6

പ്രധാന ആശയങ്ങൾ

- C++ പ്രോഗ്രാമിന്റെ ഘടന
  - പ്രിപ്രൊസ്സർ നിർദ്ദേശം
  - ഹെഡർഫയലുകൾ
  - വേരിയബിളുകൾ
  - C++ മാതൃക പ്രോഗ്രാം
- പ്രോഗ്രാം എഴുതുന്നതിനുള്ള മാർഗ്ഗ നിർദ്ദേശങ്ങൾ
- വേരിയബിളിന് പ്രാരംഭ വില നൽകൽ
- const ആക്സസ് മോഡിഫയറിന്റെ ഉപയോഗം
- ടൈപ്പ് മോഡിഫയറുകൾ
- കൂടുതൽ ഓപ്പറേറ്ററുകൾ
  - അരിത്മെറ്റിക് അസൈൻമെന്റ്
  - ഇൻക്രിമെന്റ്, ഡിക്രിമെന്റ്
  - ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം
- ഇനം മാറ്റൽ (type conversion)
  - ആന്തരിക ഇനം മാറ്റൽ (implicit conversion)
  - ബാഹ്യ ഇനം മാറ്റൽ (explicit conversion)

R2R6J1

പ്രോഗ്രാമിങ് – ഒരു ആമുഖം

C++ പ്രോഗ്രാം എഴുതുവാനുള്ള IDE നമ്മൾ പരിചയപ്പെട്ടു കഴിഞ്ഞു. എങ്ങനെയാണ് ഓപ്പറേഷനുകളെ C++ ൽ പ്രതിനിധാനം ചെയ്യുന്നത് എന്ന് കഴിഞ്ഞ അധ്യായത്തിൽ ചർച്ച ചെയ്തു. കൂടാതെ കമ്പ്യൂട്ടറിന് നിർദ്ദേശങ്ങൾ നൽകുന്നതിന് ആവശ്യമായ വിവിധതരം C++ പ്രസ്താവനകളും നാം പഠിച്ചു കഴിഞ്ഞു. ഇത്തരം പ്രസ്താവനകൾ ഉപയോഗിച്ച് ലളിതമായ പ്രശ്നപരിഹാരത്തിനുള്ള പ്രയോഗങ്ങൾ എഴുതുവാൻ നമുക്കിപ്പോൾ സാധ്യമാണ്. എന്നാൽ ഒരു കൂട്ടം പ്രസ്താവനകൾ മാത്രം കൂട്ടിച്ചേർത്താൽ ഒരു പ്രോഗ്രാമായി മാറുന്നില്ല. ഒരു C++ പ്രോഗ്രാമിന് ഒരു പ്രത്യേക ഘടനയുണ്ട്. ഈ അധ്യായത്തിൽ C++ പ്രോഗ്രാമിന്റെ ഘടനയും, അതിനനുസരിച്ച് പ്രോഗ്രാം എഴുതുന്ന രീതിയും ചർച്ച ചെയ്യുന്നു. പ്രോഗ്രാമുകൾ ഒതുക്കത്തോടെ എഴുതി വേഗത്തിൽ പ്രവർത്തിപ്പിക്കുന്നതിനാവശ്യമായ കൂടുതൽ ഓപ്പറേറ്ററുകളും ഇവിടെ പരിചയപ്പെടുത്തുന്നു.

6.1 C++ പ്രോഗ്രാമിന്റെ ഘടന:

നാം ഇതുവരെ ചർച്ച ചെയ്ത പ്രസ്താവനകൾ ഉപയോഗിച്ച് ലഘുവായ പ്രശ്നങ്ങൾ പരിഹരിക്കാവുന്ന സ്ഥിതിയിൽ നാം ഇപ്പോൾ എത്തി കഴിഞ്ഞു. എന്നാൽ ഒരു കൂട്ടം പ്രസ്താവനകൾ 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 ഫങ്ക്ഷനുകളും വ്യത്യസ്ത പേരുകളിൽ ആയിരിക്കണം തിരിച്ചറിയപ്പെടേണ്ടത്. എല്ലാ പ്രോഗ്രാമിലും 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.

6.1.1 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.

