Creating and Using a Library with Microsoft Visual C++ 2005

Matthew B. Gately

February 27, 2007

Table of Contents

Introduction

This document is designed to explain how to create and use a library using Microsoft Visual Studio 2005. It is intended as an introductory tutorial suitable for the average programmer. This tutorial will first explain libraries, what they are as well as how and why they are used. It will then cover creating a library, and finally implementing it.

For the purposes of this document three functions will be declared for extracting information about divisors.

What Is a Library

A library is simply a file that contains precompiled source code to do something. It has a ".lib" extension, and can be included into any C++ project. A library created on one computer can be used by a programmer working on another.

Why Use a Library

Libraries are used primarily as a matter of convenience and security. They provide code portability while hiding implementation details from the user. This allows a programmer to distribute code for use by others without revealing how it works.

The Two File Strategy

When creating a library, it is best to use a two file strategy. The first is file is the .h file. This should contain all of the function and class definitions, as well as an explanation to the user about how to use them, and that meaning of values that they return. The second is the .cpp file, which should contain the implementation details.

The header file will remain readable to the user and is their only source of information regarding what the library does. The source file will be compiled into the actual library and will not be readable. This approach allows for portable, reusable code that maintains the privacy of the programmers implementation details.

For example, a function may created called "gdc(int value1, int value2)" to return the greatest common divisor between two integers. The function prototype, along with pertinent information regarding the return value (for example, what is returned if no common divisor is found) should be placed in the header file. The actual function will then be placed in the source file.

Creating a Library

The first step in creating a library is starting a new project. To do this, open Microsoft Visual Studio 2005. Next start a New Project by selecting *File* and then *New Project* as shown in Figure 1.

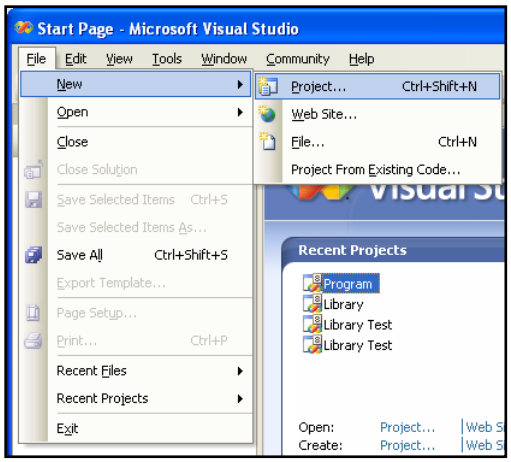

Figure 1: Starting a New Project

Under “Visual C++” select “Win32” and in the right pane select “Win32 Project”. Enter a name and press “Ok”. This is shown in Figure 2.

