**OOP in C++ : အပိုင္း (၂၁) - Structures (၄)**

**A Card Game Example**

ဒီအပတ္မွာေတာ႔ ေနာက္ထပ္ ဥပမာေလး တစ္ခုကို သံုးၿပီး structures ေတြကို ေလ႔လာၾကည္႔ၾကရေအာင္။ ဖဲသံုးခ်ပ္ ကစားနည္းကို ျမင္ဖူးၾကမယ္ ထင္ပါတယ္။ မ်ားေသာအားျဖင္႔ ဘုရားပြဲေတြမွာ ေတြ႔ရတတ္ပါတယ္။ အဲဒီကစားနည္းမွာ cardsharp လို႔ ေခၚတဲ႔ ဖဲဝိဇၨာက ဖဲခ်ပ္ သံုးခ်ပ္ကို ျပထားၿပီး စားပြဲေပၚေမွာက္ထားလိုက္ပါတယ္။ ကၽြန္ေတာ္တို႔ မ်က္ေစ႔ေရွ႕မွာပဲ အဲဒီဖဲသံုးခ်ပ္ကို ေနရာအႀကိမ္ႀကိမ္ေျပာင္းလိုက္ပါတယ္။ ၿပီးတဲ႔အခါမွာ ကၽြန္ေတာ္တို႔ကို ဖဲတစ္ခ်ပ္ကို ေရြးခိုင္းပါတယ္။ သတ္မွတ္ထားတဲ႔ ဖဲျဖစ္ရင္ ကၽြန္ေတာ္တို႔ ႏိုင္ပါတယ္။ မွားခဲ႔ရင္ေတာ႔ ရွံႈးသြားမွာပါ။ ဖဲဝိဇၨာက ဖဲကို ေရႊ႕တာ အရမ္းကို ျမန္ၿပီး ရႈပ္ေထြးတာေၾကာင္႔ ကစားသူက မ်က္စိနဲ႔ လိုက္မမွတ္ႏိုင္ပါဘူး။ ဒါေၾကာင္႔ မွန္းဆဖို႔ ခက္ခဲပါတယ္။ အဲဒီကစားနည္းကို ေအာက္မွာ structure အသံုးျပဳထားတဲ႔ ပရိုဂရမ္ေလးနဲ႔ ေရးျပထားပါတယ္။ ပထမဆံုး ဖဲခ်ပ္ေတြကို ကိုယ္စားျပဳဖို႔ card ဆိုတဲ႔ structure တစ္ခုကို ဖန္တီးလိုက္ပါတယ္။

struct card

{

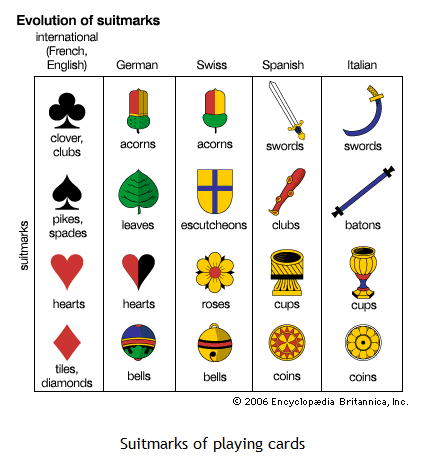
int number;

int suit;

};

အဲဒီ structure က ဖဲတစ္ခ်ပ္ကို ကိုယ္စားျပဳပါတယ္။ သူ႔ထဲမွာပါတဲ႔ number ဆိုတဲ႔ member ကေတာ႔ 2 ကေန 14 အထိ ဂဏန္းေတြကို သိမ္းေပးမွာပါ။ ဒီေနရာမွာ ဖဲထုပ္ထဲမွာ 2 ကေန 10 အထိပဲ ဂဏန္းရွိတာေၾကာင္႔ 11,12,13 နဲ႔ 14 ေတြဟာ ဂ်က္၊ ကြင္း၊ ကင္းနဲ႔ ေအ႔ ေတြကို အသီးသီး ကိုယ္စားျပဳေပးမွာပါ။ suit ကေတာ႔ 0 ကေန 3 အထိ ပါဝင္မွာ ျဖစ္ၿပီး ကလပ္၊ ဒိုင္းမြန္း၊ ဟတ္ နဲ႔ စပိတ္ ေတြကို ကိုယ္စားျပဳမွာပါ။ စာဖတ္သူတို႔ အေထြေထြ ဗဟုသုတ ရႏိုင္ေစဖို႔အတြက္ ၿဗိတိသွ်စြယ္စံုက်မ္း Encyclopædia Britannica ထဲက ေရွးေဟာင္းဖဲခ်ပ္ေလးေတြကို ကူးယူေဖာ္ျပေပးလိုက္ပါတယ္။





cards.cpp ကို ေလ႔လာၾကည္႔ပါ။

// cards.cpp

// demonstrates structures using playing cards

#include <iostream>

using namespace std;

const int clubs = 0; //suits

const int diamonds = 1;

const int hearts = 2;

const int spades = 3;

const int jack = 11; //face cards

const int queen = 12;

const int king = 13;

const int ace = 14;

