**OOP in C++ : အပိုင္း (၂၇) - Functions (5)**

Function ေတြဟာ C++ ရဲ႕ အေျခခံက်တဲ႔ အစိတ္အပိုင္းေတြ ျဖစ္တဲ႔ အတြက္ အေသးစိတ္ သိရွိထားဖို႔လည္း လိုအပ္ပါတယ္။ ဒါေၾကာင္႔ ဒီအပတ္မွာလည္း function ေတြ အေၾကာင္းကိုပဲ ဆက္လက္ေဆြးေႏြးသြားမွာ ျဖစ္ပါတယ္။

**Names in the Declaration**

Function declarations ေတြကို ေၾကျငာတဲ႔ ေနရာမွာ ပိုမိုရွင္းလင္းေအာင္ ေရးသားႏိုင္ပါေသးတယ္။ Input arguments အေနနဲ႔ ထည္႔ေပးရမယ္႔ data type ေတြနဲ႔ အတူ အဓိပၸါယ္ရွိတဲ႔ variable name ေတြ ထည္႔ေပးလိုက္ဖို႔ပဲ ျဖစ္ပါတယ္။ စာေရးသူအေနနဲ႔ကေတာ႔ function declarations ေတြျပဳလုပ္ဖို႔ ဒီနည္းကို အျမဲအသံုးျပဳပါတယ္။ ဥပမာအေနနဲ႔ ကြန္ပ်ဴတာ ဖန္သားျပင္ေပၚမွာ အမွတ္တစ္ခုကို ျပသေပးတဲ႔ function ေလးတစ္ခုကို အသံုးျပဳေနတယ္ ဆိုၾကပါစို႔။ ယခင္ ဥပမာမ်ားအရ ေအာက္ပါအတိုင္း data typed ေတြကို ထည္႔သြင္းၿပီး ေရးသားႏိုင္ပါတယ္။

void display\_point(int, int); //declaration

အထက္ပါ function declaration ေလးကို ေအာက္ပါအတိုင္း ပိုမိုရွင္းလင္းေအာင္ ျပင္ေရးႏိုင္ပါတယ္။

void display\_point(int horiz, int vert); //declaration

Compiler အတြက္ေတာ႔ အထက္ပါ declarations ႏွစ္ခုစလံုးကဟာ အတူတူပါပဲ။ ဒါေပမယ္႔ ပထမ declaration မွာ ပါတဲ႔ (int, int) ဟာ ထည္႔ေပးရမယ္႔ arguments ေတြနဲ႔ ပက္သက္ၿပီး ဘာမွ မေျပာထားတဲ႔ အတြက္ ဘယ္တစ္ခုက vertical coordinate အတြက္ ျဖစ္ၿပီး၊ ဘယ္တစ္ခုက horizontal coordinate အတြက္ ျဖစ္တယ္ဆိုတာကို မသိႏိုင္ပါဘူး။

ဒုတိယနည္းလမ္းကို အသံုးျပဳၿပီး ေၾကျငာလိုက္တဲ႔ အခါမွာ ပရိုဂရမ္မာေတြ အတြက္ ပိုမိုရွင္းလင္း လြယ္ကူေစပါတယ္။ ဒီလို declaration မ်ိဳးကို ဖတ္မိတဲ႔ မည္သူမဆို အဲဒီ function ကို ေခၚယူတဲ႔ ေနရာမွာ arguments ေတြကို မွန္မွန္ကန္ကန္ ထည္႔သြင္းမိဖို႔မ်ားပါတယ္။ မွတ္သားရမယ္႔ အခ်က္ တစ္ခုကေတာ႔ အဲဒီ declarations ေတြထဲမွာ အသံုးျပဳထားတဲ႔ variable names ေတြဟာ function တစ္ခုကို ေခၚယူတဲ႔ အခါမွာ အသံုးျပဳမယ္႔ variable names ေတြအေပၚမ်ာ အက်ိဳးသက္ေရာက္မႈ မရွိတာပဲ ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ အဲဒီလို ေၾကျငာဖို႔ ႀကိဳက္ႏွစ္သက္ရာ variable names ေတြကို လြတ္လြတ္လပ္လပ္ အသံုးျပဳႏိုင္ပါတယ္။ ဥပမာ-

display\_point(x, y); // function call

ဒါေၾကာင္႔ ကၽြန္ေတာ္ေရးသားမယ္႔ ေနာက္ပိုင္း သင္ခန္းစာမ်ားမွာ စာဖတ္သူမ်ား လြယ္ကူစြာ နားလည္ႏိုင္ေစဖို႔ ရည္ရြယ္ၿပီး ဒုတိယနည္းလမ္းျဖစ္တဲ႔ variable name နဲ႔ data type တြဲသံုးၿပီး ေရးသားျပသြားမွာ ျဖစ္ပါတယ္။

