The Zen of C++

2nd Edition

Adnan Zejnilovic

Chapter 4

Loops

Introduction

- What is a loop?
  - A repetitive statement that executes as long as certain condition is true
  - Similar to conditional statement, except that it repeats
  - Depending on where the condition is evaluated loops could be classified as
    - Pre-test loops
    - Post-test loops

Loop Classifications

- Depending on where the condition (test) takes place, loops could be
  - Pre-test loop
    - Evaluate the expression (condition) first and determine whether to execute the body of the loop
    - No guarantees that will execute at all
      - May execute or may not execute depending on how the expression (condition) evaluates
  - Post-test loops
    - Executes the body of the loop at least once
    - Tests the expression (condition) to determine whether it needs to execute more than once
    - Guaranteed to execute at least once

The While Loop

Pre-test loop

Syntax:

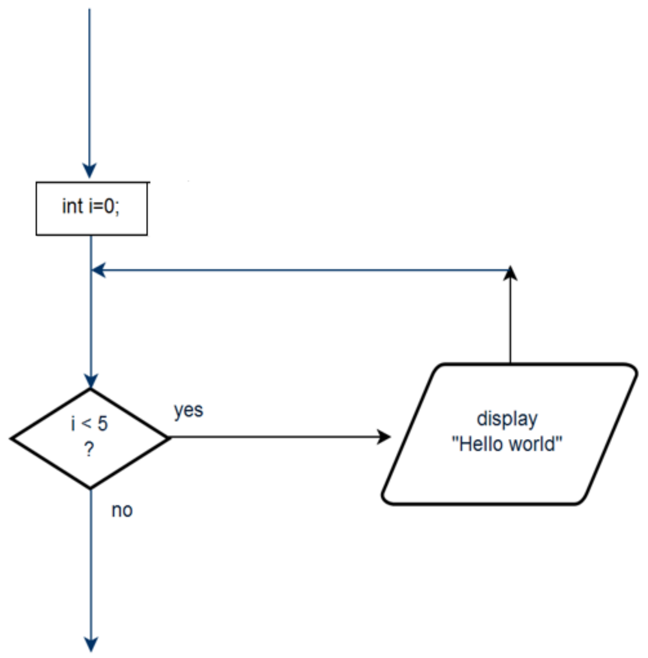
- Keyword “**while**” (expression/condition) followed by
- The body of the loop

Example:

Notice: no “;” at the end of the expression

```
int i=0;
while (i < 5)
     cout << “Hello world”;
```

The While Loop contd.



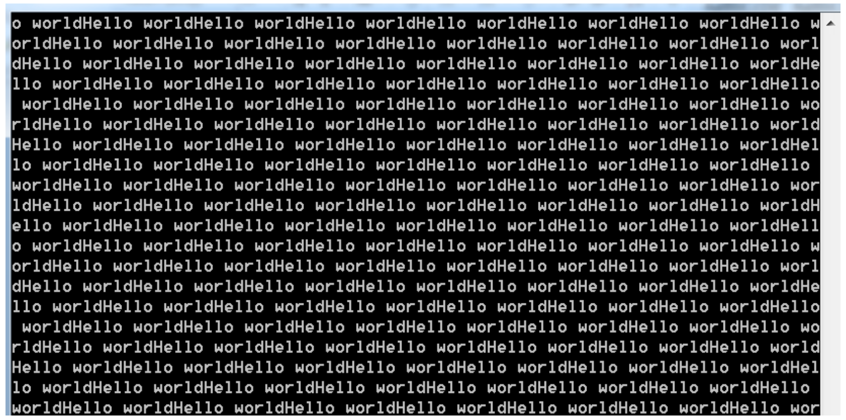
```
int i=0;
while (i < 5)
     cout << “Hello world”;
```

```
Step1: Evaluate the expression (i< 5)
Step2: if true,execute the body of the loop
Step3: go to Step1.
```

