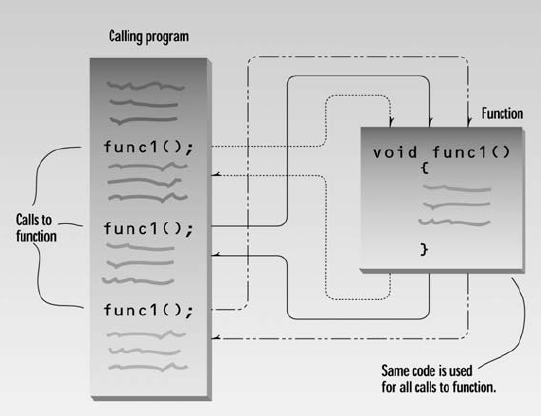
**OOP in C++ : အပိုင္း (၂၃) - Functions (1)**

Function ဆိုသည္မွာ ပရိုဂရမ္ statements ကုဒ္ အခ်ိဳ႕ကို ယူနစ္ တစ္ခုအျဖစ္ စုစည္းၿပီး နာမည္တစ္ခု ေပးထားျခင္းပင္ ျဖစ္သည္။ အဆိုပါ ယူနစ္မ်ားကို ပရိုဂရမ္၏ အျခားအစိတ္အပိုင္းမ်ားမွ ေခၚယူ အသံုးျပဳႏိုင္ပါသည္။ Functions မ်ားကို အသံုးျပဳရျခင္း၏ အဓိက အေၾကာင္းရင္းမွာ ပရိုဂရမ္တစ္ပုဒ္အား conceptual organization ျပဳလုပ္ရာတြင္ အေထာက္အကူျပဳရန္ပင္ ျဖစ္သည္။ ထို႔ျပင္ ပရိုဂရမ္ တစ္ပုဒ္အား functions မ်ားအျဖစ္ ခြဲျခား ေရးသားျခင္းသည္ structured programming ၏ အဓိက ဥပေဒသ တစ္ခုလည္း ျဖစ္ပါသည္။ (သို႔ရာတြင္ object-oriented programming တြင္ ၄င္းထက္ ပိုမို အားေကာင္းေသာ နည္းလမ္းမ်ားကို အသံုးျပဳႏိုင္ပါသည္။)

ေနာက္ထပ္ ရည္ရြယ္ခ်က္ တစ္ခုမွာ ပရိုဂရမ္၏ အရြယ္အစားကို ေလ်ာ႔ခ်ရန္ ျဖစ္သည္။ တစ္ႀကိမ္ထက္ ပိုသံုးထားေသာ ကုဒ္မ်ားကို function အျဖစ္ ေျပာင္းလဲ အသံုးျပဳသင္႔ေပသည္။ Functions မ်ားကို ပရိုဂရမ္အတြင္း အႀကိမ္မ်ားစြာ ေခၚယူ အသံုးျပဳေလ႔ ရွိေသာ္လည္း function ၏ ကုဒ္မ်ားကိုမူ memory ၏ ေနရာ တစ္ခုတည္းတြင္သာ သိမ္းဆည္းထားျခင္း ျဖစ္သည္။ ပရိုဂရမ္၏ အစိတ္အပိုင္း အသီးသီးမွ function တစ္ခုအား ေခၚယူ အသံုးျပဳပံုကို ေအာက္ပါအတိုင္း သရုပ္ေဖာ္ျပသထားပါသည္။



C++ ႏွင္႔ C တြင္ အသံုးျပဳရေသာ functions မ်ားသည္ အျခား ပရိုဂရမ္းမင္း ဘာသာစကားမ်ားရွိ subroutines၊ procedures မ်ားႏွင္႔ သေဘာတရားခ်င္း တူညီၿပီး ေရးသားပံု ဆင္တူေပသည္။

