ഹയർ സെക്കന്ററി കോഴ്സ്

XI

# 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 ഉത്ക്കല 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

# 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 സാംസ്കാരികത്തനിമയുടെ തിരിച്ചറിയൽ കൂടിയാണ്. അതുകൊണ്ടാണ് വികസിതരാജ്യങ്ങൾ മാതൃഭാഷയെ മുഖ്യബോധന മാധ്യമമായി സ്വീകരിച്ചിരിക്കുന്നത്. ഇന്ത്യയിലാകട്ടെ, ദേശീയതലത്തിലുള്ള പ്രധാന പരീക്ഷകളെല്ലാം പ്രാദേശിക ഭാഷകളിൽക്കൂടി നടത്തുന്നതിനുള്ള സംവിധാനവും ഉണ്ടായിവരികയാണ്. ഈയൊരു സാഹചര്യത്തിൽ നമ്മുടെ കുട്ടികളും മാതൃഭാഷയുടെ ശക്തിസൗന്ദര്യങ്ങൾ തിരിച്ചറിഞ്ഞ് വിവിധ വിഷയങ്ങളിൽ ജ്ഞാനനിർമിതിയിൽ ഏർപ്പെടേണ്ടതുണ്ട്. അതിന് അവരെ സജ്ജരാക്കുകയാണ് ഈ പാഠപുസ്തകങ്ങളുടെ മുഖ്യ ലക്ഷ്യം.

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 പാഠപുസ്തക രചന 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 നിർമ്മാണ സമിതി

## കമ്പ്യൂട്ടർ സയൻസ്

**ശ്രീ. ജോയി ജോൺ**
എച്ച്.എസ്.എസ്.ടി, സെന്റ്. ജോസഫ്
എച്ച്.എസ്.എസ്. തിരുവനന്തപുരം

**ശ്രീ. അസീസ് വി.**
എച്ച്.എസ്.എസ്.ടി, ജി.എച്ച്.എസ്.എസ്
വെള്ളിയോട്, കോഴിക്കോട്

**ശ്രീ. റോയ് ജോൺ**
എച്ച്.എസ്.എസ്.ടി, അലോഷ്യസ് എച്ച്.എസ്.എസ്.
എൽതുരുത്ത്, തൃശൂർ

**ശ്രീ. അബൂബക്കർ പി.**
എച്ച്.എസ്.എസ്.ടി, ഗവ. ജി.എച്ച്.എസ്.എസ്.
ചാലപ്പുറം, കോഴിക്കോട്

**ശ്രീ. ഷാജൻ ജോസ് എൻ.**
എച്ച്.എസ്.എസ്.ടി, സെന്റ്. ജോസഫ്
എച്ച്.എസ്.എസ്. പാവറട്ടി, തൃശൂർ

**ശ്രീ. അഫ്സൽ കെ.എ.**
എച്ച്.എസ്.എസ്.ടി, ജി. എച്ച്.എസ്.എസ്. ശിവപുരം,
കരിയാത്തൻകര പി.ഒ., കോഴിക്കോട്

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

**ശ്രീ. വിനോദ് വി.**
എച്ച്.എസ്.എസ്.ടി, എൻ.എസ്.എസ്.
എച്ച്.എസ്.എസ്., പ്രാക്കുളം, കൊല്ലം

**ശ്രീ. രാജമോഹൻ സി.**
എച്ച്.എസ്.എസ്.ടി, നാവമുകുന്ദ എച്ച്.എസ്.എസ്.
തിരുനാവായ, മലപ്പുറം

**ശ്രീ. എ.എസ്. ഇസ്മൈൽ**
എച്ച്.എസ്.എസ്.ടി, എച്ച്.എസ്.എസ്.
പ്ലാപ്പെട്ടി, മലപ്പുറം

**ശ്രീ. സുനിൽ കാര്യത്തൻ**
എച്ച്.എസ്.എസ്.ടി, ഗവ. ബ്രണ്ണൻ എച്ച്.എസ്.എസ്.,
തലശ്ശേരി

**ശ്രീ. സായ് പ്രകാശ് എസ്.**
എച്ച്.എസ്.എസ്.ടി, സെന്റ്. തോമസ്
എച്ച്.എസ്.എസ്. പൂന്തുറ, തിരുവനന്തപുരം

### വിദഗ്ധർ

**ഡോ. ലജീഷ് വി.എൽ.**
അസിസ്റ്റന്റ് പ്രൊഫസർ, ഡിപ്പാർട്ട്മെന്റ് ഓഫ്
കമ്പ്യൂട്ടർ സയൻസ്, കാലിക്കറ്റ് യൂണിവേഴ്സിറ്റ്

**ഡോ. മധു എസ്. നായർ**
അസിസ്റ്റന്റ് പ്രൊഫസർ, ഡിപ്പാർട്ട്മെന്റ് ഓഫ്
കമ്പ്യൂട്ടർ സയൻസ്, കേരള സർവകലാശാല

**ശ്രീ. മധു വി.ടി.**
ഡയറക്ടർ, കമ്പ്യൂട്ടർ സെന്റർ,
കാലിക്കറ്റ് യൂണിവേഴ്സിറ്റി

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

**ഡോ. സുശീൽ കുമാർ ആർ.**
അസോസിയേറ്റ് പ്രൊഫസർ, ഡിപ്പാർട്ട്മെന്റ് ഓഫ്
ഇംഗ്ലീഷ്, ഡി.ബി. കോളേജ്, ശാസ്താംകോട്ട

**ഡോ. വിനീത് കെ. പലേരി**
പ്രൊഫസർ, ഡിപ്പാർട്ട്മെന്റ് ഓഫ് സയൻസ് ആന്റ്
എഞ്ചിനീയറിംഗ്, എൻ.ഐ.ടി., കോഴിക്കോട്

**മഹേശ്വരൻ നായർ. വി.**
സബ് ഡിവിഷണൽ എഞ്ചിനീയർ, റീജിയണൽ
ടെലികോം ട്രെയിനിങ് സെന്റർ, തിരുവനന്തപുരം

### ആർട്ടിസ്റ്റ്

സുധീർ വൈ  വിനീത് വി

### അക്കാദമിക് കോർഡിനേറ്റർ

**ഡോ. മീന എസ്.**
റിസർച്ച് ഓഫീസർ, എസ്.സി.ഇ.ആർ.ടി

**ശ്രീമതി. ജാൻസി റാണി എ.കെ.**
റിസർച്ച് ഓഫീസർ, എസ്.സി.ഇ.ആർ.ടി

# പാഠപുസ്തക പരിഭാഷ സമിതി
## (മലയാളം)


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

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

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

**ശ്രീ. മുഹമ്മദ് സർഫറാസ്**
എച്ച്.എസ്.എസ്.ടി, ഡി.എച്ച്.എസ്.എസ്.
നെല്ലിപ്പുഴ

**ശ്രീ. വിനോദ് കുമാർ വി.**
എച്ച്.എസ്.എസ്.ടി, ഗവ. എച്ച്.എസ്.എസ്.,
ഇളംകുന്നപ്പുഴ,
എറണാകുളം

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

**ശ്രീ. സുനിൽകുമാർ മാത്തോടൻ**
എച്ച്.എസ്.എസ്.ടി, ജി.എച്ച്.എസ്.എസ്.
അഴീക്കോട്, കണ്ണൂർ

**ശ്രീ. സുധീർ പി.എസ്.**
എച്ച്.എസ്.എസ്.ടി, എ.കെ.എൻ.എം.എ.
മെമ്മോറിയൽ എച്ച്.എസ്.എസ്.
കാട്ടുകുളം, പാലക്കാട്

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

**ശ്രീ. ശ്രീജിത്ത് പി.**
സി.ജെ. എച്ച്.എസ്.എസ്.ടി, ചെമ്മനാട്,
കാസർഗോഡ്

**അക്കാദമിക് കോർഡിനേറ്റർ**

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

# ഉള്ളടക്കം



Z8X4C9

പാഠപുസ്തകത്തിൽ ഉപയോഗിച്ചിരിക്കുന്ന സൂചനകൾ

നമുക്ക് ചെയ്യാം

നിങ്ങളുടെ പുരോഗതി അറിയുക

ഇൻഫർമേഷൻ ബോക്സ്

നമുക്ക് പരിശീലിക്കാം

നമുക്ക് സംഗ്രഹിക്കാം

1

# കമ്പ്യൂട്ടിങ് വിജ്ഞാനശാഖ

## പ്രധാന ആശയങ്ങൾ

- **കമ്പ്യൂട്ടിങ്ങിലെ നാഴികക്കല്ലുകളും യന്ത്രപരിണാമവും**
  - എണ്ണലും, സംഖ്യാനസമ്പ്രദായത്തിന്റെ വളർച്ചയും
  - കമ്പ്യൂട്ടിങ് യന്ത്രങ്ങളുടെ വളർച്ച
- **കമ്പ്യൂട്ടറിന്റെ തലമുറകൾ**
  - ഒന്നാം തലമുറ കമ്പ്യൂട്ടറുകൾ
  - രണ്ടാം തലമുറ കമ്പ്യൂട്ടറുകൾ
  - മൂന്നാം തലമുറ കമ്പ്യൂട്ടറുകൾ
  - നാലാം തലമുറ കമ്പ്യൂട്ടറുകൾ
  - അഞ്ചാം തലമുറ കമ്പ്യൂട്ടറുകൾ
- **കമ്പ്യൂട്ടിങ്ങിന്റെ പരിണാമം**
  - പ്രോഗ്രാമിംഗ് ഭാഷകൾ
  - അൽഗോരിതവും കമ്പ്യൂട്ടർ പ്രോഗ്രാമും
  - കമ്പ്യൂട്ടിങ്ങിന്റെ സിദ്ധാന്തം


ഒരു തരത്തിലല്ലെങ്കിൽ മറ്റൊരുതരത്തിൽ കമ്പ്യൂട്ടർ ഇന്ന് ജീവിതത്തിന്റെ മിക്കവാറും എല്ലാ മേഖലകളിലും സ്വാധീനം ചെലുത്തിക്കൊണ്ടിരിക്കുന്നു. ഏകദേശം എല്ലാവരും തന്നെ ഇന്ന് കമ്പ്യൂട്ടർ ഉപയോഗിക്കുന്നവരാണ്. ഇതിൽ പലരും പ്രോഗ്രാം തയാറാക്കാൻ കഴിവുള്ളവരും ആണ്. കമ്പ്യൂട്ടറിനെ നമ്മുടെ ഇഷ്ടാനുസരണം പ്രവർത്തിപ്പിക്കുക എന്നത് ശ്രമകരമായ പ്രവൃത്തിയാണ്. ഉയർന്നതലത്തിൽ ചിന്തിച്ചാൽ കമ്പ്യൂട്ടർ സയൻസ്, കമ്പ്യൂട്ടർ ഉപയോഗിച്ചുള്ള പ്രശ്ന പരിഹരണത്തിന്റെ ശാസ്ത്രശാഖയാണ്. കമ്പ്യൂട്ടർ ശാസ്ത്രജ്ഞൻമാർ യഥാർഥ ജീവിതത്തിലെ പ്രശ്നങ്ങൾ വിശകലനം ചെയ്യാനും, അവ പരിഹരിക്കാനും കഴിവുള്ളവരായിരിക്കണം. ഈ വിജ്ഞാന ശാഖയ്ക്ക് അൽഗോരിത രൂപീകരണം പോലെയുള്ള സൈദ്ധാന്തിക വിഷയങ്ങളും, കമ്പ്യൂട്ടർ ആപ്ലിക്കേഷൻ തയാറാക്കൽ പോലുള്ള പ്രായോഗിക വിഷയങ്ങളും കൈകാര്യം ചെയ്യാനുള്ള ശേഷിയുണ്ട്. വിവരങ്ങളുടെ വിവരണത്തിനും രൂപമാറ്റത്തിനും ഉതകുന്ന അൽഗോരിത പ്രവർത്തനങ്ങളുടെ (സിദ്ധാന്തം, വിശകലനം, നിർമ്മാണം, കാര്യക്ഷമത, നടപ്പിൽ വരുത്തൽ, പ്രയോഗം തുടങ്ങിയവ) ചിട്ടയായ പഠനം കമ്പ്യൂട്ടർ സയൻസ് എന്ന വിജ്ഞാനശാഖയിൽ ഉൾപ്പെടുന്നു.

കമ്പ്യൂട്ടിങ് എന്ന ആശയം പഴയകാല അബാക്കസു മുതൽ ഇന്നത്തെ സൂപ്പർ കമ്പ്യൂട്ടർ വരെയുള്ള യന്ത്രങ്ങളുടെ പ്രവർത്തനങ്ങളിൽ നിന്ന് ഉരുത്തിരിഞ്ഞുണ്ടായതാണ്. വിവിധ കമ്പ്യൂട്ടിങ് യന്ത്രങ്ങളുടെ പരിണാമത്തെക്കുറിച്ചും കമ്പ്യൂട്ടറിന്റെ വിവിധ തലമുറകളെക്കുറിച്ചും ഈ അധ്യായത്തിൽ ചർച്ച ചെയ്യുന്നു. കൂടാതെ പ്രോഗ്രാമിങ് ഭാഷകളുടെ വളർച്ചയെക്കുറിച്ചും അലൻട്യൂറിംഗ് എന്ന ശാസ്ത്രജ്ഞന്റെ സംഭാവനകളെക്കുറിച്ചും ഇവിടെ വിവരിക്കുന്നു.

# 1.1. കമ്പ്യൂട്ടിങ്ങിലെ നാഴികക്കല്ലുകളും യന്ത്രപരിണാമവും (Computing milestones and machine evolution)

പ്രാചീനകാലത്ത് മനുഷ്യൻ എണ്ണുവാൻ വേണ്ടി ഉപയോഗിച്ചിരുന്നത് കല്ലുകളായിരുന്നു. അവർ ചുമരിൽ വരകൾ കോറിയിട്ടും, ചരടിൽ കെട്ടുകളിട്ടും വിവരങ്ങൾ രേഖപ്പെടുത്തിയിരുന്നു. ഇതിന്റെ തുടർച്ചയായി മനുഷ്യന്റെ കമ്പ്യൂട്ടിങ് ശക്തിയെ യന്ത്രങ്ങളുപയോഗിച്ച് ചെയ്യുവാൻ വേണ്ടിയുള്ള ശ്രമം നടന്നു. എണ്ണുന്നതിനുള്ള പഴയരീതികളെക്കുറിച്ചും, സ്ഥാനീയ സംഖ്യാനസമ്പ്രദായ (Positional Number System) ത്തെക്കുറിച്ചും നമുക്ക് ഇവിടെ ചർച്ച ചെയ്യാം.

## 1.1.1. എണ്ണലും സംഖ്യാസമ്പ്രദായ (Number System) ത്തിന്റെ ക്രമാനുഗതമായ വളർച്ചയും (Counting and the evolution of the positional number system)

ചരിത്രം എഴുതപ്പെടുന്നതിനും എത്രയോ മുമ്പ് തന്നെ മനുഷ്യന്റെ സംഖ്യാ ബോധവും എണ്ണൽ പ്രക്രിയയും വികാസം പ്രാപിച്ചിരുന്നു. ആദിമ മനുഷ്യന് തന്നെ ''കൂടുതൽ'', ''കുറവ്'' എന്നീ ആശയങ്ങൾ അറിമായിരുന്നു എന്ന് വിശ്വസിക്കപ്പെടുന്നു. മനുഷ്യൻ വർഗങ്ങളും ഗോത്രങ്ങളും ആയി ജീവിച്ചപ്പോൾ സ്വന്തം വിഭാഗത്തിലെയും ശത്രുപാളയത്തിലെയും അംഗങ്ങളുടെ എണ്ണം അറിയേണ്ടത് അത്യാവശ്യമായി വന്നു. ആട്ടിൻപറ്റങ്ങളുടെയും മറ്റ് വളർത്തുമൃഗങ്ങളുടെയും എണ്ണം കൂടുതലാണോ കുറവാണൊ എന്നറിയുക പ്രാധാന്യമുള്ള വിഷയമായിരുന്നു. എണ്ണൽ എന്ന പ്രക്രിയയ്ക്ക് ആദ്യകാലത്ത് വടികളും കല്ലുകളും ആയിരുന്നു ഉപയോഗിച്ചിരുന്നത്.

ഇനി നമുക്ക് വിവിധ സംഖ്യാ സമ്പ്രദായങ്ങൾ എങ്ങനെ വികസിച്ചു വന്നു എന്ന് ചിന്തിക്കാം. എന്തായാലും ഇന്നു കാണുന്ന സംഖ്യാ സമ്പ്രദായങ്ങൾ അനേകായിരം വർഷങ്ങൾ കൊണ്ട് വികസിച്ചു വന്നതാണെന്ന് നമുക്ക് അനുമാനിക്കാം. ഈ വികാസത്തിന് അനേകം ഗോത്രവർഗങ്ങളും, സംസ്കാരങ്ങളും അതിന്റേതായ സംഭാവനകൾ നൽകിയിട്ടുണ്ട്. സംഖ്യാ സമ്പ്രദായമെന്നാൽ സംഖ്യകൾ എഴുതുന്നതിനുള്ള വിവിധ രീതികളാണ്. സംഖ്യാ സമ്പ്രദായത്തിന്റെ ക്രമാനുഗതമായ വളർച്ചാചരിത്രം ചുവടെ വിവരിക്കുന്നു.

3000 BC യിൽ ആവിർഭവിച്ച ഈജിപ്ഷ്യൻ സംഖ്യാ സമ്പ്രദായത്തിൽ നമുക്ക് തുടങ്ങാം. ഈ സംഖ്യ സമ്പ്രദായം 10 നെ ആധാരസംഖ്യ (Base) ആയി ഉപയോഗിച്ചു. ഈ സമ്പ്രദായത്തിൽ 1 മുതൽ 9 വരെയും, 10 മുതൽ 90 വരെയും, 100 മുതൽ 900 വരെയും, 1000 മുതൽ 9000 വരെയും വ്യത്യസ്തമായ ചിഹ്നങ്ങൾ (Symbols) ഉപയോഗിച്ചു. ഈജിപ്തുകാർ സംഖ്യ വലത്തു നിന്ന് ഇടത്തോട്ടാണ് എഴുതിയത്. അതായത് ഒരു സംഖ്യയിൽ 10 ന്റെ ഏറ്റവും വലിയ മൂല്യം വലത്തേ അറ്റത്തെ അക്കത്തിനായിരുന്നു.

പിന്നീട് സുമേറിയൻ /ബാബിലോണിയൻ സംഖ്യാ സമ്പ്രദായത്തിന്റെ കാലഘട്ടമായിരുന്നു. ഇവിടെ 60 ആയിരുന്നു ആധാര സംഖ്യ. ഇത് സെക്സാജസിമൽ സംഖ്യാസമ്പ്രദായമെന്ന് അറിയപ്പെട്ടു. സംഖ്യയിൽ അക്കങ്ങൾ ഇടത്ത് നിന്ന് വലത്തോട്ടാണ് എഴുതിയിരുന്നത്. സംഖ്യ സമ്പ്രാദയങ്ങളുടെ ചരിത്രത്തിൽ ഏറ്റവും വലിയ ആധാരസംഖ്യ ഈ സംഖ്യാ സമ്പ്രദായത്തിനായിരുന്നു. ഇവർ പൂജ്യം അടയാളപ്പെടുത്തുന്നതിന് ചിഹ്നം ഉപയോഗിച്ചിരുന്നില്ല. പക്ഷേ, പൂജ്യം എന്ന ആശയം നിലനിന്നിരുന്നു. പൂജ്യം രേഖപ്പെടുത്തേണ്ടി വരുമ്പോൾ അവർ സംഖ്യയിൽ ഒരു ഒഴിവ് (Space) രേഖപ്പെടുത്തുമായിരുന്നു.

2500 BC യിൽ ആണ് ചൈനീസ് സംഖ്യാസമ്പ്രദായം ഉപയോഗിച്ചിരുന്നത്. ഇത് എളുപ്പമുള്ളതും വളരെ സൗകര്യപ്രദവുമായിരുന്നു. 1 മുതൽ 9 വരെയുള്ള അക്കങ്ങൾ ഉപയോഗിച്ചു. 10 ആയിരുന്നു ആധാരസംഖ്യ. ഈ രീതിക്ക് ഇന്നത്തെ സംഖ്യാസമ്പ്രദായവുമായി വളരെ സാദൃശ്യമുണ്ടായിരുന്നു. മുളങ്കമ്പുകൾ കൊണ്ടാണ് അന്ന് സംഖ്യ രേഖപ്പെടുത്തിയിരുന്നത്.

ഏകദേശം 500 BC യിൽ ഗ്രീക്ക് സംഖ്യാസമ്പ്രദായം (അയോണിയൻ സംഖ്യാ സമ്പ്രദായം) വികസിക്കപ്പെട്ടു. ഇത് 10 ആധാരമായ സമ്പ്രദായമായിരുന്നു. ഇവർക്കും പൂജ്യം അടയാളപ്പെടുത്താൻ ചിഹ്നമില്ലായിരുന്നു.

റോമൻകാർ ഗണിതശാസ്ത്രം പ്രായോഗിക തലത്തിൽ കൂടുതലായി ഉപയോഗിക്കുന്നതിന് തുടക്കം കുറിച്ചു. റോഡ്, പാലം തുടങ്ങിവയുടെ നിർമ്മാണത്തിൽ ഇവർ ഗണിതം ഉപയോഗിച്ചു. 7 ചിഹ്നങ്ങൾ (I,V, X, L, C, D, M) ഉപയോഗിച്ചാണ് ഇവർ സംഖ്യകൾ അടയാളപ്പെടുത്തിയിരുന്നത്.

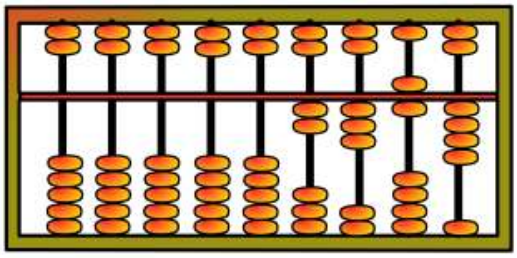
മായൻകാർ 20 ആധാരമായ സംഖ്യാ സമ്പ്രദായമായിരുന്നു ഉപയോഗിച്ചിരുന്നത്. നമ്മുടെ കൈകാലുകളിൽ ആകെ 20 വിരലുകൾ ഉള്ളതിനാലാണ് 20 എന്ന ആധാര സംഖ്യ സ്വീകരിച്ചത്. ഈ സംഖ്യാ സമ്പ്രദായം ശരിയായ ജ്യോതിശ്ശാസ്ത്രനിരീക്ഷണങ്ങൾക്കും കൂടുതൽ കൃത്യമായ അളവുകൾ രേഖപ്പെടുത്തുന്നതിനും വ്യാപകമായി ഉപയോഗിച്ചു.

ഏതാണ്ട് 1500 വർഷം മുമ്പാണ് ഇന്ന് നാം ഉപയോഗിക്കുന്ന ഹിന്ദു-അറബിക് സംഖ്യാനസമ്പ്രദായം ഇന്ത്യയിൽ പിറവിയെടുത്തത്. ഇത് സ്ഥാനവില ക്രമമുള്ള ഒരു ദശസംഖ്യാന സമ്പ്രദായമായിരുന്നു. ഇതിൽ പൂജ്യം അടയാളപ്പെടുത്തുന്നതിന് പ്രത്യേകം ചിഹ്നം ഉപയോഗിച്ചു. പൂജ്യത്തിന്റെ കണ്ടുപിടിത്തം, ലോകത്തിനുള്ള ഇന്ത്യയുടെ ഒരു വലിയ സംഭാവനയായ് കണക്കാക്കപ്പെടുന്നു. പിന്നീട് പല രാജ്യങ്ങളും ഈ സംഖ്യാസമ്പ്രദായം ഉപയോഗിച്ചു. ഇനി നമുക്ക് കമ്പ്യൂട്ടിങ് യന്ത്രങ്ങളുടെ പരിണാമത്തെ കുറിച്ച് ചർച്ച ചെയ്യാം.

## 1.1.2. കമ്പ്യൂട്ടിങ് യന്ത്രങ്ങളുടെ പരിണാമം (Evolution of the computing machine)

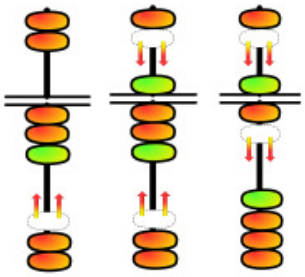
3000 BC മുതൽ 1450 AD വരെയുള്ള കാലഘട്ടത്തിൽ മനുഷ്യൻ ലളിതമായ ചിത്രങ്ങളിലൂടെയും, പിന്നീട് എഴുത്തുകളിലൂടെയും ആശയവിനിമയം നടത്തിയിരുന്നു. സംഖ്യകളുടെ കണ്ടുപിടിത്തം അബാക്കസ് എന്ന യന്ത്രത്തിന്റെ കണ്ടുപിടിത്തത്തിലേക്ക് വഴി തെളിച്ചു. അബാക്കസ്സാണ് ആദ്യത്തെ കമ്പ്യൂട്ടിങ് യന്ത്രമായി അറിയപ്പെടുന്നത്. താഴെ കൊടുത്ത വിവരണത്തിൽ നിന്നും കമ്പ്യൂട്ടർ യന്ത്രപരിണാമത്തിലെ ചില പ്രധാന നാഴികക്കല്ലുകൾ നമുക്കു പരിചയപ്പെടാം.

### a. അബാക്കസ് (Abacus)



*ചിത്രം 1.1 അബാക്കസ്*

ഏകദേശം 3000 BC യോടടുത്താണ് മെസൊപ്പൊട്ടാമിയൻമാർ അബാക്കസ് കണ്ടുപിടിച്ചത്. അബാക്കസ് എന്ന വാക്കിന്റെ അർഥം കണക്കുകൂട്ടുന്ന ബോർഡ് എന്നാണ്. ചെറിയ കമ്പിയിലൂടെ ചലിപ്പിക്കാൻ കഴിയുന്ന മുത്തുകളാണ് (Beeds) അബാക്കസ്സിന്റെ പ്രധാന ഘടകം. കമ്പികളെ രണ്ട് ഭാഗമായി വിഭജിച്ചിരിക്കുന്നു. അബാക്കസ്സിനെ അടിസ്ഥാന ഗണിതക്രിയകൾ ചെയ്യാനുള്ള ആദ്യത്തെ ഉപകരണമായി കണക്കാക്കുന്നു. അബാക്കസ്സിന്റെ ചിത്രം 1.1. ൽ കാണിച്ചിരിക്കുന്നു.

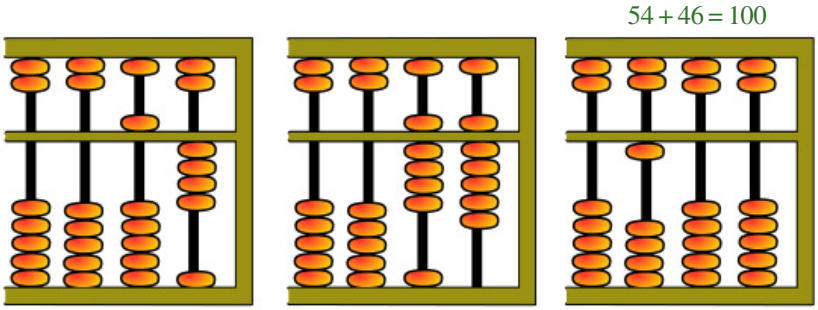



*ചിത്രം 1.2 (a) അബാക്കസ് ഉപയോഗിച്ചുള്ള കണക്ക് കൂട്ടൽ*

അബാക്കസ്സിനെ കൗണ്ടിംഗ് ഫ്രെയിം എന്നും വിളിച്ചിരുന്നു. ഇത് അടിസ്ഥാന ഗണിതക്രിയകൾ ചെയ്യാനുള്ള ഒരു ഉപകരണമാണ്. ഇതിൽ ലംബമായുള്ള മൂന്ന് വയറുകളിൽ ഓരോന്നിലും 7 മുത്തുകൾ ചേർത്ത് ചൈനക്കാർ അബാക്കസ്സിനെ പരിഷ്കരിച്ചു. ഒരു തിരശ്ചീനമായ ബാർ 7 മുത്തുകളെയും മുകളിൽ രണ്ട്, താഴെ അഞ്ച് എന്ന ക്രമത്തിൽ വിഭജിക്കുന്നു. അക്കങ്ങളുടെ സ്ഥാനവിലയ്ക്കനുസരിച്ച് മുത്തുകൾ ക്രമപ്പെടുത്തി സംഖ്യകളുടെ സങ്കലനവും ഗുണനവും അബാക്കസ്സിൽ ചെയ്യാവുന്നതാണ്.

അബാക്കസ്സ് പ്രവർത്തിക്കുന്നത് സംഖ്യകളുടെ സ്ഥാനവില ക്രമത്തിനനുസരിച്ചാണ്. അബാക്കസ്സിൽ കാണിക്കുന്ന സംഖ്യ നമ്മൾ എഴുതിയ സംഖ്യ വായിക്കുന്നതു പോലെ വായിക്കാം. ബാറിന്റെ താഴെഭാഗത്ത് കാണുന്ന അഞ്ച് മുത്തുകളുടെ ഓരോന്നിന്റെയും വില 1 ഉം മുകളിൽ കാണുന്ന 2 മുത്തുകളുടെ ഓരോന്നിന്റെയും വില 5 ഉം ആണ്. അബാക്കസ്സിലെ ബാറിനരികിലേക്ക് തള്ളി നീക്കിയ മുത്തുകൾ സംഖ്യയെ പ്രതിനിധികരിക്കുന്നു. ചിത്രം 1.1 ൽ അടയാളപ്പെടുത്തിയ സംഖ്യ 2364.

ഇന്നും അബാക്കസ് കുട്ടികൾ കണക്ക് കൂട്ടുന്നതിനുവേണ്ടി ഉപയോഗിക്കുന്നു. നന്നായ് പരിശീലിച്ചാൽ ഒരു അബാക്കസ്സിൽ കാൽക്കുലേറ്ററിന്റെ വേഗതയിൽ കണക്കു കൂട്ടാനാകും. ചിത്രം 1.2 (a) കാണിക്കുന്നത് ഒറ്റ അക്കമുള്ള രണ്ട് സംഖ്യകളുടെ സങ്കലനമാണ് ചിത്രം 1.2 (b) കാണിക്കുന്നത് രണ്ട് സംഖ്യകൾ (54 ഉം 46 ഉം) പരസ്പരം കൂട്ടുന്നതാണ്.




*ചിത്രം 1.2 (b) അബാക്കസ് ഉപയോഗിച്ചുള്ള കണക്കുകൂട്ടൽ*

## b. നാപിയറിന്റെ ബോണുകൾ (Napier's bones)

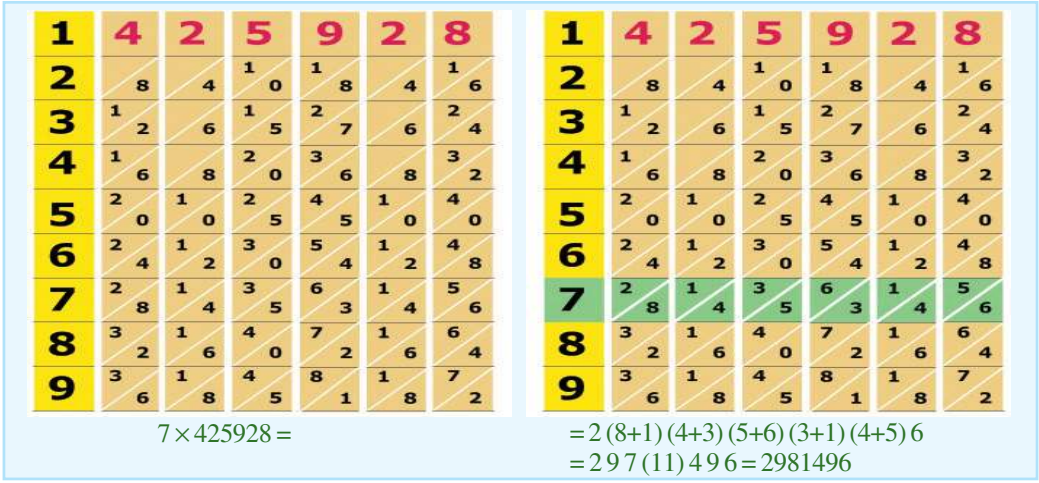
AD 1617 ൽ ജോൺ നാപ്പിയർ എന്ന ഗണിത ശാസ്ത്രജ്ഞൻ സംഖ്യകൾ രേഖപ്പെടുത്തിയ ചില ദണ്ഡുകൾ കണ്ടുപിടിച്ചു. ഈ ദണ്ഡുകൾ നാപ്പിയർ ബോൺസ് എന്നറിയപ്പെട്ടു. ഇതുപയോഗിച്ച് ഗുണനക്രിയകൾ എളുപ്പത്തിൽ ചെയ്യാൻ സാധിച്ചിരുന്നു. ഏതു സംഖ്യയെയും 2 മുതൽ 9 വരെയുള്ള അക്കങ്ങൾ കൊണ്ട് ഗുണിക്കാൻ ഈ ഉപകരണത്തിന് കഴിയുമായിരുന്നു. ഒരു ഗുണനക്രിയയിലെ ആദ്യസംഖ്യയെ പ്രതിനിധീകരിക്കാൻ 0 മുതൽ 9 വരെയുള്ള ബോണുകളും രണ്ടാമത്തെ അക്കത്തെ സൂചിപ്പിക്കാൻ പതിനൊന്നാമത്തെ ബോണും ഉപയോഗിച്ചു.

| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| 3 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 |
| 4 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| 5 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| 6 | 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 |
| 7 | 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 |
| 8 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
| 9 | 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 |

*ചിത്രം 1.3 ജോൺനാപ്പിയറും (1550 - 1671) നാപ്പിയർ ബോണുകളും*

പ്രയാസമുള്ള ഗുണനക്രിയകൾ സങ്കലനത്തിലൂടെ എളുപ്പമാക്കാൻ സഹായിക്കുന്ന ലോഗരിത പട്ടിക 1614 ൽ കണ്ടെത്തിയതും ജോൺ നാപ്പിയറാണ്. നാപ്പിയർ ബോണുകൾ ചിത്രം 1.3 ൽ കാണിച്ചിരിക്കുന്നു.

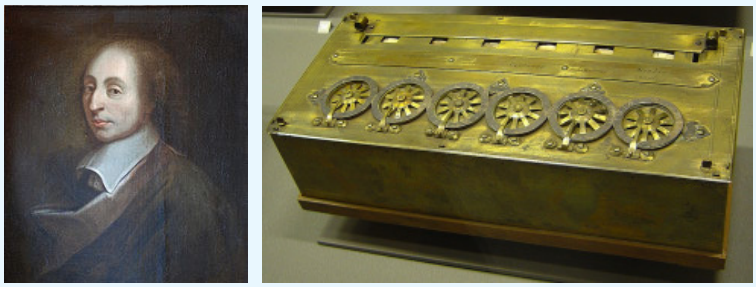
നാപ്പിയർ ബോണിന്റെ സ്ട്രിപ്പുകൾ തവണകളുടെ പട്ടികയാണ്. 2x സംഖ്യ, 3x സംഖ്യ തുടങ്ങിയ വിലകളാണ് ഓരോ ചതുരത്തിലും നൽകുന്നത്. പക്ഷെ പത്തുകളും ഒന്നുകളും എഴുതിയിരിക്കുന്നത് യഥാക്രമം ചരിഞ്ഞവരയുടെ മുകളിലും താഴെയുമാണ്. ഒരു വലിയ സംഖ്യയെ ഒറ്റ അക്കം കൊണ്ട് ഗുണിക്കാൻ നാപ്പിയർ ബോൺ അനുയോജ്യമാണ്. ഉദാഹരണമായി 425928 നെ 7 കൊണ്ട് ഗുണിക്കണമെന്നിരിക്കട്ടെ, ആദ്യം 4,2,5,9,2,8 എന്നീ അക്കങ്ങളുടെ സ്ട്രിപ്പുകൾ യഥാക്രമം അടുക്കുക എന്നിട്ട് 7ന് നേരെയുള്ള ചതുരങ്ങൾ ശ്രദ്ധിക്കുക (ചിത്രത്തിൽ പച്ചക്കളർ നൽകിയിരിക്കുന്നു) ഇനി അക്കങ്ങൾ വായിക്കുക. ചരിഞ്ഞ രേഖയിൽ ഉള്ള അക്കങ്ങൾ കൂട്ടിക്കാണിക്കണം. അപ്പോൾ ഉത്തരം 2(8+1) (4+3) (5+6) (3+1) (4+5)6. അതായത് 297 (11) 496. 11 ഒഴികെയുള്ള എല്ലാ അക്കങ്ങളും കൃത്യ സ്ഥാനത്താണ്. 11 ലെ 10 ഇടത്തോട്ട് ക്യാരി (carry) ചെയ്യുക. അപ്പോൾ 29 (7+1) 1496 എന്നാകുന്നു. അങ്ങനെ ഗുണനഫലമായ 2981496 എന്ന സംഖ്യ ലഭിക്കുന്നു.




*ചിത്രം 1.4 നാപ്പിയർ ബോൺ ഉപയോഗിച്ചുള്ള ഗുണനം*

## c. പാസ്കലൈൻ (Pascaline)

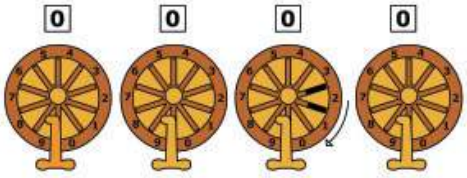
ബ്ലെയ്സി പാസ്കൽ എന്ന ഫ്രഞ്ച് ഗണിത ശാസ്ത്രജ്ഞനാണ് ആദ്യമായി കാൽക്കുലേറ്റർ നിർമ്മിച്ചത്. 1642 ൽ തന്റെ 19-ാമത്തെ വയസ്സിലാണ് അദ്ദേഹം കമ്പ്യൂട്ടിങ് യന്ത്രം നിർമ്മിച്ചത്. ആ യന്ത്രത്തിന് രണ്ട് സംഖ്യകൾ കൂട്ടാനും കുറയ്ക്കാനും കഴിയുമായിരുന്നു. കൂടാതെ ആവർത്തന പ്രക്രിയയിലൂടെ ഗുണനവും ഹരണവും ചെയ്യാൻ ആ യന്ത്രത്തിനു കഴിയുമായിരുന്നു. ടാക്സ് കളക്ഷൻ സൂപ്പർവൈസസറായ തന്റെ പിതാവിനെ സഹായിക്കുന്നതിനാണ് പാസ്കൽ ഈ യന്ത്രം കണ്ടുപിടിച്ചത്. ഈ യന്ത്രത്തിൽ അനേകം ചക്രങ്ങളും ഗിയറുകളും സിലിണ്ടറുകളും ഉപയോഗിച്ചിരുന്നു. ഈ യന്ത്രത്തിനെ പാസ്കലൈൻ എന്ന് വിളിച്ചു. (ചിത്രം 1.5)



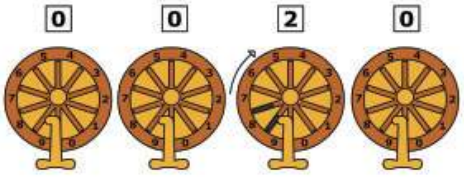
*(ചിത്രം 1.5) ബ്ലെയ്സി പാസ്ക്കൽ (1623 - 1662) ഉം പാസ്കലൈനും*

20,81 എന്നീ സംഖ്യകൾ പാസ്കലൈൻ ഉപയോഗിച്ച് കൂട്ടുന്നതിന്റെ ഒരു ഉദാഹരണം നമുക്ക് പരിശോധിക്കാം. 6 അക്കങ്ങളെ പ്രതിനിധാനം ചെയ്യുന്ന 0-6 ചക്രങ്ങളാണ് ശരിക്കും പാസ്കലൈനിൽ ഉണ്ടാവുക. ചിത്രത്തിൽ - (ചിത്രം 1.6) - 4 അക്കങ്ങൾ പ്രതിനിധീകരിക്കുന്ന 4 ചക്രങ്ങളാണ് ഉള്ളത്, വലത്ത് നിന്ന് ഇടത്തോട്ട് ചക്രം - 1, ചക്രം - 2, ചക്രം - 3, ചക്രം - 4 എന്നി

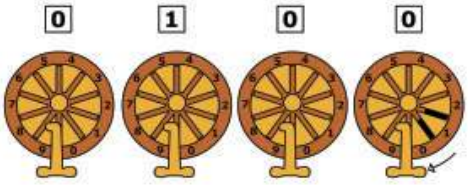
*Pascaline wheel*



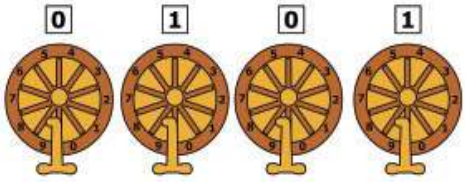

*1. Dialing the digit 2 of 20*




*2. Dialing the digit 8 of 81*




*3. Dialing the digit 1 of 81*




*4. Sum of 20 and 81, 101 on Pascaline*

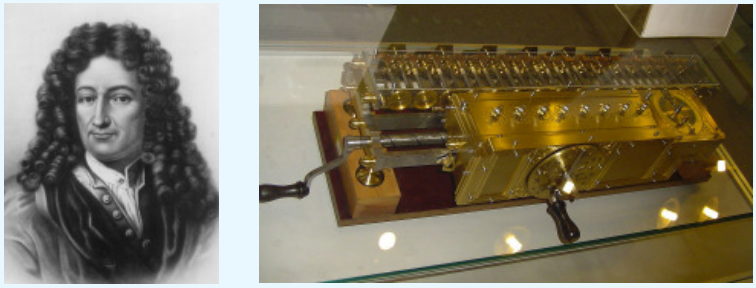
*ചിത്രം 1.6. പാസ്കലൈൻ ഉപയോഗിച്ചുള്ള സങ്കലനം*

ങ്ങനെ ഇവയ്ക്ക് യഥാക്രമം പേര് നൽകാം. 20 ഡയൽ ചെയ്യാൻ ആദ്യം നിങ്ങൾ 2 നു നേരെയുള്ള വിടവിൽ വിരൽ അമർത്തി ചക്രം 2 താഴെത്തെ സ്റ്റോപ്പറിൽ മുട്ടുംവരെ വലത്തോട്ട് തിരിക്കുക. ഈ കറക്കം 2 നെ യന്ത്രത്തിൽ രേഖപ്പെടുത്തുന്നു. ഇപ്പോൾ യന്ത്രത്തിൽ 0020 എന്ന സംഖ്യ തെളിയുന്നു.

81 രേഖപ്പെടുത്താൻ മുമ്പ് ചെയ്തത് പോലെ ആദ്യം ചക്രം-2ലെ 8ന് നേരെയുള്ള വിടവിൽ വിരലമർത്തി താഴെ സ്റ്റോപ്പറിൽ മുട്ടുംവരെ തിരിക്കുക. ചക്രം-2, 9 എന്ന സംഖ്യ കടക്കുമ്പോൾ പാസ്കലൈനിനുള്ളിലെ ഗിയർ ബാക്കി ഒന്നിനെ ചക്രം - 3ൽ രേഖപ്പെടുത്തുകയും യന്ത്രത്തിലെ സംഖ്യ 0100 എന്നാവുകയും ചെയ്യും. ഇനി 81 ലെ 1 നെ രേഖപ്പെടുത്താൻ ചക്രം- 1 തിരിക്കുക. ഇപ്പോൾ യന്ത്രത്തിൽ 0101 എന്ന സംഖ്യ തെളിയുന്നു. അതായത് 20+81=101.

## d. ലെബ്നീസിന്റെ കാൽക്കുലേറ്റർ (Leibniz's calculator)

1673ൽ ജർമ്മൻ ഗണിത ശാസ്ത്രജ്ഞനും തത്വചിന്തകനുമായ ഗോട്ഫ്രൈഡ് വില്യം വൺ ലെബനിസ് സ്റ്റെപ്പ് റക്കണർ എന്ന പേരിൽ ഒരു കണക്കുകൂട്ടൽ യന്ത്രം നിർമ്മിച്ചു. പാസ്കലിന്റെ യന്ത്രത്തിന്റെ മെച്ചപ്പെടുത്തിയ ഒരു രൂപമായിരുന്നു ഇത്. പാസ്കലിന്റെ ആശയം കുറച്ചു കൂടി വികസിപ്പിച്ച് ഗുണനത്തിനും ഹരണത്തിനും ഉപയോഗിക്കാവുന്ന തരത്തിലായിരുന്നു ഈ

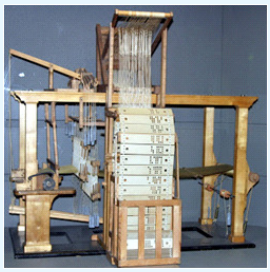


*ചിത്രം 1.7. ഗോട്ഫൈഡ് വില്യംവോൺ ലെബനീസും (1646 - 1716) ലെബനീസ് കാൽക്കുലേറ്ററും*

യന്ത്രം തയാറാക്കിയത്. പിന്നീട് ലെബനീസ് ഈ യന്ത്രം വിജയകരമായി വിപണിയിലിറക്കി. അദ്ദേഹം പ്രത്യേകമായി ഉപയോഗിച്ച ഡ്രം ആകൃതിയിലുള്ള ഗിയറുകൾ പിന്നീട് പല കണക്കുകൂട്ടൽ യന്ത്രങ്ങളുടേയും നിർമ്മാണത്തിന് അടിസ്ഥാനമായി.

## e. ജാക്വേഡിന്റെ തറി (Jacquard's loom)

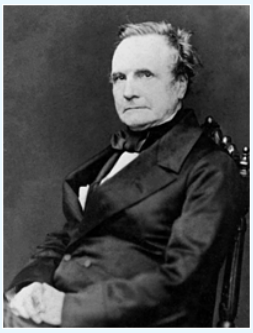
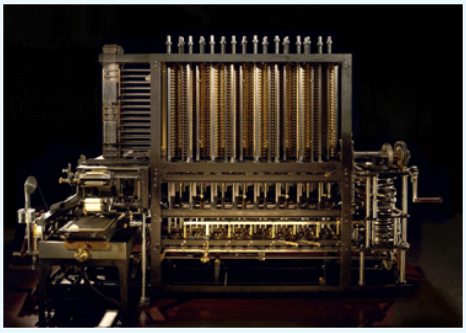
1801 ൽ ജോസഫ് മേരി ജാക്വേഡ് ഒരു യന്ത്രത്തറി നിർമ്മിച്ചു. സങ്കീർണമായ ഡിസൈനുകൾ ഉപയോഗിക്കുന്ന ടെക്സ്റ്റൈലുകളുടെ പ്രവർത്തനങ്ങൾ ലളിതമാക്കാൻ ഈ യന്ത്രത്തിന് കഴിഞ്ഞു. ഈ യന്ത്രത്തെ നിയന്ത്രിച്ചത് സുഷിരങ്ങളിട്ടിട്ടുള്ള കാർഡുകളായിരുന്നു (പഞ്ച്ഡ് കാർഡ്). ഈ കാർഡിലെ ഒരു വരി സുഷിരങ്ങൾ, ഒരുവരി ഡിസൈനിനെ പ്രതിനിധീകരിക്കുന്നു. ഇങ്ങനെയുള്ള പലവരികൾ ചേർന്നതായിരുന്നു ഒരു പഞ്ച്ഡ് കാർഡ്. ഒരേ തരത്തിലുള്ള കാർഡുകൾ ക്രമത്തിൽ അടുക്കി ചില പാറ്റേണുകൾക്ക് രൂപം നൽകിയിരുന്നു. പാറ്റേണുകൾ ഉപയോഗിച്ച് ആവർത്തിച്ചുള്ള ടെക്സ്റ്റൈൽ നിർമ്മാണം നടത്തിയിരുന്നു. വിവരശേഖരണത്തിന്റെയും, പുനരുപയോഗത്തിന്റെയും ആദ്യരൂപമായി ഇത് പരിഗണിച്ചിരുന്നു. ഇതിലെ പഞ്ച്ഡ് കാർഡ് എന്ന ആശയം ചാൾസ് ബാബേജ് തന്റെ അനെലറ്റിക് എഞ്ചിനിലും പിന്നീട് ഹോളറിത്ത് തന്റെ കണ്ടുപിടിത്തങ്ങളിലും ഉപയോഗിച്ചു.

*ചിത്രം 1.8 ജോസഫ് മേരി ജാക്വേഡും (1752 - 1834) ജാക്വേഡിന്റെ തറിയും*

## f. ഡിഫറൻസ് എഞ്ചിൻ (Difference engine)

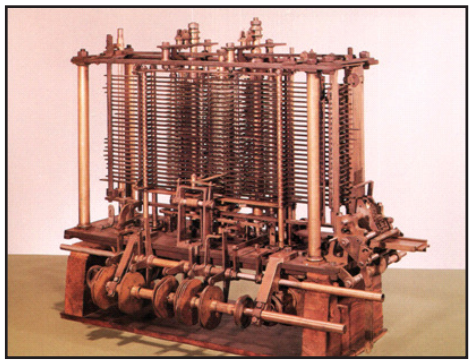
കമ്പ്യൂട്ടറിന്റെ കണ്ടുപിടിത്തത്തിലേക്കുള്ള ആദ്യ ചുവടുവയ്പ്പ് നടത്തിയ ഗണിതശാസ്ത്രജ്ഞനാണ് ചാൾസ് ബാബേജ്. കണക്കുകൂട്ടൽ യന്ത്രങ്ങളിൽ മനുഷ്യന്റെ ഇടപെടൽ കഴിയുന്നത്ര ഒഴിവാക്കുക എന്നതായിരുന്നു ബാബേജിന്റെ സ്വപ്നം. വലിയ ഗണിതശാസ്ത്രക്രിയകളെ ചെറിയ ഓപ്പറേഷനുകളായി വിഭജിച്ച് ഒരു ഓട്ടോമാറ്റിക് യന്ത്രത്തിന്റെ തുടർച്ചയായ പ്രവർത്തനത്തിലൂടെ നിർധാരണം ചെയ്യുന്ന രീതിയായിരുന്നു അദ്ദേഹം ആസൂത്രണം ചെയ്തത്. ആദ്യം അദ്ദേഹം തയാറാക്കിയത് ഒരു ഡിഫറൻസ് എഞ്ചിനായിരുന്നു. അതിന് ഗണിതക്രിയകൾ ചെയ്യാനും ഫലം പ്രദർശിപ്പിക്കാനുമുള്ള കഴിവുണ്ടായിരുന്നു. ഗണിത പട്ടികകൾ സമാഹരിക്കാൻ 1822ലാണ് ബാബേജ് ഡിഫറൻസ് എഞ്ചിൻ തയ്യാറാക്കിയത്. ഈ യന്ത്രത്തിന്റെ നിർമ്മാണത്തോടുകൂടി ഏതുതരം കണക്കു കൂട്ടലും ചെയ്യാൻ ശേഷിയുള്ള ഒരു പുതിയ യന്ത്രത്തിന്റെ ആശയം ബാബേജിന്റെ മനസ്സിൽ രൂപീകൃതമായി.

*ചിത്രം 1.9 ചാൾസ് ബാബേജും (1791 - 1871) ഡിഫറൻസ് എഞ്ചിനും*

## g. അനലിറ്റിക്കൽ എഞ്ചിൻ (Analytical engine)

ആധുനിക കമ്പ്യൂട്ടറിന്റെ പൂർവകാല രൂപമായ അനലിറ്റിക്കൽ എഞ്ചിൻ തയാറാക്കുന്ന പ്രവർത്തനം 1833ലാണ് ചാൾസ് ബാബേജ് തുടങ്ങിയത്. സാധാരണ ഗണിതക്രിയക്രിയകൾ പരിഹരിക്കുന്നതിൽ നിന്നും പൊതു ആവശ്യത്തിനുവേണ്ടിയുള്ള കണക്കു കൂട്ടലുകൾ നടത്തുന്നതിലേക്കുള്ള കമ്പ്യൂട്ടിംഗ് യന്ത്രങ്ങളുടെ വളർച്ചയെ സൂചിപ്പിക്കുന്നതായിരുന്നു ഇത്. ഈ യന്ത്രത്തിൽ ഇന്നത്തെ ഡിജിറ്റൽ കമ്പ്യൂട്ടറിലെ പല ഘടകങ്ങളും ഒത്തുചേർന്നിരുന്നു. ഇതിൽ

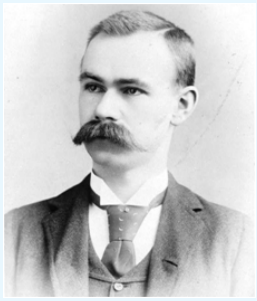


*ചിത്രം 1.10 അനലറ്റിക്കൽ എഞ്ചിന്റെ മാതൃക*

സംഖ്യകളും പ്രവർത്തന ഫലങ്ങളും സൂക്ഷിക്കുന്നതിനുള്ള സൗകര്യമുണ്ടായിരുന്നു. മിൽ (Mill) എന്നറിയപ്പെടുന്ന പ്രോസസ്സറാണ് ഇതിൽ ഗണിതക്രിയകൾ ചെയ്തിരുന്നത്. ഇതിനകത്തെ ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഉപാധികൾ നിർദേശങ്ങളടങ്ങിയ പഞ്ച്ഡ് കാർഡ് രൂപത്തിലായിരുന്നു. നിർദേശങ്ങൾ തയാറാക്കിയത് ബാബേജിന്റെ അസിസ്റ്റന്റായിരുന്ന അഗസ്റ്റ അഡ കിംഗ് (Agust Ada King) ആയിരുന്നു. അങ്ങനെ അഗസ്റ്റ അഡ കിംഗ് ലോകത്തെ ആദ്യത്തെ പ്രോഗ്രമറായി അറിയപ്പെട്ടു. അന്നത്തെ സാങ്കേതികവിദ്യയുടെ കുറവ് മൂലം അനലറ്റിക്കൽ എഞ്ചിൻ പൂർണമായി നിർമ്മിക്കപ്പെട്ടിരുന്നില്ല. ഒരു മാതൃകയും കമ്പ്യൂട്ടർ നിർമ്മാണത്തിനുള്ള എല്ലാ തത്വങ്ങളും ബാബേജ് തയാറാക്കി. ഡിഫറൻസ് എഞ്ചിന്റെയും അനലറ്റിക്കൽ എഞ്ചിന്റെയും കണ്ടുപിടിത്തം ചാൾസ് ബാബേജിന് ''കമ്പ്യൂട്ടറിന്റെ പിതാവ്'' (Father of Computer) എന്ന ബഹുമതി നേടിക്കൊടുത്തു.

## h. ഹോളറിത്തിന്റെ യന്ത്രം (Hollerith's machine)

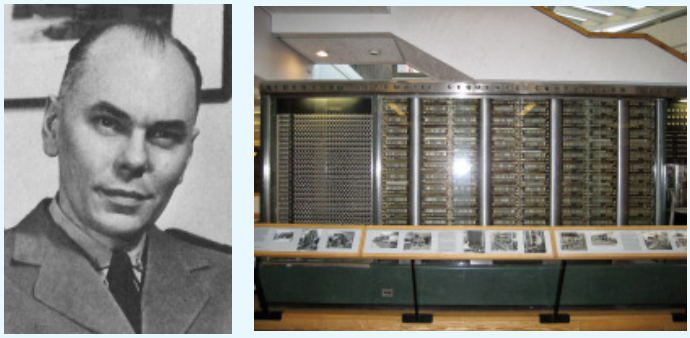
1887 ൽ അമേരിക്കക്കാരനായ ഹെർമ്മൻ ഹോളറിത്ത് (Herman Hollarith) ഒരു ഇലക്ട്രോ മെക്കാനിക്കൽ പഞ്ച്ഡ് കാർഡ് ടാബുലേറ്റർ നിർമ്മിച്ചു. ഈ യന്ത്രത്തിൽ നിർദേശങ്ങൾ നൽകാനും ഇൻപൂട്ട്, ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങൾക്കും പഞ്ച്ഡ് കാർഡാണ് ഉപയോഗിച്ചിരുന്നത്. കാർഡിൽ സുഷിരങ്ങൾ ഒരു പ്രത്യേക മാതൃകയിൽ നൽകി വിവിധ തരം ഡാറ്റയ്ക്ക് രൂപം നൽകി. 1880 ൽ അമേരിക്കൻ സെൻസസ് ബ്യൂറോ (US Census Bureau) യ്ക്ക് ധാരാളം ഡാറ്റ പട്ടികപ്പെടുത്തുവാനും വിശകലനം ചെയ്യാനുമുണ്ടായിരുന്നു. മനുഷ്യൻ ചെയ്യുകയാണെങ്കിൽ 10 വർഷത്തോളം എടുക്കുന്ന ഈ ജോലി ഒരു വർഷത്തിനുള്ളിൽ പൂർത്തിയാക്കാൻ ഹോളറിത്ത് യന്ത്രത്തിനു കഴിഞ്ഞു. ഈ യന്ത്രത്തിന്റെ ഏറ്റവും വലിയ സവിശേഷത എന്തെന്നാൽ അത് വൈദ്യുതി ഉപയോഗിച്ച് പഞ്ച്ഡ് കാർഡിനെ വായിക്കുകയും, എണ്ണുകയും, തരംതിരിക്കുകയും ചെയ്തു എന്നുള്ളതാണ്. 1896 ൽ ഹോളറിത്ത് ടാബുലേറ്റിംഗ് മെഷീൻ കോർപ്പറേഷൻ എന്ന കമ്പനിക്ക് രൂപം കൊടുത്തു. തുടർച്ചയായ ലയനങ്ങൾക്കു ശേഷം 1924 ൽ ഈ കമ്പനി IBM (ഇന്റർ നാഷണൽ ബിസിനസ് മെഷീൻ) ആയി മാറുകയും ചെയ്തു.



*ചിത്രം 1.11 ഹെർമൻ ഹോളറിത്ത് (1860 - 1929) ഹോളറിത്തിന്റെ സെൻസസ് യന്ത്രവും*

### i. മാർക്ക് - I (Mark - I)

1944 ൽ ഹോവാഡ് ഐക്കൻ (Howard Aiken) IBM കമ്പനിയിലെ എഞ്ചിനീയർമാരുമായി ചേർന്ന് ഒരു വലിയ ഇലക്ട്രോ മെക്കാനിക്കൽ കമ്പ്യൂട്ടർ നിർമ്മിച്ചു. ഈ യന്ത്രം ഹാർവാഡ് മാർക്ക്- I എന്ന് അറിയപ്പെട്ടു. ഈ യന്ത്രത്തിന്റെ പ്രവർത്തനം ബാബേജിന്റെ അനലിറ്റിക്കൽ എഞ്ചിന്റെ ചുവടുപിടിച്ചായിരുന്നു. 23 സ്ഥാനങ്ങളുള്ള സംഖ്യകളുടെ നാലുവിധത്തിലുള്ള ഗണിത ക്രിയകൾ ചെയ്യാൻ ഈ യന്ത്രത്തിന് കഴിവുണ്ടായിരുന്നു. ലോഗരിതവും, ത്രികോണമിതി ക്രിയകളും ചെയ്യാവുന്ന തരത്തിൽ ഈ യന്ത്രത്തെ പ്രോഗ്രാം ചെയ്തിരുന്നു. മാർക്ക്- I ഉപയോഗിച്ച് രണ്ട് സംഖ്യകൾ കൂട്ടാൻ ഏകദേശം 3 മുതൽ 6 സെക്കണ്ട് വരെ സമയം എടുത്തിരുന്നു. ഇൻപുട്ട്, ഔട്ട്പുട്ട് എന്നീ പ്രവർത്തനങ്ങൾക്കു വേണ്ടി ഈ യന്ത്രം പേപ്പർടേപ്പ് റീഡർ, കാർഡ് റീഡർ, കാർഡ് പഞ്ച്, ടൈപ്പ്റൈറ്റർ എന്നിവ ഉപയോഗിച്ചു.



*ചിത്രം 1.12 ഹോവാഡ് ഐക്കനും (1900 - 1973) മാർക്ക് - I കമ്പ്യൂട്ടറും*

**നിങ്ങൾക്ക് പരിശോധിക്കാം**

1. സുമേറിയൻ സംഖ്യാ സമ്പ്രദായത്തിന്റെ മറ്റൊരു പേരാണ് -------
2. ഹിന്ദു-അറബിക് സംഖ്യാ സമ്പ്രദായത്തിന്റെ പ്രത്യേകതകൾ എന്തെല്ലാം?
3. ബാബിലോണിയൻ സംഖ്യാ സമ്പ്രദായത്തിൽ പൂജ്യം എങ്ങനെയായിരുന്നു രേഖപ്പെടുത്തിയിരുന്നത്.
4. ആരാണ് ലോകത്തിലെ ആദ്യത്തെ പ്രോഗ്രാമർ.
5. ബ്ലെയ്സി പാസ്കൽ നിർമ്മിച്ച കമ്പ്യൂട്ടിംഗ് യന്ത്രം ---------- എന്ന് അറിയപ്പെട്ടു.

## 1.2. കമ്പ്യൂട്ടറിന്റെ തലമുറകൾ (Generations of computers)

16-ാം നൂറ്റാണ്ടിൽ തുടക്കം കുറിച്ച കമ്പ്യൂട്ടിംഗ് യന്ത്രങ്ങളുടെ ക്രമാനുഗതമായ വികാസമാണ് ഇന്നത്തെ ആധുനിക കമ്പൂട്ടറിൽ എത്തിനിൽക്കുന്നത്. ആദ്യത്തെ കമ്പ്യൂട്ടർ മുതൽ, നിർമ്മിത ബുദ്ധിയുള്ള കമ്പ്യൂട്ടർ വരെ അഞ്ച് തലമുറയായിട്ടാണ് കമ്പ്യൂട്ടറിന്റെ പരിണാമം എഴുതപ്പെട്ടിരിക്കുന്നത്. ഓരോ തലമുറയും അതിന്റെ അടിസ്ഥാനയന്ത്രഘടകങ്ങളിൽ വ്യത്യാസപ്പെട്ടിരിക്കുന്നു. തലമുറകൾ കഴിയുന്തോറും കമ്പ്യൂട്ടർ കൂടുതൽ ചെറുതും, വില കുറഞ്ഞതും കൂടുതൽ പ്രവർത്തനക്ഷമതയുള്ളതുമായി എന്നത് കമ്പ്യൂട്ടറിന്റെ പരിണാമത്തിലെ പ്രത്യേകതയാണ്.

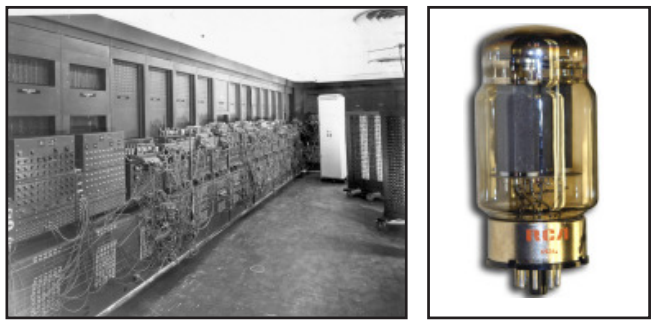
വളർച്ചയുടെ വിവിധ ഘട്ടങ്ങളെ ആസ്പദമാക്കി കമ്പ്യൂട്ടർ പരിണാമത്തിലെ വിവിധ തലമുറ കൾ ചുവടെ ചേർക്കുന്നു.

- ഒന്നാം തലമുറ കമ്പ്യൂട്ടറുകൾ (1940 – 1956)
- രണ്ടാം തലമുറ കമ്പ്യൂട്ടറുകൾ (1956 – 1963)
- മൂന്നാം തലമുറ കമ്പ്യൂട്ടറുകൾ (1964 – 1971)
- നാലാം തലമുറ കമ്പ്യൂട്ടറുകൾ (1971 – ഇന്ന് വരെ)
- അഞ്ചാം തലമുറ കമ്പ്യൂട്ടറുകൾ (നിലവിലുള്ളതും വരാനിരിക്കുന്നതും)

### 1.2.1. ഒന്നാം തലമുറ കമ്പ്യൂട്ടറുകൾ (1940 – 1956) (First generation computers 1940 - 1956)

ഒന്നാം തലമുറ കമ്പ്യൂട്ടറുകൾ നിർമ്മിച്ചത് വാക്വം ട്യൂബുകൾ ഉപയോഗിച്ചായിരുന്നു. ഈ തല മുറയിലാണ് സ്റ്റോർഡ് പ്രോഗ്രാം (Stored Program) എന്ന ആശയത്തിനു തുടക്കം കുറിച്ചത്. ഒരു അടച്ച ട്യൂബിലെ ശൂന്യതയിലൂടെ വൈദ്യുതിയെ നിയന്ത്രിക്കുന്ന ഉപകരണമാണ് വാക്വം ട്യൂബ്. ഈ സിലിണ്ടറാകൃതിയിലുള്ള ട്യൂബ് സുതാര്യമായ ഗ്ലാസ്സ് കൊണ്ടായിരുന്നു നിർമി ച്ചിരുന്നത്. ഇവിടെ ഇൻപുട്ട് നടത്തിയിരുന്നത് പഞ്ച്ഡ് കാർഡും പേപ്പർടേപ്പും ഉപയോഗിച്ചായിരു ന്നു. കൂടാതെ ഔട്ട്പുട്ട് പ്രിന്റൗട്ടുകളായും നൽകിയിരുന്നു.

ആദ്യത്തെ ഇലക്ട്രോണിക് കമ്പ്യൂട്ടറായ ഇലക്ട്രോണിക് ന്യൂമറിക്കൽ ഇന്റഗ്രേറ്റർ ആന്റ് കാൽക്കുലേറ്റർ (ENIAC) ഈ തലമുറയിലാണ് ഉൾപ്പെടുന്നത്. ENIAC നിർമ്മിച്ചത് ജെ.പ്രസ്പർ ഇക്കേർട്ടും (J. Presper Eckert) ജോൺ വി. മോഷ്ലി (John V. Mauchly)യും ചേർന്നാണ്. ഈ യന്ത്രത്തിന് 30–50 ഫീറ്റ് ഉയരവും 30 ടൺ ഭാരവും ഉണ്ടായിരുന്നു. 18000 വാക്വം ട്യൂബുകളും, 70000 റജിസ്റ്ററുകളും, 10000 കപ്പാസിറ്ററുകളും ഈ യന്ത്രത്തിൽ ഉപയോഗിച്ചിരുന്നു. 1,50,000 w വൈദ്യുതി ഈ യന്ത്രം ഉപയോഗിച്ചിരുന്നു. ആദ്യതലമുറ കമ്പ്യൂട്ടറുകൾ വളരെ വലുപ്പമുള്ളവ



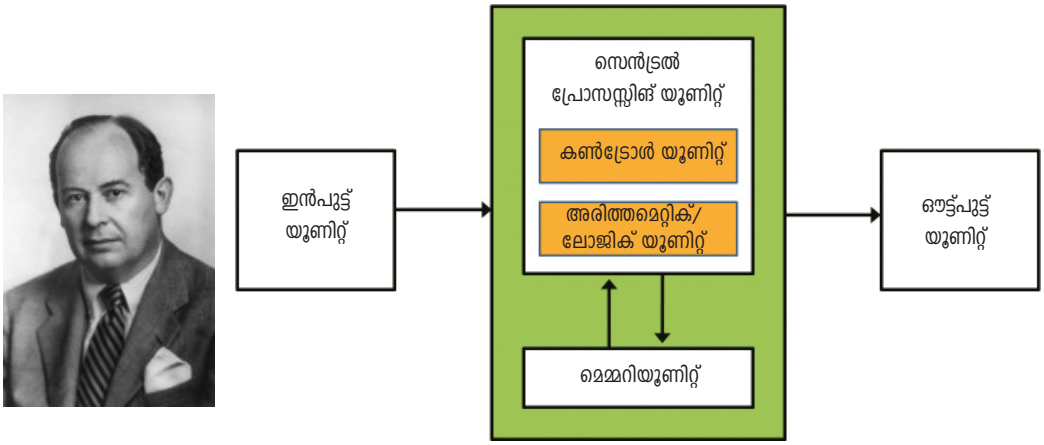
*ചിത്രം 1.13: - ENIAC ഉം വാക്വം ട്യൂബും*

യായിരുന്നു. ഇവ സ്ഥാപിക്കാൻ വലിയ മുറികളാവശ്യമായിരുന്നു . ഉയർന്ന താപം പുറത്തു വിടുന്നത് കാരണം എയർകണ്ടീഷൻ ഇതിന്റെ ശരിയായ പ്രവർത്തനത്തിന് ആവശ്യമായിരുന്നു.

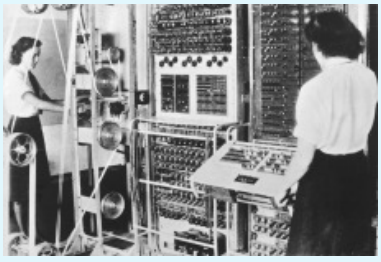
ENIAC നിർമാണം പൂർത്തീകരിക്കുന്നതിന് മുമ്പ്, വോൺ ന്യൂമാൻ (Von Newman) എന്ന ശാസ്ത്ര ജ്ഞൻ ഇലക്ട്രോണിക് ഡിസ്ക്രീറ്റ് വേരിയബിൾ ഓട്ടോമാറ്റിക് കമ്പ്യൂട്ടർ (EDVAC) എന്ന മറ്റൊരു യന്ത്രം നിർമിച്ചു. ഈ യന്ത്രത്തിൽ ഡാറ്റയും പ്രോഗ്രാമും ശേഖരിച്ചുവയ്ക്കാനുള്ള മെമ്മറി ഉണ്ടായിരുന്നു. പിന്നീട് 1952 ൽ ഇക്കേർട്ട് (Eckert) ഉം മോഷ്ലി (Mauchly) യും ചേർന്ന് നിർമിച്ച യൂണിവേഴ്സൽ ഓട്ടോമാറ്റിക് കമ്പ്യൂട്ടർ (UNIVAC) കമ്പ്യൂട്ടർ ചരിത്രത്തിലെ ഒരു വ്യാവ സായിക വിജയമായിരുന്നു.

## വോൺന്യൂമാന്റെ രൂപഘടന (Von Neumann architecture)

ജോൺ വോൺ ന്യൂമൻ എന്ന ഗണിത ശാസ്ത്രജ്ഞനാണ് ഇന്ന് ഉപയോഗത്തിലുള്ള കമ്പ്യൂട്ടറിന്റെ രൂപഘടന ആദ്യമായി നിർമ്മിച്ചത്. ഈ മാതൃക വോൺന്യൂമാന്റെ രൂപഘടന (Architechture) എന്ന പേരിൽ പ്രസിദ്ധമായി. അരിത്തമറ്റിക് ലോജിക് യൂണിറ്റും (ALU) കൺട്രോൾ യൂണിറ്റും (CU) അടങ്ങിയ ഒരു സെന്ട്രൽ പ്രോസ്സിംഗ് യൂണിറ്റ് (CPU), ഇൻപുട്ട് യൂണിറ്റ്, ഔട്ട്പുട്ട് യൂണിറ്റ്, ഡാറ്റയും നിർദേശവും ശേഖരിക്കുന്നതിനുള്ള സ്റ്റോറേജ് യൂണിറ്റ് എന്നിവ ഉൾപ്പെട്ടതാണ് ഈ മാതൃക. ഇതിൽ 'സ്റ്റോർഡ് പ്രോഗ്രാം കൺസപ്റ്റ്' ഉപയോഗിച്ചിരുന്നു. അതായത് ഡാറ്റയും പ്രോഗ്രാമും മെമ്മറിയിൽ ശേഖരിച്ചശേഷമാണ് പ്രോസസ്സിംഗ് നടന്നിരുന്നത്.



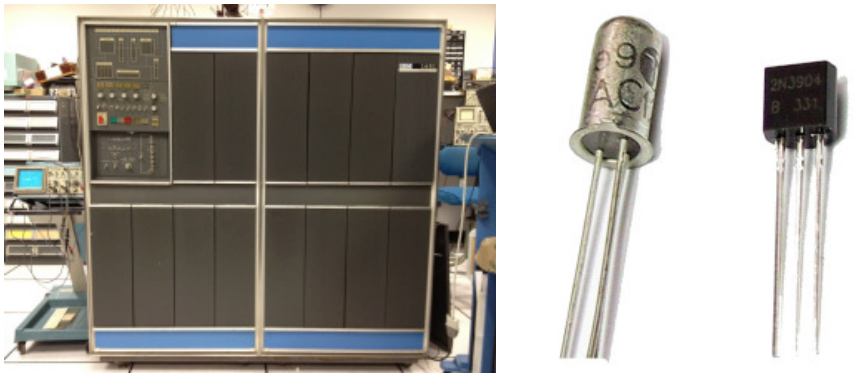

*ചിത്രം 1.14. ജോൺവോൺ ന്യൂമാനും (1903 - 1957) ന്യൂമാന്റെ രൂപഘടനയും*



*1943 ൽ ബ്രിട്ടീഷുകാർ കൊളോസസ്സ് (colossus) എന്ന പേരിൽ, ജർമ്മൻ ഭാഷയിലെ സന്ദേശങ്ങൾ ഡീകോഡ് ചെയ്യാൻ ഒരു രഹസ്യ കോഡ്-ബ്രേക്കിംഗ് കമ്പ്യൂട്ടർ നിർമിച്ചു. ടോമി ഫ്ളവേഴ്സ് (Tomy Flowers) എന്ന എഞ്ചിനീയർ വാക്വംട്യൂബ് ഉപയോഗിച്ചായിരുന്നു ഈ യന്ത്രം നിർമ്മിച്ചിരുന്നത്. കൊളോസസ്സിന് കമ്പ്യൂട്ടറിന്റെ വളർച്ചയിൽ കാര്യമായി സംഭാവന ചെയ്യാൻ കഴിയാത്തതിന് രണ്ട് കാരണങ്ങൾ ചൂണ്ടിക്കാണിക്കപ്പെടുന്നു. ഒന്നാമതായി ഇത് രഹസ്യസന്ദേശങ്ങൾ ഡീകോഡ് ചെയ്യാൻ മാത്രമാണ് ഉപയോഗിച്ചിരുന്നത്. രണ്ടാമതായി 1970 വരെ ഇങ്ങനെ ഒരു യന്ത്രമുണ്ടെന്ന വിവരം രഹസ്യമാക്കി വച്ചിരുന്നു. അതുകൊണ്ടു തന്നെ കൊളോസ്സിന്റെ പ്രശസ്തി ലോകം അറിയപ്പെടാതെ പോകുകയും ഇതിന്റെ രൂപകൽപ്പനയിൽ പങ്കാളികളായിരുന്ന വ്യക്തികളുടെ സംഭാവനകൾ പരിഗണിക്കപ്പെടാതിരിക്കുകയും ചെയ്തു.*

### 1.2.2. രണ്ടാംതലമുറ കമ്പ്യൂട്ടറുകൾ (1956–1963)
### (Second generation computers (1956-1963)

രണ്ടാംതലമുറ കമ്പ്യൂട്ടറുകളിൽ വാക്വം ട്യൂബുകൾക്ക് പകരം ട്രാൻസിസ്റ്റർ ഉപയോഗിച്ചു. 1947 ൽ ബെൽ ലബോറട്ടറിയിൽ വച്ച് ജോൺബർഡീൻ, വാൾട്ടർ ബ്രട്ടൈൻ, വില്യം ഷോക്ക്ലി എന്നിവർ ചേർന്നാണ് ട്രാൻസിസ്റ്റർ നിർമ്മിച്ചത്. ട്രാൻസിസ്റ്റർ ഉപയോഗിച്ചതു കാരണം കമ്പ്യൂട്ടറിന്റെ വലുപ്പം, വില, വൈദ്യുതി ഉപയോഗം, താപപ്രസരണം എന്നിവ കുറയുകയും പ്രവർത്തനശേഷി, വേഗത എന്നിവ വർധിക്കുകയും ചെയ്തു.

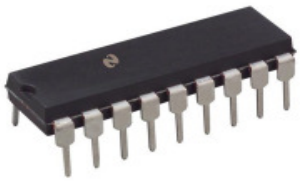


*ചിത്രം 1.15 IBM 1401 ഉം ട്രാൻസിസ്റ്ററുകളും*

രണ്ടാംതലമുറയിലാണ് പ്രോഗ്രാമിങ് ഭാഷ എന്ന ആശയം ഉടലെടുത്തത്. ഈ തലമുറ പ്രൈമറി മെമ്മറിയായി മാഗ്നറ്റിക് കോർ മെമ്മറിയും സെക്കണ്ടറി മെമ്മറിയായി മാഗ്നറ്റിക് ഡിസ്ക് മെമ്മറിയും യഥാക്രമം ഉപയോഗിച്ചു. ഇതേ കാലയളവിൽ ഒന്നും പൂജ്യവും ഉപയോഗിക്കുന്ന മെഷീൻ ലാംഗ്വേജിൽ നിന്ന് സൂചകങ്ങൾ ഉപയോഗിക്കുന്ന അസംബ്ലി ലാംഗ്വേജിലേക്ക് പ്രോഗ്രാം നിർമ്മാണം പുരോഗമിച്ചു. ഇംഗ്ലീഷ് ഭാഷയിലെ വാക്കുകൾക്ക് സമാനമായ നിർദേശങ്ങൾ അടങ്ങിയ ഹൈലെവൽ പ്രോഗ്രാമിങ് ഭാഷകളായ FORTRAN, COBOL തുടങ്ങിയവ ഇക്കാലത്താണ് നിലവിൽ വന്നത്. IBM 1401, IBM 1620 ഉം എന്നിവ ഈ തലമുറയിലെ പ്രചാരമുള്ള കമ്പ്യൂട്ടറുകളായിരുന്നു.

### 1.2.3. മൂന്നാംതലമുറ കമ്പ്യൂട്ടറുകൾ (1964–1971)
### (Third generation computers (1964-1971)

മൂന്നാംതലമുറ കമ്പ്യൂട്ടറുകൾ രൂപത്തിൽ വളരെ ചെറുതായിരുന്നു. കാരണം അവ ഐ.സി. ചിപ്പ് (I C ചിപ്പ്) എന്ന യന്ത്രഘടകമായിരുന്നു ഉപയോഗിച്ചിരുന്നത്. അനേകം ട്രാൻസിസ്റ്ററുകൾ ഉൾക്കൊള്ളിച്ചു നിർമിച്ച സിലിക്കൺ ചിപ്പാണ്, ഐ.സി. ചിപ്പ്. ടെക്സാസ് ഇൻസ്ട്രുമന്റ് സ് എന്ന കമ്പനിയിലെ ജാക് കിൽബി എന്ന എഞ്ചിനീയറാണ് ഐ.സി. ചിപ്പ് നിർമിച്ചത്. ഐ.സി. ചിപ്പ് കമ്പ്യൂട്ടറിന്റെ വലുപ്പം കുറയ്ക്കുകയും, കമ്പ്യൂട്ടറിന്റെ സ്പീഡും കാര്യക്ഷമതയും വർധിപ്പിക്കുകയും ചെയ്തു. പിന്നീട് മൾട്ടി ലെയർ പ്രിന്റഡ് സർക്യൂട്ടുപയോഗിക്കുകയും മാഗ്നറ്റിക് കോർമെമ്മറിക്ക് പകരം കൂടുതൽ വേഗതയും സംഭരണ ശേഷിയുമുള്ള സോളിഡ് സ്റ്റേറ്റ് മെമ്മറികൾ ഉപയോഗിക്കുകയും ചെയ്തു.

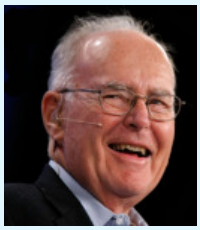


*ചിത്രം 1.16 IBM 360 ഉം ഇന്റഗ്രേറ്റഡ് സർക്യൂട്ടും*

ഈ തലമുറ കമ്പ്യൂട്ടറുകൾ കൂടുതൽ വേഗതയുള്ളവയും വൈദ്യുത ഉപഭോഗം കുറഞ്ഞവയും ചെലവ് കുറഞ്ഞവയുമായിരുന്നു. ഇന്റഗ്രേറ്റഡ് സർക്യൂട്ടുകൾ, കുറച്ചു കൂടി മെച്ചപ്പെട്ട സെക്കന്ററി സ്റ്റോറേജ് ഉപകരണങ്ങൾ, കീബോഡ്, മോണിറ്റർ തുടങ്ങിയ ഇൻപുട്ട്, ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ എന്നിവ ഉപയോഗിക്കപ്പെട്ടു. ഗണിതക്രിയകൾ മൈക്രോ സെക്കന്റിലും, നാനോസെക്കന്റിലും പൂർത്തിയാക്കിയിരുന്നു.

ഈ കമ്പ്യൂട്ടറുകൾ ഒരു പ്രോസസ്സറും മെമ്മറിയും ഉപയോഗിച്ച് ഒരേ സമയത്ത് പല പ്രോഗ്രാമുകൾ പ്രവർത്തിപ്പിച്ചിരുന്നു. പ്രോഗ്രാമിംഗ് ലോകത്തെ എളുപ്പമുള്ള ഭാഷയായ ബേസിക് (BASIC) എന്ന ഹൈലെവൽ ലാംഗ്വേജ് ഈ കാലത്താണ് തയ്യാറാക്കിയത്. കമ്പ്യൂട്ടറിന് വിലയും വലുപ്പവും കുറഞ്ഞതോടെ മുൻകാലത്തെ അപേക്ഷിച്ച് കൂടുതൽ ജനങ്ങൾ കമ്പ്യൂട്ടറുകൾ ഉപയോഗിച്ചു തുടങ്ങി. IBM 360, IBM 370 എന്നിവ ഈ തലമുറയിലെ കമ്പ്യൂട്ടറുകളാണ്.

*മൂറിന്റെ നിയമം പറയുന്നത് "ഐ.സി. ചിപ്പിലെ ട്രാൻസിസ്റ്ററുകളുടെ എണ്ണം ഓരോ രണ്ട് വർഷങ്ങളിലും ഇരട്ടിച്ചു കൊണ്ടിരിക്കും" എന്നാണ്. ഗോൾഡൻ ഇ. മൂർ എന്ന ശാസ്ത്രജ്ഞൻ 1958 മുതൽ 1965 വരെയുള്ള മാറ്റങ്ങൾ മനസ്സിലാക്കി 1965 ൽ ആണ് ഈ പ്രവചനം നടത്തിയത്. പത്തുവർഷം കൂടി ഈ അവസ്ഥ തുടരുമെന്ന് മൂർ പ്രവചിച്ചു. കേവലം ഒരു നിരീക്ഷണമാണെങ്കിലും ഈ നിയമം പിന്നീട് 50 കൊല്ലത്തോളം കൃത്യമായി പാലിക്കപ്പെട്ടു എന്നത് ഒരു അത്ഭുത പ്രതിഭാസമായി നിലനിൽക്കുന്നു.*



## 1.2.4. നാലാം തലമുറ കമ്പ്യൂട്ടറുകൾ (1971 മുതൽ ഇന്നു വരെ)
## (Fourth generation computers (1971 onwards)

ഇന്ന് നാം ഉപയോഗിക്കുന്ന കമ്പ്യൂട്ടറുകൾ ഈ തലമുറയിൽപെടുന്നു. ഈ കമ്പ്യൂട്ടറുകൾ മൈക്രോപ്രോസ്സസർ ഉപയോഗിക്കുന്നതിനാൽ ഇവ മൈക്രോ കമ്പ്യൂട്ടർ എന്നറിയപ്പെടുന്നു. മൈക്രോപ്രോസ്സസർ എന്നാൽ ലാർജ് സ്കെയിൽ ഇന്റഗ്രേഷൻ (LSI) നടത്തിയിട്ടുള്ള ഒരു സിലിക്കൺ ചിപ്പാണ്. ഇതിൽ ആയിരക്കണക്കിന് ട്രാൻസിസ്റ്ററുകൾ, റെസിസ്റ്ററുകൾ, കപ്പാസിറ്ററുകൾ എന്നിവ ഉൾക്കൊള്ളുന്നു. മൈക്രോപ്രോസസ്സർ നിലവിൽ വന്നതോടെ കമ്പ്യൂട്ടറിന്റെ CPU തന്നെ ഒറ്റ സിലിക്കൺ ചിപ്പിലേക്ക് ഉൾക്കൊള്ളിക്കാൻ സാധിച്ചു. ഇതോടെ ഡാറ്റാപ്രോസ്സസിംഗ് ശേഷി മുൻപത്തേതിനേക്കാൾ


*ചിത്രം 1.17. VLSI ചിപ്പ്*

വർധിച്ചു. ഇന്റഗ്രേഷൻ സെക്യിൽ വീണ്ടും വർധിച്ച് VLSI (Very Large Scale Integration) ചിപ്പുകൾ പുറത്തിറങ്ങി. ഈ തലമുറ കമ്പ്യൂട്ടറുകൾ രൂപത്തിൽ വളരെ ചെറുതും പൊതുവേ വേഗത ഏറിയവയും ആയിരുന്നു.

പഴയകാലത്ത് ഒരു മുറി നിറഞ്ഞിരുന്ന കമ്പ്യൂട്ടറുകൾ ഇപ്പോൾ ഒരു കൈവെള്ളയിൽ ഒതുങ്ങി. ഇവ കമ്പ്യൂട്ടർ ശൃംഖലയാൽ പരസ്പരം ബന്ധിപ്പിക്കപ്പെട്ടു. തുടർന്ന് ഇന്റർനെറ്റ് പോലുള്ള വലിയ നെറ്റ്വർക്കുകൾ രൂപപ്പെട്ടു. വില കുറയുകയും ഉപയോക്തൃ സൗഹൃദമാവുകയും ചെയ്തതോടെ കൂടുതൽ ആളുകൾ വ്യക്തിപരമായ ആവശ്യങ്ങൾക്കുവേണ്ടി കമ്പ്യൂട്ടറുകൾ വാങ്ങുവാൻ തുടങ്ങി. IBM PC, APPLE II എന്നിവ ഈ തലമുറയിലെ പ്രധാനപ്പെട്ട കമ്പ്യൂട്ടറുകളാണ്.

### 1.2.5. അഞ്ചാം തലമുറ കമ്പ്യൂട്ടറുകൾ (ഭാവി തലമുറ) (Fifth generation computers (1956-1963)

അഞ്ചാം തലമുറ നിർമ്മിത ബുദ്ധിയെ (Artificial Intelligence) അടിസ്ഥാനമാക്കിയുള്ളതാണ്. നിർമ്മിത ബുദ്ധി എന്നത് മനുഷ്യബുദ്ധി യന്ത്രത്തിൽ സന്നിവേശിപ്പിക്കുന്ന രീതിയാണ്. ഇത്തരം യന്ത്രങ്ങളുടെ നിർമ്മാണം ഇപ്പോൾ പുരോഗമിക്കുകയാണ്. സ്പീച്ച് റക്കഗ്നിഷൻ, ഫേസ് റെക്കഗ്നിഷൻ, റോബോട്ടിക് വിഷൻ തുടങ്ങിയ നൂതന മേഖലകളിലൂടെ നിർമ്മിത ബുദ്ധി എന്ന ആശയം വികസിച്ച് കൊണ്ടേയിരിക്കുന്നു.

മനുഷ്യബുദ്ധി ഉപയോഗിച്ച് സങ്കീർണപ്രശ്നങ്ങൾ നിർധാരണം ചെയ്യുന്ന അതേ രീതിയിൽ പ്രശ്ന പരിഹാരത്തിനുതകുന്ന കമ്പ്യൂട്ടർ പ്രോഗ്രാമുകൾ (Intelligent System) തയാറാക്കുന്ന കമ്പ്യൂട്ടർ സയൻസിലെ ഒരു പഠനശാഖയാണ് നിർമിത ബുദ്ധി. (Atificial Intelligence) ഈ മേഖലയിലെ പ്രധാന പ്രോഗ്രാമിംഗ് ഭാഷകളായി PROLOG, LISP എന്നിവ ഉപയോഗിക്കുന്നു. അഞ്ചാം തലമുറ കമ്പ്യൂട്ടറുകൾ നമ്മുടെ സാധാരണ ഭാഷകൾ കൈകാര്യം ചെയ്യുന്നതിലും മിടുക്ക് കാണിക്കുന്നു. സ്വയം പഠനവും സ്വയം ക്രമീകരണവും ഈ തലമുറ കമ്പ്യൂട്ടറുകളുടെ പ്രത്യേകതകളാണ്.

*ആദ്യത്തെ പ്രോസസ്സറായ ഇന്റൽ 4004 നിർമ്മിച്ചത് 1971 ലാണ്. ഇന്റൽ കോർപ്പറേഷൻ നിർമ്മിച്ച ഈ ചിപ്പിൽ 2300 ട്രാൻസിസ്റ്ററുകൾ ഉപയോഗിച്ചു. ചില പ്രധാന പ്രോസസ്സറുകളും അതിലെ ട്രാൻസിസ്റ്ററുകളുടെ എണ്ണവും ചുവടെ ചേർക്കുന്നു.*

| *പ്രോസസ്സർ* | *ട്രാൻസിസ്റ്ററുകളുടെ എണ്ണം* |
|---|---|
| ഇന്റൽ 8086 | 29,000 |
| മോട്ടറോള 68000 (ആപ്പിൾ കമ്പ്യൂട്ടറിൽ ഉപയോഗിക്കുന്നത്.) | 68,000 |
| ഇന്റൽ പെന്റിയം | 31,00,000 |
| എ.എം.ഡി. കെ-7 | 2,20,00,000 |
| കോർ ഐ - 7 | 73,10,00,000 |

| | തലമുറ | | | | |
|---|---|---|---|---|---|
| | ഒന്ന് | രണ്ട് | മൂന്ന് | നാല് | അഞ്ച് |
| സാങ്കേതിക വിദ്യ | വാക്വം ട്യൂബ് | ട്രാൻസിസ്റ്റർ | ഐ.സി | മൈക്രോ പ്രോസസ്സർ | ആർട്ടിഫിഷ്യൽ ഇന്റലിജൻസ് |
| ഓപ്പറേറ്റിംഗ് സിസ്റ്റം | ഇല്ല | ഇല്ല | ഉണ്ട് | ഉണ്ട് | ഉണ്ട് |
| ഭാഷ | മെഷീൻ | അസംബ്ലി | ഹൈലവൽ | ഹൈലവൽ | ഹൈലവൽ |
| കാലഘട്ടം | 1940-1956 | 1956-1963 | 1964-1971 | 1971 മുതൽ ഇന്ന് വരെ | ഇന്നത്തേതും വരാനിരിക്കുന്നതും |

*പട്ടിക 1.1 അഞ്ചു തലമുറ കമ്പ്യൂട്ടറുകളെ താരതമ്യം ചെയ്യുന്നു.*

**നിങ്ങൾക്ക് പരിശോധിക്കാം**

1. കൊളോസസ് ആരാണ് നിർമ്മിച്ചത്?
2. സ്റ്റോർഡ് പ്രോഗ്രാം കൺസപ്റ്റ് എന്നാൽ എന്ത്?
3. മൂറിന്റെ നിയമം എഴുതുക
4. 'കമ്പ്യൂട്ടറിന് മെഷീൻ ലാംഗ്വേജ് മാത്രമേ തിരിച്ചറിയാൻ കഴിയൂ' - ശരി/തെറ്റ്
5. ഒന്നാം തലമുറ കമ്പ്യൂട്ടറുകളിലെ പ്രധാന ഘടകമാണ് .....

## 1.3. കമ്പ്യൂട്ടിങ്ങിന്റെ പരിണാമം (Evolution of computing)

വിവരങ്ങൾ ശേഖരിക്കാനും, പ്രോസസ്സ് ചെയ്യാനും, ഉത്തരം പ്രദർശിപ്പിക്കാനും കമ്പ്യൂട്ടിങ് യന്ത്രങ്ങൾ ഉപയോഗിച്ചു. ഇതിൽ പ്രോസസിങ് നടന്നിരുന്നത് യന്ത്രത്തിലേക്ക് നൽകുന്ന നിർദ്ദേശങ്ങൾക്ക് അനുസരിച്ചായിരുന്നു. 1940 കളിൽ നിർമിച്ച ആദ്യകാല കമ്പ്യൂട്ടർ ഒരു കാൽക്കുലേറ്ററിനെപ്പോലെ ഒരു കൂട്ടം പ്രവൃത്തികൾ ചെയ്യാൻ മാത്രം പര്യാപ്തമായിരുന്നു. ഈ പ്രത്യേകതരം യന്ത്രങ്ങൾ പ്രോഗ്രാം ചെയ്യപ്പെട്ടത് ഒരു നിര മെക്കാനിക്കൽ സ്വിച്ചുകളോ ജമ്പർ വയർ പ്ലഗ് വഴിയോ ആയിരുന്നു. ബ്രാഞ്ചിംഗ് ലൂപ്പിംഗ് വാചകങ്ങളുടെ പ്രയോഗവും സബ്റൂട്ടീൻ കാളുകളും ഈ യന്ത്രങ്ങൾക്ക് അപ്രാപ്യമായിരുന്നു. എന്നാൽ പിന്നീട് കമ്പ്യൂട്ടറുകൾ ഈ പ്രശ്നം ജോൺവോൺ ന്യൂമാന്റെ സ്റ്റോർഡ് പ്രോഗ്രാം കൺസപ്റ്റ് ഉപയോഗിച്ച് തരണം ചെയ്തു. ഈ ആശയപ്രകാരം ഡാറ്റയും പ്രോഗ്രാമും മെമ്മറിയിൽ ആയിരുന്നു ശേഖരിച്ചിരുന്നത്. കമ്പ്യൂട്ടറിൽ ഇങ്ങനെ ഒരു കൂട്ടം പ്രവൃത്തികൾ ചെയ്യാനുള്ള നിർദ്ദേശങ്ങളുടെ കൂട്ടത്തെ **പ്രോഗ്രാം** എന്ന് വിളിച്ചിരുന്നു.

*ചിത്രം 1.18. അഗസ്റ്റ അഡ ലവ്ലേസ് (1815 - 1852)*

### അഗസ്റ്റ അഡ ലവ്ലേസ്

അഗസ്റ്റ അഡ കിംഗ്: കൗണ്ടസ് ഓഫ് ലവ്ലേസ് സാധാരണയായി അറിയപ്പെട്ടിരുന്നത് അഡ ലവ്ലേസ് എന്നായിരുന്നു. അവർ ഒരു ഇംഗ്ലീഷ് ഗണിത ശാസ്ത്രജ്ഞയായിരുന്നു. ചാൾസ് ബാബേജിന്റെ

ആദ്യത്തെ മെക്കാനിക്കൽ ജനറൽ പർപ്പസ് കമ്പ്യൂട്ടറായ അനലറ്റിക്കൽ എഞ്ചിനിൽ പ്രവർത്തനങ്ങൾ തയാറാക്കിയതാണ് അഡ ലവ്‌ലേസിനെ പ്രശസ്തയാക്കിയത്. അവരുടെ ആദ്യകാല നോട്ടുകൾ ആദ്യത്തെ അൽഗോരിതമായി കണക്കാക്കപ്പെട്ടു. അങ്ങനെ അഗസ്റ്റ അഡ ലവ്‌ലേസ് ലോകത്തിലെ ഒന്നാമത്തെ പ്രോഗ്രാമറായി അറിയപ്പെട്ടു.

### 1.3.1. പ്രോഗ്രാമിങ് ഭാഷകൾ (Proramming language)

പ്രോഗ്രാമിങ് ഭാഷ എന്നാൽ കമ്പ്യൂട്ടറിലേക്ക് നിർദേശങ്ങൾ നൽകാൻ വേണ്ടി തയാറാക്കിയ ഭാഷയാണ്. കമ്പ്യൂട്ടറിനെ നിയന്ത്രിക്കാനുതകുന്നതും അൽഗോരിതത്തെ പ്രതിനിധാനം ചെയ്യുന്നതുമായ പ്രോഗ്രാമുകളെ നിർമ്മിക്കാനുപയോഗിക്കുന്ന ഭാഷയാണ് പ്രോഗ്രാമിംഗ് ഭാഷ.

കമ്പ്യൂട്ടറിൽ ഉപയോഗിച്ച ആദ്യത്തെ പ്രോഗ്രാമിങ് ഭാഷയാണ് മെഷീൻ ലാംഗ്വേജ്. മെഷീൻ ലാംഗ്വേജിൽ ബൈനറി അക്കങ്ങളായ പൂജ്യവും ഒന്നും ചേർന്നുണ്ടാവുന്ന കൂട്ടങ്ങളാണ് ഉപയോഗിക്കുന്നത്. ഈ ഭാഷയുടെ കണ്ടുപിടിത്തം പ്രോഗ്രാമിങ്ങിന്റെ വേഗതയും കാര്യക്ഷമതയും വർധിപ്പിച്ചു. എന്നാൽ മെഷീൻ ലാംഗ്വേജ് പ്രോഗ്രാമിങ്ങിന് പല പരിമിതികളുമുണ്ടായിരുന്നു. ഇതിൽ തെറ്റുകൾ കണ്ടുപിടിക്കാനും അവ തിരുത്താനും പ്രയാസമായിരുന്നു. ഒറ്റവായനയിൽ അർഥം ഗ്രഹിക്കാൻ കഴിയാത്ത തരത്തിൽ വിപുലമായിരുന്നു മെഷീൻ ലാംഗ്വേജ് പ്രോഗ്രാം. മെഷീനുമായി അടുത്ത ബന്ധം പുലർത്തുന്നതിനാൽ കമ്പ്യൂട്ടറിന്റെ ഘടന നന്നായി അറിഞ്ഞാൽ മാത്രമേ ഈ ഭാഷയിൽ പ്രോഗ്രാമിങ് സാധ്യമായിരുന്നുള്ളൂ.

പ്രോഗ്രാമിങ് കുറച്ചുകൂടി എളുപ്പമാക്കാൻ ഒന്നിനും പൂജ്യത്തിനും പകരം കൃത്യമായ അർഥം നിർവചിച്ചിട്ടുള്ള ചെറിയ ഇംഗ്ലീഷ് കോഡുകൾ ഉപയോഗിച്ചിരുന്ന അസംബ്ലി ലാംഗ്വേജാണ് പിന്നീട് നിർമ്മിക്കപ്പെട്ടത്. 1949ൽ നിർമ്മിച്ച ഇലക്ട്രോണിക് ഡിലെ സ്റ്റോറേജ് ഓട്ടോമാറ്റിക് കാൽക്കുലേറ്റർ (EDSAC) എന്ന കമ്പ്യൂട്ടറിൽ ആദ്യമായി അസംബ്ലി ലാംഗ്വേജ് പ്രോഗ്രാമുകൾ ഉപയോഗിച്ചു. പ്രോഗ്രാം എഴുത്ത് എളുപ്പമാക്കിയെങ്കിലും, ഈ ഭാഷയുടെ അമിതമായ ഹാർഡ്‌വെയർ ബന്ധം മൂലം കമ്പ്യൂട്ടർ മാറുമ്പോൾ വീണ്ടുംവീണ്ടും പ്രോഗ്രാമുകൾ മാറ്റേണ്ട സ്ഥിതിവിശേഷം സംജാതമായി. ഈ ഭാഷയിൽ എഴുതിയ പ്രോഗ്രാമുകൾ ഒരു കമ്പ്യൂട്ടറിൽ നിന്ന് മറ്റൊരു കമ്പ്യൂട്ടറിലേക്ക് മാറ്റുവാൻ സാധ്യമായിരുന്നില്ല.

*ചിത്രം 1.19 ഡോ. ഗ്രേസ് ഹോപ്പർ (1906 - 1992)*

ഈ പ്രശ്നം പ്രോഗ്രാമിങ് ഭാഷകളുടെ ലോകത്തിലെ പുതിയ ഭാഷയായ ഹൈലെവൽ ലാംഗ്വേജിന്റെ കണ്ടുപിടിത്തത്തിന് വഴി തെളിച്ചു. ഹൈലെവൽ ലാംഗ്വേജിൽ ഇംഗ്ലീഷിലേതുപോലുള്ള വാക്കുകൾ ഉപയോഗിക്കുകയും, അമിത മെഷീൻ ബന്ധമില്ലാത്ത പ്രോഗ്രാമുകൾ തയാറാക്കുകയും ചെയ്തു. ഹൈലെവൽ ലാംഗ്വേജ് പഠിക്കാനും ഉപയോഗിക്കാനും എളുപ്പമാണ്. കമ്പ്യൂട്ടർ ഹാർഡ്‌വെയർ പരിചയമില്ലാത്തവർക്കും ഈ ഭാഷയിൽ പ്രോഗ്രാമുകൾ തയാറാക്കാം. റിയർ അഡ്മിറൽ ഡോ. ഗ്രേസ്ഹോപ്പർ 1952ൽ തയാറാക്കിയ A-0 പ്രോഗ്രാമിങ് ഭാഷ UNIVAC കമ്പ്യൂട്ടറിന് വേണ്ടി തയാറാക്കിയ ആദ്യത്തെ ഹൈലെവൽ ലാംഗ്വേജ് ആണ്. IBM 370 യ്ക്ക് വേണ്ടി ജോൺ വോക്കസ് തയാറാക്കിയ FORTRAN ഉം, ടീം ഹാർട്ടും മൈക്ക് ലെവിനും MITയിൽ തയ്യാറാക്കിയ LISP എന്ന ഭാഷയും ഹൈലെവൽ ലാംഗ്വേജുകൾക്ക് ചില ഉദാഹരണങ്ങളാണ്.

### 1.3.2. അൽഗോരിതവും കമ്പ്യൂട്ടർ പ്രോഗ്രാമും (Algorithm and computer programs)

പ്രശ്നം നിർധാരണം ചെയ്യാൻ അറിയാമെങ്കിൽ മാത്രമേ ഒരു പ്രോഗ്രാമർക്ക് ശരിയായ പ്രോഗ്രാം എഴുതാൻ കഴിയൂ. അതുകൊണ്ട് പ്രോഗ്രാം എഴുതുന്നതിനു മുമ്പ് തന്നെ പ്രശ്നം നിർധാരണം ചെയ്യാനുള്ള പ്രവർത്തനങ്ങൾ ക്രമീകരിക്കണം. ഇങ്ങനെ പ്രോഗ്രാം പദ്ധതികൾ ക്രമമായി എഴുതുന്നതാണ് അൽഗോരിതം. മറ്റൊരു രീതിയിൽ പറഞ്ഞാൽ ഒരു പ്രശ്നത്തിന്റെ ക്രമീകരിക്കപ്പെട്ട പരിഹാര മാർഗമാണ് അൽഗോരിതം. അൽഗോരിതത്തിലെ പ്രസ്താവനകളാണ് പിന്നീട് ഒരു പ്രോഗ്രാമിങ് ഭാഷയിലെ നിർദേശങ്ങളായി മാറ്റപ്പെടുന്നത്.

### 1.3.3. കമ്പ്യൂട്ടിങ്ങിന്റെ സിദ്ധാന്തം (Theory of computing)

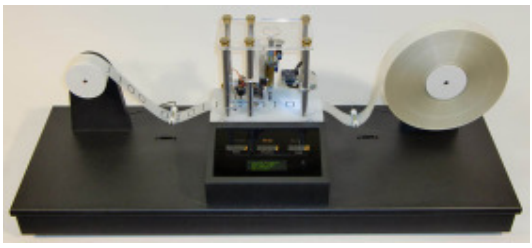
വിവിധ കമ്പ്യൂട്ടേഷൻ മാതൃകകളും അനുബന്ധ അൽഗോരിതങ്ങളും ഉപയോഗിച്ച് ഒരു പ്രശ്നം എങ്ങനെ പരിഹരിക്കപ്പെടുന്നു എന്ന വിഷയം കൈകാര്യം ചെയ്യുന്ന പഠനശാഖയാണ് കമ്പ്യൂട്ടിങ്ങിന്റെ സിദ്ധാന്തം. കമ്പ്യൂട്ടേഷനെക്കുറിച്ച് ആധികാരികമായി പഠിക്കാൻ കമ്പ്യൂട്ടർ ശാസ്ത്രജ്ഞർ ഇതിന്റെ ഗണിതപരമായ മോഡലുകൾ തയാറാക്കിയിരുന്നു. പല മോഡലുകൾ ഉപയോഗത്തിലുണ്ടെങ്കിലും അലൻ ട്യൂറിങ്ങിന്റെ കണ്ടുപിടിത്തമായ ട്യൂറിങ് മെഷീൻ എന്ന മോഡലാണ് ഈ മേഖലയിൽ പൊതുവെ പരീക്ഷിക്കപ്പെട്ടത്.

#### a. അലൻ ട്യൂറിങ്ങിന്റെ സംഭാവന

ചിത്രം 1.20: അലൻ ട്യൂറിംഗ് (1912 - 1954)

അലൻ എം ട്യൂറിങ് എന്ന ബ്രിട്ടീഷ് ഗണിതശാസ്ത്രജ്ഞൻ (1912–1954) കമ്പ്യൂട്ടർ സയൻസിന്റെ വളർച്ചയ്ക്ക് വേണ്ടി അനേകം സംഭാവനകൾ നൽകിയിട്ടുണ്ട്. അദ്ദേഹം ഒരു തർക്കശാസ്ത്ര പണ്ഡിതനും ക്രിപ്റ്റോഗ്രാഫറും കമ്പ്യൂട്ടർ ശാസ്ത്രജ്ഞനുമായിരുന്നു. അൽഗോരിതത്തിന്റെയും കമ്പ്യൂട്ടേഷന്റെയും നൂതന രീതികൾ അദ്ദേഹം തന്റെ കണ്ടുപിടിത്തമായ ട്യൂറിങ് മെഷീനിലൂടെ നിർവഹിച്ചു. ട്യൂറിംഗ് മെഷീൻ ഒരു കമ്പ്യൂട്ടറിന്റെ സൈദ്ധാന്തിക മോഡലായിട്ടാണ് കരുതപ്പെടുന്നത്. 1950 തിൽ അലൻ ട്യൂറിങ് മുമ്പോട്ടു വച്ച ചോദ്യം "യന്ത്രങ്ങൾക്ക് ചിന്തിക്കുവാൻ കഴിയുമോ?" എന്നതായിരുന്നു. പിന്നീട് നടന്ന പ്രവർത്തനങ്ങൾ കമ്പ്യൂട്ടിങ് മെഷീൻ ഇന്റലിജൻസിന്റെ പഠനത്തിന് കൃത്യമായ അടിത്തറ നൽകി. ഒരു ചെറിയ നിരീക്ഷണത്തിലൂടെ യന്ത്രത്തിന്റെ ബുദ്ധി (Machine Intelligence) അളക്കാൻ ടൂറിങ് ശ്രമം നടത്തി. പിൽക്കാലത്ത് ഈ ടെസ്റ്റ് ട്യൂറിങ് ടെസ്റ്റ് എന്നറിയപ്പെട്ടു. അങ്ങനെ അലൻ ട്യൂറിങ് പിന്നീട് കമ്പ്യൂട്ടർ സയൻസിന്റെയും നിർമ്മിത ബുദ്ധിയുടെയും (Artifical Intelligence) പിതാവായി അറിയപ്പെട്ടു.

#### b. ട്യൂറിങ് യന്ത്രം (Turing Machine)



ചിത്രം 1.21. ട്യൂറിംഗ് യന്ത്രം.

1936 ൽ അലൻ ട്യൂറിങ് സംഭാവന ചെയ്ത കമ്പ്യൂട്ടർ മോഡലാണ് ട്യൂറിങ് യന്ത്രം. ഇത് കമ്പ്യൂട്ടറിന്റെ ഉത്തമ മാതൃകയാണ്. ഏതൊരു കമ്പ്യൂട്ടേഷൻ പ്രവർത്തനവും കുറേ സ്റ്റെപ്പുകളിലൂടെ നടപ്പാക്കാമെന്നും, ഓരോ സ്റ്റെപ്പിനും

ഇടയിൽ കിട്ടുന്ന ഡാറ്റയെ ശേഖരിച്ച് അത് വീണ്ടും യന്ത്രത്തിലേക്ക് നൽകി, യന്ത്രത്തിനെ പുതിയ അവസ്ഥയിലേക്ക് എത്തിക്കാമെന്നും, ഈ പ്രവർത്തനം ആവർത്തിച്ച് അവസാന ഫല ത്തിലേക്ക് എത്താമെന്നും അലൻ ട്യൂറിങ് ചിന്തിച്ചു. യഥാർഥത്തിൽ അന്നത്തെ ട്യൂറിങ് യന്ത്ര ത്തിന് തുടർച്ചയായ നിർദേശങ്ങൾക്കനുസരിച്ച് പേപ്പർ ടേപ്പിനു മുകളിൽ ചിഹ്നങ്ങൾ ആലേ ഖനം ചെയ്യാനുള്ള കഴിവാണുണ്ടായിരുന്നത്.

ഈ ട്യൂറിങ് യന്ത്രത്തിൽ അനന്തമായി നീളുന്ന ഒരു പേപ്പർ ടേപ്പ് ഉണ്ട്. ഇത് യഥാർഥത്തിൽ കമ്പ്യൂട്ടർ മെമ്മറിയെ പോലെ പ്രവർത്തിക്കുന്നു. ടേപ്പിലെ സെല്ലുകൾ തുടക്കത്തിൽ ശൂന്യമാ യിരിക്കും. പിന്നീട് ഇതിനുമുകളിൽ നിർദേശമനുസരിച്ച് ഒന്നും പൂജ്യവും ആലേഖനം ചെയ്യ പ്പെടും. അങ്ങിനെ 0,1, ശൂന്യം (blank) എന്നീ മൂന്ന് ചിഹ്നങ്ങൾ രേഖപ്പെടുത്തുന്നതിനാൽ ഈ ട്യൂറിങ് യന്ത്രം 3-സിമ്പൽ ട്യൂറിങ് യന്ത്രം എന്നറിയപ്പെട്ടു. (ചിത്രം 1.22 നോക്കുക). ഓരോ സ്റ്റെപ്പിലും ഹെഡിനടിയിലെ സെല്ലിനകത്തുള്ള ചിഹ്നം റീഡ് ചെയ്യാനും, ചിഹ്നത്തിൽ മാറ്റം വരുത്തി ടാപ്പിനെ ഇടത്തോട്ടോ വലത്തോട്ടോ ഒരു സെൽ നീക്കാനും ഈ യന്ത്രത്തിന് കഴിവു ണ്ട്. അതുകൊണ്ട് ഈ യന്ത്രത്തിന് സമീപ സംഖ്യയെ (neighbouring number) വായിക്കാനും തിരുത്താനും കഴിയും.


*ചിത്രം 1.22: ടേപ്പിനു മുകളിലൂടെയുള്ള ഹെഡിന്റെ നീക്കം*

ഓരോ ഘട്ടത്തിലും ഹെഡ് എന്ത് ചെയ്യണം എന്നതിനുള്ള നിയമത്തിനനുസരിച്ചാണ് ട്യൂറിങ് യന്ത്രത്തിന്റെ പ്രവർത്തനം നിർവചിക്കപ്പെട്ടിരിക്കുന്നത്. ട്യൂറിങ് യന്ത്രത്തിന്റെ പ്രവർത്തനങ്ങൾ നിയന്ത്രിക്കുന്നതിനുള്ള ഘടകങ്ങൾ ഇവയാണ്.

a. യന്ത്രത്തിന്റെ ഇപ്പോഴത്തെ അവസ്ഥ.

b. യന്ത്രം ഇപ്പോൾ റീഡ് ചെയ്യുന്ന ചിഹ്നം

c. യന്ത്രത്തിന്റെ അവസ്ഥാന്തര നിയമങ്ങൾ (ഇത് യന്ത്രത്തിന്റെ പ്രോഗ്രാമാണ്.)

ആധുനിക രീതിയിൽ പറഞ്ഞാൽ, ട്യൂറിങ് യന്ത്രത്തിന്റെ ടേപ്പ് മെമ്മറി ആയും റീഡ്/റൈറ്റ് ഹെഡ് ഈ മെമ്മറിയിൽ ഡാറ്റാ എഴുതുന്ന മെമ്മറി ബസ് ആയും പ്രവർത്തിക്കുന്നു. ടേപ്പിനു മുകളിൽ ആദ്യഘട്ടത്തിൽ എഴുതിയിരിക്കുന്ന ചിഹ്നങ്ങൾ നമുക്ക് ഇൻപുട്ടായി പരിഗണിക്കാം. ട്യൂറിങ് യന്ത്രത്തിന്റെ ഘട്ടങ്ങൾ നമുക്ക് കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തനമായി കാണാം. നൽകുന്ന ഇൻപുട്ടിനനുസരിച്ച് കമ്പ്യൂട്ടർ ഏത് രീതിയിൽ പ്രവർത്തിക്കുമെന്ന്, നൽകിയിരിക്കുന്ന നിയമം പ്രസ്താവിക്കുന്നു. ഇങ്ങനെ പലരീതിയിലും പ്രവർത്തനത്തിൽ ആധുനിക കമ്പ്യൂട്ടറുമായി സാമ്യ മുണ്ടെങ്കിലും ട്യൂറിങ് യന്ത്രം പലകാര്യത്തിലും കമ്പ്യൂട്ടറിൽ നിന്ന് വ്യത്യസ്തമാണ്.

### ട്യൂറിംഗ് പരീക്ഷണം

*'കമ്പ്യൂട്ടിംഗ് യന്ത്രവും ബുദ്ധിശക്തിയും' എന്ന ഗവേഷണ പ്രബന്ധത്തിൽ അലൻ ട്യൂറിംഗ് ഒരു പരീക്ഷണം വിവരിക്കുന്നുണ്ട് ഇത് **ട്യൂറിംഗ് ടെസ്റ്റ്** എന്നറിയപ്പെടുന്നു.*

*ഈ പരീക്ഷണത്തിൽ ഒരു മനുഷ്യനായ ചോദ്യകർത്താവും, മത്സരാർഥികളായി ഒരു കമ്പ്യൂട്ടറും മറ്റൊരു മനുഷ്യനും ആയിരിക്കും ഉണ്ടാകുക. രണ്ട് മത്സരാർഥികളും ചോദ്യകർത്താവും വ്യത്യസ്ത മുറികളിൽ പരസ്പരം കാണാനാവാത്തവിധം ആയിരിക്കുമിരിക്കുന്നത്. ചോദ്യകർത്താവ് ഒരേ ചോദ്യം ഒരേ സമയം രണ്ട് മത്സരാർഥികൾക്കും നൽകുന്നു. ഉത്തരം നൽകുന്നതിന് അനുവദിക്കപ്പെട്ട സമയത്തിന് ശേഷം മത്സരാർഥികളിൽ ഏതാണ് കമ്പ്യൂട്ടർ എന്നും ഏതാണ് മനുഷ്യൻ എന്നും ചോദ്യകർത്താവിന് തിരിച്ചറിയാൻ കഴിഞ്ഞില്ലെങ്കിൽ മത്സരാർഥിയായ കമ്പ്യൂട്ടർ ട്യൂറിംഗ് ടെസ്റ്റിൽ വിജയച്ചതായി കണക്കാക്കുന്നു. ഈ കമ്പ്യൂട്ടറിനെ ബുദ്ധിയുള്ള കമ്പ്യൂട്ടറായി തിരഞ്ഞെടുക്കുന്നു. രണ്ടായിരമാണ്ടിൽ യന്ത്രങ്ങൾ ഈ പരീക്ഷണത്തിൽ വിജയിക്കും എന്ന് ട്യൂറിംഗ് പ്രവചിച്ചു. മനുഷ്യസ്വഭാവ സാദൃശ്യങ്ങൾ പരീക്ഷിക്കുന്നതിന് പല പ്രോഗ്രാമുകൾ തയാറാക്കി പല ടെസ്റ്റുകൾ നടത്തിയെങ്കിലും ഇതുവരെ ഒരു യന്ത്രവും പൂർണമായും ട്യൂറിംഗ് ടെസ്റ്റ് വിജയിച്ചിട്ടില്ല.*


## നമുക്ക് സംഗ്രഹിക്കാം

ഈ അധ്യായത്തിൽ വിവിധ സംഖ്യാസമ്പ്രദായങ്ങളുടെയും എണ്ണലിന്റെയും ആവിർഭാവവും പരിണാമവും നമ്മൾ മനസ്സിലാക്കി. പല കമ്പ്യൂട്ടിംഗ് യന്ത്രങ്ങളുടെ പ്രവർത്തനങ്ങളും നമ്മൾ പരിചയപ്പെട്ടു. ആധുനിക കമ്പ്യൂട്ടറിന്റെ ഘടന മനസ്സിലാക്കി. കമ്പ്യൂട്ടിംഗിന്റെ പരിണാമത്തെക്കുറിച്ച് ചർച്ച ചെയ്തു. വിവിധ പ്രോഗ്രാമിങ് ഭാഷകളെ കുറിച്ചും കമ്പ്യൂട്ടറിന്റെ തലമുറകളെക്കുറിച്ചും, അൽഗോരിതങ്ങളെക്കുറിച്ചും നമ്മൾ വിശദമായി മനസ്സിലാക്കി. അവസാനമായി നമ്മൾ കമ്പ്യൂട്ടേഷന്റെ സിദ്ധാന്തത്തെക്കുറിച്ചും അലൻ ട്യൂറിംഗിന്റെ സംഭാവനകളെക്കുറിച്ചും ട്യൂറിംഗ് യന്ത്രത്തെക്കുറിച്ചും ഉള്ള അറിവുകൾ നേടി.

## മാതൃകചോദ്യങ്ങൾ

### പഠനനേട്ടങ്ങൾ

**ഈ അധ്യായം പഠിച്ച് തീർത്തതിന് ശേഷം പഠിതാവ് ആർജിക്കേണ്ട ശേഷികൾ**

- കമ്പ്യൂട്ടിങ്ങിന്റെ ചരിത്രത്തിലെ നാഴികകല്ലുകളുടെ അടിസ്ഥാന വിവരങ്ങൾ വർഗീകരിക്കുക.
- ആധുനിക കമ്പ്യൂട്ടിങ് യന്ത്രത്തിന്റെ ഘടന മനസ്സിലാക്കുക.
- ഇന്നത്തെ ലോകത്തിൽ ജോൺവോൺ ന്യൂമാന്റെ രൂപകൽപ്പനയുടെ സ്വാധീനം തിരിച്ചറിയുക.
- കമ്പ്യൂട്ടർ സയൻസിലെ മഹാരഥന്മാരെ തിരിച്ചറിയുക.
- ഓരോ തലമുറ കമ്പ്യൂട്ടറിന്റെയും സ്വഭാവ സവിശേഷതകൾ ലിസ്റ്റ് ചെയ്യുക.
- ട്യൂറിങ് യന്ത്രമെന്ന ആശയവും അലൻ ട്യൂറിംഗിന്റെ സംഭാവനകളും വിവരിക്കുക.

#### ഹ്രസ്വോത്തര ചോദ്യങ്ങൾ

1. മായൻ സംഖ്യാന സമ്പ്രദായത്തിന്റെ ബേസ് എന്ത്?
2. ഗ്രീക്ക് സംഖ്യാന സമ്പ്രദായം ..... എന്നറിയപ്പെടുന്നു.
3. അടിസ്ഥാന ഗണിത ക്രിയകൾ ചെയ്യാനുപയോഗിച്ച ആദ്യത്തെ കമ്പ്യൂട്ടറേത്?
4. ലോഗരിതം ആരാണ് കണ്ടുപിടിച്ചത്?
5. ബ്ലെയ്സി പാസ്കൽ നിർമ്മിച്ച യന്ത്രത്തിന്റെ പേര്?
6. ലോകത്തിലെ ആദ്യത്തെ പ്രോഗ്രാമർ ആര്?
7. കമ്പ്യൂട്ടിങ് യന്ത്രങ്ങൾക്ക് തിരിച്ചറിയാൻ കഴിയുന്ന ഭാഷയാണ്.....
8. EDVAC ന്റെ പൂർണ രൂപം എഴുതുക.
9. ഒരു സാധാരണ കമ്പ്യൂട്ടിങ് യന്ത്രത്തിന്റെ പേരെഴുതുക.

#### ലഘു ഉപന്യാസ ചോദ്യങ്ങൾ

1. ഈജിപ്തുകാരുടെ കാലഘട്ടം മുതൽ ചൈനീസുകാരുടെ കാലഘട്ടം വരെയുള്ള സംഖ്യാന സമ്പ്രദായങ്ങൾ വിവരിക്കുക.
2. ഹിന്ദു-അറബിക് സംഖ്യാന സമ്പ്രദായം ലോകത്തിന് നൽകിയ സംഭാവനയെന്ത്?
3. മായൻ സംഖ്യാന സമ്പ്രദായവും റോമൻ സംഖ്യാന സമ്പ്രദായവും താരതമ്യം ചെയ്യുക.
4. അബാക്കസ്സിന്റെ പ്രത്യേകതകൾ വിവരിക്കുക.
5. ചാൾസ് ബാബേജിന്റെ അനലറ്റിക്കൽ എഞ്ചിനും ഡിഫറൻസ് എഞ്ചിനും താരതമ്യം ചെയ്യുക

6. ഹോളറീത്ത് യന്ത്രത്തിന്റെ പ്രാധാന്യം വ്യക്തമാക്കുക.
7. രണ്ടാം ലോകമഹായുദ്ധകാലത്ത് നടന്ന കമ്പ്യൂട്ടിങ് യന്ത്രത്തിന്റെ വളർച്ച എന്ത്?
8. മൂറിന്റെ നിയമം എന്താണ്? ഇതിന്റെ പ്രാധാന്യം വിവരിക്കുക.
9. കമ്പ്യൂട്ടർ ഭാഷകളുടെ പരിണാമം വിവരിക്കുക.
10. ട്യൂറിങ് യന്ത്രത്തിന്റെ പ്രവർത്തനം വിവരിക്കുക.

