6

H9U7I1

ഡാറ്റ ഇനങ്ങളും ഓപ്പറേറ്ററുകളും

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)

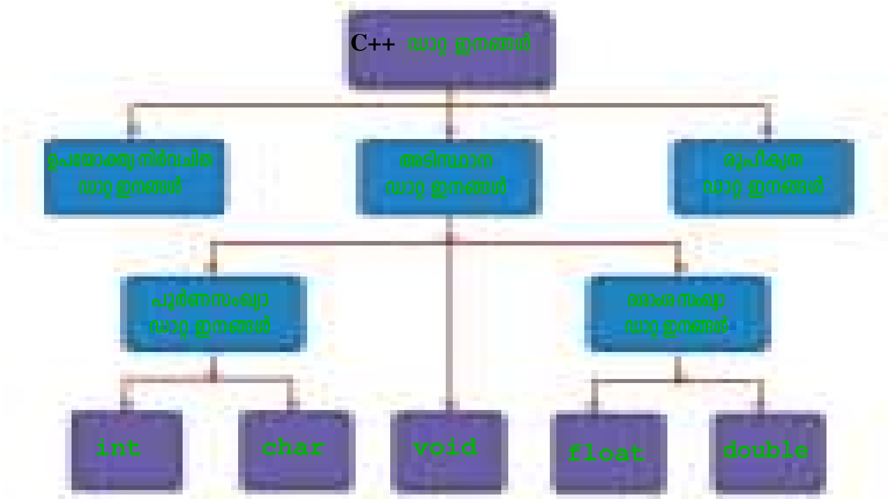
പരീക്ഷയ്ക്കുശേഷം ഒരു വിദ്യാർത്ഥിയുടെ പ്രോഗ്രസ് കാർഡ് തയ്യാറാക്കുന്നതിനായി നമുക്ക് അയാളുടെ രജിസ്റ്റർ നമ്പർ, റോൾ നമ്പർ, പേര്, വിലാസം, വിവിധ വിഷയങ്ങൾക്ക് ലഭിച്ച സ്കോർ, ഗ്രേഡുകൾ തുടങ്ങിയ ഡാറ്റ ആവശ്യമുണ്ട്. ഇത് കൂടാതെ, വിദ്യാർത്ഥിയുടെ സ്കോർ, ഹാജർ എന്നിവ ശതമാനത്തിൽ പ്രദർശിപ്പിക്കേണ്ടതുണ്ട്. ശാസ്ത്ര സംബന്ധിയായ 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 അധ്യായത്തിൽ 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.

അധ്യായം 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 സാധാരണ യായി ദശാംശ സംഖ്യാ ഡാറ്റ ഇനം (floating point data type) എന്ന് അറിയപ്പെടുന്നു. ഇവയെ വ്യാപ്തിയുടെയും (range) കൃത്യതയുടെയും (presion) അടിസ്ഥാനത്തിൽ float, double എന്നിങ്ങനെ രണ്ടായി തരം തിരിച്ചിരിക്കുന്നു.

ഉപോയോക്തൃ നിർവചിത ഡാറ്റ ഇനങ്ങൾ: പ്രോഗ്രാമർക്ക് സ്വന്തമായി ഡാറ്റ ഇനം നിർവചിക്കുവാനുള്ള സൗകര്യം C++ൽ ഉണ്ട്. സ്ട്രക്ചർ (struct), എന്യൂമറേഷൻ (enum), യൂണിയൻ (union), ക്ലാസ്സ് (class) തുടങ്ങിയവ ഇത്തരം ഡാറ്റ ഇനങ്ങൾക്കുള്ള ഉദാഹര ണങ്ങളാണ്.

രൂപീകൃത ഡാറ്റ ഇനങ്ങൾ: അടിസ്ഥാന ഡാറ്റ ഇനങ്ങളെ ഗണങ്ങൾ ആക്കി മാറ്റിയോ വലിപ്പ മാറ്റം വരുത്തിയോ നിർമ്മിക്കപ്പെടുന്ന ഡാറ്റ ഇനങ്ങൾ രൂപീകൃത ഡാറ്റ ഇനങ്ങൾ എന്ന് അറിയപ്പെടുന്നു. അറേകൾ, പോയിന്ററുകൾ, ഫങ്ഷനുകൾ തുടങ്ങിയവ രൂപീ കൃത ഡാറ്റ ഇനങ്ങൾക്കുള്ള ഉദാഹരണങ്ങളാണ്.