6.1.2 ഹെഡർ ഫയലുകൾ (Header files)

ഫംഗ്ഷനുകൾ, ഒബ്ജക്ടുകൾ മുൻനിർവചിത - രൂപീകൃത ഡാറ്റഇനങ്ങൾ എന്നിവയെക്കുറിച്ചുള്ള വിവരങ്ങൾ കംപൈലറിനോടൊപ്പം 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.

6.1.3. 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 നെയിംസ്പേസുമായി 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.

6.1.4 `main()` ഫങ്ഷൻ (The `main ()` function)

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.

6.1.5 ഒരു മാതൃക പ്രോഗ്രാം (A sample program)

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.

6.2 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. കംപൈലറുകൾ അവയെ അവഗണിക്കുന്നു. C++ ൽ കമന്റുകൾ എഴുതുന്നതിന് രണ്ടു മാർഗ്ഗങ്ങളുണ്ട്.

ഒറ്റവരി കമന്റ്: '//' ചിഹ്നങ്ങളാണ് ഒറ്റവരി കമന്റുകൾ എഴുതാനായി ഉപയോഗിക്കുന്നത്. ഒരു വരിയിൽ // ന് ശേഷമുള്ള വാക്യങ്ങൾ കമന്റുകളായി C++ കംപൈലർ കണക്കാക്കുന്നു.

ഖണ്ഡിക കമന്റ് (multiline comment): /* നും */ നും ഇടയിൽ എഴുതുന്ന എന്തിനെയും കംപൈലർ 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.

6.3. 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 6.1: 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.

6.4. 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.

6.5 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. അതിനുള്ള 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`	Character	1 byte	`signed`: -128 to 127 `unsigned`: 0 to 255
`short int` (`short`)	Short Integer	2 bytes	`signed`: -32768 to 32767 `unsigned`: 0 to 65535
`int`	Integer	4 bytes	`signed`: -2147483648 to 2147483647 `unsigned`: 0 to 4294967295
`long int` (`long`)	Long integer	4 bytes	`signed`: -2147483648 to 2147483647 `unsigned`: 0 to 4294967295
`float`	Floating point number	4 bytes	$-3.4\times10^{+/-38}$ to $+3.4 \times 10^{+/-38}$ with approximately 7 significant digits
`double`	Double precision floating point number	8 bytes	$-1.7 \times 10^{+/-308}$ to $+1.7 \times 10^{+/-308}$ with approximately 15 significant digits
`long double`	Long double precision floating point number	12 bytes	$-3.4 \times 10^{+/-4932}$ to $+3.4 \times 10^{+/-4932}$ With approximately 19 significant digits

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

6.6 കൂടുതൽ ഓപ്പറേറ്റുകൾ

സാധാരണയായി ഉപയോഗിക്കുന്ന C++ ഓപ്പറേറ്ററുകൾ നമ്മൾ ചർച്ച ചെയ്തു കഴിഞ്ഞു. പ്രോഗ്രാമുകളെ ഒതുക്കമുള്ളതാക്കുന്നതിന് ഉതകുന്ന രീതിയിലുള്ള ചില പ്രത്യേക ഓപ്പറേറ്ററുകൾ C++ൽ ഉണ്ട്. ഇവ രണ്ട് ഓപ്പറേഷനുകളെ കൂട്ടിയോജിപ്പിക്കുന്നു. അസൈൻമെന്റ് ഓപ്പറേഷനും അരിത്തമെറ്റിക് ഓപ്പറേഷനും കൂട്ടിയോജിപ്പിക്കുന്നു. ഏതാനും ചിലതു മാത്രം നമ്മൾ ഇവിടെ ചർച്ച ചെയ്യുന്നു. അരിത്തമെറ്റിക് അസൈൻമെന്റ്, ഇൻക്രിമെന്റ്, ഡിക്രിമെന്റ് എന്നീ ഓപ്പറേറ്ററുകൾ ചില ഉദാഹരണങ്ങളാണ്.