**Returning Values from Functions**

Function တစ္ခုကို ေဆာင္ရြက္ၿပီးသြားတဲ႔ အခါမွာ တန္ဖိုးတစ္ခုကို ေခၚယူတဲ႔ ပရိုဂရမ္ကို ျပန္ပို႔ေပးပါတယ္။ မ်ားေသာအားျဖင္႔ အဲဒီ တန္ဖိုးဟာ function က သက္ဆိုင္ရာ ျပႆနာ တစ္ခုကို ေျဖရွင္းေပးလိုက္တဲ႔ အေျဖေတြ ပါဝင္ေလ႔ ရွိပါတယ္။ ေအာက္ပါ ဥပမာမွာ အေလးခ်ိန္တစ္ခုကို ေပါင္ကေန ကီလိုဂရမ္ ေျပာင္းေပး မွာ ျဖစ္ပါတယ္။ convert.cpp ေလးကို ေလ႔လာၾကည္႔ၾကရေအာင္-

// convert.cpp

// demonstrates return values, converts pounds to kg

#include <iostream>

using namespace std;

float lbstokg(float); //declaration

int main()

{

float lbs, kgs;

cout << “\nEnter your weight in pounds: “;

cin >> lbs;

kgs = lbstokg(lbs);

cout << “Your weight in kilograms is “ << kgs << endl;

return 0;

}

//--------------------------------------------------------------

// lbstokg()

// converts pounds to kilograms

float lbstokg(float pounds)

{

float kilograms = 0.453592 \* pounds;

return kilograms;

}

ေအာက္မွာ အဲဒီ ပရိုဂရမ္ကို နမူနာ စမ္းသပ္ျပထားပါတယ္။ output ေတြကို ေလ႔လာၾကည္႔ပါ-

Enter your weight in pounds: 182

Your weight in kilograms is 82.553741

Function တစ္ခုမွ တန္ဖိုးတစ္ခုကို return ျပန္ေပးႏိုင္ရန္အတြက္ အဆိုပါ တန္ဖိုး၏ data type ကို သတ္မွတ္ေပးထားရမည္ ျဖစ္သည္။ ထိုသို႔သတ္မွတ္ျခင္းကို function declaration အတြင္း ထည္႔သြင္းေရးသားရမည္ ျဖစ္သည္။ အထက္ပါ ဥပမာျပထားတဲ႔ အတိုင္း function declaration နဲ႔ definition နွစ္ခုစလံုးမွာ (float) data type ကို function name ရဲ႕ အေရွ႕မွာ ေရးသားသတ္မွတ္ရမွာ ျဖစ္ပါတယ္။

ယခင္က ဥပမာေပး ေရးသားျပထားတဲ႔ နမူနာ ပရိုဂရမ္မ်ားမွာ တန္ဖိုးျပန္ေပးတာ မရွိခဲ႔ပါဘူး။ ဒါေၾကာင္႔ return type ကို void လို႔ ေရးသားေပးရပါတယ္။ convert.cpp မွာ ေရးသားထားတဲ႔ lbstokg() မွာေတာ႔ float type ကို return ျပန္ေပးထားပါတယ္။ ဒါေၾကာင္႔ သူရဲ႕ declaration က ေအာက္ပါအတိုင္း ျဖစ္ပါတယ္-

float lbstokg(float);

ေရွ႕ဆံုးမွာ ရွိတဲ႔ float က return type ကို သတ္မွတ္ေပးတာ ျဖစ္ပါတယ္။ ေနာက္က လဲသဲကြင္းထဲမွာ ရွိတဲ႔ float ကေတာ႔ အဲဒီ function (lbstokg()) ကို ထည္႔သြင္းေပးရမယ္႔ data type ကို သတ္မွတ္ေပးထားတာ ျဖစ္ပါတယ္။

