

# If Statements 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 Dev 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.

Most computer programs have some kind of decision making, as in doing certain things based on user input or calculations done behind the scenes. A program might give a message if a math problem was done correctly and a different message if the problem was done incorrectly. The if statement is one of the decision makers in programming.

For this program, the number used will be input by the user. Then the program will determine if the number is even or odd and tell the user the result. The even/odd determination will be done by using mod. If the result is zero, then the number is even (there is no remainder when dividing by two). If the result is one, then the number is odd (there is a remainder when dividing by two).

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 if statements to determine if  
// a number input by the user is even or odd
```

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 10/11/08
```

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. This is helpful since there is no input from the user and everything is done behind the scenes. 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 gets a number from the user.\n";  
    cout << "Then the program determines if the number is even or odd.\n";  
    cout << "The program displays the number and whether ";  
    cout << "it is even or odd.\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 after the welcome function:

```
// askNumber function asks for an integer from the user
// and passes it back to the main function
int askNumber()
{
    int response; // declares int variable response
    cout << "\nPlease type an integer: "; // displays prompt to get input
    cin >> response; // places the user input into response variable
    return response; // sends the contents of response back to main
} // end of askNumber function
```

The askNumber function gets a number from the user and sends it back to the main function. This means a variable needs to be declared in the main function to receive the input from 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 user input
```

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
```

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

```
void evenOdd(int number);
```

Void means nothing will be returned from this function. The statement to the user for even or odd will be done within the same function as determining even or odd. It will all be handled in the if statements. The number variable is passed to the function to use in the function.

Type the following line of code after the `char evenOdd(int number);` statement:

```
void displayEvenOdd(char evenOddChar, int number);
```

Void means there will be nothing returned to the main function from the `displayEvenOdd` function. A character is passed to the function which came from the `evenOdd` function. This drives the print statements in the `displayEvenOdd` function. The random number generated in the `randomNumber` function is also passed, so that the number and its even or oddness can be displayed in the same statement.

Now that a number has been received from the user and passed back to the main function, it needs to be passed to the `evenOdd` function to determine whether it is even or odd. It would help to type out the function before it gets called in the main function, so type the following code at the end of the program, since the function prototype is already typed at the top of the program:

```
// evenOdd function determines if the number passed to it
// is even or odd and displays a message to the user
void evenOdd(int number)
{
    // The modulus operator is used to look at the remainder
    // to determine if the number is even or odd.
    // An odd number would have a remainder of 1 if divided by 2
    if (number % 2 == 1)
        cout << "\nThe number " << number << " is odd.\n";

    // An even number would have a remainder of 0 if divided by 2
}
```

```
else if (number % 2 == 0)
    cout << "\nThe number " << number << " is even.\n";

    // This statement should never execute.
    // If it does, then something is wrong.
else
    cout << "\nThe number is neither even nor odd.\n";
} // end of evenOdd function
```

The if statement is used in this function to determine which message should be given to the user based on dividing the number by two. This is an example of using the if with the else if and an else statement. If the first if statement is satisfied, the program falls to the cout statement. If the first if statement is not satisfied, then the program goes to the else if statement. If this statement is satisfied, then the program falls to the cout statement. If for some reason the number is neither even or odd, then something is wrong and the program falls to the else statement and tells the user something is wrong with the program.

Notice the semicolon is not placed until the end of the if statement. An if statement can also have braces to start and end it with a semicolon after the closing brace. The cout statement is the end of the if statement, so the semicolon needs to be after the cout statement to keep the cout statement part of the if statement.

The new function needs to be called from main or else it will not be used in the program. Type in the following line after the askNumber function is called in the main function.

```
evenOdd(number); // Passes number to evenOdd function
```

This passes the number variable to the evenOdd function and prints the results of the evenOdd function.

