

For 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 for loop is used for looping a specific number of times, as opposed to the while and do while loops which look for specific conditions to break out of the loop. A variable is used in the for loop to control how many times the loop is performed. The for loop contains an initialization of the counter variable, the condition for when the loop will stop and a way for the counter variable to be increased or decreased.

This means that the conditions for the loop need to be set up properly so that the loop will eventually end, otherwise an endless loop can be created by accident. 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 calculate a factorial (example: 5 factorial or 5! is $5 \times 4 \times 3 \times 2 \times 1 = 120$) then print the result on the screen. The program will ask the user what number they would like to see used and there will be a print statement to show the progress through the multiple loops. Then the final answer will be given.

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 for loop to calculate the factorial of a number
// provided by the user. There will be a print statement in the loop to
// show the progress of the calculation.
```

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 << "This program calculates the factorial of a number. ";  
    cout << "For example, 3 factorial or 3! is 3 x 2 x 1 = 6. ";
```

```
cout << "The user will be asked what number to use.\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. 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 for the function prototype.

At the end of the program, type in the askNumber 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  
    cout << "\n"; // starts a new line  
    return response; // sends the contents of response back to main  
} // end of askNumber function
```

This function uses a variable to hold the input from the user. The variable name in the function is only for that function and a separate variable needs to be declared in the main function in order to handle the data returned from the askNumber function. I am using different variable names to illustrate the

fact that they are actually different, even though they share the same information.

Declare the variable in the main function. Then call the askNumber function to assign the returned value to the newly declared variable.

To do this, type the following code after the first curly brace in the main function and before the welcome function is called:

```
int num; // declares the variable
```

Leave a blank line of space after the variable is declared to separate the variable declaration from the function calls. Type the following code into the main function after the welcome function has been called:

```
num = askNumber(); // calls askNumber and gives the result to num
```

Now that we know what number will be used in the factorial calculation, we need to create the function to do the calculations.

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

```
int factorialCalc(int num);
```

int means an integer will be passed back to the main function. It will be the result of the calculation. factorialCalc is the name of the function. int num in the parentheses means an integer will be passed to the factorialCalc function. This is the number that will be used for the factorial calculation.

Type the code for the function at the end of the program:

```
// factorialCalc is a function which will calculate the factorial
// of the number being passed to it and will return the result
// of the calculation back to where it was called.
int factorialCalc(int num)
{
    int number = num + 1; // setting up number to be used in for statement
    int calculation = 1; // declaring variable to hold the result
```

```
for (int n = 1; n < number; n++)
{
    calculation = calculation * n;
    // The print statement in the loop shows the progress of the calculation.
    cout << "Now it's " << calculation << "\n";
}

return calculation; // sends the result of the calculation back
} // end of factorialCalc function
```

Since we are sending a variable back from the factorialCalc function, we need to declare a variable to receive it from the function. Type the following line of code after the int num; line of code in the main function:

```
int answer; // declares the variable to receive the calculation
```

Now we need to call the factorialCalc function. Type the following line of code after the num = askNumber(); line of code in the main function:

```
answer = factorialCalc(num); // sends num and receives answer
```

Now we need a function to display the answer to the calculation. Type the following line of code after the int factorialCalc(int num); before the main function:

```
void displayAnswer(int answer, int num);
```

That was the function prototype and now the function needs to be typed at the end of the program. Type the following code after the factorialCalc function at the end of the program:

```
// displayAnswer receives the answer and the number for the calculation,
// then displays the complete message to the user
void displayAnswer(int answer, int num)
{
    cout << "\n" << num << " factorial is " << answer << "\n";
} // end of displayAnswer function
```

The last step is to call the displayAnswer function from the main function. Type the following code after the answer = factorialCalc(num); line of code in the main function:

```
displayAnswer(answer, num); // sends answer and num to displayAnswer
```

The program is finished. Here is the code:

```
// This program uses the for loop to calculate the factorial of a number
// provided by the user. There will be a print statement in the loop to
// show the progress of the calculation.
// Written by Janine Bouyssounouse on 12/21/08

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

using namespace std;
void welcome();
int askNumber();
int factorialCalc(int num);
void displayAnswer(int answer, int num);

int main(int argc, char *argv[])
{
    int num; // declares the variable
    int answer; // declares the variable to receive the calculation

    welcome(); // This calls the welcome function
    num = askNumber(); // calls askNumber and gives the result to num
    answer = factorialCalc(num); // sends num and receives answer
    displayAnswer(answer, num); // sends answer and num to displayAnswer

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

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

```

void welcome()
{
    cout << "This program calculates the factorial of a number. ";
    cout << "For example, 3 factorial or 3! is 3 x 2 x 1 = 6. ";
    cout << "The user will be asked what number to use.\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
    cout << "\n"; // starts a new line
    return response; // sends the contents of response back to main
} // end of askNumber function

// factorialCalc is a function which will calculate the factorial
// of the number being passed to it and will return the result
// of the calculation back to where it was called.
int factorialCalc(int num)
{
    int number = num + 1; // setting up number to be used in for statement
    int calculation = 1; // declaring variable to hold the result

    for (int n = 1; n < number; n++)
    {
        calculation = calculation * n;
        // The print statement in the loop shows the progress of the calculation.
        cout << "Now it's " << calculation << "\n";
    }

    return calculation; // sends the result of the calculation back
} // end of factorialCalc function

// displayAnswer receives the answer and the number for the calculation,
// then displays the complete message to the user
void displayAnswer(int answer, int num)
{

```

```
cout << "\n" << num << " factorial is " << answer << "\n";  
} // end of displayAnswer 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 calculates the factorial of a number. For example, 3 factorial or 3! is $3 \times 2 \times 1 = 6$. The user will be asked what number to use.

Please type an integer: 4

Now it's 1

Now it's 2

Now it's 6

Now it's 24

4 factorial is 24

Press any key to continue . . .

Please note the adjustment made to the num variable in the factorialCalc function allows the for statement to loop through the correct number of times to get the correct calculation. The num variable is not changed during this process, so it can still be used to display the answer in the displayAnswer function at the end of the program.

Now write a program of your own.

Exercise 1: Write a program to print dollar signs across the screen. Let the user decide how many dollar signs will show on the screen. Choose names for the functions that tell what they do.

Sample Code for Exercise 1:

```
// This program uses the for loop to display dollar signs on the screen.
// The user determines how many will be displayed.
// Written by Janine Bouyssounouse on 12/21/08

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

using namespace std;
void welcome();
int askNumber();
void displaySigns(int num);

int main(int argc, char *argv[])
{
    int num; // declares the variable

    welcome(); // This calls the welcome function
    num = askNumber(); // calls askNumber and gives the result to num
    displaySigns(num); // sends num to displaySigns

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

// welcome function displays an opening message to
// explain the program to the user
void welcome()
{
    cout << "This program prints dollar signs on the screen. ";
    cout << "The user will be asked how many to display.\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
cout << "\n"; // starts a new line
return response; // sends the contents of response back to main
} // end of askNumber function

// displaySigns is a function which will display dollar signs
// on the screen the specified number of times
void displaySigns(int num)
{
    int number = num + 1; // setting up number to be used in for statement

    for (int n = 1; n < number; n++)
    {
        // The print statement displays the dollar sign once per loop.
        cout << "$";
    }

    cout << "\n"; // ends the line after printing the dollar signs
} // end of displaySigns function

```

Display from Sample Code for Exercise 1:

This program prints dollar signs on the screen. The user will be asked how many to display.

Please type an integer: 5

\$\$\$\$\$

Press any key to continue . . .