BTE2313

Chapter 6: CONTROL STRUCTURES Repetition/Looping

by

Sulastri Abdul Manap
Faculty of Engineering Technology
sulastri@ump.edu.my

Repetition/Looping

- There are 3 commands in C++ that are used to create loops:

→ `for` : implements a set of instructions for a definite number of times

→ `while` : re-iterates a set of instructions from 0 to several times, providing that the given condition is true

→ `do..while` : implements a set of instructions at lease ONCE, and then from 0 to several times, providing that the given condition is true.

`for` Statement

- This statements is used when the number of repetition is known beforehand by the programmer.
- As an example, the following codes are used to print a message "Hello world", a hundred times. Variable count used as the loop control variable (LCV).

```
for (counter=1; counter<=100; counter++)
  cout <<"Hello world."<<endl;
```

`for` Statement (cont.)

- The system will:

a) Set the value of variable `counter` to 1

b) Check the value of `counter`

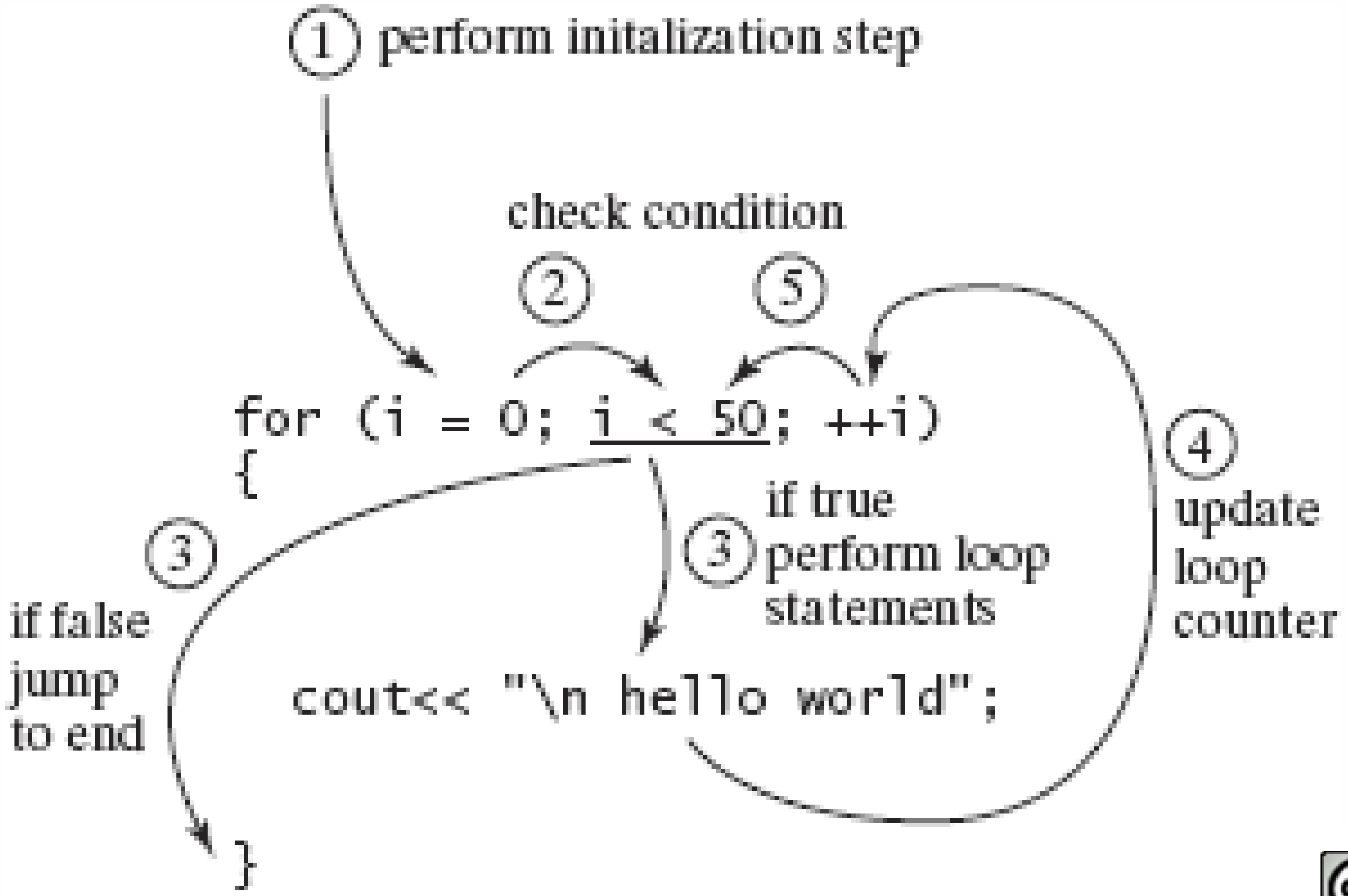
c) Exit from the repetition structure if the value of variable `counter` is bigger than or equal to 100