**ഉപന്യാസ ചോദ്യങ്ങൾ**

1. കമ്പ്യൂട്ടറിന്റെ വിവിധ തലമുറകൾ ലിസ്റ്റ് ചെയ്ത് വിവരിക്കുക.
2. സംഖ്യാന സമ്പ്രദായത്തിന്റെ പരിണാമത്തെക്കുറിച്ച് ഒരു സെമിനാർ റിപ്പോർട്ട് തയാറാക്കുക.
3. 1900 വരെ പുറത്തിറങ്ങിയ വിവിധ കമ്പ്യൂട്ടിങ് യന്ത്രങ്ങളെക്കുറിച്ച് വിവരിക്കുക.

2

## പ്രധാന ആശയങ്ങൾ

- സംഖ്യാ സമ്പ്രദായം
  - ദശസംഖ്യ (Decimal Number), ദ്വയസംഖ്യ (Binary Number), അഷ്ടസംഖ്യ (Octal Number), ഷോഡശസംഖ്യ (Hexadecimal Number) ഹെക്സാഡെസിമൽ സംഖ്യ (Hexadecimal Number)
- സംഖ്യാ പരിവർത്തനങ്ങൾ
- ഡാറ്റ പ്രതിനിധീകരണം
  - പൂർണസംഖ്യകളുടെയും ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യകളുടെയും പ്രതിനിധാനം
  - അക്ഷരങ്ങളുടെ പ്രതിനിധാനം ആസ്കി, എബ്സിഡിക്, ഇസ്കി, യൂണികോഡ്
  - ശബ്ദം, ചിത്രം, വീഡിയോ ഡാറ്റ പ്രതിനിധാനം
- ബൈനറി ഗണിതം
  - സങ്കലനം, വ്യവകലനം, പൂരകങ്ങൾ
- ബൂളിയൻ ബീജഗണിതം ഒരു ആമുഖം
  - ലോജിക് ഓപ്പറേറ്ററുകളും ഗേറ്റുകളും
- ബൂളിയൻ അടിസ്ഥാന അംഗീകൃത തത്വം
- ലളിതമായ ബൂളിയൻ പദപ്രയോഗങ്ങളും സർക്യൂട്ടുകളുടെ രൂപകൽപ്പനയും
- യൂനിവേഴ്സൽ ഗേറ്റുകൾ

R8L9A6

# ഡാറ്റയുടെ പ്രതിനിധാനവും ബൂളിയൻ ബീജഗണിതവും

വ്യത്യസ്തതരം ഡാറ്റകൾ കൈകാര്യം ചെയ്യാൻ ശേഷിയുള്ള ഒരു ഉപകരണമാണ് കമ്പ്യൂട്ടർ. ഡാറ്റ പ്രോസസ്സിംഗ് നടത്തുന്നതിന് വേണ്ടി സംഖ്യകൾ, അക്ഷരങ്ങൾ, ചിത്രങ്ങൾ, വീഡിയോകൾ, ശബ്ദങ്ങൾ തുടങ്ങിയ ഡാറ്റരൂപങ്ങൾ നമ്മൾ കമ്പ്യൂട്ടറിലേക്ക് നൽകാറുണ്ട്. വൈദ്യുതിയുടെ രണ്ട് അവസ്ഥകളായ ഓൺ (ON), ഓഫ് (OFF) എന്നിവയെ അടിസ്ഥാനമാക്കി പ്രവർത്തിക്കുന്ന ഒരു ഇലക്ട്രോണിക് ഉപകരണമാണ് കമ്പ്യൂട്ടർ എന്നത് നമുക്ക് അറിയാം. അതുപോലെ എല്ലാ ഇലക്ട്രോണിക് സർക്യൂട്ടുകൾക്കും തുറന്നിരിക്കുന്നതും അടഞ്ഞിരിക്കുന്നതും ആയ രണ്ടവസ്ഥകളാണുള്ളത്. തുറന്നിരിക്കുന്ന അവസ്ഥയെ സൂചിപ്പിക്കാനായി ഓഫ് (OFF) അല്ലെങ്കിൽ പൂജ്യവും അടഞ്ഞിരിക്കുന്ന അവസ്ഥയെ സൂചിപ്പിക്കാനായി ഓൺ (ON) അല്ലെങ്കിൽ ഒന്നും ഉപയോഗിക്കുന്നു. ഈ രണ്ടവസ്ഥയിലുള്ള പ്രവർത്തനത്തെ ബൈനറി ഓപ്പറേഷൻ (Binary Operation) എന്ന് പറയുന്നു. അതുകൊണ്ട് കമ്പ്യൂട്ടറിലേക്ക് നൽകുന്ന ഡാറ്റയും ബൈനറി രൂപത്തിലായിരിക്കണം. സംഖ്യകൾ, അക്ഷരങ്ങൾ, ചിത്രങ്ങൾ, വീഡിയോകൾ, ശബ്ദങ്ങൾ എന്നിവയെ പ്രതിനിധാനം ചെയ്യുന്നതിനുള്ള വിവിധ രീതികളാണ് ഈ അധ്യായത്തിൽ ചർച്ച ചെയ്യുന്നത്.


*ചിത്രം 2.1 ഡാറ്റയുടെ ബാഹ്യവും ആന്തരികവുമായ രൂപങ്ങൾ*

ഒരു കമ്പ്യൂട്ടറിൽ ഡാറ്റയെ ആന്തരികമായി പ്രതിനിധീകരിക്കുന്നതിന് ഉപയോഗിക്കുന്ന രീതിയാണ് ഡാറ്റയുടെ പ്രതിനിധാനം (Data representation). സംഖ്യകളുടെ പ്രതിനിധാനം ചർച്ച ചെയ്യുന്നതിന് മുമ്പായി സംഖ്യാ സമ്പ്രദായം (Number System) എന്താണെന്നു നമുക്ക് നോക്കാം.

## 2.1 സംഖ്യാ സമ്പ്രദായം (Number systems)

എണ്ണുന്നതിനും, അടയാളപ്പെടുത്തുന്നതിനും, അളക്കുന്നതിനും ഉള്ള ഗണിതശാസ്ത്രപരമായ ഉപാധിയാണ് സംഖ്യ. ചിട്ടയോടെ സംഖ്യകളെ പ്രതിനിധാനം ചെയ്യുന്ന രീതിയാണ് സംഖ്യാന സമ്പ്രദായം. പത്ത് അക്കങ്ങൾ ഉപയോഗിച്ച് കൊണ്ടുള്ള ദശസംഖ്യാ സമ്പ്രദായമാണ് (Decimal Number System) നമ്മൾ നിത്യജീവിതത്തിൽ ഉപയോഗിച്ച് വരുന്നത്. 289 എന്ന സംഖ്യയെ ഇരുന്നൂറ്റി എൺപത്തി ഒൻപത് എന്നാണ് ഉച്ചരിക്കുന്നത്. ഇതിൽ 2, 8, 9 എന്നീ അക്കങ്ങൾ അടങ്ങിയിട്ടുണ്ട്. അതുപോലെ മറ്റ് സംഖ്യാ സമ്പ്രദായങ്ങളും നിലവിലുണ്ട്. ഓരോന്നിനും അതിന്റേതായ ചിഹ്നങ്ങളും രീതികളുമാണ് അവയിലെ സംഖ്യ രൂപകൽപന ചെയ്യുന്നതിന് ഉപയോഗിക്കുന്നത്. ഓരോ സംഖ്യാ സമ്പ്രദായത്തിനും തനതായ ആധാരം ഉണ്ട്. ഇത് ആ സംഖ്യാ സമ്പ്രദായത്തിലെ ചിഹ്നങ്ങളുടെ എണ്ണത്തെ ആശ്രയിച്ചിരിക്കുന്നു. ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽ ഉപയോഗിക്കുന്ന അക്കങ്ങളുടെ അല്ലെങ്കിൽ ചിഹ്നങ്ങളുടെ എണ്ണത്തെ ആ സംഖ്യാ സമ്പ്രദായത്തിലെ ആധാരം (Base) അല്ലെങ്കിൽ മൂലസംഖ്യ (Radix ) എന്ന് പറയുന്നു.

ചില സംഖ്യാ സമ്പ്രദായങ്ങളെ കുറിച്ച് നമുക്ക് ചർച്ച ചെയ്യാം.

### 2.1.1 ദശസംഖ്യാ സമ്പ്രദായം (Decimal number system)

ദശസംഖ്യാ സമ്പ്രദായത്തിൽ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 തുടങ്ങിയ പത്ത് അക്കങ്ങളാണ് സംഖ്യാ രൂപീകരണത്തിന് ഉപയോഗിക്കുന്നത്. ദശസംഖ്യാ സമ്പ്രദായത്തിൽ പത്ത് അക്കങ്ങൾ ഉപയോഗിക്കുന്നതുകൊണ്ട് അതിന്റെ ആധാരം (Basc) 10 ആകുന്നു. അതുകൊണ്ടു ദശസംഖ്യാ സമ്പ്രദായത്തെ 10 ആധാരമാക്കിയ സംഖ്യാ സമ്പ്രദായം എന്നു കൂടി വിളിക്കുന്നു.

743, 347 എന്നീ രണ്ട് ദശസംഖ്യകൾ പരിഗണിക്കുക.

743 = ഏഴ് നൂറുകൾ + നാലു പത്തുകൾ + മൂന്ന് ഒന്നുകൾ $(7\times10^{2} + 4\times10^{1} + 3\times10)$

347 = മൂന്ന് നൂറുകൾ + നാലു പത്തുകൾ + ഏഴ് ഒന്നുകൾ $(3\times10^{2} + 4\times10^{2} + 7\times10)$

ഇവിടെ ഒന്നാമത്തെ സംഖ്യയായ 743 ൽ 7 ന്റെ സ്ഥാനവില (Weight) $10^2 = 100$ ആകുന്നു. എന്നാൽ രണ്ടാമത്തെ സംഖ്യയായ 347 ൽ 7 ന്റെ സ്ഥാനവില $10^0 = 1$ ആകുന്നു. ഒരു സംഖ്യയുടെ സ്ഥാനവില അതിന്റെ ആപേക്ഷിക സ്ഥാനത്തെ ആശ്രയിച്ചിരിക്കുന്നു. അത്തരം സംഖ്യാ സമ്പ്രദായത്തെ സ്ഥാനീയ സംഖ്യാ സമ്പ്രദായം (Positional Number System) എന്നു പറയുന്നു. എല്ലാ സ്ഥാനീയ സംഖ്യാ സമ്പ്രദായത്തിനും ഒരു ആധാരം (Base) ഉണ്ടായിരിക്കും. ഒരു അക്കത്തിന്റെ സ്ഥാനവില ആധാരത്തിന്റെ ചില കൃത്യങ്കം (Power) ആയിരിക്കും. ഓരോ ദശസംഖ്യ അക്കത്തിന്റെ സ്ഥാനവില 10 ന്റെ കൃത്യങ്കം ആയിരിക്കും ($10^0, 10^1, 10^2, \ldots\ldots$). 5876 എന്ന ദശസംഖ്യ പരിഗണിക്കുക. ഈ സംഖ്യ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ചു എഴുതാം.

| സ്ഥാനവില (Weight) | $10^3$ | $10^2$ | $10^1$ | $10^0$ |
|---|---|---|---|---|
| ദശസംഖ്യ | 5 | 8 | 7 | 6 |

$= 5\times10^3 + 8\times10^2 + 7\times10^1 + 6\times10^0$

$= 5\times1000 + 8\times100 + 7\times10 + 6\times1$

$= 5000 + 800 + 70 + 6$

$= 5876$

മുകളിൽ കൊടുത്തിരിക്കുന്ന ഉദാഹരണത്തിൽ 5 എന്ന അക്കത്തിന് ഏറ്റവും കൂടിയ സ്ഥാനവിലയായ $10^3$=1000 ഉം 6 എന്ന അക്കത്തന് ഏറ്റവും കുറഞ്ഞ സ്ഥാനവിലയായ $10^0$=1 ഉം ആണ്. ഏറ്റവും കൂടിയ സ്ഥാനവിലയുള്ള അക്കത്തെ കൂടിയ പ്രബലതയുള്ള അക്കം (Most Significant Digit - MSD) എന്നും ഏറ്റവും കുറഞ്ഞ സ്ഥാനവിലയുള്ള അക്കത്തെ കുറഞ്ഞ പ്രബലതയുള്ള അക്കം (Least Significant Digit - LSD) എന്നും വിളിക്കുന്നു. അതിനാൽ മുകളിൽ കൊടുത്തിരിക്കുന്ന സംഖ്യയിൽ MSD എന്നത് 5 ഉം LSD എന്നത് 6 ഉം ആകുന്നു.

***ഒരു സംഖ്യയുടെ ഏറ്റവും ഇടതു വശത്തുള്ള അക്കം MSD ഉം ഏറ്റവും വലതു വശത്തുള്ള അക്കം LSD ഉം ആകുന്നു.***

ദശാംശ സംഖ്യകളിൽ ദശാംശ ബിന്ദുവിന് വലുത് ഭാഗത്തുള്ള സംഖ്യകളുടെ സ്ഥാനവില 10 ന്റെ നെഗറ്റിവ് കൃത്യങ്കങ്ങൾ ആണ് ($10^{-1}$, $10^{-2}$, $10^{-3}$, …….). 249.367 എന്ന സംഖ്യ ഉദാഹരണമായി എടുക്കാം.

| സ്ഥാനവില (Weight) | $10^2$ | $10^1$ | $10^0$ | $10^{-1}$ | $10^{-2}$ | $10^{-3}$ |
|---|---|---|---|---|---|---|
| ദശസംഖ്യ | 2 | 4 | 9 | 3 | 6 | 7 |
| | MSD | | | (.) | | LSD |

$= 2\times10^2 + 4\times10^1 + 9\times10^0 + 3\times10^{-1} + 6\times10^{-2} + 7\times10^{-3}$

$= 2\times100 + 4\times10 + 9\times1 + 3\times0.1 + 6\times0.01 + 7\times0.001$

$= 200 + 40 + 9 + 0.3 + 0.06 + 0.007$

$= 249.367$

ഇതുവരെ 10 അക്കങ്ങൾ ഉപയോഗിച്ചുകൊണ്ടുള്ള ഒരു സംഖ്യാ സമ്പ്രദായത്തെക്കുറിച്ചാണ് നമ്മൾ ചർച്ചചെയ്തത്. ഇനി നമുക്ക് വ്യത്യസ്ത ആധാരങ്ങളിലുള്ള മറ്റ് സംഖ്യാ സമ്പ്രദായങ്ങൾ രൂപീകരിക്കുന്നതെങ്ങനെ എന്ന് പഠിക്കാം.

## 2.1.2 ദ്വയസംഖ്യാ സമ്പ്രദായം (Binary number system)

ഒരു സംഖ്യ രൂപീകരിക്കാൻ 0, 1 എന്നീ രണ്ടക്കങ്ങൾ മാത്രം ഉപയോഗിക്കുന്ന സംഖ്യാ സമ്പ്രദായത്തെയാണ് ദ്വയസംഖ്യാ സമ്പ്രദായം (Binary Number System) എന്ന് പറയുന്നത്. ഇംഗ്ലീഷിൽ bi

(ബൈ) എന്ന വാക്ക് കൊണ്ട് അർഥമാക്കുന്നത് 2 എന്നാണ്. അതിനാൽ ഈ സംഖ്യാ സമ്പ്രദായത്തിന്റെ ആധാരം 2 ആകുന്നു. അതുകൊണ്ട് ഇതിനെ 2 ആധാരമാക്കിയുള്ള സംഖ്യാ സമ്പ്രദായം എന്ന് കൂടി വിളിക്കുന്നു. ഒരു സംഖ്യ ദ്വയസംഖ്യയാണെന്ന് സൂചിപ്പിക്കാൻ ആ സംഖ്യയോടു കൂടി 2 കീഴ്ക്കുറിപ്പ് (Subscript) ആയി ഉപയോഗിക്കുന്നു.

ഉദാഹരണങ്ങൾ $(1101)_2$, $(101010)_2$, $(1101.11)_2$

ഒരു ദ്വയസംഖ്യയിലെ ഓരോ അക്കത്തെയും ബിറ്റ് (bit) എന്നാണ് വിളിക്കുന്നത്. ഇംഗ്ലീഷിൽ bit ന്റെ പൂർണരൂപം binary digit എന്നാകുന്നു. ദ്വയസംഖ്യാന സമ്പ്രദായവും ഒരു സ്ഥാനീയ സംഖ്യാന സമ്പ്രദായമാണ്. ഓരോ ദ്വയസംഖ്യ അക്കത്തിന്റെയും സ്ഥാന വില 2 ന്റെ കൃത്യങ്കം (Power) ആണ്. $(1101)_2$ എന്ന ദ്വയസംഖ്യ ഉദാഹരണമായി പരിഗണിക്കുക. ഈ ദ്വയസംഖ്യ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ച് എഴുതാം.

| സ്ഥാനവില (Weight) | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|
| Binary Number | 1 | 1 | 0 | 1 |
| | MSB | | | LSB |

$= 1\times2^3 + 1\times 2^2 + 0\times2^1 + 1\times2^0$
$= 1\times8 + 1\times4 + 0\times2 + 1\times1$
$= 8 + 4 + 0 + 1$
$= 13$

ദ്വയസംഖ്യയിലെ ഏറ്റവും വലതുവശത്തു നിൽക്കുന്ന അക്കത്തിനെ കുറഞ്ഞ പ്രബലതയുള്ള ബിറ്റ് (Least Significant Bit - LSB) എന്നും ഏറ്റവും ഇടതുവശത്തു നിൽക്കുന്ന അക്കത്തിനെ കൂടുതൽ പ്രബലതയുള്ള ബിറ്റ് (Most Significant Bit - MSB) എന്നും വിളിക്കുന്നു.

1101 എന്ന ദ്വയസംഖ്യ 13 എന്ന ദശസംഖ്യയ്ക്ക് തുല്യമാണ്. എന്നാൽ 1101 എന്ന സംഖ്യ ദശസംഖ്യാ സമ്പ്രദായത്തിലും ഉണ്ട്. പക്ഷെ അതിനെ വ്യാഖ്യാനിക്കുന്നത് ആയിരത്തി ഒരുന്നൂറ്റി ഒന്ന് എന്നാണ്. ഈ ആശയക്കുഴപ്പം ഒഴിവാക്കുവാൻ വേണ്ടി ദശസംഖ്യാ സമ്പ്രദായം ഒഴികെയുള്ള എല്ലാ സംഖ്യാ സമ്പ്രദായങ്ങളിലും ആധാരം വ്യക്തമായി സൂചിപ്പിക്കണം. അതിന്റെ പൊതുവായ ഘടന താഴെ കൊടുത്തിരിക്കുന്നു.

$$(\text{സംഖ്യ})_{\text{ആധാരം}}$$

വ്യത്യസ്ത ആധാരത്തിലുള്ള സംഖ്യകളെ തരംതിരിച്ചറിയുവാൻ ഈ അടയാളപ്പെടുത്തൽ സഹായിക്കുന്നു. ഉദാഹരണമായി 1101 എന്ന ദ്വയസംഖ്യയെ $(1101)_2$ എന്ന് എഴുതുകയും അതിനെ 'ഒന്ന് ഒന്ന് പൂജ്യം ഒന്ന് ആധാരം രണ്ട് ' എന്ന് വായിക്കുകയും ചെയ്യണം. ഒരു സംഖ്യയ്ക്ക് ആധാരം നൽകിയിട്ടില്ലെങ്കിൽ അതിനെ ദശസംഖ്യയായി പരിഗണിക്കണം. അതായത് ദശസംഖ്യക്ക് ആധാരം സൂചിപ്പിക്കണമെന്ന് നിർബന്ധമില്ല.

ഭിന്നകമായ ഒരു ദ്വയസംഖ്യയുടെ അംശബിന്ദുവിന് (Binary Point) വലതുഭാഗത്തുള്ള അക്കങ്ങളുടെ സ്ഥാനവില 2 ന്റെ നെഗറ്റിവ് കൃത്യങ്കം ആയിരിക്കും. $(2^{-1}, 2^{-2}, 2^{-3},....)$. $(111.011)_2$ എന്ന സംഖ്യ ഉദാഹരണമായി എടുക്കാം.

| സ്ഥാനവില (Weight) | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
|---|---|---|---|---|---|---|
| ദ്വയസംഖ്യ (Binary Number) | 1 | 1 | 1 | 0 | 1 | 1 |

MSB (.) LSB

$= 1\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$

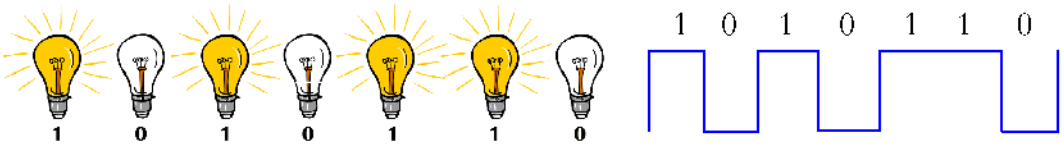
$= 1\times4 + 1\times2 + 1\times1 + 0\times\frac{1}{2} + 1\times\frac{1}{4} + 1\times\frac{1}{8}$

$= 4 + 2 + 1 + 0 + 0.25 + 0.125$

$= 7.375$

### കമ്പ്യൂട്ടറിൽ ദ്വയസംഖ്യയുടെ പ്രധാന്യം

ദ്വയസംഖ്യാ സമ്പ്രദായം 1, 0 എന്നീ അക്കങ്ങൾ അടിസ്ഥാനമാക്കിയാണെന്നു നമ്മൾ കണ്ടല്ലോ. ചിത്രം 2.2 ൽ വൈദ്യുതിയുടെ ഓൺ (ON) ആയിരിക്കുന്ന അവസ്ഥ 1 കൊണ്ടും ഓഫ് (OFF) ആയിരിക്കുന്ന അവസ്ഥ 0 കൊണ്ടും സൂചിപ്പിക്കുന്നു. ഇക്കാരണത്താൽ, കമ്പ്യൂട്ടറിൽ ഡാറ്റയെ പ്രതിനിധാനം ചെയ്യുന്നതിന് അടിസ്ഥാന സംഖ്യാ സമ്പ്രദായമായി ദ്വയസംഖ്യാന സമ്പ്രദായം ഉപയോഗിക്കുന്നു.




*ചിത്രം 2.2 ON ഉം OFF ന്റെയും ഡിജിറ്റൽ രൂപത്തിന്റെ പ്രതിനിധാനം*

## 2.1.3 അഷ്ടസംഖ്യാ സമ്പ്രദായം (Octal number system)

എട്ട് അക്കങ്ങളായ 0, 1, 2, 3, 4, 5, 6, 7 എന്നിവ ഉപയോഗിച്ചുണ്ടാക്കുന്ന സംഖ്യാ സമ്പ്രദായത്തെ അഷ്ടസംഖ്യാ സമ്പ്രദായം (Octal Number System) എന്ന് പറയുന്നു. ഇംഗ്ലീഷിൽ Octa (ഒക്ട) എന്ന വാക്ക് കൊണ്ട് അർഥമാക്കുന്നത് 8 എന്നാണ്. അതിനാലാണ് ഈ സംഖ്യാ സമ്പ്രദായത്തെ ഒക്ടൽ സംഖ്യാ സമ്പ്രദായം എന്ന് പറയുന്നത്. ഈ സംഖ്യാ സമ്പ്രദായത്തിന്റെ ആധാരം 8 ആകുന്നു. അതുകൊണ്ട് ഇതിനെ 8 ആധാരമായ സംഖ്യാ സമ്പ്രദായം എന്നും വിളിക്കുന്നു. ഉദാഹരണമായി $(236)_8$ പരിഗണിക്കുക. ഓരോ ഒക്ടൽ അക്കത്തിന്റെയും സ്ഥാനവില 8 ന്റെ കൃത്യങ്കം (Power) ആയിരിക്കും ($8^0, 8^1, 8^2, 8^3, \ldots$). $(236)_8$ എന്ന സംഖ്യ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ചു എഴുതാം.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|
| ഒക്ടൽ സംഖ്യ | 2 | 3 | 6 |

$= 2\times8^2 + 3\times8^1 + 6\times8^0$

$= 2\times64 + 3\times8 + 6\times1$

$= 128 + 24 + 6$

$= 158$

ഭിന്നകമായ ഒരു അഷ്ടസംഖ്യയുടെ അംശബിന്ദുവിന് വലതുഭാഗത്തുള്ള അക്കങ്ങളുടെ സ്ഥാനവില 8 ന്റെ നെഗറ്റിവ് കൃത്യങ്കം ആയിരിക്കും ($8^{-1}, 8^{-2}, 8^{-3}, \ldots\ldots$). $(172.4)_8$ എന്ന സംഖ്യ ഉദാഹരണമായി എടുക്കാം.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ | $8^{-1}$ |
|---|---|---|---|---|
| ഒകൂൽ സംഖ്യ | 1 | 7 | 2 | 4 |

$$= 1\times8^2 + 7\times8^1 + 2\times8^0 + 4\times8^{-1}$$
$$= 64 + 56 + 2 + 4\times\frac{1}{8}$$
$$= 122 + 0.5$$
$$= 122.5$$

## 2.1.4 ഷോഡശ (ഹെക്സാഡെസിമൽ) സംഖ്യാ സമ്പ്രദായം (Hexadecimal number system)

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F എന്നീ 16 ചിഹ്നങ്ങൾ ഉപയോഗിച്ചുണ്ടാക്കുന്ന സംഖ്യാ സമ്പ്രദായത്തെ ഹെക്സാഡെസിമൽ സംഖ്യാ സമ്പ്രദായം എന്ന് പറയുന്നു. ഈ സംഖ്യാ സമ്പ്രദായത്തിൽ 16 ചിഹ്നങ്ങൾ ഉപയോഗിക്കുന്നതുകൊണ്ട് ഇതിന്റെ ആധാരം 16 ആകുന്നു. ആയതിനാൽ ഇതിനെ 16 ആധാരമായ സംഖ്യാ സമ്പ്രദായം എന്നും വിളിക്കുന്നു. ഈ സംഖ്യാ സമ്പ്രദായത്തിലെ A, B, C, D, E, F എന്നീ ചിഹ്നങ്ങൾ ഉപയോഗിക്കുന്നത് യഥാക്രമം ദശസംഖ്യാ സമ്പ്രദായത്തിലെ 10, 11, 12, 13, 14, 15 എന്ന സംഖ്യകളെ സൂചിപ്പിക്കുന്നതിനാണ്. ഹെക്സാഡെസിമൽ അക്കങ്ങളും അവയ്ക്ക് തുല്യമായ ദശസംഖ്യകളും ചുവടെ കാണിച്ചിരിക്കുന്നു.

| ഹെക്സാഡെസിമൽ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ദശസംഖ്യ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

ഉദാഹരണമായി $(12AF)_{16}$ എന്ന ഹെക്സാഡെസിമൽ സംഖ്യ പരിഗണിക്കുക. ഓരോ ഹെക്സാഡെസിമൽ അക്കത്തിന്റെയും സ്ഥാനവില 16 ന്റെ കൃത്യങ്കം (Power) ആയിരിക്കും ($16^0, 16^1, 16^2, 16^3, \ldots$). ഈ സംഖ്യയെ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ചു എഴുതാം.

| സ്ഥാനവില (Weight) | $16^3$ | $16^2$ | $16^1$ | $16^0$ |
|---|---|---|---|---|
| ഹെക്സാഡെസിമൽ അക്കം | 1 | 2 | A | F |

$$= 1\times16^3 + 2\times16^2 + 10\times16^1 + 15\times16^0$$
$$= 1\times4096 + 2\times256 + 10\times16 + 15\times1$$
$$= 4096 + 512 + 160 + 15$$
$$= 4783$$

ഭിന്നകമായ ഒരു ഹെക്സാഡെസിമൽ സംഖ്യയുടെ അംശബിന്ദുവിന് വലതുഭാഗത്തുള്ള അക്കങ്ങളുടെ സ്ഥാനവില 16 ന്റെ നെഗറ്റിവ് കൃത്യങ്കം ആയിരിക്കും ($16^{-1}, 16^{-2}, 16^{-3}, \ldots$) $(2D.4)_{16}$ എന്ന സംഖ്യാ ഉദാഹരണമായി എടുക്കാം.

| സ്ഥാനവില (Weight) | $16^1$ | $16^0$ | $16^{-1}$ |
|---|---|---|---|
| ഹെക്സാഡെസിമൽ അക്കം | 2 | D | 4 |

$$= 2\times16^1 + 13\times16^0 + 4\times\frac{1}{16}$$
$$= 32 + 13 + 0.25$$
$$= 45.25$$

പട്ടിക 2.1 ൽ വിവിധ സംഖ്യാന സമ്പ്രദായത്തിൽ ഉപയോഗിക്കുന്ന ആധാരവും ചിഹ്നങ്ങളും കാണിച്ചിരിക്കുന്നു.

| സംഖ്യാന സമ്പ്രദായം | ആധാരം | ഉപയോഗിക്കുന്ന ചിഹ്നങ്ങൾ |
|---|---|---|
| ബൈനറി | 2 | 0, 1 |
| ഒക്ടൽ | 8 | 0, 1, 2, 3, 4, 5, 6, 7 |
| ഡെസിമൽ | 10 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| ഹെക്സാ ഡെസിമൽ | 16 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F |

*പട്ടിക 2.1 വിവിധ സംഖ്യാ സമ്പ്രദായത്തിലുമുള്ള ആധാരവും ചിഹ്നങ്ങളും*

**ഒക്ടൽ, ഹെക്സാഡെസിമൽ സംഖ്യാ സമ്പ്രദായങ്ങളുടെ പ്രാധാന്യം**

കമ്പ്യൂട്ടറിൽ ഡാറ്റ പ്രോസസ്സ് ചെയ്യുന്നതിനും അതിനെ പ്രതിനിധാനം ചെയ്യുന്നതിനും ബൈനറി സംഖ്യാ സമ്പ്രദായമാണ് ഉപയോഗിക്കുന്നത് എന്ന് നമ്മൾ മനസ്സിലാക്കിക്കഴിഞ്ഞു. ബൈനറി സംവിധാനത്തിൽ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുന്നതിനും അവയുടെ പ്രവർത്തനങ്ങൾക്കും കൂടുതൽ ബിറ്റുകളും പ്രയത്നങ്ങളും ആവശ്യമാണ്. മൂന്നു ബിറ്റുകളുടെ ഗ്രൂപ്പിനെ ഒരു ഒക്ടൽ അക്കമായും (കാരണം $2^3 = 8$) നാലു ബിറ്റുകളുടെ ഗ്രൂപ്പിനെ ഒരു ഹെക്സാഡെസിമൽ അക്കമായും (കാരണം $2^4 = 16$) മാറ്റാവുന്നതും ഇത്തരം ഗ്രൂപ്പുകളെ അവയുടെ തത്തുല്യമായ ഒക്ടൽ, ഹെക്സാഡെസിമൽ ചിഹ്നങ്ങളിലേക്കു മാറ്റാവുന്നതാണ്. ബൈനറി സംഖ്യകളുടെ ഒക്ടൽ, ഹെക്സാഡെസിമൽ സംഖ്യാ സമ്പ്രദായത്തിലേക്കുള്ള ഇത്തരം മാറ്റവും തിരിച്ചുള്ള മാറ്റവും വളരെ എളുപ്പമാണ്. ഇലക്ട്രോണിക് സർക്യൂട്ടുകളുടെ രൂപകൽപ്പനയിലും പ്രവർത്തനത്തിലും ഈ പരിവർത്തന പ്രക്രിയ വലിയ തോതിൽ ഉപയോഗിക്കുന്നു.

**സ്വയം വിലയിരുത്താം**

1. ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽ ഉപയോഗിച്ചിരിക്കുന്ന ചിഹ്നങ്ങളുടെ എണ്ണത്തെ ..... എന്ന് വിളിക്കുന്നു.
2. താഴെ കൊടുത്തിരിക്കുന്നതിൽ നിന്ന് അസാധുവായ സംഖ്യകൾ തിരഞ്ഞെടുക്കുക.
   i) $(10101)_8$  ii) $(123)_4$  iii) $(768)_8$  iv) $(ABC)_{16}$
3. ബിറ്റ് എന്ന പദം നിർവചിക്കുക.
4. 7854.25. എന്ന ദശസംഖ്യയുടെ എം.എസ്.ഡി (MSD) കണ്ടുപിടിക്കുക.
5. ഹെക്സാഡെസിമൽ സംഖ്യാ സമ്പ്രദായത്തിന്റെ ആധാരം ..... ആകുന്നു.

## 2.2 സംഖ്യകളുടെ പരിവർത്തനങ്ങൾ (Number conversions)

വിവിധ സംഖ്യാ സമ്പ്രദായങ്ങളെക്കുറിച്ച് നമ്മൾ പഠിച്ചു കഴിഞ്ഞു, ഒരാധാരത്തിലുള്ള സംഖ്യകളെ മറ്റൊരാധാരത്തിലുള്ള തത്തുല്യ സംഖ്യകളാക്കി പരിവർത്തനം ചെയ്യുന്നതെങ്ങനെയാണെന്നു നമുക്ക് ചർച്ച ചെയ്യാം. ദശസംഖ്യയിൽ നിന്ന് ബൈനറി, ബൈനറിയിൽ നിന്ന് ദശസംഖ്യ, ദശസംഖ്യയിൽ നിന്ന് ഒക്ടൽ എന്നിങ്ങനെ പല വിധത്തിലുള്ള സംഖ്യാ സമ്പ്രദായത്തിലേക്കു പരിവർത്തനം ചെയ്യാം. ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽ നിന്ന് മറ്റൊരു സംഖ്യാ സമ്പ്രദായത്തിലേക്ക് എങ്ങനെ പരിവർത്തനം ചെയ്യാമെന്ന് നമുക്ക് നോക്കാം.

### 2.2.1 ദശസംഖ്യയിൽ നിന്ന് ബൈനറിസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Decimal to binary conversion)

ആവർത്തിച്ചുള്ള ഹരണം വഴിയാണ് ദശസംഖ്യയെ ബൈനറി സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യുന്നത്. ഈ രീതിയിൽ ദശസംഖ്യയെ 2 കൊണ്ട് തുടർച്ചയായി ഹരിക്കുകയും (സംഖ്യ 0 ആകുന്നത് വരെ), അതിന്റെ ശിഷ്ടങ്ങൾ രേഖപ്പെടുത്തുകയും ചെയ്യുന്നു. MSB അവസാന ശിഷ്ടമായും LSB ആദ്യത്തെ ശിഷ്ടമായും എടുത്ത് ശിഷ്ടങ്ങളെ കൂട്ടമായി എഴുതിയാൽ ദ്വയസംഖ്യക്ക് തുല്യമായ സംഖ്യ ലഭിക്കുന്നു. ഓരോ ഘട്ടത്തിലും ഹരിക്കുമ്പോൾ കിട്ടുന്ന ശിഷ്ടങ്ങൾ ഒന്നുകിൽ 0 അല്ലെങ്കിൽ 1 എന്നീ ബൈനറി അക്കങ്ങൾ ആയിരിക്കും.

**ഉദാഹരണങ്ങൾ**

25 എന്ന ദശസംഖ്യയുടെ ബൈനറിക്ക് തുല്യമായ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 2 | 25 | | |
| 2 | 12 | 1 | LSB |
| 2 | 6 | 0 | |
| 2 | 3 | 0 | |
| 2 | 1 | 1 | |
| | 0 | 1 | MSB |

$(25)_{10} = (11001)_2$

$(80)_{10}$ ന് തുല്യമായ ബൈനറി സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 2 | 80 | | |
| 2 | 40 | 0 | LSB |
| 2 | 20 | 0 | |
| 2 | 10 | 0 | |
| 2 | 5 | 0 | |
| 2 | 2 | 1 | |
| 2 | 1 | 0 | |
| | 0 | 1 | MSB |

$(80)_{10} = (1010000)_2$

**സൂചന:** ഒറ്റ സംഖ്യയായ ദശസംഖ്യയ്ക്ക് തുല്യമായ ബൈനറി സംഖ്യ 1 ൽ അവസാനിക്കുകയും ഇരട്ട സംഖ്യയായ ദശസംഖ്യക്ക് തുല്യമായ ബൈനറി സംഖ്യ 0 ൽ അവസാനിക്കുകയും ചെയ്യുന്നു.

### ദശാംശ സംഖ്യകൾ ബൈനറിയിലേക്ക് പരിവർത്തനം ചെയ്യൽ

ദശാംശ സംഖ്യകൾ ബൈനറിയിലേക്ക് മാറ്റാൻ അതിനെ തുടർച്ചയായി 2 കൊണ്ട് ഗുണിക്കുന്ന രീതിയാണ് നാം ഉപയോഗിക്കുന്നത്. ഉത്തരത്തിന്റെ പൂർണസംഖ്യഭാഗം ബൈനറി ഭിന്നകത്തിലെ MSB ആയിരിക്കും. അടുത്ത ബൈനറി ഭിന്നകത്തിന്റെ പ്രബലതയുള്ള ബിറ്റ് കിട്ടുന്നതിന് വീണ്ടും ഭിന്നക ഭാഗത്തിന്റെ ഉത്തരത്തെ 2 കൊണ്ട് ഗുണിക്കുന്നു. ഭിന്നക ഭാഗം പൂജ്യം ആകുന്നതു വരെയോ അല്ലെങ്കിൽ ആവശ്യമുള്ളത്ര കൃത്യത (Precision) ലഭിക്കുന്നത് വരെയോ ഈ നടപടിക്രമം തുടരുന്നു.

**ഉദാഹരണങ്ങൾ:**

(0.75 നെ ബൈനറിയിലേക്ക് മാറ്റുക.

| | |
|---|---|
| | 0.75 × 2 = 1.50 |
| 1 | .50 × 2 = 1.00 |
| 1 | .00 |

$(0.75)_{10} = (0.11)_2$

(0.625 നെ ബൈനറിയിലേക്ക് മാറ്റുക.

| | |
|---|---|
| | 0.625 × 2 = 1.25 |
| 1 | .25 × 2 = 0.50 |
| 0 | .50 × 2 = 1.00 |
| 1 | .00 |

$(0.625)_{10} = (0.101)_2$

15.25 നെ ബൈനറിയിലേക്ക് മാറ്റുക.

15 നെ ബൈനറിയിലേക്കു മാറ്റുക.

| | | |
|---|---|---|
| 2 | 15 | ശിഷ്ടങ്ങൾ |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | 0 | 1 |

(0.25നെ ബൈനറിയിലേക്കു മാറ്റുക

| | |
|---|---|
| | 0.25 × 2 = 0.50 |
| 0 | .50 × 2 = 1.00 |
| 1 | .00 |

$(15.25)_{10} = (1111.01)_2$

## 2.2.2 ദശസംഖ്യയിൽ നിന്ന് ഒക്ടലിലേക്കുള്ള പരിവർത്തനം (Decimal to Octal conversion)

ആവർത്തിച്ചുള്ള ഹരണം വഴിയാണ് ദശസംഖ്യയെ ഒക്ടൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യുന്നത്. ദശസംഖ്യയെ 8 കൊണ്ട് തുടർച്ചയായി ഹരിക്കുകയും (സംഖ്യ 0 ആകുന്നത് വരെ), അതിന്റെ ശിഷ്ടങ്ങൾ രേഖപ്പെടുത്തുകയും ചെയ്യുന്നു. MSD അവസാന ശിഷ്ടമായും LSD ആദ്യത്തെ

ശിഷ്ടമായും എടുത്ത് ശിഷ്ടങ്ങളെ കൂട്ടമായി എഴുതിയാൽ ഒക്ടൽസംഖ്യക്ക് തുല്യമായ സംഖ്യ ലഭിക്കുന്നു. ഓരോ ഘട്ടത്തിലും ഹരിക്കുമ്പോൾ കിട്ടുന്ന ശിഷ്ടങ്ങൾ 0, 1, 2, 3, 4, 5, 6 , 7.എന്നിവയിൽ ഏതെങ്കിലും ആയിരിക്കും.

**ഉദാഹരണങ്ങൾ:**

125 എന്ന ദശസംഖ്യക്ക് തുല്യമായ ഒക്ടൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 8 | 125 | | |
| 8 | 15 | 5 | LSD |
| 8 | 1 | 7 | |
| | 0 | 1 | MSD |

$(125)_{10} = (175)_8$

$(400)_{10}$ ന് തുല്യമായ ഒക്ടൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ |
|---|---|---|
| 8 | 400 | |
| 8 | 50 | 0 |
| 8 | 6 | 2 |
| | 0 | 6 |

$(400)_{10} = (620)_8$

## 2.2.3 ദശസംഖ്യയിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കുള്ള പരിവർത്തനം (Decimal to hexadecimal conversion)

ആവർത്തിച്ചുള്ള ഹരണം വഴിയാണ് ദശസംഖ്യയെ ഹെക്സാഡെസിമൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യുന്നത്. ദശസംഖ്യയെ 16 കൊണ്ട് തുടർച്ചയായി ഹരിക്കുകയും (സംഖ്യ 0 ആകുന്നത് വരെ), അതിന്റെ ശിഷ്ടങ്ങൾ രേഖപ്പെടുത്തുകയും ചെയ്യുന്നു. MSD അവസാന ശിഷ്ടമായും LSD ആദ്യ ശിഷ്ടമായും എടുത്ത് ശിഷ്ടങ്ങളെ കൂട്ടമായി എഴുതിയാൽ ഹെക്സാഡെസിമൽ സംഖ്യക്ക് തുല്യമായ സംഖ്യ ലഭിക്കുന്നു. ഓരോ ഘട്ടത്തിലും ഹരിക്കുമ്പോൾ കിട്ടുന്ന ശിഷ്ടങ്ങൾ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 എന്നിവയിൽ ഏതെങ്കിലും ആയിരിക്കും. കിട്ടുന്ന ശിഷ്ടങ്ങൾ 10, 11, 12, 13, 14, 15 ആണെങ്കിൽ അതിനെ യഥാക്രമം A, B, C, D, E, F എന്നിങ്ങനെ രേഖപ്പെടുത്തണം.

**ഉദാഹരണങ്ങൾ:**

155 എന്ന ദശസംഖ്യക്ക് തുല്യമായ ഹെക്സാഡെസിമൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 16 | 155 | | |
| 16 | 9 | 11 (B) | → LSD |
| | 0 | 9 | → MSD |

$(155)_{10} = (9B)_{16}$

**ഉദാഹരണങ്ങൾ:** $380_{10}$ തുല്യമായ ഹെക്സാഡെസിമൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | |
|---|---|---|
| 16 | 380 | ശിഷ്ടങ്ങൾ |
| 16 | 23 | 12 (C) ↑ |
| 16 | 1 | 7 |
| | 0 | 1 |

$(380)_{10} = (17C)_{16}$

## 2.2.4 ബൈനറിയിൽ നിന്ന് ദശസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Binary to decimal conversion)

ബൈനറി സംഖ്യകൾക്ക് തുല്യമായ ദശസംഖ്യ കാണുന്നതിന്, ബൈനറി സംഖ്യയിലെ ഓരോ അക്കത്തിനെയും, അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. സ്ഥാനവില 2 ന്റെ കൃത്യങ്കം ആയിരിക്കും $(2^0, 2^1, 2^2, 2^3, ...)$..

**ഉദാഹരണങ്ങൾ:**

$(10110)_2$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 | 1 | 0 |

$(10110)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 16 + 0 + 4 + 2 + 0$

$= 22$

$(10110)_2 = (22)_{10}$

$(11011)_2$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 1 | 0 | 1 | 1 |

$(11011)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 16 + 8 + 2 + 1$

$= 27$

$(11011)_2 = (27)_{10}$

$(1100010)_2$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

$(1100010)_2 = 1\times2^6 + 1\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 64 + 32 + 2$

$= 98$

**$(1100010)_2 = (98)_{10}$**

പട്ടിക 2.2 ൽ രണ്ടിന്റെ 10 വരെയുള്ള കൃത്യങ്കങ്ങൾ കൊടുത്തിരിക്കുന്നു.

| $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

*പട്ടിക 2.2 രണ്ടിന്റെ കൃത്യങ്കങ്ങൾ*

## ബൈനറി ഭിന്നകങ്ങൾ ദശസംഖ്യയിലേക്ക് പരിവർത്തനം ചെയ്യൽ

ഒരു ബൈനറി ഭിന്നസംഖ്യ ദശസംഖ്യയിലേക്ക് മാറ്റുന്നതിന്, ഓരോ അക്കത്തിനെയും അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. ബൈനറി അംശബിന്ദുവിന് ശേഷമുള്ള അക്കത്തിന്റെ സ്ഥാനവില 2 ന്റെ നെഗറ്റീവ് കൃത്യങ്കം ആയിരിക്കും $(2^{-1}, 2^{-2}, 2^{-3}, ...)$

**ഉദാഹരണങ്ങൾ:**

$(0.1011)_2$ നെ ദശസംഖ്യയിലേക്ക് മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 | 1 |

$(0.1011)_2 = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 0.5 + 0 + 0.125 + 0.0625$

$= 0.6875$

**$(0.1011)_2 = (0.6875)_{10}$**

$(0.101)_2$നെ ദശസംഖ്യയിലേക്ക് മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 |

$(0.101)_2 = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$= 0.5 + 0 + 0.125$

$= 0.625$

**$(0.101)_2 = (0.625)_{10}$**

$(1010.11)_2$ നെ ദശസംഖ്യയിലേക്ക് മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 | 1 |

$(1010)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 8 + 0 + 2 + 0$

$= 10$ $\qquad (1010)_2 = (10)_{10}$

$(0.11)_2 = 1\times2^{-1} + 1\times2^{-2}$

$= 0.5 + 0.25$

$= 0.75$ $\qquad (0.11)_2 = (0.75)_{10}$

| സ്ഥാനവില (Weight) | $2^{-1}$ | $2^{-2}$ |
|---|---|---|
| ബൈനറി അക്കം | 1 | 1 |

$(1010.11)_2 = (10.75)_{10}$

പട്ടിക 2.3 ൽ രണ്ടിന്റെ നെഗറ്റീവ് കൃത്യങ്കങ്ങൾ കാണിച്ചിരിക്കുന്നു.

| $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ | $2^{-5}$ |
|---|---|---|---|---|
| 0.5 | 0.25 | 0.125 | 0.0625 | 0.03125 |

*പട്ടിക 2.3 രണ്ടിന്റെ നെഗറ്റീവ് കൃത്യങ്കങ്ങൾ*

## 2.2.5 ഒക്ടൽ സംഖ്യയിൽ നിന്ന് ദശസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Octal to decimal conversion)

ഒക്ടൽ സംഖ്യയെ ദശസംഖ്യയിലേക്കു മാറ്റുന്നതിന്, ഒക്ടൽ സംഖ്യയിലെ ഓരോ അക്കത്തിനെയും, അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. സ്ഥാനവില 8 ന്റെ കൃത്യങ്കം ആയിരിക്കും $(8^0, 8^1, 8^2, 8^3, \ldots)$.

**ഉദാഹരണങ്ങൾ:**

$(257)_8$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|
| ഒക്ടൽ അക്കം | 2 | 5 | 7 |

$(257)_8 = 2\times8^2 + 5\times8^1 + 7\times8^0$

$= 128 + 40 + 7$

$= 175$

$(257)_8 = (175)_{10}$

$(157)_8$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|
| ഒക്ടൽ അക്കം | 1 | 5 | 7 |

$(157)_8 = 1\times8^2 + 5\times8^1 + 7\times8^0$

$= 64 + 40 + 7$

$= 111$

**$(157)_8 = (111)_{10}$**

$(1005)_8$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില Weight | $8^3$ | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|---|
| ഒക്ടൽ അക്കം | 1 | 0 | 0 | 5 |

$(1005)_8 = 1\times8^3 + 0\times8^2 + 0\times8^1 + 5\times8^0$

$= 512+5$

$= 517$

**$(1005)_8 = (517)_{10}$**

## 2.2.6 ഹെക്സാഡെസിമൽ സംഖ്യയിൽ നിന്ന് ദശസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Hexadecimal to decimal conversion)

ഹെക്സാഡെസിമൽ സംഖ്യയെ ദശസംഖ്യയിലേക്കു മാറ്റുന്നതിന്, ഹെക്സാഡെസിമൽ സംഖ്യയിലെ ഓരോ അക്കത്തിനെയും, അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. സ്ഥാനവില 16 ന്റെ കൃത്യങ്കം ആയിരിക്കും ($16^0, 16^1, 16^2, ...$). ഹെക്സാഡെസിമൽ അക്കങ്ങൾ A, B, C, D, E, F ആണെങ്കിൽ അത് യഥാക്രമം 10, 11, 12, 13, 14, 15 എന്നിങ്ങനെ മാറ്റി എഴുതണം.

**ഉദാഹരണങ്ങൾ:**

$(AB)_{16}$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാന വില (Weight) | $16^1$ | $16^0$ |
|---|---|---|
| ഹെക്സാഡെസിമൽ അക്കം | A | B |

A = 10 B = 11

$(AB)_{16} = 10\times16^1+11\times16^0$

$= 160 + 11$

$= 171$

**$(AB)_{16} = (171)_{10}$**

**ഉദാഹരണങ്ങൾ:** $(2D5)_{16}$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാന വില (Weight) | $16^1$ | $16^1$ | $16^0$ |
|---|---|---|---|
| ഹെക്സാഡെസിമൽ അക്കം | 2 | D | 5 |

D = 13

$(2D5)_{16}$ $= 2\times16^2+13\times16^1+5\times16^0$

$= 512+208+5$

$= 725$

**$(AB)_{16} = (171)_{10}$**

## 2.2.7 ഒക്ടലിൽ നിന്ന് ബൈനറിയിലേക്കുള്ള പരിവർത്തനം (Octal to binary conversion)

ഓരോ ഒക്ടൽ അക്കവും തത്തുല്യമായ 3 ബിറ്റ് ബൈനറി അക്കത്തിലേക്ക് മാറ്റി എഴുതിയാൽ ഒക്ടൽ സംഖ്യ ബൈനറി സംഖ്യയായി പരിവർത്തനം ചെയ്യാനാകും. സാധ്യമായ എട്ട് ഒക്ടൽ അക്കങ്ങളും അവയുടെ തത്തുല്യ ബൈനറി അക്കങ്ങളും പട്ടിക 2.4 ൽ നൽകിയിരിക്കുന്നു.

| ഒക്ടൽ അക്കം | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| തുല്യമായ ബൈനറി | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

*പട്ടിക 2.4 ഒക്ടൽ അക്കങ്ങളുടെ തത്തുല്യമായ ബൈനറി സംഖ്യകൾ.*

**ഉദാഹരണങ്ങൾ:**

$(437)_8$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

ഓരോ ഒക്ടൽ അക്കത്തിനും തുല്യമായ 3 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

| 4 | 3 | 7 |
|---|---|---|
| ↓ | ↓ | ↓ |
| 100 | 011 | 111 |

**$(437)_8=(100011111)_2$**

$(7201)_8$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

ഓരോ ഒക്ടൽ അക്കത്തിനും തുല്യമായ 3 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

| 7 | 2 | 0 | 1 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 111 | 010 | 000 | 001 |

**$(7201)_8 = (111010000001)_2$**

## 2.2.8 ഹെക്സാഡെസിമലിൽ നിന്ന് ബൈനറിയിലേക്കുള്ള പരിവർത്തനം (Hexadecimal to binary conversion)

ഓരോ ഹെക്സാഡെസിമൽ അക്കവും തത്തുല്യമായ 4 ബിറ്റ് ബൈനറി ആക്കി മാറ്റി എഴുതിയാൽ ഹെക്സാഡെസിമൽ സംഖ്യ ബൈനറിയായി പരിവർത്തനം ചെയ്യാനാകും. ഹെക്സാഡെസിമൽ അക്കങ്ങളും അവയ്ക്കു തുല്യമായ ബൈനറി അക്കങ്ങളും പട്ടിക 2.5 ൽ നൽകിയിരിക്കുന്നു.

**ഉദാഹരണങ്ങൾ:**

$(AB)_{16}$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

ഓരോ ഹെക്സാഡെസിമൽ അക്കത്തിനും തുല്യമായ 4 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

| A | B |
|---|---|
| ↓ | ↓ |
| 1010 | 1011 |

$$(AB)_{16} = (10101011)_2$$

$(2F15)_{16}$ നെ ബൈനറിയിലേക്കു മാറ്റുക

ഓരോ ഹെക്സാഡെസിമൽ അക്കത്തിനും തുല്യമായ 4 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

| 2 | F | 1 | 5 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 0010 | 1111 | 0001 | 0101 |

$$(2F15)_{16} = (10111100010101)_2$$

| ഹെക്സാഡെസിമൽ അക്കം | തുല്യമായ ബൈനറി |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

*പട്ടിക 2.5 ഹെക്സാഡെസിമൽ അക്കങ്ങളുടെ തത്തുല്യമായ ബൈനറി അക്കങ്ങൾ.*

## 2.2.9 ബൈനറിയിൽ നിന്നും ഒക്ടലിലേക്കുള്ള പരിവർത്തനം (Binary to octal conversion)

തന്നിരിക്കുന്ന ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 3 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി അതിന്റെ തത്തുല്യമായ ഒക്ടൽ അക്കം എഴുതിയാൽ ഒരു ബൈനറി സംഖ്യ ഒക്ടൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യാം. മൂന്നിന്റെ കൂട്ടങ്ങൾ ആക്കുമ്പോൾ ഏറ്റവും ഇടത് വശത്തെ കൂട്ടത്തിൽ 3 ബിറ്റുകൾ തികയുന്നില്ലെങ്കിൽ ഇടത് വശത്ത് ആവശ്യമായ പൂജ്യങ്ങൾ കൊടുത്ത് 3 ബിറ്റ് രൂപത്തിൽ ആക്കണം.

**ഉദാഹരണങ്ങൾ:**

$(101100111)_2$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 101100111 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതുപോലെ കൂട്ടങ്ങളാക്കാം.

| 101 | 100 | 111 |
|---|---|---|
| ↓ | ↓ | ↓ |
| 5 | 4 | 7 |

$$(101100111)_2 = (547)_8$$

$(10011000011)_2$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 10011000011 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതു പോലെ കൂട്ടങ്ങളാക്കാം.

കൂട്ടങ്ങളാക്കിയശേഷം ഏറ്റവും ഇടത് ഭാഗത്തെ കൂട്ടത്തിൽ 3 ബിറ്റുകൾ ഇല്ലെങ്കിൽ ആവശ്യമായ 0 ചേർത്ത് 3 ബിറ്റുകൾ ആക്കുക.

| 010 | 011 | 000 | 011 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 2 | 3 | 0 | 3 |

$(10011000011)_2 = (2303)_8$

## 2.2.10 ബൈനറിയിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കുള്ള പരിവർത്തനം (Binary to hexadecimal conversion)

തന്നിരിക്കുന്ന ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 4 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി അതിന്റെ തത്തുല്യമായ ഹെക്സാഡെസിമൽ അക്കം എഴുതിയാൽ ഒരു ബൈനറി സംഖ്യയെ ഹെക്സാഡെസിമൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യാം. നാലിന്റെ കൂട്ടങ്ങൾ ആക്കുമ്പോൾ ഏറ്റവും ഇടത് വശത്തെ കൂട്ടത്തിൽ 4 ബിറ്റുകൾ തികയുന്നില്ലെങ്കിൽ ഇടത് വശത്ത് ആവശ്യമായ പൂജ്യങ്ങൾ കൊടുത്ത് 4 ബിറ്റ് രൂപത്തിൽ ആക്കണം.

**ഉദാഹരണങ്ങൾ:**

$(101100111010)_2$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 101100111010 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതുപോലെ കൂട്ടങ്ങളാക്കാം.

| 1011 | 0011 | 1010 |
|---|---|---|
| ↓ | ↓ | ↓ |
| B | 3 | A |

$(101100111010)_2 = (B3A)_{16}$

$(110111100001100)_2$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 110111100001100 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതു പോലെ കൂട്ടങ്ങളാക്കാം.

കൂട്ടങ്ങളാക്കിയശേഷം ഏറ്റവും ഇടത് ഭാഗത്തെ കൂട്ടത്തിൽ 4 ബിറ്റുകൾ ഇല്ലെങ്കിൽ ആവശ്യമായ 0 ചേർത്ത് 4 ബിറ്റുകൾ ആക്കുക.

| 0110 | 1111 | 0000 | 1100 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 6 | F | 0 | C |

$(110111100001100)_2 = (6F0C)_{16}$

## 2.2.11 ഒക്ടലിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കുള്ള പരിവർത്തനം. (Octal to Hexadecimal conversion)

ഒക്ടൽ സംഖ്യയിൽ നിന്ന് ഹെക്സാഡെസിമൽ സംഖ്യയിലേക്ക് മാറ്റുന്നതിന് രണ്ട് ഘട്ടങ്ങൾ ഉണ്ട്. ആദ്യം ഒക്ടൽ സംഖ്യ ബൈനറിയായി പരിവർത്തനം ചെയ്യുക. ഈ ബൈനറി സംഖ്യ തത്തുല്യമായ ഹെക്സഡെസിമൽ സംഖ്യയിലേക്കു മാറ്റുക.

**ഉദാഹരണം:**

$(457)_8$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

ഘട്ടം 1. $(457)_8$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

$$(457)_8 = \underset{\downarrow}{4}\ \underset{\downarrow}{5}\ \underset{\downarrow}{7}$$
$$100\ \ 101\ \ 111$$
$$= (100101111)_2$$

ഘട്ടം 2. $(100101111)_2$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

$(100101111)_2$ നെ 4 ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി മാറ്റുക.

$$(100101111)_2 = \underset{\downarrow}{0001}\ \underset{\downarrow}{0010}\ \underset{\downarrow}{1111}$$
$$= 1\ \ 2\ \ F$$
$$= (12F)_{16}$$

$$\mathbf{(457)_8 = (12F)_{16}}$$

## 2.2.12 ഹെക്സാഡെസിമലിൽ നിന്ന് ഒക്ടലിലേക്കുള്ള പരിവർത്തനം (Hexadecimal to Octal conversion)

ഹെക്സാഡെസിമൽ സംഖ്യയിൽ നിന്ന് ഒക്ടൽ സംഖ്യയിലേക്ക് മാറ്റുന്നതിന് രണ്ട് ഘട്ടങ്ങൾ ഉണ്ട്. ആദ്യം ഹെക്സാഡെസിമൽ സംഖ്യ ബൈനറിയായി പരിവർത്തനം ചെയ്യുക. ഈ ബൈനറി സംഖ്യ തത്തുല്യമായ ഒക്ടൽ സംഖ്യയിലേക്കു മാറ്റുക..

**ഉദാഹരണം:**

$(A2D)_{16}$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

ഘട്ടം 1. $(A2D)_{16}$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

$$(A2D)_{16} = \underset{\downarrow}{A}\ \underset{\downarrow}{2}\ \underset{\downarrow}{D}$$
$$1010\ \ 0010\ \ 1101$$
$$= (101000101101)_2$$

ഘട്ടം 2. $(101000101101)_2$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

$(101000101101)_2$ നെ 3 ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി മാറ്റുക.

$$(101000101101)_2 = \underset{\downarrow}{101}\ \underset{\downarrow}{000}\ \underset{\downarrow}{101}\ \underset{\downarrow}{101}$$
$$5\ \ 0\ \ 5\ \ 5$$
$$= (5055)_8$$

$$\mathbf{(A2D)_{16} = (5055)_8}$$

**പട്ടിക 2.6 ൽ വിവിധ സംഖ്യാ പരിവർത്തനങ്ങളുടെ നടപടിക്രമങ്ങൾ കാണിച്ചിക്കുന്നു.**

| സംഖ്യ പരിവർത്തനം | നടപടിക്രമം |
|---|---|
| ദശസംഖ്യയിൽ നിന്ന് ബൈനറിയിലേക്ക് | തുടർച്ചയായി 2 കൊണ്ട് ഹരിച്ച് ശിഷ്ടങ്ങൾ കൂട്ടങ്ങളാക്കുക. |
| ദശസംഖ്യയിൽ നിന്ന് | തുടർച്ചയായി 8 കൊണ്ട് ഹരിച്ച് ശിഷ്ടങ്ങൾ കൂട്ടങ്ങളാക്കുക. |
| ഒക്ടലിലേക്ക് | |
| ദശസംഖ്യയിൽ നിന്ന് | തുടർച്ചയായി 16 കൊണ്ട് ഹരിച്ച് ശിഷ്ടങ്ങൾ കൂട്ടങ്ങളാക്കുക. |
| ഹെക്സാഡെസിമലിലേക്ക് | |
| ബൈനറിയിൽ നിന്ന് ദശസംഖ്യയിലേക്ക് | ബൈനറി സംഖ്യയിലെ ഓരോ അക്കത്തിന്റെയും സ്ഥാനവില (2 ന്റെ കൃത്യങ്കം) കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കാണുക. |
| ഒക്ടലിൽ നിന്ന് ദശസംഖ്യയിൽലേക്ക് | ഒക്ടൽ സംഖ്യയിലെ ഓരോ അക്കത്തിന്റെയും സ്ഥാനവില (8 ന്റെ കൃത്യങ്കം) കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കാണുക. |
| ഹെക്സാഡെസിമലിൽ നിന്ന് ദശസംഖ്യയിലേക്ക് | ഹെക്സാഡെസിമൽ സംഖ്യയിലെ ഓരോ അക്കത്തിന്റെയും സ്ഥാനവില (16 ന്റെ കൃത്യങ്കം) കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കാണുക. |
| ഒക്ടലിൽ നിന്ന് ബൈനറിയിലേക്ക് | ഓരോ ഒക്ടൽ അക്കവും 3 ബിറ്റ് ബൈനറി സംഖ്യ ആയി പരിവർത്തനം ചെയ്യുക. |
| ഹെക്സാഡെസിമലിൽ നിന്ന് ബൈനറിയിലേക്ക് | ഓരോ ഹെക്സാഡെസിമൽ അക്കവും 4 ബിറ്റ് ബൈനറി സംഖ്യ ആയി പരിവർത്തനം ചെയ്യുക. |
| ബൈനറിയിൽ നിന്ന് ഒക്ടലിലേക്ക് | ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 3 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി അതിന്റെ തുല്യമായ ഒക്ടൽ അക്കം എഴുതുക. |
| ബൈനറിയിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്ക് | ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 4 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി അതിന്റെ തുല്യമായ ഹെക്സാഡെസിമൽ അക്കം എഴുതുക. |
| ഒക്ടലിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്ക് | ഒക്ടലിനെ ബൈനറിയിലേക്കും തുടർന്ന് ബൈനറിയിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കും മാറ്റുക. |
| ഹെക്സാഡെസിമലിൽ നിന്ന് ഒക്ടലിലേക്ക് | ഹെക്സാഡെസിമലിനെ ബൈനറിയിലേക്കും തുടർന്ന് ബൈനറിയിൽ നിന്ന് ഒക്ടലിലേക്കും മാറ്റുക. |

*പട്ടിക 2.6 വിവിധ സംഖ്യാ പരിവർത്തനങ്ങളുടെ നടപടിക്രമങ്ങൾ*

## സ്വയം വിലയിരുത്താം

1 31 എന്ന ദശസംഖ്യ ബൈനറിയിലേക്കു മാറ്റുക.

2 $(10001)_2$ നു തത്തുല്യമായ ദശസംഖ്യ കണ്ടുപിടിക്കുക.

3 $(x)_8 = (101011)_2$, ആയാൽ x ന്റെ വില കാണുക.

4 വിട്ട ഭാഗം പൂരിപ്പിക്കുക.

a) ( _____ )$_2$ = $(AB)_{16}$

b) (___D__)$_{16}$ = $(1010\_\_\_\_1000)_2$

c) $0.25_{10}$ = ( _____ )$_2$

5 താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകളിൽ ഏറ്റവും വലിയ സംഖ്യ കണ്ടുപിടിക്കുക.

(i) $(1001)_2$ (ii) $(A)_{16}$ (iii) $(10)_8$ (iv) $(11)_{10}$

## 2.3 ബൈനറി അരിത്തമറ്റിക്

ദശസംഖ്യാ സമ്പ്രദായത്തിലുള്ളത് പോലെ ദ്വയസംഖ്യാ സമ്പ്രദായത്തിലും ഗണിത ക്രിയകൾ ചെയ്യാം. നമ്മൾ രണ്ട് ദശസംഖ്യകൾ കമ്പ്യൂട്ടറിൽ സങ്കലനം (addition) ചെയ്യാൻ നിർദേശം നൽകുമ്പോൾ, കമ്പ്യൂട്ടർ അതിന്റെ തുല്യമായ ബൈനറി സംഖ്യകൾ ആണ് കൂട്ടുന്നത്. ബൈനറി സംഖ്യകളുടെ സങ്കലനവും (addition) വ്യവകലനവും (subtraction) എങ്ങനെയാണ് ചെയ്യുന്നത് എന്ന് നമുക്ക് നോക്കാം.

### 2.3.1 ബൈനറി സംഖ്യകളുടെ സങ്കലനം (Binary addition)

രണ്ട് ബിറ്റുകൾ കൂട്ടുവാനുള്ള നിയമങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നു.

| A | B | തുക | ശിഷ്ടം |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

ഒന്നും ഒന്നും കൂട്ടുമ്പോൾ മാത്രമാണ് ശിഷ്ട (ക്യാരി) ബിറ്റ് 1 ഉണ്ടാകുന്നത് എന്ന് ശ്രദ്ധിക്കുക. മൂന്നു ഒന്നുകൾ കൂട്ടുമ്പോൾ (1+1+1) തുക 1 ഉം ശിഷ്ടം (ക്യാരി) ബിറ്റ് 1 ഉം കിട്ടുന്നു.

**ഉദാഹരണങ്ങൾ:**

ബൈനറി സംഖ്യകളായ 1011 ന്റെയും 1001 ന്റെയും തുക കണ്ടുപിടിക്കുക.

```
  1 0 1 1 +
  1 0 0 1
---------
1 0 1 0 0
```

ബൈനറി സംഖ്യകളായ 110111 ന്റെയും 100110 ന്റെയും തുക കണ്ടുപിടിക്കുക.

```
  1 1 0 1 1 1 +
  1 0 0 1 1 0
-------------
1 0 1 1 1 0 1
```

### 2.3.2 ബൈനറി സംഖ്യകളുടെ വ്യവകലനം (Binary subtraction)

ഒരു ബൈനറി ബിറ്റിൽ നിന്ന് മറ്റൊരു ബൈനറി ബിറ്റ് കുറയ്ക്കുവാനുള്ള നിയമം താഴെ കൊടുത്തിരിക്കുന്നു.

| A | B | വ്യത്യാസം | കടം |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

0 ൽ നിന്ന് 1 കുറച്ചാൽ വ്യത്യാസം 1 ആണ്, എന്നാൽ ആദ്യത്തെ ബിറ്റിന്റെ ഇടതു ഭാഗത്ത് തൊട്ടടുത്തുള്ള ബിറ്റിൽ നിന്ന് 1 കടമെടുക്കുന്നു. വലിയ ബൈനറി സംഖ്യയിൽ നിന്ന് ഒരു ചെറിയ ബൈനറി സംഖ്യ കുറയ്ക്കുവാൻ മാത്രമേ മേൽപ്പറഞ്ഞ നിയമങ്ങൾ ഉപയോഗിക്കുവാൻ സാധിക്കുകയുള്ളു.

**ഉദാഹരണങ്ങൾ:**

$(10101)_2$ ൽ നിന്നും $(11111)_2$.

$$\begin{array}{r} 1\,1\,1\,1\,1 \\ -\ 1\,0\,1\,0\,1 \\ \hline 0\,1\,0\ \ 1\,0 \end{array}$$

$(10111)_2$ ൽ നിന്നും $(101000)_2$.

$$\begin{array}{r} 1\,0\,1\,0\,0\,0 \\ -\ \ 1\,0\,1\,1\,1 \\ \hline 1\,0\,0\,0\,1 \end{array}$$

## 2.4 ഡാറ്റയുടെ പ്രതിനിധാനം (Data representation)

സംഖ്യകൾ, അക്ഷരങ്ങൾ, ചിത്രങ്ങൾ, ശബ്ദങ്ങൾ, വീഡിയോകൾ തുടങ്ങിയ ഡാറ്റയെ കമ്പ്യൂട്ടറുകളിൽ നിശ്ചിത ബിറ്റുകളുടെ കൂട്ടങ്ങളായിട്ടാണ് പ്രതിനിധാനം ചെയ്യുന്നത്. എന്നിരുന്നാലും എല്ലാതരം ഡാറ്റകളെയും കമ്പ്യൂട്ടർ പ്രതിനിധാനം ചെയ്യുന്നതും പ്രോസസ്സ് ചെയ്യുന്നതും നിശ്ചിത എണ്ണം ബിറ്റുകളായിട്ടാണ്. ഒരു കമ്പ്യൂട്ടറിൽ ആന്തരികമായി ഡാറ്റയെ പ്രതിനിധീകരിക്കുന്നതിന് ഉപയോഗിക്കുന്ന രീതിയാണ് ഡാറ്റ പ്രതിനിധാനം. കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ വ്യത്യസ്ത തരത്തിലുള്ള ഡാറ്റ എങ്ങനെ പ്രതിനിധീകരിക്കുന്നു എന്ന് നമുക്ക് നോക്കാം.

### 2.4.1 സംഖ്യകളുടെ പ്രതിനിധാനം (Representation of numbres)

സംഖ്യകളെ പൂർണസംഖ്യകൾ, ദശാംശസംഖ്യകൾ എന്നിങ്ങനെ രണ്ടായി തിരിക്കാം. പൂർണസംഖ്യകൾ ഭിന്നസംഖ്യാ ഭാഗം ഇല്ലാത്ത സംഖ്യകൾ ആകുന്നു. ദശാംശസംഖ്യ (Floating Point Number) അല്ലെങ്കിൽ രേഖീയസംഖ്യ ഭിന്നഭാഗത്തോട് കൂടിയ സംഖ്യ ആകുന്നു. ഈ രണ്ടു സംഖ്യകളെയും കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ വ്യത്യസ്തമായിട്ടാണ് കൈകാര്യം ചെയ്യുന്നത്. പൂർണസംഖ്യകൾ എങ്ങനെയാണ് മെമ്മറിയിൽ പ്രതിനിധാനം ചെയ്യുന്നത് എന്ന് നമുക്ക് നോക്കാം.

### എ. പൂർണസംഖ്യകളുടെ പ്രതിനിധാനം (Representation of integers)

ഒരു പൂർണ സംഖ്യ കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കുന്നത് മൂന്ന് രീതിയിലാണ്.

i) ചിഹ്നവും മൂല്യവും കൊണ്ടുള്ള പ്രതിനിധാനം (Sign and magnitude representation)

ii) 1 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (1's complement representation)

iii) 2 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (2's complement representation)

കമ്പ്യൂട്ടർ പ്രോസസ്സർ ഒരു യൂണിറ്റായി കൈകാര്യം ചെയ്യുന്ന നിശ്ചിത വ്യാപ്തിയിലുള്ള ഒരു കൂട്ടം ബിറ്റുകളെയാണ് പദം (Word) എന്ന് പറയുന്നത്. ഒരു പദത്തിലെ ബിറ്റുകളുടെ എണ്ണത്തെ പദദൈർഘ്യം (Word length) എന്ന് പറയുന്നു. കമ്പ്യൂട്ടർ രൂപകൽപ്പന ചെയ്യുന്ന വിദഗ്ധരാണ് അതിന്റെ പദദൈർഘ്യം തീരുമാനിക്കുന്നത്. 8, 16, 32, 64 എന്നിവ സാധാരണയായി നിലവിലുള്ള

ചില പദദൈർഘ്യങ്ങളാണ്. പദങ്ങൾ ബിറ്റുകളുടെ കൂട്ടമായതുകൊണ്ട് പദദൈർഘ്യം രണ്ടിന്റെ കൃത്യങ്കങ്ങൾ ആയിരിക്കും.

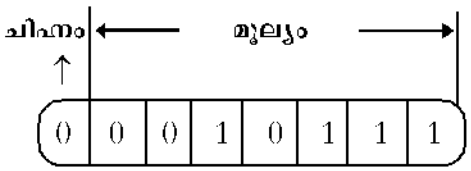
ഇനി ഡാറ്റയെ പ്രതിനിധാനം ചെയ്യുന്ന രീതികൾ (8 ബിറ്റ് പദദൈർഘ്യത്തെ അടിസ്ഥാനമാക്കി) വിശദമായി പരിശോധിക്കാം.

### i. ചിഹ്നവും മൂല്യവും കൊണ്ടുള്ള പ്രതിനിധാനം (Sign and magnitude representation)

ഈ രീതിയിൽ, ഇടതുഭാഗത്തെ ആദ്യത്തെ ബിറ്റ് (MSB) പൂർണസംഖ്യയുടെ ചിഹ്നത്തെയും ബാക്കിയുള്ള 7 ബിറ്റുകൾ സംഖ്യയുടെ മൂല്യത്തെയും പ്രതിനിധാനം ചെയ്യുന്നു. ചിഹ്നത്തെ പ്രതിനിധാനം ചെയ്യുന്ന ബിറ്റ് 1 ആണെങ്കിൽ അത് നെഗറ്റീവ് പൂർണസംഖ്യയും 0 ആണെങ്കിൽ പോസിറ്റീവ് പൂർണസംഖ്യയുമായിരിയ്ക്കും.

**ഉദാഹരണങ്ങൾ:**

+ 23 നെ ചിഹ്നവും മൂല്യവും ഉപയോഗിച്ച് പ്രതിനിധാനം ചെയ്യുക.

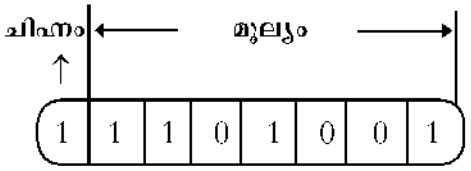



സംഖ്യ പോസിറ്റീവ് ആയതിനാൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 0 ആകുന്നു.

23 ന് തുല്യമായ 7 ബിറ്റ് ബൈനറി സംഖ്യ= $(0010111)_2$

അതുകൊണ്ട് +23 നെ $(00010111)_2$ കൊണ്ട് പ്രതിനിധീകരിക്കാം.

-105 നെ ചിഹ്നവും മൂല്യവും രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക




സംഖ്യ നെഗറ്റീവ് ആയതിനാൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 1 ആകുന്നു.

7 ബിറ്റ് ബൈനറി സംഖ്യ 105 = $(1101001)_2$

-105 ന് തുല്യമായ 7 ബിറ്റ് ബൈനറി സംഖ്യ = $(11101001)_2$

അതിനാൽ -105 നെ $(11101001)_2$ കൊണ്ട് പ്രതിനിധീകരിക്കാം

**കുറിപ്പ്** : ഈ രീതിയിൽ 8 ബിറ്റ് പദം കൊണ്ട് $2^8-1 = 255$ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ കഴിയുന്നു. സംഖ്യകൾ $-(2^7 - 1)$ മുതൽ $+(2^7 - 1)$ വരെ ആയിരിക്കും. (അതായത് -127 മുതൽ +127 വരെ). അതുപോലെ 16 ബിറ്റ് പദം കൊണ്ട് $2^{16} - 1 = 65535$ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാനും കഴിയുന്നു (അതായത് -32767 മുതൽ +32767 വരെ). പൊതുവായി, *n* ബിറ്റ് പദം കൊണ്ട് $2^n-1$ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ കഴിയും (അതായത് $-(2^{n-1}-1)$ മുതൽ $+(2^{n-1}-1)$. വരെ). പൂർണസംഖ്യയായ പൂജ്യത്തെ +0 = 00000000 എന്നും 0 = 10000000 എന്നും രണ്ട് രീതിയിൽ പ്രതിനിധാനം ചെയ്യാം.

### ii. 1 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (1's complement representation)

ഈ രീതിയിൽ, പൂർണസംഖ്യയുടെ കേവല വിലയ്ക്ക് തത്തുല്യമായ 8 ബിറ്റ് ബൈനറി സംഖ്യ കണ്ടുപിടിക്കുന്നു. ബൈനറി സംഖ്യയ്ക്ക് 8 ബിറ്റുകൾ ഇല്ലെങ്കിൽ ഇടതുവശത്ത് ആവശ്യമായ പൂജ്യം ചേർത്ത് 8 ബിറ്റ് സംഖ്യ ആക്കുക. സംഖ്യയിലെ ഓരോ പൂജ്യത്തിനും പകരം ഒന്ന് എന്നും ഓരോ

ഒന്നിന് പകരം പൂജ്യം എന്നും മാറ്റി എഴുതിയാൽ ആ സംഖ്യയുടെ 1 ന്റെ പൂരകം ലഭിക്കും. ചില ബൈനറി സംഖ്യകളും അവയുടെ 1ന്റെ പൂരക പ്രതിനിധാനങ്ങളും താഴെ കൊടുത്തിരിക്കുന്നു.

| പൂർണസംഖ്യ | ബൈനറി സംഖ്യ | 1 ന്റെ പൂരക പ്രതിനിധാനം |
|---|---|---|
| +25 | 00011001 | 00011001 |
| - 25 | 00011001 | 11100110 |

സംഖ്യ നെഗറ്റീവ് ആണെങ്കിൽ അതിന്റെ തത്തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപത്തെ 1 ന്റെ പൂരകമായി പ്രതിനിധീകരിക്കുന്നു. എന്നാൽ സംഖ്യ പോസിറ്റീവ് ആണെങ്കിൽ സംഖ്യയുടെ 8 ബിറ്റ് പ്രതിനിധാനവും 1ന്റെ പൂരക പ്രതിനിധാനവും ഒരു പോലെയായിരിക്കും.

**ഉദാഹരണങ്ങൾ: -**

119 നെ 1ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക.

119 ന്റെ 8 ബിറ്റ് ബൈനറി രൂപം = $(01110111)_2$

-119 ന്റെ 1ന്റെ പൂരക പ്രതിനിധാന രൂപം = $(10001000)_2$

+119 നെ 1ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക

119 ന്റെ 8 ബിറ്റിൽ ഉള്ള ബൈനറി രൂപം = $(01110111)_2$

+119 ന്റെ 1ന്റെ പൂരക പ്രതിനിധാന രൂപം = $(01110111)_2$

*(സംഖ്യ പോസിറ്റീവ് ആയതിനാൽ 1 ന്റെ പൂരക പ്രതിനിധാനം കണ്ടുപിടിക്കേണ്ടതില്ല)*

**കുറിപ്പ്** : ഇത്തരം പ്രതിനിധീകരണത്തിൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 0 ആണെങ്കിൽ സംഖ്യ പോസിറ്റീവും MSB 1 ആണെങ്കിൽ സംഖ്യ നെഗറ്റീവും ആയിരിക്കും. 8 ബിറ്റ് പദദൈർഘ്യം കൊണ്ട് -127 (10000000) മുതൽ +127 (01111111) വരെ പ്രതിനിധാനം ചെയ്യാൻ കഴിയുന്നു. ഈ സംവിധാനത്തിലൂടെ പൂജ്യത്തിനെ +0 = 00000000 എന്നും -0 = 11111111 എന്നും രണ്ട് രീതിയിൽ പ്രതിനിധാനം ചെയ്യാം. പൊതുവായി, *n* ബിറ്റ് പദം കൊണ്ട് $2^n$-1 സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ കഴിയും (അതായത് – $(2^{n-1} - 1)$ മുതൽ $+(2^{n-1} - 1)$ വരെ.

### iii. 2 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (2's complement representation)

ഈ രീതിയിൽ, പൂർണസംഖ്യയുടെ കേവലവിലയ്ക്ക് തത്തുല്യമായ 8 ബിറ്റ് ബൈനറി സംഖ്യ കണ്ടുപിടിക്കുന്നു. സംഖ്യ നെഗറ്റീവ് ആണെങ്കിൽ 8 ബിറ്റ് ബൈനറിയുടെ 2 ന്റെ പൂരകരൂപത്തിൽ അതിനെ പ്രതിനിധാനം ചെയ്യുന്നു. എന്നാൽ സംഖ്യ പോസിറ്റീവ് ആണെങ്കിൽ 8 ബിറ്റ് ബൈനറി സംഖ്യ തന്നെയാണ് അതിന്റെ 2 ന്റെ പൂരക പ്രതിനിധാനം. ഒരു ബൈനറി സംഖ്യയുടെ 1 ന്റെ പൂരകത്തോട് 1 കൂട്ടിയാൽ അതിന്റെ 2 ന്റെ പൂരകം കിട്ടുന്നു.

ഉദാഹരണമായി നമുക്ക് $(10101)_2$ ന്റെ 2 ന്റെ പൂരകം കണ്ടുപിടിക്കാം.

$(00010101)_2$ ന്റെ 1 ന്റെ പൂരകം = $(11101010)_2$

$(10101)_2$ ന്റെ 2 ന്റെ പൂരകം = 11101010 +

1

= $(11011010)_2$

**ഉദാഹരണങ്ങൾ:**

-38 നെ 2ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക.

38 ന്റെ 8 ബിറ്റിലുള്ള ബൈനറി രൂപം = $(00100110)_2$

-38 ന്റെ 2 ന്റെ പൂരക പ്രതിനിധാനം = 11011001+

1

= $(11011010)_2$

+38 നെ 2ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക..

38 ന്റെ 8 ബിറ്റിലുള്ള ബൈനറി രൂപം = $(00100110)_2$

+38 ന്റെ 2 ന്റെ പൂരക പ്രതിനിധാനം = $(00100110)_2$ (സംഖ്യ പോസിറ്റീവ് ആയതിനാൽ 2ന്റെ പൂരക പ്രതിനിധാനം കണ്ടുപിടിക്കേണ്ടതില്ല)

**കുറിപ്പ്** : ഈ രീതിയിൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 0 ആണെങ്കിൽ സംഖ്യ പോസിറ്റീവും MSB 1 ആണെങ്കിൽ സംഖ്യ നെഗറ്റീവും ആയിരിക്കും. ഇവിടെ പൂജ്യം എന്ന പൂർണസംഖ്യ 00000000 എന്ന രീതിയിൽ മാത്രമേ പ്രതിനിധീകരിക്കുവാൻ കഴിയൂ. 8 ബിറ്റ് പദം കൊണ്ട് -128 (10000000) മുതൽ +127 (01111111) വരെ പ്രതിനിധാനം ചെയ്യാൻ കഴിയുന്നു. പൊതുവായി, *n* ബിറ്റ് പദം കൊണ്ട് $2^n$ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുവാൻ കഴിയും. സംഖ്യകൾ $-(2^{n-1})$ മുതൽ $+(2^{n-1}-1)$. വരെ ആകുന്നു. ഈ രീതിയാണ് പൂർണസംഖ്യ പ്രതിനിധീകരിക്കുന്നതിന് സർവസാധാരണമായി ഉപയോഗിക്കുന്നത്. പട്ടിക 2.7 ൽ പൂർണസംഖ്യകളെ 8 ബിറ്റ് പദദൈർഘ്യത്തിൽ പ്രതിനിധീകരിക്കുന്നതിനുള്ള വിവിധ രീതികൾ താരതമ്യം ചെയ്യുന്നു.

| സവിശേഷത | ചിഹ്നവും മൂല്യവും | 1 ന്റെ പൂരകം | 2 ന്റെ പൂരകം | കുറിപ്പ് |
|---|---|---|---|---|
| പരിധി | -127 മുതൽ +127 വരെ | -127 മുതൽ +127 വരെ | -128 മുതൽ +127 വരെ | 2 ന്റെ പൂരകത്തിൽ പരിധി കൂടുതലാണ് |
| ആകെ സംഖ്യകൾ | 255 | 255 | 256 | |
| 0 ന്റെ പ്രതിനിധാനം | 2 രീതിയിലുള്ള പ്രതിനിധാനം | 2 രീതിയിലുള്ള പ്രതിനിധാനം | ഒരേയൊരു രീതിയിലുള്ള പ്രതിനിധാനം | പൂജ്യത്തെ 2 ന്റെ പൂരകത്തിൽ പ്രതിനിധീകരിക്കുന്നതിന് ഒരു അവ്യക്തതയും ഇല്ല. |
| പോസിറ്റീവ് സംഖ്യകളുടെ പ്രതിനിധാനം | സംഖ്യയ്ക്ക് തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപം | സംഖ്യയ്ക്ക് തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപം | സംഖ്യയ്ക്ക് തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപം | മൂന്നു രീതിയിലും ഒരേ പോലെയാണ് |
| നെഗറ്റീവ് സംഖ്യകളുടെ പ്രതിനിധാനം | ചിഹ്നം 1 ബിറ്റിലും മൂല്യം 7 ബിറ്റ് ബൈനറി രൂപത്തിലും പ്രതിനിധീകരി ക്കുന്നു | 8 ബിറ്റ് ബൈനറി രൂപത്തിലാക്കിയ ശേഷം അതിന്റെ 1 ന്റെ പൂരകം കാണുന്നു. | 8 ബിറ്റ് ബൈനറി രൂപത്തിലാക്കിയശേഷം അതിന്റെ 2 ന്റെ പൂരകം കാണുന്നു. | എല്ലാ നെഗറ്റീവ് സംഖ്യകളുടെയും MSB 1 ആകുന്നു |

*പട്ടിക 2.7 ൽ പൂർണ സംഖ്യകളുടെ 8 ബിറ്റ് പദദൈർഘ്യത്തിലുള്ള വിവിധ പ്രതിനിധാനങ്ങളുടെ താരതമ്യം*

താഴെ കൊടുത്തിരിക്കുന്ന പട്ടികയിൽ 4 ബിറ്റ് പദദൈർഘ്യം ഉപയോഗിച്ച് പൂർണ സംഖ്യകളുടെ 3 രീതിയിലുള്ള പ്രതിനിധാനങ്ങൾ വിശദീകരിച്ചിരിക്കുന്നു.

| സംഖ്യ | ചിഹ്നവും മൂല്യവും | 1 ന്റെ പൂരകം | 2 ന്റെ പൂരകം |
|---|---|---|---|
| -8 | സാധ്യമല്ല | സാധ്യമല്ല | 1000 |
| -7 | 1111 | 1000 | 1001 |
| -6 | 1110 | 1001 | 1010 |
| -5 | 1101 | 1010 | 1011 |
| -4 | 1100 | 1011 | 1100 |
| -3 | 1011 | 1100 | 1101 |
| -2 | 1010 | 1101 | 1110 |
| -1 | 1001 | 1110 | 1111 |
| 0 | 1000 അല്ലെങ്കിൽ 0000 | 0000 അല്ലെങ്കിൽ 1111 | 0000 |
| 1 | 0001 | 0001 | 0001 |
| 2 | 0010 | 0010 | 0010 |
| 3 | 0011 | 0011 | 0011 |
| 4 | 0100 | 0100 | 0100 |
| 5 | 0101 | 0101 | 0101 |
| 6 | 0110 | 0110 | 0110 |
| 7 | 0111 | 0111 | 0111 |

പൂർണ സംഖ്യകളുടെ മൂന്നു രീതിയിലുള്ള പ്രതിനിധാനത്തിലും MSB സംഖ്യയുടെ ചിഹ്നം സൂചിപ്പിക്കുന്നു. ബിറ്റ് 1 ആണെങ്കിൽ സംഖ്യ നെഗറ്റീവും ബിറ്റ് 0 ആണെങ്കിൽ സംഖ്യ പോസിറ്റീവും ആണ്. തന്നിരിക്കുന്ന പദദൈർഘ്യം കൊണ്ട് സംഖ്യകളെ ഏറ്റവും കൂടുതൽ പ്രതിനിധീകരിക്കുവാൻ സാധിക്കുന്നത് 2 ന്റെ പൂരക രീതിയിലാണെന്ന് പട്ടികയിൽ കാണുന്നു. 4 പദദൈർഘ്യം ഉപയോഗിച്ചാൽ 7 നെക്കാൾ ചെറുതും +7 നെക്കാൾ വലുതും ആയ സംഖ്യകൾ പ്രതിനിധീകരിക്കാൻ ചിഹ്നവും മൂല്യവും രീതിയിലും 1 ന്റെ പൂരക രീതിയിലും സാധ്യമല്ല. അതുകൊണ്ട് 8 ബിറ്റ് പദദൈർഘ്യമുള്ള പ്രതിനിധാനം ഉപയോഗിക്കുന്നു. അതുപോലെ 2 ന്റെ പൂരക പ്രതിനിധാന രീതിയിൽ -8 മുതൽ +7 പരിധിക്ക് പുറത്തുള്ള സംഖ്യകൾ കൈകാര്യം ചെയ്യുന്നതിനായി 8 ബിറ്റ് ആവശ്യമാണ്.

8 ബിറ്റ് പദദൈർഘ്യം ഉപയോഗരീതിയിൽ -128 മുതൽ +127 വരെയുള്ള സംഖ്യകൾ 2 ന്റെ പൂരക രീതിയിൽ പ്രതിനിധീകരിക്കാം. എന്നാൽ മറ്റു രണ്ടു രീതികളായ 1 ന്റെ പൂരകത്തിലും ചിഹ്നവും മൂല്യത്തിലും -127 മുതൽ +127 വരെ പരിധിയുള്ള സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ സാധിക്കുകയുള്ളൂ. മേൽപ്പറഞ്ഞ പരിധിക്ക് പുറത്തുള്ള സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ നമ്മൾ 16 ബിറ്റ് ഉപയോഗിക്കുന്നു.

## പൂരകം ഉപയോഗിച്ചുള്ള വ്യവകലനം (Subtraction using complements)

ഒരു ബൈനറി സംഖ്യയിൽ നിന്ന് മറ്റൊരു ബൈനറി സംഖ്യ വ്യവകലനം ചെയ്യുന്ന രീതി നമ്മൾ ചർച്ച ചെയ്തു. പക്ഷേ, ഈ രീതിയിലുള്ള ബൈനറി വ്യവകലനം, ഒരു ഇലക്ട്രോണിക്ക് സർക്യൂട്ട് രൂപകൽപ്പന ചെയ്ത് പ്രാവർത്തികമാക്കുക എന്നത് വളരെ സങ്കീർണവും പ്രയാസമുള്ളതുമാണ്. എന്നാൽ ബൈനറി സങ്കലനത്തിന്റെ ഇലക്ട്രോണിക് സർക്യൂട്ട് വളരെ ലളിതമാണ്. അതിനാൽ വ്യവകലനം സങ്കലനക്രിയ വഴി ചെയ്യുന്നതാകും നല്ലത്. വ്യവകലനം ബൈനറി സംഖ്യയുടെ പൂരകം എന്ന ആശയം ഉപയോഗിച്ച് സങ്കലന ക്രിയയിലൂടെയാണ് ചെയ്യുന്നത്. ഇതിന് രണ്ടു രീതികൾ ഉപയോഗിക്കുന്നു.

1 ന്റെ പൂരകം ഉപയോഗിച്ചുള്ള വ്യവകലനം

ഒരു വലിയ ബൈനറി സംഖ്യയിൽ നിന്ന് ഒരു ചെറിയ ബൈനറി സംഖ്യ കുറയ്ക്കുന്നതിനുള്ള ഘട്ടങ്ങൾ.

**ഘട്ടം 1:** ചെറിയ ബൈനറി സംഖ്യയുടെ ഇടതുവശത്ത് ആവശ്യമായ 0 ചേർത്ത് വലിയ ബൈനറി സംഖ്യയുടെ ബിറ്റുകളുടെ എണ്ണത്തിന് തുല്യമാക്കുക.

**ഘട്ടം 2:** ഏതു സംഖ്യകൊണ്ടാണോ കുറയ്ക്കേണ്ടത് അതിന്റെ 1 ന്റെ പൂരകം കാണുക. (ഇവിടെ ചെറിയ ബൈനറി സംഖ്യ)

**ഘട്ടം 3:** ഏതു സംഖ്യയിൽ നിന്നാണോ കുറക്കേണ്ടത് അതും, 1 ന്റെ പൂരകവും തമ്മിൽ കൂട്ടുക. (ഇവിടെ വലിയ ബൈനറി സംഖ്യ )

**ഘട്ടം 4:** തുകയോട് ശിഷ്ടം വരുന്ന ബിറ്റ് (ക്യാരി) കൂട്ടക്കിട്ടുന്നതാണ് ഉത്തരം.

**ഉദാഹരണം:** 1 ന്റെ പൂരക രീതി ഉപയോഗിച്ച് $(1010)_2$ ൽ നിന്നും $(100)_2$ കുറയ്ക്കുക.

ആദ്യമായി $(100)_2$ നെ നാല് ബിറ്റ് രൂപത്തിലേക്ക് മാറ്റുക $=(0100)_2$

ഏതു സംഖ്യയിൽ നിന്നാണോ കുറക്കേണ്ടത് അതും $(0100)_2$ ന്റെ പൂരക സംഖ്യയും തമ്മിൽ കൂട്ടുക

```
        1010     +
        1011
      1 0101
MSB     0101     +
        -----
കാരി കൂട്ടുക      1
ഉത്തരം         0110
```

MSB ഒഴിവാക്കിക്കൊണ്ട് ഉത്തരത്തോട് കൂട്ടുക.

### 2 ന്റെ പൂരകം ഉപയോഗിച്ചുള്ള വ്യവകലനം

ഒരു വലിയ ബൈനറി സംഖ്യയിൽ നിന്ന് ഒരു ചെറിയ ബൈനറി സംഖ്യ കുറയ്ക്കുന്നതിനുള്ള ഘട്ടങ്ങൾ.

**ഘട്ടം 1:** ചെറിയ ബൈനറി സംഖ്യയുടെ ഇടതുവശത്ത് ആവശ്യമായ 0 ചേർത്ത് വലിയ ബൈനറി സംഖ്യയുടെ ബിറ്റുകളുടെ എണ്ണത്തിന് തുല്യമാക്കുക.

**ഘട്ടം2:** ഏതു സംഖ്യകൊണ്ടാണോ കുറയ്ക്കേണ്ടത് അതിന്റെ 2 ന്റെ പൂരകം കാണുക. (ഇവിടെ ചെറിയ ബൈനറി സംഖ്യ )

**ഘട്ടം3:** ഏതു സംഖ്യയിൽ നിന്നാണോ കുറയ്ക്കേണ്ടത് അതും, 2 ന്റെ പൂരകവും തമ്മിൽ കൂട്ടുക. (ഇവിടെ വലിയ ബൈനറി സംഖ്യ )

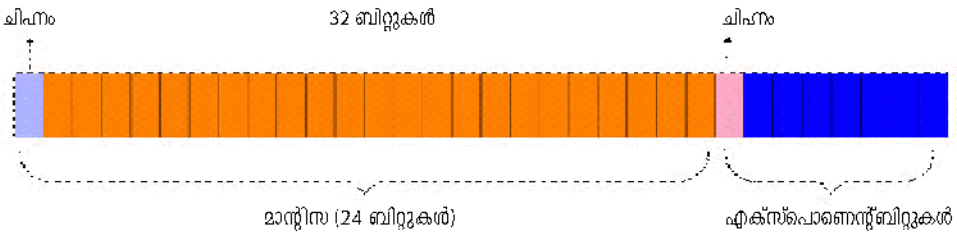
**ഘട്ടം 4:** തുകയിൽ ശിഷ്ടം വരുന്ന ബിറ്റ് (ക്യാരി) ഒഴിവാക്കി കിട്ടുന്നതാണ് ഉത്തരം.

**ഉദാഹരണം:** 2 ന്റെ പൂരക രീതി ഉപയോഗിച്ച് $(1010)_2$ ൽ നിന്നും $(100)_2$ കുറയ്ക്കുക.

| | |
|---|---|
| $(100)_2$ നെ നാല് ബിറ്റ് രൂപത്തിലേക്ക് മാറ്റുക | 0100 |
| $(0100)_2$ ന്റെ 2 ന്റെ പൂരകം കണ്ടുപിടിക്കുക | 1011 + |
| | 1 |
| | 1100 |
| ഏതു സംഖ്യയിൽ നിന്നാണോ കുറക്കേണ്ടത് | 1010 + |
| അതും 2 ന്റെ പൂരക സംഖ്യയും തമ്മിൽ കൂട്ടുക | 1100 |
| | 10110 |
| ശിഷ്ടം ഒഴിവാക്കി കിട്ടുന്നതാണ് ഉത്തരം | |
| ഉത്തരം | 0110 |

## ബി. ഫ്ളോട്ടിംഗ് പോയിന്റ് സംഖ്യകളുടെ പ്രതിനിധാനം (Representation of floating point numbers)

ഒരു ഫ്ളോട്ടിംഗ് പോയിന്റ് സംഖ്യ അല്ലെങ്കിൽ രേഖീയ സംഖ്യയിൽ പൂർണസംഖ്യാഭാഗവും ഭിന്നകഭാഗവും അടങ്ങിയിട്ടുണ്ട്. ഒരു രേഖീയ സംഖ്യയെ ഫ്ളോട്ടിംഗ് പോയിന്റ് എന്ന സവിശേഷമായ ചിഹ്നസമ്പ്രദായം ഉപയോഗിച്ച് എഴുതാവുന്നതാണ്. ഈ ചിഹ്നസമ്പ്രദായം ഉപയോഗിച്ച് എഴുതുമ്പോൾ ഏത് സംഖ്യയ്ക്കും മാന്റിസ, എക്സ്പൊണെന്റ് എന്നീ രണ്ട് ഭാഗങ്ങൾ ഉണ്ടാകും. ഉദാഹരണമായി 25.45 നെ $0.2545 \times 10^2$ എന്നെഴുതാം. ഇതിൽ 0.2545 എന്നത് മാന്റിസയും കൃത്യങ്കം 2 എന്നത് എക്സ്പൊണെന്റുമാണ്. (ക്രമാനുസൃതമായ (Normalised) ഫ്ളോട്ടിംഗ് പോയിന്റ് പ്രതിനിധാനത്തിൽ മാന്റിസ 0.1നും 1നും ഇടയിലായിരിക്കും). അതുപോലെ 0.0035 എന്ന സംഖ്യ $-0.35 \times 10^{-2}$ എന്ന് എഴുതാം. ഇവിടെ -0.35 എന്നത് മാന്റിസയും കൃത്യങ്കം -2 എന്നത് എക്സ്പൊണെന്റുമാണ്.




*ചിത്രം 2.3 ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യകളുടെ പ്രതിനിധാനം*

32 ബിറ്റ് പദദൈർഘ്യമുള്ള കമ്പ്യൂട്ടറിൽ ഒരു രേഖീയ സംഖ്യ എങ്ങനെയാണ് പ്രതിനിധാനം ചെയ്യുന്നതെന്ന് നോക്കാം. ചിത്രം 2.3 ൽ കാണുന്നതുപോലെ, ഇതിൽ 24 ബിറ്റുകൾ മാന്റിസ

*രേഖീയസംഖ്യകളിൽ ബൈനറി അംശബിന്ദു മാന്റിസ, എക്സ്പൊണെന്റ് ഭാഗങ്ങളുടെ വിവരങ്ങൾ സൂക്ഷിക്കുന്നു. ബൈനറി അംശബിന്ദുവിന്റെ സ്ഥാനം സ്ഥിരമല്ലാത്തതിനാൽ മാന്റിസ എക്സ്പൊണെന്റ് എന്നിവയുടെ വിലകൾ സംഖ്യകൾ തോറും മാറുന്നു. മറ്റൊരു വിധത്തിൽപ്പറഞ്ഞാൽ അത് ഫ്ളോട്ട് ചെയ്യുകയാണ് (വെള്ളത്തിൽ പൊങ്ങിക്കിടക്കുന്നതുപോലെ) അതിനാൽ ഈ പ്രതിനിധാനത്തെ ഫ്ളോട്ടിംഗ് പോയിന്റ് പ്രതിനിധാനം എന്നറിയപ്പെടുന്നു.*

രേഖപ്പെടുത്താനും (അതിൽ ആദ്യത്തെ ബിറ്റ് മാന്റിസയുടെ ചിഹ്നത്തിനുവേണ്ടിയാണ്), 8 ബിറ്റുകൾ എക്സ്പൊണെന്റ് രേഖപ്പെടുത്താനും (അതിൽ ആദ്യത്തെ ബിറ്റ് എക്സ്പൊണെന്റിന്റെ ചിഹ്നത്തിനുവേണ്ടി) ഉപയോഗിക്കുന്നു. ദശാംശബിന്ദു മാന്റിസയുടെ ചിഹ്നം സൂചിപ്പിക്കുന്ന ബിറ്റിന്റെ വലത് ഭാഗത്താണെന്ന് അനുമാനിക്കുക. ദശാംശസ്ഥാനം സാങ്കല്പികമായതിനാൽ അത് രേഖപ്പെടുത്താൻ പ്രത്യേകമായി ബിറ്റുകൾ ആവശ്യമില്ല.

ഉദാഹരണമായി 25.45 എന്ന രേഖീയ സംഖ്യ മാന്റിസ എക്സ്പൊണെന്റ് രീതിയിൽ $0.2545 \times 10^2$ എന്ന് എഴുതാം. ഇവിടെ മാന്റിസയായ 0.2545 നെയും എക്സ്പൊണെന്റായ 2 നെയും ബൈനറി രൂപത്തിലേക്കു മാറ്റി അവയെ അതാതു സ്ഥാനങ്ങളിൽ രേഖപ്പെടുത്തുന്നു. മാന്റിസയും എക്സ്പൊണെന്റും രേഖപ്പെടുത്താൻ വ്യത്യസ്തങ്ങളായ മാനദണ്ഡങ്ങൾ ഉപയോഗിക്കുന്നു. പദദൈർഘ്യം മാറുന്നതിനനുസരിച്ച് മാന്റിസയും എക്സ്പൊണെന്റും രേഖപ്പെടുത്താൻ ഉപയോഗിക്കുന്ന ബിറ്റുകളുടെ എണ്ണത്തിലും മാറ്റം ഉണ്ടാകും.

## 2.4.2 അക്ഷരങ്ങളുടെ പ്രതിനിധാനം (Representation of characters)

കംപ്യൂട്ടറിന്റെ മെമ്മറിയിൽ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുന്നത് എങ്ങനെയെന്ന് നമ്മൾ കണ്ടു. അതുപോലെ അക്ഷരങ്ങളെ (Characters) പ്രതിനിധാനം ചെയ്യുന്നതിന് വ്യത്യസ്തങ്ങളായ സമ്പ്രദായങ്ങളുണ്ട്. അവയിൽ ചിലതിനെക്കുറിച്ച് ചുവടെ പ്രതിപാദിക്കുന്നു.

### എ. ആസ്കി (ASCII)

കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ 7 ബിറ്റുകൾ ഉപയോഗിച്ച് ഓരോ അക്ഷരവും പ്രതിനിധാനം ചെയ്യാൻ ഉപയോഗിക്കുന്ന ASCII (ആസ്കി) കോഡ് Amercican Standard Code for Information Interchange (അമേരിക്കൻ സ്റ്റാൻഡേർഡ് കോഡ് ഫോർ ഇൻഫർമേഷൻ ഇന്റർചേഞ്ച്) എന്നതിന്റെ ചുരുക്ക രൂപമാണ്. അമേരിക്കൻ സർക്കാർ അംഗീകരിച്ച ആസ്കികോഡ് വ്യാപകമായി സ്വീകരിക്കപ്പെട്ടു കഴിഞ്ഞു. ഇതിൽ ഓരോ അക്ഷരത്തിനും വത്യസ്ത പൂർണസംഖ്യ നിശ്ചയിച്ചിരിക്കുന്നു. ആസ്കി കോഡ് എന്ന് വിളിക്കുന്ന ഈ പൂർണസംഖ്യ മെമ്മറിയിൽ സൂക്ഷിക്കുന്നതിനായി ബൈനറി സംഖ്യയിലേക്ക് പരിവർത്തനം ചെയ്യുന്നു. ഉദാഹരണത്തിന് A എന്ന അക്ഷരത്തിന്റെ ആസ്കി കോഡ് 65 ആകുന്നു. ഇതിന് തുല്യമായ 7 ബിറ്റ് ബൈനറി 1000001 ആണ്. 7 ബിറ്റുകൾ കൊണ്ട് വത്യസ്തങ്ങളായ 128 സംയോഗങ്ങൾ (Unique combination) സൃഷ്ടിക്കാനാകും. ആയതിനാൽ 7 ബിറ്റ് ആസ്കി ഉപയോഗിച്ച് 128 അക്ഷരങ്ങളുടെ കോഡുകൾ ഉണ്ടാക്കാം.

ഓരോ അക്ഷരത്തിനും 8 ബിറ്റ് ഉപയോഗിക്കുന്ന ഇതിന്റെ മറ്റൊരു പതിപ്പിനെ ആസ്കി 8 അഥവാ എക്സ്റ്റന്റഡ് ആസ്കി (Extended ASCII) എന്ന് വിളിക്കുന്നു. 8 ബിറ്റ് ആസ്കി കൊണ്ട് 256 വ്യത്യസ്താക്ഷരങ്ങളുടെ കോഡുകൾ ഉണ്ടാക്കാം. ഉദാഹരണമായി A എന്ന അക്ഷരത്തെ 01000001 എന്നും B എന്ന അക്ഷരത്തെ 01000010 എന്നും കോഡ് ചെയ്യപ്പെടുന്നു. സാധാരണ കീബോർഡിലെ മുഴുവൻ അക്ഷരങ്ങൾക്കും കോഡ് നൽകുവാൻ ആസ്കി 8 ന് കഴിയുന്നു.

### ബി. എബ്സിഡിക് (EBCDIC)

എക്സറ്റന്റഡ് ബൈനറി കോഡഡ് ഡെസിമൽ ഇന്റർചേഞ്ച് കോഡ് (Extended Binary Coded Decimal Interchange Code) എന്നതിന്റെ ചുരുക്ക രൂപമാണിത്. ഇന്റർനാഷണൽ ബിസിനസ് മെഷീൻ (ഐ.ബി.എം) നിർമ്മിക്കുന്ന കമ്പ്യൂട്ടറുകളിൽ, ആസ്കിയെ പോലെ ഇതിലും 8 ബിറ്റ് കോഡ് ഉപയോഗിക്കുന്നു. ഇതുപയോഗിച്ച് 256 അക്ഷരങ്ങൾക്ക് കോഡ് നൽകാനാവും. ആസ്കിയിൽ കോഡ് ചെയ്യപ്പെട്ട ഡാറ്റ എബ്സിഡിക് കോഡ് ഉപയോഗിക്കുന്ന കമ്പ്യൂട്ടറിൽ ഉപയോഗിക്കണമെങ്കിൽ ആസ്കി കോഡിൽ നിന്ന് എബ്സിഡിക് കോഡിലേക്ക് മാറ്റേണ്ടതുണ്ട്. അതുപോലെ, എബ്സിഡിക് കോഡ് ഉപയോഗിച്ചുണ്ടാക്കിയ ഡാറ്റ ആസ്കി കമ്പ്യൂട്ടറിൽ ഉപയോഗിക്കണമെങ്കിൽ, ആസ്കിയിലേക്കും മാറ്റേണ്ടതുണ്ട്.

### സി. ഇസ്കി (ISCII)

ഇന്ത്യൻ സ്റ്റാൻഡേർഡ് കോഡ് ഫോർ ഇൻഫർമേഷൻ ഇന്റർചേഞ്ച് (Indian Standard Code for Information Interchage) അല്ലെങ്കിൽ ഇന്ത്യൻ സ്ക്രിപ്റ്റ് കോഡ് ഫോർ ഇൻഫർമേഷൻ ഇന്റർചേഞ്ച് (Indian Script Code for Information Interchage) എന്നതിന്റെ ചുരുക്കരൂപമാണിത്. വിവിധ ഇന്ത്യൻഭാഷകളിലെ അക്ഷരങ്ങളുടെ എൻകോഡിംഗ് (Encoding) വ്യവസ്ഥയാണിത്. 8 ബിറ്റ് ഉപയോഗിച്ചാണ് ഇസ്കി ഡാറ്റ പ്രതിനിധാനം ചെയ്യുന്നത്. 1986 ൽ ഇലക്ട്രോണിക് വകുപ്പിന് കീഴിലുള്ള നിലവാരം നിശ്ചയിക്കൽ സമിതി ചിട്ടപ്പെടുത്തിയ ഈ വ്യവസ്ഥ ബ്യൂറോ ഓഫ് ഇന്ത്യൻ സ്റ്റാൻഡേർഡ്സ് (BIS) അംഗീകരിച്ചതാണ്. ഇസ്കിക്ക് പകരം യൂനിക്കോഡാണ് ഇപ്പോൾ ഉപയോഗിക്കുന്നത്.

### ഡി. യൂണികോഡ് (Unicode)

8 ബിറ്റുകൾ ഉപയോഗിക്കുന്ന ആസ്കിക്ക് 256 അക്ഷരങ്ങൾ മാത്രമേ പ്രതിനിധാനം ചെയ്യാനാകൂ. ലോകം മുഴുവനുമുള്ള ലിഖിതഭാഷകളിലെ അക്ഷരങ്ങളെയും ചിഹ്നങ്ങളേയും പ്രതിനിധാനം ചെയ്യാൻ ഇത് മതിയാകില്ല. ഈ പ്രശ്നം പരിഹരിക്കാനാണ് യൂണിക്കോഡ് വികസിപ്പിച്ചെടുത്തത്. ആഗോളവും കാര്യക്ഷമവും നിലവാരമുള്ളതും ആയ അക്ഷരങ്ങളുടെ എൻകോഡിങ് രീതിയാണ് ഇതിന്റെ ലക്ഷ്യം. ഏത് ഭാഷയായാലും ഏത് പ്ലാറ്റ്ഫോമായാലും (Platform) അവയ്ക്കെല്ലാം വ്യത്യസ്തമായ ഒരക്കം ഇത് നൽകുന്നു.

യൂണിക്കോഡിൽ മൗലികമായി 16 ബിറ്റുകളാണ് ഉപയോഗിക്കുന്നത്. അതിന് 65,536 അക്ഷരങ്ങൾ പ്രതിനിധീകരിക്കാൻ കഴിയും. യൂണിക്കോഡ് കൺസോർഷ്യം എന്ന ലാഭേച്ഛയില്ലാത്ത സംഘടനയാണ് ഇത് ചിട്ടപ്പെടുത്തുന്നത്. കൺസോർഷ്യം 1991 ൽ ആദ്യപതിപ്പായ 1.0.0 പ്രസിദ്ധീകരിച്ചു. അതിനെ അടിസ്ഥാനമാക്കി നിലവാരം മെച്ചപ്പെടുത്തുന്നതിനുള്ള ശ്രമം തുടരുകയാണ്. ഈ കാലയളവിൽ യൂണികോഡ് ഉപയോഗിക്കുന്നത് 16ൽ അധികം ബിറ്റുകളാണ്. അതിനാൽ ധാരാളം അക്ഷരങ്ങളെ പ്രതിനിധാനം ചെയ്യാൻ അതിന് സാധിക്കും. ലോകത്തിലെ എല്ലാ ലിഖിത ഭാഷകളുടെയും അക്ഷരങ്ങളെ പ്രതിനിധാനം ചെയ്യുവാൻ യൂണിക്കോഡിന് സാധിക്കുന്നു.

## 2.4.3 ശബ്ദ, ചിത്രം, വീഡിയോ എന്നിവയുടെ പ്രതിനിധാനം (Representation of audio, image and video)

ഇതിന് മുമ്പുള്ള ഭാഗത്തിൽ അക്കങ്ങളും അക്ഷരങ്ങളും ഉൾപ്പെട്ട വിവരങ്ങൾ കമ്പ്യൂട്ടറിൽ പ്രതിനിധാനം ചെയ്യുന്ന വിധവും അവയുടെ വ്യത്യസ്ത മാനദണ്ഡങ്ങളും നാം പരിചയപ്പെട്ടു. ഡിജിറ്റൽ കമ്പ്യൂട്ടറുകളുടെ സഹായത്തോടെ നിത്യജീവിതത്തിലെ പ്രശ്നങ്ങൾ കൈകാര്യം ചെയ്യുമ്പോൾ മിക്കപ്പോഴും അക്കങ്ങളോ അക്ഷരങ്ങളോ അല്ലാത്ത വിവരങ്ങൾ രേഖപ്പെടുത്തുകയോ പ്രോസസ്സ് ചെയ്യേണ്ടതായോ വരാം. അക്കങ്ങളെയും അക്ഷരങ്ങളെയും പോലെ ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിവയിലും ധാരാളം വിവരങ്ങൾ അടങ്ങിയിട്ടുണ്ട്. ഇവ സംഭരിക്കുന്നതിനുള്ള വിവിധ ഫയൽ ഘടനകളെക്കുറിച്ച് നമുക്ക് ചർച്ച ചെയ്യാം.

### ഡിജിറ്റൽ ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിവയുടെ ഫയൽ ഘടനകൾ (Digital audio, image and video file formats)

ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിങ്ങനെയുള്ള മൾട്ടിമീഡിയ ഡാറ്റ വ്യത്യസ്ത ഫയൽ ഘടനകളിലാണ് സംഭരിക്കുന്നത്. ഡാറ്റയുടെ വലുപ്പം കുറയ്ക്കുന്നതിനും ചുരുക്കുന്നതിനും വിവിധ കെട്ടുകളാക്കുന്നതിനും വിവിധ സമീപനരീതികൾ ഉപയോഗിക്കുമ്പോൾ അവ വ്യത്യസ്ത ഫയൽഘടനയ്ക്ക് കാരണമാകുന്നു. ഉദാഹരണത്തിന് ഒരു ചിത്രം സാധാരണനിലയിൽ ജോയിന്റ് പിക്ചർ എക്സ്പേർട്ട്സ് ഗ്രൂപ്പ് (ജെപെഗ് - JPEG) ഫയൽ ഘടനയിലാണ് സംഭരിക്കുന്നത്. ഈ ചിത്രത്തിന്റെ ഫയലിൽ തലക്കെട്ട് (Header) വിവരങ്ങളും ചിത്രത്തിന്റെ (Image) ഡാറ്റയും അടങ്ങിയിരിക്കുന്നു. ഫയലിന്റെ പേര്, വലുപ്പം, പരിഷ്കരിച്ച ഡാറ്റ, ഫയൽഘടന മുതലായ വിവരങ്ങൾ തലക്കെട്ട് ഭാഗത്താണ് സംഭരിക്കുന്നത്. പിക്സലുകളുടെ തീവ്രതയെക്കുറിച്ചുള്ള വിവരങ്ങൾ ഡാറ്റ ഭാഗത്തും രേഖരിക്കുന്നു.

ഫയലിന്റെ വലുപ്പം കുറയ്ക്കുന്നതിന് ഡാറ്റ ചുരുക്കിയോ അല്ലാതെയോ സംഭരിക്കാം. സാധാരണനിലയിൽ ചിത്രം ഡാറ്റയെ ചുരുക്കിയാണ് സംഭരിക്കുന്നത്. എന്താണ് ചുരുക്കൽ (Compression) എന്ന് നോക്കാം. 400x400 പിക്സൽ വലുപ്പമുള്ള, കറുപ്പ് നിറമുള്ള ഒരു ചിത്രം ഉദാഹരണമായി എടുക്കാം. 1,60,000 (400x400) പിക്സലിലും കറുപ്പ്, കറുപ്പ്, .....കറുപ്പ് എന്നിങ്ങനെ ആവർത്തിച്ച് സംഭരിക്കാം. ഇത് ചുരുക്കാതെയുള്ള രൂപമാണ്. അതേസമയം, കറുപ്പ് എന്ന് ഒരു തവണ രേഖപ്പെടുത്തുകയും 1,60,000 തവണ ആവർത്തനം എന്നും രേഖപ്പെടുത്തുന്നതാണ് ചുരുക്കി സംഭരിക്കൽ. ചുരുക്കലിനായി ഇത്തരം നിരവധി രീതികൾ ഉപയോഗിക്കാറുണ്ട്. ബിറ്റ്മാപ് (BMP), ടാഗ്ഡ് ഇമേജ് ഫയൽ ഫോർമാറ്റ് (TIFF), ഗ്രാഫിക്സ് ഇന്റർചേഞ്ച് ഫോർമാറ്റ് (GIF), പോർട്ടബിൾ പബ്ലിക് നെറ്റ്വർക്ക് ഗ്രാഫ് (PNG) തുടങ്ങി വിവിധ തരത്തിലുള്ള ഫയൽ ഘടനകളിൽ ചിത്രങ്ങൾ ഉപയോഗത്തിനനുസരിച്ച് സംഭരിക്കപ്പെടുന്നു.

ചിത്രത്തിന്റെ കാര്യത്തിൽപ്പറഞ്ഞ തലക്കെട്ട് ഫയൽ വിവരങ്ങൾ, ശബ്ദം, വീഡിയോ എന്നീ ഫയലുകൾക്കും ബാധകമാണ്. WAV, MP3, MIDI, AIFF മുതലായ വ്യത്യസ്ത ഫയൽ ഘടനകളിൽ ഡിജിറ്റൽ ശബ്ദ ഡാറ്റ സംഭരിക്കാൻ കഴിയും. ഡിജിറ്റൽ ശബ്ദഡാറ്റ സംഭരിക്കുന്നതിന് ഒരു ശബ്ദ ഫയൽ ഘടന വിവരിക്കുന്നുണ്ട്. ചില സമയങ്ങളിൽ ഇത് കണ്ടെയ്നർ ഫോർമാറ്റ് (Container Format) എന്ന് സൂചിപ്പിക്കാറുണ്ട്. ഉദാഹരണമായി WAV ചുരുക്കാത്ത ശബ്ദവും, MP3 ഫയലുകളിൽ ചുരുക്കിയ ശബ്ദവുമാണ് ഉൾക്കൊള്ളുക. സംശ്ലേഷണം (synchronous) ചെയ്ത സംഗീത

**സ്വയം വിലയിരുത്താം**

1. -80 നെ ചിഹ്നവും മൂല്യവും രൂപത്തിൽ പ്രതിനിധാനം ചെയ്താൽ അതിന്റെ MSB ഏതാണ്?
2. 28.756 നെ മാന്റിസ എക്സ്പൊണെന്റ് രൂപത്തിൽ എഴുതുക.
3. ASCII യുടെ പൂർണരൂപം എഴുതുക.
4. 60 നെ 1 ന്റെ പൂരകമായി പ്രതിനിധാനം ചെയ്യുക.
5. യൂണികോഡ് നിർവചിക്കുക.
6. ഏതെങ്കിലും രണ്ട് ചിത്രഫയൽ ഘടനകൾ എഴുതുക.

ഡാറ്റ ശേഖരിച്ചുവയ്ക്കുന്ന സംവിധാനമാണ് MIDI (Musical Instrument Digital Interface). അതുപോലെ AVI (Audio Video Interleave) എന്നത് വീഡിയോഫയൽ ശേഖരിച്ചുവയ്ക്കുന്ന മറ്റൊരു സംവിധാനമാണ്. MP3, JPEG-2, WMV എന്നീ ഫയൽ ഘടനകൾ ശബ്ദം, വീഡിയോ എന്നിവ സംഭരിച്ചുവയ്ക്കുന്നതിനും ഒരേ സമയം പ്രവർത്തിപ്പിക്കുന്നതിനും ഉപയോഗിക്കുന്നു.

## 2.5 ബൂളിയൻ ബീജ ഗണിതത്തിന് ഒരാമുഖം (Introduction to Boolean algebra)

ശരി അല്ലെങ്കിൽ തെറ്റ് എന്ന് ഉത്തരം നൽകേണ്ട നിരവധി സന്ദർഭങ്ങൾ നമ്മുടെ ജീവിതത്തിൽ ഉണ്ടാകാറുണ്ട്. അതുപോലെ നമ്മുടെ ചിന്തയുടെ ഒരു വലിയ ഭാഗം ശരി അല്ലെങ്കിൽ തെറ്റ് എന്ന് ഉത്തരം പറയേണ്ടവയാണ്. ഇപ്രകാരം സത്യം കണ്ടെത്തുന്നതിനെ വ്യക്തി വിചിന്തനം (ഹ്യൂമൺ റീസണിങ് - Human Reasoning) അഥവാ യുക്തി വിചിന്തനം ( Logical Reasoning) എന്ന് പറയുന്നു. ശരി അല്ലെങ്കിൽ തെറ്റ് എന്നത് സംഖ്യാപരമായി ഒന്നോ പൂജ്യമോ ആകാം. ഈ രണ്ട് മൂല്യങ്ങളെ ദ്വന്ദമൂല്യങ്ങൾ (Binary Values) അല്ലെങ്കിൽ ബൂളിയൻ മൂല്യങ്ങൾ (Boolean Values) എന്ന് പറയുന്നു. 1 അല്ലെങ്കിൽ 0 എന്നിവയാൽ പ്രതിനിധാനം ചെയ്യപ്പെടുന്ന ചരങ്ങളുമായി (variables) ബന്ധപ്പെട്ട ക്രിയകൾ ചെയ്യുന്ന യുക്തിപരമായ ബീജഗണിതശാസ്ത്രശാഖയാണ് ബൂളിയൻ ബീജഗണിതം (Boolean Algebra). യുക്തിയും ഗണിതശാസ്ത്രവും തമ്മിൽ ബന്ധം സ്ഥാപിച്ച ബ്രിട്ടീഷ് ഗണിതശാസ്ത്രജ്ഞനായ ജോർജ് ബൂളിനെ ആദരിച്ചു കൊണ്ടാണ് ഈ പേര് നൽകിയിരിക്കുന്നത്. അദ്ദേഹത്തിന്റെ An Investigation of the Law of Thought എന്ന വിപ്ലവകരമായ പ്രബന്ധമാണ് ബൂളിയൻ ആൾജിബ്രയുടെ ഉൽപ്പത്തിക്ക് വഴിയൊരുക്കിയത്.

