന്റിഫയറുകൾക്ക് കാമൽകേസ് (camelCase) പേരുകളാണ് ഉപയോഗിക്കുക. ഉദാഹരണത്തിന് firstName, checkData തുടങ്ങിയവ. ഒന്നിൽകൂടുതൽ വാക്കുകൾ ഒറ്റവാക്കായി ഉപയോഗിക്കുന്ന ഒരു നാമകരണ രീതിയാണ് കാമൽകേസ്. ഓരോ വാക്കിന്റെയും ആദ്യ അക്ഷരം വലിയ അക്ഷരം ഉപയോഗിക്കുന്നതിനെ 'അപ്പർകാമൽകേസ്' എന്ന് വിളിക്കുന്നു. ആദ്യത്തെ വാക്കൊഴികെ മറ്റെല്ലാ വാക്കിന്റെയും ആദ്യ അക്ഷരം വലിയ അക്ഷരം ഉപയോഗിക്കുന്നതിനെ 'ലോവർകാമൽകേസ്' എന്നു വിളിക്കുന്നു. കാമൽകേസ് രീതി വാക്കുകൾ വായിക്കുവാൻ എളുപ്പമുള്ളതാക്കുന്നു.

ഇനി നമുക്ക് താഴെകൊടുത്തിരിക്കുന്ന HTML ഫയൽ പരിഗണിക്കാം

ഉദാഹരണം 6.1 : ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ച് ഒരു വെബ് പേജ് നിർമ്മാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Welcome</TITLE> </HEAD>
<BODY>
   <SCRIPT Language= "JavaScript">
      document.write("Welcome to JavaScript.");
   </SCRIPT>
</BODY>
</HTML>
```



ചിത്രം 6.1: ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിക്കുന്ന വെബ് പേജ്

മുകളിലുള്ള കോഡ് ഏതെങ്കിലും ടെക്സ്റ്റ് എഡിറ്ററിൽ ടൈപ്പ് ചെയ്യാം. മുൻ അധ്യായങ്ങളിൽ HTML പേജുകൾ സൃഷ്ടിക്കുന്നതിന് നിങ്ങൾ ഉപയോഗിച്ചിരിക്കുന്ന ജീനി എഡിറ്ററിലും ഇത് ടൈപ്പ് ചെയ്യാവുന്നതാണ്. code6.html എന്ന പേരിൽ ഫയൽ സൂക്ഷിക്കുക. HTML പേജിൽ ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ചിട്ടുണ്ടെങ്കിലും അത് സേവ് ചെയ്തിരിക്കുന്നത്. html എക്സ്റ്റൻഷനിലാണ് എന്നത് പ്രത്യേകം ശ്രദ്ധിക്കേണ്ടതാണ്. ഇനി ഏതെങ്കിലും ഒരു ബ്രൗസറിൽ നമ്മൾ സേവ് ചെയ്ത HTML ഫയൽ തുറക്കുക. ചിത്രം 6.1 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ ഒരു വെബ് പേജ് ലഭിക്കും. 'document.write' എന്ന പ്രസ്താവന ചെറിയ അക്ഷരങ്ങളിലാണ് എഴുതിയിരിക്കുന്നത് എന്നതും ശ്രദ്ധിക്കുക, കാരണം ജാവാസ്ക്രിപ്റ്റ് കേസ് സെൻസിറ്റീവ് സ്ക്രിപ്റ്റിംഗ് ഭാഷയാണ്. സ്ക്രിപ്റ്റിലെ കീവേഡുകൾ എല്ലാം ചെറിയ അക്ഷരത്തിലാണ് എഴുതേണ്ടത്.

മുകളിൽ കൊടുത്തിരിക്കുന്ന HTML കോഡിൽ, document.write() എന്നത് ഒരു ജാവാസ്ക്രിപ്റ്റ് നിർദേശം ആണ്. അത് HTML പേജിന്റെ ബോഡിക്കകത്ത് ഒരു ടെക്സ്റ്റ് ഉൾ

ക്കൊള്ളിക്കുവാൻ ഉപയോഗിക്കുന്നു. അതായത്, മുകളിൽ നൽകിയിരിക്കുന്ന HTML കോഡ് നിർമിക്കുന്ന അതേ വെബ്പേജ് താഴെ നൽകിയിരിക്കുന്ന ജാവാ സ്ക്രിപ്റ്റ് ഇല്ലാത്ത HTML കോഡും നിർമ്മിക്കുന്നു.

ഉദാഹരണം 6.2 : HTML ഉപയോഗിച്ച് ഒരു വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Welcome</TITLE> </HEAD>
<BODY>
   Welcome to JavaScript.
</BODY>
</HTML>
```

മേൽ വിവരിച്ച ഉദാഹരണങ്ങൾ 6.1 ഉം 6.2 ഉം താരതമ്യം ചെയ്യാം. രണ്ടാമത്തെ HTML കോഡിൽ, "Welcome to JavaScript" എന്നത് ബോഡി ഭാഗത്തിൽ നേരിട്ട് എഴുതിയിരിക്കുന്നു. ആദ്യത്തെ കോഡിൽ "Welcome to JavaScript" എന്നത് ജാവാസ്ക്രിപ്റ്റ് രീതിയായ document.write( ) ഉപയോഗിച്ച് ബോഡി ഭാഗത്ത് ഉൾപ്പെടുത്തിയിരിക്കുന്നു.

യഥാർഥത്തിൽ 'document' എന്നത് വെബ് പേജിന്റെ ബോഡി വിഭാഗത്തെ പ്രതിനിധാനം ചെയ്യുന്നു. അതിനാൽ document.write() എന്ന ജാവാസ്ക്രിപ്റ്റ് ഫങ്ഷൻ വെബ് പേജിലെ ബോഡി ഭാഗത്തിൽ ഒരു ടെക്സ്റ്റ് ഉൾപ്പെടുത്തും. C++ പോലെ സ്ക്രിപ്റ്റിലെ ഓരോ പ്രസ്താവനയും ഒരു അർദ്ധവിരാമത്തിൽ (;) അവസാനിക്കുന്നു എന്നത് ശ്രദ്ധിക്കുക.

`<SCRIPT Language= "JavaScript">` എന്ന HTML നിർദേശം സൂചിപ്പിക്കുന്നത് തുടർന്നുവരുന്നത് ജാവാസ്ക്രിപ്റ്റ് കോഡ് ആണെന്നുള്ളതാണ്. ഇനി നമുക്ക് ജാവാസ്ക്രിപ്റ്റിനെ ബ്രൗസർ എങ്ങനെ കൈകാര്യം ചെയ്യുന്നുവെന്ന് കാണാം. എല്ലാ ബ്രൗസറിലും ജാവാസ്ക്രിപ്റ്റ് എൻജിൻ ഉണ്ട്. പ്രവർത്തിപ്പിക്കുന്ന സമയത്ത് സ്ക്രിപ്റ്റ് കോഡിനെ ജാവാസ്ക്രിപ്റ്റ് എൻജിൻ വ്യാഖ്യാനിക്കും. ജാവാസ്ക്രിപ്റ്റ് കോഡ് നിർവഹണത്തിനുള്ള വിർച്വൽ മെഷിനാണ് (virtual machine) ജാവാസ്ക്രിപ്റ്റ് എഞ്ചിൻ. ബ്രൗസർ, ജാവാസ്ക്രിപ്റ്റ് കോഡ് കാണുമ്പോൾ അതിനെ സ്ക്രിപ്റ്റ് എൻജിനിലൂടെ കടത്തിവിടുന്നു. ജാവാസ്ക്രിപ്റ്റ് എഞ്ചിൻ കോഡിനെ പ്രവർത്തിപ്പിക്കുന്നു. ഒരു HTML പേജിൽ ജാവാസ്ക്രിപ്റ്റ് കോഡ് ഇല്ലെങ്കിൽ ബൗസറിന് ഒറ്റയ്ക്ക് തന്നെ HTML പേജ് പ്രദർശിപ്പിക്കാൻ കഴിയും. ജാവാസ്ക്രിപ്റ്റ് കോഡ് ഉണ്ടെങ്കിൽ ജാവാസ്ക്രിപ്റ്റ് എഞ്ചിന്റെ സഹായത്തോടുകൂടി മാത്രമേ HTML പേജ് പ്രദർശിപ്പിക്കുവാൻ കഴിയൂ. ഇപ്രകാരം ജാവാസ്ക്രിപ്റ്റ് ഇല്ലാത്ത HTML ഫയലുകൾ എപ്പോഴും ജാവാസ്ക്രിപ്റ്റ് ഉള്ളവയെക്കാൾ വേഗത്തിൽ പ്രദർശിപ്പിക്കാൻ കഴിയും.

ഉദാഹരണം 6.3 ൽ ജാവാസ്ക്രിപ്റ്റ് കോഡും HTML ടാഗുകളും ഇടകലർത്തി നൽകിയിരിക്കുന്നു. ഇതിന്റെ ഔട്ട്പുട്ട് വെബ്പേജ് ചിത്രം 6.2 ൽ നൽകിയിരിക്കുന്നു.

ഉദാഹരണം 6.3: ഹെഡ്ഡിങ് ടാഗുകൾ ഉൾക്കൊള്ളുന്ന ഒരു വെബ് പേജ് നിർമ്മാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Welcome</TITLE> </HEAD>
<BODY>
   <H1>
   <SCRIPT Language= "JavaScript">
      document.write("This is in H1 Head");
   </SCRIPT>
   </H1>
   <BR>
   <H2>
   <SCRIPT Language= "JavaScript">
      document.write("This is in H2 Head");
   </SCRIPT>
   </H2>
</BODY>
</HTML>
```

ചിത്രം 6.2: ഹെഡ്ഡിംഗ് ടാഗുകൾ അടങ്ങുന്ന വെബ് പേജ്

മുകളിൽ നൽകിയിരിക്കുന്ന HTML ടാഗുകൾക്കിടയിൽ ഒന്നിലധികം തവണ സ്ക്രിപ്റ്റിംഗ് കോഡുകൾ ഉപയോഗിച്ചിട്ടുണ്ട്. അതുപോലെ നിങ്ങൾക്ക് HTML ടാഗുകൾക്കിടയിൽ എത്ര തവണ വേണമെങ്കിലും സ്ക്രിപ്റ്റ് കോഡുകൾ ഉപയോഗിക്കാം. സ്ക്രിപ്റ്റ് കോഡുകൾ ഉപയോഗിക്കുന്നത് എവിടെയായിരുന്നാലും <SCRIPT>, </SCRIPT> ടാഗുകൾക്കിടയിൽ അവ എഴുതിയിരിക്കണം.

ഒരു വെബ് പേജിൽ ജാവാസ്ക്രിപ്റ്റിന്റെ പ്രവർത്തനം അനുവദിക്കുന്നതിനോ നിരാകരിക്കുന്നതിനോ ഉള്ള സൗകര്യം എല്ലാ വെബ് ബ്രൗസറുകളും ഉപയോക്താക്കൾക്ക് നൽകുന്നുണ്ട്. ജാവാസ്ക്രിപ്റ്റ് നിരാകരിക്കുന്നതിലൂടെ, ആ ബ്രൗസറിലെ ജാവാസ്ക്രിപ്റ്റ് എഞ്ചിന്റെ പ്രവർത്തനമാണ് നിരാകരിക്കുന്നത്. മോസില്ലയിൽ, Tools -> Options -> Content -> Enabled Java Script തിരഞ്ഞെടുത്ത് ജാവാസ്ക്രിപ്റ്റ് പ്രവർത്തനക്ഷമമാക്കുകയോ പ്രവർത്തനക്ഷമമല്ലാതാക്കുകയോ ചെയ്യാം. Google Chrome Â, Tools -> Settings -> Show Advanced Settings -> Content Settings -> Do not allow any site to run JavaScript രീതി ഉപയോഗിക്കാം. ഒരു വെബ് ബ്രൗസറിൽ ജാവാസ്ക്രിപ്റ്റ് പ്രവർത്തനക്ഷമമല്ലാതാക്കിയിട്ടുണ്ടെങ്കിൽ ആ ബ്രൗസറിൽ സ്ക്രിപ്റ്റ് പ്രവർത്തിക്കുകയില്ല അതായത്, ബ്രൗസർ <SCRIPT> ... </ SCRIPT> ടാഗുകൾക്കിടയിലുള്ള ഉള്ളടക്കം അവഗണിക്കും. ഒരു ബ്രൗസറിന്റെ പ്രവർത്തനം പ്രധാനമായും അതിന്റെ സ്ക്രിപ്റ്റ് എഞ്ചിന്റെ പ്രകടനത്തെ ആശ്രയിച്ചിരിക്കുന്നു. ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിക്കാത്ത ഒരു ഡൈനാമിക് വെബ് പേജ് കാണാൻ പ്രയാസമാണ്. മികച്ചതും, വേഗതയേറിയതും ശക്തവുമായ ജാവാസ്ക്രിപ്റ്റ് എഞ്ചിനുകളുടെ നിർമാണത്തിനായി എല്ലാ ബ്രൗസർ നിർമ്മാതാക്കളും പരസ്പരം മത്സരിക്കുന്നു.

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. <SCRIPT> ടാഗിന്റെ ഒരു ആട്രിബ്യൂട്ടിന്റെ പേര് എഴുതുക.
2. ജാവാസ്ക്രിപ്റ്റിൽ ഒരു HTML പേജിന്റെ ബോഡി ഭാഗത്തിൽ ഒരു ടെക്സ്റ്റ് പ്രിന്റ് ചെയ്യാൻ ഉപയോഗിക്കുന്ന ഫങ്ഷൻ ________ ആണ്
3. ജാവാസ്ക്രിപ്റ്റ് കോഡ് ആണ് എഴുതിയിരിക്കുന്നത് എന്ന് വ്യക്തമാക്കുന്നത് <SCRIPT> ടാഗിന്റെ language ആട്രിബ്യൂട്ടിന് നൽകിയ ________ എന്ന മൂല്യമാണ്.
4. ജാവാസ്ക്രിപ്റ്റ് എഞ്ചിന്റെ ഉപയോഗം എന്താണ്?
5. താഴെ പറയുന്ന പ്രസ്താവനകൾ ശരിയോ തെറ്റോ എന്ന് എഴുതുക.

a. ജാവ സ്ക്രിപ്റ്റ് മാത്രമാണ് ക്ലൈന്റ് വശം സ്ക്രിപ്റ്റിംഗ് ഭാഷ.

b. ഒരു HTML പേജിൽ ക്ലയന്റ് വശം സ്ക്രിപ്റ്റിംഗ് ഭാഷ ഉൾപ്പെടുത്താൻ <SCRIPT> ടാഗാണ് ഉപയോഗിക്കുന്നത്.

c. ഒരു HTML ഫയലിൽ ഒരു <SCRIPT> ടാഗ് മാത്രമേ അടങ്ങിയിരിക്കാവൂ.

d. നമുക്ക് HTML കോഡിൽ ജാവാസ്ക്രിപ്റ്റ് കോഡ് ഇടകലർത്തുവാൻ കഴിയും.

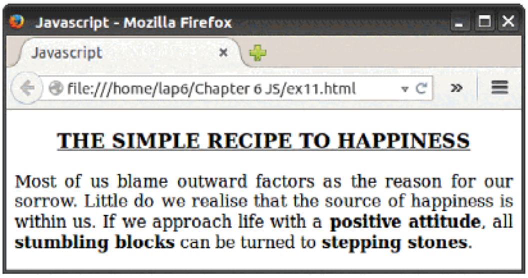
e. ജാവാസ്ക്രിപ്റ്റ് കോഡ് എല്ലായ്പ്പോഴും <SCRIPT>, </ SCRIPT> എന്നതിന് അകത്തതായിരിക്കണം.

f. ഓരോ ജാവാസ്ക്രിപ്റ്റ് പ്രസ്താവനകളും അർദ്ധവിരാമം(;) കൊണ്ട് അവസാനിക്കുന്നു.

നമുക്കു ചെയ്യാം

പേജിന്റെ ബോഡി വിഭാഗത്തിനുള്ളിൽ മാത്രം ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ച് താഴെ പറയുന്ന വെബ് പേജ് സൃഷ്ടിക്കാൻ ഒരു HTML കോഡ് എഴുതുക. അതായത്, ബോഡി വിഭാഗം താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ ആയിരിക്കണം.

```
<BODY>
<SCRIPT Language= "JavaScript">
................
................
</SCRIPT>
</BODY>
```



6.2 ജാവാസ്ക്രിപ്റ്റിലെ ഫങ്ഷനുകളുടെ നിർമാണം (Creating functions in JavaScript)

C++ ലെ ഫങ്ഷനുകളെക്കുറിച്ച് നാം നേരത്തെ മനസ്സിലാക്കിയിട്ടുണ്ട്. ജാവാസ്ക്രിപ്റ്റിലും പ്രവർത്തനങ്ങൾ നിർവചിക്കപ്പെട്ടിട്ടുള്ളതും വിളിക്കപ്പെടുന്നതും C++ ലെ അതേ രീതിയിൽ തന്നെയാണ്. ഒരു പേരുള്ള നിർദ്ദേശങ്ങളുടെ കൂട്ടമാണ് ഫങ്ഷൻ. വ്യത്യസ്ത

ആവശ്യങ്ങൾക്ക് ഉപയോഗിക്കാൻ കഴിയുന്ന ധാരാളം അന്തർനിർമിത ഫങ്ഷനുകൾ ജാവാസ്ക്രിപ്റ്റിൽ ഉണ്ട്. ഇവയിൽ ചില ഫങ്ഷനുകൾ ഈ അധ്യായത്തിൽ പിന്നീട് ചർച്ചചെയ്യും. ഈ അന്തർനിർമിത ഫങ്ഷനുകൾക്ക് പുറമെ നമുക്കും ഫങ്ഷനുകൾ നിർവ്വചിക്കാം. വെബ് പ്രോഗ്രാമിന്റെ ഒരു ഭാഗം ഒന്നിൽ കൂടുതൽ തവണ പ്രവർത്തിപ്പിക്കാം എന്നതാണ് ഒരു ഫങ്ഷൻ ഉപയോഗിക്കുന്നതുകൊണ്ടുള്ള ഏറ്റവും വലിയ മെച്ചം. ഈ ഭാഗം ഒരു പ്രാവശ്യം ഫങ്ഷനായി എഴുതിയാൽ മതി. താഴെ പറയുന്ന കോഡ് നോക്കുക.

```
function print()
{
    document.write("Welcome to JavaScript.");
}
```

ഇവിടെ, ഒരു ഫങ്ഷൻ നിർവചിക്കുവാനുള്ള കീവേഡ് 'function' ഉം ഫങ്ഷന്റെ പേര് print ഉം ആണ്. ഫങ്ഷന്റെ പേര് സാധുതയുള്ള ഏതൊരു ഐഡന്റിഫയറും ആകാം. C++ ൽ ഐഡന്റിഫയറുകൾ നാമകരണം ചെയ്യുന്ന അതേ നിയമങ്ങൾ ജാവാസ്ക്രിപ്റ്റിലും ഉപയോഗിക്കുന്നു. ഇവിടെ print() ഫങ്ഷനിൽ ഒരു പ്രസ്താവന മാത്രമേ ഉള്ളൂ, ആവശ്യമെങ്കിൽ ഒന്നിലധികം പ്രസ്താവനകൾ ഒരു ഫങ്ഷനിൽ ഉൾപ്പെടുത്താവുന്നതാണ്. ഒരു ഫങ്ഷൻ നിർവചിക്കുന്നത് കൊണ്ട് അത് സ്വയം പ്രവർത്തിക്കണം എന്നില്ല, അതിനെ വിളിക്കുമ്പോൾ മാത്രമേ അത് പ്രവർത്തിക്കൂ. ഇതിനർത്ഥം ഒരു വെബ് പേജിലെ ഒരു ഫങ്ഷൻ വിളിക്കാതിരിക്കുകയാണെങ്കിൽ ആ ഫങ്ഷൻ ഒരിക്കലും പ്രവർത്തിക്കുകയില്ല. ഫങ്ഷന്റെ പേര് ഉപയോഗിച്ച് താഴെ പറയുന്ന ഫങ്ഷനെ വിളിക്കാം.

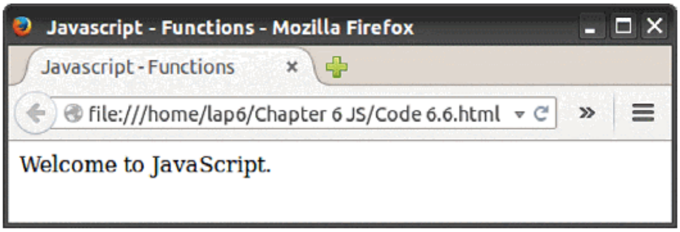
```
print();
```

ഫങ്ഷന്റെ പേരിന് ശേഷമുള്ള അർധവിരാമം ശ്രദ്ധിക്കുക. നമുക്ക് മുകളിലെ print() ഫങ്ഷൻ ഒരു HTML പേജിൽ ഉപയോഗിക്കുന്നത് എങ്ങനെ എന്ന് നോക്കാം.

ഉദാഹരണം 6.4: print() ഫംഗ്ഷൻ അടങ്ങിയ വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Functions</TITLE>
    <SCRIPT Language= "JavaScript">
        function print()
        {
            document.write("Welcome to JavaScript.");
        }
    </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

HTML പേജിന്റെ ബോഡി ഭാഗത്തിൽ ഒന്നും എഴുതിയിട്ടില്ലെന്നത് ശ്രദ്ധിക്കുക. ഈ കോഡ് ബ്രൗസർ വിൻഡോയിലെ 'Welcome to JavaScript' പ്രദർശിപ്പിക്കില്ല. കാരണം ഫങ്ഷൻ നിർവചിച്ചിട്ടുണ്ടെങ്കിലും, അത് പേജിലെ ഏതെങ്കിലും സ്ഥലത്തുനിന്ന് വിളിച്ചിട്ടില്ല. അതിനാൽ, ഫങ്ഷൻ ഒരിക്കലും പ്രവർത്തിക്കുകയില്ല, സ്ക്രീനിൽ ഒന്നും പ്രദർശിപ്പിക്കുകയുമില്ല. ബ്രൗസർ ജാലകത്തിൽ "Welcome to JavaScript" എന്ന് ഔട്ട്പുട്ട് ലഭിക്കുന്നതിന് HTML പേജിന്റെ ബോഡി ഭാഗം താഴെ കാണുന്ന രീതിയിൽ പരിഷ്കരിക്കേണ്ടതുണ്ട്.



ചിത്രം 6.3: ഫങ്ഷൻ ഉപയോഗിച്ചിരിക്കുന്ന വെബ് പേജ്

```
<BODY>
<SCRIPT Language= "JavaScript">
    print();
</SCRIPT>
</BODY>
```

ഇനി നമുക്ക് ജാവാ സ്ക്രിപ്റ്റ് ഫങ്ഷന്റെ വാക്യഘടന പരിചയപ്പെടാം

```
function function_name()
{
    statements;
}
```

ഇവിടെ function_name () എന്ന വരിയിൽ ഫങ്ഷൻ ഹെഡറും, ബ്രാക്കറ്റിലുള്ള കോഡ് ഫങ്ഷൻ ബോഡി എന്നു വിളിക്കുന്നു. C++ ൽ ഫങ്ഷന് റിട്ടേൺ ടൈപ്പ് ഉണ്ടെങ്കിലും ജാവാസ്ക്രിപ്റ്റിൽ ഫങ്ഷന് റിട്ടേൺ ടൈപ്പില്ല. ജാവാസ്ക്രിപ്റ്റിലും C++ ലേതുപോലെ നിങ്ങൾക്ക് ഫങ്ഷനിൽ നിന്ന് ചില മൂല്യങ്ങൾ തിരികെ നൽകാം. ഈ അധ്യായത്തിൽ ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗത്തെക്കുറിച്ചുള്ള അടിസ്ഥാന ആശയങ്ങൾ മാത്രം നൽകാൻ ഉദ്ദേശിച്ചിട്ടുള്ളതിനാൽ, ഇത്തരം കാര്യങ്ങൾ ഇവിടെ ചർച്ച ചെയ്യുന്നില്ല. മറ്റൊരു വ്യത്യാസം C++ൽ ഒരു ഫങ്ഷൻ നിർവചിക്കാൻ function എന്ന കീവേഡ് ഉപയോഗിക്കില്ല, എന്നാൽ ജാവാസ്ക്രിപ്റ്റിൽ function എന്ന കീവേഡ് ഉപയോഗിക്കുന്നു. HTML പേജിന്റെ ഹെഡ് ഭാഗത്തിനുള്ളിൽ ഫങ്ഷൻ നിർവചിക്കപ്പെട്ടിരിക്കുന്നതായി നിങ്ങൾ ശ്രദ്ധിച്ചിരിക്കാം. ഹെഡ് ഭാഗത്തിനുള്ളിൽ തന്നെ ഫങ്ഷൻ നിർവചിക്കണം എന്ന് നിർബന്ധമില്ല, ബോഡി ഭാഗത്തും ഫങ്ഷൻ നിർവചിക്കാം.

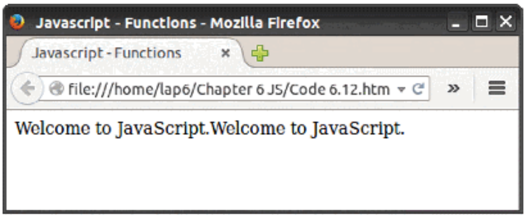
```
<BODY>
    <SCRIPT Language= "JavaScript">
        function print()
```

```
        {
            document.write("Welcome to JavaScript.");
        }
        print();
    </SCRIPT>
</BODY>
```

മുകളിൽ പറഞ്ഞിരിക്കുന്ന കോഡ് ഇതേ ഔട്ട്പുട്ട് നൽകുന്നത് ചിത്രം 6.3 ൽ കൊടുത്തിരിക്കുന്നു.

ഫങ്ഷൻ ബോഡി ഭാഗത്തിനകത്താണ് നിർവചിക്കപ്പെട്ടിരിക്കുന്നതെങ്കിൽപ്പോലും അതിന്റെ പ്രവർത്തിപ്പിക്കുന്നതിനുവേണ്ടി അതിനെ വിളിക്കണം എന്നത് ശ്രദ്ധിക്കുക. ഉദാഹരണത്തിന് താഴെ കൊടുത്തിരിക്കുന്ന കോഡ് സ്ക്രീനിൽ ഒന്നും പ്രദർശിപ്പിക്കുന്നില്ല.

```
<BODY>
<SCRIPT Language= "JavaScript">
function print()
{
document.write("Welcome to JavaScript.");
}
</SCRIPT>
</BODY>
```



ചിത്രം 6.4: ബ്രേക്ക് ഇല്ലാതെ രണ്ട് പ്രിന്റ് ഫങ്ഷനുകൾ ഉപയോഗിക്കുന്ന വെബ് പേജ്

ഒരു HTML പേജിൽ എവിടെയും ഒരു ഫങ്ഷൻ നിർവചിക്കപ്പെടാമെങ്കിലും, ഹെഡ് ഭാഗത്തിൽ ഫങ്ഷൻ നിർവചനം ഉൾപ്പെടുത്തുന്നതാണ് നല്ലത്.

`print()` ഫങ്ഷൻ രണ്ടു തവണ വിളിച്ചിരിക്കുന്ന താഴെ കൊടുത്തിരിക്കുന്ന കോഡ് പരിഗണിക്കുക, ഇത് ചിത്രം 6.4ൽ കാണുന്ന രീതിയിലുള്ള ഔട്ട്പുട്ട് നൽകുന്നു.

```
<BODY>
    <SCRIPT Language= "JavaScript">
        print();
        print();
    </SCRIPT>
</BODY>
```

ഔട്ട്പുട്ട് ചിത്രം 6.4 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ ആയിരിക്കും. എന്തുകൊണ്ട്?

വാസ്തവത്തിൽ, നമ്മൾ രണ്ടുതവണ ഫങ്ഷനെ വിളിക്കുമ്പോൾ, അത് 'Welcome to JavaScript' എന്ന് രണ്ടുതവണ ബോഡി ഭാഗത്ത് എഴുതും. അതിനാൽ മുകളിലുള്ള കോഡിന്റെ ബോഡി ഭാഗം താഴെ എഴുതിയിരിക്കുന്നതിനു തുല്യമായിരിക്കും.

```
<BODY>
    Welcome to JavaScript.Welcome to JavaScript.
</BODY>
```

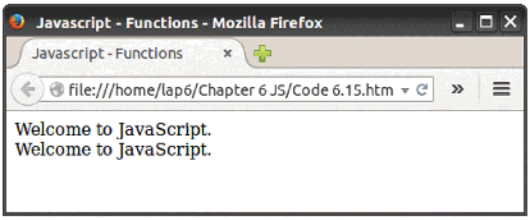
സന്ദേശം രണ്ട് വ്യത്യസ്ത വരികളിൽ പ്രദർശിപ്പിക്കുന്നതിന് നിർബന്ധമായും
 ടാഗ് ഉപയോഗിക്കണം.