6.3 അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ (fundamental data types):

അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ മൗലികമായ സ്വഭാവ വിശേഷമുള്ളവയാണ്. അവയെ വീണ്ടും ചെറിയ ഭാഗങ്ങളായി വിഭജിക്കുവാൻ കഴിയില്ല. കമ്പൈലറിൽ നിർചിക്കപ്പെട്ടിരിക്കുതി നാൽ അവയുടെ വലിപ്പം (അനുവദിക്കപ്പെട്ട മെമ്മറിയുടെ അളവ്) കമ്പൈലറിനെ ആശ്ര യിച്ചിരിക്കുന്നു. നാം ഉപയോഗിക്കുന്നത് GCC -ൽ ലഭ്യമായ കമ്പൈലർ ആയതിനാൽ ഡാറ്റയുടെ വലിപ്പവും അതുപോലെ ഡാറ്റയുടെ വ്യാപ്തിയും അതിന് അനുസൃതമായി രിക്കും. ടർബോ C++ IDE പോലുള്ള കംപയിലറുകൾ നിങ്ങൾ ഉപയോഗിക്കുമ്പോൾ ഇത് വ്യത്യസ്തമാകാം. അഞ്ചു അടിസ്ഥാന ഡാറ്റ ഇനങ്ങൾ താഴെ വിശദീകരിച്ചിരിക്കുന്നു.

int ഡാറ്റഇനം (പൂർണ സംഖ്യകൾക്കായി): ദശാംശ ഭാഗമില്ലാത്ത പൂർണ്ണ സംഖ്യക ളാണ് ഇന്റിജറുകൾ. ഇവ പോസിറ്റീവോ, പൂജ്യമോ, നെഗറ്റീവോ ആകാം. ഒരു പ്രത്യേക പരിധിക്കുള്ളിലുള്ള ഇന്റിജറുകളെ പ്രതിനിധീകരിക്കാൻ ഉപയോഗിക്കുന്ന കീ വേർഡ് ആണ് int. int ഇനത്തിലുള്ള ഇന്റിജറുകൾക്ക് GCC നാല് ബൈറ്റ് മെമ്മറി അനുവദിക്കു ന്നു. 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 അനുവദനീയമാ യ 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 ഡാറ്റഇന മായി പരിഗണിക്കപ്പെടുന്നു. Char കീ വേർഡ് C++ ലെ ക്യാരക്ടർ ലിറ്ററലുകളെ പ്രതിനി ധീകരിക്കുന്നു 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 0.4728197×10^5 എന്നാണ്. ഇതിന്റെ ആദ്യഭാഗത്തെ (0.4728197) ഭിന്നാംശം (mantissa) എന്നും പത്തിന്റെ വർഗ്ഗമായ 5നെ (10^5) കൃത്യാങ്കം (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 പേര്	Description വിശദീകരണം	Size വലിപ്പം	Range പരിധി
char	ക്യാരക്ടർ	1 ബൈറ്റ്	signed: -128 to 127 unsigned: 0 to 255
short int (short)	ഷോർട്ട് ഇന്റിജർ	2 ബൈറ്റ്സ്	signed: -32768 to 32767 unsigned: 0 to 65535
int	ഇന്റിജർ	4 ബൈറ്റ്സ്	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	ലോംഗ് ഇന്റിജർ	4 ബൈറ്റ്സ്	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
float	ഫ്ളോട്ടിംഗ് പോയിന്റ് നമ്പർ	4 ബൈറ്റ്സ്	$-3.4\times10^{+/-38}$ to $+3.4 \times 10^{+/-38}$ ഏകദേശം 7 ദശാംശ സ്ഥാനങ്ങൾ
double	ഡബിൾ പ്രിസിഷനൻ ഫ്ളോട്ടിംഗ് പോയിന്റ് നമ്പർ	8 ബൈറ്റ്സ്	$-1.7 \times 10^{+/-308}$ to $+1.7 \times 10^{+/-308}$ ഏകദേശം 15 ദശാംശ സ്ഥാനങ്ങൾ
long double	ലോംഗ് ഡബിൾ പ്രിസിഷനൻ ഫ്ളോട്ടിംഗ് പോയിന്റ് നമ്പർ	12 ബൈറ്റ്സ്	$-3.4 \times 10^{+/-4932}$ to $+3.4 \times10^{+/-4932}$ ഏകദേശം 19 ദശാംശ സ്ഥാനങ്ങൾ

