**OOP in C++ : အပိုင္း (၆) - Data types Conversion**

ယခင္ သင္ခန္းစာမ်ားမွာ data types မ်ား အသံုးျပဳပံုကို အေသးစိတ္ ရွင္းျပခဲ႔ပါတယ္။ ယခု သင္ခန္းစာမွာေတာ႔ အဲဒီ data types မ်ားကို တစ္ခုကေန အျခားတစ္ခုသို႔ ေျပာင္းလဲနည္းမ်ား အေၾကာင္း ေဆြးေႏြး တင္ျပသြားမွာ ျဖစ္ပါတယ္။

C language မွာလိုပဲ C++ မွာ မတူညီတဲ႔ data type ေတြ တြဲဘက္ အသံုးျပဳခြင္႔ ေပးထားပါတယ္။ ဥပမာအေနနဲ႔ ေအာက္ပါ mixed.cpp ပရိုဂရမ္ေလးကို ေလ႔လာၾကည္႔ရအာင္။

// mixed.cpp

// shows mixed expressions

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

int count = 7;

float avgWeight = 155.5F;

double totalWeight = count \* avgWeight;

cout << "totalWeight = " << totalWeight << endl;

\_getch();

return 0;

}

ဒီ ဥပမာမွာ double data type (totalWeight) ရရွိဖို႔အတြက္ int data type (count) ကို float data type (avgWeight) နဲ႔ ေျမွာက္ထားတာ ေတြ႕ရမွာ ျဖစ္ပါတယ္။ ကၽြန္ေတာ္တို႔အေနနဲ႔ RUN ၾကည္႔တဲ႔ အခါမွာ Error မျပဘဲ အလုပ္လုပ္တာကို ေတြ႕ရပါတယ္။ compiler အေနနဲ႔ အဲဒီလို မတူညီတဲ႔ data types ေတြ အခ်င္းခ်င္း ေပါင္း၊ ႏုတ္၊ ေျမွာက္၊ စား အစရွိတဲ႔ arithmetic operation ျပဳလုပ္တာေတြကို လက္ခံပါတယ္။

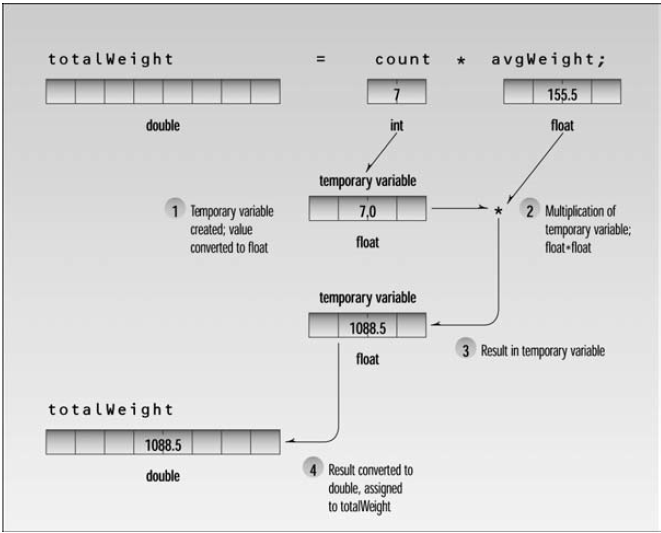
အခ်ိဳ႕ ပရိုဂရမ္းမင္း ဘာသာစကားေတြမွာ ေရးသားရတာ ဒီေလာက္ မလြတ္လပ္ပါဘူး။ အခုလို မတူညီတဲ႔ data type ေတြ ေရာေရးတဲ႔အခါမ်ိဳးမွာ error ျပမွာ ျဖစ္ပါတယ္။ အဲဒီ ဘာသာစကားေတြရဲ႕ compiler ေတြက ကၽြန္ေတာ္တို႔အေနနဲ႔ မွားယြင္းေရးသားမိတယ္ လို႔ ယူဆၿပီး အဲဒီအမွားကေန ကယ္တင္ဖို႔ ႀကိဳးစားၾကတာပဲ ျဖစ္ပါတယ္။ C နဲ႔ C++ မွာေတာ႔ ကၽြန္ေတာ္တုိ႔ အဲဒီလို ေရာေႏွာေရးသားမႈကို ရည္ရြယ္ခ်က္ တစ္ခုေၾကာင္႔လို႔ ယူဆနားလည္ေပးၿပီး ကူညီ လုပ္ေဆာင္ေပးမွာ ျဖစ္ပါတယ္။ အဲဒီ အခ်က္ဟာ C နဲ႔ C++ ေခတ္စားတဲ႔ အေၾကာင္းတစ္ခုလဲပဲ ျဖစ္ပါတယ္။ ဒါေပမယ္႔ ပိုၿပီးလြတ္လပ္တဲ႔ ဘာသာစကား ေရးသားနည္း တစ္ခုဟာ ပိုၿပီး အမွား လုပ္မိေစတတ္ ပါတယ္။