```
<BODY>
    Welcome to JavaScript.<BR>Welcome to JavaScript.
</BODY>
```

ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ച് ഇതേ റിസൾട്ട് ലഭിക്കുന്നതിന് താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ മാറ്റം വരുത്താം.

```
<SCRIPT Language= "JavaScript">
    function print()
    {
        document.write("Welcome to JavaScript.<BR>");
    }
</SCRIPT>
```

മുകളിൽ പറഞ്ഞിരിക്കുന്ന HTML കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.5 ൽ കൊടുത്തിരിക്കുന്നു.



ചിത്രം 6.5: രണ്ട് പ്രിന്റ് ഫങ്ഷനുകൾ ഉപയോഗിക്കുന്ന വെബ് പേജ്

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. താഴെ കൊടുത്തിരിക്കുന്ന പ്രസ്താവന ശരിയോ തെറ്റോ എന്ന് പറയുക.
   a. ഒരു വെബ് പേജ് ബ്രൗസറിൽ തുറക്കുമ്പോൾ ഫങ്ഷൻ സ്വമേധയാ പ്രവർത്തിക്കും.
   b. ഒരു ഫംഗ്ഷൻ സാധാരണയായി HTML പേജിന്റെ ഹെഡ്ഡ് ഭാഗത്തിൽ ഉൾപ്പെടുത്തുന്നു.
   c. ഒരു ഫംഗ്ഷൻ എത്ര തവണ വേണമെങ്കിലും വിളിക്കാം.
   d. ബോഡി ഭാഗത്തിനകത്ത് ഒരു ഫങ്ഷൻ നിർവചിച്ചിട്ടുണ്ടെങ്കിലും അത് വിളിച്ചിട്ടില്ലെങ്കിൽ പ്രവർത്തിക്കില്ല.
2. ജാവാസ്ക്രിപ്റ്റിൽ ഫങ്ഷനുകൾ ഉപയോഗിക്കുന്നതിന്റെ ഗുണദോഷങ്ങൾ എഴുതുക.

നമുക്കു ചെയ്യാം

1. താഴെക്കൊടുത്തിരിക്കുന്ന രണ്ട് HTML കോഡുകൾ (കോഡ് A ഉം കോഡ് B ഉം) പരിഗണിച്ച് ഔട്ട്പുട്ട് എഴുതുക.

കോഡ് *A*

```
<HTML>
<HEAD>
<SCRIPT Language= "JavaScript">
    function print()
    {
        document.write("Welcome to JavaScript");
    }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    print();
    document.write("<BR>");
    print();
</SCRIPT>
</BODY>
</HTML>
```

കോഡ് *B*

```
<HTML>
<HEAD>
<SCRIPT Language= "JavaScript">
function print()
{
      document.write("Welcome to JavaScript");
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
      print();
</SCRIPT>
<BR>
<SCRIPT Language= "JavaScript">
      document.write("<BR>");
      print();
</SCRIPT>
</BODY>
</HTML>
```

2. ചിത്രം 6.6 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ ഔട്ട്പുട്ട് ലഭിക്കുന്നതിന് ഒരു HTML കോഡ് താഴെ കൊടുക്കുന്നു. കോഡിൽ `startGreen()`, `stopGreen()`, `startRed()` and `stopRed()` എന്നിങ്ങനെ നാലു ഫങ്ഷനുകൾ ഉണ്ട്. ഈ ഫങ്ഷനുകൾ HTML പേജിന്റെ ബോഡി ഭാഗത്തിൽ വ്യത്യസ്ത സ്ഥലങ്ങളിൽ നിന്നും വിളിക്കുന്നു. `stopGreen()` എന്ന ഫങ്ഷൻ ഒഴികെയുള്ള എല്ലാ ഫങ്ഷൻ നിർവചനങ്ങളുടെയും ബോഡി ശൂന്യമാണ്. നിങ്ങൾക്ക് ചിത്രത്തിൽ കാണിച്ചിരിക്കുന്നതുപോലെ ഔട്ട്പുട്ട് ലഭിക്കുന്നതിന് മറ്റു ഫങ്ഷനുകളുടെ നിർവചനം പൂർത്തിയാക്കുക.

```
<HTML>
<HEAD>
<SCRIPT Language= "JavaScript">
      function startGreen()
      {
            .............
            .............
      }
      function stopGreen()
      {
            document.write("</FONT>");
      }
      function startRed()
      {
            .............
            .............
      }
      function stopRed()
      {
            .............
            .............
      }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
      startGreen();
```

```
</SCRIPT>
This is in Green colour with size 5
<SCRIPT Language= "JavaScript">
     stopGreen();
</SCRIPT>
<SCRIPT Language= "JavaScript">
     startRed();
</SCRIPT>
This is in Red colour with size 3
<SCRIPT Language= "JavaScript">
     stopRed();
</SCRIPT>
<SCRIPT Language= "JavaScript">
     startGreen();
</SCRIPT>
This is in Green colour with size 5
<SCRIPT Language= "JavaScript">
     stopGreen();
</SCRIPT>
</BODY>
</HTML>
```

6.3 ജാവാസ്ക്രിപ്റ്റിലെ ഡാറ്റ ഇനങ്ങൾ (Data types in JavaScript)

മെമ്മറി ഫലപ്രദമായി ഉപയോഗപ്പെടുത്തുന്നതിന് പ്രോഗ്രാമിങ് ഭാഷകൾ ഡാറ്റയെ വ്യത്യസ്ത വിഭാഗങ്ങളായി തരം തിരിച്ചിരിക്കുന്നു. C++ ലെ അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ int, char, float, double, void തുടങ്ങിയവയാണെന്ന് നമ്മൾ മനസ്സിലാക്കിയിട്ടുണ്ട്. ഈ ഡാറ്റ ഇനങ്ങളൊടൊപ്പം, ടൈപ്പ് മോഡിഫയറുകളും C++ ൽ ഉണ്ടായിരുന്നു. അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളുടെ എണ്ണം 3 ആയി പരിമിതപ്പെടുത്തിയിരിക്കുന്നതിലൂടെ ജാവാസ്ക്രിപ്റ്റ് ഇതിന്റെ സങ്കീർണത കുറയ്ക്കുന്നു. താഴെ പറയുന്നവയാണ് ജാവാസ്ക്രിപ്റ്റിൽ ഉപയോഗിക്കുന്ന മൂന്ന് അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ.

നമ്പർ (Number)

എല്ലാ നമ്പറുകളും ഈ വിഭാഗത്തിൽ വരുന്നു. പോസിറ്റീവ് സംഖ്യകളും, നെഗറ്റീവ് സംഖ്യകളും, എല്ലാ ഫ്ളോട്ട് വിലകളും (ഭിന്നസംഖ്യകൾ), പൂർണ സംഖ്യകളും നമ്പർ (number) ഡാറ്റ ഇനം ആയി പരിഗണിക്കപ്പെടും. അതിനാൽ, 27, -300, 1.89, -0.0082 എന്നിവ ജാവാസ്ക്രിപ്റ്റിലെ നമ്പർ ഡാറ്റ ഇനത്തിനുള്ള ഉദാഹരണങ്ങളാണ്.

സ്ട്രിംഗ് (String)

ഇരട്ട ഉദ്ധരണികൾക്കുള്ളിൽ ഉള്ള പ്രതീകങ്ങൾ, അക്കങ്ങൾ അല്ലെങ്കിൽ മറ്റേതെങ്കിലും ചിഹ്നങ്ങളുടെ കൂട്ടത്തെ ജാവാസ്ക്രിപ്റ്റിൽ ഒരു സ്ട്രിംഗായി പരിഗണിക്കും. ഉദാഹരണം "Kerala", "Welcome", "SCHOOL", "1234", "Mark20", "abc$" and "sanil@123".

ബൂളിയൻ (Boolean)

ഈ വിഭാഗത്തിൽ രണ്ട് മൂല്യങ്ങൾ മാത്രമേ ഉള്ളൂ, അവ true, false എന്നിവയാണ്. ഈ മൂല്യങ്ങൾ ഇരട്ട ഉദ്ധരണികളിലല്ല. ഇരട്ട ഉദ്ധരണികൾക്കുള്ളിൽ ആണെങ്കിൽ അവ സ്ട്രിംഗ് ആയിരിക്കും. ജാവാസ്ക്രിപ്റ്റ് കേസ് സെൻസിറ്റീവ് ആയതുകൊണ്ട് ബൂളിയൻ വിലകളെ പ്രതിനിധാനം ചെയ്യുന്നതിന് TRUE, FALSE എന്നിങ്ങനെ ഉപയോഗിക്കുവാൻ പാടില്ല.

6.4 ജാവാ സ്ക്രിപ്റ്റിലെ വേരിയബിളുകൾ (Variables in JavaSript)

വിലകൾ സംഭരിക്കുന്നതിന് വേരിയബിളുകൾ ഉപയോഗിക്കാമെന്ന് നമുക്കറിയാം. ജാവാ സ്ക്രിപ്റ്റിൽ **var** എന്ന കീവേഡ് ഉപയോഗിച്ചാണ് എല്ലാത്തരം വേരിയബിളുകളും പ്രഖ്യാപിക്കുന്നത്.

```
var  x;
```

ഇവിടെ 'x' എന്നത് വേരിയബിളിന്റെ പേരാണ്, ഒരു വേരിയബിളിന് C++ ലേതുപോലെ ഏതു പേരും ഉപയോഗിക്കാം. അത് ഒരു സാധുതയുള്ള ഐഡന്റിഫയർ ആയിരിക്കണമെന്നുമാത്രം. C++ ൽ വ്യത്യസ്ത തരം വേരിയബിളുകൾ പ്രഖ്യാപിക്കാൻ കീവേഡുകൾ `int`, `float`, `char` തുടങ്ങിയവ ഉപയോഗിച്ചു. പക്ഷേ, ജാവാസ്ക്രിപ്റ്റിൽ **var** എന്ന കീവേഡ്' മാത്രമാണ് എല്ലാത്തരം വേരിയബിളുകൾ പ്രഖ്യാപിക്കുന്നതിനും ഉപയോഗിക്കുന്നത്. ജാവാസ്ക്രിപ്റ്റിൽ വേരിയബിളിന് വില നൽകുമ്പോൾ മാത്രം നിർവചനം പൂർത്തിയാകുന്നു. ഒരു വേരിയബിളിന് മൂല്യം നൽകുമ്പോൾ മാത്രമാണ് വേരിയബിളിന്റെ തരം ജാവാസ്ക്രിപ്റ്റ് മനസ്സിലാക്കുന്നത്.

താഴെകൊടുത്തിരിക്കുന്ന പ്രസ്താവന പരിഗണിക്കുക.

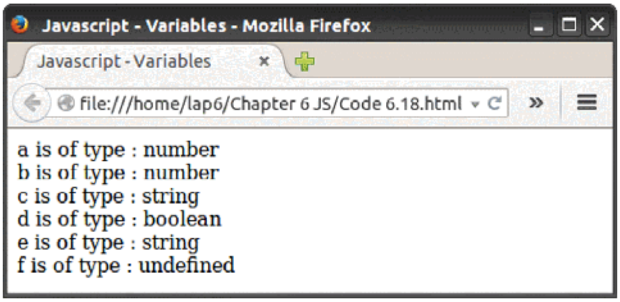
```
var x, y;
x = 25;
y = "INDIA";
```

മുകളിൽ പറഞ്ഞിരിക്കുന്ന ഉദാഹരണത്തിൽ, x എന്നത് നമ്പർ തരത്തിലും y സ്ട്രിംഗ് തരത്തിലുംപെട്ട വേരിയബിളുകൾ ആണ്. എന്നാൽ കീവേഡ് number, string എന്നൊന്നും ഇവിടെ ഉപയോഗിച്ചിട്ടില്ല. താഴെ കൊടുത്തിരിക്കുന്ന ഉദാഹരണത്തിൽ ജാവാ സ്ക്രിപ്റ്റ് എങ്ങനെയാണ് വേരിയബിളുകൾ നിർവചിക്കുന്നത് എന്ന് വിവരിക്കുന്നു.

ഉദാഹരണം 6.5 : വേരിയബിളുകളുടെ ഉപയോഗം വിവരിക്കുന്നു.

```
<HTML>
<HEAD> <TITLE>Javascript - Variables</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
   var a, b, c, d, e, f;
   a = 25;
   b = 18.5;
   c = "INDIA";
   d = true;
   e = "true";
   document.write("a is of type : ");
   document.write(typeof(a));
   document.write("<BR>b is of type : ");
   document.write(typeof(b));
   document.write("<BR>c is of type : ");
   document.write(typeof(c));
   document.write("<BR>d is of type : ");
   document.write(typeof(d));
   document.write("<BR>e is of type : ");
   document.write(typeof(e));
   document.write("<BR>f is of type : ");
   document.write(typeof(f));
</SCRIPT>
</BODY>
</HTML>
```

ഇവിടെ, `typeof()` എന്നൊരു ഫങ്ഷൻ പുതുതായി ഉപയോഗിച്ചിരിക്കുന്നു. പേര് സൂചിപ്പിക്കുന്നതുപോലെ ഒരു വേരിയബിളിന്റെ തരം കണ്ടുപിടിക്കുവാൻ ഈ ഫങ്ഷൻ ഉപയോഗിക്കുന്നു. ഈ ഫങ്ഷനെ പിന്നീട് വിശദമായി പഠിക്കാം. മുകളിലുള്ള ഉദാഹരണത്തിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.6 ൽ കൊടുത്തിരിക്കുന്നു.



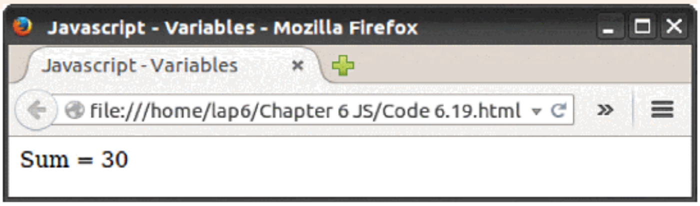
ചിത്രം 6.6: വേരിയബിളുകളുടെ തരം ഉപയോഗം പ്രദർശിപ്പിക്കുന്ന വെബ് പേജ്.

വേരിയബിൾ f പ്രഖ്യാപിച്ചിട്ടുണ്ടെങ്കിലും അതിന് ഒരു വില നൽകിയിട്ടില്ല. അതുകൊണ്ട് സ്ക്രിപ്റ്റ് എഞ്ചിന് അതിന്റെ തരം മനസ്സിലാക്കുവാനോ പ്രദർശിപ്പിക്കുവാനോ സാധിക്കു

ന്നില്ല. നിർവചിക്കാത്ത വേരിയബിളിനെ പ്രതിനിധാനം ചെയ്യുന്നതിനായി സ്ക്രിപ്റ്റിൽ ഉപയോഗിക്കുന്ന ഒരു പ്രത്യേക ഡാറ്റ ഇനമാണ് undefined. ഒരൊറ്റ var കീവേഡ് ഉപയോഗിച്ച് എത്ര വേരിയബിളുകൾ വേണമെങ്കിലും പ്രഖ്യാപിക്കാനാവും (ഡിക്ലയർ ചെയ്യാനാവും). വേരിയബിളുകൾ കോമ (,) ഉപയോഗിച്ച് വേർതിരിക്കേണ്ടതാണ്. വേരിയബിളുകൾ ഉപയോഗിക്കുന്ന ഒരു ജാവാസ്ക്രിപ്റ്റിന്റെ പ്രവർത്തനം നമുക്ക് പരിഗണിക്കാം.

ഉദാഹരണം 6.6: രണ്ട് സംഖ്യകളുടെ തുക കണ്ടു പിടിക്കുന്നതിനുള്ള വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD><TITLE>Javascript - Variables</TITLE>
<SCRIPT Language= "JavaScript">
function add()
{
    var m, n, sum;
    m = 20;
    n = 10;
    sum = m + n;
    document.write("Sum = ");
    document.write(sum);
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    add();
</SCRIPT>
</BODY>
</HTML>
```



ചിത്രം 6.7: രണ്ടു സംഖ്യകളുടെ തുക കണ്ടുപിടിക്കുന്നതിനുള്ള വെബ്പേജ്

മുകളിൽ പറഞ്ഞ ഉദാഹരണത്തിൽ, ബോഡി ഭാഗത്തിൽ ഒരു ജാവാസ്ക്രിപ്റ്റ് പ്രസ്താവന മാത്രമേ അടങ്ങിയിട്ടുള്ളു, അത് add() എന്ന ഫങ്ഷനെ വിളിക്കുന്നു. ഫങ്ഷൻ പ്രവർത്തനത്തിന്റെ ഫലം ചിത്രം 6.7 ൽ കാണിച്ചിരിയ്ക്കുന്നു.

6.5 ജാവാസ്ക്രിപ്റ്റിലെ ഓപ്പറേറ്ററുകൾ (Operators in JavaScript)

ജാവാ സ്ക്രിപ്റ്റിലെ മിക്കവാറും എല്ലാ ഓപ്പറേറ്ററുകളും C++ ലേതിന് സമാനമാണ്. അവ ഏതെല്ലാമാണെന്ന് നോക്കാം

6.5.1 അരിത്തമാറ്റിക് ഓപ്പറേറ്ററുകൾ (Arithmetic Operatiors)

ജാവാസ്ക്രിപ്റ്റിൽ ഉപയോഗിക്കുന്ന അരിത്തമാറ്റിക് ഓപ്പറേറ്ററുകൾ പട്ടിക 6.1 കാണിക്കുന്നു.

ഓപ്പറേഷൻ	വിവരണം	ഉദാഹരണം	y യുടെ മൂല്യം	ഉത്തരം (x)
+	സങ്കലനം	x = y + 10	15	25
-	വ്യവകലനം	x = y - 10	15	5
*	ഗുണനം	x = y * 3	15	45
/	ഹരണം	x = y / 2	15	7.5
%	മോഡുലസ് ഹരണത്തിലെ ശിഷ്ടം	x = y % 2	15	1
++	വർദ്ധനവ് Increment	x = ++y x = y++	15 15	16 15
--	കുറവ് Decrement	x = --y x = y--	15 15	14 15

പട്ടിക 6.1: അരിത്തമാറ്റിക് ഓപ്പറേറ്ററുകൾ

മുകളിലുള്ള പട്ടികയിലെ എല്ലാ അരിത്തമാറ്റിക് ഓപ്പറേററുകളും C++ ൽ പ്രവർത്തിക്കുന്നപോലെതന്നെ പ്രവർത്തിക്കുന്നുവെന്ന് കാണാം.

6.5.2 വിലനൽകൽ ഓപ്പറേറ്ററുകൾ (Assignment Operators)

ജാവാസ്ക്രിപ്റ്റിലെ വിവിധ വിലനൽകൽ ഓപ്പറേറ്ററുകളുടെ ഉപയോഗരീതി പട്ടിക 6.2ൽ കാണിച്ചിരിക്കുന്നു.

ഓപ്പറേറ്റർ	വിവരണം	ഉദാഹരണം	a യുടെ വില	b യുടെ വില	ഉത്തരം (a)
=	വില നൽകുന്നു	a = b	10	3	3
+=	സങ്കലനത്തിനുശേഷം വില നൽകുന്നു	a+=b	10	3	13
-=	വ്യവകലനത്തിനുശേഷം വില നൽകുന്നു	a-=b	10	3	7
*=	ഗുണനത്തിനുശേഷം വില നൽകുന്നു	a*=b	10	3	30
/=	ഹരണത്തിനുശേഷം വില നൽകുന്നു	a/=b	10	3	3.33
%=	മോഡുലസ് പ്രവർത്തനത്തിനുശേഷം വില നൽകുന്നു	a%=b	10	3	1

പട്ടിക 6.2: ജാവാസ്ക്രിപ്റ്റിലെ വിവിധ വിലനൽകൽ ഓപ്പറേറ്ററുകൾ

മുകളിലുള്ള പട്ടികയിൽ നിന്ന് ഓരോ ഓപ്പറേറ്ററുകളുടെയും പ്രവർത്തനം മനസ്സിലാക്കുക വളരെ എളുപ്പമാണ്. a യുടെയും b യുടെയും വിലകളുപയോഗിച്ച് 'ഉദാഹരണം ' എന്ന

കോളത്തിലെ പ്രസ്താവന പ്രവർത്തിപ്പിച്ച ശേഷമുള്ള ഫലം 'ഉത്തരം' എന്ന കോളത്തിൽ കാണിച്ചിരിക്കുന്നു.

6.5.3 റിലേഷണൽ ഓപ്പറേറ്റർ (താരതമ്യ ഓപ്പറേറ്റർ) (Relational/Comparison operator)

ജാവാസ്ക്രിപ്റ്റിൽ ഉപയോഗിക്കുന്ന വിവിധ തരം താരതമ്യ ഓപ്പറേറ്ററുകൾ പട്ടിക 6.3 ൽ കാണിച്ചിരിക്കുന്നു.

ഓപ്പറേറ്റർ	വിവരണം	ഉദാഹരണം	a യുടെ വില	b യുടെ വില	ഉത്തരം (a)
==	തുല്യം	a==b	10	3	false
!=	തുല്യമല്ല	a!=b	10	3	true
<	കുറവ്	a<b	10	3	false
<=	കുറവോ അല്ലെങ്കിൽ തുല്യമോ	a<=b	10	3	false
>	കൂടുതൽ	a>b	10	3	true
>=	കൂടുതലോ അല്ലെങ്കിൽ തുല്യമോ	a>=b	10	3	true

പട്ടിക 6.3: റിലേഷണൽ ഓപ്പറേറ്റർ

ഒരു താരതമ്യ/റിലേഷണൽ ഓപ്പറേഷന്റെ ഫലം ശരിയോ തെറ്റോ എന്ന് പട്ടികയിൽ നിന്ന് വ്യക്തമാണ്. ഈ ഓപ്പറേറ്ററുകൾ രണ്ട് വശത്തും ഉള്ള മൂല്യങ്ങൾ താരതമ്യം ചെയ്യുകയും അതിനനുസരിച്ച് ഫലം നൽകുകയും ചെയ്യുന്നു.

6.5.4 ലോജിക്കൽ ഓപ്പറേറ്റർ (Logical Operators)

ജാവാസ്ക്രിപ്റ്റിലെ വിവിധ ലോജിക്കൽ ഓപ്പറേറ്ററുകളുടെ പട്ടിക 6.4 ൽ ഉദാഹരണത്തോടൊപ്പം കാണിച്ചിരിക്കുന്നു.

ഓപ്പറേറ്റർ	വിവരണം	ഉദാഹരണം	a യുടെ വില	b യുടെ വില	ഉത്തരം (a)
&&	ആന്റ്	a && b	true	false	false
||	ഓർ	a || b	true	false	true
!	നോട്ട്	!a	true		false

പട്ടിക 6.4: ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ

മുകളിൽ കൊടുത്തിരിക്കുന്ന പട്ടികയിൽ നിന്ന്, ഈ ഓപ്പറേറ്ററുകൾ എല്ലാം തന്നെ C++ ലേതിന് സമാനമാണെന്നത് വളരെ വ്യക്തമാണ്. വിവിധ ആവശ്യങ്ങൾക്കായി മറ്റ് ധാരാളം ഓപ്പറേറ്ററുകൾ ജാവാസ്ക്രിപ്റ്റ് നൽകുന്നു. ജാവാസ്ക്രിപ്റ്റിനെക്കുറിച്ചുള്ള നമ്മുടെ ചർച്ച ഈ അധ്യായത്തിൽ മാത്രം ഒതുങ്ങിയിരിക്കുന്നതിനാൽ അത്തരം

ഓപ്പറേറ്ററുകളെക്കുറിച്ചുള്ള ചർച്ച ഈ പുസ്തകത്തിന്റെ പരിധിക്കു പുറത്താണ്. കൂടാതെ, മുകളിൽ വിവരിച്ച ഓപ്പറേറ്ററുകൾ ഒരു തുടക്കക്കാരനു വേണ്ട എല്ലാ പ്രവർത്തനങ്ങളും നടത്തുന്നതിന് പര്യാപ്തമാണ്. എന്നിരുന്നാലും, താഴെ പറയുന്ന സ്ട്രിംഗ് ഓപ്പറേറ്റർ വിവിധ സാഹചര്യങ്ങളിൽ നിങ്ങൾക്ക് ഉപയോഗപ്രദമാകും. ഈ ഓപ്പറേറ്റർ C++ ൽ ലഭ്യമല്ല.

6.5.5 സ്ട്രിംഗ് അഡീഷൻ ഓപ്പറേറ്റർ (+) (String addition operator (+))

രണ്ട് സംഖ്യകൾ കൂട്ടുന്നതിന് + ഓപ്പറേറ്റർ ഉപയോഗിക്കാമെന്ന് നമുക്ക് അറിയാം. അതേ ഓപ്പറേറ്ററായ '+' രണ്ട് സ്ട്രിംഗുകൾ കൂട്ടിച്ചേർക്കുന്നതിനും ഉപയോഗിക്കാം. കൺകാറ്റ് നേഷൻ (concatenation) എന്നതിനർഥം രണ്ട് സ്ട്രിംഗുകൾ കൂട്ടിചേർക്കുക എന്നാണ്.

ഉദാഹരണം

```
var x, y;
x = "A good beginning ";
y = "makes a good ending.";
z = x + y;
```

+ ഓപ്പറേറ്റർ രണ്ട് സ്ട്രിംഗുകൾ കൂട്ടിച്ചേർക്കും, അങ്ങനെ വേരിയബിൾ z ന്റെ മൂല്യം 'A good beginning makes a good ending' എന്നാകും. ഒരേ ഓപ്പറേറ്റർ വ്യത്യസ്ത ഓപ്പറന്റുകളുടെ തരം മനസ്സിലാക്കിയാണ് പ്രവർത്തിക്കുന്നത്. ഓപ്പറന്റുകൾ സംഖ്യ ആണെങ്കിൽ, അവയുടെ തുക കണ്ടുപിടിക്കും. മറിച്ച് സ്ട്രിംഗുകൾ ആണെങ്കിൽ അവ കൂട്ടിച്ചേർക്കും. ഇനി, താഴെ പറഞ്ഞിരിക്കുന്ന കോഡിൽ z ന്റെ മൂല്യം പ്രവചിക്കുക?

```
var  x, y;
x = "23";
y = 5;
z = x + y;
```

ഇതിന്റെ ഉത്തരം 235 ആണ്. ഓപ്പറന്റുകളിൽ ഏതെങ്കിലും ഒരെണ്ണം സ്ട്രിങ്ങാണെങ്കിൽ + ഓപ്പറേറ്റർ എല്ലാ ഓപ്പറന്റുകളെയും സ്ട്രിങ്ങായി കരുതി കൂട്ടിച്ചേർക്കുന്നു. അങ്ങനെ z ന്റെ വിലയായി 235 ലഭിക്കുന്നു. x ന്റെയും y യുടെയും തുക സംഖ്യാരൂപത്തിൽ കാണണമെന്നുകരുതുക. ഇതിനായി താഴെ കൊടുത്തിരിക്കുന്ന രീതിയിൽ പ്രസ്താവന മാറ്റിയെഴുതാം.

```
z = Number(x) + y;
```

മുകളിൽ പറഞ്ഞിരിക്കുന്ന പ്രസ്താവന പ്രവർത്തിച്ച ശേഷം വേരിയബിൾ z ന്റെ വില 28 ആയിരിക്കും. Number() എന്നത് ജാവാ സ്ക്രിപ്റ്റിലെ ഒരു ഫംഗ്ഷൻ ആണ്, അത് സ്ട്രിംഗ് തരം ഡാറ്റയെ നമ്പറായി മാറ്റുന്നു. ഈ അധ്യായത്തിലെ ചില ഉദാഹരണങ്ങളിൽ ഈ ഫംഗ്ഷൻ ഉപയോഗിക്കുന്നുണ്ട്.

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. നമ്പർ, സ്ട്രിംഗ്, ______ എന്നിവയാണ് ജാവാസ്ക്രിപ്റ്റിലെ അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ.
2. 'true' ഒരു ______ ഇനം ഡാറ്റയാണ്.
3. 'false' ഒരു ______ ഇനം ഡാറ്റയാണ്.
4. ജാവാസ്ക്രിപ്റ്റിൽ ഒരു വേരിയബിൾ പ്രഖ്യാപിക്കാൻ വേണ്ട കീവേഡ് ______ ആണ്.
5. ജാവാസ്ക്രിപ്റ്റിൽ ഡാറ്റ ഇനം അറിയാൻ ഉപയോഗിക്കുന്നതിനുള്ള ഫങ്ഷൻ ______ ആണ്.
6. % ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നത് എന്തിനാണ്?
7. ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ ഏവ?

