While Loop*

Kenneth Leroy Busbee

Abstract

An introduction to the while control structure with examples in the C++ programming language.

1 Introduction to Test Before Loops

There are two commonly used test before loops in the iteration (or repetition) category of control structures. They are: while and for. This module covers the: while.

1.1 Understanding Iteration in General – while

The concept of iteration is connected to possibly wanting to repeat an action. Like all control structures we ask a question to control the execution of the loop. The term loop comes from the circular looping motion that occurs when using flowcharting. The basic form of the while loop is as follows:

```
initialization of the flag
while the answer to the question is true then do
  some statements or action
  some statements or action
  some statements or action
  update the flag
```

In almost all languages the question (called a **test expression**) is a **Boolean expression**. The Boolean data type has two values – true and false. Let's rewrite the structure to consider this:

```
initialization of the flag
while the expression is true then do
  some statements or action
  some statements or action
  some statements or action
  update the flag
```

*Version 1.8: Mar 26, 2010 9:19 am +0000
†http://creativecommons.org/licenses/by/3.0/

Within the while control structure there are four attributes to a properly working loop. They are:

- Initializing the flag
- Test expression
- Action or actions
- Update of the flag

The initialization of the flag is not technically part of the control structure, but a necessary item to occur before the loop is started. The English phrasing is, "While the expression is true, do the following actions". This is looping on the true. When the test expression is false, you stop the loop and go on with the next item in the program. Notice, because this is a test before loop the action **might not happen**. It is called a **test before loop** because the test comes before the action. It is also sometimes called a pre-test loop, meaning the test is pre (or Latin for before) the action and update.

1.2 Human Example of the while Loop

Consider the following one-way conversation from a mother to her child.

Child: The child says nothing, but mother knows the child had Cheerios for breakfast and history tells us that the child most likely spilled some Cheerios on the floor.

Mother says: "While it is true that you see (As long as you can see) a Cheerio on floor, pick it up and put it in the garbage."

Note: All of the elements are present to determine the action (or flow) that the child will be doing (in this case repeating). Because the question (can you see a Cheerios) has only two possible answers (true or false) the action will continue while there are Cheerios on the floor. Either the child 1) never picks up a Cheerio because they never spilled any or 2) picks up a Cheerio and keeps picking up Cheerios one at a time while he can see a Cheerio on the floor (that is until they are all picked up).

2 The while Structure within C++

2.1 Syntax

The syntax for the while control structure within the C++ programming language is:

```
statement;        // This statement initializes the flag;
while (expression)
  {
  statement;
  statement;
  statement;
  statement;    // This statement updates the flag;
  }
```

NOTE: The test expression is within the parentheses, but this is not a function call. The parentheses are part of the control structure. Additionally, there is not a semicolon after the parenthesis following the expression.

2.2 An Example

Example 1: C++ source code: while

```
loop_response = 'y';
while (loop_response == 'y')
  {
  cout ≪ "\nWhat is your age? ";
  cin ≫ age_user;
  cout ≪ "\nWhat is your friend's age? ";
  cin ≫ age_friend;
  cout ≫ "\nTogether your ages add up to: ";
  cout ≫ (age_user + age_friend);
  cout ≪ "\nDo you want to do it again? y or n ";
  cin ≫ loop_response;
  }
```

The four attributes of a test before loop are present. The initialization of the flag. The test is the equality relational comparison of the value in the flag variable to the lower case character of y. The action part consists of the 6 lines that prompt for data and then displays the total of the two ages. The update of the flag is the displaying the question and getting the answer for the variable loop_response.

This type of loop control is called an event controlled loop. The flag updating is an event where someone decides if they want the loop to execute again.

Using indentation with the alignment of the loop actions and flag update is normal industry practice within the C++ community.

2.3 Infinite Loops

At this point it's worth mentioning that good programming always provides for a method to insure that the loop question will eventually be false so that the loop will stop executing and the program continues with the next line of code. However, if this does not happen then the program is in an infinite loop. Infinite loops are a bad thing. Consider the following code:

Example 2: C++ source code: infinite loop

```
loop_response = 'y';
while (loop_response == 'y')
  {
  cout ≪ "\nWhat is your age? ";
  cin ≫ age_user;
  cout ≪ "\nWhat is your friend's age? ";
  cin ≫ age_friend;
  cout ≫ "\nTogether your ages add up to: ";
  cout ≫ (age_user + age_friend);
  }
```