**Simple Functions**

ပထမဦးစြာ ၾကယ္ ၄၅ လံုး print ထုတ္ေပးေသာ function တစ္ခုကို စတင္ ေလ႔လာၾကရေအာင္။ ေအာက္ပါ table.cpp ပရိုဂရမ္သည္ ဇယားတစ္ခုကို ဖန္တီးေပးၿပီး အဆိုပါ ဇယားကို ပိုမို ဖတ္ရႈရလြယ္ကူေစရန္ ၾကယ္မ်ားျဖင္႔ ျပဳလုပ္ထားေသာ လိုင္းမ်ားျဖင္႔ တားေပးထား ပါသည္။

// table.cpp

// demonstrates simple function

#include <iostream>

#include <conio.h>

using namespace std;

void starline(); //function declaration

// (prototype)

int main()

{

starline(); //call to function

cout << “Data type Range” << endl;

starline(); //call to function

cout << “char -128 to 127” << endl

<< “short -32,768 to 32,767” << endl

<< “int System dependent” << endl

<< “long -2,147,483,648 to 2,147,483,647” << endl;

starline(); //call to function

\_getch();

return 0;

}

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

// starline()

// function definition

void starline() //function declarator

{

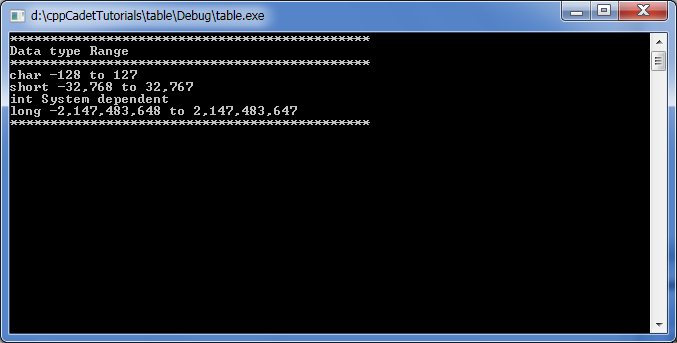
for(int j=0; j<45; j++) //function body

cout << ‘\*’;

cout << endl;

}

ပရိုဂရမ္၏ output မွာ ေအာက္ပါအတိုင္း ျဖစ္ပါသည္။



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

Data type Range

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

char -128 to 127

short -32,768 to 32,767

int System dependent

long -2,147,483,648 to 2,147,483,647

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

အထက္ပါ ပရိုဂရမ္တြင္ main() ႏွင္႔ starline() ဟူေသာ functions ႏွစ္ခုပါဝင္ပါသည္။ ကၽြန္ေတာ္တို႔ အေနျဖင္႔ ယခင္ သင္ခန္းစာမ်ားတြင္ main() function တစ္ခုတည္းကိုသာ ႀကိမ္ဖန္မ်ားစြာ ေတြ႔ျမင္လာခဲ႔ၾကရပါသည္။ ထိုသို႔ ပရိုဂရမ္ တစ္ပုဒ္တြင္ functions မ်ား ထပ္မံေပါင္းထည္႔ရန္ လိုသည္႔ အခ်က္ သံုးခ်က္ ရွိပါသည္။ ၄င္းတို႔မွာ function declaration, calls to the function ႏွင္႔ function definition မ်ား ျဖစ္ၾကပါသည္။

**The Function Declaration**

ကၽြန္ေတာ္တို႔အေနျဖင္႔ compiler အား variable တစ္ခုကို ႀကိဳတင္ေၾကျငာထားျခင္း မရွိဘဲ အသံုးမျပဳႏိုင္သကဲ႔သို႔ function တစ္ခုကိုလည္း ႀကိဳတင္ေၾကျငာေပးထားရန္ လိုအပ္ေပသည္။

ထိုသို႔ ျပဳလုပ္ႏိုင္သည္႔ နည္းလမ္း ႏွစ္ခု ရွိပါသည္။ ပထမတစ္နည္းမွာ function အား မေခၚယူမီ declare ျပဳလုပ္ျခင္းျဖစ္ပါသည္။ (ဒုတိယနည္းလမ္းမွာ function အားမေခၚယူမီ define ျပဳလုပ္ျခင္း ျဖစ္ၿပီး ၄င္းကို ေနာက္ပိုင္းတြင္ ဆက္လက္ေလ႔လာသြားမည္။)

table.cpp ပရိုဂရမ္တြင္ starline() ဆိုေသာ function ကို void starline(); ဆိုေသာ ကုဒ္ျဖင္႔ declare ေၾကျငာေပးထားသည္။ ဤနည္းျဖင္႔ ေနာင္တစ္ခ်ိန္တြင္ starline ဟုေခၚေသာ function တစ္ခုကို ေခၚယူရန္ စီစဥ္ထားေၾကာင္း compiler အား အသိေပးလိုက္ျခင္းပင္ ျဖစ္သည္။