6.6 ജാവാസ്ക്രിപ്റ്റിലെ കൺട്രോൾ സ്ട്രക്ചറുകൾ (Control structures in JavaScript)

ഒരു പ്രോഗ്രാമിന്റെ പ്രവർത്തനത്തിന്റെ നിയന്ത്രണത്തിന് കൺട്രോൾ സ്ട്രക്ചറുകൾ ഉപയോഗിക്കുന്നു. C++ ൽ പഠിച്ചിട്ടുള്ള എല്ലാ കൺട്രോൾ സ്ട്രക്ചറുകളും ജാവാസ്ക്രിപ്റ്റിലും വ്യത്യാസമില്ലാതെ ഉപയോഗിക്കാവുന്നതാണ്. ജാവാ സ്ക്രിപ്റ്റിൽ ഉപയോഗിക്കുന്ന വിവിധ കൺട്രോൾ സ്ട്രക്ചറുകൾ ഉദാഹരണസഹിതം നമുക്ക് പരിശോധിക്കാം.

6.6.1 if

എല്ലാ പ്രോഗ്രാമിങ് ഭാഷയിലും ഏറ്റവുമധികം ഉപയോഗിക്കപ്പെടുന്ന കൺട്രോൾ സ്ട്രക്ചറാണിത്. ചില വ്യവസ്ഥകൾ അടിസ്ഥാനമാക്കി ഒരു പ്രസ്താവന അല്ലെങ്കിൽ ഒരു കൂട്ടം പ്രസ്താവനകൾ പ്രവർത്തിപ്പിക്കാൻ ഇത് ഉപയോഗിക്കുന്നു. പട്ടിക 6.5 ൽ നൽകിയിരിക്കുന്നതുപോലെ രണ്ട് രീതിയിൽ ഇത് ഉപയോഗിക്കാം

if ന്റെ വാക്യഘടന	if ...... ന്റെ വാക്യഘടന

`if (test_expression)`
`{`
`    statements;`
`}`
	`if (test_expression)`
	`{`
	`    statements;`
	`}`
	`else`
	`{`
	`    statements;`
	`}`

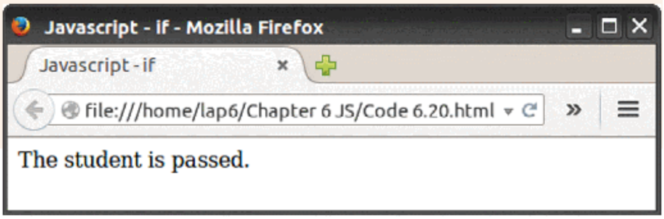
പട്ടിക 6.5: if, else ഓടുകൂടിയ if പ്രസ്താവനയുടെ വാക്യഘടന

if, if .... else പ്രസ്താവനയുടെ ഉപയോഗം കാണിക്കുന്ന ഒരു ഉദാഹരണം നോക്കാം.

ഉദാഹരണം 6.12: ഒരു വിദ്യാർഥി ജയിച്ചോ ഇല്ലയോ എന്നു പരിശോധിക്കുന്നതിനുള്ള ഒരു വെബ് പേജ് നിർമാണം.

ഉദാഹരണം 6.7: രണ്ട് സംഖ്യകളുടെ തുക കണ്ടു പിടിക്കുന്നതിനുള്ള വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - if</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
   var score;
   score = 35;
   if (score < 30)
   {
      document.write("The student is failed.");
   }
   else
   {
      document.write("The student is passed.");
   }
</SCRIPT>
</BODY>
</HTML>
```



ചിത്രം 6.8: if പ്രസ്താവന വ്യക്തമാക്കുന്ന വെബ് പേജ്

മുകളിൽ കൊടുത്തിരിക്കുന്ന പ്രോഗ്രാം കോഡ് if - else പ്രസ്താവന ഉപയോഗിക്കുന്ന വിധം കാണിക്കുന്നു. പ്രോഗ്രാമിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.8 ൽ നൽകിയിരിക്കുന്നു. നിങ്ങൾ 30 ൽ താഴെയുള്ള സംഖ്യ score ന്റെ വിലയായി നൽകി ഫലത്തിൽ വരുന്ന മാറ്റം ശ്രദ്ധിക്കുക.

6.6.2 സ്വിച്ച് (switch)

സ്വിച്ച് ഒരു 'മൾട്ടി ബ്രാഞ്ചിങ്' പ്രസ്താവനയാണ്. ഇതുപയോഗിച്ച് ഒരു എക്സ്പ്രഷന്റെ വില അടിസ്ഥാനമാക്കി വ്യത്യസ്ത പ്രോഗ്രാം കോഡുകൾ പ്രവർത്തിപ്പിക്കുന്നതിനായി തിരഞ്ഞെടുക്കാം. സ്വിച്ചിന്റെ വാക്യഘടന

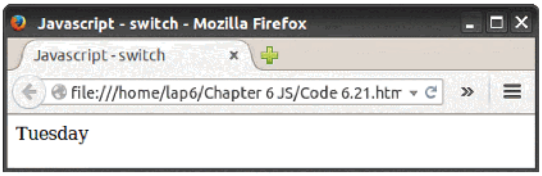
```
   switch  (പരിശോധന പ്രയോഗങ്ങൾ)
   {
      case value1:
               പ്രസ്താവനകൾ;
               break;
      case value2:
               പ്രസ്താവനകൾ;
               break;
      .................
      .................
      default:
               പ്രസ്താവനകൾ;
   }
```

എക്സ്പ്രഷന്റെ വില അടിസ്ഥാനമാക്കി ഉചിതമായ കേസ് ഭാഗം പ്രവർത്തിക്കുന്നു. ഇവിടെ എക്സ്പ്രഷൻ ഒരു വേരിയബിളിന്റെ പേര് ആവാം. നൽകുന്ന നമ്പറിന് അനുയോജ്യമായ ദിവസം പ്രിന്റ് ചെയ്യുന്ന ഒരു HTML കോഡ് താഴെ കൊടുത്തിരിക്കുന്നു.

ഉദാഹരണം 6.8: നൽകുന്ന നമ്പറിന് അനുയോജ്യമായ ദിവസം പ്രിന്റ് ചെയ്യുന്ന വെബ് പേജ് നിർമിക്കുന്നതിനുള്ള HTML കോഡ്

```
<HTML>
<HEAD> <TITLE>Javascript - switch</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var d;
    d = 3;
    switch(d)
    {
        case 1:
            document.write("Sunday");
            break;
        case 2:
            document.write("Monday");
            break;
        case 3:
            document.write("Tuesday");
            break;
        case 4:
            document.write("Wednesday");
            break;
        case 5:
            document.write("Thursday");
            break;
        case 6:
            document.write("Friday");
            break;
        case 7:
            document.write("Saturday");
            break;
        default:
            document.write("Invalid Day");
    }
</SCRIPT>
</BODY>
</HTML>
```

ഈ കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.9 ൽ നൽകിയിരിക്കുന്നു.



ചിത്രം 6.9: സ്വിച്ച് പ്രസ്താവനയുടെ ഉപയോഗം വ്യക്തമാക്കുന്ന വെബ് പേജ്

6.6.3 for ലൂപ്പ് (for loop)

ഒരു കൂട്ടം നിർദ്ദേശങ്ങൾ ആവർത്തിച്ച് നടപ്പാക്കാൻ for ലൂപ്പ് ഉപയോഗിക്കുന്നു. for ലൂപ്പിന്റെ ആവർത്തനം നിയന്ത്രിക്കുന്നതിനായി ഒരു ലൂപ്പ് വേരിയബിൾ ഉപയോഗിയ്ക്കുന്നു.

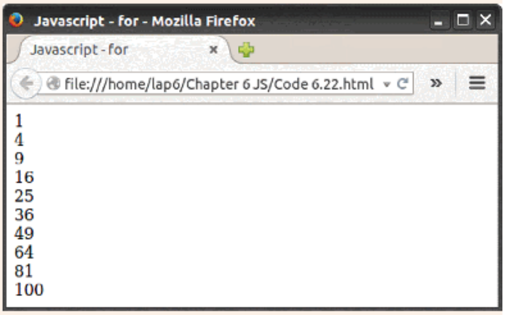
for ലൂപ്പിന്റെ വാക്യഘടന

```
for(initialisation; test_expression; update_statement)
{
    പ്രസ്താവനകൾ;
}
```

initialisation ലൂപ്പ് വേരിയബിളിന് ആദ്യ വില നൽകുവാൻ ഉപയോഗിക്കുന്നു. test_expression ലൂപ്പ് തുടരണോ വേണ്ടയോ എന്ന് പരിശോധിക്കാനുള്ള വ്യവസ്ഥയും update_statement ലൂപ്പ് വേരിയബിളിന്റെ വില വർധിപ്പിക്കുകയോ കുറയ്ക്കുകയോ ചെയ്യുന്നതിനുമുള്ള പ്രസ്താവനയുമാണ്. താഴെ പറയുന്ന ഉദാഹരണം ലൂപ്പിന്റെ പ്രവർത്തനങ്ങൾ വിശദീകരിക്കുന്നു.

ഉദാഹരണം 6.9: ആദ്യത്തെ 10 നമ്പറുകളുടെ വർഗങ്ങൾ പ്രദർശിപ്പിക്കുന്നതിനുള്ള ഒരു വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - for</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var i, s;
    for (i=1; i<=10; i++)
    {
        s = i*i;
        document.write(s);
        document.write("<BR>");
    }
</SCRIPT>
</BODY>
</HTML>
```



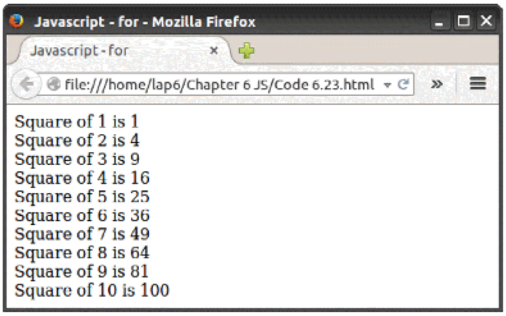
ചിത്രം 6.10: വേരിയബിളുകളുടെ തരം ഉപയോഗം പ്രദർശിപ്പിക്കുന്ന വെബ് പേജ്.

ഇതിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.10 ൽ കൊടുത്തിരിക്കുന്നു.

ചിത്രം 6.11: ഔട്ട്പുട്ട് 'Square of 1 is 1' എന്ന് ലഭിക്കുന്നതിന് മുകളിലുള്ള കോഡ് നമുക്ക് പരിഷ്കരിക്കാം.

```
    for (i=1; i<=10; i++)
    {
        s = i*i;
        document.write("Square of " + i + " is " + s);
        document.write("<BR>");
    }
```

ഔട്ട്പുട്ട് നിർമ്മിക്കുന്നതിന് സ്ട്രിംഗ് കൂട്ടിച്ചേർക്കുന്നതിനുള്ള (സ്ട്രിംഗ് അഡീഷൻ) ഓപ്പറേറ്റർ ആയി + ഉപയോഗിക്കുന്നു.



ചിത്രം 6.11: നമ്പറുകളുടെ വർഗ്ഗം പ്രദർശിപ്പിക്കുന്നതിനുള്ള വെബ് പേജ്

6.6.4 while ലൂപ്പ്

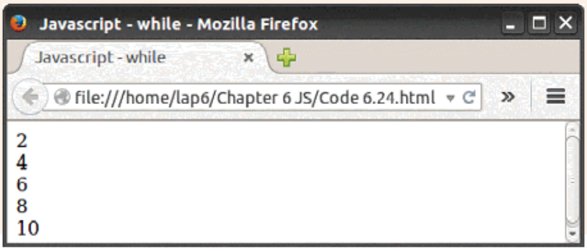
ഒരു വ്യവസ്ഥയുടെ അടിസ്ഥാനത്തിൽ ഒരു കൂട്ടം പ്രസ്താവനകൾ ആവർത്തിച്ച് പ്രവർത്തിപ്പിക്കുന്നതിനാണ് ലൂപ്പ് എന്ന്പറയുന്നത്.

```
    while (test_expression)
    {
        statements;
    }
```

ഇവിടെ test_expression എന്നത് ഒരു വ്യവസ്ഥയാണ്. വ്യവസ്ഥ ശരിയായിരിക്കുന്നിടത്തോളം ലൂപ്പിനുള്ളിലെ പ്രസ്താവനകൾ തുടർച്ചയായി പ്രവർത്തിച്ചുകൊണ്ടിരിക്കും. താഴെക്കാണുന്ന ഉദാഹരണം 10 വരെയുള്ള ഇരട്ട സംഖ്യകൾ പ്രദർശിപ്പിക്കുന്നു.

ഉദാഹരണം 6.10: പത്തു വരെയുള്ള ഇരട്ട സംഖ്യകൾ പ്രദർശിപ്പിക്കുന്നതിനുള്ള വെബ്പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - while</TITLE> </HEAD>
<BODY>
<SCRIPT Language= "JavaScript">
    var i;
    i = 2;
    while (i<=10)
    {
        document.write(i);
        document.write("<BR>");
        i += 2;
    }
</SCRIPT>
</BODY>
</HTML>
```



ചിത്രം 6.12: while ലൂപ്പിന്റെ ഉപയോഗത്തെ വ്യക്തമാക്കുന്ന വെബ് പേജ്

തന്നിരിക്കുന്ന കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.12 ൽ നൽകിയിരിക്കുന്നു.

ജാവാസ്ക്രിപ്റ്റിൽ C++ ലേതിന് സമാനമായ do while ലൂപ്പ് ലഭ്യമാണ്. മിക്കവാറും എല്ലാ ജോലികളും for ലൂപ്പിലും while ലൂപ്പിലും ചെയ്യാമെന്നതിനാൽ ഈ അധ്യായത്തിൽ നമ്മൾ മറ്റ് ലൂപ്പുകൾ ചർച്ച ചെയ്യുന്നില്ല.

ഉദാഹരണം 6.10 ൽ document.write("<BR>"); എന്ന പ്രസ്താവന ഒഴിവാക്കുകയാണെങ്കിൽ എന്ത് ഔട്ട്പുട്ട് ലഭിക്കും;

നമുക്കു ചെയ്യാം

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. ഒരു കൂട്ടം പ്രസ്താവനകൾ തിരഞ്ഞെടുത്ത് പ്രവർത്തിപ്പിക്കുന്നതിന് ________ ഉം ______ ഉം കൺട്രോൾ സ്ട്രക്ചറുകൾ ഉപയോഗിക്കുന്നു.
2. ജാവാസ്ക്രിപ്റ്റിലെ ലൂപ്പിങ് പ്രസ്താവനകൾക്ക് ഉദാഹരണങ്ങൾ എഴുതുക.
3. ___________ ഒരു മൾട്ടി ബ്രാഞ്ചിങ് പ്രസ്താവനയാണ്.
4. ശരിയോ തെറ്റോ എന്നെഴുതുക.
   a. break പ്രസ്താവന switch ബ്ലോക്കിനുള്ളിൽ ഉപയോഗിക്കുന്നു.
   b. switch, പ്രസ്താവന ഉപയോഗിക്കുകയാണെങ്കിൽ, ചുരുങ്ങിയത് ഒരു തവണയെങ്കിലും അതിൽ break പ്രസ്താവന ഉപയോഗിച്ചിരിക്കണം.
   c. if-else ഉപയോഗിച്ച് എഴുതിയ പ്രോഗ്രാമുകൾ എല്ലാം സ്വിച്ച് പ്രസ്താവന ഉപയോഗിച്ച് മാറ്റി എഴുതാം.
   d. switch പ്രസ്താവന ഉപയോഗിച്ച് എഴുതിയ പ്രോഗ്രാമുകൾ എല്ലാം if-else ഉപയോഗിച്ച് മാറ്റി എഴുതാം.
5. for ലൂപ്പും while ലൂപ്പും തമ്മിലുള്ള വ്യത്യാസം എന്താണ്?

6.7 അന്തർ നിർമിത ഫങ്ഷനുകൾ (Built-in function)

ജാവാസ്ക്രിപ്റ്റിൽ ധാരാളം അന്തർനിർമിത ഫങ്ഷനുകൾ ലഭ്യമാണ്. ഫങ്ഷനുകളെ മെത്തേഡ് എന്നും വിളിക്കുന്നു. സാധാരണയായി ഉപയോഗിക്കുന്ന ഏതാനും ഫങ്ഷനുകളെക്കുറിച്ച് ചർച്ച ചെയ്യാം.

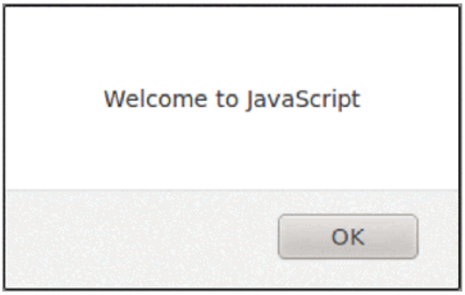
a. alert()

സ്ക്രീനിൽ ഒരു സന്ദേശം പ്രദർശിപ്പിക്കാൻ ഈ ഫങ്ഷൻ ഉപയോഗിക്കുന്നു.

ഉദാഹരണത്തിന് :

```
alert("Welcome to JavaScript");
```

എന്ന പ്രസ്താവന ഉപയോഗിച്ച് ചിത്രം 6.13 ൽ കാണിച്ചിരിക്കുന്ന സന്ദേശ ജാലകം ബ്രൗസറിൽ കാണിക്കാം. ഡാറ്റയുടെ സാധുത ഉറപ്പുവരുത്തുന്ന സമയത്ത് ഉപയോക്താവിന് സന്ദേശം നൽകുന്നതിന് ഈ ഫങ്ഷൻ ഉപയോഗിക്കുന്നു.



ചിത്രം 6.13: സന്ദേശജാലകം

b. isNaN()

തന്നിരിക്കുന്ന വില ഒരു സംഖ്യയാണോ അല്ലയോ എന്നു പരിശോധിക്കുന്നതിന് ഈ ഫങ്ഷൻ ഉപയോഗിക്കുന്നു. ഈ ഫങ്ഷനിൽ, `NaN` എന്നത് നോട്ട് എ നമ്പർ എന്ന് സൂചിപ്പിക്കുന്നു. നൽകിയിരിക്കുന്ന വില ഒരു സംഖ്യയല്ലെങ്കിൽ ഫങ്ഷൻ `true` എന്ന് തിരികെ നൽകുന്നു. ഉദാഹരണത്തിന്,

1. `isNaN("welcome");`
2. `isNaN("A123");`
3. `isNaN("Score50");`
4. `isNaN("A");`

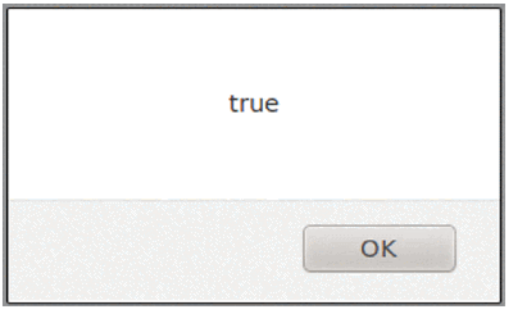
താഴെ പറയുന്ന പ്രസ്താവനകൾ `false` എന്ന വില തിരികെ നൽകുന്നു.

1. `isNaN("13");`
2. `isNaN(13);`
3. `isNaN("13.5");`
4. `isNaN("0.123");`

താഴെ പറയുന്ന പ്രസ്താവന ചിത്രം 6.14 ൽ കാണിച്ചിരിക്കുന്ന സന്ദേശ ജാലകം ബ്രൗസറിൽ പ്രദർശിപ്പിക്കുന്നു.

```
alert(isNaN("A"));
```

ഡാറ്റയുടെ സാധൂകരണത്തിനായി ഈ ഫങ്ഷൻ വളരെ ഉപയോഗപ്രദമാണ്. ഉദാഹരണത്തിന്, ഒരു വിദ്യാർഥിയുടെ പ്രായം നൽകുന്നതിന് വെബ് പേജിൽ ഒരു ടെക്സ്റ്റ് ബോക്സ് ഉണ്ട് എന്ന് കരുതുക, അബദ്ധത്തിൽ അതിൽ സംഖ്യക്ക് പകരം ഒരു ക്യാരക്ടർ നൽകിയേക്കാം. അപ്പോൾ ഈ ഫങ്ഷൻ, ഇൻപുട്ട് നൽകിയത് ഒരു നമ്പർ ആണോയെന്ന് പരിശോധിക്കുകയും ഇത് ഒരു സംഖ്യയല്ലെങ്കിൽ, `alert()` ഫങ്ഷൻ ഉപയോഗിച്ച് ഒരു സന്ദേശം പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്നു.



ചിത്രം 6.14: isNaN() ന്റെ ഔട്ട്പുട്ട്

c. toUpperCase()

ഈ ഫങ്ഷൻ തന്നിരിക്കുന്ന സ്ട്രിംഗിനെ അപ്പർ കേസാക്കി മാറ്റുന്നു. താഴെ തന്നിരിക്കുന്ന ഉദാഹരണം നോക്കുക.

```
var x, y;
x = "JavaScript";
y = x.toUpperCase();
alert(y);
```

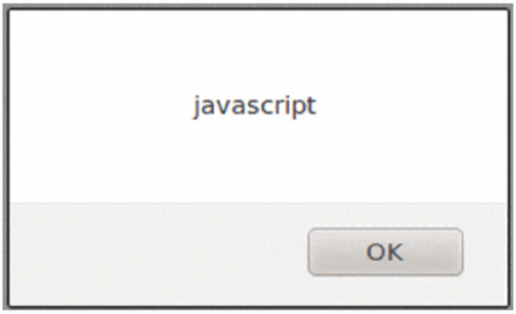
ചിത്രം 6.15: LowerCase() ന്റെ ഔട്ട്പുട്ട്

ഈ കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.15 ൽ കാണിച്ചിരിക്കുന്നു. toUpperCase() എന്ന ഫങ്ഷൻ സ്ട്രിംഗ് വേരിയബിൾ x ന്റെ പേരിനോട് ചേർത്ത് വിളിക്കുന്നു. അതായത് toUpperCase() എന്നത് സ്ട്രിംഗിനെ വലിയ അക്ഷരമാക്കി തിരികെ നൽകുന്നു. ജാവസ്ക്രിപ്റ്റ് കേസ് സെൻസിറ്റീവ് ഭാഷയാണ്. അതിനാൽ കോഡിൽ ഫങ്ഷൻ ഏതു തരം അക്ഷരത്തിലാണോ അതേ രീതിയിൽ തന്നെ ഉപയോഗിക്കണം.

d. toLowerCase()

ഈ ഫങ്ഷൻ തന്നിരിക്കുന്ന സ്ട്രിംഗിനെ ചെറിയ അക്ഷരമാക്കി തിരികെ നൽകുന്നു. താഴെ നൽകിയിരിക്കുന്ന ഉദാഹരണം ശ്രദ്ധിക്കുക

```
var x, y;
x = "JavaScript";
y = x.toLowerCase();
alert(y);
```

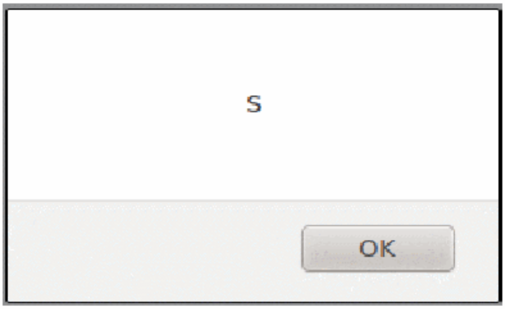


ചിത്രം 6.16: toLowerCase() ന്റെ ഔട്ട്പുട്ട്

മുകളിലെ കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.16ൽ കാണിച്ചിരിക്കുന്നു. സ്ട്രിംഗിൽ എല്ലാ ക്യാരക്ടറുകളും ഇപ്പോൾ ചെറിയ അക്ഷരത്തിലാണ് അതിനാൽ toLowerCase() അതേ സ്ട്രിംഗ് തിരികെ നൽകുന്നു.

e. charAt()

ഈ ഫങ്ഷൻ ഒരു പ്രത്യേക സ്ഥാനത്തുള്ള ക്യാരക്ടർ തിരികെ നൽകുന്നു. charAt(0) എന്നത് സ്ട്രിംഗിലെ ആദ്യത്തെ അക്ഷരം തിരികെ നൽകുന്നു. charAt(1) സ്ട്രിംഗിലെ രണ്ടാമത്തെ അക്ഷരം തിരികെ നൽകുന്നു. ഉദാഹരണം നോക്കുക.



ചിത്രം 6.17: charAt() ന്റെ റിസൽട്ട്

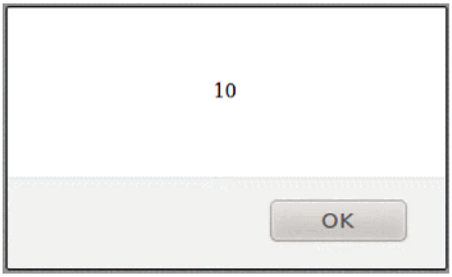
```
var x;
x = "JavaScript";
y = x.charAt(4);
alert(y);
```

ഇതിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.17 ൽ നൽകിയിരിക്കുന്നു. വേരിയബിൾ x ലെ സ്ട്രിംഗിന്റെ അഞ്ചാമത്തെ അക്ഷരം 'S' ആയതിനാൽ ബ്രൗസർ ജാലകത്തിൽ B അക്ഷരം കാണിക്കുന്നു.

f. length പ്രോപ്പർട്ടി

ഫങ്ഷനുകൾ കൂടാതെ, പ്രോഗ്രാമർക്ക് ഉപയോഗപ്രദമായ ചില പ്രോപ്പർട്ടികളും സ്ട്രിംഗ് വേരിയബിളിനോടൊപ്പം ഉപയോഗി ക്കാം. length പ്രോപ്പർട്ടി സ്ട്രിംഗിന്റെ നീളം തിരികെ നൽകുന്നു. സ്ട്രിംഗിലെ ക്യാരക്ടറുകളുടെ എണ്ണമാണ് നീളം എന്നതു കൊണ്ട് ഉദ്ദേശിക്കുന്നത്. ഉദാഹരണത്തിന്

```
var x, n;
x = "JavaScript";
n = x.length;
alert(n);
```



ചിത്രം 6.18: length പ്രോപ്പർട്ടിയുടെ റിസൾട്ട്

ഇവിടെ length പ്രോപ്പർട്ടി എങ്ങനെ വിളിക്കുന്നു എന്ന് നോക്കാം. x.length എന്ന രൂപത്തിൽ x എന്ന വേരിയബിളിന്റെ പേര് ചേർത്തതാണ് ഈ പ്രോപ്പർട്ടി വിളിക്കുന്നത്. മുകളിലെ കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.18 ൽ കാണിച്ചിരിക്കുന്നു. ഫങ്ഷനും പ്രോപ്പർട്ടിയും തമ്മിലുള്ള പ്രധാന വ്യത്യാസം, ഫങ്ഷനോടൊപ്പം () ബ്രാക്കറ്റ് ഉപയോഗിക്കുന്നു (ചില സന്ദർഭങ്ങളിൽ ബ്രാക്കറ്റിൽ പരാമീറ്ററുകളും) എന്നാൽ പ്രോപ്പർട്ടിയോടൊപ്പം () ബ്രാക്കറ്റിന്റെ ആവശ്യം ഇല്ല എന്നതാണ്.

നിങ്ങളുടെ പുരോഗതി അറിയാം

താഴെപ്പറയുന്നവയിൽ വേരിയബിൾ y യുടെ വില എഴുതുക.

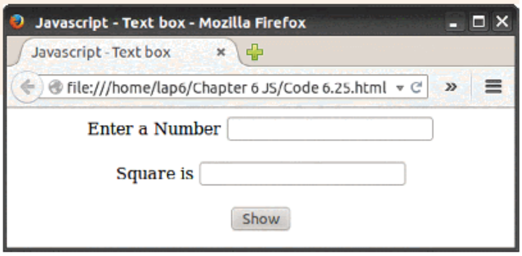
1.
```
x = "welcome";
y = x.length;
```
2.
```
x = "WELCOME";
y = x.toLowerCase();
```
3.
```
x = "Welcome";
y = x.toUpperCase();
```
4.
```
x = "welcome";
y = x.toLowerCase();
```
5.
```
x = "welcome";
y = isNaN(x);
```
6.
```
x = "welcome";
y = charAt(3);
```

6.8 ജാവാസ്ക്രിപ്റ്റിലൂടെ ടെക്സ്റ്റ് ബോക്സിലെ വിലകൾ സ്വീകരിക്കുന്നവിധം (Accessing values in a textbox using JavaScript)

മുൻ അധ്യായത്തിൽ ഒരു വെബ് പേജിൽ ടെക്സ്റ്റ് ബോക്സ്, ചെക്ക്ബോക്സ്, റേഡിയോ ബട്ടൺ, സബ്മിറ്റ് ബട്ടൺ തുടങ്ങിയ വിവിധ കൺട്രോളുകൾ (controls) എങ്ങനെ ഉപയോഗിക്കാം എന്ന് പഠിച്ചു. ഇത്തരം കൺട്രോളുകളെ ജാവാസ്ക്രിപ്റ്റിന്റെ സഹായത്താൽ എങ്ങിനെ ഉപയോഗിക്കാം എന്നാണ് ഇവിടെ ചർച്ചചെയ്യുന്നത്. ഈ അധ്യായത്തിൽ നമ്മൾ ചർച്ച ചെയ്ത ഒരു പ്രോഗ്രാമിലും ഉപയോക്താവിൽ നിന്ന് യാതൊരുവിധ ഇൻപുട്ടും സ്വീകരിച്ചിട്ടില്ല. പ്രോസസ്സിംഗിന് ആവശ്യമായ ഡാറ്റ നേരിട്ട് പ്രോഗ്രാം കോഡിൽ തന്നെ നൽകുകയാണ് ചെയ്തത്. ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിച്ച് ഒരു വെബ് പേജ് ഘടകങ്ങളെ എങ്ങനെ സ്വീകരിക്കുന്നു എന്ന് നോക്കാം. ഈ പാഠഭാഗത്തിനുശേഷം നമുക്ക് ശരിയായ ഒരു ഇന്ററാക്ടീവ് വെബ് പേജ് രൂപപ്പെടുത്തുവാൻ കഴിയും. അതായത്, ഉപയോക്താവ് ടെക്സ്റ്റ് ബോക്സിൽ വില നൽകാനും അതിന്മേൽ ചില പ്രോസസിംങ് നടത്തി ഫലം മറ്റൊരു ടെക്സ്റ്റ് ബോക്സിൽ പ്രദർശിപ്പിക്കാനും കഴിയും. ഈ കാര്യങ്ങൾ മനസ്സിലാക്കുന്നതിന് താഴെ കൊടുത്തിരിക്കുന്ന HTML കോഡ് ശ്രദ്ധാപൂർവം വായിക്കുക.

ഉദാഹരണം 6.11: ഒരു വെബ് ഫോം (web form) പ്രദർശിപ്പിക്കുന്ന ഒരു വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Text box</TITLE> </HEAD>
<BODY>
   <FORM Name= "frmSquare">
      <CENTER>
         Enter a Number
         <INPUT Type= "text" Name= "txtNum">
         <BR><BR>
         Square is
         <INPUT Type= "text" Name= "txtSqr">
         <BR><BR>
         <INPUT Type= "button" Value= "Show">
      </CENTER>
   </FORM>
</BODY>
</HTML>
```



