**OOP in C++ : အပိုင္း (၅) - Data types and Variables (၃)**

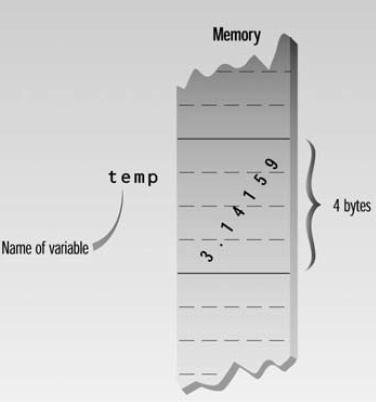
ကၽြန္ေတာ္႔အေနနဲ႔ ဒီသင္ခန္းစာမ်ားကို စတင္ေရးသားစဥ္ကပင္ ေလ႔လာသူမ်ားကို reference အေနနဲ႔ အသံုးျပဳႏိုင္ေစရန္ ရည္ရြယ္ၿပီး အတတ္ႏိုင္ဆံုး အေသးစိတ္ ရွင္းလင္းတင္ျပတဲ႔ ပံုစံနဲ႔ ေရးသားဖို႔ ဆံုးျဖတ္ခဲ႔တာပါ။ ဒါေၾကာင္႔ စိတ္ရွည္ရွည္နဲ႔ အပတ္စဥ္ ဖတ္ရႈအားေပးၾကပါလို႔ ေမတၱာရပ္ခံပါတယ္။

ယခင္ သင္ခန္းစာမ်ားမွာ int နဲ႔ char ဆိုတဲ႔ type ႏွစ္မ်ိဳးအေၾကာင္းကို ရွင္းျပခဲ႔ပါတယ္။ တကယ္ေတာ႔ အဲဒီ type ႏွစ္ခု စလံုးက ဒႆမ မပါတဲ႔ integer ဂဏန္းေတြနဲ႔ ကိုယ္စားျပဳ ေဖာ္ျပခဲ႔ၾကတာပါ။ ဒီအပတ္မွာေတာ႔ floating-point variable မ်ားအေၾကာင္းကို ဆက္လက္ ေဆြးေႏြးသြားမွာ ျဖစ္ပါတယ္။

Floating Point Types

Floating-point variable ေတြဟာ ဒႆမ ပါတဲ႔ ဂဏန္းေတြကို ကိုယ္စားျပဳ သိမ္းဆည္းထားႏိုင္ပါတယ္။ ဥပမာ - 3.1415927, 0.0000625, 10.2 စတာေတြပဲ ျဖစ္ပါတယ္။ ၄င္းတို႔ အားလံုးမွာ decimal point (.) ရဲ႕ ဘယ္ဘက္က integer part ေတြ ျဖစ္ၾကၿပီး၊ ညာဘက္ကေတာ႔ fractional part ေတြပဲ ျဖစ္ပါတယ္။ floating-point variable ေတြဟာ သခၤ်ာ ပညာရွင္ေတြအေနနဲ႔ အကြာအေဝး၊ ဧရိယာ၊ အပူခ်ိန္ စတာေတြကို တိုင္းတာဖို႔ သံုးတဲ႔ real numbers ေတြကို ကိုယ္စားျပဳပါတယ္။ အဲဒီ တိုင္းတာမႈေတြမွာ မ်ားေသာအားျဖင္႔ အပိုင္းကိန္း အစိတ္အပိုင္းေတြ ပါေနေလ႔ ရွိပါတယ္။ C++ မွာေတာ႔ float, double နဲ႔ long double ဆိုၿပီး floating-point variable သံုးမ်ိဳး ရွိပါတယ္။