*ചിത്രം 2.4: ജോർജ്ബൂൾ (1815 1864)*

### 2.5.1 ദ്വന്ദമൂല്യമുള്ള അളവുകൾ (Binary Valued Quantities)

താഴെ പറയുന്നവ നോക്കുക.

1. ഞാൻ ഒരു കുട എടുക്കുന്നതല്ലേ നല്ലത്?
2. താങ്കളുടെ പേന എനിക്ക് തരാൻ സമ്മതമാണല്ലോ?

3. ജോർജ് ബൂൾ ഒരു ബ്രിട്ടീഷ് ഗണിതശാസ്ത്രജ്ഞനാണ്.

4. കേരളം ഇന്ത്യയിലെ ഏറ്റവും വലിയ സംസ്ഥാനമാണ്.

5. ഇന്നലെ ഹാജരാകാതിരുന്നത് എന്തുകൊണ്ട് ?

6. ബൂളിയൻ ബീജഗണിതത്തെക്കുറിച്ച് നിങ്ങളുടെ അഭിപ്രായം എന്താണ്?

1 ഉം 2 ഉം വാക്യങ്ങൾ അതെ (YES) അല്ലെങ്കിൽ അല്ല (NO) എന്ന് ഉത്തരം പറയാവുന്ന ചോദ്യങ്ങളാണ്. ഇത്തരം സന്ദർഭങ്ങളെ ബൈനറി തീരുമാനങ്ങൾ എന്നും ഇതിന്റെ ഉത്തരങ്ങളെ ബൈനറി മൂല്യങ്ങൾ (Binary Values) എന്നും വിളിക്കുന്നു. മൂന്നാമത്തെ വാക്യത്തിന്റെ ഉത്തരം ശരി (TRUE) എന്നും നാലാമത്തേതിന്റെ തെറ്റ് (FALSE) എന്നുമാണ്. എന്നാൽ അഞ്ചും ആറും ചോദ്യങ്ങൾക്ക് ഇങ്ങനെ ഉത്തരം പറയാനാകില്ല. TRUE, FALSE എന്ന് ഉത്തരം നൽകാവുന്ന വാക്യങ്ങൾ യുക്തി പ്രസ്താവനകൾ (Logical Statements) അല്ലെങ്കിൽ ട്രൂത്ത് ഫങ്ക്ഷൻ (Truth function) എന്നും അതിന്റെ ഉത്തരമായി നൽകുന്ന TRUE, FALSE എന്നിവയെ ലോജിക്കൽ സ്ഥിരമൂല്യങ്ങൾ (Logical Constants) അല്ലെങ്കിൽ ബൈനറി മൂല്യങ്ങൾ (Binary Values) എന്നും പറയുന്നു. ശരി എന്നത് 1 ഉം തെറ്റ് എന്നത് 0 ഉം ആണ്. ഈ ലോജിക്കൽ കോൺസ്റ്റന്റുകൾ കൈകാര്യം ചെയ്യുന്ന വേരിയബിളിനെ ലോജിക്കൽ വേരിയബിൾ അല്ലെങ്കിൽ ബൂളിയൻ വേരിയബിൾ എന്ന് പറയുന്നു.

### 2.5.2 ബൂളിയൻ ഓപ്പറേറ്ററുകളും ലോജിക്കൽ ഗേറ്റുകളും (Boolean operators and logic gates)

കമ്പ്യൂട്ടറിലേക്ക് നമ്മൾ നൽകുന്ന വിവരങ്ങൾ 1, 0 എന്നിവയുടെ ഗണങ്ങളായി മാറ്റേണ്ടതുണ്ട്. കമ്പ്യൂട്ടറിലെ എല്ലാ ഡാറ്റയും, വിവരങ്ങളും, ക്രിയകളും 0, 1 എന്നീ അക്കങ്ങളിലാണ് പ്രതിനിധാനം ചെയ്യുന്നത്. ബൂളിയൻ മൂല്യങ്ങൾ അടിസ്ഥാനമാക്കി നിർവഹിക്കപ്പെടുന്ന ഈ പ്രവർത്തങ്ങളെ ബൂളിയൻ പ്രവർത്തനങ്ങൾ (Boolean Operation) എന്ന് വിളിക്കുന്നു. നമുക്കറിയാവുന്നതുപോലെ, ഈ ക്രിയകൾ ചെയ്യുവാൻ ഓപ്പറേറ്ററുകൾ ആവശ്യമാണ്. ഇത്തരം ഓപ്പറേറ്ററുകളെ ബൂളിയൻ ഓപ്പറേറ്ററുകൾ അല്ലെങ്കിൽ ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ എന്ന് വിളിക്കുന്നു. ബൂളിയൻ ബീജഗണിതത്തിൽ മൂന്ന് അടിസ്ഥാന ഓപ്പറേറ്ററുകളാണ് ഉള്ളത്. ഈ ഓപ്പറേറ്ററുകളും ബീജഗണിതത്തിൽ അവയുടെ പ്രവർത്തനങ്ങളും ചുവടെ ചേർത്തിരിക്കുന്നു.

OR → യുക്തിപരമായ സങ്കലനം (Logical Addition)

AND → യുക്തിപരമായ വ്യവകലനം (Logical Multiplication)

NOT → ലോജിക്കൽ നിഷേധം (Logical Negation)

ആദ്യത്തെ രണ്ട് ഓപ്പറേറ്റുകളോടൊപ്പം രണ്ട് ഓപ്പറന്റുകളും, മൂന്നാമത്തേതിനോടുകൂടി ഒരു ഓപ്പറന്റും ഉപയോഗിക്കുന്നു. ഇവിടെ ഓപ്പറന്റുകൾ എല്ലായ്പ്പോഴും ബൂളിയൻ വേരിയബിൾ അല്ലെങ്കിൽ കോൺസ്റ്റന്റുകൾ ആയിരിക്കും. ഇത്തരം ഓപ്പറേഷനുകളുടെ ഉത്തരം ഒന്നുകിൽ TRUE (1) അല്ലെങ്കിൽ FALSE (0) ആയിരിക്കും.

ഇലക്ട്രോണിക് സർക്യൂട്ടുകൾ ഉപയോഗിച്ചാണ് കമ്പ്യൂട്ടറുകൾ ഇത്തരം പ്രവർത്തങ്ങൾ നിർവഹിക്കുന്നത് അവയെ ലോജിക്കൽ സർക്യൂട്ട് (Logical circuite) എന്ന് വിളിക്കുന്നു. ഓരോ വ്യക്തിഗത യൂണിറ്റുകളാലും നിർമ്മിക്കപ്പെടുന്ന ലോജിക്കൽ സർക്യൂട്ടിനെ ഗേറ്റ് (Gate) എന്ന് പറയുന്നു. ഒരു

ഗേറ്റ് ഒരു ബൂളിയൻ പ്രവർത്തനത്തെ പ്രതിനിധാനം ചെയ്യുന്നു. ഒന്നോ അതിലധികമോ ലോജിക്കൽ ഇൻപുട്ടുകളിൽ ലോജിക്കൽ പ്രവർത്തനം നടത്താനും ഒരേ ഒരു ലോജിക്കൽ ഔട്ട്പുട്ട് നൽകാനും കഴിയുന്ന ഭൗതികസാമഗ്രിയാണ് ഒരു ലോജിക് ഗേറ്റ് (Logic Gate). ഇലക്ട്രോണിക് സ്വിച്ചുകളായി പ്രവർത്തിക്കുന്ന ഡയോഡുകളോ ട്രാൻസിസ്റ്ററുകളോ ഉപയോഗിച്ചാണ് ലോജിക്കൽ ഗേറ്റുകൾ നിർമ്മിക്കുന്നത്. ബൂളിയൻ ബീജഗണിതത്തിലെ അടിസ്ഥാനപരമായ മൂന്ന് പ്രവർത്തനങ്ങളായ OR, AND, NOT എന്നിവ നിർവഹിക്കുന്നത് യഥാക്രമം OR, AND, NOT എന്നീ ലോജിക്കൽ ഗേറ്റുകൾ ഉപയോഗിച്ചാണ്.

## a. OR ഓപ്പറേറ്ററും OR ഗേറ്റും

ജീവിതത്തിലെ ഒരു സന്ദർഭം നമുക്ക് നോക്കാം. എപ്പോഴാണ് നാം കുട ഉപയോഗിക്കുക? മഴ ഉള്ളപ്പോൾ മാത്രമാണോ? വെയിലുള്ളപ്പോഴും നാം കുട ഉപയോഗിക്കാറുണ്ടല്ലോ. ഈ രണ്ട് സാഹചര്യങ്ങളും ചേർത്ത് ഒരു സംയുക്ത പ്രസ്താവന ഇങ്ങനെ ഉണ്ടാക്കാം. 'മഴ ഉള്ളപ്പോഴോ വെയിലുള്ളപ്പോഴോ നാം കുട ഉപയോഗിക്കുന്നു.' (If it is raining or if it is sunny, we use an umbrella). ഈ പ്രസ്താവനയിൽ OR ന്റെ ഉപയോഗം ശ്രദ്ധിക്കുക. പട്ടിക 2.8 ൽ ഈ പ്രസ്താവനയുടെ വിശകലനം കാണാം. മേൽക്കൊടുത്തിരിക്കുന്ന ഉദാഹരണത്തിൽ 'കുടയുടെ ആവശ്യം' എന്ന യുക്തി വിചിന്തനത്തിന് ബൂളിയൻ OR പ്രവർത്തനവുമായി വളരെയധികം സാദൃശ്യമുണ്ട്.

| മഴ | വെയിൽ | കുടയുടെ ആവശ്യം |
|---|---|---|
| ഇല്ല | ഇല്ല | ഇല്ല |
| ഇല്ല | ഉണ്ട് | ഉണ്ട് |
| ഉണ്ട് | ഇല്ല | ഉണ്ട് |
| ഉണ്ട് | ഉണ്ട് | ഉണ്ട് |

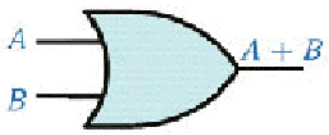
*പട്ടിക 2.8 ലോജിക്കൽ OR ന്റെ പ്രവർത്തനം*

OR ഓപ്പറേറ്റർ ലോജിക്കൽ സങ്കലനം നടത്തുന്നു. ഇതിനായി + ചിഹ്നം ഉപയോഗിക്കുന്നു. അതിനാൽ A+B എന്ന പദപ്രയോഗം വായിക്കേണ്ടത് A അല്ലെങ്കിൽ B (A OR B) എന്നാണ്. താഴെ കൊടുത്തിരിക്കുന്ന പട്ടിക 2.9 OR പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിളാണ്. A, B എന്നിവ ഇൻപുട്ടുകളും (ഓപ്പറന്റുകൾ), A+B ഔട്ട്പുട്ടും (ഉത്തരം) ആണെന്ന് അനുമാനിക്കുക. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആണെങ്കിൽ ഔട്ട്പുട്ട് 1 ആയിരിക്കുമെന്ന് ട്രൂത്ത് ടേബിളിൽ വ്യക്തമാക്കാം.

| A | B | A + B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

*പട്ടിക 2.9 OR പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ*

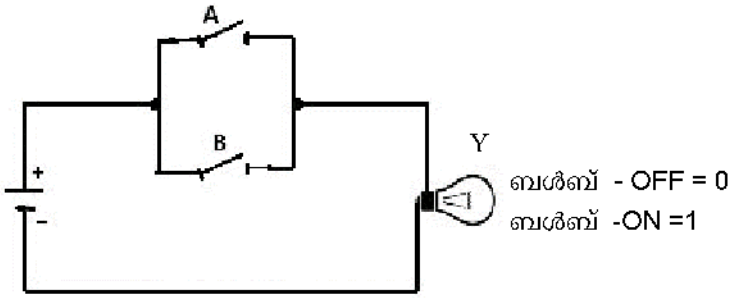
ബൂളിയൻ പ്രവർത്തനവും അതിന്റെ ഫലവും കാണിക്കുന്ന പട്ടികയാണ് ട്രൂത്ത് ടേബിൾ. ഒരു നിശ്ചിത പ്രവർത്തനത്തിന്റെ സാധ്യമായ എല്ലാ ഇൻപുട്ടുകളും അതനുസരിച്ചുള്ള ഫലവും ഈ പട്ടികയിൽ ഉണ്ടായിരിക്കും. സാധാരണ നിലയിൽ ഇത്തരം പ്രവർത്തനങ്ങളിൽ ഓപ്പറന്റ് വേരിയബിളുകളും, ഓപ്പറേറ്ററുകളും ഉണ്ടായിരിക്കും. ഓപ്പറന്റുകളും ഓപ്പറേറ്ററുകളും ചേർന്നതാണ് ബൂളിയൻ പദപ്രയോഗം (Boolean Expression). m ഓപ്പറന്റുകളും n ഓപ്പറേറ്ററുകളുള്ള ഒരു ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ട്രൂത്ത് ടേബിളിൽ $2^m$ നിരകളും m+n കോളങ്ങളും ഉണ്ടായിരിക്കും.




*ചിത്രം 2.5 ലോജിക്കൽ OR ഗേറ്റ്*

ലോജിക് സർക്യൂട്ടുകളുടെ രൂപകല്പനയിൽ ലോജിക്കൽ OR പ്രവർത്തനം നിർവഹിക്കുന്ന ഗേറ്റിനെ ലോജിക്കൽ OR ഗേറ്റ് എന്ന് വിളിക്കുന്നു. ചിത്രം 2.5 ബൂളിയൻ ബീജഗണിതത്തിലെ OR ഗേറ്റിന്റെ ലോജിക്കൽ ചിഹ്നം കാണിച്ചിക്കുന്നു.

ഒരു ഇലക്ട്രോണിക് സർക്യൂട്ടിന്റെ സഹായത്തോടെ ഈ ഗേറ്റിന്റെ പ്രവർത്തനം വിശദീകരിക്കാം. ചിത്രം 2.6 ൽ സമാന്തരമായി വിന്യസിച്ചിട്ടുള്ള രണ്ട് സ്വിച്ചുകൾ OR ഗേറ്റ് സർക്യൂട്ടിനെക്കുറിച്ചുള്ള സങ്കല്പം വ്യക്തമാക്കുന്നു. ഇതിൽ A യും B യും രണ്ട് സ്വിച്ചുകളും Y ഒരു ബൾബുമാണ്. ഇതിൽ സ്വിച്ചുകൾ ഒന്നുകിൽ അടഞ്ഞതോ (ON) അല്ലെങ്കിൽ തുറന്നതോ (OFF) ആയ അവസ്ഥയിലായിരിക്കും. സർക്യൂട്ടിലെ ഒരു സ്വിച്ചെങ്കിലും അടഞ്ഞിരുന്നാൽ ബൾബ് പ്രകാശിക്കും. രണ്ടു



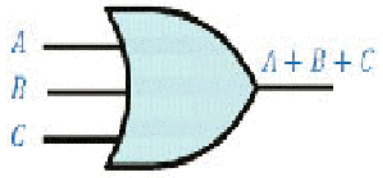

സ്വിച്ച് A - തുറന്നിരിക്കുന്നു = 0(OFF), അടഞ്ഞിരിക്കുന്നു = 1 (ON)
സ്വിച്ച് B - തുറന്നിരിക്കുന്നു = 0(OFF), അടഞ്ഞിരിക്കുന്നു = 1 (ON)

*ചിത്രം 2.6 രണ്ട് സ്വിച്ചും ഒരു ബൾബും സമാന്തരമായി ബന്ധിപ്പിച്ചിട്ടുള്ള സർക്യൂട്ട്*

സ്വിച്ചുകളും തുറന്നിരിക്കുമ്പോൾ മാത്രമാണ് ബൾബ് പ്രകാശിക്കാതിരിക്കുന്നത്. മേൽപ്പറഞ്ഞ സർക്യൂട്ടിന്റെ പ്രവർത്തനം OR ഗേറ്റിന്റെ പ്രവർത്തനവുമായി നമുക്ക് ബന്ധപ്പെടുത്താം. OFF പ്രതിനിധീകരിക്കുന്നത് ലോജിക്കൽ LOW (0) അവസ്ഥയെയും ON ലോജിക്കൽ HIGH (1) അവസ്ഥയെയുമാണ് A, B സ്വിച്ചുകൾ OR ഗേറ്റിന്റെ ഇൻപുട്ടും ബൾബിന്റെ അവസ്ഥ OR ഗേറ്റിന്റെ ഔട്ട് പുട്ടുമാണെന്ന് കരുതുക. ട്രൂത്ത് ടേബിൾ 2.9 ഇത്തരം ഒരു OR ഗേറ്റിന്റെ പ്രവർത്തനം വിശദീകരിക്കുന്നു. OR ഗേറ്റിന്റെ ബൂളിയൻ പദപ്രയോഗം Y=A+B എന്നാണ് എഴുതുന്നത്.

| A | B | C | A + B + C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.10 മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ*




*ചിത്രം 2.7 മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റ്*

ഒരു OR ഗേറ്റിന് രണ്ടിലേറെ ഇൻപുട്ടുകൾ കൈകാര്യം ചെയ്യാൻ സാധിക്കും. മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റിന്റെ ബൂളിയൻ പ്രതിനിധാനവും ലോജിക്കൽ ചിഹ്നവും എങ്ങനെയായിരിക്കുമെന്ന്

നോക്കാം. മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റിന്റെ ബൂളിയൻ പ്രതിനിധാനം Y=A+B+C എന്നാണ്. ഇതിന്റെ ലോജിക്കൽ ചിഹ്നം ചിത്രം 2.7 ൽ കൊടുത്തിരിക്കുന്നു. OR ഗേറ്റിന്റെ ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആണെങ്കിൽ ഔട്ട്പുട്ടും 1 ആയിരിക്കുമെന്ന് പട്ടിക 2.9, 2.10 എന്നിവയിൽ നിന്ന് നമുക്ക് മനസിലാക്കാം. എല്ലാ ഇൻപുട്ടുകളും 0 ആണെങ്കിൽ മാത്രമേ OR ഗേറ്റിന്റെ ഔട്ട്പുട്ട് 0 ആകുന്നുള്ളു.

| റെസ്റ്റോറന്റ് | പണം | ഭക്ഷണം |
|---|---|---|
| ഇല്ല | ഇല്ല | ഇല്ല |
| ഇല്ല | ഉണ്ട് | ഇല്ല |
| ഉണ്ട് | ഇല്ല | ഇല്ല |
| ഉണ്ട് | ഉണ്ട് | ഉണ്ട് |

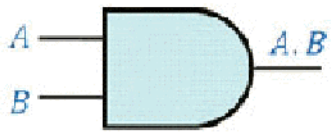
*പട്ടിക 2.11 ലോജിക്കൽ AND ന്റെ പ്രവർത്തനം*

## b. AND ഓപ്പറേറ്ററും AND ഗേറ്റും

AND എന്ന ബൂളിയൻ ക്രിയ മനസ്സിലാക്കുവാൻ നമ്മുക്ക് മറ്റൊരു ഉദാഹരണം നോക്കാം. നിങ്ങൾ വീട്ടിൽ നിന്നും അകലയാണെന്നും ഇപ്പോൾ ഉച്ചഭക്ഷണത്തിനുള്ള സമയമായെന്നും വിചാരിക്കുക. ഉച്ച ഭക്ഷണം ലഭിക്കുന്നതിന് ഇവിടെ രണ്ടു കാര്യങ്ങൾ ആവശ്യമാണ്. 1. ഒരു റെസ്റ്റോറന്റ് ഉണ്ടായിരിക്കണം. 2. ഭക്ഷണം വാങ്ങുന്നതിനുള്ള പണം നിങ്ങളുടെ പക്കൽ ഉണ്ടായിരിക്കണം. ഇവ രണ്ടും ബന്ധിപ്പിച്ചുകൊണ്ട് നമുക്ക് ഒരു സംയുക്തപ്രസ്താവന ഉണ്ടാക്കാം. ഒരു റെസ്റ്റോറന്റും ഭക്ഷണം വാങ്ങുന്നതിനുള്ള പണവും ഉണ്ടെങ്കിൽ ഭക്ഷണം ലഭിക്കും. ഈ പ്രസ്താവനയിൽ 'ഉം' എന്നതിന്റെ ഉപയോഗം ശ്രദ്ധിക്കുക. പട്ടിക 2.11 ൽ ഭക്ഷണം ലഭിക്കുന്നതിനുള്ള യുക്തിപരമായ വിശകലനം കാണിക്കുന്നത് അതിന് ബൂളിയൻ AND ക്രിയയുമായുള്ള സാദൃശ്യമാണ്.

| A | B | A.B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

*പട്ടിക 2.12 AND പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ*

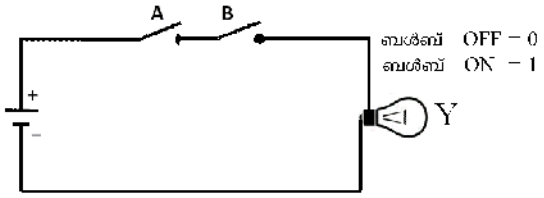



*ചിത്രം 2.8 ലോജിക്കൽ AND ഗേറ്റ്*

AND ഓപ്പറേറ്റർ ലോജിക്കൽ ഗുണിതം നിർവഹിക്കുന്നു. ഈ പ്രവർത്തനം സൂചിപ്പിക്കുന്നത് (ഡോട്ട്) ചിഹ്നം ഉപയോഗിച്ചാണ്. അതിനാൽ A.B എന്ന പദപ്രയോഗം A AND B എന്ന് വായിക്കാം. AND പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ പട്ടിക 2.12 ൽ കൊടുത്തിരിക്കുന്നു. A, B എന്നിവ ഇൻപുട്ടുകളും (ഓപ്പറന്റ്) A.B ഔട്ട്പുട്ടും (ഉത്തരം) ആണെന്ന് അനുമാനിക്കാം. എല്ലാ ഇൻപുട്ടുകളും 1 ആകുമ്പോൾ മാത്രമാണ് ഔട്ട്പുട്ട് 1 ലഭിക്കുന്നത്. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 0 ആകുമ്പോൾ ഔട്ട്പുട്ടും 0 ആകുന്നു.

സർക്യൂട്ടുകൾ രൂപകല്പന ചെയ്യുമ്പോൾ യുക്തിപരമായ AND പ്രവർത്തനം നിർവഹിക്കാൻ ഉപയോഗിക്കുന്ന ലോജിക്കൽ ഗേറ്റിനെ AND ഗേറ്റ് എന്ന് വിളിക്കുന്നു. ബൂളിയൻ ബീജഗണിതത്തിലെ AND ഗേറ്റിന്റെ ചിഹ്നം ചിത്രം 2.8 ൽ കാണിച്ചിരിക്കുന്നു. ചിത്രം 2.9 ൽ കൊടുത്തിരിക്കുന്ന ഇലക്ട്രോണിക് സർക്യൂട്ട് വഴി ഈ ഗേറ്റിന്റെ പ്രവർത്തനം വിശദമാക്കാം. ഈ സർക്യൂട്ടിൽ AND ഗേറ്റ് എന്ന സങ്കല്പം വിശദമാക്കുന്ന ശ്രേണിയായി രണ്ട് സ്വിച്ചുകളുണ്ട്. ഇതിൽ A, B എന്നിവ സ്വിച്ചുകളും Y ബൾബുമാണ്. സർക്യൂട്ടിലെ രണ്ട് സ്വിച്ചുകളും അടഞ്ഞിരിക്കുമ്പോൾ മാത്രമാണ് ബൾബ് പ്രകാശിക്കുന്നത് എന്ന് കാണാം. ഏതെങ്കിലും ഒരു സ്വിച്ച് തുറന്നിരുന്നാൽ ബൾബ്

പ്രകാശിക്കുന്നില്ല. മേൽപ്പറഞ്ഞ സർക്യൂട്ടിന്റെ പ്രവർത്തനം AND ഗേറ്റിന്റെ പ്രവർത്തനവുമായി നമുക്ക് ബന്ധപ്പെടുത്താം. OFF പ്രതിനിധീകരിക്കുന്നത് ലോജിക്കൽ LOW (0) അവസ്ഥയെയും ON ലോജിക്കൽ HIGH (1) അവസ്ഥയെയുമാണ്. A, B സ്വിച്ചുകൾ AND ഗേറ്റിന്റെ ഇൻപുട്ടും ബൾബിന്റെ അവസ്ഥ ANDഗേറ്റിന്റെ ഔട്ട്പുട്ടുമാണെന്ന് കരുതുക. ട്രൂത്ത് ടേബിൾ 2.9 ഇത്തരം ഒരു AND ഗേറ്റിന്റെ പ്രവർത്തനം വിശദീകരിക്കുന്നു. AND ഗേറ്റിന്റെ ബൂളിയൻ പദപ്രയോഗം Y=A.B എന്നാണ് എഴുതുന്നത്.




*സ്വിച്ച് A തുറന്നിരിക്കുന്നു = 0 (OFF),*
*അടഞ്ഞിരിക്കുന്നു = 1 (ON)*
*സ്വിച്ച് B തുറന്നിരിക്കുന്നു = 0 (OFF),*
*അടഞ്ഞിരിക്കുന്നു = 1 (ON)*

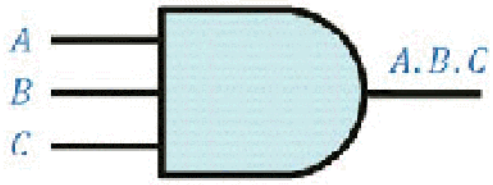
*ചിത്രം 2.9 രണ്ട് സ്വിച്ചും ഒരു ബൾബും ശ്രേണിയായി ബന്ധിപ്പിച്ചിട്ടുള്ള സർക്യൂട്ട്.*

| A | B | C | A.B.C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.13 : മൂന്ന് ഇൻപുട്ടുകളുള്ള AND ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ*

AND ഗേറ്റിന് രണ്ടിലേറെ ഇൻപുട്ടുകൾ കൈകാര്യം ചെയ്യാനാകും. ഇതിൽ മൂന്ന് ഇൻപുട്ട് വരുമ്പോൾ ബൂളിയൻ പ്രതിനിധാനവും ചിഹ്നവും എങ്ങനെയാവും എന്ന് നോക്കാം. മൂന്ന് ഇൻപുട്ടുള്ള AND ഗേറ്റിന്റെ ബൂളിയൻ പദപ്രയോഗം Y=A.B.C എന്നാണ്.

ചിത്രം 2.10 ൽ മൂന്ന് ഇൻപുട്ടുകളുള്ള AND ഗേറ്റ് കാണിച്ചിരിക്കുന്നു. AND ഗേറ്റിൽ ഏതെങ്കിലും ഇൻപുട്ട് 0 ആണെങ്കിൽ ഔട്ട്പുട്ടും 0 ആയിരിക്കും. എല്ലാ ഇൻപുട്ടുകളും 1 ആണെങ്കിൽ മാത്രമേ ഔട്ട്പുട്ടായി 1 ലഭിക്കുകയുള്ളൂ. 2.12, 2.13 എന്നീ ട്രൂത്ത് ടേബിളുകൾ ഇത് കാണിച്ചു തരുന്നു.




*ചിത്രം 2.10 : മൂന്ന് ഇൻപുട്ടുകളുള്ള AND ഗേറ്റ്*

## c. NOT ഓപ്പറേറ്ററും NOT ഗേറ്റും

ബൂളിയൻ NOT പ്രവർത്തനം മനസ്സിലാക്കുന്നതിന് നമുക്ക് മറ്റൊരു സാഹചര്യം ചർച്ച ചെയ്യാം. എന്നും രാവിലെ നിങ്ങൾ വ്യായാമത്തിനായി നടക്കാൻ പോകുന്നയാളാണെന്ന് കരുതുക. എല്ലാ ദിവസവും നിങ്ങൾക്ക് അതിന് കഴിയുമോ? മഴയുണ്ടെങ്കിൽ നിങ്ങൾക്ക് അതിന് സാധിക്കുമോ? പട്ടിക 2.14 ൽ ഇതുമായി ബന്ധപ്പെട്ട എല്ലാ സാധ്യതകളും കാണിച്ചിരിക്കുന്നു. ഇതിന് ബൂളിയൻ NOT പ്രവർത്തനവുമായി സാമ്യമുണ്ട്.

| മഴ | വ്യായാമം |
|---|---|
| ഇല്ല | ഉണ്ട് |
| ഉണ്ട് | ഇല്ല |

*പട്ടിക 2.14 ലോജിക്കൽ NOT*

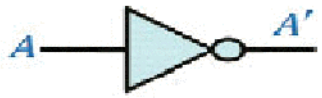
ഇത് ഒരു യൂനറി (Unary) ഓപ്പറേറ്ററാണ്. അതിനാൽ ഇതിന് ഒരേ ഒരു ഓപ്പറന്റ് മാത്രമെ ആവശ്യമുള്ളൂ. NOT ഓപ്പറേറ്റർ യുക്തിപരമായ നിഷേധം (Logical Negation) നിർവഹിക്കുന്നു. അതിന്റെ ചിഹ്നം ¯ (over bar) മുകൾവര ആണ്.

NOT ന്റെ പദപ്രയോഗത്തെ $\overline{A}$ എന്നോ A' എന്നോ അടയാളപ്പെടുപ്പെടുത്താം. $\overline{A}$ എന്നതിനെ A ബാർ എന്നും A' എന്നതിനെ A ഡാഷ് എന്നും ഉച്ചരിക്കുന്നു. ടേബിൾ 2.15 ൽ NOT പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ കൊടുത്തിരിക്കുന്നു.

| A | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

*പട്ടിക 2.15 NOT പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ*

NOT ഓപ്പറേറ്ററിൽ A എന്നത് ഇൻപുട്ടും (ഓപ്പറന്റും) $\overline{A}$ എന്നത് ഔട്ട്പുട്ടും (ഉത്തരം) ആണെന്ന് അനുമാനിക്കുക. ഇൻപുട്ടിന്റെ വിപരീത മൂല്യമാണ് ഔട്ട്പുട്ട് എന്ന് ട്രൂത്ത് ടേബിളിൽ നിന്ന് മനസ്സിലാക്കാം. NOT പ്രവർത്തനം നിർവഹിക്കുന്ന ലോജിക് ഗേറ്റിനെ NOT ഗേറ്റ് എന്ന് വിളിക്കുന്നു. ചിത്രം 2.11 ൽ NOT ഗേറ്റിന്റെ ചിഹ്നം കാണിച്ചിരിക്കുന്നു.




*ചിത്രം 2.11 NOT ഗേറ്റ്*

**സ്വയം വിലയിരുത്താം**

1. ബൂളിയൻ വേരിയബിൾ എന്ന പദം നിർവചിക്കുക.
2. ഓരോ വ്യക്തിഗത യൂണിറ്റിനാലും നിർമ്മിക്കപ്പെടുന്ന ലോജിക്കൽ സർക്യൂട്ടിനെ ..... എന്ന് വിളിക്കുന്നു.
3. എല്ലാ ഇൻപുട്ടുകളും ഉയർന്നതായിരിക്കുമ്പോൾ (High) മാത്രം ഉയർന്ന (High) ഔട്ട്പുട്ട് തരുന്ന ലോജിക്കൽ ഓപ്പറേറ്റർ/ഗേറ്റ് ന്റെ പേര് എഴുതുക.
4. ട്രൂത്ത് ടേബിൾ എന്ന പദം നിർവചിക്കുക.
5. AND ഓപ്പറേഷനിൽ ലോജിക്കൽ ..... ഉം OR ഓപ്പറേറ്ററിൽ ലോജിക്കൽ ..... ഉം ആണ് നടക്കുന്നത്.
6. OR ഗേറ്റിന്റെ ലോജിക് ചിഹ്നം വരക്കുക.

ഒരു NOT ഗേറ്റിനെ ഇൻവർട്ടർ (Inverter) എന്നും വിളിക്കുന്നു. ഇതിന് ഒരു ഇൻപുട്ടും ഒരു ഔട്ട്പുട്ടും മാത്രമേയുള്ളൂ. ഇൻപുട്ട് എല്ലായ്പോഴും വിപരീതാവസ്ഥയിലേക്ക് പരിവർത്തനം ചെയ്യുന്നു. ഇൻപുട്ട് 0 ആണെങ്കിൽ അതിന്റെ ഔട്ട്പുട്ട് വിപരീതം അല്ലെങ്കിൽ പൂരകം 1 ആയിരിക്കും. അതുപോലെ ഇൻപുട്ട് 1 ആണെങ്കിൽ ഔട്ട്പുട്ട് അതിന്റെ പൂരകമായ 0 ആയിരിക്കും.

## 2.6 ബൂളിയൻ ബീജഗണിതത്തിലെ അടിസ്ഥാന അംഗീകൃത തത്വങ്ങൾ (Basic postulates of Boolean algebra)

ചില അടിസ്ഥാന നിയമങ്ങളോടുകൂടിയ ഒരു ഗണിതശാസ്ത്ര ശാഖയാണ് ബൂളിയൻ ബീജഗണിതം. ഈ അടിസ്ഥാന നിയമങ്ങളെ അംഗീകൃത തത്വങ്ങൾ (Postulates) എന്ന് വിളിക്കുന്നു. ഇവയ്ക്ക് ശാസ്ത്രീയമായ അടിത്തറയില്ലെങ്കിലും കൃത്യമായ ഘടനയോടുകൂടിയ ശാസ്ത്ര തത്വങ്ങൾ നിർമ്മിക്കുവാൻ ഇവ ഉപയോഗിക്കുന്നു. മറ്റൊരു തരത്തിൽപ്പറഞ്ഞാൽ ഇത്തരം അംഗീകൃത തത്വങ്ങളും നിയമങ്ങളും ഉപയോഗിച്ച് തെളിയിക്കാൻ കഴിയുന്ന ചില സിദ്ധാന്തങ്ങൾ ബൂളിയൻ ബീജഗണിതത്തിൽ ഉണ്ട്.

**അംഗീകൃത തത്വം 1 : 0 ന്റെയും 1 ന്റെയും സിദ്ധാന്തങ്ങൾ**

$A \neq 0$, ആണെങ്കിൽ $A = 1$ ഉം $A \neq 1$ ആണെങ്കിൽ $A = 0$ ആകുന്നു.

**അംഗീകൃത തത്വം 2 : OR ഓപ്പറേഷൻ (യുക്തിപരമായ സങ്കലനം)**

$0 + 0 = 0 \qquad 0 + 1 = 1 \qquad 1 + 0 = 1 \qquad 1 + 1 = 1$

**അംഗീകൃത തത്വം 3 : AND ഓപ്പറേഷൻ (യുക്തിപരമായ ഗുണനം)**

$0 . 0 = 0 \qquad 0 . 1 = 0 \qquad 1 . 0 = 0 \qquad 1 . 1 = 1$

**അംഗീകൃത തത്വം 4 : NOT ഓപ്പറേഷൻ (യുക്തിപരമായ നിഷേധം അല്ലെങ്കിൽ പൂരക നിയമം)**

$\bar{0} = 1 \qquad \bar{1} = 0$

### ദ്വൈതസിദ്ധാന്തം (Principle of Duality)

ബൂളിയൻ വേരിയബിളും, വിലകളും, ബൂളിയൻ ഓപ്പറേറ്റർ വഴി കൂട്ടിച്ചേർത്തു ബൂളിയൻ പദപ്രയോഗം നിർമ്മിക്കാം. $X+Y$, $A+1$ എന്നിവ ബൂളിയൻ പദപ്രയോഗങ്ങൾക്ക് ഉദാഹരണങ്ങളാണ്. അതുപോലെ ബൂളിയൻ ബീജഗണിതത്തിന്റെ അംഗീകൃത തത്വങ്ങളായ 2, 3, 4 എന്നിവയും ബൂളിയൻ പദപ്രയോഗങ്ങളാണ്. അംഗീകൃത തത്വം 2 ലെ പ്രസ്താവനകൾ നമുക്ക് നോക്കാം. ഇതിൽ പൂജ്യത്തിനെ ഒന്നായും ഒന്നിനെ പൂജ്യമായും അതുപോലെ ഓപ്പറേറ്ററുകളായ OR (+) നെ AND (.) ആയും മാറ്റംവരുത്തിയാൽ അംഗീകൃത തത്വം 3 ലഭിക്കുന്നു. അതുപോലെ അംഗീകൃത തത്വം 2 പ്രസ്താവനയിൽ പൂജ്യത്തിനെ ഒന്നായും ഒന്നിനെ പൂജ്യമായും അതുപോലെ ഓപ്പറേറ്ററുകളായ AND (.) നെ OR (+) ആയും മാറ്റം വരുത്തിയാൽ അംഗീകൃത തത്വം 3 ലഭിക്കുന്നു. ഈ ആശയത്തെയാണ് ദ്വൈതസിദ്ധാന്തം എന്ന് പറയുന്നത്.

ഏതൊരു ബൂളിയൻ പ്രസ്താവനയ്ക്കും ഒരു ദ്വൈത രൂപം ഉണ്ടായിരിക്കും. അത് താഴെ പറയുന്ന മൂന്ന് നിയമങ്ങൾ അനുസരിച്ചാണ് വികസിപ്പിക്കുന്നതെന്ന് ദ്വൈതസിദ്ധാന്തം പ്രസ്താവിക്കുന്നു.

(i) ഓരോ OR (+) ഉം AND (.) ആയി മാറ്റുക.

(ii) ഓരോ AND (.) ഉം OR (+) ആയി മാറ്റുക.

(iii) ഓരോ 0 ഉം 1 ആയും 1 നെ 0 ആയും മാറ്റുക.

## 2.7 ബൂളിയൻ ബീജഗണിതത്തിലെ അടിസ്ഥാന തത്വങ്ങൾ (Basic theorems of Boolean algebra)

ഓരോ സിദ്ധാന്തത്തിനും ചില അടിസ്ഥാനവും അംഗീകൃതവുമായ നിയമങ്ങളും ഉണ്ട്. സിദ്ധാന്തത്തിന്റെ നിയമാവലികൾ സ്വയം പ്രമാണം (Axioms) എന്നറിയപ്പെടുന്നു. മേൽപ്പറഞ്ഞ സ്വയം പ്രമാണങ്ങളോ അടിസ്ഥാന തത്വങ്ങളോ ഉപയോഗിച്ചുള്ള അനുമാനങ്ങളിൽ നിന്ന് ഒരു നിഗമനത്തിൽ എത്തിച്ചേരാവുന്നതാണ്. ഇത്തരം നിഗമനങ്ങളെ നിയമങ്ങൾ അല്ലെങ്കിൽ സിദ്ധാന്തങ്ങൾ എന്ന് വിളിക്കുന്നു. ബൂളിയൻ പദപ്രയോഗം ലഘൂകരിക്കുന്നതിനും കൈകാര്യം ചെയ്യുന്നതിനും ഉള്ള സാമഗ്രികൾ ബൂളിയൻ ആൾജിബ്രയുടെ സിദ്ധാന്തങ്ങൾ തരുന്നു. ഇവിടെ ചില ബൂളിയൻ നിയമങ്ങളെയും സിദ്ധാന്തങ്ങളെയെയും കുറിച്ച് ചർച്ച ചെയ്യാം. മുൻകൂട്ടി തെളിയിച്ചിട്ടുള്ള ബൂളിയൻ നിയമങ്ങളുടെയും ട്രൂത്ത് ടേബിളുകളുടെയും സഹായത്താൽ മേൽപ്പറഞ്ഞ സിദ്ധാന്തങ്ങളും നിയമങ്ങളും തെളിയിക്കാം.

### 2.7.1 അനന്യത നിയമം (Identity law)

X ഒരു ബൂളിയൻ വേരിയബിൾ ആണെങ്കിൽ, അനന്യത നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാണ്.

(i) $0 + X = X$ (ii) $1 + X = 1$

(iii) $0 . X = 0$ (iv) $1 . X = X$

പ്രസ്താവന (i) ഉം (ii) ഉം സങ്കലന അനന്യത എന്നും പ്രസ്താവന (iii) ഉം (iv) ഉം ഗുണന അനന്യത എന്നും അറിയപ്പെടുന്നു. പ്രസ്താവന (i), പ്രസ്താവന (iv) ന്റെ ദ്വൈത രൂപവും (Dual form), അതുപോലെ തിരിച്ചുമാകുന്നു. അതുപോലെ പ്രസ്താവന (ii), പ്രസ്താവന (iii) ന്റെ ദ്വൈത രൂപങ്ങളാകുന്നു. ട്രൂത്ത് ടേബിളുകൾ 2.16(a), 2.16(b), 2.17(a), 2.17(b) ഈ നിയമങ്ങൾ തെളിയിക്കുന്നു.

| 0 | X | 0 + X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |

പട്ടിക 2.16(a) സങ്കലന അനന്യത നിയമം

| 1 | X | 1 + X |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |

പട്ടിക 2.16(b) സങ്കലന അനന്യത നിയമം

പട്ടിക 2.16 (a) ലെ 2 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്യമാണെന്നു കാണാം. അതിനാൽ $0 + X = X$ എന്ന് തെളിയിക്കപ്പെടുന്നു. അതുപോലെ പട്ടിക 2.16 (b) ലെ 1 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്ല്യമായതിനാൽ $1 + X = 1$ എന്ന പ്രസ്താവനയും ശരിയാകുന്നു.

| 0 | X | 0 . X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |

പട്ടിക 2.17(a) ഗുണന അനന്യത നിയമം

| 1 | X | 1 . X |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |

പട്ടിക 2.17(b) ഗുണന അനന്യത നിയമം

പട്ടിക 2.17(a) ലെ 2 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്യമാണെന്നു കാണാം. അതിനാൽ $0.X = 0$ എന്ന് തെളിയിക്കപ്പെടുന്നു. അതുപോലെ പട്ടിക 2.17(b) ലെ 2 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്യമായതിനാൽ $1.X = X$ എന്ന പ്രസ്താവനയും ശരിയാകുന്നു.

### 2.7.2 വർഗസമ നിയമം (Idempotent law)

വർഗസമ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു. (i) $X + X = X$

(ii) $X . X = X$

X ന്റെ വില 0 ആയാൽ പ്രസ്താവന സത്യമാകുന്നു, എന്തുകൊണ്ടെന്നാൽ $0 + 0 = 0$ (അംഗീകൃത തത്വം 2) ഉം $0.0 = 0$ ഉം (അംഗീകൃത തത്വം 3). പ്രകാരം പ്രസ്താവന സത്യമാകുന്നു. അതുപോലെ X ന്റെ വില 1 ആയാലും പ്രസ്താവനകൾ ശരിയാകുന്നു. ട്രൂത്ത് ടേബിളുകൾ 2.18(a), 2.18(b) എന്നിവ ഈ നിയമങ്ങളുടെ തെളിവ് കാണിച്ച് തരുന്നു.

| X | X | X + X |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

പട്ടിക 2.18 (a) : വർഗ്ഗസമനിയമം

| X | X | X . X |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

പട്ടിക 2.18 (b) : വർഗ്ഗസമനിയമം

### 2.7.3 വർഗപൂരക നിയമം ( Involution law)

ഈ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു. $\overline{\overline{X}} = X$

അംഗീകൃത തത്വം 4 പ്രകാരം $X = 0$ ആയാൽ $\overline{X}$ = 1ആകുന്നു. അതിന്റെ പൂരകം $\overline{\overline{X}} = \overline{1} = 0$ എന്നത് X നു തുല്യമാകുന്നു. X ന്റെ വില 1 ആയാലും പ്രസ്താവന സത്യമാകുന്നു.

ട്രൂത്ത് ടേബിൾ 2.19 ലെ 1 ഉം 3 ഉം നിരകൾ $\overline{\overline{X}} = X$ എന്നത് തെളിയിക്കുന്നു.

| X | $\overline{X}$ | $\overline{\overline{X}}$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |

*പട്ടിക 2.19 വർഗപൂരക നിയമം*

### 2.7.4 പൂരക നിയമം (Complementary law)

പൂരക നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു. (i) $X + \overline{X} = 1$

(ii) $X \cdot \overline{X} = 0$

X ന്റെ വില 0 ആയാൽ $\overline{X}$ എന്നത് 1 ആകുന്നു. ആയതിനാൽ $X + \overline{X}$ എന്നത് $0 + 1 = 1$ ആകുന്നു (അംഗീകൃത തത്വം 2). അതുപോലെ X ന്റെ വില 1 ആയാൽ $\overline{X} = 0$ ആകുന്നു. ട്രൂത്ത് ടേബിളുകൾ 2.20(a), 2.20(b) എന്നിവ സാധ്യമായ വിലകൾ ഉൾകൊള്ളിച്ചു കൊണ്ട് ഈ നിയമം തെളിയിക്കുന്നു. പ്രസ്താവനകൾ പരസ്പരം ദ്വന്ദ്വങ്ങൾ ആണെന്ന് ശ്രദ്ധിക്കുക.

| X | $\overline{X}$ | $X + \overline{X}$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 1 |

*പട്ടിക 2.20(a) പൂരക നിയമം*

| X | $\overline{X}$ | $X \cdot \overline{X}$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 0 |

*പട്ടിക 2.20(b) പൂരക നിയമം*

### 2.7.5 ക്രമനിയമം (Commutative law)

OR, AND എന്നീ ഓപ്പറേഷനുകളിൽ വേരിയബിളിന്റെ സ്ഥാനം മാറ്റുന്നതിന് ക്രമനിയമം അനുവദിക്കുന്നു. X ഉം Y ഉം രണ്ട് വേരിയബിളുകൾ ആണെങ്കിൽ, ഈ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു.

(i) $X + Y = Y + X$

(ii) $X \cdot Y = Y \cdot X$

ട്രൂത്ത് ടേബിളുകൾ 2.21(a), 2.21(b) എന്നിവ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | X + Y | Y + X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.21(a) ക്രമനിയമം*

| X | Y | X.Y | Y.X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.21(b) ക്രമനിയമം*

OR, AND എന്നിവയുടെ ഓരോ ഓപ്പറേഷനിലും ഓപ്പറന്റുകളുടെ ക്രമം ഔട്ട്പുട്ടിനെ ബാധിക്കുകയില്ലെന്ന് നിയമം ഉറപ്പുവരുത്തുന്നു.

## 2.7.6 സംയോജന നിയമം (Associative law)

OR, AND എന്നീ ഓപ്പറേഷനുകളിൽ മൂന്നു ഓപ്പറന്റുകൾ വരുന്ന സന്ദർഭങ്ങളിൽ, ഓപ്പറന്റുകളെ വ്യത്യസ്ത രീതിയിൽ ഗ്രൂപ്പ് ചെയ്യുവാൻ സംയോജന നിയമം അനുവദിക്കുന്നു. X, Y, Z എന്നിവ മൂന്ന് വേരിയബിളുകൾ ആണെങ്കിൽ സംയോജന നിയമം ഇങ്ങനെയാകുന്നു.

(i) $X + (Y + Z) = (X + Y) + Z$

(ii) $X . (Y . Z) = (X . Y) . Z$

ട്രൂത്ത് ടേബിളുകൾ 2.22(a), 2.22(b) എന്നിവ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | Z | X + Y | Y + X | X+(Y + Z) | (X+Y)+Z |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.22(a) സംയോജന നിയമം 1*

പട്ടിക 2.22(a) യിലെ 6 ഉം 7 ഉം നിരകൾ X + (Y + Z) = (X + Y) + Z എന്ന നിയമത്തെയും പട്ടിക 2.22(b) യിലെ 6 ഉം 7 ഉം നിരകൾ X . (Y . Z) = (X . Y) . Z എന്ന നിയമത്തെയും സാധൂകരിക്കുന്നു.

| X | Y | Z | X . Y | Y . Z | X . (Y . Z) | (X . Y) . Z |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.22(b) സംയോജന നിയമം 2*

OR (യുക്തിപരമായ സങ്കലനം) അല്ലെങ്കിൽ AND (യുക്തിപരമായ ഗുണിതം) പ്രവർത്തനങ്ങളിലെ വേരിയബിളുകളുടെ ക്രമവും കൂടിച്ചേരലും അന്തിമ ഔട്ട്പുട്ടിനെ ബാധിക്കുന്നില്ലെന്ന് സംയോജന നിയമം ഉറപ്പാക്കുന്നു.

## 2.7.7 വിതരണ നിയമം (Distributive law)

ബൂളിയൻ പദപ്രയോഗങ്ങൾ സാധാരണ ബീജഗണിതത്തിലെപ്പോലെ സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം വഴിയും അതുപോലെ ഗുണനത്തിനുമേലുള്ള സങ്കലന വിതരണം വഴിയും വിപുലീകരിക്കാൻ വിതരണ നിയമപ്രകാരം സാധിക്കുന്നു. X, Y, Z എന്നിവ വേരിയബിളുകൾ ആണെങ്കിൽ, ഈ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു.

(i) $X . ( Y + Z ) = X . Y + X . Z$

(ii) $X + Y . Z = ( X + Y ) . ( X + Z )$

ചുവടെ ചേർത്തിരിക്കുന്ന ട്രൂത്ത് ടേബിളുകൾ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | Z | Y + Z | X.(Y+Z) | X.Y | X.Z | X.Y + X.Z |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.23(a) സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം*

പട്ടിക 2.23(a) യിലെ 5 ഉം 8 ഉം നിരകൾ $X . (Y + Z) = X . Y + X . Z$ എന്ന നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | Z | Y . Z | X + Y . Z | X+Y | X+Z | (X+Y) . (X+Z) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.23(b) ഗുണനത്തിനുമേലുള്ള സങ്കലന വിതരണം*

പട്ടിക 2.23(b) യിലെ 5 ഉം 8 ഉം നിരകൾ $X + Y \,.\, Z = (X + Y) \,.\, (X + Z)$ എന്ന നിയമത്തെ സാധൂകരിക്കുന്നു.

## 2.7.8 സ്വാംശീകരണ നിയമം (Absorption law)

രണ്ടു വേരിയബിളുകൾ ഉപയോഗിക്കുകയും അതിൽ ഒന്ന് ഉത്തരം ആകുകയും ചെയ്യുന്ന തരത്തിലുള്ള വിതരണ നിയമമാണ് സ്വാംശീകരണ നിയമം. X ഉം Y ഉം വേരിയബിളുകൾ ആണെങ്കിൽ, ഈ നിയമം ഇങ്ങനെ പ്രസ്താവിക്കുന്നു.

(i) $X + (X \,.\, Y) = X$

(ii) $X \,.\, (X + Y) = X$

ട്രൂത്ത് ടേബിളുകൾ 2.24(a), 2.24(b) എന്നിവ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | X . Y | X+(X.Y) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.24(a) സ്വാംശീകരണ നിയമം 1*

| X | Y | X+Y | X.(X+Y) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.24(b) സ്വാംശീകരണ നിയമം 2*

പട്ടിക 2.24(a) യിലെ 1 ഉം 4 ഉം നിരകളും പട്ടിക 2.24(b) യിലെ 1 ഉം 4 ഉം നിരകളും ഈ നിയമം ശരിയാണെന്നു തെളിയിച്ചിരിക്കുന്നു.

പട്ടിക 2.25 നമ്മൾ ചർച്ച ചെയ്ത എല്ലാ ബൂളിയൻ നിയമങ്ങളും സൂക്ഷ്മമായി ചിത്രീകരിക്കുന്നു.

| നം. | ബൂളിയൻ നിയമം | പ്രസ്താവന 1 | പ്രസ്താവന 2 |
|---|---|---|---|
| 1 | സങ്കലന അനന്യത (Additive Identity) | $0 + X = X$ | $1 + X = 1$ |
| 2 | ഗുണന അനന്യത (Multiplicative Identity) | $0 \,.\, X = 0$ | $1 \,.\, X = X$ |
| 3 | വർഗസമ നിയമം (Idempotent Law) | $X + X = X$ | $X \,.\, X = X$ |
| 4 | വർഗപൂരക നിയമം (Involution Law) | $\overline{\overline{X}} = X$ | |
| 5 | പൂരകനിയമം (Complimentary Law) | $X + \overline{X} = 1$ | $X \,.\, \overline{X} = 0$ |
| 6 | ക്രമനിയമം (Commutative Law) | $X + Y = Y + X$ | $X \,.\, Y = Y \,.\, X$ |
| 7 | സംയോജനനിയമം (Associative Law) | $X + (Y + Z) = (X + Y) + Z$ | $X \,.\, (Y \,.\, Z) = (X \,.\, Y) \,.\, Z$ |
| 8 | വിതരണനിയമം (Distributive Law) | $X \,.\, (Y + Z) = X \,.\, Y + X \,.\, Z$ | $X + (Y \,.\, Z) = (X+Y) \,.\, (X+Z)$ |
| 9 | സ്വാംശീകരണനിയമം (Absorption Law) | $X + (X \,.\, Y) = X$ | $X \,.\, (X + Y) = X$ |

*പട്ടിക 2.25 ബൂളിയൻ നിയമങ്ങൾ*

നമ്മൾ ഇതുവരെ ചർച്ച ചെയ്ത എല്ലാ നിയമങ്ങളും ട്രൂത്ത് ടേബിളിലൂടെയാണ് തെളിയിച്ചിട്ടുള്ളത്. അവയിൽ ചിലത് മറ്റ് ചില നിയമങ്ങൾ ഉപയോഗിച്ച് തെളിയിക്കാനാകും. ഇങ്ങനെ തെളിയിക്കുന്ന രീതിയെ ബീജഗണിത രീതിയിലുള്ള തെളിവ് (Algebraic proof) എന്ന് വിളിക്കുന്നു. അവയിൽ ചിലത് നമുക്ക് നോക്കാം.

**i. $X . (X + Y) = X$ എന്ന് തെളിയിക്കുക. (അബ്സോർപ്ഷൻ നിയമം).**

LHS $= X . (X + Y)$

$= X . X + X . Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X + X . Y$ *(വർഗസമനിയമം (Idempotent law))*

$= X . (1 + Y)$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണനവിതരണം (Distribution of multiplication over addition))*

$= X . 1$ *(സങ്കലന അനന്യത (Additive identity))*

$= X$ *(ഗുണന അനന്യത (Multiplicative identity))*

$=$ RHS

**ii. $X + (X . Y) = X$ എന്ന് തെളിയിക്കുക. (അബ്സോർപ്ഷൻ നിയമം).**

LHS $= X + (X . Y)$

$= X . 1 + X . Y$ *(ഗുണന അനന്യത (Multiplicative identity))*

$= X . (1 + Y)$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X . 1$ *(സങ്കലന അനന്യത (Additive identity))*

$= X$ *(ഗുണന അനന്യത (Multiplicative identity))*

$=$ RHS

**iii. $X + (Y . Z) = (X + Y).(X+Z)$ എന്ന് തെളിയിക്കുക. (വിതരണ നിയമം).**

പ്രസ്താവനയിലെ RHS ലെ പദപ്രയോഗം നമുക്ക് എടുക്കാം.

$(X+Y) . (X+Z)$

$= (X+Y) . X + (X+Y) . Z$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X . (X+Y) + Z . (X+Y)$ *(ക്രമനിയമം (Commutative law))*

$= X . X + X . Y + Z . X + Z . Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X + X . Y + Z . X + Z . Y$ *(വർഗസമ നിയമം (Idempotent law))*

$= X . 1 + X . Y + Z . X + Z . Y$ *(ഗുണന അനന്യത (Multiplicative identity))*

$= X . (1 + Y) + Z . X + Z . Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X . 1 + Z . X + Z . Y$ *(സങ്കലന അനന്യത (Additive identity))*

$= X . (1 + Z) + Z . Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X . 1 + Z . Y$ *(സങ്കലന അനന്യത (Additive identity))*

$= X + Y . Z$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$=$ LHS

കിട്ടിയിരിക്കുന്ന പദപ്രയോഗം തന്നിരിക്കുന്ന പ്രസ്താവനയുടെ LHS ആകുന്നു. ആയതിനാൽ സിദ്ധാന്തം തെളിയിക്കപ്പെട്ടിരിക്കുന്നു.

## 2.8 ഡി മോർഗൻസ് സിദ്ധാന്തം (De Morgan's theorems)

ലണ്ടൻ യൂണിവേഴ്സിറ്റി കോളേജിലെ തർക്കശാസ്ത്ര വിദഗ്ധനും ഗണിത ശാസ്ത്രജ്ഞനുമായിരുന്ന അഗസ്റ്റസ് ഡി മോർഗൻ (1806-1871) സങ്കീർണമായ ബൂളിയൻ പദപ്രയോഗങ്ങൾ ലളിതമാക്കാൻ രണ്ടു സിദ്ധാന്തങ്ങൾ നിർദേശിച്ചു. ഈ സിദ്ധാന്തങ്ങൾ ഡി മോർഗൻസ് സിദ്ധാന്തങ്ങൾ എന്നറിയപ്പെടുന്നു. ഇവ താഴെ കൊടുത്തിരിക്കുന്നു.

(i) $\overline{X+Y} = \overline{X} \cdot \overline{Y}$

(ii) $\overline{X.Y} = \overline{X} + \overline{Y}$

ഈ സിദ്ധാന്തങ്ങൾ ഇങ്ങനെ പ്രസ്താവിക്കാം.

(i) 'ബൂളിയൻ വേരിയബിളുകളുടെ തുകയുടെ പൂരകവും അവയുടെ ഓരോന്നിന്റെയും പൂരകങ്ങളുടെ ഗുണനഫലവും തുല്യമായിരിക്കും.'

(ii) 'ബൂളിയൻ വേരിയബിളുകളുടെ ഗുണനഫലത്തിൻറെ പൂരകവും അവയുടെ ഓരോന്നിന്റെയും പൂരകങ്ങളുടെ തുകയും തുല്യമായിരിക്കും.'

**ഒന്നാമത്തെ സിദ്ധാന്തത്തിന്റെ ബീജഗണിത രീതിയിലുള്ള തെളിവ്.**

നമുക്ക് തെളിയിക്കേണ്ടത് $\overline{X+Y} = \overline{X} \cdot \overline{Y}$ എന്നാണ്.

$Z = X + Y$ _____ (1) എന്ന് നമുക്ക് അനുമാനിക്കാം.

എങ്കിൽ $\overline{Z} = \overline{X+Y}$ _____ (2)

$Z + \overline{Z} = 1$ _____ (3)

$Z \cdot \overline{Z} = 0$ _____ (4)

പൂരകനിയമപ്രകാരം സമവാക്യം (3) ഉം (4) ഉം ശരിയാണെന്നു നമുക്കറിയാം

(1), (3) എന്നീ സമവാക്യങ്ങൾ (2), (4) എന്നീ സമവാക്യങ്ങളിൽ യഥാക്രമം പ്രയോഗിച്ചാൽ താഴെ കാണുന്ന (5), (6) എന്നീ സമവാക്യങ്ങൾ കിട്ടുന്നു.

$(X + Y) + (\overline{X+Y}) = 1$ _____ (5)

$(X + Y) \cdot (\overline{X+Y}) = 0$ _____ (6)

ഡി മോർഗന്റെ ഒന്നാമത്തെ സിദ്ധാന്തം ശരിയാണെന്നു നമ്മൾ അനുമാനിക്കുക. അങ്ങനെയാണെങ്കിൽ (5), (6) എന്നീ സമവാക്യങ്ങളിൽ $(\overline{X}+\overline{Y})$ എന്നതിന് പകരം $(\overline{X}.\overline{Y})$ എന്ന് കൊടുക്കാമല്ലോ. എങ്കിൽ (5), (6) എന്നീ സമവാക്യങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നതുപോലെ മാറ്റിയെഴുതാം.

$(X+ Y) + (\overline{X} \cdot \overline{Y}) = 1$_____ (7)

$(X + Y) \cdot (\overline{X} \cdot \overline{Y}) = 0$_____ (8)

(7), (8) എന്നീ സമവാക്യങ്ങൾ ഓരോന്നും പ്രത്യേകം തെളിയിച്ചാൽ, ആ സമവാക്യങ്ങൾ രൂപീകരിക്കാൻ നമ്മൾ നടത്തിയ അനുമാനവും ശരിയാണെന്ന് നമുക്ക് പരിഗണിക്കാം. അതായത് (7), (8) സമവാക്യങ്ങൾ ശരിയാണെങ്കിൽ ഡി മോർഗന്റെ സിദ്ധാന്തവും ശരിയാണ്.

സമവാക്യം (7) ന്റെ LHS പരിഗണിക്കുക.

$(X + Y) + (\overline{X} . \overline{Y})$ $= (X + Y + \overline{X}) . (X + Y + \overline{Y})$ *(വിതരണ നിയമം (Distributive Law))*

$= (X + \overline{X} + Y) . (X + Y + \overline{Y})$ *(സംയോജന നിയമം (Associative Law))*

$= (1 + Y) . (X + 1)$ *(പൂരക നിയമം (Complimentary Law))*

$= 1 . 1$ *(സങ്കലന അനന്യത (Additive Identity))*

$= 1$

$=$ RHS

സമവാക്യം (8) ന്റെ LHS പരിഗണിക്കുക.

$(X + Y) . (\overline{X} . \overline{Y})$ $= (X . \overline{X} . \overline{Y}) + (Y . \overline{X} . \overline{Y})$ *(വിതരണ നിയമം (Distributive Law))*

$= (X . \overline{X} . \overline{Y}) + (Y . \overline{Y} . \overline{X})$ *(സംയോജന നിയമം (Associative Law))*

$= (0 . \overline{Y}) + (0 . \overline{X})$ *(പൂരക നിയമം (Complimentary Law))*

$= 0 + 0$ *(സങ്കലന അനന്യത (Additive Identity))*

$= 0$

$=$ RHS

(7), (8) എന്നീ സമവാക്യങ്ങൾ ബീജഗണിതത്തിലൂടെ നമ്മൾ തെളിയിച്ചിരിക്കുന്നു. ഇത് അർഥമാക്കുന്നത് ഡി മോർഗന്റെ ഒന്നാമത്തെ സിദ്ധാന്തം തെളിയിച്ചിരിക്കുന്നു എന്നാണ്. ഈ സിദ്ധാന്തം ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചും തെളിയിക്കാവുന്നതാണ്. അത് നിങ്ങൾ ചെയ്ത് പരിശീലിക്കുക.

**രണ്ടാമത്തെ സിദ്ധാന്തത്തിന്റെ ബീജഗണിത രീതിയിലുള്ള തെളിവ്**

നമുക്ക് തെളിയിക്കേണ്ടത് $\overline{X . Y} = \overline{X} + \overline{Y}$ എന്നാണ്

എന്ന് നമുക്ക് അനുമാനിക്കാം. $Z = X . Y$ _____ (11)

എങ്കിൽ, $\overline{Z} = \overline{X . Y}$ _____ (12)

$Z + \overline{Z} = 1$ _____ (13)

$Z . \overline{Z} = 0$ _____ (14)

പൂരക നിയമ പ്രകാരം സമവാക്യം (13) ഉം (14) ഉം ശരിയാണെന്നു നമുക്കറിയാം.

$(X . Y) + (\overline{X . Y}) = 1$ _____ (15)

$(X . Y) . (\overline{X . Y}) = 0$ _____ (16)

ഡി മോർഗന്റെ ഒന്നാമത്തെ സിദ്ധാന്തം ശരിയാണെന്ന് നമ്മൾ അനുമാനിക്കുക. അങ്ങനെയാണെങ്കിൽ (15), (16) എന്നീ സമവാക്യങ്ങളിൽ $(\overline{X . Y})$ എന്നതിന് പകരം $(\overline{X} + \overline{Y})$ എന്ന് കൊടുക്കാമല്ലോ. എങ്കിൽ (15), (16) എന്നീ സമവാക്യങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നതുപോലെ മാറ്റിയെഴുതാം.

$(X . Y) + (\overline{X} + \overline{Y}) = 1$ _____ (17)

$(X . Y) . (\overline{X} + \overline{Y}) = 0$ _____ (18)

(17), (18) എന്നീ സമവാക്യങ്ങൾ ഓരോന്നും പ്രത്യേകം തെളിയിച്ചാൽ, ആ സമവാക്യങ്ങൾ രൂപീകരിക്കാൻ നമ്മൾ നടത്തിയ അനുമാനവും ശരിയാണെന്ന് നമുക്ക് പരിഗണിക്കാം. അതായത് (17), (18) സമവാക്യങ്ങൾ ശരിയാണെങ്കിൽ ഡി മോർഗന്റെ സിദ്ധാന്തവും ശരിയാണ്.

സമവാക്യം (17) ന്റെ LHS പരിഗണിക്കുക.

$(X . Y) + (\overline{X} + \overline{Y})$ $= (\overline{X} + \overline{Y}) + (X . Y)$ *(ക്രമനിയമം (Commutative law))*

$= (\overline{X} + \overline{Y} + X) . (\overline{X} + \overline{Y} + Y)$ *(വിതരണ നിയമം (Distributive Law))*

$= (\overline{X} + X + \overline{Y}) . (\overline{X} + \overline{Y} + Y)$ *(സംയോജന നിയമം (Associative Law))*

$= (1 + \overline{Y}) . (\overline{X} + 1)$ *(പൂരക നിയമം (Complimentary Law))*

$= 1 . 1$ *(സങ്കലന അനന്യത (Additive Identity))*

$= 1$

$=$ RHS

സമവാക്യം (18) ന്റെ LHS പരിഗണിക്കുക.

$(X . Y) . (\overline{X} + \overline{Y})$ $= (X . Y . \overline{X}) + (X . Y . \overline{Y})$ *(വിതരണ നിയമം (Distributive Law))*

$= (X . \overline{X} . Y) + (X . Y . \overline{Y})$ *(സംയോജന നിയമം (Associative Law))*

$= (0 . Y) + (X . 0)$ *(പൂരക നിയമം (Complimentary Law))*

$= 0 + 0$ *(സങ്കലന അനന്യത (Additive Identity))*

$= 0$

$=$ RHS

(17), (18) എന്നീ സമവാക്യങ്ങൾ ബീജഗണിതത്തിലൂടെ നമ്മൾ തെളിയിച്ചിരിക്കുന്നു. ഇത് അർഥമാക്കുന്നത് ഡി മോർഗന്റെ രണ്ടാമത്തെ സിദ്ധാന്തം തെളിയിച്ചിരിക്കുന്നു എന്നാണ്. ഈ സിദ്ധാന്തം ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചും തെളിയിക്കാവുന്നതാണ്. അത് നിങ്ങൾ ചെയ്ത് പരിശീലിക്കുക.

ചുവടെ കാണിച്ചിരിക്കുന്നതുപോലെ ഡി മോർഗൻന്റെ സിദ്ധാന്തം ഉപയോഗിച്ച് എത്ര വേരിയബിളുകളെയും നമുക്ക് വിപുലീകരിച്ചെഴുതാം.

$$\overline{A+B+C+D+\ldots} = \overline{A} . \overline{B} . \overline{C} . \overline{D} . \ldots$$

$$\overline{A.B.C.D\ldots} = \overline{A} + \overline{B} + \overline{C} + \overline{D} + \ldots$$

മുകളിൽ കൊടുത്തിരിക്കുന്ന പദപ്രയോഗങ്ങൾ ഡി മോർഗൻസ് സിദ്ധാന്തം ഉപയോഗിച്ചുള്ള രൂപമാറ്റത്തെ പ്രതിനിധീകരിക്കുന്നുണ്ടെങ്കിലും ചുവടെ കൊടുത്തിരിക്കുന്ന ഘട്ടങ്ങളിലൂടെ ഈ പ്രക്രിയയെ കുറെകൂടി ലളിതമാക്കാം.

(i) ഫൻങ്ക്ഷനെ മുഴുവനായും പൂരകമാക്കുക.

(ii) എല്ലാ AND (.) കളെയും OR (+) ആയും എല്ലാ OR (+) കളെയും AND (.) ആയും മാറ്റുക.

(iii) ഓരോ വേരിയബിളിനെയും പ്രത്യേകം പൂരകമാക്കുക. ഈ പ്രക്രിയയെ ഡിമോർഗനൈസേഷൻ (Demorganisation) എന്ന് വിളിക്കുന്നു. ലളിതമായി പറഞ്ഞാൽ ഡിമോർഗനൈസേഷൻ എന്നത് 'ചിഹ്നങ്ങൾ മാറ്റിക്കൊണ്ട് വാക്യങ്ങളെ വിഘടിപ്പിക്കുക' (Break the line cahnge the sign) എന്നാകുന്നു.

**സ്വയം വിലയിരുത്താം**

1. A.B+B.C=1 എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ദ്വന്ദങ്ങൾ കണ്ടുപിടിക്കുക.?
2. A+A=A എന്ന് പ്രസ്താവിക്കുന്ന ബൂളിയൻ നിയമത്തിന്റെ പേരെഴുതുക.
   (a) ക്രമ നിയമം (Commutative law) (b) വർഗസമ നിയമം (Idempotent Law) (c) സ്വാംശീകരണ നിയമം (Absorption Law)
3. ഡി മോർഗന്റെ സിദ്ധാന്തം പ്രസ്താവിക്കുക.

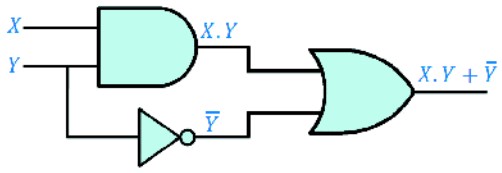
## 2.9 ലളിതമായ ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകല്പന (Circuit designing for simple Boolean expression)

അടിസ്ഥാന ഗേറ്റുകൾ ഉപയോഗിച്ച് ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകല്പന ചെയ്യാം. A.B എന്ന ബൂളിയൻ പദപ്രയോഗത്തെ പ്രതിനിധീകരിക്കുന്നതിന് AND ഗേറ്റും, A+B യെ പ്രതിനിധീകരിക്കുന്നതിന് OR ഗേറ്റും, $\bar{A}$ നെ പ്രതിനിധീകരിക്കുന്നതിന് NOT ഗേറ്റും ഉപയോഗിക്കാമെന്നു നമ്മൾ മനസ്സിലാക്കിക്കഴിഞ്ഞല്ലോ. അതുപോലെ മറ്റ് ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകൽപ്പന ചെയ്യുന്നത് എങ്ങനെയാണെന്ന് നമുക്ക് നോക്കാം.

$\bar{A}$ + B എന്ന ബൂളിയൻ പദപ്രയോഗം പരിഗണിക്കുക. ഇതിൽ രണ്ട് ഇൻപുട്ടുകളുള്ള ഒരു ഓപ്പറേഷനും അതിൽ ഒരു ഇൻപുട്ട് വിപരീതമായതുമാണ് (Inverted). അതുകൊണ്ട് ഈ സർക്യൂട്ടിന്റെ രേഖാചിത്രം ചിത്രം 2.12 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ വരയ്ക്കാം.

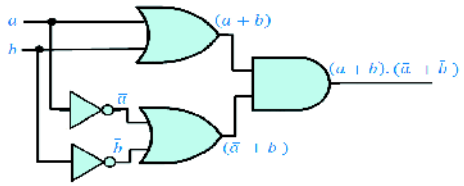

*ചിത്രം 2.12 : f (A, B) = $\bar{A}$+B*

**ഉദാഹരണം:** $f(X, Y) = X.Y + \bar{Y}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് ഉണ്ടാക്കുക.



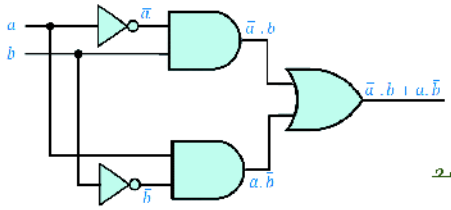

*ചിത്രം 2.13 : f (X, Y) = X.Y + $\bar{Y}$*

**ഉദാഹരണം:** $f(a, b) = (a + b) . (\bar{a} + \bar{b})$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് ഉണ്ടാക്കുക.




*ചിത്രം 2.14 : f (a, b) = (a + b) . ($\bar{a}$ + $\bar{b}$)*

**ഉദാഹരണം:** $\bar{a}.b + a.\bar{b}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് ഉണ്ടാക്കുക.



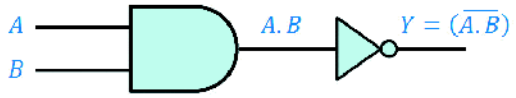

*ചിത്രം 2.15 : f (a, b) = $\bar{a}$ .b + a. $\bar{b}$*

## 2.10 യൂണിവേഴ്സൽ ഗേറ്റ് (Universal gates)

NAND, NOR എന്നീ ഗേറ്റുകളാണ് യൂണിവേഴ്സൽ ഗേറ്റ് എന്ന് അറിയപ്പെടുന്നത്. മറ്റേതൊരു ഗേറ്റും ഉപയോഗിക്കാതെ ഏതൊരു ബൂളിയൻ ഫങ്ഷനും രൂപകല്പന ചെയ്യാൻ കഴിയുന്ന ഗേറ്റ് ആണ് യൂണിവേഴ്സൽ ഗേറ്റ്. ഭൂരിഭാഗം ഡിജിറ്റൽ ഇന്റഗ്രേറ്റഡ് ചിപ്പ് (IC) കളിലും അടിസ്ഥാന ഗേറ്റുകളായി ഉപയോഗിക്കുന്നത് NAND, NOR എന്നീ ഗേറ്റുകളാണ്, എന്തുകൊണ്ടെന്നാൽ ഈ ഗേറ്റുകൾ ചെലവുകുറഞ്ഞതും എളുപ്പത്തിൽ നിർമ്മിക്കാവുന്നതുമാണ്.

### 2.10.1 NAND ഗേറ്റ്

AND ഗേറ്റിന്റെ ഔട്ട്പുട്ട് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കുന്ന (Inverted) ഗേറ്റ് ആണ് NAND ഗേറ്റ്. ചിത്രം 2.16 NAND ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട് സജ്ജീകരണം കാണിക്കുന്നു. A, B എന്നിവ AND ഗേറ്റിന്റെ ഇൻപുട്ടുകളും A.B എന്നത് അതിന്റെ ഔട്ട്പുട്ടുമായാൽ NAND ഗേറ്റിന്റെ ഔട്ട്പുട്ട് എന്നത് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കിയ $Y = \overline{(A.B)}$ ആണ്. അതുകൊണ്ട് NAND ഗേറ്റിന്റെ ലോജിക്കൽ പദപ്രയോഗം $\overline{(A.B)}$ എന്നാകുന്നു.

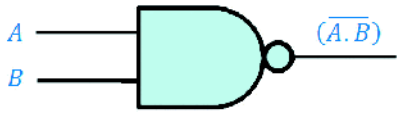


ചിത്രം 2.16 NAND ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട്

| A | B | Y= (A.B) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

പട്ടിക 2.26 NAND ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ

NAND ഗേറ്റിന്റെ ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആയാൽ അതിന്റെ ഔട്ട്പുട്ടും 1 ആയിരിക്കുമെന്ന് ട്രൂത്ത് ടേബിളിലൂടെ (പട്ടിക 2.26) കാണിച്ചു തരുന്നു. എല്ലാ ഇൻപുട്ടുകളും 1 ആകുമ്പോൾ മാത്രമാണ് അതിന്റെ ഔട്ട്പുട്ട് 0 ആകുന്നത്. NAND ഗേറ്റ് എന്നത് AND ഗേറ്റിന്റെ വിപരീത പ്രവർത്തനമാണ് ചെയ്യുന്നത്. അതുകൊണ്ട് NAND ഗേറ്റ് ഇൻവെർട്ടഡ് (Inverted) AND ഗേറ്റ് എന്നറിയപ്പെടുന്നു.

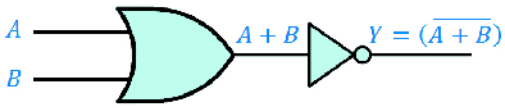


ചിത്രം 2.17 NAND ഗേറ്റ്

NAND ഗേറ്റിന്റെ പ്രതീകം ചിത്രം 2.17 ൽ കാണിച്ചിരിക്കുന്നു. AND ഗേറ്റിന്റെ പ്രതീകത്തിന്റെ ഔട്ട്പുട്ടിൽ ഒരു ചെറു വൃത്തം ചേർക്കുമ്പോൾ അത് NAND ഗേറ്റിന്റെ പ്രതീകം ആകും.

### 2.10.2 NOR ഗേറ്റ്

OR ഗേറ്റിന്റെ ഔട്ട്പുട്ട് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കുന്ന ഗേറ്റ് ആണ് NOR ഗേറ്റ്. ചിത്രം 2.18 ൽ NOR ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട് സജ്ജീകരണം കാണിക്കുന്നു. A, B എന്നിവ OR ഗേറ്റിന്റെ ഇൻപുട്ടുകളും A + B എന്നത് അതിന്റെ ഔട്ട്പുട്ടുമായാൽ NOR ഗേറ്റിന്റെ ഔട്ട്പുട്ട് എന്നത് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കിയ $Y = \overline{(A+B)}$ ആണ്. അതുകൊണ്ട് NOR ഗേറ്റിന്റെ ലോജിക്കൽ പദപ്രയോഗം $\overline{(A+B)}$ എന്നാകുന്നു.



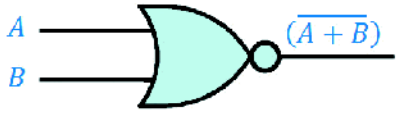
ചിത്രം 2.18 NOR ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട്

NOR ഗേറ്റിൽ രണ്ട് ഇൻപുട്ടുകളും 0 ആകുമ്പോൾ മാത്രമാണ് ഇതിന്റെ ഔട്ട്പുട്ട് 1 ആകുന്നതെന്ന് ട്രൂത്ത് ടേബിളിലൂടെ (പട്ടിക 2.27) കാണിച്ചു തരുന്നു. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആയാൽ അതിന്റെ ഔട്ട്പുട്ട് 0 ആയിരിക്കും. NOR ഗേറ്റ് എന്നത് OR ഗേറ്റിന്റെ വിപരീത പ്രവർത്തനമാണ് ചെയ്യുന്നത്. അതുകൊണ്ട് ഇതിനെ OR ന്റെ വിപരീതം (Inverted) എന്നുകൂടി പറയുന്നു.

| A | B | $Y=\overline{(A+B)}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

*പട്ടിക 2.27 NOR ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ*

NOR ഗേറ്റിന്റെ പ്രതീകം ചിത്രം 2.19 ൽ കാണിച്ചിരിക്കുന്നു. OR ഗേറ്റിന്റെ ഔട്ട്പുട്ടിൽ ചെറിയ വൃത്തത്തോടു കൂടിയതാണ് NOR ന്റെ പ്രതീകം.



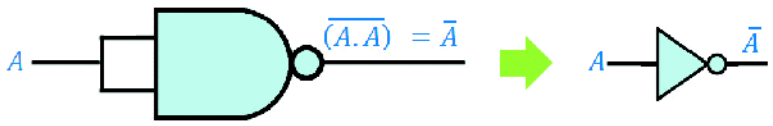

*ചിത്രം 2.19 NOR ഗേറ്റ്*

## 2.10.3 NAND ഗേറ്റും NOR ഗേറ്റും ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകളുടെ രൂപകല്പന (Implementation of basic gates using NAND and NOR)

NAND അല്ലെങ്കിൽ NOR ഗേറ്റ് ഉപയോഗിച്ച് എല്ലാ അടിസ്ഥാന ഗേറ്റുകളും നമുക്ക് രൂപകല്പന ചെയ്യാം. NAND ഗേറ്റ് ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകൾ രൂപകല്പന ചെയ്യുന്നത് എങ്ങനെയെന്ന് നമുക്ക് നോക്കാം.

### NAND ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.20 ൽ കാണുന്നതുപോലെ ഒരു NAND ഗേറ്റിന്റെ എല്ലാ ഇൻപുട്ടുകളിലും ഒരേ വില നൽകികൊണ്ട് NOT ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.20 NAND ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

$$A \text{ NAND } A = \overline{(A.A)}$$

എന്തുകൊണ്ടെന്നാൽ $= \bar{A}$ $A.A = A$

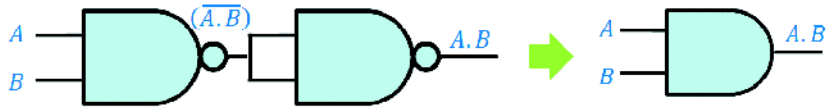
NAND ഗേറ്റ് കൊണ്ട് NOT ഗേറ്റ് ഉണ്ടാക്കുവാൻ കഴിയുമെന്ന് താഴെ കൊടുത്തിരിക്കുന്ന ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ (പട്ടിക 2.28) തെളിയിക്കുന്നു.

| A | AA | $\overline{(A.A)}$ | $\bar{A}$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

*പട്ടിക 2.28 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## NAND ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.21 ൽ കാണുന്നതുപോലെ ഒരു NAND ഗേറ്റിനു തുടർച്ചയായി, ഔട്ട്പുട്ട് വിപരീതമാക്കുവാൻ മറ്റൊരു NAND ഗേറ്റ് ഉപയോഗിച്ച്, AND ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.21NAND ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

നമുക്കറിയാം A NAND B $= (\overline{A.B})$

(A NAND B) NAND (A NAND B) $= (\overline{A.B})$ NAND $(\overline{A.B})$

$= \overline{((\overline{AB}).(\overline{AB}))}$

$= ((\overline{\overline{AB}}))$ എന്തുകൊണ്ടെന്നാൽ $A.A = A$

$= A.B$ എന്തുകൊണ്ടെന്നാൽ $(\overline{\overline{A}}) = A$

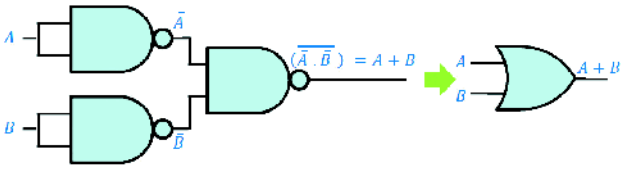
NAND ഗേറ്റ് കൊണ്ട് AND ഗേറ്റ് ഉണ്ടാക്കാൻ കഴിയുമെന്ന് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ (പട്ടിക 2.29) തെളിയിക്കുന്നു.

| A | B | A.B | $(\overline{A.B})$ | $(\overline{A.B}) \cdot (\overline{A.B})$ | $\overline{((\overline{AB}).(\overline{AB}))}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

*പട്ടിക 2.29 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## NAND ഗേറ്റ് ഉപയോഗിച്ച് OR ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.22 ൽ കാണുന്നതുപോലെ NAND ഗേറ്റിന്റെ എല്ലാ ഇൻപുട്ടുകളെയും പൂരകങ്ങൾ ആക്കി, OR ഗേറ്റിനെ NAND ഗേറ്റ് ഉപയോഗിച്ച് പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.22 NAND ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

$$A \text{ NAND } A = (\overline{A.A})$$
$$= \overline{A}$$

അതുപോലെ, $B \text{ NAND } B = \overline{B}$

അതുകൊണ്ട്, $(A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B) = \overline{A} \text{ NAND } \overline{B}$

$$= (\overline{\overline{A}.\overline{B}})$$

$= \overline{\overline{A}} + \overline{\overline{B}}$ എന്തുകൊണ്ടെന്നാൽ $(\overline{A.B}) = \overline{A} + \overline{B}$

$= A + B$ എന്തുകൊണ്ടെന്നാൽ $(\overline{\overline{A}}) = A$

NAND ഗേറ്റ് കൊണ്ട് OR ഗേറ്റ് ഉണ്ടാക്കുവാൻ കഴിയുമെന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിൽ (പട്ടിക 2.30) കാണിച്ചിരിക്കുന്നു.

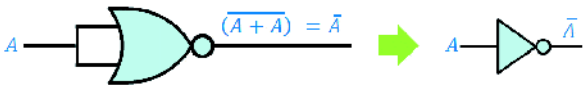
| A | B | $\overline{A}$ | $\overline{B}$ | $\overline{A} \cdot \overline{B}$ | $(\overline{\overline{A}.\overline{B}})$ | A + B |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |

*പട്ടിക 2.30 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

NAND ഗേറ്റ് ഒരു യൂണിവേഴ്സൽ ഗേറ്റ് ആകുന്നു. എന്തുകൊണ്ടെന്നാൽ ഇതുപയോഗിച്ച് AND, OR, NOT എന്നീ അടിസ്ഥാന ഗേറ്റുകൾ ഉണ്ടാക്കുവാൻ കഴിയുന്നു. NOR എന്ന മറ്റൊരു യൂണിവേഴ്സൽ ഗേറ്റ് ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകൾ എങ്ങനെ രൂപകല്പന ചെയ്യാം എന്ന് നോക്കാം.

## NOR ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.23 ൽ കാണുന്നതുപോലെ ഒരു NOR ഗേറ്റിന്റെ രണ്ടു ഇൻപുട്ടുകളിലും ഒരേ വില നൽകികൊണ്ട്, NOT ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.23 NOR ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

$A \text{ NOR } A = (\overline{A + A})$

$= \overline{A}$ എന്തുകൊണ്ടെന്നാൽ

$A + A = A$

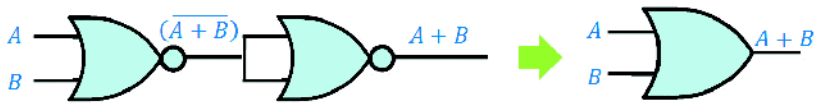
NOR ഗേറ്റ് കൊണ്ട് NOT ഗേറ്റ് ഉണ്ടാക്കാൻ കഴിയുന്നു എന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ പട്ടിക 2.31 ൽ കൊടുത്തിരിക്കുന്നു.

| A | A+A | $(\overline{A + A})$ | $\overline{A}$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

*പട്ടിക 2.31 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## NOR ഗേറ്റ് ഉപയോഗിച്ച് OR ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.24 ൽ കാണുന്നതുപോലെ ഒരു NOR ഗേറ്റിനു തുടർച്ചയായി മറ്റൊരു NOR ഗേറ്റ് ഉപയോഗിച്ച്, OR ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.24 NOR ഗേറ്റ് ഉപയോഗിച്ച് OR ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

നമുക്കറിയാം $A \text{ NOR } B = (\overline{A+B})$

അതുകൊണ്ട് $(A \text{ NOR } B) \text{ NOR } (A \text{ NOR } B) = (\overline{A+B}) \text{ NOR } (\overline{A+B})$

$= \overline{((\overline{A+B})+(\overline{A+B}))}$

$= ((\overline{\overline{A+B}}))$ എന്തുകൊണ്ടെന്നാൽ $A+A=A$

$= A+B$ എന്തുകൊണ്ടെന്നാൽ $(\overline{\overline{A}}) = A$

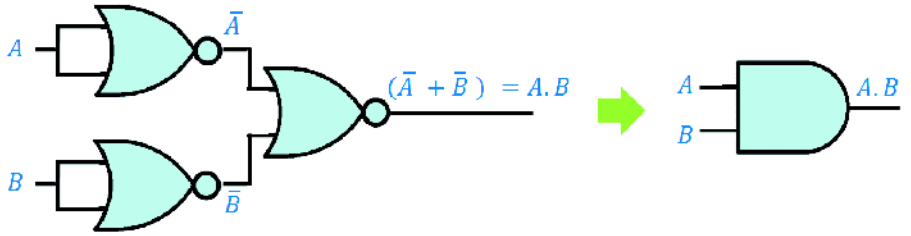
NOR ഗേറ്റ് കൊണ്ട് OR ഗേറ്റ് ഉണ്ടാക്കാൻ കഴിയുന്നു എന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ പട്ടിക 2.32 ൽ കാണിച്ചിരിക്കുന്നു.

| A | B | A+B | $(\overline{A+B})$ | $(\overline{A+B}) + (\overline{A.B})$ | $((\overline{A+B}).(\overline{A+B}))$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

*പട്ടിക 2.32 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## NOR ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.25 ൽ കാണുന്നതുപോലെ NOR ഗേറ്റിന്റെ എല്ലാ ഇൻപുട്ടുകളും പൂരകങ്ങൾ ആക്കി, AND ഗേറ്റിനെ NOR ഗേറ്റ് ഉപയോഗിച്ച് പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.25 NOR ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

$A \text{ NOR } A = (\overline{A+A}) = \overline{A}$

അതുപോലെ, $B \text{ NOR } B = (\overline{B+B}) = \overline{B}$

അതുകൊണ്ട്, $(A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B) = \overline{A} \text{ NOR } \overline{B}$

$= \overline{(\overline{A}+\overline{B})}$

$= (\overline{\overline{A}}) . (\overline{\overline{B}})$ എന്തുകൊണ്ടെന്നാൽ $(\overline{A+B}) = \overline{A} . \overline{B}$

$= A.B$ എന്തുകൊണ്ടെന്നാൽ $\overline{\overline{A}} = A$

NOR ഗേറ്റ് ഒരു യൂണിവേഴ്സൽ ഗേറ്റ് ആകുന്നു എന്തുകൊണ്ടെന്നാൽ ഇതുപയോഗിച്ച് AND, OR, NOT എന്നീ അടിസ്ഥാന ഗേറ്റുകളുടെ പ്രവർത്തനങ്ങൾ നടപ്പിലാക്കുവാൻ കഴിയുന്നു. NOR ഗേറ്റ് കൊണ്ട് AND ഗേറ്റ് ഉണ്ടാക്കുവാൻ കഴിയുന്നു എന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ പട്ടിക 2.33 ൽ കാണിച്ചിരിക്കുന്നു.

| A | B | $\overline{A}$ | $\overline{B}$ | $\overline{A}+\overline{B}$ | $(\overline{\overline{A}+\overline{B}})$ | A.B |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |

*പട്ടിക 2.33 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## നമുക്ക് പരിശോധിക്കാം

1. $X+\overline{Y}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് വരയ്ക്കുക.
2. യൂണിവേഴ്സൽ ഗേറ്റുകൾ ഏതൊക്കെയാണ്?.
3. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആകുമ്പോൾ ഔട്ട്പുട്ട് 0 തരുന്ന ഗേറ്റ് _____ ആകുന്നു.
   (a) OR (b) AND (c) NAND (d) NOR
4. A NAND B = _____.
   (a) A+B (b) A.B (c) $(\overline{A+B})$ (d) $(\overline{A.B})$

## നമുക്ക് സംഗ്രഹിക്കാം

വിവിധ രീതിയിലുള്ള ഡാറ്റ പ്രതിനിധീകരണത്തെ കുറിച്ച് ഈ അധ്യായത്തിൽ ചർച്ച ചെയ്തു. ഡാറ്റ പ്രതിനിധീകരണം ചർച്ച ചെയ്യുന്നതിനുമുമ്പ് വ്യത്യസ്ത സംഖ്യാ സമ്പ്രദായങ്ങളും അവയുടെ പരിവർത്തനങ്ങളും പരിചയപ്പെട്ടു. അതിനു ശേഷം പൂർണ സംഖ്യ, ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യ, എന്നിവയുടെ പ്രതിനിധീകരണവും, ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിവയുടെ പ്രതിനിധീകരണത്തിന്റെ വിവിധ രീതികളും നാം മനസ്സിലാക്കി. ബൂളിയൻ ബീജഗണിതത്തിലെ ആശയങ്ങൾക്കൊപ്പം ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ, ഗേറ്റുകൾ, ബൂളിയൻ നിയമങ്ങൾ എന്നിവ നാം ചർച്ച ചെയ്തു. അടിസ്ഥാന ലോജിക് സർക്യൂട്ട് രൂപകല്പന ചെയ്യുന്ന രീതികൾ മനസ്സിലാക്കുന്നതോടൊപ്പം യൂണിവേഴ്സൽ ഗേറ്റിന്റെ പ്രാധാന്യവും ചർച്ച ചെയ്തുകൊണ്ടാണ് ഈ അധ്യായം അവസാനിപ്പിക്കുന്നത്.

## പഠനനേട്ടങ്ങൾ

ഈ അധ്യായം പൂർത്തിയാകുന്നതോടുകൂടി പഠിതാവിനു താഴെ പറയുന്നവ സാധ്യമാകും.

- വിവിധ സംഖ്യാ സമ്പ്രദായങ്ങളുടെ സവിശേഷതകൾ വിശദീകരിക്കാൻ.
- ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽനിന്ന് മറ്റൊരു സംഖ്യാ സമ്പ്രദായത്തിലേക്ക് പരിവർത്തനം ചെയ്യാൻ.
- ബൂളിയൻ ഗണിതങ്ങൾ ചെയ്യാൻ.
- സംഖ്യകളും അക്ഷരങ്ങളും കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കാൻ.
- ശബ്ദം, ചിത്രം, വീഡിയോ എന്നീ ഫയലുകളുടെ ഘടനകൾ.
- ബൂളിയൻ ബീജഗണിതത്തിന്റെ ആശയങ്ങൾ.
- ലോജിക് ഓപ്പറേറ്ററുകളുടെയും ഗേറ്റുകളുടെയും പ്രവർത്തനങ്ങൾ ഉദാഹരണ സഹിതം വിശദീകരിക്കാൻ.
- ബൂളിയൻ ബീജഗണിതത്തിലെ അംഗീകൃതതത്വങ്ങൾ, സിദ്ധാന്തങ്ങൾ, നിയമങ്ങൾ എന്നിവ പ്രസ്താവിക്കാനും തെളിയിക്കാനും.
- ലളിതമായ അടിസ്ഥാന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകല്പന ചെയ്യാൻ.
- യൂനിവേഴ്സൽ ഗേറ്റുകൾ ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകളുടെ രൂപകല്പന.

## മാതൃകചോദ്യങ്ങൾ

### ഒറ്റ വാചകത്തിൽ ഉത്തരമെഴുതുക.

1. $(296)_{10}$ എന്ന സംഖ്യയിൽ 9 ന്റെ സ്ഥാനവില എത്രയാണ്?
2. 55 എന്ന ദശസംഖ്യ (Decimal) ക്ക് തുല്യമായ അഷ്ടസംഖ്യ (Octal) കണ്ടുപിടിക്കുക.
3. താഴെ കൊടുത്തിരിക്കുന്ന ശ്രേണികളിൽ വിട്ടുപോയ സംഖ്യകൾ പൂരിപ്പിക്കുക.
   a) $101_2$ , $1010_2$, $1111_2$, _____, _____
   b) $15_8$ , $16_8$, $17_8$, _____, _____.
   c) $18_{16}$, $1A_{16}$, $1C_{16}$ ,_____, _____.
4. If $(X)_2 - (1010)_2 = (1000)_2$ ആയാൽ X ന്റെ വില കണ്ടുപിടിക്കുക.
5. ലോകത്തിലെ എല്ലാ ലിഖിതഭാഷകളിലെയും അക്ഷരങ്ങളെ പ്രതിനിധാനം ചെയ്യാൻ ഉപയോഗിക്കുന്ന സംവിധാനത്തിന്റെ പേരെഴുതുക.
6. താഴെ കൊടുത്തിരിക്കുന്നവയിൽ നിന്ന് ലോജിക്കൽ പ്രസ്താവനകൾ കണ്ടുപിടിക്കുക.
   a) നിങ്ങൾ എന്തുകൊണ്ടാണ് വൈകിയത്?
   b) നിങ്ങൾ എന്നോടൊപ്പം കമ്പോളത്തിൽ വരാൻ തയ്യാറാണോ?
   c) ഇന്ത്യ എന്റെ രാജ്യം ആകുന്നു.
   d) ക്ലാസ് മുറിയിലേക്ക് പോകുക.
7. മൂന്ന് അടിസ്ഥാന ഗേറ്റുകളുടെ പേരെഴുതുക.
8. ഇൻവെർട്ടർ എന്നറിയപ്പെടുന്ന ഗേറ്റ് ഏതാണ്?
9. രണ്ട് പൂരക നിയമങ്ങൾ എഴുതുക.
10. $\overline{(A+B)}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തെ _____ ഗേറ്റ് പ്രതിനിധാനം ചെയ്യുന്നു.
    a) AND  b) NOR  c) OR  d) NAND

### ഒന്നോ രണ്ടോ വാക്യത്തിൽ ഉത്തരമെഴുതുക.

1. ഡാറ്റ പ്രതിനിധാനം എന്ന പദം നിർവചിക്കുക.
2. സംഖ്യാ സമ്പ്രദായം എന്നത് കൊണ്ട് നിങ്ങൾ അർഥമാക്കുന്നത് എന്താണ്? ഏതെങ്കിലും നാല് സംഖ്യാ സമ്പ്രദായങ്ങൾ എഴുതുക.
3. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ മറ്റു മൂന്ന് സംഖ്യാ സമ്പ്രദായങ്ങളിലേക്കു മാറ്റിയെഴുതുക.
   a) $(125)_8$  b) 98  c) $(101110)_2$  d) $(A2B)_{16}$
4. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ മറ്റു മൂന്ന് സംഖ്യാ സംമ്പ്രദായങ്ങളിലേക്ക് പരിവർത്തനം ചെയ്യുക.
   a) $(7F.1)_{16}$  b) $(207.13)_8$  c) 93.25  d) $(10111011.1101)_2$

5. If $(X)_2 = (Y)_8 = (Z)_{16} = (28)_{10}$ ആയാൽ X, Y, Z എന്നിവയുടെ വില കണ്ടുപിടിക്കുക.

6. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ അവരോഹണക്രമത്തിലാക്കുക.

   a) $(101)_{16}$ b) $(110)_{10}$ c) $(111000)_2$ d) $(251)_8$

7. $(X)_2 = (10111)_2 + (11011)_2 - (11100)_2$ ആയാൽ X വില കണ്ടുപിടിക്കുക.

8. പൂർണസംഖ്യകൾ കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധാനം ചെയ്യുവാൻ ഉപയോഗിക്കുന്ന രീതികൾ എന്തെല്ലാമാണ് ?

9. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ ചിഹ്നവും മൂല്യവും, 1 ന്റെ പൂരകം, 2 ന്റെ പൂരകം എന്നീ രീതികളിൽ പ്രതിനിധീകരിക്കുക.

   a) -19 b) +49 c) -97 d) -127

10. ചിഹ്നവും മൂല്യവും രീതിയിൽ പ്രതിനിധാനം ചെയ്ത $(10011001)_2$ എന്ന സംഖ്യയുടെ പൂർണസംഖ്യ കണ്ടുപിടിക്കുക.

11. 32 ബിറ്റ് ഉപയോഗിക്കുന്ന കമ്പ്യൂട്ടറിൽ ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുന്ന രീതി വിശദീകരിക്കുക.

12. കമ്പ്യൂട്ടർ മെമ്മറിയിൽ അക്ഷരങ്ങൾ പ്രതിനിധീകരിക്കുന്നതിനുള്ള രീതികൾ എന്തൊക്കെയാണ് ?

13. അക്ഷരങ്ങളുടെ പ്രതിനിധീകരണത്തിൽ യൂണികോഡിന്റെ പ്രാധാന്യം ചുരുക്കി വിവരിക്കുക.

14. ചേരുംപടി ചേർക്കുക:

| A | | B |
|---|---|---|
| i) | ഏതെങ്കിലും ഇൻപുട്ട് 1 ആയാൽ ഔട്ട്പുട്ട് 1 ആകുന്നു. | a) NAND |
| ii) | ഏതെങ്കിലും ഇൻപുട്ട് 0 ആയാൽ ഔട്ട്പുട്ട് 0 ആകുന്നു. | b) OR |
| iii) | ഏതെങ്കിലും ഇൻപുട്ട് 0 ആയാൽ ഔട്ട്പുട്ട് 1 ആകുന്നു. | c) NOR |
| iv) | ഏതെങ്കിലും ഇൻപുട്ട് 1 ആയാൽ ഔട്ട്പുട്ട് 0 ആകുന്നു. | d) AND |

15. താഴെ കൊടുത്തിരിക്കുന്ന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ ദ്വൈത (dual) രൂപങ്ങൾ എഴുതുക.

   a) X.Y+Z b) A.C+A.1+A.C c) $(A+0).(A.1.\overline{A})$

16. താഴെ കൊടുത്തിരിക്കുന്ന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ പൂരകം (complement) എഴുതുക.

   a) $\overline{A}\,\overline{B}$ b) $\overline{A.B} + \overline{C.D}$

17. താഴെ കൊടുത്തിരിക്കുന്ന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ ലോജിക് സർക്യൂട്ട് നിർമിക്കുക.

   (i) $\overline{ab} + c$ (ii) $ab + \bar{a}b + \overline{ab}$ (iii) $(a+\bar{b}).(\bar{a}+\bar{b})$

18. NAND, NOR എന്നീ ഗേറ്റുകളെ യൂണിവേഴ്സൽ ഗേറ്റുകൾ എന്ന് വിളിക്കുന്നത് എന്തുകൊണ്ടാണ്? ഉദാഹരണ സഹിതം സാധൂകരിക്കുക.

## ഉപന്യസിക്കുക

1. സംഖ്യകൾ കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കുന്നതിനുള്ള വിവിധ രീതികളെക്കുറിച്ച് വിശദീകരിക്കുക.
2. അക്ഷരങ്ങൾ കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കുന്നതിനുള്ള വിവിധ രീതികളെക്കുറിച്ച് വിശദീകരിക്കുക.
3. ശബ്ദം, ചിത്രം, വീഡിയോ എന്നീ ഫയലുകൾ കമ്പ്യൂട്ടറിൽ സംഭരിക്കുന്നതിന്റെ ഘടന വിവരിക്കുക.
4. മൂന്ന് ഇൻപുട്ടുകൾ ഉള്ള AND ഗേറ്റിന്റെ പ്രതീകം, ബൂളിയൻ പദപ്രയോഗം, ട്രൂത്ത് ടേബിൾ എന്നിവ എഴുതുക.
5. എല്ലാ അടിസ്ഥാന ഗേറ്റുകളും NOR ഗേറ്റ് ഉപയോഗിച്ച് നിർമ്മിച്ച് യൂണിവേഴ്സൽ ഗേറ്റ് ആണെന്ന് തെളിയിക്കുക.

3

# കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ ഘടകങ്ങൾ

## പ്രധാന ആശയങ്ങൾ

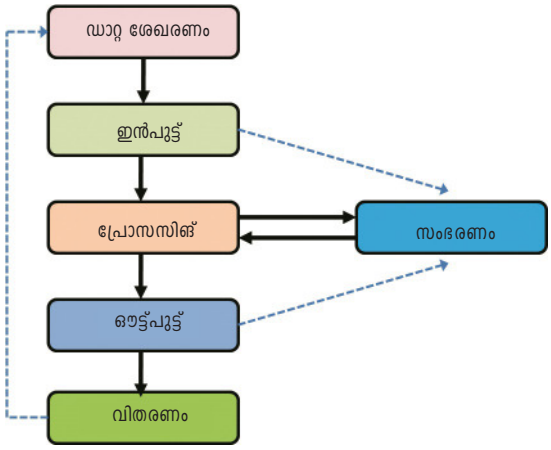
- ഡാറ്റ പ്രോസസ്സിങ്
- കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തന യൂണിറ്റുകൾ
- ഹാർഡ്‌വെയർ
  - പ്രോസസ്സറുകൾ
  - മദർബോർഡ്
  - പെരിഫെറലുകളും പോർട്ടുകളും
  - മെമ്മറി - പ്രാഥമിക മെമ്മറി, ദ്വിതീയ മെമ്മറി
  - കമ്പ്യൂട്ടറിൽ മെമ്മറിയുടെ ഉപയോഗം
  - ഇൻപുട്ട് ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ
- ഇ-വേസ്റ്റ്
  - നിർമാർജന മാർഗങ്ങൾ
  - ഹരിത കമ്പ്യൂട്ടിംഗ്
- സോഫ്റ്റ്‌വെയർ
  - സിസ്റ്റം സോഫ്റ്റ്‌വെയർ (ഓപ്പറേറ്റിങ്സിസ്റ്റം, ഭാഷ പ്രോസസ്സറുകൾ, യൂട്ടിലിറ്റി സോഫ്റ്റ്‌വെയർ)
  - ആപ്ലിക്കേഷൻ സോഫ്റ്റ്‌വെയർ (പൊതു ഉപയോഗം, പ്രത്യേക ഉപയോഗം)
  - സ്വതന്ത്ര ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്‌വെയർ
  - ഫ്രീവെയറും ഷെയർവെയറും
  - ഉടമസ്ഥാവകാശമുള്ള സോഫ്റ്റ്‌വെയർ (Proprietary software)
- ഹ്യൂമൺവെയർ/ലൈവ്‌വെയർ


ഇന്നത്തെ ലോകത്ത് നമുക്ക് കമ്പ്യൂട്ടറുകളും അതിന്റെ ഉപയോഗങ്ങളും പരിചയമുണ്ട്. നൽകുന്ന നിർദേശങ്ങൾക്കനുസരിച്ച് ഡാറ്റ സ്വീകരിക്കുകയും പ്രോസസ്സ് ചെയ്ത് ഔട്ട്പുട്ട് പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്ന വേഗതയേറിയ ഒരു ഇലക്ട്രോണിക് ഉപകരണമാണ് കമ്പ്യൂട്ടർ. ഒരു കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ അടിസ്ഥാന രൂപകൽപ്പനയുടെ അവലോകനം ഈ പാഠഭാഗത്ത് അവതരിപ്പിക്കുന്നു. ഒരു കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ വിവിധ ഘടകങ്ങൾ എങ്ങിനെ ക്രമീകരിക്കപ്പെട്ടിരിക്കുന്നു എന്നും, ഒരു പ്രത്യേക ചുമതല നിർവ്വഹിക്കാൻ ഏതെല്ലാം പ്രവർത്തനങ്ങൾ നടത്തുന്നു എന്നും ഇവിടെ ചർച്ച ചെയ്യുന്നു. ഒരു കമ്പ്യൂട്ടറിന് പ്രധാനമായും രണ്ട് ഘടകങ്ങളാണുള്ളത്. ഹാർഡ്‌വെയറും സോഫ്റ്റ്‌വെയറും. ഒരു കമ്പ്യൂട്ടർ സംവിധാനവുമായി ബന്ധപ്പെട്ട എല്ലാ ഭൗതിക ഘടകങ്ങളേയും ഹാർഡ്‌വെയർ എന്ന് സൂചിപ്പിക്കുന്നു. ഒരു നിർദ്ദിഷ്ട പ്രവൃത്തി ചെയ്യുന്ന ഹാർഡ്‌വെയറിനുള്ള ഒരു കൂട്ടം നിർദ്ദേശങ്ങളാണ് സോഫ്റ്റ്‌വെയർ. കമ്പ്യൂട്ടർ ഉപയോഗിച്ച് നിത്യജീവിതത്തിലെ ഏതെങ്കിലും പ്രശ്നം നമുക്ക് പരിഹരിക്കേണ്ടി വരുമ്പോൾ, ഡാറ്റ പ്രോസസ്സ് ചെയ്ത് അതിനാവശ്യമായ വിവരങ്ങൾ നാം ഉത്പാദിപ്പിക്കുന്നു. ഈ അധ്യായത്തിൽ ആദ്യം ഡാറ്റ പ്രോസസ്സിങ്ങിന്റെ ആശയങ്ങൾ അവതരിപ്പിക്കുകയും, ഡാറ്റ പ്രോസസ്സിങ്ങിന് ഒരു കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തന ഭാഗങ്ങൾ സഹായിക്കുന്നത് എങ്ങനെ എന്ന് ചർച്ചചെയ്യുകയും ചെയ്യുന്നു. അതിനുശേഷം വിവിധ ഹാർഡ്‌വെയർ ഘടകങ്ങൾ അവതരിപ്പിക്കുന്നു. അതിനു ശേഷം ഇലക്ട്രോണിക് വേസ്റ്റ്, അവയുടെ നിർമാർജ്ജന രീതികൾ, ഹരിത കമ്പ്യൂട്ടിങ്ങിന്റെ ആശയം എന്നിവ വിവരിക്കുന്നു. പിന്നീട് വിവിധതരത്തിലുള്ള കമ്പ്യൂട്ടർ ഭാഷകൾക്കൊപ്പം സോഫ്റ്റ്‌വെയറുകളുടെ വിശദമായ തരംതിരിവും നൽകിയിരിക്കുന്നു. ഓപ്പൺ സോഴ്സ്, ഫ്രീവെയർ, ഷെയർ വെയർ, ഉടമസ്ഥാവകാശമുള്ള സോഫ്റ്റ്‌വെയർ എന്നിവയുടെ ആശയങ്ങളെക്കുറിച്ചും നമ്മൾ ചർച്ച ചെയ്യുന്നു.

## 3.1. ഡാറ്റ പ്രോസസ്സിങ്ങ് (Data Processing)

വിവരങ്ങൾ ലഭിക്കുന്നതിനായി ഡാറ്റയിൽ നടത്തുന്ന ഓപ്പറേഷൻ അഥവാ പ്രവർത്തനങ്ങളെയാണ് ഡാറ്റ പ്രോസസ്സിങ്ങ് എന്ന് സൂചിപ്പിക്കുന്നത്. സംഖ്യ, പദം, അളവ്, തുക തുടങ്ങിയ അസംസ്കൃതവസ്തുതകളും അക്കങ്ങളും ഡാറ്റ പ്രതിനിധീകരിക്കുന്നു. ഇത് പ്രോസസ്സ് ചെയ്യാനോ കൈകാര്യം ചെയ്യാനോ സാധിക്കും. അർത്ഥവത്തായതോ പ്രോസസ്സ് ചെയ്തതോ ആയ ഡാറ്റയുടെ രൂപമാണ് ഇൻഫർമേഷൻ (വിവരം). ഇത് നമ്മുടെ അറിവ് വർദ്ധിപ്പിക്കുകയും തീരുമാനങ്ങൾ എടുക്കാൻ സഹായിക്കുകയും ചെയ്യുന്നു. ചിത്രം 3.1 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ ഡാറ്റ പ്രോസസ്സിങ്ങ് ആറു ഘട്ടങ്ങളിലൂടെ കടന്നുപോകുന്നു.

a) ഡാറ്റ ശേഖരണം (Data Capturing)
b) ഡാറ്റയെ ഇൻപുട്ട് ചെയ്യുന്നു. (Input of Data)
c) ഡാറ്റ സംഭരണം/സമാഹരിക്കൽ
d) പ്രോസസ്സിങ്ങ്/ഡാറ്റ കൈകാര്യം ചെയ്യുന്നു.
e) വിവരം ഔട്ട്പുട്ട് ചെയ്യുന്നു.
f) വിവരം വിതരണം ചെയ്യുന്നു.



*ചിത്രം 3.1 ഡാറ്റ പ്രോസസ്സിങ്ങിന്റെ ഘട്ടങ്ങൾ*

നമുക്ക് ഈ ഘട്ടങ്ങളെ വിശദമായി പരിശോധിക്കാം.

**ഡാറ്റ ശേഖരണം (Data Capturing):** ഡാറ്റ പ്രോസസ്സിങ്ങിന്റെ ആദ്യഘട്ടമാണിത്. ഇവിടെ ഉറവിട രേഖകൾ (source document) എന്നറിയപ്പെടുന്ന ഒരു ഫോറം, ഡാറ്റ കൃത്യമായി ശേഖരിച്ച് ക്രമീകരിക്കാൻ ഉപയോഗിക്കുന്നു.

**ഇൻപുട്ട്:** ഈ ഘട്ടത്തിൽ ഉറവിട രേഖകൾ വഴി ശേഖരിച്ച ഡാറ്റ പ്രോസസ്സ് ചെയ്യുന്നതിനായി കമ്പ്യൂട്ടറിന് നൽകുന്നു. എന്നാൽ ഇക്കാലത്ത് പല സന്ദർഭങ്ങളിലും ഉറവിട രേഖകൾ ഉപയോഗിക്കാതെ തന്നെ കമ്പ്യൂട്ടറിൽ നേരിട്ട് ഡാറ്റ നൽകുന്നു.

**ഡാറ്റ സംഭരണം:** ഈ ഘട്ടത്തിൽ ഡാറ്റ പ്രോസസ്സിങ്ങിന് ആവശ്യമായ ഇൻപുട്ട് ചെയ്ത ഡാറ്റ സംഭരിക്കപ്പെടുന്നു. പ്രോസസ്സിങ്ങിന് ശേഷം ലഭിക്കുന്ന വിവരങ്ങളും സംഭരിക്കപ്പെടുന്നു.

**പ്രോസസ്സ്:** കമ്പ്യൂട്ടറിൽ സംഭരിച്ചിരിക്കുന്ന ഡാറ്റ, പ്രോസസ്സിങ്ങിനുവേണ്ടി തിരിച്ചെടുക്കുന്നു. പ്രോസസ്സിങ്ങിന്റെ ഭാഗമായി കണക്കുകൂട്ടൽ, തരംതിരിക്കൽ, താരതമ്യം ചെയ്യൽ, ക്രമീകരണം, സംഗ്രഹിക്കൽ, അരിച്ചെടുക്കൽ തുടങ്ങിയ പ്രവർത്തനങ്ങൾ നടത്തപ്പെടുന്നു.

**ഔട്ട്പുട്ട്:** ഈ ഘട്ടത്തിൽ പ്രോസസ്സ് ചെയ്ത് എടുത്ത ഡാറ്റ വിവരത്തിന്റെ രൂപത്തിലാണ് ലഭിക്കുന്നത്. ഔട്ട്പുട്ടിനെ ഭാവി ആവശ്യങ്ങൾക്കു വേണ്ടി സംഭരിക്കാം. ഇത് മറ്റ് സന്ദർഭങ്ങളിൽ വിവര നിർമ്മാണത്തിന് ഉപയോഗിക്കാം.

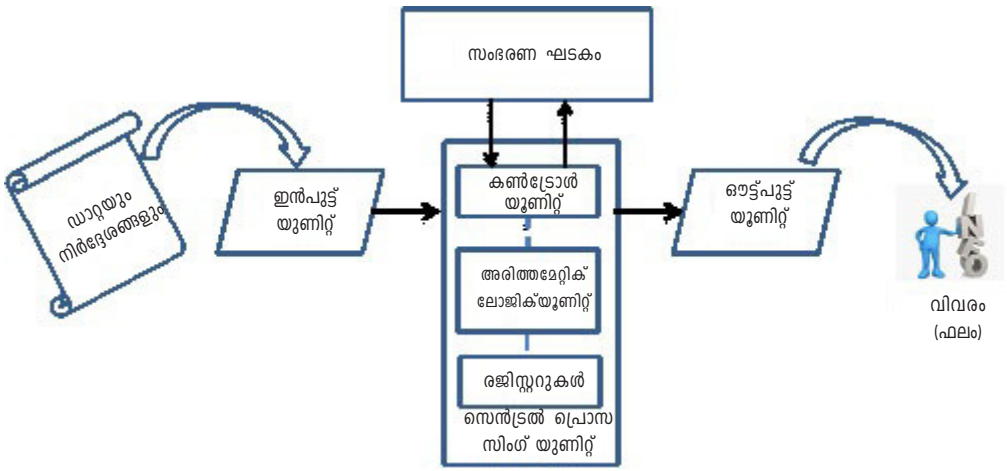
**വിവരങ്ങളുടെ വിതരണം:** ഔട്ട്പുട്ട് ഘട്ടത്തിൽ നിന്നും ലഭ്യമാകുന്ന വിവരങ്ങൾ ഗുണഭോക്താക്കൾക്ക് വിതരണം ചെയ്യപ്പെടുന്നു. അവർ ഈ വിവരങ്ങൾ അടിസ്ഥാനമാക്കി തീരുമാന

ങ്ങൾ എടുക്കുകയും പ്രശ്നപരിഹാരം കണ്ടെത്തുകയും ചെയ്യുന്നു. ഡാറ്റ പ്രോസസ്സിങ്ങിൽ അടങ്ങിയിരിക്കുന്ന പ്രവർത്തനങ്ങൾ നമ്മൾ കണ്ടല്ലോ. ഇങ്ങിനെയുള്ള പ്രവൃത്തികൾ നടത്താൻ വേണ്ടിയാണ് കമ്പ്യൂട്ടറുകൾ രൂപകൽപ്പന ചെയ്തിട്ടുള്ളത്. കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തന ഭാഗങ്ങൾ എങ്ങനെയാണ് ക്രമീകരിച്ചിരിക്കുന്നത് എന്ന് നമുക്ക് നോക്കാം.

## 3.2. കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തന ഘടകങ്ങൾ (Functional units of a Computer)

ജോൺവോൺ ന്യൂമാൻ (1903 1957)

വർഷങ്ങളായി കമ്പ്യൂട്ടറുകൾ, അതിന്റെ വലുപ്പം, രൂപം, വില, പ്രകടനം എന്നിവയിൽ വ്യത്യസ്തത പുലർത്തുന്നുണ്ടെങ്കിലും അതിന്റെ ഘടന അടിസ്ഥാനപരമായി ഒന്നുതന്നെയാണ്. അധ്യായം 1 ൽ ചർച്ച ചെയ്തതു പോലെ ഗണിതശാസ്ത്രജ്ഞനും കമ്പ്യൂട്ടർ ശാസ്ത്രജ്ഞനുമായ ജോൺ വോൺ ന്യൂമാൻ ആണ് ഈ അടിസ്ഥാനഘടന നിർദ്ദേശിച്ചിട്ടുള്ളത്. ഇതിൽ ഇൻപുട്ട് യൂണിറ്റ്, സെൻട്രൽ പ്രോസസ്സിംങ് യൂണിറ്റ് (CPU), സംഭരണ യൂണിറ്റ്, ഔട്ട്പുട്ട് യൂണിറ്റ് എന്നീ പ്രവർത്തന ഭാഗങ്ങൾ അടങ്ങിയിരിക്കുന്നു. (ചിത്രം 3.2. ശ്രദ്ധിക്കുക) ഇതിലെ ഓരോ യൂണിറ്റുകളും ഒരു പ്രത്യേക പ്രവൃത്തി നിർവ്വഹിക്കാൻ നിയോഗിക്കപ്പെട്ടിരിക്കുന്നു. ഈ ഘടകങ്ങളുടെ പ്രവർത്തനങ്ങൾ നമുക്ക് ചർച്ച ചെയ്യാം.




ചിത്രം 3.2 കമ്പ്യൂട്ടറിന്റെ അടിസ്ഥാന രൂപഘടന

### a) ഇൻപുട്ട് യൂണിറ്റ് (Input unit)

ശേഖരിച്ച ഡാറ്റയും അവ പ്രോസസ്സ് ചെയ്യാനുള്ള നിർദ്ദേശങ്ങളും കമ്പ്യൂട്ടറിലേക്ക് നൽകുന്നത് ഇൻപുട്ട് യൂണിറ്റ് വഴിയാണ്. അവ മെമ്മറിയിൽ സംഭരിക്കപ്പെടുന്നു. അക്കം, അക്ഷരം, ചിത്രം, ശബ്ദം, വീഡിയോ തുടങ്ങിയ പല രൂപങ്ങളിലും ഡാറ്റ കാണപ്പെടുന്നു. ഇങ്ങിനെയുള്ള ഡാറ്റകൾ കമ്പ്യൂട്ടറിലേക്ക് കൊടുക്കുവാൻ പലതരത്തിലുള്ള ഉപകരണങ്ങൾ ഉപയോഗിക്കുന്നു. കീബോർഡ്, മൗസ്, സ്കാനർ, മൈക്ക്, ഡിജിറ്റൽ ക്യാമറ തുടങ്ങിയവയാണ് സാധാരണയായി ഉപയോഗിക്കുന്ന ഇൻപുട്ട് ഉപകരണങ്ങൾ. ചുരുക്കത്തിൽ ഇൻപുട്ട് യൂണിറ്റ് നടത്തുന്ന പ്രവർത്തനങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നു.

1. നിർദ്ദേശങ്ങളും ഡാറ്റയും പുറമേനിന്ന് സ്വീകരിക്കുക.
2. ഈ നിർദ്ദേശങ്ങളും ഡാറ്റയും കമ്പ്യൂട്ടറിന് സ്വീകാര്യമായ രൂപത്തിൽ പരിവർത്തനം ചെയ്യുന്നു.
3. പരിവർത്തനം ചെയ്ത നിർദ്ദേശങ്ങളും ഡാറ്റയും കമ്പ്യൂട്ടറിലേക്ക് പ്രോസസ്സ് ചെയ്യാനായി നൽകുന്നു.