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

struct card

{

int number; //2 to 10, jack, queen, king, ace

int suit; //clubs, diamonds, hearts, spades

};

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

int main()

{

card temp, chosen, prize; //define cards

int position;

card card1 = { 7, clubs }; //initialize card1

cout << “Card 1 is the 7 of clubs\n”;

card card2 = { jack, hearts }; //initialize card2

cout << “Card 2 is the jack of hearts\n”;

card card3 = { ace, spades }; //initialize card3

cout << “Card 3 is the ace of spades\n”;

prize = card3; //copy this card, to remember it

cout << “I’m swapping card 1 and card 3\n”;

temp = card3; card3 = card1; card1 = temp;

cout << “I’m swapping card 2 and card 3\n”;

temp = card3; card3 = card2; card2 = temp;

cout << “I’m swapping card 1 and card 2\n”;

temp = card2; card2 = card1; card1 = temp;

cout << “Now, where (1, 2, or 3) is the ace of spades? “;

cin >> position;

switch (position)

{

case 1: chosen = card1; break;

case 2: chosen = card2; break;

case 3: chosen = card3; break;

}

if(chosen.number == prize.number && // compare cards

chosen.suit == prize.suit)

cout << “That’s right! You win!\n”;

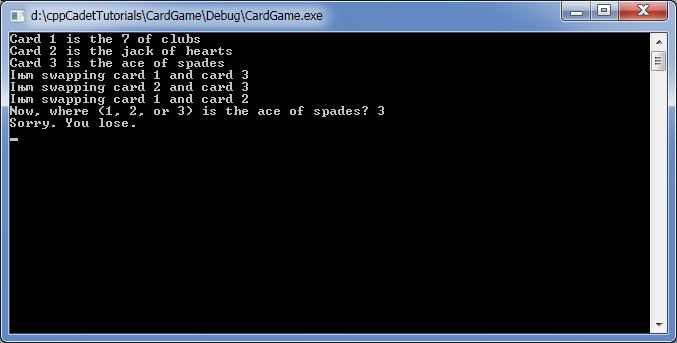
else

cout << “Sorry. You lose.\n”;

return 0;

}

ေအာက္မွာ ပရိုဂရမ္ရဲ႕ output ကို ျပထားပါတယ္-



အဲဒီ ပရိုဂရမ္မွာ အေျဖမွန္က ၂ ျဖစ္ပါတယ္။

ပရိုဂရမ္ရဲ႕ အစမွာ face card ေတြနဲ႔ suit တန္ဖိုးေတြ ထည္႔သြင္းဖို႔ const int variables ေတြ ဖန္တီးပါတယ္။ အဲဒီ variables ေတြ အကုန္လံုးကိုေတာ႔ ဒီပရိုဂရမ္ထဲမွာ အသံုးမျပဳထားပါဘူး။ ဒါေပမယ္႔ ျပည္႔စံုေအာင္ ထည္႔ေပးထားတဲ႔ သေဘာပါ။ card structure ကို သတ္မွတ္ၿပီးတဲ႔ အခါမွာ အဲဒီ card structure အမ်ိဳးအစား variable သံုးခုျဖစ္တဲ႔ temp, chosen နဲ႔ prize တို႔ကို initialize မလုပ္ဘဲ ဖန္တီးလိုက္ပါတယ္။ ေနာက္ပိုင္း ဖဲခ်ပ္ေတြ ေရႊ႕ဖို႔နဲ႔ ကစားသမား ေရြးခ်ယ္တဲ႔ ဖဲခ်ပ္မွန္မမွန္ တိုက္ဆိုင္ စစ္ေဆးတဲ႔ ေနရာမွာ အသံုးျပဳဖို႔ပဲ ျဖစ္ပါတယ္။

ကစားတဲ႔ ေနရာမွာ အသံုးျပဳတဲ႔ ဖဲသံုးခ်ပ္ အတြက္ကေတာ႔ ေနာက္ထပ္ card1, card2 နဲ႔ card3 ဆိုတဲ႔ card structure သံုးခုကို တန္ဖိုးေတြနဲ႔ initialized လုပ္ေပးလိုက္ပါတယ္။ အဲဒီေနာက္မွာ ကစားသမား သိရွိႏိုင္ေစရန္ ရည္ရြယ္ၿပီး အဲဒီ ဖဲခ်ပ္ေတြကို print out ထုတ္ေပးလိုက္ပါတယ္။ ကစားသူမွ ခန္႔မွန္း ေရြးခ်ယ္ရမယ္႔ ဖဲခ်ပ္ကို မွတ္သားထားဖို႔ အတြက္ prize ဆိုတဲ႔ variable ကို အသံုးျပဳပါတယ္။

