**OOP in C++ : အပိုင္း (၁၀) - while loop**

**The while Loop**

အရင္အပတ္တုန္းက တိက်တဲ႔ အႀကိမ္အေရအတြက္ အတိုင္း looping ပတ္ဖို႔အတြက္ for loop ကို အသံုးနည္းမ်ားကို ေလ႔လာခဲ႔ၿပီးပါၿပီ။ ဒါေပမယ္႔ တကယ္လို႔ ကၽြန္ေတာ္တို႔ ပတ္မယ္႔ loop အေရအတြက္ကို ႀကိဳမသိႏိုင္ခဲ႔ဘူးဆိုရင္ အျခားနည္းကို သံုးရပါလိမ္႔မယ္။ အဲဒါကေတာ႔ while loop ပဲ ျဖစ္ပါတယ္။ ဥပမာအေနနဲ႔ endon0.cpp ပရိုဂရမ္ကို ေရးျပထားပါတယ္။ အဲဒီ မွာ user ကို ဂဏန္းေတြ တစ္ခုၿပီး တစ္ခု ေတာင္းပါမယ္။ တကယ္လို႔ user က 0 ကို ထည္႔သြင္းလိုက္တဲ႔ အခါမွသာ ပရိုဂရမ္ ၿပီးဆံုးသြားမွာ ျဖစ္ပါတယ္။ ဒီေနရာမွာ ပရိုဂရမ္ အေနနဲ႔ user က 0 မထည္႔သြင္းခင္ ဘယ္ႏွစ္ႀကိမ္ ဂဏန္းေတြ ထည္႔ေပးမယ္ဆိုတာကို ႀကိဳမသိပါဘူး။ user စိတ္ႀကိဳက္ ထည္႔သြင္းႏိုင္ပါလိမ္႔မယ္။

// endon0.cpp

// demonstrates WHILE loop

#include <iostream>

using namespace std;

int main()

{

int n = 99; // make sure n isn’t initialized to 0

while( n != 0 ) // loop until n is 0

cin >> n; // read a number into n

cout << endl;

return 0;

}

ေအာက္မွာေတာ႔ နမူနာအေနနဲ႔ ပရိုဂရမ္ထဲကို ဂဏန္းေတြ ထည္႔သြင္းျပထားပါတယ္။ 0 ကို ထည္႔သြင္းလိုက္တဲ႔ အခါမွာေတာ႔ loop ကေနထြက္ၿပီး ပရိုဂရမ္ ၿပီးဆံုးသြားမွာ ျဖစ္ပါတယ္။

1

27

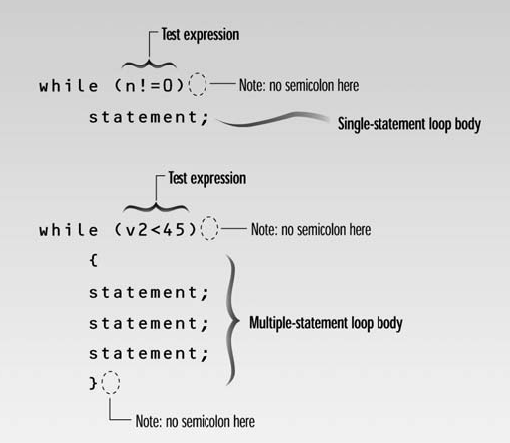
33

144

9

0

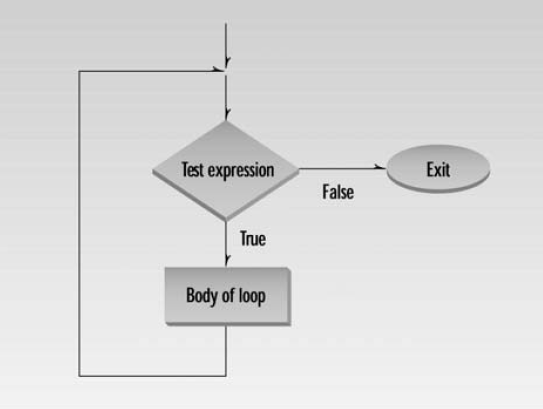
while loop ဟာ for loop ကို ပိုမိုရိုးရွင္းေအာင္ ျပဳလုပ္ထားတာနဲ႔ တူေနပါတယ္။ သူ႔မွာ test expression သာပါရွိၿပီး initialization နဲ႔ increment expression ေတြ မပါဝင္ပါဘူး။ ေအာက္က ပံုေလးမွာ while loop ရဲ႕ syntax ကို ျပထားပါတယ္။