## b) സെൻട്രൽ പ്രോസസ്സിങ്ങ് യൂണിറ്റ് (CPU)

CPU കമ്പ്യൂട്ടറിന്റെ മസ്തിഷ്കമാണ്. മനുഷ്യ ശരീരത്തിൽ പ്രധാന തീരുമാനങ്ങളെല്ലാം എടുക്കുന്നത് മസ്തിഷ്കമാണ്. മസ്തിഷ്കത്തിന്റെ നിർദ്ദേശപ്രകാരമാണ് മറ്റു ശരീരഭാഗങ്ങളുടെ പ്രവർത്തനങ്ങൾ നടക്കുന്നത്. അതുപോലെ, ഒരു കമ്പ്യൂട്ടർ സംവിധാനത്തിൽ എല്ലാ പ്രധാന കണക്കുകൂട്ടലുകളും താരതമ്യങ്ങളും സി.പി.യു. വിനുള്ളിൽ ചെയ്യപ്പെടുന്നു. കമ്പ്യൂട്ടറിന്റെ മറ്റ് ഘടങ്ങളുടെ പ്രവർത്തനങ്ങൾ സജീവമാക്കുന്നതിനും നിയന്ത്രിക്കുന്നതിനും സി.പി.യു ഉത്തരവാദിത്തപ്പെട്ടിരിക്കുന്നു. CPU വിന്റെ പ്രവർത്തനങ്ങൾ നിർവ്വഹിക്കുന്നത് പ്രധാനമായും മൂന്ന് ഘടകങ്ങളാണ് – അരിത്തമെറ്റിക് ലോജിക് യൂണിറ്റ് (ALU), കൺട്രോൾ യൂണിറ്റ് (Control Unit - CU), രജിസ്റ്ററുകൾ (Registers)

### 1. അരിത്തമെറ്റിക് ലോജിക് യൂണിറ്റ് (ALU)

നിർദ്ദേശങ്ങൾക്കനുസരിച്ച് പ്രവർത്തനങ്ങൾ നടക്കുന്നത് ALU വിലാണ്. കണക്കുകൂട്ടലുകളും, ലോജിക്കൽ പ്രവർത്തനങ്ങളായ താരതമ്യം ചെയ്യലും തീരുമാനങ്ങൾ എടുക്കലും നിർവ്വഹിക്കപ്പെടുന്നത് ഈ യൂണിറ്റിലാണ്. സംഭരണ യൂണിറ്റിൽ സംഭരിക്കപ്പെട്ടിരിക്കുന്ന ഡാറ്റയും നിർദ്ദേശങ്ങളും ALU വിലേക്ക് മാറ്റപ്പെടുകയും അവിടെ പ്രോസസ്സിങ്ങ് നടക്കുകയും ചെയ്യുന്നു. ALU വിൽ ഡാറ്റ പ്രൊസസ്സിങ്ങിനിടയിൽ നിർമ്മിക്കപ്പെടുന്ന ഫലങ്ങളെ സംഭരണ ഘടകത്തിലേക്ക് താൽക്കാലികമായി മാറ്റുന്നു. പിന്നീട് കൂടുതൽ പ്രോസസ്സുകൾക്ക് ആവശ്യമുള്ളപ്പോൾ അവ തിരിച്ചെടുക്കുന്നു. അങ്ങനെ സംഭരണയൂണിറ്റിനും ALU വിനും ഇടയിൽ ഡാറ്റ പ്രോസസ്സിങ്ങ് പൂർത്തിയാക്കുന്നതിനു മുമ്പേ പലതവണ ഡാറ്റയുടെ ഒഴുക്ക് ഉണ്ടാകുന്നു.

### 2. കൺട്രോൾ യൂണിറ്റ് (CU)

ഓരോ ഫങ്ഷണൽ യൂണിറ്റിനും സ്വന്തമായി പ്രവൃത്തികൾ നിർവ്വചിക്കപ്പെട്ടിട്ടുണ്ടെങ്കിലും അവ ആവശ്യാനുസരണം മാത്രമെ നിർവ്വഹിക്കപ്പെടുന്നുള്ളു. ഈ പ്രവൃത്തികൾ ചെയ്യുന്നത് കൺട്രോൾ യൂണിറ്റാണ്. കൺട്രോൾ യൂണിറ്റിന്റെ നിർദേശം കിട്ടിയാൽ മറ്റ് യൂണിറ്റുകൾ അതിന്റെ പ്രവൃത്തി ഏറ്റെടുക്കുന്നു. കമ്പ്യൂട്ടറിന്റെ മറ്റെല്ലാ യൂണിറ്റുകളും കൈകാര്യം ചെയ്യുകയും ഏകോപിപ്പിക്കുകയും ചെയ്യുന്ന പ്രധാന നാഡീവ്യൂഹ വ്യവസ്ഥയാണ് കൺട്രോൾ യൂണിറ്റ്. മെമ്മറിയിൽ സംഭരിച്ചിരിക്കുന്ന പ്രോഗ്രാമുകൾ സ്വീകരിക്കുകയും, അവ പ്രവർത്തിപ്പിക്കുന്നതിനായി സിസ്റ്റത്തിൽ ബന്ധപ്പെട്ട ഘടകങ്ങൾക്ക് നിർദേശം നൽകുകയും അവയുടെ പ്രവർത്തനങ്ങളെ നടപ്പിൽ വരുത്തുകയുമാണ് കൺട്രോൾ യൂണിറ്റിന്റെ ധർമ്മം.

### 3. രജിസ്റ്ററുകൾ (Registers)

CPU വിന്റെ പ്രവർത്തനങ്ങൾ സുഗമമാക്കുന്ന താൽക്കാലിക സംഭരണ ഘടകങ്ങളാണ് ഇവ. ഡാറ്റ, നിർദേശം, മെമ്മറി, അഡ്രസ്, ഫലങ്ങൾ തുടങ്ങിയവ സംഭരിക്കുന്നതിനായി വ്യത്യസ്ഥ തരത്തിലുള്ള രജിസ്റ്ററുകൾ രൂപകൽപ്പന ചെയ്തിട്ടുണ്ട്.

### c. സംഭരണ ഘടകം (Storage Unit)

ഇൻപുട്ട് യൂണിറ്റ് വഴി നൽകിയ ഡാറ്റയും നിർദ്ദേശങ്ങളും യഥാർത്ഥ പ്രോസസ്സിങ്ങ് ആരംഭിക്കുന്നതിന് മുമ്പ് കമ്പ്യൂട്ടറിൽ സൂക്ഷിക്കുന്നു. അതുപോലെ പ്രോസസ്സിങ്ങിനുശേഷമുള്ള വിവരങ്ങൾ അല്ലെങ്കിൽ ഫലങ്ങൾ ഔട്ട്പുട്ട് യൂണിറ്റിലേക്ക് അയക്കുന്നതിനു മുമ്പ് കമ്പ്യൂട്ടറിന്റെ അകത്ത് സൂക്ഷിക്കുന്നു. കൂടാതെ, ഡാറ്റ പ്രോസസ്സിങ്ങിനിടയിൽ ഉള്ള ഫലങ്ങളും പിന്നീട് പ്രോസസ്സ് ചെയ്യുന്നതിന് വേണ്ടി കമ്പ്യൂട്ടറിൽ സംഭരിക്കുന്നു. കമ്പ്യൂട്ടറിന്റെ സംഭരണ യൂണിറ്റ് ഈ ഉദ്ദേശ്യങ്ങളെല്ലാം നിറവേറ്റുന്നു. ചുരുക്കത്തിൽ താഴെകൊടുത്ത വിവരങ്ങൾ സംഭരിക്കുക അല്ലെങ്കിൽ സൂക്ഷിക്കുക എന്നതാണ് ഈ യൂണിറ്റിന്റെ ധർമ്മം.

1. പ്രോസസ്സിങ്ങിന് ആവശ്യമായി വരുന്ന ഡാറ്റയും നിർദ്ദേശങ്ങളും.
2. നടന്നുകൊണ്ടിരിക്കുന്ന പ്രവർത്തനങ്ങൾക്കുവേണ്ട ഇടക്കാലഫലങ്ങൾ (Intermediate results)
3. ഔട്ട്പുട്ട് യൂണിറ്റിലേക്ക് വിടുന്നതിനുമുമ്പുള്ള പ്രോസസ്സിങ്ങിന്റെ അന്തിമഫലങ്ങൾ.

സംഭരണയൂണിറ്റ് പ്രധാനമായും രണ്ട് തരത്തിലാണുള്ളത്. അവ വിശദമായി താഴെ കൊടുക്കുന്നു.

**i. പ്രാഥമിക സംഭരണം (Primary Storage):** ഇത് പ്രധാന മെമ്മറി (main memory) എന്നറിയപ്പെടുന്നു. ഇതിനെ വീണ്ടും രണ്ടായി തിരിച്ചിരിക്കുന്നു - റാൻഡം ആക്സസ് മെമ്മറി (RAM), റീഡ് ഓൺലി മെമ്മറി (ROM). RAM ഒരു അസ്ഥിരമെമ്മറിയാണ് (volatile). നിർദ്ദേശങ്ങൾ, ഡാറ്റ പ്രോസസ്സിങ്ങിന്റെ ഇടക്കാലഫലങ്ങൾ എന്നിവ സൂക്ഷിക്കുന്നതിനാണ് ഇത് ഉപയോഗിക്കുന്നത്. കമ്പ്യൂട്ടർ തൊട്ടുമുമ്പ് ചെയ്ത പ്രവർത്തനങ്ങളുടെ ഫലങ്ങളും RAM - ൽ ഉണ്ടായിരിക്കും. കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തനങ്ങൾ ആരംഭിക്കാനുള്ള നിർദ്ദേശങ്ങൾ റോം (ROM) ൽ അടങ്ങിയിരിക്കുന്നു. പ്രാഥമിക സംഭരണി ഉയർന്ന വേഗതയുള്ളതാണെങ്കിലും താരതമ്യേന കുറഞ്ഞ സംഭരണശേഷിയുള്ളതുമാകുന്നു.

**ii. ദ്വിതീയ സംഭരണം (Secondary Storage):** ഇതിനെ സഹായക (auxilary) സംഭരണ ഘടകം എന്നും അറിയപ്പെടുന്നു. പ്രാഥമിക സംഭരണത്തിന്റെ പോരായ്മകളെ ഇത് പരിഹരിക്കുന്നു. സംഭരണ സ്ഥിരതയും (permanent) ഉയർന്ന സംഭരണശേഷിയും ഇതിന്റെ പ്രത്യേകതകളാണ്. ഡാറ്റയും പ്രോഗ്രാമുകളും വിവരങ്ങളും സ്ഥിരമായി ദ്വിതീയ മെമ്മറിയിൽ സൂക്ഷിക്കുന്നു. പക്ഷെ അതിനുള്ള നിർദ്ദേശങ്ങൾ പുറമെ നിന്നും നൽകിയിരിക്കണം. ഡി.വി.ഡി.കൾ, സി.ഡി കൾ, ഹാർഡ് ഡിസ്ക്കുകൾ, മെമ്മറിസ്റ്റിക്കുകൾ തുടങ്ങിയവ ഇതിന് ചില ഉദാഹരണങ്ങളാണ്.

### d. ഔട്ട്പുട്ട് യൂണിറ്റ് (output unit)

ഡാറ്റാ പ്രോസസ്സിങ്ങിനുശേഷം ലഭ്യമാകുന്ന വിവരങ്ങൾ മനുഷ്യന് വായിക്കാവുന്ന രീതിയിൽ പുറംലോകത്തേക്ക് നൽകാനാണ് ഔട്ട്പുട്ട് യൂണിറ്റ് ഉപയോഗിക്കുന്നത്. മോണിറ്ററും പ്രിന്ററുമാണ് സാധാരണയായി ഉപയോഗിച്ചു വരുന്ന ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ. ഔട്ട്പുട്ട് യൂണിറ്റിന്റെ പ്രധാന ധർമ്മങ്ങൾ ചുവടെ കൊടുക്കുന്നു.

1. CPU വിൽ നിന്ന് പ്രോഗ്രാം രൂപത്തിലുള്ള ഫലങ്ങൾ സ്വീകരിക്കുന്നു.
2. ഈ ഫലങ്ങൾ മനുഷ്യന് വായിച്ചെടുക്കാവുന്ന രീതിയിലേക്ക് മാറ്റുന്നു.
3. ഈ ഫലങ്ങൾ പുറം ലോകത്തേക്ക് നൽകുന്നു.

## 3.3. ഹാർഡ്‌വെയർ (Hardware)

ഹാർഡ്‌വെയറും സോഫ്ട്‌വെയറും അടങ്ങുന്നതാണ് കമ്പ്യൂട്ടർ സംവിധാനം എന്ന് നമുക്കറിയാം. തൊട്ടറിയാൻ കഴിയുന്നതും കാണാൻ സാധിക്കുന്നതുമായ കമ്പ്യൂട്ടറിന്റെ ഭാഗങ്ങളാണ് ഹാർഡ് വെയർ എന്ന പദം കൊണ്ടുദ്ദേശിക്കുന്നത്. മാത്രമല്ല ഇലക്ട്രോ മെക്കാനിക്കൽ ഘടകങ്ങളും ഇതിൽ ഉൾപ്പെടുന്നു. കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തന ഭാഗങ്ങൾ ഹാർഡ്‌വെയർ ഘടകങ്ങളുമായി ബന്ധപ്പെട്ടിരിക്കുന്നു. നമുക്ക് ഈ പ്രവർത്തന ഘടകങ്ങളെ പരിചയപ്പെടാം.

### 3.3.1. പ്രോസസ്സറുകൾ (Processors)

കമ്പ്യൂട്ടറിൽ എല്ലാ കണക്കുകൂട്ടലുകൾ നടത്തുന്നതും യുക്തിപരമായ തീരുമാനങ്ങൾ എടു ക്കുന്നതും മറ്റു പ്രവർത്തനങ്ങൾ ഏകോപിപ്പിക്കുന്നതും CPU ആണെന്ന് നാം കഴിഞ്ഞ ഭാഗങ്ങ ളിൽ പഠിച്ചുവല്ലോ. CPU വിന്റെ പ്രവർത്തനം കമ്പ്യൂട്ടറിന്റെ മൊത്തത്തിലുള്ള പ്രകടനത്തെ നിർണ്ണയിക്കുന്നു. CPU എന്നത് ഒരു ഇന്റഗ്രേറ്റഡ് സർക്യൂട്ട് (IC) പാക്കേജ് ആണ്. ഇത് ദശ ലക്ഷ കണക്കിന് ട്രാൻസിസ്റ്ററുകളും അനുബന്ധഘടകങ്ങളും കൂടിചേർന്ന ഒരു സിലിക്കൺ ചിപ്പാണ്. ഇതിനെ മൈക്രോപ്രൊസസ്സർ എന്ന്വിളിക്കുന്നു. ചിത്രം 3.3 ൽ ചില കമ്പനികളുടെ പ്രോസസ്സറുകൾ കാണിച്ചിരിക്കുന്നു. കമ്പ്യൂട്ടറിലെ പ്രധാന ബോർഡായ മദർ ബോർഡിലെ വലിയ സോക്കറ്റുമായാണ് CPU സാധാരണയായി ബന്ധിപ്പിച്ചിരിക്കുന്നത്. CPU പ്രവർത്തിക്കു മ്പോൾ ധാരാളം താപം പുറംതള്ളുന്നതുകൊണ്ട് ഫാനും ഹീറ്റ് സിങ്കും ഉൾപ്പെട്ട കൂളിംഗ് സിസ്റ്റം ഇതിൽ ഘടിപ്പിച്ചിരിക്കുന്നു. സാധാരണയായി ഉപയോഗിക്കുന്ന പ്രോസസ്സറുകളാണ് ഇന്റൽ കോർ i3, കോർ i5, കോർ i7, AMD Quadcore തുടങ്ങിയവ.

*ചിത്രം.3.3: വിവിധതലം പ്രോസസ്സറുകൾ*

CPU വിന് ഉള്ളിലെ സംഭരണ സ്ഥലങ്ങളാണ് രജിസ്റ്ററുകൾ, മറ്റ് മെമ്മറി ഭാഗങ്ങളേക്കാൾ വേഗത്തിൽ ഇതിന്റെ ഉള്ളടക്കത്തെ CPU വിന് ഉപയോഗിക്കാൻ കഴിയും. നിർദ്ദേശ ങ്ങളും ഡാറ്റയും താൽക്കാലികമായി സംഭ രിക്കാനുള്ള സ്ഥലമാണ് രജിസ്റ്ററുകൾ. ഇവ മെമ്മറിയുടെ ഭാഗമല്ല. എന്നാൽ കമ്പ്യൂട്ടറിന് വേഗത പ്രധാനം ചെയ്യുന്ന പ്രത്യേക സംഭരണ സ്ഥലങ്ങളാണ്. രജിസ്റ്ററുകൾ പ്രവർത്തിക്കുന്നത് കൺട്രോൾ യൂണിറ്റിന്റെ നിർദ്ദേശമനുസരിച്ചാ ണ്. ഡാറ്റയും നിർദേശങ്ങളും ശേഖരിച്ച് വളരെ വേഗത്തിൽ അരിത്തമെറ്റിക് അഥവാ ലോജി ക്കൽ പ്രവർത്തനങ്ങൾ നിർവ്വഹിക്കാൻ ഇവ സഹായിക്കുന്നു. പ്രോഗ്രാമിന്റെ കൃത്യനിർവ്വഹണം ഇത് വേഗത്തിലാക്കുന്നു. CPU വിനുള്ളിലെ പ്രധാനപ്പെട്ട രജിസ്റ്ററുകൾ താഴെ കൊടുക്കുന്നു.

എല്ലാ കമ്പ്യൂട്ടറുകളിലും ഒരു ക്ലോക്ക് ഉണ്ട്. അത് നിർദ്ദേശങ്ങൾ നിർവ്വഹിക്കുന്ന നിരക്ക് ക്രമീകരിക്കുന്നു. ഓരോ നിർദ്ദേശവും നിർവ്വഹിക്കാൻ CPU വിന് ഒരു നിശ്ചിത എണ്ണം ക്ലോക്ക് ടിക്കുകൾ (ക്ലോക്ക് ആവൃത്തി) ആവശ്യമാണ്. ക്ലോ ക്കിന്റെ വേഗത കൂടുമ്പോൾ, സി.പി.യു.വിന് ഒരു സെക്കന്റിൽ കൂടുതൽ നിർദേ ശങ്ങൾ നിർവ്വഹിക്കുവാൻ സാധിക്കും. മറ്റൊരു ഘടകം ചിപ്പിന്റെ രൂപഘടനയാണ്. ഒരു സമയം പ്രോസസ്സറിന് പ്രോസസ്സ് ചെയ്യുവാൻ കഴിയുന്ന ബിറ്റുകളുടെ സംഖ്യയെ word size എന്ന് വിളിക്കുന്നു. വിവിധ word size ഉള്ള പ്രോസസ്സറുകൾ ഉപയോഗിക്കപ്പെടുന്നു. (ഉദാഹരണം 8-bit, 16-bit, 32-bit, 64-bit തുടങ്ങിയവ)

i. **അക്യുമുലേറ്റർ (Accumulator)**: അരിത്തമെറ്റിക് ആന്റ് ലോജിക് യൂണിറ്റിന്റെ (ALU) ഒരു ഭാഗമാണ് അക്യുമുലേറ്റർ. അരിത്തമെറ്റിക് ലോജിക്കൽ പ്രവർത്തനങ്ങൾ നിർവ്വഹിക്കുമ്പോൾ അതിന്റെ ഫലങ്ങൾ സൂക്ഷിക്കാൻ ഈ രജിസ്റ്റർ ഉപയോഗിക്കുന്നു. ഇതിനെ രജിസ്റ്റർ A എന്നും വിളിക്കാറുണ്ട്.

ii. **മെമ്മറി അഡ്രസ്സ് രജിസ്റ്റർ (MAR):** ഡാറ്റ സംഭരിക്കപ്പെടേണ്ടതോ അല്ലെങ്കിൽ എവിടെനിന്നാണോ വീണ്ടെടുക്കപ്പെടേണ്ടത് ആ മെമ്മറി ലൊക്കേഷന്റെ വിലാസം സൂക്ഷിക്കുന്ന രജിസ്റ്ററാണ് മെമ്മറി അഡ്രസ് രജിസ്റ്റർ.

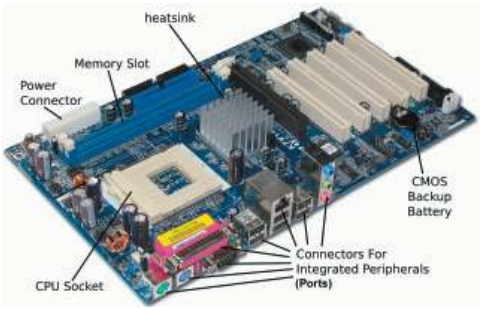
iii. **മെമ്മറി ബഫർ രജിസ്റ്റർ (MBR):** ഡാറ്റാ പ്രോസസ്സിങ്ങിനുവേണ്ടി പ്രോസസ്സർ എടുക്കുന്നതോ പ്രോസസിങ്ങിനുശേഷം കൊടുക്കേണ്ടതോ ആയ ഡാറ്റ താൽക്കാലികമായി സൂക്ഷിക്കുന്ന രജിസ്റ്ററാണ് മെമ്മറി ബഫർ രജിസ്റ്റർ.

iv. **ഇൻസ്ട്രക്ഷൻ രജിസ്റ്റർ (I R):** ഏത് നിർദ്ദേശമാണോ പ്രൊസസ്സർ നിർവ്വഹിക്കേണ്ടത്, ആ നിർദ്ദേശം സൂക്ഷിച്ചുവെക്കുന്ന രജിസ്റ്ററാണ് ഇൻസ്ട്രക്ഷൻ രജിസ്റ്റർ.

v. **പ്രോഗ്രാം കൗണ്ടർ (PC):** പ്രോസസ്സർ അടുത്തതായി നിർവ്വഹിക്കേണ്ട നിർദ്ദേശത്തിന്റെ മെമ്മറി വിലാസം സൂക്ഷിക്കുന്ന രജിസ്റ്ററാണ് പ്രോഗ്രാം കൗണ്ടർ

## 3.3.2 മദർബോർഡ് (Motherboard)

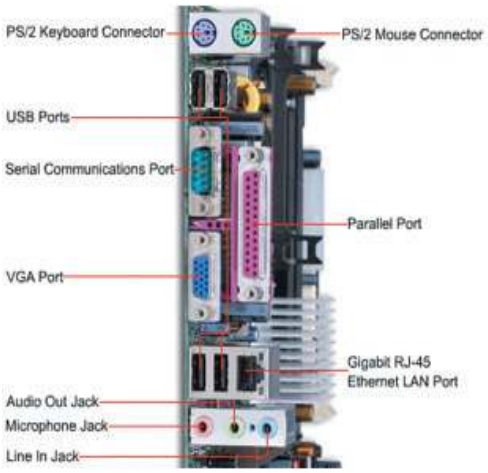
പ്രൊസസ്സറും അതിനോട് അനുബന്ധിച്ചുള്ള പ്രധാനപ്പെട്ട ഘടകങ്ങളും അടങ്ങിയിട്ടുള്ള വലിയ പ്രിന്റഡ് സർക്യൂട്ട് ബോർഡാണ് മദർബോർഡ് (PCB). മെമ്മറി, ഗ്രാഫിക്സ് കാർഡ്, സൗണ്ട് കാർഡ് തുടങ്ങിയുള്ള സർക്യൂട്ട് ബോർഡുകൾ ആവശ്യാനുസരണം വേണ്ടിവന്നാൽ ഉൾപ്പെടുത്താനുള്ള എക്സപാൻഷൻ സ്ലോട്ടുകൾ ഇതിലുണ്ട്. (ചിത്രം 3.4: നോക്കുക) മദർബോർഡ് നിർബ്ബന്ധമായും പ്രോസസ്സറിന് അനുയോജ്യമായിരിക്കണം.




*ചിത്രം 3.4 മദർബോർഡ്*

## 3.3.4 പെരിഫറലുകളും പോർട്ടുകളും (Peripherals and ports)

കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ കഴിവുകൾ വർദ്ധിപ്പിക്കാൻ അതിനോട് ഘടിപ്പിച്ചിരിക്കുന്ന അനുബന്ധ ഉപകരണങ്ങളാണ് പെരിഫറലുകൾ. പുറമേയുള്ള ഉപകരണങ്ങളെ മദർബോർഡിൽ ഘടിപ്പിക്കാനാണ് പോർട്ടുകൾ ഉപയോഗിക്കുന്നത്. ഇൻപുട്ട് ഉപകരണങ്ങൾ, ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ, ബാഹ്യ സംഭരണ ഉപകരണങ്ങൾ, ആശയവിനിമയ ഉപകരണങ്ങൾ എന്നിവ പെരിഫറലുകളിൽ ഉൾപ്പെടുന്നു. ഈ ഉപകരണങ്ങൾ മദർബോർഡുമായി ആശയവിനിമയം നടത്തുന്നത് VGA, PS/2, USB, Ethernet, HDMI തുടങ്ങിയ പോർട്ടുകൾ വഴിയാണ്. പേഴ്സണൽ കമ്പ്യൂട്ടറിൽ ഉപയോഗിക്കുന്ന ചില പോർട്ടുകൾ ചിത്രം 3.5 ൽ കാണിച്ചിരിക്കുന്നു.




*ചിത്രം 3.5 പോർട്ടുകൾ*

### 3.3.5. മെമ്മറി (Memory)

ഡാറ്റയോ നിർദ്ദേശങ്ങളോ ഫലങ്ങളോ താൽക്കാലികമായോ സ്ഥിരമായോ സൂക്ഷിച്ചു വെക്കാനുള്ള സ്ഥലമാണ് മെമ്മറി. മെമ്മറിയെ രണ്ടായി തരംതിരിച്ചിരിക്കുന്നു. പ്രാഥമിക മെമ്മറി, ദ്വിതീയ മെമ്മറി. മദർബോർഡിൽ സ്ഥിതി ചെയ്യുന്നതും പ്രോസസ്സറുമായി നേരിട്ട് ബന്ധപ്പെടുന്നതുമായ മെമ്മറിയാണ് പ്രാഥമിക മെമ്മറി. സ്ഥിരമായി വിവരങ്ങൾ സൂഷിക്കാൻ ഉപയോഗിക്കുന്നതും പ്രോസസ്സറുമായി പ്രാഥമിക മെമ്മറിയിലൂടെ മാത്രം വിവരങ്ങൾ കൈമാറുന്നതുമായ മെമ്മറിയാണ് ദ്വീതിയ മെമ്മറി. മെമ്മറിയെക്കുറിച്ച് കൂടുതൽ പഠിക്കുന്നതിന് മുമ്പായി മെമ്മറി അളക്കുന്ന യൂണിറ്റുകളെ മനസിലാക്കാം. താഴെപറയുന്നവയാണ് ഈ അളവിന്റെ ഏകകങ്ങൾ.

| | |
|---|---|
| ബൈനറി ഡിജിറ്റ് = 1 ബിറ്റ് | 1 MB (മെഗാ ബൈറ്റ്)= 1024 KB |
| 1 നിബ്ബ്ൾ = 4 ബിറ്റ് | 1 GB (ജിഗാബൈറ്റ്) = 1024 MB |
| 1 ബൈറ്റ് = 8 ബിറ്റ്സ് | 1 TB (ടെറാ ബൈറ്റ്) = 1024 GB |
| 1 KB(കിലോ ബൈറ്റ്) = 1024 ബൈറ്റ്സ് | 1 PB (പെറ്റാ ബൈറ്റ്) = 1024 TB |

### a. പ്രാഥമിക മെമ്മറി (Primary Storage)

പ്രാഥമിക മെമ്മറി എന്നത് സെമികണ്ടക്ടർ മെമ്മറിയാണ്. ഇതിനെ CPU നേരിട്ട് കൈകാര്യം ചെയ്യുന്നു. ഇതിന് ഡാറ്റ വളരെ വേഗത്തിൽ അയയ്ക്കുന്നതിനും സ്വീകരിക്കുന്നതിനുമുള്ള കഴിവുണ്ട്. 3 തരത്തിലുള്ള മെമ്മറിയാണ് ഇതിൽ ഉൾപ്പെട്ടിരിക്കുന്നത്. റാം, റോം, ക്യാഷ് മെമ്മറി എന്നിവയാണ് അവ.

#### i. റാൻഡം ആക്സസ് മെമ്മറി (RAM)

മൈക്രോ പ്രൊസസ്സറിന് ഡാറ്റ സംഭരിക്കാനും തിരിച്ചയയ്ക്കാനും സാധിക്കുന്ന RAM എന്ന പ്രാഥമിക മെമ്മറി ചിത്രം 3.6 ൽ കാണിച്ചിരിക്കുന്നു. RAM നുള്ളിൽ ഡാറ്റ എവിടെ നിന്ന് വേണമെങ്കിലും ശേഖരിക്കാനും തിരിച്ചെടുക്കാനും സാധിക്കും. CPU പ്രൊസസ്സ് ചെയ്യുന്ന ഡാറ്റയോ നിർദ്ദേശങ്ങളോ റാമിനുള്ളിൽ നിർബന്ധമായും ഉണ്ടായിരിക്കണം. വൈദ്യുത ബന്ധം നിലയ്ക്കുമ്പോൾ റാമിനുള്ളിൽ അടങ്ങിയിരിക്കുന്നതെല്ലാം നഷ്ടപ്പെടുന്നു. അതിനാൽ റാം ഒരു അസ്ഥിരമെമ്മറിയാണ് റാമിന്റെ സംഭരണശേഷി സാധാരണ ജിഗാബൈറ്റിലാണ് പറയാറുള്ളത്.

*ചിത്രം 3.6 (RAM)*

എത്രമാത്രം വേഗത്തിൽ ഡാറ്റ മെമ്മറിയിൽ സംഭരിക്കുന്നു/തിരിച്ചെടുക്കുന്നു എന്നതാണ് റാമിന്റെ വേഗത കൊണ്ടുദ്ദേശിക്കുന്നത്. ഇത് അളക്കുന്നത് മെഗാ ഹെർട്സിൽ ആണ് (MHz). ഒരു കമ്പ്യൂട്ടർ പ്രവർത്തിച്ചുകൊണ്ടിരിക്കുമ്പോൾ അതിലെ റാമിൽ താഴെ പറയുന്നവ ഉണ്ടായിരിക്കും.

1. ഓപ്പറേറ്റിംഗ് സിസ്റ്റം
2. നിലവിൽ ഉപയോഗിച്ചു കൊണ്ടിരിക്കുന്ന അപ്ലിക്കേഷൻ സോഫ്ട്വെയർ
3. പ്രൊസസ്സ് ചെയ്തുകൊണ്ടിരിക്കുന്ന ഡാറ്റ

#### ii. റീഡ് ഒൺലി മെമ്മറി (ROM)

ROM എന്നത് സ്ഥിരമായ മെമ്മറിയാണ്. അതിൽ നിന്ന് ഡാറ്റ വീണ്ടെടുക്കുവാൻ മാത്രമേ കഴിയൂ. ഇതിന്റെ ഉള്ളടക്കം എളുപ്പത്തിൽ മാറ്റാൻ സാധിക്കില്ല. വൈദ്യുതി ബന്ധം നിലച്ചാലും

ഇതിലെ ഉള്ളടക്കം മാഞ്ഞു പോകാതെ നിലനിൽക്കുന്നു. BIOS എന്ന പേരിൽ അറിയപ്പെടുന്ന ഒരു പ്രത്യേക ബൂട്ട് അപ്പ് (boot up) പ്രോഗ്രാം റോമിൽ അടങ്ങിയിരിക്കുന്നു. ROM ചിപ്പാണ് ചിത്രം 3.7 ൽ കാണിച്ചിരിക്കുന്നത്. കമ്പ്യൂട്ടർ ഓൺ ചെയ്യുമ്പോഴോ 'ബൂട്ട് അപ്പ്' ചെയ്യുമ്പോഴോ ഈ സോഫ്റ്റ്വെയർ പ്രവർത്തിക്കുന്നു. ഇത് കമ്പ്യൂട്ടർ ഹാർഡ്വെയറിനെ പരിശോധിക്കുകയും ഓപ്പറേറ്റിംങ്ങ് സിസ്റ്റത്തിനെ മെമ്മറിയിലേക്ക് കൊണ്ടുവരുകയും ചെയ്യുന്നു. ROM ന്റെ ചില പരിഷ്കരിച്ച രൂപങ്ങൾ താഴെകൊടുക്കുന്നു.

*ചിത്രം 3.7: ROM ചിപ്പ്*

1. പ്രോഗ്രാമബിൾ റീഡ് ഒൺലി മെമ്മറി PROM (ഒരിക്കൽ മാത്രം പ്രോഗ്രാം ചെയ്യപ്പെടുന്ന മെമ്മറിയാണിത്)
2. ഇറേസബിൾ പ്രോഗ്രാമബിൾ റീഡ് ഒൺലി മെമ്മറി (EPROM) (അൾട്രാവയലറ്റ് റേഡിയേഷൻ ഉപയോഗിച്ച് വീണ്ടും എഴുതപ്പെട്ടാൻ കഴിയുന്ന മെമ്മറിയാണിത്)
3. ഇലക്ട്രിക്കലി ഇറേസബിൾ പ്രോഗ്രാമബിൾ റീഡ് ഒൺലി മെമ്മറി (EEPROM) (വൈദ്യുതി ഉപയോഗിച്ച് വീണ്ടും എഴുതപ്പെടാൻ കഴിയുന്ന മെമ്മറിയാണിത്.)

പട്ടിക 3.1 ൽ റാമിന്റെയും റോമിന്റെയും താരതമ്യം കൊടുത്തിരിക്കുന്നു.

| റാൻഡം ആക്സസ് മെമ്മറി (RAM) | റീഡ് ഒൺലി മെമ്മറി (ROM) |
|---|---|
| • ഇത് റോമിനേക്കാൾ വേഗത കൂടിയതാണ്<br>• കമ്പ്യൂട്ടർ പ്രവർത്തിക്കുമ്പോൾ ഡാറ്റയും, അപ്ലിക്കേഷൻ പ്രോഗ്രാമും ഓപ്പറേറ്റിംങ്ങും സിസ്റ്റവും ഇത് സൂക്ഷിക്കുന്നു.<br>• ഡാറ്റയുടെ സംഭരണവും വീണ്ടെടുക്കലും ഇത് അനുവദിക്കുന്നു.<br>• കമ്പ്യൂട്ടർ ഓഫാക്കിയാൽ ഇതിലെ ഉള്ളടക്കം നഷ്ടപ്പെടുന്നതിനാൽ ഇതൊരു അസ്ഥിരമെമ്മറിയാണ്. | • ഇതിന് വേഗത കുറവാണ്.<br>• കമ്പ്യൂട്ടർ ബൂട്ട് ചെയ്യുവാനുള്ള പ്രോഗ്രാം ഇത് സൂക്ഷിക്കുന്നു.<br>• സാധാരണയായി ഇതിൽ നിന്ന് ഡാറ്റ തിരിച്ചെടുക്കാൻ മാത്രമേ സാധിക്കുകയുള്ളൂ.<br>• കമ്പ്യൂട്ടർ ഓഫ് ആക്കിയാലും ROM ലെ ഉള്ളടക്കം നഷ്ടപ്പെടാത്തതിനാൽ ഇതൊരു സ്ഥിര മെമ്മറിയാണ്. |

*പട്ടിക 3.1: RAM- ROM എന്നിവയുടെ താരതമ്യം*

### iii. ക്യാഷ് മെമ്മറി

പ്രൊസസ്സറിന്റെയും റാമിന്റെയും (അഥവാ മെയിൻ മെമ്മറിയുടെയും) ഇടയ്ക്കുള്ള ചെറുതും വേഗതയേറിയതുമായ മെമ്മറിയാണ് ക്യാഷ് മെമ്മറി. കൂടെകൂടെ ആവശ്യമായി വരുന്ന ഡാറ്റയും, നിർദ്ദേശങ്ങളും ഇടക്കാല ഫലങ്ങളും വേഗത്തിൽ എടുക്കുവാൻവേണ്ടി ക്യാഷ് മെമ്മറിയിൽ സൂക്ഷിക്കുന്നു. പ്രോസസ്സർ, റാമിലെ ഒരു ലൊക്കേഷൻ സംഭരിക്കുകയോ അതിൽ നിന്ന് തിരിച്ചെടുക്കുകയോ ചെയ്യുമ്പോൾ ആദ്യം ക്യാഷ് മെമ്മറിയിൽ ഡാറ്റയുടെ ഒരു കോപ്പി ഉണ്ടോ എന്ന് പരിശോധിക്കുന്നു. അങ്ങനെയാണെങ്കിൽ പ്രൊസസ്സർ ക്യാഷ് മെമ്മറിയിൽ നിന്നും ഇത് പെട്ടെന്ന് റീഡ് ചെയ്യുന്നു. റാമിനെക്കാൾ വേഗത്തിൽ ക്യാഷ് മെമ്മറിയിൽ നിന്നും ഡാറ്റ തിരിച്ചെടുക്കുന്നു. റാമിനേക്കാളും വിലയേറിയതാണ് ക്യാഷ് മെമ്മറി. CPU വിന്റെയും മദർബോർഡിന്റെയും ഉള്ളിലുള്ള ക്യാഷ് മെമ്മറി, സിസ്റ്റത്തിന്റെ പ്രകടനം മെച്ചപ്പെടുത്തുന്നതിന് സഹായിക്കുന്നു.

## b. ദ്വിതീയ സംഭരണി (Secondary / Auxilary storage)

ദ്വിതീയ മെമ്മറി സ്ഥിരമാണ്. റാമിൽ നിന്നും വ്യത്യസ്തമായി, കമ്പ്യൂട്ടർ ഓഫ് ചെയ്താലും ഈ ഉപകരണങ്ങളിൽ സംഭരിച്ചിട്ടുള്ള ഡാറ്റ ഒരിക്കലും നഷ്ടപ്പെടുന്നില്ല. ദ്വിതീയ മെമ്മറി റാമിനേക്കാൾ വളരെ വലുപ്പമുള്ളതാണ്. എന്നാൽ ഇതിന് വേഗത കുറവാണ്. പ്രോഗ്രാമും ഡാറ്റയും ഇതിൽ സൂക്ഷിക്കുന്നുണ്ടെങ്കിലും പ്രോസസ്സറിന് അവയെ നേരിട്ട് ഉപയോഗിക്കാൻ സാധിക്കില്ല. ഒരു കമ്പ്യൂട്ടറിൽ നിന്നും വേറൊരു കമ്പ്യൂട്ടറിലെക്ക് ഡാറ്റയോ, പ്രോഗ്രാമുകളോ കൈമാറ്റം ചെയ്യാൻ ദ്വിതീയ മെമ്മറി ഉപയോഗിക്കുന്നു. ഇത് ഒരു ബാക്ക് അപ്പ് ആയും ഉപയോഗിക്കുന്നു. സംഭരണ ഉപകരണങ്ങളുടെ പ്രധാന തരം തിരിവുകളാണ് മാഗ്നറ്റിക്, ഒപ്റ്റിക്കൽ സെമികണ്ടക്ടർ മെമ്മറി എന്നിവ.

### i. കാന്തിക സംഭരണ ഉപകരണങ്ങൾ (Magnetic Storage Devices)

കാന്തിക വസ്തുക്കൾ പുതിയ പ്ലാസ്റ്റിക് ടേപ്പോ മെറ്റൽ/പ്ലാസ്റ്റിക് ഡിസ്കോ ആണ് കാന്തിക സംഭരണ ഉപകരണങ്ങളായി ഉപയോഗിക്കുന്നത്. ഈ ഉപകരണങ്ങളിൽ ഡാറ്റ കാന്തികമായി റെക്കോർഡ് ചെയ്യപ്പെടുന്നു. ഡാറ്റ ഈ ഉപകരണങ്ങളിൽ നിന്നും ഉപയോഗിക്കുന്നത്, റീഡ്/റൈറ്റ് ഹെഡ് ഉപയോഗിച്ചാണ്. ചില പ്രശസ്തമായ കാന്തിക സംഭരണ ഉപകരണങ്ങളാണ് മാഗ്നറ്റിക് ടേപ്പുകൾ (Magnetic Tapes), ഹാർഡ് ഡിസ്കുകൾ (Hard Disks) തുടങ്ങിയവ.

### ii. ഒപ്റ്റിക്കൽ സ്റ്റോറേജ് ഉപകരണങ്ങൾ (Optical storage devices)

ശക്തി കുറഞ്ഞ ലേസർ രശ്മികൾ ഉപയോഗിച്ചാണ് ഒപ്റ്റിക്കൽ സ്റ്റോറേജ് ഉപകരണങ്ങളിൽ ഡാറ്റ സംഭരിക്കുകയോ അവിടെ നിന്ന് തിരിച്ചെടുക്കുകയോ ചെയ്യുന്നത്. വൃത്താകൃതിയിലുള്ള രണ്ട് പ്ലാസ്റ്റിക് ഡിസ്ക്കുകളുടെ ഇടയിൽ അലുമിനിയത്തിന്റെ ഒരു തകിട് ചേർത്തു വെച്ചാണ് ഇത് ഉണ്ടാക്കുന്നത്. തുടർച്ചയായ ഈ സ്പൈറലിൽ പിറ്റ്സ് ആന്റ് ലാന്റ്സ് രൂപത്തിലാണ് ഡാറ്റ എഴുതപ്പെട്ടിരിക്കുന്നത്. ലേസർ രശ്മികൾ ഈ പിറ്റ്സ് ആന്റ് ലാന്റ്സിനെ പൂജ്യമായും ഒന്നായും എടുക്കുന്നു. കൂടുതൽ അളവിലും കുറഞ്ഞ ചിലവിലും നിർമിക്കാൻ സാധിക്കുന്നതും പ്രശസ്തവുമായ ദ്വിതീയ സംഭരണ ഉപകരണമാണ് ഒപ്റ്റിക്കൽ ഡിസ്ക്. പ്രധാനപ്പെട്ട ഒപ്റ്റിക്കൽ സ്റ്റോറേജ് ഉപകരണങ്ങളാണ് CD, DVD, and Blue-Ray എന്നിവ.

### iii. അർദ്ധചാലക സംഭരണി (Flash Memory)

ഫ്ളാഷ് ഡ്രൈവുകൾ ഡാറ്റാ സംഭരണത്തിനായി EEPROM ചിപ്പുകൾ ഉപയോഗിക്കുന്നു. ചലനാത്മകമായ ഭാഗങ്ങളൊന്നും തന്നെ ഇതിൽ ഇല്ലാത്തതിനാൽ ആഘാതത്തെ പ്രതിരോധിക്കുന്നതാണ് ഇത്. മറ്റ് ദ്വിതീയ മെമ്മറികളുമായി താരതമ്യം ചെയ്യുമ്പോൾ ഫ്ളാഷ് മെമ്മറി വേഗതയേറിയതും കേടുകൂടാതെ കുറേക്കാലം നിൽക്കുന്നതുമാണ്. ഒരു നിശ്ചിത പ്രാവശ്യം മാത്രമേ എഴുതുവാൻ സാധിക്കുകയുള്ളൂ എന്നത് ഇതിന്റെ ഒരു പരിമിതിയാണ്. USB ഫ്ളാഷ് മെമ്മറി, ഫ്ളാഷ് മെമ്മറി കാർഡ് എന്നിവ വിവിധ തരത്തിലുള്ള ഫ്ളാഷ് മെമ്മറികളാണ്. ചിത്രം 3.8ൽ പലതരത്തിലുള്ള ഫ്ളാഷ് മെമ്മറികൾ കാണിച്ചിരിക്കുന്നു.


*ചിത്രം 3.8: ഫ്ളാഷ്ഡ്രൈവും മെമ്മറികാർഡുകളും*

എങ്ങിനെയാണ് രജിസ്റ്ററുകളും പ്രാഥമിക മെമ്മറിയും ദ്വിതീയ മെമ്മറിയും ഒരുമിച്ച് ഒരു കമ്പ്യൂട്ടറിൽ പ്രവർത്തിക്കുന്നത് എന്ന് മനസിലാക്കുന്നതിന് താഴെയുള്ള ഉദാഹരണം ശ്രദ്ധിക്കുക.

അടുക്കളയിൽ സാലഡ് ഉണ്ടാക്കുന്നതിന് നമുക്ക് ആവശ്യമുള്ളവ.

- സാലഡ് ഉണ്ടാക്കുന്നതിന് ആവശ്യമായ പച്ചക്കറികൾ സൂക്ഷിക്കുന്നതിനുള്ള റഫ്രിജറേറ്റർ
- പച്ചക്കറികൾ നുറുക്കുവാൻ ആവശ്യമായ പലക വെക്കുവാനുള്ള മേശ
- പച്ചക്കറികൾ വെച്ച് അരിയുവാൻ മേശമേൽ വെക്കുന്ന പലക.
- ഏതൊക്കെ പച്ചക്കറികൾ അരിയണം എന്നതിനുള്ള പാചകക്കുറിപ്പ്.
- ഭാഗികമായി അരിഞ്ഞ പച്ചക്കറികൾ വീണ്ടും ചെറുതാക്കുന്നതിന് വേണ്ടിയോ ഭാഗികമായി അരിഞ്ഞ് മറ്റു പച്ചക്കറികളുമായി കൂട്ടികലർത്തുന്നതിന് വേണ്ടിയോ പലകയുടെ മൂലകൾ ശൂന്യമായി സൂക്ഷിക്കണം.
- സാലഡ് നന്നായി ഇളക്കുവാൻ ആവശ്യമായ പാത്രം.
- സാലഡ് ഉണ്ടാക്കി കഴിഞ്ഞാൽ അത് സൂക്ഷിച്ചുവെക്കുവാൻ ആവശ്യമായ ഫ്രിഡ്ജിലെ സ്ഥലം.

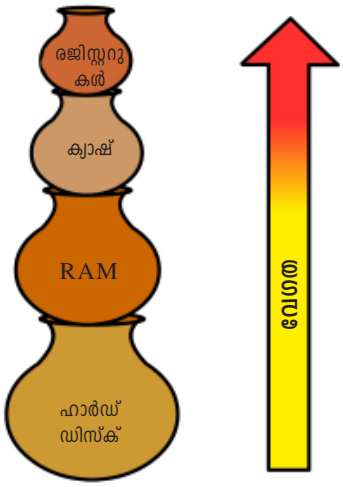
സാലഡ് ഉണ്ടാക്കുന്നവിധം: റഫ്രിജറേറ്റിൽ നിന്ന് പച്ചക്കറികൾ മേശമേൽ എത്തിക്കുന്നു. പാചകക്കുറിപ്പിന് അനുസരിച്ച് ചില പച്ചക്കറികൾ എടുത്ത് അരിയുന്ന ബോക്സിൽവെക്കുക, മുറിക്കുന്ന ബോർഡിന്റെ അരികിലേക്ക് ചില പച്ചക്കറികൾ താൽക്കാലികമായി മാറ്റിവയ്ക്കുന്നു. അവ ചെറിയ കഷണങ്ങളാക്കുന്നു. പകുതി മുറിച്ച കഷ്ണങ്ങൾ താൽക്കാലികമായി പലകയുടെ മൂലയിലേക്ക് മാറ്റിവയ്ക്കാവുന്നതാണ്. മുറിച്ചെടുത്ത പച്ചക്കറികൾ പാത്രത്തിൽവെക്കുക. തീൻ മേശയിൽ അത് അപ്പോൾ വിതരണം ചെയ്യുന്നില്ലെങ്കിൽ റഫ്രിജറേറ്ററിലേക്ക് സൂക്ഷിക്കുക.

ഇവിടെ റഫ്രിജറേറ്റർ ഒരു ദ്വിതീയ സംഭരണി അഥവാ ഹാർഡ് ഡിസ്ക് ആയി പ്രവർത്തിക്കുന്നു. കൂടുതൽ പച്ചക്കറികൾ കൂടുതൽ കാലത്തേക്ക് സൂക്ഷിച്ചു വെക്കുന്നു. പച്ചക്കറി അരിയാനുപയോഗിച്ച മേശ കമ്പ്യൂട്ടറിന്റെ മദർബോർഡിനെപോലെ പ്രവർത്തിക്കുന്നു. എല്ലാ പ്രവർത്തനങ്ങളും അവിടെവച്ചാണ് നടക്കുന്നത് (കമ്പ്യൂട്ടറിനുള്ളിലുള്ള). കഷണം നുറുക്കാനുള്ള ബോർഡാണ് ALU - പ്രവർത്തനങ്ങൾ നടക്കുന്നത് അവിടെയാണ്. പാചകക്കുറിപ്പാണ് കൺട്രോൾ യൂണിറ്റ്. മുറിക്കേണ്ട പലകയിൽ എന്താണ് ചെയ്യേണ്ടത് (ALU) എന്ന് ഇതിൽ പറയുന്നു. മേശപ്പുറത്തെ ഒഴിഞ്ഞ ഭാഗമാണ് റാം പെട്ടെന്ന് എടുക്കുന്നതിനുവേണ്ടി എല്ലാ പച്ചക്കറികളും റഫ്രിജറേറ്ററിൽ നിന്ന് എടുത്ത് (Counter top) മേശപ്പുറത്ത് വെക്കുന്നു. ഇവിടെ പച്ചക്കറികൾ റഫ്രിജറേറ്ററിൽ നിന്നും (disk) എടുക്കുന്നതിനേക്കാൾ വേഗത്തിൽ മേശപ്പുറത്തു നിന്ന് എടുക്കാൻ സാധിക്കും. എന്നാൽ കൂടുതൽ അളവിൽ, കുറെ നേരം വെച്ചിരിക്കാൻ സാധിക്കുകയില്ല. ഭാഗികമായി മുറിച്ച പച്ചക്കറികൾ താൽക്കാലികമായി വച്ചിരിക്കുന്ന പലകയുടെ മൂലകൾ രജിസ്റ്ററുകൾക്ക് തുല്യമാണ്. ഈ മൂലകളിൽ വച്ചിരിക്കുന്ന പച്ചക്കറികൾ വളരെ വേഗത്തിൽ എടുക്കാൻ സാധിക്കും, എന്നാൽ കൂടുതൽ നേരം വെച്ചിരിക്കാൻ സാധിക്കുകയില്ല. സാലഡ് വെച്ചിരിക്കുന്ന പാത്രം ക്യാഷ് മെമ്മറി പോലെയാണ്. ഇത് പലകയുടെ മൂലയിൽ താൽക്കാലികമായി മാറ്റപ്പെടുന്ന മുറിച്ച പച്ചക്കറികൾ സൂഷിക്കുന്നതിനോ (അവിടെ കൂടുതൽ ഉണ്ടെങ്കിൽ), സാലഡ് തിരിച്ച് റഫ്രിജറേറ്ററിലേക്ക് വെക്കുന്നതിനോ (ഡാറ്റ തിരിച്ച് ഡിസ്കിലേക്ക് വെക്കുന്നതുപോലെ) അല്ലെങ്കിൽ ഡിന്നർ ടേബിളിലേക്ക് വെക്കുന്നതിനോ ആണ്. (ഔട്ട്പുട്ട് ആയി ഡാറ്റ ഔട്ട്പുട്ട് ഉപകരണത്തിലേക്ക് നൽകുന്നതുപോലെ).

### 3.3.6 കമ്പ്യൂട്ടറിൽ മെമ്മറിയുടെ പ്രാധാന്യം (Role of memories in computers)

തൊഴിലാളികളുടെ ശമ്പളം തയാറാക്കാൻ ഉപയോഗിക്കുന്ന ഒരു പേ റോൾ പ്രോഗ്രാം പരിഗണിക്കുക. എല്ലാ തൊഴിലാളികളുടെയും ഡാറ്റ ഹാർഡ് ഡിസ്കിൽ ലഭ്യമായിരിക്കും. ഓരോ തൊഴിലാളികളെക്കുറിച്ചുള്ള വിവരങ്ങൾ റാമിലേക്ക് എടുക്കുന്നു. അവിടെ നിന്നും ശമ്പളം കണക്കുകൂട്ടുന്നതിനാവശ്യമായ വിവരങ്ങൾ (ബോണസ്, കുറയ്ക്കേണ്ടവ എന്നിവ) ക്യാഷ് മെമ്മറിയിലേക്കും എടുക്കുന്നു. എത്ര മണിക്കൂർ ജോലി ചെയ്തുവെന്നും അതിനുള്ള ശമ്പളം എത്രയാണെന്നുമുള്ള ഡാറ്റ ബന്ധപ്പെട്ട രജിസ്റ്ററുകളിലേക്ക് മാറുന്നു. കൺട്രോൾ യൂണിറ്റിൽ നിന്നുള്ള നിർദേശത്തിനനുസരിച്ച് ജോലി ചെയ്ത സമയം പരിഗണിച്ച് ALU ശമ്പളം കണക്കാക്കുന്നു. (അധിക സമയം ജോലി ചെയ്തത്, ബോണസ് എന്നിവ) ക്യാഷ് മെമ്മറിയിൽ നിന്നും രജിസ്റ്ററിലേക്ക് മാറ്റുന്നു. ഒരാളുടെ ശമ്പളത്തിന്റെ കണക്കുകൂട്ടലുകൾ CPU പൂർത്തിയാക്കി കഴിയുമ്പോൾ അടുത്ത ആളുടെ വിവരങ്ങൾ ദ്വിതീയ മെമ്മറിയിൽ നിന്നും റാമിലേക്ക് കൊണ്ടു വരുന്നു. അവിടെനിന്ന് ക്യാഷ് മെമ്മറിയിലേക്കും തുടർന്ന് രജിസ്റ്ററിലേക്കും എത്തിക്കുന്നു.

സംഭരണശേഷിയുടെയും അതിന്റെ വേഗതയുടെയും അടിസ്ഥാനത്തിൽ വിവിധതരം മെമ്മറികളുടെ ഒരു ശ്രേണി ചിത്രം 3.9ൽ കൊടുത്തിരിക്കുന്നു.




*ചിത്രം 3.9: മെമ്മറി ശ്രേണി*

വിവിധ തരം ഡാറ്റ സംഭരണികളുടെ സവിശേഷതകൾ പട്ടിക 3.2 ൽ സംഗ്രഹിച്ചിരിക്കുന്നു.

| സംഭരണം | വേഗത | സംഭരണശേഷി | ആനുപാതിക മൂല്യം | അസ്ഥിരമായത് (Volatile) |
|---|---|---|---|---|
| രജിസ്റ്റർ | അതിവേഗം | വളരെ കുറവ് | ഏറ്റവും കൂടുതൽ | അതെ |
| ക്യാഷ് | കൂടിയ വേഗം | കുറവ് | വളരെ കൂടുതൽ | അതെ |
| റാം (RAM) | വളരെവേഗം | കുറവ്/മിതം കുറവ് | കൂടുതൽ | അതെ |
| ഹാർഡ് ഡിസ്ക് | മിത വേഗം | വളരെ കൂടുതൽ | വളരെ കുറവ് | അല്ല |

*പട്ടിക 3.2: വിവിധതരം മെമ്മറികളുടെ സവിശേഷതകളുടെ താരതമ്യം*

**സ്വയം പരിശോധിക്കുക**

1. കമ്പ്യൂട്ടറിലെ അതിവേഗതയുള്ള മെമ്മറിയാണ് ----------
2. ഡാറ്റ പ്രൊസസ്സിംഗ് നിർവ്വചിക്കുക
3. എന്താണ് ക്യാഷ് മെമ്മറി?
4. പ്രോഗ്രാം കൗണ്ടർ രജിസ്റ്ററിന്റെ ഉപയോഗമെന്താണ്?
5. ALU വിന്റെ ഉപയോഗമെന്ത്?

## ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ (Input/output Devices)

പുറംലോകവുമായി ആശയവിനിമയം സാധ്യമാകുന്നില്ലെങ്കിൽ കമ്പ്യൂട്ടർ കൊണ്ട് യാതൊരു ഉപയോഗവും ഇല്ല. കമ്പ്യൂട്ടറും ഉപയോക്താവും തമ്മിൽ ആശയവിനിമയം നടത്തുന്നതിന് ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ ആവശ്യമാണ്. ചുരുക്കത്തിൽ ഇൻപുട്ട് ഉപകരണങ്ങൾ കമ്പ്യൂട്ടറിലെക്ക് ഡാറ്റയും നിർദ്ദേശങ്ങളും കൊടുക്കുകയും ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ കമ്പ്യൂട്ടറിലുള്ള വിവരങ്ങൾ അവതരിപ്പിക്കുകയും ചെയ്യുന്നു. വിവിധതരത്തിലുള്ള പോർട്ടുകൾ വഴിയോ വയർലെസ് സാങ്കേതികവിദ്യയുടെ സഹായത്താലോ ഈ ഇൻപുട്ട്/ഔട്ട്പുട്ട് ഉപകരണങ്ങളെ CPU വുമായി ബന്ധിപ്പിക്കാം. ഇവ CPUവിന് പുറത്തായതിനാൽ അവയെ പെരിഫറലുകൾ എന്നുവിളിക്കുന്നു.

താഴെ കൊടുത്തിരിക്കുന്ന പട്ടികയിൽ വിവിധതരത്തിലുള്ള ഇൻപുട്ട് ഉപകരണങ്ങളും അവയുടെ ഉപയോഗങ്ങളും കാണിച്ചിരിക്കുന്നു.

| ഉപകരണം | സവിശേഷതകൾ/ഉപയോഗങ്ങൾ |
|---|---|
| **കീബോർഡ്**  | ഉപയോക്താവിന് കമ്പ്യൂട്ടറിലേക്ക് അക്ഷരങ്ങൾ, സംഖ്യകൾ, ചിഹ്നങ്ങൾ തുടങ്ങിയവ അടങ്ങുന്ന ടെക്സ്റ്റ് ഡാറ്റ ഇൻപുട്ട് ചെയ്യുവാൻ സഹായിക്കുന്നു. നമ്മൾ അമർത്തുന്ന കീ തിരിച്ചറിഞ്ഞ് അതിന് അനുസൃതമായി കമ്പ്യൂട്ടറിന് മനസിലാകുന്ന ആസ്കി കോഡ് (ASCII) സൃഷ്ടിക്കുന്നു. കേബിൾ ഉള്ളതും ഇല്ലാത്തതുമായ കീബോർഡുകൾ വിപണിയിൽ ലഭ്യമാണ്. |
| **മൗസ്**  | ഒരു പരന്ന പ്രതലത്തിൽ/മൗസ് പാഡിൽ വെച്ച് കമ്പ്യൂട്ടറിന്റെ സ്ക്രീനിലെ മൗസ് പോയിന്ററിനെ എങ്ങോട്ടു വേണമെങ്കിലും ചലിപ്പിക്കാൻ സാധിക്കുന്നതോ കർസർ ഒരു സ്ഥാനത്തെത്തിക്കുവാൻ ഉപയോഗിക്കുന്നതോ ആയ കയ്യിൽ ഒതുങ്ങുന്ന ചെറിയ ഉപകരണമാണിത്. ബാൾ, ഒപ്റ്റിക്കൽ, ലേസർ എന്നിവ വിവിധ തരം മൗസുകളാണ്. കേമ്പിൾ ഇല്ലാത്ത മൗസും ലഭ്യമാണ്. |
| **ലൈറ്റ് പെൻ**  | പേനയുടെ ആകൃതിയിലുള്ള ഒരു പോയിന്റിംങ്ങ് ഉപകരണമാണിത്. സ്ക്രീനിൽ നേരിട്ട് വരക്കാൻ കഴിയുന്നു എന്നതാണ് ഇതിന്റെ മേന്മ. കമ്പ്യൂട്ടറിന്റെ സഹായത്തോടെയുള്ള രൂപകൽപ്പനക്കും (CAD) ചിത്രരചനക്ക് വേണ്ടി കലാകാരൻമാരും ഫാഷൻ ഡിസൈനർമാരും എഞ്ചിനീയർമാരും ഇത് ഉപയോഗിക്കുന്നു. |
| **ടച്ച് സ്ക്രീൻ**  | സ്ക്രീനിൽ തൊട്ടുകൊണ്ട് പ്രവർത്തിപ്പിക്കുവാനോ / തിരഞ്ഞെടുക്കുവാനോ ഉപയോക്താവിനെ അനുവദിക്കുന്ന ഉപകരണമാണിത്. കൂടുതൽ കൃത്യതയ്ക്കായി സ്റ്റൈലസ് എന്ന ഉപകരണം ഉപയോഗിച്ചു ഇത് പ്രവർത്തിപ്പിക്കാവുന്നതാണ് |
| **ഗ്രാഫിക് ടാബ്ലറ്റ്**  | എഴുതുവാൻ ഉപയോഗിക്കുന്ന ഇലക്ട്രോണിക് പ്രതലവും അതിനുള്ള പ്രത്യേകതരം പേനയുമാണ് ഗ്രാഫിക് ടാബ്ലറ്റിനുള്ളത്. ചിത്രരചനകൾ നടത്തുന്നതിനായി ഇത് കലാകാരൻമാരെ അനുവദിക്കുന്നു. |

| | |
|---|---|
| **ജോയ്സ്റ്റിക്ക്**  | വീഡിയോ ഗെയിമുകൾ കളിക്കുവാനും റോബോട്ട്, ട്രയിനിംങ് സിമുലേറ്റർ എന്നിവ നിയന്ത്രിക്കുവാനും ഉപയോഗിക്കുന്നു. ഇതിന് ഏതു ദിശയിലേക്കും ചലിപ്പിക്കാവുന്ന ലംബമായ ഒരു വടിയും അതിന് മുകളിൽ ഒരു ബട്ടനുമുണ്ട്. കർസർ സൂചിപ്പിക്കുന്നത് തിരഞ്ഞെടുക്കുവാൻ ഇത് ഉപയോഗിക്കാം. |
| **മൈക്രോഫോൺ**  | അനലോഗ് രൂപത്തിലുള്ള ശബ്ദത്തെ ഇൻപുട്ട് ആയി സ്വീകരിച്ച് അതിനെ ഡിജിറ്റൽ രൂപത്തിലാക്കി മാറ്റുന്നു. ഡിജിറ്റൽ രൂപത്തിലാക്കിയ ശബ്ദത്തെ കമ്പ്യൂട്ടറിൽ സൂക്ഷിച്ചുവെച്ച് പിന്നീടുള്ള പ്രവർത്തനങ്ങൾക്കു വേണ്ടിയോ വീണ്ടും കേൾപ്പിക്കുന്നതിനോ ഉപയോഗിക്കുന്നു. |
| **സ്കാനർ**  | അക്ഷരരൂപത്തിലോ ചിത്രരൂപത്തിലോ ഉള്ള വിവരങ്ങൾ ശേഖരിച്ച് അതിനെ ഡിജിറ്റൽ രൂപത്തിലാക്കി മാറ്റുന്നു. പിന്നീട് അതിനെ കമ്പ്യൂട്ടർ ഉപയോഗിച്ച് എഡിറ്റ് ചെയ്യാവുന്നതാണ്. ഗുണമേന്മയുള്ള ചിത്രങ്ങൾ ലഭിക്കുന്നത് സ്കാനറിന്റെ റെസല്യൂഷനെ ആശ്രയിച്ചിരിക്കുന്നു. പരന്നതലമുള്ള (flat bed), ഷീറ്റ് ഫീഡ് (sheet feed), കയ്യിൽ പിടിക്കാവുന്ന (hand held) സ്കാനർ എന്നിവ വിവിധ തരത്തിലുള്ള സ്കാനറുകളാണ്. സ്കാൻ ചെയ്ത ചിത്രത്തിലെ ടെക്സ്റ്റിനെ തിരിച്ചറിയുന്നതിനും അതിനെ ടെക്സ്റ്റ് ആയി മാറ്റുന്നതിനും ഒപ്റ്റിക്കൽ ക്യാരക്ടർ റക്ഗ്നിഷൻ (OCR) സോഫ്റ്റ്വെയർ ഉപയോഗിക്കുന്നു. ഈ ടെക്സ്റ്റിനെ ഒരു ടെക്സ്റ്റ് എഡിറ്റർ ഉപയോഗിച്ച് മാറ്റം വരുത്താവുന്നതാണ്. |
| **ഒപ്റ്റിക്കൽ മാർക്ക് റീഡർ** (OMR)  | ഒരു പ്രിന്റ് ചെയ്ത ഫോമിലെ നിശ്ചയിച്ച സ്ഥാനങ്ങളിലുള്ള അടയാളങ്ങൾ ശേഖരിക്കുന്നതിനും അവ സംഭരിക്കുന്നതിനും ഉപയോഗിക്കുന്ന മറ്റൊരു സ്കാനിംഗ് ഉപകരണമാണിത്. കൈകൊണ്ട് പൂർത്തിയാക്കിയ ഫോമുകൾ വളരെ കൃത്യതയോടും വേഗത്തിലും പ്രോസസ്സ് ചെയ്യുന്നതിന് ഉപയോഗപ്രദമാണ് ഇത്. ഒബ്ജക്റ്റീവ് തരത്തിലുള്ള മത്സരപരീക്ഷകളുടെ മൂല്യനിർണ്ണയം, ചോദ്യാവലികൾ എന്നിവ ഉദാഹരണങ്ങളാണ്. |
| **ബാർകോഡ് റീഡർ/ ക്യുക്ക് റെസ് പോൺസ് (ദ്രുത പ്രതികരണ) QR കോഡ് റീഡർ**   | ഒരു നമ്പർ പ്രതിനിധീകരിക്കുന്ന വ്യത്യസ്ത കട്ടിയും അകലവുമുള്ള ലംബരേഖകളുടെ ഒരു കൂട്ടമാണ് ബാർകോഡ്. അത്തരം ബാർകോഡുകളുടെ കൂട്ടം ഉപയോഗിച്ച് ഡാറ്റ ഇൻപുട്ട് ചെയ്യാൻ ബാർകോഡ് റീഡറുകൾ സഹായിക്കുന്നു. കയ്യിൽ പിടിക്കാവുന്ന സ്കാനറുകൾ, ക്യാമറയോടു കൂടിയ മൊബൈൽ ഫോൺ സ്പെഷ്യൽ സോഫ്ട്വെയർ എന്നിവ ബാർകോഡ് റീഡറുകൾ ആയി ഉപയോഗിക്കുന്നു. QR കോഡ് ബാർകോഡ് പോലെയാണ്. ബാർകോഡുകൾ ഏകമാനം ആണ്. എന്നാൽ QR കോഡ് ദ്വിമാനം ആണ്. ദ്വിമാന രീതിയിലുള്ള ഡാറ്റ സംഭരണം, QR കോഡിന് സാധാരണ ബാർകോഡിനെ അപേക്ഷിച്ച് കൂടുതൽ ഡാറ്റ സംഭരിക്കാൻ അനുവദിക്കുന്നു. വെബ്സൈറ്റിന്റെ യു.ആർ.എല്ലും (URL). പ്ലെയിൻ ടെക്സ്റ്റും ഫോൺ നമ്പറും |

| | |
|---|---|
|  | ഇ-മെയിൽ അഡ്രസ്സും എന്നുവേണ്ട എല്ലാ അക്കങ്ങളും അക്ഷരങ്ങളും ഈ കോഡ് സൂക്ഷിച്ചു വെയ്ക്കുന്നു. QR കോഡ് വായിച്ചെടുക്കാൻ സഹായിക്കുന്നത് ബാർകോഡ് റീഡർ ഉപയോഗിച്ചോ ക്യാമറയോടു കൂടിയ മൊബൈൽ ഫോൺ ഉപയോഗിച്ചോ ഇൻസ്റ്റാൾ ചെയ്തിരിക്കുന്ന പ്രത്യേക സോഫ്റ്റ്വെയർ ഉപയോഗിച്ചോ ആണ്. |
| **ബയോമെട്രിക് സെൻസർ**  | മനുഷ്യന്റെ അനന്യമായ ശാരീരിക പ്രത്യേകതകൾ ഏറ്റവും കൃത്യതയോടെ വായിച്ചെടുക്കാൻ സഹായിക്കുന്ന ഒരു ഉപകരണമാണ് ബയോമെട്രിക് സെൻസർ. ശാരീരിക സവിശേഷതകളായ വിരലടയാളം, റെറ്റിന, ഐറിസ് എന്നിങ്ങിനെയുള്ളവ തിരിച്ചറിഞ്ഞ്, ഒരാളുടെ ഐഡന്റിറ്റി ഉറപ്പുവരുത്താൻ ഉപയോഗിക്കുന്ന ബയോമെട്രിക് സംവിധാനത്തിലെ ഒഴിവാക്കാനാവാത്ത ഘടകമാണ് ഇത്. |
| **സ്മാർട്ട് കാർഡ് റീഡർ**  | ഡാറ്റ സൂക്ഷിക്കുവാനും കൈമാറ്റം ചെയ്യുവാനും സാധിക്കുന്ന ഒരു പ്ലാസ്റ്റിക് കാർഡാണ് സ്മാർട്ട് കാർഡ്. ഇതിൽ ഒരു മൈക്രോ പ്രൊസസ്സറോ, മെമ്മറിയോ കാണപ്പെടുന്നു. ബാങ്കിംഗ്, ആരോഗ്യമേഖല, ടെലിഫോൺ വിളികൾ, ഇലക്ട്രോണിക് ക്യാഷ് പെയ്മെന്റുകൾ എന്നീ വിവിധ പ്രവർത്തനങ്ങൾക്ക് സ്മാർട്ട് കാർഡ് ഉപയോഗിക്കുന്നു. സ്മാർട്ട് കാർഡിലെ ഡാറ്റ എടുക്കുവാൻ ഉപയോഗിക്കുന്ന ഉപകരണമാണ് സ്മാർട്ട് കാർഡ് റീഡർ. |
| **ഡിജിറ്റൽ ക്യാമറ**  | ചിത്രങ്ങളും വീഡിയോകളും എടുത്ത് ഡിജിറ്റൽ രൂപത്തിലേക്ക് മാറ്റാൻ ഡിജിറ്റൽ ക്യാമറ ഉപയോഗിക്കുന്നു. ചിത്രങ്ങൾ മെമ്മറിയിൽ സൂക്ഷിച്ചു വയ്ക്കുകയും അതിനുശേഷം കമ്പ്യൂട്ടറിലേക്ക് മാറ്റുകയും ചെയ്യുന്നു. ഡിജിറ്റൽ ക്യാമറയുടെ ചെലവ് കുറഞ്ഞതും ഒതുക്കമുള്ളതുമായ പതിപ്പാണ് വെബ്ബ് ക്യാമറ. വീഡിയോ കോളിംഗ്, വീഡിയോ ചാറ്റിംങ്ങ് എന്നിവയ്ക്കായി കമ്പ്യൂട്ടറിൽ ഇത് ഉപയോഗിക്കുന്നു. ഇതിന് ആന്തരിക മെമ്മറി ഉണ്ടായിരിക്കുകയില്ല. |

*പട്ടിക 3.3: ഇൻപുട്ട് ഉപകരണങ്ങളും അവയുടെ ഉപയോഗവും*

**ചില ഔട്ട്പുട്ട് ഉപകരണങ്ങളും അവയുടെ പ്രത്യേകതകളും നമുക്ക് ഇപ്പോൾ നോക്കാം. വിവിധ ഔട്ട്പുട്ട് ഉപകരണങ്ങളും അവയുടെ ഉപയോഗവും പട്ടിക 3.4 ൽ കാണാം**

| ഉപകരണം | സവിശേഷതകൾ/ഉപയോഗങ്ങൾ |
|---|---|
| **വിഷ്വൽ ഡിസ്പ്ലേ യൂണിറ്റ് (VDU)**  | CRT മോണിറ്ററുകൾ,ഗ്യാസ് പ്ലാസ്മ മോണിറ്ററുകൾ, എൽ.സി.ഡി മോണിറ്ററുകൾ, ടി.എഫ്.ടി മോണിറ്ററുകൾ, എൽ.ഇ.ഡി മോണിറ്ററുകൾ, ഓർഗാനിക് ലൈറ്റ് എമിറ്റിങ്ങ് ഡയോഡ് മോണിറ്ററുകൾ (OLED) തുടങ്ങിയവ ഡിസ്പ്ലേ ഉപകരണങ്ങളാണ്. ഡിസ്പ്ലേ ഉപകരണങ്ങളിൽ കാണുന്ന വിവരങ്ങളെ സോഫ്റ്റ്കോപ്പി എന്നു വിളിക്കുന്നു. സ്ക്രീനിന് കുറുകെ ഡയഗണലായി ഇഞ്ചുകളിൽ ആണ് മോണിറ്ററിന്റെ വലുപ്പം കണക്കാക്കുന്നത്. |

**എൽ.സി.ഡി. പ്രൊജക്ടർ (LCD Projector)**

വീഡിയോയും ചിത്രങ്ങളും കമ്പ്യൂട്ടറിലെ ഡാറ്റയും ഒരു വലിയ സ്ക്രീനിലോ മറ്റേതെങ്കിലും പരന്ന പ്രതലത്തിലോ പ്രദർശിപ്പിക്കാനുപയോഗിക്കുന്ന ഒരുതരം വീഡിയോ പ്രൊജക്ടറാണ് എൽ.സി.ഡി. പ്രൊജക്ടർ. ഉയർന്ന തീവ്രതയിലുള്ള പ്രകാശം എൽ.സി.ഡി. യിലുള്ള ആയിരക്കണക്ക് പിക്സലുകളിലൂടെ കടത്തിവിടുകയും അവയെ ലെൻസ് ഉപയോഗിച്ച് കേന്ദ്രീകരിക്കുകയും ചെയ്യുന്നു.

**പ്രിന്റർ (Printer)**

ഡോട്മെട്രിക്സ് പ്രിന്റർ

ഇൻക്ജെറ്റ് പ്രിന്റർ

ലേസർ പ്രിന്റർ

തെർമൽ പ്രിന്റർ

ഹാർഡ് കോപ്പി ഔട്ട്പുട്ട് നിർമ്മിക്കുന്നതിനാണ് പ്രിന്റർ ഉപയോഗിക്കുന്നത്. പേപ്പറുകളിൽ പ്രിന്റ് ചെയ്തെടുക്കുന്ന ഔട്ട്പുട്ടുകൾ ഹാർഡ് കോപ്പി എന്നാണറിയപ്പെടുന്നത്. പ്രിന്ററിനെ ഇംപാക്ട് പ്രിന്റർ, നോൺ ഇംപാക്ട് പ്രിന്റർ എന്നിങ്ങിനെ രണ്ടായി തരംതിരിച്ചിരിക്കുന്നു. ഡോട്ട് മെട്രിക്സ് എന്നത് ഇംപാക്ട് പ്രിന്റർ വിഭാഗത്തിൽപെടുന്നു. വളരെ കുറഞ്ഞ പ്രിന്റിങ്ങ് ചെലവിൽ കാർബൺ കോപ്പികൾ പ്രിന്റ് ചെയ്യുന്നു. ഒരു പ്രിന്ററിന്റെ വേഗത അളക്കുന്നത്, ഒരു നിശ്ചിത സമയത്ത് എത്ര കാരക്ടറുകൾ പ്രിന്റ് ചെയ്യുന്നു, കാരക്ടേഴ്സ് പെർ സെക്കണ്ട് (CPS), ലൈൻസ് പെർ മിനുട്ട് (LPM) അഥവാ പേജസ് പെർ മിനുട്ട് (PPM) എന്നിവയെ ആശ്രയിച്ചാണ്. ഈ പ്രിന്ററുകൾ വേഗത കുറഞ്ഞവയും ശബ്ദം ഉണ്ടാക്കുന്നവയുമാണ്. ഇങ്ക്ജെറ്റ് പ്രിന്ററുകൾ നോൺഇംപാക്ട് പ്രിന്റർ എന്ന വിഭാഗത്തിൽപെടുന്നു. ഇങ്ക്ജെറ്റ് പ്രിന്ററുകളിൽ അക്ഷരങ്ങൾ രൂപപ്പെടുത്തുന്നത് പ്രിന്റ് ഹെഡിലൂടെ ചെറിയ മഷിത്തുള്ളികൾ പേപ്പറിലേക്ക് സ്പ്രേ ചെയ്താണ്. ഇങ്ക്ജെറ്റ് പ്രിന്ററുകൾ ചെലവ് കുറഞ്ഞവയാണെങ്കിലും ദീർഘകാലത്തെക്ക് ഉപയോഗിക്കുമ്പോൾ അവയുടെ ചിലവ് വർദ്ധിക്കുന്നു. ലേസർ പ്രിന്ററുകൾ, നോൺ ഇംപാക്ട് വിഭാഗത്തിൽപെടുന്നു. നല്ല നിലവാരത്തിലുള്ള ചിത്രങ്ങൾ സൃഷ്ടിക്കാൻ ഇവ ഉപയോഗിക്കുന്നു. ഡോട്ട് മാട്രിക്സ് പ്രിന്ററിനെ അപേക്ഷിച്ച് ശബ്ദമില്ലാതെയും വളരെ വേഗത്തിലും പ്രിന്റ് ചെയ്യാൻ ലേസർ പ്രിന്ററുകൾ ഉപയോഗിക്കുന്നു. മോണോക്രോം ലേസർ പ്രിന്ററുകളും കളർ ലേസർ പ്രിന്ററുകളും വിപണിയിൽ ലഭ്യമാണ്. പലവിധത്തിലുള്ള കളർ ടോണർ കാട്രിഡ്ജുകൾ ഉപയോഗിച്ച് വിവിധ നിറങ്ങളിലുള്ള ഔട്ട്പുട്ടുകൾ സൃഷ്ടിക്കാൻ കളർ ലേസർ പ്രിന്ററുകൾ ഉപയോഗിക്കുന്നു. ഇവ ചെലവേറിയവയാണ്. ലേസർ പ്രിന്ററുകൾ വേഗതയേറിയവയാണ്. ഇവയുടെ വേഗത അളക്കുന്നത് പേജസ് പെർ മിനുട്ടിലാണ് (PPM). തെർമൽ പ്രിന്റർ നോൺ ഇംപാക്ട് പ്രിന്റർ വിഭാഗത്തിൽപ്പെടുന്നു. ഹീറ്റ് സെൻസിറ്റീവ് തെർമൽ പേപ്പർ എന്ന പ്രത്യേകതരം പേപ്പർ ആണ് ഇവിടെ ഉപയോഗിക്കുന്നത്. ഈ പേപ്പർ ഉപയോഗിച്ച് പ്രിന്റ്

| | |
|---|---|
| | ഹെഡിന് മുകളിലൂടെ ഇത് നീങ്ങുമ്പോൾ അച്ചടിക്കേണ്ടതായ വിവരത്തിന്റെ പ്രതിരൂപം സൃഷ്ടിക്കുന്നു. എടുത്തു കൊണ്ടു പോകാൻ സാധിക്കുന്ന പ്രചാരത്തിലുള്ള പ്രിന്ററാണിത്. |
| **പ്ലോട്ടർ**  | ഗ്രാഫിന്റേയും രൂപകൽപ്പനയുടെയും (design) ഹാർഡ്കോപ്പി നിർമ്മിക്കാൻ ഉപയോഗിക്കുന്ന ഔട്ട്പുട്ട് ഉപകരണമാണ് പ്ലോട്ടർ. വലിയ തോതിലുള്ള ഗ്രാഫുകളും, ഭൂപടങ്ങളും, പോസ്റ്ററുകളും, നിർമ്മാണത്തിനാവശ്യമായ മാപ്പുകളും, എഞ്ചിനീയറിങ്ങിനാവശ്യമായ ചിത്രങ്ങളും പ്രിന്റ് ചെയ്യാൻ ഉപയോഗിക്കുന്ന ഉപകരണമാണ് പ്ലോട്ടറുകൾ. രണ്ട് തരത്തിലുള്ള പ്ലോട്ടറുകളുണ്ട്. ഡ്രം പ്ലോട്ടർ (Drum Plotter), ഫ്ളാറ്റ് ബെഡ് പ്ലോട്ടർ (Flat bed plotter). ഡ്രം പ്ലോട്ടർ റോളർ പ്ലോട്ടർ എന്നും ഫ്ളാറ്റ് ബെഡ് പ്ലോട്ടർ ടേബിൾ പ്ലോട്ടർ എന്നും അറിയപ്പെടുന്നു |
| **3D പ്രിന്റർ**  | 3D വസ്തുക്കൾ പ്രിന്റ് ചെയ്യാൻ ഉപയോഗിക്കുന്ന ഔട്ട്പുട്ട് ഉപകരണമാണ് 3D പ്രിന്റർ. പദാർത്ഥങ്ങളിൽ പലതരത്തിലുള്ള വസ്തുക്കൾ നിർമ്മിക്കുവാൻ സാധിക്കുന്നു. പ്രിന്റ് ചെയ്യേണ്ട വസ്തുവിനെ ആയിരക്കണക്കിന് വളരെ ചെറിയ കഷണങ്ങളാക്കി മാറ്റുന്നു. താഴെ മുതൽ മുകൾ വരെ കഷ്ണം കഷ്ണമായി പ്രിന്റ് ചെയ്യുന്നു. ഈ ചെറിയ ലയറുകൾ ഒന്നിച്ച് കൂട്ടിചേർത്ത് യഥാർത്ഥ വസ്തുവിന്റെ മാതൃക ഉണ്ടാക്കുന്നു. സെറാമിക് കപ്പ്, കളിപ്പാട്ടങ്ങൾ എന്നിവ ഉദാഹരണങ്ങളാണ്. |
| **ഓഡിയോ ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ**  | കമ്പ്യൂട്ടർ ഉണ്ടാക്കുന്ന ശബ്ദം പുറത്തേക്ക് ലഭ്യമാക്കുന്ന ഉപകരണങ്ങളാണ് ഓഡിയോ ഔട്ട്പുട്ട്. ശബ്ദം സൃഷ്ടിക്കുന്ന ഔട്ട്പുട്ട് ഉപകരണമാണ് സ്പീക്കറുകൾ. ഓഡിയോ പോർട്ടുകൾ വഴിയാണ് ഇവ കമ്പ്യൂട്ടറുമായി ബന്ധിപ്പിക്കുന്നത്. |

*പട്ടിക 3.4: ഔട്ട്പുട്ട് ഉപകരണങ്ങളും അവയുടെ ഉപയോഗവും*

നമ്മൾ പലതരത്തിലുള്ള പ്രിന്ററുകൾ കണ്ടുവല്ലോ. ഈ പ്രിന്ററുകളുടെ വ്യത്യസ്ത സ്വഭാവ സവിശേഷതകളുടെ താരതമ്യം പട്ടിക 3.5 ൽ കൊടുത്തിരിക്കുന്നു.

| സവിശേഷതകൾ | ലേസർ പ്രിന്ററുകൾ | ഇങ്ക്ജെറ്റ് പ്രിന്ററുകൾ | തെർമൽ പ്രിന്ററുകൾ | ഡോട്ട്മെട്രിക്സ് പ്രിന്ററുകൾ |
|---|---|---|---|---|
| **പ്രിന്റ് ചെയ്യാനുപയോഗിക്കുന്ന വസ്തു** | പൊടി രൂപത്തിലുള്ള മഷി (ഇങ്ക് പൗഡർ) | ദ്രവ രൂപത്തിലുള്ള മഷി | ഹീറ്റ് സെൻസിറ്റീവ് പേപ്പർ | മഷി പുരട്ടിയ റിബ്ബൺ |
| **എങ്ങനെ ഇത് പ്രിന്റ് ചെയ്യുന്നു** | പേപ്പർ ചൂടാക്കി അതിൽ പൗഡർ ഒഴുക്കുന്നു | വളരെ ചെറിയ ദ്വാരങ്ങളിലൂടെ ദ്രവരൂപത്തിലുള്ള മഷി പേപ്പറിൽ സ്പ്രേ ചെയ്യുന്നു | തെർമൽ പ്രിന്റ് ഹെഡിന് മുകളിലൂടെ തെർമൽ പേപ്പർ പാസ് ചെയ്യുന്നു | പേപ്പറിന് മുകളിലുള്ള റിബ്ബണിൽ പിന്നുകൾ അമർത്തുന്നു |
| **പ്രിന്റിങ്ങിന്റെ വേഗത** | മിനുട്ടിൽ 20 പേജ് | മിനുട്ടിൽ 6 പേജ് | സെക്കന്റിൽ 150 mm | ഒരു സെക്കന്റിൽ 30 മുതൽ 550 ക്യാരക്ടറുകൾ വരെ |
| **ഗുണമേന്മ** | ഗുണമേൻമയുള്ള പ്രിന്റിംഗ്, ബ്ലാക്ക് & വൈറ്റിന് നല്ലത് | ഗുണമേന്മയുള്ളത് പ്രത്യേകിച്ചു ചെറിയ ഫോണ്ടുകൾക്ക് | ചിത്രങ്ങൾ പ്രിന്റ് ചെയ്യുന്നതിന് യോജിച്ചതല്ല, ടെക്സ്റ്റുകൾ പ്രിന്റ് ചെയ്യാൻ നല്ലത്. | ചിത്രങ്ങൾ പ്രിന്റ് ചെയ്യാൻ യോജിച്ചതല്ല. ടെക്സ്റ്റിന്റെ അടിസ്ഥാനത്തിൽ നല്ലത്. |
| **ഗുണങ്ങൾ** | ശബ്ദമില്ലാതെ പ്രിന്റ് ലഭിക്കും, വേഗത്തിൽ പ്രിന്റ് ലഭിക്കും, ഉയർന്ന പ്രിന്റ് നിലവാരം | ശബ്ദമില്ല, ഉയർന്ന പ്രിന്റ് നിലവാരം, ഉപകരണത്തിന് പ്രത്യേക തയാറെടുപ്പ് ആവശ്യമില്ല, കുറഞ്ഞ ഉപകരണ വില | ശബ്ദമില്ല, വേഗത, ചെറുത്, ഭാരം കുറവ്, കുറഞ്ഞ വൈദ്യുതി ഉപയോഗം, കൊണ്ടു നടക്കാവുന്നത് | റിബ്ബണിന് വില കുറവായതു കൊണ്ട് പ്രിന്റിങ്ങിന് വില കുറവ്, കാർബൺ കോപ്പി എടുക്കാം. |
| **ദോഷങ്ങൾ** | പേപ്പർ കുടുങ്ങാനുള്ള സാധ്യത കൂടുതൽ, ടോണറിന്റെ വില കൂടുതൽ, ഉപകരണത്തിനും വില കൂടുതൽ | മഷിക്ക് വില കൂടുതൽ, മഷി വാട്ടർ പ്രൂഫ് അല്ല, അതിനാൽ സ്പ്രേ ചെയ്യുമ്പോൾ തടസ്സങ്ങൾ നേരിടാം. | സ്പെഷ്യൽ തെർമൽ പേപ്പർ ആവശ്യമായി വരുന്നു. പ്രിന്റിംഗ് ഗുണമേന്മ കുറവാണ്. | പ്രാരംഭ മുടക്ക് മുതൽ കൂടുതലാണ്, വേഗത്തിൽ പ്രിന്റ് ചെയ്യുന്നില്ല, ശബ്ദം കൂടുതലാണ്. |

*പട്ടിക 3.5: പ്രിന്ററുകളുടെ താരതമ്യം*

## 3.4 ഇ-വേസ്റ്റ് (e-Waste)

ഇലക്ട്രോണിക് ഉപകരണങ്ങൾ ഉപയോഗശൂന്യമാകുമ്പോൾ അവ അറിയപ്പെടുന്നത് ഇ-വേസ്റ്റ് എന്ന പേരിലാണ്. ഉപേക്ഷിക്കപ്പെട്ട കമ്പ്യൂട്ടറുകൾ, ഓഫീസ് ഇലക്ട്രോണിക് ഉപകരണങ്ങൾ, വിനോദത്തിനുപയോഗിക്കുന്ന ഉപകരണങ്ങൾ, മൊബൈൽ ഫോൺ, ടെലിവിഷൻ, റഫ്രിജറേറ്റർ ഇവയെല്ലാം ഇ-വേസ്റ്റ് എന്ന വിഭാഗത്തിൽപെടുന്നു. പുനരുപയോഗം, പുനർവിൽപ്പന, വീണ്ടെടുക്കൽ, പുനരുൽപാദനം എന്നിങ്ങനെ നിർമാർജ്ജനം ചെയ്യാവുന്ന ഉപയോഗം കഴിഞ്ഞ ഇലക്ട്രോണിക് ഉപകരണങ്ങളെയും ഇ-വേസ്റ്റ് ആയി പരിഗണിക്കാം.

ആധുനിക ജീവിതത്തിൽ ഒഴിച്ചു കൂടാൻ കഴിയാത്ത ഇലക്ട്രോണിക് ഉപകരണങ്ങളാണ് ഡസ്ക്ടോപ്പ് കമ്പ്യൂട്ടർ, ലാപ്ടോപ്പ്, മൊബൈൽ ഫോൺ, റഫ്രിജറേറ്റർ, ടെലിവിഷൻ എന്നിവ. നമ്മുടെ ആവശ്യങ്ങൾക്കനുസരിച്ച് ഓരോ വർഷവും പുതിയ ഉപകരണങ്ങൾ നമ്മൾ വാങ്ങുന്നു. ഓരോ വർഷവും 300 ദശലക്ഷം കമ്പ്യൂട്ടറുകളും 1 ലക്ഷം കോടി സെൽഫോണുകളും ഉൽപ്പാദിപ്പിക്കപ്പെടുന്നു. രണ്ടോ മൂന്നോ വർഷത്തിനുള്ളിൽ തന്നെ ഈ ഇലക്ട്രോണിക് ഉപകരണങ്ങൾ ഉപയോഗശൂന്യമായി തീരുന്നു. ഓരോ വർഷവും ലോകത്താകമാനമുള്ള വേസ്റ്റ് 8% നിരക്കിൽ വർദ്ധിക്കുന്നു എന്ന് കണക്കുകൾ സൂചിപ്പിക്കുന്നു.

മാറിക്കൊണ്ടിരിക്കുന്ന സാങ്കേതികവിദ്യയും, മാധ്യമങ്ങളിൽ വരുന്ന മാറ്റങ്ങളും, വിലയിടിവും, ആസൂത്രിതമായ അസ്ഥിരതയും ലോകത്താകമാനമുള്ള ഇ-വേസ്റ്റ് വർദ്ധിക്കുന്നതിന് കാരണമാകുന്നു. 50 ദശലക്ഷം ടൺ ഇ-വേസ്റ്റ് ഓരോ വർഷവും സൃഷ്ടിക്കപ്പെടുന്നു എന്ന് കണക്കുകൾ സൂചിപ്പിക്കുന്നു. ഇതിൽ 15 മുതൽ 20% വരെ മാത്രമേ പുതുക്കി ഉപയോഗിക്കുന്നുള്ളൂ. ബാക്കിയെല്ലാം മണ്ണിൽ കുഴിച്ചു മൂടപ്പെടുകയോ കത്തിച്ചു കളയുകയോ, മറ്റെവിടെയെങ്കിലും ഉപക്ഷിക്കുകയോ ചെയ്യുന്നു. ഇന്ത്യ, ചൈന പോലുള്ള രാജ്യങ്ങളിലും ആഫ്രിക്ക, ലാറ്റിൻ അമേരിക്ക തുടങ്ങിയ ഭൂഖണ്ഡങ്ങളിലും ഇലക്ട്രോണിക് ഉൽപ്പന്നങ്ങളുടെ വിൽപ്പന അടുത്ത 10 വർഷത്തിനുള്ളിൽ കുത്തനെ ഉയരാൻ സാധ്യതയുണ്ട്.

### 3.4.1. ഇ-വേസ്റ്റിനെ കുറിച്ച് നാം എന്തുകൊണ്ട് ഉത്കണ്ഠപ്പെടണം? (Why should we be concerned about e-Waste?)

ഇലക്ട്രോണിക് വേസ്റ്റ് വെറുമൊരു വേസ്റ്റ് അല്ല. മെർക്കുറി, ലെഡ്, കാഡ്മിയം, ബ്രോമിനേറ്റഡ് ഫ്ളൈം റിട്ടാർഡന്റ്സ് എന്നിങ്ങനെ ആരോഗ്യത്തിന് ഹാനികരമായേക്കാവുന്ന വിഷവസ്തുക്കൾ ഇതിൽ അടങ്ങിയിരിക്കുന്നു. വേണ്ടവിധം നിയന്ത്രിച്ചില്ലെങ്കിൽ ഈ വിഷവസ്തുക്കൾ കാൻസർ, പ്രത്യുൽപ്പാദനശേഷി കുറവ്, മറ്റ് ആരോഗ്യപ്രശ്നങ്ങൾ എന്നിവക്ക് കാരണമാകുന്നു. ഇ-വേസ്റ്റ് കുഴിച്ചു മൂടുന്നതിനാൽ 40% വരെ ലെഡ് മണ്ണിൽ കലരുവാൻ ഇടയാക്കുന്നു.

പട്ടിക 3.6. ൽ അപകടകരമായ ചില രാസപദാർത്ഥങ്ങൾ, അവയുടെ ഉറവിടം, പ്രത്യാഘാതം എന്നിവ കൊടുത്തിരിക്കുന്നു.

| രാസപദാർത്ഥം | ഉറവിടം | പ്രത്യാഘാതം |
|---|---|---|
| ലെഡ് | കമ്പ്യൂട്ടർ മോണിറ്റർ ഗ്ലാസ്സിലും PCB സോൾഡറിലും ഇത് കാണുന്നു. | കേന്ദ്ര നാഡീവ്യൂഹത്തെ ബാധിക്കുന്നു. രക്തചംക്രമണത്തെയും കിഡ്നിയേയും ബാധിക്കുന്നു. |
| മെർക്കുറി | PCB, എൽ.സി.ഡി. സ്ക്രീനിന്റെ ഉള്ളിലെ ലൈറ്റുകളിലും കാണുന്നു. | ചെറിയ കുട്ടികളുടെ തലച്ചോറിനെയും നാഡീവ്യൂഹത്തെയും ബാധിക്കുന്നു. മുതിർന്നവരിൽ അവയവ വൈകല്യങ്ങൾ, മാനസിക വൈകല്യങ്ങൾ, മറ്റനേകം രോഗ ലക്ഷണങ്ങൾ എന്നിവ പ്രകടമാകുന്നു. |
| കാഡ്മിയം | ചിപ്പ് റസിസ്റ്ററുകളിലും സെമികണ്ടക്ടറുകളിലും കാണുന്നു | പലതരത്തിലുള്ള കാൻസറുകൾക്ക് കാരണമാകുന്നു. കാഡ്മിയം കിഡ്നിയിൽ കുമിഞ്ഞുകൂടി അതിന് ദോഷം ചെയ്യുന്നു. |
| ബ്രോമിനേറ്റഡ് ഫ്ളെയിം റിട്ടാർഡന്റ്സ് (BFRs) | PCB യിലും ചില പ്ലാസ്റ്റിക്കുകളിലും കാണപ്പെടുന്നു. | കാൻസർ സാധ്യത വർദ്ധിപ്പിക്കുന്നു. |

*പട്ടിക 3.6: അപകടകരമായ രാസപദാർത്ഥങ്ങളും അതിന്റെ ഉറവിടവും പ്രത്യാഘാതവും.*

## 3.4.2 ഇ-വേസ്റ്റിന് എന്ത് സംഭവിക്കുന്നു (What happens to the e-Waste?)

നിർഭാഗ്യവശാൽ ഒരു ചെറിയശതമാനം ഇ-വേസ്റ്റ് മാത്രമേ പുതുക്കി ഉപയോഗിക്കുന്നുള്ളൂ. പുതുക്കൽ കേന്ദ്രങ്ങളിൽ കൊണ്ടുപോയാലും നാം പ്രതീക്ഷിക്കുന്നതുപോലെ മിക്കപ്പോഴും അവ യഥാർത്ഥത്തിൽ പുതുക്കപ്പെടുന്നില്ല. CRT മോണിറ്ററുകളിൽ ഡിസ്പ്ലേക്ക് സഹായകരമാകുന്ന ഫോസ്ഫറസും ലെഡും താരതമ്യേന കൂടിയ അളവിൽ കാണപ്പെടുന്നു. ഉപയോഗശൂന്യമായ CRT കളെ 'അപകടകരമായ വേസ്റ്റ്' എന്ന ഗണത്തിലാണ് അമേരിക്കൻ പരിസ്ഥിതി സംരക്ഷണ ഏജൻസി (United States Environmental Protection Agency- EPA) ഉൾപ്പെടുത്തിയിരിക്കുന്നത്.

ഭൂരിഭാഗം ഇ-വേസ്റ്റുകളും വേസ്റ്റ് കൂനകളിൽ തള്ളുകയോ ഇൻസിനറേറ്റർ ഉപയോഗിച്ച് കത്തിക്കുകയോ ആണ് ചെയ്യുന്നത്. ഇത്തരം വേസ്റ്റിന്റെ അനുചിതമായ നശീകരണ മാർഗ്ഗങ്ങൾ അവ ലംബിക്കുന്നതോടെ അവയിൽ നിന്നും വിലപിടിപ്പുള്ള വസ്തുക്കൾ ശേഖരിക്കുവാനോ അപകടകരമായ വിഷവസ്തുക്കളെ നിയന്ത്രിക്കാനോ സാധിക്കാതെ വരുന്നു. ഇതിന്റെ ഭാഗമായി ഇവ നമ്മുടെ മണ്ണിനെയും ജലത്തെയും വായുവിനെയും മലിനപ്പെടുത്തുന്നു.

ഇ-വേസ്റ്റ് യാതൊരു കാരണവശാലും മറ്റു ഗൃഹമാലിന്യങ്ങൾക്കൊപ്പം ഉപേക്ഷിക്കാൻ പാടില്ല. ഇവ എവിടെയാണോ ഉള്ളത് അവിടെ വച്ചുതന്നെ വേർതിരിക്കപ്പെടേണ്ടതും വിവിധ സന്നദ്ധ സംഘടനകൾക്ക് കൈമാറ്റം ചെയ്യപ്പെടേണ്ടതുമാണ്. ഇ-വേസ്റ്റ് എന്ന രൂക്ഷമായ പ്രശ്നം പരിഗണിക്കുമ്പോൾ ഗവൺമെന്റോ, വ്യാവസായിക സ്ഥാപനങ്ങളോ, പൊതുജനങ്ങളോ ഇവ നിയന്ത്രിക്കേണ്ടതിനാവശ്യമായ നടപടികൾ കൈക്കൊള്ളേണ്ടത് അത്യന്താപേക്ഷിതമാണ്.

*ചിത്രം 3.10: കേടുവന്നതും കാലഹരണപ്പെട്ടതുമായ ഇലക്ട്രോണിക് വസ്തുക്കൾ*

ഇന്ത്യാ ഗവൺമെന്റിന്റെ കേന്ദ്ര മാലിന്യ നിയന്ത്രണ ബോർഡ് (CPCB -സെൻട്രൽ പൊല്യൂഷൻ കൺട്രോൾ ബോർഡ്) രൂപീകരിച്ച 'ഇ-വേസ്റ്റ് നിയന്ത്രണ നിയമങ്ങൾ- 2011', 01—5-2012 മുതൽ നിലവിൽ വന്നു. ഈ നിയമങ്ങൾ ഇലക്ട്രിക്കൽ, ഇലക്ട്രോണിക് ഉപകരണങ്ങളുടെ നിർമ്മാണവും വിൽപ്പനയും പ്രവർത്തനവുമായി ബന്ധപ്പെട്ടു കിടക്കുന്ന എല്ലാവർക്കും (നിർമ്മാതാക്കൾ, ഉപഭോക്താക്കൾ, ശേഖരണകേന്ദ്രം) ബാധകമാണ്. സംസ്ഥാനത്ത് ഇതിന്റെ നടത്തിപ്പും മേൽനോട്ടവും നിർവ്വഹിക്കുന്നത് സംസ്ഥാന മാലിന്യ നിയന്ത്രണ ബോർഡിന്റെ നേതൃത്വത്തിലാണ്.

ഇ-വേസ്റ്റിന്റെ ശേഖരണത്തിനും നിർമ്മാർജ്ജനത്തിനുംവേണ്ടി കേരള ഗവൺമെന്റ് പ്രത്യേക നിർദ്ദേശം കൊടുത്തിട്ടുണ്ട്. നിർമ്മാതാക്കളുടെയും തദ്ദേശസ്വയംഭരണസ്ഥാപനങ്ങളുടെയും മാലിന്യ നിയന്ത്രണബോർഡിന്റെയും ചുമതലകൾ ഗവൺമെന്റ് വ്യക്തമായി നിർവ്വചിച്ചിട്ടുണ്ട്. നിർമ്മാതാക്കളുടെ തിരികെ വാങ്ങൽ പദ്ധതിയിലൂടെയോ തദ്ദേശസ്വയംഭരണ സ്ഥാപനങ്ങളുടെ നേതൃത്വത്തിലുള്ള തിരിച്ചെടുക്കൽ സംവിധാനങ്ങളിലൂടെയോ ഇലക്ട്രിക്കൽ, ഇലക്ട്രോണിക്സ് പോലുള്ള ഇ-വേസ്റ്റുകൾ ശേഖരിക്കപ്പെടുന്നു. ഇവയെല്ലാം അംഗീകൃത പുതുക്കൽ കേന്ദ്രത്തിന് കൈമാറുന്നു. പ്രധാനപ്പെട്ട ബ്രാൻഡുകൾ, ഉപയോഗം കഴിഞ്ഞ ഉൽപ്പന്നങ്ങളെല്ലാം തന്നെ നിർമ്മാതാക്കൾക്കു തന്നെ തിരിച്ച് ഏൽപ്പിക്കാൻ നിർദ്ദേശം നൽകിയിട്ടുണ്ട്. അതുമല്ലെങ്കിൽ തദ്ദേശസ്വയംഭരണ സ്ഥാപനങ്ങൾ തയ്യാറാക്കിയിട്ടുള്ള ശേഖരണ കേന്ദ്രങ്ങളിലും തിരികെ ഏൽപ്പിക്കാനുള്ള സംവിധാനമുണ്ട്. ഇ-വേസ്റ്റ് നിർമ്മാർജ്ജനത്തെക്കുറിച്ച് ബോധവൽക്കരണ പരിപാടികൾ സംഘടിപ്പിക്കുന്നതിനും പുതുക്കൽ അഥവാ മാലിന്യം നിർമ്മാർജ്ജനം ചെയ്യുന്നതിനും ഉള്ള സംവിധാനങ്ങൾ പൊല്യൂഷൻ കൺട്രോൾ ബോർഡ് നിർവ്വഹിച്ചു പോരുന്നുണ്ട്.

### 3.4.3 ഇ-വേസ്റ്റ് നിർമ്മാർജ്ജന മാർഗ്ഗങ്ങൾ (e-waste disposal methods)

ഇ-വേസ്റ്റ് നിർമ്മാർജ്ജനത്തിനുവേണ്ടി താഴെ പറയുന്ന മാർഗ്ഗങ്ങൾ ഉപയോഗിക്കാം

a) **പുനരുപയോഗം (Reuse):** സെക്കന്റ് ഹാൻഡ് ഉപയോഗം അഥവാ കേടുപാടുകൾ പരിഹരിച്ച് മെച്ചപ്പെടുത്തി ഉപയോഗിക്കുക എന്നതാണ് പുനരുപയോഗം കൊണ്ട് ഇവിടെ ഉദ്ദേശിക്കുന്നത്. മിക്ക പഴയ കമ്പ്യൂട്ടറുകളും ബന്ധുക്കൾക്കോ സുഹൃത്തുക്കൾക്കോ ചില്ലറക്കച്ചവടം നടത്തുന്നവർക്കോ പൈസക്കോ അല്ലാതെയോ കൈമാറാം. ചിലത് സന്നദ്ധ സംഘടനകൾ, വിദ്യാഭ്യാസ സ്ഥാപനങ്ങൾ എന്നിവയ്ക്കോ കൈമാറാം. ഇങ്ക്ജെറ്റ് കാട്രിഡ്ജുകളും ലേസർ ടോണറുകളും പുനരുപയോഗം ചെയ്യാം. ഇത് ഇ-വേസ്റ്റിന്റെ തോത് കുറയ്ക്കാൻ സഹായിക്കും.

b) **കത്തിച്ചു കളയൽ (Incineration):** പ്രത്യേകം രൂപകൽപ്പന ചെയ്ത ഇൻസിനറേറ്ററിൽ 900 മുതൽ 1000 ഡിഗ്രി സെൽഷ്യസ് വരെ ഉള്ള ഉയർന്ന ഊഷ്മാവിൽ നിയന്ത്രണവിധേയമായി കത്തിച്ചു കളയുന്നു.

c) **ഇ-വേസ്റ്റിന്റെ പുനരുൽപാദനം (Recycling):** ഉൽപ്പന്നങ്ങളിൽ നിന്നും ഉപയോഗിക്കാൻ പറ്റുന്ന ഘടകങ്ങളുപയോഗിച്ച് പുതിയ ഉപകരണങ്ങൾ നിർമ്മിക്കുന്നതിനെയാണ് പുനരുൽപാദനം എന്ന് പറയുന്നത്. മോണിറ്ററുകൾ, കീബോർഡുകൾ, ലാപ്ടോപ്പുകൾ, മോഡം, ടെലിഫോൺ ബോർഡുകൾ, ഹാർഡ് ഡ്രൈവുകൾ, സിഡി, മൊബൈലുകൾ, ഫാക്സ് മെഷീൻ, പ്രിന്റർ, സി.പി.യു, മെമ്മറി ചിപ്പുകൾ, കൂട്ടിയോജിപ്പിക്കുന്ന വയറുകൾ, കേബിളുകൾ എന്നിവയെല്ലാം പുതുക്കി ഉപയോഗിക്കാം.

d) **മണ്ണിട്ടുമൂടൽ (Land Filling)** : ഏറ്റവും കൂടുതൽ ഉപയോഗിക്കുന്നതും എന്നാൽ ശുപാർശ ചെയ്യപ്പെടാത്തതുമായ മാർഗ്ഗമാണ് മണ്ണിട്ടു മൂടൽ.

### ഇ-വേസ്റ്റ് നിർമാർജനത്തിൽ വിദ്യാർത്ഥികളുടെ പങ്ക്

- ആവശ്യമില്ലാതെ ഇലക്ട്രോണിക് ഉപകരണങ്ങൾ വാങ്ങുന്നത് നിർത്തുക.
- ഉപകരണങ്ങൾ കേടുവരുമ്പോൾ പുതിയത് വാങ്ങുന്നതിന് പകരം അവ നന്നാക്കി ഉപയോഗിക്കുക.
- ഉപകരണങ്ങൾ പാഴാക്കാതെ അവ മറ്റുള്ളവർക്ക് വിൽക്കുകയോ സംഭാവന ചെയ്തോ അവയുടെ പ്രവർത്തന കാലയളവ് ദീർഘിപ്പിക്കുക.
- പുതിയ ഇലക്ട്രോണിക് ഉപകരണങ്ങൾ വാങ്ങുമ്പോൾ അപകടകരമായ പദാർത്ഥങ്ങൾ അടങ്ങിയിട്ടില്ലെന്ന് ഉറപ്പു വരുത്തുകയും പുതുക്കി ഉപയോഗിക്കാവുന്നവയാണെന്നും ഊർജ്ജം കുറച്ച് മാത്രം ഉപയോഗിക്കുന്നവയാണെന്നും കൂടുതൽ കാലം ഉപയോഗിക്കാവുന്നവയാണെന്നും വളരെ കുറഞ്ഞ ദുർവ്യയം മാത്രം സൃഷ്ടിക്കുന്നവയാണെന്നും ഉറപ്പു വരുത്തുക.
- ഉപയോഗ ശൂന്യമായാൽ ഉപകരണങ്ങൾ തിരികെ എടുക്കുന്ന പദ്ധതിയുണ്ടോ എന്ന് നിർമ്മാതാക്കളുടെ ഫോൺ നമ്പർ വഴിയോ വെബ്സൈറ്റ് വഴിയോ മനസ്സിലാക്കുക.
- ബാറ്ററിയിൽ പ്രവർത്തിക്കുന്ന ഉപകരണങ്ങളിൽ ഉപയോഗശേഷം കളയുന്ന ബാറ്ററിക്കു പകരം റീചാർജ്ജ് ചെയ്യാവുന്ന ബാറ്ററികൾ ഉപയോഗിക്കുക.
- ഗുണമേൻമ ഉത്തരവാദിത്വം (Warranty) ഉള്ളതും തിരികെ എടുക്കുന്നതുമായ ഉൽപ്പന്നങ്ങൾ വാങ്ങിക്കുക.

## 3.4.4. ഹരിത കമ്പ്യൂട്ടിംഗ്/ഹരിത സാങ്കേതികവിദ്യ (Green computing or Green IT)

പരിസ്ഥിതിക്ക് നാശം വരുത്താതെയുള്ള സാങ്കേതികവിദ്യയുടെ പഠനവും പ്രയോഗവുമാണ് ഹരിത കമ്പ്യൂട്ടിംഗ് അഥവാ ഹരിത സാങ്കേതികവിദ്യ. കമ്പ്യൂട്ടറിന്റെയും അനുബന്ധ ഘടകങ്ങളുടെയും രൂപകൽപ്പന, നിർമ്മാണം, ഉപയോഗം, നിർമ്മാർജ്ജനം അതുമായി ബന്ധപ്പെട്ട ഘടകങ്ങളായ മോണിറ്ററുകൾ, പ്രിന്ററുകൾ, സംഭരണ ഉപകരണങ്ങൾ എന്നിവ ഫലപ്രദമായി പരിസ്ഥിതിക്ക് യോജിക്കുന്നവിധം നടപ്പാക്കുന്നതിനെയാണ് ഹരിത കമ്പ്യൂട്ടിംഗ് എന്ന് വിളിക്കുന്നത്.

ഹരിത കമ്പ്യൂട്ടിംഗിന്റെ ആദ്യകാലത്തെ തുടക്കം 'എനർജി സ്റ്റാർ' എന്നറിയപ്പെടുന്ന സ്വമേധയാ ഉള്ള ലേബലിംഗ് പ്രോഗ്രാം ആയിരുന്നു. എല്ലാവിധ ഹാർഡ്വെയറുകളിലും ഊർജ്ജ കാര്യക്ഷമത വർദ്ധിപ്പിക്കുന്നതിനായി 1992-ൽ EPA ആണ് ഇത് നടപ്പിലാക്കിയത്. നോട്ട്ബുക്ക് കമ്പ്യൂട്ടറുകളിലും, ഡിസ്പ്ലേകളിലും എനർജി സ്റ്റാർ ലേബൽ സാധാരണ കാഴ്ചയാണ്. യൂറോപ്പിലും, ഏഷ്യയിലും ഈ പദ്ധതി നടപ്പാക്കപ്പെട്ടു. ചിത്രം 3.11ൽ സാധാരണയായി ഉപയോഗിക്കുന്ന എനർജി സ്റ്റാർ അടയാളം കാണിച്ചിരിക്കുന്നു.

*ചിത്രം 3.11: എനർജി സ്റ്റാർ ലേബൽ*

ഹരിത കമ്പ്യൂട്ടിംഗ് എന്ന ആശയത്തിന്മേലുള്ള ഗവൺമെന്റിന്റെ നിയന്ത്രണം ഭാഗികം മാത്രമാണ്. ആഗോള പരിസ്ഥിതിയെ ദോഷകരമായി ബാധിക്കാതെയുള്ള ഒരു തൊഴിൽ സംസ്കാരം വളർത്തിയെടുക്കുവാൻ കമ്പ്യൂട്ടർ ഉപയോഗിക്കുന്നവരും സ്ഥാപനങ്ങളും ശ്രദ്ധിക്കേണ്ടതുണ്ട്. അതിനായി ചെയ്യേണ്ട ചില കാര്യങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നു.

- കമ്പ്യൂട്ടർ ഉപയോഗിക്കാത്തപ്പോൾ അത് ഓഫ് ചെയ്യുക.
- ലേസർ പ്രിന്റർ പോലുള്ള പെരിഫറലുകൾ ആവശ്യമുള്ളപ്പോൾ മാത്രം ഓണാക്കുക.
- ഊർജ്ജ സംരക്ഷണ സമ്പ്രദായം ഉപയോഗിക്കുക.
- ഡസ്ക്ടോപ്പ് കമ്പ്യൂട്ടറിനുപകരം ലാപ്പ്ടോപ്പ് സാധിക്കുന്നിടത്തോളം ഉപയോഗിക്കുക.
- ആവശ്യമാണെങ്കിൽ മാത്രം പ്രിന്റൗട്ട് എടുക്കുക.
- CRT മോണിറ്ററുകൾക്കു പകരം LCD മോണിറ്ററുകൾ ഉപയോഗിക്കുക.
- എനർജി സ്റ്റാർ അടയാളമുള്ള ഹാർഡ്വെയർ, സോഫ്റ്റ്വെയർ ഉപയോഗിക്കുക.
- കേന്ദ്ര, സംസ്ഥാന, പ്രാദേശിക നിയന്ത്രണങ്ങൾക്കനുസരിച്ച് ഇ-വേസ്റ്റ് നിർമ്മാർജ്ജനം ചെയ്യുക.
- സൗരോർജ്ജം പോലുള്ള ബദൽ ഊർജ്ജ ഉറവിടങ്ങൾ ഉപയോഗിക്കുക.

## കമ്പ്യൂട്ടറുകളെ എങ്ങിനെ ഹരിതാഭമാക്കാം (How to make computers Green)

കമ്പ്യൂട്ടറുകളെ ഹരിതാഭമാക്കി മാറ്റുന്നതിൽ അവയുടെ വലിപ്പം, കാര്യക്ഷമത, അതിലടങ്ങിയിരിക്കുന്ന വസ്തുക്കൾ എന്നിവ പ്രാധാന്യമർഹിക്കുന്നു. ചെറിയ കമ്പ്യൂട്ടറുകൾ കൂടുതൽ ഹരിതാഭമാണ്. എന്തുകൊണ്ടെന്നാൽ അവ കുറച്ച് വസ്തുക്കളെ ഉപയോഗിക്കുന്നുള്ളൂ. കൂടാതെ അവയുടെ പ്രവർത്തനത്തിന് കുറഞ്ഞ വൈദ്യുതിയേ ആവശ്യമുള്ളൂ. ഹരിത കമ്പ്യൂട്ടിങ്ങിന്റെ കാര്യത്തിൽ ഊർജ്ജത്തിന്റെ കാര്യക്ഷമമായ വിനിയോഗം പ്രാധാന്യമർഹിക്കുന്നു. ലാപ്ടോപ്പുകളിൽ വലിയ കമ്പ്യൂട്ടറുകളെ അപേക്ഷിച്ച് ഊർജ്ജ വിനിയോഗം കുറവാണ്. അതുപോലെ തന്നെ LCD സ്ക്രീനുകൾ CRT മോഡലുകളെ അപേക്ഷിച്ച് വളരെ കുറച്ച് ഊർജ്ജം മാത്രമേ ഉപയോഗിക്കുന്നുള്ളൂ. ലെഡ്, മെർക്കുറി പോലുള്ള അപകടകരമായ വസ്തുക്കളുടെ ഉപയോഗം കുറക്കണം.

ഹരിത കമ്പ്യൂട്ടിംഗ് പ്രോത്സാഹിപ്പിക്കുന്നതിനുവേണ്ടി താഴെ പറയുന്ന 4 സമീപനങ്ങൾ ഉൾക്കൊള്ളിച്ചിരിക്കുന്നു.

**ഹരിത രൂപകൽപ്പന (Green Design):** കമ്പ്യൂട്ടറുകൾ, സെർവറുകൾ, പ്രിന്ററുകൾ, പ്രൊജക്ടറുകൾ, മറ്റ് ഡിജിറ്റൽ ഉപകരണങ്ങൾ രൂപകൽപ്പന ചെയ്യുമ്പോൾ അവ പരിസ്ഥിതിക്ക് അനുയോജ്യമായും ഊർജ്ജ കാര്യക്ഷമമായും ഉപയോഗിക്കാൻ തരത്തിൽ തയ്യാറാക്കുക.

**ഹരിത നിർമ്മാണം (Green Manufacturing):** കമ്പ്യൂട്ടറും മറ്റ് അനുബന്ധ ഘടകങ്ങളും നിർമ്മിക്കുമ്പോൾ ദുർവ്യയം പരമാവധി കുറച്ചു കൊണ്ട് പരിസ്ഥിതിക്ക് ദോഷമെന്നും ഉണ്ടാക്കാത്ത രീതിയിൽ തയ്യാറാക്കുക.

**ഹരിത ഉപയോഗം (Green Use):** കമ്പ്യൂട്ടറിന്റെയും അനുബന്ധ ഉപകരണങ്ങളുടെയും വൈദ്യുത ഉപഭോഗം കുറച്ചു കൊണ്ട് പരിസ്ഥിതി സൗഹാർദ്ദപരമായി ഉപയോഗിക്കുക.

**ഹരിത നിർമ്മാർജ്ജനം (Green Disposal) :** കമ്പ്യൂട്ടർ കേടുപാടുകൾ തീർത്ത് ഉപയോഗിക്കുക, ഉചിതമായ രീതിയിൽ നിർമ്മാർജ്ജനം ചെയ്യുക, ആവശ്യമില്ലാത്ത ഇലക്ട്രോണിക് ഉപകരണങ്ങൾ പുതുക്കി ഉപയോഗിക്കുക.

**സ്വയം വിലയിരുത്തുക**

1. പാരിസ്ഥിതിക ഉത്തരവാദിത്വത്തോടും പ്രകൃതിക്ക് യോജിച്ചതുമായ രീതിയിൽ കമ്പ്യൂട്ടറുകളും അവയുടെ വിഭവങ്ങളും ഉപയോഗിക്കുന്നതിനെ _____ എന്നറിയപ്പെടുന്നു
2. ഉൽപ്പന്നങ്ങളിൽ നിന്നും ഉപയോഗിക്കാൻ പറ്റുന്ന ഘടകങ്ങൾ ഉപയോഗിച്ച് പുതിയ ഉപകരണങ്ങൾ നിർമിക്കുന്നതിനെ _____ എന്നു വിളിക്കുന്നു.
3. കമ്പ്യൂട്ടറുകളിലും അവയുടെ വിഭവങ്ങളിലും ഊർജ്ജ കാര്യക്ഷമത വർദ്ധിപ്പിക്കുന്ന ലേബലിങ്ങ് പരിപാടിയെ _____ എന്നു വിളിക്കുന്നു.
4. ഏതെങ്കിലും രണ്ട് ഇൻപുട്ട്/ ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ വീതം പട്ടികപ്പെടുത്തുക.

1. *പരിസ്ഥിതിക്കും ജനങ്ങളുടെ ആരോഗ്യത്തിനും ഇ-വേസ്റ്റ് ഉണ്ടാക്കുന്ന പ്രത്യാഘാതത്തെക്കുറിച്ച് പഠിക്കുന്നതിനായി ഒരു സർവ്വെ നിങ്ങളുടെ പ്രദേശത്ത് നടത്തി റിപ്പോർട്ട് എഴുതുക.*

ചെയ്തു നോക്കൂ

2. *ഹരിത കമ്പ്യൂട്ടിംഗിന്റെ പ്രധാന്യത്തെ കുറിച്ച് ചർച്ച ചെയ്യുക.*

## 3.5 സോഫ്റ്റ്വെയർ (Software)

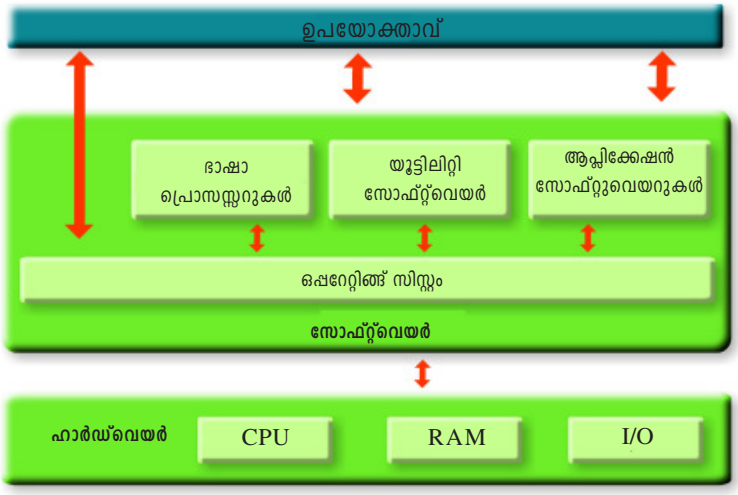
ഒരുകൂട്ടം പ്രോഗ്രാമുകൾ ഉപയോഗിച്ച് കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെയും മറ്റ് ഇലക്ട്രോണിക് ഉപകരണങ്ങളുടെയും പ്രവർത്തനം കാര്യക്ഷമവും ഫലപ്രദവുമായി നടത്താൻ സഹായിക്കുന്നവയാണ് സോഫറ്റ്വെയർ. ഹാർഡ്വെയർ കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ ശരീരം രൂപപ്പെടുത്തുമെന്ന് പറയാമെങ്കിൽ സോഫ്റ്റ്വെയർ അതിന്റെ മനസ്സോ ആത്മാവോ ആകുന്നു. രണ്ടു തരത്തിലുള്ള സോഫ്റ്റ്വെയറാണുള്ളത്.

- സിസ്റ്റം സോഫ്റ്റ്വെയർ
- ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയർ

### 3.5.1. സിസ്റ്റം സോഫ്റ്റ്വെയർ (System software)

കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തനങ്ങൾ നിയന്ത്രിക്കുന്നതിനായി രൂപകൽപ്പന ചെയ്തിട്ടുള്ള ഒരു കൂട്ടം പ്രോഗ്രാമുകളെയാണ് സിസ്റ്റം സോഫ്റ്റ്വെയർ എന്ന് പറയുന്നത്. കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തനങ്ങളെ നിയന്ത്രിച്ചുകൊണ്ടും, കമ്പ്യൂട്ടർ സംവിധാനത്തിന്റെ അകത്തേയ്ക്കും പുറത്തേയ്ക്കും ഡാറ്റാ എത്തിച്ചുകൊണ്ടും, ആപ്ലിക്കേഷൻ പ്രോഗ്രാമുകളുടെ കൃത്യനിർവഹണത്തിന്റെ എല്ലാ ഘട്ടങ്ങളും ചെയ്തുകൊണ്ടും കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ ഉപയോഗത്തിൽ മനുഷ്യനെ സഹായിക്കാൻ വേണ്ടി രൂപകൽപ്പന ചെയ്ത പൊതു പ്രോഗ്രാമുകളാണ് അവ. ചുരുക്കത്തിൽ സിസ്റ്റം സോഫ്റ്റ്വെയർ മറ്റ് സോഫ്റ്റ്വെയറുകളുടെ പ്രവർത്തനത്തെ പിന്തുണക്കുകയും പെരിഫറൽ ഉപകരണങ്ങളുമായി ആശയവിനിമയം നടത്തുകയും ചെയ്യുന്നു. കമ്പ്യൂട്ടർ ഫലപ്രദമായി ഉപയോഗിക്കുന്നതിന് ഉപയോക്താക്കളെ ഇത് സഹായിക്കുന്നു. കമ്പ്യൂട്ടറിലെ വിഭവങ്ങൾ നിയന്ത്രിക്കുവാൻ സിസ്റ്റം സോഫ്റ്റ്വെയർ സഹായിക്കുന്നു എന്ന് ഇത് സൂചിപ്പിക്കുന്നു.