ചിത്രം 6.19: വെബ് ഫോം പ്രദർശിപ്പിക്കുന്ന ഒരു വെബ് പേജ്

ഈ കോഡിന്റെ പ്രവർത്തന ഫലം ചിത്രം 6.19 ൽ കൊടുത്തിരിക്കുന്നു. ഇവിടെ ഫോമിന് frmSquare എന്നും, ടെക്സ്റ്റ് ബോക്സുകൾക്ക് txtNum, txtSquare എന്നും

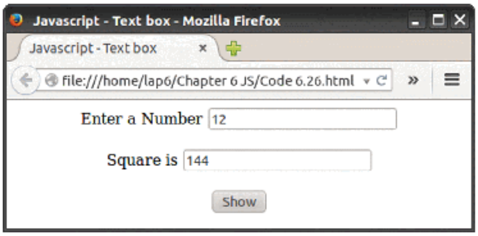
പേരുകൾ നൽകിയിരിക്കുന്നു. ജാവസ്ക്രിപ്റ്റിൽ ഇത്തരം ഘടകങ്ങൾക്ക് ഉപയോഗിക്കുന്നതിന് അവയ്ക്ക് പേര് നൽകേണ്ടത് വളരെ പ്രധാനമാണ്. ഒരു വെബ് പേജ് ഘടകത്തിന് നമ്മൾ പേര് നൽകുന്നില്ലെങ്കിൽ, ജാവസ്ക്രിപ്റ്റിന് അവയെ ഉപയോഗിക്കാൻ കഴിയില്ല. Show എന്ന് എഴുതിയിരിക്കുന്ന സബ്മിറ്റ് ബട്ടന് പേര് നൽകിയിട്ടില്ലെന്നത് ശ്രദ്ധിക്കുക. കാരണം, ഈ ബട്ടൺ ജാവസ്ക്രിപ്റ്റിൽ നിന്ന് പരാമർശിക്കാൻ കഴിയില്ല.

ഇനി നമുക്ക് ഈ പ്രോഗ്രാം കോഡിൽ കുറച്ചു മാറ്റം വരുത്തി നോക്കാം.

ഉദാഹരണം 6.12: നൽകുന്ന സംഖ്യയുടെ വർഗം (സ്ക്വയർ) പ്രദർശിപ്പിക്കുന്നതിനുള്ള ഒരു വെബ് പേജ് നിർമാണം.

```
<HTML>
<HEAD> <TITLE>Javascript - Text box</TITLE>
   <SCRIPT Language= "JavaScript">
      function showSquare()
      {
         var num, ans;
         num = document.frmSquare.txtNum.value;
         ans = num * num;
         document.frmSquare.txtSqr.value = ans;
      }
   </SCRIPT>
</HEAD>
<BODY>
   <FORM Name= "frmSquare">
   <CENTER>
      Enter a Number
      <INPUT Type= "text" Name= "txtNum">
      <BR><BR>
      Square is
      <INPUT Type= "text" Name= "txtSqr">
      <BR><BR>
      <INPUT Type= "button" Value= "Show"
                                 onClick= "showSquare()">
   </CENTER>
   </FORM>
</BODY>
</HTML>
```

മുകളിലുള്ള കോഡ് ശ്രദ്ധാപൂർവം പരിശോധിച്ച് കൂട്ടിച്ചേർത്തിട്ടുള്ളവ ശ്രദ്ധിക്കുക (ഉദാഹരണം 6.17). വെബ് പേജിന്റെ ഹെഡ് ഭാഗത്ത് `showSquare()` എന്ന ഫങ്ഷൻ നിർവചിച്ചിരിക്കുന്നു. താഴെ കൊടുത്തിരിക്കുന്ന രീതിയിൽ ഈ ഫങ്ഷനെ വിളിക്കുന്നു.



ചിത്രം 6.20: ഒരു സംഖ്യയുടെ വർഗം പ്രദർശിപ്പിക്കുന്നതിനുള്ള വെബ് പേജ്

```
<INPUT Type= "button" Value= "Show" onClick= "showSquare()">
```

```
onclick= ""
```

എന്നത് ഇൻപുട്ട് ടാഗിന്റെ ഉള്ളിലാണ് എഴുതിയിട്ടുള്ളത്. ഉപയോക്താവ് ഈ ബട്ടൺ ക്ലിക്കുചെയ്യുമ്പോൾ, `showSquare()` എന്ന ഫങ്ഷൻ പ്രവർത്തിക്കുന്നു. ബട്ടനിൽ ക്ലിക്കുചെയ്തുകൊണ്ട് ഒരു ഫങ്ഷൻ എങ്ങനെ വിളിക്കാം എന്ന് മനസ്സിലായായല്ലോ.

ഫങ്ഷൻ നിർവചനത്തിലെ താഴെപ്പറയുന്ന വരി ശ്രദ്ധിക്കുക

```
num = document.frmSquare.txtNum.value;
```

`document.frmSquare.txtNum.value` നോക്കുക. ഇവിടെ 'document' എന്നത് വെബ് പേജിന്റെ ബോഡി ഭാഗത്തെ സൂചിപ്പിക്കുന്നു. `frmSquare` എന്നത് body ഭാഗത്ത് നൽകിയിട്ടുള്ള ഫോമിന്റെ പേരാണ്. 'txtNum' എന്നത് 'frmSquare'- ൽ ഉള്ള ടെക്സ്റ്റ് ബോക്സിന്റെ പേരാണ്, `value` എന്നത് ടെക്സ്റ്റ് ബോക്സിലെ ഉള്ളടക്കത്തെ പരാമർശിക്കുന്നു. അതായത് `document.frmSquare.txtNum.value` എന്നത് ഡോക്യുമെന്റിലെ `frmSquare` എന്ന ഫോമിലെ `txtNum` ന്റെ വിലയാണ്. മുകളിൽ പറഞ്ഞ വരി ടെക്സ്റ്റ് ബോക്സിലെ വില വേരിയബിൾ `num`-ൽ ലഭ്യമാക്കും.

ഇനി പറയുന്ന വരിയുടെ അർഥം മനസ്സിലാക്കാൻ ശ്രമിക്കുക.

```
document.frmSquare.txtSqr.value = ans;
```

മുകളിലുള്ള വരി രണ്ടാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ `ans` എന്ന വേരിയബിളിന്റെ വില കാണിക്കുന്നു. ബട്ടനിൽ ക്ലിക്കുചെയ്യുമ്പോൾ വെബ്പേജിലെ ആദ്യത്തെ ടെക്സ്റ്റ് ബോക്സിൽ നൽകിയിരിക്കുന്ന സംഖ്യയുടെ വർഗം രണ്ടാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ കാണിക്കുന്നു. ഉപയോക്താവിന് ആദ്യത്തെ ടെക്സ്റ്റ് ബോക്സിൽ ഏത് സംഖ്യയും ടൈപ്പു ചെയ്യാം, അതിന്റെ വർഗ്ഗം കാണുന്നതിന് show ബട്ടൺ ക്ലിക്കുചെയ്യുക. പ്രവർത്തന സമയത്തുള്ള വെബ് പേജിന്റെ സ്ക്രീൻ ഷോട്ട് ചിത്രം 6.20 ൽ കൊടുത്തിരിക്കുന്നു

```
<INPUT Type= "button" Value= "Show" onClick= "showSquare()">
```

എന്നത്

```
<INPUT Type= "button" Value= "Show"
                        onMouseEnter= "showSquare()">
```

എന്നാക്കിയാൽ എന്ത് സംഭവിക്കുമെന്ന് നോക്കാം. ബട്ടനു മുകളിലൂടെ മൗസ് പോയിന്റർ നീക്കുമ്പോൾ ഫങ്ഷൻ പ്രവർത്തിപ്പിക്കുന്നതിനായി വിളിക്കുന്നു. ഫങ്ഷൻ പ്രവർത്തനത്തിനുവേണ്ടി നിങ്ങൾ ബട്ടണിൽ ക്ലിക്ക് ചെയ്യേണ്ടതില്ല. ഒരു വെബ് പേജിലെ ചില

ബട്ടനു മുകളിലൂടെ മൗസ് പോയിന്റർ നീക്കുമ്പോൾ, ബട്ടന്റെ നിറം മാറുന്നത് നിങ്ങൾ ശ്രദ്ധിച്ചിരിക്കാം. ബട്ടന്റെ നിറം മാറുന്നതിനുള്ള ഒരു ജാവാസ്ക്രിപ്റ്റ് ഫങ്ഷൻ എഴുതി അത് `onMouseEnter` എന്ന ഇവന്റിലൂടെ വിളിച്ച് ഇത് സാധ്യമാക്കാം.

<INPUT Type= "button" Value= "Show"onClick= "showSquare()">

`onMouseEnter`, `onClick`, `onMouseEnter`, `onMouseLeave`, `onKeyDown`, `onKeyUp` തുടങ്ങിയവ സാധാരണയായി ഉപയോഗിക്കുന്ന ചില ഇവന്റുകളാണ്.

സാധാരണ ഉപയോഗിക്കുന്ന ചില ജാവ സ്ക്രിപ്റ്റ് ഇവന്റുകളും അവയുടെ വിവരണങ്ങളും പട്ടിക 6.6 ൽ ചേർത്തിരിക്കുന്നു.

ഇവന്റ്	വിവരണം
onClick	ഉപയോക്താവ് ഒരു ഒബ്ജക്റ്റിൽ ക്ലിക്കുചെയ്യുമ്പോൾ സംഭവിക്കുന്നു
onMouseEnter	മൗസ് പോയിന്റർ ഒരു ഒബ്ജക്റ്റിലേക്ക് നീങ്ങുമ്പോൾ സംഭവിക്കുന്നു
onMouseLeave	മൗസ് പോയിന്റർ ഒരു ഒബ്ജക്റ്റിൽ നിന്നും മാറുകയാണെങ്കിൽ സംഭവിക്കുന്നു
onKeyDown	ഉപയോക്താവ് കീബോർഡിലെ ഒരു കീ അമർത്തുമ്പോൾ സംഭവിക്കുന്നു
onKeyUp	ഉപയോക്താവ് കീബോർഡിലെ ഒരു കീയിൽ നിന്ന് വിടുമ്പോൾ സംഭവിക്കുന്നു

പട്ടിക 6.6: സാധാരണ ഉപയോഗിക്കുന്ന ജാവാസ്ക്രിപ്റ്റ് ഇവന്റുകൾ

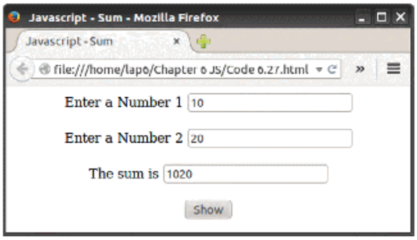
ഒരു വെബ്പേജിൽ രണ്ട് വ്യത്യസ്ത ടെക്സ്റ്റ് ബോക്സുകളിൽ ഉപയോക്താവിന് രണ്ട് സംഖ്യ നൽകാനും, ഒരു ബട്ടണിൽ അമർത്തുമ്പോൾ ഈ സംഖ്യകളുടെ തുക മൂന്നാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ കാണിക്കുന്നതിനുമുള്ള ഒരു വെബ്പേജ് നിർമ്മിക്കാം. കോഡ്, ഔട്ട്പുട്ട് എന്നിവ ചിത്രം 6.21 ൽ കൊടുത്തിരിക്കുന്നു.

ഉദാഹരണം 6.13: രണ്ട് സംഖ്യകളുടെ തുക കാണിക്കുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Sum</TITLE>
   <SCRIPT Language= "JavaScript">
      function showSum()
      {
         var num1, num2, ans;
         num1 = document.frmSum.txtNum1.value;
         num2 = document.frmSum.txtNum2.value;
         ans = num1 + num2;
         document.frmSum.txtSum.value = ans;
      }
   </SCRIPT>
```

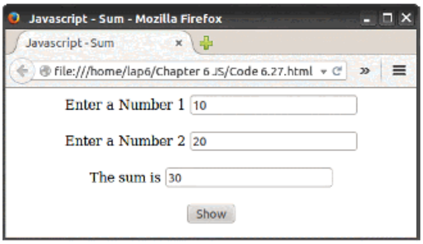
```
</HEAD>
<BODY>
<FORM Name= "frmSum">
    <CENTER>
    Enter a Number 1
    <INPUT Type= "text" Name= "txtNum1">
    <BR><BR>
    Enter a Number 2
    <INPUT Type= "text" Name= "txtNum2">
    <BR><BR>
    The sum is
    <INPUT Type= "text" Name= "txtSum">
    <BR><BR>
    <INPUT Type= "button" Value= "Show" onClick= "showSum()">
    </CENTER>
</FORM>
</BODY>
</HTML>
```

ഈ പ്രോഗ്രാമിന് ഇൻപുട്ടായി 10, 20 എന്നീ സംഖ്യകൾ നൽകിയപ്പോൾ റിസൾട്ടായി 1020 എന്നാണ് കിട്ടിയത് (ചിത്രം 6.21 ൽ കാണിച്ചിരിക്കുന്നു). സ്ട്രിങ്ങുകൾ കൂട്ടിച്ചേർക്കുന്നതിന് + ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു എന്നതിനാലാണിത്. നമ്മൾ രണ്ട് സ്ട്രിങ്ങുകൾ "10"+"20" ചേർക്കുമ്പോൾ ഉത്തരം '1020' ആണല്ലോ. ഒരു, ടെക്സ്റ്റ് ബോക്സിന്റെ തനതുവില എല്ലായ്പ്പോഴും സ്ട്രിങ്ങായിരിക്കും അതിനാൽ ടെക്സ്റ്റ് ബോക്സിലെ ഉള്ളടക്കം ഒരു നമ്പർ ആയിരുന്നാലും ഒരു വേരിയബിളിലേക്ക് നൽകുമ്പോൾ അതിനെ സ്ട്രിംഗ് ആയി മാത്രമേ പരിഗണിക്കൂ. രണ്ട് സംഖ്യകളുടെ തുക കിട്ടുന്നതിന് ഫങ്ഷൻ showSum() താഴെ കൊടുത്തിരിക്കുന്ന രീതിയിൽ പരിഷ്കരിക്കാം.



ചിത്രം 6.21 : ഓപ്പറേറ്റർ ഉപയോഗത്തെ ചിത്രീകരിക്കുന്നതിന് വെബ് പേജ്

```
    function showSum()
    {
        var num1, num2, ans;
        num1 = Number(document.frmSum.txtNum1.value);
        num2 = Number(document.frmSum.txtNum2.value);
        ans = num1 + num2;
document.frmSum.txtSum.value = ans;
    }
```



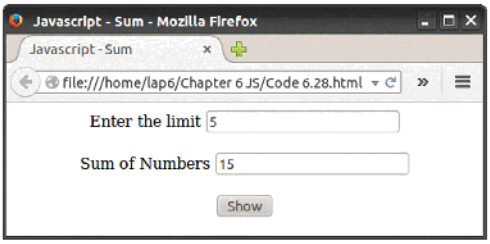
ചിത്രം 6.22: രണ്ട് സംഖ്യകളുടെ തുക കാണിക്കുന്ന വെബ് പേജ്

ഇവിടെ Number() എന്ന ഫങ്ഷൻ, ഡാറ്റയെ സംഖ്യയായി മാറ്റി ആ സംഖ്യ num1 എന്ന വേരിയബിളിലേക്ക് നൽകുന്നു. ഇവിടെ num1, num2 എന്നിവയെ നമ്പർ തരം ഡാറ്റയായി കണക്കാക്കുകയും അതുവഴി ശരിയായ തുക കിട്ടുകയും ചെയ്യും. ഉദാഹരണം 6.13 ലെ showSum() എന്ന ഫങ്ഷനിൽ മാറ്റം വരുത്തിയാൽ ചിത്രം 6.22 ൽ കാണിച്ചിരിക്കുന്ന വെബ് പേജ് ഔട്ട്പുട്ടായി ലഭിക്കും. തന്നിരിക്കുന്ന പരിധി വരെയുള്ള സംഖ്യകളുടെ തുക പ്രദർശിപ്പിക്കുന്നതിനുള്ള ഒരു വെബ് പേജ് നിർമിക്കാം. ഇതിൽ പരിധി ഉപയോക്താവിന് ടെക്സ്റ്റ് ബോക്സിലൂടെ നൽകാൻ കഴിയണം.

ഉദാഹരണം 6.14: തന്നിരിക്കുന്ന പരിധിവരെയുള്ള സംഖ്യകളുടെ തുക കാണിക്കുന്നതിനുള്ള വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Sum</TITLE>
   <SCRIPT Language= "JavaScript">
      function sumLimit()
      {
         var sum = 0, i, limit;
         limit = Number(document.frmSum.txtLimit.value);
         for(i = 1; i <= limit; i++)
            sum += i;
         document.frmSum.txtSum.value = sum;
      }
   </SCRIPT>
</HEAD>
<BODY>
   <FORM Name= "frmSum">
      <CENTER>
         Enter the limit
         <INPUT Type= "text" Name= "txtLimit">
         <BR><BR>
         Sum of Numbers
         <INPUT Type= "text" Name= "txtSum">
         <BR><BR>
         <INPUT Type= "button" Value= "Show"
                                    onClick= "sumLimit()">
      </CENTER>
   </FORM>
</BODY>
</HTML>
```

മുകളിലെ കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.23 ൽ നൽകിയിരിക്കുന്നു. ഈ വെബ് പേജിൽ ആദ്യത്തെ ടെക്സ്റ്റ് ബോക്സിൽ ഉപയോക്താവ് പരിധി നൽകണം. അതിനു ശേഷം **show** ബട്ടണിൽ ക്ലിക്കുചെയ്യുമ്പോൾ നൽകിയ പരിധി വരെയുള്ള സംഖ്യകളുടെ തുക രണ്ടാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ കാണിക്കുന്നു. ആദ്യത്തെ ടെക്സ്റ്റ് ബോക്സിൽ ഉപയോക്താവ് പരിധി നൽകാതെ **show** ബട്ടൺ ക്ലിക്കുചെയ്താൽ അത് ഒരു സന്ദേശവും പ്രദർശിപ്പിക്കുന്നില്ല. പരിധി ഇല്ലെങ്കിൽ തുക പൂജ്യമായിരിക്കും കാരണം ആദ്യത്തെ ടെക്സ്റ്റ് ബോക്സ് ശൂന്യമാകുമ്പോൾ `Number()` ഫങ്ഷൻ ശൂന്യ വിലയെ പൂജ്യമായി മാറ്റുന്നു. അതായത് `document.frmSum.txtLimit.value` ന്റെ വില പൂജ്യമായിരിക്കും. അങ്ങനെ ലൂപ്പ് പ്രവർത്തിക്കാതിരിക്കുകയും `sum` എന്ന വേരിയബിളിന്റെ ആദ്യ വിലയായ പൂജ്യം രണ്ടാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്നു. ജാവാസ്ക്രിപ്റ്റ് ഒരിക്കലും ഒരു തെറ്റ് ഉണ്ടെന്ന് കാണിച്ച് തരില്ല. സ്ക്രിപ്റ്റ് എഞ്ചിന് എന്തെങ്കിലും നിർദേശം പ്രാവർത്തികമാക്കാൻ കഴിയാത്തപ്പോൾ, അത് ഫങ്ഷനിലെ ആ വരിയും ബാക്കി ഭാഗങ്ങളും അവഗണിക്കും.



ചിത്രം 6.23: നൽകിയ പരിധിവരെയുള്ള സംഖ്യകളുടെ തുക കാണിക്കുന്ന വെബ് പേജ്

താഴെ കൊടുത്തിരിക്കുന്ന `sumLimit()` ഫങ്ഷൻ പരിശോധിക്കാം.

```
function sumLimit()
{
    var sum = 0, i, limit;
    if (document.frmSum.txtLimit.value == "")
    {
        alert("Please enter the limit!");
        return;
    }
    limit = Number(document.frmSum.txtLimit.value);
    for(i=1; i<=limit; i++)
        sum += i;
    document.frmSum.txtSum.value = sum;
}
```

പരിധി നൽകാതെ നിങ്ങൾ show ബട്ടണിൽ ക്ലിക്കുചെയ്താൽ, ഈ കോഡ് പരിധി നൽകാൻ ഓർമ്മപ്പെടുത്തുന്ന ഒരു സന്ദേശം കാണിക്കും. ബാക്കി നിർദേശങ്ങൾ ഒഴിവാക്കിക്കൊണ്ട് ഫങ്ഷനിൽ നിന്ന് പുറത്തുകടക്കുവാൻ `return` പ്രസ്താവന ഉപയോഗിക്കുന്നു. ഇത് C++ ൽ ഉപയോഗിക്കുന്ന റിട്ടേൺ പ്രസ്താവനയ്ക്ക് സമാനമാണ്. ഇനി നമുക്ക് ടെക്സ്റ്റ് ബോക്സിൽ നൽകിയ പരിധി അക്ഷരമാണോ സംഖ്യയാണോ എന്ന് പരിശോധിക്കുന്നതിനായി `isNaN()` എന്ന ഫങ്ഷൻ ഉപയോഗിക്കാം. ഉപയോക്താവ് ശരിയായ ഡാറ്റ നൽകിയിട്ടുണ്ടോ എന്ന് പരിശോധിക്കുന്നതിന് ഇത് സഹായിക്കുന്നു.

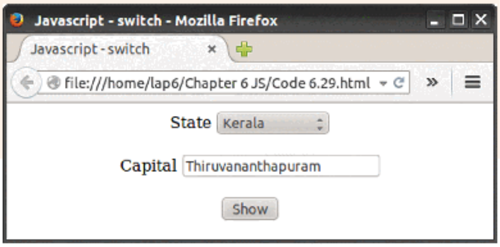
```
function sumLimit()
{
    var sum = 0, i, limit;
    if (document.frmSum.txtLimit.value == "")
    {
        alert("Please enter the limit!");
        return;
    }
    if (isNaN(document.frmSum.txtLimit.value))
    {
        alert("Please enter a number as the limit!");
        return;
    }
    limit = Number(document.frmSum.txtLimit.value);
    for(i = 1; i <= limit; i++)
        sum += i;
    document.frmSum.txtSum.value = sum;
}
```

ക്ലൈന്റ് ഭാഗത്തെ ഡാറ്റയുടെ സാധുത വിലയിരുത്തുന്നതിനായി ജാവ സ്ക്രിപ്റ്റ് ഉപയോഗിക്കുന്നു. ആവശ്യമുള്ള എല്ലാ ഡാറ്റയും നൽകിയിട്ടുണ്ടോ, നൽകിയ ഡാറ്റ ശരിയായ ഘടനയിലാണോ തുടങ്ങിയവ പരിശോധിക്കാനായി ജാവാസ്ക്രിപ്റ്റ് ഉപയോഗിക്കാം. ഏതെങ്കിലും പോരായ്മ ഉണ്ടെങ്കിൽ അത് ഓർമ്മിപ്പിക്കുന്ന ഒരു സന്ദേശം കാണിക്കുവാൻ കഴിയും. താഴെക്കൊടുത്തിരിക്കുന്ന ഉദാഹരണത്തിൽ ജാവാസ്ക്രിപ്റ്റിലെ ഒരു ഡ്രോപ്പ് ഡൗൺ ലിസ്റ്റ് ഉപയോഗിക്കുന്നു ഈ ഡ്രോപ്പ് ഡൌൺ ലിസ്റ്റ് ഉപയോഗിച്ച് ഒരു സംസ്ഥാനം തെരഞ്ഞെടുക്കാനും show ബട്ടണിൽ ക്ലിക്കുചെയ്യുമ്പോൾ, തെരഞ്ഞെടുത്ത സംസ്ഥാനത്തിന്റെ തലസ്ഥാനം ഒരു ടെക്സ്റ്റ് ബോക്സിൽ പ്രദർശിപ്പിക്കാനുമുള്ള ഒരു വെബ്പേജ് നൽകിയിരിക്കുന്നു. ഇതിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.24 ൽ കൊടുത്തിരിക്കുന്നു.