**Automatic Conversion**

compiler အေနနဲ႔ အဲဒီလို ေရာေႏွာ ေရးသားထားတဲ႔ expressions ေတြနဲ႔ ရင္ဆိုင္ရတဲ႔ အခါမွာ အလုပ္လုပ္သြားပံုကို ေလ႔လာၾကည္႔ရေအာင္။ ေအာက္မွာ ေဖာ္ျပထားတဲ႔ Table အရ data types ေတြကို higher နဲ႔ lower ဆိုၿပီး အၾကမ္းဖ်င္း ခြဲျခားထားႏိုင္ပါတယ္။

|  |  |
| --- | --- |
| **Data Type** | **Order** |
| long double | Highest |
| double |  |
| float |  |
| long |  |
| int |  |
| short |  |
| char | Lowest |

အေပါင္းနဲ႔ အေျမွာက္ တို႔လို arithmetic operators ေတြဟာ တူညီတဲ႔ data type ရွိတဲ႔ operands ေတြကိုပဲ လုပ္ေဆာင္ေပးခ်င္ပါတယ္။ ဒါေၾကာင္႔ ေရာေႏွာေနတဲ႔ data types ေတြနဲ႔ ရင္ဆိုင္ရတဲ႔ အခါမွာ data type အနိမ္႔ကို အျမင္႔သို႔ အလိုအေလ်ာက္ ေျပာင္းလဲေပးမွာ ျဖစ္ပါတယ္။ mixed.cpp မွာ int (count) float ကိုေျပာင္းလဲေပးၿပီး float (avgWeight) နဲ႔ မေျမွာက္ခင္ tempoerary variavle တစ္ခုမွာ သိမ္းဆည္းထားေပးပါတယ္။ float ႏွစ္ခု ေျမွာက္လို႔ ရလာတဲ႔ ရလာဒ္ကို double ေျပာင္းၿပီး totalWeight ထဲကို ထည္႔ေပးပါတယ္။ အဲဒီ ျဖစ္စဥ္ကို ေအာက္မွာ ပံုနဲ႔ ျပထားပါတယ္။



အဲဒီလို data types ေတြ အလိုအေလ်ာက္ ေျပာင္းလဲေပးသြားတာဟာ မျမင္သာပါဘူး။ ကၽြန္ေတာ္တို႔ အေနနဲ႔ ျဖစ္ခ်င္တာေတြကို C++ compiler က အလိုအေလ်ာက္ လုပ္ေပးသြားတာ ျဖစ္တဲ႔အတြက္ အမ်ားႀကီး စဥ္းစားေနစရာလဲ မလိုအပ္ပါဘူး။ ဒါေပမယ္႔ တစ္ခုတစ္ရံမွာ အဲဒီလို ေျပာင္းလဲတာအခ်ိဳ႕ကို compiler အေနနဲ႔ အဆင္ေျပေျပ မျပဳလုပ္ေပးႏိုင္ပါဘူး။ ေနာက္တစ္ခုက ကၽြန္ေတာ္တို႔အေနနဲ႔ objects ေတြ ျပဳလုပ္ျခင္းျဖင္႔ ကိုယ္ပိုင္ data types ေတြဖန္တီးတဲ႔ အခ်ိန္မွာ မတူညီတဲ႔ object data type ေတြကို ေရာေႏွာ အသံုးျပဳဖို႔အတြက္ compiler က ျပဳလုပ္မေပးႏိုင္ပါဘူး။ အဲဒီအခါမွာေတာ႔ ကိုယ္ပိုင္ conversion routine ေတြကို ေရးသား အသံုးျပဳဖို႔ လိုလာမွာ ျဖစ္ပါတယ္။ compiler ဟာ bulit-in dtat types ေတြကိုပဲ အလိုအေလ်ာက္ ေျပာင္းလဲ ေပးႏိုင္စြမ္း ရွိပါတယ္။

**Casts**

တစ္ခါ တစ္ရံမွာ compiler အေနနဲ႔ အလိုအေလ်ာက္ type conversion ျပဳလုပ္မေပးႏိုင္တဲ႔ အေျခအေနမ်ိဳးမွာ data type တစ္ခုကို ကၽြန္ေတာ္တို႔ သတ္မွတ္ေပးထားတဲ႔ အျခား data type တစ္ခုသို႔ ေျပာင္းလဲေပးျခင္းအား type casts သို႔မဟုတ္ Casts လုပ္တယ္လို႔ ေခၚဆိုႏိုင္ပါတယ္။