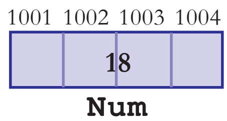
പട്ടിക 6.1: ഡാറ്റ ഇനങ്ങളും ടൈപ്പ് മോഡിഫയറുകളും

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. ഓപ്പറേറ്ററുകൾ (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 ഓപ്പറാൻഡ് കളുടെ എണ്ണം അനുസരിച്ച് ഓപ്പറേറ്ററുകളെ യൂനറി (unary), ബൈനറി (binary), ടെറിനറി (terinary) എങ്ങനെ 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 യുനറി ഓപ്പറേറ്റർ നാം പ്രയോഗിച്ചാൽ സംഖ്യയുടെ നിലവിലുള്ള ചിഹ്നം നേരേ വിപരീതമാകുന്നു. യുനറി 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 യുനറി 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 യുനറി ഓപ്പറേറ്ററുകളായും 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

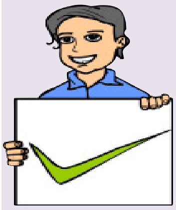
ചിത്രം 6.3 അരിത്മെറ്റിക് ഓപ്പറേറ്ററുകൾ

മോഡുലസ് ഓപ്പറേറ്ററുകൾ (Modulus operator (%)): മോഡ് അഥവാ മോഡുലസ് ഓപ്പറേറ്റർ ഹരണത്തിനുശേഷമുള്ള ശിഷ്ടം കണ്ടുപിടിക്കാൻ ഉപയോഗിക്കുന്നു. ഇത് ഇന്റിജർ ഓപ്പറാൻഡ്യുകളോടൊപ്പം മാത്രമേ ഉപയോഗിക്കാൻ കഴിയൂ. 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
   b. -y
   c. -x + -y
   d. -x – y
   e. x % -11
   f. x + y
   g. x % y
   h. x / y
   i. x * -y
   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 (=) ആവശ്യാമാണെന്നത് 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.

ഉദാഹരണം ! (100<2) ഫലം 1

! (100>2) ഫലം 0 (False)

E1	!E1
0	1
1	0

പട്ടിക 6.8നോട്ട് ഓപ്പറേറ്ററിന്റെ ഉപയോഗം

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 വില നൽകൽ 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

ഓപ്പററ്ററുകളാണ്. C++ ചുരുക്കെഴുത്തുകൾ (short hands) എന്നുകൂടി ഇവ അറിയപ്പെടുന്നു. ഇവയെല്ലാം തന്നെ ബൈനറി ഓപ്പറേറ്ററുക ളാണ്. ഇവയുടെ ഒന്നാമത്തെ ഓപ്പറാന്റ് എപ്പോഴും ഒരു വേരിയബിൾ തന്നെയായിരി ക്കണം. സങ്കലനം, വിലനൽകൽ എന്നീ പ്രവർത്തനങ്ങൾ സാധാരണ രീതിയേക്കാൾ വേഗത്തിൽ ചെയ്യുന്നതിനായി ഈ ഓപ്പറേറ്റ റുകൾ 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.സോ ഴ്സ് കോഡ് സംക്ഷിപ്തമാക്കാൻ ഈ ഓപ്പറേറ്ററുകൾ സഹായിക്കും.

ഇൻക്രിമന്റ് ഓപ്പറേറ്റർ: (++) ഒരു ഇന്റിജർ വേരിയബിളിന്റെ ഉള്ളടക്കത്തെ ഒന്നു വർദ്ധി പ്പിക്കാൻ 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 ഉപ യോഗിക്കാം. ഇത് x=x-1 അല്ലെങ്കിൽ x-=1 എന്നതിന് തുല്യമാണ്.

ഈ ഓപ്പറേറ്ററുകളുടെ രണ്ട് രീതിയിലുള്ള ഉപയോഗങ്ങളെ ഇൻക്രിമെന്റ്/ഡിക്രിമെന്റ് പ്രവർത്തനങ്ങളുടെ പ്രിഫിക്സ് രൂപമെന്നും, പോസ്റ്റ്ഫിക്സ് രൂപമെന്നും എന്ന് വിളി ക്കുന്നു.

രണ്ടു രീതികളും ഓപ്പറാന്റിൽ ഒരേ മാറ്റമാണ് വരുത്തുന്നത് എങ്കിലും മറ്റ് ഓപ്പറേറ്ററുക ളുടെ കൂടെ ഉപയോഗിക്കുമ്പോൾ ഇവയുടെ പ്രവർത്തന രീതി വ്യത്യസ്തമായിരിക്കും.

