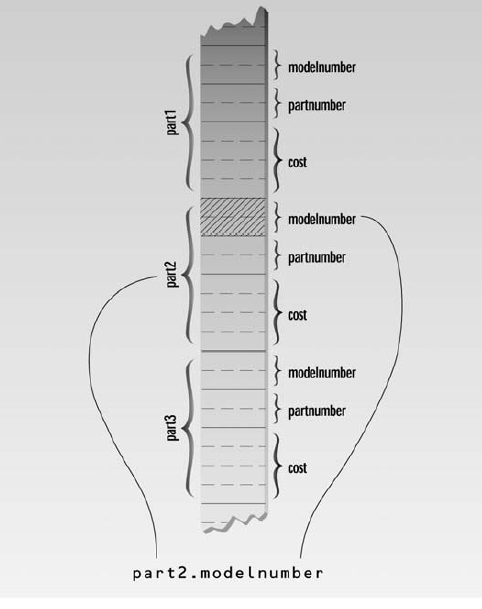
**OOP in C++ : အပိုင္း (၁၉) - Structures (၂)**

**Accessing Structure Members**

Structure variable ေတြကို defined လုပ္ၿပီးတာနဲ႔ သူရဲ႕ members ေတြကို *dot operator* အသံုးျပဳၿပီး access လုပ္ႏိုင္ပါၿပီ။ ေအာက္မွာ အဲဒီလို ေရးသားနည္းကို နမူနာျပထားပါတယ္။

part1.modelnumber = 6244;

Structure member ေတြကို အသံုးျပဳဖို႔ အပိုင္း(၃)ပိုင္း ေရးသားေပးရမွာ ျဖစ္ပါတယ္။ ပထမ အပိုင္းကေတာ႔ structure variable ရဲ႕ name ျဖစ္ပါတယ္။ အဲဒီေနာက္မွာ ဒုတိယအပိုင္းအေနနဲ႔ period (.) ကိုေရးသားရမွာျဖစ္ၿပီး ေနာက္ဆံုး တတိယ အပိုင္းအေနနဲ႔ member name (ဒီဥပမာမွာ- modelnumber)ကို ေရးသားရမွာပါ။ အဲဒီအဓိပၸါယ္ကေတာ႔ “the modelnumber member of part1.” ဒါမွမဟုတ္ part1 ရဲ႕ member တစ္ခုျဖစ္တဲ႔ modelnumber လို႔ ဆိုလိုတာပဲ ျဖစ္ပါတယ္။ Dot operator ရဲ႕ အေခၚအေဝၚ အျပည္႔အစံု က *member access operator*  ျဖစ္ေပမယ္႔ ရွည္တဲ႔အတြက္ အသံုးမျပဳခ်င္ၾကတာ ျဖစ္ပါတယ္။ အဲဒီမွာ သတိျပဳရမယ္႔ အခ်က္ကေတာ႔ ပထမဆံုး ေရးသားတဲ႔ အပိုင္းဟာ structure definition (part) မဟုတ္ဘဲ specific structure variable (part1) ျဖစ္တယ္ဆိုတာပါပဲ။ Variable name ေတြဟာ part1, part2 စသျဖင္႔ တစ္ခုနဲ႔ တစ္ခု ခြဲျခားဖုိ႔အတြက္ အသံုးျပဳတာပဲ ျဖစ္ပါတယ္။ ေအာက္က ပံု(၁၉-၁) မွာ ေလ႔လာၾကည္႔ႏိုင္ပါတယ္။



ပံု(၁၉-၁) Dot operator အသံုးျပပံု

Structure members ေတြကို အျခား variables ေတြလိုပဲ လုပ္ေဆာင္ေစပါတယ္။ ဥပမာ part1.modelnumber = 6244; ဆိုတဲ႔ statement မွာ သာမန္ assignment operator ကိုသံုးၿပီး အဲဒီ member ထဲကို တန္ဖိုး 6244 ထည္႔သြင္းေပးလိုက္တာ ျဖစ္ပါတယ္။ cout statements ေတြကို သံုးၿပီးေတာ႔လည္း members ေတြရဲ႕ တန္ဖိုးေတြကို print ထုတ္ေပးလို႔ ရပါတယ္။

cout << “\nModel “ << part1.;

