**OOP in C++ : အပိုင္း (၂၉) - Functions (7)**

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

**Overloaded Functions**

Overloaded function မ်ားသည္ ၄င္းတို႔အား ေပးပို႔လိုက္ေသာ အခ်က္အလက္မ်ား (input parameter(s)) မ်ားအေပၚ မူတည္၍ မတူညီေသာ လုပ္ေဆာင္ခ်က္မ်ားကို ျပဳလုပ္ေပးသည္။ Oveloading သည္ ကၽြန္ေတာ္ ၾကားဘူးသည္႔ ဟာသ တစ္ခုႏွင္႔ တူညီပါသည္။ ေက်ာ္ၾကားေသာ သိပၸံပညာရွင္ႀကီး တစ္ေယာက္က ဓာတ္ဗူးကို အႀကီးက်ယ္ဆံုးေသာ တီထြင္မႈဟု အေၾကာက္အကန္ ျငင္းခံုပါသည္။ ၄င္း၏ အဆိုမွာ ဓာတ္ဗူးသည္ ပူေသာ အရာမ်ားကို ပူေအာင္ ထိန္းထားႏိုင္ၿပီး ေအးေသာအရာမ်ားကို ဆက္ေအးေနေအာင္ ထိန္းထားႏိုင္စြမ္း ရွိေသာေၾကာင္႔ အလြန္ထူးဆန္း အံ႔ၾသဖြယ္ရာပင္ ျဖစ္သည္ ဟုဆိုသည္။ ဓာတ္ဗူးက ဘယ္လိုလုပ္ၿပီး သိသလဲဟု ၄င္းက ေစာဒက တက္ခဲ႔ပါသည္။ overloaded function သည္လည္း ထိုကဲ႔သို႔ပင္ အံ႔ၾသဖြယ္ရာ သိရွိၿပီး လုပ္ေဆာင္ေပးႏိုင္ပါသည္။ ၄င္းသည္ ပထမ အခ်က္အလက္ တစ္ခုအတြက္ လုပ္ေဆာင္မႈ တစ္ခုကို လုပ္ေပးၿပီး အျခားအခ်က္အလက္အတြက္ သက္ဆိုင္ရာ ေနာက္လုပ္ေဆာင္မႈ တစ္ခုကို ျပဳလုပ္ေပးႏိုင္ပါသည္။ အဆိုပါ သေဘာသဘာဝကို ဥပမာမ်ားျဖင္႔ ရွင္းလင္းသြားမည္ ျဖစ္ပါသည္။

**Different Numbers of Arguments**

ယခင္ ဥပမာမ်ား (table.cpp ႏွင္႔ tablearg.cpp) တြင္ အသံုးျပဳခဲ႔ေသာ functions မ်ားကို ျပန္လည္ ေလ႔လာၾကည္႔ၾကပါစို႔။ starline() function သည္ ၾကယ္ ၄၅ လံုးကို print ထုတ္ေပးသည္။ repchar() function တြင္မူ ပံုေသ သတ္မွတ္ထားျခင္း မရွိဘဲ ေခၚယူေသာ program မွ ထည္႔သြင္းေပးလိုက္သည္႔ character ႏွင္႔ အလံုးအေရအတြက္ကို print ထုတ္ေပးမည္ျဖစ္သည္။ အကယ္၍ charline() function ကို ေရးသားကာ calling program မွ ထည္႔သြင္းေပးလိုက္ေသာ character အလံုးေရ ၄၅ လံုးကို print ထုတ္ေပးမည္ ဆိုၾကပါစို႔။ အဆိုပါ starline(), repchar(), ႏွင္႔ charline() function သံုးမ်ိဳးစလံုးသည္ ဆင္တူေသာ လုပ္ေဆာင္ခ်က္ကို လုပ္ေဆာင္ၾကၿပီး အမည္ မတူညီသည္ကို သတိထားမိပါလိမ္႔မည္။ ပရိုဂရမ္မာမ်ား အေနျဖင္႔ အဆိုပါ functions မ်ားကို အသံုးျပဳရန္ နာမည္ သံုးခုကို မွတ္သားထားရန္ လိုၿပီး ေရးသားေနေသာ ပရိုဂရမ္အတြက္ စုစည္းထားေသာ *Function Reference* documentation တြင္ အကၡစဥ္အလိုက္ စီစဥ္ထားေသာ အထဲမွ အမည္သံုးခုစာ သံုးေနရာ လိုက္လံ ရွာေဖြေနရမည္ ျဖစ္သည္။ အကယ္၍ လုပ္ေဆာင္ခ်က္ ဆင္တူေသာ functions မ်ားကို input arguments အေရအတြက္ႏွင္႔ data types မ်ား မတူညီသည္႔တိုင္ေအာင္ နာမည္တူ ေပးခဲ႔လွ်င္ အသံုးျပဳမည္႔ ပရိုဂရမ္မာအတြက္ ပိုမို အဆင္ေျပႏိုင္မည္ ျဖစ္သည္။ ေအာက္ပါ overload.cpp ပရိုဂရမ္တြင္ ေလ႔လာၾကည္႔ၾကပါစို႔။

