

Do 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 do while loop is very similar to the while loop. The main difference is that the do while loop will go through the loop at least one time no matter what, since it checks the condition at the end of the loop. The while loop checks the condition at the beginning of the loop, so there is a possibility of the while loop never executing if the condition is right. Basically a set of statements are executed over and over until a specific situation is met, which is set up in the while statement at the end of the do loop.

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 print a string multiple times on the screen after the user enters it, until the user enters end. The program will give the user at least one chance to enter a word. At any prompt, the user may enter end, so the loop could end on the first try if the user so desires.

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 lines of code:

```
// This program uses the do while loop to print to the screen
// a word entered by the user. The program will exit the loop
// when the user enters end.
```

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 word ";  
    cout << "and then prints that word several times on the screen. ";
```

```
cout << "Type end to stop the program.\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:

```
void displayWords();
```

Void means there is no value returned to the main function. displayWords is the name of the function. The empty parentheses show that nothing is being passed to the displayWords function from the main program. The semicolon shows the end of the line of code.

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

```
// displayWords function asks for a word from the user  
// and prints that word several times on the screen until  
// the user types the word end to get out of the loop  
void displayWords()  
{  
    string response; // declares string variable response  
  
    // the do loop executes at least one time  
    // and continues until end is entered  
    do  
    {  
        // Prompt for entry from user with instructions for stopping  
        cout << "\nEnter a word to be repeated: (type end to stop) ";  
        cin >> response; // assign user input to variable
```

```
// prints response on screen three times with a space in between
cout << "\n" << response << " " << response << " " << response;

} while (response != "end"); // end of do while loop

} // end of displayWords function
```

Please note a main difference between the while and the do while function is the semicolon at the end of the statements. The while loop has no semicolon at the end of it. But the do while loop will not work until there is a semicolon at the end of it. Think about the end curly brace ending the while loop, but there is more after the end curly brace in the do while loop, so the computer needs a way to know when it is done.

Now the new function needs to be called from the main function. Type the following code into the main function after the welcome function has been called:

```
displayWords(); // Calls displayWords function
```

The program is finished. Here is the code:

```
// This program uses the do while loop to print to the screen
// a word entered by the user. The program will exit the loop
// when the user enters end.
// Written by Janine Bouyssounouse on 10/17/08

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

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

int main(int argc, char *argv[])
{
    welcome(); // This calls the welcome function
    displayWords(); // Calls displayWords 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 word ";
    cout << "and then prints that word several times on the screen. ";
    cout << "Type end to stop the program.\n\n";
} // end of welcome function

// displayWords function asks for a word from the user
// and prints that word several times on the screen until
// the user types the word end to get out of the loop
void displayWords()
{
    string response; // declares string variable response

    // the do loop executes at least one time
    // and continues until end is entered
    do
    {
        // Prompt for entry from user with instructions for stopping
        cout << "\nEnter a word to be repeated: (type end to stop) ";
        cin >> response; // assign user input to variable

        // prints response on screen three times with a space in between
        cout << "\n" << response << " " << response << " " << response;

    } while (response != "end"); // end of do while loop
} // end of displayWords 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 word and then prints that word several times on the screen. Type end to stop the program.

Enter a word to be repeated: (type end to stop) one

one one one

Enter a word to be repeated: (type end to stop) free

free free free

Enter a word to be repeated: (type end to stop) special

special special special

Enter a word to be repeated: (type end to stop) end

end end endPress any key to continue . . .

Please note that the loop does not actually check to see if it is time to get out of the loop until after it prints the word end on the screen three times. It becomes obvious where the end line has been put in the program, since there is no end line after the word end is printed three times on the screen.

Now write a program of your own.

Exercise 1: Write a program to repeat any number entered by the user, until the user enters zero (0). The 0 will end the loop. Choose names for the functions that tell what they do.

Exercise 2: Write a program that shows the square of a number and continues to ask for more numbers to square until a zero (0) is entered to end the loop. Remember squaring a number means to multiply it by itself.

Example: $4^2 = 4 \times 4 = 16$. The number 4 squared is 16.

Sample Code for Exercise 1:

```
// This program uses the do while loop to print to the screen
// an integer entered by the user. The program will exit the loop
// when the user enters zero (0).
// Written by Janine Bouyssounouse on 10/17/08

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

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

int main(int argc, char *argv[])
{
    welcome(); // This calls the welcome function
    displayNumber(); // Calls displayNumber 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 an integer ";
    cout << "and then prints that integer to the screen. ";
    cout << "Type zero (0) to stop the program.\n\n";
} // end of welcome function

// displayNumber function asks for an integer from the user
// and prints that integer to the screen until
// the user types zero (0) to get out of the loop
void displayNumber()
{
    int response; // declares int variable response

    // the do loop executes at least one time
```

```
// and continues until 0
do
{
    // Prompt for entry from user with instructions for stopping
    cout << "\nEnter an integer: (type 0 to stop) ";
    cin >> response; // assign user input to variable

    // prints response on screen
    cout << "\nYou entered: " << response << "\n";

} while (response != 0); // end of do while loop
} // end of displayNumber function
```

Display from Sample Code for Exercise 1:

This program asks the user for an integer and then prints that integer to the screen. Type zero (0) to stop the program.

Enter an integer: (type 0 to stop) 5

You entered: 5

Enter an integer: (type 0 to stop) 19

You entered: 19

Enter an integer: (type 0 to stop) 43

You entered: 43

Enter an integer: (type 0 to stop) 0

You entered: 0

Press any key to continue . . .

Sample Code for Exercise 2:

```
// This program uses the do while loop to print to the screen
// the square of an integer entered by the user. The program will
// exit the loop when the user enters zero (0).
// Written by Janine Bouyssounouse on 01/20/09

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

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

int main(int argc, char *argv[])
{
    welcome(); // This calls the welcome function
    displaySquare(); // Calls displaySquare 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 an integer ";
    cout << "and then squares that integer and displays ";
    cout << "the result to the screen.\n";
    cout << "Type zero (0) to stop the program.\n\n";
} // end of welcome function

// displaySquare function asks for an integer from the user,
// squares it and prints that result to the screen until
// the user types zero (0) to get out of the loop
void displaySquare()
{
    int response; // declares int variable response
```

```
// the do loop executes at least one time
// and continues until 0
do
{
    // Prompt for entry from user with instructions for stopping
    cout << "\nEnter an integer to be squared: (type 0 to stop) ";
    cin >> response; // assign user input to variable

    // prints response on screen and its square
    cout << "\n" << response << " squared is ";
    cout << response * response << "\n";

} while (response != 0); // end of do while loop

} // end of displaySquare function
```

Display from Sample Code for Exercise 2:

This program asks the user for an integer and then squares that integer and displays the result to the screen.
Type zero (0) to stop the program.

Enter an integer to be squared: (type 0 to stop) 12

12 squared is 144

Enter an integer to be squared: (type 0 to stop) -3

-3 squared is 9

Enter an integer to be squared: (type 0 to stop) 0

0 squared is 0

Press any key to continue . . .