**OOP in C++ : အပိုင္း (၂၂) - Enumerations**

ၿပီးခဲ႔တဲ႔ သင္ခန္းစာေတြမွာ user-defined data types အျဖစ္ structures မ်ားအား အသံုးျပဳနည္းကို ေလ႔လာခဲ႔ၾကပါတယ္။ ယခု အပတ္မွာေတာ႔ emumeration ကို အသံုးျပဳၿပီး ကိုယ္ပိုင္ data types မ်ား ဖန္တီးပံုကို ေနာက္ထပ္ နည္းလမ္းတစ္ခုအေနနဲ႔ ေဆြးေႏြးလိုပါတယ္။ ဒီနည္းလမ္းက structures ေတြေလာက္ေတာ႔ အေရးမႀကီးပါဘူး။ C++ ပရိုဂရမ္ေတြကို enumerations အေၾကာင္းမသိပဲ ေကာင္းေကာင္းေရးႏိုင္ပါတယ္။ ဒါေပမယ္႔ enumerations ေတြအေၾကာင္းကို သိရွိ အသံုးခ်ႏိုင္မယ္ဆိုရင္ ပိုမိုရွင္းလင္း လြယ္ကူၿပီး C++ ပိုဆန္တဲ႔ ပရိုဂရမ္ေတြကို ေရးသားႏိုင္လာမွာ ျဖစ္ပါတယ္။

**Days of the Week**

မ်ားေသာအားျဖင္႔ အတိအက် သိရွိထားၿပီးသား data အနည္းငယ္သာ ပါဝင္တဲ႔ data type တစ္ခုကို ဖန္တီးဖို႔ enumeration ကို အသံုးျပဳေလ႔ ရွိၾကပါတယ္။ ဥပမာအေနနဲ႔ တစ္ပတ္စာ ေန႔ေတြကို enumeration လုပ္ထားတဲ႔ dayenum.cpp ကို ေလ႔လာၾကရေအာင္ -

// dayenum.cpp

// demonstrates enum types

#include <iostream>

using namespace std;

//specify enum type

enum days\_of\_week { Sun, Mon, Tue, Wed, Thu, Fri, Sat };

int main()

{

days\_of\_week day1, day2; //define variables

//of type days\_of\_week

day1 = Mon; //give values to

day2 = Thu; //variables

int diff = day2 - day1; //can do integer arithmetic

cout << “Days between = “ << diff << endl;

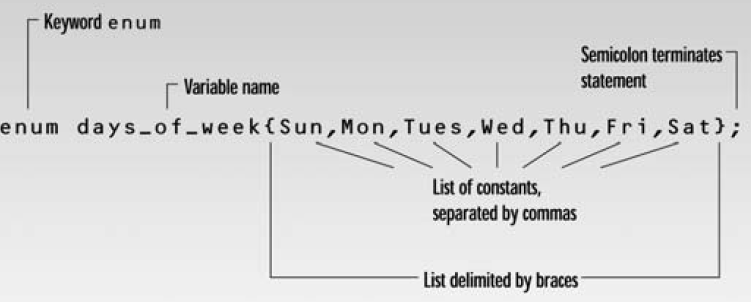
if(day1 < day2) //can do comparisons

cout << “day1 comes before day2\n”;

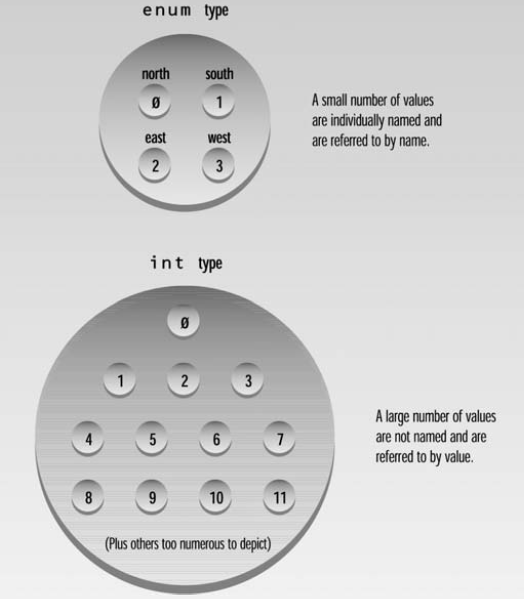
return 0;