အထက္ပါ statement ဟာ structure member ျဖစ္တဲ႔ modelnumber ရဲ႕ value ကို output ထုတ္ေပးမွာ ျဖစ္ပါတယ္။

**Other Structure Features**

Structures ေတြဟာ အံ႔ၾသစရာ ေကာင္းေလာက္ေအာင္ စြယ္စံု အသံုးဝင္ပါတယ္။ ဒါေၾကာင္႔ ေနာက္ထပ္ အသံုးတည္႔တဲ႔ structure ေတြရဲ႕ features ေတြကို ေလ႔လာၾကည္႔ၾကရေအာင္။

**Initializing Structure Members**

ေနာက္လာမယ္႔ ဥပမာ ပရိုဂရမ္ေလးမွာ structure variable ေတြကို define ျပဳလုပ္ရင္း structure members ေတြကို initialize ျပဳလုပ္နည္းေတြကို ျပသထားပါတယ္။ ဒီဥပမာမွာပဲ structure type တစ္ခုကို variable တစ္ခုထက္ပိုၿပီး အသံုးခ်ႏိုင္တာကိုလဲ ေလ႔လာႏိုင္ပါတယ္။

// partinit.cpp

// shows initialization of structure variables

#include <iostream>

using namespace std;

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

struct part //specify a structure

{

int modelnumber; //ID number of widget

int partnumber; //ID number of widget part

float cost; //cost of part

};

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

int main()

{ //initialize variable

part part1 = { 6244, 373, 217.55F };

part part2; //define variable

//display first variable

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

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

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

part2 = part1; //assign first variable to second

//display second variable

cout << “Model “ << part2.modelnumber;

cout << “, part “ << part2.partnumber;

cout << “, costs $” << part2.cost << endl;

return 0;

}

ဒီပရိုဂရမ္ေလးမွာ part ရဲ႕ type အမ်ိဳးအစား variables ႏွစ္ခု (part1, part2) ကို define ျပဳလုပ္ပါတယ္။ part1 ကိုေတာ႔ define ျပဳလုပ္ခ်ိန္မွာပဲ initialize လုပ္ေပးလိုက္ပါတယ္။ တနည္းအားျဖင္႔ တန္ဖိုးေတြ ထည္႔သြင္းေပးလိုက္တာပါ။ ၿပီးေတာ႔ part1 ထဲက members ေတြရဲ႕ တန္ဖိုးေတြကို print ထုတ္ေပးပါတယ္။ part2 ထဲကို part1 assign လုပ္ေပးပါတယ္။ ေနာက္ဆံုးမွာ part2 ထဲက members ေတြရဲ႕ တန္ဖိုးေတြကို ေအာက္ပါအတိုင္း print ထုတ္ေပးလုိက္ပါတယ္။

Model 6244, part 373, costs $217.55

Model 6244, part 373, costs $217.55

တန္ဖိုးႏွစ္ခု တူေနၾကတာကေတာ႔ variable ႏွစ္ခုဟာ တူညီေနလို႔ပဲ ျဖစ္ပါတယ္။

part part1 = { 6244, 373, 217.55 }; ဆိုတဲ႔ statement ကေတာ႔ part1 ကိုေတာ႔ define ျပဳလုပ္ခ်ိန္မွာinitialize လုပ္ေပးလိုက္တာပါ။ structure members ေတြကို assign လုပ္မယ္႔ တန္ဖိုးေတြကို တြန္႔ကြင္းနဲ႔ ခတ္ၿပီး တစ္ခုနဲ႔ တစ္ခုၾကား ေကာ္မာခံထားဖို႔ လိုပါတယ္။ ပထမဆံုး တန္ဖိုးကို ပထမ member၊ ဒုတိယ တန္ဖိုးကို ဒုတိယ member စသျဖင္႔ ထည္႔သြင္းေပးသြားမွာ ျဖစ္ပါတယ္။

**Structure Variables in Assignment Statements**

ကၽြန္ေတာ္တို႔ partinit.cpp မွာ ေတြ႔ခဲ႔ၾကတဲ႔ အတိုင္း structure variable ေတြကို တစ္ခုနဲ႔ တစ္ခု ေအာက္ပါအတိုင္း assign ျပဳလုပ္ႏိုင္ပါတယ္။

part2 = part1;

