CSC215 Programming Techniques – Due February 23 at 4:00pm
Programming Assignment #1: The Disgruntled Employee

Problem:

A company recently fired an employee and before his access could be revoked, he was able to change the images files that are needed for the company's documents, forms and web pages. Although this employee did not delete the images, he did enough damage to cripple the small company. This employee found all the images that the company used and removed the extensions and on specific image types also removed the width and height that are needed for proper web site layout. Your job is to repair all the files restoring them to their prior name.

Data:

The data shall come from a directory located somewhere on a hard disk. The directories to be processed will be provided via the command prompt. You will need to process all files within each directory. You do not need to do a subdirectory within a directory. If an image is found, you will need to rename it appropriately. If a file is not an image, you are not to rename it. The only image files that we will be concerned with renaming are bmp, gif, png, and jpg.

Program Execution:

To start your program, at least one command line argument is needed. This argument is the directory that contains the files to be processed. If the directory does not exist or is unable to be opened, output an error message and process the next directory provided.

Ex. **C:\> prog1.exe c:\dir dir1 c:\dir2\subdir ..\..\images**

Output:

While processing each file within the directory, if you determine that the file is a gif, png or bmp you will extract the width and height of the image and rename the file using the original filename, the width, the height and the extension to form a new filename.

Ex. You determine that the file name frog is a gif image with a width of 50 and a height of 75, you will rename the file to frog_50x75.gif

If you determine that the file is a jpeg file you will not need to extract the width and height of the image but you will need to rename the file adding on the extension of jpg.

Ex. You determine that the file name cat is a jpg image. You will need to rename the file to cat.jpg.

Errors:

If the directory to be processed can not be changed to or does not exist, an error message should come to the screen and your program should process the next directory. If your program is unable to open a file, just continue processing the directory without specifying an error message. If your program is unable to rename a file, output the original files name, an arrow, and then what the new file should be.

Ex.   Unable to rename: frog ==> frog_50x75.gif

Ex.   Unable to process the directory ..\..\..\webimages ⬅ this is stored in argv

Changing to a Directory:

Please see the file system document on web site.

Reading a Directory:

Please see the file system document on web site.

Determining if it is a Directory or a File:

As you get a directory listing, you are to only process files. Do not attempt to open any directories. You can use the attrib field which is a part of the _finddata_t structure. This structure is already defined for you. If the 4 bit is iset, it is a directory. Do not just see if the field is 16 because other fields may be set. You must use bitwise operators to see if it is a directory. If a possible file has the 1st and the 4th bit set, it is a hidden directory, and does not contain the value 16. Just use the constant (_A_SUBDIR) and test for that bit being set.

The following table shows which bit within the attrib that indicates it is a directory. If the bit is a zero, It is just a file.

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Important: Do not try and process any name that is "." Or "..". these are special directories. That is a single period or two periods.

Detecting the image types, and the width and height:

To detect if a file is an image of type gif, png, bmp, or jpg, you will need to open the file in binary mode, seek to a given location, read in certain bytes and compare them to a specified sequence. Each image file has its own format. To extract the information you will need to use the read function rather than the >> operator or the getline function that we normally use for input. For example if you wish to read in 1 byte from a file into the character data you would execute the following line file.read((char *) &data, 1);. The prototype for the read function is as follows:

Prototype: **ifstreamName.read((char *) , number_of_bytes);**

All reads must be into a character type and bitwise operators must be used to compute the sizes of the images. Doing it this way makes your code work on any system without needing to know if the architecture is big endian or little endian. If you do not form the image sizes using bitwise operators, you will lose 15 points.

Gif **Image**

If the file is a gif image, the first 6 bytes of the file will contain either "GIF89a" or "GIF87a". if the 6 bytes are a match, then you can extract the width and the height of the image.

The width can be found in bytes 7 and 8. Byte 7 is low order byte of a short integer and byte 8 is the high order byte of a short integer.