d) Execute the output statement if the value of variable `counter` is less than or equal to 100;

e) Increase the value of variable `count` by 1

f) Go to step b

`for` Statement (cont.)



for Statement (cont.)

- Syntax of the for statement:

```
for (<A>; <B>; <C>)
{
  // Statement to be executed in a loop
}
```

Note:

<A> is a statement to initialize the value of the LCV;

<B> is a logical expression that causes the loop to be terminated and contains the LCV

<C> is a statement to increase/decrease the value of the LCV

`for` Statement (cont.)

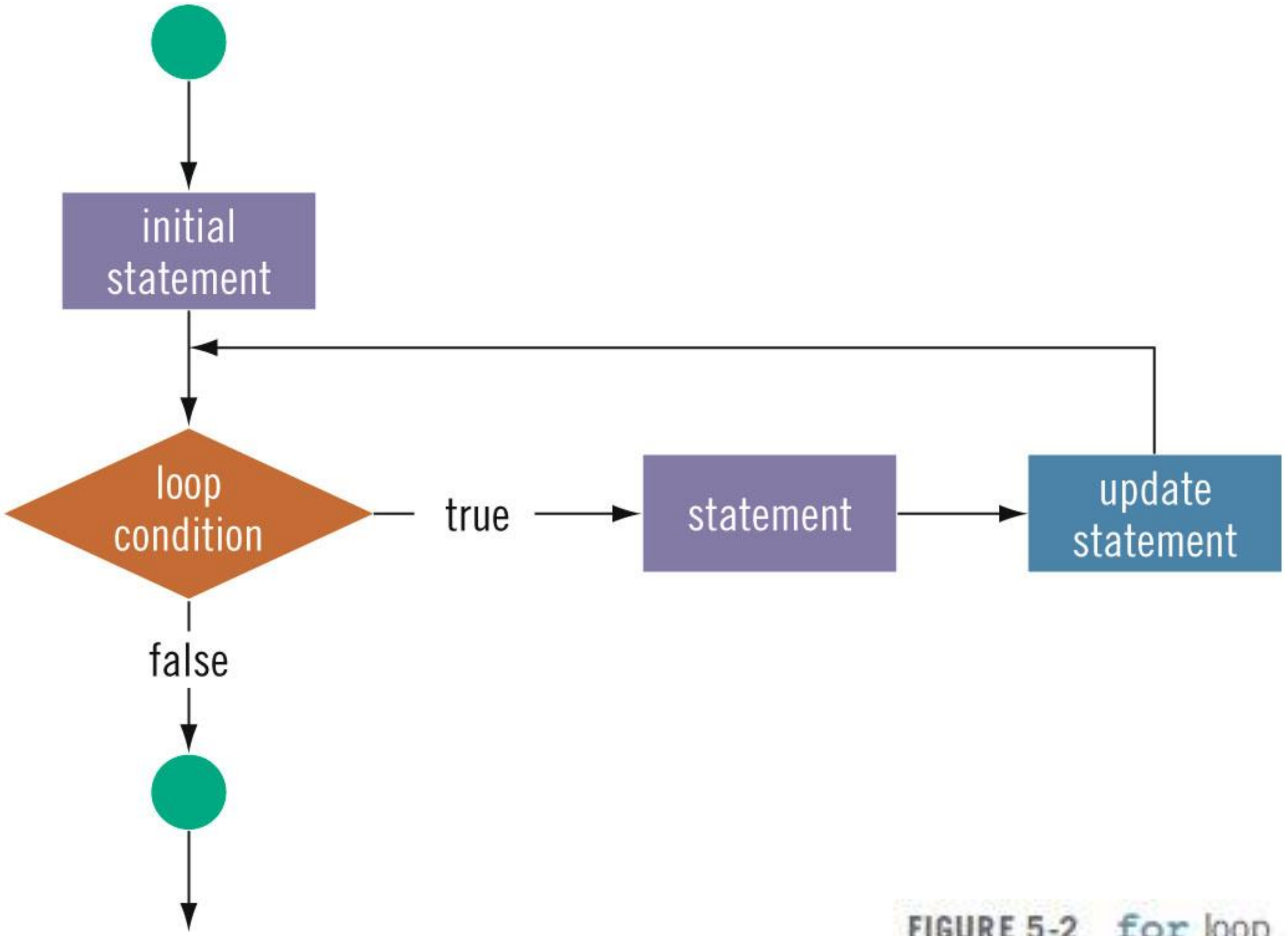


FIGURE 5-2 `for` loop

for Statement (cont.)

- Example 1: To display the sum of number from 50 to 55.

```
sum = 0;
for (num=50; num <=55; num++)
{
       sum +=num;
}
cout <<“Sum of numbers from 50 to 55 is” <<sum;
```

for Statement (cont.)

- Example 2: To display the sum and average of thirty marks, which are keyed in by the user