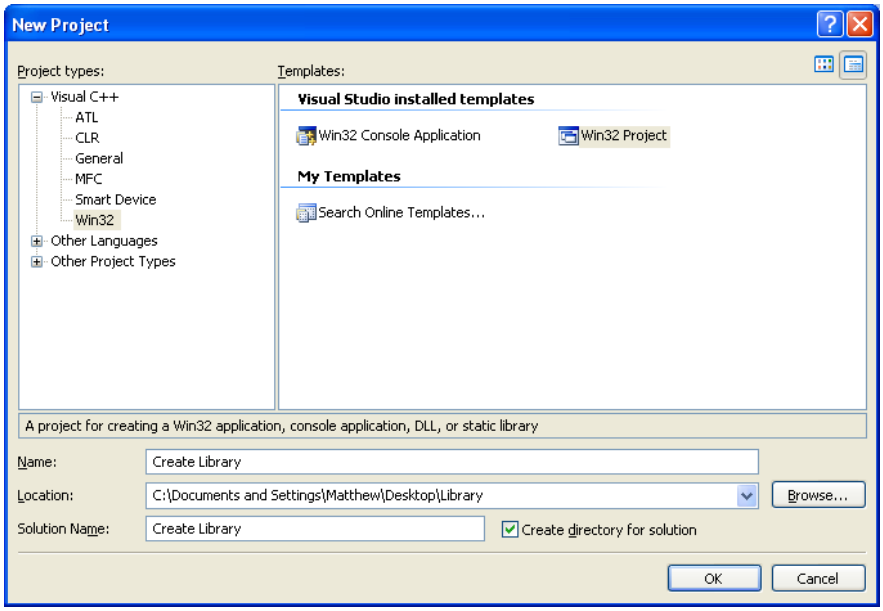

Figure 2: New Project Type

In the application wizard shown in Figure 3, click next.

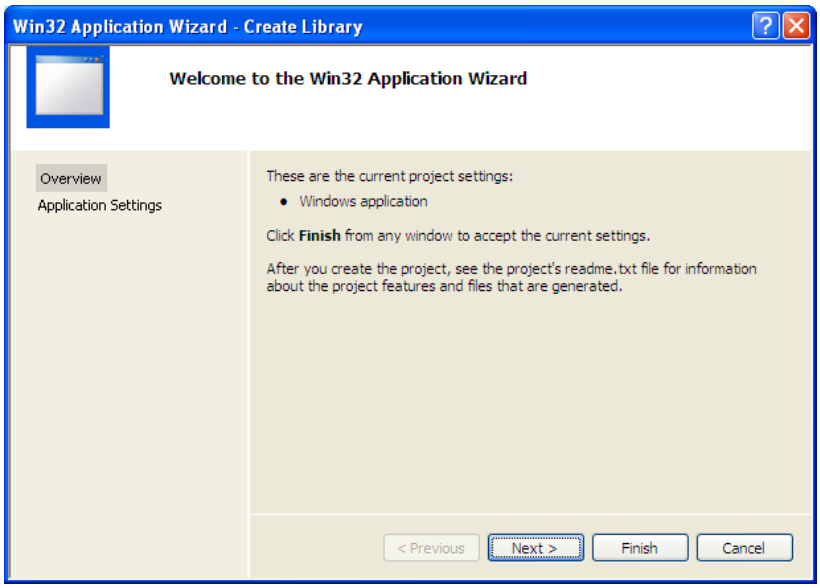

Figure 3: Application Wizard Screen 1

Select the “Static library” radio option, uncheck “Precompiled header” and press “Finish”. The proper settings are show in Figure 4.

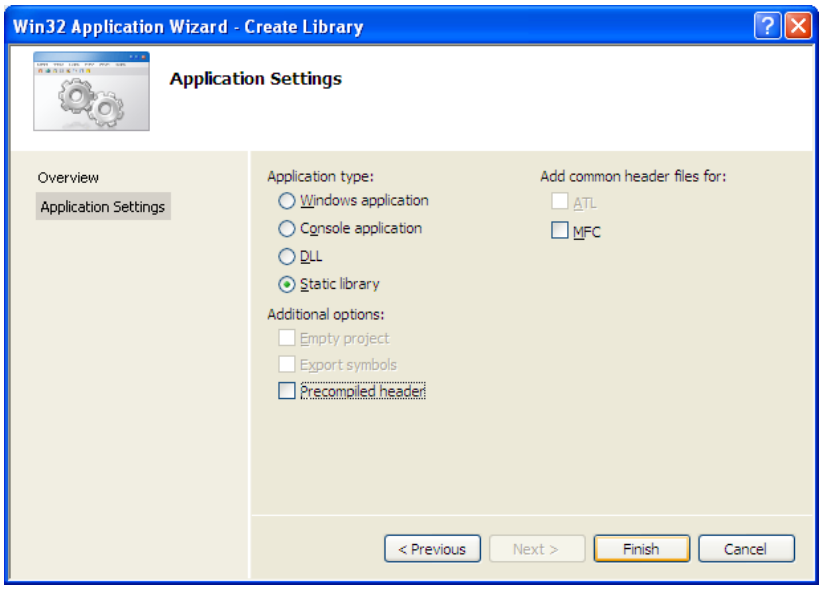

Figure 4: Application Wizard Screen 2

Two new files must now be added to the project. A header file for the definitions, and a source file for the implementation. Be sure to use the same name for both the header and for the source file. Adding a new file is shown in Figure 5 and in Figure 6.