The program is finished. Here is the code:

```
// This program uses if statements to determine if
// a number input by the user is even or odd
// Written by Janine Bouyssounouse on 10/11/08
```

```

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

using namespace std;

void welcome();
int askNumber();
void evenOdd(int number);

int main(int argc, char *argv[])
{
    int number; // Declares a variable for the user input

    welcome(); // This calls the welcome function
    number = askNumber(); // Calls askNumber function
    evenOdd(number); // Passes number to evenOdd 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 gets a number from the user.\n";
    cout << "Then the program determines if the number is even or odd.\n";
    cout << "The program displays the number and whether ";
    cout << "it is even or odd.\n\n";
} // end of welcome function

// askNumber function asks for an integer from the user
// and passes it back to the main function
int askNumber()
{
    int response; // declares int variable response
    cout << "\nPlease type an integer: "; // displays prompt to get input
    cin >> response; // places the user input into response variable
    return response; // sends the contents of response back to main
}

```

```

} // end of askNumber function

// evenOdd function determines if the number passed to it
// is even or odd and displays a message to the user
void evenOdd(int number)
{
    // The modulus operator is used to look at the remainder
    // to determine if the number is even or odd.
    // An odd number would have a remainder of 1 if divided by 2
    if (number % 2 == 1)
        cout << "\nThe number " << number << " is odd.\n";

    // An even number would have a remainder of 0 if divided by 2
    else if (number % 2 == 0)
        cout << "\nThe number " << number << " is even.\n";

    // This statement should never execute.
    // If it does, then something is wrong.
    else
        cout << "\nThe number is neither even nor odd.\n";
} // end of evenOdd 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 run three times:

This program gets a number from the user.  
Then the program determines if the number is even or odd.  
The program displays the number and whether it is even or odd.

Please type an integer: 7

The number 7 is odd.  
Press any key to continue . . .

This program gets a number from the user.  
Then the program determines if the number is even or odd.

The program displays the number and whether it is even or odd.

Please type an integer: 6

The number 6 is even.

Press any key to continue . . .

This program gets a number from the user.

Then the program determines if the number is even or odd.

The program displays the number and whether it is even or odd.

Please type an integer: -1

The number is neither even nor odd.

Press any key to continue . . .

This program waited for input on the first line. Once the user typed the input and pressed the enter key, then the second line of code printed. Please note that the program works just fine under usual circumstances, but using a negative one threw off the program output. Trying many different scenarios for a program is part of testing the program to see if it works the way it is expected to work. This program might need to be modified to not accept negative numbers or to handle the special case of the negative one. Software testing is as important as the programming itself. If the program doesn't work the way it's supposed to work, then it isn't going to do much good. Strong software testers are a valuable asset to any programming team. They help weed out the bugs in programs to make them as good as possible.

Now write a program of your own using the if statement.

Exercise 1: Write a program with an input function asking the user for two numbers, then pass this information to a function to determine which number is greater than the other number and display the results. Choose names for the functions that tell what they do.

## Sample Code for Exercise 1:

```
// This program gets input from the user, determines which is bigger
// and displays the result using functions
// Written by Janine Bouyssounouse on 10/11/08

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

using namespace std;

int askNumber();
void bigger(int num1, int num2);

int main(int argc, char *argv[])
{
    int num1, num2; // declares two variables for user input

    num1 = askNumber(); // calls askNumber and gives result to num1
    num2 = askNumber(); // calls askNumber and gives result to num2
    bigger (num1, num2); // passes variables to bigger function

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

// askNumber function asks for an integer from the user
// and passes it back to the main function
int askNumber()
{
    int response; // declares int variable response
    cout << "\nPlease type an integer: "; // displays prompt to get input
    cin >> response; // places the user input into response variable
    return response; // sends the contents of response back to main
} // end of askNumber function

// bigger function gets two integers from the main function
// and displays which is bigger to the user on the screen
void bigger(int num1, int num2)
{
```

```
if (num1 > num2)
    cout << "\n" << num1 << " is bigger than " << num2 << "\n";

else if (num2 > num1)
    cout << "\n" << num2 << " is bigger than " << num1 << "\n";

else
    cout << "\nThe numbers are the same.\n";

} // end bigger function
```

Display from Sample Code for Exercise 1:

```
Please type an integer: 4
Please type an integer: 8
8 is bigger than 4
Press any key to continue . . .

Please type an integer: 9
Please type an integer: 1
9 is bigger than 1
Press any key to continue . . .

Please type an integer: 2
Please type an integer: 2
The numbers are the same.
Press any key to continue . . .
```