ထိုသို႔ေၾကျငာရာတြင္ ပါဝင္ေသာ void ဆိုသည္႔ keyword သည္ function မွ return value ျပန္ေပးမည္ မဟုတ္ေၾကာင္း သတ္မွတ္ေပးထားျခင္းျဖစ္ၿပီး လက္သဲကြင္းထဲတြင္ အလြတ္ထားျခင္းသည္ arguments ထည္႔သြင္းေပးရန္ မလိုေၾကာင္း (လက္ခံမည္မဟုတ္ေၾကာင္း) အသိေပးထားျခင္းပင္ ျဖစ္သည္။ (ေနာက္တစ္နည္းမွာ လက္သဲကြင္းထဲတြင္ void ထည္႔ေပးျခင္းျဖစ္ၿပီး C တြင္ အသံုးမ်ားသည္။ သို႔ရာတြင္ C++ ၌မူ လက္သဲကြင္း အလြတ္ထားေလ႔ရွိသည္။) ေနာက္ပိုင္းတြင္ arguments မ်ားနွင္႔ return values မ်ား အေၾကာင္းကို ေလ႔လာသြားမည္ျဖစ္သည္။ function declaration ကို semicolon (;) ျဖင္႔ အဆံုးသတ္ေလ႔ရွိၿပီး ၄င္းစာေၾကာင္းသည္ ၿပီးျပည္႔စံုသည္႔ statement တစ္ခုပင္ ျဖစ္သည္။

Function declarations မ်ားသည္ functions မ်ား၏ model သို႔မဟုတ္ blueprint သဖြယ္ ျဖစ္ေနသျခင္႔ ၄င္းတို႔ကို prototypes မ်ားဟုလည္း ေခၚဆိုၾကသည္။ ၄င္းတို႔သည္ ဤကဲ႔သို႔ ပံုစံရွိေသာ function တစ္ခုသည္ ပရိုဂရမ္ ၏ တစ္ေနရာတြင္ ေရးသားထားမည္ ျဖစ္ေသာေၾကာင္႔ ၄င္းအား ေခၚဆို အသံုးခ်မႈမ်ားကို အဆိုပါ function အားေရးသားထားခ်က္ မတိုင္မီ ေတြ႔ရွိရပါက နားလည္ေစရန္ compiler အား အေၾကာင္းၾကားလိုက္ျခင္းပင္ ျဖစ္သည္။ အဆိုပါ declaration တြင္ ပါဝင္ေသာ return type၊ arguments အေရအတြက္ႏွင္႔ types မ်ား စသည္႔ အခ်က္အလက္မ်ားကို function ၏ signature မ်ားဟုလည္း ေခၚဆိုေလ႔ ရွိသည္။

**Calling the Function**

အထက္ပါ ပရိုဂရမ္တြင္ starline() function အား main() function အတြင္း သံုးႀကိမ္ေခၚယူ အသံုးျပဳထားေပသည္။ ထိုသို႔ ေခၚယူရန္ starline(); ဟူေသာ ကုဒ္ကို အသံုးျပဳခဲ႔သည္။ Function တစ္ခုအား ေခၚယူရန္မွာ အဆိုပါ function ၏ အမည္ေနာက္တြင္ လက္သဲကြင္းတစ္စံု ႏွင္႔ ၄င္းတို႔ကို semicolon ျဖင္႔ ပိတ္ေပးရန္သာ ျဖစ္သည္။ Function call ၏ syntax သည္ function declaration ၏ syntax ႏွင္႔ ဆင္တူၿပီး return type မပါျခင္းသာ ကြာျခားမႈ ရွိသည္။