സിസ്റ്റം സോഫ്റ്റ്വെയർ, ഉപയോക്താവിനെയും ഹാർഡ്വെയറിനെയും എങ്ങനെ ബന്ധിപ്പിച്ചിരിക്കുന്നു എന്ന് ചിത്രം 3.12 ൽ കാണിക്കുന്നു.




*ചിത്രം 3.12: സോഫ്റ്റ്വെയറിലൂടെ ഉപയോക്താവും ഹാർഡ്വെയറും തമ്മിലുള്ള സമ്പർക്കം.*

സിസ്റ്റം സോഫ്റ്റ്വെയറിന്റെ ഘടകങ്ങൾ താഴെ കൊടുക്കുന്നു.

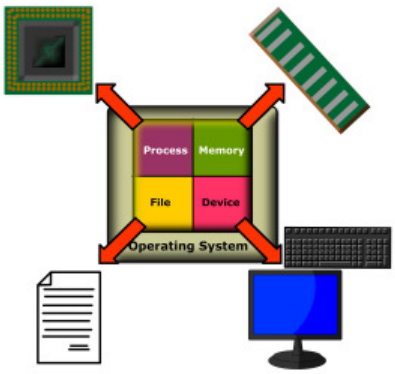
a) ഓപ്പറേറ്റിംഗ് സിസ്റ്റം

b) ഭാഷാ പ്രോസസ്സറുകൾ

c) യൂട്ടിലിറ്റി സോഫ്റ്റ്വെയർ

## a. ഓപ്പറേറ്റിംഗ് സിസ്റ്റം (Operating system)

ഉപയോക്താവിനേയും കമ്പ്യൂട്ടർ ഹാർഡ്വെയറിനെയും ബന്ധിപ്പിക്കാനായുള്ള ഒരുകൂട്ടം പ്രോഗ്രാമുകളെയാണ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റം എന്നുപറയുന്നത്. കമ്പ്യൂട്ടർ സിസ്റ്റത്തെ ഉപയോഗയോഗ്യമാക്കുക എന്നതാണ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന്റെ പ്രാഥമികലക്ഷ്യം. ഉപയോക്താവിന് പ്രോഗ്രാമുകൾ നടപ്പിലാക്കാൻ അനുയോജ്യമായ പരിസ്ഥിതി ഒരുക്കിക്കൊടുക്കുകയാണ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റം ചെയ്യുന്നത്. കാര്യക്ഷമമായ രീതിയിൽ കമ്പ്യൂട്ടർ ഹാർഡ്വെയർ ഉപയോഗിക്കാൻ സഹായിക്കുക എന്നതും ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന്റെ ജോലിയാണ്.




*ചിത്രം 3.13: ഓപ്പറേറ്റിങ് സിസ്റ്റം വിഭവങ്ങളുടെ മാനേജർ എന്ന നിലയിൽ*

കമ്പ്യൂട്ടറിലെ എല്ലാവിധ പ്രവർത്തനങ്ങളെയും നിയന്ത്രിക്കുകയും ഏകോപിപ്പിക്കുകയും ചെയ്യുന്നത് ഓപ്പറേറ്റിംഗ് സിസ്റ്റമാണ്. ചിത്രം 3.13 ൽ കമ്പ്യൂട്ടർ സിസ്റ്റത്തിലെ വിഭവങ്ങളുടെ മാനേജരായി (Resource Manager) ഓപ്പറേറ്റിംഗ് സിസ്റ്റം പ്രവർത്തിക്കുന്നത് കാണിക്കുന്നു. ഏറ്റവും പ്രധാനപ്പെട്ട സിസ്റ്റം സോഫ്റ്റ്വെയറാണ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റം. കമ്പ്യൂട്ടറിലെ ഹാർഡ് ഡിസ്കിൽ നിന്നും ആദ്യം

എടുക്കുന്ന പ്രോഗ്രാമും ഓഫാക്കുന്നതുവരെ മെമ്മറിയിൽ നിലനിൽക്കുന്ന പ്രോഗ്രാമുമാണിത്. കമ്പ്യൂട്ടറിന്റെ അനുചിതമായ ഉപയോഗവും തെറ്റുകൾ സംഭവിക്കുന്നത് തടയാനും ഇത് ശ്രമിക്കുന്നു.

ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന്റെ പ്രധാനപ്പെട്ട ധർമ്മങ്ങളാണ് പ്രോസസ്സ് കൈകാര്യംചെയ്യുക, മെമ്മറി കൈകാര്യം ചെയ്യുക, ഫയൽ കൈകാര്യചെയ്യുക, സെക്യൂരിറ്റി കൈകാര്യം ചെയ്യുക, നിർദ്ദേശങ്ങൾ വ്യാഖ്യാനിക്കുക തുടങ്ങിയവ.

**i പ്രോസസ്സ് കൈകാര്യം ചെയ്യുക**

പ്രോസസ്സുകളുടെ വിന്യാസവും തിരിച്ചെടുക്കലും, വിവിധ പ്രൊസസ്സുകൾക്ക് വിഭവങ്ങൾ നൽകുന്നതിനുള്ള പദ്ധതി തയാറാക്കൽ എന്നിവയാണ് ഓപ്പറേറ്റിംങ് സിസ്റ്റത്തിലെ പ്രോസസ്സ് കൈകാര്യം ചെയ്യുന്ന ഭാഗം ശ്രദ്ധപുലർത്തുന്നത്.

**ii മെമ്മറി കൈകാര്യം ചെയ്യുക**

പ്രാഥമിക മെമ്മറിയെ കൈകാര്യം ചെയ്യുകയോ നിയന്ത്രിക്കുകയോ ചെയ്യുന്നത് ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിലെ മെമ്മറി കൈകാര്യം ചെയ്യുന്ന ഭാഗമാണ്. ഓരോ മെമ്മറി സ്ഥാനത്തിന്റെയും അവസ്ഥ (ഒരു പ്രോസസ്സിന് നീക്കിവെച്ചിരിക്കുകയാണോ അതോ ഒഴിഞ്ഞു കിടക്കുകയാണോ) അത് സൂക്ഷിച്ചുവെക്കുന്നു. ഓരോ പ്രോസസ്സിനും എത്രമാത്രം മെമ്മറി നീക്കിവെക്കണമെന്ന് അത് കണക്കാക്കുകയും അതു പ്രകാരം നീക്കി വെക്കുകയും ചെയ്യുന്നു. മെമ്മറി പിന്നീട് ആവശ്യമില്ലെങ്കിൽ അത് തിരിച്ചെടുക്കുകയും ചെയ്യുന്നു.

**iii ഫയൽ കൈകാര്യം ചെയ്യുക**

ഫയലുകളുമായി ബന്ധപ്പെട്ട് കിടക്കുന്ന പ്രവർത്തനങ്ങളായ ആസൂത്രണം ചെയ്യുക, പേര് കൊടുക്കുക, സംഭരിക്കുക, തിരിച്ചെടുക്കുക, കൈമാറ്റം ചെയ്യുക, സംരക്ഷിക്കുക എന്നീ പ്രവർത്തനങ്ങൾ ചെയ്യുന്നു.

**iv ഡിവൈസ് കൈകാര്യം ചെയ്യുക**

കമ്പ്യൂട്ടറുമായി ബന്ധിപ്പിച്ചിരിക്കുന്ന ഉപകരണങ്ങളുടെ നിയന്ത്രണമാണ് ഡിവൈസ് കൈകാര്യം ചെയ്യൽ. ഹാർഡ്‌വെയറും സോഫ്റ്റ്‌വെയറും സംയോജിപ്പിച്ചു കൊണ്ട് അത് ഉപകരണങ്ങളെ കൈകാര്യം ചെയ്യുന്നു. ഡിവൈസ് ഡ്രൈവർ സോഫ്റ്റ്‌വെയറിലൂടെ ഓപ്പറേറ്റിംഗ് സിസ്റ്റം ഹാർഡ് വെയർ ഉപകരണങ്ങളുമായി സംവദിക്കുന്നു. ഡോസ്, വിൻഡോസ്, യുണിക്സ്, ലിനക്സ്, മാക് ഒ.എസ്. എന്നിവ വിവിധതരം ഓപ്പറേറ്റിംങ് സിസ്റ്റങ്ങൾക്കുള്ള ഉദാഹരണങ്ങളാണ്.

### b. ഭാഷ പ്രോസസ്സറുകൾ (Language Processors)

മനുഷ്യർ തമ്മിൽ ആശയവിനിമയം നടത്താൻ ഭാഷ ഉപയോഗിക്കുന്നു. കമ്പ്യൂട്ടറുമായി ആശയവിനിമയം നടത്താൻ കമ്പ്യൂട്ടറിന് അറിയാവുന്ന ഒരു ഭാഷ ഉപയോക്താവിന് ആവശ്യമായി വരുന്നു. കമ്പ്യൂട്ടർ ഭാഷകളെ ഉയർന്നതലത്തിലുള്ള കമ്പ്യൂട്ടർ ഭാഷ (HLL) എന്നും താഴ്ന്ന തലത്തിലുള്ള കമ്പ്യൂട്ടർ ഭാഷ (LLL) എന്നും രണ്ടായി തരംതിരിച്ചിരിക്കുന്നു.

താഴ്ന്നതലത്തിലുള്ള കമ്പ്യൂട്ടർ ഭാഷ യന്ത്രാധിഷ്ഠിതഭാഷ എന്നറിയപ്പെടുന്നു. കമ്പ്യൂട്ടറിൽ ലഭ്യമായ മെമ്മറിയും രജിസ്റ്ററുകളും ഉപയോഗിച്ച് ഈ ഭാഷയിലുള്ള പ്രോഗ്രാമുകൾ എഴുതപ്പെ

ടുന്നു. ഓരോ കമ്പ്യൂട്ടറിന്റെയും രൂപകൽപ്പന വ്യത്യസ്ഥമായതു കൊണ്ട് ഓരോ കമ്പ്യൂട്ടറിനും പ്രത്യേക താഴ്ന്നതലത്തിലുള്ള യന്ത്രഭാഷ ഉപയോഗിക്കുന്നു. മെഷീൻ ലാംഗ്വേജും അസംബ്ലി ലാംഗ്വേജും വിവിധ തരം താഴ്ന്നതലത്തിലുള്ള കമ്പ്യൂട്ടർ ഭാഷകളാണ്.

**യന്ത്ര ഭാഷ (Machine Language) :** പ്രത്യേകതരം സൂചനകൾ മാത്രമെ കമ്പ്യൂട്ടറിന് മനസ്സിലാക്കാൻ സാധിക്കുകയള്ളൂ എന്ന് നമുക്കറിയാമല്ലോ. 1, 0 എന്നീ ബൈനറി സംഖ്യകളിലൂടെ പ്രതിനിധീകരിക്കപ്പെടുന്ന പ്രത്യേക അടയാളങ്ങൾ മാത്രമേ കമ്പ്യൂട്ടറിന് മനസിലാകൂ എന്ന് നമുക്കറിയാം. ബൈനറി അക്കങ്ങൾ ഉപയോഗിക്കുന്ന ഭാഷകളെ യന്ത്ര ഭാഷ എന്നുവിളിക്കുന്നു. യന്ത ഭാഷയിൽ പ്രോഗ്രാം എഴുതുന്നത് വളരെ ബുദ്ധിമുട്ടാണ്. എല്ലാ നിർദേശങ്ങൾക്കും, 0 ന്റെയും 1 ന്റെയും ദൈർഘ്യമേറിയ സിട്രിങ്ങ് ഓർത്തിരിക്കാൻ സാധ്യമല്ല.

**അസംബ്ലി ഭാഷ (Assembly Language)**: അസംബ്ലി ഭാഷ മധ്യവർത്തിയായ (intermediate) പ്രോഗ്രാമിംഗ് ഭാഷയാണ്. അസംബ്ലി ഭാഷകൾ ന്യൂമോണിക്കുകൾ ഉപയോഗിക്കുന്നു. ഒരു പ്രവർത്തനത്തിന് കൊടുക്കുന്ന പ്രതീകാത്മകമായ പേരാണ് ന്യൂമോണിക്. ഉദാഹരണമായി സങ്കലനനത്തിന് ADD, വ്യവകലനത്തിന് SUB, തുടങ്ങിയവ. യന്ത്ര ഭാഷയെ അപേക്ഷിച്ച് അസംബ്ലി ഭാഷയിൽ കമ്പ്യൂട്ടർ പ്രോഗ്രാം എഴുതാൻ എളുപ്പമാണ്. ഇത് യന്ത്രാധിഷ്ഠിത ഭാഷയായതിനാൽ പ്രോഗ്രാമർക്ക് കമ്പ്യൂട്ടർ രൂപഘടനയെക്കുറിച്ച് അറിവുണ്ടായിരിക്കേണ്ടതാണ്.

**ഉയർന്നതലത്തിലുള്ള ഭാഷ (High Level Language)**: ഈ ഭാഷകൾ ഇംഗ്ലീഷ് ഭാഷയേപ്പോലെ ഉള്ളതും അസംബ്ലി ഭാഷയെക്കാളും യന്ത്ര ഭാഷയെക്കാളും ലളിതമായി മനസിലാക്കാൻ സാധിക്കുന്നതുമാണ്. ഉയർന്നതലത്തിലുള്ള കമ്പ്യൂട്ടർ ഭാഷകൾ (HLL) കമ്പ്യൂട്ടറിന് മനസ്സിലാക്കാൻ സാധിക്കുകയില്ല. ഉയർന്നതലത്തിലുള്ള ഭാഷകളിൽ എഴുതിയിരിക്കുന്ന കമ്പ്യൂട്ടർ പ്രോഗ്രാമിനെ അതിന് തുല്യമായ യന്ത്ര ഭാഷയിലേക്ക് മാറ്റുന്നു. ആയതിനാൽ ഇത്തരം ഭാഷകളെ തർജ്ജമ ചെയ്യുന്നതിന് ഒരു ഭാഷ വിവർത്തകനെ (കമ്പൈലറുകളോ ഇന്റർപ്രിറ്ററുകളോ) ആവശ്യമായി വരുന്നു. BASIC, C, C++, JAVA എന്നിവ ഉയർന്നതലത്തിലുള്ള പ്രോഗ്രാമിംഗ് ഭാഷകൾക്ക് ഉദാഹരണങ്ങളാണ്.

### ഭാഷ പ്രോസസ്സറിന്റെ ആവശ്യകത

ഉയർന്നതലത്തിലുള്ള ഭാഷകളിലോ അസംബ്ലി ഭാഷകളിലോ എഴുതിയിരിക്കുന്ന പ്രോഗ്രാമുകൾ കമ്പ്യൂട്ടറിന് മനസ്സിലാക്കാൻ സാധിക്കുകയില്ല. ഇങ്ങിനെയുള്ള പ്രോഗ്രാമുകളെ യന്ത്ര ഭാഷകളിലേക്ക് (കമ്പ്യൂട്ടറിന് മനസിലാകുന്ന ഭാഷ) മാറ്റുന്നതിന് ഭാഷ പ്രോസസ്സറുകൾ ഉപയോഗിക്കുന്നു. ഉയർന്നതലത്തിലുള്ള ഭാഷകളിലോ അസംബ്ലി ഭാഷകളിലോ എഴുതിയിരിക്കുന്ന പ്രോഗ്രാമുകളെ അതിന് സമാനമായ യന്ത്ര ഭാഷകളിലേക്ക് മാറ്റം ചെയ്യുന്ന സിസ്റ്റം പ്രോഗ്രാമുകളെയാണ് ഭാഷ പ്രൊസസ്സറുകൾ എന്നു വിളിക്കുന്നത്.

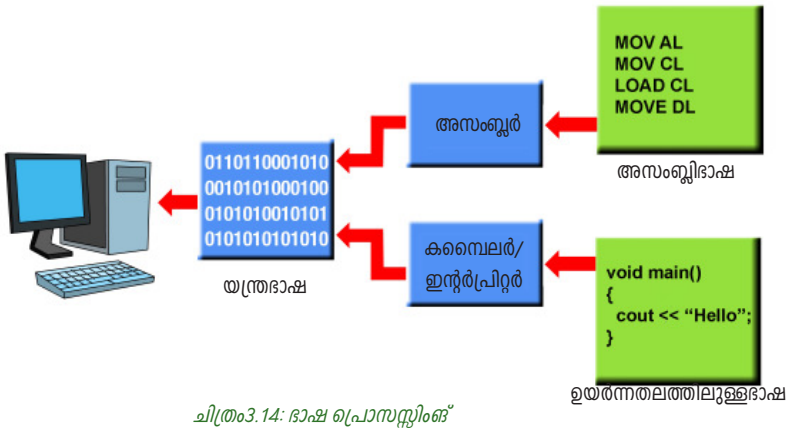
### വിവിധതരത്തിലുള്ള ഭാഷ പ്രോസസ്സറുകൾ (Types of Language Processors)

- **അസംബ്ലർ (Assembler):** അസംബ്ലി ഭാഷയിൽ എഴുതിയ പ്രോഗ്രാമുകളെ യന്ത്ര ഭാഷയിലേക്ക് തർജ്ജമ ചെയ്യുന്നതിന് അസംബ്ലർ എന്നറിയപ്പെടുന്ന ഒരു വിവർത്തകൻ ആവശ്യമുണ്ട്. പ്രോഗ്രാമിലെ പ്രവൃത്തികൾ നടപ്പിലാക്കുന്നത് തർജ്ജമ ചെയ്യപ്പെട്ട ശേഷമാണ്. കാരണം കമ്പ്യൂട്ടറിന് യന്ത്ര കോഡ് നിർദ്ദേശം മാത്രമെ മനസിലാക്കാൻ കഴിയുകയുള്ളൂ. അസംബ്ലർ യന്ത്രാധിഷ്ഠിതമാണ്.

- **ഇന്റർപ്രിറ്റർ (Interpreter)** : ഉയർന്നതലത്തിലെഴുതിയ പ്രോഗ്രാമുകളെ വരിവരിയായി യാന്ത്രിക ഭാഷയിലേക്ക് മൊഴിമാറ്റം നടത്തുന്ന ഭാഷ പ്രൊസസ്സറാണ് ഇന്റർപ്രിറ്റർ. ഏതെങ്കിലും ഒരു വരിയിൽ തെറ്റുണ്ടെങ്കിൽ, തെറ്റുകൾ വെളിപ്പെടുത്തുകയും പ്രവർത്തനം അവിടെവെച്ച് അവസാനിപ്പിക്കുകയും ചെയ്യുന്നു. തെറ്റു തിരുത്തിയതിനുശേഷം മാത്രമേ വിവർത്തനം തുടരുകയുള്ളൂ. BASIC ഒരു ഇന്റർപ്രിറ്റഡ് ഭാഷയാണ്.

- **കമ്പൈലർ (compiler):** ഉയർന്നതലത്തിലെഴുതിയ പ്രോഗ്രാമുകളെ യന്ത്ര ഭാഷയിലേക്ക് മൊഴിമാറ്റം നടത്തുന്ന ഭാഷ പ്രൊസസ്സറാണ് കമ്പൈലർ. ഒറ്റത്തവണ കൊണ്ടുതന്നെ അത് പ്രോഗ്രാം മുഴുവനായും വ്യാഖ്യാനിക്കുന്നു. ഇതിൽ എന്തെങ്കിലും തെറ്റുകൾ വന്നാൽ ആ തെറ്റുകൾ കമ്പൈലേഷന്റെ അവസാനം ക്രമനമ്പറോടു കൂടി സന്ദേശങ്ങളായി സ്ക്രീനിൽ തെളിയും. വാക്യഘടനയിൽ തെറ്റൊന്നുമില്ലെങ്കിൽ കമ്പൈലർ ഒരു ഒബ്ജക്റ്റ് ഫയൽ സൃഷ്ടിക്കും. കമ്പൈലർ ഉപയോഗിച്ചുള്ള തർജ്ജമയെ കമ്പൈലേഷൻ എന്നു പറയുന്നു. തർജ്ജമക്കുശേഷം പ്രോഗ്രാം റൺ ചെയ്യുന്നതിന് കമ്പൈലർ മെമ്മറിയിൽ ആവശ്യമില്ല. കമ്പൈലർ ഉപയോഗിക്കുന്ന പ്രോഗ്രാമിംഗ് ഭാഷകളാണ് C, C++, Pascal തുടങ്ങിയവ.

അസംബ്ലി ഭാഷയിൽ നിന്നും, ഉയർന്നതലത്തിലുള്ള ഭാഷയിൽ നിന്നും പ്രോഗ്രാമുകൾ യന്ത്ര ഭാഷ പ്രോഗ്രാമുകളായി വിവർത്തനം ചെയ്യുന്ന പ്രവൃത്തികൾ ചിത്രം 3.14 ൽ കാണിച്ചിരിക്കുന്നു.




*ചിത്രം3.14: ഭാഷ പ്രൊസസ്സിംങ്*

## c. യൂട്ടിലിറ്റി സോഫ്റ്റ്വെയർ (Utility Software)

പതിവ് ജോലികളും സിസ്റ്റം പരിപാലന ജോലികളും നിർവ്വഹിക്കാൻ ഉപയോക്താക്കളെ സഹായിക്കുന്ന ഒരുകൂട്ടം പ്രോഗ്രാമുകളാണ് യൂട്ടിലിറ്റി സോഫ്ട്വെയർ. ചില യൂട്ടിലിറ്റി പ്രോഗ്രാമുകളും അവയുടെ ധർമ്മങ്ങളും താഴെ കൊടുത്തിരിക്കുന്നു

- **കംപ്രഷൻ ടൂൾസ് (compression tools):** വലിയ ഫയലുകളെ ചുരുങ്ങിയ സംഭരണ സ്ഥലങ്ങളിലേക്ക് ചുരുക്കുക എന്നതാണ് കംപ്രഷൻ ടൂൾസിലൂടെ ചെയ്യുന്നത്. ആവശ്യാനുസരണം ഒരിക്കൽ കംപ്രഷൻ യൂട്ടിലിറ്റി ഉപയോഗിച്ച് വലിപ്പം കുറച്ച ഫയലുകളെ ഡികമ്പ്രസ് ചെയ്ത് യഥാർത്ഥ വലിപ്പത്തിലേക്ക് മാറ്റുവാൻ സാധിക്കും. ഫയലുകളുടെ കമ്പ്രഷനെ സിപ്പിംഗ് (Zipping) എന്നും ഡികമ്പ്രഷനെ അൺസിപ്പിംങ്ങ് (Unzipping) എന്നും വിളിക്കുന്നു. ഉദാഹരണം Winzip, WinRAR എന്നിവ.

- **ഡിസ്ക് ഡീഫ്രാഗ്മെന്റർ** കമ്പ്യൂട്ടർ ഹാർഡ് ഡിസ്കിലെ ഫയലുകളെ പുനക്രമീകരണം നടത്തുന്ന പ്രോഗ്രാമുകളെ ഡിസ്ക് ഡീഫ്രാഗ്മെന്റർ എന്നു പറയുന്നു. ഡിസ്കിന്റെ പല ഭാഗങ്ങളിലായി ചിതറികിടക്കുന്ന ഫയലുകളെ ക്രമീകരിച്ചാൽ കമ്പ്യൂട്ടറിന് വേഗത കൂടുകയും പ്രവർത്തനം കാര്യക്ഷമമാവുകയും ചെയ്യും.

- **ബാക്ക് അപ്പ് സോഫ്റ്റ്വെയർ (Backup Software)** : ഏതെങ്കിലും കാരണത്താൽ ഹാർഡ് ഡിസ്ക്കുകൾ പ്രവർത്തനരഹിതമാകുകയോ അബദ്ധത്തിൽ മാറ്റങ്ങൾക്ക് വിധേയമാകുകയോ ചെയ്താൽ ഡിസ്കിൽ സൂക്ഷിച്ചു വച്ചിരിക്കുന്ന വിവരങ്ങളുടെ പകർപ്പ് നമുക്ക് എടുക്കുവാൻ സാധിക്കുന്ന സോഫ്റ്റ്വെയറാണിത്. ഈ സൗകര്യം ഉപയോഗിച്ച് ഫയലുകളോ ഫോൾഡറുകളോ ഡ്രൈവുകളോ നമുക്ക് ബാക്ക് അപ്പിന് വിധേയമാക്കാം.

- **ആന്റി വൈറസ് സോഫ്ട്വെയർ:** കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തനത്തെ ദോഷകരമായി ബാധിക്കുന്ന പ്രോഗ്രാമുകളാണ് കമ്പ്യൂട്ടർ വൈറസുകൾ. ആന്റി വൈറസ് സോഫ്റ്റ്വെയർ എന്ന യൂട്ടിലിറ്റി പ്രോഗ്രാം ഉപയോഗിച്ച് കമ്പ്യൂട്ടറിനെ ബാധിച്ചിരിക്കുന്ന വൈറസിനെ കണ്ടെത്താനും അവയെ ഒഴിവാക്കാനും സാധിക്കും. പുതിയ വൈറസുകൾ സൃഷ്ടിക്കപ്പെട്ടുകൊണ്ടിരിക്കുന്നതിനനുസരിച്ച് ആന്റി വൈറസ് സോഫ്റ്റ്വെയറുകളും പുതുക്കേണ്ടതുണ്ട്. എല്ലാ ആന്റി വൈറസ് സോഫ്റ്റ്വെയറുകളിലും സ്വയം പുതുക്കുന്ന സംവിധാനങ്ങളുണ്ട്. നോർട്ടൺ ആന്റിവൈറസ്, കാസ്പേഴ്സ്കി, എ.വി.ജി എന്നിവ ഉദാഹരണങ്ങളാണ്.

### 3.5.2 ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയർ (Application Software)

ഒരു പ്രത്യേക ആവശ്യത്തിനായി വികസിപ്പിച്ചെടുക്കുന്ന സോഫ്റ്റ്വെയറുകളെയാണ് ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയർ എന്നു വിളിക്കുന്നത്. പൊതുവായ ആവശ്യങ്ങൾക്കായുള്ള സോഫ്റ്റ്വെയർ പാക്കേജുകളും പ്രത്യേക ആവശ്യങ്ങൾക്കായുള്ള സോഫ്റ്റ്വെയറുകളും ഇതിൽപ്പെടുന്നു. ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയറുകൾക്ക് ഉദാഹരണങ്ങളാണ് GIMP, Payroll system, Airline Reservation System, Tally എന്നിവ.

#### a. പൊതുവായ ആവശ്യങ്ങൾക്കുള്ള പാക്കേജുകൾ (General Purpose Software Packages)

ഒരു പ്രത്യേക ആപ്ലിക്കേഷനിലെ പ്രവർത്തനങ്ങൾ നിർവ്വഹിക്കാൻ ഉപയോഗിക്കുന്ന സോഫ്റ്റ്വെയർ പാക്കേജാണിത്. ഉപയോക്താവിന്റെ ആവശ്യങ്ങൾക്കനുസരിച്ച് ധാരാളം സവിശേഷതകൾ പ്രത്യേക പാക്കേജുകളിൽ ഉൾപ്പെടുത്തിയിരിക്കുന്നു. വേഡ് പ്രോസസറുകൾ, സ്പ്രെഡ്ഷീറ്റ് സോഫ്റ്റ്വെയർ, പ്രസന്റേഷൻ സോഫ്റ്റ്വെയർ, ഡാറ്റാബേസ് സോഫ്റ്റ്വെയർ, മൾട്ടിമീഡിയ സോഫ്റ്റ്വെയർ എന്നിങ്ങനെ ഇവയെ തരംതിരിച്ചിരിക്കുന്നു.

- **വേഡ് പ്രൊസസ്സിംങ്ങ് സോഫ്റ്റ്വെയർ (Word Processing Software):** ഡോക്യുമെന്റുകൾ നിർമ്മിക്കുന്നതിനും മാറ്റങ്ങൾ വരുത്തുന്നതിനും വേണ്ടി രൂപകൽപ്പന ചെയ്തിരിക്കുന്ന സോഫ്റ്റ്വെയർ ആണ് വേഡ്പ്രൊസസ്സിംഗ്. വളരെ എളുപ്പത്തിൽ ലിഖിത ഉള്ളടക്കം രൂപകൽപ്പന ചെയ്യുവാനും ചിട്ടപ്പെടുത്തുവാനും നിർമിക്കുവാനും പ്രിന്റ് ചെയ്യുവാനും ഇത് ഉപയോഗിക്കുന്നു, ഇതിന്റെ സഹായത്തോടെ ഫോണ്ടുകൾ സെറ്റ് ചെയ്യുവാനും പാരഗ്രാഫ് സെറ്റ് ചെയ്യുവാനും വ്യത്യസ്ഥ രീതിയിൽ അടയാളങ്ങൾ കൊടുക്കുവാനും വരിയായിനിരത്തൽ ശരിയാക്കുവാനും വ്യാകരണവും അക്ഷരവിന്യാസവും പരിശോധിക്കുവാനും ചിത്രങ്ങൾ ഉൾപ്പെടുത്തുവാനും രേഖാചിത്രങ്ങളും, പട്ടികകളും നിർമ്മിക്കുവാനും സാധിക്കുന്നു. ഡോക്യുമെന്റിന്റെ ഓരോ പേജുകളിലും തലക്കെട്ടും അടിക്കുറിപ്പും സജ്ജീകരിക്കാനും സാധിക്കുന്നു. MS Word, Open Office Writer, Apple i Work Pages എന്നിവ ഉദാഹരണങ്ങളാണ്.

- **സ്പ്രെഡ്ഷീറ്റ് സോഫ്റ്റ്വെയർ (Spreadsheet Software):** പട്ടിക രൂപത്തിലുള്ള ഈ സോഫ്റ്റ്വെയർ ഉപയോഗിച്ച് കണക്കു കൂട്ടലുകൾ എളുപ്പത്തിൽ നടത്താൻ സാധിക്കും. പേപ്പർ വർക്ക്ഷീറ്റിനെ അനുകരിച്ചുകൊണ്ട് സെല്ലുകൾ കൊണ്ട് ഒരു ഗ്രിഡ് നിർമ്മിക്കുന്നു. ചിത്രങ്ങൾ ഉൾപ്പെടുത്തുവാനും വിവിധതരത്തിലുള്ള ചാർട്ടുകൾ നിർമ്മിക്കുവാനും അത് അനുവദിക്കുന്നു. എംഎസ്എക്സൽ, ഓപ്പൺ ഓഫീസ് കാൽക്, ലോട്ടസ് 1-2-3, ആപ്പിൾ i Work നമ്പേഴ്സ്. (MS Excel, Open Office Calc, Lotus 1-2-3, Apple i Work numbers) എന്നിവ ഉദാഹരണങ്ങളാണ്.

- **പ്രസന്റേഷൻ സോഫ്റ്റ്വെയർ (Presentation Software):** സ്ലൈഡ് ഷോവിൽ ചലിക്കുന്ന ചിത്രങ്ങളും ശബ്ദങ്ങളും ഉപയോഗിച്ച് വിവരങ്ങൾ തയ്യാറാക്കാൻ ഉപയോഗിക്കുന്ന സോഫ്റ്റ് വെയറാണിത്. ചിത്രങ്ങളും ടെക്സ്റ്റുകളും ആനിമേഷനും വീഡിയോകളും ശബ്ദങ്ങളും ഉൾപ്പെടുത്തി വ്യത്യസ്ത തരത്തിലുള്ള ആശയങ്ങൾ സ്ലൈഡുകളിലൂടെ നിർമ്മിക്കാൻ പ്രസന്റേഷൻ സോഫ്റ്റ്വെയർ സഹായിക്കുന്നു. മൈക്രോസോഫ്റ്റ് പവർപോയിന്റ്, ഓപ്പൺ ഓഫീസ് ഇംപ്രസ്, ആപ്പിൾ ഐവർക്ക് കീനോട്ട് (Apple i Work Keynote) എന്നിവ ഉദാഹരണങ്ങളാണ്.

- **ഡാറ്റാബേസ് സോഫ്റ്റ്വെയർ (Database Software):** പരസ്പരം ബന്ധപ്പെട്ടു കിടക്കുന്ന ഒരുകൂട്ടം ഡാറ്റകളെ ശേഖരിച്ച് പട്ടികാ രീതിയിൽ സൂക്ഷിച്ചിരിക്കുന്നവയെയാണ് ഡാറ്റാ ബേസുകൾ എന്നുപറയുന്നത്. പരസ്പരം ബന്ധപ്പെട്ടു കിടക്കുന്ന ഡാറ്റയും അവ സ്വീകരിക്കാനുള്ള ഒരുകൂട്ടം പ്രോഗ്രാമുകളും ചേർന്നതാണ് ഡാറ്റാ ബേസ് മാനേജ്മെന്റ് സിസ്റ്റം (DBMS), ഡാറ്റാബേസിലുള്ള വിവരങ്ങൾ അതിനനുയോജ്യമായ രീതിയിലും കാര്യക്ഷമമായും വിനിയോഗിച്ച് അവ വേണ്ട രീതിയിൽ സൂക്ഷിക്കുകയും തിരിച്ചെടുക്കുകയും ചെയ്യലാണ് ഡാറ്റാ ബേസിന്റെ പ്രധാന ലക്ഷ്യം. സുരക്ഷിതത്വവും സ്വകാര്യതയും, പ്രത്യേകമാനദണ്ഡങ്ങളും ഇവ നൽകുന്ന പ്രത്യേകതകളാണ്. മൈക്രോസോഫ്റ്റ് ആക്സസ് (Microsoft access), ഒറാക്കിൾ (Oracle), പോസ്റ്റ്ഗ്രസ് എസ്.ക്യൂ.എൽ (Postgres SQL), മൈ എസ്.ക്യൂ.എൽ (My SQL) എന്നിവ ഉദാഹരണങ്ങളാണ്.

- **മൾട്ടിമീഡിയ സോഫ്റ്റ്വെയർ (Multimedia Software):** വിവിധ രൂപങ്ങളിലുള്ള മാധ്യമങ്ങളുടെ ഏകീകൃത രൂപമാണ് മൾട്ടിമീഡിയ. അക്ഷരങ്ങളും ചിത്രങ്ങളും ഗ്രാഫിക്സുകളും ഓഡിയോകളും വീഡിയോകളും കൂടി ചേർന്നുള്ള രൂപമാണ് മൾട്ടിമീഡിയ. വിവിധതരത്തിലുള്ള വിവരങ്ങൾ പ്രൊസസ്സ് ചെയ്യാൻ മൾട്ടിമീഡിയ സോഫ്റ്റ്വെയറിൽ സാധിക്കും. ഓഡിയോ, വീഡിയോ ഫയലുകൾ നിർമ്മിക്കുന്നതിനും എഡിറ്റ് ചെയ്യുന്നതിനും ഇത് സഹായിക്കുന്നു. ഒരു രൂപത്തിൽ നിന്നും വേറൊരു രൂപത്തിലേക്ക് തർജ്ജമ ചെയ്യുവാനുള്ള (ഓഡിയോ വീഡിയോ ഫയലുകൾ) സോഫ്റ്റ്വെയറുകൾ ഉണ്ട്. വി.എൽ.സി. പ്ലയർ (VLC Player), അഡോബ് ഫ്ളാഷ് (Adobe Flash), റിയൽ പ്ലയർ (Real Player), മീഡിയ പ്ലയർ (Media Player) എന്നിവ ഉദാഹരണങ്ങളാണ്.

### b. പ്രത്യേക ആവശ്യങ്ങൾക്കുള്ള സോഫ്റ്റ്വെയർ (Specific Purpose Software)

പ്രത്യേക ആവശ്യങ്ങൾക്കു മാത്രമായി തയ്യാറാക്കുന്ന സോഫ്റ്റ്വെയറുകളാണിത്. ഒരു സ്ഥാപനത്തിനുവേണ്ടി പ്രത്യേകം തയ്യാറാക്കിയിരിക്കുന്ന ഈ സോഫ്റ്റ്വെയറിനെ ടെയ്ലർ - മെയ്ഡ് സോഫ്റ്റ്വെയർ എന്നു വിളിക്കുന്നു. സാമ്പ്രദായികമായി ചിട്ടപ്പെടുത്തിയ സോഫ്റ്റ്വെയർ എന്നും ഇതിനെ പറയാറുണ്ട്. ഒരു ഉപയോക്താവിനെ മാത്രം ഉദ്ദേശിച്ചു കൊണ്ട് അയാളുടെ മുൻഗണന

കളും പ്രതീക്ഷകളും മാത്രം ഉൾപ്പെടുത്തികൊണ്ട് തയാറാക്കുന്ന സോഫ്റ്റ്വെയറാണ് സാമ്പ്രദായികമായി ചിട്ടപ്പെടുത്തിയ സോഫ്റ്റ്വെയർ. പട്ടിക 3.7 ൽ പ്രത്യേക ആവശ്യങ്ങൾക്കായുള്ള ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയർ വിശദീകരിച്ചിരിക്കുന്നു.

| അപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയർ | ഉദ്ദേശ്യങ്ങൾ |
|---|---|
| പേ റോൾ സിസ്റ്റം | ഒരു സ്ഥാപനത്തിലെ തൊഴിലാളികളുടെ വേതനത്തെക്കുറിച്ചും മറ്റ് എല്ലാവിധ വിവരങ്ങളെക്കുറിച്ചും വിശദമായി പരിപാലിക്കുന്നത് പേ റോൾ സിസ്റ്റമാണ്. |
| ഇൻവെന്ററി മാനേജ്മെന്റ് സിസ്റ്റം | ഒരു വ്യാപാര സ്ഥാപനത്തിലെ ആസ്തി വിവര പട്ടികകൾ, ഓർഡറുകൾ, വിപണനം, വിതരണം എന്നിവയെ വേണ്ടവിധം പരിപാലിക്കുന്നു. |
| ഹ്യൂമൺ റിസോഴ്സ് മാനേജ്മെന്റ് സിസ്റ്റം | ഒരു സ്ഥാപനത്തിലെ മനുഷ്യവിഭവങ്ങളെ വേണ്ട വിധം പരിപാലിക്കുന്നു. |

*പട്ടിക. 3.7: ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയറിന് ഉദാഹരണം*

ചെയ്തു നോക്കൂ

- *സോഫ്റ്റ്വെയറിന്റെ വർഗ്ഗീകരണം ചർച്ച ചെയ്യുക*
- *നിങ്ങളുടെ അധ്യാപകന്റെ സഹായത്തോടെ ലിനക്സും വിൻഡോസും തമ്മിലുള്ള സവിശേഷതകൾ താരതമ്യം ചെയ്ത് ഹ്രസ്വ കുറിപ്പുകൾ തയാറാക്കുക. യൂട്ടിലിറ്റി സോഫ്റ്റ്വെയറിന്റെ പ്രാധാന്യം ചർച്ച ചെയ്യുക.*
- *താഴെ കൊടുത്തിരിക്കുന്നവയുടെ ഹ്രസ്വ കുറിപ്പുകൾ തയാറാക്കുക.*
  - *ഭാഷാ പ്രോസസ്സുകൾ*
  - *പൊതുവായ ആവശ്യങ്ങൾക്കായുള്ള സോഫ്റ്റ്വെയർ പാക്കേജുകൾ.*

## സ്വയം പരിശോധിക്കുക

1. ഓപ്പറേറ്റിങ് സിസ്റ്റം നിർവചിക്കുക
2. ഓപ്പറേറ്റിങ് സിസ്റ്റത്തിന് 2 ഉദാഹരണം കൊടുക്കുക.
3. ഒരു പ്രോഗ്രാം കൃത്യനിർവ്വഹണത്തിന് പറയുന്ന പേരാണ്_____
4. ഓപ്പറേറ്റിങ് സിസ്റ്റത്തിന്റെ ഏതെങ്കിലും രണ്ട് ധർമ്മങ്ങൾ എഴുതുക.
5. അസംബ്ലി ഭാഷയെ യന്ത്ര ഭാഷയിലേക്ക് തർജ്ജമ ചെയ്യുന്ന സോഫ്റ്റ്വെയറിന്റെ പേരെഴുതുക
6. ഉയർന്ന തലത്തിലുള്ള ഭാഷകളെ യന്ത്രഭാഷകളാക്കി മാറ്റുന്ന 2 വ്യത്യസ്ഥ ഭാഷ പ്രൊസസ്സറുകളുടെ പേര് എഴുതുക.
7. കമ്പയിലറും ഇന്റർപ്രിറ്ററും തമ്മിലുള്ള വ്യത്യാസം എഴുതുക.
8. DBMS എന്നാൽ _____.
9. സാമ്പ്രദായികമായി ചിട്ടപ്പെടുത്തിയ സോഫ്റ്റ്വെയറിന് രണ്ട് ഉദാഹരണം എഴുതുക.
10. ഡിസ്കിലെ വിവരങ്ങളുടെ തനിപ്പകർപ്പിനെ വിളിക്കുന്ന പേര് _____.

### 3.5.3 സ്വതന്ത്ര ഓഷൺ സോഴ്സ് സോഫ്റ്റ്വെയർ (Free and open source software)

ഉപയോഗിക്കുന്നതിനും പകർപ്പ് എടുക്കുന്നതിനും വിതരണം ചെയ്യുന്നതിനും പരിശോധിക്കുന്നതിനും മാറ്റങ്ങൾ വരുത്തുന്നതിനും മെച്ചപ്പെടുത്തുന്നതിനും ഉപയോക്താവിന് സ്വാതന്ത്ര്യം നൽകുന്ന സോഫ്റ്റ്വെയറാണ് സ്വതന്ത്ര ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്വെയർ. കുറഞ്ഞ ചെലവ്, സുരക്ഷിതത്വം കുത്തക കമ്പനികളിൽ നിന്നുള്ള സ്വാതന്ത്ര്യം, കാര്യക്ഷമമായ പ്രവർത്തനം, പരസ്പര പ്രവർത്തനക്ഷമത തുടങ്ങിയവ നൽകുന്നത് കൊണ്ട് സ്വതന്ത്ര ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്വെയർ ഇപ്പോൾ വ്യാപകമായി ഉപയോഗിക്കുന്നു.

നാല് തരത്തിലുള്ള സ്വാതന്ത്ര്യം സ്വതന്ത്ര സോഫ്റ്റ്വെയർ ഫൗണ്ടേഷൻ (Free Software Foundation - FSF) നിർവചിക്കുന്നു.

*സ്വാതന്ത്ര്യം 0 (Freedom 0)* : ഏത് ആവശ്യത്തിനും ഇഷ്ടപ്രകാരം ഉപോയഗിക്കുന്നതിനുള്ള സ്വാതന്ത്ര്യം.

*സ്വാതന്ത്ര്യം 1 (Freedom 1)* : സോഫ്റ്റ്വെയർ എങ്ങനെ പ്രവർത്തിക്കുന്നു എന്ന് വിശകലനം ചെയ്യുന്നതിനുള്ള സ്വാതന്ത്ര്യം.

*സ്വാതന്ത്ര്യം 2 (Freedom 2)* : പ്രോഗ്രാമിന്റെ പകർപ്പുകൾ പുനർവിതരണം ചെയ്യുവാനുള്ള സ്വാതന്ത്ര്യം.

*സ്വാതന്ത്ര്യം 3 (Freedom 3)* : പ്രോഗ്രാമിനെ നവീകരിക്കുവാനും നവീകരിച്ചവയെ പുറത്തിറക്കുവാനുമുള്ള സ്വാതന്ത്ര്യം.

ഫ്രീ ആന്റ് ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്വെയറുകളിൽ ചിലത് താഴെ കൊടുക്കുന്നു

**ഗ്നു/ലിനക്സ് (GNU/Linux):** സ്വതന്ത്ര ഓപ്പൺ സോഫ്റ്റ്വെയർ മാതൃകയിലുള്ള ഓപ്പറേറ്റിംഗ് സിസ്റ്റം വിഭാഗത്തിൽപ്പെട്ട സോഫ്റ്റ്വെയറാണ് GNU/Linux. 1983 ൽ FSF ൽ റിച്ചാർഡ് സ്റ്റാൾമാൻ അവതരിപ്പിച്ച GNU പ്രൊജക്ടിൽ ചിട്ടപ്പെടുത്തിയ ഒന്നാണ് ഇത്.

**ജിമ്പ് (GIMP)**: GNU ഇമേജ് മാനിപുലേഷൻ പ്രോഗ്രാം എന്നാണ് ഇത് അറിയപ്പെടുന്നത്. ചിത്രങ്ങളെ ചിട്ടപ്പെടുത്തുവാൻ ഉപയോഗിക്കുന്ന സോഫ്റ്റ്വെയറാണിത്. ചിത്രങ്ങൾ നിർമ്മിക്കുവാനും, ചിട്ടപ്പെടുത്തുവാനും വേണ്ട രീതിയിൽ കൈകാര്യം ചെയ്യുവാനും ഈ സോഫ്റ്റ്വെയർ ഉപയോഗിക്കുന്നു. വിവിധ ഫയൽ ഫോർമാറ്റുകളെ പിന്തുണക്കാനും ഒരു രൂപത്തിൽ നിന്ന് മറ്റൊന്നിലേക്ക് മാറ്റുന്നതിനും GIMP സഹായിക്കുന്നു.

**മോസില്ല ഫയർഫോക്സ് (Mozilla Firefox)** : മോസില്ല കോർപ്പറേഷൻ നിർമ്മിച്ച വളരെ പ്രശസ്തമായ വെബ് ബ്രൗസർ ആണിത്. സുരക്ഷിതമായ ബ്രൗസിങ്ങിന് ഇത് അനുയോജ്യമാണ്.

**ഓപ്പൺ ഓഫീസ് .ഒആർജി (Open offfice .org)** : ഒരു സമ്പൂർണ്ണ ഓഫീസ് പാക്കേജ് ആണ് ഓപ്പൺ ഓഫീസ്. ഇതിൽ ലിഖിത ഉള്ളടക്കങ്ങൾ തയ്യാറാക്കാനും രൂപമാറ്റം ചെയ്യുവാനും വേഡ് പ്രൊസസ്സർ ആയ റൈറ്റർ, സ്പ്രെഡ് ഷീറ്റ് ആയ കാൽക്ക്, പ്രസന്റേഷൻ സോഫ്ടവെയർ ആയ ഇംപ്രസ്സ് എന്നിവ ഉപയോഗിക്കുന്നു. ഇത് ലിനക്സ്, വിൻഡോസ് സംവിധാനങ്ങളിൽ പ്രവർത്തിക്കുന്നു.

### 3.5.4 ഫ്രീവെയറും ഷെയർവെയറും (Freeware and shareware)

പ്രത്യേകമായ വിലയൊന്നും കൂടാതെ, പരിധിയൊന്നുമില്ലാതെ ഉപയോഗിക്കാൻ സാധിക്കുന്ന പകർപ്പവകാശം ഉള്ള കമ്പ്യൂട്ടർ സോഫ്റ്റ്വെയർ ആണ് ഫ്രീവെയർ.

ഒരു ചുരുങ്ങിയ കാലയളവിലേക്ക് പരീക്ഷണ അടിസ്ഥാനത്തിൽ വിതരണം ചെയ്യപ്പെടുന്ന വാണിജ്യ പരമായ സോഫ്റ്റ്വെയർ ആണ് ഷെയർവെയർ. വിലയൊന്നും കൂടാതെ വളരെ പരിമിതമായ പ്രവർത്തനക്ഷമതയോടെ വിതരണം ചെയ്യപ്പെടുന്നതാണിത്. ഇന്റർനെറ്റിൽ നിന്ന് ഡൗൺലോഡ് ചെയ്യാൻ സാധിക്കുന്ന വിധത്തിലാണ് ഷെയർവെയറുകൾ ലഭ്യമാകുന്നത്. വില കൊടുത്ത് വാങ്ങുന്നതിന് മുമ്പ് ഉപയോക്താക്കൾക്ക് ഇവയെ വിലയിരുത്തുവാനുള്ള അവസരം നൽകുന്നു എന്നതാണ് ഇതിന്റെ ലക്ഷ്യം. ചില ഷെയർവെയറുകൾ പരിമിതമായ കാലയളവിലേക്ക് മാത്രം പ്രവർത്തിപ്പിക്കുന്നവയാണ്. പട്ടിക 3.8ൽ ഫ്രീവെയറിന്റെയും ഷെയർവെയറിന്റെയും താരതമ്യം കൊടുത്തിരിക്കുന്നു

| ഫ്രീവെയർ | ഷെയർവെയർ |
|---|---|
| • ഇന്റർനെറ്റിൽ നിന്നും ആർക്കു വേണമെങ്കിലും സൗജന്യമായി ഡൗൺലോഡ് ചെയ്യുകയും ഉപയോഗിക്കുകയും ചെയ്യാം. | • വാങ്ങുന്നതിനും മുമ്പ് തന്നെ സോഫ്റ്റ്വെയറിനെ പരിചയപ്പെടാൻ സാധിക്കുന്നു. |
| • എല്ലാ സവിശേഷതകളും സൗജന്യമാണ്. | • എല്ലാ സവിശേഷതകളും ലഭ്യമാകുകയില്ല. എല്ലാ സവിശേഷതകളും ലഭിക്കണമെങ്കിൽ വില കൊടുത്തു വാങ്ങിക്കണം. |
| • ഫ്രീവെയർ പ്രോഗ്രാമുകൾ വിലയില്ലാതെയാണ് വിതരണം ചെയ്യുന്നത്. | • ഷെയർവെയർ വില കൊടുത്തും അല്ലാതെയും വിതരണം ചെയ്യുന്നു. പല സന്ദർഭങ്ങളിലും നിർമിച്ചയാളുടെ അനുമതിയോടെ മാത്രമേ ഷെയർവെയർ വിതരണം ചെയ്യുകയുള്ളു. |

*പട്ടിക 3.8 : ഫ്രീവെയറും ഷെയർവെയറും തമ്മിലുള്ള താരതമ്യം*

### 3.5.5 ഉടമസ്ഥാവകാശമുള്ള സോഫ്റ്റ്വെയർ (Proprietary Software)

സോഫ്റ്റ്വെയർ നിർമ്മാതാവ് അഥവാ പ്രസാധകന്റെ പൂർണ്ണ അധികാരപരിധിയിൽ വരുന്ന സോഫ്റ്റ്വെയറാണിത്. അനുവാദ ഉടമ്പടി ഇല്ലാതെ ഇത് പകർത്താനോ വിതരണം ചെയ്യുവാനോ പാടുള്ളതല്ല. പ്രോഗ്രാമിന്റെ സോഴ്സ് കോഡ് ലഭ്യമാകാത്തതിനാൽ ഇതിൽ മാറ്റം വരുത്തി മെച്ചപ്പെടുത്തുവാനൊന്നും ഉപയോക്താവിന് സാധിക്കുകയില്ല. മൈക്രോസോഫ്റ്റ് വിൻഡോസ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റം, എംഎസ് ഓഫീസ്, മാക് ഒഎസ് എന്നിവ ഉദാഹരണങ്ങളാണ്.

## 3.6 ഹ്യൂമൺവെയർ/ലൈവ് വെയർ

കമ്പ്യൂട്ടർ ഉപയോഗിക്കുന്ന ആളുകളെയാണ് ഹ്യൂമൺവെയർ/ലൈവ്വെയർ എന്നുപറയുന്നത്. പ്രോഗ്രാമർ, സിസ്റ്റം അനലിസ്റ്റ്, ഓപ്പറേറ്റിംങ്ങ് സ്റ്റാഫ്, കമ്പ്യൂട്ടർ സിസ്റ്റം ഉപയോഗിക്കുന്നവർ ഇവരെല്ലാം ഇതിൽ ഉൾപ്പെടുന്നു.

പട്ടിക 3.9ൽ വിവിധതരം ഹ്യൂമൺവെയറുകളും അവരുടെ ജോലികളും വിശദീകരിക്കുന്നു.

| ഹ്യൂമൺവെയർ | ജോലികൾ |
|---|---|
| സിസ്റ്റം അഡ്മിനിസ്ട്രേറ്റർ | കമ്പ്യൂട്ടർ സിസ്റ്റത്തെയും, സെർവറിനെയും പരിപാലിക്കുക. ക്രമീകരണം നടത്തുക, വിശ്വസനീയമായ പ്രവർത്തനങ്ങൾ നിർവ്വഹിക്കുക. പ്രത്യേകിച്ച് സെർവറുകളെയും ഒന്നിൽ കൂടുതൽ ഉപയോക്താക്കളുള്ള കമ്പ്യൂട്ടറുകളെയും പരിപാലിക്കുക. |
| സിസ്റ്റം മാനേജർ | ഉപഭോക്തൃ സേവനങ്ങൾ ഉത്തമമായി ഉറപ്പുവരുത്തുകയും എല്ലാ വ്യാപാര യൂണിറ്റ് സിസ്റ്റങ്ങളിലെയും വൈദഗ്ദ്ധ്യം നിലനിർത്തി പോരുകയും വിൽപ്പനക്കാരും കരാറുകാരും പോലെയുള്ളവരുമായുള്ള തൊഴിൽപരമായ ബന്ധം വളർത്തികൊണ്ടു വരികയും ചെയ്യുക. |
| സിസ്റ്റം അനലിസ്റ്റ് | പുതിയ ഐ.ടി. പരിഹാരങ്ങൾ രൂപകൽപ്പന ചെയ്ത് വ്യാപാരത്തിന്റെ കാര്യക്ഷമതയും ഉൽപ്പാദനക്ഷമതയും മെച്ചപ്പെടുത്തുക. |
| ഡാറ്റാ ബേസ് അഡ്മിനിസ്ട്രേറ്റേഴ്സ് | ഡാറ്റാബേസ് പരിഹാരങ്ങൾ രൂപകല്പന ചെയ്യുകയും നിരീക്ഷിക്കുകയും അപഗ്രഥിക്കുകയും നടപ്പിൽ വരുത്തുകയും ചെയ്യുക. |
| കമ്പ്യൂട്ടർ എഞ്ചിനീയർമാർ | കമ്പ്യൂട്ടർ സിസ്റ്റത്തിലെ ഹാർഡ്വെയറിന്റെയും സോഫ്റ്റ്വെയറിന്റെയും രൂപകൽപ്പന നടത്തുക. |
| കമ്പ്യൂട്ടർ പ്രോഗ്രാമർ | കമ്പ്യൂട്ടറുകളെ ശരിയായ രീതിയിൽ പ്രവർത്തിപ്പിക്കുവാൻ ആവശ്യമായ കോഡുകൾ എഴുതുന്നു. |
| കമ്പ്യൂട്ടർ ഓപ്പറേറ്റർ | കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ മേൽനോട്ടം നിർവ്വഹിക്കുന്നു. ഇവ വേണ്ടവിധം പ്രവർത്തിക്കുന്നു എന്ന് ഉറപ്പുവരുത്തുന്നു. ഭൗതിക സുരക്ഷിതത്വം ഉറപ്പാക്കുന്നു. തെറ്റുകൾ വരുന്ന സാഹചര്യം ഒഴിവാക്കുന്നു. |

*പട്ടിക 3.9 : വിവിധതരം ഹ്യൂമൺവെയറുകളും അവരുടെ ജോലിവിവരങ്ങളും*

## സ്വയം പരിശോധിക്കുക

1. സ്വതന്ത്ര ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്വെയറിന് ഒരു ഉദാഹരണം നൽകുക.
2. വാങ്ങുന്നതിനുമുമ്പ് ഉപഭോക്താവിന് ഉപയോഗിക്കാനുള്ള സൗകര്യം ഒരുക്കുന്ന സോഫ്റ്റ്വെയർ ആണ്_____
3. ഫ്രീ ആന്റ് ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്വെയർ എന്നാലെന്ത്?
4. ഉടമസ്ഥാവകാശമുള്ള സോഫ്റ്റ്വെയറിന് ഉദാഹരണം നൽകുക.
5. ഹ്യൂമൺവെയറിന് രണ്ട് ഉദാഹരണം എഴുതുക.

## നമുക്ക് സംഗ്രഹിക്കാം

തുടർച്ചയായ പ്രവർത്തനങ്ങളിലൂടെ ഡാറ്റയെ വിവരങ്ങളാക്കി മാറ്റുന്ന പ്രക്രിയയാണ് ഡാറ്റ പ്രോസസ്സിംഗ്. കമ്പ്യൂട്ടർ ഉപയോഗിച്ചുള്ള ഇലക്ട്രോണിക് ഡാറ്റ പ്രോസസ്സിങ് മാനുവൽ ഡാറ്റാ പ്രോസസ്സിങിന്റെ പരിമിതികൾ മറികടക്കാൻ സഹായിക്കുന്നു. ഒരു കമ്പ്യൂട്ടറിന് ഇൻപുട്ട് യൂണിറ്റ്, സെൻട്രൽ പ്രോസസ്സിങ് യൂണിറ്റ്, സംഭരണ യൂണിറ്റ്, ഔട്ട്പുട്ട് യൂണിറ്റ് എന്നിങ്ങനെ നാല് പ്രവർത്തനഘടകങ്ങൾ ഉണ്ട്. ഈ അധ്യായത്തിൽ കമ്പ്യൂട്ടർ ഓർഗനൈസേഷനെക്കുറിച്ച് മൊത്തത്തിൽ പൊതുവായ ഒരു ധാരണ നൽകിയിരിക്കുന്നു. ഇൻപുട്ട് ഔട്ട്പുട്ട് ഉപകരണങ്ങളെക്കുറിച്ചും, ഇ-വേസ്റ്റിനെക്കുറിച്ചും അതിന്റെ നിർമാർജ്ജന രീതികളെക്കുറിച്ചും ഹരിത കമ്പ്യൂട്ടിങ്ങിന്റെ പ്രാധാന്യത്തെക്കുറിച്ചും വിശദീകരിച്ചിരിക്കുന്നു. വിവിധതരം സോഫ്റ്റ്വെയറുകളെക്കുറിച്ചും ഓപ്പറേറ്റിംങ്ങ് സിസ്റ്റത്തിന്റെ ആവശ്യകതയെക്കുറിച്ചും അതിന്റെ പ്രധാന പ്രവർത്തനങ്ങളെക്കുറിച്ചും ചർച്ച ചെയ്തു. കമ്പ്യൂട്ടർ ഭാഷകളുടെ വിഭാഗങ്ങളെക്കുറിച്ച് വിശദീകരിച്ചിരിക്കുന്നു. ഓപ്പൺ സോഴ്സിന്റെ ആശയങ്ങളെകുറിച്ചും, ഫ്രീവെയർ, ഷെയർവെയർ, സ്വതന്ത്ര സോഫ്റ്റ്വെയർ, ഉടമസ്ഥാവകാശമുള്ള സോഫ്റ്റ്വെയർ എന്നിവയെക്കുറിച്ചും വിശദമായി ചർച്ച ചെയ്തു. ഈ അധ്യായം അവസാനിച്ചിരിക്കുന്നത് ഹ്യൂമൺ വെയറിന്റെ ആശയത്തിന് വിവരണം നൽകി കൊണ്ടാണ്.

## പഠന നേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ അവസാനം പഠിതാവിന് കിട്ടിയിരിക്കേണ്ട കഴിവുകൾ

- ഡാറ്റയെയും വിവരങ്ങളേയും വേർതിരിക്കുക.
- ഡാറ്റ പ്രോസസ്സിങ്ങിന്റെ വ്യത്യസ്ഥ ഘട്ടങ്ങൾ തിരിച്ചറിയൽ
- കമ്പ്യൂട്ടർ സിസ്റ്റത്തിന്റെ ബേസിക് ഓർഗനൈസേഷൻ വിവരണം
- പലതരത്തിലുള്ള ഇൻപുട്ട് ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ തിരിച്ചറിയൽ
- സിസ്റ്റം സോഫ്റ്റ്വെയറും ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയറും വേർതിരിക്കുക
- ഇ-വേസ്റ്റ് നിർമ്മാർജ്ജനത്തിന്റെ പ്രാധാന്യം തിരിച്ചറിയൽ
- ഹരിത കമ്പ്യൂട്ടിങ്ങിന്റെ ആശയം തിരിച്ചറിയൽ
- പലതരത്തിലുള്ള സോഫ്റ്റ്വെയറുകളെ തരം തിരിക്കൽ
- ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന്റെ ധർമ്മങ്ങൾ തിരിച്ചറിയുക
- വേഡ് പ്രൊസസ്സറും ഇലക്ട്രോണിക് സ്പ്രെഡ്ഷീറ്റും പ്രസന്റേഷൻ സോഫ്റ്റ്വെയറും ഉപയോഗിക്കൽ
- വിവിധതരത്തിലുള്ള കമ്പ്യൂട്ടർ ഭാഷകൾ തരംതിരിക്കൽ
- വ്യത്യസ്ഥ തരത്തിലുള്ള യൂട്ടിലിറ്റി സോഫ്റ്റ്വെയറുകൾ പട്ടികപ്പെടുത്തൽ
- ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്വെയർ പ്രോത്സാഹിപ്പിക്കൽ
- ഹ്യൂമൺവെയറും ഷെയർവെയറും വിവരിക്കൽ

## മാതൃകചോദ്യങ്ങൾ

### ഹ്രസ്വോത്തര ചോദ്യങ്ങൾ

1. ഡാറ്റ എന്നാൽ എന്ത്?
2. പ്രൊസസ്സ് ചെയ്ത ഡാറ്റ അറിയപ്പെടുന്നത് _____.
3. ഡിജിറ്റൽ കമ്പ്യൂട്ടറിന്റെ ഘടകങ്ങൾ ഏതൊക്കെയാണ്?
4. സി.പി.യു.വിന്റെ പ്രധാന ധർമ്മങ്ങൾ എഴുതുക?
5. വിവിധതരത്തിലുള്ള പ്രധാന മെമ്മറികൾ ഏതെല്ലാം?
6. EPROM ന് മീതെ EEPROM ന്റെ മേന്മകൾ എന്തെല്ലാം?
7. എപ്പോഴാണ് നമ്മൾ റോം (ROM) ഉപയോഗിക്കുന്നത്?
8. ഇൻപുട്ട് ഉപകരണം എന്നാൽ എന്ത്? സാധാരണയായി ഉപയോഗിക്കുന്ന ഇൻപുട്ട് ഉപകരണങ്ങൾ പട്ടികപ്പെടുത്തുക.
9. ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ എന്നാൽ എന്ത്? സാധാരണയായി ഉപയോഗിക്കുന്ന ഔട്ട്പുട്ട് ഉപകരണങ്ങൾ പട്ടികപ്പെടുത്തുക.
10. സംഭരണ ഉപകരണങ്ങൾ എന്താണ്? സാധാരണ ഉപയോഗിക്കുന്ന സംഭരണ ഉപകരണങ്ങൾ പട്ടികപ്പെടുത്തുക.
11. അരിത്തമെറ്റിക് ലോജിക് യൂണിറ്റി (ALU) ന്റെ പങ്ക് എന്താണ്?
12. കൺട്രോൾ യൂണിറ്റ് എന്താണ്?
13. രജിസ്റ്റേഴ്സ് എന്താണ്? ഏതെങ്കിലും രണ്ട് എണ്ണം എഴുതുകയും വിശദീകരിക്കുകയും ചെയ്യുക
14. ഹാർഡ്കോപ്പിയും സോഫ്റ്റ്കോപ്പിയും താരതമ്യം ചെയ്യുക
15. ഇ-വേസ്റ്റ് എന്നാൽ എന്താണ്?
16. ഓപ്പറേറ്റിങ് സിസ്റ്റം എന്നാൽ എന്താണ്?
17. ഭാഷ പ്രൊസസ്സർ എന്താണ്?
18. കമ്പ്യൂട്ടർ ഭാഷകളെ തരംതിരിക്കുക?
19. ഡിസ്ക് ഡിഫ്രാഗ്മെന്റർ എന്താണ്?
20. ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിനെ വിഭവ മാനേജരായി പരിഗണിക്കുന്നതെന്തുകൊണ്ട്?
21. ഉടമസ്ഥാവകാശമുള്ള സോഫ്റ്റ്വെയർ എന്താണ്?
22. ഓപ്പൺ സോഴ്സ് സോഫ്റ്റ്വെയർ കൊണ്ടുദ്ദേശിക്കുന്നതെന്താണ്?

## ലഘു ഉപന്യാസ ചോദ്യങ്ങൾ

1. ഡാറ്റയും വിവരവും വേർതിരിച്ചെഴുതുക
2. പ്ലസ് വൺ അഡ്മിഷനു വേണ്ടിയുള്ള അപേക്ഷ ഫോമിൽ വ്യക്തിപരമായ വിവരങ്ങളും വിവിധ ഗ്രൂപ്പുകളും സ്കൂളും കൊടുത്തിരിക്കുന്നു.

   a) അഡ്മിഷൻ പ്രോസസ്സിൽ വരുന്ന ഡാറ്റയും വിവരങ്ങളും തിരിച്ചറിയുക

   b) വിവരങ്ങൾ, അപേക്ഷ കൊടുക്കുന്നവരെയും സ്കൂൾ അധികാരികളെയും സഹായിക്കുന്നത് എങ്ങനെ എന്ന് വിശദീകരിക്കുക.

   c) ഡാറ്റ പ്രൊസസ്സ് ചെയ്യുമ്പോൾ ഉൾപ്പെട്ടിട്ടുള്ള പ്രവർത്തനങ്ങൾ എഴുതുക.
3. ഏതെങ്കിലും മൂന്ന് ഇൻപുട്ട് ഉപകരണങ്ങളെക്കുറിച്ച് ചുരുക്കി വിശദീകരിക്കുക.
4. സി.ആർ.ടി. (CRT) മോണിറ്ററും എൽ.ഇ.ഡി. (LED) മോണിറ്ററും താരതമ്യം ചെയ്യുക
5. റാം, റോം എന്നിവ തമ്മിലുള്ള വ്യത്യാസം എഴുതുക
6. ഇ-വേസ്റ്റ് നിർമ്മാർജ്ജനം പട്ടികപ്പെടുത്തി വിശദീകരിക്കുക
7. ഹരിത കമ്പ്യൂട്ടിംഗ് നടപ്പിൽ വരുത്തുവാൻ ആവശ്യമായ ഘട്ടങ്ങളെക്കുറിച്ച് സംഗ്രഹിക്കുക
8. സാമ്പ്രദായികമായി ചിട്ടപ്പെടുത്തിയ സോഫ്റ്റ്വെയർ കൊണ്ടുദ്ദേശിക്കുന്നതെന്താണ്? ഉദാഹരണം നൽകുക.
9. താഴ്ന്നതലത്തിലുള്ള ഭാഷകളും ഉയർന്നതലത്തിലുള്ള ഭാഷകളും വേർതിരിച്ചെഴുതുക.
10. കമ്പൈലർ, ഇന്റർപ്രിറ്റർ എന്നിവ വേർതിരിക്കുക
11. ഇലക്ട്രോണിക് സ്പ്രെഡ്ഷീറ്റിന്റെ ഉപയോഗത്തെക്കുറിച്ച് വിശദീകരിക്കുക
12. യൂട്ടിലിറ്റി സോഫ്റ്റ്വെയർ എന്താണ്? രണ്ട് ഉദാഹരണങ്ങൾ നൽകുക.
13. താഴെ കൊടുത്തിരിക്കുന്ന സോഫ്റ്റ്വെയറുകളെ ഓപ്പറേറ്റിംഗ് സിസ്റ്റം, ആപ്ലിക്കേഷൻ പാക്കേജ്, യൂട്ടിലിറ്റി പ്രോഗ്രാമുകൾ എന്നിങ്ങനെ തരംതിരിക്കുക

    Linux, Tally, Winzip, MS-Word, Windows, MS-Excel
14. ഫ്രീവെയറും ഷെയർവെയറും വേർതിരിക്കുക
15. സോഫ്റ്റ്വെയറുകൾ ഫ്രീയായും ഓപ്പൺ സോഴ്സായും നിർമ്മിക്കുമ്പോൾ ഉണ്ടായിരിക്കേണ്ട നാല് സ്വാതന്ത്ര്യങ്ങൾ ഏതെല്ലാമാണ്.
16. ഹ്യൂമൺവെയർ കൊണ്ടുദ്ദേശിക്കുന്നതെന്താണ്? ഏതെങ്കിലും രണ്ട് ഉദാഹരണങ്ങൾ നൽകുക.

## ഉപന്യാസ ചോദ്യങ്ങൾ

1. നിത്യ ജീവിതത്തിലെ ഏതെങ്കിലും ഉദാഹരണം എടുത്ത് ഡാറ്റ പ്രോസസ്സിങ്ങിന്റെ ഓരോ ഘട്ടത്തിലെയും പ്രവർത്തനങ്ങൾ ചുരുക്കി വിവരിക്കുക.
2. ബ്ലോക്ക് ഡയഗ്രത്തിന്റെ സഹായത്തോടെ, കമ്പ്യൂട്ടറിന്റെ പ്രവർത്തനഘടകങ്ങളെക്കുറിച്ച് വിശദീകരിക്കുക.
3. സെൻട്രൽ പ്രോസസ്സിങ്ങ് യൂണിറ്റിനെക്കുറിച്ച് വിശദമായി വിവരിക്കുക.
4. വിവിധതരത്തിലുള്ള മെമ്മറിയെക്കുറിച്ച് ചുരുക്കി വിവരിക്കുക.
5. പ്രിന്ററിന്റെ വർഗ്ഗീകരണത്തെക്കുറിച്ച് വിശദീകരിക്കുക.
6. നമ്മുടെ ആരോഗ്യത്തിനും പരിസ്ഥിതിക്കും ഇ-വേസ്റ്റ് ആപൽക്കരമാണ്. പ്രസ്താവന ന്യായീകരിക്കുക. ഇ-വേസ്റ്റ് നിർമ്മാർജ്ജനത്തിന് പൊതുവായി ഉപയോഗിക്കുന്ന രീതികൾ പട്ടികപ്പെടുത്തി വിശദമാക്കുക.
7. ഹരിത കമ്പ്യൂട്ടിങ് നിർവ്വചിക്കുക. നിങ്ങൾക്ക് സാധ്യമാകുന്ന രീതിയിൽ ഹരിത കമ്പ്യൂട്ടിംങ്ങിന്റെ പ്രോത്സാഹനത്തിനുള്ള ആശയങ്ങൾ വിശദമാക്കുക.
8. സോഫ്റ്റ്വെയറിന്റെ വിവിധ വിഭാഗങ്ങൾ പട്ടികപ്പെടുത്തി വിശദീകരിക്കുക.
9. വിവിധ തരത്തിലുള്ള യൂട്ടിലിറ്റി സോഫ്റ്റ്വെയറുകളുടെ ഉപയോഗം വിശദീകരിക്കുക
10. ഓപ്പറേറ്റിങ് സിസ്റ്റം നിർവ്വചിക്കുക. ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിന്റെ പ്രധാന ധർമ്മങ്ങൾ പട്ടികപ്പെടുത്തി വിശദമാക്കുക.
11. പൊതുവായ പ്രവർത്തനങ്ങൾക്കുള്ള ആപ്ലിക്കേഷൻ സോഫ്റ്റ്വെയർ ഉദാഹരണ സഹിതം വിശദമാക്കുക.
12. ഫ്രീവെയറും ഷെയർവെയറും താരതമ്യം ചെയ്യുക.

4

# പ്രോഗ്രാമിങ്ങിന്റെ തത്വങ്ങളും പ്രശ്നപരിഹാരവും

## പ്രധാന ആശയങ്ങൾ

- **കമ്പ്യൂട്ടറുകളുടെ സഹായത്തോടെയുള്ള പ്രശ്ന പരിഹാരം**
- **പ്രശ്ന പരിഹാരത്തിനുള്ള സമീപനങ്ങൾ**
  - ടോപ് ഡൗൺ രൂപകല്പന
  - ബോട്ടം അപ് രൂപകല്പന
- **പ്രോഗ്രാമിങ്ങിലെ വിവിധ ഘട്ടങ്ങൾ**
  - പ്രശ്നം തിരിച്ചറിയൽ (Problem Identification)
  - അൽഗോരിതങ്ങളും ഫ്ളോച്ചാർട്ടുകളും തയാറാക്കൽ (Preparing algorithms and Flowcharts)
  - പ്രോഗ്രാം കോഡിങ് (Program Coding)
  - പരിഭാഷ (Translation)
  - ഡീബഗ്ഗിങ് (Debugging)
  - പ്രവർത്തനവും പരീക്ഷണവും (Execution and Testing)
  - വിവരണം തയാറാക്കൽ (Documentation)
- അൽഗോരിതങ്ങളുടെ പ്രകടനം വിലയിരുത്തൽ


ഡാറ്റാ പ്രോസസ്സിങ് എന്ന ആശയവും, അതിൽ കമ്പ്യൂട്ടറുകൾക്കുള്ള പങ്കും നാം പഠിച്ചിട്ടുണ്ട്. ഹാർഡ്വെയർ, സോഫ്റ്റ്വെയർ, ഉപയോക്താക്കൾ എന്നിവ അടങ്ങുന്ന ഒരു സംവിധാനം എന്ന നിലയിലും ഈ ഘടകങ്ങളെല്ലാം മുൻ അധ്യായത്തിൽ നാം വിശദമായി ചർച്ച ചെയ്തിട്ടുണ്ട്. സോഫ്ട്വെയറിന്റെ നിർവചനം നമുക്ക് ഓർത്തെടുത്ത് നോക്കാം. ലളിതമായ രീതിയിൽ പറഞ്ഞാൽ, കമ്പ്യൂട്ടർ ഉപയോഗിച്ച് പ്രശ്നങ്ങൾ പരിഹരിക്കുന്നതിനുള്ള പ്രോഗ്രാമുകളുടെ ശേഖരമാണ് സോഫ്റ്റ്വെയർ. കമ്പ്യൂട്ടറിന് സ്വന്തമായി ഒന്നും ചെയ്യാനുള്ള കഴിവില്ലെന്ന് നമുക്കറിയാവുന്നതാണ്. ഒരു ജോലി നിർവഹിക്കണമെങ്കിൽ വ്യക്തമായ നിർദേശങ്ങൾ കമ്പ്യൂട്ടറിന് ആവശ്യമാണ്. ആയതിനാൽ ഒരു പ്രശ്ന പരിഹാരത്തിന് കമ്പ്യൂട്ടർ നിർവഹിക്കേണ്ട നിർദേശങ്ങളുടെ ക്രമം വ്യക്തമാക്കേണ്ടത് അത്യാവശ്യമാണ്. ഇത്തരത്തിൽ കമ്പ്യൂട്ടറിന് മനസ്സിലാകുന്ന ഒരു ഭാഷയിൽ എഴുതിയിരിക്കുന്ന, ക്രമത്തിലുള്ള നിർദേശങ്ങൾ 'കമ്പ്യൂട്ടർ പ്രോഗ്രാം' എന്ന പേരിൽ അറിയപ്പെടുന്നു. കമ്പ്യൂട്ടർ പ്രോഗ്രാം എഴുതുക എന്നത് ഒരു വെല്ലുവിളിയാണ്. എന്നിരുന്നാലും പ്രോഗ്രാമിങ്ങിന്റെ വിവിധ ഘട്ടങ്ങളും പ്രശ്നപരിഹാരസാങ്കേതികത്വ ആശയങ്ങളും ആർജിച്ചുകൊണ്ട് നമുക്ക് അതിനായി ശ്രമിക്കാം.

## 4.1 കമ്പ്യൂട്ടറുകൾ ഉപയോഗിച്ചുള്ള പ്രശ്നപരിഹാരം (Problem solving using computers)

കമ്പ്യൂട്ടറിലേക്ക് നമ്മൾ നിർദേശങ്ങൾ നൽകിയാൽ മാത്രമെ അതിനു പ്രശ്നങ്ങൾ പരിഹരിക്കാൻ കഴിയുകയുള്ളൂ. നിർദേശങ്ങളിൽ അടങ്ങിയിരിക്കുന്ന ക്രിയകൾ മനസ്സിലായാൽ അതിനനുസരിച്ച് കമ്പ്യൂട്ടർ പ്രവർത്തിക്കും. നിർദേശം (Instruction) ഒരു ക്രിയാധിഷ്ഠിത പ്രസ്താവനയാണ്. ഏതു പ്രവൃത്തിയാണ് നിർവഹിക്കേണ്ടത് എന്ന് കംപ്യൂട്ടറിനോട് അത് പറയുന്നു. കൃത്യതയോടും സൂക്ഷ്മതയോടും കൂടി ചെയ്യേണ്ട പ്രവൃത്തി വ്യക്തമാക്കിയാൽ മാത്രമേ ഒരു നിർദേശത്തെ കമ്പ്യൂട്ടറിനു നിർവഹിക്കാൻ കഴിയുകയുള്ളൂ.

പ്രശ്നപരിഹാരത്തിനായി പ്രോഗ്രാമർമാർ ഒരു പ്രത്യേക ക്രമത്തിൽ നിർദേശങ്ങൾ എഴുതുന്നു എന്ന് മുൻ അധ്യായത്തിൽ നാം പഠിച്ചിട്ടുണ്ട്. പ്രോഗ്രാം വികസിപ്പിക്കുകയും കമ്പ്യൂട്ടറിൽ സ്ഥിരമായി സൂക്ഷിക്കുകയും ചെയ്തു കഴിഞ്ഞാൽ നമുക്ക് ആവശ്യാനുസരണം അവ പ്രവർത്തിപ്പിക്കാൻ കമ്പ്യൂട്ടറിനോട് ആവശ്യപ്പെടാം

കമ്പ്യൂട്ടറിനു സാമാന്യബുദ്ധിയോ അന്തർജ്ഞാനമോ ഇല്ലാത്തതിനാൽ ഒരു പ്രോഗ്രാം രൂപകല്പന ചെയ്യുമ്പോൾ പ്രശ്ന പരിഹാരത്തിന്റെ യുക്തിയും നിർദേശങ്ങളുടെ ഘടനയും വ്യക്തമായി നാം മനസ്സിലാക്കിയിരിക്കണം. മനുഷ്യരായ നാം അനുഭവങ്ങളുടെ അടിസ്ഥാനത്തിൽ വിഷയാധിഷ്ഠിതവും വൈകാരികവുമായ പരിഗണനകൾക്കനുസരിച്ച്, തീരുമാനങ്ങൾ കൈക്കൊള്ളുന്നു. അത്തരം മൂല്യാധിഷ്ഠിത വിധിന്യായങ്ങൾ പലപ്പോഴും സാമാന്യബോധത്തെ ആശ്രയിച്ചിട്ടായിരിക്കും. ഇതിനു വിരുദ്ധമായി, ഒരു കമ്പ്യൂട്ടറിന് വികാരപ്രകടനമോ സാമാന്യബുദ്ധിയോ ഇല്ല. അതുകൊണ്ടാണ് കമ്പ്യൂട്ടറിന് സ്വന്തമായ ബുദ്ധിവൈഭവം ഇല്ല എന്നു നാം പറയുന്നത്.

ഒരു വിധത്തിൽ പറഞ്ഞാൽ, കമ്പ്യൂട്ടറിനെ ഒരു 'അനുസരണയുള്ള ഭൃത്യൻ' ആയി കണക്കാക്കാം. 'സാമാന്യബോധം' ഉപയോഗിക്കാതെയുള്ള അനുസരണശീലം പലപ്പോഴും അലോസരപ്പെടുത്തുന്നതും ഫലമില്ലാത്തതുമാകുന്നു.ഉദാഹരണത്തിന്, 'തപാൽ ഓഫീസിൽ പോയി പത്ത് 5 രൂപ സ്റ്റാമ്പുകൾ വാങ്ങുക'. എന്ന് നിർദേശിച്ച് തന്റെ അനുസരണയുള്ള ഭൃത്യനെ തപാൽ ഓഫീസിലേക്ക് യജമാനൻ അയച്ചു എന്ന് കരുതുക. ഭൃത്യൻ കാശുമായി തപാൽ ഓഫീസിൽ പോയിട്ട് ദീർഘ നേരമായും തിരികെ വരുന്നില്ല. ചിന്താകുലനായ യജമാനൻ ഭൃത്യനെ അന്വേഷിച്ചു തപാൽ ഓഫീസിലെത്തുമ്പോൾ കാണുന്നത് സ്റ്റാമ്പുകളുമായി നിൽക്കുന്ന ഭൃത്യനെയാണ്.കോപിഷ്ഠനായ യജമാനൻ ഭൃത്യനോട് കാരണം അന്വേഷിക്കുമ്പോൾ, തന്നോട് പത്തു 5 രൂപ സ്റ്റാമ്പുകൾ വാങ്ങുവാൻ മാത്രമേ കല്പിച്ചിട്ടുള്ളൂ എന്നും, അവയുമായി തിരികെ വരാൻ നിർദേശിച്ചിട്ടില്ല എന്നുമുള്ള മറുപടിയാണ് ഭൃത്യനിൽ നിന്ന് ലഭിക്കുന്നത്. .

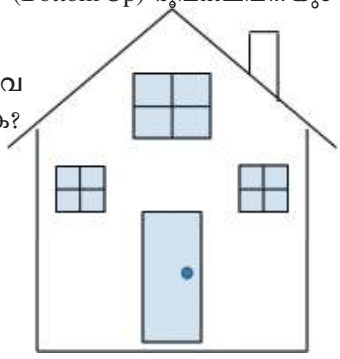
## 4.2 പ്രശ്നപരിഹാരത്തിലെ സമീപനങ്ങൾ (Approaches in problem solving)

ഒരു പ്രശ്നം വ്യത്യസ്ത രീതികളിലൂടെ പരിഹരിക്കാവുന്നതാണ്. സമീപനം പോലും വ്യത്യസ്തമായിരിക്കാം. നമ്മുടെ ദൈനംദിന ജീവിതത്തിൽ,അസുഖം ബാധിക്കുമ്പോൾ നാം ഒരു അലോപ്പതി, ആയുർവേദം അല്ലെങ്കിൽ ഹോമിയോപ്പതി ചികിത്സകനെ സമീപിച്ച് വൈദ്യചികിത്സ തേടുന്നു. ഒരേ രോഗമാണ് ചികിത്സിക്കുന്നതെങ്കിലും ഇവരിൽ ഓരോരുത്തരുടേയും സമീപനങ്ങൾ വ്യത്യസ്തമായിരിക്കും. അതു പോലെ പ്രശ്നപരിഹാരത്തിന് വ്യത്യസ്ത സമീപനങ്ങൾ ഉപയോഗിക്കുന്നു. പ്രശ്നപരിഹാരത്തിനുള്ള പ്രശസ്തമായ രണ്ടു രൂപകൽപ്പനാ രീതികൾ നമുക്ക് പരിചയപ്പെടാം ടോപ് ഡൗൺ (Top Down) രൂപകല്പനയും ബോട്ടം അപ് (Bottom Up) രൂപകല്പനയും

### 4.2.1 ടോപ് ഡൗൺ രൂപകല്പന

ചിത്രം 4.1 നോക്കുക. ഈ ചിത്രം വരയ്ക്കണമെന്ന് നിങ്ങളോട് ആവശ്യപ്പെടുകയാണെങ്കിൽ, എങ്ങനെയാണ് നിങ്ങൾ അത് വരയ്ക്കുക? ഒരു പക്ഷേ താഴെ പറയുന്ന പ്രകാരമായിരിക്കാം :

1. വീടിന്റെ രേഖാചിത്രം വരയ്ക്കുക.
2. ചിമ്മിനി വരയ്ക്കുക
3. വാതിൽ വരയ്ക്കുക
4. ജാലകങ്ങൾ വരയ്ക്കുക



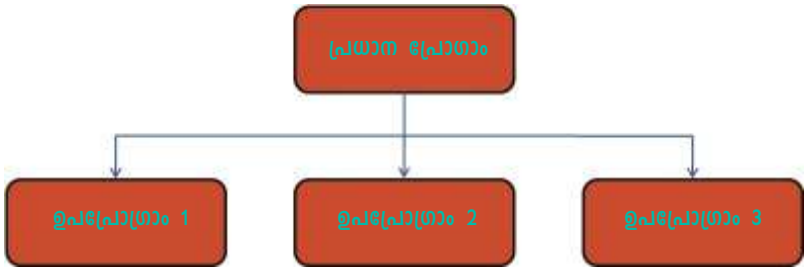
*ചിത്രം 4 .1 ഒരു വീടിന്റെ രേഖാചിത്രം*

മുകളിൽ വിവരിച്ച നടപടിക്രമങ്ങളെ ഇപ്രകാരം സംഗ്രഹിക്കാം :

| | |
|---|---|
| ഘട്ടം 3 ലെ വാതിൽ വരക്കുമ്പോഴുള്ള നടപടിക്രമം താഴെ കൊടുത്തിരിക്കുന്നു :<br>3.1 വാതിലിന്റെ ബാഹ്യരേഖ<br>3.2 ഷേഡിംഗ്<br>3.3 വാതിലിന്റെ പിടി | അത് പോലെ താഴെ പറയുന്ന പ്രകാരം ജാലകങ്ങൾ വരയ്ക്കാം<br>4.1 ജാലകത്തിന്റെ ബാഹ്യരേഖ<br>4.2 ഷേഡിംഗ്<br>4.3 തിരശ്ചീനവും ലംബവുമായ വരകൾ |

തന്നിരിക്കുന്ന പ്രശ്നത്തെ (ഇവിടെ ചിത്രം വരയ്ക്കുക എന്നത്) ചെറിയ ക്രിയകളായി (Task ) വിഭജിച്ചിരിക്കുന്നു. അതുപ്രകാരം ഈ പ്രശ്നം പരിഹരിക്കാൻ നാല് പ്രവർത്തനങ്ങൾ കണ്ടെത്തിയിട്ടുണ്ട്.ഇവയിൽ ചിലത് (ഇവിടെ വാതിലുകളും ജാലകങ്ങളും വരയ്ക്കുന്നത്) വീണ്ടും വിഭജിച്ചിട്ടുണ്ട്. അങ്ങനെ സങ്കീർണമായ ഒരു പ്രശ്നം,വിവിധ ക്രിയകളായി വിഭജിച്ച്, ഓരോ ക്രിയയെയും ലളിതമായ പ്രവർത്തനങ്ങളിലൂടെ പ്രാവർത്തികമാക്കാൻ കഴിയും. ഈ പ്രശ്ന പരിഹാര രീതി ടോപ് ഡൗൺ രൂപകല്പന (Top Down Design ) എന്ന് അറിയപ്പെടുന്നു.

ഏറ്റവും ഫലവത്താണെന്നു തെളിയിക്കപ്പെട്ട പ്രോഗ്രാമിങ് സമീപനങ്ങളിലൊന്നാണിത്. ചിത്രം 4.2 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ, ടോപ് ഡൗൺ രൂപകല്പന എന്നത് തന്നിരിക്കുന്ന നടപടിക്രമത്തെ അല്ലെങ്കിൽ കൃത്യത്തെ ഘടകങ്ങൾ ആക്കുകയും ഏറ്റവും അടിസ്ഥാനപരമായ ക്രിയകൾ അടങ്ങിയ ഘടകം ലഭിക്കുന്നത് വരെ ഓരോ ഘടകത്തെയും വീണ്ടും വിഭജിക്കുകയും ചെയ്യുന്നു പ്രക്രിയയാണ്. ഒരു പൊതുവായ പ്രശ്നം മുകളിലെ തലം മുതൽ ആരംഭിക്കുകയും അതിലെ ഓരോ ഉപവിഭാഗത്തിനും പ്രത്യേക പരിഹാരങ്ങൾ രൂപകല്പന നടത്തുകയും ചെയ്യുന്നതിനാൽ ടോപ് ഡൗൺ വിഭജനം എന്ന പേരിലും ഇത് അറിയപ്പെടുന്നു. തന്നിരിക്കുന്ന പ്രധാന പ്രശ്നത്തിന് ഫലപ്രദമായ പരിഹാരം ലഭിക്കണമെങ്കിൽ ഓരോ ഉപപ്രശ്നവും മറ്റൊന്നിൽ നിന്ന് സ്വതന്ത്രമായിരിക്കണം. അപ്രകാരമായാൽ ഓരോ ഉപപ്രശ്നവും സ്വതന്ത്രമായി പരിഹരിക്കാനും പരിശോധിക്കാനും സാധിക്കും.




*ചിത്രം 4.2: ഒരു പ്രശ്നത്തെ വിഭജിക്കുന്നു.*

വിഭജനത്തിലൂടെ പ്രശ്നം പരിഹരിക്കുന്നത് കൊണ്ടുള്ള പ്രയോജനങ്ങൾ താഴെപ്പറയുന്നവയാണ്:

- പ്രശ്നത്തെ വിഭജിക്കുന്നതു കൊണ്ട് ഓരോ ഭാഗത്തും എന്ത് പ്രവൃത്തിയാണ് ചെയ്യേണ്ടതെന്നതിനെക്കുറിച്ച് ഒരു ധാരണ ലഭിക്കാൻ നമ്മെ സഹായിക്കുന്നു.

- ഓരോ ഘട്ടത്തിലും തിരിച്ചറിയുന്ന പുതിയ ഉപപ്രശ്നങ്ങളിൽ സങ്കീർണത കുറവായതിനാൽ അതിന്റെ പ്രശ്ന പരിഹാരം എളുപ്പത്തിൽ ലഭിക്കുന്നു.
- പ്രശ്ന പരിഹാരത്തിലെ ചില ഭാഗങ്ങൾ പുനരുപയോഗിക്കാൻ കഴിയുന്നതായിരിക്കാം.
- പ്രശ്നവിഭജനത്തിലൂടെ ഒന്നിലധികം ആളുകൾക്ക് ഒരേ സമയം പ്രശ്ന പരിഹാരത്തിൽ പങ്കാളികളാകാൻ സാധിക്കുന്നു.

### 4.2.2 ബോട്ടം അപ് രൂപകല്പന (Bottom Up Design)

ഒരു വീടിന്റെ നിർമ്മാണപ്രവർത്തനം പരിഗണിക്കുക. ടോപ് ഡൗൺ രൂപകല്പനയല്ല മറിച്ചു ബോട്ടം അപ് രൂപകല്പനയാണ് നമ്മൾ ഇവിടെ പിന്തുടരുന്നത്. അസ്ഥിവാരമിടുക എന്നത് ആദ്യത്തെ പ്രവൃത്തിയും മേൽക്കൂര പണിയുക എന്നത് അവസാനത്തെ പ്രവൃത്തിയുമാണ്. ഇവിടെയും പ്രധാന പ്രവർത്തനത്തെ ഉപപ്രവർത്തനങ്ങളായി വിഭജനം നടത്തുന്നു. ഇവയിൽ ത്തന്നെ ചില പ്രവർത്തനങ്ങൾ പൂർത്തിയായാൽ മാത്രമേ മറ്റു പ്രവർത്തനങ്ങൾ നടത്തുവാൻ കഴിയൂ. എന്നാൽ താഴെത്തട്ടിലുള്ള എല്ലാ പ്രവർത്തനങ്ങളും പൂർത്തിയായാൽ മാത്രമേ പ്രധാന പ്രവർത്തനമായ മേൽക്കൂര പണിയൽ സാധ്യമാവുകയുള്ളൂ.

ഇതുപോലെ പ്രോഗ്രാമിങ്ങിലും ആകെയുള്ള നടപടിക്രമങ്ങളെ വിവിധ ഘടകങ്ങളായി വിഭജിക്കുകയും, ഏറ്റവും താഴ്ന്ന തലത്തിലുള്ള ഖണ്ഡം ലഭിക്കുന്നത് വരെ ഈ ഘടകങ്ങൾ പുനർവിഭജിക്കുകയും ചെയ്യുന്നു. ഏറ്റവും താഴ്ന്ന ഘടകം മുതൽ പ്രശ്ന പരിഹാരം ആരംഭി ക്കുന്നു. പ്രധാന പ്രശ്നത്തിനുള്ള പരിഹാരം, ഉപവിഭാഗങ്ങളുടെ പരിഹാരത്തിനു ശേഷം മാത്രമേ സാധ്യമാകൂ.ഈ രീതിയിലുള്ള സമീപനത്തെ പ്രശ്ന പരിഹാരത്തിനുള്ള ബോട്ടം അപ്പ് രൂപക ല്പന എന്ന് പറയുന്നു. മുൻപ് പറഞ്ഞത് പോലെ ഇവിടെയും ഒരു ഉപപ്രശ്നം മറ്റൊരു ഉപപ്ര ശ്നത്തിൽ നിന്ന് സ്വതന്ത്രമായിരിക്കുന്നതാണ് അഭിലഷണീയം.

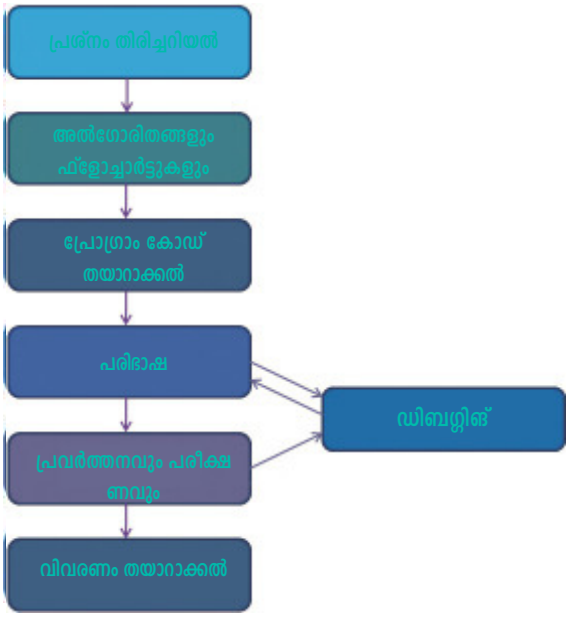
ചെയ്തു നോക്കൂ

*എല്ലാ വർഷവും നമ്മുടെ സ്കൂളുകളിൽ യുവജനോത്സവം നടത്താറുണ്ട്. ഈ അവസരത്തിൽ, സാധാരണഗതിയിൽ ചുമതലകളും ഉത്തരവാദിത്വങ്ങളും വിഭജിച്ച് നൽകുന്നു. യുവജനോത്സവത്തിന്റെ വിജയകരമായ നടത്തിപ്പിന് എങ്ങനെയാണ് ഓരോ പ്രവർത്തനവും വിഭജിക്കുകയും പ്രാവർത്തി കമാക്കുകയും ചെയ്യുന്നതെന്ന് ചർച്ച ചെയ്യുക.*

## 4.3 പ്രോഗ്രാമിങ്ങിലെ വിവിധ ഘട്ടങ്ങൾ (Phases in programming)

കമ്പ്യൂട്ടർ ഉപയോഗിച്ച് പ്രശ്നം പരിഹരിക്കുക എന്നത് ഒരു വലിയ വെല്ലുവിളിയാണ് എന്ന് നാം കണ്ടുകഴിഞ്ഞു. ഇതിനായി ചിട്ടയായി ഒരു സമീപനം അത്യന്താപേക്ഷിതമാണ്. ആവശ്യമായ പ്രോഗ്രാമുകൾ നിർമ്മിക്കുന്നതിന് വിവിധ ഘട്ടങ്ങളിലൂടെ കടന്നുപോകേണ്ടതുണ്ട്. പ്രശ്ന പരിഹാ രത്തിനുള്ള ജൻമസിദ്ധമായ കഴിവ് നമുക്കുണ്ടെങ്കിലും അത് ഫലപ്രദമായി ഉപയോഗപ്പെടുത്ത ണമെങ്കിൽ പ്രശ്നം പരിഹരിക്കുന്നതിന് ഉതകുന്ന രീതിയിലുള്ള ചിന്തയും, ആസൂത്രണവും യുക്തിസഹമായ ന്യായവാദവും വളർത്തിയെടുക്കേണ്ടതുണ്ട്. താഴെപ്പറയുന്ന ഘട്ടങ്ങൾ പിന്തു ടർന്ന് നമുക്കിത് നേടിയെടുക്കാവുന്നതാണ്.

1. പ്രശ്നം തിരിച്ചറിയൽ (Problem Identification)
2. അൽഗോരിതങ്ങളും ഫ്ളോച്ചാർട്ടുകളും തയാറാക്കൽ (Preparing algorithms and Flowcharts)
3. പ്രോഗ്രാമിങ് ഭാഷ ഉപയോഗിച്ച് പ്രോഗ്രാം കോഡ് ചെയ്യൽ (Coding the Program using Programming Language)
4. പരിഭാഷ (Translation)
5. ഡീബഗ്ഗിങ് (Debugging)
6. പ്രവർത്തനവും പരീക്ഷണവും (Execution and Testing)
7. വിവരണം തയ്യാറാക്കുക (Documentation)




*ചിത്രം 4.3 പ്രോഗ്രാമിങ്ങിലെ വിവിധ ഘട്ടങ്ങൾ*

പ്രോഗ്രാമിങ്ങിന്റെ വിവിധ ഘട്ടങ്ങളിൽ ചെയ്യുന്ന പ്രവർത്തനങ്ങളുടെ ക്രമം ചിത്രം 4.3 ൽ കാണിച്ചിരിക്കുന്നു. ഡീബഗ്ഗിങ് ഘട്ടം പരിഭാഷയുമായും നിർവഹണവുമായും ബന്ധപ്പെട്ടിരിക്കുന്നു എന്നത് ശ്രദ്ധിക്കുക. മേൽപ്പറഞ്ഞ ഏഴ് ഘട്ടങ്ങളിൽ ഉൾപ്പെട്ടിരിക്കുന്ന പ്രവർത്തനങ്ങൾ താഴെ വിശദീകരിക്കുന്നു.

### 4.3.1 പ്രശ്നം തിരിച്ചറിയൽ (Problem Identification)

നിങ്ങൾക്കു വയറ് വേദന അനുഭവപ്പെടുന്നതായി കരുതുക. ഈ പ്രശ്നം ഒരു ഡോക്ടർക്കു പരിഹരിക്കാൻ കഴിയുമെന്ന് നിങ്ങൾക്കറിയാം. വേദന തുടങ്ങിയ സമയം,മുൻപ് ഇത് പോലെ വേദന അനുഭവപ്പെട്ട സാഹചര്യം, ഭക്ഷണരീതി എന്നിവയെക്കുറിച്ചുള്ള ചില ചോദ്യങ്ങൾ ഡോക്ടർ ചോദിക്കുകയും, സ്റ്റെതസ്കോപ്പ്, എക്സ്റേ അല്ലെങ്കിൽ സ്കാൻ എന്നിവ ഉപയോഗിച്ച് നിങ്ങളുടെ ശരീരത്തിന്റെ ചില ഭാഗങ്ങൾ പരിശോധിക്കുകയും ചെയ്യുന്നു.ഇവയെല്ലാം രോഗനിർണയത്തിന്റെ ഭാഗമാണ്. ഈ നടപടിക്രമങ്ങൾക്ക് ശേഷം, ഡോക്ടർ പ്രശ്നം തിരിച്ചറിഞ്ഞ് ചില വൈദ്യശാസ്ത്ര പദങ്ങൾ ഉപയോഗിച്ച് നിങ്ങളെ ബോധ്യപ്പെടുത്തുന്നു.അടുത്ത ഘട്ടം ഈ പ്രശ്നത്തിനുള്ള പരിഹാര നടപടികൾ തയാറാക്കലാണ്. അതിനെ മരുന്ന് കുറിപ്പ് എന്ന് പറയുന്നു.

ഇതിൽ നിന്നും പ്രശ്ന പരിഹാരത്തിനുള്ള നടപടികൾ സ്വീകരിക്കുന്നതിന് മുമ്പ് പ്രശ്നം വിശകലനം ചെയ്യേണ്ടത് അത്യാവശ്യമാണ് എന്ന് വ്യക്തമാകുന്നു. ഈ ഘട്ടത്തിൽ നിങ്ങൾക്ക് നടപ്പിലാക്കേണ്ട പ്രവർത്തനത്തിന് ആവശ്യമായ ഡാറ്റ, അതിന്റെ ഇനം, അളവ്, ഉപയോഗിക്കേണ്ട സൂത്രവാക്യം, ഉൾപ്പെട്ടിരിക്കുന്ന പ്രവർത്തനങ്ങൾ ലഭിക്കേണ്ട ഔട്ട്പുട്ട് എന്നിവ തിരിച്ചറിയാൻ സാധിക്കുന്നു. പ്രശ്നം വ്യക്തമായി പഠിക്കുകയും, പ്രശ്നപരിഹാരത്തിനാവശ്യമായ കൃത്യങ്ങളുടെ ക്രമം സംബന്ധിച്ച് നമുക്ക് ബോധ്യമാകുകയും ചെയ്താൽ, അടുത്ത ഘട്ടത്തിലേക്ക് പ്രവേശിക്കാം. പ്രോഗ്രാമറുടെ (പ്രശ്ന പരിഹാരകൻ) കാര്യക്ഷമത പരമാവധി ഉപയോഗപ്പെടുത്തേണ്ടതിനാൽ തീർച്ചയായും ഇത് ഒരു വെല്ലുവിളി നിറഞ്ഞ ഘട്ടമാണ്.

### 4.3.2 അൽഗോരിതങ്ങളും ഫ്ളോച്ചാർട്ടുകളും (Algorithms and Flowcharts)

പ്രശ്നം തിരിച്ചറിഞ്ഞു കഴിഞ്ഞാൽ അത് പരിഹരിക്കാൻ പടിപടിയായുള്ള നടപടിക്രമങ്ങൾ കൃത്യമായി വികസിപ്പിക്കേണ്ടത് അത്യാവശ്യമാണ്. ഈ നടപടിക്രമം പുതിയതോ കമ്പ്യൂട്ടർ മേഖലയ്ക്കുമാത്രം പരിമിതമായതോ അല്ല. കാലാകാലമായി ജീവിതത്തിന്റെ എല്ലാ മേഖലകളിലും തുടർന്ന് കൊണ്ടിരിക്കുന്ന ഒന്നാണിത്. നിത്യ ജീവിതത്തിൽ നിന്നെടുത്ത അത്തരത്തിലുള്ള ഒരു നടപടിക്രമം താഴെ വിവരിച്ചിരിക്കുന്നു. ഒരു മാസികയിൽ നിന്നും എടുത്തിട്ടുള്ള ഓംലെറ്റ് തയാറാക്കാനുള്ള പാചകക്കുറിപ്പാണിത്

***ചേരുവകൾ***

മുട്ട 2 എണ്ണം, ഉള്ളി 1 എണ്ണം (ചെറുതായി അരിഞ്ഞത്); പച്ച മുളക് 2 (ചെറുതായി അരിഞ്ഞത്); എണ്ണ 2 ടീ സ്പൂൺ, ഉപ്പ് ഒരു നുള്ള്.

***രീതി***

ഘട്ടം 1 : മുട്ടകൾ പൊട്ടിച്ച് ഒരു പാത്രത്തിലിട്ട് നന്നായി ഇളക്കുക.

ഘട്ടം 2 : അരിഞ്ഞു വച്ച ഉള്ളി, പച്ചമുളക്, ഉപ്പ് എന്നിവ മുട്ടയിൽ ചേർത്തിളക്കുക.

ഘട്ടം 3 : അടുപ്പിൽ ഒരു പാൻ വച്ച് അടുപ്പു കത്തിക്കുക.

ഘട്ടം 4 : പാനിൽ എണ്ണ ഒഴിച്ച്, ചൂടാകുന്നതുവരെ കാത്തിരിക്കുക.

ഘട്ടം 5 : ഘട്ടം 2 ൽ തയാറാക്കിയ മിശ്രിതം പാനിൽ ഒഴിച്ച് ഒരു ഭാഗം പൊരിയുന്നത് വരെ കാത്തിരിക്കുക.

ഘട്ടം 6 : മറിച്ചിട്ട് മറുവശം നന്നായി പൊരിക്കുക.

ഘട്ടം 7 : കുറച്ച് സെക്കൻഡുകൾക്ക് ശേഷം ഇത് എടുക്കുക.

***ഫലം***

കുരുമുളക് പൊടി ചേർത്ത് കഴിക്കാൻ പാകത്തിനുള്ള ഒരു ഓംലെറ്റ് തയാർ.

മുകളിൽ കൊടുത്ത പാചകകുറിപ്പിന് താഴെപ്പറയുന്ന സവിശേഷതകൾ ഉണ്ട്:

1. ഓംലെറ്റ് പാചകം ചെയ്യാൻ ആവശ്യമായ ചേരുവകളുടെ ഒരു പട്ടിക നിർമിക്കുന്നതിലൂടെയാണ് പ്രവർത്തനം ആരംഭിക്കുന്നത്. ഈ ചേരുവകളെ ഇൻപുട്ടുകൾ എന്ന് വിളിക്കാം.

2. ഇൻപുട്ടുകൾ ഉപയോഗിച്ചുള്ള പ്രവർത്തനത്തിന് ആവശ്യമായ, ക്രമത്തിലുള്ള നിർദേശങ്ങൾ നൽകുന്നു.

3. നിർദേശങ്ങൾ നടപ്പാക്കുന്നതിന്റെ ഫലമായി ചില ഔട്ട്പുട്ടുകൾ (ഇവിടെ, ഓംലെറ്റ്) ലഭിക്കുന്നു.

എന്നാൽ ഇൻപുട്ടുകൾ ഉപയോഗിച്ച് പ്രവർത്തനങ്ങൾക്കുള്ള നിർദേശങ്ങൾ കൃത്യമല്ല. അവ അവ്യക്തമാണ്. ഉദാഹരണത്തിന്, അഞ്ചാം ഘട്ടത്തിൽ "ഒരു ഭാഗം പൊരിക്കുന്നത്", ആറാം ഘട്ടത്തിൽ "നന്നായി പൊരിക്കുക" തുടങ്ങിയ നിർദേശങ്ങളുടെ വ്യാഖ്യാനം വ്യക്തികൾക്കനുസരിച്ചു വ്യത്യാസപ്പെട്ടിരിക്കും. തന്മൂലം കൃത്യമായ നിർദേശങ്ങളും, സമാന ഇൻപുട്ടുകളുമായി ഒരേ പാചകക്കുറിപ്പ് പിന്തുടരുന്ന വ്യത്യസ്ത വ്യക്തികൾക്ക്, വലുപ്പം, ആകൃതി, രുചി എന്നിവയ്ക്കനുസരിച്ച് വ്യത്യസ്ത ഓംലെറ്റുകൾ ലഭിക്കുന്നു.

കമ്പ്യൂട്ടർ ഉപയോഗിച്ചുള്ള പ്രശ്നങ്ങൾ പരിഹരിക്കുന്നതിനുള്ള നടപടികൾ എഴുതുമ്പോൾ മുകളിൽ പറഞ്ഞ അവ്യക്തതകൾ ഒഴിവാക്കേണ്ടതാണ്.

## a. അൽഗോരിതം (Algorithm)

ചിത്രം 4.4 അബു ജഅഫർ മുഹമ്മദ് ഇബ്നു മുസാ അൽ ഖവാറീസ്മി (780 – 850)

അബു ജഅഫർ മുഹമ്മദ് ഇബ്നു മുസാ അൽ ഖവാറീസ്മി എന്ന അറബി ഗണിതജ്ഞനാണ് അൽഗോരിതം എന്ന വാക്കിന്റെ ഉപജ്ഞാതാവായി അറിയപ്പെടുന്നത്. അദ്ദേഹത്തിന്റെ പേരിന്റെ 'അൽ ഖവാരിസിമി' എന്ന പദത്തിൽ നിന്നാണ് അൽഗോരിതം എന്ന പേര് ലഭിച്ചത്.കമ്പ്യൂട്ടർ പദാവലിയിൽ ഒരു പ്രശ്നം പരിഹരിക്കുന്നതിനുള്ള ക്രമത്തിലുള്ള നിശ്ചിത നിർദേശങ്ങളെ അൽഗോരിതം എന്നു നിർവചിക്കാവുന്നതാണ്. പ്രശ്നം പരിഹരിക്കാനുള്ള ഘട്ടംഘട്ടമായ നടപടിക്രമാണ് അൽഗോരിതം. ഇതിലെ ഓരോ ഘട്ടവും ചെയ്യപ്പെടേണ്ട നിശ്ചിതമായ കൃത്യത്തെ പ്രതിനിധീകരിക്കുന്നു. എന്നിരുന്നാലും, ഒരു അൽഗോരിതം ആകണമെങ്കിൽ, ക്രമത്തിലുള്ള നിർദേശങ്ങൾക്ക് താഴെപ്പറയുന്ന സവിശേഷതകൾ ഉണ്ടായിരിക്കേണ്ടതാണ് :

(i) ഇൻപുട്ടുകൾ സ്വീകരിക്കുന്നതിനുള്ള നിർദേശം (നിർദേശങ്ങൾ) കൊണ്ടായിരിക്കണം അതിന്റെ തുടക്കം. തുടർന്നുള്ള നിർദേശങ്ങൾ വഴി ഈ ഇൻപുട്ട്കൾക്ക് മേൽ പ്രവർത്തനങ്ങൾ നടപ്പിലാക്കുന്നു. ചില സാഹചര്യങ്ങളിൽ, ഉപയോഗിക്കേണ്ട ഡാറ്റ പ്രശ്നത്തിനോടൊപ്പം തന്നെ നൽകിയിരിക്കും. അത്തരം സാഹചര്യങ്ങളിൽ, അൽഗോരിതത്തിന്റെ തുടക്കത്തിൽ ഇൻപുട്ട് സ്വീകരിക്കാനുള്ള നിർദേശങ്ങൾ ഉണ്ടായിരിക്കുകയില്ല .

(ii) ഡാറ്റയെ സൂചിപ്പിക്കാൻ വേരിയബിളുകൾ ഉപയോഗിക്കുക. ഇവിടെ വേരിയബിളുകൾ എന്നതു ഗണിതശാസ്ത്രത്തിലേതു പോലെ അക്ഷരങ്ങളും അക്കങ്ങളും അടങ്ങിയ ഉപയോക്തൃ നിർവചിത വാക്കുകളാണ്. ഡാറ്റ ഇൻപുട്ട് ചെയ്യുന്നതിനും വിലകൾ/ഫലങ്ങൾ സംഭരിക്കുന്നതിനും വേരിയബിളുകൾ തീർച്ചയായും ഉപയോഗിക്കേണ്ടതാണ്.

(iii) ഓരോ നിർദേശവും കൃത്യവും സ്പഷ്ടവും ആയിരിക്കണം. മറ്റൊരു വിധത്തിൽ പറഞ്ഞാൽ, നിർദേശങ്ങൾ അവ്യക്തമായിരിക്കരുത്. മാത്രമല്ല അവ നടപ്പിലാക്കാൻ സാധ്യമായതുമാകണം.

(iv) ഒരു വ്യക്തിക്ക് പേപ്പറും പെൻസിലും ഉപയോഗിച്ച് നിശ്ചിത സമയം കൊണ്ട് ചെയ്തു തീർക്കാവുന്ന തരത്തിൽ അടിസ്ഥാനപരമായതായിരിക്കണം ഓരോ നിർദേശവും.

(v) അൽഗോരിതത്തിൽ പറഞ്ഞിരിക്കുന്ന എല്ലാ നടപടികളും ചെയ്യുവാനുള്ള സമയം നിശ്ചിതമായിരിക്കണം.എന്തെന്നാൽ അൽഗോരിതത്തിലെ ചില നിർദേശങ്ങൾ ആവർത്തിച്ചു നിർവഹിക്കേണ്ടവയായിരിക്കും.അത്തരത്തിലുള്ള ആവർത്തനങ്ങളുടെ എണ്ണം നിശ്ചിതമായിരിക്കണം എന്നാണ് ഇത് സൂചിപ്പിക്കുന്നത്.

(vi) അൽഗോരിത്തിൽ നൽകിയിരിക്കുന്ന നിർദേശങ്ങൾ നിർവഹിച്ചാൽ പ്രതീക്ഷിച്ച ഫലം (ഔട്ട്പുട്ട്) ലഭിച്ചിരിക്കേണ്ടതാണ് .

അൽഗോരിതം സംബന്ധിച്ച ഉൾക്കാഴ്ച നേടാൻ നമുക്ക് ഒരു ലളിതമായ ഉദാഹരണം പരിഗണിക്കാം. തന്നിരിക്കുന്ന മൂന്ന് സംഖ്യകളുടെ ആകത്തുകയും (sum) ശരാശരിയും (average) നമുക്ക് കണ്ടുപിടിക്കണം.ഈ പ്രശ്നം പരിഹരിക്കാനുള്ള നടപടിക്രമം നമുക്ക് താഴെ പറയും പ്രകാരം എഴുതാം:

ഘട്ടം 1: മൂന്ന് സംഖ്യകൾ ഇൻപുട്ട് ചെയ്യുക.

ഘട്ടം 2: ആകത്തുക ലഭിക്കുന്നതിന് ഈ സംഖ്യകൾ കൂട്ടുക.

ഘട്ടം 3: ശരാശരി ലഭിക്കുന്നതിന് ആകത്തുകയെ 3 കൊണ്ട് ഹരിക്കുക.

ഘട്ടം 4: ആകത്തുകയും ശരാശരിയും പ്രിൻറ് ചെയ്യുക.

ഇവിടെ നടപടിക്രമം ശരിയാണെങ്കിലും, ഒരു അൽഗോരിതം തയാറാക്കുമ്പോൾ, ഒരു അംഗീകൃത ഘടന നമ്മൾ പിന്തുടരേണ്ടതുണ്ട്. മുകളിൽ പറഞ്ഞ പ്രക്രിയ ഒരു അൽഗോരിതം രൂപത്തിൽ എങ്ങനെ എഴുതാം എന്ന് നോക്കാം.

**ഉദാഹരണം 4.1: മൂന്ന് സംഖ്യകളുടെ ആകത്തുക,ശരാശരി എന്നിവ കാണാനുള്ള അൽഗോരിതം.**

ഇൻപുട്ട് ചെയ്യുന്ന സംഖ്യകൾ സ്വീകരിക്കാൻ A, B, C എന്ന വേരിയബിളുകൾ ഉപയോഗിക്കുക. അതുപോലെ S ആകത്തുകയ്ക്കു വേണ്ടിയും, Avg ശരാശരിക്കു വേണ്ടിയുമുള്ള വേരിയബിളുകൾ ആക്കുക.

ഘട്ടം 1: ആരംഭിക്കുക.
ഘട്ടം 2: A, B, C ഇൻപുട്ട് ചെയ്യുക.
ഘട്ടം 3: S = A+ B + C.
ഘട്ടം 4: Avg = S / 3.
ഘട്ടം 5: S, Avg പ്രിൻറ് ചെയ്യുക.
ഘട്ടം 6: അവസാനിപ്പിക്കുക

താഴെ പറയുന്ന കാരണങ്ങളാൽ മുകളിൽ പറഞ്ഞിരിക്കുന്ന നിർദേശങ്ങളുടെ കൂട്ടത്തെ അൽഗോരിതം ആയി കണക്കാക്കുന്നു:

- ഇതിന് ഇൻപുട്ട് ഉണ്ട് (ഇൻപുട്ട് ഡാറ്റ സംഭരിക്കാൻ വേരിയബിളുകൾ A, B, C ഉപയോഗിക്കുന്നു).
- ഒരു വ്യക്തിക്ക് പേപ്പറും പെൻസിലും ഉപയോഗിച്ച് കൃത്യമായി നിർവഹിക്കാവുന്ന രൂപത്തിൽ നടപടികൾ കൃത്യമായി പ്രസ്താവിച്ചിരിക്കുന്നു. (ഘട്ടം 3 ലും ഘട്ടം 4 ലും ഉചിതമായ ഓപ്പറേറ്ററുകൾ ഉപയോഗിച്ചുകൊണ്ട്)

- ഓരോ നിർദേശവും അടിസ്ഥാനപരവും അർഥവത്തുമാണ് (ഇൻപുട്ട്, പ്രിന്റ്, കൂട്ടുക, ഹരിക്കുക).
- ഇത് ആകെത്തുക (S), ശരാശരി (Avg) എന്നിങ്ങനെ രണ്ടു ഔട്ട്പുട്ടുകൾ സൃഷ്ടിക്കുന്നു.
- ആരംഭവും അവസാനവും സൂചിപ്പിക്കാനായി തുടങ്ങുക, നിർത്തുക എന്നീ നിർദേശങ്ങൾ ഉപയോഗിച്ചിരിക്കുന്നു.

### നിർദേശങ്ങളുടെ തരങ്ങൾ

നമുക്കറിയാം ഒരു കമ്പ്യൂട്ടറിന് നിർവഹിക്കാൻ കഴിയുന്ന ക്രിയകളുടെ എണ്ണം പരിമിതമാണ്. അതിനാൽ പ്രശ്നപരിഹാരത്തിന് അത്രയും നിർദേശങ്ങൾ മാത്രമേ നമുക്ക് ഉപയോഗിക്കാൻ സാധിക്കുകയുള്ളൂ. കൂടുതൽ അൽഗോരിതങ്ങൾ നിർമിക്കുന്നതിന് മുമ്പ് അൽഗോരിതം നിർമിക്കാൻ ഉപയോഗിക്കുന്ന നിർദേശങ്ങളുടെ തരം ഏതൊക്കെയാണെന്ന് നമുക്ക് നോക്കാം.

- നമ്മൾ നൽകുന്ന ഡാറ്റ കമ്പ്യൂട്ടറിന് സ്വീകരിക്കാൻ സാധിക്കുന്നു.ആയതിനാൽ ഇൻപുടിനുള്ള നിർദേശങ്ങൾ നമുക്ക് ഉപയോഗിക്കാവുന്നതാണ്. ഇൻപുട്ട്, സ്വീകരിക്കുക, വായിക്കുക മുതലായ പദങ്ങൾ നമുക്ക് ഇതിനായി ഉപയോഗിക്കാം.
- കമ്പ്യൂട്ടർ ഫലങ്ങൾ ഔട്ട്പുട്ടായി നൽകുന്നു.ആയതിനാൽ നമുക്ക് ഔട്ട്പുട്ടുമായി ബന്ധപ്പെട്ട നിർദേശങ്ങൾ ഉപയോഗിക്കാം. പ്രിന്റ്, പ്രദർശിപ്പിക്കുക, എഴുതുക മുതലായ പദങ്ങൾ നമുക്ക് ഇതിന് ഉപയോഗിക്കാം.
- ഒരു മെമ്മറി സ്ഥാനത്ത് ഡാറ്റ നേരിട്ട് ശേഖരിക്കാം അല്ലെങ്കിൽ ഡാറ്റ ഒരു സ്ഥാനത്ത് നിന്ന് മറ്റൊന്നിലേക്ക് പകർത്തുകയും ചെയ്യാം. അതുപോലെ, ഡാറ്റയുടെ മുകളിലുള്ള ഗണിത പ്രവർത്തനങ്ങളുടെ ഫലങ്ങൾ മെമ്മറി സ്ഥാനങ്ങളിൽ സംഭരിക്കാം. ഇതിനായി ഗണിതശാസ്ത്രത്തിലേതിനു സമാനമായി വിലനൽകൽ (assignment) (അല്ലെങ്കിൽ സംഭരണം) നിർദേശം നമുക്ക് ഉപയോഗിക്കാം. മൂല്യങ്ങൾ സംഭരിക്കുന്നതിന് വേരിയബിളുകൾക്കു ശേഷം സമം ചിഹ്നം (=) ഉപയോഗിക്കുന്നു. ഇവിടെ വേരിയബിളുകൾ എന്നത് മെമ്മറി സ്ഥാനങ്ങളെ സൂചിപ്പിക്കുന്നു.
- കമ്പ്യൂട്ടറിന് ഡാറ്റ മൂല്യങ്ങൾ താരതമ്യം ചെയ്ത് (ലോജിക്കൽ ഓപ്പറേഷൻ എന്ന് വിളിക്കുന്നു), അതിന്റെ അടിസ്ഥാനത്തിലുള്ള തീരുമാനങ്ങൾ എടുക്കാൻ സാധിക്കും. ഇത്തരം തീരുമാനങ്ങൾ ഒന്നോ അതിലധികമോ പ്രസ്താവനകളുടെ തിരഞ്ഞെടുക്കൽ / ഒഴിവാക്കൽ അല്ലെങ്കിൽ ഒരുകൂട്ടം നിർദേശങ്ങളുടെ ആവർത്തിച്ചുള്ള പ്രവർത്തനം എന്ന രൂപത്തിലായിരിക്കും.

## b. ഫ്ളോച്ചാർട്ടുകൾ (Flowchart)

ഒരു ചിത്രം അല്ലെങ്കിൽ രേഖാചിത്രത്തിന്റെ രൂപത്തിൽ ആവിഷ്കരിക്കപ്പെട്ടിരിക്കുന്ന ഒരു സങ്കല്പം ആണ് എഴുത്ത് രൂപത്തെക്കാൾ ആളുകൾക്ക് സ്വീകാര്യമാകുക. ചില സാഹചര്യങ്ങളിൽ അൽഗോരിതം മനസ്സിലാക്കുക എന്നത് ബുദ്ധിമുട്ടേറിയതായിരിക്കും. എന്തെന്നാൽ ചില സങ്കീർണമായ പ്രവർത്തനങ്ങളും ആവർത്തിച്ചുവരുന്ന ഘട്ടങ്ങളും അതിൽ ഉൾപ്പെട്ടേക്കാം. അതിനാൽ അൽഗോരിതം ചിത്ര രൂപത്തിൽ ആവിഷ്കരിക്കുക എന്നതായിരിക്കും മെച്ചപ്പെട്ട രീതി. നിർദേശങ്ങൾ സൂചിപ്പിക്കുന്നതിനുള്ള പ്രത്യേക ചിഹ്നങ്ങളും പ്രവർത്തനങ്ങളുടെ ക്രമം സൂചിപ്പിക്കുന്നതിന് ആരോകളും ഉപയോഗിച്ച് നിർമിക്കുന്ന അൽഗോരിതത്തിന്റെ ചിത്ര ആവിഷ്കരണമാണ് ഫ്ളോച്ചാർട്ട്. ഒരു അൽഗോരിതം ചിട്ടപ്പെടുത്തുന്നതിനും മനസ്സിലാക്കു

ന്നതിനുമുള്ള ഒരു സഹായിയായിട്ടാണ് പ്രധാനമായിട്ടും ഇത് ഉപയോഗിക്കുന്നത്. വിവിധതര ത്തിലുള്ള നിർദേശങ്ങൾ സൂചിപ്പിക്കുന്നതിനായി അടിസ്ഥാനപരമായ ജ്യാമിതീയ രൂപങ്ങൾ ഫ്ളോച്ചാർട്ടുകളിൽ സാധാരണമായി ഉപയോഗിക്കുന്നു. യഥാർഥ നിർദേശങ്ങൾ ഇത്തരം രൂപങ്ങൾ ക്കുള്ളിൽ കൃത്യവും വ്യക്തവുമായ പ്രസ്താവനകൾ ഉപയോഗിച്ച് എഴുതുന്നു. പ്രവർത്തനങ്ങളുടെ ക്രമത്തെ അതായത് നിർദേശങ്ങൾ പ്രാവർത്തികമാക്കേണ്ട ക്രമത്തെ സൂചിപ്പിക്കുന്ന തരത്തിൽ ഈ രൂപങ്ങൾ ആരോകളോട് കൂടിയ നേർരേഖകൾ വഴി പരസ്പരം ബന്ധപ്പെടുത്തിയിരിക്കുന്നു.