6.6.1 അരിത്മെറ്റിക് അസൈൻമെന്റ് ഓപ്പറേറ്ററുകൾ

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

അരിത്മെറ്റിക് അസൈൻമെന്റ് ഓപ്പറേഷൻ	തുല്യമായ അരിത്മെറ്റിക് ഓപ്പറേഷൻ
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 6.2: C++ ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം.

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.

6.6.2 ഇൻക്രിമന്റ് (++) ഡിക്രിമന്റ് (--) ഓപ്പറേറ്ററുകൾ

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 തുല്യമാണ്.

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.

6.6.3 ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം

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 6.3 ൽ നൽകിയിരിക്കുന്നു. ഒരു പ്രയോഗത്തിൽ മുൻഗണനാക്രമത്തിൽ ഒരേ സ്ഥാനം വരുന്ന ഓപ്പറേറ്ററുകൾ ഉണ്ടെങ്കിൽ അവ മിക്കവാറും ഇടത്തുനിന്ന് വലത്തേക്ക് എന്ന രീതിയിലാണ് പ്രവർത്തിക്കുക.

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 പ്രയോഗത്തിലെ 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.

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 6.3: ഓപ്പറേറ്ററുകളുടെ മുൻഗണനാക്രമം.

6.7 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.

6.7.1 ആന്തരിക ഇനം മാറ്റൽ (implicit type conversion/ type promotion):

ആന്തരിക ഇനം മാറ്റൽ C++ കമ്പൈലർ ആന്തരികമായി ചെയ്യുന്നതാണ്. വ്യത്യസ്ഥതരം ഡാറ്റ ഉള്ള ഒരു പദപ്രയോഗത്തിൽ C++ കുറഞ്ഞ വലിപ്പത്തിലുള്ള ഓപ്പറാന്റിനെ കൂടുതൽ 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).

6.7.2: 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.

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 സംഭരിക്ക പ്പെടുകയും 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;

പ്രോഗ്രാമുകൾ

പ്രോഗ്രാം 6.1 ഒരു സന്ദേശം കാണിക്കുന്നതിന്

```
    /* This program displays the message
         "Smoking is injurious to health"
         on the monitor */
#include <iostreamh>   // To use the cout object
using namespace std; // To access cout
int main() //program begins here
{     //The following output statement displays a message
      cout << "Smoking is injurious to health";
      return 0;
}     //end of the program
```

Multiline comment

Single line comment

ഇന്റന്റേഷൻ

ഔട്ട്പുട്ട് :

Smoking is injurious to health

ഇന്റന്റേഷൻ ഉപയോഗിച്ചുള്ള കൂടുതൽ ഉദാഹരണങ്ങൾ അധ്യായം 7 ൽ നൽകിയിരിക്കുന്നു.പ്രോഗ്രാം 6.2 രണ്ട് സംഖ്യകൾ ഇൻപുട്ടായി സ്വീകരിച്ച് അവയുടെ തുക പ്രദർശിപ്പിക്കുന്നു. ഈ പ്രോഗ്രാമിൽ അധ്യായം അഞ്ചിൽ പഠിച്ച പ്രസ്താവനകൾ ഉപയോഗിച്ചിരിക്കുന്നു.

നമുക്കു ചെയ്യാം

5.2, 2.7 തുടങ്ങിയ വ്യത്യസ്ത ഇൻപുട്ടുകൾ ഉപയോഗിച്ച് പ്രോഗ്രാം 6.2 റൺ ചെയ്യുക. ശരിയായ ഉത്തരം ലഭിക്കുന്നുണ്ടോ എന്ന് പരിശോധിക്കുക. ഇല്ലെങ്കിൽ കൂട്ടുകാരുമായി ചർച്ച ചെയ്ത് പ്രശ്ന പരിഹാരം കണ്ടെത്തുക.