Infinite Loop

```
int i=0;
while (i < 5)
     cout << "Hello world";
```

- This code would result in an infinite loop
- The reason?
- i is always zero
- Every time the test is performed (i<5) the answer is YES
- Question: How do we correct this?



```
o worldHello worldHello worldHello worldHello worldHello worldHello worldHello w
orldHello worldHello worldHello worldHello worldHello worldHello worldHello worl
dHello worldHello worldHello worldHello worldHello worldHello worldHello worldHe
llo worldHello worldHello worldHello worldHello worldHello worldHello worldHello
 worldHello worldHello worldHello worldHello worldHello worldHello worldHello wo
rldHello worldHello worldHello worldHello worldHello worldHello worldHello world
Hello worldHello worldHello worldHello worldHello worldHello worldHello worldHel
lo worldHello worldHello worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello worldHello worldHello wor
ldHello worldHello worldHello worldHello worldHello worldHello worldHello worldH
ello worldHello worldHello worldHello worldHello worldHello worldHello worldHell
o worldHello worldHello worldHello worldHello worldHello worldHello worldHello w
orldHello worldHello worldHello worldHello worldHello worldHello worldHello worl
dHello worldHello worldHello worldHello worldHello worldHello worldHello worldHe
llo worldHello worldHello worldHello worldHello worldHello worldHello worldHello
 worldHello worldHello worldHello worldHello worldHello worldHello worldHello wo
rldHello worldHello worldHello worldHello worldHello worldHello worldHello world
Hello worldHello worldHello worldHello worldHello worldHello worldHello worldHel
lo worldHello worldHello worldHello worldHello worldHello worldHello worldHello
worldHello worldHello worldHello worldHello worldHello worldHello worldHello wor
```

Infinite Loop Corrected

```
int i=0;
while (i < 5)
{
      cout << "Hello world";
      i++;
}
```

1. Need to find a way to stop the loop from executing!

   How?

2. **increment i !**
   With each iteration, i will get incremented (updated):
   i=0……is 0 < 5? Yes. Display "Hello World". Increment i.
   i=1….. Is 1 < 5? Yes. Display "Hello world". Increment i.
   i=2……is 2 < 5? Yes. Display "Hello world". Increment i.
   i=3……is 3 < 5? Yes. Display "Hello world". Increment i.
   i=4……is 4 < 5? Yes. Display "Hello world". Increment i.
   i=5……is 5 < 5? No. Skip the body of th e loop

Counters

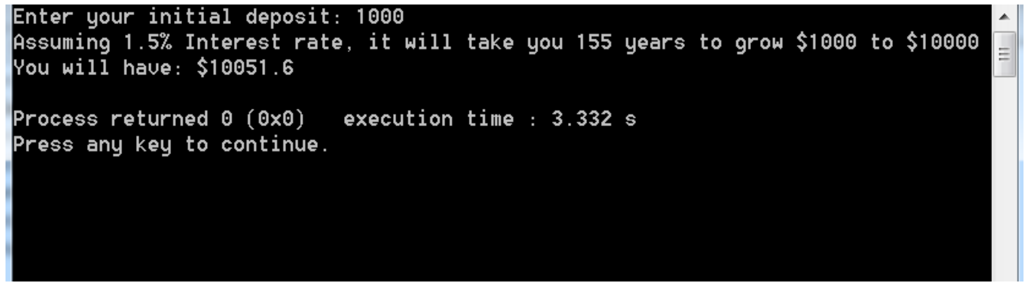
- Sometimes you will need to count how many times your loop executes.
- Examples:
  - Count how many customers enter your store during business hours
  - Count how many cars enter the toll booth
  - Count how many years (days/months/hours/minutes) some process takes until it completes
- A counter needs to have a starting (“initial”) value
- Usually the initial value is set to zero
- The initial value is incremented by one with each iteration of the loop

Counters

- Example:
  - Determine how many years will it take for a certain amount of money (principal) to reach maturity
  - Assume the bank gives you 1.5% interest
  - Assume maturity is $10,000

```cpp
int main()
{
    const double TEN_K = 10000.00;
    const double INTEREST = 0.015;     // 1.5% intestest
    int numYears = 0;
    double principalAmt, initialAmt,interest;

    cout << "Enter your initial deposit: ";
    cin >> initialAmt;
    principalAmt = initialAmt;  // copy initialAmt to principalAmt for the purpose of reporti
    while (principalAmt < TEN_K)
    {
        interest = principalAmt * INTEREST;
        principalAmt = principalAmt + interest;
        numYears = numYears + 1;
    }

    cout << "Assuming " << INTEREST * 100
         << "% Interest rate, it will take you " << numYears
         << " years to grow $" << initialAmt
         << " to $" << TEN_K << endl;

    cout << "You will have: $" << principalAmt << endl;
    return 0;
}
```

Output:



```
Enter your initial deposit: 1000
Assuming 1.5% Interest rate, it will take you 155 years to grow $1000 to $10000
You will have: $10051.6

Process returned 0 (0x0)   execution time : 3.332 s
Press any key to continue.
```

Accumulators

- Accumulators are identical to counters with one difference
- Both counters and accumulators must have initial value which is then incremented or decremented in the loop depending on programming logic.
- The difference is that accumulators are used to keep a running total
- When their value is incremented/decremented, it is incremented/decremented by the value of the variable being accumulated

Sentinels

- A Sentinel is a special value that is used to mark the end of user input.
- Usually this is a value that is not typical for the data being collected.
- Example:
    - Keeping track of goals scored by a soccer player.
    - A Sentinel value could be set to -99 because nobody scores negative goals, and even worse, 99 of them!

