**OOP in C++ : အပိုင္း (၁၈) - Structures**

ဒီအပတ္မွာေတာ႔ Loops and Desicions အေၾကာင္းကို အဆံုးသတ္တဲ႔ အေနနဲ႔ အနည္းငယ္ ေဆြးေႏြးၿပီး ေနာက္တစ္ခန္းျဖစ္တဲ႔ Structures ေတြအေၾကာင္းကို ဆက္လက္ ေလ႔လာသြားမွာ ျဖစ္ပါတယ္။ တကယ္ေတာ႔ ကၽြန္ေတာ္တို႔ အဲဒီအခန္းကို ေလ႔လာခဲ႔တာ အေတာ္ေလး ျပည္႔စံုသြားပါၿပီ။ ဒါေပမယ္႔ အသံုးမျပဳသင္႔တဲ႔ statement တစ္ခုကို ရွင္းျပဖို႔ က်န္ပါေသးတယ္။ ၄င္းကေတာ႔ goto ဆိုတဲ႔ statement ပဲ ျဖစ္ပါတယ္။ goto ကို အသံုးျပဳျခင္းအားျဖင္႔ နားလည္ရခက္ၿပီး ရႈပ္ေထြးတဲ႔ ကုဒ္ေတြ ျဖစ္လာေစႏိုင္ပါတယ္။ ဒါေၾကာင္႔ လံုးဝ အသံုးမျပဳၾကေစခ်င္ပါဘူး။ အသံုးျပဳနည္းကိုေတာ႔ တင္ျပလိုက္ပါတယ္။

ပထမဆံုး ကၽြန္ေတာ္တို႔ jump လုပ္ဖို႔ လိုအပ္တဲ႔ ကုဒ္ေနရာမွာ label တစ္ခုကို ေရးသားထားရမွာ ျဖစ္ပါတယ္။ Label ကို colon ":" နဲ႔ ပိတ္ေပးရမွာပါ။ ပရိုဂရမ္ရဲ႕ တေနရာရာမွာ goto ရဲ႕ ေနာက္ကို အဲဒီ label name ထည္႔ေပးလိုက္မယ္ဆိုရင္ runtime မွာ program control က အဲဒီေနရာေရာက္တာနဲ႔ label ထိုးထားတဲ႔ ေနရာကို jump လုပ္သြားမွာ ျဖစ္ပါတယ္။ ဥပမာ -

*goto SystemCrash;*

*// other statements*

*SystemCrash:*

*// control will begin here following goto*

ေရွ႕ပိုင္း သင္ခန္းစာမ်ားမွာ float, char, int အစရွိသျဖင္႔ ရိုးရွင္းတဲ႔ data type ေတြကို ေလ႔လာခဲ႔ၾကၿပီးပါၿပီ။ အဲဒီ data type ေတြ အားလံုးဟာ (အရပ္အျမင္႔႔၊ ကိုယ္အေလးခ်ိန္၊ ကားတန္ဖိုး စတဲ႔) သတင္းအခ်က္အလက္ တစ္ခုကိုပဲ သိမ္းဆည္းထားႏိုင္ပါတယ္။ အခ်က္အလက္ေတြကို အေျမာက္အမ်ား သိမ္းဆည္းခ်င္လာတဲ႔ အခါမွာ အဲဒီ data type ေတြဟာ မလံုေလာက္ေတာ႔ပါဘူး။ ဒါေၾကာင္႔ array, structure စတာေတြကို အသံုးခ်ၿပီး သိမ္းဆည္းဖို႔ လိုလာပါတယ္။ data type တူတဲ႔ ေဒတာ အမ်ားအျပား သိမ္းဆည္းဖို႔အတြက္ေတာ႔ ေနာက္ပိုင္းအခန္းေတြမွာ ေလ႔လာရမယ္႔ array ကို အသံုးျပဳရမွာ ျဖစ္ပါတယ္။ data type မတူညီတဲ႔ ေဒတာေတြကို သိမ္းဆည္းဖို႔အတြက္ေတာ႔ structure ကို အသံုးျပဳႏိုင္ပါတယ္။

