Programming Language (630203)
Fall 2010/2011 – Lecture Notes # 11

Repetition Structure (do…while)

Objectives of the Lecture

- **do…while Looping (Repetition) Structure.**
- **Choosing the Right Looping Structure.**
- **break and continue Statements.**

do…while Looping (Repetition) Structure

- The general form of a **do...while**:

```
do
    statement
while (expression);
```

- The statement can be either a simple or compound statement.
- The statement executes first, and then the expression is evaluated.
- To avoid an infinite loop, body must contain a statement that makes the expression false.
- Loop always iterates at least once.

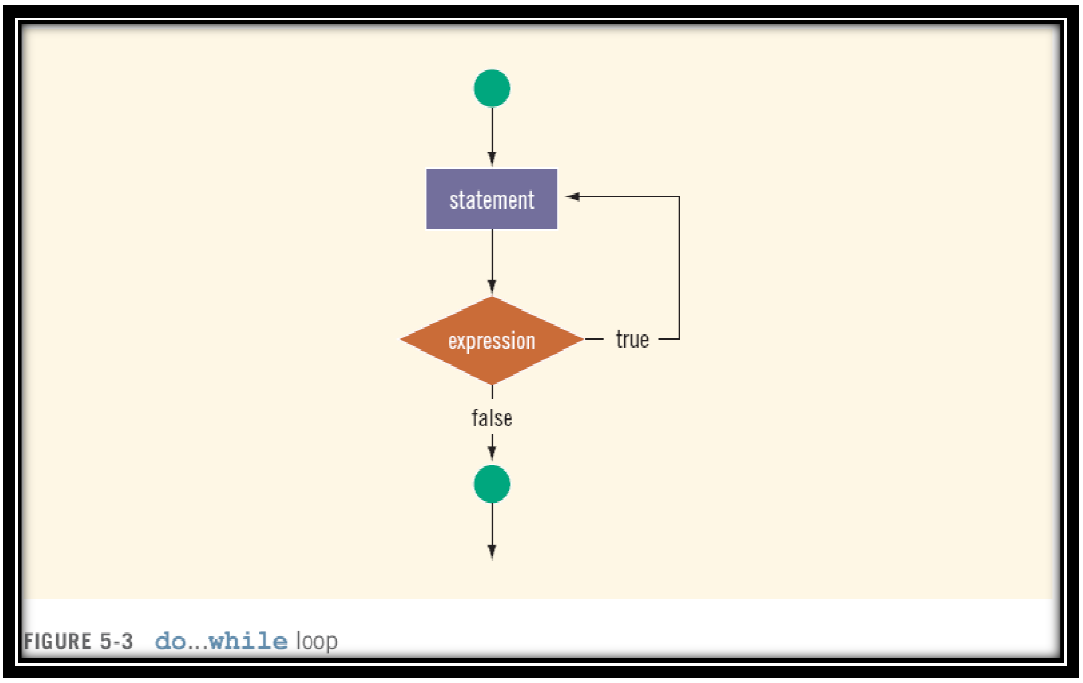

FIGURE 5-3 do...while loop

EXAMPLE 5-18

```
i = 0;

do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 20);
```

The output of this code is:

```
0 5 10 15 20
```

After 20 is output, the statement:

```
i = i + 5;
```

changes the value of i to 25 and so `i <= 20` becomes `false`, which halts the loop.

EXAMPLE 5-19

Consider the following two loops:

a.
```
i = 11;
while (i <= 10)
{
    cout << i << " ";
    i = i + 5;
}
cout << endl;
```

b.
```
i = 11;
do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 10);

cout << endl;
```

In (a), the `while` loop produces nothing. In (b), the `do...while` loop outputs the number 11 and also changes the value of i to 16.

```
sum = 0;

do
{
    sum = sum + num % 10; //extract the last digit
                          //and add it to sum
    num = num / 10;       //remove the last digit
}
while (num > 0);

cout << "The sum of the digits = " << sum << endl;

if (sum % 3 == 0)
    cout << temp << " is divisible by 3" << endl;
else
    cout << temp << " is not divisible by 3" << endl;

if (sum % 9 == 0)
    cout << temp << " is divisible by 9" << endl;
else
    cout << temp << " is not divisible by 9" << endl;
```

Choosing the Right Looping Structure

- All three loops have their place in C++
  - If you know or can determine in advance the number of repetitions needed, **the for loop is the correct choice**
  - If you do not know and cannot determine in advance the number of repetitions needed, and it could be zero, **use a while loop**
  - If you do not know and cannot determine in advance the number of repetitions needed, and it is at least one, **use a do...while loop**

break and continue Statements

- **break** and **continue** alter the flow of control (terminates the loop immediately)
- **break** statement is used for two purposes:
  - To exit early from a loop
  - To skip the remainder of the switch structure
- After the break statement executes, the program continues with the first statement after the structure.
- **continue** is used in while, for, and do…while structures
- When **continue** executed in a loop
  - It skips remaining statements and proceeds with the next iteration of the loop

// break loop example

```
#include <iostream>
using namespace std;
int main ()
{
  int n;
  for (n=10; n>0; n--)
  {
    cout << n << ", ";
    if (n==3)
    {
      cout << "countdown aborted!";
      break;
    }
  }
  return 0;    }
```

// continue loop example

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

int main ()
{
  for (int n=10; n>0; n--) {
    if (n==5) continue;
    cout << n << ", ";
  }
  cout << "FIRE!\n";
  return 0;
}
```