The height can be found in bytes 9 and 10. byte 9 is the low order byte of a short integer and byte 10 is the high order byte of a short integer.

						W1	W2	H1	H2
						Low	High	Low	High

Short integer	High byte	Low byte

Bmp Image

If the file is a bmp image, the first 2 bytes will contain “BM”. If the first two bytes are a match, then you can extract the width and the height of the image as follows:

The width can be found in bytes 19, 20, 21 and 22. Byte 19 is the low order byte, followed by 20, then 21 and byte 22, which is the high order byte.

The height can be found in bytes 23, 24, 25 and 26. Byte 23 is the low order byte, followed by 24, then 25 and byte 26, which is the high order byte.

Byte1	Byte2	Byte3	Byte4
19	20	21	22

Integer	Byte4	Byte3	Byte2	Byte1

Png Image

If the file is a png image, the first 8 bytes of the file will contain the values (137, 80, 78, 71, 13, 10, 26, 10). You can get these into an array by using an initalizer list and providing these values. Example `datatype array[8] = { 137, 80, 78, 71, 13, 10, 26, 10 };`
If these first 8 characters match, you can extract the width and height of the image as follows.

The width can be found in bytes 17, 18, 19, and 20. Byte 20 is the low order byte, followed by 19, 18, and 17 which is the high order byte.

The height can be found in bytes 21, 22, 23, and 24. Byte 24 is the low order byte, followed by 23, 22, and 21 which is the high order byte.

Byte1	Byte2	Byte3	Byte4
17	18	19	20

Integer	Byte1	Byte2	Byte3	Byte4

Jpg Image

If the file is a jpg file, the first two bytes will contain the values (255, 216) and the last two bytes of the file will contain the values (255, 217). You will not need to extract the width and height of the image since it is not needed to rename the file. They do exist, but jpg is a complex image and can contain many images inside the file. Your camera usually stores the image you just took and a thumbnail version inside it that is used for quick displaying in the camera viewer.

Renaming the file:

To rename a file from one name to another, you will need to use the function rename. This function is provided by including io.h in your program. The function will rename a file from oldname to newname and return a zero if successful. If unable to rename a file, output an error message indicating the old filename and the new file name. Then continue on processing the directory.

Prototype: `int rename ( char *oldname, char *newname );`

Example:

```
char oldname[80] = "frog";
char newname[80] = "frog_50x75.gif";

rename(oldname, newname);
```

Algorithm:

1. Check command line arguments.
2. Check that the specified directory exists and change to that directory
    a. Read in first filename
    b. Check if it is an image file
    c. If it is an image file of gif, bmp, or png extract width and height
    d. Form a new name if necessary.
    e. Rename file if necessary.
    f. Read in next filename and repeat steps a through e until all files in this directory have been tested.
3. Return to the initial directory and repeat step 2 for all directories given on the command line.

Specifications:

1. No classes allowed.
2. All files, do not open directories, are to be opened in binary mode.
3. Project name: prog1
4. File names: prog1.cpp, imageClassifier.cpp, imageDimensions.cpp, utilities.cpp, imageClassifier.h, imageDimensions.h, and utilities.h
5. The width and height are to be computed using bitwise operators and each byte is to be read into a character byte.
6. The reading of data from files should be into characters. To form the integers, you must read each byte needed into an **character** type and use bitwise operators to form the integer.
7. **You may not use the itoa to convert your number to a string, You must write your own function that is named intToString that converts an integer to a string or use a function that is within the string library. Use cplusplus.com to view these functions.**
8. You must us the seek command to move around within the file to extract data. I will not allow you to read in the data using a large block that contains all the data you need. Part of the purpose of this assignment is teaching you to seek around the file.
9. You are to use your Gitlab project labeled **csc215s18programs** and create a project name prog1 within this repository. This is just like creating a homework project.