// overload.cpp

// demonstrates function overloading

#include <iostream>

using namespace std;

void repchar(); //declarations

void repchar(char);

void repchar(char, int);

int main()

{

repchar();

repchar(‘=’);

repchar(‘+’, 30);

return 0;

}

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

// repchar()

// displays 45 asterisks

void repchar()

{

for(int j=0; j<45; j++) // always loops 45 times

cout << ‘\*’; // always prints asterisk

cout << endl;

}

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

// repchar()

// displays 45 copies of specified character

void repchar(char ch)

{

for(int j=0; j<45; j++) // always loops 45 times

cout << ch; // prints specified character

cout << endl;

}

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

// repchar()

// displays specified number of copies of specified character

void repchar(char ch, int n)

{

for(int j=0; j<n; j++) // loops n times

cout << ch; // prints specified character

cout << endl;

}

၄င္းပရိုဂရမ္ကို စမ္းသပ္ၾကည္႔လွ်င္ ေအာက္ပါအတိုင္း အကၡရာမ်ဥ္း သံုးေၾကာင္းကို ေတြ႔ရမည္ ျဖစ္ပါသည္။

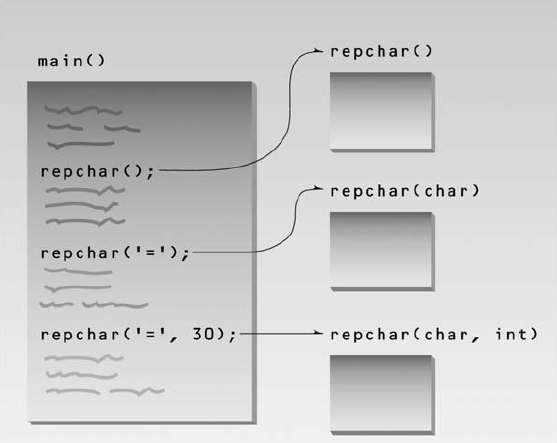
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=============================================

++++++++++++++++++++++++++++++

ပထမ ႏွစ္ေၾကာင္းသည္ အကၡရာ ၄၅ လံုးရွည္လ်ားၿပီး ေနာက္ဆံုး တတိယ အေၾကာင္းတြင္ အလံုးေရ ၃၀ ပါဝင္ပါသည္။ အဆိုပါ ပရိုဂရမ္တြင္ အမည္တူေသာ functions သံုးခု ပါဝင္ပါသည္။ function declarations သံုးခု၊ function calls သံုးခုႏွင္႔ function definitons သံုးခုတို႔ ပါဝင္ပါသည္။ ၄င္းတို႔ကို compiler က ခြဲျခားသိႏိုင္ျခင္းမွာ ၄င္းတို႔တြင္ ပါဝင္ေသာ မတူညီသည္႔ arguments အေရအတြက္ ႏွင္႔ မတူညီေသာ data types မ်ားပင္ ျဖစ္ပါသည္။

ေနာက္တစ္နည္းဆိုရလွ်င္ void repchar(); သည္ argument လံုးဝ မရွိဘဲ၊ char type argument တစ္ခုပါဝင္ေသာ void repchar(char); ႏွင္႔ char type argument တစ္ခုႏွင္႔ int type argument တစ္ခု ပါဝင္ေသာ void repchar(char, int); functions မ်ားႏွင္႔ မူညီဘဲ ကြဲျပားျခားနားေပသည္။ ၄င္းအခ်က္ကို compiler က ခြဲျခားႏိုင္စြမ္းရွိသျဖင္႔ အမည္တူေသာ္လည္း သက္ဆိုင္ရာ function ကိုသာ ေခၚယူ လုပ္ေဆာင္ေပးမည္ ျဖစ္သည္။



(ပံု-၁) function overloading ျပဳလုပ္ပံု

**Different Kinds of Arguments**

overload.cpp ပရိုဂရမ္တြင္ input arguments အေရအတြက္ မတူေသာ္လည္း အမည္တူသည္႔ functions သံုးခုကို ေရးသားထားပါသည္။ Compiler အေနျဖင္႔ arguments အေရအတြက္ မတူညီမႈျဖင္႔ ၄င္းတို႔အား ခြဲျခားၿပီး လုပ္ေဆာင္ေပးႏိုင္ခဲ႔ပါသည္။ သို႔ရာတြင္ Compiler အေနျဖင္႔ argument အေရအတြက္ တူညီေသာ္လည္း data type မတူညီျခင္းကို အေျခခံၿပီး ခြဲျခားေပးႏိုင္စြမ္းလည္း ရွိပါသည္။ overengl.cpp တြင္ ေလ႔လာၾကည္႔ႏိုင္ပါသည္။ ၄င္းတြင္ function အတြင္း ထည္႔သြင္း ေပးရမည္႔ တစ္ခုတည္းေသာ argument ကို structure type Distance ျဖစ္ေစ သို႔မဟုတ္ simple variable float type ျဖစ္ေစ အသံုးျပဳႏိုင္ပါသည္။ Compiler မွ ထည္႔သြင္းေပးလိုက္သည္႔ argument ၏ type ေပၚ မူတည္ၿပီး သက္ဆိုင္ရာ function ကို ေခၚယူေပးမည္ ျဖစ္ပါသည္။