Try with counter keyed in by user.

```
sum=0;
cout<<“Enter 10 marks”;
for (count=1; count<=10; count++)
{
   cin>>num;
   sum+=num;
}
cout<< “The summation of marks is” << sum <<endl;
cout<< “The calculated average is” << sum/10.0;
```

while Statement

- Used when it is not clear how many times a statement or a block of statements will be executed.
- A while loop can be:
  - → Counter-controlled
  - → Sentinel-controlled
- Sentinel value (a value to stop the execution of loop) is keyed in to end the loops. Example:-1

while Statement (cont.)

Syntax of the while statement:

```
<A>
while <B>
{
    //statement to be repeated <C>
}
```

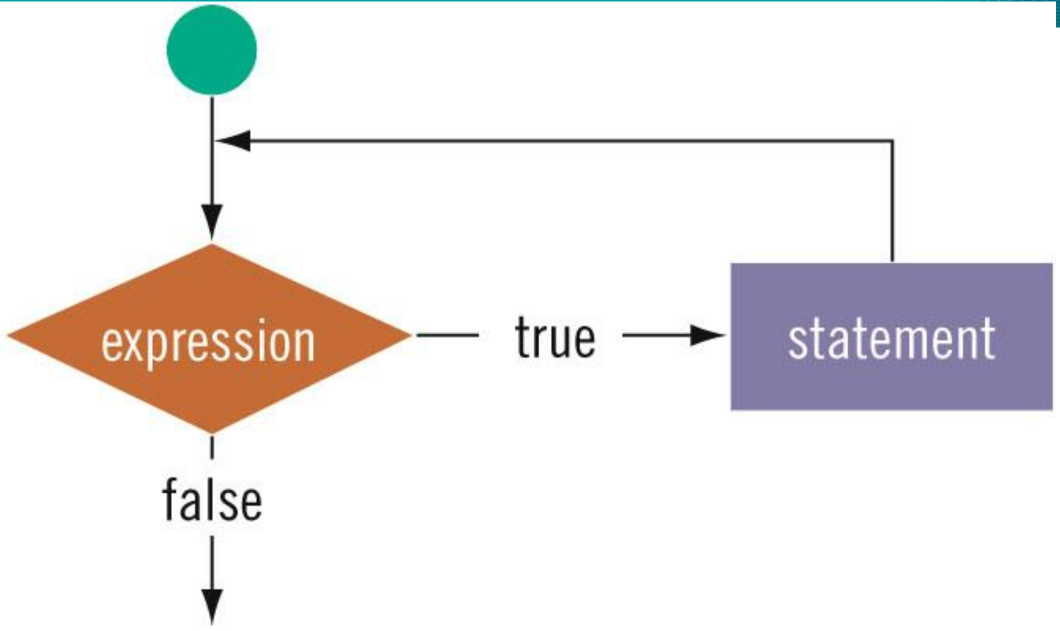
Note:

<A> is a statement to initialize the value of the LCV;

<B> is a logical expression that causes the loop to be terminated and contains the LCV

<C> is a statement to increase/decrease the value of the LCV

while Statement (cont.)



