

While Loop Using Bloodshed Dev C++

By Janine Bouyssounouse

Bloodshed Dev C++ is a free program to make writing and compiling C++ programs easy to do. It makes executable files quickly so programs can be shared with others as soon as they are written and debugged. Bloodshed Dev C++ can be downloaded from the website:

<http://www.bloodshed.net/devcpp.html>.

There is more than one way to start writing programs in Dev C++. This document discusses using a project for each program. This encourages using an organized file structure to keep each program in a separate location for future use. Remember that reusing code is a good idea and it is nice to be able to find the folder with everything for a program in one place.

Click on the File menu and select New – Project. Choose Console Application for this program. Type in a name for the project, such as Display Input, in the name field in the bottom left corner of the window. Click on the OK button. Choose a place on the computer to save the project. It's a good idea to make a folder for each project. Once the project is saved, some basic items show up on the screen. These items give a shell of a C++ program and include things that are unique to Bloodshed Dec C++ as well.

```
#include <iostream>    - This tells the compiler to include files.
#include <stdlib.h>    - This tells the compiler to include files.

using namespace std;  - This tells the compiler that you will be using key
                      words that are included in namespace std.

int main(int argc, char *argv[])    - This starts the main function.
{                                     - Main is contained between { and }.

    system("PAUSE");                - This leaves the console window open until
                                    you are ready to close it. Otherwise you
                                    wouldn't be able to see the results of the
                                    program. Not all compilers require this.

    return 0;                       - This is how the main program ends.
}                                     - This signifies the end of the code for main.
```

The code for the main part of the program will be typed between the first curly brace ({}) and the system("PAUSE") line of code. The input and print functions for this program will be outside of this area, but the main program will refer to them.

Programs sometimes have a need to loop around to do something several times. There are several ways to do this. They each have their place in programs for different circumstances. Some of them can be used for the same purpose, but one might be much easier than another to program as well as to understand.

The while loop is the easiest of the methods to loop as it has the fewest parts to make it work. Basically a set of statements are executed over and over until a specific situation is met, which is set up in the while statement. This means that the statements in the loop have to have a way to affect the condition for the while loop to end or else it will never end. Writing endless loops is a common programming mistake. Please note how to stop an endlessly looping program on the computer, such as using ctrl-alt-del if needed. Endless loops will tie up computer resources and the computer may have to be reboot if it is too bad.

This program will count down to lift off. The user will get to choose what number to start with, then the while loop will handle the printing of the count down to the screen. To make the program a little less fragile, another while statement will be used to make sure the number entered by the user is greater than zero.

Type in the first lines of the program by clicking at the beginning of the first line and pressing enter a couple of times, then moving to the first blank line. Type in these first two lines of code:

```
// This program uses the while loop to count down to take off.  
// The user selects the number to start and the count down will  
// show on the screen.
```

The next line of code can be a comment stating your name as the programmer and the date the program was written.

```
// Written by Janine Bouyssounouse on 10/17/08
```

Type the following line of code after the using namespace statement:

```
void welcome();
```

A welcome statement will be displayed to explain what this program does. There is nothing sent to or received from the welcome function, so the void and empty parentheses are used in the function prototype.

On the last line of the program, we will comment the line to show that the main function is finished, so that it is not confused with the other functions listed after it.

The last line of code should look like this:

```
} // end main
```

Next we will start typing the welcome function at the end of the program, outside of the curly braces for the main function.

Skip a line and type:

```
// welcome function displays an opening message to  
// explain the program to the user  
void welcome()
```

Notice the comments are listed on the lines before the start of the function. The first line of the function looks exactly like the function prototype, except for the missing semicolon at the end.

On the next line of code, type in the function:

```
{  
    cout << "This program asks the user for a starting number ";  
    cout << "and then counts down from that number to blast off.\n\n";  
} // end of welcome function
```

This function uses no variables at all.

Call the welcome function in the main function to display the information to the user.

To do this, type the following code after the first curly brace in the main function and before the system("PAUSE") line of code:

```
welcome(); // This calls the welcome function
```

Type the following line of code after the void welcome(); statement:

```
int askNumber();
```

Int means there is an integer value returned to the main function from the askNumber function. AskNumber is the name of the function. The empty parentheses show that nothing is being passed to the askNumber function from the main program. The semicolon shows the end of the line of code.

At the end of the program, type in the askNumber function:

```
// askNumber function asks for an integer greater than zero from the user
// and passes it back to the main function, if the incorrect
// response is received, then the user is asked again
int askNumber()
{
    int response; // declares int variable response

    // displays prompt to get input
    cout << "\nPlease type an integer greater than zero: ";
    cin >> response; // places the user input into response variable

    // The while loop asks the question until an acceptable answer is given
    while (response <= 0)
    {
        // A revised print statement is shown to the user
        cout << "\nTry again. Please type an integer greater than zero: ";
        cin >> response;
    }

    return response; // sends the contents of response back to main
```

```
} // end of askNumber function
```

Before calling the askNumber function, another variable needs to be declared in the main function to hold the result of the askNumber function. Declare the new variable above the call to the welcome function at the beginning of the main function.

```
int number; // Declares a variable for the number
```

Now the new function needs to be called from the main function. The result of the function will be stored in the newly declared variable. Type the following code into the main function after the welcome function has been called:

```
number = askNumber(); // Calls askNumber function
```

Now it's time to display the count down. To do this the number input by the user will be sent to a print function.

