**OOP in C++ : အပိုင္း (၂၅) - Functions (3)**

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

**Passing Variables**

ယခင္ အပတ္မွာ ေလ႔လာခဲ႔ၾကတဲ႔ tablearg.cpp ပရိုဂရမ္ထဲမွာ constant arguments (‘–’, 43) ေတြကို အသံုးျပဳခဲ႔ပါတယ္။ ယခုအပတ္မွာေတာ႔ constants ေတြအစား variables ေတြကို arguments မ်ား အျဖစ္ အသံုးျပဳျခင္းကို ေလ႔လာသြားမွာပါ။ ေအာက္ပါ vararg.cpp ပရိုဂရမ္ေလးမွာေတာ႔ tablearg.cpp မွာတုန္းကလိုပဲ repchar() function ပါဝင္မွာ ျဖစ္ပါတယ္။ ဒါေပမယ္႔ တစ္ခု ပိုထူးျခားလာတာကေတာ႔ user က ေရြးခ်ယ္ေပးလိုက္တဲ႔ character နဲ႔ အႀကိမ္အေရအတြက္ကို အသံုးျပဳသြားမွာပဲ ျဖစ္ပါတယ္။

// vararg.cpp

// demonstrates variable arguments

#include <iostream>

using namespace std;

void repchar(char, int); //function declaration

int main()

{

char chin;

int nin;

cout << “Enter a character: “;

cin >> chin;

cout << “Enter number of times to repeat it: “;

cin >> nin;

repchar(chin, nin);

return 0;

}

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

// repchar()

// function definition

void repchar(char ch, int n) //function declarator

{

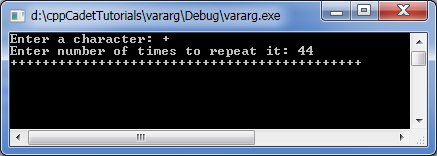
for(int j=0; j<n; j++) //function body

cout << ch;

cout << endl;

}

vararg.cpp ရဲ႕ နမူနာ output ကို ေအာက္မွာ ေလ႔လာႏိုင္ပါတယ္။



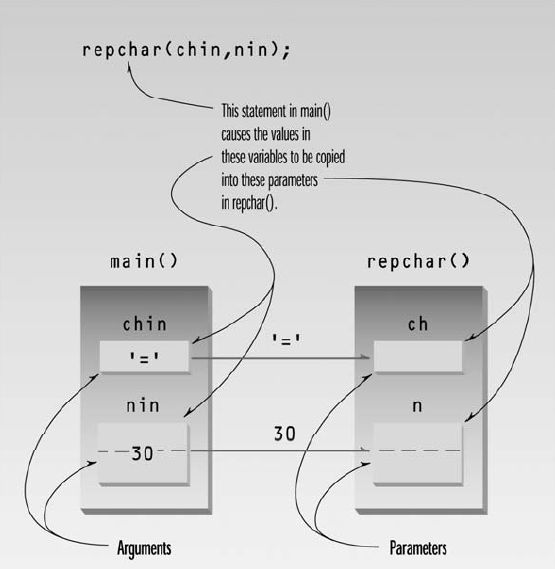
repchar() အတြက္ input arguments မ်ားအေနနဲ႔ main() ထဲက chin နဲ႔ nin တို႔ကို ေအာက္ပါအတိုင္း အသံုးျပဳထားပါတယ္။

repchar(chin, nin); // function call

variables ေတြအတြက္ အသံုးျပဳထားတဲ႔ arguments ေတြရဲ႕ data types ေတြဟာ constants ေတြ အသံုးျပဳစဥ္ကလိုပဲ function declaration နဲ႔ definition ေတြမွာ တူညီေနဖို႔ လိုအပ္ပါတယ္။ ဒါေၾကာင္႔ chin ဟာ char ျဖစ္ဖို႔လိုၿပီး nin ကေတာ႔ int ျဖစ္ဖို႔ လိုပါတယ္။