പ്രോഗ്രാം 6.2: രണ്ട് പൂർണ്ണസംഖ്യകളുടെ തുക കണ്ടുപിടിക്കാൻ

```
#include <iostream>
using namespace std;
int main()
{  //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;
}
```

ഔട്ട്പുട്ട് :

```
Enter two numbers:  5    7
Sum of the entered numbers = 12
```

യൂസർ നൽകുന്ന ഡാറ്റ സ്പേയ്സ് ഉപയോഗിച്ച് വേർതിരിച്ചിരിക്കുന്നു

നമുക്ക് മറ്റൊരു പ്രശ്നം പരിഗണിക്കാം. ഒരു കുട്ടിയ്ക്ക് മൂന്ന് തുടർമൂല്യ പ്രവർത്തനങ്ങളിൽ ലഭിച്ച സ്കോറുകൾ നൽകിയിരിക്കുന്നു. കൂടിയ സ്കോർ 20 ആണ്. ശരാശരി സ്കോർ കണ്ടുപിടിക്കുക.

പ്രോഗ്രാം 6.3 : മൂന്നു CE സ്കോറുകളുടെ ശരാശരി കണ്ടെത്താൻ

```
#include <iostream>
using namespace std;
int 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;
    return 0;
}
```

ഔട്ട്പുട്ട് :

Enter the three CE scores: 17 19 20

Average CE score is: 18.666666

ശരാശരി കാണാനുള്ള എക്സ്പ്രഷൻ അസൈമെന്റ് ഓപ്പറേറ്ററി (=) ന്റെ വലതു ഭാഗത്തു നൽകിയിരിക്കുന്നു. ഈ എക്സ്പ്രഷനിൽ രണ്ട് + ഓപ്പറേറ്ററുകളും ഒരു / ഉണ്ട്. / ഓപ്പറേറ്റർ + ഓപ്പറേറ്ററിനേക്കാൾ മുൻഗണന ഉള്ളതായതിനാൽ തുക കാണുന്ന പ്രവർത്തനം ബ്രാക്കറ്റിനുള്ളിൽ നൽകിയിരിക്കുന്നു. തുക കാണുന്നതിനുള്ള ഓപ്പറന്റുകളെല്ലാം int ഡാറ്റാ തരത്തിലുള്ളതായതിനാൽ ഫലവും ഇന്റിജർ തന്നെയായിരിക്കും. ഈ ഇന്റിജറിനെ മൂന്ന് കൊണ്ട് ഹരിക്കുമ്പോൾ ഫലം വീണ്ടും ഇന്റജർ തന്നെയായിരിക്കും. അതുകൊണ്ട് പ്രോഗ്രാം 6.3 ന്റെ ഔട്ട്പുട്ട് 18 ആയിരിക്കും. ഇത് കൃത്യമല്ല. ആയതിനാൽ ഹരിക്കാൻ വേണ്ടി ഫ്ളോട്ടിംഗ് പോയിന്റ് സംഖ്യയായ 3.0 ഉപയോഗിച്ചിരിക്കുന്നു. അങ്ങനെ ഇന്റിജർ ഫ്ളോട്ടിംഗ് തരമായി ഉയർത്തപ്പെടുകയും കൃത്യമായ ഉത്തരം ലഭിക്കുകയും ചെയ്യുന്നു.

ഒരു വൃത്തത്തിന്റെ ആരം നൽകിയെന്നിരിക്കട്ടെ. എങ്ങനെയാണ് വിസ്തീർണ്ണവും ചുറ്റളവും കണ്ടുപിടിക്കുന്നത്? വിസ്തീർണ്ണം കാണാനുള്ള സൂത്രവാക്യം r^2 ഉം ചുറ്റളവ് കാണാനുള്ള സൂത്രവാക്യം 2 r ആണല്ലോ (=3.14). പ്രോഗ്രാം 6.4 ഈ പ്രശ്നം നിർദ്ധാരണം ചെയ്യുന്നു.

പ്രോഗ്രാം 6.4 : ഒരു വൃത്തത്തിന്റെ വിസ്തീർണവും ചുറ്റളവും കണ്ടെത്താൻ

```
#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;
}
```

Escape sequence '\n' prints a new line after displaying the value of Area

ഔട്ട്പുട്ട് :

```
Enter the radius of the circle: 2.5
Area of the circle = 19.642857
Perimeter of the circle = 15.714285
```

പ്രോഗ്രാം 6.4 ന്റെ അവസാനത്തെ രണ്ട് ഔട്ട്പുട്ടുകൾ വ്യത്യസ്ത ലൈനുകളിൽ പ്രദർശിപ്പിക്കുന്നു. ഇതിനു കാരണം എസ്കേപ്പ് സിക്വൻസ് ക്യാരക്ടറായ '\n' ഉപയോഗിച്ചതാണ്.