C++ မွာ static casts, dynamic casts, reinterpret casts နဲ႔ const casts ဆိုၿပီး casts အမ်ိဳးအစား အမ်ားအျပားရွိပါတယ္။ ဒီသင္ခန္းစာမွာေတာ႔ static casts အေၾကာင္းကိုပဲ ေလ႔လာသြားမွာ ျဖစ္ၿပီး သီးျခား အေျခအေနမ်ားမွာ အသံုးျပဳရတဲ႔ က်န္ရွိေနတဲ႔ casts မ်ားကိုေတာ႔ သက္ဆိုင္ရာ သင္ခန္းစာမ်ားမွာ အလ်င္းသင္႔သလို ထည္႔သြင္း ရွင္းျပေပးသြားမွာ ျဖစ္ပါတယ္။

ေအာက္ပါ နမူနာသည္ int data type အား char data type အျဖစ္ ေျပာင္းလဲေပးေသာ ကုဒ္ ျဖစ္သည္။

aCharVar = static\_cast<char>(anIntVar);

ကၽြန္ေတာ္ တို႔ type ေျပာင္းလဲေပးခ်င္တဲ႔ variable (anIntVar) ကို လက္သဲကြင္း () ထဲ ထည္႔သြင္းထားရမွာ ျဖစ္ၿပီး ေျပာင္းလဲမည္႔ data type (char) ကိုေတာ႔ angle brackets <> ထဲမွာ ထည္႔ေပးရမွာ ျဖစ္ပါတယ္။ ဒီနည္းနဲ႔ aCharVar ထဲကို တန္ဖိုး မထည္႔သြင္းခင္မွာ char data type ကို ေျပာင္းလဲ ေပးႏိုင္ပါတယ္။

Standard C++ မတိုင္မီ C ေခတ္တုန္းက ကၽြန္ေတာ္တို႔အေနနဲ႔ အျခား နည္းလမ္းေဟာင္းမ်ားကို cast လုပ္ဖို႔ သံုးခဲ႔ၾကဘူးပါတယ္။ aCharVar = (char) anIntVar; ဒါမွ မဟုတ္ aCharVar = char(anIntVar); ဆိုၿပီး ေရးခဲ႔ၾကပါတယ္။ ဒါေပမယ္႔ အားနည္းခ်က္တစ္ခုက ကုဒ္ေတြအမ်ားႀကီးထဲမွာ casts လုပ္ထားတာကို ရွာရခက္ေလာက္ေအာင္ ေပ်ာက္ေနတတ္ပါတယ္။ Find နဲ႔ ရွာဖို႔လည္း မလြယ္ပါဘူး။ အခု static\_cast ကေတာ႔ အဲဒီ ျပႆနာကို ေျဖရွင္းေပးႏိုင္ပါတယ္။ ဒါေၾကာင္႔ ယခင္ နည္းလမ္းေဟာင္းေတြကို သံုးႏိုင္ေသးေပမယ္႔ static\_cast နည္းလမ္းကိုပဲ အသံုးျပဳၾကဖို႔ တိုက္တြန္းလိုပါတယ္။

ေနာက္အပတ္ေတြမွာေတာ႔ Operator ေတြအေၾကာင္းကို ဆက္လက္ ေဆြးေႏြး သြားပါမယ္ခင္ဗ်ာ။ ဤ ေဆာင္းပါးပါ အခ်က္အလက္မ်ားကို စမ္းသပ္ရာမွာ အခက္အခဲ ရွိေနဦးမယ္ဆိုရင္ စာေရးသူရဲ႕ တင္ျပပံု ခ်ိဳ႕ယြင္းခ်က္ေၾကာင္႔သာ ျဖစ္ပါတယ္။ သိလိုသည္မ်ား ကိုလည္း aungwh2013@gmail.com ကို ဆက္သြယ္ၿပီး ေမးျမန္းႏိုင္သလို ျပင္ဦးလြင္သားမ်ား အတြက္လည္း ေအာက္ေဖာ္ျပပါ လိပ္စာရွိ ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ ၌လည္း ေလ႔လာ စံုစမ္းႏိုင္ပါတယ္ခင္ဗ်ာ။

**အကိုးအကား**

Object-Oriented Programming in C++(4th edition), Robert Lafore, Copyright©2002 by Sams Publishing: ISBN 0-672-32308-7

Dr. ေအာင္ဝင္းထြဋ္ (bluephoenix)

http://www.tech4mm.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