}

ပရိုဂရမ္းမင္းတိုင္းမွာ ကၽြန္ေတာ္တို႔ သတ္မွတ္ခ်င္တဲ႔အရာေတြကို မ်ားေသာအားျဖင္႔ int တန္ဖိုး တစ္ခုခု သတ္မွတ္ေပးၿပီး ကြန္ပ်ဴတာကို ေျပာျပေလ႔ ရွိပါတယ္။ ဥပမာ Sunday ဆိုတာကို ကြန္ပ်ဴတာ နားလည္ႏိုင္ဖို႔ int Sunday=0; စသျဖင္႔ သတ္မွတ္ေလ႔ ရွိပါတယ္။ ဒါေၾကာင္႔ ေနာက္ပိုင္း 0 လို႔ ေျပာတိုင္း ကြန္ပ်ဴတာက Sunday လို႔ လက္ခံၿပီး အလုပ္လုပ္ေပးပါတယ္။ ဒါေပမယ္႔ အဲဒီလို အခ်က္အလက္ေတြ မ်ားလာရင္ ကြန္ပ်ဴတာက မွတ္မိေပမယ္႔ လူကေတာ႔ လိုက္မမွတ္ႏိုင္ေတာ႔ပါဘူး။ တခါတေလ အယူအဆေတြ လြဲမွားသြားႏိုင္ပါတယ္။ Enumeration ကို အသံုးျပဳျခင္းအားျဖင္႔ ကၽြန္ေတာ္တို႔ ထားခ်င္တဲ႔ သတ္မွတ္ခ်က္ေတြကို မူရင္း အေခၚအေဝၚအတိုင္း ေရးသား အသံုးျပဳႏိုင္တာေၾကာင္႔ အဲဒီ အခက္အခဲကို ေက်ာ္လႊားႏိုင္ပါတယ္။ အထက္ပါ ပရိုဂရမ္ေလးမွာပါတဲ႔ days\_of\_week ဆိုတဲ႔ enum type မွာ Sun, Mon, Tues, Wed, Thu, Fri, Sat စသျဖင္႔ enumerators (၇) ခု ပါဝင္ၿပီး သူတို႔ရဲ႕ syntax ကို ေအာက္မွာ ေလ႔လာႏိုင္ပါတယ္။



Enumeration ဆိုတာ ျဖစ္ႏိုင္ေျခ ရွိတဲ႔ တန္ဖိုးအားလံုးကို စုစည္းၿပီး စာရင္းျပဳစုထားတာပဲ ျဖစ္ပါတယ္။ အဲဒီ တန္ဖိုး တစ္ခုခ်င္းစီကိုလည္း သီးသန္႔ အမည္မ်ား ေပးထားရပါတယ္။ အထက္မွာ ေဖာ္ျပထားသလို enum တစ္ခုကို declare လုပ္ၿပီးတာနဲ႔ အဲဒီ type အမ်ိဳးအစား variables ေတြကို define လုပ္ႏိုင္ၿပီ ျဖစ္ပါတယ္။ dayenum.cpp မွာ days\_of\_week day1, day2; ဆိုၿပီး day1 နဲ႔ day2 variables ေတြကို define လုပ္ျပထားပါတယ္။ ( C မွာတုန္းကေတာ႔ enum ဆိုတဲ႔ keyword ကို define လုပ္တဲ႔ အခါမွာလည္း ထည္႔ေပးရပါတယ္။ ဥပမာ- enum days\_of\_week day1, day2; ဒါေပမယ္႔ C++ မွာေတာ႔ ထည္႔ေပးစရာ မလိုေတာ႔ပါဘူး။)