Sentinels – Example Program

This program tracks number of customers that enter the store ---------- (counter)
It also “accumulates” each customer deposit --------------------------------(accumulator)
Lastly, it terminates when -99 is entered as billAmt -------------------------(sentinel)

```
#include <iostream>

using namespace std;
const int MY_SENTINEL = -99;
int main()
{
    double billAmt,
           totalProfit = 0.0;              // accumulator
    int    numCustomers = 0;               // counter
    cout << "Enter the bill amount: ";
    cin >> billAmt;                        // priming read

    while (billAmt != MY_SENTINEL)
    {
        totalProfit += billAmt;
        numCustomers++;
        cout << "Enter the bill amount: ";
        cin >> billAmt;
    }

    cout << numCustomers << " customer visits. Total profit: " << totalProfit << endl;
    return 0;
}
```

Priming Read

- "priming read" (prime read) is a read (data capture) ahead of the loop.
- Priming read is used to determine whether to enter the loop or not
- Used with "pre-test" loops

```
cout << "Enter the bill amount: ";
cin >> billAmt;                           // priming read

while (billAmt != MY_SENTINEL)
{
    totalProfit += billAmt;
    numCustomers++;
    cout << "Enter the bill amount: ";
    cin >> billAmt;
}
```

For Loop

- When you know ahead of time how many iterations your program is going to need to accomplish a certain task you can use a so-called ***count*** controlled loop.

```
SYNTAX:
    [1]initialization
    [2]test
    [3]body of the loop
    [4]update

    for ([1]initialization; [2]test; [4]update )
    {
        [3] body of the loop

    }
```

The numbers represent the order of execution of statements in the for loop”

For Loop

```
SYNTAX:
    [1]initialization
    [2]test
    [3]body of the loop
    [4]update

    for ([1]initialization; [2]test; [4]update )
    {
        [3] body of the loop

    }
```

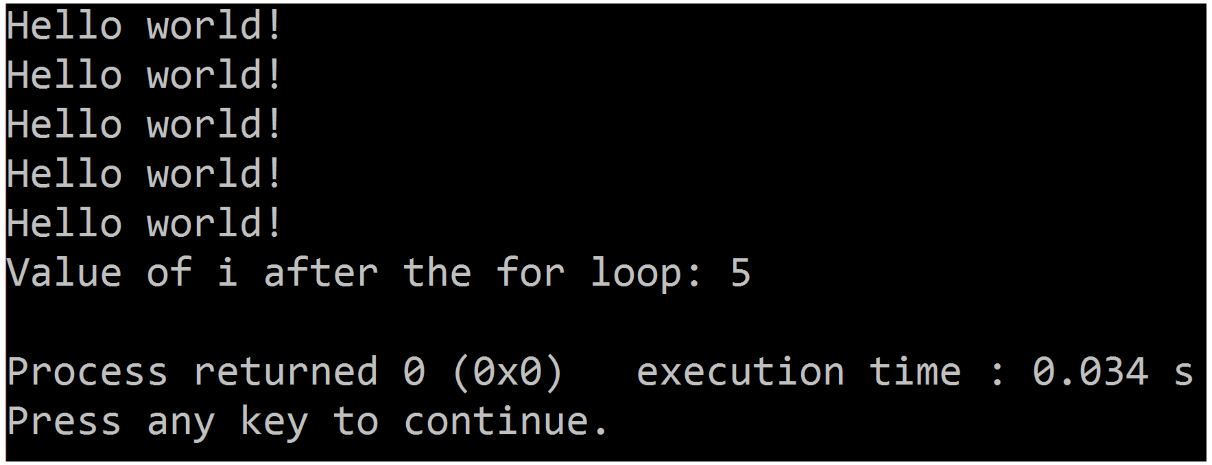
```
for (int i=0; i<5; i++)
{
    cout << "Hello world!" << endl;
}
```

