

# High Low 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.

**The Programming Challenge:** Write a program to play a guessing game. The computer will pick a random number between 1 and 100. The user will guess a number and the computer will respond by saying Too High or Too Low. The number of guesses will be counted and displayed once the number is guessed. Don't forget to include a welcome message that explains the game to the users.

**Hints:** Work on the different parts of the program, one at a time. Work on the welcome message first, since that is the easiest part.

Then work on a function to pick a random number.

Then work on a function for the main game loop. A do while loop will work great for a game loop. It can keep going until the correct guess is made. The number of guesses can be returned from the function to send to a display function to end the program.

Make a function to announce the winner of the game when the game is over. It can be called when the game loop function ends.

## **The Walk Through:** Starting with the welcome message...

The welcome message needs to explain the rules of the game to the users so they know what is happening and how to play the game.

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 is a game with one human user guessing the computer's  
// number between 1 and 100. The user enters a guess and the computer  
// tells the user if the guess is too high or too low. The total number of  
// guesses is counted and displayed at the end of the game.
```

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 12/21/08
```

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 << "The computer will choose a number between 1 and 100.\n";
    cout << "The user will try to guess the number.\n";
    cout << "The user will be told if the guess is too high or too low.\n";
    cout << "The total number of guesses will show at the end.\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
```

You can compile and run the program at this stage to see if things are working so far. It is a good idea to check the program after creating each function, so that it is easier to tell where problems are located.

**The Walk Through:** Initializing the variables...

Type the following lines of code before the `void welcome;` statement in the main function:

```
int matchTotal = 21; // initializing matchTotal for start of game
```

```
string whoseTurn = "Player 1"; // initializing whoseTurn for the first turn
int turnAmount = 0; // initializing matches taken on each turn
```

**The Walk Through:** The random number function...

The computer needs to pick a random number between 1 and 100 before the user makes a guess. Nothing needs to be sent to the function, unless variables are being used to choose the highest and lowest numbers possible (a harder programming challenge). But there needs to be a number coming back from the function to state what the number is.

Type the following line of code after the void welcome(); statement in the function prototype section before the main function:

```
int pickRandom();
```

int means there is an integer returned to the main program when the pickRandom function finishes. This integer is the number to be guessed in the game. The function doesn't need anything passed to it, so the parentheses are empty.

In order to use the random function and to initialize it with a seed of the computer time, some more lines of code need to be entered after the two standard #include statements.

```
#include <cstdlib>
#include <ctime>
```

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

```
// pickRandom function picks a random number between 1 and 100.
// The number is passed back to where it was called.
int pickRandom()
{
    srand(time(0)); // seeds the random number generator
    int number = (1 + rand() % 100); // random number between 1 and 100

    return number; // sends the number back
} // end of pickRandom function
```

Before the function is called, a variable to hold the number to be guessed needs to be created in the main function. Type the following line of code before the welcome(); call in the main function:

```
int number = 0; // creates a variable and initializes it to zero
```

Now the pickRandom function can be called. Type the following line of code after the welcome(); call in the main function:

```
number = pickRandom(); // number gets value returned from function
```

A temporary cout statement can be placed after this line of code to test to see if the pickRandom function is returning random numbers. Compile and run the program to check if it is still working.

**The Walk Through:** The game loop function...

Type the following line of code after the int pickRandom(); line of code in the function prototypes section above the main function:

```
int playGame(int number);
```

Type the following code at the end of the program:

```
// playGame function takes care of the main game loop of the game.
// It will stay in the loop until the game is won (user guesses number).
// The function is passed the number to guess variable and returns the
// number of guesses it took to get it. The user is told if the guesses are
// too high or too low until the correct guess is made.
int playGame(int number)
{
    int guess = 0; // initializing the guess made by the user
    int numGuesses = 0; // initializing amount of guesses

    // the main game loop
    do
    {
        // start play
```

```
cout << "\nWhat is your guess? "; // prompt for guess
cin >> guess; // accept guess from user and assign it to guess
numGuesses++; // increment numGuesses for each guess

// determining if Too high or Too low should be shown to user
if (guess > number)
    cout << "\nToo high.";

if (guess < number)
    cout << "\nToo low.";
} while (guess != number); // exits loop when number is guessed

return numGuesses; // returning total guesses the user took
} // end of playGame function
```

We need another variable in the main function to hold the value returned by this function. Type the following code after the `int number = 0;` line of code at the beginning of the main function:

```
int numGuesses = 0; // creates a variable and initializes it to zero
```

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

```
numGuesses = playGame(number); // calls playGame
```

Test to see if the program is working so far. Choose **Compile and Run** from the **Execute** menu.

**The Walk Through:** Displaying the end of the game...

There is only one function left to write. It's the function to tell the display how many guesses it took to guess the right number. It would be good if the correct guess and the number of guesses it took to get there were both displayed, so that requires two variables being passed the function and nothing will be returned from the function, so it will be a void function.

Type the following code after the `int playGame(int number);` statement in the function prototypes before the main function:

```
void endOfGame(int number, int numGuesses);
```

At the end of the program, type the following code:

```
// endOfGame displays the results of the game.
// The number and the number of guesses are passed to the function.
void endOfGame(int number, int numGuesses)
{
    cout << "\n\nEnd of Game!\n\n";
    cout << number << " is the number!\n\n";
    cout << "It took " << numGuesses << " guesses to get it!\n\n";
} // end of endOfGame function
```

Now the function needs to be called from the main function. Type the following code after the `numGuesses = playGame(number);` statement in the main function:

```
endOfGame(number, numGuesses); // calls the endOfGame function
```

The program is finished. Here is the code:

```
// This program is a game with one human user guessing the computer's
// number between 1 and 100. The user enters a guess and the computer
// tells the user if the guess is too high or too low. The total number of
// guesses is counted and displayed at the end of the game.
// Written by Janine Bouyssounouse on 12/21/08

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

using namespace std;
void welcome();
int pickRandom();
int playGame(int number);
```

```

void endOfGame(int number, int numGuesses);

int main(int argc, char *argv[])
{
    int number = 0; // creates a variable and initializes it to zero
    int numGuesses = 0; // creates a variable and initializes it to zero

    welcome(); // This calls the welcome function
    number = pickRandom(); // number gets value returned from function
    numGuesses = playGame(number); // calls playGame
    endOfGame(number, numGuesses); // calls the endOfGame function

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

// welcome function displays an opening message to
// explain the program to the user
void welcome()
{
    cout << "The computer will choose a number between 1 and 100.\n";
    cout << "The user will try to guess the number.\n";
    cout << "The user will be told if the guess is too high or too low.\n";
    cout << "The total number of guesses will show at the end.\n\n";
} // end of welcome function

// pickRandom function picks a random number between 1 and 100.
// The number is passed back to where it was called.
int pickRandom()
{
    srand(time(0)); // seeds the random number generator
    int number = (1 + rand() % 100); // random number between 1 and 100

    return number; // sends the number back
} // end of pickRandom function

// playGame function takes care of the main game loop of the game.
// It will stay in the loop until the game is won (user guesses number).
// The function is passed the number to guess variable and returns the
// number of guesses it took to get it. The user is told if the guesses are

```

```

// too high or too low until the correct guess is made.
int playGame(int number)
{
    int guess = 0; // initializing the guess made by the user
    int numGuesses = 0; // initializing amount of guesses

    // the main game loop
    do
    {
        // start play
        cout << "\nWhat is your guess? "; // prompt for guess
        cin >> guess; // accept guess from user and assign it to guess
        numGuesses++; // increment numGuesses for each guess

        // determining if Too high or Too low should be shown to user
        if (guess > number)
            cout << "\nToo high.";

        if (guess < number)
            cout << "\nToo low.";
    } while (guess != number); // exits loop when number is guessed

    return numGuesses; // returning total guesses the user took
} // end of playGame function

// endOfGame displays the results of the game.
// The number and the number of guesses are passed to the function.
void endOfGame(int number, int numGuesses)
{
    cout << "\n\nEnd of Game!\n\n";
    cout << number << " is the number!\n\n";
    cout << "It took " << numGuesses << " guesses to get it!\n\n";
} // end of endOfGame 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:

The computer will choose a number between 1 and 100.  
The user will try to guess the number.  
The user will be told if the guess is too high or too low.  
The total number of guesses will show at the end.

What is your guess? 50

Too low.

What is your guess? 75

Too low.

What is your guess? 90

Too high.

What is your guess? 85

Too high.

What is your guess? 80

End of Game!

80 is the number!

It took 5 guesses to get it!

Press any key to continue . . .

Enjoy playing. Try to get the lowest total number of guesses as possible.  
Think about a strategy that might help lower the number of guesses made in each game.