day1 နဲ႔ day2 လိုမ်ိဳး enumerated type ေတြကို သက္ဆိုင္ရာ enum declaration မွာ ပါဝင္တဲ႔ တန္ဖိုးေတြထဲက တစ္ခုခုကို သတ္မွတ္ေပးႏိုင္ပါတယ္။ ဒီဥပမာမွာေတာ႔ Mon နဲ႔ Thu ဆိုၿပီး ေပးထားခဲ႔ပါတယ္။ Declaration ထဲမွာ မပါတဲ႔ တန္ဖိုးေတြကိုေတာ႔ သတ္မွတ္ မေပးႏိုင္ပါဘူး။ ဥပမာ day1 = halloween; ဆိုၿပီး ေပးလို႔ မရပါဘူး။ ကၽြန္ေတာ္တို႔အေနနဲ႔ enumerated type ေတြကို standared arithmetic operators (+,-,\*,/) ေတြကို သံုးၿပီး အေပါင္း၊ အႏုတ္၊ အေျမာက္၊ အစား လုပ္ႏိုင္ပါတယ္။ ပရိုဂရမ္ထဲမွာ တန္ဖိုးႏွစ္ခုရဲ႕ ျခားနားျခင္းကို ရွာျပထားပါတယ္။ Comparison operators (<, >, <=, >=, !) ေတြကို အသံုးျပဳၿပီး တစ္ခုနဲ႔ တစ္ခုကိုလည္း ႏိႈင္းယွဥ္ႏိုင္ပါတယ္။ ပရိုဂရမ္ကို run ၾကည္႔မယ္ဆိုရင္ ေအာက္ပါ ရလဒ္ေတြကို ေတြ႔ရမွာ ျဖစ္ပါတယ္။

Days between = 3

day1 comes before day2

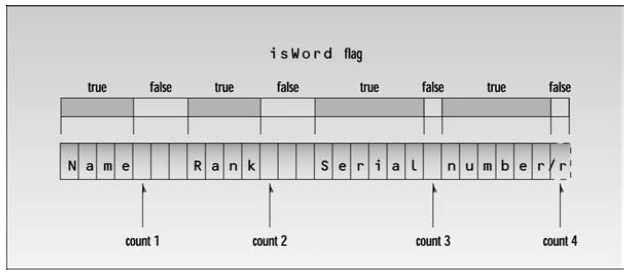
Enum types ေတြကို Arithmetic နဲ႔ relational operators ေတြသံုးလို႔ရေပမယ္႔ တစ္ခ်ိဳ႕ေနရာေတြမွာ အဓိပၸါယ္မရွိတာေၾကာင္႔ အသံုးမျပဳသင္႔ပါဘူး။ ဥပမာ အေနနဲ႔ ေအာက္က enum type declaration မွာ enum pets { cat, dog, hamster, canary, ocelot }; ဆိုၿပီး ေၾကျငာထားတယ္ ဆိုပါစို႔။ ဒီေနရာမွာ (dog + canary) နဲ႔ (cat < hamster) ဆိုတဲ႔ statements ေတြဟာ အဓိပၸါယ္ မရွိပါဘူး။

Enum types ေတြကို arithmetic နဲ႔ relational operations ေတြ လုပ္ေဆာင္ႏိုင္တာဟာ တကယ္ေတာ႔ သူတို႔ကို integers ေတြ အေနနဲ႔ internally သတ္မွတ္ယူဆထားလို႔ပဲ ျဖစ္ပါတယ္။ ဘာမွ သတ္မွတ္ မေပးခဲ႔ဘူးဆိုရင္ ပထမဆံုး တန္ဖိုးကို 0 သတ္မွတ္ေပးမွာ ျဖစ္ပါတယ္။ ေနာက္က တန္ဖိုးေတြကို တစ္ တိုးၿပီး 1, 2, 3, .. စသျဖင္႔ သတ္မွတ္ေပးမွာပါ။ ဒါေၾကာင္႔ dayenum.cpp ထဲက Sun ကေန Sat အထိ တန္ဖိုးေတြဟာ တကယ္ေတာ႔ integer တန္ဖိုး 0-6 ပဲ ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ enum types ေတြကို arithmetic operation ေတြ ျပဳလုပ္တဲ႔ အခါမွာ အဲဒီ integer တန္ဖိုးေတြကုိ အသံုးျပဳၿပီး လုပ္ေဆာင္တာ ျဖစ္ပါတယ္။