Time Line:

February 7, 2016	-- Be able to detect command line arguments, validate directories, and output all the files (only the files) to be processed in a directory.
February 12, 2016	-- be able to detect which files are image files and which ones are not.
February 17, 2016	-- Be able to extract the width and height from bmp, gif, and png's and form an integer value.
February 19, 2016	-- Be able to rename the image files correctly.
February 21, 2016	-- Be able to process all directories given on the command line.
February 23, 2016	-- Have tested your program and documented it with doxygen

Hints:

- Work on the program in small sections, never try and write the entire program at once.
- Save and compile often. Remove syntax errors as you discover them.
- Don't be afraid to write functions. No function should be over 50 lines long.
- Do not wait until the last minute to program the assignment. Work on the program early and only do small time frames. Follow the timeline. Use git commits to show your progress.
- When you encounter a logical error, don't spend hours trying to find it. Seek help from your instructor.

Other:

Adhere to the programming guidelines and coding standards
No functions should be over 50 lines long.
I don't debug programs on the day they are due.
Make sure and commit your program to the repository.

I will be checking Gitlab periodically to see that you are on track. Remember, only commit code to the Gitlab repository that compiles and works correctly. Points will be deducted if you have not been working on the project on a regular basis. This is to keep you on track so you do not wait until the last minute.

File Setup:

Prog1.cpp

Contains only the main function with the doxygen program header and function header for main.

imageClassifier.cpp & imageClassifier.h

Contains the functions that test if a file is a png, bmp, gif or jpg. These functions should not call any other functions outside this file.

Prototypes go in the header file and the functions are written in the cpp file. (required)

Prototypes
```
bool isPNG( ifstream &fin );
bool isBMP( ifstream &fin );
bool isGIF( ifstream &fin );
bool isJPG( ifstream &fin );
```

imageDimension.cpp & imageDimension.h

Contains the functions that will extract the width and height of the image. Consider writing one function for each of the three images that require width and height. Put the prototypes in the header and the functions in the cpp These functions should not call any other functions outside this file.

Utilities.cpp and utilities.h

Contains all other functions outside of main. These functions may call any function needed to complete its task. Put the prototypes in the header and the functions in the cpp.

Since this can call the image classifier and the image dimension functions, you will need to include the 2 headerfiles in utilities.h

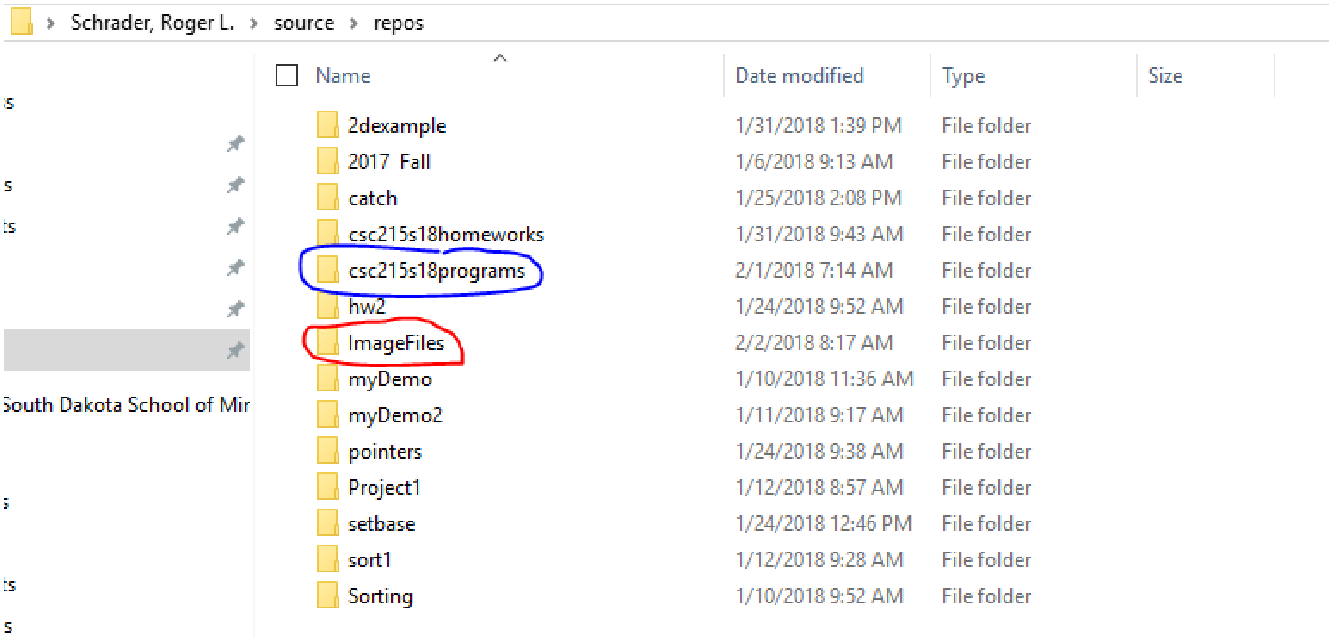
Remember, in the header files to insert the appropriate #ifndef __FILE__H__ statements just like your homeworks and put an #endif after the prototypes.