Section	Statement
[1] initialization	int i=0;
[2] test/condition	i<5;
[3] body	cout << “Hello world” << endl;
[4] update	i++

For Loop

```
int i;   // declare i
for (i=0; i<5; i++)
{
    cout << "Hello world!" << endl;
}
cout << "Value of i after the for loop: " << i << endl;
```

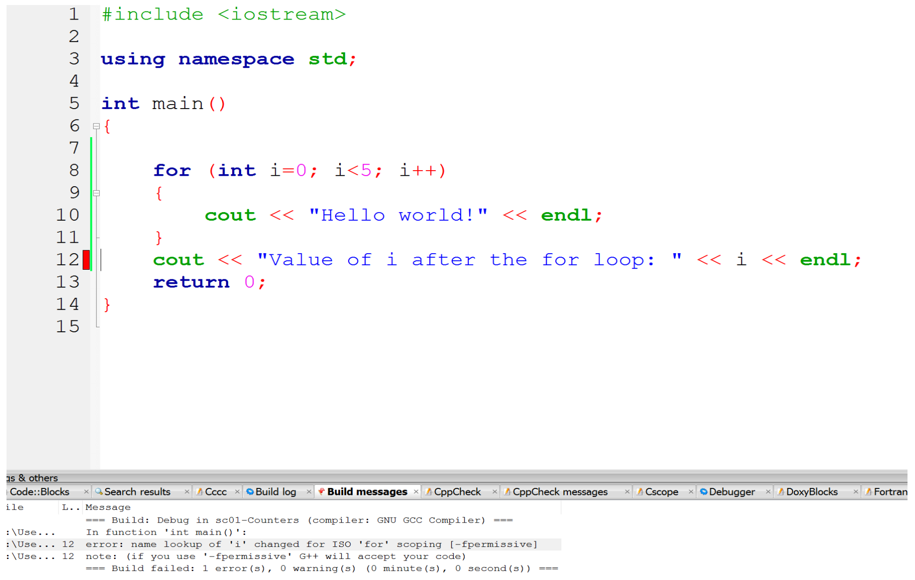
- Variable “i” is in scope even after the loop terminates.
- The program compiles and displays value of “i”:



```
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Value of i after the for loop: 5

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

```
for (int i=0; i<5; i++)
{
    cout << "Hello world!" << endl;
}
cout << "Value of i after the for loop: " << i << endl;
```

- Variable “i” is local to the loop.
- We get a syntax error if we try to display it afte
  the loop



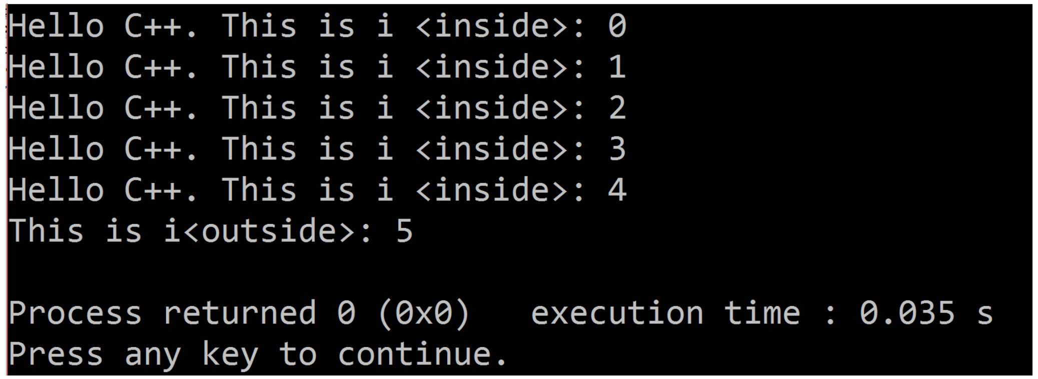
```
 1  #include <iostream>
 2
 3  using namespace std;
 4
 5  int main()
 6  {
 7
 8      for (int i=0; i<5; i++)
 9      {
10          cout << "Hello world!" << endl;
11      }
12      cout << "Value of i after the for loop: " << i << endl;
13      return 0;
14  }
15
```

```
ile        L..  Message
                === Build: Debug in sc01-Counters (compiler: GNU GCC Compiler) ===
:\Use...        In function 'int main()':
:\Use... 12     error: name lookup of 'i' changed for ISO 'for' scoping [-fpermissive]
:\Use... 12     note: (if you use '-fpermissive' G++ will accept your code)
                === Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