**Structures**

Structure ဆိုတာ တကယ္ေတာ႔ ရိုးရိုး variable ေတြကို စုစည္းထားတာ ျဖစ္ပါတယ္။ အဲဒီ variables ေတြဟာ int, float, char စသျဖင္႔ types အမ်ိဳးမ်ိဳး ျဖစ္ႏိုင္ပါတယ္။ structure ထဲမွာ ပါတဲ႔ variables ေတြကို structure ရဲ႕ *members* ေတြလို႔ ေခၚပါတယ္။

C programming စာအုပ္ေတြမွာတုန္းကေတာ႔ structures ေတြကို advanced feature အျဖစ္ ယူဆတဲ႔အတြက္ စာအုပ္တိုင္းရဲ႕ ေနာက္ဆံုး အခန္းေတြမွာပဲ ေဖာ္ျပေလ႔ ရွိၾကပါတယ္။ C++ programmers ေတြအတြက္ကေတာ႔ structures ဆိုတာ objects နဲ႔ classes ေတြကို နားလည္ႏိုင္ဖို႔ အဓိက အေျခခံ အခ်က္ ႏွစ္ခုထဲက တစ္ခုပဲ ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ အရင္ဆံုး မိတ္ဆက္ ေဖာ္ျပေပးထားျခင္း ျဖစ္ပါတယ္။

တကယ္ေတာ႔ class နဲ႔ structure ေတြမွာ syntax ေတြ အေတာ္ေလး တူညီေနပါတယ္။ structure က ေတာ႔ အခ်က္အလက္ေတြကို စုထားတာျဖစ္ၿပီး class က အခ်က္အလက္ေတြ အျပင္ function ေတြ စုထားတာေလးပဲ ကြာျခားမႈ ရွိပါတယ္။ ဒါေၾကာင္႔ structure ကို နားလည္ထားျခင္းျဖင္႔ class နဲ႔ object ေတြရဲ႕ သေဘာတရားကို ေလ႔လာရတာ လြယ္ကူေစမွာ ျဖစ္ပါတယ္။ Pascal နဲ႔ အျခား language အခ်ိဳ႕မွာ ရွိတဲ႔ *records* ရဲ႕ သေဘာတရားနဲ႔ ရည္ရြယ္ခ်က္က C နဲ႔ C++ မွာရွိတဲ႔ structure ရဲ႕ သေဘာတရားနဲ႔ အတူတူပဲ ျဖစ္ပါတယ္။

**A Simple Structure**

Integer variables ႏွစ္ခုနဲ႔ floating point number variable တစ္ခု ပါဝင္တဲ႔ structure တစ္ခုကို စတင္ ဖန္တီးၾကရေအာင္။ အဲဒီလို structure မ်ိဳးဟာ ကုမၸဏီ တစ္ခုရဲ႕ စတိုးထဲက အစိတ္အပိုင္း ေတြကို ကိုယ္စားျပဳထားပါတယ္။ အစိတ္္အပိုင္း တစ္ခုစီအတြက္ လိုအပ္တဲ႔ အခ်က္အလက္ေတြကို သတ္မွတ္ေပးထားတဲ႔ blueprint လိုပါပဲ။ Structure ရဲ႕ ပထမ member ကေတာ႔ အဲဒီအစိတ္အပိုင္း ပါဝင္တဲ႔ ပစၥည္းရဲ႕ model number ပဲ ျဖစ္ပါတယ္။ ဒုတိယ member ကေတာ႔ အစိတ္အပိုင္း number ျဖစ္ၿပီး ေနာက္ဆံုးတစ္ခုက က်သင္႔ေငြ ျဖစ္ပါတယ္။ ေအာက္မွာ ေဖာ္ျပထားတဲ႔ ပရိုဂရမ္ parts.cpp မွာ part ဆိုတဲ႔ structure တစ္ခုကို define ျပဳလုပ္ထားပါတယ္။ ထို႔ေနာက္ အဲဒီ structure type အမ်ိဳးအစား variable တစ္ခု (part1) ကို define ျပဳလုပ္ပါတယ္။ သူ႔ရဲ႕ members ေတြထဲကို values ေတြ ထည္႔သြင္းေပးၿပီး display ျပန္လုပ္ေပးမွာ ျဖစ္ပါတယ္။ ေလ႔လာၾကည္႔ၾကရေအာင္-

