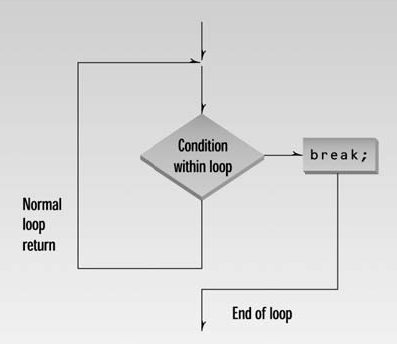
**OOP in C++ : အပိုင္း (၁၇) - Other Control Statements**

C++ မွာ အျခား control statements ေတြ ရွိပါေသးတယ္။ switch ကို ေလ႔လာတုန္းက break statement ကို အသံုးျပဳနည္းကို ေတြ႔ခဲ႔ၾကပါတယ္။ ဒါေပမယ္႔ switch မွာသာမဟုတ္ဘဲ အျခားေနရာေတြမွာလဲ break ကို သံုးလို႔ ရပါေသးတယ္။ continue ကိုေတာ႔ loops ေတြမွာပဲ သံုးလို႔ ရပါတယ္။ goto ကိုေတာ႔ မသံုးသင္႔ေတာ႔ပါဘူး။ အဲဒီ statements ေတြကို အေသးစိတ္ ေလ႔လာၾကည္႔ၾကရေအာင္။

**The break Statement**

switch မွာ ေလ႔လာခဲ႔ၾကသလိုပါပဲ break statement ဟာ loop တစ္ခုကေန ထြက္သြားေစပါတယ္။ ဒါေၾကာင္႔ break ၿပီးေတာ႔ ဆက္လုပ္မယ္႔ statement ဟာ loop အျပင္ဘက္က ပထမဆံုးေတြ႔ရမယ္႔ statement ပဲျဖစ္ပါတယ္။ ပံု (၁၇-၁) မွာ break statement ရဲ႕ လုပ္ေဆာင္ပံုေတြကို flow chat နဲ႔ ရွင္းျပထားပါတယ္။



**ပံု (၁၇-၁) မွာ break statement ရဲ႕ လုပ္ေဆာင္ခ်က္ျပပံု**

To demonstrate break, here’s a program, SHOWPRIM, that displays the distribution of prime numbers in graphical form:

break အေၾကာင္းကို ရွင္းျပဖုိ႔အတြက္ ေအာက္ပါ showprim.cpp ပရိုဂရမ္ေလးကို အသံုးျပဳပါ႔မယ္။ ၄င္းဟာ prime numbers ေတြရဲ႕ distribution ကို graphical form နဲ႔ ျပသေပးမွာ ျဖစ္ပါတယ္။

// showprim.cpp

// displays prime number distribution

#include <iostream>

using namespace std;

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

int main()

{

const unsigned char WHITE = 219; //solid color (primes)

const unsigned char GRAY = 176; //gray (non primes)

unsigned char ch;

//for each screen position

for(int count=0; count<80\*25-1; count++)

{

ch = WHITE; //assume it’s prime

for(int j=2; j<count; j++) //divide by every integer from

if(count%j == 0) //2 on up; if remainder is 0,

{

ch = GRAY; //it’s not prime

break; //break out of inner loop

}

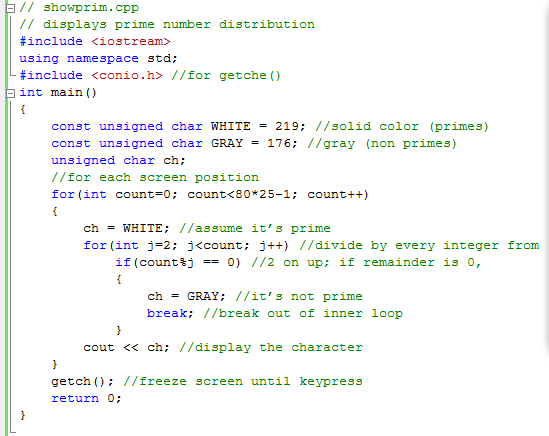
cout << ch; //display the character

}

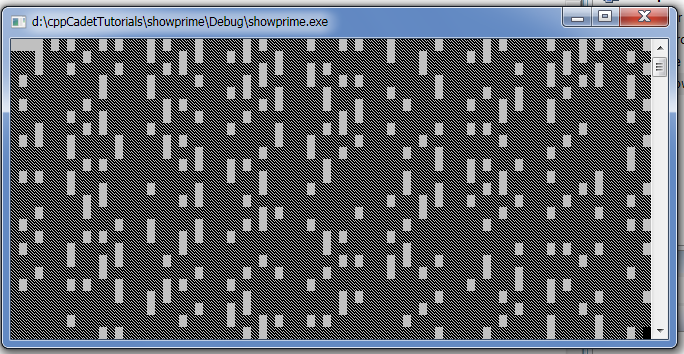
getch(); //freeze screen until keypress

return 0;