Type float



ပံု-၅.၁ ကြန္ပ်ဴတာ မွတ္ဥာဏ္၌ float type variable တစ္ခု ေနရာယူျခင္း

float ကေတာ႔ အဲဒီ floating-point variable သံုးမ်ိဳးထဲမွာ အငယ္ဆံုး ျဖစ္ၿပီး 3.4×10-38 ကေန 3.4×1038 ၾကားထဲမွာ ဂဏန္း ခုႏွစ္လံုး precision ရွိတဲ႔ အခ်က္အလက္ေတြကို သိမ္းဆည္းႏိုင္ပါတယ္။ ေအာက္ပါ ဥပမာေလးကေတာ႔ floating-point number ေတြကိုသံုးၿပီး စက္ဝိုင္း တစ္ခုရဲ႕ ဧရိယာကို ရွာတဲ႔ ပရိုဂရမ္ေလးပါ။

// circarea.cpp

// demonstrates floating point variables

#include <iostream> //for cout, etc.

#include <conio.h> //for \_getch()

using namespace std;

int main()

{

float rad; //variable of type float

const float PI = 3.14159F; //type const float

cout << "Enter radius of circle: "; //prompt

cin >> rad; //get radius

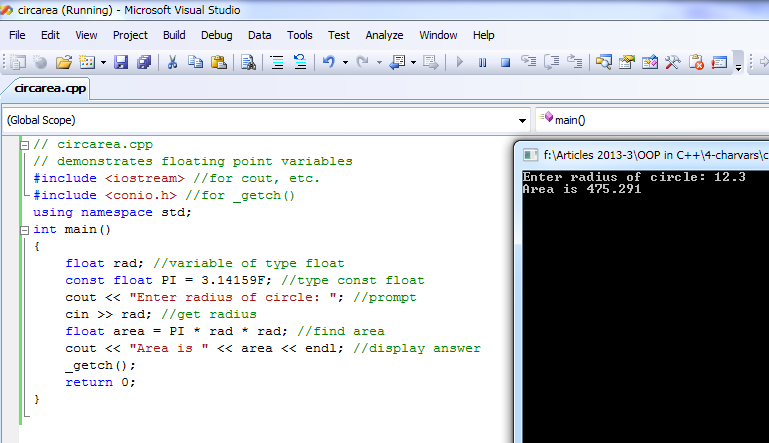
float area = PI \* rad \* rad; //find area

cout << "Area is " << area << endl; //display answer

\_getch();

return 0;

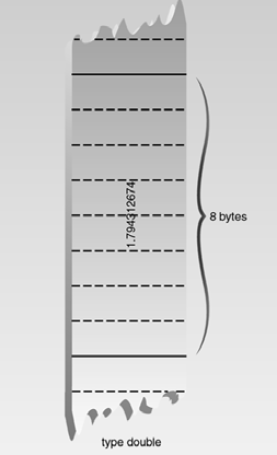
}



ပံု-၅.၂ circarea.cpp ႏွင္႔ testing

Tpye double and long double

ပိုၿပီး အရြယ္အစား ႀကီးမားတဲ႔ floating point types ေတြျဖစ္တဲ႔ double နဲ႔ long double ေတြဟာ ကြန္ပ်ဴတာ မွတ္ဥာဏ္မွာ ေနရာပိုယူၿပီး တန္ဖိုးေတြ ပိုထည္႔သြင္းႏိုင္သလို ပိုမိုတိက်လာတာက လြဲလို႔ float နဲ႔ အတူတူပါပဲ။ double type အေနနဲ႔ သိမ္းဆည္းဖို႔ 8 bytes လိုအပ္ၿပီး 1.7×10-308 ကေန 1.7×10308 ၾကား ကိန္းဂဏန္းေတြကို သိမ္းဆည္းေပးႏိုင္ၿပီး ဒႆမ ၁၅ ေနရာအထိ တိက်ပါတယ္။ long double type ကေတာ႔ ကြန္ပိုင္းလာ အေပၚ မူတည္ေနေပမယ္႔ မ်ားေသာအားျဖင္႔ double နဲ႔ တူေလ႔ ရွိပါတယ္။