ကြန္ပ်ဴတာက enum ေတြကို integers ေတြ အေနနဲ႔ သိေပမယ္႔ ကၽြန္ေတာ္တို႔ကေတာ႔ အဲဒီအခ်က္ကို သတိထားၿပီး အသံုးျပဳရမွာ ျဖစ္ပါတယ္။ day1 = 5; ဆိုၿပီး ေရးခဲ႔မိရင္ compiler က compile လုပ္ေပးေပမယ္႔ warning message ထုတ္ေပးမွာ ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ တတ္ႏိုင္သမွ် enums ေတြဟာ (တကယ္ေတာ႔) integers ေတြ ျဖစ္တယ္ဆိုတဲ႔ အခ်က္ကို ေမ႔ထားရမွာပါ။

**One Thing or Another**

ေနာက္ထပ္ ဥပမာအေနနဲ႔ user မွ ရိုက္ထည္႔လိုက္တဲ႔ စကားလံုးေတြကို ေရတြက္တဲ႔ ပရိုဂရမ္ေလး တစ္ပုဒ္ကို ထပ္မံေလ႔လာမွာ ျဖစ္ပါတယ္။ ယခင္ ဥပမာ တစ္ခုျဖစ္တဲ႔ chcount.cpp မွာ spaces မ်ားကို ေရတြက္ျခင္းျဖင္႔ စကားလံုး အေရအတြက္ကို ခန္႔မွန္းတြက္ခ်က္ခဲ႔ပါတယ္။ တကယ္လို႔ user က space တစ္ခုထက္ပိုၿပီး ေရးသားခဲ႔မယ္ဆိုရင္ စာလံုးအေရအတြက္ မွားယြင္းသြားမွာ ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ ယခု ပရိုဂရမ္မွာ space မဟုတ္တဲ႔ အကၡရာေတြအဆံုးမွာ space ကို ေတြ႔မွ စကားလံုးကို ေရတြက္တဲ႔ စနစ္ကို အသံုးျပဳထားပါတယ္။ အဲဒီနည္းစနစ္ကို ေအာက္က ပံုေလးမွာ ရွင္းျပထားပါတယ္။



This way you don’t get a false count if you type multiple spaces between words. (It still

doesn’t handle tabs and other whitespace characters.) Here’s the listing for WDCOUNT: This example shows an enumeration with only two enumerators.

ဒါေၾကာင္႔ စကားလံုးေတြၾကားမွာ spaces ေတြ အေျမာက္အမ်ားပါေနတာေၾကာင္႔ ေရတြက္မႈ မွားယြင္းတာမ်ိဳး ရွိေတာ႔မွာ မဟုတ္ပါဘူး။ wdcount.cpp မွာ အဲဒီလို ျပဳလုပ္ဖို႔ enumerators ႏွစ္ခု (NO, YES) သာပါတဲ႔ enum (itsaWord) ကို အသံုးျပဳထားပါတယ္။ ေလ႔လာၾကည္႔ၾကရေအာင္ -

// wdcount.cpp

// demonstrates enums, counts words in phrase

#include <iostream>

#include <conio.h>

using namespace std;

#include <conio.h> //for getche()

enum itsaWord { NO, YES }; //NO=0, YES=1

int main()

{

itsaWord isWord = NO; //YES when in a word,

//NO when in whitespace

char ch = ‘a’; //character read from keyboard

int wordcount = 0; //number of words read

cout << “Enter a phrase:\n”;

do {

ch = getche(); //get character

if(ch==’ ‘ || ch==’\r’) //if white space,

{

if( isWord == YES ) //and doing a word,

{ //then it’s end of word

wordcount++; //count the word

isWord = NO; //reset flag

}

} //otherwise, it’s

else //normal character

if( isWord == NO ) //if start of word,

isWord = YES; //then set flag

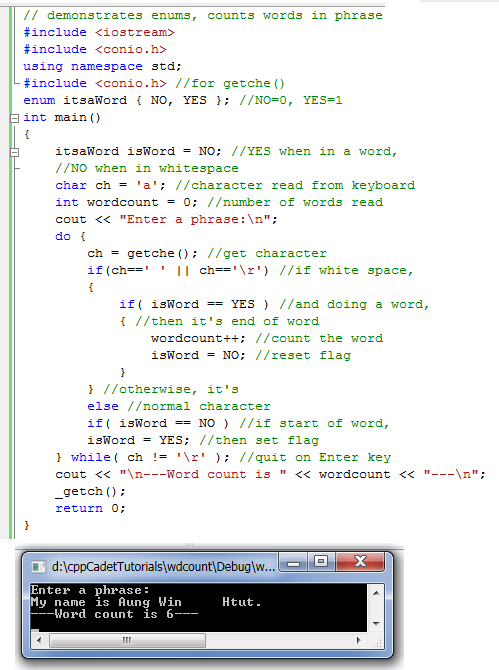
} while( ch != ‘\r’ ); //quit on Enter key

cout << “\n---Word count is “ << wordcount << “---\n”;