Function တစ္ခုက တန္ဖိုးတစ္ခုကို return ျပန္ေပးတဲ႔ အခါမွာ lbstokg(lbs) ဆိုတဲ႔ function call ကို အဲဒီ function က ျပန္ေပးတဲ႔ တန္ဖိုးအေနနဲ႔ ယူဆၿပီး အသံုးျပဳမွာ ျဖစ္ပါတယ္။ ဥပမာအားျဖင္႔ အဲဒီက ထြက္လာတဲ႔ တန္ဖိုးကို အျခား variable ေတြကဲ႔သို႔ ယူဆၿပီး assignment statement မွာ

kgs = lbstokg(lbs);

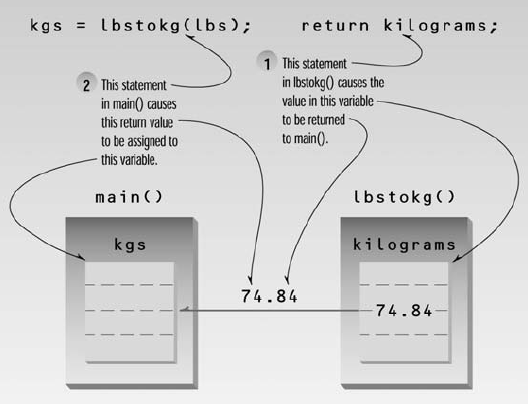
ဒီကုဒ္က lbstokg() ကေန return ျပန္ေပးလိုက္တဲ႔ တန္ဖိုးကို kgs ဆိုတဲ႔ variable ထဲ assign လုပ္ၿပီး ထည္႔ေပးသြားမွာ ျဖစ္ပါတယ္။

**The return Statement**

Function lbstokg() ထဲကို အေလးခ်ိန္ ေပါင္တန္ဖိုးေတြကို ကိုယ္စားျပဳတဲ႔ argument တစ္ခုထည္႔သြင္းေပးတဲ႔ အခါမွာ pounds ဆိုတဲ႔ parameter ထဲမွာ သိမ္းဆည္းထားလိုက္မွာ ျဖစ္ပါတယ္။ အဲဒီ ေပါင္တန္ဖိုးေတြကို ကီလိုဂရမ္ေျပာင္းလဲဖို႔ ကိန္းေသတစ္ခုနဲ႔ ေျမွာက္ၿပီး kilograms ဆိုတဲ႔ variable ထဲမွာ သိမ္းထားလိုက္ပါတယ္။ အဲဒီ variable ထဲက တန္ဖိုးကို ေအာက္ပါအတိုင္း return statement အသံုးျပဳၿပီး ေခၚယူတဲ႔ ပရိုဂရမ္ကို ျပန္ေပးမွာ ျဖစ္ပါတယ္။

return kilograms;

သတိထားရမယ္႔ အခ်က္တစ္ခုကေတာ႔ ကီလိုဂရမ္ တန္ဖိုးကို သိမ္းဆည္းဖို႔ main() function ထဲမွာ kgs ဆိုတဲ႔ variable ကို အသံုးျပဳထားၿပီး lbstokg() ထဲမွာ kilograms ဆိုတဲ႔ variable ကို အသံုးျပဳထားပါတယ္။ Calling program ျဖစ္တဲ႔ main() function က lbstokg() function ထဲက kilograms ဆိုတဲ႔ variable ကို ရယူသံုးစြဲျခင္း မျပဳဘဲ (သံုးလို႔လဲ မရပါဘူး) အဲဒီ lbstokg() function ကေန ျပန္ေပးတဲ႔ တန္ဖိုးကိုပဲ ရယူသံုးစြဲထားတာ ျဖစ္ပါတယ္။ အဲဒီ ျဖစ္စဥ္ကို ေအာက္ပါ ပံုမွာ ရွင္းလင္း တင္ျပထားပါတယ္။