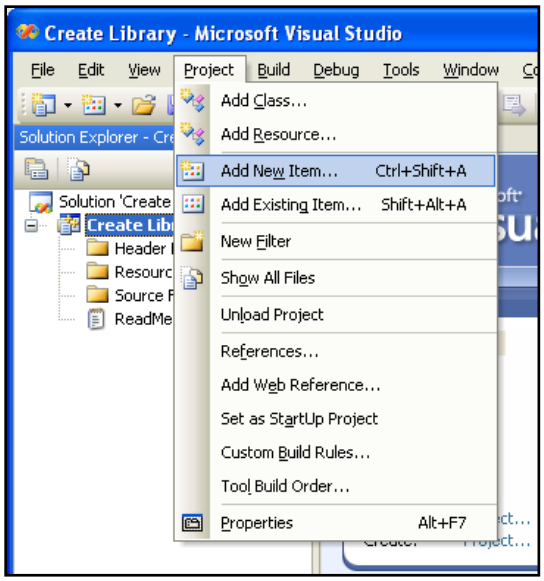

Figure 5: Adding a New Item

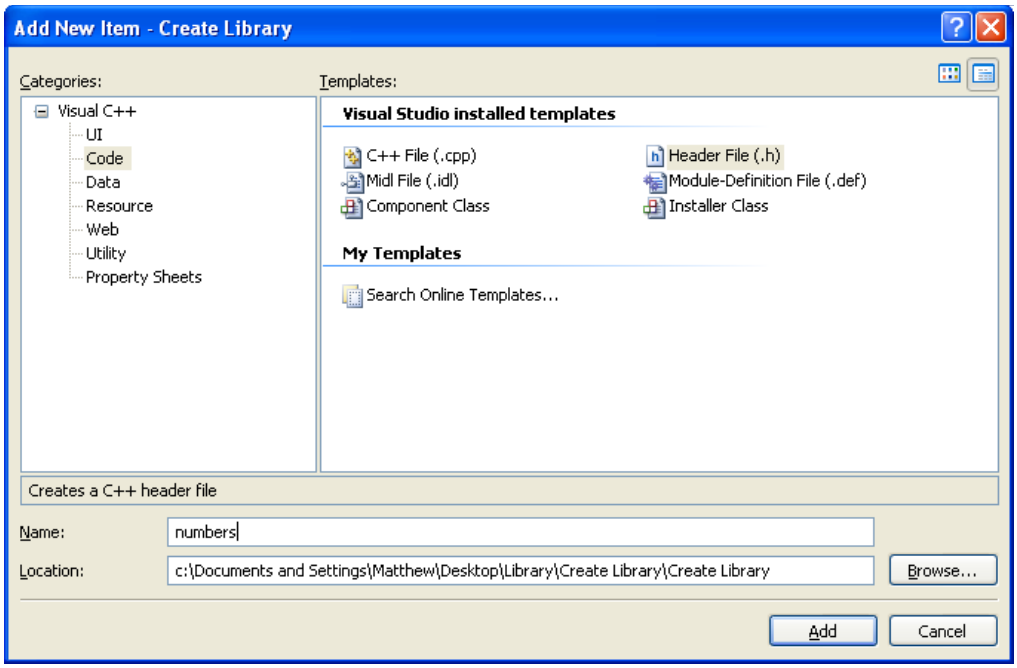

Figure 6: Add New Item

The proper source code will need to be added to the two files. The sample code used in this document is included in the appendix. From the *Build* menu, select *Build* "Your Project Name" This is shown in Figure 7.

