Cairo University
Faculty of Computers and Information
Computer Science Department

Programming-1 CS112
2017/2018

Loops

Dr. Amin Allam

1 The while statement

In some situations, we need to repeat some statements several times. Each time the statements are executed is called a *loop*. One of the ways to construct loops is the *while* statement.
A `while` statement in the form *while(expression) statement;* will execute `statement` several times as long as `expression` is `true` (or any value other than 0). `statement` can be replaced by a block of statements: {`statement1; statement2; ...statement`n`;`}.
The following example illustrates a simple program that computes the sum of n integers, where the value of n is entered by the user:

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6     int n;
7     cout << "Enter the number of integers: ";
8     cin >> n;
9
10    int sum = 0;
11
12    int i = 0;
13    while(i < n)
14    {
15       int v;
16       cout << i << ") Enter integer:  ";
17       cin >> v;
18       sum += v;
19
20       i++;
21    }
22
23    cout << "The sum = " << sum << endl;
24    return 0;
25 }
```

2 The for statement

Another way to construct loops is the *for* statement.

A `while` statement in the form *for(initstmt; expression; incstmt) statement;* will execute `statement` several times as long as `expression` is `true` (or any value other than 0). `statement` can be replaced by a block of statements: {`statement1; statement2; ...statement`n`;`}. `initstmt` will be called exactly once before the `for` statement begins. `incstmt` will be called at the end of each loop.

The following example illustrates the program that computes the sum of n integers, where the value of n is entered by the user, using `for` statement instead of `while` statement:

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter the number of integers: ";
    cin >> n;

    int sum = 0;

    for(int i = 0; i < n; i++)
    {
        int v;
        cout << i << ") Enter integer:  ";
        cin >> v;
        sum += v;
    }

    cout << "The sum = " << sum << endl;
    return 0;
}
```

Note that the variable i is visible only inside the `for` header and block. A `for` statement can replace a `while` statement and vice versa, they are just syntactic alternatives:

```
for(initstmt; expression; incstmt)
{
    statements;
}
```

```
initstmt;
while(expression)
{
    statements;
    incstmt;
}
```

3 The do-while statement

Another way to construct loops is the *do-while* statement. The `do-while` statement is similar to the `while` statement except that `expression` is checked at the end of each loop instead of the beginning. This is useful only when we know for sure that the first loop is going to be executed:

```
initstmt;
do
{
   statements;
   incstmt;
}
while(expression);
```

4 The break statement

In the `for` loop, it is possible to omit any of `initstmt`, `expression` and `incstmt`. In the `while` loop, it is not possible to omit `expression`. For example, the following are three variants of implementing an infinite loop that will never stop execution:

```
for(;;)
{
   statements;
}
```

```
while(true)
{
   statements;
}
```

```
do
{
   statements;
}
while(true);
```

A *break* statement will exit immediately any loop statement. Note that it can exit from `switch` statement, but it can not exit from `if` statement. The following program computes the sum of integers entered by user until he enters zero, which stops the loops and reports the sum:

```
int main()
{
   int sum = 0;
   while(true)
   {
      int v;
      cout << "Enter an integer:  ";
      cin >> v;
      if(v == 0) break;
      sum += v;
   }
   cout << "The sum = " << sum << endl;
   return 0;
}
```

5 The continue statement

A *continue* statement will not exit from the loop statement. It will stop execution of the current loop and continue to the execution of the following loop.

The following program computes the sum of positive integers entered by user until he enters zero. If the user enters a negative number, the program ignores it and asks the user to enter the following number:

```
int main()
{
   int sum = 0;
   while(true)
   {
      int v;
      cout << "Enter an integer:  ";
      cin >> v;
      if(v == 0) break;      // end all loops
      if(v < 0) continue;    // end current loop only (do not execute sum+=v)
      sum += v;
   }
   cout << "The sum = " << sum << endl;
   return 0;
}
```

In case of `for` statement, `continue` will execute `incstmt` before executing the following loops. The following example outputs all values from 0 to 9 except the value 7:

```
int main()
{
   for(int i=0; i<10; i++)
   {
      if(i == 7) continue; // execute i++, then check i<10, then this line again
      cout << i << " ";
   }
   return 0;    // Outputs: 0 1 2 3 4 5 6 8 9
}
```

It is possible that the loop *body* consists of one statement or no statement:

```
int main()
{
   for(int i=0; i<10; i++) cout<<i<<" "; // outputs all numbers from 0 to 9
   int k; for(k=0; k<10; k++); cout<<k;  // outputs 10
   return 0;
}
```