ഉദാഹരണം 6.15: ഒരു സംസ്ഥാനത്തിന്റെ തലസ്ഥാനം പ്രദർശിപ്പിക്കുന്നതിനുള്ള വെബ്പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - switch</TITLE>
<SCRIPT Language= "JavaScript">
function capital()
{
    var n, answer;
    n = document.frmCapital.cboState.selectedIndex;
    switch (n)
    {
```

```
    case 0:
        answer = "Thiruvananthapuram";
        break;
    case 1:
        answer = "Bengaluru";
        break;
    case 2:
        answer = "Chennai";
        break;
    case 3:
        answer = "Mumbai";
        break;
    }
    document.frmCapital.txtCapital.value = answer;
}
</SCRIPT>
</HEAD>
<BODY>
    <FORM Name= "frmCapital">
    <CENTER> State
        <SELECT Size= 1 Name= "cboState">
            <OPTION>Kerala</OPTION>
            <OPTION>Karnataka</OPTION>
            <OPTION>Tamilnadu</OPTION>
            <OPTION>Maharashtra</OPTION>
        </SELECT>
        <BR><BR>
        Capital
        <INPUT Type= "text" Name= "txtCapital">
        <BR><BR>
        <INPUT Type= "button" Value= "Show" onClick= "capital()">
    </CENTER>
    </FORM>
</BODY>
</HTML>
```



ചിത്രം 6.24: സംസ്ഥാനത്തിന്റെ തലസ്ഥാനം പ്രദർശിപ്പിക്കുന്നതിനുള്ള വെബ് പേജ്

ഈ പ്രോഗ്രാമിലെ താഴെ കൊടുത്തിരിക്കുന്ന പ്രസ്താവന പരിഗണിക്കുക.

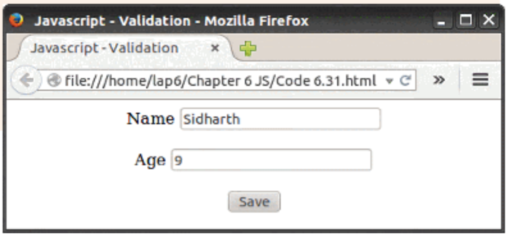
```
document.frmCapital.cboState.selectedIndex;
```

ഇവിടെ 'cboState' എന്നത് ഡ്രോപ്ഡൌൺ ലിസ്റ്റിന്റെ പേരാണ്. 'selectedIndex' എന്നത് ഡ്രോപ് ഡൌൺ ലിസ്റ്റിന്റെ സൂചിക കാണിക്കുന്നു. ഒന്നാമത്തെ ഇനം തിരഞ്ഞെടുത്താൽ സൂചിക 0-ഉം രണ്ടാമത്തെ ഇനം തിരഞ്ഞെടുത്താൽ, സൂചിക 1 ഉം ആയി കണക്കാക്കുന്നു. മുകളിലെ വരി തിരഞ്ഞെടുത്ത വേരിയബിളിന്റെ ഇൻഡക്സ് വേരിയബിൾ n ൽ നൽകുന്നു. വിദ്യാർഥിയുടെ പേരും പ്രായവും നൽകാൻ ഉപയോക്താവിനെ അനുവദിക്കുന്ന വെബ് പേജാണ് താഴെ കൊടുത്തിരിക്കുന്നത്. പേരിൽ കുറഞ്ഞത് 5 അക്ഷരങ്ങൾ ഉണ്ടായിരിക്കണം. പ്രായം 15 മുതൽ 20 വരെ ഉള്ള ഒരു സംഖ്യയായിരിക്കണം.

ഉദാഹരണം 6.16: പേരിന്റെയും പ്രായത്തിന്റെയും സാധുത പരിശോധിക്കുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Validation</TITLE>
<SCRIPT Language= "JavaScript">
function checkData()
{
   var T_name, T_age, N_age;
   T_name = document.frmValid.txtName.value;
   if (T_name == "")
   {
      alert("Please enter name!");
      return;
   }
   if (T_name.length < 5)
   {
      alert("Name must contain at least 5 characters!");
      return;
   }
   T_age = document.frmValid.txtAge.value;
   if (T_age == "")
   {
      alert("Please enter age!");
      return;
   }
   if (isNaN(T_age))
   {
      alert("Please enter a number as the age!");
      return;
   }
   N_age = Number(T_age);
```

```
    if (N_age < 15 || N_age > 20)
    {
        alert("The age must be between 15 and 20!");
        return;
    }
}
</SCRIPT>
</HEAD>
<BODY>
    <FORM Name= "frmValid">
        <CENTER>Name
            <INPUT Type= "text" Name= "txtName">
            <BR><BR>
            Age
            <INPUT Type= "text" Name= "txtAge">
            <BR><BR>
            <INPUT Type= "button" Value= "Save"
                                        onClick= "checkData()">
        </CENTER>
    </FORM>
</BODY>
</HTML>
```



ചിത്രം 6.25: ലൂപ്പിന്റെ ഉപയോഗം വിശദീകരിക്കുന്ന വെബ് പേജ്

HTML കോഡിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.25 ൽ കൊടുത്തിരിക്കുന്നു. ഡാറ്റയുടെ എല്ലാ സാധുതകളും ഇത് പരിശോധിക്കുന്നു. ആദ്യം, പേരിന്റെ ഫീൽഡിൽ ഒരു വില ഉണ്ടോ എന്ന് പരിശോധിക്കുന്നു. പിന്നീട് പേരിന്റെ ദൈർഘ്യം കുറഞ്ഞത് 5 അക്ഷരങ്ങൾ ഉണ്ടോ എന്ന് പരിശോധിക്കുന്നു. ശേഷം വയസ്സ് നൽകിയിട്ടുണ്ടോ എന്ന് പരിശോധിക്കുന്നു. പിന്നീട് നൽകിയ ഡാറ്റ സംഖ്യ ആണോ അല്ലയോ എന്ന് പരിശോധിക്കുന്നു. അന്തിമമായി പ്രായപരിധി 15 മുതൽ 20 വരെ ആണോ എന്ന് പരിശോധിക്കുന്നു.

6.9 സ്ക്രിപ്റ്റുകൾ വെബ്പേജിൽ ചേർക്കുന്നതിനുള്ള വഴികൾ (Ways to add scripts to a webpage)

പല രീതിയിൽ HTML കോഡുകൾക്കുള്ളിൽ സ്ക്രിപ്റ്റുകൾ ഉൾപ്പെടുത്താവുന്നതാണ്. മുൻ ഉദാഹരണങ്ങളിൽ നമ്മൾ ജാവാസ്ക്രിപ്റ്റ് വെബ് പേജിന്റെ ഹെഡ് ഭാഗത്താണ് ഉൾപ്പെടുത്തിയിരുന്നത്. ഇതിന് പുറമേ, സ്ക്രിപ്റ്റുകൾ <BODY> ടാഗിലോ അല്ലെങ്കിൽ ഒരു ബാഹ്യ ഫയലിലോ നൽകാവുന്നതാണ്.

ഇനി വെബ്പേജുകളിൽ സ്ക്രിപ്റ്റുകൾ കൂട്ടിച്ചേർക്കുന്നതിനുള്ള വ്യത്യസ്ത രീതികൾ പരിചയപ്പെടാം.

6.9.1 <BODY> ടാഗിന് ഉള്ളിൽ (Inside <BODY>)

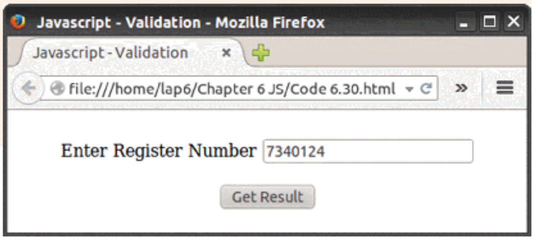
ഈ അധ്യായത്തിന്റെ തുടക്കത്തിൽ <BODY> ടാഗിന് ഉള്ളിൽ സ്ക്രിപ്റ്റുകൾ സ്ഥാപിക്കുന്നത് നാം ചർച്ചചെയ്തു. വെബ്പേജിന്റെ ഉള്ളടക്കം ബ്രൗസറിൽ പ്രദർശിപ്പിക്കുമ്പോൾ സ്ക്രിപ്റ്റുകളും പ്രവർത്തിക്കും. വെബ് പേജ് ഡോക്യുമെന്റിന്റെ ആരംഭം മുതൽക്കാണ് പ്രദർശിപ്പിക്കുക. ഒരു സ്ക്രിപ്റ്റ് കോഡ് കാണുമ്പോൾ അത് പ്രവർത്തിക്കുകയും തുടർന്ന് വെബ് പേജിന്റെ ബാക്കി ഭാഗം പ്രവർത്തിക്കുകയും ചെയ്യുന്നു.

ഒരു ഉദാഹരണം ഉപയോഗിച്ച് ഈ രീതി മനസ്സിലാക്കാം. ഒരു വിദ്യാർഥിയുടെ റിസൾട്ട് ലഭിക്കുന്നതിനുള്ള ഒരു വെബ് പേജാണ് താഴെ കൊടുത്തിരിക്കുന്നത്. ടെക്സ്റ്റ് ബോക്സിൽ ഉപയോക്താവ് ഒരു രജിസ്റ്റർ നമ്പർ നൽകണം. **Get Result** ബട്ടനിൽ ക്ലിക്കു ചെയ്യുമ്പോൾ രജിസ്റ്റർ നമ്പർ ബോക്സിൽ ഏതെങ്കിലും ഡാറ്റ ഉണ്ടോ എന്ന് പരിശോധിക്കാനുള്ള ജാവാസ്ക്രിപ്റ്റ് ഫങ്ഷൻ പ്രവർത്തിക്കണം. ടെക്സ്റ്റ് ബോക്സിൽ വില ഉണ്ടെങ്കിൽ അത് ഏഴ് അക്കമുള്ള ഒരു സംഖ്യ തന്നെ ആയിരിക്കണം. ഈ കാര്യങ്ങൾ ഫങ്ഷൻ പരിശോധിക്കണം. വെബ് പേജിന്റെ ഔട്ട്പുട്ട് ചിത്രം 6.26 ൽ നൽകിയിരിക്കുന്നു.

ഉദാഹരണം 6.17: സാധുതാപരിശോധനയ്ക്കു ശേഷം രജിസ്റ്റർ നമ്പർ സ്വീകരിക്കുന്ന വെബ് പേജ് നിർമാണം

```
<HTML>
<HEAD> <TITLE>Javascript - Validation</TITLE> </HEAD>
<BODY>
   <FORM Name= "frmValid">
      <SCRIPT Language= "JavaScript">
         function checkData()
         {
            var rno;
            rno = document.frmValid.txtRegno.value;
            if (rno == "")
            {
               alert("Please enter Register No.");
               return;
            }
            if (isNaN(rno))
            {
               alert("Invalid Register No.");
               return;
            }
            if (rno.length < 7)
```

```
            {
                alert("The Register No. must have 7 digits");
                return;
            }
        }
    </SCRIPT>
    <CENTER>
        <BR>Enter Register Number
        <INPUT Type= "text" Name= "txtRegno">
        <BR><BR>
        <INPUT Type= "button" Value= "Get Result"
                                onClick= "checkData()">
    </CENTER>
  </FORM>
</BODY>
</HTML>
```



ചിത്രം 6.26: രജിസ്റ്റർ നമ്പർ സ്വീകരിക്കാനുള്ള ഫോം പ്രദർശിപ്പിക്കുന്ന വെബ് പേജ്

സ്ക്രിപ്റ്റ് <BODY> ടാഗിനുള്ളിൽ അവസാനഭാഗത്തും ആകാം. <BODY> ടാഗിൽ അല്ലെങ്കിൽ <HEAD> ടാഗിൽ സ്ക്രിപ്റ്റുകൾ എഴുതിയിട്ടുണ്ടെങ്കിൽ അവ HTML കോഡിനൊപ്പം ബ്രൗസറിലെത്തുകയും ചെയ്യും. വെബ്പേജ് പ്രദർശിപ്പിക്കുന്നത് വൈകുവാൻ ഇത് കാരണമാകുന്നു. സ്ക്രിപ്റ്റുകൾ </BODY> ടാഗിന് മുമ്പായി കൊടുക്കുകയാണെങ്കിൽ വെബ്പേജിലെ ഉള്ളടക്കങ്ങൾ അക്ഷരങ്ങൾ, ചിത്രങ്ങൾ പോലുള്ളവ സ്ക്രീനിൽ വേഗത്തിൽ ദൃശ്യമാകും. എന്നാൽ വെബ് പേജ് ലഭ്യമാക്കുന്നതിനിടയിൽ സ്ക്രിപ്റ്റുകൾ പ്രവർത്തിക്കണമെന്നില്ല.

6.9.2 <HEAD> ടാഗിന് ഉള്ളിൽ (Inside <HEAD>)

സാധാരണ രീതിയിൽ വെബ് പേജിന്റെ ഹെഡ് ഭാഗത്താണ് സ്ക്രിപ്റ്റുകൾ ഉൾപ്പെടുത്തുന്നത്. നാം ചർച്ച ചെയ്ത എല്ലാ ഉദാഹരണങ്ങളിലും അങ്ങനെ തന്നെയാണ് ചെയ്തിരിക്കുന്നത്. ഇതിന് പ്രധാന കാരണം മിക്ക HTML പേജുകളുടെയും ബോഡി ഭാഗത്തിനുള്ളിൽ ഉള്ളടക്കത്തെ സൂചിപ്പിക്കുന്ന ഒരു വലിയ അളവ് ടെക്സ്റ്റ് ദൃശ്യമാക്കേണ്ടതായിട്ടുണ്ട്. ഫങ്ഷൻ നിർവചനം കൂടി ഇവിടെ ചേർക്കുകയാണെങ്കിൽ വെബ്പേജിൽ മാറ്റം വരുത്തുമ്പോൾ ഇത് രൂപകല്പനചെയ്യുന്നയാൾക്ക് ആശയക്കുഴപ്പം സൃഷ്ടിക്കും. ബോഡി ഭാഗത്തിന് മുൻപായി ഒരു വെബ് പേജിന്റെ ഹെഡ് ഭാഗം ലോഡ് ചെയ്യുന്നു എന്നത് ഇതിന്റെ നേട്ടമാണ്. അതിനാൽ, ബോഡി ഭാഗത്ത് വിളിക്കപ്പെടുന്ന ഫങ്ഷനുകളുടെ നിർവചനം നേരത്തെ തന്നെ മെമ്മറിയിൽ ലഭ്യമാക്കിയിരിക്കും. ഉദാഹരണം 6.17 ൽ <SCRIPT> നും </SCRIPT> നും ഇടയിലുള്ള കോഡ് HTML കോഡിന്റെ ഹെഡ് ഭാഗത്തേക്ക് മാറ്റി ഉപയോഗിക്കാവുന്നതാണ്.

6.9.3 ബാഹ്യ ജാവാസ്ക്രിപ്റ്റ് ഫയൽ (External JavaSript file)

HTML ഡോക്യുമെന്റിലെ എല്ലാ സ്ക്രിപ്റ്റുകളും ഒരു ബാഹ്യ ഫയലിലേക്ക് ശേഖരിക്കുകയും ഡോക്യുമെന്റിൽ നിന്ന് ആ ഫയലിലേക്ക് ഒരു ലിങ്ക് സ്ഥാപിക്കുകയും ചെയ്യാം. ഈ ഫയൽ '.js' എന്ന എക്സ്റ്റൻഷനോടുകൂടി സൂക്ഷിക്കുക. ബാഹ്യ ഫയലുകളിൽ ജാവാസ്ക്രിപ്റ്റ് സൗകര്യം ഉപയോഗിക്കുന്നത് കൊണ്ട് ചില ഗുണങ്ങളുണ്ട്. ഒരേ സ്ക്രിപ്റ്റ് ഒന്നിലധികം HTML പേജുകളിലോ ഒരു മുഴുവൻ വെബ് സൈറ്റിലോ ആവർത്തിക്കപ്പെടുമ്പോൾ ഇത് ഉപയോഗപ്രദമാണ്. ഇത് HTML ഉം ജാവാസ്ക്രിപ്റ്റും വേർതിരിച്ച് കാണാൻ സഹായിക്കുന്നു. ഇത് HTML കോഡിനെയും ജാവാസ്ക്രിപ്റ്റിനെയും വേർതിരിക്കുവാനും ഇവ രണ്ടിനെയും എളുപ്പത്തിൽ വായിക്കാനും നിലനിർത്താനും സഹായിക്കുകയും ചെയ്യുന്നു. ജാവാസ്ക്രിപ്റ്റ് ബാഹ്യ ഫയലുകളിൽ സംഭരിക്കുന്നത് പേജ് വേഗത്തിൽ ലഭ്യമാക്കാൻ സഹായിക്കും.

മുകളിൽ പറഞ്ഞിരിക്കുന്ന ഉദാഹരണം 6.17 ൽ ജാവാസ്ക്രിപ്റ്റ് കോഡ് "check.js" എന്ന പേരിൽ ഒരു പ്രത്യേക ഫയൽ ആയി സൂക്ഷിക്കുന്നു. ഈ ഫയലിന്റെ ഉള്ളടക്കം താഴെ കാണിച്ച രീതിയിലാണ്.

```
function checkData()
{
    var rno;
    rno = document.frmValid.txtRegno.value;
    if (rno == "")
    {
        alert("Please enter Register No.");
        return;
    }
    if (isNaN(rno))
    {
        alert("Invalid Register No.");
        return;
    }
    if (rno.length < 7)
    {
        alert("The Register No. must have 7 digits");
        return;
    }
}
```

ഈ ഫയലിൽ ജാവാസ്ക്രിപ്റ്റ് കോഡ് മാത്രമേ ഉള്ളൂ എന്നും <SCRIPT> ടാഗ് ഉപയോഗിച്ചിട്ടില്ല എന്നും ശ്രദ്ധിക്കുക. HTML ഫയലിനുള്ളിൽ മാത്രമാണ് <SCRIPT> ടാഗ് ഉപയോഗിക്കുന്നത്. <SCRIPT> ടാഗ് ഉപയോഗിച്ച് ജാവാസ്ക്രിപ്റ്റ് ഫയലിനെ HTML ഫയലുമായി ലിങ്ക് ചെയ്യാവുന്നതാണ്. Type ആട്രിബ്യൂട്ട് സൂചിപ്പിക്കുന്നത് ലിങ്കു

ചെയ്തിരിക്കുന്ന ഫയൽ ഒരു ജാവാസ്ക്രിപ്റ്റ് ഫയൽ ആണെന്നതാണ്. Src ആട്രിബ്യൂട്ട് ബാഹ്യ ജാവാസ്ക്രിപ്റ്റ് ഫയലിന്റെ സ്ഥാനവും ഫയലിന്റെ പേരും വ്യക്തമാക്കുന്നു. പരിഷ്കരിച്ച HTML കോഡ് താഴെ കൊടുത്തിരിക്കുന്നു.

```
<HTML>
<HEAD><TITLE>Javascript - Validation</TITLE>
   <SCRIPT Type= "text/JavaScript" Src= "checkdata.js">
   </SCRIPT>
</HEAD>
<BODY>
   <FORM Name= "frmValid">
      <CENTER>
         <BR>Enter Register Number
         <INPUT Type= "text" Name= "txtRegno">
         <BR><BR>
         <INPUT Type= "button" Value= "Get Result"
                                onClick= "checkData()">
      </CENTER>
   </FORM>
</BODY>
</HTML>
```

<SCRIPT> ടാഗിൽ Src ആട്രിബ്യൂട്ട് ഉണ്ടെങ്കിൽ അതിലെ ഉള്ളടക്കം അവഗണിക്കപ്പെടും. അതായത്, നിങ്ങൾക്ക് ഒറ്റ <SCRIPT> ടാഗുപയോഗിച്ച് ബാഹ്യ ഫയൽ ചേർക്കാനും ഒരു കോഡ് നടപ്പാക്കാനും ഒരുമിച്ച് സാധ്യമല്ല. ഇതിനായി രണ്ട് വ്യത്യസ്ത <SCRIPT> ടാഗുകൾ ആവശ്യമാണ്. ഇവയിൽ ഒന്ന് Src ഉപയോഗിച്ച് ഒരു ബാഹ്യ ഫയൽ ചേർക്കാനും മറ്റൊന്ന് കോഡ് ചേർക്കാനും ഉപയോഗിക്കാം

നിങ്ങളുടെ പുരോഗതി അറിയാം

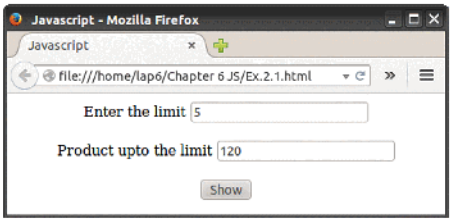
1. ജാവാസ്ക്രിപ്റ്റിൽ onMouseEnter ഇവന്റ് എപ്പോഴാണ് പ്രവർത്തിക്കുക?
2. ഡ്രോപ്ഡൗൺ ലിസ്റ്റിൽ <SELECT> നിന്നും തിരഞ്ഞെടുത്ത ഇനത്തിന്റെ സൂചിക ലഭിക്കാനുള്ള പ്രോപ്പർട്ടി ____________ ആണ്.
3. ജാവാസ്ക്രിപ്റ്റിലെ onKeyDown, onKeyUp എന്നിവ താരതമ്യം ചെയ്യുക.
4. Number() ഫങ്ഷൻ ഉപയോഗിക്കുന്നത് എന്തിനാണ്?
5. ജാവാസ്ക്രിപ്റ്റ് കോഡ് </BODY> ടാഗിന് മുൻപായി ഉപയോഗിക്കുന്നതിന്റെ ഗുണം എന്ത്?

നമുക്കു സംഗ്രഹിക്കാം

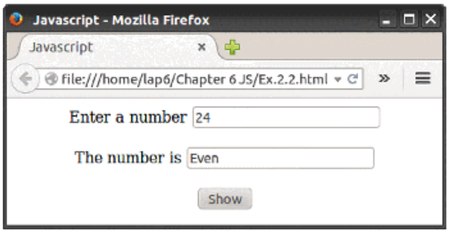
ഡാറ്റയുടെ സാധൂകരണത്തിനായി ഉപയോഗിക്കുന്ന ഒരു ക്ലൈന്റ് ഭാഗം സ്ക്രിപ്റ്റിംഗ് ഭാഷയായിട്ടാണ് ജാവാസ്ക്രിപ്റ്റിനെ ഈ അധ്യായത്തിൽ അവതരിപ്പിക്കുന്നത്. ജാവാ സ്ക്രിപ്റ്റ് കോഡ് HTML ൽ ഉൾപ്പെടുത്തുന്നതിന് ഉപയോഗിക്കുന്ന ടാഗുകളും ജാവാ സ്ക്രിപ്റ്റിലെ പ്രധാന ഫങ്ഷനുകളും ഇവിടെ വിശദീകരിച്ചിരിക്കുന്നു. ജാവാസ്ക്രിപ്റ്റിലെ ഡാറ്റാതരങ്ങളും വേരിയബിളിന്റെ ഉപയോഗവും വിശദമായി ചർച്ചചെയ്യുന്നു. ഓപ്പറേറ്ററുകളുടെയും കൺട്രോൾ സ്ട്രക്ചറുകളുടെയും ഉപയോഗം C++ലേതിന് സമാനമാണ്. വിവിധ അന്തർ നിർമ്മിത ഫങ്ഷനുകളും ജാവാസ്ക്രിപ്റ്റിലെ ഇവന്റുകളും ഉദാഹരണസഹിതം വിശദീകരിച്ചിരിക്കുന്നു. HTML പേജിൽ ജാവാസ്ക്രിപ്റ്റ് കോഡ് ഉൾപ്പെടുത്തുന്ന വ്യത്യസ്ത രീതികൾ വിശദമായി പ്രതിപാദിച്ചിരിക്കുന്നു.

നമുക്കു പരിശീലിക്കാം

1. താഴെ കാണുന്ന സ്ക്രീൻ പ്രദർശിപ്പിക്കുന്നതിന് ഒരു വെബ് പേജ് നിർമിക്കുക. ഉപയോക്താവിന് ആദ്യത്തെ ടെക്സ്റ്റ് ബോക്സിൽ ഒരു നമ്പർ നൽകാം. Show ബട്ടൺ ക്ലിക്കുചെയ്യുമ്പോൾ രണ്ടാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ 1 മുതൽ നൽകിയിട്ടുള്ള പരിധി വരെയുള്ള എല്ലാ സംഖ്യകളുടെയും ഗുണനഫലം പ്രദർശിപ്പിക്കണം.



2. താഴെ കാണുന്ന സ്ക്രീൻ പ്രദർശിപ്പിക്കുന്നതിന് ഒരു വെബ് പേജ് നിർമിക്കുക. ഉപയോക്താവിന് ആദ്യത്തെ ടെക്സ്റ്റ് ബോക്സിൽ ഒരു നമ്പർ നൽകാം. Show ബട്ടണിൽ ക്ലിക്കുചെയ്യുമ്പോൾ രണ്ടാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ ഒറ്റ സംഖ്യയോ ഇരട്ട സംഖ്യയോ എന്നതിന് അനുസൃതമായി Odd അല്ലെങ്കിൽ Even എന്ന് പ്രദർശിപ്പിക്കണം.

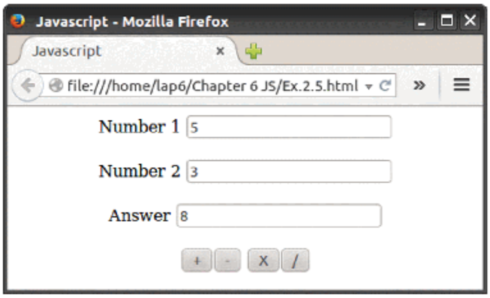


3. താഴെക്കൊടുത്തിരിക്കുന്ന സ്ക്രീൻ പ്രദർശിപ്പിക്കുന്നതിന് ഒരു വെബ് പേജ് തയാറാക്കുക. ഉപയോക്താവിന് ടെക്സ്റ്റ് ബോക്സിൽ ഒരു പ്രായം നൽകാം. ടെക്സ്റ്റ് ബോക്സിൽ ഒരു സംഖ്യക്ക് പകരം ഉപയോക്താവ് അക്ഷരമാണ് നൽകുന്നതെങ്കിൽ, Show ബട്ടണിൽ ക്ലിക്കുചെയ്യുമ്പോൾ 'Invalid Age" എന്ന ഒരു

സന്ദേശം പ്രദർശിപ്പിക്കണം അല്ലെങ്കിൽ 'Correct Data" എന്ന സന്ദേശം പ്രദർശിപ്പിക്കണം.

4. ചിത്രത്തിൽ കാണിച്ചിരിക്കുന്നതുപോലെ ഒരു ലോഗിൻ പേജ് നിർമിക്കുക. പാസ് വേഡ് നൽകുന്നതിനായി പാസ് വേഡ് ടെക്സ്റ്റ് ബോക്സും യൂസർനെയിം നൽകാനായി ഒരു ടെക്സ്റ്റ് ബോക്സും ഉണ്ടായിരിക്കണം. യൂസർനെയിമിൽ കുറഞ്ഞത് 4 ഉം പാസ് വേഡിൽ കുറഞ്ഞത് 6 ഉം ക്യാരക്ടറുകൾ ഉണ്ടായിരിക്കണം. പാസ്‌വേഡിലെ ആദ്യത്തെ രണ്ട് ക്യാരക്ടറുകൾ അക്കങ്ങൾ ആയിരിക്കണം. show ബട്ടണിൽ ക്ലിക്കു ചെയ്യുമ്പോൾ സാധുവായ ഡാറ്റയാണ് ബോക്സുകളിൽ നൽകിയിട്ടുള്ളതെങ്കിൽ 'Correct Data" എന്നും അല്ലെങ്കിൽ 'Wrong Data" എന്നുമുള്ള സന്ദേശം പ്രദർശിപ്പിക്കണം.

5. താഴെ കാണുന്ന ചിത്രത്തിലെ പോലെ ലളിതമായ ഒരു കാൽക്കുലേറ്റർ അടങ്ങിയ ഒരു വെബ് പേജ് നിർമിക്കുക. രണ്ട് സംഖ്യകൾ നൽകാൻ രണ്ടു ടെക്സ്റ്റ് ബോക്സുകൾ ഉണ്ടായിരിക്കണം. സങ്കലനം, വ്യവകലനം, ഗുണനം, ഹരണം എന്നിവയ്ക്കായി 4 ബട്ടണുകൾ ഉണ്ടായിരിക്കണം. ബട്ടനിൽ ക്ലിക്കുചെയ്യുമ്പോൾ മൂന്നാമത്തെ ടെക്സ്റ്റ് ബോക്സിൽ ഉത്തരം കാണിക്കണം.