ഇൻക്രിമന്റ്/ ഡിക്രിമന്റ് ഓപ്പറേറ്ററുകളുടെ പ്രീഫിക്സ് രൂപം: പ്രീഫിക്സ് രീതിയിൽ ഓപ്പറേറ്റർ ഓപ്പറാന്റിന്റെ മുൻപായിരിക്കും എഴുതുക. ഇവിടെ ഇൻക്രിമെന്റ് / ഡിക്രി മെന്റ് ആദ്യം ചെയ്യുകയും അങ്ങനെ കിട്ടുന്ന മൂല്യം മറ്റ് ഓപ്പറേഷനുകൾക്ക് ഉപയോഗി ക്കുകയും 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 കൂടി യ 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) എന്ന് 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 കൂട്ടിച്ചേർക്കൂ. 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 ഓപ്പറാന്ററുകളായി 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. വിവിധ തരം 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==5ǁy==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 റിലേഷണ്‍ 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 ചെറിയതിനെ 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 ഇടതു വശത്തായി 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 വാല്യു 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 കംപൈൽ സമയത്ത് മെമ്മറിയിൽ സംഭരിക്കപ്പെടുന്നു. ഇതിനായി അസ്സൈൻമെന്റ് ഓപ്പറേറ്റർ ഉപയോഗിക്കുന്നു. താഴെ കൊടുത്തിരിക്കുന്നതു പോലെ രണ്ടു രീതിയിൽ ഇത് ചെയ്യാവുന്നതാണ്.

Data type variable = value

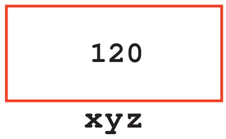
അല്ലെങ്കിൽ

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 കീ വേഡ് 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 സ്ഥിരമായിരിക്കാം.

```
const float pi=3.14;
```

ഇതിന്റെ വില പിന്നീട് മാറ്റം വരുത്താൻ സാധ്യമല്ല. വേരിയബിളിൽ മൂല്യങ്ങൾ രേഖപ്പെടുത്താനും തിരിച്ചെടുക്കാനുമുള്ള 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. x ന്റെ 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 ദശാംശസംഖ്യയായി x ൽ സംഭരിക്കുന്നതിന് ടൈപ്പ് കാസ്റ്റിങ് ഉപയോഗിച്ച് താഴെപ്പറയുന്ന രീതിയിൽ പ്രസ്താവന മാറ്റി എഴുതാവുന്നതാണ്.

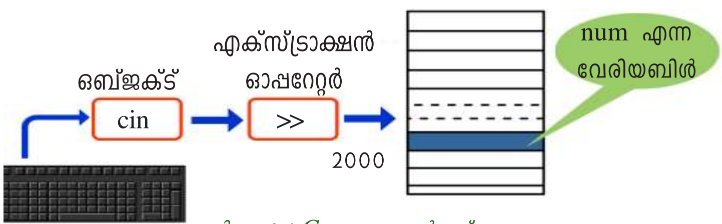
x=(float)p/q; or x=p/(float)q;

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. ഉദാഹരണത്തിന് താഴെ പറയുന്ന പ്രസ്ഥാവന കീബോർഡിൽ നിന്ന് ഡാറ്റാ സ്വീകരിക്കുകയും Num എന്ന വേരിയബിളിൽ സൂക്ഷിക്കുകയും ചെയ്യുന്നു.

```
cin>>num;
```

ഡാറ്റ കീ ബോർഡിൽ നിന്നും സ്വീകരിച്ച് എങ്ങനെ വേരിയബിളിൽ സംഭരിക്കുന്നു എന്ന് ചിത്രം 6.4. ൽ കാണിച്ചിരിക്കുന്നു.