ပံု-၅.၃ ကြန္ပ်ဴတာ မွတ္ဥာဏ္၌ double type variable တစ္ခု ေနရာယူျခင္း

Floating-Point Constants

circarea.cpp ပရိုဂရမ္ေလးမွာ အသံုးျပဳခဲ႔တဲ႔ 3.14159F ဆိုတဲ႔ ဂဏန္းေလးဟာ floating-point constant ေလးပဲ ျဖစ္ပါတယ္။ decimal point ကို ျမင္လိုက္တာနဲ႔ integer type မဟုတ္မွန္း သိႏိုင္သလို ေနာက္ဆံုးက F ကေတာ႔ float type ျဖစ္ေၾကာင္း သတ္မွတ္ေပးလိုက္တာပါ။ double အတြက္ကေတာ႔ ေနာက္ဆံုးမွာ ဘာ suffix letter မွ ထည္႔ေပးလိုက္စရာ မလိုဘဲ သာမာန္ ဒႆမကိန္း ပံုစံ ေရးေပးဖို႔ပဲ ျဖစ္ပါတယ္။ double က defaulut type ျဖစ္ေနလို႔ပါပဲ။ long double အတြက္ကေတာ႔ L ကို ထည္႔ေပးဖို႔ လိုမွာ ျဖစ္ပါတယ္။

ကၽြန္ေတာ္တို႔ အေနနဲ႔ floating-point constants ေတြကို exponential notation ကိုသံုးၿပီး ေရးသားႏိုင္ပါတယ္။ ဒီနည္းနဲ႔ သုညေတြ အမ်ားႀကီး သံုးစရာ မလိုေတာ႔ပါဘူး။ ဥပမာ 1,000,000,000 ကို 1.0E9 လို႔ ေဖာ္ျပႏိုင္ပါတယ္။ 1234.56 ကိုလည္း 1.23456E3 လို႔ ေရးႏိုင္ပါတယ္။ ဒီေတာ႔ 109 ကို E9 နဲ႔ 103 ကို E3 လို႔ အလြယ္တကူ ေရးႏိုင္ပါတယ္။ E ေနာက္မွာ ေရးတဲ႔ ဂဏန္းကို ထပ္ညႊန္း exponent လို႔ ေခၚၾကပါတယ္။ exponent ကို ၾကည္႔ၿပီး ဒႆမေနာက္ ဘယ္ေနရာ ေရႊ႕ေပးရမယ္ဆိုတာကို သိႏိုင္ပါတယ္။ exponet ေတြက positive နဲ႔ negative တန္ဖိုး ႏွစ္ခုစလံုး ျဖစ္ႏိုင္ပါတယ္။ 6.35239E-5 ဆိုတာဟာ 0.0000635239 ကို ေဖာ္ျပတာပါ။

The const Qualifier

circarea.cpp ပရိုဂရမ္ထဲမွာ float အျပင္ const ဆိုတဲ႔ စကားလံုး ပါလာပါတယ္။

const float PI = 3014159F; //type const float

constant အတြက္သံုးတဲ႔ const ဆိုတဲ႔ ေသာ႔ခ်က္ စကားလံုးကို variable ေတြေၾကျငာတဲ႔ data type ေတြရဲ႕ ေရွ႕မွာ ေရးရပါတယ္။ အဲဒီ variable ရဲ႕ တန္ဖိုးဟာ ပရိုဂရမ္ တစ္ခုလံုးမွာ ေျပာင္းလဲလို႔ မရေတာ႔ပါဘူး။ တကယ္လို႔ အဲဒီ တန္ဖိုးကို ပရိုဂရမ္ထဲမွာ ေျပာင္းလဲဖို႔ ႀကိဳးစားခဲ႔မယ္ ဆိုရင္ error message ရရွိမွာ ျဖစ္ပါတယ္။