For Loop – Omitting Sections

- You could omit sections (initialization, test, or update) and still have your loop execute.

```
int i=0;   // initialization done outside of the for loop
for ( ; i<5; i++)     // ommited initialization section
{
    cout << "Hello C++. ";
    cout << "This is i <inside>: " << i << endl;
}

cout << "This is i<outside>: " << i << endl;
```

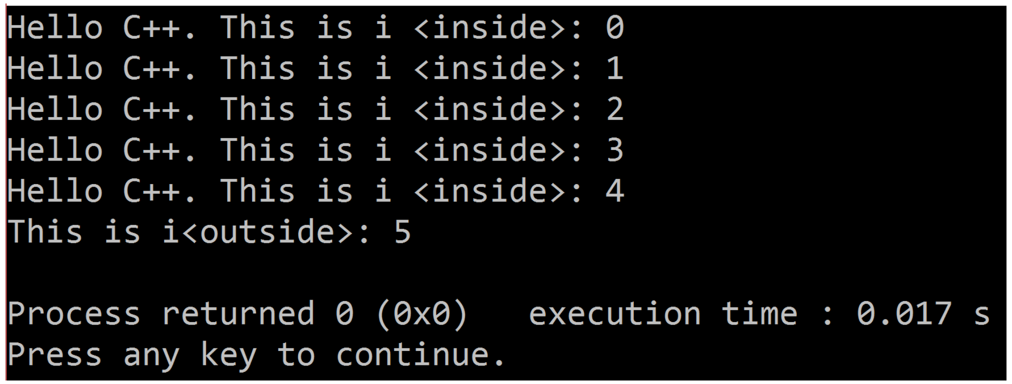


```
Hello C++. This is i <inside>: 0
Hello C++. This is i <inside>: 1
Hello C++. This is i <inside>: 2
Hello C++. This is i <inside>: 3
Hello C++. This is i <inside>: 4
This is i<outside>: 5

Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```

For Loop – Omitting Sections

```
int i=0;  // initialization done outside of the for loop
for ( ; ; i++)     // omitted initialization and test section
{
     if (i<5)
     {
          cout << "Hello C++. ";
          cout << "This is i <inside>: " << i << endl;
     }
     else
          break;
}
```

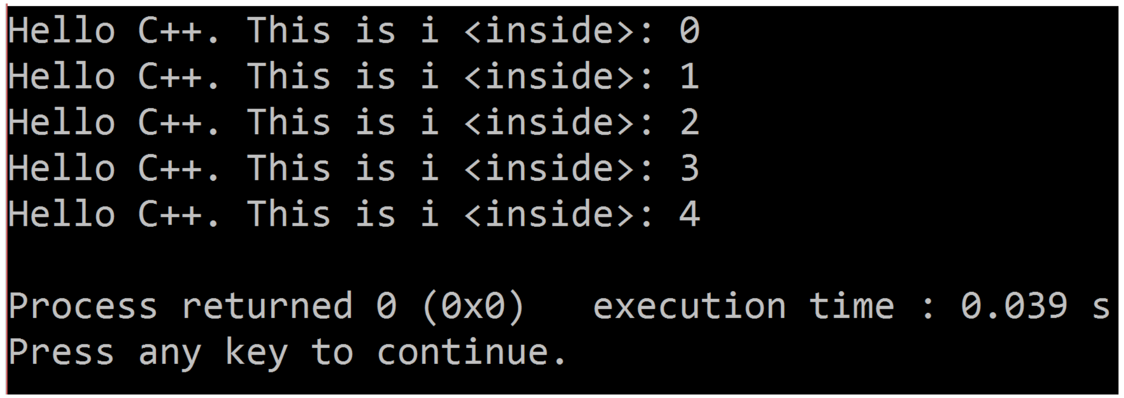
Use the break statement to terminate the loop



```
Hello C++. This is i <inside>: 0
Hello C++. This is i <inside>: 1
Hello C++. This is i <inside>: 2
Hello C++. This is i <inside>: 3
Hello C++. This is i <inside>: 4
This is i<outside>: 5

Process returned 0 (0x0)   execution time : 0.017 s
Press any key to continue.
```

For Loop – Omitting Sections