സാധാരണഗതിയിൽ ഒരു അൽഗോരിതത്തെ ഫ്ളോച്ചാർട്ടിലേക്ക് രൂപഭേദം വരുത്തിയ ശേഷമാണ് നിർദേശങ്ങൾ പ്രോഗ്രാമിങ് ഭാഷയിൽ ആവിഷ്കരിക്കുന്നത്. പ്രോഗ്രാം എഴുതുന്നതിൽ ഈ ദ്വിതല (Two step approach) സമീപനത്തിന്റെ മുഖ്യ ഗുണം എന്തെന്നാൽ ഫ്ളോച്ചാർട്ട് വരയ്ക്കുന്ന സമയത്ത് പ്രോഗ്രാം ഭാഷയുടെ വിവിധ ഘടകങ്ങളുടെ വിശദാംശങ്ങളെ പറ്റി വരയ്ക്കുന്നയാൾ ചിന്തിക്കേണ്ടതില്ല. അതിനാൽ അവന്/അവൾക്ക് നടപടിക്രമത്തിന്റെ യുക്തിയിൽ (ഘട്ടംഘട്ടമാ യുള്ള രീതി) കൂടുതൽ ശ്രദ്ധ പതിപ്പിക്കാൻ സാധിക്കുന്നു. മാത്രമല്ല, ഫ്ളോച്ചാർട്ടിൽ, പ്രവർത്ത നങ്ങളുടെ ക്രമം ചിത്രരൂപത്തിൽ സൂചിപ്പിക്കുന്നതിനാൽ നടപടിക്രമത്തിന്റെ യുക്തിയിൽ വരുന്ന തെറ്റുകൾ, പ്രോഗ്രാമിൽ തെറ്റ് കണ്ടെത്തുന്നതിനെക്കാൾ എളുപ്പത്തിൽ കണ്ടെത്താൻ കഴിയുന്നു. അൽഗോരിതവും ഫ്ളോച്ചാർട്ടും എപ്പോഴും പ്രോഗ്രാമറിനുള്ള പ്രമാണം ആകുന്നു. ഒരിക്കൽ ഇവ തയാറാകുകയും പ്രശ്ന പരിഹാരത്തിന്റെ യുക്തി ശരിയാണെന്ന് ബോധ്യമാവുകയും ചെയ്തു കഴിഞ്ഞാൽ പ്രോഗ്രാമർക്ക് പ്രോഗ്രാമിങ് ഭാഷയിലുള്ള വിവിധ നിർമ്മിതികളുടെ സഹായത്തോടെ ചെയ്യേണ്ട പ്രവർത്തനങ്ങളെ കോഡ് ചെയ്യുന്നതിൽ പ്രോഗ്രാമർക്ക് ശ്രദ്ധ ചെലുത്താൻ സാധിക്കുന്നു. പ്രോഗ്രാം തെറ്റുകൾ ഇല്ലാത്തതാണെന്നു ഉറപ്പു വരുത്താൻ സാധാരണ ഗതിയിൽ ഇതിനു കഴിയു ന്നു.

### ഫ്ളോച്ചാർട്ടിലെ ചിഹ്നങ്ങൾ

അംഗീകരിക്കപ്പെട്ട അർഥവത്തായ ചിഹ്നങ്ങൾ ഉപയോഗിക്കുന്നതിലൂടെ ഫ്ളോചാർട്ടുകൾ വഴി യുള്ള പ്രോഗ്രാം യുക്തിയുടെ ആശയ വിനിമയം എളുപ്പമായി തീരുന്നു. അവശ്യമായ ചില പ്രവർത്തനങ്ങൾ സൂചിപ്പിക്കുന്നതിന് ആവശ്യമായ ചില ചിഹ്നങ്ങൾ മാത്രമാണ് നമ്മൾ ഇവിടെ പരിചയപ്പെടുന്നത്. ഈ ചിഹ്നങ്ങൾ അമേരിക്കൻ നാഷണൽ സ്റ്റാൻഡേർഡ്സ് ഇൻസ്റ്റിറ്റ്യൂട്ട് (ANSI) അംഗീകരിച്ചതാണ്.

**1. ടെർമിനൽ (Terminal)**

പേര് സൂചിപ്പിക്കുന്നതുപോലെ, പ്രോഗ്രാം യുക്തിയുടെ ആരംഭത്തെയും അവസാനത്തെയും ഈ ചിഹ്നം സൂചിപ്പിക്കുന്നു. ഒരു ഫ്ളോച്ചാർട്ടിന്റെ ആദ്യത്തെയും അവസാനത്തെയും ചിഹ്നമാണിത്.

ഒരു അണ്ഡവൃത്തത്തിന്റെ (ellipse) ആകൃതിയാണ് ഇതിന്. തുടക്കത്തിൽ ഉപയോഗിക്കുമ്പോൾ നിർഗമനത്തിന്റെ ആരോ ഇതിൽ ഘടിപ്പിച്ചിരിക്കും. പക്ഷേ അവസാനത്തിൽ ഉപയോഗിക്കുമ്പോൾ ആഗമനത്തിന്റെ ആരോ ഇതിൽ ഘടിപ്പിച്ചിരിക്കും.

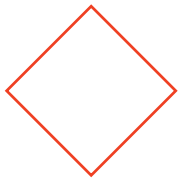
**2. ഇൻപുട്ട്/ഔട്ട്പുട്ട് (input/output)**

ഇൻപുട്ട്/ഔട്ട്പുട്ട് ചിഹ്നമായി ഉപയോഗിക്കുന്നത് ഒരു സമാന്തരഭുജ (Parallelogram) മാണ്. പ്രോഗ്രാമിൽ ഒരു ഇൻപുട്ട് /ഔട്ട്പുട്ട് ഉപകരണത്തിന്റെ ധർമ്മത്തെ ഇത് സൂചിപ്പിക്കുന്നു. എല്ലാ ഇൻപുട്ട്/ഔട്ട്പുട്ട് നിർദേശങ്ങളും ഈ ചിഹ്നം ഉപയോഗിച്ചാണ് ആവിഷ്കരിച്ചിരിക്കുന്നത്. ഇതിലേക്ക് ഒരു ആഗമന ആരോയും ഒരു നിർഗമന ആരോയും ഘടിപ്പിച്ചിരിക്കും.

**3. പ്രവർത്തനം (process)**

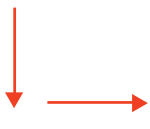
പ്രവർത്തന ഘട്ടത്തെ പ്രതിനിധീകരിക്കാൻ ദീർഘചതുരം ഉപയോഗിക്കുന്നു. സങ്കലനം, വ്യവകലനം ഗുണിനം ഹരണം തുടങ്ങിയ ഗണിത ക്രിയകൾ ചെയ്യാനും അതുപോലെ വേരിയബിളിയിലേക്ക് വില നൽകുവാനും ഈ ചിഹ്നം ഉപയോഗിക്കുന്നു. വേരിയബിളുകൾക്ക് വില നൽകുക (assignment) എന്നത് കൊണ്ടുദ്ദേശിക്കുന്നത്- ഒരു മെമ്മറി സ്ഥാനത്തു നിന്ന് മറ്റൊന്നിലേക്ക് ഡാറ്റ പകർത്തുന്നതോ (ഉദാ:a=b) അല്ലെങ്കിൽ എ.ൽ.യു. വിൽ (ALU) നിന്നും ഔട്ട്പുട്ടിനെ മെമ്മറി സ്ഥാനത്തേക്ക് പകർത്തുന്നതോ (ഉദാ:a=b+5) അല്ലെങ്കിൽ ഒരുപക്ഷേ മെമ്മറി സ്ഥാനത്തേക്ക് വില നേരിട്ട് സംഭരിക്കുന്നതോ (ഉദാ:a=2) ആകാം. പ്രോസസ്സ് ചിഹ്നത്തിനു ഒരു ആഗമന ആരോയും ഒരു നിർഗമന ആരോയും ഉണ്ടായിരിക്കും.

**4. തീരുമാനം (Decision)**



തീരുമാനങ്ങൾ സൂചിപ്പിക്കുന്നതിനായുള്ള ചിഹ്നമായി ചതുർഭുജം ഉപയോഗിക്കുന്നു. ഇത് തീരുമാനങ്ങൾ കൈക്കൊള്ളേണ്ട ഒരു ഘട്ടത്തെയാണ് സൂചിപ്പിക്കുന്നത്. ഈ ഘട്ടത്തിൽ നിന്ന് ഒന്നോ അതിലധികമോ ഇതര ഘട്ടങ്ങളിലേക്ക് വിഭജനം സാധ്യമാണ്. എല്ലാ നിർഗമന പാതകളും ഇവിടെ പരാമർശിച്ചിട്ടുണ്ടായിരിക്കും. എന്നിരുന്നാലും ഒരു വ്യവസ്ഥ (condition) പരിശോധിച്ച് അതിന്റെ ഫലത്തിന്റെ അടിസ്ഥാനത്തിൽ അടുത്ത ഒരു പാത മാത്രമേ തിരഞ്ഞെടുക്കപ്പെടുകയുള്ളൂ. സാധാരണഗതിയിൽ ഈ ചിഹ്നത്തിന് ഒരു ആഗമന മാർഗവും 2 നിർഗമന മാർഗങ്ങളും ഉണ്ടായിരിക്കും. ഒന്ന് വ്യവസ്ഥയുടെ ഫലം ശരിയാണെങ്കിൽ ചെയ്യേണ്ടുന്ന പ്രവൃത്തിയുടെ നേർക്കും, മറ്റേത് ഇതര മാർഗത്തിലേക്കും.

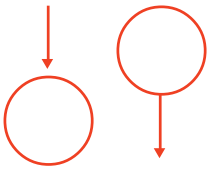
**5. ഫ്ളോ ലൈനുകൾ (flow lines)**



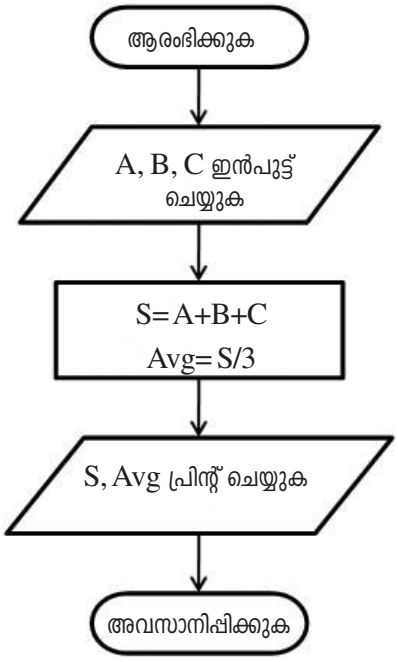
പ്രവർത്തന ക്രിയകളുടെ ഒഴുക്കിനെ സൂചിപ്പിക്കാൻ ആരോകളോട് കൂടിയ ഫ്ളോ ലൈനുകൾ ഉപയോഗിക്കുന്നു. അതായത് നിർദേശങ്ങൾ പ്രാവർത്തികമാക്കേണ്ട കൃത്യമായ ക്രമത്തെ ഇവ സൂചിപ്പിക്കുന്നു. സാധാരണഗതിയിൽ ഫ്ളോ ലൈനുകളുടെ ദിശ മുകളിൽ നിന്നും താഴേക്കും ഇടത്തുനിന്നും വലത്തേക്കും ആയിരിക്കും. എന്നാൽ ചില സാഹചര്യങ്ങളിൽ ഇത് വലതു നിന്ന് ഇടത്തേക്കും താഴെനിന്നും മുകളിലേക്കും ആവാം. ഫ്ളോ ലൈനുകൾ

പരസ്പരം ഛേദിക്കുന്നത് നല്ല പ്രവണതയല്ല. അത്തരം ഛേദനങ്ങൾ പരമാവധി ഒഴിവാക്കേണ്ടതാണ്.

### 6. കണക്ടർ (connector)



ഫ്ളോച്ചാർട്ടുകളുടെ വലിപ്പം കൂടുമ്പോൾ ഫ്ളോ ലൈനുകൾ പല ഭാഗങ്ങളിലും പരസ്പരം ഛേദിക്കുകയും പല സ്ഥലങ്ങളിലും ആശയക്കുഴപ്പം വരുത്തുകയും തൻമൂലം ഫ്ളോച്ചാർട്ടിന്റെ ഗ്രഹണം ബുദ്ധിമുട്ടാക്കുകയും ചെയ്യുന്നു. അതുപോലെ ഒരു പേജിൽ ഒതുങ്ങുന്നതിനെക്കാൾ വലിയ ഫ്ളോച്ചാർട്ട് ആകുമ്പോൾ ഫ്ളോ ലൈനുകളുടെ ഉപയോഗം അസാധ്യമായിത്തീരുന്നു. ഒരു ഫ്ളോച്ചാർട്ട് സങ്കീർണമാകുകയും ഫ്ളോ ലൈനുകളുടെ എണ്ണം, ദിശ എന്നിവയെക്കുറിച്ച് ആശയക്കുഴപ്പം സംഭവിക്കുകയോ അല്ലെങ്കിൽ ഫ്ളോച്ചാർട്ട് ഒന്നിൽ അധികം പേജുകളിലായി പടരുകയോ ചെയ്താൽ മുറിഞ്ഞുപോയ ഫ്ളോ ലൈനുകളെ തമ്മിൽ കൂട്ടിയോജിപ്പിക്കാൻ ഒരു ജോടി കണക്ടർ ചിഹ്നങ്ങൾ ഉപയോഗിക്കാം. ഫ്ളോച്ചാർട്ടിന്റെ ഒരു ഭാഗത്ത് നിന്നുമുള്ള ആഗമനത്തെ അല്ലെങ്കിൽ അതിന്റെ മറ്റൊരു ഭാഗത്തേക്കുള്ള നിർഗമനത്തെ ഈ ചിഹ്നം സൂചിപ്പിക്കുന്നു. ഒരു അക്കം അല്ലെങ്കിൽ അക്ഷരത്തോട് കൂടിയ ഒരു വൃത്തം ഉപയോഗിച്ചാണ് കണക്ടർ ചിഹ്നം സൂചിപ്പിക്കുന്നത്. ഒരേപോലെ അടയാളപ്പെടുത്തിയ ഒരു ജോഡി കണക്ടർ ചിഹ്നങ്ങൾ അൽഗോരിതത്തിന്റെ തുടർച്ചയെ സൂചിപ്പിക്കുന്നു. ഒരു വലിയ ഫ്ളോ ലൈനിന് പകരം ഒരേ ചിഹ്നങ്ങൾ / അക്കങ്ങൾ ഉള്ള രണ്ടു കണക്ടറുകൾ ഉപയോഗിക്കാം. അതായത് ഒരേപോലെ അടയാളപ്പെടുത്തിയ ഒരു ജോടി കണക്ടറുകളിൽ ഒന്ന് ഫ്ളോച്ചാർട്ടിന്റെ മറ്റൊരു ഭാഗത്തേക്കുള്ള നിർഗമനത്തെയും, രണ്ടാമത്തേത് ഫ്ളോച്ചാർട്ടിന്റെ മറ്റൊരു ഭാഗത്തേക്കുള്ള ആഗമനത്തെയും സൂചിപ്പിക്കുന്നു.




*ചിത്രം 4.5 ആകെതുകയും ശരാശരിയും കാണാനുള്ള ഫ്ളോച്ചാർട്ട്*

ഉദാഹരണം 4.1 ൽ വിശദീകരിച്ച പ്രശ്നത്തിന്റെ ഫ്ളോച്ചാർട്ട് ചിത്രം 4.5 ൽ കാണിച്ചിരിക്കുന്നു.

അൽഗോരിതത്തിലെ ഓരോ ഘട്ടത്തിലെയും നിർദേശങ്ങൾ സൂചിപ്പിക്കാൻ ഉചിതമായ ചിഹ്നങ്ങൾ ഉപയോഗിച്ചിരിക്കുന്നു.മാത്രമല്ല ഓരോ ചിഹ്നത്തിലും അതിനനുസൃതമായ നിർദേശങ്ങൾ രേഖപ്പെടുത്തുന്നു. പ്രവർത്തനങ്ങളുടെ ക്രമം ഫ്ളോ ലൈനുകൾ ഉപയോഗിച്ച് കൃത്യമായി അടയാളപ്പെടുത്തുകയും ചെയ്യുന്നു.

## ഫ്ളോച്ചാർട്ടിന്റെ ഗുണങ്ങൾ

പ്രോഗ്രാം ആസൂത്രണത്തിൽ പലരീതികളിലും ഫ്ളോച്ചാർട്ടുകൾ ഗുണപ്രദങ്ങളാണ്.

- **മികച്ച ആശയവിനിമയം :** ഫ്ളോച്ചാർട്ട് ഒരു പ്രോഗ്രാമിന്റെ ചിത്രരൂപത്തിലുള്ള സൂചകം, ആയതിനാൽ ഒരു പ്രോഗ്രാമർക്ക് മറ്റൊരു പ്രോഗ്രാമറിന് പ്രോഗ്രാമിന്റെ യുക്തി ഫ്ളോച്ചാർട്ട് ഉപയോഗിച്ച് വിശദീകരിച്ചു കൊടുക്കുന്നത് പ്രോഗ്രാം വിശദീരിക്കുന്നതിനേക്കാൾ എളുപ്പമാണ്.
- **ഫലപ്രദമായ വിശകലനം :** പ്രോഗ്രാമിലെ വിവിധ ഘട്ടങ്ങൾ കൃത്യമായി ഫ്ളോച്ചാർട്ടിൽ പ്രതിപാദിച്ചിരിക്കുന്നതിനാൽ,പ്രോഗ്രാമിനെ ഫലപ്രദമായി വിശകലനം ചെയ്യാൻ ഫ്ളോച്ചാർട്ട് സഹായിക്കുന്നു.
- **ഫലപ്രദമായ സമന്വയം :** പ്രോഗ്രാമിനെ വിവിധ ഘടകങ്ങളായി തിരിക്കുകയും അവ ഓരോന്നിന്റെയും പരിഹാരം ഫ്ളോച്ചാർട്ടുകളായി പ്രത്യേകം പ്രത്യേകമായി തയാറാക്കുകയും ചെയ്താൽ,അവയെ എല്ലാം കൂട്ടി യോജിപ്പിച്ചു മൊത്തത്തിലുള്ള സിസ്റ്റത്തിന്റെ രൂപരേഖ നമുക്ക് തയാറാക്കാവുന്നതാണ്.
- **ഫലപ്രദമായ കോഡിങ് :** ഫ്ളോച്ചാർട്ട് തയ്യാറാക്കി കഴിഞ്ഞാൽ പ്രോഗ്രാമർക്കു അനുബന്ധ പ്രോഗ്രാം തയ്യാറാക്കാൻ എളുപ്പമാണ്, എന്തെന്നാൽ ഫ്ളോച്ചാർട്ട് പ്രോഗ്രാമിന്റെ ഒരു രേഖാ ചിത്രമായി പ്രവർത്തിക്കുന്നു. പ്രോഗ്രാമിന്റെ തുടക്കം മുതലുള്ള എല്ലാ ഘട്ടങ്ങളിലൂടെയും കടന്നു പോയി ,ഒന്ന് പോലും വിട്ടുപോകാതെ അവസാനം വരെയും എത്തിച്ചേരുവാനുള്ള ഒരു സഹായിയായി ഇത് വർത്തിക്കുന്നു.

## ഫ്ളോച്ചാർട്ടിന്റെ പരിമിതികൾ

ഫ്ളോച്ചാർട്ടുകൾക്ക് ഇത്തരത്തിലുള്ള ഗുണങ്ങൾ എടുത്തു പറയാമെങ്കിലും ,ചില പരിമിതികളും അവയ്ക്കുണ്ട്.

- ഉചിതമായ ചിഹ്നങ്ങളും സ്പേസും നൽകിയുള്ള ഫ്ളോച്ചാർട്ട് നിർമാണം സമയം ചെലവഴിച്ചു ചെയ്യേണ്ടതും കഠിനാധ്വാനം ആവശ്യമായതുമാണ്, പ്രത്യേകിച്ചും സങ്കീർണ്ണമായ അൽഗോരിതങ്ങൾ ആണെങ്കിൽ.
- അൽഗോരിതത്തിന്റെ യുക്തിയിലുള്ള വളരെച്ചെറിയ മാറ്റത്തിനുപോലും പുതിയ ഫ്ളോച്ചാർട്ട് ആവശ്യമായി വരുന്നു.
- ഫ്ളോച്ചാർട്ടിൽ ഉൾപ്പെടുത്തേണ്ട വിശദശാംശങ്ങളെ പറ്റി വിശദീകരിക്കുന്ന ഒരു തരത്തിലുള്ള മാനദണ്ഡങ്ങളും നിലവിലില്ല.

വിവിധ പ്രശ്നങ്ങൾ പരിഹരിക്കുന്നതിനുള്ള അൽഗോരിതങ്ങളും ഫ്ളോച്ചാർട്ടുകളും നമുക്ക് വികസിപ്പിക്കാം.

### ഉദാഹരണം 4.2: ഒരു ചതുരത്തിന്റെ വിസ്തീർണവും ചുറ്റളവും കണ്ടെത്തുക

ചതുരത്തിന്റെ നീളവും വീതിയും ലഭ്യമായാൽ ഈ പ്രശ്നം നമുക്കു പരിഹരിക്കാം. താഴെ പറയുന്ന സമവാക്യം ഇതിന് വേണ്ടി ഉപയോഗിക്കാവുന്നതാണ്.

ചുറ്റളവ് = 2 x (നീളം + വീതി), വിസ്തീർണം = (നീളം x വീതി).

L, B എന്നീ വേരിയബിളുകൾ നീളം വീതി എന്നി സൂചിപ്പിക്കാനും P, A എന്നീ വേരിയബിളുകൾ വിസ്തീർണം, ചുറ്റളവ് എന്നിവ സൂചിപ്പിക്കാനും ഉപയോഗിക്കുന്നു എന്നിരിക്കട്ടെ

ഘട്ടം 1 : തുടങ്ങുക

ഘട്ടം 2 : L , B ഇൻപുട്ടായി സ്വീകരിക്കുക

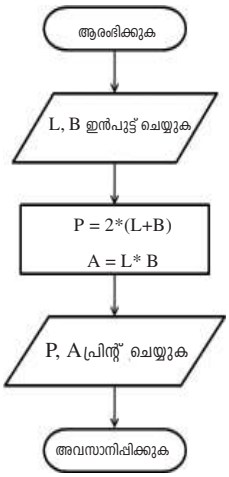
ഘട്ടം 3 : P = 2 *(L +B )

ഘട്ടം 4 : A = L *B

ഘട്ടം 5 : P ,A പ്രിൻറ് ചെയ്യുക.

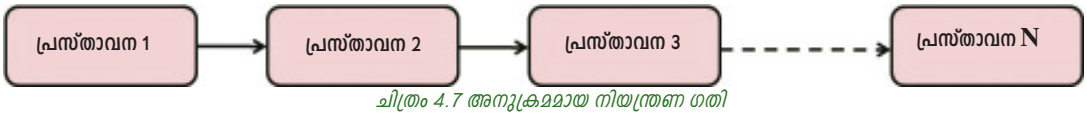
ഘട്ടം 6. അവസാനിപ്പിക്കുക.

ചിത്രം 4 .6 ൽ ഇതിനുള്ള ഫ്ളോച്ചാർട്ട് നൽകിയിരിക്കുന്നു.




*ചിത്രം 4.6 വിസ്തീർണം ചുറ്റളവ് കാണാനുള്ള ഫ്ളോച്ചാർട്ട്*

ഉദാഹരണം 4.1 ലും 4.2 ലും വികസിപ്പിച്ചെടുത്തിരിക്കുന്ന അൽഗോരിതങ്ങൾക്ക് ഓരോന്നിലും ആറു വീതം നിർദേശങ്ങളാണുള്ളത്. ചിത്രം 4.7 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ ഈ രണ്ടു സന്ദർഭങ്ങളിലും നിർദേശങ്ങൾ ഓരോന്നും അനുക്രമമായ രീതിയിലാണ് പ്രവർത്തിക്കുക. നിർദേശങ്ങളുടെ പ്രവർത്തനക്രമത്തെ നിയന്ത്രണഗതി (Flow of Cotnrol) എന്ന് പറയുന്നു. അപ്രകാരം മുകളിൽപ്പറഞ്ഞ രണ്ട് അൽഗോരിതങ്ങളും അനുക്രമമായ നിയന്ത്രണ ഗതിയാണ് പിന്തുടരുന്നതെന്നു നമുക്ക് പറയാവുന്നതാണ്.




*ചിത്രം 4.7 അനുക്രമമായ നിയന്ത്രണ ഗതി*

*സെക്കന്റുകളായി സമയം ഇൻപുട്ട് ആയി സ്വീകരിച്ച് Hr:min:Sec രൂപത്തിൽ ലഭിക്കുവാനുള്ള അൽഗോരിതവും ഫ്ളോച്ചാർട്ടും വികസിപ്പിക്കുക. (ഉദാഹരണത്തിന് 3700 എന്ന് ഇൻപുട്ട് നൽകിയാൽ 1 Hr : 1 Min : 40 sec എന്ന ഔട്ട്പുട്ട് ലഭിക്കണം)*

### ഉദാഹരണം 4.3 രണ്ടു വിദ്യാർഥികളിൽ ഉയരം കൂടിയ ആളുടെ ഉയരം കണ്ടെത്തുക

ഇവിടെ,രണ്ടു വിദ്യാർത്ഥികളുടെ ഉയരത്തെ പ്രതിനിധീകരിക്കുന്ന രണ്ടു സംഖ്യകൾ ഇൻപുട്ട് ആയി സ്വീകരിക്കേണ്ടതാണ്. അവയിലെ വലിയ സംഖ്യയാണ് ഉത്തരമായിട്ടു പരിഗണിക്കുക. ഇതിനായി ഈ സംഖ്യകളെ താരതമ്യം ചെയ്യേണ്ടതുണ്ടെന്നു നമുക്കറിയാവുന്നതാണ്. അൽഗോരിതം താഴെ നൽകിയിരിക്കുന്നു.

ഘട്ടം 1 : തുടങ്ങുക

ഘട്ടം 2 : H1 ,H2 ഇൻപുട്ട് ആയി സ്വീകരിക്കുക

ഘട്ടം 3 : അഥവാ H1 >H2 ആണെങ്കിൽ

ഘട്ടം 4 : H1 പ്രിന്റ് ചെയ്യുക

ഘട്ടം 5 : അല്ലെങ്കിൽ

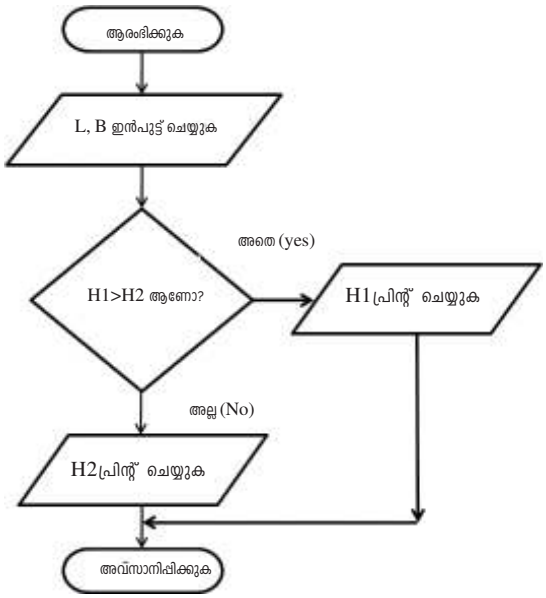
ഘട്ടം 6 : H2 പ്രിന്റ് ചെയ്യുക

ഘട്ടം 7 : പരിശോധന അവസാനിക്കുന്നു

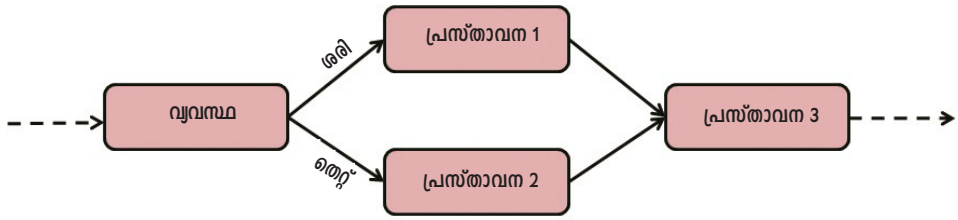
ഘട്ടം 8 : അവസാനിപ്പിക്കുക

ഈ അൽഗോരിതത്തിന്റെ ഫ്ളോച്ചാർട്ട് ചിത്രം 4.8 ൽ നൽകിയിരിക്കുന്നു. തീരുമാനങ്ങൾ കൈകൊള്ളാനുള്ള സങ്കേതം ഉപയോഗപെടുത്തുന്ന ഒരു അൽഗോരിതമാണിത്. ഘട്ടം 3 ൽ ഒരു

നിബന്ധന പരിശോധിക്കുന്നു. H1 ,H2 എന്നിവയുടെ വിലകളുടെ അടിസ്ഥാനത്തിൽ അതിന്റെ ഉത്തരം തീർച്ചയായും ശരി അല്ലെങ്കിൽ തെറ്റ് എന്നായിരിക്കും. തീരുമാനം കൈക്കൊള്ളുന്നത് ഈ നിബന്ധനയുടെ ഉത്തരത്തിനെ അടിസ്ഥാനമായിട്ടായിരിക്കും. ഉത്തരം ശരി എന്നാണെങ്കിൽ ഘട്ടം 4 പ്രവർത്തിക്കും അല്ലെങ്കിൽ ഘട്ടം 6 ആണ് പ്രവർത്തിക്കുക. ഇവിടെ ഒരു നിബന്ധനയുടെ അടിസ്ഥാനത്തിൽ രണ്ടു പ്രസ്താവനകളിൽ ഏതെങ്കിലും ഒന്നാണ് (ഘട്ടം 4 അല്ലെങ്കിൽ ഘട്ടം 6) പ്രവർത്തനത്തിനായി തിരഞ്ഞെടുക്കപ്പെടുക. ഘട്ടം 3 ൽ പ്രോഗ്രാം രണ്ടു ശാഖകളായി പിരിയുന്നു. അതായത് പ്രശ്നം പരിഹരിക്കുന്നതിന് വേണ്ടി ഈ അൽഗോരിതം ഒരു തിരഞ്ഞെടുക്കൽ ഘടന ഉപയോഗപ്പെടുത്തുന്നു. ചിത്രം 4.9 ൽ കാണിച്ചിരിക്കുന്നത് പോലെ നിബന്ധനയുടെ ഉത്തരത്തിനനുസരിച്ചു പ്രവർത്തനത്തിന്റെ ഗതി രണ്ടിൽ ഏതെങ്കിലും ഒരു പ്രസ്താവനയിലേക്ക് വിഘടിച്ചു പോകുന്നു.

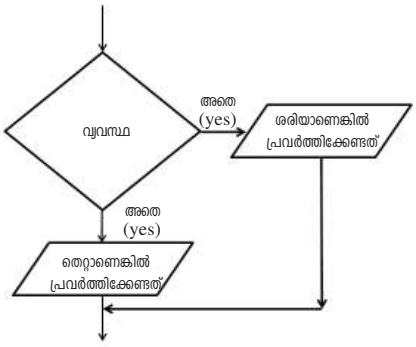



*ചിത്രം 4.8 ഉയർന്ന വില കാണാനുള്ള ഫ്ളോച്ചാർട്ട്*




*ചിത്രം 4.9 തിരഞ്ഞെടുക്കൽ ഘടന*

തിരഞ്ഞെടുക്കൽ നിർമ്മിതിയുടെ പ്രവർത്തനം ചിത്രം 4 .10 ൽ കൊടുത്തിരിക്കുന്നു. നിയന്ത്രണ ഗതി നിബന്ധനയിലേക്കു വരികയും നിബന്ധന ശരി അല്ലെങ്കിൽ തെറ്റ് എന്ന് വിലയിരുത്തപ്പെടുകയും ചെയ്യുന്നു. നിബന്ധനയുടെ ഫലം ശരി എന്നാണെങ്കിൽ, ശരിയായാൽ പ്രവർത്തിക്കേണ്ട നിർദേശങ്ങളുടെ കൂട്ടം നടപ്പിലാക്കുകയും, തെറ്റായാൽ പ്രവർത്തിക്കേണ്ട നിർദേശങ്ങളുടെ കൂട്ടത്തെ ഒഴിവാക്കുകയും ചെയ്യുന്നു. നിബന്ധനയുടെ ഫലം തെറ്റ് ആണെങ്കിൽ തെറ്റായാൽ പ്രവർത്തിക്കേണ്ട നിർദേശങ്ങളുടെ കൂട്ടം നടപ്പിലാക്കുകയും ശരിയായാൽ പ്രവർത്തിക്കേണ്ട നിർദേശങ്ങളുടെ കൂട്ടാത്തെ ഒഴിവാക്കുകയും ചെയ്യുന്നു. ഇനി നമുക്ക് മറ്റൊരു പ്രശ്നം പരിഹരിക്കാം.




*ചിത്രം 4.10 തിരഞ്ഞെടുക്കലിന്റെ ഫ്ളോച്ചാർട്ട്*

**ഉദാഹരണം 4.4 : 3 യൂണിറ്റ് ടെസ്റ്റുകളിൽ ലഭിച്ച സ്കോറുകൾ ഇൻപുട്ട് ചെയ്ത ഏറ്റവും ഉയർന്ന സ്കോർ കണ്ടെത്തുക**

ഇവിടെ സ്കോറുകൾ സൂചിപ്പിക്കുന്നതിനു വേണ്ടി മൂന്നു സംഖ്യകൾ നൽകുകയും അവയിൽ ഏറ്റവും വലിയ സംഖ്യ കണ്ടുപിടിക്കുകയും ചെയ്യുന്നു. ഇതിന്റെ അൽഗോരിതം താഴെ കൊടുത്തിരിക്കുന്നു. ചിത്രം 4.11 ൽ ഫ്ളോച്ചാർട്ട് പ്രദർശിപ്പിച്ചിരിക്കുന്നു.

ഘട്ടം 1 : ആരംഭിക്കുക

ഘട്ടം 2 : M 1, M 2 , M 3 എന്നീ സംഖ്യകൾ ഇൻപുട്ട് ആരംഭിക്കുക
ചെയ്യുക .

ഘട്ടം 3 : അഥവാ M 1 > M 2 ഉം M 1 > M 3 ആണെങ്കിൽ

ഘട്ടം 4 : M 1 പ്രിന്റ് ചെയ്യുക

ഘട്ടം 5 : അല്ലെങ്കിൽ M 2 >M 3 ആണെങ്കിൽ

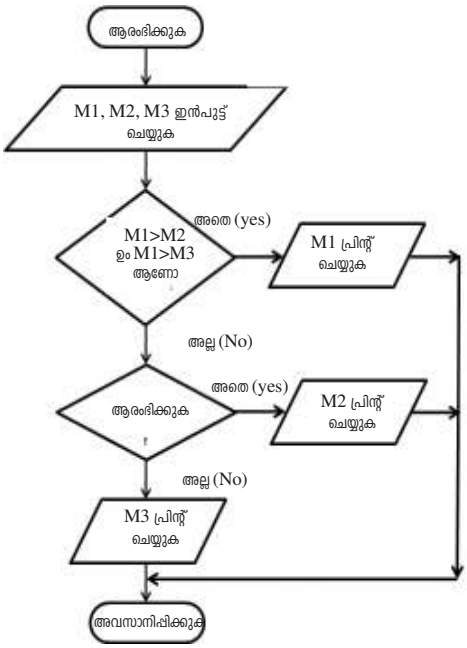
ഘട്ടം 6 : M 2 പ്രിന്റ് ചെയ്യുക

ഘട്ടം 7 : അല്ലെങ്കിൽ

ഘട്ടം 8 : M 3 പ്രിൻറ് ചെയ്യുക.

ഘട്ടം 9 : വ്യവസ്ഥാ പരിശോധനയുടെ അവസാനം

ഘട്ടം 10 : അവസാനിപ്പിക്കുക




ചിത്രം 4.11 മൂന്ന് സംഖ്യകളിൽ ഏറ്റവും വലിയ സംഖ്യ കാണാനുള്ള ഫ്ളോച്ചാർട്ട്

വ്യത്യസ്തമായ നിബന്ധനകളുടെ അടിസ്ഥാനത്തിൽ അൽഗോരിതത്തിൽ ഒന്നിലധികം തിരഞ്ഞെടുക്കൽ നിർമിതകൾ ഉപയോഗിക്കുന്നു. ഇവിടെ വ്യത്യസ്തമായ മൂന്നു പ്രവൃത്തികൾ നൽകിയിരിക്കുന്നു. എന്നാൽ അവയിൽ ഒന്ന് മാത്രമേ പ്രവർത്തിക മാകുകയുള്ളൂ. ശ്രദ്ധിക്കേണ്ട മറ്റൊരു വസ്തുത, ഇവിടെ ആദ്യത്തെ നിബന്ധനയിൽ രണ്ടു താര തമ്യങ്ങൾ അടങ്ങിയിരിക്കുന്നു എന്നതാണ്. ഇത്തരം നിബന്ധനകൾ സംയുക്ത നിബന്ധനകൾ (Compound conditions) എന്ന് പറയുന്നു.

ചെയ്തു നോക്കൂ

1. *തന്നിരിക്കുന്ന സംഖ്യ ഒറ്റയാണോ ഇരട്ടയാണോ എന്ന് പരിശോധിക്കാനുള്ള അൽഗോരിതം തയാറാക്കുക. ഫ്ളോച്ചാർട്ട് വരയ്ക്കുക.*
2. *ദിവസത്തെ സൂചിപ്പിക്കുന്ന സംഖ്യ ഇൻപുട്ട് ആയി നൽകിയാൽ ദിവസത്തിന്റെ പേര് പ്രദർശിപ്പിക്കാനുള്ള അൽഗോരിതവും ഫ്ളോച്ചാർട്ടും തയാറാക്കുക. (ഉദാഹരണത്തിന് 1 ഇൻപുട്ട് നൽകിയാൽ ഔട്ട്പുട്ട് Sunday എന്നായിരിക്കണം. അഥവാ 2 ആണ് ഇൻപുട്ടെങ്കിൽ ഔട്ട്പുട്ട് Monday എന്നായിരിക്കണം. 1 മുതൽ 7 വരെയുള്ള സംഖ്യ അല്ല ഇൻപുട്ടെങ്കിൽ 'INVALID DATA' എന്നായിരിക്കേണ്ടതാണ്.*
3. *പത്താം തരത്തിലെ മൂല്യനിർണയ വ്യവസ്ഥയുടെ അടിസ്ഥാനത്തിൽ ഒരു സ്കോർ (പരമാവധി 100) സ്വീകരിച്ചു ഗ്രേഡ് കാണാനുള്ള അൽഗോരിതം തയാറാക്കുക.*

ഒരു പ്രവൃത്തി തന്നെ ആവർത്തിച്ചു നിർവഹിക്കേണ്ട ഒരു സാഹചര്യം പരിഗണിക്കുക. ഉദാഹരണത്തിന് ആദ്യത്തെ 100 എണ്ണൽ സംഖ്യകൾ പ്രിൻറ് ചെയ്യണമെന്നിരിക്കട്ടെ എങ്ങനെയാണ് നമുക്കത് ചെയ്യാൻ സാധിക്കുക?നമുക്കറിയാം ആദ്യത്തെ സംഖ്യ 1 ആണ് .അത് പ്രിന്റ് ചെയ്യേണ്ടതാണ്.ആദ്യത്തെ സംഖ്യയോട് 1 കൂട്ടിയാൽ അടുത്ത സംഖ്യ ലഭിക്കുന്നു. അതും പ്രിന്റ് ചെയ്യണം. ഇതിൽ നിന്ന് ഒരു കാര്യം വ്യക്തമാണ്, സംഖ്യ പ്രിൻറ് ചെയ്യുക സംഖ്യയോടു 1 കൂട്ടുക എന്നീ പ്രവൃത്തികൾ ആവർത്തിച്ച് ചെയ്യേണ്ടവയാണ്. അവസാനത്തെ സംഖ്യ പ്രിൻറ് ചെയ്തു കഴിഞ്ഞാൽ പ്രവർത്തനം അവസാനിപ്പിക്കേണ്ടതാണ് .ഇതിനു വേണ്ടിയുള്ള ഒരു അൽഗോരിതം നമുക്ക് തയാറാക്കാം.

### ഉദാഹരണം 4.5: 1 മുതൽ 100 വരെയുള്ള സംഖ്യകൾ പ്രിന്റ് ചെയ്യാൻ

ഘട്ടം 1 : ആരംഭിക്കുക

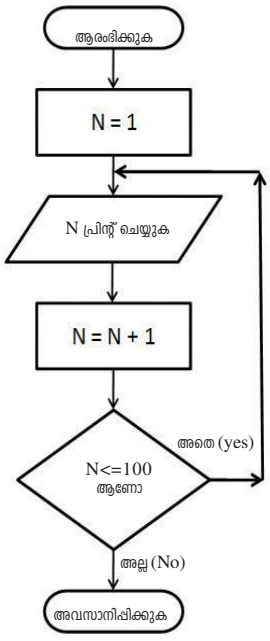
ഘട്ടം 2 : N =1

ഘട്ടം 3 : N പ്രിന്റ് ചെയ്യുക

ഘട്ടം 4 : N =N + 1

ഘട്ടം 5 : അഥവാ N <=100 ആണെങ്കിൽ
ഘട്ടം 3 ലേക്ക് പോകുക

ഘട്ടം 6 : അവസാനിപ്പിക്കുക




*ചിത്രം 4.12 1 മുതൽ 100 വരെയുള്ള സംഖ്യകൾ പ്രിന്റ് ചെയ്യാനുള്ള ഫ്ളോച്ചാർട്ട്*

ഉദാഹരണം 4.5ൽ കൊടുത്തിരിക്കുന്ന അൽഗോരിതത്തിൽ ഘട്ടം 5 ൽ ഒരു വ്യവസ്ഥ പരിശോധിക്കുന്നു. അഥവാ വ്യവസ്ഥ ശരിയാണെങ്കിൽ നിയന്ത്രണ ഗതി ഘട്ടം 3 ലേക്ക് തിരിച്ചു പോകുന്നു.അതു കാരണം വ്യവസ്ഥ ശരിയായിരിക്കുന്നതു വരെ ഘട്ടം 3 ,ഘട്ടം 4 ,ഘട്ടം 5 എന്നിവ ആവർത്തിച്ചു പ്രവർത്തിച്ചു കൊണ്ടിരിക്കും.ഇവിടെ ഒരു ലൂപ്പ് രൂപം കൊണ്ടതായി നമുക്ക് പറയാം.ഘട്ടം 3 ,4,5 എന്നിവ ചേർന്നതാണ് ആ ലൂപ്പ്.വ്യവസ്ഥ തെറ്റാകുമ്പോൾ മാത്രമെ നിയന്ത്രണം ലൂപ്പിനു പുറത്തേക്കു വരുകയുള്ളൂ.ഈ അൽഗോരിതത്തിന്റെ ഫ്ളോച്ചാർട്ട് ചിത്രം 4.12 ൽ കാണിച്ചിരിക്കുന്നു.

മുകളിൽപ്പറഞ്ഞ അൽഗോരിതത്തെ താഴെ പറയും പ്രകാരം ലഘൂകരിക്കാം

ഘട്ടം 1 : ആരംഭിക്കുക

ഘട്ടം 2 : N = 1

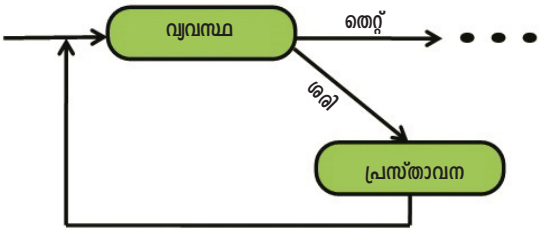
ഘട്ടം 3 : N <100 ആയിരിക്കുന്നത് വരെ ഘട്ടം 4 ഉം 5 ഉം ആവർത്തിക്കുക

ഘട്ടം 4 : N പ്രിന്റ് ചെയ്യുക

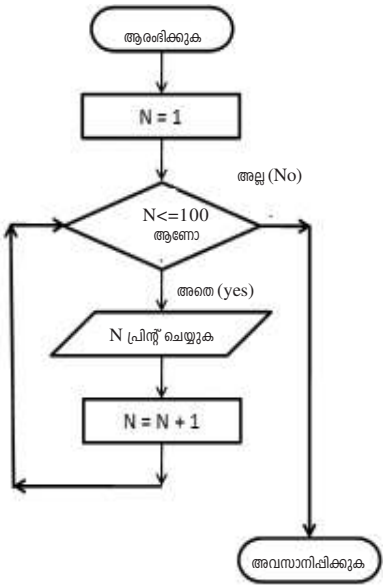
ഘട്ടം 5 : N = N + 1

ഘട്ടം 6 : അവസാനിപ്പിക്കുക

ഘട്ടം 3 ൽ 'ആവർത്തിക്കുക', 'ആയിരിക്കുന്നത് വരെ' മുതലായ വാക്കുകൾ ലൂപ്പ് നിർമ്മിക്കാൻ ഉപയോഗിക്കുന്നു. ആവർത്തിച്ചു പ്രവർത്തിക്കേണ്ട പ്രസ്താവനകൾ 'ആവർത്തിക്കുക' എന്ന വാക്കിന്റെ കൂടെ പ്രസ്താവിക്കുകയും പരിശോധിക്കേണ്ട വ്യവസ്ഥ 'ആയിരിക്കുന്നത് വരെ' എന്ന വാക്കിന്റെ കൂടെയും നൽകുന്നു. അൽഗോരിതം വ്യത്യസ്തമായിരിക്കുന്നത് പോലെ ചിത്രം 4.13 ൽ കാണിച്ചിരിക്കുന്ന ഫ്ളോച്ചാർട്ടും അല്പം വ്യത്യസ്തമായിരിക്കും. ചിത്രം 4.14 ൽ ലൂപ്പിന്റെ പ്രവർത്തന ശൈലി കാണിച്ചിരിക്കുന്നു.

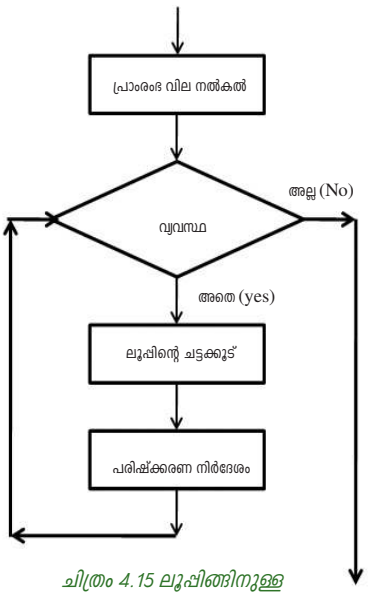



*ചിത്രം 4 .14 ലൂപ്പിംഗ് നിർമിതി*




*ചിത്രം 4.13 1 മുതൽ 100 വരെയുള്ള സംഖ്യകൾ പ്രിൻറ് ചെയ്യാനുള്ള ഫ്ളോച്ചാർട്ട്*

ഒരു ലൂപ്പിനു നാല് ഘടകങ്ങൾ ഉണ്ട്. സ്വാഭാവികമായും അവയിൽ ഒന്ന് വ്യവസ്ഥ (Condition) തന്നെ. വ്യവസ്ഥ നൽകുന്നതിനു വേണ്ടി ഒരു വേരിയബിളെങ്കിലും ഉപയോഗിക്കണം എന്നു നമുക്കറിയാം. ഇതിനെ ലൂപ്പ് നിയന്ത്രണ വേരിയബിൾ എന്ന് വിളിക്കാം. വ്യവസ്ഥ പരിശോധിക്കുന്നതിന് മുമ്പ് ലൂപ്പ് നിയന്ത്രണ വേരിയബിളിന് ഒരു വില ലഭ്യമാക്കേണ്ടതാണ്. ഇൻപുട്ട് അല്ലെങ്കിൽ വില നൽകൽ (Assignment) വഴി ഇത് സാധ്യമാകുന്നതാണ്. അത്തരത്തിലുള്ള നിർദേശങ്ങളെ ലൂപ്പിന്റെ പ്രാരംഭ വില നൽകൽ നിർദേശങ്ങൾ (Initialization Instructions) എന്ന് പറയുന്നു. പരിഷ്കരണ നിർദേശം (Update Instruction) എന്ന മൂന്നാമത്തെ ഘടകമാണ് ലൂപ്പ് നിയന്ത്രണ വേരിയബിളിന്റെ വില മാറ്റുന്നത്. ഇത് വളരെ അത്യാവശ്യമാണ്. എന്തെന്നാൽ പരിഷ്കരണ നിർദേശം ഇല്ലെങ്കിൽ ലൂപ്പിന്റെ പ്രവർത്തനം ഒരിക്കലും അവസാനിക്കില്ല. ആവർത്തിച്ചു പ്രവർത്തിക്കേണ്ട ഒരു കൂട്ടം നിർദേശങ്ങളാണ് ലൂപ്പിന്റെ ചട്ടക്കൂട് (Body of the Loop) എന്ന നാലാമത്തെ ഘടകം. ചിത്രം 4.15 ൽ കാണിച്ചിരിക്കുന്ന ഫ്ളോച്ചാർട്ട് ലൂപ്പിങ് ഘടന വ്യക്തമാക്കുന്നു.




*ചിത്രം 4.15 ലൂപ്പിങ്ങിനുള്ള ഫ്ളോച്ചാർട്ട്*

പ്രാരംഭ വില നൽകൽ നിർദേശമാണ് ആദ്യം പ്രവർത്തിക്കുക. ശേഷം വ്യവസ്ഥ പരിശോധിക്കുന്നു. വ്യവസ്ഥ ശരിയാണെങ്കിൽ ലൂപ്പിന്റെ ചട്ടക്കൂടും തുടർന്ന് പരിഷ്കരണ നിർദേശവും പ്രവർത്തിക്കുന്നു. പരിഷ്കരണ നിർദേശം പ്രവർത്തിച്ചു കഴിഞ്ഞാൽ വീണ്ടും വ്യവസ്ഥ പരിശോധിക്കുന്നു. വ്യവസ്ഥ തെറ്റാകുന്നത് വരെ ഈ പ്രക്രിയ തുടരുന്നു. ലൂപ്പിന്റെ ചട്ടക്കൂട് പ്രാവർത്തികമാകുന്നതിനു മുമ്പ് വ്യവസ്ഥ പരിശോധിക്കുന്ന ലൂപ്പിനെ ആഗമന നിയന്ത്രിത ലൂപ്പ് (entry cotnrolled Loop ) എന്ന് വിളിക്കുന്നു. മറ്റൊരു രീതിയിലുള്ള ലൂപ്പിംഗ് നിർമ്മിതിയും നിലവിലുണ്ട്. അതിൽ ലൂപ്പിന്റെ ചട്ടക്കൂടും പരിഷ്കരണ നിർദേശവും പ്രവർത്തിച്ചു കഴിഞ്ഞ ശേഷം മാത്രമേ വ്യവസ്ഥ പരിശോധിക്കുകയുള്ളു. ഇത്തരത്തിലുള്ള ലൂപ്പിനെ നിർഗമന നിയന്ത്രിത ലൂപ്പ് (Exit Cotnrolled Loop) എന്ന് വിളിക്കുന്നു.

### ഉദാഹരണം 4.6: ആദ്യത്തെ N എണ്ണൽ സംഖ്യകളുടെ തുക കണ്ടെത്തുക

ഇവിടെ N ന്റെ വില നമ്മൾ ഇൻപുട്ട് ആയി നൽകുന്നു. 1 മുതൽ N വരെയുള്ള സംഖ്യകളുടെ തുക കണ്ടുപിടിക്കേണ്ടതാണ്. തുക സംഭരിക്കുന്നതിനായി 'S' എന്ന വേരിയബിളാണെന്നിരിക്കട്ടെ, ചിത്രം 4.16 ൽ ഈ അൽഗോരിതത്തിൻറെ ഫ്ളോച്ചാർട്ട് കാണിച്ചിരിക്കുന്നു.

ഘട്ടം 1 : തുടങ്ങുക

ഘട്ടം 2 : N ലേക്ക് വില ഇൻപുട്ട് ആയി സ്വീകരിക്കുക

ഘട്ടം 3 : A=1 ,S=0

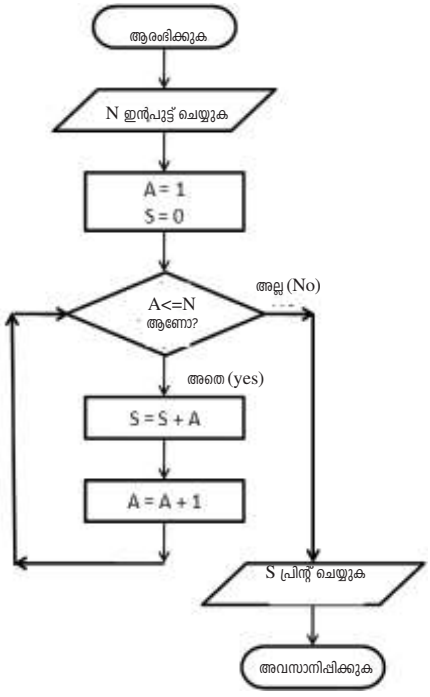
ഘട്ടം 4 : (A<=N) ആയിരിക്കുന്നത് വരെ ഘട്ടം 5 ഉം 6 ഉം ആവർത്തിക്കുക

ഘട്ടം 5 : S =S +A

ഘട്ടം 6 : A =A +1

ഘട്ടം 7 : S പ്രിന്റ് ചെയ്യുക.

ഘട്ടം 8 : അവസാനിപ്പിക്കുക




*ചിത്രം 4.16 ആദ്യത്തെ N എണ്ണൽ സംഖ്യകളുടെ ആകെത്തുക കാണാനുള്ള ഫ്ളോച്ചാർട്ട്*

ഈ അൽഗോരിതം ഒരു ആഗമന നിയന്ത്രിത ലൂപ്പ് ഉപയോഗിക്കുന്നു. അടുത്ത ഉദാഹരണത്തിൽ പ്രശ്ന പരിഹാരത്തിനായി ഒരു നിർഗമന നിയന്ത്രിത ലൂപ്പ് ഉപയോഗിക്കുന്ന അൽഗോരിതം കാണാം.

**ഉദാഹരണം 4.7 : തന്നിരിക്കുന്ന സംഖ്യയുടെ ആദ്യത്തെ 10 ഗുണിതങ്ങൾ കണ്ടെത്തുക**

ഘട്ടം 1 : തുടങ്ങുക

ഘട്ടം 2 : N ഇൻപുട്ട് ആയി സ്വീകരിക്കുക

ഘട്ടം 3 : A=1

ഘട്ടം 4 : M =A x N

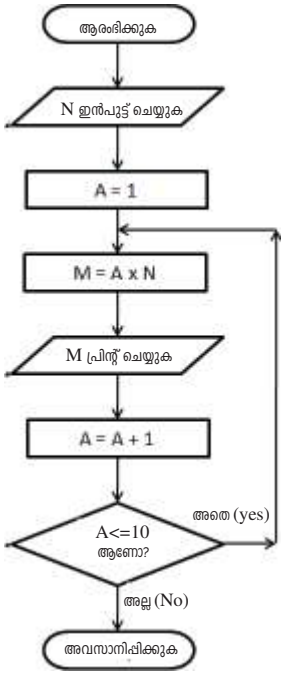
ഘട്ടം 5 : M പ്രിന്റ് ചെയ്യുക

ഘട്ടം 6 : A=A+1

ഘട്ടം 7 : (A<=10) ആയിരിക്കുന്നത് വരെ ഘട്ടം 4 മുതൽ 5 വരെ ആവർത്തിക്കുക

ഘട്ടം 8 : അവസാനിപ്പിക്കുക

ഈ അൽഗോരിതത്തിന് അനുസൃതമായ ഫ്ളോച്ചാർട്ട് ചിത്രം 4.17 ൽ നൽകിയിരിക്കുന്നു. ഇവിടെ ലൂപ്പ് ചട്ടക്കൂട് പ്രവർത്തിച്ച ശേഷം മാത്രം പരിശോധിക്കപ്പെടുന്ന ഒരു വ്യവസ്ഥ അടങ്ങിയ ഒരു ലൂപ്പ് ഉപയോഗിച്ചിരിക്കുന്നു. ആഗമന നിയന്ത്രിത ലൂപ്പും നിർഗമന നിയന്ത്രിത ലൂപ്പും തമ്മിലുള്ള താരതമ്യം പട്ടിക 4.1 ൽ കാണിച്ചിരിക്കുന്നു.




*ചിത്രം 4.17 തന്നിരിക്കുന്ന സംഖ്യയുടെ ആദ്യത്തെ 10 ഗുണിതങ്ങൾ കണ്ടുപിടിക്കാനുള്ള ഫ്ളോച്ചാർട്ട്*

| ആഗമന നിയന്ത്രിത ലൂപ്പ് | നിർഗമന നിയന്ത്രിത ലൂപ്പ് |
|---|---|
| • ലൂപ്പ് ചട്ടക്കൂട് പ്രാവർത്തികമാക്കുന്നതിനു മുമ്പ് വ്യവസ്ഥ പരിശോധിക്കുന്നു<br>• ലൂപ്പ് ചട്ടക്കൂട് ഒരിക്കൽ പോലും പ്രവർത്തിച്ചില്ല എന്ന് വരാം<br>• ലൂപ്പ് ചട്ടക്കൂട് പ്രാവർത്തികമാക്കുന്നതിൽ നിന്നും ഒഴിവാക്കണമെങ്കിൽ ഇത് ഉപയോഗിക്കുന്നു. | • ലൂപ്പ് ചട്ടക്കൂട് പ്രാവർത്തികമാക്കിയതിന് ശേഷം വ്യവസ്ഥ പരിശോധിക്കുന്നു<br>• ലൂപ്പ് ചട്ടക്കൂട് കുറഞ്ഞത് ഒരു തവണ എങ്കിലും നിർബന്ധമായും പ്രവർത്തിച്ചിരിക്കും<br>• ചട്ടക്കൂടിന്റെ സാധാരണ നിലയിലുള്ള പ്രവർത്തനം ഉറപ്പുവരുത്തണമെങ്കിൽ ഉപയോഗിക്കുന്നു |

*പട്ടിക 4.1: ലൂപ്പുകളുടെ താരതമ്യം*

പഠന പ്രവർത്തനങ്ങളുടെ ഭാഗമായി നൽകിയിരിക്കുന്ന പ്രശ്നപരിഹാരത്തിനുള്ള അൽഗോരിതങ്ങളും ഫ്ളോച്ചാർട്ടും ഉപയോഗിച്ച് നമുക്ക് പരിശീലിക്കാം

ചെയ്തു നോക്കൂ

താഴെ പറയുന്ന പ്രശ്നങ്ങൾക്കു വേണ്ടി അൽഗോരിതവും ഫ്ളോച്ചാർട്ടും തയാറാക്കുക

1. 100 ൽ താഴെയുള്ള എല്ലാ ഇരട്ട സംഖ്യകളും അവരോഹണ ക്രമത്തിൽ പ്രിന്റ് ചെയ്യുക
2. 100 നും 200 നും ഇടയിലുള്ള ഒറ്റ സംഖ്യകളുടെ തുക കണ്ടുപിടിക്കുക
3. തന്നിരിക്കുന്ന സംഖ്യയുടെ ഗുണനപ്പട്ടിക പ്രിൻറ് ചെയ്യുക
4. ഒരു സംഖ്യയുടെ ഫാക്ടോറിയൽ (FACTORIAL) കണ്ടുപിടിക്കുക
5. ഒരു സംഖ്യ ഇൻപുട്ട് ചെയ്ത് അവിഭാജ്യ സംഖ്യയാണോ എന്ന് പരിശോധിക്കുക

### 4.3.3 പ്രോഗ്രാം കോഡ് തയാറാക്കൽ (Coding the Program)

അൽഗോരിതവും ഫ്ളോച്ചാർട്ടും രൂപകൽപന ചെയ്തു കഴിഞ്ഞാൽ പ്രോഗ്രാമിങ്ങിന്റെ അടുത്ത പടി നിർദേശങ്ങൾ സൂക്ഷ്മവും സംക്ഷിപ്തവുമായ പ്രതീകങ്ങൾ ഉപയോഗിച്ച് ആവിഷ്കരിക്കുക എന്നതാണ്. അതായത് നിർദേശങ്ങൾ പ്രോഗ്രാമിങ് ഭാഷയിൽ ആവിഷ്കരിക്കുന്നു. പ്രശ്ന പരിഹാരത്തിന് വേണ്ടി ഇത്തരം പ്രോഗ്രാം നിർദേശങ്ങൾ എഴുതുന്നതിനെയാണ് കോഡിങ് എന്ന് പറയുന്നത്. കമ്പ്യൂട്ടറിൽ കോഡ് എഴുതുന്നതിനായി ടെക്സ്റ്റ് എഡിറ്റർ പ്രോഗ്രാമുകൾ ലഭ്യമാണ്.

ആശയവിനിമയത്തിനുള്ള ഒരു സംവിധാനമാണ് ഭാഷ. ഇംഗ്ലീഷ് മലയാളം മുതലായ സ്വാഭാവിക ഭാഷകൾ ഉപയോഗിച്ച് നാം നമ്മുടെ ആശയങ്ങളും വികാരങ്ങളും പരസ്പരം പങ്കുവയ്ക്കുന്നു .

അത് പോലെ ഉപയോക്താവിനും കമ്പ്യൂട്ടറിനും ഇടയിൽ ആശയവിനിമയം നടത്താൻ പ്രോഗ്രാമിങ് ഭാഷ ഉപയോഗിക്കുന്നു. കമ്പ്യൂട്ടർ പ്രോഗ്രാം എഴുതുന്ന വ്യക്തിക്ക് കംപ്യൂട്ടറിനു മനസ്സിലാകുന്ന ഭാഷയും പരിചിതമായിരിക്കണം. മനുഷ്യർക്ക് മനസ്സിലാക്കാനും ഉപയോഗിക്കാനും ബുദ്ധിമുട്ടുള്ള ദ്വയാങ്ക ഭാഷ (BInary Language) മാത്രമേ കമ്പ്യൂട്ടറിന് അറിയുകയുള്ളൂ എന്നത് നമ്മൾ നേരത്തെ കണ്ടതാണ്. അതിനാൽ അധ്യായം മൂന്നിൽ പഠിച്ചതു പോലെ ഇംഗ്ലീഷ് ഭാഷയ്ക്ക് സമാനമായതും മനുഷ്യർക്ക് സഹൃദയമായതുമായ ഉയർന്നതല ഭാഷ (High Level Language) (HLL) നമുക്ക് ഉപയോഗിക്കാവുന്നതാണ്. ഉയർന്നതല ഭാഷയിൽ എഴുതിയിരിക്കുന്ന പ്രോഗ്രാമുകളെ യന്ത്ര ഭാഷയിലേക്കു വിവർത്തനം അല്ലെങ്കിൽ മൊഴിമാറ്റം ചെയ്യാൻ ലാംഗ്വേജ് പ്രോസസ്സർ ഉപയോഗിക്കുന്നു. ഉയർന്നതല ഭാഷയിൽ എഴുതിയിരിക്കുന്ന പ്രോഗ്രാം, സോഴ്സ് കോഡ് (Source Code) എന്നറിയപ്പെടുന്നു.

ഒരു പ്രോഗ്രാമർ ആകണമെങ്കിൽ പ്രോഗ്രാമിലെ നിർദേശങ്ങൾ ആവിഷ്ക്കരിക്കുന്നതിനു വേണ്ടിയുള്ള ബേസിക് (BASIC), കോബോൾ (COBOL), പാസ്കൽ (PASCAL), C++ തുടങ്ങിയ ഉയർന്നതല ഭാഷകളിൽ ഏതിലെങ്കിലും ഒന്നിൽ നൈപുണ്യം കൈവരിക്കേണ്ടതാണ്. പ്രോഗ്രാമുകൾ തയാറാക്കുന്നതിന് വേണ്ടി ഓരോ പ്രോഗ്രാമിങ് ഭാഷയും അതിന്റേതായ അക്ഷരമാലകളും (Charecter Set), ശബ്ദകോശങ്ങളും (Voccabulary), വ്യാകരണങ്ങളും (Grammar) (നമുക്കതിനെ വാക്യ ഘടന (Syntax) എന്ന് വിളിക്കാം) ഉണ്ടായിരിക്കും.

ഈ ഭാഷ ഉപയോഗിച്ച് പ്രോഗ്രാം തയ്യാറാക്കി കഴിഞ്ഞാൽ അത് ഒരു ഫയലിൽ (സോഴ്സ് ഫയൽ) സൂക്ഷിക്കുകയും പ്രോഗ്രാമിങ്ങിന്റെ അടുത്ത ഘട്ടത്തിലേക്ക് കടക്കുകയും ചെയ്യുന്നു.

### 4.3.4 പരിഭാഷ (Translation)

സോഴ്സ് കോഡ് വികസിപ്പിക്കുന്നതിനായി ഭാഷ തിരഞ്ഞെടുക്കുമ്പോൾ ഉപയോഗിക്കുന്ന ഡാറ്റയുടെ അളവ്, പ്രവർത്തനത്തിന്റെ സങ്കീർണ്ണത, ഫയലുകളുടെ ഉപയോഗം മുതലായ ചില മാനദണ്ഡങ്ങൾ പരിഗണിക്കേണ്ടതുണ്ട്. പ്രോഗ്രാമിങ് ഭാഷ തിരഞ്ഞെടുക്കുകയും സോഴ്സ് കോഡ് തയ്യാറാക്കുകയും ചെയ്തു കഴിഞ്ഞാൽ അതിനെ ലാംഗ്വേജ് പ്രോസെസ്സറിൻറെ സഹായത്തോടെ പരിഭാഷ ചെയേണ്ടതാണ്. ഉയർന്നതല ഭാഷയിൽ എഴുതപ്പെട്ടിരിക്കുന്ന പ്രോഗ്രാമിനെ അതിനു തുല്യമായ യന്ത്രഭാഷാ പതിപ്പിലേക്കു രൂപഭേദം വരുത്തുന്ന പ്രക്രിയയാണ് പരിഭാഷ (Translation). കംപൈലർ അല്ലെങ്കിൽ ഇൻറെർപ്രെറ്റർ ഇതിനായി ഉപയോഗിക്കുന്നു. പ്രോഗ്രാമിലുള്ള സിന്റാക്സ് എററുകൾ (Syntax Error) ഈ ഘട്ടത്തിൽ ദൃശ്യമാകുന്നു. സോഴ്സ് കോഡ് അടങ്ങിയ ഫയൽ തുറന്നു ഈ തെറ്റുകൾ തിരുത്തേണ്ടതാണ്. അതിനു ശേഷം സോഴ്സ് കോഡ് വീണ്ടും കമ്പൈൽ ചെയ്യാനായി (പരിഭാഷ) നൽകുന്നു. "No Errors or Warnings" അല്ലെങ്കിൽ "Sucessfull Compilation" എന്ന സന്ദേശം ലഭിക്കുന്നത് വരെ ഈ പ്രക്രിയ തുടർന്ന് കൊണ്ടിരിക്കും. പൂർണമായും യന്ത്രഭാഷയിലെ നിർദേശങ്ങൾ അടങ്ങിയ പ്രോഗ്രാം നമുക്കിപ്പോൾ ലഭ്യമാകുന്നു.സോഴ്സ് കോഡിൻറെ ഈ പതിപ്പ് ഒബ്ജക്റ്റ് കോഡ് (Object Code) എന്ന് അറിയപ്പെടുന്നു. കംപൈലർ തന്നെ ഈ ഒബ്ജക്റ്റ് കോഡിനെ മറ്റൊരു ഒരു ഫയലിൽ സൂക്ഷിക്കുകയും ചെയ്യുന്നു


*ചിത്രം 4.18 പരിഭാഷാ പ്രവർത്തനം*

ഇങ്ങനെ ഒബ്ജക്റ്റ് കോഡ് ലഭിച്ചു കഴിഞ്ഞാൽ പ്രോഗ്രാം ഉപയോഗിക്കുന്നിടത്തോളം കാലം ഈ കോഡ് ആ കമ്പ്യൂട്ടറിൽ തന്നെ ഉണ്ടായിരിക്കേണ്ടതാണ്.

### 4.3.5 ഡീബഗ്ഗിങ് (Debugging)

പ്രോഗ്രാമിങ്ങിലുള്ള തെറ്റുകൾ കണ്ടെത്തുകയും അവ തിരുത്തുകയും ചെയ്യുന്ന ഘട്ടമാണ് ഡീബഗ്ഗിങ്.മനുഷ്യർ കമ്പ്യൂട്ടറുകളെ പ്രോഗ്രാം ചെയ്യുന്നിടത്തോളം കാലം പ്രോഗ്രാമുകളിൽ തെറ്റുകൾ സംഭവിക്കാം. പ്രോഗ്രാമിങ്ങിലുള്ള തെറ്റുകളെ 'ബഗ്ഗ്' എന്ന് പറയുന്നു.തെറ്റുകൾ കണ്ടുപിടിക്കുകയും തിരുത്തുകയും ചെയ്യുന്ന പ്രക്രിയയെ 'ഡീബഗ്ഗിങ്' എന്ന് വിളിക്കുന്നു. പൊതുവെ പ്രോഗ്രാമിങ്ങിൽ രണ്ടു തരത്തിലുള്ള തെറ്റുകളാണ് വന്നുകൂടുക സിന്റാക്സ് എററുകളും ലോജിക്കൽ തെറ്റുകളും. പ്രോഗ്രാമിങ് ഭാഷയുടെ നിയമങ്ങൾ അല്ലെങ്കിൽ വാക്യഘടന പാലിക്കാത്തതു കൊണ്ട് സംഭവിക്കുന്ന തെറ്റുകളെയാണ് സിന്റാക്സ് എറർ എന്ന് വിളിക്കുന്നത്. തെറ്റായ ചിഹ്നങ്ങൾ ഉപയോഗിക്കുക, തെറ്റായ വാക്യപ്രയോഗം, നിർവചിക്കപ്പെടാത്ത പദങ്ങളുടെ ഉപയോഗം, നിയമവിരുദ്ധമായ വാക്യരചന അല്ലെങ്കിൽ പദങ്ങളുടെ ഉപയോഗം മുതലായ

കാരണങ്ങൾ കൊണ്ടാണ് സാധാരണ അത്തരം തെറ്റുകൾ സംഭവിക്കുക. പരിഭാഷക്കു വേണ്ടി പ്രോഗ്രാം നൽകി കഴിഞ്ഞാൽ തന്നെ ലാംഗ്വേജ് പ്രൊസസ്സറുകൾ സിൻറ്റാക്സ് തെറ്റുകൾ കണ്ടെത്തുന്നു. തെറ്റുകൾ സംഭവിച്ചിരിക്കുന്ന പ്രസ്താവനകളുടെ ലൈൻ നമ്പറുകൾ അതി നോടൊപ്പം സംഭവിച്ചിരിക്കുന്ന തെറ്റിനെ കുറിച്ചുള്ള സൂചനകളും അവ നൽകുന്നു. എറർ മെസ്സേജുകളായി പ്രദർശിപ്പിക്കുന്നു. ഇൻറ്റർപ്രെറ്ററുകളുടെ കാര്യത്തിൽ, പ്രവർത്തന ഘട്ടത്തിലാണ് സിൻറ്റാക്സ് തെറ്റുകൾ കണ്ടുപിടിച്ച് പ്രദർശിപ്പിക്കുന്നത്. ഡീബഗ്ഗിങ് പ്രക്രിയക്കായി വേണ്ടി വരുന്ന സമയവും പരിശ്രമവുമാണ് ഒരു പ്രോഗ്രാമിങ് ഭാഷ ഉപയോഗിക്കുന്നതിൽ പ്രോഗ്രാമർക്കുള്ള പ്രാപ്തി എത്രത്തോളമുണ്ട് എന്ന് തീരുമാനിക്കുന്നത്. എല്ലാ സിൻറ്റാക്സ് തെറ്റുകളും തിരുത്തിക്കഴിഞ്ഞാൽ മാത്രമേ ഒബ്ജക്റ്റ് പ്രോഗ്രാം നിർമ്മിക്കപ്പെടുകയുള്ളു.

പ്രോഗ്രാമിങ് യുക്തിയുടെ ആസൂത്രണത്തിലുള്ള അപാകതകൾ കാരണമാണ് ലോജിക്കൽ എറർ എന്ന് പേരുള്ള രണ്ടാമത്തെ തരം തെറ്റുകൾ സംഭവിക്കുന്നത്.സിൻറ്റാക്സ് എററുകൾ ഇല്ലെങ്കിൽ ലാംഗ്വേജ് പ്രൊസസ്സറുകൾ വിജയകരമായി സോഴ്സ് കോഡിനെ പരിഭാഷപ്പെടുത്തുന്നു. പ്രോഗ്രാമിന്റെ പ്രവർത്തന ഘട്ടത്തിൽ കമ്പ്യൂട്ടർ, പ്രോഗ്രാം നിർദേശങ്ങൾ പിന്തുടരുകയും അവക്ക് അനുസൃതമായ ഔട്ട്പുട്ട് നൽകുകയും ചെയ്യുന്നു.പക്ഷെ ഈ ഔട്ട്പുട്ട് ശരിയായിരിക്കണമെന്നില്ല. ഇതിനെയാണ് ലോജിക്കൽ എറർ എന്ന് പറയുന്നത്. ലോജിക്കൽ എറർ സംഭവിക്കുമ്പോൾ പ്രോഗ്രാം തെറ്റായ ഔട്ട്പുട്ട് ആണ് തരുന്നത് എന്ന് മാത്രമേ നമുക്ക് ഗ്രഹിക്കാൻ സാധിക്കുകയുള്ളു.എന്താണ് തെറ്റ് എന്ന് കമ്പ്യൂട്ടർ നമ്മോടു പറയുന്നില്ല. പ്രോഗ്രാമർ അല്ലെങ്കിൽ ഉപയോക്താവാണ് അത് കണ്ടെത്തേണ്ടത്. ലോജിക്കൽ തെറ്റുകൾ ഉണ്ടോ ഇല്ലയോ എന്നറിയുന്നതിനായി പ്രോഗ്രാം പരീക്ഷിക്കപ്പെടേണ്ടതാണ്. അതിനായി പ്രോഗ്രാമിങ്ങിന്റെ അടുത്ത ഘട്ടത്തിലേക്ക് നമുക്ക് കടക്കാം.

### 4.3.6 പ്രവർത്തനവും പരീക്ഷണവും (Execution and Testing)

മുകളിൽ പറഞ്ഞത് പോലെ ലോജിക്കൽ തെറ്റുകൾ കൂടി തിരുത്തിയാൽ മാത്രമേ ഒരു പ്രോഗ്രാം തെറ്റുകളിൽ നിന്നും മുക്തമാണ് എന്ന് നമുക്ക് പറയാൻ കഴിയൂ. ആയതിനാൽ കംപൈൽ ചെയ്യപ്പെട്ട പ്രോഗ്രാമിന്റെ പതിപ്പ് പരീക്ഷണത്തിനായി പ്രവർത്തിപ്പിക്കേണ്ടതാണ്. ശരിയായ ഫലങ്ങൾ ലഭ്യമാകുന്നുണ്ടോ എന്ന് പരിശോധിക്കുകയാണ് പരീക്ഷണത്തിന്റെ ഉദ്ദേശ്യം. 'അറിയാവുന്ന ഫലങ്ങൾ' ലഭിക്കുന്നതിന് വേണ്ടിയുള്ള പരീക്ഷണ ഡാറ്റ നൽകി പ്രോഗ്രാം പ്രാവർത്തികമാക്കുക എന്ന പ്രക്രിയയാണ് പരീക്ഷണ ഘട്ടത്തിൽ ഉൾപ്പെട്ടിരിക്കുന്നത്. അതായത് പ്രോഗ്രാമിൽ ഉൾപ്പെട്ടിരിക്കുന്ന ക്രിയകൾ മനുഷ്യൻ തന്നെ ചെയ്യുകയും, ലഭ്യമാകുന്ന ഔട്ട്പുട്ട് കമ്പ്യൂട്ടർ മുഖേന ചെയ്യുമ്പോൾ ലഭിക്കുന്ന ഔപുട്ടുമായി താരതമ്യം ചെയ്യുകയും വേണം. പ്രോഗ്രാം യുക്തിയുടെ കൃത്യത ഈ പരീക്ഷണ ഘട്ടം കൊണ്ട് നിർണയിക്കാവുന്നതാണ്. പരീക്ഷണത്തിനായുള്ള ഡാറ്റ തിരഞ്ഞെടുക്കുമ്പോൾ പ്രോഗ്രാം യുക്തിയുടെ എല്ലാ വശങ്ങളും പരീക്ഷിക്കപ്പെടുന്നുണ്ട് എന്ന് ഉറപ്പാക്കേണ്ടതാണ്. അതിനാൽ തന്നെ ഉചിതമായ ഡാറ്റ തിരഞ്ഞെടുക്കുക എന്നത് പ്രോഗ്രാം പരീക്ഷണത്തിൽ വളരെ പ്രാധാന്യമുള്ളതാണ്.

തെറ്റായ യുക്തി മൂലം ലഭിക്കുന്ന തെറ്റായ ഔട്ട്പുട്ടുകളെ കുറിച്ചാണ് നാം ഇതുവരെ ചർച്ച ചെയ്തു കൊണ്ടിരുന്നത്. എന്നാൽ പ്രോഗ്രാം പ്രവർത്തനത്തെ തടസ്സപ്പെടുത്താവുന്ന മറ്റൊരു തരം തെറ്റ് സംഭവിക്കാൻ സാധ്യതയുണ്ട്. ഒരു ക്രിയയിൽ അനുചിതമായ ഡാറ്റ വരുന്നത് മൂലം സംഭവിക്കാവുന്ന ഒന്നാണത്.ഉദാഹരണത്തിന്, A =B / C എന്ന നിർദേശം പരിഗണിക്കുക. Cയുടെ വില പൂജ്യം ആയാൽ ഈ പ്രസ്താവന പ്രോഗ്രാമിന്റെ പ്രവർത്തനത്തെ തടസ്സപ്പെടുത്തുന്നു (പൂജ്യം കൊണ്ടുള്ള ഹരണം മൂലം). ഇത്തരം സാഹചര്യങ്ങളിൽ പ്രോഗ്രാമിംഗ് ഭാഷയിലുള്ള തെറ്റുകൾ കൈകാര്യം ചെയ്യുന്ന ഫങ്ക്ഷനുകൾ (Error handling function) എറർ മെസ്സേജുകൾ പ്രദർശിപ്പിക്കുന്നു.ഇത്തരം തെറ്റുകളെ റൺ ടൈം എറർ എന്ന് വിളിക്കുന്നു.ഡാറ്റ പ്രോസസ്സ് ചെയ്യപ്പെടുന്നതിനു മുമ്പ് ഡാറ്റയുടെ സാധുത പരിശോധിക്കാനുള്ള അനുബന്ധ നിർദേശങ്ങൾ പ്രോഗ്രാമിൽ ഉൾപ്പെടുത്തുക വഴി ഇത്തരം തെറ്റുകൾ തിരുത്താവുന്നതാണ്.

### 4.3.7 വിവരണം തയ്യാറാക്കൽ (Documentation)

ഉചിതമായ രീതിയിയിൽ വിവരണം തയ്യാറാക്കാത്ത ഒരു കംപ്യൂട്ടർവൽകൃത സംവിധാനം ഒരിക്കലും പൂർണമാണെന്നു പറയാൻ നമുക്ക് കഴിയില്ല. വാസ്തവത്തിൽ, പ്രശ്നപഠന ഘട്ടം മുതൽ അത് പ്രയോഗത്തിൽ വരുത്തുന്നത് വരെ തുടർച്ചയായി നടന്നുകൊണ്ടിരിക്കുന്ന ഒരു പ്രക്രിയയാണ് വിവ രണം തയ്യാരാക്കൽ. ഇതിന്റെ ഭാഗമായി സോഴ്സ് കോഡിൽ നമുക്ക് കമെന്റുകൾ ഉൾപ്പെടു ത്താവുന്നതാണ്. ഇത് ആന്തരിക വിവരണം (Internal Documentation) എന്നറിയപ്പെടുന്നു. ഡീബഗ്ഗിങ് ഘട്ടത്തിലും പിൽക്കാലത്ത് പ്രോഗ്രാമിൽ വരുന്ന മാറ്റങ്ങൾ ഉൾപ്പെടുത്താനും ഇത് സഹായിക്കുന്നു. പ്രോഗ്രാം നിർമാണ സമയത്ത് ഉപയോഗിച്ച യുക്തി പിന്നീട് നമ്മുടെ തന്നെ പ്രോഗ്രാമിലൂടെ കടന്നു പോകുമ്പോൾ നമുക്ക് ഓർമയുണ്ടായിരിക്കണമെന്നില്ല. മാത്രമല്ല ചില സാഹചര്യങ്ങളിൽ ഒരു വ്യക്തി എഴുതിയ പ്രോഗ്രാം മറ്റൊരു വ്യക്തിക്ക് ഭാവിയിൽ മാറ്റേണ്ടതായി വരാം.ഒരു പ്രോഗ്രാമിൽ കൃത്യമായി വിവരണം തയ്യാറാക്കിയാൽ നമ്മൾ ഉപയോഗിച്ച യുക്തി മനസ്സിലാക്കാനും പ്രോഗ്രാമിൽ ഒരു പ്രസ്താവന എന്ത് കൊണ്ടാണ് ഉപയോഗിച്ചിരിക്കുന്നത് എന്ന് മനസ്സിലാക്കാനും സാധിക്കുന്നു. എന്നിരുന്നാലും പ്രോഗ്രാം പരിഭാഷക്കായി നൽകുമ്പോൾ ലാംഗ്വേജ് പ്രൊസസ്സറുകൾ പ്രോഗ്രാമിന്റെ വിവരണ ഭാഗങ്ങൾ പരിഭാഷക്കായി പരിഗണിക്കുകയില്ല.

പ്രോഗ്രാമിന്റെ ഭാഗമായ കമെന്റുകൾ വിവരണം തയ്യാറാക്കുന്ന ഘട്ടത്തിന്റെ ഒരു ഭാഗം മാത്രമാണ്. സിസ്റ്റം മാനുവൽ, ഉപയോക്തൃ മാനുവൽ എന്നിവ തയ്യാറാക്കുക എന്നത് വിവരണം തയ്യാറാക്കു ന്നതിന്റെ മറ്റൊരു പ്രക്രിയയാണ്. കമ്പ്യൂട്ടർ വ്യവസ്ഥയുടെ പ്രവർത്തനം, അവയുടെ ആവശ്യകത, പ്രോഗ്രാമുകൾ ഇൻസ്റ്റാൾ ചെയ്യുക, അവ ഉപയോഗിക്കുന്ന രീതികൾ എന്നിവ ഉൾപ്പെടുന്ന ഹാർഡ് കോപ്പികളാണിവ. വിവിധ ആവശ്യങ്ങൾക്ക് വേണ്ടിയുള്ള സോഫ്റ്റ്വെയറുകൾ തയ്യാറാക്കുമ്പോൾ ഇത്തരം മാനുവലുകൾ നിർബന്ധമാണ്. ഇത്തരം വിവരണം തയ്യാറാക്കുന്നതിനെയാണ് ബാഹ്യമായ വിവരണം (External Documentation) എന്ന് പറയുന്നത്.

ഇപ്പോൾ നാം ഒരു പ്രശ്നത്തെ വിശകലനം ചെയ്യുകയും പരിഹാരത്തിനുള്ള യുക്തി കണ്ടെ ത്തുകയും, ഫ്ളോ ചാർട്ട് രൂപത്തിൽ പ്രതിപാദിക്കുകയും, പ്രോഗ്രാമിങ് ഭാഷയിൽ കോഡ് തയ്യാ റാക്കുകയും, സിന്റാക്സിലെ പിശകുകൾ നീക്കം ചെയ്ത ശേഷം പരിഭാഷപ്പെടുത്തുകയും ചെയ്തു. കൂടാതെ ലോജിക്കൽ തെറ്റുകളും റൺ ടൈം തെറ്റുകളും നീക്കം ചെയ്ത് ശേഷം ഔട്ട്പുട്ടിന്റെ കൃത്യത പരിശോധിക്കുകയും അവസാനമായി പ്രോഗ്രാമിന്റെ വിവരണം തയ്യാറാ ക്കുകയും ചെയ്തു.

**സ്വയം പരിശോധിക്കുക**

1. അൽഗോരിതം എന്നാൽ എന്താണ് ?
2. അൽഗോരിതത്തിന്റെ ചിത്ര ആവിഷ്കരണമാണ് _____.
3. ഏതു ഫ്ളോച്ചാർട്ട് ചിഹ്നമാണ് എപ്പോഴും ജോഡികളായി ഉപയോഗിക്കുന്നത്?
4. ഏതു ഫ്ളോച്ചാർട്ട് ചിഹ്നത്തിനാണ് ഒരു ആഗമന മാർഗവും രണ്ടോ അതിലധികമോ നിർഗമന മാർഗങ്ങളും ഉള്ളത് ?
5. HLL ൽ എഴുതിയിരിക്കുന്ന പ്രോഗ്രാം _____ എന്ന് അറിയപ്പെടുന്നു.
6. ഡീബഗ്ഗിങ് എന്നാലെന്ത് ?
7. ഒബ്ജക്റ്റ് കോഡ് എന്നാലെന്ത് ?

## 4.4 അൽഗോരിതങ്ങളുടെ പ്രകടനം വിലയിരുത്തൽ

വിവിധ പ്രശ്നങ്ങൾ പരിഹരിക്കുന്നതിനായി നമ്മൾ അൽഗോരിതങ്ങൾ തയാറാക്കിയിട്ടുണ്ട്. ചിലപ്പോൾ ചില പ്രശ്നങ്ങൾ പരിഹരിക്കുന്നതിന് വ്യത്യസ്തമായ യുക്തി പ്രയോഗിക്കാമായിരുന്നു എന്ന് നമുക്ക് തോന്നാം. ഒരേ പ്രശ്നം തന്നെ വ്യത്യസ്തമായ ഒരു കൂട്ടം നിർദേശങ്ങൾ ഉപയോഗിച്ച് പരിഹരിക്കാവുന്നതാണ്. എന്നിരുന്നാലും വളരെ കുറഞ്ഞ കമ്പ്യൂട്ടർ വിഭവങ്ങൾ ഉപയോഗപ്പെടുത്തി, കുറഞ്ഞ സമയം കൊണ്ട് വളരെ കൃത്യമായ ഫലം നൽകുന്ന പ്രോഗ്രാം തയ്യാറാക്കുന്നയാളാണ് ഒരു സമർത്ഥനായ പ്രോഗ്രാമർ എന്ന് പറയുന്നത്. ഒരു അൽഗോരിതത്തിന്റെ പ്രകടനം അളക്കുന്നത് ടൈം കോംപ്ലക്സിറ്റി (Time complexity), സ്പേസ് കോംപ്ലക്സിറ്റി (Space complexity) എന്നിവയുടെ അടിസ്ഥാനത്തിലാണ്. ഏറ്റവും കുറവ് മെമ്മറി ഉപയോഗപ്പെടുത്തി ഏറ്റവും വേഗത്തിൽ പ്രവർത്തിക്കുന്ന അൽഗോരിതത്തെയാണ് പ്രശ്ന പരിഹാരത്തിനുള്ള ഏറ്റവും നല്ല അൽഗോരിതം ആയി കണക്കാക്കുന്നത്.

| അൽഗോരിതം 1 | | അൽഗോരിതം 2 | |
|---|---|---|---|
| ഘട്ടം 1 : | തുടങ്ങുക | ഘട്ടം 1 : | തുടങ്ങുക |
| ഘട്ടം 2 : | A ,B ,C ഇൻപുട്ട് ചെയ്യുക | ഘട്ടം 2 : | A ,B ,C ഇൻപുട്ട് ചെയ്യുക |
| ഘട്ടം 3 : | S =A +B +C | ഘട്ടം 3 : | S=A+B+C |
| ഘട്ടം 4 : | AVG =S / 3 | ഘട്ടം 4 : | AVG =(A +B +C) / 3 |
| ഘട്ടം 5 : | S, AVG പ്രിന്റ് ചെയ്യുക | ഘട്ടം 5 : | S, AVG പ്രിന്റ് ചെയ്യുക |
| ഘട്ടം 6 : | അവസാനിപ്പിക്കുക | ഘട്ടം 6 : | അവസാനിപ്പിക്കുക |

പട്ടിക 4 .2 മൂന്നു സംഖ്യകളുടെ തുകയും ശരാശരിയും കാണാനുള്ള രണ്ടു അൽഗോരിതങ്ങൾ

മൂന്നു സംഖ്യകളുടെ തുകയും ശരാശരിയും കാണാനുള്ള രണ്ടു അൽഗോരിതങ്ങൾ പട്ടിക 4.2 ൽ കൊടുത്തിരിക്കുന്നു. അവയെ നമുക്ക് താരതമ്യം ചെയ്യാം. രണ്ടു അൽഗോരിതങ്ങളും ഘട്ടം 4 ൽ വ്യത്യാസപ്പെട്ടിരിക്കുന്നു. അൽഗോരിതം2 ൽ ഒരേ ഡാറ്റയിൽ രണ്ടു തവണ സങ്കലനം ചെയ്യുന്നു (ഘട്ടം 3, ഘട്ടം 4).സ്വാഭാവികമായിട്ടും ഈ അൽഗോരിതം അൽഗോരിതം1 നെക്കാൾ കൂടുതൽ സമയം പ്രവർത്തനത്തിനായി ഉപയോഗിക്കുന്നു. അതിനാൽ കോഡിങ്ങിനു നല്ലതു അൽഗോരിതം 1 ആകുന്നു.

ഒരു പ്രസ്താവന തിരഞ്ഞെടുക്കുന്നതിന് വേണ്ടി താരതമ്യ ക്രിയകൾ ഉൾപ്പെടുന്ന മറ്റൊരു ഉദാഹരണം നമുക്കു പരിശോധിക്കാം. മൂന്നു സംഖ്യകളിൽ വലുത് ഏതാണെന്നു കണ്ടെത്താനുള്ള അൽഗോരിതം ഉദാഹരണം 4.4 ൽ നമ്മൾ ചർച്ച ചെയ്യുകയുണ്ടായി. അതേ പ്രശ്നം പരിഹരിക്കുന്നതിനുള്ള രണ്ടു അൽഗോരിതങ്ങൾ പട്ടിക 4.3 ൽ നൽകിയിരിക്കുന്നു.

| അൽഗോരിതം 1 | | അൽഗോരിതം 2 | |
|---|---|---|---|
| ഘട്ടം 1 : | തുടങ്ങുക | ഘട്ടം 1 : | തുടങ്ങുക |
| ഘട്ടം 2 : | M 1 ,M 2 ,M 3 ഇൻപുട്ട് ആയി സ്വീകരിക്കുക | ഘട്ടം 2 : | M 1, M 2, M 3 ഇൻപുട്ട് ആയി സ്വീകരിക്കുക |
| ഘട്ടം 3 : | അഥവാ M 1 >M 2 ഉം M 1 > M 3 ഉം ആണെങ്കിൽ | ഘട്ടം 3 : | അഥവാ M 1 >M 2 ആണെങ്കിൽ |
| ഘട്ടം 4 : | M 1 പ്രിന്റ് ചെയ്യുക | ഘട്ടം 4 : | Big = M 1 |
| ഘട്ടം 5 : | അഥവാ M 2 >M 1 ഉം M 2 > M 3 ഉം ആണെങ്കിൽ | ഘട്ടം 5 : | അല്ലെങ്കിൽ |
| ഘട്ടം 6 : | M 2 പ്രിന്റ് ചെയ്യുക | ഘട്ടം 6 : | Big = M 2 |
| ഘട്ടം 7 : | അഥവാ M 3 >M 1 ഉം M 3 > M 2 ഉം ആണെങ്കിൽ | ഘട്ടം 7 : | അഥവാ M 3 >Big ആണെങ്കിൽ Big = M 3 |
| ഘട്ടം 8 : | M3 പ്രിന്റ് ചെയ്യുക | ഘട്ടം 8 : | Big പ്രിന്റ് ചെയ്യുക |
| ഘട്ടം 9 : | അവസാനിപ്പിക്കുക | ഘട്ടം 9 : | അവസാനിപ്പിക്കുക |

*പട്ടിക 4.3 മൂന്നു സംഖ്യകളിൽ ഏറ്റവും വലുത് കണ്ടെത്താനുള്ള അൽഗോരിതങ്ങൾ*

ഉദാഹരണം 4.3 ൽ മൂന്ന് താരതമ്യ ക്രിയകളും ഒരു ലോജിക്കൽ ക്രിയയും ഉപയോഗിച്ചിരിക്കുന്നു. ഏറ്റവും വലിയ വില M3 ൽ (മൂന്നാമത്തെ വേരിയബിൾ) വരുമ്പോളാണ് ഈ ക്രിയകൾ എല്ലാം പ്രവർത്തിക്കുക. പ്രവർത്തനത്തിന്റെ വേഗത അളക്കുന്നതിനായി താരതമ്യ ക്രിയക്ക് 1 സെക്കന്റ് സമയമെടുക്കുന്നു എന്ന് നമുക്ക് അനുമാനിക്കാം. അങ്ങനെയാണെങ്കിൽ ഏറ്റവും വേഗം കൂടിയ ഫലം 3 സെക്കന്റ് വേഗം കുറവുള്ള ഫലം 4 സെക്കന്റ് വേണ്ടി വരുന്നതായി കാണാം. ശരാശരി വേഗം 3.5 സെക്കന്റ് ആയിരിക്കും.

ഇനി നമുക്ക് പട്ടിക 4.3 ലുള്ള അൽഗോരിതം1 വിശകലനം ചെയ്യാം. അതിൽ മൂന്ന് താരതമ്യ ക്രിയകൾ അടങ്ങിയ മൂന്ന് If (അഥവാ) പ്രസ്താവനകൾ അടങ്ങിയിരിക്കുന്നു. മുകളിൽ പ്രസ്താവിച്ചിട്ടുള്ള അനുമാനം പിന്തുടരുകയാണെങ്കിൽ വേരിയബിലിന്റെ വില ഏതാണെങ്കിലും നമുക്ക് ഫലം 9 സെക്കന്റിൽ ലഭിക്കുന്നതാണ്. അപ്പോൾ ശരാശരി വേഗം എന്ന് പറയുന്നത്തു 9 സെക്കന്റ് ആണ്. പക്ഷെ പട്ടിക 4.3 ൽ പറയുന്ന അൽഗോരിതം2ൽ രണ്ടു If (അഥവാ) പ്രസ്താവനകൾ ഉപയോഗിച്ചിരിക്കുന്നു. അൽഗോരിതം2 ൽ വേരിയബിളുകളിൽ ഏതു വില വന്നാലും താരതമ്യം ചെയ്യാൻ വേണ്ട സമയം 2 സെക്കന്റ് ആയിരിക്കും. അപ്പോൾ ശരാശരി വേഗം എന്ന് പറയുന്നത് 2 സെക്കന്റ് ആയിരിക്കും.എന്നു പറഞ്ഞാൽ അൽഗോരിതം 2 മറ്റു രണ്ടു അൽഗോരിതങ്ങളെക്കാൾ മെച്ചപ്പെട്ടതാണ്.

ലൂപ്പുകൾ അടങ്ങിയ മറ്റൊരു ഉദാഹരണം കൂടി നമുക്ക് നോക്കാം. 100 നും 200 നും ഇടയിലുള്ള എല്ലാ ഒറ്റ സംഖ്യകളുടെ തുകയും എല്ലാ ഇരട്ട സംഖ്യകളുടെ തുകയും കണ്ടുപിടിക്കാനുള്ള രണ്ടു അൽഗോരിതങ്ങൾ പട്ടിക 4.4 ൽ കൊടുത്തിരിക്കുന്നു.

| അൽഗോരിതം 1 | | അൽഗോരിതം 2 | |
|---|---|---|---|
| ഘട്ടം 1 : | തുടങ്ങുക | ഘട്ടം 1 : | തുടങ്ങുക |
| ഘട്ടം 2 : | N =100, ES =0 | ഘട്ടം 2 : | N =100 ,ES =0,OS =0 |
| ഘട്ടം 3 : | (N <=200) ആയിരിക്കുന്നതു വരെ ഘട്ടം 4 മുതൽ 6 വരെ ആവർത്തിക്കുക | ഘട്ടം 3 : | (N <=200) ആയിരിക്കുന്നതു വരെ ഘട്ടം 4 മുതൽ 8 വരെ ആവർത്തിക്കുക |
| ഘട്ടം 4 : | അഥവാ N/2 ന്റെ ശിഷ്ടം = 0 ആണെങ്കിൽ | ഘട്ടം 4 : | അഥവാ N/2 ന്റെ ശിഷ്ടം = 0 ആണെങ്കിൽ |
| ഘട്ടം 5 : | ES =ES + N | ഘട്ടം 5 : | ES =ES + N |
| ഘട്ടം 6 : | N =N +1 | ഘട്ടം 6 : | അല്ലെങ്കിൽ |
| ഘട്ടം 7 : | ES പ്രിന്റ് ചെയ്യുക | ഘട്ടം 7 : | OS =OS + N |
| ഘട്ടം 8 : | N =100, OS =0 | ഘട്ടം 8 : | N =N +1 |
| ഘട്ടം 9 : | (N <=200 ) ആയിരിക്കുന്നതു വരെ ഘട്ടം 10 മുതൽ 12 വരെ ആവർത്തിക്കുക | ഘട്ടം 9 : | ES പ്രിന്റ് ചെയ്യുക |
| ഘട്ടം 10 : | അഥവാ N/2 ന്റെ ശിഷ്ടം = 1 ആണെങ്കിൽ | ഘട്ടം 10 : | OS പ്രിന്റ് ചെയ്യുക |
| ഘട്ടം 11 : | OS =OS + N | ഘട്ടം 11 : | അവസാനിപ്പിക്കുക |
| ഘട്ടം 12 : | N =N +1 | | |
| ഘട്ടം 13 : | OS പ്രിന്റ് ചെയ്യുക | | |
| ഘട്ടം 14 : | അവസാനിപ്പിക്കുക | | |

*പട്ടിക 4.4 ഒറ്റ സംഖ്യകളുടെയും ഇരട്ട സംഖ്യകളുടെയും തുക കാണാനുള്ള അൽഗോരിതങ്ങൾ*

അൽഗോരിതം1 രണ്ടു ലൂപ്പുകളും അൽഗോരിതം2 ഒരു ലൂപ്പും ഉപയോഗിക്കുന്നു. സ്വാഭാവികമായിട്ടും അൽഗോരിതം2 നെ അപേക്ഷിച്ച് അൽഗോരിതം1 ന് പ്രാരംഭ വിലനൽകാനും, പരിശോധനയ്ക്ക് വേണ്ടിയും ലൂപ്പ് വേരിയബിൾ പുതുക്കുന്നതിന് വേണ്ടിയും മറ്റും ഇരട്ടി സമയം ആവശ്യമായി വരുന്നു. അൽഗോരിതം2 മെച്ചപ്പെട്ടതും കാര്യക്ഷമവുമാണ് എന്ന് പട്ടികയിൽ നിന്ന് തന്നെ വ്യക്തമാണ്. അതിനാൽ തന്നെ പ്രശ്ന പരിഹാരത്തിനുള്ള യുക്തി വികസിപ്പിക്കുന്നതിനു മുമ്പ് വ്യത്യസ്തവും വിഭിന്നവുമായി ചിന്തിക്കേണ്ടതു അനിവാര്യമാണ്.

## നമുക്ക് ചുരുക്കിപ്പറയാം

ഒരു കമ്പ്യൂട്ടർ ഭാഷയിൽ ക്രമത്തിൽ എഴുതിയിരിക്കുന്ന നിർദേശങ്ങളാണ് പ്രോഗ്രാം. പ്രോഗ്രാമിങ് പ്രക്രിയ ചില ഘട്ടങ്ങളിലൂടെ കടന്നു പോകുന്നു. അൽഗോരിതവും ഫ്ളോച്ചാർട്ടും തയ്യാറാക്കുന്നത് പ്രോഗ്രാമിങ്ങിന്റെ യുക്തി വികസിപ്പിക്കാൻ സഹായിക്കുന്നു. HLL ൽ തയ്യാറാക്കിയിരിക്കുന്ന പ്രോഗ്രാമിനെ സോഴ്സ് കോഡ് എന്ന് വിളിക്കുന്നു. അതിനെ യന്ത്ര ഭാഷയിലേക്കു പരിഭാഷപ്പെടുത്തേണ്ടതാണ്. അതിന്റെ ഫലമായി ലഭിക്കുന്ന കോഡ്, ഒബ്ജക്റ്റ് കോഡ് എന്ന് അറിയപ്പെടുന്നു. ഡീബഗ്ഗിങ് എന്ന് വിളിക്കുന്ന പ്രക്രിയയിലൂടെ പ്രോഗ്രാമിൽ സംഭവിച്ചിരിക്കുന്ന തെറ്റുകളെ നീക്കം ചെയ്യുന്നു. പരിഭാഷപ്പെടുത്തിയ പതിപ്പ് കമ്പ്യൂട്ടർ പ്രവർത്തിപ്പിക്കുന്നു. ഉചിതമായ വിവരണം തയ്യാറാക്കുന്നതിന് പിൽക്കാലത്തു പ്രോഗ്രാമിൽ മാറ്റം വരുത്തുന്നതിന് സഹായകമാകുന്നു. പ്രശ്ന പരിഹാരത്തിന് വ്യത്യസ്തങ്ങളായ യുക്തികൾ പ്രയോഗിക്കാമെങ്കിലും പ്രോഗ്രാമിന്റെ പ്രകടനം അളക്കുന്നത് ടൈം കോംപ്ലക്സിറ്റിയുടെയും സ്പേസ് കോംപ്ലക്സിറ്റിയുടെയും അടിസ്ഥാനത്തിലാണ്.

## പഠന നേട്ടങ്ങൾ

ഈ അദ്ധ്യായം പൂർത്തിയാകുന്നതോടെ പഠിതാവിന്

- പ്രശ്ന പരിഹാരത്തിന്റെ വിവിധ വശങ്ങൾ വിശദീകരിക്കാൻ സാധിക്കുന്നു.
- പ്രശ്നങ്ങൾ പരിഹരിക്കാനുള്ള അൽഗോരിതങ്ങൾ വികസിപ്പിക്കാൻ സാധിക്കുന്നു.
- അൽഗോരിതത്തിൽ കൃത്യത ഉറപ്പു വരുത്താൻ ഫ്ളോച്ചാർട്ടുകൾ വരയ്ക്കാൻ സാധിക്കുന്നു.
- പ്രശ്നം പരിഹരിക്കാനുള്ള ഏറ്റവും നല്ല അൽഗോരിതം തിരഞ്ഞെടുക്കാൻ സാധിക്കുന്നു.

## മാതൃകാ ചോദ്യങ്ങൾ

### ഒറ്റ വാക്യത്തിൽ ഉത്തരം എഴുതുക

1. അൽഗോരിതം എന്നാലെന്ത് ?
2. പ്രശ്ന പരിഹാരത്തിൽ കമ്പ്യൂട്ടറിന്റെ പങ്ക് എന്താണ് ?
3. ഫ്ളോച്ചാർട്ടിൽ കണക്ടറിന്റെ ഉപയോഗമെന്ത് ?
4. പ്രോഗ്രാമിൽ ലോജിക്കൽ തെറ്റുകൾ എന്നാലെന്ത് ?

### ലഘു വിവരണാത്മകം

1. കമ്പ്യൂട്ടർ പ്രോഗ്രാം എന്നാലെന്ത് ? പ്രോഗ്രാമുകൾ തയ്യാറാക്കുന്നതിന് അൽഗോരിതങ്ങൾ എങ്ങനെ സഹായിക്കുന്നു ?
2. 3 സംഖ്യകളുടെ തുകയും ശരാശരിയും കണ്ടുപിടിക്കാനുള്ള അൽഗോരിതം എഴുതുക
3. ആദ്യത്തെ 100 എണ്ണൽ സംഖ്യകൾ പ്രദർശിപ്പിക്കാനുള്ള ഫ്ളോച്ചാർട്ട് വരയ്ക്കുക
4. ഫ്ളോച്ചാർട്ടുകളുടെ പരിമിതികൾ എന്തെല്ലാം?
5. ഡീബഗ്ഗിങ് എന്നാലെന്ത്?
6. ഒരു പ്രോഗ്രാമിൽ വിവരണം തയ്യാറാക്കുന്നതിന്റെ ആവശ്യകത എന്ത്?

### വിവരണാത്മകം

1. അൽഗോരിതത്തിന്റെ സവിശേഷതകൾ എന്തെല്ലാം?
2. ഫ്ളോച്ചാർട്ട് ഉപയോഗിക്കുന്നത് കൊണ്ടുള്ള ഗുണങ്ങൾ എന്തെല്ലാം?
3. പ്രോഗ്രാമിങ്ങിന്റെ വിവിധ ഘട്ടങ്ങളെ പറ്റി ചുരുക്കത്തിൽ വിവരിക്കുക

5

# C++ പ്രോഗ്രാമിങ് - ഒരു ആമുഖം

## പ്രധാന ആശയങ്ങൾ

- C++ -ലെ ക്യാരക്ടർ സെറ്റ്
- ടോക്കണുകൾ
  - കീവേഡുകൾ
  - ഐഡന്റിഫയറുകൾ
  - ലിറ്ററലുകൾ
  - പഞ്ചുവേറ്ററുകൾ
  - ഓപ്പറേറ്ററുകൾ
- ഇന്റഗ്രേറ്റഡ് ഡവലപ്മെന്റ് എൻവയോൺമെന്റ് (IDE)
  - ജിനി (IDE)


Bjarne Stroupstrup വികസിപ്പിച്ച ശക്തവും ജനപ്രിയവുമായ ഒരു ഒബ്ജക്ട് ഓറിയന്റഡ് പ്രോഗ്രാമിങ് ഭാഷയാണ് C++ (ഉച്ചരിക്കുന്നത് c പ്ലസ് പ്ലസ്). C++ എന്ന ആശയം വന്നത് C-യോടൊപ്പം + + ഓപ്പറേറ്റർ കൂടി ചേർന്നാണ്. അങ്ങനെ C++ എന്നത് C ഭാഷയുടെ ഒരു വിപുലീകരിച്ച രൂപമായിത്തീർന്നു.

നാം കഴിഞ്ഞ അധ്യായത്തിൽ ചർച്ച ചെയ്ത ക്രമപ്പെടുത്തൽ, തിരഞ്ഞെടുക്കൽ, ആവർത്തനം തുടങ്ങിയ വിവിധ തരം പ്രോഗ്രാമിങ് ആശയങ്ങൾ പ്രാവർത്തികമാക്കാൻ C++ ഭാഷ ഉപയോഗിക്കുന്നു. ഈ അധ്യായത്തിൽ, C++ -ന്റെ അടിസ്ഥാന ആശയങ്ങളെക്കുറിച്ച് ഒരു അവലോകനം നടത്താം. C++ പ്രോഗ്രാം എഴുതുന്നതിനുള്ള വിവിധ ഭാഷാ പ്രൊസസ്സർ പാക്കേജുകളും നമുക്ക് പരിചയപ്പെടാം.

ഏതൊരു ഭാഷയേയും പോലെ, C++ ഭാഷാപഠനവും തുടങ്ങുന്നത് അതിലെ അടിസ്ഥാന ചിഹ്നങ്ങളായ അക്ഷരങ്ങൾ പരിചയപ്പെട്ടുകൊണ്ടാണ്. അതിനുശേഷം വാക്കുകൾ, വാക്യങ്ങൾ (എക്സ്പ്രഷനുകൾ), പ്രസ്താവനകൾ തുടങ്ങിയവയിലൂടെ പഠനക്രമം തുടരും. അക്ഷരങ്ങൾ പഠിച്ച് കൊണ്ട് നമുക്ക് തുടങ്ങാം.

## 5.1 ക്യാരക്ടർ സെറ്റ് (Character set)

നമുക്ക് അറിയാവുന്നത് പോലെ ഇംഗ്ലീഷ്, മലയാളം, ഹിന്ദി തുടങ്ങിയ ഏതൊരു ഭാഷാപഠനവും ആരംഭിക്കുന്നത് അക്ഷരമാലയിലാണ്. അതുപോലെ C++ ഭാഷയ്ക്കും അതിന്റേതായ അക്ഷരമാലയുണ്ട്. ഒരു പ്രോഗ്രാമിങ് ഭാഷയുടെ അക്ഷരമാലയെ അതിന്റെ ക്യാരക്ടർ സെറ്റ് എന്ന് വിളിക്കുന്നു. ഭാഷയിലെ അംഗീകരിക്കപ്പെട്ട ചിഹ്നങ്ങളുടെ ഗണമാണ് അത്. അതിൽ അക്ഷരങ്ങളും, അക്കങ്ങളും മറ്റ് ചിഹ്നങ്ങളും ഉൾപ്പെടുന്നു.

*ന്യൂജഴ്സിയിലെ (USA) മുറേഹില്ലിലുള്ള, AT&T ബെൽ ലബോറട്ടറിയിലാണ് Bjarne Stroustrup C++ വികസിപ്പിച്ചത്. ഇപ്പോൾ അദ്ദേഹം കൊളംബിയ സർവകലാശാലയിലെ സന്ദർശക അധ്യാപകനും ടെക്സാസ് A&M സർവകലാശാലയിലുള്ള കോളേജ് ഓഫ് എഞ്ചിനിയറിംഗിലെ കമ്പ്യൂട്ടർ ശാസ്ത്ര വിഭാഗത്തിന്റെ ചുമതലയുള്ളയാളുമാണ്. അദ്ദേഹത്തിന് നിരവധി പുരസ്കാരങ്ങൾ ലഭിച്ചിട്ടുണ്ട്. C++ ന്റെ ആദ്യകാല നാമം, ക്ലാസ്സുകളോട് കൂടിയ C എന്നായിരുന്നു. പിന്നീട് 1983-ൽ C++ എന്ന് പുനർനാമകരണം ചെയ്തു.*

*Bjarne Stroustrup*

C++ അക്ഷരമാല ചുവടെ ചേർക്കുംവിധം തരം തിരിച്ചിരിക്കുന്നു.

(i) അക്ഷരങ്ങൾ – A മുതൽ Z വരെ
a മുതൽ z വരെ

(ii) അക്കങ്ങൾ – 0 മുതൽ 9 വരെ

(iii) പ്രത്യേക ചിഹ്നങ്ങൾ – + - * / ^ \ ( ) [ ] { } = < > . ‘ “ $ , ; : % ? _ # @

(iv) വൈറ്റ് സ്പെയ്സുകൾ – സ്പെയ്സ് ബാർ, ഹൊറിസോണ്ടൽ ടാബ് (→), ക്യാരിയേജ് റിട്ടേൺ ↵, ന്യൂ ലൈൻ, ഫോം ഫീഡ് തുടങ്ങിയവ കൊണ്ട് ഉണ്ടാക്കുന്ന വൈറ്റ് സ്പെയ്സുകൾ

(v) മറ്റു ചിഹ്നങ്ങൾ – 256 ASCII ചിഹ്നങ്ങളിൽ ഏതിനേയും C++ ലിറ്ററൽ ആയോ ഡാറ്റ ആയോ പ്രക്രിയയ്ക്കു വിധേയമാക്കാൻ സാധിക്കും.

*അടുത്തടുത്ത വാക്കുകളും സംഖ്യകളും വേർതിരിക്കാൻ ഉപയോഗിക്കുന്ന അക്ഷരങ്ങളാണ് വൈറ്റ് സ്പെയ്സുകൾ.*

## 5.2 ടോക്കണുകൾ (Tokens)

അക്ഷരമാല പഠിച്ചതിന് ശേഷം അക്ഷരങ്ങൾ ചേർത്തുണ്ടാകുന്ന വാക്കുകളെക്കുറിച്ചുള്ള പഠനമാണ് അടുത്തത്. സ്വാഭാവിക ഭാഷയിലെ വാക്കുകൾക്ക് സമാനമാണ് C++ ഭാഷയിലെ ‘ടോക്കൺ’. ഒരു പ്രോഗ്രാം വികസിപ്പിക്കുന്നതിനുള്ള അടിസ്ഥാന ഘടകങ്ങളാണ് ടോക്കണുകൾ. അവ ലെക്സിക്കൽ യൂണിറ്റുകൾ എന്നും അറിയപ്പെടുന്നു. C++ -ൽ താഴെപ്പറയും വിധം അഞ്ച് തരത്തിലുള്ള ടോക്കണുകളുണ്ട്.

1 കീവേർഡുകൾ
2 ഐഡന്റിഫയറുകൾ
3 ലിറ്ററലുകൾ
4 പഞ്ചുവേറ്ററുകൾ
5 ഓപ്പറേറ്ററുകൾ

### 5.2.1 കീവേഡുകൾ (Keywords)

ഭാഷാ കംബയിലറിന് ഒരു പ്രത്യേക അർഥം നൽകുന്ന വാക്കുകളാണ് (ടോക്കണുകളാണ്) കീവേഡുകൾ. പ്രത്യേക കാര്യങ്ങൾക്കായി ഭാഷ മാറ്റിവച്ച വാക്കുകളായതിനാലും മറ്റാവശ്യങ്ങൾക്ക് പുനർനിർവചിക്കാൻ സാധിക്കാത്തതിനാലും ഇവയെ കീവേഡുകൾ എന്നു വിളിക്കുന്നു. C++ -ലെ 48 കീവേഡുകൾ പട്ടിക 5.1-ൽ അക്ഷരമാലാക്രമത്തിൽ രേഖപ്പെടുത്തിയിട്ടുണ്ട്. അവയുടെ ഉപയോഗം പിന്നീട് വിശദീകരിക്കാം.

| | | | | | |
|---|---|---|---|---|---|
| asm | continue | float | new | signed | try |
| auto | default | for | operator | sizeof | typedef |
| break | delete | friend | private | static | union |
| case | do | goto | protected | struct | unsigned |
| catch | double | if | public | switch | virtual |
| char | else | inline | register | template | void |
| class | enum | int | return | this | volatile |
| const | extern | long | short | throw | while |

*പട്ടിക 5.1 : C++ - ലെ കീവേഡുകൾ*

### 5.2.2 ഐഡന്റിഫയറുകൾ (Identifiers)

സ്ഥലങ്ങൾ, വ്യക്തികൾ, വസ്തുക്കൾ എന്നിവ തിരിച്ചറിയുന്നതിനായി നാം പേരുകൾ നൽകാറുണ്ട്. ഇതിനു വേണ്ടിയാണ് C++ -ൽ ഐഡന്റിഫയറുകൾ ഉപയോഗിക്കുന്നത്. മെമ്മറി സ്ഥാനങ്ങൾ, വാചകങ്ങൾ, ഫങ്ഷനുകൾ, ഒബ്ജക്ടുകൾ, ക്ലാസ്സുകൾ തുടങ്ങിയ പ്രോഗ്രാമിലെ ഘടകങ്ങൾക്ക് പേര് നൽകുന്നതിന് ഉപയോക്താവ് നിർവചിക്കുന്ന വാക്കുകളാണ് ഐഡന്റിഫയറുകൾ. മെമ്മറി സ്ഥാനങ്ങളുടെ ഐഡന്റിഫയറുകളെ വേരിയബിളുകൾ എന്ന് വിളിക്കുന്നു. വാചകങ്ങൾക്ക് നൽകുന്ന ഐഡന്റിഫയറുകളെ ലേബലുകൾ എന്ന് വിളിക്കുന്നു. ഒരു കൂട്ടം പ്രസ്താവനകൾ പ്രതിനിധാനം ചെയ്യുന്ന ഐഡന്റിഫയറുകളാണ് ഫങ്ഷൻ നാമങ്ങൾ.

ഒരു പ്രോഗ്രാമിൽ ഐഡന്റിഫയറുകൾ നിർമ്മിക്കുമ്പോൾ, ചില നിബന്ധനകൾ കൃത്യമായി പാലിക്കേണ്ടതാവശ്യമാണ്. ആ നിയമങ്ങൾ താഴെപ്പറയും വിധമാണ്.

- അക്ഷരങ്ങൾ, അക്കങ്ങൾ, അണ്ടർ സ്കോർ (_) തുടങ്ങിയവയുടെ ഒരു ശ്രേണിയാണ് ഐഡന്റിഫയർ.
- ഐഡന്റിഫയർ തുടങ്ങുന്നത് അക്ഷരത്തിലോ, അണ്ടർ സ്കോറിലോ ( _ ) ആയിരിക്കണം.
- വൈറ്റ് സ്പെയ്സോ പ്രത്യേക ചിഹ്നങ്ങളോ അനുവദനീയമല്ല.
- കീവേഡുകൾ ഐഡന്റിഫയറായി ഉപയോഗിക്കാൻ പാടില്ല.
- ചെറിയ അക്ഷരങ്ങളും വലിയ അക്ഷരങ്ങളും വ്യത്യസ്തമായാണ് C++-ൽ ഉപയോഗിക്കുന്നത്. അതായത് C++ കെയ്സ് സെൻസിറ്റീവാണ്.

സാധുവായ ചില ഐഡന്റിഫയറുകൾക്ക് ഉദാഹരണം: `Count, sumof2numbers, Average_Height, _1stRank, Main, FOR`

താഴെപ്പറയുന്നവ ചില അസാധുവായ ഐഡന്റിഫയറുകളാണ്. അതിനുള്ള കാരണങ്ങളും സൂചിപ്പിച്ചിരിക്കുന്നു.

| | | |
|---|---|---|
| Sum of digits | → | ശൂന്യസ്ഥലങ്ങൾ ഉപയോഗിച്ചിരിക്കുന്നു |
| 1styear | → | അക്കം ആദ്യത്തെ അക്ഷരമായി ഉപയോഗിച്ചിരിക്കുന്നു |
| First . Jan | → | പ്രത്യേക ചിഹ്നം (.) ഉപയോഗിച്ചിരിക്കുന്നു |
| for | → | അതൊരു കീവേഡാണ് |

നമുക്ക് ചെയ്യാം

താഴെ തന്നിരിക്കുന്നവയിൽ നിന്ന് അസാധുവായ ഐഡന്റിഫയറുകൾ കണ്ടെത്തുക. അതിനുള്ള കാരണവും എഴുതുക.

`Data_rec, _data, Idata, data1, my.file, asm, switch, goto, break`

## 5.2.3 ലിറ്ററലുകൾ (Literals)

11-ാം ക്ലാസ് വിദ്യാർഥികൾക്കായുള്ള ഏകജാലക പ്രവേശന സംവിധാനം പരിഗണിക്കുക. പ്രവേശന ഫോറത്തിൽ നിങ്ങൾ ജനനത്തീയതി നൽകിയിട്ടുണ്ടാകും. ജനനത്തീയതി ഒരിക്കലും മാറുന്നില്ല. ഗണിത ശാസ്ത്രത്തിൽ π ഒരു സ്ഥിരാംഗവും ഗുരുത്വാകർഷണ സ്ഥിരാംഗം 'g' യുടെ വില (9.8 മീ./സെ.) ഒരിക്കലും മാറാത്തതുമാണെന്ന് നമുക്കറിയാം. അതുപോലെ, C++ -ൽ ഡാറ്റാ ഇനങ്ങളെ പ്രതിനിധാനം ചെയ്യാൻ നാം ഉപയോഗിക്കുന്ന ടോക്കണുകളായ ലിറ്ററലുകൾ, പ്രോഗ്രാം പ്രവർത്തിച്ച് തീരും വരെ വില മാറ്റം വരാതെ തുടരുന്നു. അവയെ പലപ്പോഴും സ്ഥിരാംഗങ്ങൾ എന്ന് വിളിക്കുന്നു. ലിറ്ററലുകൾ താഴെ പറയും വിധം നാലായി തരം തിരിക്കപ്പെട്ടിരിക്കുന്നു.