Modify the .gitignore

Just like we did for the homeworks, modify the .gitignore file to exclude the html directory.
Scroll to the bottom.

Add the comment line	# csc215 custom ignores
Exclude the html directory	html/

DO NOT ADD ANY IMAGES TO THE PROJECT DIRECTORY. PUT YOUR TESTING DATA OUTSIDE THE REPOSITORY.

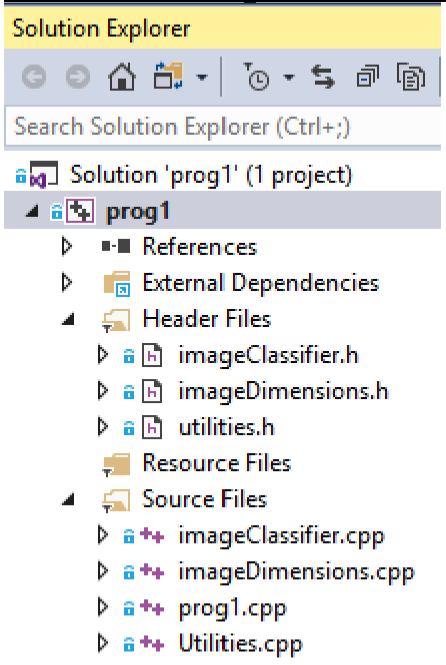

Schrader, Roger L. › source › repos

South Dakota School of Mir

Name	Date modified	Type	Size
2dexample	1/31/2018 1:39 PM	File folder	
2017 Fall	1/6/2018 9:13 AM	File folder	
catch	1/25/2018 2:08 PM	File folder	
csc215s18homeworks	1/31/2018 9:43 AM	File folder	
csc215s18programs	2/1/2018 7:14 AM	File folder	
hw2	1/24/2018 9:52 AM	File folder	
ImageFiles	2/2/2018 8:17 AM	File folder	
myDemo	1/10/2018 11:36 AM	File folder	
myDemo2	1/11/2018 9:17 AM	File folder	
pointers	1/24/2018 9:38 AM	File folder	
Project1	1/12/2018 8:57 AM	File folder	
setbase	1/24/2018 12:46 PM	File folder	
sort1	1/12/2018 9:28 AM	File folder	
Sorting	1/10/2018 9:52 AM	File folder	

The blue is the repo, the red is a directory with test files.