ကၽြန္ေတာ္တို႔ အေနနဲ႔ function တစ္ခုကို arguments ေတြအမ်ားႀကီး ထည္႔သြင္းေပးလို႔ ရေပမယ္႔ return ျပန္ေပးတဲ႔ အခါမွာေတာ႔ argument တစ္ခုပဲ ျပန္ေပးႏိုင္ပါတယ္။ တကယ္လို႔ function တစ္ခုကေန အခ်က္အလက္အမ်ားအျပား ေပးခ်င္လာတဲ႔ အခါမွာ အဲဒီ ကန္႔သတ္ခ်က္ ေၾကာင္႔ အခက္အခဲ ျဖစ္လာႏိုင္ပါတယ္။ ဒါေၾကာင္႔ အဲဒီအခက္အခဲကို ေက်ာ္လႊားဖို႔အတြက္ arguments ေတြကို pass by reference နည္းလမ္းနဲ႔ေသာ္ လည္းေကာင္း၊ အခ်က္အလက္ အမ်ားအျပားကို စုစည္းၿပီး structure တစ္ခုတည္ေဆာက္ကာ return ျပန္ေပးျခင္းျဖင္႔ ေသာ္လည္းေကာင္း ေျဖရွင္းႏိုင္ပါတယ္။

ကၽြန္ေတာ္တို႔ အေနျဖင္႔ function တစ္ခု၏ return type ကို function declaration မွာ မျဖစ္မေန ထည္႔သြင္းေပးရမွာ ျဖစ္ပါတယ္။ အကယ္၍ function အေနျဖင္႔ return ျပန္ေပးရန္ မလိုအပ္ပါက void ကို ထည္႔သြင္းေပးျခင္းျဖင္႔ ညႊန္ျပႏိုင္ပါသည္။ function declaration တြင္ return type ထည္႔သြင္းမေပးပါက compiler အေနျဖင္႔ int value ျပန္ေပးလိမ္႔မည္ဟု ယူဆမည္ ျဖစ္သည္။ ဥပမာအေနျဖင္႔ ေအာက္ပါ declaration ကို ေလ႔လာၾကည္႔ပါ-

somefunc(); // declaration -- assumes return type is int tells the compiler that somefunc() တြင္ return type ကို သတ္မွတ္ေပးထားျခင္းမရွိ၍ int ဟု ယူဆမည္ ျဖစ္သည္။ ဤသို႔ ယူဆရျခင္းမွာ အေစာပိုင္း C versions မ်ား အသံုးျပဳပံုကို အေျခခံထားျခင္းေၾကာင္႔ ပင္ျဖစ္သည္။ သို႔ရာတြင္ ၄င္းနည္းလမ္းကို လက္ေတြ႔တြင္ အသံုးမျပဳသင္႔ေပ။ retun type သည္ int ျဖစ္ေနရင္ေတာင္မွ declaration တြင္ မျဖစ္မေန သတ္မွတ္ေပးသင္႔ေပသည္။ သို႔မွသာ ကုဒ္မ်ား တစ္ပံုစံတည္း ျဖစ္ၿပီး ဖတ္ရလြယ္မည္ ျဖစ္သည္။

**Eliminating Unnecessary Variables**

convert.cpp ပရိုဂရမ္တြင္ တကယ္တမ္း မလိုအပ္ဘဲ ဖတ္ရႈရ ရွင္းလင္းေစရန္ ထည္႔သြင္း အသံုးျပဳထားေသာ variables အမ်ားအျပား ပါဝင္သည္။ ေအာက္ပါ convert2.cpp ပရိုဂရမ္တြင္ မလိုအပ္ေသာ variables မ်ားကို ျဖဳတ္၍ ေရးသားျပထားသည္။

// convert2.cpp

// eliminates unnecessary variables

#include <iostream>

using namespace std;

float lbstokg(float); //declaration

int main()

{

float lbs;

cout << “\nEnter your weight in pounds: “;

cin >> lbs;

cout << “Your weight in kilograms is “ << lbstokg(lbs)

<< endl;

return 0;

}

//--------------------------------------------------------------

// lbstokg()

// converts pounds to kilograms

float lbstokg(float pounds)

{

return 0.453592 \* pounds;

}

main() function တြင္ convert.cpp တြင္ အသံုးျပဳထားေသာ variable kgs ကို ျဖဳတ္ပစ္ၿပီး function call - lbstokg(lbs) ကို cout statement အတြင္းတြင္ ေအာက္ပါအတိုင္း တိုက္ရိုက္ အသံုးျပဳထားသည္။

cout << “Your weight in kilograms is “ << lbstokg(lbs) << endl;