1. ഇന്റിജർ ലിറ്ററൽ
2. ഫ്ളോട്ടിംഗ് പോയിന്റ് ലിറ്ററൽ
3. ക്യാരക്ടർ ലിറ്ററൽ
4. സ്ട്രിങ് ലിറ്ററൽ

### ഇന്റിജർ ലിറ്ററലുകൾ (പൂർണസംഖ്യാ സ്ഥിരാംഗങ്ങൾ)

1776, 707, -273 എന്നീ സംഖ്യകൾ പരിഗണിക്കുക. അവ പൂർണ ദശാങ്ക വിലകൾ സൂചിപ്പിക്കുന്ന ഇന്റിജർ സ്ഥിരാംഗങ്ങളാണ്. അക്കങ്ങൾ മാത്രം ചേർത്തുണ്ടായിട്ടുള്ള ടോക്കണുകളായ ഇന്റിജർ ലിറ്ററലുകൾ അവിഭാജ്യ ഘടകങ്ങളില്ലാത്ത പൂർണ സംഖ്യകളാണ്. ഇന്റിജർ ലിറ്ററലുകളുടെ സ്വഭാവ സവിശേഷതകൾ താഴെപ്പറയുന്നവയാണ്.

- ഒരു പൂർണ സംഖ്യാ സ്ഥിരാംഗത്തിന് ഒരു അക്കമെങ്കിലും ഉണ്ടായിരിക്കണം. ദശാംശം ഉണ്ടാകരുത്.

നമുക്ക് ചെയ്യാം

*താഴെത്തന്നിട്ടുള്ളവ സാധുവോ അസാധുവോ ആയ പൂർണ്ണ സംഖ്യാ സ്ഥിരാംഗങ്ങളായി തരംതിരിക്കുകയും അവ അസാധുവായതിന്റെ കാരണം കണ്ടെത്തുകയും ചെയ്യുക.*

| | | | | |
|---|---|---|---|---|
| 77,000 | 70 | 314. | -5432 | +15346 |
| +23267 | -.7563 | -02281+0 | 1234E56 | -9999 |

C++ -ൽ ദശാംശ സംഖ്യകൾ (ബേസ്-10) കൂടാതെ അഷ്ടാംഗസംഖ്യകളും (ബേസ്-8) ഷോഡശ സംഖ്യകളും (ബേസ്-16) ലിറ്ററലുകളായി (സ്ഥിരാംഗങ്ങളായി) ഉപയോഗിക്കുന്നു. അഷ്ടാംഗ സംഖ്യയെ സൂചിപ്പിക്കുന്നതിന് 0 (പൂജ്യം എന്ന ചിഹ്നം) മുന്നിൽ നൽകുകയും ഷോഡശ സംഖ്യയെ സൂചിപ്പിക്കുന്നതിന് 0x (പൂജ്യം, x) മുന്നിൽ നൽകുകയും ചെയ്യുന്നു. ഉദാഹരണമായി പൂർണസംഖ്യാ സ്ഥിരാംഗങ്ങളായ 75 (ബേസ്-10)-ഉം 0113 (ബേസ്-8)-ഉം 0 x 4B (ബേസ്-16)-ഉം എല്ലാം പരസ്പരം തുല്യമാണ്.

- സംഖ്യ പോസിറ്റീവ് എന്നോ നെഗറ്റീവ് എന്നോ കാണിക്കുന്ന + അല്ലെങ്കിൽ - ചിഹ്നം ആദ്യ അക്ഷരമായി വരാവുന്നതാണ്.
- ചിഹ്നം ഇല്ലാത്ത സംഖ്യ പോസിറ്റീവ് സംഖ്യയായി കണക്കാക്കുന്നു.
- മറ്റ് ചിഹ്നങ്ങൾ അനുവദിക്കില്ല.

### ഫ്ളോട്ടിങ് പോയിന്റ് (അസ്ഥിര ദശാംശ സംഖ്യ) ലിറ്ററലുകൾ

3.1459, $3.0x10^8$, $1.6x10^{-19}$, 3.0 തുടങ്ങിയ സംഖ്യകൾ നിങ്ങൾക്ക് പരിചിതമാണല്ലോ. ഇവ സാധുവായ നാല് വിലകളാണ്. ഒന്നാമത്തേത് $\pi$ (പൈ)-യുടെ വിലയും, രണ്ടാമത്തേത് പ്രകാശ വേഗത മീറ്റർ/സെക്കന്റിൽ ഉള്ളതും, മൂന്നാമത്തേത് ഇലക്ട്രോണിന്റെ വൈദ്യുത ചാർജും (വളരെ ചെറിയ സംഖ്യ) അവസാനത്തേത് 3 എന്ന സംഖ്യയെ അസ്ഥിര ദശാംശസംഖ്യ ലിറ്ററലായി പ്രസ്താവിച്ചിട്ടുള്ളതുമാണ്.

**അസ്ഥിര ദശാംശസംഖ്യ ലിറ്ററലുകൾ** അവിഭാജ്യ ഘടകങ്ങളുള്ള രേഖീയ സ്ഥിരാംഗങ്ങൾ എന്നും അറിയപ്പെടുന്നു. ഇവ അവിഭാജ്യ രൂപമായോ കൃത്യങ്കരൂപമായോ എഴുതാൻ സാധിക്കും.

രേഖീയ സ്ഥിരാംഗങ്ങൾ അവിഭാജ്യ രൂപത്തിൽ ചിഹ്നമുള്ളതോ ഇല്ലാത്തതോ ആയതും, അക്കങ്ങൾ ചേർന്നതും, അവയ്ക്കിടയിൽ ദശാംശമുള്ളവയുമായിരിക്കും. രേഖീയ സ്ഥിരാംഗങ്ങൾ അവിഭാജ്യ രൂപത്തിലെഴുതുന്നതിനുള്ള നിയമങ്ങൾ ചുവടെ ചേർക്കുന്നു.

- ഒരു രേഖീയ സ്ഥിരാംഗത്തിന് അതിന്റെ അവിഭാജ്യ രൂപത്തിൽ ഒരക്കവും ഒരു ദശാംശവും എങ്കിലും ഉണ്ടായിരിക്കണം.
- അതിന് ഒന്നുകിൽ + (അധികം) അല്ലെങ്കിൽ - (ന്യൂനം) ചിഹ്നം തുടക്കത്തിൽ ഉണ്ടാകാം.
- ചിഹ്നം ഇല്ലാത്ത ഒരു സ്ഥിരാംഗത്തെ അധിക ചിഹ്നമുള്ള സംഖ്യയായി കണക്കാക്കാം.

കൃത്യങ്കരൂപത്തിൽ ഒരു രേഖീയ സ്ഥിരാംഗത്തിന് രണ്ട് ഭാഗങ്ങളുണ്ടാകും - അപൂർണ സംഖ്യാ

ഭാഗവും (മാന്റിസ്സ) കൃത്യങ്കവും. ഉദാഹരണമായി, 5.8 എന്നത് $0.58 \times 10^1 = 0.58$ E1 എന്നെഴുതാനാകും. ഇതിൽ കൃത്യങ്കഭാഗം 1-ഉം (E ക്ക് ശേഷം വരുന്ന ഭാഗം) മാന്റിസ്സ ഭാഗം 0.58-ഉം (E യ്ക്ക് മുമ്പുള്ള ഭാഗം) ആണ്. E 1 എന്നത് $10^1$-നെ സൂചിപ്പിക്കുന്നു. കൃത്യങ്ക രൂപത്തിൽ രേഖീയ സ്ഥിരാംഗം എഴുതുന്നതിനുള്ള നിയമങ്ങൾ താഴെപ്പറയും വിധമാണ്.

- കൃത്യങ്ക രൂപത്തിൽ രേഖീയ സ്ഥിരാംഗങ്ങൾക്ക് രണ്ട് ഭാഗങ്ങളുണ്ടായിരിക്കും. ഒരു പൂർണ സംഖ്യാ ഭാഗവും ഒരു കൃത്യങ്കവും.
- അപൂർണ സംഖ്യാ ഭാഗം ഒന്നുകിൽ ഒരു പൂർണ സംഖ്യാ രൂപത്തിലോ അല്ലെങ്കിൽ സാധുവായ അവിഭാജ്യ രൂപത്തിലോ ആയിരിക്കും.
- അപൂർണ സംഖ്യാ ഭാഗത്തെ തുടർന്ന് E അല്ലെങ്കിൽ e എന്ന അക്ഷരവും കൃത്യങ്കവുമുണ്ടാകും.
- കൃത്യങ്കം പൂർണസംഖ്യ ആയിരിക്കണം.

താഴെത്തന്നിട്ടുള്ളവ സാധുവായ രേഖീയ സ്ഥിരാംഗങ്ങളാണ്.

| | | | |
|---|---|---|---|
| 52.0 | 107.5 | -713.8 | -.00925 |
| 453.E-5 | 1.25E08 | .212E04 | 562.0E09 |
| 152E+8 | 1520E04 | -0.573E-7 | -.097 |

ചില അസാധുവായ രേഖീയ സ്ഥിരാംഗങ്ങൾ കാരണസഹിതം ഇതോടൊപ്പം ചേർക്കുന്നു.

58,250.262 (കോമ ഉപയോഗിച്ചിരിക്കുന്നു), 5.8E (കൃത്യങ്ക ഭാഗം ഇല്ല), 0.58E2.3 (കൃത്യങ്കമായി അവിഭാജ്യ സംഖ്യ ഉപയോഗിച്ചിരിക്കുന്നു).

നമുക്ക് ചെയ്യാം

*താഴെ തന്നിട്ടുള്ളവയെ കാരണസഹിതം സാധുവോ അസാധുവോ ആയ രേഖീയ സ്ഥിരാംഗങ്ങളായി തരംതിരിക്കുക.*

| | | | | |
|---|---|---|---|---|
| 77, 00,000 | 7.0 | 3.14 | -5.0E5.4 | +53.45E-6 |
| +532.67. | .756E-3 | -0.528E10 | 1234.56789 | 34,56.24 |
| 4353 | +34/2 | 5.6E | 4356 | 0 |

## ക്യാരക്ടർ ലിറ്ററൽ

സാധാരണ ലിംഗഭേദത്തെ കാണിക്കുന്നതിനായി ആൺ (Male) എന്നതിന് 'M' അല്ലെങ്കിൽ 'm' എന്നും പെൺ (Female) എന്നതിന് 'F' അല്ലെങ്കിൽ 'f' എന്നും ഉപയോഗിക്കാറുണ്ട്. അതുപോലെ, അതെ (Yes) എന്നതിന് 'y' അല്ലങ്കിൽ 'Y' എന്നും അല്ല (No) എന്നതിന് 'n' അല്ലെങ്കിൽ 'N' എന്നും നാം ഉപയോഗിക്കുന്നു. ഇവയെല്ലാം ഒറ്റ അക്ഷരങ്ങളാണ്. പ്രോഗ്രാം പ്രവർത്തിച്ച് തീരും വരെ അവ വില മാറ്റമില്ലാതെ തുടരുന്നു. ഒരു ജോഡി ഏക സൂചകങ്ങൾക്കുള്ളിൽ (ഒറ്റ ഉദ്ധരണികൾക്ക് ഉള്ളിൽ) ഉപയോഗിച്ചിരിക്കുന്ന അക്ഷരത്തെ ക്യാരക്ടർ ലിറ്ററൽ അല്ലെങ്കിൽ ക്യാരക്ടർ സ്ഥിരാംഗം എന്നു വിളിക്കുന്നു.

ഉദ്ധരണി ഇല്ലാതെ ഉപയോഗിക്കുന്ന X ഐഡന്റിഫയറും ഉദ്ധരണിക്കുള്ളിൽ ഉപയോഗിക്കുന്ന 'X' ക്യാരക്ടർ ലിറ്ററലുമാണ്. ക്യാരക്ടർ സ്ഥിരാംഗത്തിന്റെ വില എന്നത് അതിന്റെ ASCII വിലയാണ്. 'c' -യുടെ വില എന്നത് 'c' -യുടെ ASCII വിലയായ 99-ഉം 'A'-യുടെ വില എന്നത് അതിന്റെ ASCII വിലയായ 65-ഉം ആണ്.

C++ ഭാഷയിൽ ചില ചിത്രീകരിക്കാനാകാത്ത ക്യാരക്ടർ സ്ഥിരാംഗങ്ങളുണ്ട്, അവയെ കീ ബോർഡിൽ നിന്ന് നേരിട്ട് ടൈപ്പ് ചെയ്യാൻ സാധിക്കില്ല. ഉദാഹരണമായി, ക്യാരിയേജ് റിട്ടേൺ അല്ലെങ്കിൽ എന്റർ കീ, ടാബ് കീ, ബാക്ക് സ്പെയ്സ് കീ എന്നിവ ഒരു രീതിയിലും പ്രകടിപ്പിക്കാൻ സാധ്യമല്ല. ഇത്തരം പ്രകടിപ്പിക്കാനാകാത്ത ചിഹ്നങ്ങളെ പ്രതിനിധീകരിക്കാനായി **എസ്കേപ്പ് സീക്വൻസുകൾ** ഉപയോഗിക്കാം. അവയിൽ ബാക്ക് സ്ലാഷും തുടർന്ന് വരുന്ന ഒന്നോ അതിലധികമോ അക്ഷരങ്ങളും അടങ്ങിയിരിക്കും. എസ്കെയ്പ് സീക്വൻസുകൾ ഒരു ജോഡി ഏക സൂചകങ്ങൾക്കുള്ളിലുള്ള ഒന്നിലധികം അക്ഷരങ്ങൾ ചേർന്നതാണെങ്കിലും ആനുപാതികമായ ഒരു ASCII കോഡുപയോഗിച്ചാണ് അത് സൂചിപ്പിക്കുന്നത് എന്ന കാര്യം ശ്രദ്ധിക്കേണ്ടതാണ്. അതുകൊണ്ടാണ് അവയെ ക്യാരക്ടർ സ്ഥിരാംഗങ്ങളായി കണക്കാക്കുന്നത്. പട്ടിക 5.2-ൽ എസ്കേപ്പ് സീക്വൻസുകളും ആനുപാതികമായ ക്യാരക്ടറുകളും കുറിച്ചിട്ടുണ്ട്.

പട്ടിക 5.2-ൽ \', \", \? എന്നിവ നിങ്ങൾക്ക് കാണാൻ സാധിക്കും. ഈ ചിഹ്നങ്ങൾ (അക്ഷരങ്ങൾ) കീ ബോർഡിൽ നിന്ന് ടൈപ്പ് ചെയ്യാൻ സാധിക്കുമെങ്കിലും എസ്കേപ്പ് സീക്വൻസ് ഇല്ലാതെ ഉപയോഗിച്ചാൽ അവ പ്രത്യേക ഉദ്ദേശ്യവും കാര്യവുമായിരിക്കും നിർവഹിക്കുന്നത്. അവ അതുപോലെ തന്നെ പ്രദർശിപ്പിക്കുകയോ അച്ചടിക്കുകയോ ചെയ്യേണ്ടി വന്നാൽ എസ്കേപ്പ് സീക്വൻസ് ഉപയോഗിക്കണം. ചില സാധുവായ ക്യാരക്ടർ സ്ഥിരാംഗങ്ങൾക്ക് ഉദാഹരണങ്ങളാണ്:

's', 'S', '$', '\n', '+', '9'

ചില അസാധുവായ ക്യാരക്ടർ സ്ഥിരാങ്കങ്ങൾ കാരണ സഹിതം തന്നിരിക്കുന്നു.

A (ഏക ഉദ്ധരണി ഇല്ല), '82' (ഒന്നിലധികം അക്ഷരങ്ങൾ), "K" (ഏക ഉദ്ധരണിക്ക് പകരം ജോഡിയായ ഉദ്ധരണികൾ), '\g' (അസാധുവായ എസ്കേപ്പ് സീക്വൻസ് അല്ലെങ്കിൽ ഒന്നിലധികം അക്ഷരങ്ങൾ).

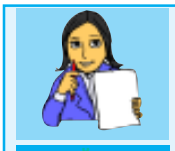
| എസ്കേപ്പ് സീക്വൻസ് | ആനുപാതികമായ ചിത്രീകരിക്കാനാകാത്ത ചിഹ്നങ്ങൾ |
|---|---|
| \a | ഓഡിയബിൾ ബെൽ (അലെർട്ട്) |
| \b | ബാക്ക് സ്പെയ്സ് |
| \f | ഫോം ഫീഡ് |
| \n | ന്യൂലൈൻ അല്ലെങ്കിൽ ലൈൻഫീഡ് |
| \r | ക്യാരിയേജ് റിട്ടേൺ |
| \t | ഹൊറിസോണ്ടൽ ടാബ് |
| \v | വെർട്ടിക്കൽ ടാബ് |
| \\ | ബാക്ക് സ്ലാഷ് |
| \' | സിംഗിൾ ക്വോട്ട് (ഏകസൂചകം) |
| \" | ഡബിൾ ക്വോട്ട് (ജോഡിയായ സൂചകം) |
| \? | ക്വസ്റ്റ്യൻ മാർക്ക് (ചോദ്യ ചിഹ്നം) |
| \0 | നൾ ക്യാരക്ടർ |

*പട്ടിക 5.2: എസ്കേപ്പ് സീക്വൻസുകൾ*

ഒക്ടൽ സംഖ്യകളെയും ഹെക്സാ ഡെസിമൽ സംഖ്യകളെയും എസ്കേപ്പ് സീക്വൻസുകളുടെ സഹായത്താലാണ് C++-ൽ സൂചിപ്പിക്കുന്നത്. \0n-ഉം \xHn-ഉം യഥാക്രമം ഒക്ടൽ സംഖ്യാ സമ്പ്രദായത്തിലും ഹെക്സാ ഡെസിമൽ സംഖ്യാന സമ്പ്രദായത്തിലും ഉള്ള സംഖ്യകളെ സൂചിപ്പിക്കുന്നു.

### സ്ട്രിങ് ലിറ്ററൽ

നന്ദന ഒരു വിദ്യാർഥിനിയാണ്. അവൾ ബാപ്പുജി നഗറിൽ താമസിക്കുന്നു. ഇവിടെ "Nandana" എന്നത് ഒരു പെൺകുട്ടിയുടെ പേരും "Bapuji Nagar" എന്നത് ഒരു സ്ഥല നാമവുമാണ്. ഈ രീതിയിലുള്ള ഡാറ്റ, പ്രോഗ്രാമുകളുടെ സഹായത്താൽ പ്രോസസ്സ് ചെയ്യേണ്ടി വരും. അത്തരം ഡാറ്റയെ സ്ട്രിങ് സ്ഥിരാംഗങ്ങളായി പരിഗണിക്കുകയും ജോഡികളായ ഉദ്ധരണികൾക്കുള്ളിൽ ഉൾപ്പെടുത്തുകയും ചെയ്യുന്നു. തുടർച്ചയായ ഒന്നോ അതിലധികമോ അക്ഷരങ്ങൾ (ചിഹ്നങ്ങൾ), രണ്ട് ജോടിയായ സൂചകങ്ങൾക്കുള്ളിൽ (ഉദ്ധരണികൾക്കുള്ളിൽ) അടക്കം ചെയ്തതാണ് സ്ട്രിങ് സ്ഥിരാംഗങ്ങൾ. ഉദാഹരണമായി "Hello friends", "123", "C++", "Baby\'s Day Out" എന്നിവ സാധുവായ സ്ട്രിങ് സ്ഥിരാംഗങ്ങളാണ്.



*താഴെ തന്നിട്ടുള്ളവ പല വിഭാഗങ്ങളിലുള്ള ലിറ്ററലുകളായി (സ്ഥിരാംഗങ്ങളായി) തരംതിരിക്കുക.*

```
'a'     'rita'     -124      12.5      -12e-1
"raju\'s pen"   0          -11.999   '\\'      32760
```

## 5.2.4 പഞ്ചുവേറ്ററുകൾ (പൂർണ വിരാമ ചിഹ്നങ്ങൾ) (Punctuators)

വാക്യങ്ങളിലെ വ്യാകരണത്തിന്റെ പൂർണതയ്ക്കായി പഞ്ചുവേഷൻ അടയാളങ്ങൾ ഇംഗ്ലീഷ്, മലയാളം തുടങ്ങിയ ഭാഷകളിലുപയോഗിക്കുന്നുണ്ട്. ഉദാഹരണമായി, 'ആരാണ് C++ വികസിപ്പിച്ചത്?' എന്ന വാക്യം ശ്രദ്ധിക്കുക. ഇവിടെ '?' എന്നത് പഞ്ചുവേഷൻ അടയാളവും അത് സൂചിപ്പിക്കുന്നത് ഒരു ചോദ്യ പ്രസ്താവനയെയുമാണ്. അതുപോലെ എല്ലാ വാക്യങ്ങളുടെയും അവസാനം നാം പൂർണവിരാമ ചിഹ്നം (.) ഇടുന്നു. അത്തരത്തിൽ അർഥ സംബന്ധമായോ, പദവിന്യാസപരമായോ ഉള്ള പൊരുൾ കംബയിലറിന് എത്തിക്കുവാനായി C++ -ലും ചില പ്രത്യേക ചിഹ്നങ്ങൾ ഉണ്ട്. അവയെ പഞ്ചുവേറ്ററുകൾ എന്ന് വിളിക്കുന്നു.

അവയ്ക്കുള്ള ഉദാഹരണങ്ങളാണ് # ; ' " ( ) { } [ ] എന്നിവ. ഓരോ പഞ്ചുവേറ്ററിന്റെയും ഉപയോഗം പിന്നീട് ചർച്ച ചെയ്യാം.

## 5.2.5 ഓഷറേറ്ററുകൾ (Operators)

5-ഉം 3-ഉം തമ്മിൽ കൂട്ടേണ്ടി വരുമ്പോൾ, അവ 5+3 എന്ന് നാം സൂചിപ്പിക്കുന്നു. ഇവിടെ + എന്നത് സങ്കലനം എന്ന പ്രവൃത്തിയെ കാണിക്കുന്ന ഓപ്പറേറ്റർ ആണ്. C++-ൽ ഇതുപോലെയുള്ള ധാരാളം ഓപ്പറേറ്ററുകൾ ഉണ്ട്. കംബയിലറിനോട് ഒരു പ്രത്യേക പ്രവൃത്തിയെക്കുറിച്ച് പറയുന്നതിന് ഉപയോഗിക്കുന്ന ചിഹ്നമാണ് ഓപ്പറേറ്റർ. ഏതെങ്കിലും തരത്തിലുള്ള പ്രവൃത്തിയെ ഉത്തേജിപ്പിക്കുന്ന ടോക്കണുകളാണ് അവ. ഓപ്പറന്റുകൾ എന്ന് വിളിക്കുന്ന ഒരു കൂട്ടം ഡാറ്റാകളിൽ ഓപ്പറേറ്ററുകൾ പ്രയോഗിക്കുന്നു. അരിത്മെറ്റിക്, ലോജിക്കൽ, റിലേഷണൽ, കണ്ടീഷണൽ, അസൈൻമെന്റ് തുടങ്ങിയ വിവിധ തരം ഓപ്പറേറ്ററുകൾ C++ -ൽ ഉണ്ട്. ഓപ്പറേറ്ററുകളെക്കുറിച്ച് കൂടുതലായി നമുക്ക് അടുത്ത അധ്യായത്തിൽ ചർച്ച ചെയ്യാം.

*താഴെ തന്നിട്ടുള്ളവ വിവിധയിനം ടോക്കണുകളായി തരംതിരിക്കുക.*

```
/     -24              +     -12e-1    '"KL01"
Sum   "raju\'s pen"    if    rita        '\\'
break    }
```

## 5.3 ഇന്റഗ്രേറ്റഡ് ഡവലപ്മെന്റ് എൻവയോൺമെന്റ് – IDE

ഒരു C++ പ്രോഗ്രാമിലെ അടിസ്ഥാന ഘടകങ്ങൾ നാം ഇപ്പോൾ പഠിച്ചു കഴിഞ്ഞു. C++ പ്രോഗ്രാമുകൾ എഴുതിത്തുടങ്ങുന്നതിന് മുൻപ് എവിടെയാണ് പ്രോഗ്രാം ടൈപ്പ് ചെയ്യേണ്ടതെന്ന് നാം അറിഞ്ഞിരിക്കണം. മറ്റ് പ്രോഗ്രാമിങ് ഭാഷകളെപ്പോലെ ഒരു ടെക്സ്റ്റ് എഡിറ്റർ ഉപയോഗിച്ച് C++ പ്രോഗ്രാം നിർമ്മിക്കാം. C++ പ്രോഗ്രാമുകൾ സൃഷ്ടിക്കാനായി ടർബോ C++, ബോറോലാന്റ് C++, GCC തുടങ്ങിയ കമ്പയിലറുകൾ IDE ലഭ്യമാക്കുന്നു. ഇവയിൽ പല IDE-കളിലും ടൈപ്പിങ്, എഡിറ്റിങ്, സേർച്ചിങ്, ലിങ്കിങ്, കമ്പയിലിങ്, എക്സിക്യൂട്ടിങ് എന്നീ സൗകര്യങ്ങൾ ഉണ്ട്. ഇത്തരം പ്രവർത്തനങ്ങൾക്കായി നാം ഇവിടെ ജിനി IDE (ഐ.ടി.@ സ്കൂൾ ഉബുണ്ടു ലിനക്സ് 14.04) ഉപയോഗിക്കുന്നു.

### GCC യോടൊപ്പമുള്ള ജിനി IDE

ലിനക്സ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തോടൊപ്പം ലഭ്യമാകുന്ന ഒരു സ്വതന്ത്ര സോഫ്റ്റ്വെയറാണ് GCC കമ്പയിലർ. GCC എന്നത് GNU കമ്പയിലർ കളക്ഷനെ പ്രതിനിധാനം ചെയ്യുന്നതും ISO C++ മാനദണ്ഡങ്ങൾ അനുസരിച്ച് പ്രവർത്തിക്കുന്നതുമായ ഒരു ജനപ്രിയ കമ്പയിലറാണ്. C++ പ്രോഗ്രാമുകൾ എഴുതുന്നതിനും, കമ്പയിൽ ചെയ്യുന്നതിനും എക്സിക്യൂട്ട് ചെയ്യുന്നതിനുമുള്ള ഒരു ക്രോസ് - പ്ലാറ്റ്ഫോം IDE ആണ് ജിനി.

### A. എഡിറ്റ് വിൻഡോ തുറക്കുന്ന വിധം

ഉബുണ്ടു ലിനക്സിന്റെ ആപ്ലിക്കേഷൻസ് മെനുവിൽ നിന്നുമാണ് ജിനി IDE-യുടെ എഡിറ്റ് വിൻഡോ തുറക്കുന്നത്.

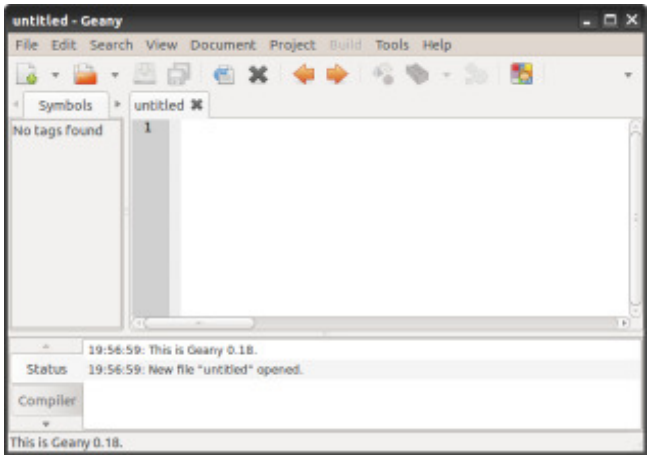
**അപ്ലിക്കേഷനുകൾ → പ്രോഗ്രാമിങ് →ജിനി**

ജിനി IDE അതിന്റെ വിൻഡോ തുറക്കുന്നത് ചിത്രം 5.1-ൽ കാണിച്ചിരിക്കും വിധമാണ്. അതിന് ഒരു ടൈറ്റിൽ ബാർ, മെനു ബാർ, ടൂൾ ബാർ, കോഡ് എഡിറ്റ് ചെയ്യാനുള്ള സ്ഥലം എന്നിവ ഉണ്ടാകും. അൺ ടൈറ്റിൽഡ് എന്ന പേരിലുള്ള ഒരു ടാബും അവിടെ കാണാൻ സാധിക്കും. വിൻഡോസ് ഓപ്പറേറ്റിംഗ് സിസ്റ്റത്തിൽ ജിനി 1.24 ആണ് നാം ഉപയോഗിക്കുന്നതെങ്കിൽ, തുറക്കുന്ന ജാലകം (വിൻഡോ) ചിത്രം 5.2-ൽ കാണിച്ചിരിക്കും വിധമായിരിക്കും. മുകളിൽ പരാമർശിച്ച രണ്ട് ജാലകങ്ങളും (വിൻഡോകളും) സമാനമാണെന്ന് നമുക്ക് കാണാം.

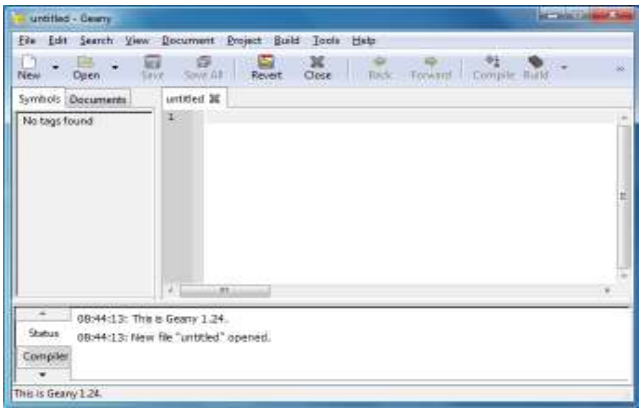
അൺടൈറ്റിൽഡ് എന്ന തനത് പേരുള്ള ഫയലിൽ നമുക്ക് പ്രോഗ്രാം എഡിറ്റ് ചെയ്യാം. ഒരു പുതിയ ഫയൽ തുറക്കാനായി ഫയൽ മെനുവിലെ ന്യൂ ഓപ്ഷൻ തിരഞ്ഞെടുക്കുകയോ ടൂൾ ബാറിലുള്ള ന്യൂ ബട്ടൺ  ക്ലിക്ക് ചെയ്യുകയോ ആകാം. ഇതേ ആവശ്യത്തിനായി Ctrl+N എന്നീ കീ-കൾ ഒരുമിച്ചും ഉപയോഗിക്കാം.

## B. പ്രോഗ്രാം സേവ് ചെയ്യുന്നു

ഒരിക്കൽ ഫയൽ തുറന്നാൽ C++ പ്രോഗ്രാം നൽകുകയും അനുയോജ്യമായ ഫയൽ നാമത്തോടൊപ്പം .cpp എന്ന എക്സ്റ്റൻഷൻ കൊടുത്ത് അത് സേവ് ചെയ്യുകയും ആകാം. GCC എന്നത് ഒരു കൂട്ടം കമ്പയിലറുകൾ ആയതിനാൽ കോഡിന്റെ കമ്പയിലേഷന് വേണ്ടി ഏത് കമ്പയിലർ തിരഞ്ഞെടുക്കണമെന്ന് തീരുമാനിക്കുന്നത് അതിന്റെ എക്സ്റ്റൻഷൻ നോക്കിയാണ്. അതുകൊണ്ട് ഫയൽ എക്സ്റ്റൻഷൻ കൃത്യമായി നൽകിയിരിക്കണം. പ്രോഗ്രാം ടൈപ്പ് ചെയ്യുന്നതിന് മുൻപായി നാം ഫയൽ നാമം നൽകുകയാണെങ്കിൽ, പ്രോഗ്രാമിലുള്ള വിവിധയിനം ടോക്കണുകളെ വേർതിരിച്ച് കാണിക്കാനായി GCC സ്വയം പല നിറങ്ങൾ ലഭ്യമാക്കും. സോഴ്സ് കോഡിലെ വാചകങ്ങളുടെ സ്ഥാനം തിരിച്ചറിയുന്നതിനുള്ള ഇൻഡന്റേഷനും ഇത് ഉപയോഗിക്കുന്നു. ഇൻഡന്റേഷൻ എന്ന ആശയം നമുക്ക് പിന്നീട് ചർച്ച ചെയ്യാം.




*ചിത്രം 5.1: ഉബുണ്ടു ലിനക്സിലെ ജിനി IDE- - യുടെ പ്രാരംഭ സ്ക്രീൻ*



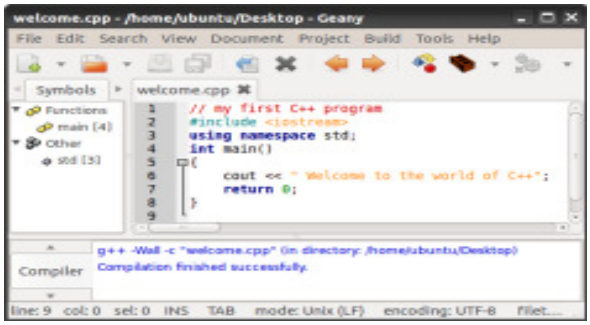

*ചിത്രം 5.2: വിൻഡോസ് ഓപ്പറേറ്റിങ് സിസ്റ്റത്തിൽ*

പ്രോഗ്രാം 5.1-ൽ കാണിച്ചിരിക്കുന്നത് പോലെ ലളിതമായ ഒരു പ്രോഗ്രാം നമുക്ക് എഴുതുകയും welcome.cpp എന്ന പേരിൽ അത് സേവ് ചെയ്യുകയും ആകാം.

**പ്രോഗ്രാം 5.1: IDE മനസ്സിലാക്കുന്നതിനുള്ള ഒരു പ്രോഗ്രാം.**

```
// my first C++ program
#include<iostream>
using namespace std;
int main()
{
      cout << "Welcome to the world of C++";
      return 0;
}  //end of program
```

പ്രോഗ്രാം 5.1 കൊടുത്തശേഷമുള്ള IDE വിൻഡോ ചിത്രം 5.3-ൽ കാണിച്ചിരിക്കുന്നു. ടോക്കണുകളിൽ ഉപയോഗിച്ചിട്ടുള്ള വിവിധ നിറങ്ങൾ നിരീക്ഷിക്കുക.




*ചിത്രം 5.3: ജിനി IDE-യിൽ ഒരു പേര് നൽകി പ്രോഗ്രാം സേവ് ചെയ്യുന്നു.*

പ്രോഗ്രാം സേവ് ചെയ്യുന്നതിനായി, ഫയൽ മെനുവിലെ സേവ് ഓപ്ഷൻ എടുക്കുകയോ കീബോർഡിലെ Ctrl+S എന്ന കുറുക്കുവഴി ഉപയോഗിക്കുകയോ ചെയ്യുക. ടൂൾ ബാറിലെ സേവ് ബട്ടൺ ക്ലിക്ക് ചെയ്ത് കൊണ്ടും ഫയൽ സേവ് ചെയ്യാവുന്നതാണ്. ഇടയ്ക്കിടെ Ctrl+S അമർത്തിക്കൊണ്ട് പ്രോഗ്രാം സേവ് ചെയ്യുന്നതാണ് ഉചിതമായ രീതി  .

അപ്രതീക്ഷിതമായ സിസ്റ്റത്തിന്റെ പിഴവോ, വൈദ്യുതിത്തകരാറോ മൂലം ഡാറ്റ നഷ്ടപ്പെടുന്നത് ഒഴിവാക്കാൻ ഇത് സഹായിക്കുന്നു. ഒരിക്കൽ, പ്രോഗ്രാം ടൈപ്പ് ചെയ്യുന്നത് പൂർണമായാൽ കമ്പയിൻ ചെയ്യുകയോ വ്യത്യാസം വരുത്തുകയോ ചെയ്യുന്നതിന് മുൻപായി ഫയൽ സേവ് ചെയ്യുന്നതാണ് ഏറ്റവും ഉചിതം. താൽക്കാലികമായ അസ്ഥിര പ്രാഥമിക മെമ്മറിയിൽ (വോളട്ടൈൽ പ്രൈമറി മെമ്മറിയിൽ) നിന്ന് ഒരു ഫയൽ ദീർഘകാലത്തേക്കുള്ള സുസ്ഥിരമായ ദ്വീതീയ മെമ്മറിയിലേക്ക് (നോൺ വോളട്ടൈൽ സെക്കന്ററി മെമ്മറിയിലേക്ക്) പകർത്തുന്നതിനെയാണ് പ്രോഗ്രാം സേവ് ചെയ്യുക എന്ന് പറയുന്നത്.

*വിവിധ C++ കമ്പയിലറുകൾക്കനുസരിച്ച് പല എക്സ്റ്റൻഷനുകളാണ് പ്രോഗ്രാം ഫയലുകൾക്ക് നൽകുന്നത്. അതായത് വിവിധ കമ്പയിലറുകൾ വ്യത്യസ്തങ്ങളായ ഫയൽ എക്സ്റ്റൻഷനുകൾ പിൻതുടരുന്നു. ഉദാഹരണമായി .cpp, .cxx, .cc, .c++ തുടങ്ങിയ എക്സ്റ്റൻഷനുകൾ ശ്രദ്ധിക്കുക.*

## C. പ്രോഗ്രാമിന്റെ കമ്പയിലിങ്ങും ലിങ്കിങ്ങും

പ്രോഗ്രാം കമ്പയിൽ ചെയ്യുകയും, തെറ്റ് കണ്ടെത്തിയാൽ തിരുത്തുകയും ചെയ്യുന്ന ഘട്ടമാണ് അടുത്തത്. ഇതിന് ബിൽഡ് മെനുവിലെ കമ്പയിൽ ഓപ്ഷൻ എടുക്കുകയോ കമ്പയിൽ ബട്ടൺ  ഉപയോഗിക്കുകയോ ചെയ്യാം. തെറ്റുകളുണ്ടായാൽ, ആ തെറ്റുകൾ താഴെ ഭാഗത്തുള്ള കമ്പയിലറിന്റെ സ്ഥിതി കാണിക്കുന്ന വിൻഡോയിൽ പ്രദർശിപ്പിക്കുകയും മറിച്ചാണെങ്കിൽ, കമ്പയിലേഷൻ വിജയകരമായി പൂർത്തീകരിച്ചു എന്ന സന്ദേശം പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്നു (ചിത്രം 5.3 പരിശോധിക്കുക). വിജയകരമായ കമ്പയിലേഷന് ശേഷം ലിങ്ക് ചെയ്യാനായി ടൂൾ ബാറിലെ ബിൽഡ് ബട്ടൺ ക്ലിക്ക്  ചെയ്യുകയോ ബിൽഡ് മെനുവിലെ ബിൽഡ് ഓപ്ഷൻ തിരഞ്ഞെടുക്കുകയോ ചെയ്യുക. ഇപ്പോൾ പ്രോഗ്രാം എക്സിക്യൂഷന് തയാറായിക്കഴിഞ്ഞിരിക്കുന്നു.

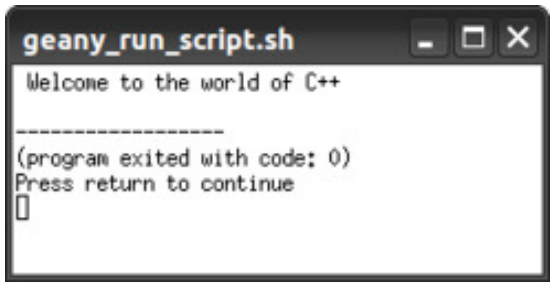
## D. പ്രോഗ്രാം റൺ/എക്സിക്യൂട്ട് ചെയ്യുക.

ഒരു കമ്പ്യൂട്ടർ പ്രോഗ്രാമിലെ നിർദേശങ്ങൾ കമ്പ്യൂട്ടർ പ്രാവർത്തികമാക്കുന്ന പ്രക്രിയയാണ് പ്രോഗ്രാം റണ്ണിംഗ്. പ്രോഗ്രാം റൺ ചെയ്യാനായി ബിൽഡ് മെനുവിലെ ഏക്സിക്യൂട്ട് ഓപ്ഷൻ

തിരഞ്ഞെടുക്കുക. ടൂൾ ബാറിലുള്ള എക്സിക്യൂട്ട് ബട്ടൺ ക്ലിക്ക്  ചെയ്തുകൊണ്ടും പ്രോഗ്രാം എക്സിക്യൂട്ട് ചെയ്യാവുന്നതാണ്. ഔട്ട്പുട്ട് ചിത്രം 5.4-ൽ കാണുന്നത് പോലെ ഒരു പുതിയ വിൻഡോയിൽ പ്രദർശിപ്പിക്കുന്നതാണ്.

## E. IDE യുടെ പ്രവർത്തനം അവസാനിപ്പിക്കുക

ഒരിക്കൽ നാം പ്രോഗ്രാം എക്സിക്യൂട്ട് ചെയ്യുകയും പ്രതീക്ഷിച്ച ഔട്ട്പുട്ട് ലഭിക്കുകയും ചെയ്ത് കഴിഞ്ഞാൽ, ഫയൽ മെനുവിൽ നിന്ന് ക്ലോസ് ഓപ്ഷൻ തിരഞ്ഞെടുത്തോ സജീവമായ ടാബിലെ ക്ലോസ് ബട്ടൺ (X) ക്ലിക്ക് ചെയ്തോ പ്രോഗ്രാം അവസാനിപ്പിക്കാം. പ്രോഗ്രാമിങ് തുടരുന്നുവെങ്കിൽ മുമ്പ് വിവരിച്ചത് പോലെ പുതിയ ഫയൽ തുറന്ന് അതിൽ പ്രോഗ്രാം ടൈപ്പ് ചെയ്യാവുന്നതാണ്. അല്ലെങ്കിൽ ടൈറ്റിൽ ബാറിലെ ക്ലോസ് ബട്ടൺ (X) ക്ലിക്ക് ചെയ്ത് IDE- യുടെ പ്രവർത്തനം അവസാനിപ്പിക്കാം. ഇതിനു പുറമേ Ctrl+Q കീ-കൾ ഒരുമിച്ചു ഉപയോഗിച്ചോ ഫയൽ മെനുവിൽ നിന്ന് ക്വിറ്റ് ഓപ്ഷൻ തിരഞ്ഞെടുത്തോ IDE-ൽ നിന്ന് പുറത്തുവരാം.




*ചിത്രം 5.4: ഔട്ട് പുട്ട് വിൻഡോ*

ഉബുണ്ടു ലിനക്സിലെ ജിനി IDE-യിൽ .cpp എന്ന എക്സ്റ്റൻഷൻ നൽകി സേവ് ചെയ്ത ഒരു സോഴ്സ് പ്രോഗ്രാം കമ്പയിൽ ചെയ്യുമ്പോൾ ലഭിക്കുന്ന ഫയലിന്റെ എക്സ്റ്റൻഷൻ .o (ഒബ്ജക്ട് ഫയൽ) എന്നായിരിക്കും. പ്രോഗ്രാം ലിങ്ക് ചെയ്യുമ്പോൾ ഒബ്ജക്ട് ഫയലിൽ നിന്ന് പ്രൊസസ്സറിന് എക്സിക്യൂട്ട് ചെയ്യാൻ വേണ്ട .out എന്ന എക്സ്റ്റൻഷനോട് കൂടിയ എക്സിക്യൂട്ടബിൾ ഫയൽ ലഭിക്കും. ഈ ഫയലാണ് പ്രൊസസ്സർ എക്സിക്യൂട്ട് ചെയ്യുന്നത്.

നമുക്ക് ചെയ്യാം

1. *"SMOKING IS INJURIOUS TO HEALTH" എന്ന സന്ദേശം സ്ക്രീനിൽ പ്രദർശിപ്പിക്കാനായി ഒരു പ്രോഗ്രാം എഴുതുക.*
2. *"TOBOCCO CAUSES CANCER" എന്ന സന്ദേശം മോണിട്ടറിൽ പ്രദർശിപ്പിക്കാനായി ഒരു പ്രോഗ്രാം എഴുതുക*

## നമുക്ക് സംഗ്രഹിക്കാം

1980-കളുടെ തുടക്കത്തിൽ Bjarne Stroustrup ആണ് C++ വികസിപ്പിച്ചത്. C++-ന് അതിന്റേതായ ക്യാരക്ടർ സെറ്റുണ്ട്. ടോക്കൺ എന്നത് പ്രോഗ്രാമിന്റെ ഏറ്റവും ചെറിയ ഘടകവും അവ നിർമ്മിക്കപ്പെട്ടിരിക്കുന്നത് C++-ലെ ഒന്നോ അതിലധികമോ അക്ഷരങ്ങൾ ചേർന്നുമാണ്. കീവേഡുകൾ, ഐഡന്റിഫയറുകൾ, ലിറ്ററലുകൾ, പഞ്ചുവേറ്ററുകൾ, ഓപ്പറേറ്ററുകൾ എന്നിങ്ങനെയുള്ള അഞ്ച് തരം ടോക്കണുകളുണ്ട്. പ്രോഗ്രാമുകൾ കമ്പ്യൂട്ടറിൽ എഴുതുന്നത് ഒരു എഡിറ്ററിന്റെ സഹായത്താലാണ്. സോഴ്സ് കോഡ് കമ്പ്യൂട്ടറിൽ നൽകുന്നതിനും അത് കമ്പയിൽ ചെയ്യുന്നതിനും ഒബ്ജക്ട് കോഡ് എക്സിക്യൂട്ട് ചെയ്യുന്നതിനും വേണ്ടി GCC-യും ജിനി IDE-യും പോലുള്ള സോഫ്റ്റ്വെയറുകൾ സൗകര്യമൊരുക്കുന്നു.

## പഠന നേട്ടങ്ങൾ

ഈ അധ്യായത്തിന്റെ പൂർത്തീകരണത്തിന് ശേഷം പഠിതാവിന് കഴിയും.

- C++-ലെ ക്യാരക്ടർ സെറ്റ് ലിസ്റ്റ് ചെയ്യുക.
- വിവിധ ടോക്കണുകളെ തരം തിരിക്കുക.
- കീവേഡുകളെ തിരിച്ചറിയുക.
- സാധുവായ ഐഡന്റിഫയറുകൾ എഴുതുക.
- ലിറ്ററലുകൾ തരം തിരിക്കുക.
- ജിനി IDE-യുടെ പ്രധാന ഭാഗങ്ങൾ തിരിച്ചറിയുക.
- ഒരു ലളിതമായ പ്രോഗ്രാം എഴുതി കമ്പയിൽ ചെയ്ത് റൺ ചെയ്യുക.

## മാതൃകാ ചോദ്യങ്ങൾ

**വളരെ ലഘുവായ ഉത്തരങ്ങൾ എഴുതുന്നവ.**

1. C++ ക്യാരക്ടർ സെറ്റിലെ വിവിധ ക്യാരക്ടറുകൾ എന്തെല്ലാം?
2. എന്താണ് എസ്കേപ്പ് സീക്വൻസ്?
3. ആരാണ് C++ വികസിപ്പിച്ചത്?
4. ടോക്കണുകൾ എന്നാലെന്ത്? C++-ലെ ടോക്കണുകളുടെ പേരെഴുതുക.
5. എന്താണ് C++-ലെ ഒരു ക്യാരക്ടർ സ്ഥിരാങ്കം?
6. എങ്ങനെയാണ് ചിത്രീകരിക്കാനാകാത്ത ക്യാരക്ടറുകൾ C++-ൽ പ്രതിനിധാനം ചെയ്യുന്നത്? ഉദാഹരണം നൽകുക.
7. \(സ്ലാഷ്), ‘ (ഏക ഉദ്ധരണി), “ (ജോഡിയായ ഉദ്ധരണി), ? (ചോദ്യച്ചിഹ്നം) തുടങ്ങിയ ക്യാരക്ടറുകൾ എസ്കേപ്പ് സീക്വൻസുകൾ ഉപയോഗിച്ച് ടൈപ്പ് ചെയ്യുന്നത് എന്ത് കൊണ്ടാണ്?
8. ഏത് എസ്കേപ്പ് സീക്വൻസുകളാണ് ന്യൂലൈൻ ക്യാരക്ടറിനേയും നൾ ക്യാരക്ടറിനെയും പ്രതിനിധാനം ചെയ്യാൻ ഉപയോഗിക്കുന്നത്?
9. ഒരു എസ്കേപ്പ് സീക്വൻസ് ..... ക്യാരക്ടറുകളെ പ്രതിനിധാനം ചെയ്യുന്നു.
10. താഴെ തന്നിട്ടുള്ളവയിൽ ഏതൊക്കെയാണ് C++-ലെ സാധുവായ ക്യാരക്ടറുകൾ/സ്ട്രിങ് സ്ഥിരാംഗങ്ങൾ?

    ‘c’ ‘anu’ “anu” mine ‘min’s’ “ ”

    ‘char ‘\ \’

11. എന്താണ് ഫ്ളോട്ടിങ് പോയിന്റ് സ്ഥിരാംഗം? അവ പ്രതിനിധാനം ചെയ്യുന്നതിനുള്ള വിവിധ മാർഗങ്ങൾ എന്തെല്ലാം?

12. എന്താണ് C++-ലെ സ്ട്രിങ് ലിറ്ററലുകൾ? ക്യാരക്ടർ സ്ഥിരാംഗങ്ങളും സ്ട്രിങ് ലിറ്ററലുകളും തമ്മിലുള്ള വ്യത്യാസമെന്ത്?

13. റൺ ചെയ്യുവാനുപയോഗിക്കുന്ന ഒരു C++ ഫയലിന്റെ എക്സ്റ്റൻഷൻ എന്താണ്?

14. താഴെ തന്നിട്ടുള്ളവയിൽ നിന്ന് അസാധുവായ ഐഡന്റിഫയറുകൾ കണ്ടെത്തുക. അസാധുവായതിന് കാരണമെഴുതുക.

(a) Principal amount (b) Continue (c) Area (d) Date-of-join (e) 9B

15. ലേബൽ എന്നത് C++-ലെ ഒരു .....ആണ്

(a) കീവേർഡ് (b) ഐഡന്റിഫയർ (c) ഓപ്പറേറ്റർ (d) ഫങ്ഷൻ

16. താഴെത്തന്നിട്ടുള്ള ടോക്കണുകൾ C++ പ്രോഗ്രാമിൽ നിന്ന് എടുത്തിട്ടുള്ളവയാണ്. താഴെ തന്നിട്ടുള്ള പട്ടികയിൽ അവ യഥാസ്ഥാനത്ത് നിരത്തുക.

(int, cin, %, do, =, “break”, 25.7, digit)

| കീവേഡുകൾ | ഐഡന്റിഫയറുകൾ | ലിറ്ററലുകൾ | ഓപ്പറേറ്ററുകൾ |
|---|---|---|---|
| | | | |

### ലഘു ഉത്തരങ്ങളെഴുതുന്നവ.

1. ഐഡന്റിഫയറുകളെ പരിപാലിക്കുന്ന നിയമങ്ങളെഴുതുക.

2. എന്താണ് C++-ലെ ടോക്കണുകൾ? എത്ര തരം ടോക്കണുകൾ C++-ൽ അനുവദിക്കുന്നു? അവ ലിസ്റ്റ് ചെയ്യുക.

3. കീവേഡുകളും, ഐഡന്റിഫയറുകളും വേർതിരിക്കുക

4. പൂർണ സംഖ്യാ സ്ഥിരാംഗങ്ങളെ C++-ൽ എങ്ങനെ പ്രതിനിധാനം ചെയ്യുന്നു? ഉദാഹരണ സഹിതം വിശദീകരിക്കുക

5. C++-ലെ ക്യാരക്ടർ സ്ഥിരാംഗങ്ങൾ എന്തെല്ലാം? എങ്ങനെ അവ പ്രയോഗത്തിൽ വരുത്തുന്നു?

### വിവരണാത്മക ഉത്തരങ്ങളെഴുതുന്നവ.

1. വിവിധയിനം ടോക്കണുകളെക്കുറിച്ച് ചുരുക്കി വിവരിക്കുക.

2. വിവിധയിനം ലിറ്ററലുകളെക്കുറിച്ച് ഉദാഹരണ സഹിതം വിശദീകരിക്കുക.

3. ജിനി IDE-യെക്കുറിച്ചും അതിന്റെ പ്രത്യേകതകളെക്കുറിച്ചും ചുരുക്കി വിശദീകരിക്കുക.

6

# ഡാറ്റ ഇനങ്ങളും ഓപ്പറേറ്ററുകളും

H9U7I1

## പ്രധാന ആശയങ്ങൾ

- ഡാറ്റ ഇനങ്ങൾ എന്ന ആശയം
- C++ ഡാറ്റ ഇനങ്ങൾ
- അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ
- ടൈപ്പ് മോഡിഫയറുകൾ
- വേരിയബിളുകൾ
- ഓപ്പറേറ്ററുകൾ
  - അരിത്മറ്റിക്
  - റിലേഷണൽ
  - ലോജിക്കൽ
  - ഇൻപുട്ട്/ഔട്ട്പുട്ട്
  - അസൈൻമെന്റ്
  - അരിത്മെറ്റിക് അസൈൻമെന്റ്
  - ഇൻക്രിമെന്റും ഡിക്രിമെന്റും
  - കണ്ടീഷണൽ
  - സൈസ് ഓഫ്
  - ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം
- പദപ്രയോഗങ്ങൾ
  - അരിതമെറ്റിക്
  - റിലേഷണൽ
  - ലോജിക്കൽ
- ഇനം മാറ്റൽ
- പ്രസ്താവനകൾ
  - പ്രഖ്യാപനം
  - അസൈൻമെന്റ്
  - ഇൻപുട്ട്/ഔട്ട്പുട്ട്
- ഒരു C++ പ്രോഗ്രാമിന്റെ ഘടന
  - പ്രീ പ്രോസസ്സർ നിർദ്ദേശകങ്ങൾ
  - ഹെഡർ ഫയലുകൾ
  - നെയിംസ്പേസ് എന്ന ആശയം
  - main () ഫങ്ഷൻ
  - ഒരു മാതൃകാ പ്രോഗ്രാം
- കോഡിംഗിനുള്ള മാർഗനിർദ്ദേശങ്ങൾ

ഇതിനു മുൻപുള്ള അധ്യായത്തിൽ C++ ന്റെ അടിസ്ഥാന നിർമ്മാണ ഘടകങ്ങളും പ്രോഗ്രാം വികസിപ്പിക്കുന്നതിനുപയോഗിക്കുന്ന IDE യും നാം പരിചയപ്പെട്ടു കഴിഞ്ഞു. കമ്പ്യൂട്ടറുകളിൽ നടക്കുന്ന പ്രധാന പ്രവർത്തനം ഡാറ്റ പ്രോസസ്സിംഗ് ആണെന്ന് നമുക്കറിയാമല്ലോ. എല്ലാ പ്രോഗ്രാമ്മിംഗ് ഭാഷകളും ഡാറ്റ കൈകാര്യം ചെയ്യുന്നതിന് പ്രാധാന്യം നൽകുന്നുണ്ട്. ചില പ്രത്യേകമായ സങ്കേതങ്ങൾ ഉപയോഗിച്ചുകൊണ്ടാണ് കമ്പ്യൂട്ടറുകളിൽ ഇൻപുട്ട് ചെയ്യപ്പെടുന്ന ഡാറ്റയുടെ ക്രമീകരണവും സംഭരണവും നടക്കുന്നത്. ഡാറ്റ സംഭരിക്കുന്നതിന് C++ന് മുൻകൂട്ടി നിർവചിച്ച ഒരു രൂപരേഖയുണ്ട്. സംഭരിക്കപ്പെട്ട ഡാറ്റ പിന്നീട് ഓപ്പറേറ്ററുകൾ ഉപയോഗിച്ച് പ്രോസസ്സ് ചെയ്യപ്പെടുന്നു. ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനങ്ങൾ എന്നു വിളിക്കുന്ന പുതിയ ഡാറ്റ ഇനങ്ങൾ നിർവചിക്കുന്നതിന് ഉപയോക്താവിനെ C++ അനുവദിക്കുന്നു.

C++ഭാഷയുടെ മുഖ്യ ആശയങ്ങളായ ഡാറ്റ ഇനങ്ങൾ, ഓപ്പറേറ്ററുകൾ, പദപ്രയോഗങ്ങൾ, പ്രസ്താവനകൾ എന്നിവയെക്കുറിച്ച് നമുക്ക് ഈ അദ്ധ്യായത്തിൽ വിശദമായി പഠിക്കാം.

## 6.1 ഡാറ്റ ഇനങ്ങൾ എന്ന ആശയം (Concept of data types)

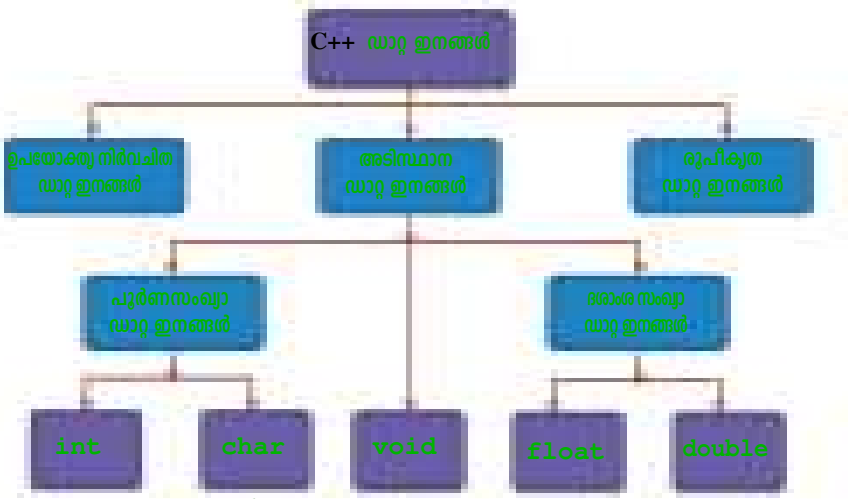
പരീക്ഷയ്ക്കുശേഷം ഒരു വിദ്യാർത്ഥിയുടെ പ്രോഗ്രസ് കാർഡ് തയ്യാറാക്കുന്നതിനായി നമുക്ക് അയാളുടെ രജിസ്റ്റർ നമ്പർ, റോൾ നമ്പർ, പേര്, വിലാസം, വിവിധ വിഷയങ്ങൾക്ക് ലഭിച്ച സ്കോർ, ഗ്രേഡുകൾ തുടങ്ങിയ ഡാറ്റ ആവശ്യമുണ്ട്. ഇത് കൂടാതെ, വിദ്യാർത്ഥിയുടെ സ്കോർ, ഹാജർ എന്നിവ ശതമാനത്തിൽ പ്രദർശിപ്പിക്കേണ്ടതുണ്ട്. ശാസ്ത്ര സംബന്ധിയായ ഡാറ്റ പ്രോസസിംഗ് പരിഗണിക്കുമ്പോൾ പ്രകാശത്തിന്റെ വേഗതയായ ($3x10^8 m/s$), ഗുരുത്വാകർഷണത്തിന്റെ വിലയായ ($9.8\ m/s^2$), ഇലക്ട്രോണിന്റെ ഇലക്ട്രിക് ചാർജ്ജായ ($-1.6x10^{-19}c$), തുടങ്ങിയ അക്കങ്ങളുടെ രൂപത്തിലുള്ള ഡാറ്റ ആവശ്യമായി വരാം.

മേൽ പറഞ്ഞതിൽ നിന്ന്, ക്യാരക്ടർ (character), ഇന്റിജർ (integer), ഭിന്ന സംഖ്യ (Real Number), വാക്കുകൾ (string) മുതലായ വിവിധതരം ഡാറ്റയുണ്ടെന്ന് നമുക്ക് അനുമാനിക്കാം. C++ലെ സാധുവായ ഒരു അക്ഷരം ഒരു ജോഡി ഏക ഉദ്ധരണികൾക്കുള്ളിൽ (single quotes) രേഖപ്പെടുത്തിയാൽ അത് ഒരു ക്യാരക്ടർ ഡാറ്റയെ പ്രതിനിധീകരിക്കുന്നു എന്ന് നാം കഴിഞ്ഞ അദ്ധ്യായത്തിൽ പഠിച്ചതാണല്ലോ. അതുപോലെ ദശാംശസ്ഥാനമില്ലാത്ത സംഖ്യകൾ ഇന്റിജർ (integer) ഡാറ്റയെ പ്രതിനിധാനം ചെയ്യുന്നു, ഭിന്നസംഖ്യകൾ ഫ്ളോട്ടിംഗ് പോയിന്റ് (floating point) ഡാറ്റ എന്നും, ഇരട്ട ഉദ്ധരണികളിൽ രേഖപ്പെടുത്തിയിരിക്കുന്നവ സ്ട്രിംഗ് (string) ഡാറ്റ എന്നും അറിയപ്പെടുന്നു. വിവിധ തരം ഡാറ്റ കൈകാര്യം ചെയ്യേണ്ടതിനാൽ എല്ലാ പ്രോഗ്രാമിംഗ് ഭാഷകളും അതിനുള്ള സംവിധാനം നൽകേണ്ടതാണ്. ഡാറ്റ ഇനങ്ങൾക്ക് പേരുകൾ നൽകികൊണ്ട് വിവിധതരം ഡാറ്റ കൈകാര്യം ചെയ്യുന്നതിനുള്ള സംവിധാനം C++ നൽകുന്നു. ഡാറ്റ ഇനങ്ങൾ (data types) എന്നാൽ ഡാറ്റയുടെ സ്വഭാവം, അതിൻമേൽ നടത്തുന്ന പ്രവർത്തനങ്ങൾ എന്നിവ തിരിച്ചറിയുന്നതിനുള്ള ഉപാധിയാണ്. ഈ സവിശേഷതകൾ വേർതിരിക്കുന്നതിനായി C++-ൽ വിവിധ ഡാറ്റ ഇനങ്ങൾ നിർവചിച്ചിരിക്കുന്നു.

അദ്ധ്യായം നാലിലെ അൽഗോരിതങ്ങളിൽ ഡാറ്റയെ സൂചിപ്പിക്കുവാൻ വേരിയബിളുകളാണ് നാം ഉപയോഗിച്ചത്. പ്രോഗ്രാമുകളിലും വേരിയബിളുകൾ തന്നെയാണ് ഡാറ്റയെ സൂചിപ്പിക്കുവാൻ ഉപയോഗിക്കുന്നത്. C++ ഭാഷയിൽ നാം പ്രോഗ്രാമുകൾ എഴുതുമ്പോൾ വേരിയബിളുകളെ അവ ഉപയോഗിക്കുന്നതിനുമുമ്പായി പ്രഖ്യാപിക്കേണ്ടതുണ്ട് (declaration of variable) ഈ വേരിയബിളുകൾ പ്രഖ്യാപിക്കുന്നതിന് ഡാറ്റ ഇനങ്ങൾ ആവശ്യമാണ്.

## 6.2 C++ ലെ ഡാറ്റ ഇനങ്ങൾ (C++ Data Types)

C++ വിവിധതരം ഡാറ്റാഇനങ്ങൾ കൊണ്ട് സമ്പുഷ്ടമാണ്. ഇവയെ സ്വഭാവം, വലിപ്പം, അവയുമായി ബന്ധപ്പെട്ട പ്രവർത്തനങ്ങൾ എന്നിവയുടെ അടിസ്ഥാനത്തിൽ ചിത്രം 6.1ൽ കാണുന്നതു പോലെ പലതായി തരം തിരിച്ചിട്ടുണ്ട്. ഡാറ്റ ഇനങ്ങളെ അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ അല്ലെങ്കിൽ അന്തർ നിർമിത ഡാറ്റ ഇനങ്ങൾ (built-in data types), രൂപീകൃത ഡാറ്റ ഇനങ്ങൾ (Derived data types), ഉപയോക്ത നിർവചിത ഡാറ്റ ഇനങ്ങൾ (user defined data types) എന്നിങ്ങനെ തരം തിരിച്ചിരിക്കുന്നു.




*ചിത്രം 6.1 : C++ ഡാറ്റ ഇനങ്ങളുടെ തരംതിരിക്കൽ*

**അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ:** C++ കംപൈലറിൽ അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ നിർവ്വചിച്ചിരിക്കുന്നു. അന്തർ നിർമിത ഡാറ്റ ഇനങ്ങൾ എന്നും അവ അറിയപ്പെടുന്നു. ഇവ വീണ്ടും വിഭജിക്കുവാൻ കഴിയാത്ത ഏറ്റവും ചെറിയ ഘടകങ്ങളാണ്. char, int, float, double, void എന്നിവയാണ് C++ലെ അഞ്ച് അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ. ഇവയിൽ int, char, എന്നിവ പൂർണസംഖ്യാ ഡാറ്റ ഇനത്തിനു കീഴിൽ വരുന്നതാണ്. പൂർണ്ണ സംഖ്യകളെ മാത്രമേ ഇവയ്ക്ക് കൈകാര്യം ചെയ്യാൻ കഴിയുകയുള്ളൂ. ഭിന്നസംഖ്യകൾ സാധാരണയായി ദശാംശ സംഖ്യാ ഡാറ്റ ഇനം (floating point data type) എന്ന് അറിയപ്പെടുന്നു. ഇവയെ വ്യാപ്തിയുടെയും (range) കൃത്യതയുടെയും (presion) അടിസ്ഥാനത്തിൽ float, double എന്നിങ്ങനെ രണ്ടായി തരം തിരിച്ചിരിക്കുന്നു.

**ഉപയോക്തൃ നിർവചിത ഡാറ്റ ഇനങ്ങൾ:** പ്രോഗ്രാമർക്ക് സ്വന്തമായി ഡാറ്റ ഇനം നിർവചിക്കുവാനുള്ള സൗകര്യം C++ൽ ഉണ്ട്. സ്ട്രക്ചർ (struct), എന്യൂമറേഷൻ (enum), യൂണിയൻ (union), ക്ലാസ്സ് (class) തുടങ്ങിയവ ഇത്തരം ഡാറ്റ ഇനങ്ങൾക്കുള്ള ഉദാഹരണങ്ങളാണ്.

**രൂപീകൃത ഡാറ്റ ഇനങ്ങൾ:** അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളെ ഗണങ്ങൾ ആക്കി മാറ്റിയോ വലിപ്പ മാറ്റം വരുത്തിയോ നിർമ്മിക്കപ്പെടുന്ന ഡാറ്റ ഇനങ്ങൾ രൂപീകൃത ഡാറ്റ ഇനങ്ങൾ എന്ന് അറിയപ്പെടുന്നു. അറൈകൾ, പോയിന്ററുകൾ, ഫങ്ഷനുകൾ തുടങ്ങിയവ രൂപീകൃത ഡാറ്റ ഇനങ്ങൾക്കുള്ള ഉദാഹരണങ്ങളാണ്.

## 6.3 അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ (fundamental data types):

അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ മൗലികമായ സ്വഭാവ വിശേഷമുള്ളവയാണ്. അവയെ വീണ്ടും ചെറിയ ഭാഗങ്ങളായി വിഭജിക്കുവാൻ കഴിയില്ല. കമ്പൈലറിൽ നിർചിക്കപ്പെട്ടിരിക്കുതിനാൽ അവയുടെ വലിപ്പം (അനുവദിക്കപ്പെട്ട മെമ്മറിയുടെ അളവ്) കമ്പൈലറിനെ ആശ്രയിച്ചിരിക്കുന്നു. നാം ഉപയോഗിക്കുന്നത് GCC -ൽ ലഭ്യമായ കമ്പൈലർ ആയതിനാൽ ഡാറ്റയുടെ വലിപ്പവും അതുപോലെ ഡാറ്റയുടെ വ്യാപ്തിയും അതിന് അനുസൃതമായിരിക്കും. ടർബോ C++ IDE പോലുള്ള കംപയിലറുകൾ നിങ്ങൾ ഉപയോഗിക്കുമ്പോൾ ഇത് വ്യത്യസ്തമാകാം. അഞ്ചു അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ താഴെ വിശദീകരിച്ചിരിക്കുന്നു.

**int ഡാറ്റഇനം (പൂർണ സംഖ്യകൾക്കായി):** ദശാംശ ഭാഗമില്ലാത്ത പൂർണ്ണ സംഖ്യകളാണ് ഇന്റിജറുകൾ. ഇവ പോസിറ്റീവോ, പൂജ്യമോ, നെഗറ്റീവോ ആകാം. ഒരു പ്രത്യേക പരിധിക്കുള്ളിലുള്ള ഇന്റിജറുകളെ പ്രതിനിധീകരിക്കാൻ ഉപയോഗിക്കുന്ന കീ വേർഡ് ആണ് int. int ഇനത്തിലുള്ള ഇന്റിജറുകൾക്ക് GCC നാല് ബൈറ്റ് മെമ്മറി അനുവദിക്കുന്നു. ആയതിനാൽ -2147483648 മുതൽ +2147483647 വരെയുള്ള സംഖ്യകളെ int ഡാറ്റാ ഇനം ഉപയോഗിച്ച് സൂചിപ്പിക്കാം. 69, 0, -112, 17, -32768, +32767 എന്നിവ int ഡാറ്റാ ഇനത്തിനുള്ള ഉദാഹരണങ്ങളാണ്. 22000000.00, - 2147483649 എന്നിവ അനുവദനീയമായ പരിധിക്ക് പുറത്തായതിനാൽ int ഡാറ്റാ ഇനമായി പരിഗണിക്കുകയില്ല.

**char ഡാറ്റ ഇനം (ക്യാരക്ടർ സ്ഥിരാംഗങ്ങൾക്കുവേണ്ടി) :** C++ ഭാഷയിലെ ക്യാരക്ടർ സെറ്റിലുൾപ്പെടുന്ന ചിഹ്നങ്ങൾ ആണ് ക്യാരക്ടറുകൾ. എല്ലാ അക്ഷരങ്ങൾ അക്കങ്ങൾ പ്രത്യേക ചിഹ്നങ്ങൾ വിരാമ ചിഹ്നങ്ങൾ തുടങ്ങിയവ ഈ വിഭാഗത്തിൽ ഉൾപ്പെടുന്നു. ഈ ക്യാരക്ടറുകൾ ഡാറ്റയായി ഉപയോഗിക്കുമ്പോൾ അവയെ C++ ലെ char ഡാറ്റഇനമായി പരിഗണിക്കപ്പെടുന്നു. Char കീ വേർഡ് C++ ലെ ക്യാരക്ടർ ലിറ്ററലുകളെ പ്രതിനിധീകരിക്കുന്നു എന്നു നമുക്ക് പറയാം.ഓരോ char ഇനത്തിലുള്ള ഡാറ്റായ്ക്കും 1 ബൈറ്റ് മെമ്മറി അനുവദിക്കുന്നു. 'a', '+', '/t', '0' മുതലായവ char ഡാറ്റാ ഇനത്തിൽപ്പെടുന്നവയാണ്.

char ഡാറ്റയെ ഇന്റിജർ ആയിട്ടാണ് പരിഗണിക്കുന്നത് എന്തുകൊണ്ടെന്നാൽ ASCII കോഡ് ഉപയോഗിച്ചാണ് കമ്പ്യൂട്ടർ ക്യാരക്ടറുകളെ തിരിച്ചറിയുന്നത്. മെമ്മറിയിൽ ക്യാരക്ടർ ഡാറ്റാ സംഭരിക്കുന്നത് അതിന്റേതായ ASCII കോഡ് ഉപയോഗിച്ചാണ്. ASCII കോഡ് ഇന്റിജറായതിനാലും അവ 8 ബിറ്റ്/1 ബൈറ്റ് സ്ഥലത്ത് സംഭരിക്കപ്പെടേണ്ടതിനാലും char ഡാറ്റാ ഇനത്തിന്റെ പരിധി -128 മുതൽ +127 വരെയാണ്.

**float ഡാറ്റഇനം (ദശാംശ സംഖ്യകൾക്കായി):** ദശാംശ ഭാഗത്തോടുകൂടിയ സംഖ്യകളെ ഫ്ളോട്ടിംഗ് പോയിന്റ് സംഖ്യകളെന്നു വിളിക്കുന്നു. കമ്പ്യൂട്ടറിൽ ഫ്ളോട്ടിംഗ് പോയിന്റ് സംഖ്യകൾ രേഖപ്പെടുത്തുന്നത് ശാസ്ത്രീയമായ പ്രതീകങ്ങൾ ഉപയോഗിച്ചാണ്. ഉദാഹരണത്തിന് -47281.97 എന്ന് സംഖ്യയെ ശാസ്ത്രീയ പ്രതീകങ്ങൾ ഉപയോഗിച്ചാണ് എഴുതുന്നത് $0.4728197 \times 10^5$ എന്നാണ്. ഇതിന്റെ ആദ്യഭാഗത്തെ (0.4728197) ഭിന്നാംശം (mantissa) എന്നും പത്തിന്റെ വർഗ്ഗമായ 5നെ ($10^5$) കൃത്യംങ്കം (exponent) എന്നും പറയുന്നു. ഫ്ളോട്ടിംഗ് പോയിന്റ് വിലകളെ പ്രതിനിധീകരിക്കാൻ കമ്പ്യൂട്ടർ സാധാരണയായി കൃത്യങ്ക മാതൃക (E-notation) ഉപയോഗിക്കുന്നു. അത് പ്രകാരം 47281.97 എന്നത് 0.4728197E5 ആയാണ് രേഖപ്പെടുത്തുന്നത്. Eക്ക് മുമ്പുള്ള സംഖ്യ ഭിന്നാംശവും Eക്ക് ശേഷമുള്ള സംഖ്യ കൃത്യങ്കവുമാണ്. C++ൽ float എന്ന കീ വേർഡാണ് ഇത്തരം സംഖ്യകളെ സൂചിപ്പിക്കാൻ ഉപയോഗിക്കുന്നത്. float ഡാറ്റ ഇനത്തിൽപ്പെട്ട സംഖ്യകൾക്ക് 4 ബൈറ്റ് മെമ്മറി GCC അനുവദിക്കുന്നു. ഈ ഡാറ്റ തരത്തിലുള്ള സംഖ്യകൾക്ക് സാധാരണയായി ദശാംശത്തിനുശേഷം 7 അക്കങ്ങൾ വരെ ആവാം.

**Double ഡാറ്റ ഇനം (ഡബിൾ പ്രിസിഷൻ ദശാംശ സംഖ്യകൾക്കായി):** ദശാംശത്തിനുശേഷം കൂടുതൽ അക്കങ്ങൾ വേണ്ട സംഖ്യകൾക്ക് അതായത് കൂടതുൽ കൃത്യത വേണ്ട ദശാംശ സംഖ്യകൾക്ക് ഉപയോഗിക്കുന്ന ഡാറ്റ ഇനമാണ് double. ഫ്ളോട്ട് ഡാറ്റാ ഇനം ഉപയോഗിച്ച് കൈകാര്യം ചെയ്യാവുന്ന സംഖ്യകളുടെ പരിധി ഈ ഡാറ്റാ ഇനം ഉപയോഗിച്ച് വർധിപ്പിക്കാവുന്നതാണ്. എന്തുകൊണ്ടെന്നാൽ ഈ ഡാറ്റാ ഇനം ഫ്ളോട്ട് ഡാറ്റാ ഇനത്തെക്കാൾ ഇരട്ടി മെമ്മറി ഉപയോഗിക്കുന്നു. C++ ൽ double ന്റെ കൃത്യതയും പരിധിയും കുറഞ്ഞത് float ന്റെ അത്രയെങ്കിലും ആയിരിക്കണം. gcc ഇത്തരത്തിലുള്ള ഡാറ്റ സംഭരിക്കുന്നതിന് 8 ബൈറ്റ് മെമ്മറി നീക്കിവച്ചിരിക്കുന്നു. ഡബിൾ ഡാറ്റാ ഇനത്തിൽ ദശാംശ സ്ഥാനത്തിനു ശേഷം 15 അക്കങ്ങൾ വരെ ആകാം.

**void ഡാറ്റ തരം (എം.റ്റി. സെറ്റ് ഡാറ്റക്കായി):** എംറ്റി സെറ്റിലെ ഡാറ്റയെ സൂചിപ്പിക്കാൻ ഉപയോഗിക്കുന്ന കീ വേഡാണ് വോയിഡ് (void). തീർച്ചയായും ഇതിന് മെമ്മറി ആവശ്യമില്ല. ഈ ഡാറ്റാ ഇനത്തിന്റെ വിശദമായ ഉപയോഗം അധ്യായം 10 ൽ ചർച്ച ചെയ്യാം.

അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളെ അവയുടെ വലിപ്പത്തിന്റെ അവരോഹണ ക്രമത്തിൽ double, float, int, char.void എന്ന് ക്രമീകരിക്കാം.

## 6.4 ടൈപ്പ് മോഡിഫയറുകൾ (Type modifiers)

വ്യാപ്തി വർദ്ധിപ്പിച്ചു കൊണ്ട് അധിക സാധനങ്ങൾ ഉൾപ്പെടുത്താവുന്ന തരത്തിലുള്ള യാത്രാ ബാഗുകൾ നിങ്ങൾ കണ്ടിട്ടുണ്ടോ? സാധാരണയായി നമ്മൾ ഈ അധിക സ്ഥലം ഉപയോഗിക്കാറില്ല. ബാഗിൾ ഘടിപ്പിച്ചിട്ടുള്ള സിബ് അവയുടെ വ്യാപ്തി കൂടുന്നതിനോ കുറയ്ക്കുന്നതിനോ നമ്മെ സഹായിക്കുന്നു. അൽപ്പം വലിപ്പം കൂടിയതോ കുറഞ്ഞതോ ആയ ഡാറ്റ ഉൾക്കൊള്ളാവുന്ന ഡാറ്റ ഇനങ്ങൾ C++ലും നമുക്ക് ആവശ്യമാണ്. അതി

നുള്ള C++ലെ സംവിധാനമാണ് ഡാറ്റാ ടൈപ്പ് മോഡിഫെയറുകൾ (type modifiers). ഇവ C++ൽ ഡാറ്റ ഇനങ്ങളുടെ വലിപ്പം (size), പരിധി (range), ദശാംശത്തിനുശേഷമുള്ള സംഖ്യയുടെ വലിപ്പം (precision) എന്നിവ ക്രമീകരിക്കാൻ ഉപയോഗിക്കുന്നു. വേരിയബിൾ പ്രഖ്യാപനത്തിൽ ഡാറ്റാ ഇനത്തിന്റെ പേരിന് മുൻപായി മോഡിഫെയറുകൾ ചേർക്കുന്നു. അതുകൊണ്ട് ഒരു ഡാറ്റാ ഇനത്തിന് അനുവദിച്ചിരിക്കുന്ന സ്ഥലവും ഡാറ്റയുടെ ചിഹ്നവും മാറ്റം വരുത്തി വിലകളുടെ പരിധി വ്യത്യാസപ്പെടുത്താൻ അനുവദിക്കുന്നു. signed, unsigned, long, short എന്നിവയാണ് പ്രധാനപ്പെട്ട മോഡിഫയറുകൾ.

ഡാറ്റ ഇനങ്ങളുടെ ശരിയായ വലിപ്പം നിങ്ങൾ ഉപയോഗിക്കുന്ന കംപ്യൂട്ടറിനെയും കംപയിലറിനെയും ആശ്രയിച്ചിരിക്കുന്നു. താഴെ പറയുന്നവ ഇത് ഉറപ്പു നൽകുന്നു.

- ഒരു double ഡാറ്റ ഇനത്തിന് ഏറ്റവും കുറഞ്ഞത് float ഡാറ്റ ഇനത്തിന്റെ വലിപ്പം ഉണ്ടാകണം.
- ഒരു long double ന് ഏറ്റവും കുറഞ്ഞത് double ഡാറ്റ ഇനത്തിന്റെയെങ്കിലും വലിപ്പമുള്ളതായിരിക്കും.

ഓരോ ഡാറ്റ ഇനങ്ങളും അവയുടെ മോഡിഫയറുകളും ടേബിൾ 6.1 ൽ കാണിച്ചിരിക്കുന്നു. (GCC കമ്പൈലറിനെ അടിസ്ഥാനമാക്കി)

| Name<br>പേര് | Description<br>വിശദീകരണം | Size<br>വലിപ്പം | Range<br>പരിധി |
|---|---|---|---|
| `char` | ക്യാരക്ടർ | 1<br>ബൈറ്റ് | `signed`: -128 to 127<br>`unsigned`: 0 to 255 |
| `short int (short)` | ഷോർട്ട് ഇന്റിജർ | 2<br>ബൈറ്റ്സ് | `signed`: -32768 to 32767<br>`unsigned`: 0 to 65535 |
| `int` | ഇന്റിജർ | 4<br>ബൈറ്റ്സ് | `signed`: -2147483648 to 2147483647<br>`unsigned`: 0 to 4294967295 |
| `long int (long)` | ലോംഗ് ഇന്റിജർ | 4<br>ബൈറ്റ്സ് | `signed`: -2147483648 to 2147483647<br>`unsigned`: 0 to 4294967295 |
| `float` | ഫ്ളോട്ടിംഗ് പോയിന്റ് നമ്പർ | 4<br>ബൈറ്റ്സ് | $-3.4\times10^{+/-38}$ to $+3.4\times10^{+/-38}$ ഏകദേശം 7 ദശാംശ സ്ഥാനങ്ങൾ |
| `double` | ഡബിൾ പ്രിസിഷനൻ ഫ്ളോട്ടിംഗ് പോയിന്റ് നമ്പർ | 8<br>ബൈറ്റ്സ് | $-1.7\times10^{+/-308}$ to $+1.7\times10^{+/-308}$ ഏകദേശം 15 ദശാംശ സ്ഥാനങ്ങൾ |
| `long double` | ലോംഗ് ഡബിൾ പ്രിസിഷനൻ ഫ്ളോട്ടിംഗ് പോയിന്റ് നമ്പർ | 12<br>ബൈറ്റ്സ് | $-3.4\times10^{+/-4932}$ to $+3.4\times10^{+/-4932}$ ഏകദേശം 19 ദശാംശ സ്ഥാനങ്ങൾ |

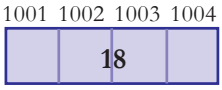
*പട്ടിക 6.1: ഡാറ്റ ഇനങ്ങളും ടൈപ്പ് മോഡിഫയറുകളും*

*ഡാറ്റ ഇനങ്ങൾ എങ്ങനെ വ്യത്യാസപ്പെട്ടിരിക്കുന്നു എന്ന് നിങ്ങൾക്ക് അറിയുന്നതിനുള്ള ഉദാഹരണങ്ങളാണ് ടേബിൾ 6.1 ൽ കൊടുത്തിരിക്കുന്നത്. ഇതിലുള്ള പല വിലകളും നിങ്ങളുടെ കമ്പ്യൂട്ടറിൽ വ്യത്യസ്തമായിരിക്കാം*

**വേരിയബിളുകൾ (variables):** ഡാറ്റ പരാമർശിക്കുന്നതിന് മെമ്മറിയിൽ അതിന്റെ സ്ഥാനങ്ങൾ തിരിച്ചറിയേണ്ടതുണ്ട്. മെമ്മറി സ്ഥാനങ്ങൾക്ക് നൽകുന്ന പേരുകളാണ് വേരിയബിളുകൾ. മെമ്മറി സ്ഥാനങ്ങളിൽ ഡാറ്റയെ സ്റ്റോർ ചെയ്യാനും വീണ്ടെടുക്കാനും ഉപയോഗിക്കുന്ന C++ -ലെ ഐഡന്റിഫയറുകളാണ് വേരിയബിളുകൾ. ഒരു വേരിയബിളിൽ സ്റ്റോർ ചെയ്തിട്ടുള്ള ഡാറ്റയുടെ സ്വഭാവവും അതിന്റെ വലിപ്പവും ആ വേരിയബിളിന്റെ ഡാറ്റ ഇനത്തിന് അനുസരിച്ചിരിക്കും. ഒരു വേരിയബിളിന് മൂന്ന് പ്രധാനപ്പെട്ട സ്വഭാവ സവിശേഷതകളുണ്ട്.

i. **വേരിയബളിന്റെ പേര് (variable name):** മെമ്മറിയിലെ ഒരു സ്ഥലത്ത് സംഭരിച്ചിരിക്കുന്ന ഡാറ്റ പരാമർശിക്കുന്നതിന് വേണ്ടി പ്രതീകാത്മകമായ ഉപയോഗിക്കുന്ന പേരാണിത്.

ii. **മെമ്മറി വിലാസം (memory address):** ഒരു ബൈറ്റു ഡാറ്റ വീതം സംഭരിക്കാൻ കഴിയുന്ന സെല്ലുകളുടെ (cell) ശേഖരമാണ് കമ്പ്യൂട്ടറിന്റെ RAM. RAM ലുള്ള ഓരോ സെല്ലും (ബൈറ്റ്) ഉപയോഗിക്കുന്നതിന് അവയ്ക്ക് തനതായ വിലാസങ്ങൾ നൽകപ്പെട്ടിരിക്കുന്നു. എല്ലാ വേരിയബിളുകളും RAM ലുള്ള ഒന്നോ അതിലധികമോ മെമ്മറി സ്ഥാനങ്ങളുമായി ബന്ധപ്പെട്ടിരിക്കുന്നു. അനുവദിച്ചിട്ടുള്ള മെമ്മറിയുടെ ആരംഭത്തിലെ സെല്ലിന്റെ വിലാസത്തെ പ്രാരംഭ വിലാസം (base address) എന്നു പറയുന്നു. സാധാരണഗതിയിൽ ഈ വിലാസം അനുവദിക്കുന്നത് കംപൈലർ ആണ്. ഈ വിലാസത്തെ വേരിയബിളിന്റെ എൽ മൂല്യം (L-Value) എന്നും വിളിക്കുന്നു. ചിത്രം 6.2 ൽ Num വേരിയബിളിന്റെ പ്രാരംഭ വിലാസം 1001 ആണ്.



*ചിത്രം 6.2 : ഒരു വേരിയബിളിന്റെ മെമ്മറി പ്രതിനിധാനം*

iii. **ഉള്ളടക്കം (Content):** ഒരു മെമ്മറി സ്ഥാനത്ത് സംഭരിച്ചിരിക്കുന്ന മൂല്യത്തെ വേരിയബിളിന്റെ ഉള്ളടക്കം എന്ന് വിളിയ്ക്കുന്നു. ഇതിനെ വേരിയബിളിന്റെ ആർ. മൂല്യം (R-value) എന്നും വിളിയ്ക്കുന്നു. ഉള്ളടക്കത്തിന്റെ തരവും വലിപ്പവും വേരിയബിളിന്റെ ഡാറ്റാ ഇനത്തെ ആശ്രയിച്ചിരിക്കുന്നു.

ചിത്രം 6.2 ഒരു വേരിയബിളിന്റെ മെമ്മറിയിലെ പ്രതിനിധാനം കാണിച്ചിരിക്കുന്നു. ഇവിടെ Num എന്നതു വേരിയബിളിന്റെ പേരും 1001, 1002, 1003, 1004 എന്നീ നാലു മെമ്മറി വിലാസങ്ങൾ ഉപയോഗിക്കുന്ന 4 ബൈറ്റ് മെമ്മറിയുമാണ്. ഈ വേരിയബിളിന്റെ ഉള്ളടക്കം 18 ആണ്. അതായത് Num ന്റെ L മൂല്യം. 1001 ഉം R മൂല്യം 18 ഉം ആണ്.

## 6.6. ഓഷറേറ്ററുകൾ (operators):

കമ്പ്യൂട്ടറുകളിൽ പ്രവർത്തനങ്ങൾ (operations) നടപ്പിലാക്കുന്നതിന് പ്രേരിപ്പിക്കുന്ന മുൻകൂട്ടി നിശ്ചയിച്ചിട്ടുള്ള ചിഹ്നങ്ങളാണ് ഓപ്പറേറ്ററുകൾ. ഒരു ഓപ്പറേഷനിൽ പങ്കെടുക്കുന്നവയെ ഓപ്പറാൻഡ്സ് (operands) എന്നു വിളിക്കുന്നു. ഒരു ഓപ്പറന്റ് വേരിയയബിളോ സ്ഥിരാംഗമോ ആകാം.

ഉദാഹരണത്തിന് a+b എന്ന അരിത്മെറ്റിക് ഓപ്പറേഷനിൽ + (സങ്കലനം) ഓപ്പറേറ്ററും, a, b എന്നിവ ഓപ്പറാൻഡുകളും ആണ്. വിവിധ മാനദണ്ഡങ്ങൾക്കനുസൃതമായി C++ലെ ഓപ്പറേറ്ററുകളെ തരം തിരിച്ചിരിക്കുന്നു. C++ൽ ഓപ്പറേഷനുപയോഗിക്കുന്ന ഓപ്പറാൻഡ്കളുടെ എണ്ണം അനുസരിച്ച് ഓപ്പറേറ്ററുകളെ യൂനറി (unary), ബൈനറി (binary), ടെറിനറി (terinary) എിങ്ങനെ മൂന്നായി തരം തിരിച്ചിരിക്കുന്നു.

**യൂനറി ഓപ്പറേറ്ററുകൾ (Unary Operators ):** ഒരു ഓപ്പറാന്റ് മാത്രമുള്ള പ്രവർത്തനങ്ങളിലെ ഓപ്പറേറ്ററുകളാണ് യൂനറി ഓപ്പറേറ്ററുകൾ. ഒരു സംഖ്യ പോസിറ്റീവ് അല്ലെങ്കിൽ നെഗറ്റീവ് എന്നു കാണിക്കാനുപയോഗിക്കുന്ന +, (-) ചിഹ്നങ്ങളാണ് സാധാരണ ഉപ

യോഗിക്കുന്ന യൂനറി ഓപ്പറേറ്ററുകൾ ചിഹ്നത്തോടു കൂടിയ ഒരു നമ്പറിന് മുൻപിൽ + ഓപ്പറേറ്റർ നൽകുമ്പോൾ നിലവിലുള്ള ചിഹ്നത്തിന് മാറ്റമുണ്ടാകുന്നില്ല. എന്നാൽ - നൽകുമ്പോൾ വിലയിൽ മാറ്റമുണ്ടാകുന്നു. ചിഹ്നത്തോടു കൂടിയ ഒരു സംഖ്യയിൽ യൂനറി ഓപ്പറേറ്റർ നാം പ്രയോഗിച്ചാൽ സംഖ്യയുടെ നിലവിലുള്ള ചിഹ്നം നേരേ വിപരീതമാകുന്നു. യൂനറി ഓപ്പറേറ്ററിന്റെ ഉപയോഗം പട്ടിക 6.2ൽ കൊടുത്തിരിക്കുന്നു.

| വേരിയബിൾ x | യൂനറി + +x | യൂനറി - -x |
|---|---|---|
| 8 | 8 | -8 |
| 0 | 0 | 0 |
| -9 | -9 | 9 |

*പട്ടിക 6.2 - യൂനറി ഓപ്പറേറ്ററുകൾ*

ഇൻക്രിമന്റ് (increment) ++ (decriment) -- എന്നിവയും യൂനറി ഓപ്പറേറ്ററുകൾക്ക് ഉദാഹരണങ്ങളാണ്.

**ബൈനറി ഓപ്പറേറ്ററുകൾ (Binary Operator):** ബൈനറി ഓപ്പറേറ്ററുകൾ രണ്ട് ഓപ്പറാൻഡുകളിൽ പ്രവർത്തിക്കുന്നു. അരിത്മെറ്റിക് ഓപ്പറേറ്ററുകൾ (arithmatic), റിലേഷണൽ ഓപ്പറേറ്ററുകൾ (relational), ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ (logical) മുതലായവയാണ് സാധാരണയായി ഉപയോഗിക്കുന്ന ബൈനറി ഓപ്പറേറ്ററുകൾ.

**ടെറിനറി ഓപ്പറേറ്റർ (Terinary operator):** ടെർനറി ഓപ്പറേറ്ററുകൾ മൂന്ന് ഓപ്പറാൻഡുകളിൽ പ്രവർത്തിക്കുന്നു. കണ്ടീഷണൽ (conditional) ഓപ്പറേറ്റർ (**?:** ) ഇതിന് ഒരു ഉദാഹരണമാണ്.

മുകളിൽ പറഞ്ഞിരിക്കുന്ന ഓപ്പറേറ്ററുകളിൽ ചിലത് അടുത്ത ഭാഗങ്ങളിലും മറ്റു ചിലത് അധ്യായം ഏഴിലും ചർച്ച ചെയ്യാം.

പ്രവർത്തനരീതി അടിസ്ഥാനമാക്കി ഓപ്പറേറ്ററുകളെ അരിത്മാറ്റിക് (arithmatic), റിലേഷണൽ (relational), ലോജിക്കൽ (logical), ഇൻപുട്ട്/ ഔട്ട്പുട്ട് (input/output), അസൈൻമെന്റ് (assignment), ഷോർട്ട്-ഹാൻഡ് (short-hand), ഇൻക്രിമെന്റ് / ഡിക്രിമെന്റ് (increment/ decrement) എന്നിങ്ങനെ തരംതിരിച്ചിരിക്കുന്നു.

### 6.6.1 അരിത്മെറ്റിക് ഓപ്പറേറ്ററുകൾ (Arithmetic operators)

അടിസ്ഥാന ഗണിതപ്രക്രിയകളായ സങ്കലനം, വ്യവകലനം, ഗുണനം, ഹരണം എന്നിവയ്ക്ക് ഉപയോഗിക്കുന്ന ഓപ്പറേറ്ററുകളാണ് അരിത്മെറ്റിക് ഓപ്പറേറ്ററുകൾ. യഥാക്രമം +, ,* ,/എന്നീ ചിഹ്നങ്ങൾ ഇതിനായി ഉപയോഗിക്കുന്നു. ഹരണത്തിനു ശേഷമുള്ള ശിഷ്ടം ലഭിക്കുന്നതിനായി C++ ൽ മോഡുലസ് ഓപ്പറേറ്റർ (%) എന്നൊരു പ്രത്യേക ഓപ്പറേറ്ററും ഉപയോഗിക്കുന്നു. ഇവയെല്ലാം ബൈനറി ഓപ്പറേറ്ററുകളാണ്. + ഉം, - ഉം യൂനറി ഓപ്പറേറ്റുകളായും ഉപയോഗിക്കുന്നു എന്നത് ശ്രദ്ധിക്കുക. ഈ പ്രവർത്തനങ്ങൾക്ക് സംഖ്യാസംബന്ധിയായ ഓപ്പറന്റുകളാണ് ആവശ്യമായിട്ടുള്ളത്. ഈ പ്രവർത്തനത്തിന് ശേഷം ലഭിക്കുന്ന ഫലവും ഒരു സംഖ്യയായിരിക്കും. പട്ടിക 6.3ൽ. ബൈനറി അരിത്മെറ്റിക് പ്രവർത്തനത്തിന്റെ ചില ഉദാഹരണങ്ങൾ കാണിച്ചിരിക്കുന്നു.

| വേരിയബിൾ x | വേരിയബിൾ y | സങ്കലനം x + y | വ്യവകലനം x - y | ഗുണനം x * y | ഹരണം x / y |
|---|---|---|---|---|---|
| 10 | 5 | 15 | 5 | 50 | 2 |
| -11 | 3 | -8 | -14 | -33 | -3.66667 |
| 11 | -3 | 8 | 14 | -33 | -3.66667 |
| -50 | -10 | -60 | -40 | 500 | 5 |

*ചിത്രം 6.3 അരിത്മെറ്റിക് ഓപ്പറേറ്ററുകൾ*

**മോഡുലസ് ഓപ്പറേറ്ററുകൾ (Modulus operator (%)):** മോഡ് അഥവാ മോഡുലസ് ഓപ്പറേറ്റർ ഹരണത്തിനുശേഷമുള്ള ശിഷ്ടം കണ്ടുപിടിക്കാൻ ഉപയോഗിക്കുന്നു. ഇത് ഇന്റിജർ ഓപ്പറാൻഡ്കുകളോടൊപ്പം മാത്രമേ ഉപയോഗിക്കാൻ കഴിയു. മോഡുലസ് പ്രക്രിയയുടെ ചില ഉദാഹരണങ്ങൾ പട്ടിക 6.4 ൽ കാണിച്ചിരിക്കുന്നു. ഈ പ്രക്രിയയുടെ ഫലത്തിന്റെ ചിഹ്നം ഒന്നാമത്തെ ഓപ്പറാൻഡിന്റെ ചിഹ്നം തന്നെ ആയിരിക്കുമെന്നത് ശ്രദ്ധിക്കുക. ഇവിടെ പട്ടികയിൽ ഒന്നാമത്തെ ഓപ്പറാന്റ് x ആണ്. ഉദാഹരണം പട്ടിക 6.4ൽ.

| വേരിയബിൾ x | വേരിയബിൾ y | മോഡുലസ് ഓപ്പറേഷൻ x % y | വേരിയബിൾ x | വേരിയബിൾ y | മോഡുലസ് ഓപ്പറേഷൻ x % y |
|---|---|---|---|---|---|
| 10 | 5 | 0 | 100 | 100 | 0 |
| 5 | 10 | 5 | 32 | 11 | 10 |
| -5 | 11 | -5 | 11 | -5 | 1 |
| 5 | -11 | 5 | -11 | 5 | -1 |
| -11 | -5 | -1 | -5 | -11 | -5 |

*പട്ടിക 6.4: മോഡുലസ് ഓപ്പറേറ്റർ ഉപയോഗിച്ചുള്ള പ്രവർത്തനങ്ങൾ*

**സ്വയം പരിശോധിക്കുക**

1. അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളെ ആരോഹണ ക്രമത്തിൽ ക്രമീകരിക്കുക.
2. ഒരു സംഭരണ സ്ഥാനത്തിനു നൽകുന്ന പേര് ..... എന്ന് അറിയപ്പെടുന്നു.
3. C++ ലെ ഒരു ടെർനറി ഓപ്പറേറ്ററുടെ പേരെഴുതുക.
4. x = -5, y = 3 ആയാൽ താഴെ കൊടുത്തിരിക്കുന്ന ഓപ്പറേഷനുകളുടെ ഔട്ട്പുട്ട് പ്രവചിക്കുക.

| | |
|---|---|
| a. –x | f. x + y |
| b. –y | g. x % y |
| c. –x + -y | h. x / y |
| d. –x – y | i. x * –y |
| e. x % –11 | j. –x % -5 |

### 6.6.2 റിലേഷണൽ ഓപ്പറേറ്ററുകൾ (Relational Operators) :

സംഖ്യ സംബന്ധിയായ ഡാറ്റയെ താരതമ്യം ചെയ്യുന്നതിനാണ് റിലേഷനൽ ഓപ്പറേറ്ററുകൾ ഉപയോഗിക്കുന്നത്. ഇവ ബൈനറി ഓപ്പറേറ്ററുകളാണ്. ഏതൊരു റിലേഷണൽ ഓപ്പറേഷന്റെയും ഫലം ശരി (true) അല്ലെങ്കിൽ തെറ്റ് (false) എന്നതായിരിക്കും. C++ൽ True നെ 1 കൊണ്ടും False നെ 0 കൊണ്ടും പ്രതിനിധീകരിക്കുന്നു. <(ചെറുതാണ്), < = (ചെറുതോ, തുല്യമോ ആണ്), > (വലുതാണ്), > = (വലുതോ, തുല്യമോ ആണ്), == (തുല്യമാണ്), ! = (തുല്യമല്ല). എന്നിങ്ങനെ 6 റിലേഷണൽ ഓപ്പറേറ്ററുകളാണ് C++ൽ ഉള്ളത്. തുല്യതാ പരിശോധനയ്ക്ക് രണ്ട് തുല്യ ചിഹ്നങ്ങൾ (=) ആവശ്യമാണെന്നത് ശ്രദ്ധിക്കുക. വിവിധ റിലേഷണൽ ഓപ്പറേറ്ററുകളുടെ ഉപയോഗവും അവയുടെ ഫലങ്ങളും പട്ടിക 6.5 ൽ കാണിച്ചിരിക്കുന്നു.

| m | n | m<n | m>n | m<=n | m>=n | m!=n | m==n |
|---|---|---|---|---|---|---|---|
| 12 | 5 | 0 | 1 | 0 | 1 | 1 | 0 |
| -7 | 2 | 1 | 0 | 1 | 0 | 1 | 0 |
| 4 | 4 | 0 | 0 | 1 | 1 | 0 | 1 |

*പട്ടിക 6.5 റിലേഷണൽ ഓപ്പറേറ്ററുകൾ ഉപയോഗിച്ചുള്ള പ്രവർത്തനങ്ങൾ*

### 6.6.3 ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ (Logical Operators):

റിലേഷണൽ ഓപ്പറേറ്ററുകൾ ഉപയോഗിച്ച് നമുക്ക് വിലകൾ താരതമ്യം ചെയ്യാം. ഉദാഹരണത്തിന് 3<5,num !=10 മുതലായവ C++ൽ ഇത്തരം താരതമ്യ പ്രവർത്തനങ്ങളെ റിലേഷണൽ പദപ്രയോഗങ്ങൾ എന്ന് വിളിക്കുന്നു. ചില സാഹചര്യങ്ങളിൽ രണ്ടോ അതിലധികമോ താരതമ്യങ്ങൾ സംയോജിപ്പിക്കേണ്ടതായി വരും. ഗണിതശാസ്ത്രത്തിൽ a>b>c എന്ന രീതിയിലുള്ള പദപ്രയോഗങ്ങൾ നമുക്ക് ഉപയോഗിക്കാം. എന്നാൽ C++ൽ ഇത് സാധ്യമല്ല. ഇവയെ a>b എന്നും b>c എന്നും വേർതിരിച്ച് & എന്ന ലോജിക്കൽ ഓപ്പറേറ്റർ ഉപയോഗിച്ച് സംയോജിപ്പിക്കുന്നു. അതായത് (a>b) && (b>c). ഇത്തരം ലോജിക്കൽ സംയോഗങ്ങളുടെ ഫലവും (true) (1) അല്ലെങ്കിൽ (false) (0) ആയിരിക്കും. && (ലോജിക്കൽ ആൻഡ് (AND),! (ലോജിക്കൽ ഓർ (OR)), ! (ലോജിക്കൽ നോട്ട് (NOT)) എന്നിവയാണ് C++ ലെ ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ.

**ലോജിക്കൽ ആന്റ് (logical AND) ഓപ്പറേറ്റർ:** E1, E2 എന്നീ രണ്ട് റിലേഷൻ പദപ്രയോഗങ്ങൾ logical AND ഉപയോഗിച്ച് സംയോജിപ്പിക്കുമ്പോൾ ഫലം true(1) ലഭിക്കണമെങ്കിൽ E1, E2 എന്നിവ രണ്ടും true(1) തന്നെ ആയിരിക്കണം. അല്ലാത്ത എല്ലാ സന്ദർഭങ്ങളിലും ഫലം false(0) ആയിരിക്കും. വിവിധ ഇൻപുട്ടുകൾക്ക് അനുസരിച്ചുള്ള ലോജിക്കൽ AND പ്രക്രിയയുടെ ഫലം പട്ടിക 6.6 ൽ കാണിച്ചിരിക്കുന്നു.

| E1 | E2 | E1&&E2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

*പട്ടിക 6.6ആന്റ് ഓപ്പറേറ്ററിന്റെ ഉപയോഗം*

ഉദാഹരണം 10>5&&15<25 ഫലം true (1). 10>5 &&100<25 ഫലം false (0).

**ലോജിക്കൽ ഓർ (logical OR) ഓപ്പറേറ്റർ**: E1, E2 എന്നീ രണ്ട് റിലേഷൻ പദപ്രയോഗങ്ങൾ logical OR ഉപയോഗിച്ച് സംയോജിപ്പിക്കുമ്പോൾ ഫലം false(0) ലഭിക്കുമെങ്കിൽ E1, E2 എന്നിവ രണ്ടും false (0) ആയിരിക്കണം. അല്ലാത്ത എല്ലാ സന്ദർഭങ്ങളിലും ഫലം true(1) ആയിരിക്കും. വിവിധ ഇൻപുട്ടുകൾക്ക് അനുസരിച്ചുള്ള ലോജിക്കൽ OR പ്രക്രിയയുടെ ഫലം പട്ടിക 6.7ൽ കാണിച്ചിരിയ്ക്കുന്നു.

| E1 | E2 | E1□E2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

*പട്ടിക 6.7 ഓർ ഓപ്പറേറ്ററിന്റെ ഉപയോഗം*

ഉദാഹരണം: 10>15 | |100<25 ഫലം true(1), 10>15 | |100<90 ഫലം false (0).

**ലോജിക്കൽ നോട്ട് (logical NOT) ഓപ്പറേറ്റർ**: റിലേഷനൽ പദപ്രയോഗങ്ങളുടെ ഫലം വിപരീതമാക്കാനാണ് logical NOT ഉപയോഗിക്കുന്നത്. ഇത് ഒരു യൂനറി ഓപ്പറേഷനാണ്.

വിവിധ ഇൻപുട്ടുകൾക്ക് അനുസരിച്ചുള്ള ലോജിക്കൽ NOT പദപ്രയോഗത്തിന്റെ ഫലം പട്ടിക 6.8ൽ കാണിച്ചിരിക്കുന്നു.

| E1 | !E1 |
|---|---|
| 0 | 1 |
| 1 | 0 |

പട്ടിക 6.8നോട്ട് ഓപ്പറേറ്ററിന്റെ ഉപയോഗം

ഉദാഹരണം ! (100<2) ഫലം 1

! (100>2) ഫലം 0 (False)

### 6.6.4 ഇൻപുട്ട് / ഔട്ട്പുട്ട് ഓപ്പറേറ്ററുകൾ (Input/Output operators)

ഇൻപുട്ട് പ്രവർത്തനങ്ങൾക്ക് സാധാരണയായി ഉപയോക്താവിന്റെ ഇടപെടൽ ആവശ്യമാണ്. ഇൻപുട്ട് പ്രോസസ്സിൽ കീബോഡ് വഴി നൽകുന്ന ഡാറ്റ മെമ്മറി ലൊക്കേഷനുകളിൽ സൂക്ഷിക്കുന്നു. C++ൽ ഇൻപുട്ട് ഓപ്പറേഷൻ ചെയ്യുന്നതിനായി >> ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു. ഈ ഓപ്പറേറ്റർ ഗേറ്റ് ഫ്രം (get from) അഥവാ എക്സ്ട്രാക്ഷൻ (extraction) ഓപ്പറേറ്റർ എന്ന് അറിയപ്പെടുന്നു. രണ്ട് > ചിഹ്നങ്ങൾ ഉപയോഗിച്ചാണ് ഈ ചിഹ്നം നിർമ്മിച്ചിരിക്കുന്നത്.

ഇതുപോലെ ഔട്ട്പുട്ട് പ്രവർത്തനങ്ങളിൽ ഡാറ്റ റാമിൽ നിന്നും ഔട്ട്പുട്ട് ഉപകരണത്തിലേക്ക് മാറ്റുന്നു. സാധാരണയായി ഫലം നേരിട്ട് ലഭിക്കാൻ ഉപയോഗിക്കുന്ന ഔട്ട്പുട്ട് ഉപകരണം മോണിറ്ററാണ്. പുട്ട് ടു (Put to) അഥവാ ഇൻസേർഷൻ (insertion) ഓപ്പറേറ്റർ എന്നു വിളിക്കുന്ന <<ഓപ്പറേറ്റർ ഔട്ട്പുട്ട് പ്രവർത്തനത്തിന് ഉപയോഗിക്കുന്നു. ഇത് രണ്ട് <ചിഹ്നങ്ങൾ ഉപയോഗിച്ചാണ് നിർമ്മിച്ചിരിക്കുന്നത്.

### 6.6.5 വിലനൽകൽ ഓപ്പറേറ്റർ (Assignment operators) (=)

സാധാരണയായി ഒരു വില മെമ്മറിയിൽ സംഭരിക്കുന്നതിനായി വിലനൽകൽ ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു. ഇത് ഒരു ബൈനറി ഓപ്പറേറ്ററായതിനാൽ ഇവയ്ക്ക് രണ്ട് ഓപ്പറാന്റുകൾ ആവശ്യമാണ്. ആദ്യത്തെ ഓപ്പറാന്റ് ഒരു വേരിയബിൾ ആയിരിക്കണം. അതിലാണ് രണ്ടാമത്തെ ഓപ്പറാന്റിന്റെ മൂല്യം സൂക്ഷിക്കുന്നത്.

ചില ഉദാഹരണങ്ങൾ പട്ടിക 6.9ൽ കാണിച്ചിരിക്കുന്നു.

| ഇനം | വിശദീകരണം |
|---|---|
| a=b | വേരിയബിൾ **b** യുടെ വില **a** ൽ സംഭരിക്കുന്നു |
| a=3 | സ്ഥിരാംഗം **3** വേരിയബിൾ **a** ൽ സംഭരിക്കുന്നു |

പട്ടിക 6.9 അസൈൻമെന്റ് ഓപ്പറേറ്റർ

റിലേഷണൽ ഓപ്പറേറ്ററായ = = ഉപയോഗത്തെപ്പറ്റി ഭാഗം 6.6.2ൽ നമ്മൾ ചർച്ച ചെയ്തിരുന്നു. ഈ രണ്ടു ഓപ്പറേറ്ററുകൾ തമ്മിലുള്ള വ്യത്യാസം ശ്രദ്ധിക്കുക. = ചിഹ്നം ഒരു വേരിയബിളിനു വില നൽകുന്നതിനും എന്നാൽ = = ചിഹ്നം രണ്ട് വിലകളെ തമ്മിൽ താരതമ്യം ചെയ്ത് true അല്ലെങ്കിൽ false എന്ന ഉത്തരം നൽകുന്നതിനും ഉപയോഗിക്കുന്നു.

### 6.6.6 അരിത്മെറ്റിക് വിലനൽകൽ ഓപ്പറേറ്ററുകൾ (Arithmetic assignment operators)

ലളിതമായ ഒരു അരിത്മാറ്റിക് പ്രസ്താവന ചുരുക്കി സൂചിപ്പിക്കാൻ അരിത്തമാറ്റിക് വിലനൽകൽ ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു. ഉദാഹരണത്തിന് a = a +10 എന്നത് a + = 10 എന്നും എഴുതാം. ഇവിടെ + = എന്നത് അരിത്മാറ്റിക് വിലനലൽകൽ ഓപ്പറേറ്റർ ആണ്. ഈ രീതി എല്ലാ അരിത്മാറ്റിക് ഓപ്പറേറ്ററുകളിലും ഉപയോഗിക്കാവുന്നതാണ്. അവ പട്ടിക 6.10 കാണിച്ചിരിക്കുന്നു. +=, =, *=, /=, %= എന്നിവ. C++ലെ അരിത്മാറ്റിക് വിലനൽകൽ

ഓപ്പറേറ്ററുകളാണ്. C++ ചുരുക്കെഴുത്തുകൾ (short hands) എന്നുകൂടി ഇവ അറിയപ്പെടുന്നു. ഇവയെല്ലാം തന്നെ ബൈനറി ഓപ്പറേറ്ററുകളാണ്. ഇവയുടെ ഒന്നാമത്തെ ഓപ്പറാന്റ് എപ്പോഴും ഒരു വേരിയബിൾ തന്നെയായിരിക്കണം. സങ്കലനം, വിലനൽകൽ എന്നീ പ്രവർത്തനങ്ങൾ സാധാരണ രീതിയേക്കാൾ വേഗത്തിൽ ചെയ്യുന്നതിനായി ഈ ഓപ്പറേറ്ററുകൾ ഉപയോഗിക്കുന്നു.

| അരിത്മെറ്റിക് വിലനൽകൽ പ്രവർത്തനം | തത്തുല്യ അരിത്മെറ്റിക് പ്രവർത്തനം |
|---|---|
| x += 10 | x = x + 10 |
| x -= 10 | x = x - 10 |
| x *= 10 | x = x * 10 |
| x /= 10 | x = x / 10 |
| x %= 10 | x = x % 10 |

പട്ടിക 6.10 C++ ചുരുക്കെഴുത്തുകൾ

### 6.6.7 ഇൻക്രിമന്റ് (Increment) (++) ഡിക്രിമന്റ് (Decrement) (--) ഓപ്പറേറ്ററുകൾ (opertors)

C++ലെ രണ്ടു പ്രത്യേക ഓപ്പറേറ്ററുകളാണ് ഇൻക്രിമെന്റ്, ഡിക്രിമെന്റ് ഓപ്പറേറ്ററുകൾ. ഇവ യൂനറി ഓപ്പറേറ്ററുകളാണ്. അവയുടെ ഓപ്പറാന്റ് വേരിയബിൾ ആയിരിക്കണം.സോഴ്സ് കോഡ് സംക്ഷിപ്തമാക്കാൻ ഈ ഓപ്പറേറ്ററുകൾ സഹായിക്കും.

**ഇൻക്രിമന്റ് ഓപ്പറേറ്റർ:** (++) ഒരു ഇന്റിജർ വേരിയബിളിന്റെ ഉള്ളടക്കത്തെ ഒന്നു വർദ്ധിപ്പിക്കാൻ ഈ ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു. ++x പ്രീഇൻക്രിമെന്റ് (pre increment), x++ പോസ്റ്റ്ഇൻക്രിമെന്റ് (post increment) എന്നിങ്ങനെ ഇതിനെ രണ്ടു രീതിയിൽ എഴുതാം. ഇത് x=x+1 അല്ലെങ്കിൽ x+=1 എന്നതിനു തുല്യമാണ്.

**ഡിക്രിമെന്റ് ഓപ്പറേറ്റർ:** ഇൻക്രിമെന്റ് ഓപ്പറേറ്ററിന്റെ നേർവിപരീതമായ ഡിക്രിമെന്റ് ഓപ്പറേറ്റർ ഇന്റിജർ വേരിയബിളിന്റെ ഉള്ളടക്കത്തെ കുറക്കുന്നു. ---x, x- എന്നിങ്ങനെ ഈ ഇൻക്രിമെന്റ്/ഡിക്രിമെന്റ് പ്രവർത്തനങ്ങളുടെ ഓപ്പറേറ്ററും രണ്ടു രീതിയിൽ ഉപയോഗിക്കാം. ഇത് x=x-1 അല്ലെങ്കിൽ x-=1 എന്നതിന് തൂല്യമാണ്.

ഈ ഓപ്പറേറ്ററുകളുടെ രണ്ട് രീതിയിലുള്ള ഉപയോഗങ്ങളെ ഇൻക്രിമെന്റ്/ഡിക്രിമെന്റ് പ്രവർത്തനങ്ങളുടെ പ്രിഫിക്സ് രൂപമെന്നും, പോസ്റ്റ്ഫിക്സ് രൂപമെന്നും എന്ന് വിളിക്കുന്നു.

രണ്ടു രീതികളും ഓപ്പറാന്റിൽ ഒരേ മാറ്റമാണ് വരുത്തുന്നത് എങ്കിലും മറ്റ് ഓപ്പറേറ്ററുകളുടെ കൂടെ ഉപയോഗിക്കുമ്പോൾ ഇവയുടെ പ്രവർത്തന രീതി വ്യത്യസ്തമായിരിക്കും.

**ഇൻക്രിമന്റ്/ ഡിക്രിമന്റ് ഓപ്പറേറ്ററുകളുടെ പ്രീഫിക്സ് രൂപം:** പ്രീഫിക്സ് രീതിയിൽ ഓപ്പറേറ്റർ ഓപ്പറാന്റിന്റെ മുൻപായിരിക്കും എഴുതുക. ഇവിടെ ഇൻക്രിമെന്റ് / ഡിക്രിമെന്റ് ആദ്യം ചെയ്യുകയും അങ്ങനെ കിട്ടുന്ന മൂല്യം മറ്റ് ഓപ്പറേഷനുകൾക്ക് ഉപയോഗിക്കുകയും ചെയ്യും. അതുകൊണ്ട് ഈ രീതിയെ മാറ്റുക പിന്നീട് ഉപയോഗിക്കുക (change, then use) എന്ന് വിളിക്കുന്നു.

a, b, c, d എന്നീ വേരിയബിളുകൾ പരിഗണിക്കുക. അവയിൽ a, യുടെ വില 10 ഉം b യുടെ വില 5 ഉം ആണ്. C=++a എന്ന പ്രസ്താവനയിൽ a യുടെ മൂല്യം 11 ഉം c യുടെ മൂല്യം 11 ഉം ആയി ലഭിക്കും. ഇവിടെ ആദ്യം a യുടെ മൂല്യം ഒന്ന് വർദ്ധിച്ച് 11 ആകും. ഈ കൂടിയ മൂല്യമാണ് cക്ക് നൽകുന്നത്. അതുകൊണ്ടാണ് രണ്ടിനും ഒരേ വില ലഭിക്കുന്നത്. അതുപോലെ തന്നെ d= --b എന്നതിൽ d യുടെയും b യുടെയും മൂല്യം 4 ആകും.

**ഇൻക്രിമന്റ്/ഡിക്രിമെന്റ് ഓപ്പറേറ്ററുകളുടെ പോസ്റ്റ് ഫിക്സ് രൂപം:** പോസ്റ്റ്ഫിക്സ് രീതിയിൽ ഇൻക്രിമന്റ് /ഡിക്രിമന്റ് ഓപ്പറേഷൻ നടത്തുമ്പോൾ ഓപ്പറേറ്റർ ഓപ്പറാന്റിനു

ശേഷമാണ് എഴുതുക. വേരിയബിളിന്റെ നിലവിലുള്ള വിലയാണ് മറ്റ് ഓപ്പറേഷനുകൾക്ക് ഉപയോഗിക്കുക. അതിനുശേഷം മൂല്യം വർദ്ധിപ്പിക്കുകയോ കുറയ്ക്കുകയോ ചെയ്യും. അതിനാൽ ഈ രീതിയെ ഉപയോഗിക്കുക, പിന്നീട് മാറ്റുക (use, then change) എന്നു വിളിക്കുന്നു.

മുകളിൽ കൊടുത്ത ഉദാഹരണം അതേ തുടക്ക വിലകൾ ഉപയോഗിച്ച് നിരീക്ഷിക്കുമ്പോൾ c = a++ എന്ന പ്രയോഗത്തിൽ aയുടെ മൂല്യം 11, c യുടെ മൂല്യം 10 എന്ന് ലഭിക്കും. ഇവിടെ a യുടെ മൂല്യം c ക്ക് നൽകുകയും അതിന് ശേഷം a യുടെ മൂല്യം ഒന്ന് വർദ്ധിപ്പിക്കുകയും ചെയ്യുന്നു. അതായത് a യുടെ മൂല്യം വർദ്ധിപ്പിക്കുന്നതിന് മുൻപുതന്നെ c ക്ക് ആ വില നൽകുന്നു. അതുപോലെ d = b- - എന്ന പ്രവർത്തനത്തിനുശേഷം d യുടെ മൂല്യം 5 ഉം b യുടെ മൂല്യം 4 ഉം ആയിരിക്കും.

### 6.6.8 കണ്ടീഷണൽ ഓപ്പറേറ്റർ (conditional operator) (?:)

മൂന്ന് ഓപ്പറാന്റുകൾക്കുമേൽ ഉപയോഗിക്കുന്ന ഒരു ടെറിനറി ഓപ്പറേറ്ററാണ്. ഇതിൽ ആദ്യത്തെ ഓപ്പറാന്റ് ഒരു ലോജിക്കൽ പ്രയോഗവും (വ്യവസ്ഥ) മറ്റു രണ്ടെണ്ണം വിലകളും ആയിരിക്കും. അവ സ്ഥിരാംഗമോ വേരിയബിളോ പ്രയോഗമോ ആവാം. ആദ്യം വ്യവസ്ഥ പരിശോധിച്ച് ശരിയാണെങ്കിൽ രണ്ടാമത്തെ ഓപ്പറാന്റ് തിരഞ്ഞെടുക്കുന്നു. അല്ലെങ്കിൽ മൂന്നാമത്തെ ഓപ്പറാന്റ് തിരഞ്ഞെടുക്കുന്നു.

വാക്യഘടന

പ്രയോഗം 1 ? പ്രയോഗം 2 :പ്രയോഗം 3;

താഴെ കൊടുത്തിരിക്കുന്ന പ്രവർത്തനം നമുക്ക് നോക്കാം.

result= score >50? 'p':'f';

എന്ന പ്രയോഗത്തിൽ scroreന്റെ മൂല്യം 50 ൽ കൂടുതൽ ആണെങ്കിൽ 'p' എന്ന വിലയും അല്ലെങ്കിൽ 'f' എന്ന വിലയുമാണ് result എന്ന വേരിയബിളിൽ സംഭരിക്കുന്നത്. ഈ ഓപ്പറേറ്ററുകളെ കുറിച്ചുള്ള കൂടുതൽ കാര്യങ്ങൾ ഏഴാം അദ്ധ്യായത്തിൽ നമ്മൾ ചർച്ച ചെയ്യും.

### 6.6.9 സൈസ് ഓഫ് ഓപ്പറേറ്റർ (size of operator)

സൈസ് ഓഫ് ഓപ്പറേറ്റർ യൂനറി കംപൈൽ ടൈം ഓപ്പറേറ്ററാണ്. ഒരു ഓപ്പറാന്റ് എത്ര മെമ്മറി ഉപയോഗിച്ചു എന്ന് കണക്കാക്കുന്നതിനാണ് ഇത് ഉപയോഗിക്കുന്നത്. ഇതിലുപയോഗിക്കുന്ന ഓപ്പറാന്റ് സ്ഥിരാംഗമോ, വേരിയബിളോ, ഡാറ്റയുടെ ഇനമോ ആവാം.