// parts.cpp

// uses parts inventory to demonstrate structures

#include <iostream>

using namespace std;

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

struct part //declare a structure

{

int modelnumber; //ID number of widget

int partnumber; //ID number of widget part

float cost; //cost of part

};

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

int main()

{

part part1; //define a structure variable

part1.modelnumber = 6244; //give values to structure members

part1.partnumber = 373;

part1.cost = 217.55F;

//display structure members

cout << “Model “ << part1.modelnumber;

cout << “, part “ << part1.partnumber;

cout << “, costs $” << part1.cost << endl;

return 0;

}

Run ၾကည္႔တဲ႔အခါ ေအာက္ပါ output ေတြကို ရရွိမွာ ျဖစ္ပါတယ္။

Model 6244, part 373, costs $217.55

parts.cpp ပရိုဂရမ္မွာ အဓိက အစိတ္အပိုင္း ၃ ခု ပါဝင္ပါတယ္။ ၁) structure ကို define ျပဳလုပ္ျခင္း ၂) structure variable ကို define ျပဳလုပ္ျခင္းနဲ႔ ၃) structure ရဲ႕ members ေတြကို ရယူသံုးစြဲျခင္းတို႔ပဲ ျဖစ္ပါတယ္။ တစ္ခုခ်င္းကို ေအာက္မွာ အေသးစိတ္ ေလ႔လာၾကည္႔ၾကရေအာင္။

**Defining the Structure**

structure definition ဆိုတာကေတာ႔ structure တစ္ခု ဘယ္လို ဖြဲ႕စည္းထားတယ္ ဆိုတာကို ေဖာ္ျပေပးပါတယ္။ structure မွာ ပါဝင္မယ္႔ members ေတြကို သတ္မွတ္ေပးပါတယ္။ ဥပမာ -

struct part

{

int modelnumber;

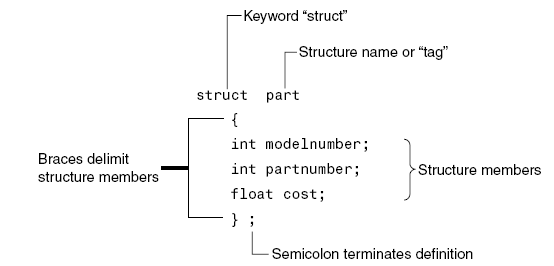
int partnumber;

float cost;

};

**Syntax of the Structure Definition**

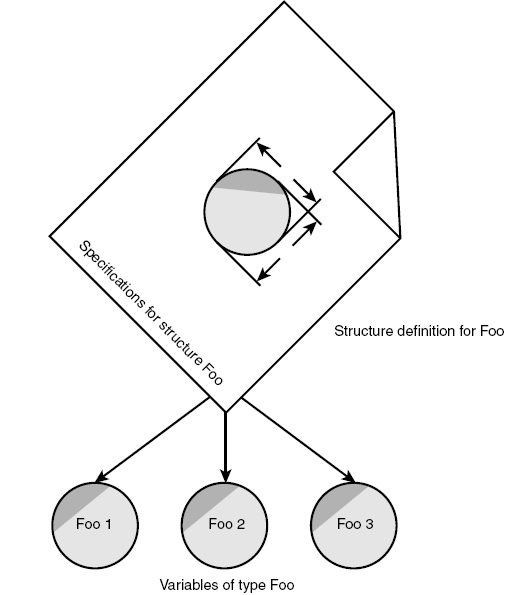
struct ဆိုတဲ႔ keyword နဲ႔ structure definition ကို အစျပဳထားပါတယ္။ ေနာက္က ကပ္ရပ္ေရးရတာက *structure name* ဒါမွမဟုတ္ *tag* ျဖစ္ပါတယ္။ ဒီဥပမာမွာေတာ႔ part လို႔ ေပးထားပါတယ္။ modelnumber, partnumber နဲ႔ cost ဆိုတဲ႔ structure members ေတြရဲ႕ declarations ကို တြန္႔ကြင္း အဖြင္႔အပိတ္ၾကားမွာ ေရးသားရပါတယ္။ တြန္႔ကြင္း အပိတ္ရဲ႕ ေနာက္မွာ semicolon နဲ႔ ပိတ္ၿပီး structure တစ္ခုလံုးကို အဆံုးသတ္ထားပါတယ္။ အျခား code block ေတြ (ဥပမာ- loop, decisions နဲ႔ functions) ကိုေတာ႔ တြန္႔ကြင္း အဖြင္႔အပိတ္နဲ႔ပဲ အစျပဳ၊ အဆံုးသတ္ေလ႔ ရွိၿပီး structure မွာေတာ႔ semicolon နဲ႔ အဆံုးသတ္တာ ျဖစ္ပါတယ္။ ေအာက္မွာ structure declaration တစ္ခုရဲ႕ syntax ကို ေဖာ္ျပေပးထားပါတယ္။