ပရိုဂရမ္က ဖဲခ်ပ္ေတြကို ေနရာေရႊ႕ေျပာင္းပါတယ္။ ပထမဖဲကို တတိယဖဲနဲ႔ လဲပါတယ္။ ေနာက္ ဒုတိယဖဲကို တတိယဖဲနဲ႔ လဲပါတယ္။ ေနာက္တစ္ခါမွာေတာ႔ ပထမဆံုးဖဲကို ဒုတိယဖဲ နဲ႔ ေနရာလဲလိုက္ပါတယ္။ ေနရာလဲသမွ်ကိုလည္း ကစားသူကို အသိေပးေနမွာပါ။ (ဒီပရိုဂရမ္ေလးမွာေတာ႔ နမူနာအေနနဲ႔ ေရးျပထားတာျဖစ္လို႔ လြယ္ကူစြာ ခန္႔မွန္းႏိုင္ေပမယ္႔ တကယ္တမ္း ရာနဲ႔ခ်ီၿပီး ေနရာလဲလိုက္မယ္ အသိေပးခ်က္ေတြကိုလဲ အခ်ိန္အနည္းငယ္သာ ျပသေပးလိုက္မယ္ဆိုရင္ ခန္႔မွန္းဖို႔ ခက္သြားမွာ ျဖစ္ပါတယ္။)

ေနာက္ဆံုးမွာ ကစားသူကို သတ္မွတ္ထားတဲ႔ဖဲခ်ပ္ ရွိတဲ႔ ေနရာကို ခန္႔မွန္းခိုင္းပါတယ္။ အဲဒီတန္ဖိုးကို chosen ထဲကို ထည္႔ပါတယ္။ မွန္မွန္ကို သိဖို႔ prize card နဲ႔ ႏိႈင္းယွဥ္ စစ္ေဆးပါတယ္။ တကယ္လို႔ chosen နဲ႔ prize တန္ဖိုးေတြ တူခဲ႔ရင္ ကစားသူ အႏိုင္ရရွိမွာ ျဖစ္ၿပီး မတူခဲ႔ရင္ေတာ႔ ရႈံးနိမ္႔မွာ ျဖစ္ပါတယ္။

ဖဲခ်ပ္ေတြ လဲလွယ္ဖို႔အတြက္ကေတာ႔ structures ေတြ ျဖစ္ေပမယ္႔ assignment operator ကိုပဲ သံုးၿပီး အလြယ္တကူ ျပဳလုပ္ထားပါတယ္။ temp = card3; card3 = card1; card1 = temp;

ဒီေနရာမွာ ျပႆနာတစ္ခုက structure ေတြကို ေပါင္းလို႔မရသလို == operator ကိုသံုးၿပီး ( chosen == prize ဆိုတဲ႔ ပံုစံမ်ိဳး) အလြယ္တကူ ႏိႈင္းယွဥ္လို႔ မရႏိုင္တာ ျဖစ္ပါတယ္။ ဒီျပႆနာကို Class အေၾကာင္း ေလ႔လာတဲ႔အခါမွာ ေတြ႔ရွိရမယ္႔ operator overloading နည္းလမ္းနဲ႔ ေျဖရွင္းပံုကို ေနာက္ပိုင္းမွာ ဆက္လက္ ရွင္းျပေပးသြားမွာ ျဖစ္ပါတယ္။

**Structures and Classes**

ၾကံဳႀကိဳက္လာတဲ႔အတြက္ structure နဲ႔ class အေၾကာင္း အနည္းငယ္ ရွင္းျပလိုပါတယ္။ sturcture ေတြကို data ေတြ စုစည္းဖို႔ အသံုးျပဳၿပီး class ေတြကိုေတာ႔ data ေတြနဲ႔ function ေတြ စုစည္းထားဖို႔ အသံုးျပဳတယ္လို႔ ေျပာခဲ႔ပါတယ္။ C မွာေတာ႔ structure ထဲမွာ data ပဲ ထည္႔လို႔ ရပါတယ္။ ဒါေပမယ္႔ C++ မွာ structures ေတြကို data ေတြသာမကာ function ေတြကိုပါ ထည္႔သြင္းႏိုင္ပါတယ္။ သို႔ေသာ္ C++ ပရိုဂရမ္မာ အမ်ားစုဟာ structures ေတြကို data ေတြအတြက္ပဲ သီးသန္႔သံုးေလ႔ ရွိၿပီး function ေတြ ထည္႔သံုးဖို႔လိုလာတိုင္း classes ေတြကိုသာ အသံုးျပဳတတ္ၾကပါတယ္။ ဒါေၾကာင္႔ စာေရးသူအေနနဲ႔လဲ structures ေတြကို data ေတြ စုစည္းဖို႔ အသံုးျပဳနည္းမ်ားကိုသာ ဆက္လက္တင္ျပေပးသြားမွာ ျဖစ္ပါတယ္။

ေနာက္အပတ္မွာေတာ႔ Enumerations အေၾကာင္းကို ဆက္လက္ ေဆြးေႏြးေပးသြားမွာပါ။ 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
* "playing card." Encyclopædia Britannica. Encyclopædia Britannica Ultimate Reference Suite.  Chicago: Encyclopædia Britannica, 2012.

Dr. ေအာင္ဝင္းထြဋ္ (bluephoenix)

http://engineer4myanmar.blogspot.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