```cpp
for (int i=0; i<5; )      // omitted update section
{
    cout << "Hello C++. ";
    cout << "This is i <inside>: " << i << endl;
    i++;        // performing update here
}
```

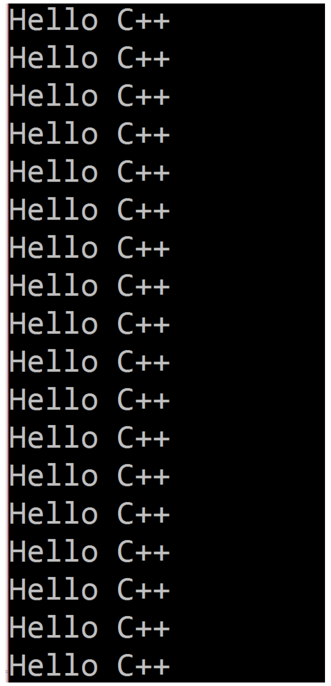


```
Hello C++. This is i <inside>: 0
Hello C++. This is i <inside>: 1
Hello C++. This is i <inside>: 2
Hello C++. This is i <inside>: 3
Hello C++. This is i <inside>: 4

Process returned 0 (0x0)   execution time : 0.039 s
Press any key to continue.
```

For Loop – Omitting Sections

```
for ( ; ; )     // ommited all sections
{
    cout << "Hello C++" << endl;
}
```

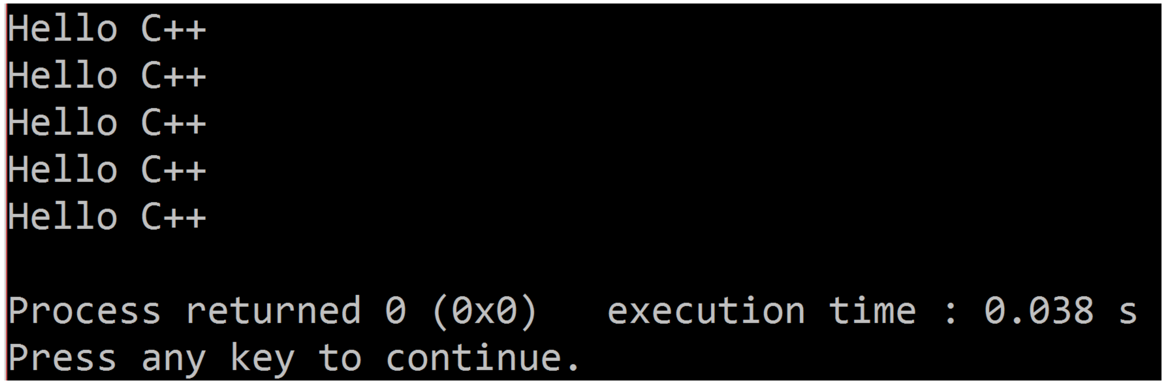
Results in an infinite loop:



```
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++
```

- Use the break statement to terminate the loop

```
int i=0;
for ( ; ; )     // ommited all sections
{
    cout << "Hello C++" << endl;
    if (i==4)
        break;   // break out of the loop
    i++;         // increment i;
}
```



```
Hello C++
Hello C++
Hello C++
Hello C++
Hello C++

Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

What Not to Do

Do not put a “;” at the end of the loop header

```
for (int i=0; i<5;i++ );
```

- While it is syntactically correct, the body of the loop will not execute

Do not update loop control variable in the body of the loop:

```
for (int i=0; i<5;i++)
{
    cout << "Hello C++. ";
    cout << "This is i <inside>: " << i << endl;
    i+=2;      // another update here
}
```

- As a programmer you want to control your code

Rewriting a for Loop as a while Loop

- Both are pretest loops so it is possible to rewrite for as a while:

```
int numGames;         // number of games played
int totalPoints = 0; // accumulator
int points;           // points scored in a game
double avg;           // average points per game


cout << "How many games: ";
cin >> numGames;
      [1] Initialization        [2] Test          [4] Update
for (int game=0; game < numGames; game++)
{
   cout << "Game #" << game+1 << " points: ";     [3] Body
   cin >> points;
   totalPoints = totalPoints + points; // add points to the accumulator
}

// calculate the average
avg = static_cast<double>(totalPoints)/numGames;

// display the result
cout << "The player averaged: " << avg << " points in "
     << numGames << " games" << endl;
```

Rewritten as a while loop:

```
//[1] initialization
int game=0;

//[2] condition
while (game < numGames)
{
    //[3] statement(s)
    cout << "Game #" << game+1 << " points: ";
    cin >> points;

    totalPoints = totalPoints + points;

    // [4] update
    game++;
}


// calculate the average
avg = static_cast<double>(totalPoints)/numGames;

// display the result
// display the result
cout << "The player averaged: " << avg << " points in "
     << numGames << " games" << endl;
```