အဲဒီအခါမွာ part1 ရဲ႕ member အားလံုးဟာ part2 ရဲ႕ သက္ဆိုင္ရာ member ေတြအားလံုးကို assign ျပဳလုပ္ေပးပါတယ္။ Structure တစ္ခုမွာ members ေတြ ဒါဇင္နဲ႔ခ်ီၿပီး ပါဝင္ႏိုင္တာေၾကာင္႔ ကြန္ပ်ဴတာအေနနဲ႔ ဒီ assignment ျဖစ္စဥ္ကို ေဆာင္ရြက္ဖို႔ အလုပ္ေတာ္ေတာ္မ်ားမ်ား လုပ္ေပးရမွာ ျဖစ္ပါတယ္။

မွတ္သားရမွာ တစ္ခုကေတာ႔ အဲဒီလို assign ျပဳလုပ္မယ္႔ structures ေတြဟာ structure type တူညီဖို႔ လိုအပ္ပါတယ္။ တကယ္လို႔ မတူညီတဲ႔ structures ႏွစ္ခုကို assign ျပဳလုပ္မယ္ဆိုရင္ error message ျပမွာ ျဖစ္ပါတယ္။

**A Measurement Example**

ေနာက္ထပ္ ဥပမာတစ္ခုအေနနဲ႔ မတူညီတဲ႔ သတင္းအခ်က္အလက္ ေတြကို အုပ္စုဖြဲ႔ၿပီး အသံုးခ်ဖို႔ structure တစ္ခု တည္ေဆာက္ျပပါမယ္။ အေဆာက္အဦ တစ္ခုရဲ႕ ပံုၾကမ္းကို ေလ႔လာၾကည္႔မယ္ဆိုရင္ အကြာအေဝးေတြဟာ ေပေတြ လက္မေတြနဲ႔ တိုင္းတာထားတာကို ေတြ႔ရွိရမွာပါ။ (တစ္ေပမွာ ၁၂ လက္မ ရွိတယ္ဆိုေတာ႔ အားလံုး သိၿပီးသားျဖစ္ပါတယ္)။ ဧည္႔ခန္းတစ္ခုရဲ႕ အလ်ားကို 15’–8” လို႔ ျပထားမယ္ဆိုရင္ အဲဒီမွာ သံုးထားတဲ႔ hyphen ဟာ အႏုတ္ လကၡဏာ မဟုတ္ဘဲ ေပနဲ႔ လက္မကို ျခားထားေပးတာပါ။ ၁၅ ေပ နဲ႔ ၈ လက္မလို႔ ဆိုလိုပါတယ္။ အဲဒီလို စနစ္မ်ိဳးအတြက္ ဆိုရင္ ေပအတြက္ variable တစ္ခုနဲ႔ လက္မအတြက္ variable တစ္ခုဆိုၿပီး ႏွစ္ခု သံုးရပါလိမ္႔မယ္။ ေအာက္က englstrc.cpp ပရိုဂရမ္ေလးမွာ အဲဒီျပႆနာကို structure အသံုးခ်ၿပီး ေျဖရွင္းျပထားပါတယ္။ အဲဒီ structure ထဲမွာ မတူညီတဲ႔ Distance type ႏွစ္ခုကို အတူတူ ထည္႔သြင္းေရးသား အသံုးျပဳထားတာ ျဖစ္ပါတယ္။

// englstrc.cpp

// demonstrates structures using English measurements

#include <iostream>

using namespace std;

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

struct Distance //English distance

{

int feet;

float inches;

};

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

int main()

{

Distance d1, d3; //define two lengths

Distance d2 = { 11, 6.25 }; //define & initialize one length

//get length d1 from user

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

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

//add lengths d1 and d2 to get d3

d3.inches = d1.inches + d2.inches; //add the inches

d3.feet = 0; //(for possible carry)

if(d3.inches >= 12.0) //if total exceeds 12.0,

{ //then decrease inches by 12.0

d3.inches -= 12.0; //and

d3.feet++; //increase feet by 1

}

d3.feet += d1.feet + d2.feet; //add the feet

//display all lengths

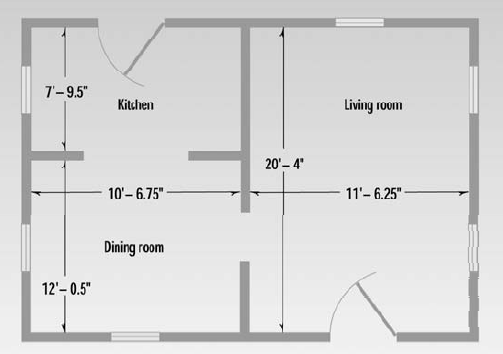
cout << d1.feet << “\’-” << d1.inches << “\” + “;

cout << d2.feet << “\’-” << d2.inches << “\” = “;

cout << d3.feet << “\’-” << d3.inches << “\”\n”;

return 0;