ထိုနည္းတူ lbstokg() function တြင္လည္း variable kilograms ကို အသံုးမျပဳေတာ႔ဘဲ 0.453592\*pounds ဟူသည္႔ expression ကို return statement တြင္ တိုက္ရိုက္ထည္႔သြင္းထားသည္။

return 0.453592 \* pounds;

ယခင္ နည္းလမ္းအရ တြက္ခ်က္မႈမွ ရရွိေသာ တန္ဖိုးကို variable အတြင္း ထည္႔သြင္းကာ calling program ကို return ျပန္ေပးမည္ျဖစ္သည္။ ယခု နည္း၌မူ တြက္ခ်က္ရေသာ တန္ဖိုးကို calling program သို႔ တိုက္ရိုက္ return ျပန္ေပးမည္ျဖစ္သည္။ ပိုမိုရွင္းလင္းေစရန္အတြက္ ပရိုဂရမ္မာ အမ်ားစုက expression ကို လက္သဲကြင္းျဖင္႔ ပိတ္၍ ေအာက္ပါအတိုင္း ေရးသားေလ႔ ရွိသည္။

return (0.453592 \* pounds);

Compiler အေနျဖင္႔ အဆိုပါ လက္သဲကြင္းမ်ားကို မလိုအပ္ေသာ္လည္း ကၽြန္ေတာ္တို႔ ပရိုဂရမ္မာမ်ား ဖတ္ရႈရ လြယ္ကူေစရန္ ထည္႔သြင္းေပးသင္႔သည္။ ကၽြမ္းက်င္ေသာ C++ (သို႔မဟုတ္ C) ပရိုဂရမ္မာမ်ား အေနျဖင္႔ ပိုမိုရွည္လ်ားေသာ convert.cpp ပံုစံမ်ိဳးထက္ တိုေတာင္းေသာ convert2.cpp ပံုစံမ်ိဳးကို ႀကိဳက္ႏွစ္သက္ေလ႔ ရွိၾကသည္။ သို႔ရာတြင္ convert2.cpp ပံုစံသည္ နားလည္ရန္ ခက္ခဲႏိုင္ေသာေၾကာင္႔ မကၽြမ္းက်င္ေသးသူမ်ားအဖို႔ အသံုးမျပဳသင္႔ေပ။ ကၽြန္ေတာ္တို႔ အေနျဖင္႔ တိုေတာင္းမႈ ႏွင္႔ ရွင္းလင္းမႈကို ေရြးခ်ယ္ရသည္႔အခါတြင္ ကၽြန္ေတာ္တို႔၏ ေရးသားမႈစတိုင္ႏွင္႔ အဆိုပါ ကုဒ္မ်ားကို မည္သူက ဖတ္ရႈမည္ဟူေသာ အခ်က္ကို အေျခခံ၍ ေရြးခ်ယ္ၾကရမည္ ျဖစ္ပါသည္။

**Returning Structure Variables**

ေရွ႕ပိုင္း သင္ခန္းစာမ်ားတြင္ functions မ်ားအတြင္း arguments မ်ားအေနျဖင္႔ structures မ်ားကို ထည္႔သြင္းေပးႏိုင္ေၾကာင္း ေလ႔လာခဲ႔ၾကပါသည္။ ထိုနည္းတူစြာ structure တစ္ခုကို return value အျဖစ္လည္း အသံုးျပဳႏိုင္ပါသည္။ retstrc.cpp တြင္ Distance variables မ်ားကို ေပါင္းကာ ရရွိလာေသာ Distance data type တန္ဖိုးကို return ျပန္ေပးမည္ ျဖစ္ပါသည္။

// retstrc.cpp

// demonstrates returning a structure

#include <iostream>

using namespace std;

////////////////////////////////////////////////////////////////

struct Distance //English distance

{

int feet;

float inches;

};

////////////////////////////////////////////////////////////////

Distance addengl(Distance, Distance); //declarations

void engldisp(Distance);

int main()

{

Distance d1, d2, d3; //define three lengths

//get length d1 from user

cout << “\nEnter feet: “; cin >> d1.feet;

cout << “Enter inches: “; cin >> d1.inches;

//get length d2 from user

cout << “\nEnter feet: “; cin >> d2.feet;

cout << “Enter inches: “; cin >> d2.inches;

d3 = addengl(d1, d2); //d3 is sum of d1 and d2

cout << endl;

engldisp(d1); cout << “ + “; //display all lengths

engldisp(d2); cout << “ = “;

engldisp(d3); cout << endl;

return 0;