တကယ္ေတာ႔ const qualifier ေလးဟာ ကၽြန္ေတာ္တို႔ ပံုေသ သတ္မွတ္ထားခ်င္တဲ႔ တန္ဖိုးတစ္ခုကို မေတာ္တဆ ေျပာင္းလဲမိျခင္းမွ ကာကြယ္ဖို႔ သံုးရတာ ျဖစ္ပါတယ္။ ကုဒ္ ကို ဖတ္မိတဲ႔ လူတိုင္းကိုလဲ ဒီတန္ဖိုးဟာ ေျပာင္းလဲလို႔ မရဘူးဆိုတာ သတိေပးထားတာပဲ ျဖစ္ပါတယ္။

The #define Directive

C++ မွာ သိပ္မသံုးၾကေတာ႔တဲ႔ ပံုစံ တစ္ခု ျဖစ္ပါတယ္။ constants ေတြကို #define ဆိုတဲ႔ preprocessor directive နဲ႔ သတ္မွတ္ေပးႏိုင္ပါတယ္။ ဒီနည္းနဲ႔ identifier ေတြနဲ႔ ညီမွ် text phrase ေတြကို ဖန္တီးေပးမွာပါ။ ဥပမာ - #define PI 3.14159 ဆိုတဲ႔ စာေၾကာင္းကို ပရိုဂရမ္ရဲ႕ အစမွာ ေရးသားျခင္းျဖင္႔ PI ဆိုတဲ႔ စာသားကို 3.14159 ျဖင္႔ အစားထိုး ေပးမွာ ျဖစ္ပါတယ္။ ဒီလို ေရးသားနည္းကို C language မွာ အေတာ္ေလး ေခတ္စားခဲ႔ဘူးပါတယ္။ ဒါေပမယ္႔ ဘယ္ data type ဆိုတာကို မသတ္မွတ္ႏိုင္တဲ႔ အားနည္းခ်က္ေၾကာင္႔ အမွားအယြင္းေတြ ျဖစ္လာႏိုင္ပါတယ္။ ဒါေၾကာင္႔ C++ နဲ႔ C# မွာ const ကို သံုးၿပီး ေရးသားၾကတာပါ။ ဒါေပမယ္႔ ပရိုဂရမ္ အေဟာင္းေတြကို ဖတ္ၾကည္႔မယ္ ဆိုရင္ေတာ႔ #define ေတြ အမ်ားႀကီး ေတြ႔ရႏိုင္ပါတယ္။

Type bool

int data type မွာ ျဖစ္ႏိုင္ေျခရွိတဲ႔ တန္ဖိုးေတြ ဘီလီယံနဲ႔ ခ်ီၿပီး ရွိႏိုင္ပါတယ္။ char မွာေတာ႔ ၂၅၆ လံုး ျဖစ္ႏိုင္ပါတယ္။ bool ကေတာ႔ true နဲ႔ false ဆိုၿပီး ႏွစ္မ်ိဳးပဲ ျဖစ္ႏိုင္ပါတယ္။ သီအိုရီအရ bool ဟာ one bit ပဲ ေနရာယူပါတယ္။ ဒါေပမယ္႔ တကယ္႔လက္ေတြ႔မွာ bit အေနနဲ႔ သိမ္းခဲ႔ရင္ byte ကေန ျပန္ၿပီး ခြဲထုတ္ယူရတာ ျဖစ္လို႔ အခ်ိန္ပိုၾကာတာေၾကာင္႔ ကြန္ပိုင္လာက ကိုင္တြယ္ရယူဖို႔ ျမန္ဆန္တဲ႔ byte အေနနဲ႔ ပဲ သိမ္းဆည္းတာ ျဖစ္ပါတယ္။ ႏိႈင္းယွဥ္ စစ္ေဆးခ်က္ေတြရဲ႕ ရလာဒ္ေတြကို သိမ္းဆည္းဖို႔ bool type ကို အသံုးမ်ားပါတယ္။ ဥပမာ ကိုကို ဟာ ေမာင္ေမာင္ထက္ အသက္ႀကီးလား ဆိုတဲ႔ ႏိႈ္င္းယွဥ္ခ်က္ မွာ မွန္ခဲ႔ရင္ bool value - true ကို ရရွိမွာ ျဖစ္ၿပီး မွားခဲ႔ပါက false ကို ရရွိမွာ ျဖစ္ပါတယ္။ bool ဆိုတဲ႔ နာမည္ကို true-or-false တန္ဖိုးေတြ နဲ႔ logical operators ေတြကို သံုးၿပီး စဥ္းစားေတြးေခၚနည္းကို တီထြင္ခဲ႔တဲ႔ ၁၉ ရာစု အဂၤလိပ္ သခၤ်ာပညာရွင္ George Boole အား အစြဲျပဳၿပီး ဂုဏ္ျပဳေခၚဆိုၾကတာပါ။

