

# Chapter 6

## Looping

Dale/Weems/Headington

1

## Chapter 6 Topics

- **While Statement Syntax**
- **Count-Controlled Loops**
- **Event-Controlled Loops**
- **Using the End-of-File Condition to Control Input Data**
- **Using a While Statement for Summing and Counting**
- **Nested While Loops**
- **Loop Testing and Debugging**

2

## **What is a loop?**

- **A loop is a repetition control structure.**
- **it causes a single statement or block to be executed repeatedly**

3

## **Two Types of Loops**

**count controlled loops**

**repeat a specified number of times**

**event-controlled loops**

**some condition within the loop body changes and this causes the repeating to stop**

4

## while Statement

### SYNTAX

```
while ( Expression )  
{  
    .  
    .           // loop body  
    .  
}
```

**NOTE:** Loop body can be a single statement, a null statement, or a block.

6

## Count-controlled Loop

```
int count ;  
  
count = 4;           // initialize loop variable  
  
while (count > 0)   // test expression  
{  
    cout << count << endl ; // repeated action  
  
    count -- ;      // update loop variable  
}  
cout << "Done" << endl ;
```

10

## Count-controlled Loop

```
int count ;
```

```
count = 4;
```

```
while (count > 0)
```

```
{
```

```
    cout << count << endl ;
```

```
    count -- ;
```

```
}
```

```
cout << "Done" << endl ;
```

count

OUTPUT

11

## Count-Controlled Loop Example

**myInfile contains 100 blood pressures**

**Use a while loop to read the 100 blood pressures and find their total**

12

```
ifstream  myInfile ;
int       thisBP ;
int       total ;
int       count ;

count = 0 ;                               // initialize

while ( count < 100 )                     // test expression
{
    myInfile >> thisBP ;
    total = total + thisBP ;
    count++ ;                               // update
}

cout << "The total = " << total << endl ;
```

13

## Event-controlled Loops

- **Sentinel controlled**  
keep processing data until a special value which is not a possible data value is entered to indicate that processing should stop
- **End-of-file controlled**  
keep processing data as long as there is more data in the file
- **Flag controlled**  
keep processing data until the value of a flag changes in the loop body

14

## Examples of Kinds of Loops

<b>Count controlled loop</b>	<b>Read exactly 100 blood pressures from a file.</b>
<b>End-of-file controlled loop</b>	<b>Read all the blood pressures from a file no matter how many are there.</b>

15

## Examples of Kinds of Loops

<b>Sentinel controlled loop</b>	<b>Read blood pressures until a special value (like -1) selected by you is read.</b>
<b>Flag controlled loop</b>	<b>Read blood pressures until a dangerously high BP (200 or more) is read.</b>

16

## A Count-Controlled Loop

### SYNTAX

```
for ( initialization ; test expression ; update )  
{  
    0 or more statements to repeat  
}
```

27

### The for loop contains

an initialization

an expression to test for  
continuing

an update to execute after each  
iteration of the body

28

## Example of Repetition

```
int num;

for ( num = 1 ; num <= 3 ; num++ )
{
    cout << num << "Potato" << endl;
}
```

29

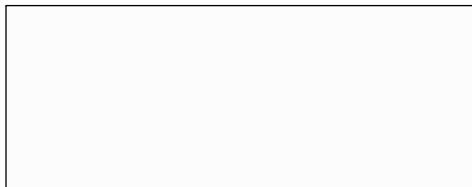
num

?

## Example of Repetition

```
int num;

for ( num = 1 ; num <= 3 ; num++ ) {
    cout << num << "Potato" << endl;
}
```

**OUTPUT**

30

## Do-While Statement

Is a looping control structure in which the loop condition is tested after each iteration of the loop.

### SYNTAX

```
do
{
    Statement
} while ( Expression );
```

Loop body statement can be a single statement or a block.

34

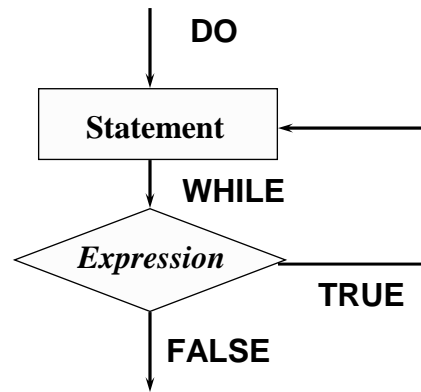
## Do-While Loop vs. While Loop

- POST-TEST loop (exit-condition)
- The looping condition is tested after executing the loop body.
- Loop body is always executed at least once.

- PRE-TEST loop (entry-condition)
- The looping condition is tested before executing the loop body.
- Loop body may not be executed at all.

35

## Do-While Loop



When the expression is tested and found to be false, the loop is exited and control passes to the statement that follows the do-while statement.

36