നമുക്കു വിലയിരുത്താം

1. താഴെപ്പറയുന്നവയിൽ വേരിയബിൾ z എന്ന വേരിയബിളിന്റെ വില എഴുതുക

a.
```
var x, y, z;
x = 5;
y = 3;
z = ++x - y--;
```

b.
```
var x, y, z;
x = "12";
y = 13;
z = x + y;
```

c.
```
var x, y, z;
x = 20;
y = 8;
x %= y;
z = x++;
```

d.
```
var x, y, z;
x = 1;
y = 4;
z = !(x < y);
```

e.
```
var x, y, z;
x = 5;
y = 6 ;
z = (x > y) || (y % 2 == 0);
```

2. താഴെകൊടുത്തിരിക്കുന്നവയുടെ ഔട്ട്പുട്ട് കണ്ടെത്തുക.

a.
```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var i;
for (i = 10; i >= 1; i--)
      document.write(i + "<BR>");
</SCRIPT>
</BODY>
</HTML>
```

b.
```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var i, s = 0;
for (i = 1; i <= 100; i += 2)
      s += i;
document.write("Sum = " + s);
</SCRIPT>
</BODY>
</HTML>
```

c.
```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var n, s = 0;
n = 0;
```

```
while (n <= 50)
{
      s = s + n;
      n = n + 5;
}
document.write("Sum = " + s);
</SCRIPT>
</BODY>
</HTML>
```

d.

```
<HTML>
<BODY>
<SCRIPT Language= "JavaScript">
var n, f = 1;
n = 5;
while ( n > 0)
{
      f = f * n;
      n--;
}
document.write("Product  = " + f);
</SCRIPT>
  </BODY>
  </HTML>
```

3. താഴെ കൊടുത്തിരിക്കുന്ന html കോഡ് ശ്രദ്ധിക്കുക

```
<FORM Name= "frmStud">
<INPUT Type= "text" Name= "studentName">
</FORM>
```

ടെക്സ്റ്റ് ബോക്സിൻറെ വില n എന്ന വേരിയബിളിൽ സംഭരിക്കുന്നതിനായി വിട്ടഭാഗം പൂരിപ്പിക്കുക.

```
var n;
n = ............................;
```

4. നിങ്ങൾ checkData () എന്ന് പേരിൽ ഒരു ജാവാസ്ക്രിപ്റ്റ് ഫങ്ഷൻ എഴുതിയിട്ടുണ്ടെന്ന് കരുതുക. മൗസ് പോയിന്റർ ബട്ടനുമുകളിലൂടെ നീക്കുമ്പോൾ മാത്രമേ ഫങ്ഷൻ പ്രവർത്തിക്കേണ്ടതുള്ളൂ. അതിനുവേണ്ടി താഴെക്കൊടുത്തിരിക്കുന്ന കോഡ് എങ്ങനെ പൂർത്തീകരിക്കാം.

```
<INPUT Type= "button" .......... = "checkData()">
```

5. <SCRIPT> ടാഗും അതിന്റെ ആട്രിബ്യൂട്ടുകളും വിശദമാക്കുക.
6. ജാവാസ്ക്രിപ്റ്റിൽ ഒരു അന്തർനിർമിത ഫങ്ഷൻ നിർവചിക്കുന്നതിനുള്ള വാക്യ ഘടന എഴുതുക.
7. താഴെക്കൊടുത്തിരിക്കുന്ന വിലകൾ ജാവാസ്ക്രിപ്റ്റിലെ അനുയോജ്യമായ ഡാറ്റാ ഇനങ്ങൾ ഉപയോഗിച്ച് വേർതിരിക്കുക.
   "Welcome", "123", "true", 67.4, .98, false, "hello"
8. ജാവാസ്ക്രിപ്റ്റിൽ undefined ഡാറ്റ ഇനം എന്നതുകൊണ്ട് എന്താണ് അർഥമാക്കുന്നത്?
9. ജാവാസ്ക്രിപ്റ്റിലെ ഓപ്പറേറ്ററുകൾ വിശദീകരിക്കുക.
10. താഴെപ്പറയുന്നവ ചെയ്യുന്നതിനുള്ള ജാവാസ്ക്രിപ്റ്റ് ഫങ്ഷനുകളുടെ പേര് എഴുതുക
    a. N എന്ന വേരിയബിളിൽ സംഖ്യ ആണോ എന്ന് പരിശോധിക്കാൻ.
    b. "scert" എന്ന സ്ട്രിംഗ് വലിയ അക്ഷരങ്ങളായി മാറ്റാൻ.
    c. "HTML" എന്ന സ്ട്രിംഗ് ചെറിയ അക്ഷരങ്ങളായി മാറ്റാൻ.
    d. "Welcome to functions" എന്ന സന്ദേശം പ്രദർശിപ്പിക്കാൻ.
    e. "Computer" എന്ന സ്ട്രിംഗിലെ മൂന്നാമത്തെ ക്യാരക്ടർ പ്രദർശിപ്പിക്കാൻ.
11. 'Computer" എന്ന സ്ട്രിങ്ങിന്റെ നീളം പ്രദർശിപ്പിക്കുന്നതിനുള്ള ജാവാസ്ക്രിപ്റ്റ് കോഡ് എഴുതുക.
12. ഒരു വെബ് പേജിൽ ഒരു ബട്ടൺ ഉണ്ട്. താഴെപ്പറയുന്ന ഇവന്റുകൾക്കനുസരിച്ച് Message( ) എന്ന ഫങ്ഷൻ പ്രവർത്തിപ്പിക്കാനുള്ള HTML കോഡ് എഴുതുക.
    a. ഉപയോക്താവ് ബട്ടണിൽ മൗസ് ക്ലിക്ക് ചെയ്യുമ്പോൾ.
    b. ഉപയോക്താവ് ബട്ടണിനുമേൽ മൗസ് പോയിന്റർ നീക്കുമ്പോൾ.
13. ഒരു HTML പേജിൽ ഹെഡ് ഭാഗത്ത് ജാവാസ്ക്രിപ്റ്റ് കോഡ് എഴുതുന്നതിന്റെ ഗുണങ്ങൾ ഏവ?
14. ഒരു വിഷയത്തിന്റെ മാർക്ക് നൽകുവാൻ ഒരു ടെക്സ്റ്റ് ബോക്സ് അടങ്ങിയിരിക്കുന്ന ഒരു വെബ് പേജ് രൂപകൽപന ചെയ്യുക.
    a. ഈ വെബ് പേജിനുവേണ്ടിയുള്ള HTML കോഡ് എഴുതുക.
    b. ഈ ടെക്സ്റ്റ് ബോക്സിലെ വിലയുടെ സാധൂകരണത്തിന് ഒരു ബാഹ്യ ജാവാസ്ക്രിപ്റ്റ് ഫയൽ നിർമിച്ച് അത് HTML ഡോക്യുമെന്റുമായി ബന്ധിപ്പിക്കുക. വിലയിരുത്തുന്നതിനുള്ള സൂചകങ്ങൾ
       (i) അത് ശൂന്യമായിരിക്കരുത്
       (ii) അത് ഒരു സംഖ്യയായിരിക്കണം
       (iii) 0-നും 60-നും ഇടയിലായിരിക്കണം മാർക്ക്.
    c. സ്ക്രിപ്റ്റ് ഒരു ബാഹ്യ ഫയൽ ആയി എഴുതുന്നതിന്റെ ഗുണഫലങ്ങൾ എഴുതുക.

S4G1S1

7 വെബ് ഹോസ്റ്റിങ്

പ്രധാന പഠനനേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പഠനം പൂർത്തിയാകുന്നതോടെ പഠിതാവ് ആർജിക്കേണ്ട പഠനനേട്ടങ്ങൾ

- വെബ്സെർവറിന്റെ ഉപയോഗം വെബ്ഹോസ്റ്റിങ് ആശയം എന്നിവ വിവരിക്കുന്നു.
- വ്യത്യസ്ത തരത്തിലുള്ള ഹോസ്റ്റിങ് രീതി മനസിലാക്കുന്നു.
- ഹോസ്റ്റിങ് സ്ഥലം വാങ്ങുന്നതിനുള്ള മാർഗ്ഗം വിശദമാക്കുന്നു.
- ഡൊമൈൻ നെയിം രജിസ്റ്റർ ചെയ്യും എഫ്.റ്റി.പി. ക്ലയന്റ് സോഫ്റ്റ്‌വെയർ ഉപയോഗിച്ച് ഹോസ്റ്റ് ചെയ്യുന്നു.
- സൗജന്യ ഹോസ്റ്റിംഗിന്റെ സവിശേഷതകൾ മനസ്സിലാക്കുന്നു.
- കണ്ടന്റ് മാനേജ്മെന്റ് സംവിധാനത്തിന്റെ ഉപയോഗം തിരിച്ചറിയുന്നു.
- റെസ്പോൺസീവ് വെബ് രൂപകൽപനയുടെ ആവശ്യകത തിരിച്ചറിയുന്നു.

വെബ് പേജുകൾ എങ്ങനെ നിർമ്മിക്കാമെന്ന് മുൻ അധ്യായങ്ങളിൽ നാം പഠിച്ചുവല്ലോ? ഒട്ടേറെ വെബ്പേജുകൾ ചേർത്തു നിർമിക്കപ്പെടുന്ന വെബ്സൈറ്റ് ഒരു സ്ഥാപനത്തിന്റെയോ ഉല്പന്നത്തിന്റേയോ സേവനത്തിന്റേയോ വിവരങ്ങൾ നൽകുന്നവയായിരിക്കും. നമ്മുടെ സ്കൂളിന്റെ ഒരു വെബ്സൈറ്റ് HTML ഉപയോഗിച്ച് നിർമിച്ചു എന്നിരിക്കട്ടെ. ഈ വെബ്സൈറ്റ് ഇന്റർനെറ്റിൽ എങ്ങനെ ലഭ്യമാക്കാം? അങ്ങനെ ലഭ്യമാക്കണമെങ്കിൽ ഈ വെബ്പേജുകളെ ഇന്റർനെറ്റുമായി ബന്ധമുള്ള ഒരു വെബ്സെർവറിൽ സൂക്ഷിക്കണം. വെബ് ഹോസ്റ്റിംഗിനെപ്പറ്റിയുള്ള അവലോകനം, വിവിധതരം വെബ്ഹോസ്റ്റിങ്ങുകൾ അതിന്റെ പ്രത്യേകതകൾ തുടങ്ങിയവ ഈ അധ്യായത്തിൽ പ്രതിപാദിക്കുന്നു. കൂടാതെ ഒരു വെബ്സൈറ്റിനെ തിരിച്ചറിയാനുള്ള ഡൊമൈൻനാമം (ഡൊമൈൻ നെയിം) എങ്ങനെ തെരഞ്ഞെടുക്കാം എന്നും അത് എങ്ങനെ രജിസ്റ്റർ ചെയ്യാം എന്നും ഇവിടെ ചർച്ചചെയ്യുന്നു. വിവിധ FTP ക്ലയന്റ് സോഫ്റ്റ്‌വെയർ ഉപയോഗിച്ച് നമ്മുടെ കമ്പ്യൂട്ടറിൽ നിന്ന് സെർവറിലേയ്ക്ക് ഫയലുകൾ (വെബ്പേജുകൾ, ചിത്രങ്ങൾ മുതലായവ) എങ്ങനെ മാറ്റാമെന്നും ഇതിൽ വിശദമാക്കുന്നു. ഈ അധ്യായം പഠിച്ചു കഴിയുമ്പോൾ ഒരു ഡൊമൈൻ നാമം രജിസ്റ്റർ ചെയ്യാനും വെബ്സൈറ്റ് ഹോസ്റ്റ് ചെയ്യാനും പഠിതാവിന് കഴിയും.

7.1 വെബ് ഹോസ്റ്റിംഗ് (Web hosting)

നമ്മുടെ സ്കൂളിനായി ഒട്ടേറെ പേജുകളുള്ള ഒരു വെബ്സൈറ്റ് എങ്ങനെ നിർമിക്കുമെന്ന് മുൻ അധ്യായങ്ങളിൽ നാം പഠിച്ചിട്ടുണ്ട്. ഏതെങ്കിലും ഒരു ടെക്സ്റ്റ് എഡിറ്റർ അല്ലെങ്കിൽ ഒരു വെബ്ഡിസൈൻ ടൂൾ ഉപയോഗിച്ച് നമ്മുടെ വെബ്സൈറ്റ് നിർമിക്കാം. ഹോം പേജിൽ സ്കൂളിലെ സൗകര്യങ്ങൾ, കോഴ്സുകൾ മേൽവിലാസം മറ്റ് വെബ് പേജുകളിലേയ്ക്കുള്ള ലിങ്കുകൾ എന്നിവ ഉൾപ്പെടുത്തി ആകർഷണീയമാക്കാം.

നമ്മുടെ കമ്പ്യൂട്ടറിൽ സ്കൂളിന്റെ വെബ്സൈറ്റ് നിർമാണം പൂർത്തിയാക്കിയ ശേഷം അത് ഇന്റർനെറ്റിൽ ലഭ്യമാക്കണം. അതിനായി നമ്മുടെ കൈവശമുള്ള വെബ്സൈറ്റ് ഏതെങ്കിലും വെബ്സെർവറിലേയ്ക്ക് അപ്ലോഡ് ചെയ്തെങ്കിൽ മാത്രമേ ലോകത്തുള്ള എല്ലാവർക്കും അത് വീക്ഷിക്കാനാവൂ. വെബ്സൈറ്റ് സൂക്ഷിക്കുവാനായി വെബ്സെർവറിലെ കുറച്ച് സ്ഥലം വാടകയ്ക്ക് എടുക്കുകയോ അല്ലെങ്കിൽ നമ്മുടേതായ ഒരു വെബ് സെർവർ ക്രമീകരിക്കുകയോ ചെയ്യാം. ഒരു വെബ്സെർവർ ക്രമീകരിക്കുക എന്നത് വളരെ ചെലവേറിയ ഒരു പ്രക്രിയയാണ്. ആയതിനാൽ നിലവിലുള്ള ഒരു വെബ്സെർവറിന്റെ സ്ഥലം വാടകയ്ക്കെടുക്കുന്നതായിരിക്കും ഉചിതം.

വെബ്ഹോസ്റ്റിംഗ് എന്നാൽ ഇന്റർനെറ്റിൽ ലഭ്യമാക്കേണ്ട വെബ്സൈറ്റിലെ ഫയലുകളെ വെബ്സെർവറിൽ സൂക്ഷിക്കുകയും അതിനാവശ്യമായ സേവനം നൽകുകയും ചെയ്യുന്ന പ്രക്രിയയാണ്. വെബ്ഹോസ്റ്റിങ് സേവനങ്ങൾ നൽകുന്ന കമ്പനികളെ വെബ്ഹോസ്റ്റ് എന്നുവിളിക്കും. വെബ് സെർവറുകൾ വെബ്ഹോസ്റ്റുകളുടെ ഉടമസ്ഥതയിലും നിയന്ത്രണത്തിലും ഉള്ളതായിരിക്കും. തടസമില്ലാതെ ഇന്റർനെറ്റ് ബന്ധം നൽകുവാനും PHP, JAVA, ASP.NET തുടങ്ങിയ പ്രോഗ്രാമിങ് ഭാഷകൾക്ക് ഡാറ്റ ബേസ് പിൻബലം ആവശ്യമുള്ളപ്പോൾ നൽകാനും ഇത്തരം സെർവറുകൾക്ക് കഴിയും.

7.1.1 വിവിധ തരം വെബ് ഹോസ്റ്റിംഗുകൾ (Types of web hosting)

HTML, ചിത്രങ്ങൾ എന്നിവ അടങ്ങിയ നമ്മുടെ സ്കൂൾ വെബ്സൈറ്റിന് 4 എം.ബി. സ്ഥലമാണ് വെബ്സെർവറിൽ വേണ്ടത് എന്ന് കരുതുക. എന്നാൽ വെബ്ഹോസ്റ്റുകൾ 10 എം.ബി., 20 എം.ബി തുടങ്ങിയ പാക്കേജുകളായാണ് ഇന്ന് സാധാരണയായി ലഭ്യമാക്കുന്നത്. അപ്പോൾ നമ്മുടെ ആവശ്യത്തിനനുസരിച്ച് ഏറ്റവും അനുയോജ്യമായ ഒരു പാക്കേജ് തിരഞ്ഞെടുക്കേണ്ടതുണ്ട്. നമ്മുടെ വെബ്സൈറ്റിൽ എത്ര ആളുകൾ സന്ദർശിക്കുമെന്നത് ഈ തിരഞ്ഞെടുപ്പിന് ഒരു മാനദണ്ഡമാണ്. നമ്മുടെ വെബ്സൈറ്റിൽ ഡാറ്റബേസും സ്ക്രിപ്റ്റുകളും ഉണ്ടെങ്കിൽ അതിനാവശ്യമായ സേവനം ലഭ്യമാകുന്ന വെബ്ഹോസ്റ്റുകളെയാണ് തിരഞ്ഞെടുക്കേണ്ടത്. വെബ്ഹോസ്റ്റുകൾ വിവിധതരത്തിലുള്ള ഹോസ്റ്റിങ് പാക്കേജുകൾ ലഭ്യമാക്കുന്നു. ഷെയേർഡ് ഹോസ്റ്റിങ്, വെർച്ച്വൽ പ്രൈവറ്റ് ഹോസ്റ്റിങ്, ഡെഡിക്കേറ്റഡ് ഹോസ്റ്റിങ് എന്നിങ്ങനെയാണ് അവയെ തരം തിരിച്ചിരിക്കുന്നത്.

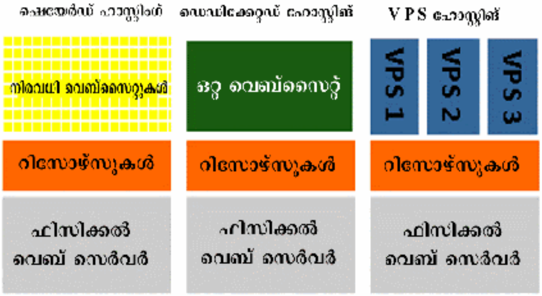
a. ഷെയേർഡ് ഹോസ്റ്റിങ് (Shared hosting): വെബ് ഹോസ്റ്റിങിലെ സർവസാധാരണമായ ഹോസ്റ്റിങ് രീതിയാണ് ഷെയേർഡ് വെബ്ഹോസ്റ്റിങ്. പല വെബ്സൈറ്റുകൾ ഒരു വെബ്സെർവറിൽ സൂക്ഷിക്കുകയും സെർവറിന്റെ RAM, CPU എന്നിവ പങ്കിടുകയും ചെയ്യുന്നതുകൊണ്ടാണ് ഇതിനെ ഷെയേർഡ് ഹോസ്റ്റിങ് എന്നു വിളിക്കുന്നത്. ഇത് ഉയർന്ന ബാൻഡ് വിഡ്ത്തും ഉയർന്ന സംഭരണ സ്ഥലവും ആവശ്യമുള്ള വെബ്സൈ

റ്റുകൾക്ക് അനുയോജ്യമല്ല. സന്ദർശകർ കുറവുള്ള വെബ്സൈറ്റുകൾക്കാണ് ഷെയേർഡ് ഹോസ്റ്റിംഗ് കൂടുതൽ ഇണങ്ങുന്നത്. ഇത് വളരെ ചെലവ് കുറഞ്ഞതും എളുപ്പത്തിൽ ഉപയോഗിക്കാവുന്നതുമാണ്. അതുകൊണ്ട് തന്നെ ഇതുവളരെ വ്യാപകമായി ഉപയോഗിക്കുന്നു. സെർവർ നവീകരണവും സോഫ്റ്റ്വെയർ ഇൻസ്റ്റാൾ ചെയ്യുമ്പോൾ ഉണ്ടാകുന്ന സുരക്ഷാപ്രശ്നങ്ങളും ഹോസ്റ്റിങ് കമ്പനി തന്നെ പരിഹരിക്കും. നിരവധി വെബ്സൈറ്റുകൾ ഇതിന്റെ ബാൻഡ് വിഡ്ത്ത് പങ്കിടുന്നതുകൊണ്ട് ഏതെങ്കിലും ഒരു വെബ്സൈറ്റിന് വലിയ അളവിലുള്ള ട്രാഫിക്ക് ഉണ്ടാവുകയാണെങ്കിൽ ഈ വെബ്സെർവറിലെ എല്ലാ വെബ്സൈറ്റുകളുടെയും പ്രവർത്തനം മന്ദഗതിയിലാകും എന്നതാണ് ഇതിന്റെ പ്രധാനപ്പെട്ട ന്യൂനത.

b. ഡെഡിക്കേറ്റഡ് ഹോസ്റ്റിങ് (Dedicated hosting): ഉപഭോക്താവ് ഒരു വെബ്സെർവറും അനുബന്ധ സൗകര്യങ്ങളും പൂർണ്ണമായും വാടകയ്ക്ക് എടുക്കുന്നതിനെ ഡെഡിക്കേറ്റഡ് ഹോസ്റ്റിങ് എന്ന് വിളിക്കുന്നു. ഈ വെബ് സെർവർ മറ്റ് വെബ്സൈറ്റുകളുമായി വിഭവങ്ങൾ പങ്കിടുന്നില്ല. വലിയ സ്ഥാപനങ്ങളും സർക്കാർ വകുപ്പുകളും ധാരാളം സന്ദർശകർ ഉള്ളതുകൊണ്ട് ഡെഡിക്കേറ്റഡ് വെബ്ഹോസ്റ്റിങ്ങാണ് തിരഞ്ഞെടുക്കുന്നത്. ഇവിടെ ഹാർഡ്‌വെയറും സോഫ്റ്റ്വെയറും തിരഞ്ഞെടുക്കുന്നതിനുള്ള സ്വാതന്ത്ര്യവും വെബ്സെർവറിന്റെ പൂർണ്ണ നിയന്ത്രണവും ഉപയോക്താവിനാണ്. ഡെഡിക്കേറ്റഡ് സെർവറിന്റെ കാര്യക്ഷമത എപ്പോഴും ഉയർന്ന നിലവാരത്തിലുള്ളതും, എന്നാൽ വളരെ ചെലവേറിയതുമാണ്. ഡെഡിക്കേറ്റഡ് സെർവറുകൾ സാധാരണ ഗതിയിൽ ഡാറ്റ സെന്ററുകളിൽ സ്ഥാപിച്ചിരിക്കുന്നതിനാൽ ഇന്റർനെറ്റ് കണക്ഷനും വൈദ്യുതിയും വിദഗ്ധ സേവനവും എപ്പോഴും ലഭ്യമാകും. ഇതുമൂലം ഇന്റർനെറ്റ് കണക്ഷനും വൈദ്യുതി സംബന്ധമായുള്ള സൗകര്യങ്ങളും ക്രമീകരിക്കുന്നതിനേക്കാൾ കുറവായിരിക്കും. ബാന്റ് വിഡ്ത്ത് പങ്കുവയ്ക്കപ്പെടാത്തതിനാൽ വളരെ വേഗത്തിൽ വെബ്സൈറ്റുകൾ ഉപയോഗിക്കുവാൻ ഉപയോക്താവിന് സാധ്യമാകും. സേവനദാതാവിന്റെ ഭൗതിക സൗകര്യത്തിനുള്ളിൽ ഉപയോക്താവ് സ്വന്തമായി വാങ്ങിയ വെബ് സെർവർ പ്രവർത്തിക്കുവാൻ ക്രമീകരണം ഒരുക്കുകയാണെങ്കിൽ അതിനെ കോ ലോക്കേഷൻ എന്നുവിളിക്കും.

c. വെർച്വൽ പ്രൈവറ്റ് സെർവർ (Virtual Private Server): വെർച്വൽ പ്രൈവറ്റ് സെർവർ എന്നാൽ വെർച്വൽ ടെക്നോളജി ഉപയോഗിച്ച് പലതായി വിഭജിക്കപ്പെട്ട ഒരു വെബ്സെർവറാണ്. ഓരോ VPS ഉം ഡെഡിക്കേറ്റഡ് സെർവർ പോലെ പ്രവർത്തിക്കും. അതിനായി പ്രത്യേക ഓപ്പറേറ്റിങ് സിസ്റ്റം വെബ്സെർവർ സോഫ്റ്റ് വെയർ പാക്കേജുകളായ ഇ-മെയിൽ, ഡാറ്റാബേസ് തുടങ്ങിയവയും അതിൽ സ്ഥാപിച്ചിരിക്കും. ഷെയേർഡ് ഹോസ്റ്റിങ് പോലെ ഇവയെ കരുതേണ്ടതില്ല. കാരണം VPS-ൽ ഒരു നിശ്ചിതശതമാനം RAM ഓരോ വെർച്വൽ വെബ്സെർവറിനുമായി മാറ്റിവച്ചിരിക്കുന്നു. ഓരോ VPS-ഉം പൂർണ്ണ സ്വാതന്ത്ര്യമുള്ള വെബ്സെർവറായി പ്രവർത്തിക്കും. അതായത് ഒരു പ്രത്യേക ഭൗതിക സെർവറിൽ പ്രവർത്തിക്കുന്നതുപോലെ VPS-ന്റെ ഉപഭോക്താവിന് സോഫ്റ്റ് വെയർ ഇൻസ്റ്റാൾ ചെയ്യാനും അവ ക്രമീകരിക്കാനും പൂർണ്ണ അവകാശം ഉണ്ടായിരിക്കും. മറ്റ് വെർച്വൽ സെർവറുകളെ ബാധിക്കാതെ പുനരാരംഭിക്കാൻ ഓരോ VPS-നും അവകാശമുണ്ട്. VPS-ഹോസ്റ്റിങ് സെർവറിലെ ഓരോ വെബ്സൈറ്റിനും തനതായ ബാന്റ് വിഡ്ത്ത് ലഭ്യമാക്കും. ഇതുമൂലം സെർവർ ഷെയർ ചെയ്യപ്പെട്ടാലും ഡെഡിക്കേറ്റഡ്

ഹോസ്റ്റിങ്ങിന്റെ നേട്ടം കൈവരിക്കാനാകും. ഷെയേർഡ് ഹോസ്റ്റിങ്ങിനെക്കാളും സവിശേഷതകൾ ആവശ്യമുള്ള വെബ്സൈറ്റുകൾക്കാണ് ഇത്തരം ഹോസ്റ്റിങ് യോജിക്കുന്നത്. പക്ഷേ ഡെഡിക്കേറ്റഡ് ഹോസ്റ്റിങ്ങിന്റെ എല്ലാ പ്രത്യേകതകളും ഇവിടെ ലഭ്യമല്ല. ഡെഡിക്കേറ്റഡ് ഹോസ്റ്റിങ്ങിന്റെ ഒട്ടുമിക്ക സർവ്വീസുകളും മിതമായ നിരക്കിൽ നൽകുവാൻ VPS-ഹോസ്റ്റിങ്ങിന് കഴിയും. അറിയപ്പെടുന്ന ചില സെർവർ വെർച്ചലൈസേഷൻ സോഫ്റ്റ് വെയറുകളാണ് VM വെയർ, വെർച്ചൽബോക്സ്, ഫ്രീ VPS, യൂസർ മോഡ് ലിനക്സ്, മൈക്രോസോഫ്റ്റ് ഹൈപ്പർ വി തുടങ്ങിയവ.

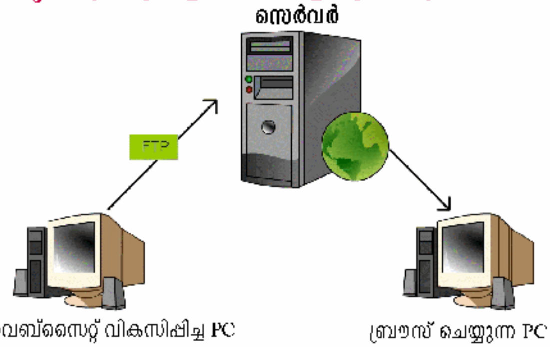


ചിത്രം 7.1 : വെബ് ഹോസ്റ്റിംഗ് തരങ്ങൾ

വിവിധതരത്തിലുള്ള ഹോസ്റ്റിംഗ് പാക്കേജുകളെപ്പറ്റിയുള്ള പ്രതീകാത്മക വിവരണം ചിത്രം 7.1 ൽ നൽകിയിരിക്കുന്നു.

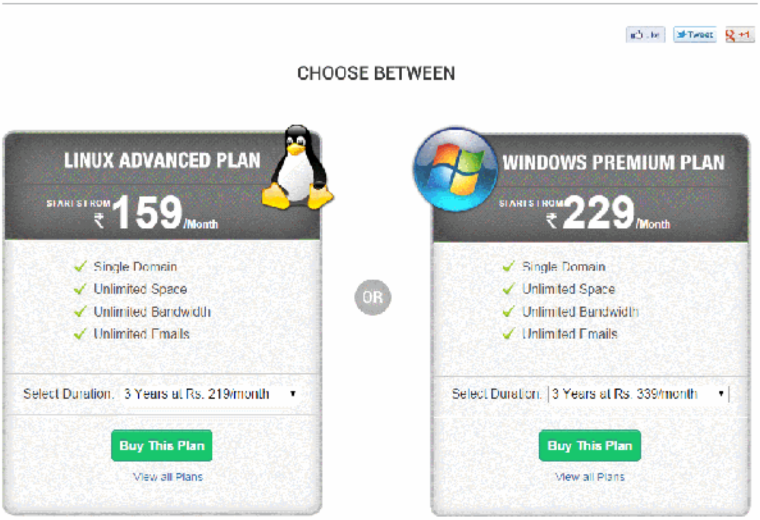
7.1.2 ഹോസ്റ്റിങ്ങിന് സ്ഥലം വാങ്ങുക (Buying hosting space)

സ്കൂളിന് വേണ്ടി നിർമ്മിച്ച വെബ്സൈറ്റ് ഇപ്പോൾ നമ്മുടെ കമ്പ്യൂട്ടറിലെ ഒരു ഫോൾഡറിൽ സൂക്ഷിച്ചിരിക്കുകയാണ്. ഇത് ഇന്റർനെറ്റിൽ ലഭ്യമാകാൻ ഈ ഫയലുകളെ വെബ്സെർവറിലേയ്ക്ക് പകർത്തണം. ചിത്രം 7.2 ൽ ഇത് കാണിച്ചിരിക്കുന്നു.