The setw Manipulator

ကၽြန္ေတာ္တို႔ အေနနဲ႔ endl အေၾကာင္းကို ေရးသားစဥ္က အခ်က္အလက္ေတြကို screen မွာ ျပသတဲ႔အခါ insertion operator (<<) နဲ႔ တြဲသံုးတဲ႔ manipulator ျဖစ္ေၾကာင္း ရွင္းျပခဲ႔ဘူးပါတယ္။ အခု သင္ခန္းစာမွာေတာ႔ output ရဲ႕ field width ကို ေျပာင္းလဲ ေပးႏိုင္တဲ႔ setw manipulator အေၾကာင္းပဲ ျဖစ္ပါတယ္။

ကၽြန္ေတာ္တို႔ cout ကို သံုးၿပီး တန္ဖိုး တစ္ခုကို screen မွာ ျပသမယ္ဆိုရင္ အဲဒီတန္ဖိုးဟာ တိက်တဲ႔ width ရွိတဲ႔ မျမင္ရတဲ႔ ေလးေထာင္႔ ေနရာတစ္ခုကို ယူထားတယ္လို႔ စိတ္ကူး ၾကည္႔ႏိုင္ပါတယ္။ default field ကေတာ႔ အဲဒီ တန္ဖိုး ဆန္႔သေလာက္ ေနရာယူထားတာပါ။ ဥပမာ- integer 567 ဟာ အကၡရာ သံုးလံုးစာ ေနရာယူမွာ ျဖစ္ၿပီး "pajamas" ဆိုတဲ႔ string ကေတာ႔ ခုႏွစ္ေနရာ ယူမွာ ျဖစ္ပါတယ္။ ဒါေပမယ္႔ အခ်ိဳ႕ ေနရာေတြမွာေတာ႔ ကၽြန္ေတာ္တို႔ စိတ္ႀကိဳက္ ျပင္ဆင္ သတ္မွတ္ေပးဖို႔ လိုလာပါတယ္။ နမူနာ အေနနဲ႔ width1.cpp ပရိုဂရမ္ေလးကို ေလ႔လာၾကည္႔ရေအာင္။ ဒီပရိုဂရမ္ေလးမွာ ၿမိဳ႕သံုးခုရဲ႕ အမည္ကို ေကာ္လံ တစ္ခုမွာ ျပသထားၿပီး အျခားေကာ္လံမွာေတာ႔ သက္ဆိုင္ရာ လူဦးေရမ်ားကို ေဖာ္ျပေပးမွာ ျဖစ္ပါတယ္။

// width1.cpp

// demonstrates need for setw manipulator

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

long pop1=2425785, pop2=47, pop3=9761;

cout << "LOCATION " << "POP." << endl

<< "Portcity " << pop1 << endl

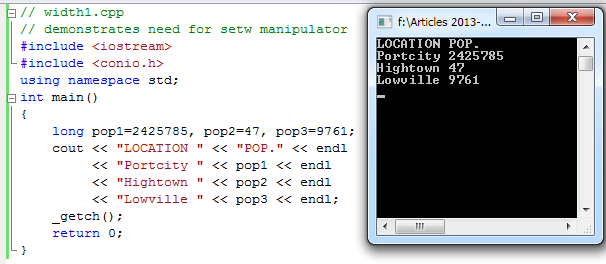
<< "Hightown " << pop2 << endl

<< "Lowville " << pop3 << endl;

