

Chapter 21 – While Loops

The *while* statement is a loop control structure that is much simpler than the *for* statement. It does not have a built-in counter. The while loop has the following form:

```
while (Boolean-expression)
    statement;
```

where *Boolean-expression* is a Boolean expression that determines when the loop stops, the loop will continue until the Boolean expression becomes false

statement is a single statement or a compound statement that is performed until the loop halts

In order to write the unknown-number-of-grades program example (see Chapter 20) using a *while-loop*, both counters are still required. Both loops will have to be created and managed by the programmer independently of the looping structure. In addition, there need be no *if* statement or use of a *break*. Instead, there will be two *cin* statements in the program. One will be placed before the *while* statement so that the Boolean expression will have a value to test the first time around. (This is often referred to as a priming read.) The other will be placed at the end of the compound statement after the grade value has been added to the counter variable *sum*. Placing the second *cin* statement at the end of the body of the loop means that the loop will stop before the terminal value can be added to the *sum* counter.

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

void main( ) {
    int grade, number=0, sum=0;
    cout << "Enter the numeric grade: ";
    cin >> grade;
    while (grade>=0) {
        number++;
        sum += grade;
        cout << "Enter the numeric grade: ";
        cin >> grade;
    }
    cout << endl << "Average = " << sum / number << endl;
}
```

If the user enters the grades 80, 66, 89, 59, 77, 85, 85, and the terminal value -5, the run will appear as follows:

```
Enter the numeric grade (<0 quits): 80
Enter the numeric grade (<0 quits): 66
Enter the numeric grade (<0 quits): 89
Enter the numeric grade (<0 quits): 59
Enter the numeric grade (<0 quits): 77
Enter the numeric grade (<0 quits): 85
Enter the numeric grade (<0 quits): 85
Enter the numeric grade (<0 quits): -5
```

Average = 77

Exercises

1. What is the output of the following loop?

```
n = 0;
while (n < 5) {
    n++;
    cout << n << endl;
}
```

2. What is the output of the following loop?

```
n = 0;
while (n < 5) {
    cout << n << endl;
    n++;
}
```

3. How many times will the following loop print out “Hi!”?

```
A = 33;
STOP = false;
While (!STOP) {
    A++;
    cout << “Hi!” << endl;
    if (A > 36)
        STOP = true;
}
```

4. Give a value that will cause the following loop to halt.

```
s = 0;
cout << "Enter a number: ";
cin >> n;
while (n > 0) {
    s += n;
    cout << "Enter a number: ";
    cin >> n;
}
cout << "The sum of the inputs is: " << s << endl;
```

5. If the given input is entered into the program of problem number 4, what will be the output of the cout statement after the loop?

7, 8, 3, -2, 4, 5, 0

Programming Assignment 21.1

Write a program that uses a while loop to output the odd integers from -99 to 99.

Programming Assignment 21.2

Write a program that accepts the heights in inches of everyone in the class, then outputs the average height of the people in the class.

Programming Assignment 21.3

Write a program that will allow the user to guess a computer-generated number between 0 and 1000. Let the program loop until the user guesses the correct number. Each time the user makes a guess that is incorrect, the program should respond with a hint telling the user if the guess was too high or too low.

Programming Assignment 21.4

Change the interface member function(s) of the FNPB of programming assignment 20.14 so that the interface continues to function until the user decides to quit the program.

Programming Assignment 21.5

Make the following changes to *class track*.

- Add the private data member *length* to store the length of the track.
- In the default constructor, set *length* to 40.
- Add a second constructor that accepts the length of the track from the programmer.
- Add a public function *raceOver()* to determine if the race is over (i.e. one of the cars has crossed the finish line). *raceOver()* should return 0 if the race is not over and 1 if the race is over.

After making the above changes, write a program that use class *track* to stage a race. In the program, use a *while-loop* to run the race until there is a winner. The program should, of course, announce the winner.