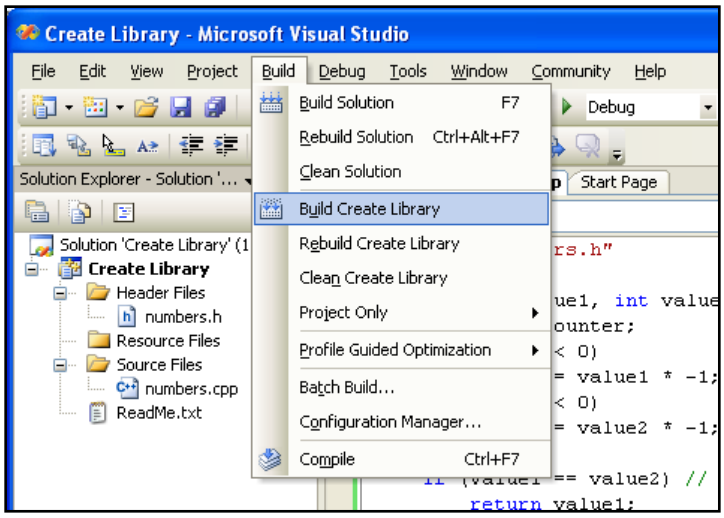

Figure 7: Build Files

When the program compiles successfully, open the containing folder. It will be necessary to copy two files to save for later use. They are the compiled library itself, as well as the header file. The compiled library is stored in the "DEBUG" folder and is shown in Figure 8. The header file containing the definitions is also needed and can be found in the main folder. The library has now been successfully created.

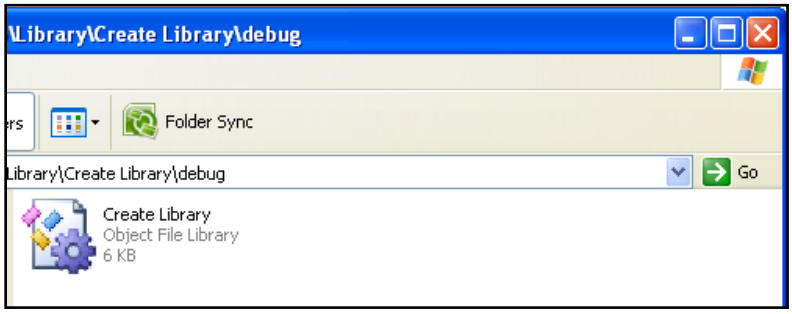

Figure 8: Library File

Using a Library

The first step in using a library is to create a project that will need the library. To create a new project, open Microsoft Visual Studio 2005. Next, Start a new Project by selecting *File* and then *New Project* as shown previously in Figure 1. Under "Visual C++" select "Win32" and in the right pane select "Win32 Project". Enter a name and press "Ok". This is shown in Figure 9.

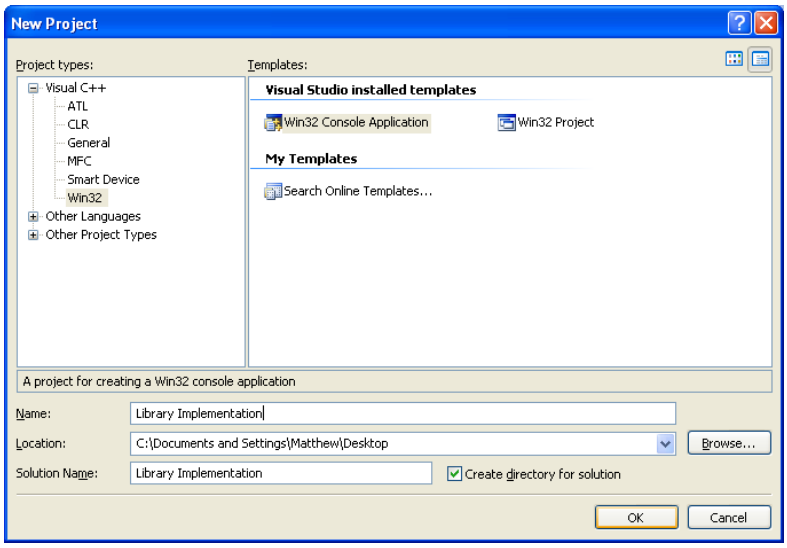

Figure 9: Creating a New Win32 Console Application

In the application wizard click “Next” as shown in Figure 10.