അടുത്തതായി നമുക്ക് സാധാരണ പലിശ കാണുന്നതിനുള്ള ഒരു പ്രോഗ്രാം തയ്യാറാക്കാം. പ്രിൻസിപ്പൽ എമൗണ്ട്, റെയ്റ്റ് ഓഫ് ഇൻട്രസ്റ്റ്, പീരിഡ് എന്നിവ ഇതിലെ ഇൻപുട്ടുകളാണെന്ന് നിങ്ങൾക്കറിയാമല്ലോ.

പ്രോഗ്രാം 6.5 : സാധാരണ പലിശ കണ്ടെത്താൻ

```
#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;
}
```

ഔട്ട്പുട്ട് :

```
Enter the principal amount in Rupees: 100
Enter the number of years for the deposit: 2
Enter the rate of interest in percentage: 10
```

പ്രോഗ്രാം 6.5 ലെ അവസാനത്തെ പ്രസ്താവന അഞ്ച് വരികളിലേക്ക് പടർന്നിരിക്കുന്നു. ഇത് ഒരു പ്രസ്താവനയാണ്. ഇതിനിടയിൽ സെമികോളൻ ഉപയോഗിച്ചിട്ടില്ലായെന്ന് ശ്രദ്ധിക്കുക. പ്രോഗ്രാം പ്രവർത്തിക്കുമ്പോൾ ഫലം മോണിട്ടറിന്റെ റെസലൂഷൻ അനുസരിച്ച് രണ്ട് വ്യത്യസ്ത വരികളിൽ പ്രദർശിപ്പിക്കപ്പെടാൻ സാധ്യതയുണ്ട്.

പ്രോഗ്രാം 6.6 ഒരു കൺവേർഷൻ പ്രശ്നമാണ്. ഇവിടെ ഡിഗ്രി സെൽഷ്യസിലുള്ള താപം ഫാരൻ ഹീറ്റിലേക്ക് മാറ്റുന്നു.

പ്രോഗ്രാം 6.6 : സെൽഷ്യസിലുള്ള താപം ഫാരൻ ഹീറ്റിലേക്ക് മാറ്റാൻ

```
#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;
}
```

ഔട്ട്പുട്ട് :

```
Enter the Temperature in Celsius: 37
37 Degree Celsius = 98.599998 Degree Fahrenheit
```

നമുക്കറിയാം C++ ലെ ഓരോ ക്യാരക്ടറിനും തത്തുല്യമായ ASCII കോഡ് ഉണ്ട്. ഇത് ഇന്റിജർ കോഡാണ്. തന്നിരിക്കുന്ന ക്യാരക്ടറിന്റെ ASCII കോഡ് കണ്ടുപിടിക്കാൻ നമുക്ക് ഒരു പ്രോഗ്രാം എഴുതാം.

പ്രോഗ്രാം 6.7 : ക്യാരക്ടറിന്റെ ASCII കോഡ് കണ്ടെത്താൻ

```
#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;
}
```