ചിത്രം 7.2 : ഇന്റർനെറ്റിൽ ഒരു വെബ്സൈറ്റ് ലഭ്യമാക്കുന്നു

ഇതിനായി അനുയോജ്യമായ ഒരു വെബ്ഹോസ്റ്റിങ് രീതി നാം തിരഞ്ഞെടുക്കേണ്ടതുണ്ട്. ഷെയേർഡ് ഹോസ്റ്റിങ് നമ്മുടെ വെബ്സൈറ്റിന് അനുയോജ്യമായതിനാലും, ചെലവ് കുറവായതിനാലും അത് തിരഞ്ഞെടുക്കുന്നതാവും കൂടുതൽ ഉത്തമം. ഏതു തരം ഹോസ്റ്റിങ് സെർവറാണെന്ന് തരം തീരുമാനിച്ചു കഴിഞ്ഞാൽ നമ്മുടെ വെബ്സൈറ്റിലെ ഫയലുകൾ സൂക്ഷിക്കാനാവശ്യമായ



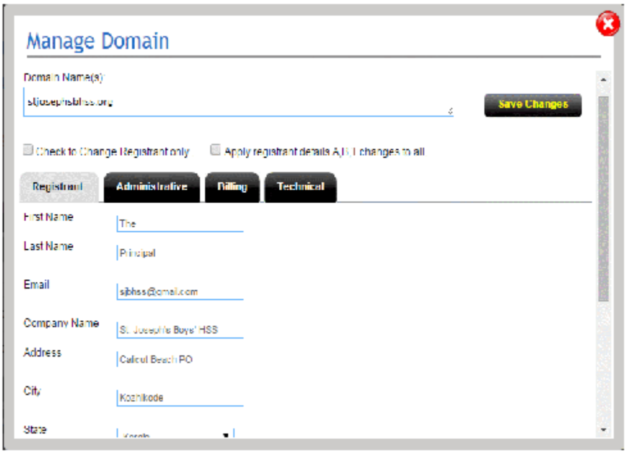
ചിത്രം 7.3 : ഹോസ്റ്റിംഗ് ഓപ്ഷനുകൾ

സെർവർ സ്ഥലം സേവന ദാതാവിൽ നിന്ന് വാങ്ങണം. ഹോസ്റ്റിങ് സ്ഥലം വാങ്ങുമ്പോൾ നിരവധി കാര്യങ്ങൾ പരിഗണിക്കേണ്ടതായുണ്ട്. ഒന്നാമതായി നമ്മുടെ വെബ്സൈറ്റിന് എത്രസ്ഥലം ആവശ്യമുണ്ടെന്ന് കണ്ടെത്തണം. വെബ്പേജിൽ പ്രോഗ്രാമിങ് ഉള്ളടക്കം ഉണ്ടെങ്കിൽ വെബ്സെർവറിൽ ഈ സേവനം നൽകുന്ന സാങ്കേതിക വിദ്യ ഉണ്ടായിരിക്കണം. വെബ്പേജ് നിർമ്മാണത്തിനായി ഉപയോഗിച്ചിരിക്കുന്ന പ്രോഗ്രാം, വിന്റോസ് ഹോസ്റ്റിങ്ങിനാണോ ലിനക്സ് ഹോസ്റ്റിങ്ങിനാണോ അനുയോജ്യമെന്ന് കണ്ടെത്തണം. അതിനുശേഷം വിൻഡോസ് സെർവറോ ലിനക്സ് സെർവറോ തിരഞ്ഞെടുക്കണം. ചിത്രം 7.3 ഇത് വ്യക്തമാക്കുന്നു. വെബ്സൈറ്റിൽ HTML കോഡ് മാത്രമാണ് ഉപയോഗിച്ചിരിക്കുന്നതെങ്കിൽ ഏത് തരം സെർവറും ഉപയോഗിക്കാം. എന്നാൽ ഡാറ്റാ ബേസ്, ഇ-മെയിൽ എന്നിവയ്ക്ക് ആവശ്യമായ സാങ്കേതിക സൗകര്യം ആവശ്യമുണ്ടെങ്കിൽ അത് കൂടി പരിഗണിച്ച് വേണം വെബ്ഹോസ്റ്റ് തിരഞ്ഞെടുക്കാൻ.

7.1.3 ഡൊമൈൻ നെയിം രജിസ്റ്റർ ചെയ്യുക (Domain name registration)

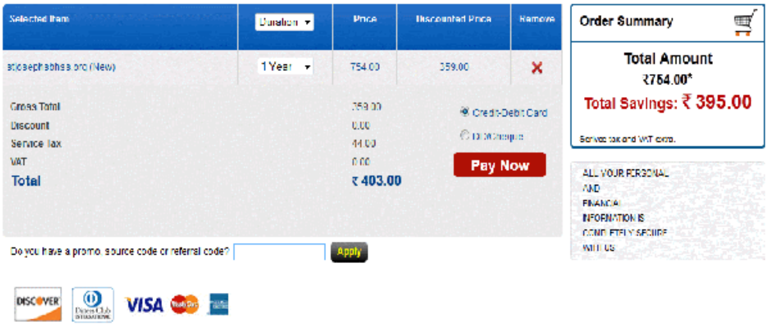
സ്കൂൾ വെബ്സൈറ്റിന് ആവശ്യമായ ഹോസ്റ്റിങ് സ്ഥലം നാം വാങ്ങിക്കഴിഞ്ഞു. ഇന്റർനെറ്റിൽ ഓരോ വെബ്സൈറ്റും തിരിച്ചറിയുന്നതിനായി പ്രത്യേകം URL ആവശ്യമാണ്. ഇതിനായി അനുയോജ്യമായ ഒരു ഡൊമൈൻ നാമം സ്കൂൾ വെബ്സൈറ്റിനായി നാം തിരഞ്ഞെടുക്കണം. ഇന്റർനെറ്റിൽ ഒരു വെബ്സൈറ്റ് തിരിച്ചറിയാൻ ഡൊമൈൻ സഹായിക്കുന്നതാണ്. മിക്ക വെബ്ഹോസ്റ്റിങ് കമ്പനികളും ഡൊമൈൻ നാമം രജിസ്റ്റർ ചെയ്യുവാനുള്ള സേവനം നൽകുന്നുണ്ട്. നമ്മുടെ വെബ്സൈറ്റിന് അനുയോജ്യമായ ഒരു ഡൊമൈൻ നാമം തിരഞ്ഞെടുത്തശേഷം ഇവ ലഭ്യമാണോ എന്നും മറ്റാരെങ്കിലും ഇതേ പേരിൽ രജിസ്റ്റർ ചെയ്തിട്ടുണ്ടോ എന്നും പരിശോധിക്കണം. www.whois.net പോലെയുള്ള സൈറ്റുകളും വെബ്ഹോസ്റ്റിങ് കമ്പനികളുടെ വെബ്സൈറ്റുകളും ഇത്തരം പരിശോധനയ്ക്ക് നമ്മെ സഹായിക്കുന്നു. രജിസ്റ്റർ ചെയ്യപ്പെട്ട എല്ലാ ഡൊമൈൻ നാമങ്ങളുടെയും ഡാറ്റാബേസായ ICANN ഡാറ്റാബേസ് പരിശോധിച്ച് മറുപടി നൽകുന്നു. ചിത്രം 7.4 ഇത് സൂചിപ്പിക്കുന്നു. ഈ ഡൊമൈൻ നാമം ലഭ്യമാണെങ്കിൽ ICANN ന്റെ whois.net ഡാറ്റാബേസിൽ പേര്,

ചിത്രം 7.4 : ഡൊമൈൻ നാമ രജിസ്ട്രേഷന്റെ തിരയൽ ഫലം



ചിത്രം 7.5 : WHOIS വിവരങ്ങൾ

മേൽവിലാസം, ഫോൺ നമ്പർ, ഇ-മെയിൽ എന്നിവ ചിത്രം 7.5-ൽ കാണിച്ചിട്ടുള്ളതു പോലെ രേഖപ്പെടുത്തണം. ഈ വിവരങ്ങൾ രജിസ്റ്റർ ചെയ്യുന്ന ആളിന്റെ ഇഷ്ടാനുസരണം പൊതുവായോ വ്യക്തിപരമായോ സൂക്ഷിക്കുന്നതാണ്. വാർഷിക വരിസംഖ്യ ഓൺലൈനായി അടച്ച് നമ്മുടെ പേരിൽ ഡൊമൈൻനാമം രജിസ്റ്റർ ചെയ്യാവുന്നതാണ്. വാങ്ങുന്നതിന്റെ ഷോപ്പിങ് കാർട്ട് ചിത്രം 7.6 ൽ പ്രദർശിപ്പിച്ചിരിക്കുന്നു.



ചിത്രം 7.6 : ഡൊമൈൻ വാങ്ങൽ

ഇപ്പോൾ നമ്മൾ വെബ്സെർവറിൽ സ്കൂൾ വെബ്സൈറ്റിനായുള്ള സ്ഥലം വാങ്ങി ഡൊമൈൻ നാമം രജിസ്റ്റർ ചെയ്തു. ഏതെങ്കിലും ഉപയോക്താവ് നമ്മുടെ ഡൊമൈൻ നാമം (stjosephhss.org) ബ്രൗസർ വിൻഡോയിൽ രേഖപ്പെടുത്തിയാൽ നാം വെബ്സെർവറിൽ സൂക്ഷിച്ചിരിക്കുന്ന വെബ്പേജ് ദൃശ്യമാകും. DNS നമ്മുടെ സെർവറിന്റെ IP മേൽവിലാസം ബ്രൗസറിൽ ലഭ്യമാക്കുകയും ആ IP മേൽവിലാസം ഉപയോഗിച്ച് നമ്മുടെ വെബ്പേജ് സൂക്ഷിച്ചിരിക്കുന്ന വെബ് സെർവറുമായി ബന്ധമുണ്ടാക്കുകയും ചെയ്യുന്നതുമൂലമാണ് ഇത് സാധ്യമാകുന്നത്. വെബ് സെർവറുമായി ബന്ധിപ്പിച്ചിട്ടുള്ള ഡൊമൈൻ നാമത്തിന്റെ IP മേൽവിലാസം സൂക്ഷിച്ചുവയ്ക്കുന്നതിനാണ് A.റിക്കോർഡ് ഉപയോഗിക്കുന്നത്. ഡൊമൈനിന്റെ കൺട്രോൾ പാനലിൽ ലോഗ് ഇൻ ചെയ്തതിനുശേഷം A. റിക്കോർഡ് പുതുക്കാവുന്നതാണ്.

ചിത്രം 7.7 : ഡൊമൈനിന്റെ A റിക്കോർഡ് മാറ്റുന്ന ജാലകം

വെബ്സെർവറിന്റെ IP മേൽവിലാസം സൂചിപ്പിക്കുന്നതിനായി ചിത്രം 7.7 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ A റിക്കോർഡ് ക്രമീകരിക്കാം. ഇതിന് ശേഷം DNS, സെർവറുകൾക്ക് നാം നൽകിയ വെബ്സൈറ്റ് സൂക്ഷിച്ചിരിക്കുന്ന വെബ്സെർവറിലേയ്ക്ക് ബ്രൗസർ വഴി ബന്ധപ്പെടാനുള്ള IP മേൽവിലാസം ലഭ്യമാകും. അതു വഴി നമ്മുടെ വെബ്സെർവറിലേയ്ക്ക് വളരെ വേഗം ബന്ധപ്പെടാനാവും.

ഒരു ഡൊമൈനെകുറിച്ചുള്ള വിവരങ്ങൾ **WHO IS search** -ൽ ലഭ്യമാകും. ഡൊമൈൻ ഉടമസ്ഥത എവിടെ എപ്പോൾ രജിസ്റ്റർ ചെയ്തു, കാലാവധി തുടങ്ങിയ വിവരങ്ങൾ ഇതിൽ ഉൾപ്പെടും. ഒരു പ്രത്യേക ഡൊമൈൻ പേര് ലഭ്യമാണോ എന്ന് കണ്ടെത്തുന്നതിന് ഇത് ഉപയോഗിക്കും. **www.kerala.gov.in** എന്നത് **WHOIS** ൽ നോക്കുമ്പോൾ അത് താഴെ പറയുന്ന വിവരങ്ങൾ ലഭ്യമാക്കും.

```
Domain ID:D8944-AFIN
Domain Name:KERALA.GOV.IN
Created On:31-Dec-2003 05:00:00 UTC
Last Updated On:16-Jul-2014 11:37:59 UTC
Expiration Date:31-Dec-2016 05:00:00 UTC
Sponsoring Registrar:National Informatics Centre (R12-AFIN)
Status:OK
Registrant ID:R-R03120114034
Registrant Name:Government of Kerala
Registrant Organization:
Registrant Street1:Chief Minister's Office
Registrant Street2:
Registrant Street3:
Registrant City:Government Secretariate, Trivandrum, 69500
Registrant State/Province:Kerala
Registrant Postal Code:695001
Registrant Country:IN
```

നമുക്കു ചെയ്യാം

ജനപ്രീതിയാർജ്ജിച്ച വെബ്ഹോസ്റ്റുകളുടെയും അവയുടെ ഏറ്റവും കുറഞ്ഞ അളവിലുള്ള ഹോസ്റ്റിംഗ് സ്ഥലത്തിന് ഒരു വർഷത്തേക്ക് നൽകേണ്ട വിലകളും താരതമ്യം ചെയ്യുന്ന ടേബിൾ തയ്യാറാക്കുക. വിൻഡോസിനും, ലിനക്സ് വെബ്ഹോസ്റ്റിങ്ങിനും ഉള്ള വിലകൾ പ്രത്യേകമായി തയ്യാറാക്കുക.

പ്രശസ്തരായ വെബ്ഹോസ്റ്റ് സേവനദാതാക്കളുടെ .org, .com ഡൊമൈൻ രജിസ്ട്രേഷനുകൾക്ക് ഒരു വർഷത്തെ വിലവിവരപ്പട്ടിക താരതമ്യം ചെയ്യുന്ന ടേബിൾ തയ്യാറാക്കുക.

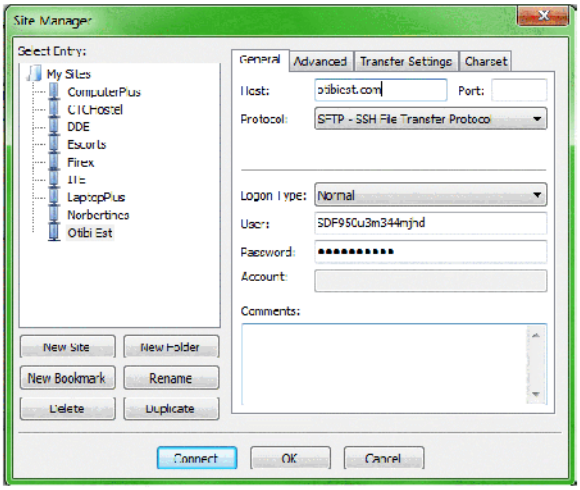
www.scert.kerala.gov.in, www.dhsekerala.gov.in എന്നീ വെബ്സൈറ്റുകളുടെ WHOIS വിവരങ്ങൾ www.whois.net ൽ തിരഞ്ഞശേഷം അവയുടെ ചാർട്ട് തയ്യാറാക്കുക.

നിങ്ങളുടെ പുരോഗതി അറിയാം

1. വെബ്ഹോസ്റ്റിങ് സേവനം നൽകുന്ന കമ്പനികളെ ____________ എന്നു പറയുന്നു.
2. വെബ്ഹോസ്റ്റിങ് തരം നിശ്ചയിക്കുന്ന ഘടകങ്ങൾ തയ്യാറാക്കുക.
3. വി.പി.എസ്. എന്നത്
   a. Virtual Premium Service b. Virtual Private Service
   c. Virtual Premium Server e. Virtual Private Server
4. കോ-ലൊക്കേഷൻ എന്നാലെന്ത്?
5. WHOIS വിവരങ്ങൾ ഉൾക്കൊള്ളുന്നതെന്തൊക്കെ?
6. ഒരു ഡൊമൈൻ നെയിമിന് A റിക്കോർഡ് പ്രാധാന്യമുള്ളതാകുന്നതെന്തുകൊണ്ട്?

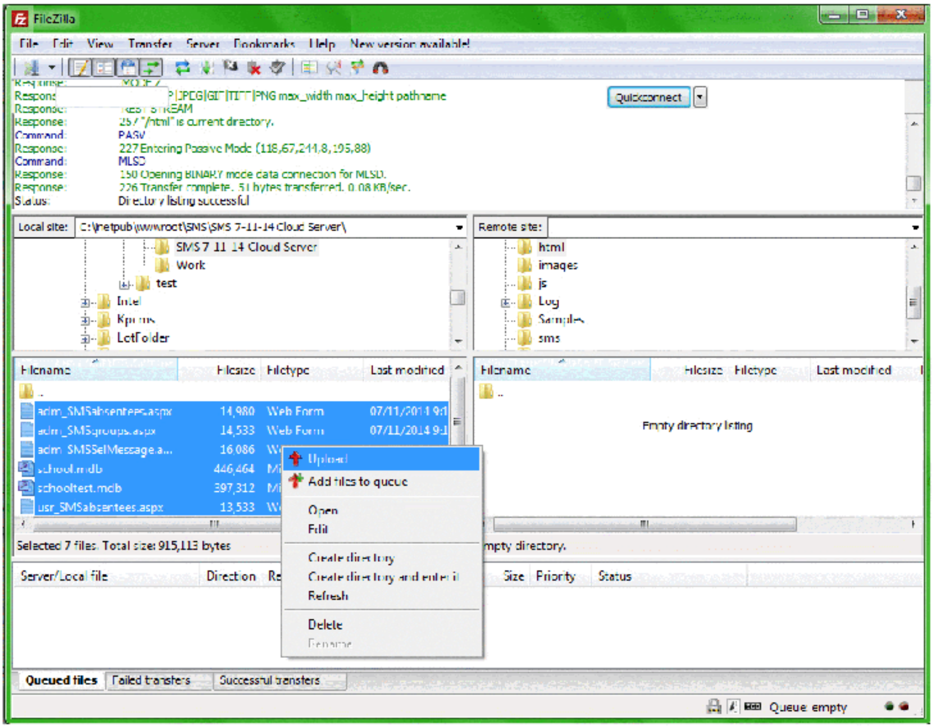
7.1.4 എഫ്.റ്റി.പി. ക്ലയന്റ് സോഫ്റ്റ്‌വെയർ (FTP client software)

ഹോസ്റ്റിങ്ങിനായി സെർവർ സ്ഥലവും ഡൊമൈൻ നാമവും വാങ്ങിയശേഷം നമ്മുടെ കമ്പ്യൂട്ടറിൽ സൂക്ഷിച്ചിരിക്കുന്ന വെബ്സൈറ്റ് ഫയലുകളെ വെബ്സെർവറിലേയ്ക്ക് പകർത്തേണ്ടതായുണ്ട്. ഇതിനായി എഫ്.റ്റി.പി. ക്ലയന്റ് സോഫ്റ്റ്‌വെയർ ഉപയോഗിക്കുന്നു.



ചിത്രം 7.8 : FTP സോഫ്റ്റ്‌വെയറിന്റെ ലോഗിൻ പേജ്

FTP യെ കുറിച്ച് കമ്പ്യൂട്ടർ ശൃംഖലകൾ എന്ന അധ്യായത്തിൽ 11-ാം ക്ലാസിൽ നാം ചർച്ച ചെയ്തു. ഇന്റർനെറ്റുമായി ബന്ധിപ്പിച്ചിരിക്കുന്ന ഒരു കമ്പ്യൂട്ടറിൽ നിന്ന് മറ്റൊരു കമ്പ്യൂട്ടറിലേയ്ക്ക് ഫയൽ നീക്കുന്നതിന് FTP ഉപയോഗിക്കുന്നു. FTP ക്ലയന്റ് സോഫ്റ്റ്‌വെയർ വിദൂര സെർവറുമായി ബന്ധപ്പെടുകയും അതിനുശേഷം കമ്പ്യൂട്ടറിലെ ഫയലുകൾ സെർവർ കമ്പ്യൂട്ടറിലേയ്ക്ക് മാറ്റുകയും ചെയ്യുന്നു. FTP സെർവറുമായി ബന്ധം ഉണ്ടാക്കുവാൻ FTP ക്ലയന്റ് സോഫ്റ്റ്‌വെയറിന് യൂസർനെയിം, പാസ്‌വേർഡ്, ഡൊമൈൻ നാമം എന്നിവ

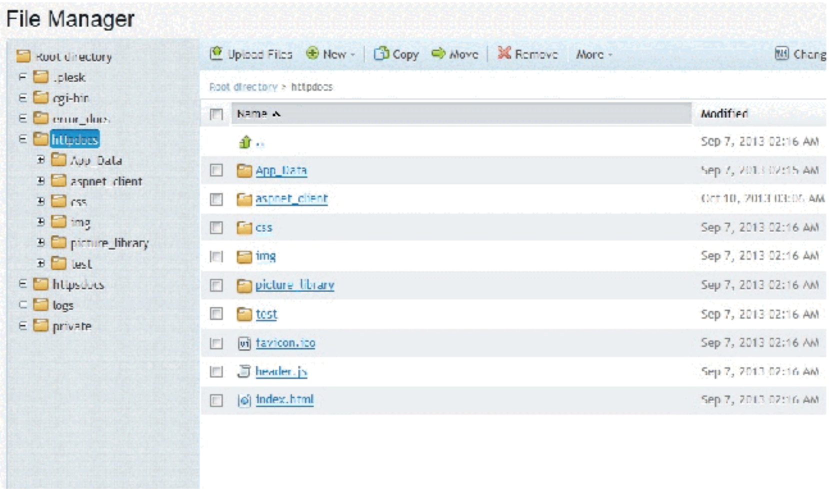


ചിത്രം 7.9 : FTP ക്ലൈന്റ് സോഫ്റ്റ്‌വെയറിന്റെ IDE

ആവശ്യമാണ്. ചിത്രം 7.8 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ സൈറ്റ് മാനേജർ ഡയലോഗ് ബോക്സിൽ ഇവ ലഭ്യമാക്കണം. സുരക്ഷിതമല്ലാത്ത ഒരു ലളിതമായ ടെക്സ്റ്റിന്റെ രൂപത്തിൽ ഈ യൂസർനാമവും പാസ്‌വേർഡും സെർവറിലേയ്ക്ക് FTP അയയ്ക്കും. എന്നാൽ ഇപ്പോൾ SSH FTP പ്രോട്ടോക്കോൾ ഈ വിവരങ്ങൾ മറ്റൊരു രൂപത്തിലാക്കി വിവർത്തനം ചെയ്ത് (ENCRYPT) വെബ്സെർവറിലേയ്ക്ക് FTP യുടെ സഹായത്തോടെ അയയ്ക്കുന്നു. സെക്യൂർഷെൽ പ്രോട്ടോക്കോൾ (SSH)ഉപയോഗിച്ച് സുരക്ഷിതമായ ഫയൽനീക്കത്തിന് SFTP (SSH FTP) സഹായിക്കുകയും ചെയ്യും.

FTP ക്ലയന്റ് സോഫ്റ്റ് വെയറിന്റെ ആധികാരികത ഉറപ്പിച്ച് കഴിഞ്ഞാൽ അതിന്റെ IDE-ചിത്രം 7.9 -ലേതുപോലെ പ്രത്യക്ഷമാകും. ഈ ചിത്രത്തിൽ ഇടതുവശത്ത് പ്രദർശിപ്പിച്ചിരിക്കുന്ന ഫോൾഡർ നമ്മുടെ കമ്പ്യൂട്ടറിലേതും വലതുവശത്തേത് വെബ്സെർവർ കമ്പ്യൂട്ടറിലേതും ആകുന്നു. മെനു ഉപയോഗിച്ചോ ഡ്രാഗ് ചെയ്തോ ഇടതുവശത്തുള്ള ഫയലിനെ വലതുവശത്തുള്ള ജാലകത്തിലേക്കാക്കി വെബ്സെർവറിലേയ്ക്ക് അപ്‌ലോഡ് ചെയ്യാം. ഇങ്ങനെ നമ്മുടെ കമ്പ്യൂട്ടറിലെ ഫയലുകൾ വെബ്സെർവറിലേക്ക് പകർത്താം. പ്രമുഖമായ ചില FTP സോഫ്റ്റ് വെയറുകളാണ് ഫയൽസില്ല, ക്യൂട്ട് FTP., സ്മാർട്ട് FTP. തുടങ്ങിയവ.

ചില വെബ്ഹോസ്റ്റിങ് കമ്പനികൾ ഫയലുകൾ അപ്‌ലോഡ് ചെയ്യുന്നതിന് അവരുടേതായ കൺട്രോൾ പാനലുകൾ സജ്ജീകരിച്ചിട്ടുണ്ട്. ഇവ പുറത്ത് \ ¶ pÅ FTP ക്ലൈന്റ് സോഫ്റ്റ് വെയർ വഴി ഫയൽ അപ്‌ലോഡ് ചെയ്യുവാൻ അനുവദിക്കുകയില്ല. ചിത്രം 7.10 ഇത്തരത്തിലുള്ള ഒരു കൺട്രോൾ പാനൽ പ്രദർശിപ്പിച്ചിരിക്കുന്നു.



ചിത്രം 7.10 : ഹോസ്റ്റിംഗ് സൗകര്യങ്ങൾ ലഭ്യമാക്കുന്ന FTP കൺട്രോൾ പാനൽ

7.2 സൗജന്യ ഹോസ്റ്റിങ് (Free hosting)

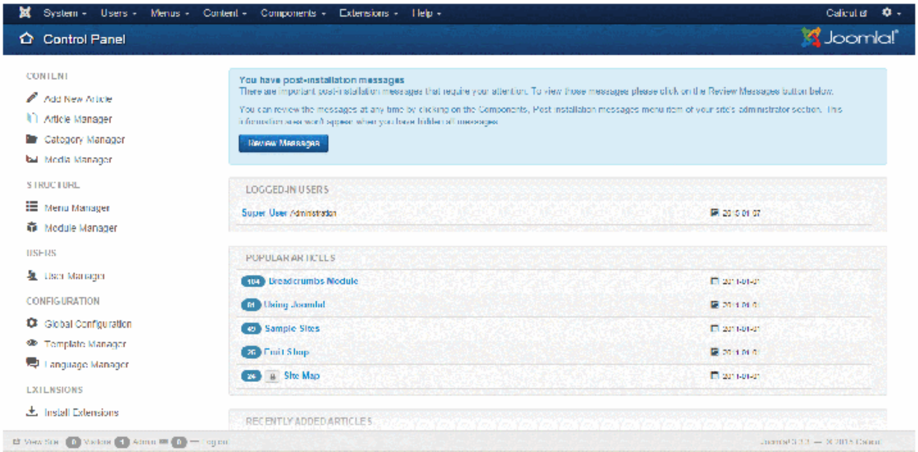
വെബ് ഹോസ്റ്റിങ് സൗകര്യം സൗജന്യമായി നൽകുന്നതാണ് സൗജന്യ ഹോസ്റ്റിംഗ്. ഇവിടെ സേവന ദാതാവ് ഈ വെബ്സൈറ്റിൽ ചില പരസ്യങ്ങൾ നൽകി ഇതിന്റെ ചെലവ് കണ്ടെത്തുന്നു. സൗജന്യവെബ്ഹോസ്റ്റിംഗ് സൈറ്റുകൾ നമ്മുടെ വെബ്സൈറ്റിലെ ഫയലുകളെ സെർവറിൽ സൂക്ഷിക്കുന്നു. എന്നാലിതിന് ചില നിയന്ത്രണങ്ങളുണ്ട്. ഫയലിന്റെ വലുപ്പം നിയന്ത്രണവിധേയമായിരിക്കും. (5 MBവരെ) ശബ്ദവും വീഡിയോയും ചിലതിൽ അനുവദനീയമല്ല. ചില വെബ്സൈറ്റുകൾ അതിന്റെ ഒരു ടെംപ്ലേറ്റ്

അടിസ്ഥാനമാക്കി വെബ്സൈറ്റ് രൂപകൽപന ചെയ്യുവാൻ മാത്രമേ അനുവദിക്കുകയുള്ളൂ. മറ്റ് ഫയലുകൾ വെബ്സെർവറിൽ അപ്‌ലോഡ് ചെയ്യുവാനും അനുവദിക്കില്ല. ചില സൗജന്യ വെബ്ഹോസ്റ്റിങ് കമ്പനികൾ ഡൊമൈൻ നെയിം രജിസ്ട്രേഷനും അനുവദിക്കുന്നു. ഈ വെബ്സൈറ്റുകൾ ഫയലുകളെ അപ്‌ലോഡ് ചെയ്യുന്നതിനുള്ള കൺട്രോൾ പാനലുകൾ ചിത്രം 7.10 -ലേതുപോലെ നൽകുന്നു. ചില വെബ്സൈറ്റുകൾ മുൻ നിശ്ചയിച്ച ഘടനയിൽ മാത്രമേ വെബ്സൈറ്റ് നിർമിക്കുവാൻ അനുവദിക്കുകയുള്ളൂ. സൗജന്യ വെബ്ഹോസ്റ്റിംഗ് സൗകര്യം അവരുടേതായ ഒരു സബ് ഡൊമൈനോ അല്ലെങ്കിൽ ഒരു ഡയറക്ടറി സർവ്വീസോ ഉപയോഗിച്ചാണ് നമ്മുടെ വെബ്സൈറ്റിനെ ലഭ്യമാക്കുവാൻ സഹായിക്കുന്നത്. ഒരേ ആശയങ്ങളുള്ളവരുമായി വിവരങ്ങൾ പങ്കിടാനും, ലാഭേച്ഛയില്ലാത്ത പ്രസ്ഥാനങ്ങൾക്കും അധിക ചെലവ് വഹിക്കാനാകാത്തവർക്കും സൗജന്യ വെബ്ഹോസ്റ്റിങ് സഹായകരമാണ്. വളരെ ചെലവ് കുറഞ്ഞ രീതിയിൽ വെബ്ഹോസ്റ്റിങ് സൗകര്യം നിലവിൽ വന്നതോടുകൂടി സൗജന്യ വെബ്ഹോസ്റ്റിങ് കുറഞ്ഞുവരുന്നു. sites.google.com, yola.com എന്നിവ സൗജന്യ വെബ്ഹോസ്റ്റിങ് സർവ്വീസുകളാണ്. ഹയർസെക്കന്ററി അധ്യാപകർക്കും വിദ്യാർഥികൾക്കും ഏറെ പ്രയോജനകരമായ വിവരങ്ങൾ നൽകുന്ന hsslive.in എന്ന വെബ്സൈറ്റിന് സൗജന്യ ഹോസ്റ്റിങ് ആണ് ഉപയോഗിച്ചിരിക്കുന്നത്. ചിത്രം 7.11 ൽ ഇവ പ്രദർശിപ്പിച്ചിരിക്കുന്നു.