ഇതിന്റെ വാക്യഘടന താഴെ കൊടുത്തിരിക്കുന്നു

size of (data type);

size of (variable name);

size of (constant)

ഓപ്പറാന്റായി ഡാറ്റ ഇനം നൽകുമ്പോൾ അത് ആവരണ ചിഹ്നത്തിനുള്ളിൽ ആയിരിക്കണമെന്നത് ശ്രദ്ധിക്കുക. മറ്റുള്ള ഓപ്പറാന്റുകൾക്ക് ആവരണം നിർബന്ധമില്ല. പല രൂപത്തിലുള്ള സൈസ് ഓഫ് ഓപ്പറേറ്ററിന്റെ ഉപയോഗം പട്ടിക 6.11 ൽ കാണിച്ചിരിക്കുന്നു.

| ഇനം | വിശദീകരണം |
|---|---|
| `sizeof(int)` | 4 എന്ന വില ലഭിക്കുന്നു (GCC ൽ int ഡാറ്റ ഇനത്തിന്റെ വലിപ്പം 4 ബൈറ്റ് ആണ്). |
| `sizeof 3.2` | ഫലം 8 ലഭിക്കുന്നു (ഫ്ളോട്ടിംഗ് പോയിന്റ് സ്ഥിരാംഗം എന്നത് double ഡാറ്റാ ഇനമായി കണക്കാക്കുന്നു. |
| `sizeof p;` | p എന്നത് float തരത്തിലുള്ള വേരിയബിൾ ആണെങ്കിൽ 4 എന്ന വില നൽകുന്നു. |

പട്ടിക 6.11, സൈസ് ഓഫ് ഓപ്പറേറ്ററിന്റെ വിവിധ പ്രയോഗങ്ങൾ

### 6.6.10 ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം (Precedence of operators)

പലതരം ഓപ്പറേറ്ററുകൾ ഒരു പ്രയോഗത്തിൽ ഉപയോഗിക്കുമ്പോൾ ഏത് ക്രമത്തിലാണ് ആ ക്രിയകൾ ചെയ്യേണ്ടത് എന്ന് അറിയേണ്ടതുണ്ട്. C++ ൽ അവയുടെ മുൻഗണനാക്രമം എങ്ങനെയാണെന്ന് നോക്കാം. വിവിധ ഓപ്പറേറ്ററുകൾ ഉപയോഗിക്കുന്ന ഒരു പ്രയോഗത്തിൽ ആവരണ ചിഹ്നത്തിനാണ് ആദ്യ പരിഗണന. ( ) ആവരണ ചിഹ്നം ഇല്ലെങ്കിൽ മുൻനിശ്ചയിച്ചപ്രകാരമുള്ള ഒരു മുൻഗണനാക്രമത്തിലാണ് അവ വിലയിരുത്തപ്പെടുക. ഓപ്പറേറ്ററുകളുടെ ഈ മുൻഗണനാക്രമം പട്ടിക 6.12 ൽ നൽകിയിരിക്കുന്നു. ഒരു പ്രയോഗത്തിൽ മുൻഗണനാക്രമത്തിൽ ഒരേ സ്ഥാനം വരുന്ന ഓപ്പറേറ്ററുകൾ ഉണ്ടെങ്കിൽ അവ മിക്കവാറും ഇടത്തുനിന്ന് വലത്തേക്ക് എന്ന രീതിയിലാണ് പ്രവർത്തിക്കുക.

| മുൻഗണന | പ്രവർത്തനങ്ങൾ |
|---|---|
| 1 | ( ) ആവരണം |
| 2 | ++, --, ! , യൂനറി + , യൂനറി –, sizeof |
| 3 | * (ഹരണം), / (ഗുണനം), % (മോഡുലസ്) |
| 4 | + (സങ്കലനം), – (വ്യവകലനം) |
| 5 | < (ചെറുതാണ്), <= (ചെറുതോ തുല്യമോ ആണ്), > (വലുതാണ്), >= (വലുതോ തുല്യമോ ആണ്) |
| 6 | == (തുല്യമാണ്), != (തുല്യമല്ല) |
| 7 | && (ലോജിക്കൽ AND) |
| 8 | \|\| (ലോജിക്കൽ OR) |
| 9 | ? : (കണ്ടീഷണൽ പ്രയോഗം) |
| 10 | = (അസൈൻമെന്റ് ഓപ്പറേറ്റർ), *=, /=, %=, +=, –= (അരിത്മെറ്റിക് അസൈൻമെന്റ് ഓപ്പറേറ്ററുകൾ) |
| 11 | , (അല്പവിരാമം) |

*പട്ടിക 6.12: ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം.*

a=3, b=5, c=4, d=2, x എന്നീ വേരിയബിളുകളും അവയുടെ വിലകളും പരിഗണിക്കുക.

x=a+b*c-d എന്ന പ്രയോഗം ചെയ്തു കഴിയുമ്പോൾ x ന്റെ വില 21 ആയിരിക്കും. ഇവിടെ * (ഗുണനത്തി)ന് +(സങ്കലനം), -(വ്യവകലനം) എന്നിവയേക്കാൾ മുൻഗണനയുള്ളതിനാൽ b, c എന്നീ വേരിയബിളുകൾ തമ്മിൽ ഗുണിച്ചശേഷമേ അതിന്റെ ഫലം a യോടൊപ്പം കൂട്ടിചേർക്കൂ. ആ കിട്ടുന്ന ഉത്തരത്തിൽ നിന്നും d കുറച്ചാൽ അന്തിമഫലം ലഭ്യമാകും. അങ്ങനെ കിട്ടുന്ന ഉത്തരം xന് നൽകുന്നു. പ്രോഗ്രാമറുടെ ആവശ്യാനുസരണം പ്രയോ

ഗത്തിലെ ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം മാറ്റുന്നതിനായി ( ) (ആവരണചിഹ്നം) ഉപയോഗിച്ചാൽ മതിയാകും. ഉദാഹരണത്തിന് a=5, b=4, c=3, d=2 എന്നായാൽ a+b-c*d എന്നതിന്റെ ഉത്തരം 3 ആയിരിക്കും. പ്രോഗ്രാമർക്ക് ആദ്യം വ്യവകലനം, പിന്നീട് സങ്കലനം, ഗുണനം എന്ന ക്രമത്തിൽ ചെയ്യണമെങ്കിൽ അതിനായി ശരിയായ രീതിയിൽ ആവരണ ചിഹ്നങ്ങൾ ഉപയോഗിച്ച് (a+(b-c))*d എന്ന് എഴുതണം. ഇതിന്റെ ഉത്തരം 12 ആയിരിക്കും. മുൻഗണനാക്രമം മാറ്റുതിനായി [], {} ഇത്തരം ആവരണചിഹ്നങ്ങൾ ഉപയോഗിക്കാൻ സാധിക്കില്ല.

**അറിവ്പെട്ടി**

ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം വിവിധ കപലറുകളിൽ വ്യത്യസ്തമായിരിക്കും. Tubo, C++ പോസ്റ്റ്ഫിക്സ് രൂപത്തെക്കാൾ ഉയർന്ന് മുൻഗണ പ്രീഫിക്സ് ഇൻക്രിമെന്റ് /ഡിക്രമെന്റിനെ നൽകുന്നു.

ഉദാഹരണമായി a യുടെ പ്രാരംഭ വില 5 ആണെങ്കിൽ b = a++ + ++ a എന്ന പ്രയോഗത്തിൽ b യുടെയും a യുടെയും വിലകൾ യഥാക്രമം 12,7 എന്നായിരിക്കും. ഇവ a = a+1 (പ്രഫിക്സ് പൂർണ്ണരൂപം), b=a+a, a=a+1 (പോസ്റ്റ് ഫിക്സ് പൂർണ്ണ രൂപം) എന്നിവക്ക് തുല്യമായിരിക്കും.

## 6.7. പദപ്രയോഗങ്ങൾ (expressions)

ഒരു പദപ്രയോഗം ഓപ്പറേറ്ററുകളും ഓപ്പറാന്റുകളും ചേർന്നതാണ്. ഓപ്പറാന്റുകൾ സ്ഥിരാംഗങ്ങളോ വേരിയബിളുകളോ ആകാം. എല്ലാ പദപ്രയോഗങ്ങളും പൂർത്തീകരിച്ചതിനുശേഷമേ ആ പ്രയോഗത്തിന്റെ അന്തിമ ഫലം ലഭ്യമാകൂ. ഈ ഫലം പദപ്രയോഗം തിരികെ നൽകിയ വില എന്ന് അറിയപ്പെടുന്നു. ഉപയോഗിച്ചിരിക്കുന്ന ഓപ്പറേറ്ററുകളുടെ അടിസ്ഥാനത്തിൽ പദപ്രയോഗങ്ങളെ പ്രധാനമായും അരിത്മാറ്റിക് പദപ്രയോഗങ്ങൾ, റിലേഷണൽ പദപ്രയോഗങ്ങൾ, ലോജിക്കൽ പദപ്രയോഗങ്ങൾ എന്നിങ്ങനെ തരംതിരിച്ചിരിക്കുന്നു.

### 6.7.1 അരിത്മാറ്റിക് പദപ്രയോഗങ്ങൾ (Arithmetic expressions)

അരിത്മാറ്റിക് ഓപ്പറേറ്ററുകൾ മാത്രം ഉപയോഗിച്ചിട്ടുള്ള പദപ്രയോഗങ്ങളെ അരിത്മാറ്റിക് പദപ്രയോഗങ്ങൾ എന്ന് വിളിക്കുന്നു. ഇവിടെ ഓപ്പറാന്റുകൾ സംഖ്യകളാണ്. അവ വേരിയബിളുകളോ സ്ഥിരാംഗങ്ങളോ ആകാം. ഈ പദപ്രയോഗത്തിൽ നിന്നും ലഭ്യമാകുന്ന വിലയും ഒരു സംഖ്യ ആയിരിക്കും. അരിത്മാറ്റിക് പദപ്രയോഗങ്ങളെ വീണ്ടും പൂർണ്ണ സംഖ്യാപദപ്രയോഗങ്ങൾ, ദശാംശസംഖ്യാ (real) പദപ്രയോഗങ്ങൾ, സ്ഥിരാംഗ പദപ്രയോഗങ്ങൾ എന്നിങ്ങനെ തരം തിരിച്ചിരിക്കുന്നു.

**പൂർണ്ണസംഖ്യാ പദപ്രയോഗങ്ങൾ:** ഒരു അരിത്മെറ്റിക് പദപ്രയോഗത്തിൽ പൂർണ്ണസംഖ്യകൾ മാത്രമേ ഉൾകൊള്ളുന്നുള്ളൂ എങ്കിൽ അതിനെ പൂർണ്ണസംഖ്യാപദപ്രയോഗം എന്ന് വിളിക്കുന്നു. ഇവയുടെ ഫലവും ഒരു പൂർണ്ണസംഖ്യ ആയിരിക്കും.

ഉദാഹരണത്തിന്: x, y എന്നിവ പൂർണ്ണസംഖ്യാ വേരിയബിളുകൾ ആണെങ്കിൽ ചില പൂർണ്ണ സംഖ്യാ പദപ്രയോഗവും അവയുടെ ഫലങ്ങളും പട്ടിക 6.13 ൽ കൊടുത്തിരിക്കുന്നു. എല്ലാ പദപ്രയോഗങ്ങളുടെയും ഫലം ഒരു പൂർണ്ണസംഖ്യ ആയിരിക്കും എന്നത് ശ്രദ്ധിക്കുക.

| x | y | x + y | x / y | -x + x * y | 5 + x / y | x % y |
|---|---|---|---|---|---|---|
| 5 | 2 | 7 | 2 | 5 | 7 | 1 |
| 6 | 3 | 9 | 2 | 12 | 7 | 0 |

*പട്ടിക 6.13 പൂർണ്ണ സംഖ്യാ പ്രയോഗങ്ങളും അവയുടെ ഫലങ്ങളും*

**ദശാംശസംഖ്യാ പദപ്രയോഗങ്ങൾ (floating point/ real expression):** ഒരു അരിത്മാറ്റിക് പദപ്രയോഗത്തിൽ എല്ലാ വിലകളും ദശാംശസഖ്യകൾ ആണെങ്കിൽ അവയെ ദശാംശസഖ്യ അഥവാ ഭിന്ന സംഖ്യാപദപ്രയോഗം എന്ന് വിളിക്കുന്നു. ഇതിന്റെ ഫലം തീർച്ചയായും ഒരു ദശാംശസംഖ്യ ആയിരിക്കും. x, y എന്നിവ ദശാംശസംഖ്യാ വേരിയബിൾ ആണെന്ന് കരുതുക. ചില ദശാംശസംഖ്യാപദപ്രയോഗങ്ങളും അവയുടെ ഫലങ്ങളും പട്ടിക 6.14ൽ കൊടുത്തിരിക്കുന്നു.

| x | y | x + y | x / y | -x + x * y | 5 + x / y | x * x / y |
|---|---|---|---|---|---|---|
| 5.0 | 2.0 | 7.0 | 2.5 | 5.0 | 7.5 | 12.5 |
| 6.0 | 3.0 | 9.0 | 2.0 | 12.0 | 7.0 | 12.0 |

*പട്ടിക 6.14: ദശാംശസംഖ്യാപദപ്രയോഗങ്ങളും അവയുടെ ഫലങ്ങളും*

മുകളിൽ കൊടുത്തിരിക്കുന്ന എല്ലാ പദപ്രയോഗങ്ങളുടെയും ഉത്തരം ദശാംശസംഖ്യകളാണ് എന്ന് കാണാൻ കഴിയും.

ഒരു അരിത്മാറ്റിക് പദപ്രയോഗത്തിൽ ഉപയോഗിക്കുന്ന എല്ലാ ഓപ്പറാന്റുകളും സ്ഥിരാംഗങ്ങളാണെങ്കിൽ അതിനെ സ്ഥിരാംഗപദപ്രയോഗം (const. expression) എന്നു വിളിക്കുന്നു. ഉദാ: 20+5/2.0. സ്ഥിരാംഗങ്ങളായ 15,3.14, 'a' എന്നിവയും സ്ഥിരാംഗപദപ്രയോഗങ്ങളായി അറിയപ്പെടുന്നു.

### 6.7.2 റിലേഷണൽ പദപ്രയോഗങ്ങൾ (relational expressions)

റിലേഷണൽ ഓപ്പറേറ്ററുകൾ ഉപയോഗിക്കുന്ന പദപ്രയോഗങ്ങളെ റിലേഷണൽ പദപ്രയോഗങ്ങൾ എന്ന് വിളിക്കുന്നു. ഇവ true(1) അല്ലെങ്കിൽ false(0) എന്ന ഫലം നൽകുന്നു. ഇത്തരം പദപ്രയോഗങ്ങളിൽ ഓപ്പറാന്റുകളായി സംഖ്യകളാണ് ഉപയോഗിക്കുക. ഇവയുടെ ചില ഉദാഹരണം പട്ടിക 6.15 ൽ കാണിച്ചിരിക്കുന്നു.

| x | y | x > y | x == y | x+y !=y | x-2 == y+1 | x*y == 6*y |
|---|---|---|---|---|---|---|
| 5.0 | 2.0 | 1 (True) | 0 (False) | 1 (True) | 1 (True) | 0 (False) |
| 6 | 13 | 0 (False) | 0 (False) | 1 (True) | 0 (False) | 1 (True) |

*പട്ടിക 6.15 റിലേഷണൽ പദപ്രയോഗങ്ങളും അവയുടെ ഫലങ്ങളും*

അരിത്മാറ്റിക് ഓപ്പറേറ്ററുകൾക്ക് റിലേഷണൽ ഓപ്പറേറ്ററേക്കാൾ മുൻഗണയുണ്ടെന്ന്നമുക്കറിയാം. ഒരു റിലേഷണൽ ഓപ്പറേറ്ററിന്റെ ഇരുവശങ്ങളിലായി അരിത്മാറ്റിക് പദപ്രയോഗങ്ങൾ ഉപയോഗിക്കുമ്പോൾ ആദ്യം അരിത്മാറ്റിക് ഓപ്പറേഷനുകൾ ചെയ്യുകയും അതിന് ശേഷം ആ ഫലങ്ങൾ താരതമ്യം ചെയ്യുന്നു. പട്ടികയിലെ ചില പദപ്രയോഗങ്ങളിൽ അരിത്മാറ്റിക് ഓപ്പറേറ്ററും റിലേഷണൽ ഓപ്പറേറ്ററുകളും ഉൾപ്പെട്ടിരിക്കുന്നു. വിവിധതരം ഓപ്പറേറ്ററുകൾ ഉൾപ്പെട്ടിട്ടുണ്ടെങ്കിലും ഇവയുടെ ഫലം true(1) അല്ലെങ്കിൽ false (0) ആയതിനാൽ അവയെ റിലേഷണൽ പദപ്രയോഗങ്ങൾ എന്നാണ് വിളിക്കുന്നത്.

### 6.7.3 ലോജിക്കൽ പ്രയോഗങ്ങൾ (logical expressions)

രണ്ടോ അതിലധികമോ റിലേഷണൽ പദപ്രയോഗങ്ങളെ ലോജിക്കൽ ഓപ്പറേറ്റർ ഉപയോഗിച്ച് ലോജിക്കൽ പദപ്രയോഗങ്ങൾ സംയോജിപ്പിക്കുന്നു. ഇവയുടെ ഫലം true(1) അല്ലെങ്കിൽ false (0) എന്നായിരിക്കും. ലോജിക്കൽ പദപ്രയോഗത്തിൽ വേരിയബിളുകൾ, സ്ഥിരാംഗങ്ങൾ ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ, റിലേഷനൽ ഓപ്പറേറ്ററുകൾ എന്നിവ ഉൾപ്പെടാവുന്നതാണ്. ചില ഉദാഹരണങ്ങൾ പട്ടിക 6.16 ൽ കാണിച്ചിരിക്കുന്നു.

| x | y | x>=y && x==20 | x==5\|\|y==0 | x==y && y+2==0 | !(x==y) |
|---|---|---|---|---|---|
| 5.0 | 2.0 | 0 (False) | 1 (True) | 0 (False) | 1 (True) |
| 20 | 13 | 1 (True) | 0 (False) | 0 (False) | 1 (True) |

*പട്ടിക 6.16 ലോജിക്കൽ പദ പ്രയോഗങ്ങളും അവയുടെ ഫലങ്ങളും*

പട്ടിക 6.16 ൽ കാണുന്നതു പോലെ ചില പദപ്രയോഗങ്ങളിൽ ലോജിക്കൽ ഓപ്പറേറ്ററുകളെ കൂടാതെ അരിതമാറ്റിക് ഓപ്പറേറ്ററുകളും റിലേഷൺ ഓപ്പറേറ്ററുകളും ഉൾപ്പെട്ടിട്ടുണ്ടെങ്കിലും ഈ പ്രയോഗങ്ങളെ ലോജിക്കൽ പ്രയോഗങ്ങളായി കണക്കാക്കുന്നു. അവസാനം ചെയ്യുന്ന പ്രവർത്തനം ലോജിക്കൽ പ്രവർത്തനം ആയതിനാലും അതിന്റെ ഫലം True അല്ലെങ്കിൽ False ആയത് കൊണ്ടുമാണ് ഇത്.

**സ്വയം പരിശോധിക്കാം.**

1. x = 5, y=3 ആയാൽ താഴെ കൊടുത്തിരിക്കുന്ന പ്രവർത്തനങ്ങളുടെ ഔട്ട്പുട്ട് പ്രവചിക്കുക
   a) x> = 10<<y>=4, b) x>=1<<y>=3, c) x>=11y>=4, d)x>=1 11y>=3
2. p=5, q=3, x=2 ആണെങ്കിൽ ഔട്ട്പുട്ട് പ്രവചിക്കുക
   a) ++P-q×r/2 b) p×q--+r c) p-q-r×2+p d) p+=5×q+r×r/2

## 6.8 ഇനം മാറ്റൽ (type conversion)

പൂർണ്ണസംഖ്യ പദപ്രയോഗം, ദശാംശസംഖ്യാ പദപ്രയോഗം എന്നിങ്ങനെ രണ്ട്തരം അരിത്മെറ്റിക് പദപ്രയോഗങ്ങൾ ഉണ്ടെന്ന് നാം മുമ്പ് ചർച്ച ചെയ്തുവല്ലോ. ഇവ രണ്ടിലും അരിത്മാറ്റിക് ഓപ്പറേഷനിൽ ഉൾപ്പെട്ടിരിക്കുന്ന ഓപ്പറാന്റുകൾ ഒരേ ഡാറ്റാ ഇനത്തിലുള്ളവയാണ്. എന്നാൽ വ്യത്യസ്ത ഇനം സംഖ്യകൾ ഉപയോഗിക്കേണ്ട സാഹചര്യങ്ങളും ഉണ്ടാകാം. ഉദാഹരണമായി C++ ൽ പൂർണ്ണസംഖ്യാ പദപ്രയോഗം 5/2, എന്നത് 2 എന്ന ഫലം തരുമ്പോൾ ദശാംശസംഖ്യാ പദപ്രയോഗമായ 5.0/2.0 എന്നത് 2.5 എന്ന ഫലം തരുന്നു. പക്ഷെ 5/2.0, അല്ലെങ്കിൽ 5.0/2 എന്നിവയുടെ ഉത്തരം എന്തായിരിക്കും? ഇനം മാറ്റൽ രീതിയാണ് ഈ സാഹചര്യത്തിൽ ഉപയോഗിക്കേണ്ടി വരുക. ഒരു ഓപ്പറാന്റിന്റെ ഡാറ്റാ ഇനം മറ്റൊന്നിലേക്ക് മാറ്റുകയാണ് ചെയ്യേണ്ടത്. ഇതിനെ ഇനം മാറ്റൽ എന്ന് പറയാം. ഇത് ആന്തരിക ഇനം മാറ്റൽ, ബാഹ്യഇനം മാറ്റൽ എന്നിങ്ങനെ രണ്ടു രീതിയിൽ ചെയ്യാം.

### 6.8.1 ആന്തരിക ഇനം മാറ്റൽ (implicit type conversion/ type promotion):

ആന്തരിക ഇനം മാറ്റൽ C++ കമ്പൈലർ ആന്തരികമായി ചെയ്യുന്നതാണ്. വ്യത്യസ്ഥതരം ഡാറ്റ ഉള്ള ഒരു പദപ്രയോഗത്തിൽ C++ കുറഞ്ഞ വലിപ്പത്തിലുള്ള ഓപ്പറാന്റിനെ കൂടു

തൽ വലുപ്പമുള്ളതിന്റെ ഡാറ്റാ ഇനമാക്കി മാറ്റുന്നു. അതായത് എല്ലായ്പ്പോഴും ചെറിയ തിനെ വലുതാക്കുക മാത്രമാണ് ചെയ്യുന്നത്. ആയതിനാൽ ഇതിനെ ടൈപ്പ് പ്രമോഷൻ എന്നും പറയുന്നു. ഡാറ്റ ഇനങ്ങൾ വലിപ്പത്തിന്റെ അവരോഹണ ക്രമത്തിൽ താഴെ പറയുന്ന രീതിയിലായിരിക്കും.

long double, double, float, unsigned long, long int, unsigned int short in

ഫലത്തിന്റെ ഡാറ്റാ ഇനം വലിയ ഓപ്പറാന്റിന്റെ ഡാറ്റാ ഇനമായിരിക്കും. ഉദാഹരണമായി 5/2*3+2.5 എന്ന പ്രയോഗത്തിന്റെ ഫലം 8.5 ആണ്. ഇത് എങ്ങിനെ ലഭിക്കുന്നു എന്ന് നോക്കാം.

ഘട്ടം 1: 5/2 → 2 പൂർണ്ണസംഖ്യയുടെ ഹരണം

ഘട്ടം 2: 2*3 → 6 പൂർണ്ണസംഖ്യയുടെ ഗുണനം

ഘട്ടം 3: 6+2.5→8.5 (ദശാംശസംഖ്യാ സങ്കലനം ഇവിടെ 6 നെ 6.0 ആക്കി മാറ്റുന്നു).

### 6.8.2: ബാഹ്യഇനംമാറ്റൽ (explicit type conversion /type casting):

ആന്തരിക ഡാറ്റാ ഇനം മാറ്റലിൽ നിന്നും വ്യത്യസ്തമായി ചില സാഹചര്യങ്ങളിൽ ചില പ്രോഗ്രാമർ തന്നെ ഫലത്തിന്റെ ഡാറ്റ ഇനം തീരുമാനിക്കേണ്ടതായി വരും. അങ്ങനെ വരുമ്പോൾ പ്രോഗ്രാമർ തന്നെ ഡാറ്റ ഇനം ആവരണ ചിഹ്നത്തിൽ ഓപ്പറാന്റിന്റെ ഇടതുവശത്തായി നൽകണം. ഇത് ഇനം മാറ്റാൻ ഉപകരിക്കുന്നു. ഈ രീതിയിൽ പ്രോഗ്രാമർ തന്നെ ആവശ്യമായ ഇനത്തിലേക്ക് ഡാറ്റയെ ഇനം മാറ്റുന്നതിനെ ബാഹ്യഇനം മാറ്റൽ അഥവാ ടൈപ്പ് കാസ്റ്റിംഗ് എന്നു പറയുന്നു. സാധാരണയായി പദപ്രയോഗത്തിലെ വേരിയബിളുകളുടെ ഇനം മാറ്റത്തിനാണ് ഇത് ഉപയോഗിക്കുന്നത്. കൂടുതൽ ഉദാഹരണങ്ങൾ ഭാഗം 6.9.2 ൽ ചർച്ച ചെയ്യാം.

## 6.9. പ്രസ്താവനകൾ (statements)

ഒരു ഭാഷയുടെ പഠനശ്രേണി എന്നത് അക്ഷരമാല, പദങ്ങൾ, ശൈലികൾ, വാക്യങ്ങൾ, ഖണ്ഡികകൾ തുടങ്ങിയവയാണ്. അതുപോലെ C++ന്റെ പഠനത്തിൽ അക്ഷരമാല (character set), ടോക്കണുകൾ (tokens), പദപ്രയോഗങ്ങൾ എന്നിവ നമ്മൾ മനസ്സിലാക്കി കഴിഞ്ഞു. പ്രസ്താവനകളുടെ സഹായത്തോടെ കമ്പ്യൂട്ടറുമായി യുക്തിപരമായും അർത്ഥവത്തായും സംവദിക്കാവുന്ന രീതിയിൽ നാമിപ്പോൾ എത്തിയിട്ടുണ്ട്. ഒരു പ്രോഗ്രാമിംഗ് ഭാഷയിലെ ഏറ്റവും ചെറിയ പ്രവർത്തന ഘടകമാണ് പ്രസ്താവനകൾ. ഒരു പ്രസ്താവന അവസാനിച്ചു എന്ന് സൂചിപ്പിക്കുവാൻ C++; (Semi column) ഉപയോഗിക്കുന്നു. C++ൽ വ്യത്യസ്ത ആവശ്യങ്ങൾക്കായി പ്രഖ്യാപന പ്രസ്താവനകൾ (declaration), വില നൽകുന്ന (assignment) പ്രസ്താവനകൾ, ഇൻപുട്ട് (input) പ്രസ്താവനകൾ, നിയന്ത്രണ പ്രസ്താവനകൾ (control), ഔട്ട്പുട്ട് (output) പ്രസ്താവനകൾ തുടങ്ങിയവ ഉപയോഗിക്കുന്നു. ഒരു C++പ്രോഗ്രാമിലെ ഓരോ പ്രസ്താവനക്കും അതിന്റേതായ ലക്ഷ്യങ്ങളുണ്ട്. ഇവയിൽ പ്രഖ്യാപന പ്രസ്താവനകൾ ഒഴികെയുള്ളവ ചില പ്രത്യേക പ്രവർത്തനങ്ങൾ കമ്പ്യൂട്ടർ ഉപയോഗിച്ച് ചെയ്യാനുള്ളവയാണ്. നിർവഹണ പ്രസ്താവനകൾ (execulable statements) കമ്പ്യൂട്ടറനുള്ള നിർദ്ദേശങ്ങളാണ്. നിയന്ത്രണപ്രസ്താവനകളുടെ പ്രവർത്തനം അദ്ധ്യായം 7ൽ ചർച്ച ചെയ്യാം.

മറ്റു ചില പ്രസ്താവനകളെ നമുക്കിവിടെ ചർച്ചചെയ്യാം.

### 6.9.1. പ്രഖ്യാപന പ്രസ്താവനകൾ (Declaration statement)

എല്ലാ ഉപയോക്തൃ നിർവചിത വാക്കുകളും പ്രോഗ്രാമിൽ അവ ഉപയോഗിക്കുന്നതിനു മുൻപുതന്നെ നിർവചിക്കേണ്ടതാണ്. ഒരു വേരിയബിൾ എന്നത് ഉപയോക്താവ് നിർവചിക്കുന്നതാണെന്നും മെമ്മറിയിലെ ഒരിടത്തിനെ സൂചിപ്പിക്കുന്നതാണെന്നും നാം കണ്ടു. ഉപയോഗിക്കുന്നതിന് മുൻപ് പ്രോഗ്രാമിൽ ഇവ പ്രഖ്യാപിക്കപ്പെടേണ്ടതുണ്ട്. നാം ഒരു വേരിയബിളിനെ പ്രഖ്യാപിക്കുമ്പോൾ അതിൽ സൂക്ഷിച്ചിരിക്കുന്ന ഡാറ്റയുടെ ഇനം ഏതാണെന്ന് കംപൈലറിനെ അറിയിക്കുകയാണ് ചെയ്യുന്നത്. വേരിയബിൾ പ്രഖ്യാപിക്കുന്നതിന്റെ വാക്യഘടന:

Data Type<variable>, [<variable 2>, < variable 3>...];

Data Type എന്നത് C++ലെ ഏതെങ്കിലും അംഗീകൃതമായ ഡാറ്റ ഇനം ആകാം. ഒന്നിലധികം വേരിയബിളുകൾ പ്രയോഗിക്കുമ്പോൾ അവയെ വേർതിരിക്കാൻ കോമ (,) ഉപയോഗിക്കണം. ഒരു പ്രഖ്യാപന പ്രസ്താവന അർദ്ധവിരാമം (;) ത്തോട് കൂടി അവസാനിക്കുന്നു. സാധാരണയായി വേരിയബിളുകൾ പ്രഖ്യാപിക്കുന്നത് അവ ഉപയോഗിക്കുന്നതിന് തൊട്ട് മുൻപോ അല്ലെങ്കിൽ പ്രോഗ്രാമിന്റെ തുടക്കത്തിലോ ആയിരിക്കും. വാക്യഘടനയിൽ [ ] ൽ നൽകിയിരിക്കുന്നത് ആവശ്യമുണ്ടെങ്കിൽ മാത്രം ഉപയോഗിച്ചാൽ മതി എന്ന അർത്ഥത്തിലാണ്. താഴെ കൊടുത്തിരിക്കുന്ന പ്രസ്താവനകൾ വേരിയബിൾ പ്രഖ്യാപനങ്ങൾക്ക് ഉദാഹരണങ്ങളാണ്:

```
int roll number;
double wgpa, avg-score;
```

ഒന്നാമത്തെ പ്രസ്താവനയിൽ വേരിയബിൾ roll number ഒരു int ഡാറ്റ ഇനമായതിനാൽ ഇതിനായി 4 ബൈറ്റ് മെമ്മറി മാറ്റിവക്കപ്പെടുന്നു. (gcc അനുസരിച്ച്) ഇതിൽ 2147483648 മുതൽ +2147483647 വരെയുള്ള ഏതെങ്കിലും പൂർണ്ണസംഖ്യ സൂക്ഷിക്കാവുന്നതാണ്.

രണ്ടാമത്തെ പ്രസ്താവന wgpa, avg-score എന്നീ double ഡാറ്റ ഇനത്തിലുള്ള വേരിയബിളുകൾ നിർവചിക്കുന്നു. ഇവ ഓരോന്നിനും 8 ബൈറ്റ് മെമ്മറി വീതം നീക്കി വയ്ക്കുന്നു. പ്രോഗ്രാം കംപൈൽ ചെയ്യുന്ന സമയത്ത് ഇവക്കുള്ള മെമ്മറി നീക്കി വയ്ക്കുന്നു.

#### വേരിയബിളുകൾക്ക് പ്രാരംഭവില നൽകൽ (Veriable initialisation)

വേരിയബിളുമായി ബന്ധപ്പെട്ട L മൂല്യം (വിലാസം), R മൂല്യം (ഉള്ളടക്കം) എന്നിങ്ങനെ രണ്ട് വിലകൾ ഉണ്ടെന്ന് ഭാഗം 6.5ൽ നാം കണ്ടു. ഒരു വേരിയബിൾ പ്രഖ്യാപിക്കുമ്പോൾ അതിനായി വിലാസത്തോടുകൂടിയ ഒരു മെമ്മറി ഭാഗം നീക്കി വെക്കുന്നു. എന്തായിരിക്കും അതിന്റെ ഉള്ളടക്കം? അത് പൂജ്യം, ശൂന്യം, സ്ഥലം/ വിടവ് എന്നിവയൊന്നും ആയിരിക്കില്ല! വേരിയബിളുകൾ int ഡാറ്റയായി പ്രഖ്യാപിക്കുമ്പോൾ വേരിയബിളുകളുടെ ഉള്ളടക്കം അഥവാ R വാല്യൂ എന്നത് അനുവദനീയമായ പരിധിക്ക് അകത്തുള്ള ഒരു പൂർണ്ണസംഖ്യ ആയിരിക്കും. എന്നാൽ ഈ സംഖ്യ പ്രവചിക്കാൻ സാധ്യമല്ല, എല്ലായിപ്പോഴും ഒരേ വില ആയിരിക്കണമെന്നുമില്ല. അതുകൊണ്ട് ഇതിനെ ഗാർബേജ് വില (garbage value) എന്ന് വിളിക്കുന്നു. നാം വേരിയബിളിന് ഒരു വില നൽകുമ്പോൾ അതിന്റെ പഴയ വിലയെ മാറ്റി പുതിയ വില ആക്കുന്നു. വേരിയബിളിന് കംപൈലേഷൻ സമയത്തോ പ്രോഗ്രാമിന്റെ പ്രവർത്തന (execution) സമയത്തോ വില നൽകാവുന്നതാണ്.

പ്രഖ്യാപന സമയത്തുതന്നെ വേരിയബിളിന് വില നൽകുന്നതിന് പ്രാരംഭ വിലനൽകൽ (variable initialisation) എന്നു പറയുന്നു. ഈ വില കപൈൽ സമയത്ത് മെമ്മറിയിൽ സംഭരിക്കപ്പെടുന്നു. ഇതിനായി അസ്സൈൻമെന്റ് ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു. താഴെ കൊടുത്തിരിക്കുന്നതു പോലെ രണ്ടു രീതിയിൽ ഇത് ചെയ്യാവുന്നതാണ്.

Data type variable = value

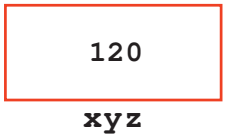
അല്ലെങ്കിൽ

Data type variable (value)

xyz എന്നൊരു വേരിയബിൾ പ്രഖ്യാപിച്ച് അതിന് 120 എന്ന വില നൽകുന്നതിനായി താഴെ പറയുന്ന രണ്ട് രീതികൾ സ്വീകാര്യമാണ്.

int xyz=120;

int xyz (120); ഈ രണ്ടു പ്രസ്താവനകളും xyz എന്ന ഇന്റിജർ വേരിയബിൾ പ്രഖ്യാപിച്ച് 120 എന്ന വില ചിത്രം 6.3 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ സൂക്ഷിക്കുകയും ചെയ്യുന്നു.



*ചിത്രം 6.3: വേരിയബിളിനു പ്രാരംഭവില നൽകൽ*

കൂടുതൽ ഉദാഹരണങ്ങൾ:

```
float val=0.12, b=5.234;
char k='A';
```

പ്രോഗ്രാമിന്റെ പ്രവർത്തനസമയത്തും വേരിയബിളുകൾക്ക് പ്രാരംഭവില നൽകാവുന്നതാണ്. ഇത് ഡൈനാമിക് പ്രാരംഭവില നൽകൽ എന്ന് അറിയപ്പെടുന്നു (dynamic initialisation). താഴെ കൊടുത്തിരിക്കുന്ന പ്രസ്താവനകളിലുള്ളതു പോലെ ഒരു പ്രയോഗത്തെ വേരിയബിളിലേക്ക് അസൈൻ ചെയ്യാൻ കഴിയും.

```
float product = x*y;
float interest = p*n*r/100.0;
```

ഒന്നാമത്തെ പ്രസ്താവനയിൽ x, y എന്നിവ പ്രവർത്തന സമയത്ത് ഗുണിച്ചു കിട്ടുന്ന ഫലമാണ് product എന്ന വേരിയബിളിന്റെ പ്രാരംഭ വില. രണ്ടാമത്തേതിൽ p*n*r/100.0; എന്നതിന്റെ ഫലമാണ്, interest എന്ന വേരിയബിളിൽ സംഭരിക്കുന്നത്.

ഡൈനാമിക് പ്രാരംഭ വിലനൽകുമ്പോൾ അസൈൻമെന്റ് ഓപ്പറേറ്ററിന് വലതുവശത്തുള്ള എല്ലാ വേരിയബിളുകളിലും സാധുവായ വില ഉണ്ടായിരിക്കണമെന്നത് ശ്രദ്ധിക്കുക. അല്ലെങ്കിൽ അപ്രതീക്ഷിത ഫലങ്ങൾ അത് സൃഷ്ടിക്കും.

### Const- ആക്സസ് മോഡിഫയർ

സംഖ്യ സ്ഥിരാംഗങ്ങൾ ഉപയോഗിക്കുതിനേക്കാൾ നല്ല രീതി അവയുടെ പ്രതീകങ്ങൾ ഉപയോഗിക്കുന്നതാണ്. ഉദാഹരണമായി 3.14 അല്ലെങ്കിൽ 22.0/7.0 എന്ന് ഉപയോഗിക്കുന്നതിന് പകരം pi എന്ന പ്രതീകം നമുക്ക് ഉപയോഗിക്കാം. ഇതിനായി const എന്ന കീവേഡ് ആണ് ഉപയോഗിക്കേണ്ടത്. const എന്ന സൂചികപദം (keyword) ഉപയോഗിച്ച് ഒരു പ്രതീകാത്മക സ്ഥിരാംഗം നിർമിക്കുമ്പോൾ ആ വേരിയബിളിന്റെ വില പ്രവർത്തന സമയത്ത് മാറ്റം വരുത്താൻ കഴിയാത്തതായിതീരുന്നു. താഴെ കൊടുത്തരിക്കുന്ന പ്രസ്താവന പരിഗണിക്കുക.

```
float Pi=3.14;
```

Pi എന്ന ദശാംശസംഖ്യാ വരിയബിളിന് 3.14 എന്ന പ്രാരംഭവില നൽകിയിരിക്കുന്നു. Pi എന്നതിന്റെ മൂല്യം പ്രോഗ്രാമിന്റെ പ്രവർത്തന സമയത്ത് മാറ്റം വരുത്താവുന്നതാണ്. ഈ പ്രഖ്യാപനത്തെ താഴെപറയുന്ന രീതിയിൽ നാം പരിഷ്കരിച്ചാൽ Pi യുടെ വില പ്രോഗ്രാമിന്റെ പ്രവർത്തനസമയം മുഴുവൻ സ്ഥിരമായിരിക്കും.

```
const float pi=3.14;
```

ഇതിന്റെ വില പിന്നീട് മാറ്റം വരുത്താൻ സാധ്യമല്ല. വേരിയബിളിൽ മൂല്യങ്ങൾ രേഖപ്പെടുത്താനും തിരിച്ചെടുക്കാനുള്ള അവകാശം (read/write accessibility) പരിഷ്കരിച്ച് തിരിച്ചെടുക്കുക എന്നത് മാത്രമാക്കി മാറ്റുന്നു. അതിനാൽ const എന്നത് ഒരു ആക്സസ് മോഡിഫയർ ആയി പ്രവർത്തിക്കുന്നു.

സോഫ്റ്റ്വെയർ വികസിപ്പിക്കുമ്പോൾ വലിയ പ്രോഗ്രാമുകൾ വികസിപ്പിക്കുന്നത് സംയുക്ത സംരംഭങ്ങളായിട്ടാണ്. ഒരേ പ്രോഗ്രാമിന്റെ പല ഭാഗങ്ങളിൽ ധാരാളം ആളുകൾ ജോലി ചെയ്യുന്നുണ്ടാകും അവർ ഒരേ വേരിയബിൾ പങ്കുവെക്കുന്നുണ്ടാകാം. ഇത്തരം സന്ദർഭങ്ങളിൽ ഒരാൾ വേരിയബിളിന്റെ ഉള്ളടക്കത്തിൽ വരുത്തുന്ന മാറ്റം മറ്റൊരാൾ തയ്യാറാക്കുന്ന കോഡിനെ ദോഷകരമായി ബാധിച്ചേക്കാം. ഇവിടെ മറ്റുള്ളവരുടെ പ്രവർത്തി വേരിയബിളിന്റെ ഉള്ളടക്കത്തെ ബാധിക്കാതെ നോക്കണം. const ഉപയോഗിച്ച് കൊണ്ട് ഇത് ചെയ്യാൻ കഴിയും.

### 6.9.2 അസൈൻമെന്റ് പ്രസ്താവനകൾ (Assignment statements)

ഒരു വേരിയബിളിലേക്ക് വില നൽകുന്നതിനാണ് അസൈൻമെന്റ് ഓപ്പറേറ്റർ (=) ഉപയോഗിക്കുന്നത്. ഇങ്ങനെ ഉപയോഗിക്കുന്നതിനുള്ള പ്രസ്താവനകളെ അസൈൻമെന്റ് പ്രസ്താവന എന്ന് വിളിക്കുന്നു.

താഴെപറയുന്ന ഏതെങ്കിലും രീതികളിൽ അവ എഴുതാം.

```
variable = constant;
variable1 = variable2;
variable = expression;
variable = function();
```

ഒന്നാമത്തേതിൽ ഒരു സ്ഥിരാംഗം വേരിയബിളിൽ സംഭരിക്കുന്നു. രണ്ടാമത്തേതിൽ വേരിയബിളിന്റെ വില മറ്റൊരു വേരിയബിളിൽ സംഭരിക്കുന്നു. മൂന്നാമത്തേതിൽ പദപ്രയോഗത്തിന്റെ ഫലം വേരിയബിളിൽ സംഭരിക്കുന്നു. അതുപോലെ നാലാമത്തേതിൽ ഫങ്ഷൻ തിരിച്ചുനൽകുന്ന വിലയാണ് വേരിയബിളിലേക്ക് സംഭരിക്കുന്നത്. ഫങ്ഷൻ എന്ന ആശയത്തെക്കുറിച്ച് അദ്ധ്യായം 10ൽ ചർച്ച ചെയ്യാം.

അസൈൻമെന്റ് പ്രസ്താവനകൾക്കുള്ള ചില ഉദാഹരണങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നു.

```
A = 15;
b = 5.8;
c = a+b;
d = (a+b) & (c+d)
r = sqrt (25);
```

അവസാനം നൽകിയിരിക്കുന്ന ഉദാഹരണത്തിൽ sqrt ( ) എന്നത് ഒരു ഫങ്ഷനാണ്. r എന്ന വേരിയബിളിൽ 25 ന്റെ വർഗമൂലമായ 5 ആണ് സംഭരിക്കപ്പെടുക.

അസൈൻമെന്റ് പ്രസ്താവനകളിൽ ഇടതുവശത്ത് ഒരു വേരിയബിൾ തന്നെ ആയിരിക്കണം. പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോൾ ആദ്യം വലതുവശം പ്രവർത്തിച്ചശേഷം കിട്ടുന്ന ഫലം ഇടതുവശത്തെ വേരിയബിളിൽ (RHS) സംഭരിക്കുന്നു.

താഴെകാണിച്ചിരിക്കുന്നതുപോലെ ഒന്നിൽ കൂടുതൽ അസൈൻമെന്റുകൾ കൂട്ടിച്ചേർത്ത് ഒരേ സമയം ചെയ്യാവുന്നതാണ്.

ഉദാഹരണത്തിന് x = y = z =13;

ഇവിടെ 13 എന്ന വില z,y,x എന്നീ ക്രമത്തിൽ മൂന്ന് വേരിയബിളുകൾക്കും നൽകുന്നു. അസൈൻമെന്റ് പ്രസ്താവനയ്ക്കുമുമ്പ് വേരിയബിളുകൾ പ്രഖ്യാപിച്ചിരിക്കണം. ഒരു വേരിയബിളിന് നാം വില നൽകുകയാണെങ്കിൽ അതിലുള്ള പഴയ വില മാറ്റി പുതിയ വില നൽകുന്നു.

### ഇനം അനുയോജ്യപ്പെടുത്തൽ

ഒരു വില നൽകൽ പ്രസ്താവന നടപ്പിലാക്കുമ്പോൾ RHS പ്രയോഗത്തിന്റെ ഡാറ്റ ഇനം LHS വേരിയബിളിൽ നിന്നും വ്യത്യസ്തമാകാം, അതിന് രണ്ട് സാധ്യതകളുണ്ട്. LHS-ലുള്ള വേരിയബിളിന്റെ ഡാറ്റ ഇനത്തിന്റെ വലിപ്പം RHS ലുള്ള വേരിയബിൾ അല്ലെങ്കിൽ പദപ്രയോഗത്തിലേതിനേക്കാളും കൂടുതലാകാം. ഈ സാഹചര്യത്തിൽ, RHS ലെ മൂല്യത്തിന്റെ ഡാറ്റ ഇനം LHSലെ വേരിയബിളിലേക്ക് ഉയർത്തപ്പെടുന്നതാണ് (ടൈപ്പ് പ്രൊമോഷൻ). താഴെയുള്ള കോഡ് ശകലം പരിഗണിക്കുക:

```
int a=5, b=2;
float p, q;
p = b;
q = a / p;
```

ഇവിടെ b യുടെ ഡാറ്റ ഇനം ടൈപ്പ് പ്രൊമോഷനിലൂടെ float ആക്കി മാറ്റുകയും 2.0 എന്നത് p യിൽ സൂക്ഷിക്കുകയും ചെയ്യുന്നു. q=a/p എന്ന പ്രസ്താവനയിൽ a യുടെ ഡാറ്റാഇനം ടൈപ്പ് പ്രൊമോഷൻ ചെയ്തുകഴിയുമ്പോൾ ഉത്തരം 2.5 എന്ന് ലഭിക്കുകയും q യിൽ സൂക്ഷിയ്ക്കപ്പെടുകയും ചെയ്യും.

LHS ന്റെ ഡാറ്റാഇനത്തിന്റെ വലിപ്പം RHS ന്റേതിനേക്കാൾ കുറവായിരിക്കാം എന്നതാണ് രണ്ടാമത്തെ സാധ്യത. RHS ന്റെ വില (truncate) ചുരുക്കി LHS നു അനുയയോജ്യമാക്കുന്നു. താഴെയുള്ള കോഡ് ഇത് വിശദീകരിക്കുന്നു.

```
float a=2.6;
int p, q;
p = a;
q = a * 4;
```

ഇവിടെ p യുടെ വില 2 ഉം q എന്നത് 10 ആകുന്നു. a*4 എന്ന പദപ്രയോഗത്തിന്റെ വില കാണുമ്പോൾ 10.4 എന്ന് ലഭിക്കുന്നു. എന്നാൽ q int ഇനത്തിൽപ്പെട്ടതിനാൽ 10 മാത്രമേ അതിൽ സൂക്ഷിക്കൂ. ഓപ്പറേറ്ററുകളുടെ ഡാറ്റ ഇനങ്ങളിൽ ചേർച്ചയില്ലാതെ വരുമ്പോൾ

ആഗ്രഹിക്കുന്ന ഫലം ലഭിക്കുന്നതിനായി പ്രോഗ്രാമർക്ക് ബാഹ്യഇനം മാറ്റൽ രീതി പ്രയോഗിക്കാവുനനതാണ്.

താഴെക്കൊടുത്തിരിക്കുന്ന പ്രോഗ്രാം ശകലം ശ്രദ്ധിക്കുക.

```
int p=5, q=2;
float x, y;
x = p/q;
y = (x+p)/q;
```

മുകളിലെ കോഡ് ശകലം പ്രവർത്തിച്ചു കഴിയുമ്പോൾ x ന്റെ വില 2.0 , y യുടെ വില 3.5 ആയിരിക്കും. p/q എന്ന പദപ്രയോഗം ഒരു പൂർണ്ണസംഖ്യാ പദപ്രയോഗം ആയതിനാൽ അതിന്റെ ഫലമായി 2 ലഭിക്കുകയും, ഫ്ളോട്ടിംഗ് പോയിന്റ് വിലയായി x ൽ സംഭരിക്കപ്പെടുകയും ചെയ്യുന്നു. X+p ബ്രാക്കറ്റിനകത്ത് ആയതിനാൽ മുൻഗണന ലഭിക്കുന്നു. xന്റെ ഇനം float ആയതിനാൽ p ടൈപ്പ് പ്രമോഷൻ ചെയ്യുകയും ഫലം 7.0 എന്ന് ലഭിക്കുകയും ചെയ്യും. പിന്നീട് 7.0 ഹരണക്രിയയുടെ ഒന്നാമത്തെ ഓപ്പെറന്റ് ആക്കി ഹരണക്രിയ നടത്തുന്നു q നെ float ആക്കി മാറ്റി ഫലം 3.5 എന്ന് ലഭിക്കുകയും ചെയ്യുന്നു. നമുക്ക് p /q ന്റെ ഫലം ദശാശസംഖ്യയായി x ൽ സംഭരിക്കുന്നതിന് ടൈപ്പ് കാസ്റ്റിങ് ഉപയോഗിച്ച് താഴെപ്പറയുന്ന രീതിയിൽ പ്രസ്താവന മാറ്റി എഴുതാവുന്നതാണ്.

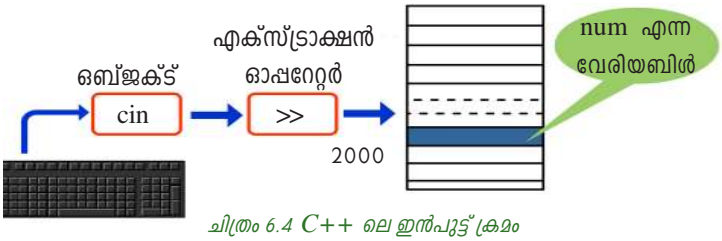
```
x=(float)p/q; or x=p/(float)q;
```

### 6.9.3 ഇൻപുട്ട് പ്രസ്താവനകൾ (Input statement)

പ്രോഗ്രാമിന്റെ പ്രവർത്തനസമയത്ത് ഉപയോക്താവിന് ഡാറ്റ മെമ്മറിയിൽ സംഭരിക്കുന്നതിന് ഉപയോഗിക്കുന്ന പ്രസ്താവനകൾ ഇൻപുട്ട് പ്രസ്താവനകൾ എന്നറിയപ്പെടുന്നു. ഗെറ്റ് ഫ്രം, എക്സ്ട്രാക്ഷൻ എന്നീ പേരുകളിലറിയപ്പെടുന്ന >> ഓപ്പറേറ്ററാണ് ഇൻപുട്ട് ഓപ്പറേറ്റർ എന്നും നാം കണ്ടതാണ്. ഡാറ്റ സൂക്ഷിക്കേണ്ട RAMലെ സ്ഥാനവും ഇൻപുട്ട് നൽകുന്ന ഉപകരണവുമാണ് ഇൻപുട്ട് ഓപ്പറേറ്ററുടെ രണ്ട് ഓപെറന്റുകൾ. ഒരു അംഗീകൃത ഇൻപുട്ട് ഉപകരണമായ കീബോർഡിൽ നിന്ന് വരുന്ന തുടർച്ചയായ ഡാറ്റാ പ്രവാഹത്തെ വേരിയബിളുകൾ സൂചിപ്പിക്കുന്ന മെമ്മറി സ്ഥാനങ്ങളിൽ സംഭരിക്കപ്പെടുന്നു.. c++ ഒരു ഒബ്ജക്റ്റ് ഓറിയന്റഡ് ഭാഷയായതിനാൽ കീബോർഡ് ഒരു അംഗീകൃത ഇൻപുട്ട് സ്ട്രീം ഉപകരണമായാണ് കണക്കാക്കപ്പെടുന്നത്. സി ഇൻ (cin) എന്ന പേരിലുള്ള ഒരു ഒബ്ജക്റ്റ് ആയി തിരിച്ചറിയപ്പെടുകയും ചെയ്യുന്നു. ഒരു ഇൻപുട്ട് പ്രസ്താവനയുടെ ലളിതമായ രൂപം താഴെ കൊടുത്തിരിക്കുന്നു.

```
streamobject >> variable;
```

കീ ബോർഡ് ഒരു ഇൻപുട്ട് ഉപകരണമായി നാം ഉപയോഗിക്കുന്നതിനാൽ മുകളിൽ പറഞ്ഞ വാക്യഘടനയിൽ stream object നു പകരം cin എന്നു എഴുതുന്നു. >> എന്ന ഓപ്പറേറ്ററിനു നിർബന്ധമായും ഒരു വേരിയബിൾ തന്നെയാകണം ഓപ്പറന്റ് ആയി ഉപയോഗിക്കേ




*ചിത്രം 6.4 C++ ലെ ഇൻപുട്ട് ക്രമം*

ണ്ടത്. ഉദാഹരണത്തിന്ന് താഴെ പറയുന്ന പ്രസ്ഥാവന കീബോർഡിൽ നിന്ന് ഡാറ്റാ സ്വീകരിക്കുകയും Num എന്ന വേരിയബിളിൽ സൂക്ഷിക്കുകയും ചെയുന്നു.

```
cin>>num;
```

ഡാറ്റ കീ ബോർഡിൽ നിന്നും സ്വീകരിച്ച് എങ്ങനെ വേരിയബിളിൽ സംഭരിക്കുന്നു എന്ന് ചിത്രം 6.4. ൽ കാണിച്ചിരിക്കുന്നു.

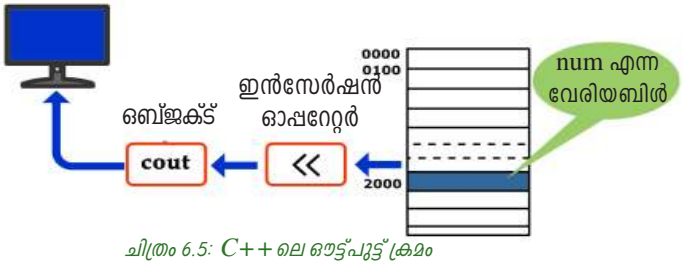
### 6.9.4 ഔട്ട്പുട്ട് പ്രസ്താവനകൾ (Output statements)

ഏതൊരു ഔട്ട്പുട്ട് ഉപകരണത്തിലൂടെയും ഉപയോക്താക്കൾക്ക് ഫലം ലഭ്യമാക്കുന്നതാണ് ഔട്ട്പുട്ട് പ്രസ്താവന. പുട്ട് ടു അല്ലെങ്കിൽ ഇൻസേർഷൻ എന്നീ പേരുകളിൽ അറിയപ്പെടുന്ന ഓപ്പറേറ്ററാണ് ഈ പ്രവർത്തനത്തിന് ഉപയോഗിക്കുന്നത്. ഇവിടെ ഔട്ട്പുട്ട് ചെയ്യേണ്ട ഡാറ്റയും ഔട്ട്പുട്ട് ഉപകരണവുമാണ് രണ്ട് ഓപെറന്റുകൾ. ഔട്ട്പുട്ട് പ്രസ്താവനയുടെ വാക്യഘടന ഇതാണ്.

```
streamobject << data;
```

stream object ഏതെങ്കിലും ഔട്ട്പുട്ട് ഉപകരണമാകാം. Data ഒരു സ്ഥിരാംഗമോ ഒരു വേരിയബിളോ അല്ലെങ്കിൽ ഒരു പദപ്രയോഗമോ ആകാം. മോണിറ്റർ ആണ് സാധാരണ ഉപയോഗിക്കുന്ന ഔട്ട്പുട്ട് ഉപകരണം. C++ ൽ cout (സി ഔട്ട് എന്ന് ഉച്ചരിക്കുന്നു) എന്നതാണ് മോണിറ്ററിനെ സൂചിപ്പിക്കുന്ന ഒബ്ജക്റ്റിന്റെ പേര്. മോണിറ്റർ ഔട്ട്പുട്ട് ഉപകരണമായി ഉപയോഗിക്കുന്ന ചില ഔട്ട്പുട്ട് പ്രസ്താവനകൾക്കുള്ള ഉദാഹരണങ്ങളാണ് താഴെ പറയുന്നവ.

```
cout << num;
cout << "hello friends";
cout << num+12;
```




*ചിത്രം 6.5: C++ ലെ ഔട്ട്പുട്ട് ക്രമം*

ഒന്നാമത്ത പ്രസ്താവന num ന്റെ വില മോണിറ്ററിൽ പ്രദർശിപ്പിക്കുന്നു. രണ്ടാമത്തേത് hello friends എന്ന സ്ട്രിംഗ് പ്രദർശിപ്പിക്കുന്നു. അവസാനത്തേതിൽ num നോടുകൂടി 12 കൂട്ടികിട്ടുന്ന ഫലം പ്രദർശിപ്പിക്കുന്നു. (num ൽ സംഖ്യയാണെന്ന് കരുതുക). മെമ്മറി സ്ഥാനം num ൽ നിന്ന് ഡാറ്റ എങ്ങനെയാണ് stream object (മോണിറ്റർ)ൽ ചേർത്തിരിക്കുന്നത് എന്ന് ചിത്രം 6.5. ൽ കാണിച്ചിരിക്കുന്നു.

ടോക്കണുകളായ cin ഉം cout ഉം കിവേഡുകളല്ല. C++ ഭാഷയുടെ ഭാഗമല്ലാത്ത മുൻ നിർവചിത വാക്കുകളാണിവ. ഉപയോക്താവിന് ഇവയെ പുനർ വ്യാഖ്യാനം ചെയ്യാനാവുന്നതാണ്. C++ ഭാഷയുടെ ലൈബ്രറിയിൽ നിർവ്വചിച്ചിട്ടുള്ള അടിസ്ഥാനവാക്കാണിത്. വ്യക്തമായി പറയുകയാണെങ്കിൽ മുൻ നിർവചിത വാക്കുകളെ പുനർവ്യാഖ്യാനം ചെയ്ത് മാറ്റാവശ്യങ്ങൾക്ക് ഉപയോഗിക്കുന്നത് അപകടകരവും ചിന്താക്കുഴപ്പം വരുത്തുന്നതുമാണ്. ഇത് ഒഴിവാക്കേണ്ടതാണ്. എല്ലാ മുൻ നിർവചിത ഐഡന്റി ഫയറുകളെയും കീവേർഡുകളെ പോലെ കരുതുന്നതാണ് സുരക്ഷിതവും ഏളുപ്പവുമായ രീതി.

### I/O ഓപ്പറേറ്ററുകളുടെ സംയോജനം

നമുക്ക് x,y,z എന്നീ മൂന്നു പേരുകളിലായി ഡാറ്റ ഇൻപുട്ട് ചെയ്യുന്നതിനായി

```
cin>>x;
cin>>y;
cin>>z;
```

ഇങ്ങനെ 3 പ്രസ്താവനകൾ ഉപയോഗിക്കാം; താഴെ പറയുന്ന രീതിയിൽ ഇവ മൂന്നും കൂട്ടി യോജിപ്പിച്ച് ഒറ്റ പ്രസ്താവനയായി ഉപയോഗിക്കാവുന്നതാണ്.

```
cin>>x>>y>>z;
```

ഒന്നിൽകൂടുതൽ ഇൻപുട്ട് ഔട്ട്പുട്ട് ഓപ്പറേറ്ററുകൾ ഒറ്റ പ്രസ്താവനയിൽ ഉപയോഗിക്കുന്നതിന് സംയോജിപ്പിച്ച് ഉണ്ടാക്കുമ്പോൾ ഓഫ് ഇൻപുട്ട് ഔട്ട്പുട്ട് ഓപ്പറേറ്ററുകളുടെ സംയോജനം എന്നു പറയുന്നു. ഇൻപുട്ട് ഓപ്പറേറ്ററുകൾ ചെയ്യുമ്പോൾ ആദ്യം നൽകുന്ന വില ആദ്യത്തെ വേരിയബിളിന് ലഭിക്കും. രണ്ടാമത്തേ വില രണ്ടാമത്തേതിന് അങ്ങനെ ഇടത്തുനിന്ന് വലത്തേക്ക് വില ലഭിക്കും. ഉദാഹരണമായി cin>>x>>y>>z; ഒന്നാമത് നൽകുന്ന വില x നും രണ്ടാമത്തേത് y ക്കും മൂന്നാമത്തേത് z നും ലഭിക്കും. പ്രവർത്തനസമയത്ത് വിലനൽകുമ്പോൾ വേരിയബിളുകളുടെ വിലകൾ തമ്മിൽ വേർതിരിക്കുന്നതിന് സ്പൈസ്, ബാർ, ടാബ്, അല്ലെങ്കിൽ എന്റർ കീ ഇവ ഏതെങ്കിലും ഉപയോഗിക്കാം.

ഇതുപോലെ ഒന്നിലധികം വേരിയബിളുകളുടെ വിലകൾ മോണിറ്ററിൽ കാണിക്കുന്നതിനായി താഴെ പറയുന്ന രീതി ഉപയോഗിക്കാം.

```
cout<<x<<y<<z;
```

വേരിയബിളുകൾ സ്ഥിരാംഗങ്ങൾ പദപ്രയോഗങ്ങൾ എന്നിവ ഒരുമിച്ച് ഔട്ട്പുട്ട് ചെയ്യാനായി താഴെ പറയുന്ന രീതി ഉപയോഗിക്കാം

```
cout<<"The number is "<<z;
```

ഔട്ട്പുട്ട് ഓപ്പറേറ്ററുകൾ സംയോജിപ്പിച്ച് ഉപയോഗിക്കുമ്പോൾ വലത്തുനിന്ന് ഇടത്തേക്കായിരിക്കും ഔട്ട്പുട്ട് വിലകൾ ക്രമമാക്കുന്നത്. താഴെ കൊടുത്തിരിക്കുന്ന കോഡ് ശകലം ശ്രദ്ധിക്കുക.

```
int x=5;
cout<<x<<'\t'<<++x;
```

ഔട്ട്പുട്ട് : 6 6

ഔട്ട്പുട്ട് 5 6 എന്ന് ആയിരിക്കില്ല.

<<,>> ഓപ്പറേറ്ററുകളെ ഒരേ പ്രസ്താവനയിൽ ഉപയോഗിക്കാൻ കഴിയില്ല എന്നത് ശ്രദ്ധിക്കുക. x=y=z=5; എന്ന പ്രസ്താവനയിൽ = ഓപ്പറേറ്റർ സംയോജിപ്പിച്ചു ഉപയോഗിച്ചിരിക്കുന്നു. ഇവിടെയും വലത്തുനിന്ന് ഇടത്തോട്ടാണ് സംയോജനം നടക്കുന്നത്.

## 6.10 ഒരു C++ പ്രോഗ്രാമിന്റെ ഘടന (Structure of a C++ program)

ഇതുവരെ ചർച്ച ചെയ്ത പ്രസ്താവനകൾ ഉപയോഗിച്ച് ലഘുവായ പ്രശ്നങ്ങൾ പരിഹരിക്കാവുന്ന സ്ഥിതിയിൽ നാം ഇപ്പോൾ എത്തിക്കഴിഞ്ഞു. എന്നാൽ ഒരു കൂട്ടം പ്രസ്താ

വനകൾ മാത്രം ചേർന്നാൽ ഒരു പ്രോഗ്രാമാകുകയില്ല. C++ പ്രോഗ്രാമിന് ഒരു സവിശേഷ ഘടനയുണ്ട്. അത് ഒന്നോ അതിലധികമോ ഫങ്ക്ഷനുകളുടെ ശേഖരമാണ്. ഫങ്ഷൻ എന്നാൽ ഒരു പേരിൽ സ്വരൂപിച്ചിരിക്കുന്നതും ഒരു പ്രത്യേക കാര്യം ചെയ്യുന്നതിനായുമുള്ള പ്രസ്താവനകളുടെ കൂട്ടമാണ്. ഒരു C++ പ്രോഗ്രാമിൽ ഒന്നിലധികം ഫങ്ക്ഷനുകൾ ഉപയോഗിക്കുന്നതിനാൽ അവ ഓരോന്നും അനന്യമായ പേരുകളിൽ ആയിരിക്കണം തിരിച്ചറിയപ്പെടേണ്ടത്. എല്ലാപ്രോഗ്രാമിലും ഏറ്റവും അത്യാവശ്യമായി ഉണ്ടായിരിക്കേണ്ട ഫങ്ക്ഷനാണ് main() ഫങ്ഷൻ.

ഒരു C++ പ്രോഗ്രാമിന്റെ ഘടന താഴെ കൊടുത്തിരിക്കുന്നു

```
#include <header file>
using namespace identifier;
int main()
{
    statements;
      :
      :
      :
    return 0;
}
```

ഒന്നാമത്തെ വരിയെ പ്രീ - പ്രൊസസ്സർ നിർദ്ദേശം എന്നും രണ്ടാമത്തെ വരിയെ നെയിം സ്പേസ് പ്രസ്താവന എന്നും വിളിക്കുന്നു. മൂന്നാമത്തെ വരിയിൽ ഫങ്ഷൻ ഹെഡറും തുടർന്നുള്ള വരികളിൽ ഒരു ജോഡി ബ്രാക്കറ്റുകൾക്കുള്ളിൽ ഉള്ള ഒരു കൂട്ടം പ്രസ്ഥാവനകളുമാണ് അടങ്ങിയിരിക്കുന്നത്.

പ്രോഗ്രാമിലെ ഈ ഓരോ ഭാഗങ്ങളും നമുക്ക് ചർച്ച ചെയ്യാം.

### 6.10.1 പ്രീപ്രൊസസ്സർ നിർദ്ദേശങ്ങൾ (Preprocessor directive)

പ്രീ പ്രോസസ്സർ നിർദ്ദേശങ്ങളോടു കൂടിയാണ് ഒരു C++ പ്രോഗ്രാം ആരംഭിക്കുന്നത്.ഇവ കംപൈലറിനുള്ള നിർദേശ പ്രസ്താവനകളാണ്. കംമ്പൈലേഷൻ ആരംഭിക്കുന്നതിനു മുമ്പ് കമ്പൈലർ ചെയ്യേണ്ടുന്ന കാര്യങ്ങൾ ഇത് നിർദ്ദേശിക്കുന്നു. പ്രോഗ്രാം പ്രസ്താവനകൾ അല്ലാത്തതും എന്നാൽ പ്രോഗ്രാമിൽ ഉൾപ്പെട്ടിട്ടുള്ളതുമായ വരികളാണ് പ്രീപ്രൊസസ്സർ നിർദ്ദേശങ്ങൾ. ഈ വരികൾ എപ്പോഴും # ചിഹ്നത്തോടു കൂടിയാണ് തുടങ്ങുന്നത്. പ്രോഗ്രാമിന് ആവശ്യമായ സൗകര്യങ്ങൾ ലഭ്യമാക്കാൻ C++ ലൈബ്രറിയിലെ ശീർഷക ഫയലുകളെ #include എന്നുതുടങ്ങുന്ന പ്രീപ്രോസസ്സർ നിർദ്ദേശം ഉപയോഗിച്ച് ബന്ധിപ്പിക്കുന്നു. ഈ വരികളുടെ അവസാനം അർദ്ധവിരാമം (;) ആവശ്യമില്ല. വിവിധ ശീർഷക ഫയലുകൾക്ക് വേണ്ടി വ്യത്യസ്ത #include പ്രസ്താവനകൾ ഉപയോഗിക്കണം. #define, #undef മുതലായവ മറ്റു ചില പ്രീപ്രോസസ്സർ നിർദ്ദേശങ്ങളാണ്.

### 6.10.2 ഹെഡർ ഫയലുകൾ (Header files)

ഫംഗ്ഷനുകൾ, ഒബ്ജക്ടുകൾ മുൻനിർവചിത - രൂപീകൃത ഡാറ്റഇനങ്ങൾ എന്നിവയെക്കുറിച്ചുള്ള വിവരങ്ങൾ കപൈലറിനോടൊപ്പം ലഭ്യമായിട്ടുള്ള ശീർഷകംമായിട്ടുള്ള

ശീർഷക ഫയലുകളിൽ അടങ്ങിയിരിക്കുന്നു. ലൈബ്രറിയിൽ സൂക്ഷിച്ചിട്ടുള്ള ഇത്തരം നിരവധി ഫയലുകൾ C++ പ്രോഗ്രാമുകളെ പിന്തുണയ്ക്കുന്നു. ഇത്തരത്തിലുള്ള ഏതെങ്കിലും വിവരങ്ങൾ ആവശ്യമുള്ള പ്രോഗ്രാമുകളിൽ അവ ഉപയോഗിക്കുന്നതിന് അതുമായി ബന്ധപ്പെട്ട ഹെഡർ ഫയൽ ഉൾപ്പെടുത്തണം. ഉദാഹരണത്തിന് മുൻ നിർവചിത ഒബ്ജക്ടുകളായ cin, cout എന്നിവ നമുക്ക് ഉപയോഗിക്കേണ്ടതായി വരുമ്പോൾ പ്രോഗ്രാമിന്റെ തുടക്കത്തിൽ താഴെ പറയുന്ന പ്രസ്താവന നമ്മൾ ഉപയോഗിക്കണം.

#include <iostream> എന്ന ഹെഡർ ഫയലിൽ cin, cout എന്നീ ഒബ്ജക്ടുകളുടെ വിവരങ്ങൾ അടങ്ങിയിരിക്കും. ഹെഡർ ഫയലുകൾക്ക് h എക്സ്റ്റൻഷൻ ഉണ്ടെങ്കിലും GCC ൽ അതു ഉപയോഗിക്കരുത്. പക്ഷേ ടർബോ C++ IDE പോലെയുള്ള മറ്റു ചില കംപൈലറുകൾക്ക് ഈ എക്സ്റ്റൻഷൻ നിർബന്ധമാണ്.

### 6.10.3. നെയിംസ്പേസ് എന്ന ആശയം (Concept of namespace)

ഒരു പ്രോഗ്രാമിൽ ഒരേ വ്യാപ്തിയിൽ ഒരേ പേരിലുള്ള ഒന്നിലധികം ഐഡന്റിഫയറുകൾ (വേരിയബിളുകൾ അല്ലെങ്കിൽ ഫംങ്ഷനുകൾ) ഉണ്ടായിരിക്കാൻ പാടില്ല. ഉദാഹരണത്തിന് നമ്മുടെ വീട്ടിൽ രണ്ടോ അതിലധികമോ ആളുകൾക്ക് (അല്ലെങ്കിൽ ജീവജാലങ്ങൾക്ക്) ഒരേ പേരുണ്ടാവില്ല. അങ്ങനെയുണ്ടെങ്കിൽ തീർച്ചയായും വീട്ടിനുള്ളിൽ അവരെ പേരു കൊണ്ട് തിരിച്ചറിയുക എന്നത് വിഷമകരമാകും. അതുകൊണ്ട് നമ്മുടെ വീട്ടിൽ ഓരോ പേരും അനന്യമായിരിക്കണം. എന്നാൽ നമ്മുടെ അയൽപക്കത്തെ വീട്ടിൽ സമാനമായ പേരുള്ള ഒരാൾ (അല്ലെങ്കിൽ ജീവജാലം) ഉണ്ടായിരിക്കാം. അതാത് പരിധിക്കുള്ളിൽ വ്യക്തികളെ പേരു കൊണ്ട് തിരിച്ചറിയുന്നതിന് ഇത് യാതൊരു ആശയക്കുഴപ്പവുമുണ്ടാക്കില്ല. പക്ഷേ പുറമേ നിന്നൊരു വ്യക്തിക്ക് പേരു മാത്രം ഉപയോഗിച്ച് കൊണ്ട് ഇവരെ തിരിച്ചറിയാൻ കഴിയില്ല. അതിനായി വീട്ടുപേരു കൂടി പരാമർശിക്കേണ്ടതുണ്ട്.

നെയിം സ്പേസ് എന്ന ആശയം വീട്ടുപേരിനു സമാനമാണ്. ഒരു പ്രത്യേക നെയിംസ് പേസുമായി വ്യത്യസ്ത ഐഡന്റിഫയറുകൾ ബന്ധപ്പെട്ടിരിക്കുന്നു. ഓരോ ഇനവും വ്യത്യസ്തമായിരിക്കുന്ന ഒരു ഗണത്തിന്റെ പേരാണിത്. വേരിയബിളുകൾക്കും ഫംഗ്ഷനുകൾക്കുമായി പ്രത്യേകം നെയിം സ്പേസുകൾ സൃഷ്ടിക്കുന്നതിന് ഉപയോക്താവിനു അനുവാദമുണ്ട്. ഒരു നെയിംസ്പേസിനു പേരു കൊടുക്കാൻ നമുക്ക് ഒരു ഐഡന്റിഫയർ ഉപയോഗിക്കാം. പ്രോഗ്രാമിംഗിൽ ഉപയോഗിക്കുന്ന ഘടകങ്ങളെ ഏത് നെയിം സ്പേസിൽ തിരയണമെന്ന് using എന്ന കീവേർഡ് സാങ്കേതികമായി കംപൈലറിനോട് പറയുന്നു. C++ ൽ standard എന്നതിന്റെ ചുരുക്കെഴുത്താണ് std. cin, cout തുടങ്ങി മറ്റ് പല ഒബ്ജക്ടുകളും നിർവചിച്ചിട്ടുള്ള ഒരു നെയിം സ്പേസ് ആണിത്. അതിനാൽ ഒരു പ്രോഗ്രാമിൽ ഇവ ഉപയോഗിക്കണമെങ്കിൽ std::cin, std::cout എന്ന മാതൃക നാം പിന്തുടരേണ്ടതാണ്. using name space std എന്ന പ്രസ്താവന പ്രോഗ്രാമിൽ ഉപയോഗിക്കുന്നതിലൂടെ ഇത്തരത്തിലുള്ള വിശദമായ പരാമർശങ്ങൾ ഒഴിവാക്കാവുന്നതാണ്. അത്തരമൊരു സാഹചര്യത്തിൽ കംപൈലർ cin, cout, endl മുതലായവയ്ക്കായി ഈ നെയിംസേപസിൽ തിരയുന്നു. cin, cout, endl അല്ലെങ്കിൽ അതുപോലെയുള്ളവ എപ്പോഴൊക്കെ ഒരു C++ പ്രോഗ്രാമിൽ കമ്പ്യൂട്ടർ കാണുന്നുവോ, അവയെ std::cin, std::cout, std::endl എന്നിങ്ങനെ വ്യഖ്യാനിക്കുന്നു.

using name space std എന്ന പ്രസ്താവന യഥാർത്ഥത്തിൽ പ്രോഗ്രാമിലേക്ക് ഒരു ഫംഗ്ഷനും കൂട്ടിച്ചേർക്കുന്നില്ല, #include<iostream> എന്ന നിർദ്ദേശമാണ് cin, cout, endl അതുപോലെയുള്ളവ ഉൾപ്പെടുത്തുന്നത്.

### 6.10.4 main() ഫങ്ഷൻ

എല്ലാ C++ പ്രോഗ്രാമിലും main() എന്നു പേരുള്ള ഫങ്ഷൻ ഉൾപ്പെട്ടിരിക്കുന്നു. പ്രോഗ്രാമിന്റെ പ്രവർത്തനം ആരംഭിക്കുന്നതും അവസാനിക്കുന്നതും main() ഫങ്ഷനിലാണ്. മറ്റ് ഏതെങ്കിലും ഫങ്ഷനുകൾ നാം പ്രോഗ്രാമിൽ ഉപയോഗിക്കുന്നുണ്ടെങ്കിൽ അവയെ വിളിക്കുന്നത് main() ഫങ്ഷനിൽ നിന്നാണ്. സാധാരണയായി main() ഫങ്ഷൻ മുമ്പായി ഒരു ഡാറ്റ ഇനം ഉണ്ടായിരിക്കും. GCC -ൽ ഇത് int ആയിരിക്കണം.

main () ഫങ്ഷന്റെ ഹെഡറിനെ തുടർന്ന് ഒരു ജോഡി ബ്രാക്കറ്റുകൾക്കുള്ളിൽ ഒന്നോ അതിലധികമോ പ്രസ്താവനകൾ അടങ്ങിയ ഫങ്ഷന്റെ ചട്ടക്കൂടും ഉണ്ടായിരിക്കും. ഈ ഘടന main ഫങ്ഷന്റെ നിർവചനം എന്ന് അറിയപ്പെടുന്നു. ഓരോ പ്രസ്താവനയും ഒരു അർദ്ധവിരാമത്തിൽ ';' അവസാനിക്കുന്നു. പ്രസ്താവനകൾ നിർവഹിക്കാവുന്നവയോ നിർവഹിക്കാനാവത്തവയോ ആകാം. കമ്പ്യൂട്ടർ ചെയ്യേണ്ട കാര്യങ്ങൾക്കുള്ള നിർദേശങ്ങളാണ് നിർവഹണ പ്രസ്താവനകൾ പ്രതിനിധാനം ചെയ്യുന്നത്. നിർവഹിക്കാനാവാത്ത പ്രസ്താവനകൾ കമ്പയിലറിനെയോ പ്രോഗ്രാമറെയോ ഉദ്ദേശിച്ചുള്ളവയാണ്. അവ വിവരാധിഷ്ഠിത പ്രസ്താവനകൾ ആണ്. main() ഫംഗ്ഷന് ഉള്ളിലെ അവസാനത്തെ പ്രസ്താവന return 0 എന്നായിരിക്കും. ഈ പ്രസ്താവനകൾ നാം ഉപയോഗിച്ചില്ലെങ്കിലും പ്രോഗ്രാമിൽ അത് ഒരു തരത്തിലുള്ള പിശകും വരുത്തുന്നില്ല. ഇതിന്റെ പ്രസക്തി അധ്യായം 10-ൽ ചർച്ച ചെയ്യാം. ഓരോ പ്രസ്താവനകളും പുതിയ വരികളിൽ തുടങ്ങണമെന്ന് നിർബ്ബന്ധമില്ലാത്തതിനാൽ C++ ഒരു സ്വതന്ത്ര രൂപത്തിലുള്ള ഭാഷയാണ്. അതുപോലെ ഒരു പ്രസ്താവനക്ക് ഒന്നിൽ കൂടുതൽ വരികൾ ഉപയോഗിക്കാവുന്നതാണ്.

### 6.10.5 ഒരു മാതൃക പ്രോഗ്രാം

പൂർണ്ണമായ ഒരു പ്രോഗ്രാം നമുക്ക് പരിശോധിച്ച് അതിന്റെ സവിശേഷതകൾ വിശദമായി പരിചയപ്പെടാം. ഈ പ്രോഗ്രാം സ്ക്രീനിൽ ഒരു വാചകം പ്രദർശിപ്പിക്കും.

```
#include<iostream.h>
void main()
{
        cout<<"Hello, Welcome to C++";
}
```

താഴെ പറയുന്ന രീതിയിൽ ഈ പ്രോഗ്രാമിന് ഏഴു വരികളുണ്ട്.

വരി 1. പ്രീ പ്രോസസ്സർ നിർദേശം #include എന്നത് iostream.h എന്ന ഹെഡർ ഫയലിനെ ഈ പ്രോഗ്രാമുമായി ബന്ധിപ്പിക്കുന്നു.

വരി 2. using name space std എന്ന പ്രസ്താവന cout എന്ന ഐഡന്റി ഫയറിനെ പ്രോഗ്രാമിൽ ലഭ്യമാക്കുന്നു.

വരി 3. പ്രോഗ്രാമിൽ നിർബന്ധമായും ഉണ്ടായിരിക്കേണ്ട മെയിൻ എന്ന ഫങ്ഷന്റെ ഹെഡർ

വരി 4. ആദ്യത്തെ ബ്രാക്കറ്റ് ({)പ്രസ്താവനകൾ തുടങ്ങുന്നു എന്നു കാണിക്കുന്നു.

വരി 5. നാം പ്രോഗ്രാം പ്രവർത്തിപ്പിക്കുമ്പോൾ ഈ ഔട്ട്പുട്ട് പ്രസ്താവന പ്രവർത്തിച്ച് Hello, welcome to C++ എന്ന് മോണിറ്ററിൽ പ്രദർശിപ്പിക്കുന്നു. Cout പ്രസ്താവന ഈ പ്രോഗ്രാമിൽ ഉപയോഗിക്കുന്നതിന് iostream എന്ന് ഹെഡർ ഫയൽ ഉൾപ്പെടുത്തിയിട്ടുണ്ട്.

വരി 6. return പ്രസ്താവന main() ഫങ്ഷന്റെ പ്രവർത്തനം അവസാനിപ്പിക്കുന്നു. ഈ പ്രസ്താവന main ഫങ്ഷനെ സംബന്ധിച്ചിടത്തോളം നിർബന്ധമല്ല.

വരി 7. അവസാന ബ്രാക്കറ്റ് (}) ഈ പ്രോഗ്രാം അവസാനിപ്പിച്ചതായി സൂചിപ്പിക്കുന്നു.

## 6.11 പ്രോഗ്രാം എഴുതുന്നതിനുള്ള മാർഗ്ഗ നിർദ്ദേശങ്ങൾ

ഒരു പ്രോഗ്രാം കോഡ് യുക്തിസഹവും സ്പഷ്ടവും തെറ്റുകൾ പെട്ടെന്ന് കണ്ടെത്തുവാൻ കഴിയുന്നതുമാണെങ്കിൽ അത് ഒരു നല്ല സോഴ്സ് കോഡ് ആയിരിക്കും. പ്രോഗ്രാമുകൾ എഴുതുമ്പോൾ ചില രീതികൾ പിന്തുടരുകയാണെങ്കിൽ ഈ സവിശേഷതകൾ നമുക്ക് അനുഭവവേദ്യമാക്കാം.

ശൈലീപരമായ പ്രോഗ്രാമുകൾ എഴുതുന്നതിനുള്ള ചില മാർഗ്ഗ നിർദ്ദേശങ്ങൾ ഈ ഭാഗത്ത് ചർച്ച ചെയ്യുന്നു.

**ഐഡന്റിഫയറുകൾക്ക് യോജിച്ച പേര് നൽകുക.**

ഒരു ജോലിക്കാരന്റെ കിഴിവുകൾക്കുശേഷമുള്ള ശമ്പളം നമുക്ക് കണക്കാക്കണം എന്നിരിക്കട്ടെ. താഴെ കാണുന്ന രീതിയിൽ നമുക്ക് കോഡ് ചെയ്യാവുന്നതാണ്.

ഇവിടെ A എന്നത് മിച്ച ശമ്പളവും b മൊത്ത ശമ്പളവും c ആകെ കിഴിവും ആണ്, എന്നാൽ ഈ പേരുകൾ അവയുടെ ഉപയോഗത്തെ പ്രതിഫലിപ്പിക്കുന്നില്ല. ഇതേ പ്രസ്താവന താഴെ പറയുന്ന രീതിയിലായാൽ കൂടുതൽ വ്യക്തമായിരിക്കും.

```
Net_Slary=Gross_Slary-Deduction;
```

ഇവിടെ വേരിയബിളുകളുടെ പേരുകൾ അവയുടെ മൂല്യവുമായി പൊരുത്തമുള്ളതും പെട്ടെന്ന് ഓർത്തിരിക്കാൻ പറ്റുന്നതുമാണ്. ഈ പേരുകൾ അവയുടെ ഉദ്ദേശ്യത്തെ പ്രതിഫലിപ്പിക്കുന്നു.ഇത്തരം പേരുകളെ ന്യൂമോണിക് പേരുകൾ (mnemonic names) എന്നു വിളിക്കുന്നു. പേരുകൾ സ്വീകരിക്കുമ്പോൾ താഴെപ്പറയുന്ന കാര്യങ്ങൾ ശ്രദ്ധിക്കണം.

1. വേരിയബിളുകൾ, ഫങ്ഷനുകൾ, പ്രൊസീഡറുകൾ എന്നിവക്ക് നല്ല ന്യൂമോണിക് പേരുകൾ തിരഞ്ഞെടുക്കാം.

   ഉദാഹരണം: avg_hgt, Roll_No, emp_code, Sum Of Digits, തുടങ്ങിയവ

2. ബന്ധപ്പെട്ട വേരിയബിളുകൾക്ക് നിലവാരമുള്ള പിൻ വാക്കുകളും, മുൻ വാക്കുകളും ഉപയോഗിക്കാം.

   ഉദാഹരണം: (മൂന്ന് സംഖ്യകൾക്കായി) num1, num2, num3

3. പ്രോഗ്രാമിന്റെ തുടക്കത്തിൽതന്നെ സ്ഥിരാംഗങ്ങൾക്ക് പേരുകൾ നൽകുക.

ഉദാഹരണം: const float PI = 3.14;

**വ്യക്തവും ലളിതവുമായ പ്രയോഗങ്ങൾ ഉപയോഗിക്കുക**

പ്രവർത്തനസമയം കുറയ്ക്കുന്നതിനായി പ്രോഗ്രാമുകളുടെ ലാളിത്യം നഷ്ടപ്പെടുത്തുന്ന പ്രവണത ചില ആളുകൾക്കുണ്ട്. ഇത് ഒഴിവാക്കേണ്ടതാണ്. താഴെ പറയുന്ന ഉദാഹരണം പരിഗണിക്കുക. x നെ yകൊണ്ട് ഹരിച്ച്കിട്ടുന്ന ശിഷ്ടം കാണുന്നതിന് y=x-(x/n)*n; എന്ന പ്രസ്താവന ഉപയോഗിക്കാം. ഇതേ കാര്യത്തിനായി താഴെ കാണുന്ന ലളിതവും സുന്ദരവുമായ കോഡ് ഉപയോഗിക്കാവുന്നതാണ്.

y=x%n;

അതു കൊണ്ട് ഒരു പ്രോഗ്രാമിനെ ലളിതവും വ്യക്തവുമാക്കുന്നതിന് ലളിതമായ കോഡുകൾ ഉപയോഗിക്കുന്നതാണ് നല്ലത്.

**ആവശ്യമുള്ളിടത്ത് കമന്റുകൾ ഉപയോഗിക്കുക.**

ഒരു പ്രോഗ്രാമിന്റെ മുകളിൽ വിവരണം നൽകുന്നതിന് കമന്റുകൾ വളരെ പ്രധാനപ്പെട്ട പങ്ക് വഹിക്കുന്നു. അവയെ പ്രോഗ്രാമുകളെ വിശദീകരിക്കുവാനുള്ള വരികളായിട്ടാണ് പ്രോഗ്രാമിനുള്ളിൽ കൂട്ടി ചേർത്തിരിക്കുന്നത്. കമ്പൈലറുകൾ അവയെ അവഗണിക്കുന്നു. C++ ൽ കമന്റുകൾ എഴുതുന്നതിന് രണ്ടു മാർഗ്ഗങ്ങളുണ്ട്.

**ഒറ്റവരി കമന്റ്:** '//' ചിഹ്നങ്ങളാണ് ഒറ്റവരി കമന്റുകൾ എഴുതാനായി ഉപയോഗിക്കുന്നത്. ഒരു വരിയിൽ // ന് ശേഷമുള്ള വാക്യങ്ങൾ കമന്റുകളായി C++ കമ്പൈലർ കണക്കാക്കുന്നു.

**ഖണ്ഡിക കമന്റ് (multiline comment):** /* നും */ നും ഇടയിൽ എഴുതുന്ന എന്തിനെയും കമ്പൈലർ കമന്റ് ആയി കണക്കാക്കുന്നു. ആയതിനാൽ ഒരു കമന്റിൽ എത്ര വരികൾ വേണമെങ്കിലും ഉൾപ്പെടുത്താം. പക്ഷേ പ്രോഗ്രാമിൽ ആവശ്യമായ പ്രസ്താവനകൾ കമന്റുകളിൽ ആയി പോകാതിരിക്കാൻ പ്രത്യേകം ശ്രദ്ധിക്കേണ്ടതാണ്.

**കമന്റുകൾ നൽകുമ്പോൾ താഴെ പറയുന്ന കാര്യങ്ങൾ ശ്രദ്ധിക്കണം**

- പ്രോഗ്രാമിന്റെ തുടക്കത്തിലുള്ള കമന്റുകൾ പ്രോഗ്രാമിന്റെ ഉദ്ദേശ്യത്തെ സംഗ്രഹിക്കുന്നതായിരിക്കണം.
- ഓരോ വേരിയബിളും, സ്ഥിരാംഗവും പ്രഖ്യാപിക്കുമ്പോൾ കമന്റുകൾ ഉപയോഗിക്കുക.
- സങ്കീർണമായ പ്രോഗ്രാം ഘട്ടങ്ങൾ വിശദീകരിക്കുന്നതിന് കമന്റുകൾ ഉപയോഗിക്കുക.
- പ്രോഗ്രാം എഴുതുന്ന സമയത്തു തന്നെ കമന്റുകൾ ഉൾപ്പെടുത്തുന്നതാണ് നല്ലത്.
- ലളിതവും വ്യക്തവുമായി കമന്റുകൾ എഴുതുക.

**ഇന്റന്റേഷന്റെ ആവശ്യകത**

കമ്പ്യൂട്ടർ പ്രോഗ്രാമിംഗിൽ പ്രോഗ്രാം ഘടന വ്യക്തമാക്കുന്നതിന് കോഡുകൾ മാർജിനിൽ നിന്നും നിശ്ചിത അകലത്തിൽ എഴുതുന്ന സമ്പ്രദായമുണ്ട്. ഇതിനെ ഇന്റന്റേഷൻ എന്ന് പറയുന്നു. ഇത് പ്രസ്താവനകളുടെ വായന സുഗമമാക്കുകയും പ്രോഗ്രാമിന് വ്യക്തത വരുത്തുകയും ചെയ്യുന്നു.

ഇത് പ്രോഗ്രാമിലെ പ്രസ്താവനകളുടെ വിവിധ നിലകളെ സൂചിപ്പിക്കുന്നു.

ഈ മാർഗ്ഗ നിർദ്ദേശങ്ങളുടെ ഉപയോഗം അടുത്ത ഭാഗത്തുള്ള പ്രോഗ്രാമുകളിൽ നിരീക്ഷിക്കാവുന്നതാണ്.

## പ്രോഗ്രാമുകൾ

പ്രോഗ്രാം 6.1 ഒരു സന്ദേശം പ്രദർശിപ്പിക്കുന്നു.

കോഡിംഗ് മാർഗനിർദേശങ്ങൾ അനുസരിച്ച് ചില പ്രശ്നങ്ങൾ നിർദ്ധാരണം ചെയ്യുന്നതിനുള്ള പ്രോഗ്രാമുകൾ നമുക്ക് ഇപ്പോൾ എഴുതാം. തന്നിരിക്കുന്ന കോഡുകൾ പ്രോഗ്രാമിന്റെ ഭാഗമല്ല.

**പ്രോഗ്രാം 6.1 ഒരു സന്ദേശം കാണിക്കുന്നതിന്**

```
    /*  This program displays the message
        "Smoking is injurious to health"
        on the monitor */
#include <iostream.h> // To use the cout object
void main() //program begins here
{   //The following output statement displays a message
    cout << "Smoking is injurious to health";
}   //end of the program
```

ഖണ്ഡിക കമന്റ്

ഒറ്റവരി കമന്റ്

ഇൻന്റെന്റേഷൻ

പ്രോഗ്രാം 6.1 പ്രവർത്തിക്കുമ്പോൾ താഴെ കൊടുത്തിരിക്കുന്ന ഔട്ട്പുട്ട് ലഭിക്കും.

ഔട്ട്പുട്ട് :

Smoking is injurious to health

അദ്ധ്യായം 7ലെ ഉദാഹരണങ്ങളിൽ ഇന്റന്റേഷന്റെ ഉപയോഗം കൂടുതൽ വിവരിച്ചിരിക്കുന്നു.

പ്രോഗ്രാം 6.2 ഉപയോക്താവിൽ നിന്ന് രണ്ടു പൂർണ്ണ സംഖ്യകൾ സ്വീകരിക്കുകയും അവയുടെ തുക കണ്ടുപിടിക്കുകയും ഫലം പ്രദർശിപ്പിക്കുകയും ചെയ്യുന്നു.

**പ്രോഗ്രാം 6.2: രണ്ട് പൂർണ്ണസംഖ്യകളുടെ തുക കണ്ടുപിടിക്കാൻ**

```
#include<iostream.h>
{     //Program begins
/* Two variables are declared to read user inputs and the variable
sum is declared to store the result
*/
      int num1, num2, sum;
```

```
      cout<<"Enter two numbers: "; //Prompt for input
      cin>>num1>>num2; //Cascading to get two numbers
      sum=num1+num2; //Assignment statement to find the sum
      cout<<"Sum of the entered numbers = "<<sum;
/* The result is displayed with proper message.
      Cascading of output operator is utilized */
return  0;
}
```

പ്രോഗ്രാം 6.2 ന്റെ ഒരു മാതൃക ഔട്ട്പുട്ട് ചുവടെ കൊടുത്തിരിക്കുന്നു.

ഔട്ട്പുട്ട് :

Enter two numbers: 5 7

ഉപയോക്താവിന്റെ ഇൻപുട്ടുകൾ സ്പേസ് ഉപയോഗിച്ച്

Sum of the entered numbers = 12

നമുക്ക് മറ്റൊരു പ്രശ്നം പരിഗണിക്കാം. തുടർച്ചയായ മൂന്നു മൂല്യനിർണയങ്ങളിൽ ഒരു വിദ്യാർഥിക്ക് മൂന്ന് സ്കോറുകൾ ലഭിച്ചു. ഒരു പ്രവർത്തിയിലെ കൂടിയ സ്കോർ 20 ആണ്. വിദ്യാർഥിയുടെ ശരാശരി സ്കോർ കണ്ടുപിടിക്കുക.

**പ്രോഗ്രാം 6.3 : മൂന്നു CE സ്കോറുകളുടെ ശരാശരി കണ്ടുപിടിക്കുന്നതിന്**

```
#include<iostream.h>
void main()
{
      int score_1, score_2, score_3;
      float avg;
      //Average of 3 numbers can be a floating point value
      cout << "Enter the three CE scores: ";
      cin >> score_1 >> score_2 >> score_3;
      avg = (score_1 + score_2 + score_3) / 3.0;
/* The result of addition will be an integer value. If 3 is written
instead of 3.0, integer division will be performed and will not get
the correct result */
      cout << "Average CE score is: " << avg;
}
```

CE സ്കേറുകളായി 17,19,20 എന്നിവ നൽകുമ്പോൾ പ്രോഗ്രാം 6.3 താഴെയുള്ള ഔട്ട്പുട്ട് നൽകുന്നു.

```
Enter the three CE scores: 17 19 20
Average CE score is: 18.666666
```

ശരാശരി വില കാണുന്നതിനായി വിലനൽകൽ പ്രസ്താവന, വില നൽകൽ ഓപ്പറേറ്റ റിന്റെ (=) വലതുഭാഗത്ത് ഒരു പദപ്രയോഗം ഉപയോഗിക്കുന്നു. ഈ പദപ്രയോഗത്തിൽ രണ്ടു+ ഓപ്പറേറ്ററുകളും ഒരു / ഓപ്പറേറ്ററുമുണ്ട്./ ഓപ്പറേറ്റർക്ക് + ഓപ്പറേറ്ററിനുമേലുള്ള മുൻഗണന, സങ്കലനത്തിനായി ആവരണ ചിഹ്നങ്ങൾ ഉപയോഗിച്ച് മാറ്റം വരുത്തുന്നു. സങ്കലന ഓപ്പറേറ്ററുകളുടെ ഓപ്പറന്റുകളെല്ലാം int ഇനം ഡാറ്റയായുതുകൊണ്ടുതന്നെ ഫലവും ഇന്റിജർ ആയിരിക്കും. ഈ ഇന്റിജർ ഫലത്തെ 3 കൊണ്ടു ഗുണിക്കുമ്പോൾ ഔട്ട്പുട്ട് വീണ്ടുമൊരു ഇന്റിജർ ആയിരിക്കും. അങ്ങനെയായിരുന്നെങ്കിൽ പ്രോഗ്രാം 6.3ന്റെ ഔട്ട് പുട്ട് 18 ആകുമായിരുന്നു. അത് കൃത്യമാകുകയില്ല. അതിനാൽ / ഓപ്പറേറ്റ റുടെ രണ്ടാമത്തെ ഓപ്പറേറ്ററായി 3.0 എന്ന ദശാംശ സ്ഥിരാംഗം ഉപയോഗിക്കുന്നു. ഇത് ഡാറ്റാ ഇനം ഉയർത്തലിലൂടെ ഇന്റിജർ അംശത്തെ float ആക്കുന്നു.

ഒരു വൃത്തത്തിന്റെ ആരം 'r' എന്നു നൽകി അതിന്റെ വിസ്തീർണവും ചുറ്റളവും കണക്കു കൂട്ടുവാൻ നിങ്ങളോട് അഭ്യർഥിക്കുന്നു എന്നിരിക്കട്ടെ. നമുക്കറിയാവുന്നതുപോലെ വൃത്ത ത്തിന്റെ വിസ്തീർണം $r^2$ എന്ന സമവാക്യം ഉപയോഗിച്ചും, ചുറ്റളവ് 2 r ഉപയോഗിച്ചും ( ഇവിടെ =3.14) മാണ് കണക്കാക്കുന്നത്. പ്രോഗ്രാം 6.4 ഈ പ്രശ്നം പരിഹരിക്കുന്നു.

**പ്രോഗ്രാം 6.4: തന്നിരിക്കുന്ന ആരത്തിന് അനുസരിച്ച് ഒരു വൃത്തത്തിന്റെ വിസ്തീർണവും ചുറ്റളവും കണ്ടുപിടിക്കാൻ വേണ്ടി.**

π

```
#include <iostream>
using namespace std;
int main()
{
    const float PI = 22.0/7; //Use of const access modifier
    float radius, area, perimeter;
    cout<<"Enter the radius of the circle: ";
    cin>>radius;
    area = PI * radius * radius;
    perimeter = 2 * PI * radius;
    cout<<"Area of the circle = "<<area<<"\n";
    cout<<"Perimeter of the circle = "<<perimeter;
    return 0;
}
```

Areaയുടെ വില പ്രദർശിപ്പിച്ച ശേഷം' \n ' എന്ന എസ്കേപ് സീക്വൻസ് ഒരു പുതിയ വരിയിലേക്ക് കർസറിനെ കൊണ്ടുപോകുന്നു.

പ്രോഗ്രാം 6.4ന്റെ ഒരു മാതൃക ഔട്ട്പുട്ട് താഴെ കൊടുത്തിരിക്കുന്നു.

```
Enter the radius of the circle: 2.5
Area of the circle = 19.642857
Perimeter of the circle = 15.714285
```

പ്രോഗ്രാം 6.4ന്റെ അവസാനത്തെ രണ്ട് ഔട്ട്പുട്ട് പ്രസ്താവനകൾ രണ്ടു ഫലങ്ങളെയും വ്യത്യസ്ത വരികളിലായി പ്രദർശിപ്പിക്കുന്നു. അവസാനത്തെ ഔട്ട്പുട്ട് പ്രസ്താവന പ്രവർത്തിക്കുന്നതിനു മുമ്പായി എന്ന എസ്കേപ്പ് സീക്വൻസ് കർസറിനെ പുതിയ വരിയിലേക്ക് വരുത്തുന്നു.

വെറും പലിശ കണക്കാക്കുന്നതിന് മറ്റൊരു പ്രോഗ്രാം നമുക്ക് എഴുതി നോക്കാം. നമുക്കറിയാവുന്നതു പോലെ ഫലം കണക്കാക്കുന്നതിനായി മൂലധനം, പലിശയുടെ ശതമാനം, കാലാവധി എന്നിവ ഇൻപുട്ട് ആയി നൽകേണ്ടതാണ്.

**പ്രോഗ്രാം 6.5: വെറും പലിശ കണ്ടുപിടിക്കുക**

```
#include <iostream>
using namespace std;
int main()
{
    float p_Amount, n_Year, i_Rate, int_Amount;
    cout<<"Enter the principal amount in Rupees: ";
    cin>>p_Amount;
    cout<<"Enter the number of years for the deposit: ";
    cin>>n_Year;
    cout<<"Enter the rate of interest in percentage: ";
    cin>>i_Rate;
    int_Amount = p_Amount * n_Year * i_Rate /100;
    cout <<"Simple interest for the principal amount "
         <<p_Amount<<" Rupees for a period of "<<n_Year
         <<" years at the rate of interest "<<i_rate
         <<" is "<<int_Amount<<" Rupees";
    return 0;
}
```

പ്രോഗ്രാം 6.5ന്റെ ഒരു മാതൃക ഔട്ട്പുട്ട് താഴെ കൊടുത്തിരിക്കുന്നു.

```
Enter the principal amount in Rupees: 100
Enter the number of years for the deposit: 2
Enter the rate of interest in percentage: 10
Simple interest for the principal amount 100 Rupees for a
period of 2 years at the rate of interest 10 is 20 Rupees
```

പ്രോഗ്രാം 6.5ലെ അവസാനത്തെ പ്രസ്താവനയാണ് ഔട്ട്പുട്ട് പ്രസ്താവന. ഇത് നാലു വരികളിലായി സജ്ജീകരിച്ചിരിക്കുന്നു. ഓരോ വരിയുടെയും അവസാനത്തിൽ അർദ്ധവിരാമം ഇല്ല എന്നതു പ്രത്യേകം ശ്രദ്ധിക്കുക. അതിനാൽ ഇത് ഒറ്റ പ്രസ്താവനയായിട്ടാണ് പരിഗണിക്കപ്പെടുന്നത്. പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോൾ നിങ്ങളുടെ കമ്പ്യൂട്ടർ

മോണിട്ടറിന്റെ വലിപ്പത്തിനും റെസലൂഷനും അനുസരിച്ച് ഒന്നിലധികം വരികളിലായി പ്രദർശിപ്പിക്കപ്പെടുന്നു.

പ്രോഗ്രാം 6.6 ഒരു താപനില പരിവർത്തന പ്രശ്നം പരിഹരിക്കുന്നു. താപനില ഡിഗ്രി സെൽഷ്യസിൽ ഇൻപുട്ട് ആയി നൽകുകയും ഔട്ട്പുട്ട് തത്തുല്യമായ ഫാരൻ ഹീറ്റിലും ആയിരിക്കും.

**പ്രോഗ്രാം 6.6: താപനില സെൽഷ്യസിൽ നിന്നും ഫാരൻഹീറ്റിലേക്ക് പരിവർത്തനം ചെയ്യാൻ വേണ്ടി**

```
#include <iostream>
using namespace std;
int main()
{
    float celsius, fahrenheit;
    cout<<"Enter the Temperature in Celsius: ";
    cin>>celsius;
    fahrenheit=1.8*celsius+32;
    cout<< celsius<<" Degree Celsius = "
        << fahrenheit<<" Degree Fahrenheit";
    return 0;
}
```

പ്രോഗ്രാം 6.6 ന്റെ ഒരു മാതൃക ഔട്ട്പുട്ട് താഴെ കൊടുത്തിരിക്കുന്നു.

```
Enter the Temperature in Celsius: 37
37 Degree Celsius = 98.599998 Degree Fahrenhei
```

C++ ലെ ഓരോ ക്യാരക്ടർ സ്ഥിരാംഗത്തിനും ASCIIകോഡ് എന്നു വിളിക്കുന്നു. ഒരു അനന്യവിലയുണ്ടെന്ന് നമുക്കറിയാം. ഈ വിലകൾ ഇന്റിജറുകളാണ്. തന്നിരിക്കുന്ന ക്യാരക്ടറിന്റെ ASCII കോഡ് കണ്ടെത്തുവാനുള്ള പ്രോഗ്രാം നമുക്ക് എഴുതി നോക്കാം.

**പ്രോഗ്രാം 6.7 ഒരു ക്യാരക്ടറിന്റെ ASCII വില കണ്ടെത്തുവാൻ വേണ്ടി**

```
#include <iostream>
using namespace std;
int main()
{
    char ch;
    int asc;
    cout << "Enter the character: ";
    cin >> ch;
    asc = ch;
    cout << "ASCII value of "<<ch<<"  = " << asc;
    return 0;
```

പ്രോഗ്രാം 6.7 ന്റെ ഒരു മാതൃക ഔട്ട്പുട്ട് താഴെ കൊടുത്തിരിക്കുന്നു.

```
Enter the character: A
ASCII value of A = 65
```

## നമുക്ക് സംഗ്രഹിക്കാം

ഡാറ്റയുടെ തരം തിരിച്ചറിയുന്നതിനും അവയുമായി ബന്ധപ്പെട്ട ക്രിയകളെ കൈകാര്യം ചെയ്യുന്നതിനുമുള്ള ഒരു ഉപാധിയാണ് ഡാറ്റ ഇനങ്ങൾ. ഓരോ ഡാറ്റ ഇനത്തിലെ ഡാറ്റയ്ക്കും അതിന്റേതായ വലിപ്പവും പരിധിയുമുണ്ട്. വേരിയബിളുകൾ നിർവചിക്കാൻ ഡാറ്റ ഇനങ്ങൾ ഉപയോഗിക്കുന്നു. C++ ൽ വിവിധ ക്രികകൾക്കായി വ്യത്യസ്തതരം ഓപ്പറേറ്ററുകൾ ലഭ്യമാണ്. ഓപ്പറേറ്ററുകൾ ഓപ്പറന്റുകളുമായി (ഡാറ്റ) കൂട്ടി ചേർക്കുമ്പോൾ പ്രയോഗങ്ങൾ രൂപപ്പെടുന്നു. മൂന്നു തരത്തിലുള്ള പ്രയോഗങ്ങളാണുള്ളത്. അരിത്മാറ്റിക്, റിലേഷണൽ, ലോജിക്കൽ അരിത്മാറ്റിക് പ്രയോഗങ്ങളിൽ നിന്നും ഉദ്ദേശിച്ചു ഫലങ്ങൾ ലഭ്യമാകുന്നതിന് ചില സാഹചര്യങ്ങളിൽ തരം മാറ്റൽ ഉപയോഗിക്കുന്നു. ഒരു പ്രോഗ്രാമിന്റെ ഏറ്റവും ചെറിയ പ്രവർത്തന ഭാഗമാണ് പ്രസ്താവന വേരിയബിളിനെ പ്രഖ്യാപിക്കുന്ന പ്രസ്താവനകൾ. പ്രോഗ്രാമിൽ ഒരു വേരിയബിളിലെ നിർവചിക്കുകയും അവക്ക് മെമ്മറി സ്ഥാനം നീക്കി വയ്ക്കുകയും ചെയ്യുന്നു. വില നൽകൽ പ്രസ്താവന, ഇൻപുട്ട് പ്രസ്താവനകൾ, ഔട്ട് പുട്ട് പ്രസ്താവനകൾ മുതലായവ കമ്പ്യൂട്ടറുകൾക്ക് നിർദ്ദേശങ്ങൾ നൽകാൻ സഹായിക്കുന്നു. അരിത്മാറ്റിക് വിലനൽകൽ, പ്രസ്താവന, ഡിക്രിമെന്റ് മുതലായ ചില പ്രത്യേക ഓപ്പറേറ്ററുകൾ പ്രയോഗങ്ങളെയും പ്രസ്താവനകളെയും ചുരുക്കുകയും തൻമൂലം പ്രോഗ്രാമിന്റെ പ്രവർത്തന വേഗം കൂട്ടുകയും ചെയ്യുന്നു. C++ പ്രോഗ്രാമിന് തനതായ ഒരു ഘടനയുണ്ട്. പ്രോഗ്രാമുകൾ തയാറാക്കുമ്പോൾ അത് തീർച്ചയായും പിന്തുടരേണ്ടതാണ്. പ്രോഗ്രാം ആകർഷണീയവും മനുഷ്യർക്കിടയിൽ വിനിമയ യോഗ്യവുമാക്കുവാൻ ശൈലീപരമായ മാർഗ്ഗരേഖകൾ പിന്തുടരേണ്ടതാണ്.

## പഠന നേട്ടങ്ങൾ

ഈ അധ്യായത്തിൽ പൂർത്തികരണത്തോടെ പഠിതാവിന്

- C++ ലെ വിവിധ ഡാറ്റ ഇനങ്ങൾ തിരിച്ചറിയാൻ സാധിക്കുന്നു.
- ഉചിതമായ ഡാറ്റ ഇനങ്ങളുടെ തരം മാറ്റൽ പട്ടികപ്പെടുത്താനും ആവശ്യമായത് തിരഞ്ഞെടുക്കുവാനും സാധിക്കുന്നു.
- ഉചിതമായ വേരിയബിളുകൾ തിരഞ്ഞെടുക്കുവാൻ സാധിക്കുന്നു.
- വിവിധ ഓപ്പറേറ്ററുകൾ പരീക്ഷിച്ചു നോക്കുവാൻ സാധിക്കുന്നു.
- വിവിധ I/O ഓപ്പറേറ്ററുകൾ പ്രയോഗിക്കുവാൻ സാധിക്കുന്നു.
- വിവിധ പ്രയോഗങ്ങളും പ്രസ്താവനകളും എഴുതുവാൻ സാധിക്കുന്നു.

- ഒരു ലളിതമായ C++ പ്രോഗ്രാമിന്റെ ഘടന തിരിച്ചറിയാൻ സാധിക്കുന്നു.
- പ്രോഗ്രാമിൽ ശൈലിപരമായ മാർഗ്ഗരേഖകളുടെ ആവശ്യകത തിരിച്ചറിയാൻ സാധിക്കുന്നു.
- C++ പ്രോഗ്രാമുകൾ എഴുതുവാൻ സാധിക്കുന്നു.

## ലാബ് പ്രവർത്തനം

1. ഉപയോഗക്താവിനോട് തൂക്കം ഗ്രാമിൽ നൽകാൻ ആവശ്യപ്പെടുകയും പിന്നീട് തത്തുല്യ കിലോഗ്രാമായി പ്രദർശിപ്പിക്കുകയും ചെയ്യാനുള്ള ഒരു പ്രോഗ്രാം എഴുതുക.
2. താഴെ പറയുന്ന പട്ടിക നിർമ്മിക്കാനുള്ള പ്രോഗ്രാം എഴുതുക

| | |
|---|---|
| 2013 | 100% |
| 2012 | 99.9% |
| 2011 | 95.5% |
| 2010 | 90.81% |
| 2009 | 85% |

ഔട്ട്പുട്ടിനായി ഒറ്റ പ്രസ്താവന ഉപയോഗിക്കുക.

(സൂചന: \n ഉം \t ഉം ഉപയോഗിക്കുക)

4. നിങ്ങളുടെ ഉയരം മീറ്ററിലും സെന്റീമീറ്ററിലും ആവശ്യപ്പെടുകയും അതിനെ ഫീറ്റിലും ഇഞ്ചിലുമായി പരിവർത്തനം ചെയ്യുന്നതിലുള്ള ഒരു ചെറിയ പ്രോഗ്രാം എഴുതുക. (1 ഫൂട്ട് =12 ഇഞ്ച്, 1 ഇഞ്ച് = 2.54 cm)
5. വെറും പലിശയും കൂട്ടുപലിശയും കണക്കാക്കുന്നതിനുള്ള പ്രോഗ്രാം എഴുതുക.
6. പ്രോഗ്രാം എഴുതുക : (i) തന്നിരിക്കുന്ന അക്കത്തിന്റെ ASCII കോഡ് പ്രിന്റ് ചെയ്യാൻ, (ii) ബാക്ക് സ്പേസിന്റെ ASCII കോഡ് പ്രിന്റ് ചെയ്യാൻ. (സൂചന: ബാക്ക് സ്പേസിന്റെ എസ്കേപ്പ് സീക്വൻസ് ഒരു ഇന്റിജർ വേരിയബിളിൽ സംഭരിക്കുക)
7. സമയം സെക്കന്റുകളായി സ്വീകരിച്ച് hrs: mins: secs രൂപത്തിലേക്ക് പരിവർത്തനം ചെയ്യാനുള്ള ഒരു പ്രോഗ്രാം എഴുതുക. ഉദാഹരണത്തിന്, ഇൻപുട്ട് 3700 സെക്കന്റുകൾ ആണെങ്കിൽ ഔട്ട്പുട്ട് 1hr: 1 min: 40 secsഎന്നായിരിക്കും.

## മാതൃകാ ചോദ്യങ്ങൾ

### ഹ്രസ്വ ചോദ്യങ്ങൾ

1. ഡാറ്റ ഇനങ്ങൾ എന്നാലെന്ത്? C++ലെ എല്ലാ മുൻനിർമ്മിത ഡാറ്റ ഇനങ്ങളും എഴുതുക.
2. സ്ഥിരാംഗം എന്നാലെന്ത്?

3. പ്രവർത്തന സമയത്തെ വേരിയബിളുകളുടെ പ്രാരംഭ വിലനൽകൽ (dynamic initialisation) എന്നാൽ എന്ത്?
4. ഇനം മാറ്റൽ എന്നാലെന്ത്?
5. പ്രഖ്യാപന പ്രസ്താവനയുടെ ഉദ്ദേശമെന്ത്?
6. പ്രോഗ്രാമുകളിൽ cin, coutഎന്നിവ ഉപയോഗിക്കുന്നതിനായി ഉൾപ്പെടുത്തേണ്ട ഹെഡർ ഫയലിന്റെ പേര് എഴുതുക.
7. ">>" ഇൻപുട്ട് ഓപ്പറേറ്ററിനെയും "<<" ഔട്ട് പുട്ട് ഓപ്പറേറ്ററിനെയും എന്താണ് വിളിക്കുന്നത്?
8. a (i) float ഉം (ii) int ഉം ആയാൽ a = 5/3ന്റെ ഫലമെന്തായിരിക്കും?
9. പ്രാരംഭത്തിൽ **i** എന്നത് 22 ഉം P= 3ഉം ആണെങ്കിൽ P= P++ + ++i യുടെ വില എന്തായിരിക്കും?
10. പ്രാരംഭത്തിൽ j =5 ആണെങ്കിൽ താഴെ കാണുന്ന പദപ്രയോഗങ്ങളുടെ വില കണക്കാക്കുക.

    (i) (5*++j)%6 (ii) (5*j++)%6
11. താഴെ പറയുന്ന പദപ്രയോഗങ്ങളുടെ പ്രവർത്തന ക്രമം എന്തായിരിക്കും.

    (i) i+5>=j-6 (ii) s+10<p-2+2*q
12. പ്രാരംഭത്തിൽ ans എന്നത് 6 ആണെങ്കിൽ താഴെ കാണുന്നവയുടെ ഫലം എന്തായിരിക്കും

    (i) cout <<ans = 8 ; (ii) cout << ans == 8

## ലഘു വിവരണാത്മകം

1. വേരിയബിൾ എന്നാലെന്ത്? അതുമായി ബന്ധപ്പെട്ട രണ്ടു വിലകൾ എഴുതുക.
2. C++ എത്ര തരത്തിൽ വേരിയബിളുകളെ പ്രഖ്യാപിക്കാം?
3. C++ ലെ വേരിയബിളിന്റെ പ്രഖ്യാപനത്തിൽ ഇനം മാറ്റൽ ഉപയോഗിക്കുന്നതിന്റെ അനന്തരഫലമെന്ത്?
5. 'const' കീവേർഡിന്റെ പങ്കെന്താണ്?
6. വർദ്ധനവ് പ്രവർത്തനത്തിൽ എങ്ങനെയാണ് പ്രീഫിക്സ് രൂപം പോസ്റ്റ്ഫിക്സ് രൂപത്തിൽ നിന്നും വ്യത്യാസപ്പെട്ടിരിക്കുന്നത് എന്ന് വിവരിക്കുക.
8. sizeof ഓപ്പറേറ്റർ നിർവഹിക്കുന്ന പ്രവർത്തനത്തെപറ്റി എഴുതും.
9. ഇനം മാറ്റലിന്റെ രണ്ടു രീതികളെ പറ്റി വിവരിക്കുക.
10. ഒരു പ്രോഗ്രാമിൽ main() ഇല്ലെങ്കിൽ എന്ത് സംഭവിക്കും?
11. താഴെ കാണുന്ന കോഡ് ശകലത്തിലുള്ള തെറ്റുകൾ തിരിച്ചറിയുക.

(a) int main()

{ cout << "Enter two numbers"

cin >> num >> auto

float area = Length * breadth ; }

(b) #include <iostream>

using namespace std

void Main()

{ int a, b

cin <<a <<b

max=(a > b) a:b

cout>max

}

12. താഴെ കാണുന്ന C++ പ്രസ്താവനകളിൽ തെറ്റുകളുണ്ടെങ്കിൽ കണ്ടെത്തുക.

(i) `cout<< "a=" a;`
(ii) `m=5,n=12;015`
(iii) `cout << "x" ; <<x;`
(iv) `cin >> y`
(v) `cin >> "\n" >> y ;`
(vi) `cout >> \n "abc"`
(vii) `a = b + c`
(viii) `break = x`

13. റിലേഷണൽ ഓപ്പറേറ്ററുകളുടെ കർത്തവ്യമെന്ത്? == ഉം = ഉം തമ്മിൽ വേർതിരിക്കുക.

14. ഒരു പ്രോഗ്രാമിന്റെ ഗ്രഹണവും വായനാക്ഷമതയും വർദ്ധിപ്പിക്കാൻ കമന്റുകൾ ഉപകരിക്കുന്നു. ഈ പ്രസ്താവന ഉദാഹരണസഹിതം സമർഥിക്കുക.

### വിവരണാത്മകം

1. C++ ലെ ഓപ്പറേറ്ററുകൾ വിശദമായി വിവരിക്കുക.
2. C++ ലെ വിവിധതരം പദപ്രയോഗങ്ങളും ഇനം മാറ്റൽ രീതികളും വിശദമായി വിവരിക്കുക.
3. അരിതമറ്റിക് വിലനൽകൽ ഓപ്പറേറ്ററിന്റെ പ്രവർത്തനം എഴുതുക. ഉദാഹരണ സഹിതം എല്ലാ അരിത്മറ്റിക് വിലനൽകൽ ഓപ്പറേറ്ററുകളും വിവരിക്കുക.

## പദാവലി

| | | |
|---|---|---|
| 1 ന്റെ പൂരകം | : | 1's Complement |
| 2 ന്റെ പൂരകം | : | 2's Complement |
| അംഗീകൃത തത്വങ്ങൾ | : | postulates |
| അനന്യത നിയമം | : | identtiy law |
| അഷ്ടസംഖ്യ | : | octal number |
| അസംബ്ലി ഭാഷ | : | assembly language |
| അസ്ഥിര ദശാംശ സംഖ്യ ലിറ്ററൽ | : | floating point numeric literal/floating point literal |
| അസ്ഥിര പ്രാഥമിക മെമ്മറി | : | volatile primary memory |
| ആധാരം | : | base |
| ആസ്കി | : | ASCII |
| ഇടം നൽകൽ | : | allocation |
| ഇസ്കി | : | ISCII |
| ഉപയോക്തൃ നിർവചിക്കുന്ന | : | user-defined |
| ഏക ഉദ്ധരണി (ഏക സൂചകം) | : | single quote |
| ക്രമ നിയമം | : | commutative law |
| ഗണിതം | : | arithmetic |
| ചിഹ്നവും മൂല്യവും | : | sign and magnitude |
| തലക്കെട്ട് | : | header |
| തനത് | : | default |
| ദശസംഖ്യാ സമ്പ്രദായം | : | decimal number system |
| ദ്വയസംഖ്യ | : | binary number |
| ദ്വൈത സിദ്ധാന്തം | : | principle of duality |
| ദ്വീതീയ സംഭരണം | : | secondary storage |
| നൽകിയ ഇടം തിരികെ എടുക്കൽ | : | de-allocation |

| | | |
|---|---|---|
| നിർദ്ദേശം വ്യാഖ്യാനിക്കുക. | : | command interpretation |
| പദപ്രയോഗം | : | expression |
| പരിവർത്തനം | : | conversion |
| പൂരകം | : | complement |
| പൂരക നിയമം | : | complementary law |
| പൂർണസംഖ്യ | : | integer |
| പ്രതിനിധാനം | : | representation |
| പ്രാഥമിക സംഭരണം | : | primary storage |
| പ്രൊസസ്സ് കൈകാര്യം ചെയ്യുക | : | process management |
| പ്രസ്താവന | : | statement |
| ഫങ്ഷൻ നാമം | : | function name |
| ഫയൽ നാമം | : | file name |
| ഫയൽ കൈകാര്യം ചെയ്യുക | : | file management |
| ഫ്ളോട്ടിങ് പോയിന്റ് നമ്പർ | : | floating point number |
| ഭാഷ പ്രൊസസ്സർ | : | language processor |
| ഭൗതിക ഘടകങ്ങൾ | : | physcial component |
| ബീജഗണിതം | : | algebra |
| മൂലസംഖ്യ | : | radix |
| മെമ്മറി കൈകാര്യം ചെയ്യുക | : | memory management |
| മെമ്മറി സ്ഥാനം | : | memory location |
| ട്രൂത്ത് ടേബിൾ | : | truth table |
| യുക്തി വിചിന്തനം (ലോജിക്കൽ റീസണിംഗ്) | : | logical reasoning |
| യുക്തിപരമായ നിഷേധം | : | logical negation |
| യന്ത്ര ഭാഷ | : | machine language |
| വർഗപൂരക നിയമം | : | involution law |
| വർഗസമ നിയമം | : | idempotent law |
| വിതരണ നിയമം | : | distributive law |

| | | |
|---|---|---|
| വേരിയബിൾ | : | variable |
| ശ്രേണി | : | sequence |
| സംയോജന നിയമം | : | associative law |
| സിദ്ധാന്തം | : | theorem |
| സ്ഥാന വില | : | weight |
| സ്ഥാനീയ സംഖ്യ | : | positional number |
| സ്വാംശീകരണ നിയമം | : | absorption law |
| സ്വതന്ത്ര ഓപ്പൺ സോഴ്സ് | : | free and open source |
| സുസ്ഥിര ദ്വിതീയ മെമ്മറി | : | non-volatile secondary memory |