Rewriting a while Loop as a for Loop

while loop

[1] initialization

[2] Test

[3] Body

[4] Update

```
int j=0;
while (j<10)
{
    cout << "This is j: " << j << endl;
    j++;
}
```

for loop

[1] initialization

[2] Test

[4] Update

[3] Body

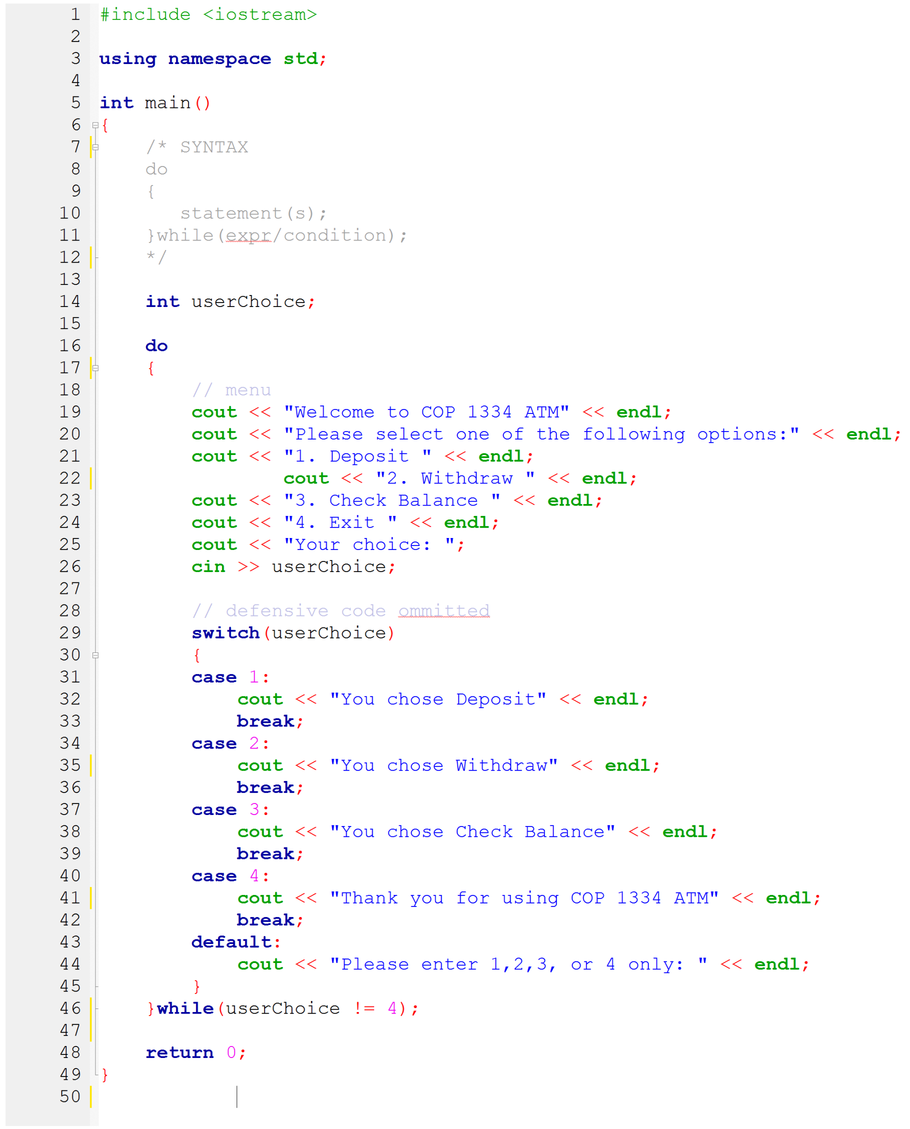
```
j=0;
for (; j<10; j++)
{
    cout << "This is j: " << j << endl;
}
```

The do-while Loop

The general format of the do while loop is:

```
do
{
    statement(s);
} while (expression);
```