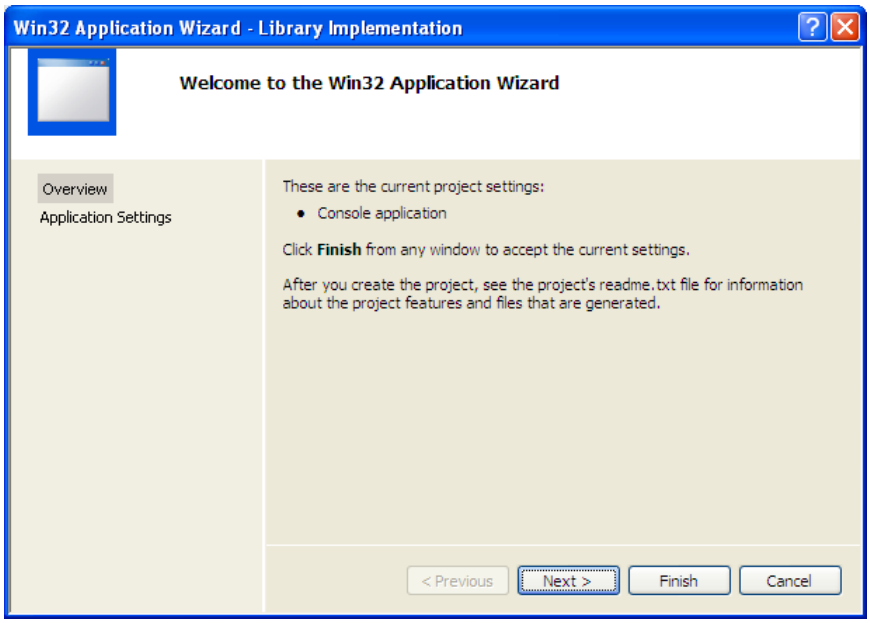

Figure 10: Application Wizard Screen 1

Select the “Console application” radio option, check “Empty project” and press “Finish”. The proper settings are shown in Figure 11.

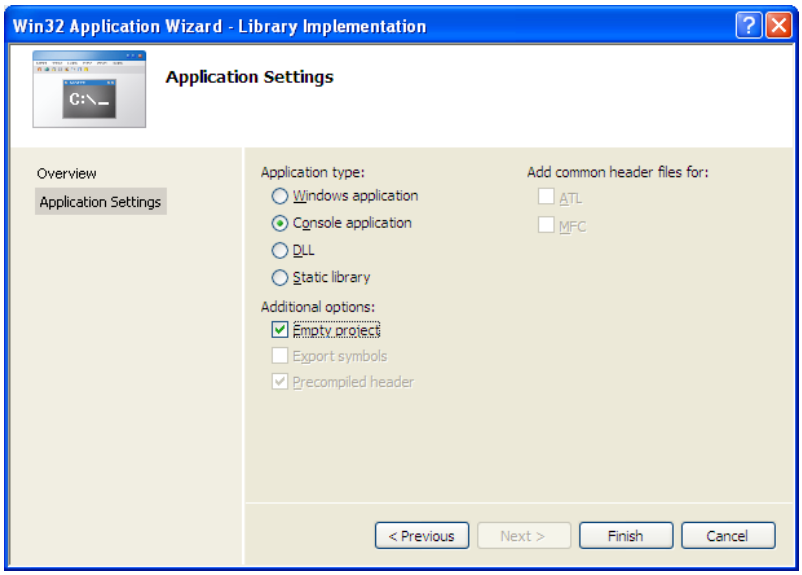

Figure 11: Application Wizard Screen 2

Now a new source file must be added to the document. This was shown previously in Figure 5 and Figure 6. This is the file that contains the code that will use functions or classes in the library. It will now be necessary to copy the existing header file and the library file into the project folder (where the new .cpp file is located). This is shown in Figure 12.

