

Print Functions 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 Print Functions, 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 print function for this program will be outside of this area, but the main program will refer to it. It's a little like outsourcing or placing someone in charge on a specific task and then coming back for more instructions.

Commenting code is an important step in programming. Adding comments to the code explains to you and others what the program is trying to do. It can help with debugging the program or for future updates once the memory fades on what this program was intended to do.

The first line of the program can be a comment telling who wrote the program and a brief description of what the program does. Comments can be done in multiple ways, for now, the double slash will be used.

Type in the first line 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 this first line of code:

```
// This program shows the use of functions for print statements
```

The second 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 06/13/08
```

Comments such as these can be on their own lines of code or they can be at the end of another line of code to explain that line. This can seem redundant, but to someone new to writing programs it can help with understanding what the program is doing at each step. It will also help others to figure out what is going on at a later date. Many programmers spend time updating code written by other programmers.

Since functions will be used in this program, function prototypes will also be used. This tells the program what to expect. Function prototypes are listed after the include statements and before the main function. We will be typing the actual function at the end of the program.

Type the following line of code after the include namespace statement:

```
void welcome();
```

Void means there is nothing returned to the main function from the welcome function. Welcome is the name of the function. The empty parentheses show that nothing is being passed to the welcome function from the main program. There will be more information on this later. The semicolon shows the end of the line of code.

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:

```
void welcome() // intro text displayed at beginning of program
```

Notice the comment is listed after the empty parentheses and there is no semicolon after the empty parentheses. So the first line of the function looks exactly like the function prototype, except for the semicolon at the end.

On the next line of code, type in the function:

```
{  
    cout << "\tWelcome to my program!\n\n";  
    cout << "This program just prints this welcome statement."  
    cout << " There is a space in front of this sentence.\n\n";  
    cout << "There is a blank line above this sentence."  
} // end welcome function
```

Now that the function has a prototype and the function is typed at the end of the program, we can call the function in the main part of the program. If we

don't call the function, it won't be used in the program. This allows you to use it where you want it in the main program. There can be several functions that are called multiple times in a program. The main function takes care of when the other functions are used.

To call the function, type the function name with the parentheses on a line of code in the main function.

After the first curly brace in the main function and before the `system("PAUSE")` line of code, type the following:

```
welcome(); // this calls the welcome function
```

The program is finished. Here is the code:

```
// This program shows the use of functions for print statements
// Written by Janine Bouyssounouse on 06/13/08

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

using namespace std;

void welcome();

int main(int argc, char *argv[])
{
    welcome(); // this calls the welcome function

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

void welcome() // intro text displayed at beginning of program
{
    cout << "\tWelcome to my program!\n\n";
    cout << "This program just prints this welcome statement.";
    cout << " There is a space in front of this sentence.\n\n";
    cout << "There is a blank line above this sentence.";
```

```
} // end welcome 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:

```
Welcome to my program!
```

```
This program just prints this welcome statement. There is a space in front of  
th  
is sentence.
```

```
There is a blank line above this sentence.Press any key to continue . . .
```

Notice there is a space in front of the first line of text that was caused by the `\t` in front of it in the line of code. The `\t` moves the text over on the screen, or tabs it as in a word processor program. Text runs together unless spaces, tabs or new lines are typed into the code. To add a blank line before the "press any key to continue..." message, just type `\n\n` to add two new lines at the end of the last print statement in the welcome function. Then save, compile and run again to see the change.

Now write a program of your own to write your own print function.

Exercise 1: Write a program with a print function to display your favorite colors. Choose a name for the function that tells what it does.

Exercise 2: Write a program with two print functions and call them each from the main function. Write one function to list your favorite foods and another to list your hobbies.

Exercise 3: Write a program using a print function to print a right triangle made out of stars (*) which is five stars high and five stars wide at the base of the triangle.

Sample Code for Exercise 1:

```
// This program shows the use of functions for print statements
// Written by Janine Bouyssounouse on 06/13/08

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

using namespace std;

void colors();

int main(int argc, char *argv[])
{
    colors(); // this calls the colors function
    system("PAUSE");
    return 0;
} // end main

void colors() // text to display favorite colors
{
    cout << "\tThese are my favorite colors:\n\n";
    cout << "Red\nBlue\nPurple\nYellow\n\n";
} // end colors function
```

Display from Sample Code for Exercise 1:

```
    These are my favorite colors:

Red
Blue
Purple
Yellow

Press any key to continue . . .
```

Sample Code for Exercise 2:

```
// This program shows the use of functions for print statements
// Written by Janine Bouyssounouse on 06/13/08

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

using namespace std;

void foods();
void hobbies();

int main(int argc, char *argv[])
{
    foods(); // this calls the foods function
    hobbies(); // this calls the hobbies function
    system("PAUSE");
    return 0;
} // end main

void foods() // text to display favorite foods
{
    cout << "\tThese are my favorite foods:\n\n";
    cout << "Ice Cream\n\tChicken\n\t\tColeslaw\n\t\t\tMango\n\n";
} // end foods function

void hobbies() // text to display hobbies
{
    cout << "\tThese are some of my hobbies:\n\n";
    cout << "\t\tSewing\n\t\tQuilting\n\tEgg Decorating\nPlaying
Cards\n\n";
} // end hobbies function
```

Display of Sample Code for Exercise 2:

These are my favorite foods:

```
Ice Cream
  Chicken
    Coleslaw
      Mango

  These are some of my hobbies:

    Sewing
      Quilting
        Egg Decorating
  Playing Cards

Press any key to continue . . .
```

Sample Code for Exercise 3:

```
// This program shows the use of functions for print statements
// by printing a triangle made of stars (*) on the screen.
// Written by Janine Bouyssounouse on 01/06/09

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

void welcome();
void triangle();

using namespace std;

int main(int argc, char *argv[])
{
  welcome(); // this calls the welcome function
  triangle(); // this calls the triangle function

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

// The welcome function displays the intro text
```

```

// at the beginning of the program
void welcome()
{
    cout << "This program prints a right triangle.\n";
    cout << "The triangle should be five stars (*) high and\n";
    cout << "five stars wide at the base using a print function.\n\n";
} // end welcome function

// The triangle function displays a triangle made of stars (*)
void triangle()
{
    cout << "\n*\n";
    cout << "**\n";
    cout << "***\n";
    cout << "****\n";
    cout << "*****\n";
} // end triangle function

```

Display of Sample Code for Exercise 3:

This program prints a right triangle.
The triangle should be five stars (*) high and
five stars wide at the base using a print function.

```

*
**
***
****
*****

```

Press any key to continue . . .