}



Console screen ရဲ႕ အက်ယ္ဟာ 80 columns နဲ႔ 25 rows (lines) ျဖစ္တဲ႔ အတြက္ ဂဏန္းအေရအတြက္အရဆိုရင္ 0-1999 (800\*25-1) ရွိပါတယ္။ သက္ဆိုင္ရာ ေနရာမွာ ရွိတဲ႔ ဂဏန္းေတြဟာ prime ျဖစ္ခဲ႔ရင္ အျဖဴေရာင္ ျခယ္ၿပီး မဟုတ္ခဲ႔ရင္ အညိဳေရာင္ျခယ္ေပးမွာပါ။ ပံု (၁၇-၂) မွာ showprime.cpp ရဲ႕ ရလာဒ္ကို ျပသေပးထားပါတယ္။ တိတိက်က်ေျပာရမယ္ဆိုရင္ သုည နဲ႔ တစ္ ဟာ prime ေတြ မဟုတ္ၾကပါဘူး။ ဒါေပမယ္႔ ပရိုဂရမ္ကို ပိုမိုရႈပ္ေထြးမသြားေအာင္ prime မ်ားနည္းတူ အျဖဴေရာင္ျခယ္ေပးထားပါတယ္။ ပထမဆံုး စာေၾကာင္းရဲ႕ ေကာ္လံေတြကို သုည ကေန ၇၉ အထိ ဂဏန္းေတြလို႔ စဥ္းစားၾကည္႔ရေအာင္။ စံုဂဏန္းေတြ ေနရာတိုင္းမွာ ၂ ကလြဲရင္ အားလံုး prime မဟုတ္ၾကပါဘူး။ ၂ နဲ႔ စားလို႔ ျပတ္ေနတာေၾကာင္႔ပါ။ ဒါဆို အျခား ဂဏန္းေတြအတြက္ ပံုေသ ပံုစံ တစ္ခုေရာ မရွိႏိုင္ဘူးလား? တကယ္လို႔သာ ေပးထားတဲ႔ ဂဏန္းေတြကို prime ဟုတ္မဟုတ္ ခန္႔မွန္းႏိုင္မယ္႔ pattern တစ္ခုမ်ား ရွာေတြ႔ႏိုင္ခဲ႔ရင္ ကမာၻ႔ သခၤ်ာ ပညာရွင္ေတြအတြက္ စိတ္လႈပ္ရွားစရာ သတင္းတစ္ခု ျဖစ္သြားႏိုင္ပါတယ္။



ပံု(၁၇-၂) showprim.cpp ၏ ရလာဒ္ကို ေတြ႔ရစဥ္

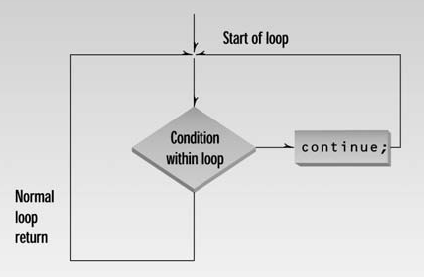
အတြင္းက loop က ဂဏန္းတစ္ခုကို prime မဟုတ္ဘူးလို႔ ဆံုးျဖတ္ခဲ႔ရင္ output ထုတ္ေပးမယ္႔ character ch ကို GRAY လို႔ သတ္မွတ္ေပးၿပီးတာနဲ႔ အဲဒီ inner loop ကေန break သံုးၿပီး ထြက္သြားမွာ ျဖစ္ပါတယ္။ (ပရိုဂရမ္ တစ္ခုလံုးက ထြက္မသြားဘဲ အျခား ဂဏန္းေတြကို စစ္ေဆးဖို႔ လိုေနပါေသးတယ္)။ အဲဒီမွာ သတိထားရမွာက break ဟာ loop ေတြ အားလံုးက ထြက္သြားေစတာ မဟုတ္ဘဲ ၄င္းရွိတဲ႔ အတြင္းအက်ဆံုး loop ကေနပဲ ထြက္သြားေစႏိုင္တာ ျဖစ္ပါတယ္။ ဘယ္လို loop အမ်ိဳးအစားပဲ ငံုထား ငံုထား အဲဒီအတိုင္းပဲ ျပဳမူမွာ ျဖစ္ပါတယ္။ ဥပမာ switch ကို loop တစ္ခုက ငံုထားခဲ႔ရင္ switch ထဲက break statement ဟာ switch ကိုပဲ ထြက္သြားေစျပီး loop ထဲကေတာ႔ ထြက္သြားမွာ မဟုတ္ပါဘူး။ ေနာက္ဆံုး cout statement ကေတာ႔ loop မၿပီးမခ်င္း prime test က သတ္မွတ္ေပးလိုက္တဲ႔ graphics character ေတြကို print ထုတ္ေပးမွာပဲ ျဖစ္ပါတယ္။