while loop မွာ test expression မွန္ေနသမွ် ကာလပတ္လံုး loop ကို ပတ္ေနမွာ ျဖစ္ပါတယ္။ endon0.cpp မွာပါရွိတဲ႔ n!=0 (n note equal to 0) ဆိုတဲ႔ test expression ေၾကာင္႔ n တန္ဖိုး 0 မျဖစ္မခ်င္း test result မွန္ကန္ေနၿပီး loop ကို ပတ္ေနမွာ ျဖစ္ပါတယ္။ while loop မွာ initialization expression မပါေပမယ္႔ loop variable (ဒီဥပမာမွာေတာ႔ n) ကို loop မစခင္မွာ initialize လုပ္ေပးထားဖို႔ လိုအပ္မွာ ျဖစ္ပါတယ္။ ေနာက္တစ္ခုက loop body ထဲမွာ အဲဒီ loop variable ကို ေျပာင္းလဲေပးတဲ႔ statement ပါဖို႔လိုအပ္ပါတယ္။ မပါခဲ႔ရင္ loop က ထြက္ဖို႔ မျဖစ္ႏိုင္ေတာ႔ပါဘူး။ endon0.cpp မွာေတာ႔ cin<<n; ဆိုတဲ႔ ကုဒ္က n တန္ဖိုးကို ေျပာင္းလဲေစပါတယ္။

**Multiple Statements in a while Loop**

ေနာက္ထပ္ နမူနာ ပရိုဂရမ္ တစ္ခုကို ေရးျပပါဦးမယ္။ while4.cpp မွာ while loop အတြင္း multiple statements ေတြ ေရးျပထားပါတယ္။ တကယ္ေတာ႔ ဒီလို multiple statements ေတြ ေရးတဲ႔ အေၾကာင္းကို cubelist.cpp မွာတုန္းက foor loop ထဲမွာ ေရးျပခဲ႔ၿပီးပါၿပီ။ ဒီပရိုဂရမ္မွာေတာ႔ ေလးထပ္ကိန္းေတြကို ရွာျပထားပါတယ္။ ေလးထပ္ကိန္းရဲ႕ တန္ဖိုးေတြကို ဂဏန္းေလးလံုး အထိပဲ ရွာဖို႔ ဆံုးျဖတ္ထားတယ္ ဆိုၾကပါစို႔။ ဒါဆိုရင္ အမ်ားဆံုး တန္ဖိုးက 9999 ထက္ မႀကီးရပါဘူး။ အဲဒီလို အေျခအေနမ်ိဳးကို ကန္႔သတ္ဖို႔ အတြက္ဆိုရင္ တြက္မၾကည္႔ရေသးပဲနဲ႔ ဘယ္ဂဏန္းရဲ႕ ေလးထပ္ကိန္းက အဲဒီ limit ကို ေက်ာ္သြားမယ္ဆိုတာကို မသိႏိုင္ပါဘူး။ ဒါေၾကာင္႔ test expression မွာ အဲဒီ အခ်က္ကို စစ္ေဆးခိုင္းၿပီး limit ေက်ာ္သြားတာနဲ႔ loop ကို ရပ္လိုက္မွာ ျဖစ္ပါတယ္။



// while4.cpp

// prints numbers raised to fourth power

#include <iostream>

#include <iomanip> //for setw

using namespace std;

int main()

{

int pow=1; //power initially 1

int numb=1; //numb goes from 1 to ???

while( pow<10000 ) //loop while power <= 4 digits

{

cout << setw(2) << numb; //display number

cout << setw(5) << pow << endl; //display fourth power

++numb; //get ready for next power

pow = numb\*numb\*numb\*numb; //calculate fourth power

}

cout << endl;

return 0;

}

ေလးထပ္ကိန္းကို ရွာဖို႔အတြက္ကေတာ႔ အဲဒီဂဏန္းကို ေလးခါ ေျမွာက္ေပးလိုက္တာပါ။ loop တစ္ခါပတ္တိုင္း numb ဆိုတဲ႔ variable ကို 1 တိုးေပးသြားမွာပါ။ ဒါေပမယ္႔ test expression မွာေတာ႔ numb ကို မစစ္ဘဲ ေလးထပ္ကိန္းတန္ဖိုး pow ကိုပဲ စစ္ေဆးၿပီး loop ကို ဘယ္အခ်ိန္မွာ ထြက္ရမယ္ဆိုတာကို ဆံုးျဖတ္သြားမွာပါ။ ေအာက္မွာ မူရင္းဂဏန္းေတြနဲ႔ သူတို႔ရဲ႕ ေလးထပ္ကိန္းေတြကို တြက္ျပထားပါတယ္။ ေလးထပ္ကိန္းတန္ဖိုး 9999 ထက္ႀကီးသြားတာနဲ႔ ပရိုဂရမ္ၿပီးဆံုးသြားမွာ ျဖစ္ပါတယ္။