Type the following line of code after the int askNumber(); statement:

```
void displayCountDown(int number);
```

Void means there is nothing returned to the main function from the displayCountDown function. An integer is passed to the displayCountDown function to display it on the screen for the user to see.

Type the following at the end of the program:

```
// displayCountDown function receives an integer to  
// display on the screen to start the count down  
void displayCountDown(int number)  
{  
    cout << "\nHere is the count down: \n"; // displayed one time  
  
    while (number > 0)  
    {  
        cout << number << ", "; // the numbers are displayed with a comma  
        number --; // this decreases the number each time through the loop  
    }  
}
```

```
}  
    cout << "...Blast Off!\n"; // prints only one time, it's outside the loop  
}  
} // end of the displayCountDown function
```

Now all that is left to do is call this new function from the main function. Type the following code after the askNumber function is called in the main function:

```
displayCountDown(number); // Passes variable to function
```

The program is finished. Here is the code:

```
// This program uses the while loop to count down to take off.  
// The user selects the number to start and the count down will  
// show on the screen.  
// Written by Janine Bouyssounouse on 10/17/08  
  
#include <iostream>  
#include <stdlib.h>  
  
using namespace std;  
void welcome();  
int askNumber();  
void displayCountDown(int number);  
  
int main(int argc, char *argv[])  
{  
    int number; // Declares a variable for the number  
  
    welcome(); // This calls the welcome function  
    number = askNumber(); // Calls askNumber function  
    displayCountDown(number); // Passes variable to function  
  
    system("PAUSE");  
    return 0;  
} // end main  
  
// welcome function displays an opening message to
```

```

// explain the program to the user
void welcome()
{
    cout << "This program asks the user for a starting number ";
    cout << "and then counts down from that number to blast off.\n\n";
} // end of welcome function

// askNumber function asks for an integer greater than zero from the user
// and passes it back to the main function, if the incorrect
// response is received, then the user is asked again
int askNumber()
{
    int response; // declares int variable response

    // displays prompt to get input
    cout << "\nPlease type an integer greater than zero: ";
    cin >> response; // places the user input into response variable

    // The while loop asks the question until an acceptable answer is given
    while (response <= 0)
    {
        // A revised print statement is shown to the user
        cout << "\nTry again. Please type an integer greater than zero: ";
        cin >> response;
    }

    return response; // sends the contents of response back to main
} // end of askNumber function

// displayCountDown function receives an integer to
// display on the screen to start the count down
void displayCountDown(int number)
{
    cout << "\nHere is the count down: \n"; // displayed one time

    while (number > 0)
    {
        cout << number << ", "; // the numbers are displayed with a comma
        number --; // this decreases the number each time through the loop
    }
}

```

```
cout << "...Blast Off!\n"; // prints only one time, it's outside the loop
} // end of the displayCountDown function
```

Save, compile and run the program to see if it works. Choose Compile and Run from the Execute menu.

Here is a display of the program:

This program asks the user for a starting number and then counts down from that number to blast off.

Please type an integer greater than zero: -3

Try again. Please type an integer greater than zero: 0

Try again. Please type an integer greater than zero: 10

Here is the count down:

10, 9, 8, 7, 6, 5, 4, 3, 2, 1, ...Blast Off!

Press any key to continue . . .

Please note that a test was done to see if the while statement was working in the askNumber function. Two invalid responses were given and then a valid response was given. It is very important to test different cases of program input to make sure the program is working properly. This becomes more important the more complex the program is.

Now write a program of your own.

Exercise 1: Write a program with an incomplete phrase, such as "Mary had a little lamb," but the word lamb is missing. Write a while loop to ask for the correct ending to the phrase to get out of the loop. Choose names for the functions that tell what they do.

Sample Code for Exercise 1:

```
// This program uses the while loop to ask for an ending
// to a phrase. The loop won't end until the correct word
// is entered.
// Written by Janine Bouyssounouse on 10/17/08

#include <iostream>
#include <stdlib.h>

using namespace std;
void welcome();
void askWord();

int main(int argc, char *argv[])
{
    welcome(); // This calls the welcome function
    askWord(); // Calls askNumber function

    system("PAUSE");
    return 0;
} // end main

// welcome function displays an opening message to
// explain the program to the user
void welcome()
{
    cout << "This program asks the user to finish a phrase. \n";
    cout << "The loop won't end until the phrase is complete.\n\n";
} // end of welcome function

// askWord function asks for a word to complete a phrase.
// It loops until the correct word is entered.
void askWord()
{
    string response; // declares string variable response

    // displays prompt to get input
    cout << "\nPlease finish the phrase: Mary had a little _____: ";
    cin >> response; // places the user input into response variable
```

```
// The while loop asks the question until an acceptable answer is given
while (response != "lamb")
{
    // A revised print statement is shown to the user
    cout << "\nTry again. Mary had a little _____: ";
    cin >> response;
}

cout << "\n\n Congratulations! Mary did have a little lamb.\n\n";
} // end of askWord function
```

Display from Sample Code for Exercise 1:

This program asks the user to finish a phrase.
The loop won't end until the phrase is complete.

Please finish the phrase: Mary had a little _____: cow

Try again. Mary had a little _____: pig

Try again. Mary had a little _____: lamb

Congratulations! Mary did have a little lamb.

Press any key to continue . . .