ഔട്ട്പുട്ട് :

```
Enter the character: A
ASCII value of A = 65
```

നമുക്ക് സംഗ്രഹിക്കാം

C++ പ്രോഗ്രാമിൽ കൃത്യമായ ഒരു ഘടനയുണ്ട്. പ്രോഗ്രാം ആകർഷകമാക്കാനും ആശയവിനിമയം ഉറപ്പു വരുത്താനും കൃത്യമായ മാർഗ്ഗ നിർദ്ദേശങ്ങൾ പാലിക്കപ്പെടേണ്ടതായിട്ടുണ്ട്. വേരിയബിളുകൾ പ്രഖ്യാപിക്കുമ്പോൾ തന്നെ അതിനു പ്രാരംഭ വില നൽകേതാണ്ടിയിട്ടുണ്ട്. 'const' ആക്സസ് മോഡിഫയർ ഉപയോഗിച്ചാൽ വേരിയബിളിന്റെ വാല്യു നമ്മൾക്ക് പ്രോഗ്രാമിലൂടെ മാറ്റാൻ സാധിക്കില്ല. കൂടുതൽ റെഞ്ചിലുള്ള ഡാറ്റ കൈകാര്യം ചെയ്യുമ്പോൾ ടൈപ്പ് മോഡിഫയറുകൾ സഹായകമാകുന്നു. ടൈപ്പ് മോഡിഫയറുകൾ പ്രഖ്യാപന സമയത്ത് ഡാറ്റ ഇനത്തിന്റെ കൂടെ ഉപയോഗിക്കുന്നു. C++ ൽ അരിത്തമെറ്റിക് ഓപ്പറേറ്റർ, അസൈൻമെന്റ് ഓപ്പറേറ്റർ എന്നിവ ഒന്നിച്ചുപയോഗിക്കാനുള്ള സൗകര്യം ഉണ്ട്. അരിത്തമെറ്റിക് ഓപ്പറേഷനുകളിൽ നിന്നും ഉദ്ദേശിച്ച ഫലം ലഭിക്കുന്നതിന് ഇനം മാറ്റൽ രീതികളും C++ ൽ ഉപയോഗിക്കപ്പെടുന്നു.

പഠന നേട്ടങ്ങൾ

ഈ അധ്യായത്തിൽ പൂർത്തികരണത്തോടെ പഠിതാവിന്

- ഡാറ്റ ടൈപ്പ് മോഡിഫയറുകൾ തിരഞ്ഞെടുക്കാൻ സാധിക്കുന്നു.
- വിവിധ ഓപ്പറേറ്ററുകൾ ഉപയോഗിക്കാൻ കഴിയുന്നു.
- I/O ഓപ്പറേഷനുകൾ നടത്താൻ കഴിയുന്നു.
- C++ പ്രോഗ്രാമിന്റെ ഘടന തിരിച്ചറിയുന്നു.
- പ്രോഗ്രാം എഴുതുമ്പോൾ ശ്രദ്ധിക്കേണ്ട മാർഗ്ഗ നിർദ്ദേശങ്ങൾ ലഭിക്കുന്നു.
- C++ IDE ഉപയോഗിച്ച് ചെറിയ പ്രോഗ്രാമുകൾ ചെയ്യാൻ സാധിക്കുന്നു.

ലാബ് പ്രവർത്തനങ്ങൾ

1. ഉപയോക്താവിൽ നിന്ന് ഭാരം ഗ്രാമിൽ സ്വീകരിച്ച് കിലോഗ്രാമിൽ പ്രദർശിപ്പിക്കുന്നതിനുള്ള C++ പ്രോഗ്രാം എഴുതുക.
2. താഴെ കൊടുത്തിരിക്കുന്ന പട്ടിക പ്രദർശിപ്പിക്കുന്നതിനുള്ള പ്രോഗ്രാം എഴുതുക.

2013	100%
2012	99.9%
2011	95.5%
2010	90.81%
2009	85%

(സൂചന : '\n', '\t' എന്നിവ ഉപയോഗിച്ച് ഒറ്റ ഔട്ട് പ്രസ്താവനയിൽ ഉത്തരമെഴുതുക.)

3. നിങ്ങളുടെ ഉയരം മീറ്ററിലും സെന്റീമീറ്ററിലുമായി സ്വീകരിച്ച് അടിയിലും ഇഞ്ചിലുമായി മാറ്റുന്നതിനാവശ്യമായ C++ പ്രോഗ്രാം എഴുതുക.

(1 അടി = 12 ഇഞ്ച്, 1 ഇഞ്ച് = 2.54 സെ.മീ.)

4. ത്രികോണത്തിന്റെ വിസ്തീർണം കാണുവാനുള്ള പ്രോഗ്രാമെഴുതുക.
5. സാധാരണ പലിശ, കൂട്ടുപലിശ എന്നിവ കാണുവാനുള്ള പ്രോഗ്രാം എഴുതുക.
6. തന്നിരിക്കുന്ന ഡിജിന്റെ ASCII കോഡും, ബാക്ക് സ്പേസിന്റെ ASCII കോഡും പ്രദർശിപ്പിക്കുവാനുള്ള പ്രോഗ്രാം എഴുതുക.

