Programming in C++

Repetition and Loops in C++

Dr. Zeyad Safaa Younus Saffawi

Repetition and Loops

- **Repetition** (also called ***loops***) is used in programming when we want to **execute a block of code multiple times** until a specific condition is met.
- Instead of writing the same statements many times, we use loops to make the program **shorter, faster, and easier to read**.

Type of loop:

- **For loop:**
- **While loop**
- **Do while loop**

for loop Statement

The for loop is used when the number of repetitions is known in advance.

Syntax:

```
for (initialization; condition; increment | decrement)
    {
        // statements
    }
```

It has three parts:

- **Initialization** – set the starting value.
- **Condition** – test if the loop continues.
- **Increment | Decrement** – change the loop variable.

for loop Statement

- **Example: print numbers from 1 -10**

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    for(i = 1; i <= 10; i++)
                cout << "The numbers: " << i << endl;
return 0;
}
```

for loop Statement

Example 5: Find the factorial of a number (fact = n*(n-1)*(n-2)*…*1)

```
#include <iostream>
using namespace std;
int main()
  {
   int n, fact = 1;
   cout << "Enter a number: ";
   cin >> n;
   for (int i = n; i >= 1; i--)
       {
         fact *= i;
        }
   cout << "Factorial of " << n << " = " << fact;
   return 0;
  }
```

Nested for loop

- **Example: multiplication table**

```
#include <iostream>
using namespace std;
int main()
    {
    for (int row = 1; row <= 3; row++)
        {
            for (int col = 1; col <= 4; col++)
                {
                  cout << row * col << "\t";
                }
              cout << endl;
        }
    return 0;
}
```

While statement

- The **while loop** is used when the number of repetitions is **unknown**, and it continues **as long as the condition is true**.

Syntax:

```
while (condition)
{
statement(s);
}
```

While statement

Example: print the numbers from 1-5

```
#include <iostream>
using namespace std;
int main()
   {
    int i = 1;
    while (i <= 5)
        {
         cout << i << " ";
         i++;
         }
     return 0;
}
```

nested while loop statement

- **Syntax**

```
while(condition)
      {
      while(condition)
            {
             statement(s);
             }
      statement(s); // you can put more statements.
       }
```

nested while loop statement

Example: Multiplication Table

```
#include <iostream>
using namespace std;
int main()
    {
    int i = 1; // Counter for the Outer Loop (The number being multiplied)
     int j;    // Counter for the Inner Loop (The multiplier/factor)
    while (i <= 10)
        {
        j = 1; // Initialize the inner counter before starting each outer loop iteration
        cout << "Multiplication Table for " << i << ":" << endl;
        while (j <= 10)
            {
            cout << i << " * " << j << " = " << (i * j) << endl;
             j++;
             }
         cout << "----------------------" << endl; // Separator between tables
      i++;
   }
   return 0;
}
```

do...while statement

- This loop is similar to while, but it **executes the block at least once** even if the condition is false.
- Unlike **for** and **while** loops, which **test the loop condition at the top of the loop**, the **do...while** loop **checks its condition at the bottom of the loop**.

Syntax:

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

do...while statement

- Example: Write a C++ program that uses a do-while loop to print the numbers from 1 to 5.

```
#include <iostream>
using namespace std;
int main()
   {
   int i = 1;
   do
    {
      cout << i << " ";
      i++;
    }
   while (i <= 5);
   return 0;
   }
```

nested do...while loop statement

```
do
   {
       statement(s); // you can put more statements.
       do
         {
          statement(s);
          }
           while( condition );
    }
     while( condition );
```

nested do...while loop statement

Example: Write a C++ program that uses nested do-while loops to print numbers from 1 to 5 in three rows.

```
#include <iostream>
using namespace std;
int main() {
   int row = 1;
   do {
         int col = 1;
         do {
               cout << col << " ";
               col++;
               } while (col <= 5);
         cout << endl;
         row++;
       } while (row <= 3);
   return 0;
}
```

Infinite Loop

- An **infinite loop** is a loop that **never stops executing** because the **condition never becomes false**.
- It keeps repeating endlessly — unless we **manually stop it** using a control statement like **break** or by closing the program.
- **Example:** Basic infinite loop using **for**

```
#include <iostream>
using namespace std;
int main()
    {
     for( ; ; )
        {                    // no condition or increment → always true
           cout << "Running forever..." << endl
        }
     return 0;
}
```

- ***Notes***:
  - There is **no condition** inside the parentheses of for(; ;), so, the loop runs forever.
  - The only way to stop it is by pressing **Ctrl + C** or adding a **break** statement.

Loop control statements

- **Loop control statements** change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

1. **break** Statement terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
2. **continue** statement causes the loop to skip the remainder of its body and immediately retest its condition before reiterating.
3. **goto** statement transfers control to the labeled statement. However, it is not advised to use the goto statement in your program.

Break Statement

- When the **break** statement is encountered inside a loop, the loop is terminated immediately, and program control resumes at the next statement following the loop.
- It can be used to terminate a case in the **switch** statement.

- Syntax:

```
break;
```

Break Statement

Example:

```
#include <iostream>
using namespace std;
int main()
    {
   int counter = 1;
   for( ; ; )
       {
         cout << "Count = " << counter << endl;
         counter++;
         if(counter > 5)  // stop when counter exceeds 5
          break;
        }
   cout << "Loop stopped!";
   return 0;
}
```

continue Statement

- The **continue** statement works somewhat like the break statement. Instead of forcing termination, however, continue forces the next iteration of the loop to take place, skipping any code in between.
- For the **for** loop, continue causes the conditional test and increment portions of the loop to execute. For the **while** and **do...while** loops, program control passes to the conditional tests.

Syntax:

continue;

continue statement

Example:

```
#include <iostream>
using namespace std;
int main ()
{
   int a = 10;
   do
     {
      if( a == 15)
          {
           // skip the iteration.
           a = a + 1;
           continue;
          }
       cout << "value of a: " << a << endl;
       a = a + 1;
        }
         while( a < 20 );
  return 0;
  }
```

References

- Gaddis, T. (2014). *Starting out with C++: From control structures through objects* (8th ed.). Pearson.
- Soulié, J. (2007, April 24). C++ language tutorial. cplusplus.com.
- Tutorials Point. (n.d.). *Learn C++ programming language*. Tutorials Point.