- This is a post-test loop
- Guaranteed to execute at least once

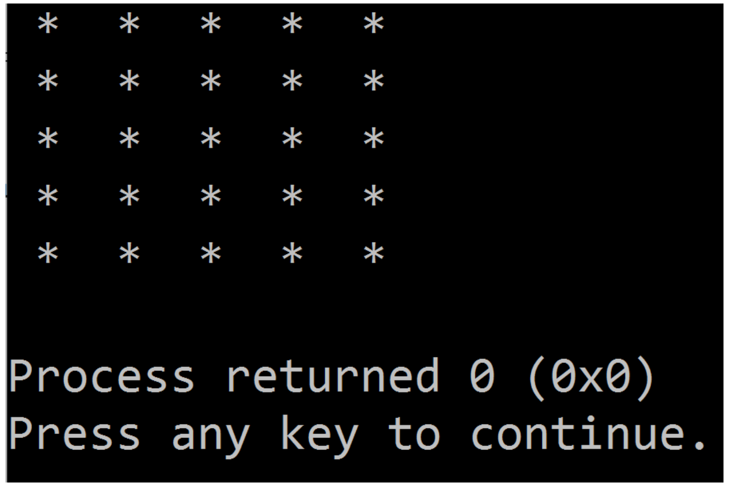


```cpp
#include <iostream>

using namespace std;

int main()
{
    /* SYNTAX
    do
    {
       statement(s);
    }while(expr/condition);
    */

    int userChoice;

    do
    {
        // menu
        cout << "Welcome to COP 1334 ATM" << endl;
        cout << "Please select one of the following options:" << endl;
        cout << "1. Deposit " << endl;
                cout << "2. Withdraw " << endl;
        cout << "3. Check Balance " << endl;
        cout << "4. Exit " << endl;
        cout << "Your choice: ";
        cin >> userChoice;

        // defensive code ommitted
        switch(userChoice)
        {
        case 1:
            cout << "You chose Deposit" << endl;
            break;
        case 2:
            cout << "You chose Withdraw" << endl;
            break;
        case 3:
            cout << "You chose Check Balance" << endl;
            break;
        case 4:
            cout << "Thank you for using COP 1334 ATM" << endl;
            break;
        default:
            cout << "Please enter 1,2,3, or 4 only: " << endl;
        }
    }while(userChoice != 4);

    return 0;
}
```

Nested Loops

- Could be used to print data in a table
- Print a pattern

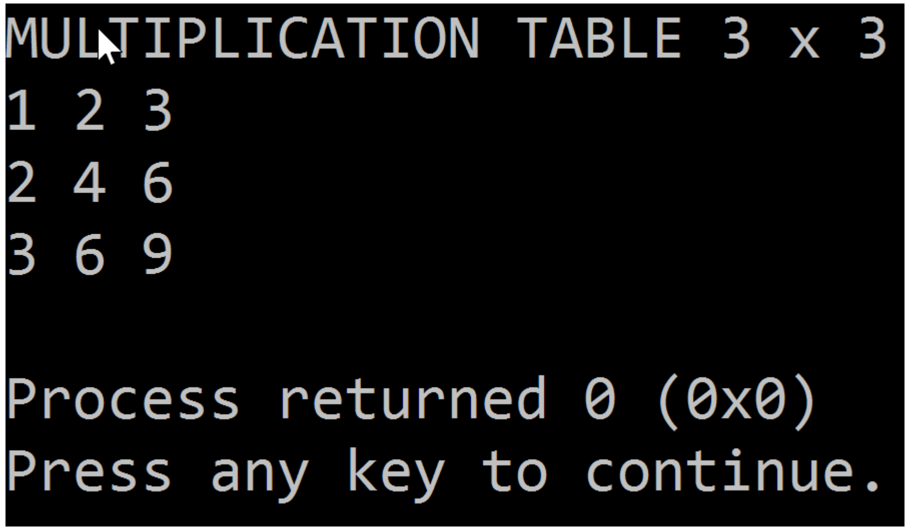
```
for (int row=0; row < 5; row++)        // OUTTER LOOP
{
    for (int col=0; col<5; col++) // INNER LOOP
    {
        cout << " * " ;
    }
    cout << endl;
}
```



```
 *  *  *  *  *
 *  *  *  *  *
 *  *  *  *  *
 *  *  *  *  *
 *  *  *  *  *


Process returned 0 (0x0)
Press any key to continue.
```

Nested Loops

```
cout << "MULTIPLICATION TABLE 3 x 3" << endl;

for (int row=0; row < 3; row++)       // OUTTER LOOP
{
    for (int col=0; col<3; col++) // INNER LOOP
    {
        cout << (row+1) * (col + 1) << " ";
    }
    cout << endl;
}
```



```
MULTIPLICATION TABLE 3 x 3
1 2 3
2 4 6
3 6 9

Process returned 0 (0x0)
Press any key to continue.
```