void starline(); //function declaration

starline(); //function calling

Call statement ကို လုပ္ေဆာင္သည္႔အခါ control သည္ function သို႔ ေရာက္ရွိသြားၿပီး (ေအာက္တြင္ ဆက္လက္ ရွင္းျပသြားမည္႔) function definition ရွိ ကုဒ္မ်ားကို လုပ္ေဆာင္မည္ျဖစ္သည္။ ထုိ႔ေနာက္ control သည္ function call ၏ ေနာက္တြင္ ရွိေသာ ကုဒ္မ်ားသို႔ ျပန္လည္ေရာက္ရွိသြားမည္ ျဖစ္သည္။

**The Function Definition**

Function တစ္ခုအတြက္ တကယ္႔ကုဒ္မ်ား ေရးသားသည္႔ ေနရာကို function definition ဟုေခၚသည္။ ေအာက္တြင္ starline() function ၏ definition ကို ေလ႔လာၾကည္႔ရေအာင္-

void starline() //declarator

{

for(int j=0; j<45; j++) //function body

cout << ‘\*’;

cout << endl;

}

The definition consists of a line called the *declarator*, followed by the function *body*.

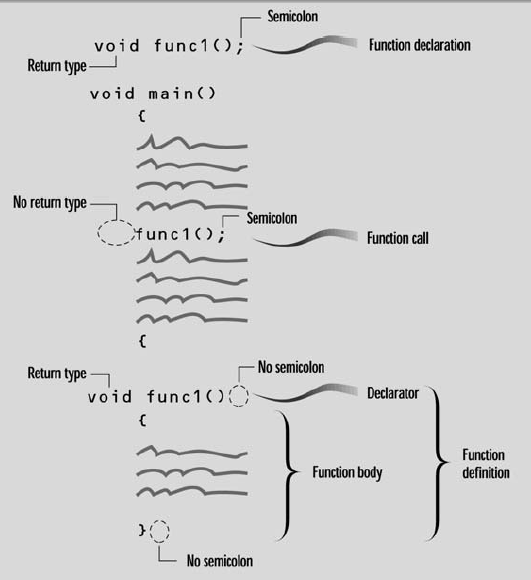
Function definition တြင္ declarator ဟုေခၚေသာ စာေၾကာင္းႏွင္႔ function body ဟူ၍ ပါဝင္သည္။

Function body သည္ function ၏ ကုဒ္မ်ား ပါဝင္ၿပီး တြန္႔ကြင္း အဖြင္႔အပိတ္ျဖင္႔ သတ္မွတ္ထားသည္။

The declarator တြင္ ပါဝင္ေသာ အခ်က္မ်ားသည္ declaration ႏွင္႔ ကိုက္ညီမႈ ရွိရမည္ ျဖစ္သည္။ ေအာက္ပါအခ်က္မ်ား တူညီရမည္ျဖစ္သည္-

1. function name
2. same argument types in the same order (if there are arguments)
3. return type.

declarator အားsemicolon လံုးဝ မထည္႔ေပးရေပ။ ေအာက္ပါပံုတြင္ function declaration, function call ႏွင္႔ function definition တို႔၏ syntax ကို ေလ႔လာႏိုင္ပါသည္။



Function တစ္ခုအား ေခၚယူလိုက္သည္႔အခါ control သည္ function body ရွိ ပထမဆံုး ကုဒ္ကို ကူးေျပာင္းသြားၿပီး စတင္လုပ္ေဆာင္မည္ ျဖစ္သည္။ function body ရွိ ကုဒ္မ်ား အားလံုး လုပ္ေဆာင္ၿပီး၍ တြန္႔ကြင္း အပိတ္အား ေတြ႔လိုက္သည္႔အခ်ိန္တြင္ control သည္ calling ျပဳလုပ္သည္႔ ေနရာသို႔ ျပန္ေရာက္သြားမည္ ျဖစ္သည္။

ေနာက္အပတ္ေတြမွာေတာ႔ function အေၾကာင္းကိုပဲ ဒုတိယပိုင္း အေနနဲ႔ 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)

aungwh2013@gmail.com

http://engineer4myanmar.blogspot.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