\_getch();

return 0;

}



ပရိုဂရမ္ အစမွာ do loop ကို အသံုးျပဳၿပီး keyboard က အကၡရာေတြကို ဖတ္ရႈပါတယ္။ space မဟုတ္တဲ႔ အကၡရာေတြကို ေက်ာ္သြားၿပီး space ကို စေတြ႔တာနဲ႔ အေရွ႕က ဖတ္ခဲ႔တဲ႔ အကၡရာဟာ စကားလံုး ဟုတ္မဟုတ္ကို enum variable - isWord ကို YES ျဖစ္မျဖစ္နဲ႔ စစ္ၿပီး ဟုတ္ခဲ႔ရင္ cont ကို ၁ တိုးလိုက္ပါတယ္။ အဲဒီေနာက္ isWord ကို NO ေျပာင္းလိုက္ပါတယ္။ ဒါမွ ေနာက္ထပ္ space ေတြ ကပ္လ်က္ ရွိေနခဲ႔ရင္ count မလုပ္မွာ ျဖစ္ပါတယ္။ enum - itsaWord မွာ ျဖစ္ႏိုင္ေျခရွိတဲ႔ တန္ဖိုး ႏွစ္ခု (NO နဲ႔ YES) ပဲ ရွိပါတယ္။ NO ကိုေရွ႕ကထားတာ သတိထားမိပါလိမ္႔မယ္။ ဒါေၾကာင္႔ သူ႔ရဲ႕တန္ဖိုးဟာ 0 (false ရဲ႕ သေကၤတ) ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ if(isWord == YES) အစား if(isWord) ဆိုၿပီး စစ္ေဆးႏိုင္ပါတယ္။ တခါတရံ အေျခအေနေပၚ မူတည္ၿပီး အဲဒီေနရာမွာ enum အစား bool variable ကို သံုးႏိုင္ပါေသးတယ္။

**Organizing the Cards**

Here’s our final example of enum types. Remember that in the CARDS program earlier in this chapter we defined a group of constants of type const int to represent a card’s suits.

အခု enum types ေတြကို ရွင္းျပဖို႔ ေနာက္ဆံုး ဥပမာေလး တစ္ခုကို ေလ႔လာၾကရေအာင္။ ယခင္ သင္ခန္းစာမွာ ပါတဲ႔ cards.cpp ပရိုဂရမ္ေလးကို မွတ္မိဦးမယ္ ထင္ပါတယ္။ အဲဒီမွာ card suits ေတြကို const int ေတြသံုးၿပီး ေအာက္ပါအတိုင္း သတ္မွတ္ခဲ႔ပါတယ္။

const int clubs = 0;

const int diamonds = 1;

const int hearts = 2;

const int spades = 3;

ဒီလို ေရးနည္းထက္ ပိုေကာင္းတဲ႔ နည္းလမ္းကို ေအာက္က cardenum.cpp မွာ ေလ႔လာၾကရေအာင္ -

// cardenum.cpp

// demonstrates enumerations

#include <iostream>

using namespace std;

const int jack = 11; //2 through 10 are unnamed integers

const int queen = 12;

const int king = 13;

const int ace = 14;

enum Suit { clubs, diamonds, hearts, spades };

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

struct card

{

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

Suit 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 seven 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;

}

ဒီပရိုဂရမ္မွာ suits ေတြအတြက္ definitions ေတြကို enum အသံုးျပဳၿပီး ေအာက္ပါအတိုင္း declare လုပ္ထားပါတယ္။

enum Suit { clubs, diamonds, hearts, spades };

ဒီနည္းဟာ const variables ေတြ အသံုးျပဳတာထက္ ပိုၿပီး ရွင္းလင္းပါတယ္။ တကယ္လို႔ card1.suit = 5; ဆိုၿပီး သံုးခဲ႔ရင္ေတာ႔ ကြန္ပိုင္လာက သတိေပးတာ ခံရမွာ ျဖစ္ၿပီး မသံုးသင္႔ပါဘူး။

**Specifying Integer Values**

We said that in an enum declaration the first enumerator was given the integer value 0, the second the value 1, and so on. This ordering can be altered by using an equal sign to specify a starting point other than 0.

