

Random Function 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 random function is a key part of programming. There are many times when a programmer has a need for a random number. The problem with random numbers are they aren't really random. There is a random number table the program refers to and if the random number generator isn't seeded with something to start reading the table at a different place, then the "random" numbers will be the same ones every time. This can be useful for debugging purposes, but usually a program will actually need different numbers each time.

The time on the computer is a great way to seed the random number generator, since the theory is that it won't be the same time every time the program is run. There is a need for more include statements to go along with the use of the random function in C++.

This program will list off some random numbers on the screen. If all goes well, then every time the program is run, these numbers will be different. This program is an introduction to the use of random numbers in programs. There are many, many different ways they can be used in other programs. It is one of the more useful functions in programming.

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 random number generator to show  
// a listing of random numbers 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 10/15/08
```

In order to seed the random number generator with the time on the computer, another include statement must be added. Type the following code after the first two include statements:

```
#include <ctime>
```

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 generates random numbers and ";  
    cout << "prints them on the screen for the user to see.\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 randomNumber();
```

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

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

```
// randomNumber function generates a random number  
// and returns it to the main function  
int randomNumber()  
{  
    srand(time(0)); // This seeds the random number generator  
    int randNumber = rand(); // This gets a random number as an integer  
    const int MAX = 100; // Sets a constant variable for the maximum  
  
    // The MAX variable says there will be 100 numbers in the allowed  
    // responses and the number 1 says zero will not be included  
    int number = (randNumber % MAX) + 1;
```

```
    return number; // This sends the random number back to main
} // end of randomNumber function
```

The random number generator is seeded with the time from the computer. A variable is created to store the random number generated. A constant variable is used for the maximum number of the random number. This is done so that when the program needs to be changed to use a different maximum number, the programmer only needs to go to the constant declaration to make the change. The change is made where ever the MAX variable is used in the program. Then another variable is created to store the manipulated random number that fits the parameters needed by the program.

Before calling the randomNumber function, another variable needs to be declared in the main function to hold the result of the randomNumber 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 random 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 = randomNumber(); // Calls randomNumber function
```

Now it's time to display the random number. To do this the random number will be sent to a print function.

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

```
void displayRandom(int number);
```

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

Type the following at the end of the program:

```
// displayRandom function receives an integer to
// display on the screen
void displayRandom(int number)
{
    cout << "\nThe number " << number;
    cout << " is a random number.\n";
} // end of the displayRandom function
```

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

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

There are many ways to show more than one random number on the screen. For this program, a simplified way to do this will be used. The randomNumber and displayRandom functions will be called multiple times to show multiple random numbers on the screen for the user to see.

Type the following after the displayRandom function is called in the main function:

```
system("PAUSE");
number = randomNumber(); // Calls randomNumber function
displayRandom(number); // Passes variable to function
system("PAUSE");
number = randomNumber(); // Calls randomNumber function
displayRandom(number); // Passes variable to function
```

The system("PAUSE"); statements were added to slow the system down. If all three statements are run one right after the other, the random number generator will be seeded with the same time on the clock, since the program runs so fast. There are other ways to resolve this situation, but for now this will work. The user will simply press the enter key to see the next random number. This will put a delay into the program to get a different random number from the system due to the different time seed.

The program is finished. Here is the code:

```

// This program uses the random number generator to show
// a listing of random numbers on the screen
// Written by Janine Bouyssounouse on 10/15/08

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

using namespace std;
void welcome();
int randomNumber();
void displayRandom(int number);

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

    welcome(); // This calls the welcome function
    number = randomNumber(); // Calls randomNumber function
    displayRandom(number); // Passes variable to function
    system("PAUSE");
    number = randomNumber(); // Calls randomNumber function
    displayRandom(number); // Passes variable to function
    system("PAUSE");
    number = randomNumber(); // Calls randomNumber function
    displayRandom(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 generates random numbers and ";
    cout << "prints them on the screen for the user to see.\n\n";
} // end of welcome function

// randomNumber function generates a random number

```

```

// and returns it to the main function
int randomNumber()
{
    srand(time(0)); // This seeds the random number generator
    int randomNumber = rand(); // This gets a random number as an integer
    const int MAX = 100; // Sets a constant variable for the maximum

    // The MAX variable says there will be 100 numbers in the allowed
    // responses and the number 1 says zero will not be included
    int number = (randNumber % MAX) + 1;

    return number; // This sends the random number back to main
} // end of randomNumber function

// displayRandom function receives an integer to
// display on the screen
void displayRandom(int number)
{
    cout << "\nThe number " << number;
    cout << " is a random number.\n";
} // end of the displayRandom 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 generates random numbers and prints them on the screen for the user to see.

The number 88 is a random number.
Press any key to continue . . .

The number 95 is a random number.
Press any key to continue . . .

The number 11 is a random number.

Press any key to continue . . .

Now write a program of your own.

Exercise 1: Write a program with a random number generator that generates random numbers between 1 and 6 as if a six sided die were being tossed. Use a print function to display the random number. Choose names for the functions that tell what they do.

Sample Code for Exercise 1:

```
// This program uses the random number generator to show
// a random number on the screen
// Written by Janine Bouyssounouse on 10/15/08

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

using namespace std;
void welcome();
int randomNumber();
void displayRandom(int number);

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

    welcome(); // This calls the welcome function
    number = randomNumber(); // Calls randomNumber function
    displayRandom(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 generates a random number and ";
    cout << "prints it on the screen for the user to see.\n\n";
} // end of welcome function

// randomNumber function generates a random number
// and returns it to the main function
int randomNumber()
{
    srand(time(0)); // This seeds the random number generator
```

```

int randomNumber = rand(); // This gets a random number as an integer
const int MAX = 6; // Sets a constant variable for the maximum

// The MAX variable says there will be 100 numbers in the allowed
// responses and the number 1 says zero will not be included
int number = (randNumber % MAX) + 1;

return number; // This sends the random number back to main
} // end of randomNumber function

// displayRandom function receives an integer to
// display on the screen
void displayRandom(int number)
{
    cout << "\nThe number " << number;
    cout << " is a random number.\n";
} // end of the displayRandom function

```

Display from Sample Code for Exercise 1:

This program generates a random number and prints it on the screen for the user to see.

The number 6 is a random number.
Press any key to continue . . .