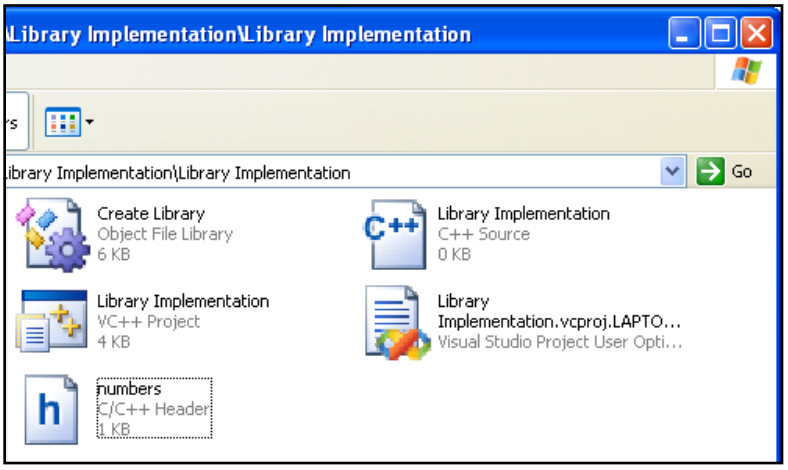

Figure 12: Copied Files

Now that the library has been physically added to the project folder, it is necessary to add it to the project using Microsoft Visual Studio 2005 so that it will be linked to the rest of the program during the compile. First add the line at the top of the .cpp file to include the header such as *#include "numbers.h"*. Next select "Project", then "Add Existing Item" as shown in Figure 13.

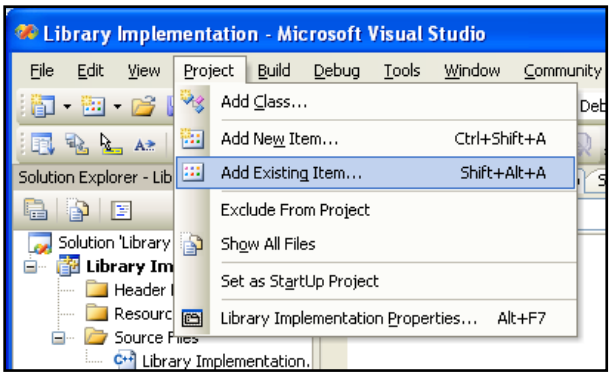

Figure 13: Add Existing Item

From the drop down menu “Files of type” select “All Files”. Select the library and click “Add”. This is shown in Figure 14. If a message appears that there is no appropriate build rule, ignore it and compile the program. By having added the include statement, it will compile correctly.