ചിത്രം 7.11 : സൗജന്യ വെബ്ഹോസ്റ്റിംഗ്

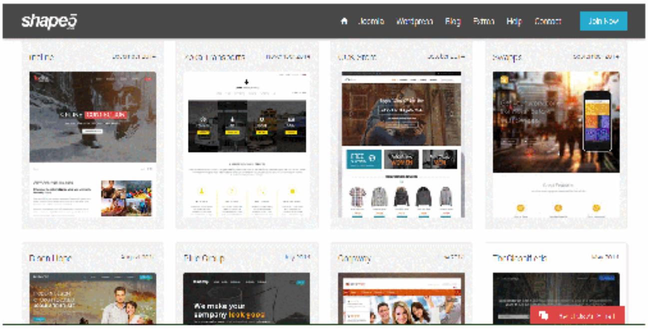
7.3 കണ്ടന്റ് മാനേജ്മെന്റ് സിസ്റ്റം (Content Management System)

വെബ്സൈറ്റുകൾ നിർമ്മിക്കുന്നതിനും നിയന്ത്രിക്കുന്നതിനും അവയിലേയ്ക്ക് പുതിയ വിവരങ്ങൾ ഉൾപ്പെടുത്തുന്നതിനും സാധ്യമായ ഒരു ഇന്റർനെറ്റ് അധിഷ്ഠിതമായ സോഫ്റ്റ് വെയർ സംവിധാനമാണ് കണ്ടന്റ് മാനേജ്മെന്റ് സിസ്റ്റം. മിക്ക CMS കളും അവയുടെ ഔദ്യോഗിക വെബ്സൈറ്റിൽ നിന്ന് സൗജന്യമായി ഡൗൺലോഡ് ചെയ്യാം. ഈ ഫയലുകളെ നമ്മുടെ വെബ്സൈറ്റിലെ ഹോസ്റ്റിങ് സ്ഥലത്തേക്ക് കോപ്പി ചെയ്യാം. അതിനു ശേഷം ചിത്രം 7.12 ലെപോലെ അതിനെ പ്രവർത്തനസജ്ജമാക്കുകയും ചെയ്യാം.



ചിത്രം 7.12 : ജൂംല ഉപയോഗിച്ച് വെബ്സൈറ്റ് പ്രവർത്തന സജ്ജമാക്കുന്നു

വെബ്സൈറ്റുകളിൽ CMS കളായി വേണ്ടി മുൻപ് രൂപകല്പന ചെയ്ത് വച്ചിട്ടുള്ള ചില ടെംപ്ലേറ്റുകൾ ലഭ്യമാകും. ചിത്രം 7.13-ൽ ഇവയുടെ ഉദാഹരണം നൽകിയിരിക്കുന്നു. ഉപയോക്താവിന് ഈ ടെംപ്ലേറ്റുകളിൽ ഒന്ന് തെരഞ്ഞെടുത്ത് നമ്മുടെ വെബ്സൈറ്റിൽ അപ്‌ലോഡ് ചെയ്യാം. അതിനു ശേഷം അതിന്റെ ശീർഷകങ്ങളും ചിത്രങ്ങളും മറ്റ് വിവരങ്ങളും നമ്മുടെ ഇഷ്ടപ്രകാരം മാറ്റുകയും ചെയ്യാം.



ചിത്രം 7.13 : ജൂംലയിൽ ലഭിക്കുന്ന ടെംപ്ലേറ്റുകൾ

ഇത് വേഡ് പ്രോസസ്സറിൽ ഒരു ഫയൽ നിർമിക്കുന്നത്ര ലളിതമാണ്. CMS-കളിൽ അടിസ്ഥാന നിയന്ത്രണ സംവിധാനങ്ങൾ അതിന്റെ ഡിസൈൻ സമയത്ത് ഉൾപ്പെടുത്തിയിട്ടുണ്ട്, ഇത് സാങ്കേതിക അറിവ് ഇല്ലാത്തവർക്ക് പോലും ഒരു വെബ്സൈറ്റ് നിർമിക്കാൻ സഹായകമാകും. CMS -ലെ ടെംപ്ലേറ്റുകൾ കോഡുകളുടെ ആവർത്തനം ഒഴിവാക്കുകയും ശീർഷകം രൂപകല്പന ചെയ്യുന്നതിനും എല്ലാപേജിലും മെനു ദൃശ്യമാക്കുന്നതിനും ചില പ്രത്യേക സംവിധാനങ്ങൾ ഒരുക്കിയിട്ടുണ്ട്. പ്രമുഖമായ ചില CMS വെബ്സൈറ്റുകളിൽ ഇവയുടെ ടെംപ്ലേറ്റുകൾ സൗജന്യമായോ മിതമായ തുകയ്ക്കോ ലഭ്യമാണ്. CMS-നെ നമ്മുടെ വെബ്സൈറ്റിന് അനുയോജ്യമാം

ചിത്രം 7.14 : ജൂംലയിൽ CMS ഉപയോഗിച്ച് വികസിപ്പിച്ച ഒരു വെബ്പേജ്

വിധം ക്രമപ്പെടുത്തി നൽകുന്ന വ്യക്തികളും സ്ഥാപനങ്ങളും ഇന്നുണ്ട്. ഒരു ഷെയേർഡ് വെബ്സെർവറിലാണ് വെബ്സൈറ്റ് ഹോസ്റ്റ് ചെയ്യുന്നതെങ്കിൽ അതിൽ CMS ഫയലുകൾ ഉൾപ്പെടുത്താനാകുമോ എന്ന് ശ്രദ്ധിക്കണം.

സാമ്പത്തികലാഭം ഉള്ളതുകൊണ്ട് ധാരാളം സംഘടനകളും ബ്ലോഗറുകളും അവരുടെ വെബ്സൈറ്റിനായി സി.എം.എസ്. ഉപയോഗിക്കുന്നു. പ്രമുഖമായ ചില CMS സോഫ്റ്റുവെയറുകളാണ് വേഡ്പ്രസ്, ദ്രുപാൽ, ജൂംല എന്നിവ ചിത്രം 1.4-ൽ ജൂംല ഉപയോഗിച്ച് നിർമ്മിച്ച മോട്ടോർ വാഹന വകുപ്പിന്റെ വെബ്സൈറ്റ് കാണാനാകും.

7.4 റെസ്പോൺസീവ് വെബ്രൂപകൽപന (Responsive webdesign)

ഇന്ന് ഇന്റർനെറ്റുമായി ബന്ധപ്പെടുവാൻ കമ്പ്യൂട്ടർ, ലാപ്ടോപ്പ്, ടാബ്ലെറ്റ്, മൊബൈൽ ഫോൺ തുടങ്ങിയ ധാരാളം ഉപകരണങ്ങൾ ഉണ്ട്. പല ഉപകരണങ്ങൾക്കും വ്യത്യസ്ത സ്ക്രീൻവലുപ്പമാണുള്ളത്. പണ്ട് നിർമിച്ചിട്ടുള്ള വെബ്പേജുകൾ ഡെസ്ക് ടോപ്പിനും ലാപ്ടോപ്പിനും അനുയോജ്യമാംവിധം ഉള്ളതാണ്. ഈ വെബ്സൈറ്റുകൾ ടാബ്ലെറ്റും, മൊബൈൽഫോണും വഴി കാണുവാൻ പ്രയാസമാണ്. അല്ലെങ്കിൽ ഉപഭോക്താവ് ഈ വെബ്പേജിന്റെ ഓരോ ഭാഗവും കാണുവാൻ സ്ക്രോൾബാർ വശങ്ങളിലേയ്ക്ക് നീക്കി നോക്കണം. മുൻകാലത്ത് മൊബൈൽവഴി വെബ്സൈറ്റ് കാണുവാൻ പ്രത്യേക വെബ് സൈറ്റുകൾ നിർമിച്ചിരുന്നു. ഇവ മൊബൈലിന്റെ സ്ക്രീൻ വലുപ്പത്തിന് അനുയോജ്യമാംവിധമാണ് നിർമിച്ചിരുന്നത്. എന്നാൽ ഒരു സ്ഥാപനത്തിന് രണ്ട് വെബ്സൈറ്റുകൾ കൈകാര്യം ചെയ്യേണ്ടിവരുന്നത് പ്രയാസമാണ്. അതുകൊണ്ട് വെബ്പേജിന്റെ വലുപ്പം സ്ക്രീനിന്റെ വലുപ്പത്തിന് അനുസരിച്ച് മാറുന്ന രീതിയിൽ രൂപപ്പെടുത്തിയാൽ അത് കൂടുതൽ പ്രയോജനകരമാകും. ഇത്തരത്തിലുള്ള വെബ്പേജ് രൂപകൽപനയെ റെസ്പോൺസീവ് വെബ്രൂപകൽപന എന്നു പറയും. വിവിധ ഉപകരണങ്ങളിൽ ഒരു റെസ്പോൺസീവ് വെബ്സൈറ്റ് എങ്ങനെ ദൃശ്യമാകുമെന്ന് ചിത്രം 7.15 ൽ കാണിച്ചിരിക്കുന്നു.

എല്ലാ ഉപകരണങ്ങളിലും പ്രവർത്തിക്കത്തക്ക വിധമാണ് റെസ്പോൺസീവ് വെബ്ഡിസൈനിംഗിൽ വെബ്സൈറ്റ് രൂപപ്പെടുത്തുന്നത്. അതുകൊണ്ടുതന്നെ സ്ക്രീൻ വലുപ്പമോ ഉപകരണമേതെന്നോ ഇവിടെ പരിഗണിക്കപ്പെടുന്നില്ല. റെസ്പോൺസീവ് വെബ്ഡിസൈനിങ് എന്ന ആശയം ഒരു സ്വതന്ത്ര ഡിസൈനറായ ഈഥാൻ മർക്കോട്ടിന്റേതാണ്. സ്ഥിരമായി മാറിക്കൊണ്ടിരിക്കുന്ന ഒരു വെബ്രൂപകൽപന രീതിയെപ്പറ്റി അദ്ദേഹം വിവരിക്കുന്നു. ഈ വെബ്രൂപകൽപന രീതി ഫ്ളക്സിബിൾ ഗ്രിഡ് ലേഔട്ടും ഫ്ളക്സിബിൾ ഇമേജും മീഡിയകൊറിയും ഉപയോഗിച്ച് പ്രാവർത്തികമാക്കിയിരിക്കുന്നു. ഫ്ളക്സിബിൾ ഗ്രിഡ് ലേഔട്ട് സൗകര്യം ഉപയോഗിച്ച് ഉപകരണത്തിന്റെ പ്രദർശനതലത്തിന്റെ വലുപ്പം പരിഗണിച്ച് വെബ്പേജിനെ അതിന് അനുസൃതമായി സജ്ജീകരിക്കുകയും ചെയ്യും. മീഡിയകൊറി ഓരോ ഉപകരണത്തിനും അനുസരിച്ചുള്ള വ്യത്യസ്ത ശൈലിയിൽ ലഭ്യമാക്കും. ഒരു വെബ് പേജിലെ തിരശ്ചീന മെനു മൊബൈലിൽ ഡ്രോപ്ഡൗൺ മെനു ആയി പരിവർത്തനം ചെയ്യപ്പെടും. CSS ഫയ

ചിത്രം 7.15 : റെസ്പോൺസീവ് വെബ്രൂപം

ലിനുള്ളിലെ മീഡിയകൊറിയാണ് ഇതിന് സഹായിക്കുന്നത്. ഇവിടെ പ്രദർശനതലത്തിന്റെ വലുപ്പം മാറുന്നതിന് അനുസരിച്ച് വെബ്സൈറ്റിന്റെ വലുപ്പവും ഭാവവും മാറിക്കൊണ്ടിരിക്കും.

നമുക്കു ചെയ്യാം

ഇന്ത്യൻ ക്രിക്കറ്റ് ടീം അംഗങ്ങളുടെ പേര്, വയസ്സ്, നേടിയ റണ്ണുകളുടെ എണ്ണം, എടുത്ത വിക്കറ്റുകളുടെ എണ്ണം എന്നിവ ഉൾക്കൊളളുന്ന ഒരു വെബ്സൈറ്റ് രൂപകൽപന ചെയ്ത് ഫ്രീ ഹോസ്റ്റിങ് സേവനങ്ങൾ ഉപയോഗിച്ച് ഇതിനെ ഹോസ്റ്റ് ചെയ്യുക.
ജനകീയമായ CMS ദാതാക്കളുടെ ഒരു ലിസ്റ്റ് തയ്യാറാക്കുന്നതിനോടൊപ്പം അവരുടെ സവിശേഷതകളും തയ്യാറാക്കുക.
നിങ്ങൾ തയാറാക്കിയ സ്കൂൾ വെബ്സൈറ്റുകളുടെ ഫയലുകൾ ഏതെങ്കിലും FTP സോഫ്റ്റ്വെയർ ഉപയോഗിച്ച് ലഭ്യമായ വെബ് ഹോസ്റ്റിങ് സ്ഥലത്ത് അപ്‌ലോഡ് ചെയ്യുക.

നമുക്കു സംഗ്രഹിക്കാം

ഒരു വെബ്സൈറ്റ് നിർമ്മിച്ചതിനുശേഷം അനുയോജ്യമായ ഒരു ഹോസ്റ്റിങ് രീതി ഉപയോഗിച്ച് ഇന്റർനെറ്റിൽ ലഭ്യമാക്കാം. ചെറിയ സ്ഥാപനങ്ങൾ തങ്ങളുടെ വെബ്സൈറ്റിന് ഷെയേർഡ് ഹോസ്റ്റിങ് ഉപയോഗിക്കും. ഉയർന്ന ട്രാഫിക്കുള്ളതും ഉയർന്ന സുരക്ഷ വേണ്ടതുമായ വെബ്സൈറ്റുകൾക്ക് വി.പി.എസ്. ഹോസ്റ്റിങ് ഉപയോഗിക്കും. എന്നാൽ വലിയ സ്ഥാപനങ്ങളുടെ വെബ്സൈറ്റുകളിൽ നിരന്തരം ഉയർന്ന ട്രാഫിക്കുകൾ ഉണ്ടാകുന്നതുകൊണ്ട് ഡെഡിക്കേറ്റഡ് ഹോസ്റ്റിങ് ഉപയോഗിക്കും. വെബ്ഹോസ്റ്റിങിന് ആവശ്യമായ സ്ഥലം വാങ്ങിയതിനുശേഷം FTP സോഫ്റ്റ്വെയർ ഉപയോഗിച്ച് നമ്മുടെ വെബ്സൈറ്റ് ഫയലുകളെ വെബ്സെർവറിലേയ്ക്ക് മാറ്റാം. അതിനുമുമ്പ് ഡൊമൈൻ നാമം സേവന ദാതാവ് വഴി രജിസ്റ്റർ ചെയ്യണം. വെബ്സെർവറിലെ നമ്മുടെ ഡൊമൈൻ സൂചിപ്പിക്കുവാൻ A റിക്കോർഡ് സജ്ജമാക്കണം. സൗജന്യനിരക്കിൽ വെബ്സൈറ്റിന് ലഭ്യമാക്കുന്ന വെബ്സെർവറുകൾ ലഭ്യമാണ്. കണ്ടന്റ് മാനേജ്മെന്റ് സംവിധാനം ചില ഉപകരണങ്ങളും ചില അടിസ്ഥാന സുരക്ഷിത സൗകര്യങ്ങളും ഒരുക്കി സാങ്കേതികവിദ്യകളിൽ അറിവ് കുറഞ്ഞ ആളുകൾക്കുപോലും വെബ്സൈറ്റുകൾ നിർമ്മിക്കുവാനുളള സാധ്യത നൽകുന്നു. വ്യത്യസ്ത ഉപകരണങ്ങൾ ഉപയോഗിച്ച് ബ്രൗസ് ചെയ്യുന്നതുകൊണ്ട് സ്ക്രീനിന്റെ വലുപ്പമനുസരിച്ച് വെബ്സൈറ്റുകൾ ലഭ്യമാക്കുക എന്നത് പ്രാധാന്യത്തോടെ കാണേണ്ടിയിരിക്കുന്നു. ഇതിനായി റസ്പോൺസീവ് വെബ്രൂപകൽപന ഉപയോഗിക്കുന്നു.

നമുക്കു വിലയിരുത്താം

1. വെബ്ഹോസ്റ്റിങ് എന്നതുകൊണ്ട് എന്താണ് അർഥമാക്കുന്നത്? പലതരത്തിലുള്ള വെബ്ഹോസ്റ്റിംഗിനെ കുറിച്ച് വിശദീകരിക്കുക.
2. നഗരത്തിലുള്ള ഒരു സൂപ്പർമാർക്കറ്റ് സ്ഥാപനം അവരുടെ വ്യാപാരം ഓൺ ലൈൻ ആക്കുവാൻ ഉദ്ദേശിക്കുന്നു. വെബ്സൈറ്റ് വഴി സാധനങ്ങൾ വിൽക്കാനും പണമിട പാടുകൾ ഓൺലൈൻ വഴി നടത്തുവാനും അവർ ആസൂത്രണം ചെയ്യുന്നുവെങ്കിൽ
   a. ഏത് തരത്തിലുള്ള ഹോസ്റ്റിങ് ആണ് ഈ വെബ്പേജിന് അനുയോജ്യമാകുക?
   b. അത് തിരഞ്ഞെടുക്കുവാനുള്ള കാരണവും വിശദമാക്കുക.
3. എമിൽ സ്വന്തം മരുന്നു കടയുടെ വെബ്സെർവറിൽ വെബ് ഹോസ്റ്റിംഗ് സ്ഥലം വാങ്ങുവാൻ ആഗ്രഹിക്കുന്നു. വെബ്സെർവറിൽ ഹോസ്റ്റിംഗിനായി സ്ഥലം വാങ്ങുമ്പോൾ പരിഗണിക്കേണ്ട പ്രത്യേകതകൾ പട്ടികപ്പെടുത്തുക.
4. ഒരു ഡൊമൈൻ നാമവും വെബ്സർവറിൽ സംഭരിച്ചിട്ടുള്ള വെബ്സൈറ്റും തമ്മിൽ എങ്ങനെയാണ് പരസ്പരം ബന്ധിപ്പിക്കുന്നത്?
5. FTP സോഫ്റ്റ്വെയറിൽ SFTP (പ്രോട്ടോകോൾ) ഉപയോഗിക്കുന്നത് കൊണ്ടുള്ള മേന്മയെന്താണ്?
6. സ്വന്തം കുടുംബത്തിനായി ഒരു വെബ്സൈറ്റ് ഹോസ്റ്റ് ചെയ്യുവാൻ രാജു ആഗ്രഹിക്കുന്നു. സൗജന്യ വെബ്സൈറ്റ് ഹോസ്റ്റിംഗ് കമ്പനികൾ നൽകുന്ന സംവിധാനങ്ങൾ എന്തൊക്കെയാണ്?
7. CMS എന്നാൽ എന്താണ്? CMS -ന്റെ പ്രത്യേകതകൾ എന്തൊക്കെ? ഉദാഹരണങ്ങൾ നൽകുക.
8. ഇന്നത്തെ കാലത്ത് വെബ്സൈറ്റിന്റെ നിർമ്മാണത്തിൽ റെസ്പോൺസീവ് വെബ്സൈറ്റ് രൂപകൽപനയുടെ ആവശ്യകത വിശദീകരിക്കുക.
9. എങ്ങനെയാണ് റെസ്പോൺസീവ് വെബ്പേജ് രൂപകൽപന നടപ്പിലാക്കുന്നത്?

സാങ്കേതിക പദാവലി

അംഗങ്ങൾ	elements
അടിസ്ഥാന ക്ലാസ്	base class
അടിസ്ഥാന ഡാറ്റ ഇനം	fundamental data type
അടിസ്ഥാന വിലാസം	base address
അടിസ്ഥാനപരമായ	fundamental
അന്തർനിർമ്മിത	buitl-in
അബ്സ്ട്രാക്ട് ഡാറ്റകൾ	abstract data types
അർദ്ധവിരാമം	semicolon (;)
അസ്ഥിര ദശാംശ സംഖ്യ	floating point number
അറേ	array
ആധാരസംഖ്യ	base number
ആധികാരികത	authentication
ആർഗ്യുമെൻറ്	argument
ഇരട്ട ഉദ്ധരണി	double quotes (")
ഉപയോഗിക്കുക	access
ഉപവാക്യ ടാഗ്	phrase tag
ഉൾപ്പെടുത്തുക	inster
ഒന്നിപ്പിക്കുക	grouping
ഒബ്ജക്റ്റ്	objetc
ഒബ്ജക്റ്റ് ഓറിയന്റഡ്	objetc oriented
ഓപ്പറേറ്റർ ഓവർലോഡിങ്	operator overloading
IP മേൽവിലാസം	IP addrsse
കമ്പ്യൂട്ടിങ് യന്ത്രം	computing machine
കമ്പ്യൂട്ടിങ് വിജ്ഞാനശാഖ	computing discipline
കമ്പ്യൂട്ടിങ് സിദ്ധാന്തം	theory of computing
കുറവ് ഓപ്പറേറ്റർ	decremetn operator()
കുറുകെ വെട്ടിയ	strikethrough
ക്യൂ	queue
ക്യൂ അണ്ടർഫ്ളോ	queue underflow
ക്യൂ ഓവർഫ്ളോ	queue overflow
ക്രമ ലിസ്റ്റ്	ordered list
ക്രമപ്പെടുത്തൽ	sorting
ക്രമപ്രകാരമുള്ള വർഗീകരണം	hierarchial classification
ക്രമീകരിക്കുക	set up

ഖണ്ഡം	module
ഘടന	template
ചലനാത്മക ചിത്രങ്ങൾ	animation
ചിഹ്നം	symbol
ചൂണ്ടുന്ന	pointing
R വില	R value
ടൈപ്പ് മോഡിഫയർ	type modifier
ടോപ് ഡൗൺ	top down
ഡബിൾ ലിങ്ക്ട് ലിസ്റ്റ്	double linked list
ഡാറ്റ അംഗങ്ങൾ	data clemetns
ഡാറ്റ അബ്സ്ട്രാക്ഷൻ	data abrstaction
ഡാറ്റ ഇനം	data type
ഡാറ്റ സ്ട്രക്ചർ	data strucutre
ഡൊമൈൻ നാമം	domain name
ഡൊമൈൻ നാമകരണ സമ്പ്രദായം	domain naming dysctm
തനതുവില	default valuc
തലക്കെട്ട്	heading
തിരയൽ	search
ത്രികോണമിതി	trigonomtery
ദ്വിതീയ മെമ്മറി	secondary memory
നൾ പോയിന്റർ	null pointer
നിർമാണം/വികസനം	develope
നിർമിത ബുദ്ധി	artificial intelligence
നിർവചനം	definition
നീക്കം ചെയ്യൽ	dcletion
നീക്കി വയ്ക്കുക	allocate
നെസ്റ്റഡ്	nested
നോഡ്	node
പകർത്തുക	copy
പദപ്രയോഗം	cxpression
പരിധി	scope
പരിപാലനം	maintainabiltiy
പരിവർത്തനാത്മകം	dynamic
പശ്ചാത്തലം	background
പുനരാരംഭിക്കാൻ	resttar
പുനരുപയോഗം	rcusabiltiy
പുഷ്	push

പൊതു അവലോകനം	overview
പൊരുത്തപ്പെടൽ	alignment
പോപ്	pop
പോയിൻറർ	pointer
പോസ്റ്റ്ഫിക്സ്	postfix
പ്രഖ്യാപനം	declaration
പ്രഖ്യാപന പ്രസ്താവന	declaration statemetn
പ്രദർശിപ്പിക്കുക/ദൃശ്യമാക്കുക	dipslay
പ്രവർത്തനം	activity
പ്രവർത്തനം	operation
പ്രവർത്തിപ്പിക്കൽ/നടപ്പിലാക്കൽ	execution
പ്രശ്നങ്ങൾ നിർധാരണം ചെയ്യുക	problemsolving
പ്രസ്താവന	statement
പ്രാഥമിക മെമ്മറി	primary memory
പ്രാഥമികമായ ഡാറ്റാ സ്ട്രക്ച്ചർ	simple datarustcurte
പ്രാദേശിക DNS	local DNS
പ്രാരം വില നൽകൽ	initialization
പ്രാവർത്തികമാക്കൽ	implementation
പ്രീഫിക്സ്	prefix
പ്രോഗ്രാമിങ് മാതൃക	programming paradigm
ഫങ്ഷൻ	function
ബാഹ്യ ഇനം മാറ്റൽ	explicity type conversion
ബാഹ്യഫയൽ	external file
മെമ്മറി വലുപ്പം	memory size
മെമ്മറിസ്ഥാനം	memory location
രൂപകല്പന	design
രൂപഘടന	architecture
രൂപഘടന നൽകൽ	formatting
രൂപമാറ്റം വരുത്തുക	encrytp
രൂപീകൃത ക്ലാസ്	derived class
രൂപീകൃത ഡാറ്റ ഇനം	derived data type
രൂപീകൃത ഡാറ്റ ഇനം	derived data type
രേഖീയം	linear
രേഖീയമല്ലാത്തത്	non liner
ലയിക്കുക	merging
ലിഖിതം	document
വർദ്ധനവ് ഓപ്പറേറ്റർ	incremetn operator

വഹനീയത	portabiltiy
വാക്യ ഘടന	syntax
വിദൂര സെർവർ	remote server
വില നൽകൽ പ്രസ്താവന	assignment statement
വിലാസ സമ്പ്രദായം	Addressing scheme
വൃത്താകൃത ക്യൂ	circular queue
ശകലം	segment
ശീർഷകം	caption
സന്ദേശ ജാലകം	message window
സമഗ്രത	integrtiy
സമർപ്പണം	submission
സമ്പർക്കമുഖം	interface
സംയോചിതം/ഏകോപിതം	grouped
സംയോജിത/ഏകോപിത ഡാറ്റ സ്ട്രക്ചർ	compound data structure
സവിശേഷ വസ്തുക്കൾ	entitise
സാർഥക ചിഹ്നങ്ങൾ	reserved character
സിംഗിൾ ഇൻഹെറിറ്റൻസ്	single inheritance
സിംഗിൾ ലിങ്ക്ട് ലിസ്റ്റ്	singly linked list
സുരക്ഷിതം	secure
സുരക്ഷിതത്വം	saftey
സൂക്ഷിക്കുക/സം രിക്കുക	sorte
സെർവർ നവീകരണം	server update
സ്ക്രീൻ വലുപ്പം	screen size
സ്ട്രക്ച്ചർ	structure
സ്ഥാനീയ സംഖ്യാന സമ്പ്രദായം	positional value number system
സ്ഥാപിക്കുക	load
സ്ഥിര ഡാറ്റ സ്ട്രക്ച്ചർ	static data strucurte
സ്ഥിരവില	constant
സ്വയം സൂചിത	self referential

പരാമർശം/സൂചന (References)

- Stroustrup, B. (2013). *The C++ Programming Language*. New Delhi : Addison-Wesley Professional
- Lafore, R. (2009). *Object-Oriented Programming in C++*. Chennai : Sams Publishing
- Balagurusamy, E. (2008). *Object Oriented Programming with C++*. New Delhi: Tata McGraw-Hill Education
- Sharma, A. K. (2011). *Data structure using C*. New Delhi : Pearson Education India
- Srivastava, S. K. & Srivastava, D. (2011). *Data Structures Through C in Depth*. New Delhi : BPB Publications
- Powel, T. A. (2010). *The Complete Reference : HTML & XHTML*. New Delhi: OSborne/Tata MC Graw-Hill
- Lloyd, I. (2008). *The Ultimate HTML Reference*. Melbourne : Sitepoint
- Holzner, S. (2000). *HTML Black Book*. New Delhi : DreamTech Press
- Frain, B. (2012). *Responsive Web Design with HTML5 and CSS3*. Mumbai : Packt Publishing
- Powel, T. A. & Schenider, F. (2008). *The Complete Reference JavaScript*. New Delhi : Tata McGraw-Hill
- Zakas, N. C. (2012). *Professional JavaScript for Web Developers*. Birmingham : Wrox