The programmer assigned a value to the flag before the loop which is correct. However, he forgot to update the flag. Every time the test expression is asked it will always be true. Thus, an infinite loop because the programmer did not provide a way to exit the loop (he forgot to update the flag). Consider the following code:

Example 3: C++ source code: infinite loop

```
loop_response = 'y';
while (loop_response = 'y')
  {
  cout ≪ "\nWhat is your age? ";
  cin ≫ age_user;
  cout ≪ "\nWhat is your friend's age? ";
  cin ≫ age_friend;
  cout ≫ "\nTogether your ages add up to: ";
  cout ≫ (age_user + age_friend);
  cout ≪ "\nDo you want to do it again? y or n ";
  cin ≫ loop_response;
  }
```

No matter what the user replies during the flag update, the test expression does not do a relational comparison but does an assignment. It assigns 'y' to the variable and asks if 'y' is true? Since all non-zero values are treated as representing true within the Boolean concepts of the C++ programming language, the answer to the test expression is true. Viola, you have an infinite loop.

Example 4: C++ source code: infinite loop

```
loop_response = 'y';
while (loop_response == 'y');
  {
  cout ≪ "\nWhat is your age? ";
  cin ≫ age_user;
  cout ≪ "\nWhat is your friend's age? ";
  cin ≫ age_friend;
  cout ≫ "\nTogether your ages add up to: ";
  cout ≫ (age_user + age_friend);
  cout ≪ "\nDo you want to do it again? y or n ";
  cin ≫ loop_response;
  }
```

The undesirable semi-colon on the end of while line causes the action of the while loop to be the "nothingness" between the closing parenthesis and the semi-colon. The program will infinitely loop because there is no action (that is no action and no update). If this is the first item in your program it will appear to start but there will be no output.

3 Counting Loops

The examples above are for an event controlled loop. The flag updating is an event where someone decides if they want the loop to execute again. Often the initialization sets the flag so that the loop will execute at least once.

Another common usage of the while loop is as a counting loop. Consider:

Example 5: C++ source code: while loop that is counting

```
counter = 0;
while (counter < 5)
  {
  cout << "\nI love ice cream!";
  counter++;
  }
```

The variable counter is said to be controlling the loop. It is set to zero (called initialization) before entering the while loop structure and as long as it is less than 5 (five); the loop action will be executed. But part of the loop action uses the increment operator to increase counter's value by one. After executing the loop five times (once for counter's values of: 0, 1, 2, 3 and 4) the expression will be false and the next line of code in the program will execute. A counting loop is designed to execute the action (which could be more than one statement) a set of given number of times. In our example, the message is displayed five times on the monitor. It is accomplished my making sure all four attributes of the while control structure are present and working properly. The attributes are:

- Initializing the flag
- Test expression
- Action or actions
- Update of the flag

Missing an attribute might cause an infinite loop or give undesired results (does not work properly).

3.1 Infinite Loops

Consider:

Example 6: C++ source code: infinite loop

```
counter = 0;
while (counter < 5)
  {
  cout << "\nI love ice cream!";
  }
```

Missing the flag update usually causes an infinite loop.

3.2 Variations on Counting

In the following example, the integer variable age is said to be controlling the loop (that is the flag). We can assume that age has a value provided earlier in the program. Because the while structure is a test before loop; it is possible that the person's age is 0 (zero) and the first time we test the expression it will be false and the action part of the loop would never be executed.

Example 7: C++ source code: while as a counting loop

```
while (0 < age)
  {
  cout ≪ "\nI love candy!";
  age--;
  }
```

Consider the following variation assuming that age and counter are both integer data type and that age has a value:

Example 8: C++ source code: while as a counting loop

```
counter = 0;
while (counter < age)
  {
  cout ≪ "\nI love corn chips!";
  counter++;
  }
```

This loop is a counting loop similar to our first counting loop example. The only difference is instead of using a literal constant (in other words 5) in our expression, we used the variable age (and thus the value stored in age) to determine how many times to execute the loop. However, unlike our first counting loop example which will always execute exactly 5 times; it is possible that the person's age is 0 (zero) and the first time we test the expression it will be false and the action part of the loop would never be executed.

4 Definitions

Definition 1: while
A test before iteration control structure available in C++.

Definition 2: loop attributes
Items associated with iteration or looping control structures.

Definition 3: initialize item
An attribute of iteration control structures.

Definition 4: might not happen
Indicating that test before loops might not execute the action.

Definition 5: event controlled
Using user input to control a loop.

Definition 6: counting controlled
Using a variable to count up or down to control a loop.