**Passing by Value**

vararg.cpp ပရိုဂရမ္မွာ chin နဲ႔ nin အတြင္း ရွိတဲ႔ တန္ဖိုးမ်ားကို function call ျပဳလုပ္တဲ႔ အခါမွာ function ထဲကို pass ျပဳလုပ္ေပးမွာ ျဖစ္ပါတယ္။ function ထဲတြင္ ၄င္း တန္ဖိုးမ်ားကို သိမ္းဆည္းအသံုးျပဳရန္ variables အသစ္မ်ား ဖန္တီးေပးမည္ ျဖစ္သည္။ ထိုသို႔ ျပဳလုပ္ရန္ function ၏ declarator ျဖစ္ေသာ void repchar(char ch, int n) ထဲတြင္ သတ္မွတ္ေပးထားေသာ data type မ်ားႏွင္႔ အမည္မ်ား (char ch, int n) ကို အသံုးျပဳထားျခင္းပင္ ျဖစ္သည္။ function အတြင္း pass ျပဳလုပ္လိုက္ေသာ တန္ဖိုးမ်ားကို အဆိုပါ parameters မ်ားအတြင္း initialize ျပဳလုပ္၍ ထည္႔သြင္းေပးလိုက္ျခင္းပင္ ျဖစ္သည္။ ထို႔ေနာက္တြင္ေတာ႔ ၄င္း တန္ဖိုးမ်ားကို ပံုမွန္ variables မ်ား အသံုးျပဳသည္႔အတိုင္း function body အတြင္း အသံုးျပဳႏိုင္ၿပီ ျဖစ္သည္။ ထိုကဲ႔သို႔ passing ျပဳလုပ္လိုက္ေသာ arguments မ်ားကို function အတြင္း ကူးယူဖန္တီးသည္႔ နည္းလမ္းကို passing by value ဟု ေခၚဆိုၾကသည္။ ေနာက္ထပ္ နည္းလမ္းတစ္ခုမွာ passing by reference ဟု ေခၚၿပီး ေနာက္ပိုင္း သင္ခန္းစာမ်ားတြင္ ဆက္လက္ ေလ႔လာသြားၾကမည္ ျဖစ္သည္။ Passing by value ျဖစ္စဥ္ကို ေအာက္ပါပံုတြင္ အေသးစိတ္ ေလ႔လာႏိုင္ပါသည္။



**Structures as Arguments**

function အတြင္းသို႔ တန္ဖိုးမ်ားကို တစ္ခုခ်င္း ထည္႔သြင္းေပးႏိုင္သလို structures မ်ားကို အသံုးျပဳ၍ အစုလိုက္ အျပံဳလိုက္လည္း ထည္႔သြင္းေပးႏိုင္ေပသည္။ နမူနာ ပရိုဂရမ္မ်ားျဖင္႔ ဆက္လက္၍ ေလ႔လာၾကရေအာင္။

**Passing a Distance Structure**

ေအာက္ပါ ဥပမာတြင္ ယခင္က ေလ႔လာခဲ႔ဘူးေသာ Distance structure type ကို input argument အေနျဖင္႔ အသံုးျပဳထားသည္။ engldisp.cpp ကို ေလ႔လာၾကည္႔ၾကရေအာင္ -

// engldisp.cpp

// demonstrates passing structure as argument

#include <iostream>

using namespace std;

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

struct Distance //English distance

{

int feet;

float inches;

};

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

void engldisp( Distance ); //declaration

int main()

{

Distance d1, d2; //define two lengths

//get length d1 from user

cout << “Enter 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;

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

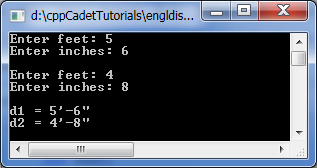
void engldisp( Distance dd ) //parameter dd of type Distance

{

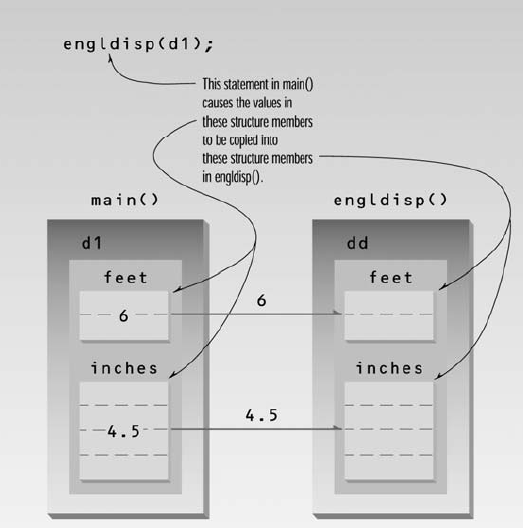
cout << dd.feet << “\’-” << dd.inches << “\””;