// overengl.cpp

// demonstrates overloaded functions

#include <iostream>

using namespace std;

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

struct Distance //English distance

{

int feet;

float inches;

};

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

void engldisp( Distance ); //declarations

void engldisp( float );

int main()

{

Distance d1; //distance of type Distance

float d2; //distance of type float

//get length d1 from user

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

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

//get length d2 from user

cout << “Enter entire distance in inches: “; cin >> d2;

cout << “\nd1 = “;

engldisp(d1); //display length 1

cout << “\nd2 = “;

engldisp(d2); //display length 2

cout << endl;

return 0;

}

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

// engldisp()

// display structure of type Distance in feet and inches

void engldisp( Distance dd ) //parameter dd of type Distance

{

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

}

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

// engldisp()

// display variable of type float in feet and inches

void engldisp( float dd ) //parameter dd of type float

{

int feet = static\_cast<int>(dd / 12);

float inches = dd - feet\*12;

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

}

အထက္ပါ ပရိုဂရမ္တြင္ user အား distances တန္ဖိုး ႏွစ္ခုကို ထည္႔သြင္းေစပါသည္။ ပထမ အႀကိမ္တြင္ ေပ ႏွင္႔ လက္မ ကို ခြဲျခား ထည္႔သြင္းေစၿပီး ဒုတိယ အႀကိမ္တြင္ လက္မဖြဲ႕ၿပီးသား အတိုင္းအတာကိုသာ ထည္႔သြင္းေစပါသည္(ဥပမာ - 9'–1.5'' အစား 109.5 inches ဟု ထည္႔သြင္းေပးရန္ ျဖစ္သည္)။ ပရိုဂရမ္မွ ပထမအႀကိမ္တြင္ Distance type argument အသံုးျပဳထားသည္႔ engldisp(Distance) function ကို ေခၚယူၿပီး ဒုတိယအႀကိမ္တြင္ float type argument ကို အသံုးျပဳထားေသာ engldisp(float) function ကို ေခၚယူမည္ ျဖစ္ပါသည္။ ေအာက္ပါအတိုင္း နမူနာ စမ္းသပ္ အသံုးျပဳျပထားပါသည္။

Enter feet: 5

Enter inches: 10.5

Enter entire distance in inches: 76.5

d1 = 5’-10.5”

d2 = 6’-4.5”

သတိျပဳရမည္႔ အခ်က္တစ္ခုမွာ engldisp() functions ႏွစ္ခုစလံုးသည္ ဆင္တူေသာ လုပ္ေဆာင္မႈ ရွိေသာ္လည္း ကုဒ္မ်ားမွာ လံုးဝ ကြဲျပားျခားနားေပသည္။ လက္မဖြဲ႕ၿပီးသား တန္ဖိုးကို အသံုးျပဳသည္႔ function သည္ ေပ ႏွင္႔ လက္မ ျပန္ဖြဲ႕ၿပီးမွ ရလဒ္ကို ျပသေပးမည္ ျဖစ္ပါသည္။

Overloaded functions can simplify the programmer’s life by reducing the number of function

names to be remembered. As an example of the complexity that arises when overloading is not

used, consider the C++ library routines for finding the absolute value of a number.

Overloaded functions မ်ားကို အသံုးျပဳျခင္းျဖင္႔ ပရိုဂရမ္မာမ်ားသည္ function အမည္မ်ားစြာကို မွတ္သားရန္ မလိုေတာ႔သျဖင္႔ ေရးသားရသည္မွာ ပိုမိုလြယ္ကူလာေပသည္။ အကယ္၍ overloaded function အသံုးမျပဳခဲ႔လွ်င္ function နာမည္မ်ားစြာကို မွတ္သားရသျဖင္႔ မွားယြင္းႏိုင္ေပသည္။ ဥပမာအားျဖင္႔ absolute value ကို ရွာရန္ ေရးသားထားသည္႔ functions မ်ားသည္ C programming အတြက္ အမ်ိဳးမ်ိဳး ကြဲျပားေနေပသည္။ int data type အတြက္ abs()၊ complex numbers အတြက္ cabs()၊ type double အတြက္ fabs() ႏွင္႔ type long အတြက္ labs() ဟူ၍ အသီးသီး ခြဲျခားေရးသားထားရေပသည္။ C++ တြင္မူ abs() တစ္ခုတည္းျဖင္႔ ႀကိဳက္ႏွစ္သက္ရာ data type ကို ထည္႔သြင္းၿပီး အသံုးျပဳႏိုင္ပါသည္။ Function overloading ကို objects မ်ားအေၾကာင္း ေလ႔လာရာတြင္ ဆက္လက္ ရွင္းျပပါဦးမည္။

ေ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။

**မွီျငမ္း**

* 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