**ASCII Extended Character Set**

ဒီပရိုဂရမ္ထဲမွာ 128 ကေန 255 ရွိတဲ႔ *extended ASCII character set* ထဲက character ႏွစ္ခုကို ယူသံုးထားပါတယ္။ 219 ကေတာ႔ အျဖဴေရာင္ အတံုးေလး တစ္ခု ရဲ႕ သေကၤတ ျဖစ္ၿပီး 176 ကေတာ႔ အညိဳေရာင္ အတံုးေလးကို ကိုယ္စားျပဳထားတာပါ။ showprim.cpp နဲ႔ ယခင္က ဥပမာေတြ အေတာ္မ်ားမ်ားမွာ getch() ကို သံုးထားတာ သတိထားမိမွာပါ။ တကယ္ေတာ႔ ၄င္းဟာ DOS prompt မွာ ပရိုဂရမ္ မထြက္သြားခင္ ရလာဒ္ေတြကို ေစာင္႔ၾကည္႔ႏိုင္ဖို႔ keyboard က key တစ္ခုကို ဖတ္ဖို႔ သံုးထားတာပါ။ ဆိုလိုတာက getch() ကို ေရာက္ရင္ keyboard က key တစ္ခုခုကို မႏွိပ္မခ်င္း ပရိုဂရမ္က ရပ္ေနမွာ ျဖစ္ပါတယ္။ ေနာက္တစ္ခ်က္က character variables အတြက္ unsigned char ကို သံုးထားတာ ျဖစ္ပါတယ္။ တကယ္ေတာ႔ char က 127 အထိပဲ ရတဲ႔အတြက္ 255 အထိ သံုးလို႔ရေအာင္ unsigned char ကို သံုးထားတာပါ။

**The continue Statement**

break statement က ကၽြန္ေတာ္တို႔ကို loop ရဲ႕ ေအာက္ေျခကို ေခၚေဆာင္သြားေပးပါတယ္။ တခါတေလမွာ ဆန္႔က်င္ဘက္အေနနဲ႔ မေမွ်ာ္လင္႔တာ တစ္ခုခု ျဖစ္လာခဲ႔ရင္ loop ရဲ႕ ထပ္ဆံုးကို ျပန္သြားဖို႔ လိုအပ္လာတတ္ပါတယ္။ continue ဆိုတဲ႔ statement က အဲဒီ ရလာဒ္ကို ေပးႏိုင္ပါတယ္။ (ဒါေပမယ္႔ အတိအက်ေျပာရရင္ေတာ႔ continue ဟာ loop ရဲ႕ ကြင္းပိတ္ကို အရင္သြားၿပီးမွ ထိပ္ဆံုးကို ျပန္ခုန္တက္လိုက္တာပါ)။ ပံု(၁၇-၃) မွာ continue အလုပ္လုပ္ပံုကို flow chat နဲ႔ ျပထားပါတယ္။



ပံု(၁၇-၃) continue အလုပ္လုပ္ပံု

ယခင္ ဥပမာေတြ ေပးတုန္းက divdo.cpp ဆိုတဲ႔ ပရိုဂရမ္ေလးကို မွတ္မိဦးမယ္ ထင္ပါတယ္။ အဲဒီ ပရိုဂရမ္ေလးမွာ ကိန္းဂဏန္းေတြကို စားျပထားပါတယ္။ ဒါေပမယ္႔ သူ႔မွာ အဓိက အားနည္းခ်က္ တစ္ခု ရွိပါတယ္။ အဲဒါကေတာ႔ စားကိန္းကို သုည ေပးခဲ႔မိရင္ Divide Error ဆိုတဲ႔ runtime error message ေပၚလာၿပီး ပရိုဂရမ္ ထြက္သြားမွာပါ။ အခု ဥပမာေပးမယ္႔ divdo2.cpp မွာေတာ႔ အဲဒီ အေျခအေနကို သိမ္သိမ္ေမြ႔ေမြ႔ေလး ေျဖရွင္းေပးထားပါတယ္။

// divdo2.cpp

// demonstrates CONTINUE statement

#include <iostream>

using namespace std;

int main()

{

long dividend, divisor;

char ch;

do {

cout << “Enter dividend: “; cin >> dividend;

cout << “Enter divisor: “; cin >> divisor;

if( divisor == 0 ) //if attempt to

{ //divide by 0,

cout << “Illegal divisor\n”; //display message

continue; //go to top of loop

}

cout << “Quotient is “ << dividend / divisor;

cout << “, remainder is “ << dividend % divisor;

cout << “\nDo another? (y/n): “;

cin >> ch;

} while( ch != ‘n’ );

return 0;

}

တကယ္လို႔ user က စားကိန္းကို သုည ထည္႔ေပးခဲ႔မယ္ဆိုရင္ program က error message ထုတ္ေပးၿပီး continue အသံုးျပဳကာ loop ရဲ႕ ထိပ္ဆံုးပိုင္းကို ျပန္သြားလို႔ ဂဏန္း ထပ္ေတာင္းမွာ ျဖစ္ပါတယ္။

Enter dividend: 10

Enter divisor: 0

Illegal divisor

Enter dividend:

တကယ္လို႔ ဒီေနရာမွာ break ကို သံုးခဲ႔မိရင္ do loop ကေရာ ပရိုဂရမ္ကပါ ထြက္သြားမွာ ျဖစ္ပါတယ္။

ေနာက္အပတ္ေတြမွာေတာ႔ Loops and Desicions အေၾကာင္းကို အဆံုးသတ္မၿပီး Structures မ်ားအေၾကာင္းကို 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