(സൂചന: ബാക്ക്സ്പേസിനെ ('\b) ഒരു int വേരിയബിളിൽ സംഭരിക്കുക.)

7. സമയം സെക്കന്റിൽ സ്വീകരിച്ച് hrs:mins:secs എന്ന മാതൃകയിൽ പ്രദർശിപ്പിക്കുന്നതിനുള്ള പ്രോഗ്രാം എഴുതുക.

ഉദാഹരണത്തിന് സമയം 3700 സെക്കന്റുകൾ ആണെങ്കിൽ ഉത്തരം 1hr:1min:40secs എന്നായിരിക്കും.

മാതൃക ചോദ്യങ്ങൾ

ഹ്രസ്വോത്തര ചോദ്യങ്ങൾ

1. വേരിയബിളിന്റെ ഡയനാമിക് ഇനിഷ്യലൈസേഷൻ എന്നാൽ എന്ത്?
2. ടൈപ്പ് പ്രമോഷൻ എന്നാൽ എന്ത്?
3. ടൈപ്പ് കാസ്റ്റിങ്ങ് എന്നാൽ എന്ത്?

ലഘു ഉപന്യാസ ചോദ്യങ്ങൾ

1. ഏതൊക്കെ തരത്തിലുള്ള വേരിയബിൾ പ്രഖ്യാപനങ്ങളാണ് താഴെ തന്നിരിക്കുന്ന പ്രോഗ്രാം ശകലത്തിലുള്ളത്?

```
{ int area, length = 10, wideth = 12, perimeter;
  area = length * width;
  perimeter = 2 * (length + width)
}
```

2. 'area', 'perimeter' എന്നീ വേരിയബിളുകളെ ഡയനാമിക് ഇനിഷ്യലൈസേഷൻ നടത്തുന്ന രീതിയിൽ മുകളിലുള്ള പ്രോഗ്രാം ശകലം പരിഷ്കരിക്കുക.
3. C++ ലെ ടൈപ്പ് മോഡിഫയർ വിശദീകരിക്കുക.
4. എന്തുകൊണ്ടാണ് C++ ൽ ധാരാളം ഡാറ്റ ഇനങ്ങൾ ലഭ്യമായിട്ടുള്ളത്.
5. 'const' എന്ന കീവേർഡിന്റെ ധർമ്മം എന്ത്?
6. പ്രീഇൻക്രിമെന്റ്, പോസ്റ്റ് ഇൻക്രിമെന്റ് എന്നീ ഓപ്പറേഷനുകൾ വിവരിക്കുക.
7. ഇനം മാറ്റൽ രീതികൾ വിവരിക്കുക.
8. iostream എന്ന ഫയൽ പ്രോഗ്രാമിൽ ഉൾപ്പെടുത്തിയിട്ടില്ലെങ്കിൽ എന്ത് സംഭവിക്കും?
9. main( ) ഒരു പ്രോഗ്രാമിൽ ഇല്ലെങ്കിൽ എന്ത് സംഭവിക്കും?
10. താഴെ കൊടുത്ത പ്രോഗ്രാം ശകലത്തിലെ തെറ്റുകൾ കണ്ടെത്തുക.

(i)
```
int main() { cout << "Enter two numbers"
cin >> num >> auto
float area = Length * breadth ; }
```

(ii)
```
#include <iostream>
using namespace;
void Main()
{ int a, b
cin <<a <<b
max=a % b
cout > max
}
```

11. പ്രോഗ്രാമിൽ കമന്റുകളുടെ ഉപയോഗം എന്ത്?

ഉപന്യാസ ചോദ്യങ്ങൾ

1. C++ ലെ വിവിധതരം പ്രയോഗങ്ങൾ വിവരിക്കുക.
2. അസൈൻമെന്റ് ഓപ്പറേറ്ററിന്റെ പ്രവർത്തനം എഴുതുക. അരിത്തമെറ്റിക് അസൈൻമെന്റ് ഓപ്പറേറ്റുകൾ ഉദാഹരണസഹിതം വിശദമാക്കുക.