}

main() function ထဲတြင္ user ထည္႔သြင္းလိုက္ေသာ distances တန္ဖိုးမ်ားကို feet-and-inches ပံုစံျဖင္႔ လက္ခံၿပီး d1 ႏွင္႔ d2 ဟူေသာ structure ႏွစ္ခုအတြင္း ထည္႔သြင္းလိုက္သည္။ ထို႔ေနာက္ engldisp() function ကို ေခၚယူကာ d1 ႏွင္႔ d2 ကို input argument မ်ားအျဖစ္ ထည္႔သြင္းေပးလိုက္သည္။ ၄င္း function ၏ ရည္ရြယ္ခ်က္မွာ စံျပဳပံုစံ ျဖစ္ေသာ 10'–2.25'' ကဲ႔သို႔ ျပသေပးရန္ ျဖစ္သည္။ ေအာက္တြင္ အဆိုပါ ပရိုဂရမ္၏ output ကို နမူနာ ျပသထားသည္။



main() အတြင္းရွိ function calls မ်ား၊ function declaration မ်ား ႏွင္႔ function body ရွိ declarator မ်ားသည္ input ထည္႔သြင္းေပးလိုက္ေသာ Distance type structure variables မ်ားကို အျခား int, char တို႔ကဲ႔သို႔ေသာ argument variables မ်ားနည္းတူ ျပဳမူေဆာင္ရြက္ေပးသည္။ main() function ထဲတြင္ engldisp() function ကို ႏွစ္ႀကိမ္ေခၚယူထားသည္။ ပထမတစ္ႀကိမ္တြင္ structure d1 အား ထည္႔သြင္းေပးၿပီး ဒုတိယအႀကိမ္တြင္ d2 အားထည္႔သြင္းေပးသည္။ engldisp() function ထဲတြင္ Distance type structure - dd ကို parameter အျဖစ္ အသံုးျပဳထားၿပီး main() function မွ ထည္႔သြင္းေပးလိုက္ေသာ structure value မ်ားကို သာမာန္ argument value မ်ားကဲ႔သို႔ပင္ dd ထဲသို႔ အလိုအေလ်ာက္ ထည္႔သြင္းေပးသည္။ ထို႔ေၾကာင္႔ engldisp() function ထဲမွ statements မ်ားသည္ dd.feet ႏွင္႔ dd.inches ဆိုသည္႔ ပံုမွန္ structure အသံုးျပဳပံု နည္းလမ္းမ်ားအတိုင္း အဆိုပါ value မ်ားကို ရယူ အသံုးျပဳႏိုင္ေပသည္။ ေအာက္ပါပံုတြင္ function တစ္ခုအတြင္းသို႔ structure တစ္ခုအား argument အေနျဖင္႔ ထည္႔သြင္းအသံုးျပဳပံုကို ရွင္းလင္းထားသည္။



ပံုမွန္ variables မ်ားကဲ႔သို႔ပင္ engldisp() function အတြင္းရွိ structure parameter - dd ကို ေျပာင္းလဲျခင္းသည္ ၄င္းအား main() function မွ ထည္႔သြင္းေပးလိုက္ေသာ structures -d1 ႏွင္႔ d2 တို႔အား ေျပာင္းလဲေစျခင္း မရွိေပ။ သီးသန္႔ ျဖစ္ေနေပသည္။ ဥပမာ -

dd.feet = 2;

dd.inches = 3.25;

တို႔သည္ main() အတြင္းရွိ d1 ႏွင္႔ d2 တို႔၏ တန္ဖိုးမ်ားအား မေျပာင္းလဲ ေစႏိုင္ေပ။

ေနာက္အပတ္ေတြမွာလဲ 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