\_getch();

return 0;

}



ပံု-၅.၄ width1.cpp ႏွင္႔ testing

ကၽြန္ေတာ္တို႔ သတိထားမိတဲ႔အတိုင္း output မွာပါတဲ႔ လူဦးေရ အေရအတြက္ ေတြဟာ ႏိႈင္းယွဥ္ရ ခက္ခဲေနပါတယ္။ တကယ္လို႔ ညာဘက္ကို ညွိထားမယ္ ဆိုရင္ ဖတ္ရ ပိုမို လြယ္ကူလာမွာ ျဖစ္ပါတယ္။ ၿမိဳ႕နာမည္ေတြ ေဖာ္ျပတဲ႔အခါမွာလည္း ေနာက္က ဂဏန္းေတြၾကားမွာ ေနရာလြတ္ (spaces) ေတြ ျခားေပးႏိုင္မယ္ဆိုရင္ ပိုၿပီး အဆင္ေျပလာမွာပါ။ အထက္က width1.cpp ကိုပဲ အနည္းငယ္ ျပင္ဆင္ၿပီး with2.cpp နာမည္နဲ႔ setw manipulator အသံုးျပဳကာ ျပန္ေရးပါမယ္။

// width2.cpp

// demonstrates setw manipulator

#include <iostream>

#include <iomanip> // for setw

#include <conio.h>

using namespace std;

int main()

{

long pop1=2425785, pop2=47, pop3=9761;

cout << setw(8) << "LOCATION" << setw(12)

<< "POPULATION" << endl

<< setw(8) << "Portcity" << setw(12) << pop1 << endl

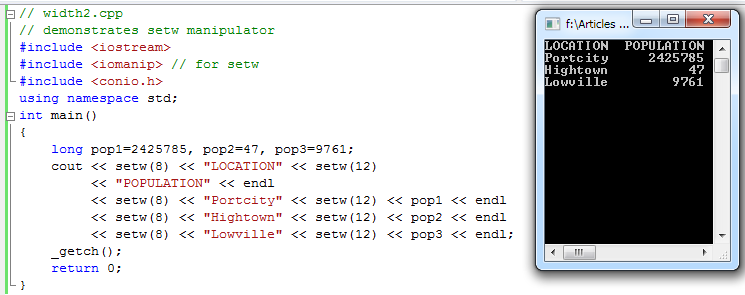
<< setw(8) << "Hightown" << setw(12) << pop2 << endl

<< setw(8) << "Lowville" << setw(12) << pop3 << endl;

\_getch();

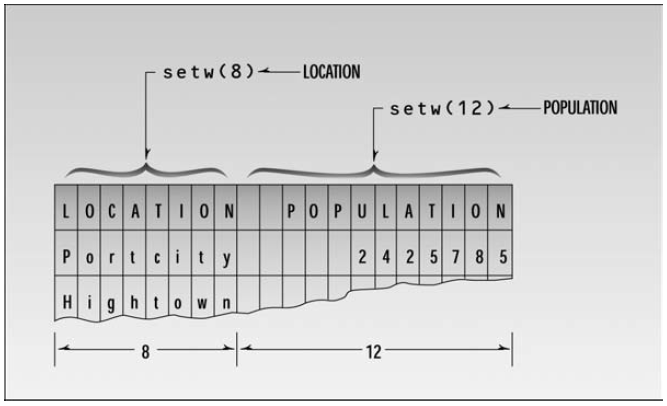
return 0;