}

//--------------------------------------------------------------

// addengl()

// adds two structures of type Distance, returns sum

Distance addengl( Distance dd1, Distance dd2 )

{

Distance dd3; //define a new structure for sum

dd3.inches = dd1.inches + dd2.inches; //add the inches

dd3.feet = 0; //(for possible carry)

if(dd3.inches >= 12.0) //if inches >= 12.0,

{ //then decrease inches

dd3.inches -= 12.0; //by 12.0 and

dd3.feet++; //increase feet

} //by 1

dd3.feet += dd1.feet + dd2.feet; //add the feet

return dd3; //return structure

}

//--------------------------------------------------------------

// engldisp()

// display structure of type Distance in feet and inches

void engldisp( Distance dd )

{

cout << dd.feet << “\’-” << dd.inches << “\””;

}

အသံုးျပဳသူမ်ားအေနျဖင္႔ ေပ ႏွင္႔ လက္မ မ်ားကို အလ်ား ႏွစ္စံုအတြက္ ထည္႔သြင္းေပးရမည္ ျဖစ္ၿပီး addengl() function ကို အသံုးျပဳ၍ တန္ဖိုးႏွစ္ခုကို ေပါင္းေပးမည္ ျဖစ္သည္။ ထို႔ေနာက္ ရလာေသာ တန္ဖိုးမ်ားကို engldisp() function ကို အသံုးျပဳ၍ ျပသေပးမည္ ျဖစ္သည္။ ေအာက္မွာ အဲဒီ ပရိုဂရမ္ကို နမူနာ စမ္းသပ္ျပထားပါတယ္။ output ေတြကို ေလ႔လာၾကည္႔ပါ-

Enter feet: 4

Enter inches: 5.5

Enter feet: 5

Enter inches: 6.5

4’-5.5” + 5’-6.5” = 10’-0”

Distance type အလ်ားႏွစ္ခုကို ေပါင္းဖို႔ addengl() function ကို ေခၚယူၿပီး ရလာတဲ႔ တန္ဖိုး ကို ေနာက္ထပ္ Distance type ျဖစ္တဲ႔ d3 ထဲကို ေအာက္ပါအတိုင္း ထည္႔သြင္းေပးလိုက္ပါတယ္။

d3 = addengl(d1, d2);

ဒီေန႔ သင္ခန္းစာေလးမွာ structure တစ္ခုကို function ရဲ႕ return value အျဖစ္ ျပန္ေပးႏိုင္တဲ႔အေၾကာင္း ေလ႔လာရသလို အထက္ပါ ပရိုဂရမ္ေလးမွာ function ႏွစ္ခုကို သံုးျပထားပါတယ္။ အဲဒီ function ေတြရဲ႕ definitions ေတြကို ေရွ႕ေနာက္ စိတ္ႀကိဳက္ စီစဥ္ထားႏိုင္ပါတယ္။ တစ္ခုတည္းေသာ စည္းမ်ဥ္းကေတာ႔ function call ေတြ မတိုင္ခင္မွာ အဲဒီ function ေတြကို declare လုပ္ထားရမယ္ဆိုတဲ႔ အခ်က္ပဲ ျဖစ္ပါတယ္ခင္ဗ်ာ။

ေနာက္အပတ္ေတြမွာလဲ က်န္ရွိေနတဲ႔ function အေၾကာင္းကိုပဲ ဆက္လက္ ေဆြးေႏြးသြားပါမယ္ခင္ဗ်ာ။ ဤအခန္းဆက္ ေဆာင္းပါးမ်ားဟာ IT ေက်ာင္းသားမ်ားအတြက္ အထူးရည္ရြယ္ၿပီး ေရးသားထားတာ ျဖစ္ပါတယ္။ သင္ရိုးကို ေရးသားျခင္း ျဖစ္တဲ႔အတြက္ တိုက္ရိုက္ ဘာသာျပန္ဆိုျခင္းနည္းကို ေနရာေတာ္ေတာ္မ်ားမ်ားမွာ သံုးထားပါတယ္။ သိလိုသည္မ်ားကို aungwh2013@gmail.com ကို ဆက္သြယ္ၿပီး ေမးျမန္းႏိုင္သလို YCC ေက်ာင္းသားမ်ား 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://engineer4myanmar.blogspot.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