Counter-Controlled `while` Statement

- When number of times of the statements that need to be repeated is known definitely
- Structure:

```
counter = 0;          //initialize the loop control variable

while (counter < N) //test the loop control variable
{
    .
    .
    .
    counter++;        //update the loop control variable
    .
    .
    .
}
```

Counter-Controlled `while` Statement

- Example

```
float marks, sum=0;
counter=1;
cout << “Enter 5 marks”;
while (counter<=5)
{
   cin >> marks;
   sum += marks;
   counter++;
}
cout << “summation of marks:” << sum <<endl;
cout << “the average is” << sum/10;
```

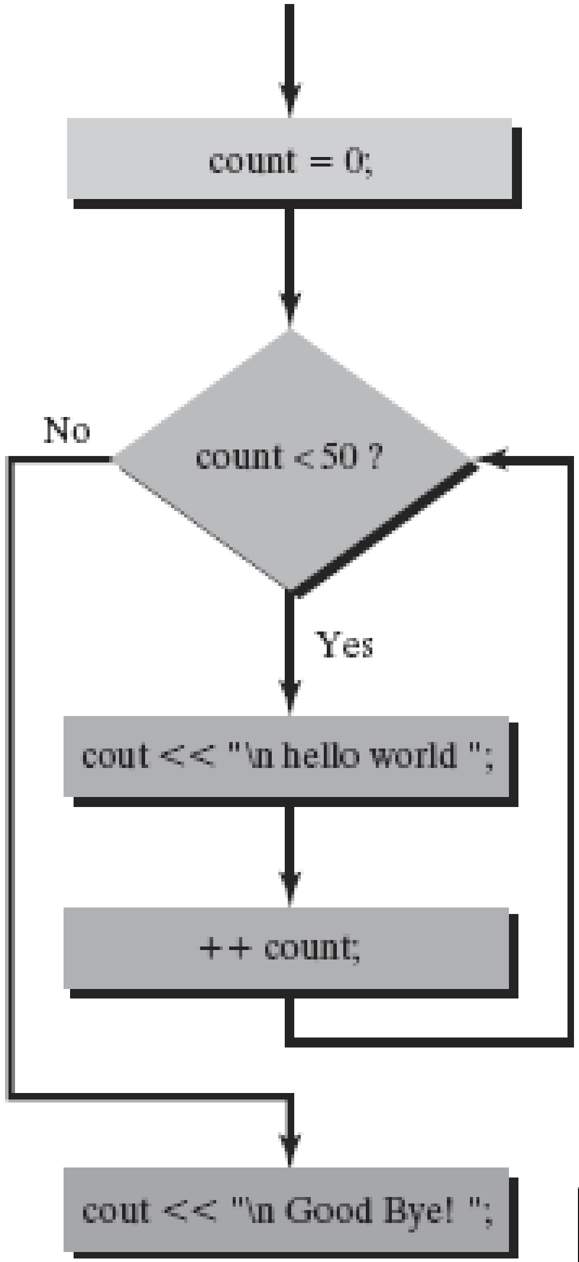
Counter-Controlled `while` Statement

```
// write hello world 50 times

count = 0;   // initialize count to zero

while (count < 50 )
{
    cout << "\n hello world";
    ++ count;   // loop altering statement
}

cout << "\n Good Bye!  ";
```



Sentinel-Controlled `while` Statement

- Sentinel variable is tested in the condition, and the repetition ends when the sentinel value is encountered
- Structure:

```
cin >> variable;                //initialize the loop control variable

while (variable != sentinel)  //test the loop control variable
{
    .
    .
    .
    cin >> variable;          //update the loop control variable
    .
    .
    .
}
```

Sentinel-Controlled `while` Statement

- Example

```
float sum=0;
float num;
int count=0;
cout << “Enter numbers to be summed (type -1 to end).”;
cin >> num;
while (num != -1)
{
    sum += num;
    count++;
    cin >> num;
}
cout << “The sum of marks is” << sum <<endl;
cout << “The average is” << sum/count;
```

do…while Statement

- The loop-continuation at the end of the loop, so the body of the loop will be executed at least once. Similar with `while` structure.