**ပံု (၁၈-၁)** *Syntax of the structure definition.*

**Use of the Structure Definition**

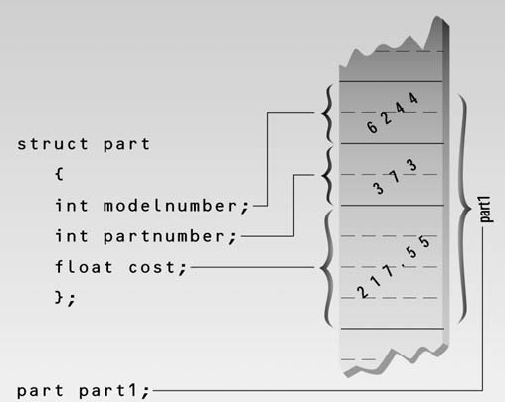
Structure definition သည္ အဆိုပါ Structure အမ်ိဳးအစား variables ကို ဖန္တီးရန္ blueprint အေနျဖင္႔သာ အသံုးျပဳထားသည္။ ဆိုလိုသည္႔အခ်က္မွာ structure definition သည္ အျခား သာမာန္ variables မ်ား၏ definition မ်ားလို memoryအတြင္း ေနရာယူူျခင္း မရွိသလို variable မ်ားကို အမည္ေပးျခင္းလည္း မဟုတ္ေပ။ ၄င္းသည္ structure variable တစ္ခုကို define ျပဳလုပ္ၿပီးလွ်င္ မည္သည္႔ ပံုစံ ရွိႏိုင္မည္ဟူေသာ ေဖာ္ျပခ်က္ သက္သက္သာ ျဖစ္သည္။ ေနာက္ပိုင္း class ကို ေလ႔လာသည္႔အခါ အလားတူ သေဘာတရားမ်ိဳးကို အသံုးျပဳထားျခင္းျဖစ္ေၾကာင္း ေတြ႔ရွိရမည္ ျဖစ္သည္။



ပံု (၁၈-၂) structure ႏွင္႔ structure variables

**Defining a Structure Variable**

main() ထဲရွိ ပထမဆံုး ေဖာ္ျပခ်က္ျဖစ္ေသာ part part1; သည္ structure part type ၏ part1 ဟူေသာ variable တစ္ခုကို defines ျပဳလုပ္ျခင္းပင္ ျဖစ္သည္။ အဆိုပါ ကုဒ္ေၾကာင္႔ memory ထဲတြင္ part1 ၏ members မ်ားကို ဖန္တီးမည္ ျဖစ္သည္။ မည္မွ် ေနရာယူမည္ဆိုသည္ကို part1 အတြင္းရွိ members မ်ားျဖစ္ေသာ modelnumber, partnumber ႏွင္႔ cost တို႔က သတ္မွတ္ေပးသည္။ အထက္ပါ ဥပမာတြင္ integer ႏွစ္ခုအတြက္ 4 bytes (32-bit) ႏွင္႔ float အတြက္ 4 bytes စုစုေပါင္း 8 bytes ေနရာယူမည္ ျဖစ္သည္။ ပံု (၁၈-၃) ကိုၾကည္႔ပါ။



ပံု (၁၈-၃) memory အတြင္း structure member မ်ား ေနရာယူပံု

ေနာက္အပတ္ေတြမွာေတာ႔ structures ေတြကို လက္ေတြ႔ အသံုးျပဳပံု နည္းလမ္းမ်ားအေၾကာင္းကို 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://www.tech4mm.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