1 1

2 16

3 81

4 256

5 625

6 1296

7 2401

8 4096

9 6561

**Precedence: Arithmetic and Relational Operators**

ေနာက္ပရိုဂရမ္ တစ္ပုဒ္ကေတာ႔ Fibonacci series ကို တြက္ထုတ္ေပးမယ္႔ fibo.ccp ပဲ ျဖစ္ပါတယ္။ Fibonacci series ဆိုတာဟာ ေရွ႕ဂဏန္း ႏွစ္လံုးေပါင္းျခင္းျဖင္႔ ေနာက္ဂဏန္း ကို ရွာယူထားတဲ႔ ဂဏန္းတြဲေတြ ျဖစ္ပါတယ္။ ၄င္းရဲ႕ အစပိုင္း ဂဏန္းတစ္ခ်ိဳ႕ကေတာ႔ 1 1 2 3 5 8 13 21 34 55 တို႔ပဲ ျဖစ္ပါတယ္။ တကယ္ေတာ႔ ဒီ series ကို လက္ေတြ႔ အသံုးခ်မႈေပါင္းမ်ားစြာ ရွိပါတယ္။ ၄င္းဟာ ဗိသုကာပညာနဲ႔ အႏုပညာမွာ သံုးေလ႔ရွိတဲ႔ golden ratio နဲ႔လဲ ဆက္စပ္ေနပါေသးတယ္။ ဒီေနရာမွာေတာ႔ အဲဒီအေၾကာင္းေတြကို အက်ယ္ရွင္းမျပေတာ႔ပါဘူး။

// fibo.cpp

// demonstrates WHILE loops using fibonacci series

#include <iostream>

using namespace std;

int main()

{ //largest unsigned long

const unsigned long limit = 4294967295;

unsigned long next=0; //next-to-last term

unsigned long last=1; //last term

while( next < limit / 2 ) //don’t let results get too big

{

cout << last << “ “; //display last term

long sum = next + last; //add last two terms

next = last; //variables move forward

last = sum; // in the series

}

cout << endl;

return 0;

}

ရလာဒ္ေတြကေတာ႔ ေအာက္ပါအတိုင္းပဲ ျဖစ္ပါတယ္။

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

1597 2584 4181 6765 10946 17711 28657 46368 75025 121393

196418 317811 514229 832040 1346269 2178309 3524578

5702887 9227465 14930352 24157817 39088169 63245986

102334155 165580141 267914296 433494437 701408733 1134903170

1836311903 2971215073

အဲဒီရလာဒ္ေတြထဲက ေနာက္ဆံုး ဂဏန္းႏွစ္လံုးကို အခ်ိဳးခ်လိုက္မယ္ဆိုရင္ 0.618033988 ကို ရရွိမယ္ျဖစ္ၿပီး ၄င္းဟာ golden ratio ပဲျဖစ္ပါတယ္။ fibo.cpp ပရိုဂရမ္မွာ အႀကီးဆံုး positive integers ေတြ ထည္႔ထားႏိုင္ဖို႔ unsigned long data type ကို အသံုးျပဳထားပါတယ္။

Test expression မွာ အဲဒီ data type ရဲ႕ ကန္႔သတ္ခ်က္ကို ေက်ာ္မသြားတဲ႔ အခ်ိန္အထိ loop ကို ပတ္ေနေအာင္ ေရးသားထားပါတယ္။ limit တန္ဖိုးကိုေတာ႔ const သတ္မွတ္ေပးထားၿပီး မေတာ္တဆ ေျပာင္းလဲေရးသားမိျခင္းကေန ကာကြယ္ထားပါသည္။ loop ကို limit တန္ဖိုးရဲ႕ တစ္ဝက္ေရာက္လာရင္ ထြက္ဖို႔ ေရးသားရပါတယ္။ ဒါမွလဲ sum တန္ဖိုးက limit ကို ေက်ာ္မသြားမွာ ျဖစ္ပါသည္။ test expression မွာ operators ႏွစ္ခု သံုးထားပါတယ္။ (next < limit / 2)။ ကၽြန္ေတာ္တို႔ရဲ႕ ရည္ရြယ္ခ်က္က next ကို limit/2 နဲ႔ ႏိႈင္းယွဥ္ဖို႔ ျဖစ္ပါတယ္။ division ကို comparison မတိုင္မီ လုပ္ေဆာင္ဖို႔ လိုပါတယ္။ အဲဒီအခ်က္ကို ေသခ်ာေစဖို႔ လက္သဲကြင္းထဲ ထည္႔ေရးလို႔ရပါတယ္။ (next < (limit/2) )။ ဒါေပမယ္႔ လက္သဲကြင္းကို ထည္႔ေရးဖို႔ မလိုပါဘူး။ arithmetic operators ေတြဟာ relational operators ေတြထက္ precedence ပိုျမင္႔လို႔ပဲ ျဖစ္ပါတယ္။

ေနာက္အပတ္ေတြမွာေတာ႔ do loop အေၾကာင္းကို ဆက္လက္ ေဆြးေႏြးသြားပါမယ္ခင္ဗ်ာ။ ဤအခန္းဆက္ ေဆာင္းပါးမ်ားဟာ IT ေက်ာင္းသားမ်ားအတြက္ အထူးရည္ရြယ္ၿပီး ေရးသားထားတာ ျဖစ္ပါတယ္။ 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