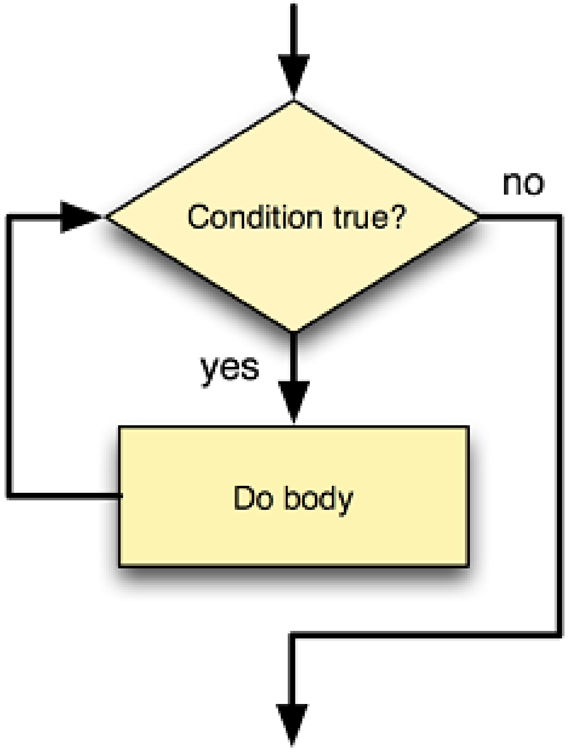
```
do
{
        // Statement to be repeated <C>
} while (<B>);
```

Note:

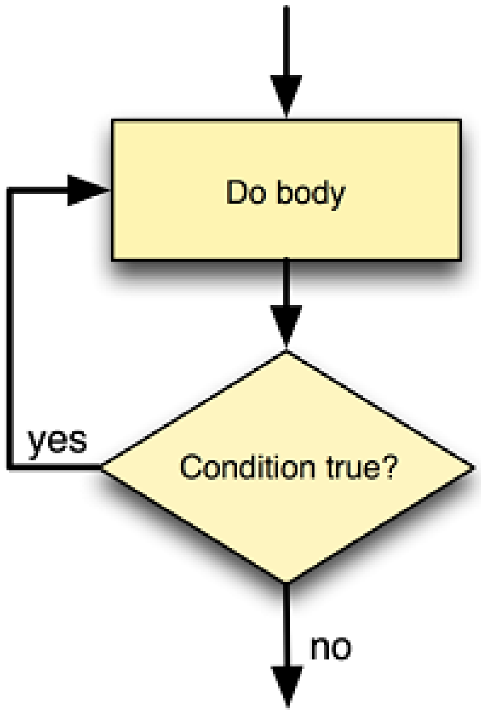
<B> is a logical expression that causes the loop to be terminated and contains the LCV

<C> is a statement to change the value LCV

`do…while` Statement (cont.)



`while` flowchart



`do/while` flowchart

do…while Statement (cont.)

- Example:

```
#include <iostream>
using namespace std;
int main()
{
 float input;
 cout << "Please enter a number between 0-20: ";
 // values must be in the range 0...20
 do
{
   cin >> input;
} while (input < 0 || input > 20);
 // input at this point is assured to be within range
 cout << "Valid number keyed in was " << in_value << endl;
}
```

How to choose the right looping?

- If number of iterations is known beforehand, use `for` loop
- If number of iterations is not known beforehand, and it could be zero, use a `while` loop
- If number of iterations is not known beforehand, and it is certain the command must be done at least once, use a `do...while` loop

break and continue Statement

- break and continue statements are used to alter the flow of control structure
- break is used for two conditions:

  → To quit prematurely from a loop, where it is able to exclude certain variables.

  →To skip the remains of a switch structure
- continue is utilized in while, for, and do…while structures

  → skips remaining statements and proceeds with the next iteration of the loop

`break` Example

```
#include <iostream>
int main()
{
   int y;
   for (y=1; y<=10; y++)
   {
   if (y==6)
      break; //stop loop if y==6
   cout << y << “”;
   }
   cout << endl << “out of loop at y==” << y <<endl;
   return 0;
}
```

How the output looks like?

continue Example

```
#include <iostream>
int main()
{
   for (int y=1; y<=10; y++)
   {
     if (y==5 || y==8)
    continue; //skip remaining code in loop if y==5
     cout << y << " ";
   }
   cout << endl << "continue to skip displaying the value 5" << endl;
return 0;
}
```

How the output looks like?

Nested Loop

- A loop can be put/nested within another loop.
- Example: To create the following pattern;

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

- Codes:

```
for (i = 0; i <= 4 ; i++)
{
        for (j = 0; j <= i; j++)
            cout << "*";
        cout << endl;
}
```

Nested Loop (cont.)

- Determine the result if the first for statement is replaced with this?

```
for (i = 4; i >= 0; i--)
```