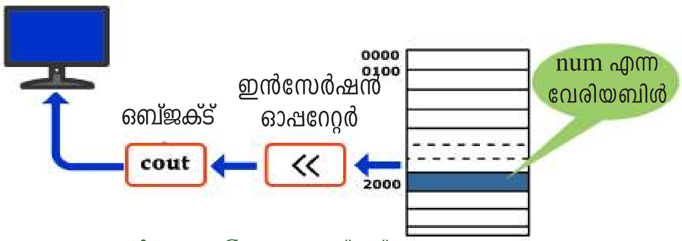
6.9.4 ഔട്ട്പുട്ട് പ്രസ്താവനകൾ (Output statements)

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

ഒന്നാമത്തെ പ്രസ്താവന num ന്റെ വില മോണിറ്ററിൽ പ്രദർശിപ്പിക്കുന്നു. രണ്ടാമത്തേത് hello friends എന്ന സ്ട്രിംഗ് പ്രദർശിപ്പിക്കുന്നു. അവസാനത്തേതിൽ num നോടുകൂടി 12 കൂട്ടി കിട്ടുന്ന 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 എളുപ്പവുമായ 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 നെയിംസ്പേസ് 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, ഒബ്ജക്റ്റുകൾ 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 നിർവ ചിത ഒബ്ജക്ടുകളായ cin, cout എന്നിവ നമുക്ക് ഉപയോഗിക്കേണ്ടതായി വരുമ്പോൾ പ്രോഗ്രാമിന്റെ തുടക്കത്തിൽ താഴെ പറയുന്ന പ്രസ്താവന നമ്മൾ ഉപയോഗിക്കണം.

#include <iostream> എന്ന ഹെഡർ ഫയലിൽ cin, cout എന്നീ ഒബ്ജക്ടുകളുടെ വിവര ങ്ങൾ അടങ്ങിയിരിക്കും. ഹെഡർ ഫയലുകൾക്ക് h എക്സ്റ്റൻഷൻ ഉണ്ടെങ്കിലും GCC ൽ അതു ഉപയോഗിക്കരുത്. പക്ഷേ ടർബോ C++ IDE പോലെയുള്ള മറ്റു ചില കംപൈല റുകൾക്ക് ഈ എക്സ്റ്റൻഷൻ നിർബന്ധമാണ്.

6.10.3. നെയിംസ്പേസ് എന്ന ആശയം (Concept of namespace)

ഒരു പ്രോഗ്രാമിൽ ഒരേ വ്യാപ്തിയിൽ ഒരേ പേരിലുള്ള ഒന്നിലധികം ഐഡന്റി ഫയറു കൾ (വേരിയബിളുകൾ അല്ലെങ്കിൽ ഫംങ്ഷനുകൾ) ഉണ്ടായിരിക്കാൻ പാടില്ല. ഉദാഹര ണത്തിന് നമ്മുടെ വീട്ടിൽ രണ്ടോ അതിലധികമോ ആളുകൾക്ക് (അല്ലെങ്കിൽ ജീവജാല ങ്ങൾക്ക്) ഒരേ പേരുണ്ടാവില്ല. അങ്ങനെയുണ്ടെങ്കിൽ തീർച്ചയായും വീട്ടിനുള്ളിൽ അവരെ പേരു കൊണ്ട് തിരിച്ചറിയുക എന്നത് വിഷമകരമാകും. അതുകൊണ്ട് നമ്മുടെ വീട്ടിൽ ഓരോ പേരും അനന്യമായിരിക്കണം. എന്നാൽ നമ്മുടെ അയൽപക്കത്തെ വീട്ടിൽ സമാ നമായ പേരുള്ള ഒരാൾ (അല്ലെങ്കിൽ ജീവജാലം) ഉണ്ടായിരിക്കാം. അതാത് പരിധിക്കു ള്ളിൽ വ്യക്തികളെ പേരു കൊണ്ട് തിരിച്ചറിയുന്നതിന് ഇത് യാതൊരു ആശയക്കുഴപ്പവു മുണ്ടാക്കില്ല. 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 വ്യത്യ സ്തമായിരിക്കുന്ന ഒരു ഗണത്തിന്റെ പേരാണിത്. വേരിയബിളുകൾക്കും ഫംഗ്ഷനുകൾക്കു മായി പ്രത്യേകം നെയിം സ്പേസുകൾ സൃഷ്ടിക്കുന്നതിന് ഉപയോക്താവിനു അനുവാ ദമുണ്ട്. ഒരു നെയിംസ്പേസിനു പേരു കൊടുക്കാൻ നമുക്ക് ഒരു ഐഡന്റിഫയർ ഉപ യോഗിക്കാം. പ്രോഗ്രാമിംഗിൽ ഉപയോഗിക്കുന്ന ഘടകങ്ങളെ ഏത് നെയിം സ്പേസിൽ തിരയണമെന്ന് using എന്ന കീവേർഡ് സാങ്കേതികമായി കംപൈലറിനോട് പറയുന്നു. C++ ൽ standard എന്നതിന്റെ ചുരുക്കെഴുത്താണ് std. cin, cout തുടങ്ങി മറ്റ് പല ഒബ്ജ ക്ടുകളും നിർവചിച്ചിട്ടുള്ള ഒരു നെയിം സ്പേസ് ആണിത്. അതിനാൽ ഒരു പ്രോഗ്രാ മിൽ ഇവ ഉപയോഗിക്കണമെങ്കിൽ std::cin, std::cout എന്ന മാതൃക നാം പിന്തുടരേണ്ടതാ ണ്. using name space std എന്ന പ്രസ്താവന പ്രോഗ്രാമിൽ ഉപയോഗിക്കുന്നതിലൂടെ ഇത്തരത്തിലുള്ള വിശദമായ പരാമർശങ്ങൾ ഒഴിവാക്കാവുന്നതാണ്. അത്തരമൊരു സാഹ ചര്യത്തിൽ കംപൈലർ cin, cout, endl മുതലായവയ്ക്കായി ഈ നെയിംസേപസിൽ തിര യുന്നു. cin, cout, endl അല്ലെങ്കിൽ അതുപോലെയുള്ളവ എപ്പോഴൊക്കെ ഒരു C++ പ്രോഗ്രാ മിൽ കമ്പ്യൂട്ടർ കാണുന്നുവോ, അവയെ std::cin, std::cout, std::endl എന്നിങ്ങനെ വ്യഖ്യാ നിക്കുന്നു.