}



ပံု-၅.၅ width2.cpp ႏွင္႔ testing

setw manipulator က သူ႔ေနာက္က ဂဏန္းေတြနဲ႔ စာသားေတြကို n character အက်ယ္ရွိတဲ႔ field အတြင္းမွာ ေဖာ္ျပေပးပါတယ္။ အဲဒီေနရာမွာ n ဆိုတာက setw(n) functin ကြင္းထဲက argument (n) ပဲ ျဖစ္ပါတယ္။ အဲဒီ တန္ဖိုးက field ထဲမွာ right-justified ျဖစ္ေနမွာပါ။ အထက္က ပံု-၅.၅ မွာ output မွာ ျမင္ရမယ္႔ ပံုကို ျပထားပါတယ္။ ထူးျခားခ်က္အေနနဲ႔ လူဦးေရေတြကို ေဖာ္ျပတဲ႔ ေနရာမွာ 2-byte ေနရာယူၿပီး အမ်ားဆံုး တန္ဖိုး 32767 ပဲ သိမ္းလို႔ရတဲ႔ အတြက္ overflow ျဖစ္ႏိုင္ေျခရွိတဲ႔ integer type ေတြကို မသံုးဘဲ long ကို သံုးထားတာပါ။



ပံု-၅.၆ width2.cpp ႏွင္႔ testing

Cascading the Insertion Operator

width1.cpp နဲ႔ width2.cpp မွာ ပါရွိတဲ႔ cout statement ထဲမွာ cout ကို တစ္ခါပဲ သံုးထားၿပီး စာေၾကာင္းေရ အေတာ္မ်ားမ်ား ေရးထားတာကို သတိထားမိမွာပါ။ ကြန္ပိုင္လာ က whitespace ေတြကို ထည္႔မစဥ္းစားတာနဲ႔ insertion operator (<<) ေတြကို cascade လုပ္ၿပီး ထပ္ခါ ထပ္ခါ ေရးလို႔ရတဲ႔ အခ်က္ေတြကို အသံုးခ်ထားတာ ျဖစ္တယ္။ cout ေလးေၾကာင္းေရးၿပီး ရလာမယ္႔ ရလာဒ္နဲ႔ အတူတူပဲ ျဖစ္ပါတယ္။

Multiple Deifinitions

variables ေတြျဖစ္တဲ႔ pop1, pop2 နဲ႔ pop3 တို႔ကို ဖန္တီးစဥ္ တန္ဖိုးေတြပါ တစ္ပါတည္း သတ္မွတ္ေပးထားတာကို ေတြ႔ရပါမယ္။ ဒီနည္းနဲ႔ data type တူညီတဲ႔ variables ေတြကို ေကာ္မာ ခံၿပီး ဖန္တီးေပးျခင္း၊ တန္ဖိုး သတ္မွတ္ေပးျခင္းျဖင္႔ ကုဒ္မ်ားကို ေလ်ာ႔ခ်ႏိုင္ပါတယ္။

The IOMANIP Header File

endl, setw() စတဲ႔ manipulators ေတြကို အသံုးျပဳဖို႔ဆိုရင္ ကၽြန္ေတာ္တို႔ iostream header file ထဲမွာ မပါတဲ႔ အတြက္ iomanip header file ကို ထပ္မံ ေၾကျငာေပးရမွာ ျဖစ္ပါတယ္။ width2.cpp မွာ #include <iomanip> // for setw ဆိုၿပီး ေရးသားသြားတာပါ။

ေနာက္အပတ္ေတြမွာေတာ႔ data type တစ္ခုကေန ေနာက္တစ္ခုကို ဘယ္လို ေျပာင္းလို႔ရတယ္ ဆိုတဲ႔ အေၾကာင္းကို ဆက္လက္ ေဆြးေႏြးသြားပါမယ္ခင္ဗ်ာ။ ဤ ေဆာင္းပါးပါ အခ်က္အလက္မ်ားကို စမ္းသပ္ရာမွာ အခက္အခဲ ရွိေနဦးမယ္ဆိုရင္ စာေရးသူရဲ႕ တင္ျပပံု ခ်ိဳ႕ယြင္းခ်က္ေၾကာင္႔သာ ျဖစ္ပါတယ္။ သိလိုသည္မ်ားကိုလည္း 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