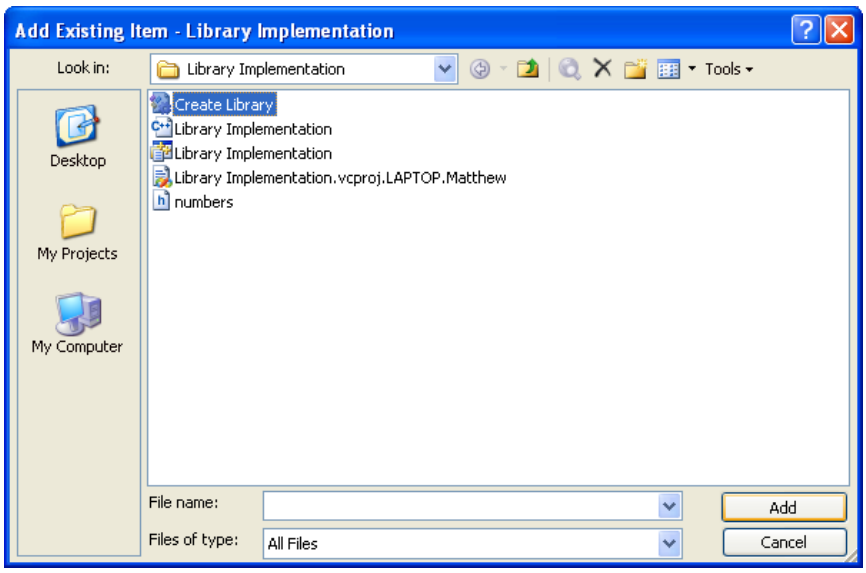

Figure 14: Adding an existing library

The appropriate source code can be added and can use the functions defined in the header file. A copy of the source code used for this document can be found in the appendix. There are no additional steps other than compiling the program. The output of the program is shown in Figure 15.

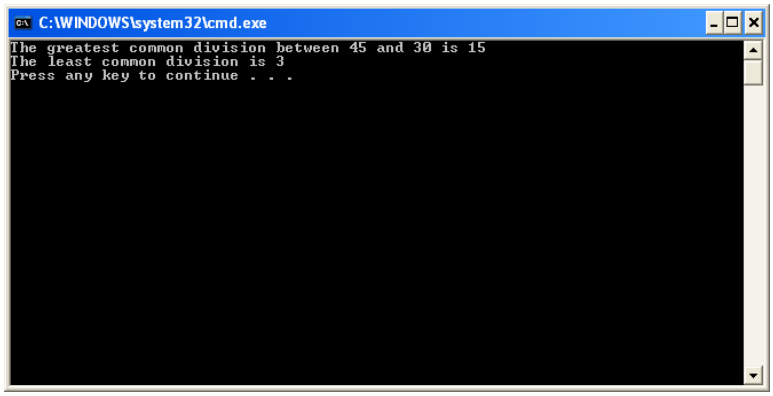

Figure 15: Output

Appendix

Source Code

Library Header File

```
#ifndef _numbers_h_
#define _numbers_h_

// Returns the largest common divisor between two numbers
// Numbers are treated as positive, -1 is returned if there is an error
int gdc(int value1, int value2);

// Returns the lowest common division between two numbers other than 1
// Returns a -1 if there is no lowest common divisor other than 1
// Numbers are treated as positive
int ldc(int value1, int value2);

// Returns a 1 if value1 is a factor of value2, otherwise returns 0
int is_factor(int value1, int value2);

#endif
```

Library Implementation File

```
#include "numbers.h"

int gdc(int value1, int value2) {
	int temp, counter;
	if (value1 < 0)
		value1 = value1 * -1;
	if (value2 < 0)
		value2 = value2 * -1;

	if (value1 == value2) // See if values are the same
		return value1;

	if (value1 < value2)
		temp = value1 / 2;
	else
		temp = value2 / 2;

	for (counter = temp; counter > 0; counter--) {
		if (value1 % counter == 0 && value2 % counter == 0)
			return counter;
	}

	return -1;
}

int ldc(int value1, int value2) {
	int counter, max;
	if (value1 < value2)
		max = value1 / 2;
	else
		max = value2 / 2;

	for (counter = 2; counter <= max; counter++)
		if (value1 % counter == 0 && value2 % counter ==0)
			return counter;

	return -1;
}

int is_factor(int value1, int value2) {
	if (value2 % value1 == 0)
		return 1;
	else
```

```
		return 0;
}
```

Project Source File

```
#include "numbers.h"
#include <iostream>

using namespace std;

void main(void) {
	cout << "The greatest common division between 45 and 30 is " << gdc(45,
30);
	cout << endl << "The least common division is " << ldc(45, 30) << endl;
}
```