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 ഐഡന്റിഫയറിനെ 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 ന്യുമോണിക് പേരുകൾ (mnemonic names) എന്നു വിളിക്കുന്നു. പേരുകൾ സ്വീകരിക്കുമ്പോൾ താഴെപ്പറയുന്ന കാര്യങ്ങൾ ശ്രദ്ധിക്കണം.

1. വേരിയബിളുകൾ, ഫങ്ഷനുകൾ, പ്രൊസീഡറുകൾ എന്നിവക്ക് നല്ല ന്യുമോണിക് 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. കംപൈലറുകൾ 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

ഇൻഡന്റേഷൻ

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

Sum of the entered numbers = 12

ഉപയോക്താവിന്റെ ഇൻപുട്ടുകൾ സ്പേസ് ഉപയോഗിച്ച്

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 സ്കോറുകളായി 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 ഓപ്പറേറ്ററിന്റെ (=) 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 / ഓപ്പറേറ്ററുടെ 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 കണക്കുകൂട്ടുവാൻ നിങ്ങളോട് അഭ്യർഥിക്കുന്നു എന്നിരിക്കട്ടെ. നമുക്കറിയാവുന്നതുപോലെ വൃത്തത്തിന്റെ വിസ്തീർണം r^2 എന്ന സമവാക്യം ഉപയോഗിച്ചും, ചുറ്റളവ് 2 r ഉപയോഗിച്ചും (ഇവിടെ =3.14) മാണ് കണക്കാക്കുന്നത്. പ്രോഗ്രാം 6.4 ഈ പ്രശ്നം പരിഹരിക്കുന്നു.

π

പ്രോഗ്രാം 6.4: തന്നിരിക്കുന്ന ആരത്തിന് അനുസരിച്ച് ഒരു വൃത്തത്തിന്റെ വിസ്തീർണവും ചുറ്റളവും കണ്ടുപിടിക്കാൻ വേണ്ടി.

```
#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 വരി യിലേക്ക് വരുത്തുന്നു.

വെറും പലിശ കണക്കാക്കുന്നതിന് മറ്റൊരു പ്രോഗ്രാം നമുക്ക് എഴുതി നോക്കാം. നമുക്ക റിയാവുന്നതു പോലെ ഫലം കണക്കാക്കുന്നതിനായി മൂലധനം, പലിശയുടെ ശതമാ നം, 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 അർദ്ധ വിരാമം ഇല്ല എന്നതു പ്രത്യേകം ശ്രദ്ധിക്കുക. അതിനാൽ ഇത് ഒറ്റ പ്രസ്താവനയായി ട്ടാണ് 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 ഉദ്ദേശിച്ച 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/0 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 ഒട്ട്പുട്ട് 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;`	(v)	`cin >> "\n" >> y ;`
(ii)	`m=5,n=12;015`	(vi)	`cout >> \n "abc"`
(iii)	`cout << "x" ; <<x;`	(vii)	`a = b + c`
(iv)	`cin >> y`	(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. ഉദാഹരണസഹിതം 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
ദ്വിതീയ 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