}



ပံု(၁၉-၂) အိမ္ပံုၾကမ္း ႏွင္႔ အတိုင္းအတာမ်ား ျပပံု

အထက္ပါ ပရိုဂရမ္ထဲမွာ feet နဲ႔ inches ဆိုတဲ႔ members ႏွစ္ခု ပါဝင္တဲ႔ Distance structure ကို အသံုးျပဳထားပါတယ္။ လက္မေတြကို မ်ားေသာအားျဖင္႔ ဒႆမကိန္းနဲ႔ ေဖာ္ျပႏိုင္တာေၾကာင္႔ inches အတြက္ float type ကို သံုးထားပါတယ္။ ေပ ကေတာ႔ အျမဲတမ္း ကိန္းျပည္႔ျဖစ္ေနတာေၾကာင္႔ feet ကို int type အသံုးျပဳထားတာပါ။

We define two such distances, d1 and d3, without initializing them, while we initialize another, d2, to 11'–6.25''.

အတိုင္းအတာေတြကို မွတ္သားဖို႔ structure variable သံုးခု (d1, d2, d3) ကို define ျပဳလုပ္ပါတယ္။ ဒါေပမယ္႔ d2 ကိုပဲ initialize ျပဳလုပ္ေပးပါတယ္။

Distance d2 = { 11, 6.25 }; //define & initialize one length

ပရုဂရမ္က user ကို အတိုင္းအတာတစ္ခုရဲ႕ ေပ နဲ႔ လက္မ တန္ဖိုးေတြကို ထည္႔သြင္းေပးဖို႔ ေတာင္းဆိုပါတယ္။ (လက္မ တန္ဖိုးဟာ ၁၂ ေအာက္ ငယ္ဖို႔ လိုပါတယ္)။ အဲဒီေနာက္မွာေတာ႔ d1 နဲ႔ d2 ကို ေပါင္းၿပီး d3 ထဲမွာ ရလဒ္ကို သိမ္းလိုက္ပါတယ္။ ေနာက္ဆံုးမွာေတာ႔ အတိုင္းအတာ သံုးခုလံုးရဲ႕ တန္ဖိုးေတြကို print ျပန္ထုတ္ေပးပါတယ္။ ေအာက္မွာ နမူနာ ျပထားပါတယ္။

Enter feet: 10

Enter inches: 6.75

10’-6.75” + 11’-6.25” = 22’-1”

အဲဒီမွာ သတိထားစရာ အခ်က္ကေတာ႔ အဲဒီ structure variable ေတြကို ေအာက္ပါအတိုင္း တန္းေပါင္းလို႔ မရတာပါပဲ။

d3 = d1 + d2; // can’t do this in ENGLSTRC

ဘာေၾကာင္႔လဲဆိုေဆာ႔ C++ မွာ Distance type variables ေတြကို ေပါင္းေပးႏိုင္တဲ႔ routine ေတြ မရွိလို႔ဘဲ ျဖစ္ပါတယ္။ + operator ဟာ float, int စတဲ႔ built-in types ေတြကို ေပါင္းေပးႏိုင္ပါတယ္။ ဒါေပမယ္႔ ကၽြန္ေတာ္တို႔ ကိုယ္တိုင္ define လုပ္လိုက္တဲ႔ type ေတြကိုေတာ႔ ေပါင္းေပးႏိုင္စြမ္း မရွိပါဘူး။ (Class ကို ေလ႔လာၾကတဲ႔ အခါမွာေတာ႔ “Operator Overloading,” အေၾကာင္းကို ရွင္းျပေပးပါ႔မယ္။)

ေနာက္အပတ္ေတြမွာေတာ႔ structures ေတြ အတြင္းမွာ ေနာက္ထပ္ structures ေတြ ထည္႔သြင္းအသံုးျပဳပံု 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://engineer4myanmar.blogspot.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