ကၽြန္ေတာ္တို႔အေနနဲ႔ အေစာပိုင္းမွာ ပထမဆံုး enumerator ကို 0 သတ္မွတ္ထားၿပီး ေနာက္ထပ္ enumerator ေတြကို တစ္တိုးၿပီး 1, 2, 3,.. စသျဖင္႔ သတ္မွတ္ေပးလိမ္႔မယ္လို႔ ေျပာခဲ႔ပါတယ္။ ဒါေပမယ္႔ ကိုယ္ႀကိဳက္ႏွစ္သက္ရာ တန္ဖိုးနဲ႔ အစျပဳဖို႔အတြက္လည္း equal sign (=) ကို အသံုးျပဳႏိုင္ပါတယ္။ ဥပမာ suits ေတြရဲ႕ တန္ဖိုးကို 0 နဲ႔ အစမျပဳဘဲ 1 နဲ႔ စခ်င္တယ္ဆိုပါစို႔။

enum Suit { clubs=1, diamonds, hearts, spades }; ဆိုၿပီး ေရးသားႏိုင္ပါတယ္။ ေနာက္က လာတဲ႔ တန္ဖိုးေတြကေတာ႔ ထံုးစံအတိုင္း တစ္တိုးၿပီး 2, 3, 4,.. စသျဖင္႔ ျဖစ္သြားမွာပါ။ တကယ္တမ္းမွာ တန္ဖိုးတစ္ခုခ်င္းစီကို equal sign (=) အသံုးျပဳၿပီး ထည္႔သြင္းေပးလိုက္ႏိုင္ပါတယ္။

**Not Perfect**

eunm ရဲ႕ အားနည္းခ်က္ကေတာ႔ ၄င္းကို C++ ရဲ႕ I/O statements ေတြက နားမလည္တာပဲ ျဖစ္ပါတယ္။ ဥပမာ ေအာက္က ကုဒ္ေတြကို run မယ္ဆိုပါစို႔

enum direction { north, south, east, west };

direction dir1 = south;

cout << dir1;

Did you guess the output would be south? That would be nice, but C++ I/O treats variables of enum types as integers, so the output would be 1.

ကၽြန္ေတာ္တို႔ကေတာ႔ output ဟာ south ျဖစ္လိမ္႔မယ္လို႔ ထင္ၾကမွာပါပဲ။ ဒါေပမယ္႔ C++ I/O က enum variables ေတြကို integers ေတြအေနနဲ႔ပဲ ကိုင္တြယ္တာ ျဖစ္လို႔ output ကို 1 အျဖစ္ပဲ ထုတ္ေပးမွာပါ။

**Other Examples**

ေနာက္ထပ္ enum အသံုးျပဳနည္းမ်ားကို ေအာက္မွာ ဥပမာေလးမ်ားနဲ႔ ျပသေပးထားပါတယ္။ ေလ႔လာၾကည္႔ၾကပါ။

enum months { Jan, Feb, Mar, Apr, May, Jun,

Jul, Aug, Sep, Oct, Nov, Dec };

enum switch { off, on };

enum meridian { am, pm };

enum chess { pawn, knight, bishop, rook, queen, king };

enum coins { penny, nickel, dime, quarter, half-dollar, dollar };

ေနာက္အပတ္ေတြမွာေတာ႔ function အေၾကာင္းကို 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
* "playing card." Encyclopædia Britannica. Encyclopædia Britannica Ultimate Reference Suite.  Chicago: Encyclopædia Britannica, 2012.

Dr. ေအာင္ဝင္းထြဋ္ (bluephoenix)

aungwh2013@gmail.com

http://engineer4myanmar.blogspot.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