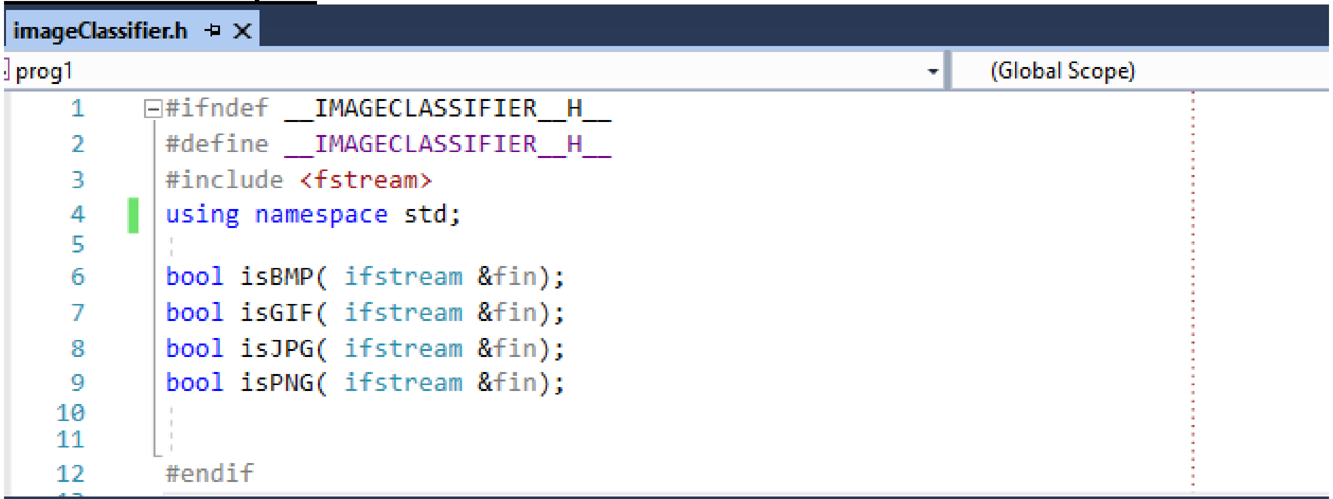
Solution Explorer View:


Solution Explorer

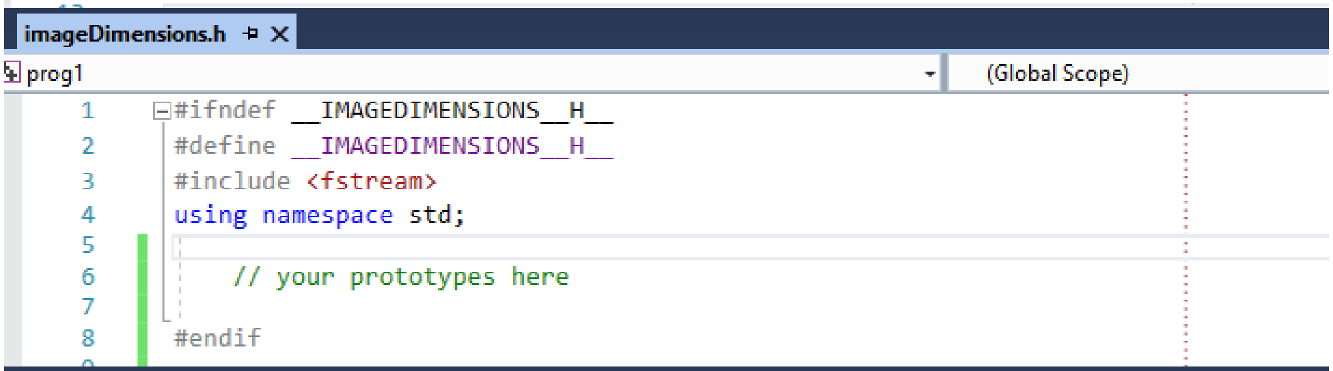
Search Solution Explorer (Ctrl+;)

```
Solution 'prog1' (1 project)
  prog1
    References
    External Dependencies
    Header Files
      imageClassifier.h
      imageDimensions.h
      utilities.h
    Resource Files
    Source Files
      imageClassifier.cpp
      imageDimensions.cpp
      prog1.cpp
      Utilities.cpp
```

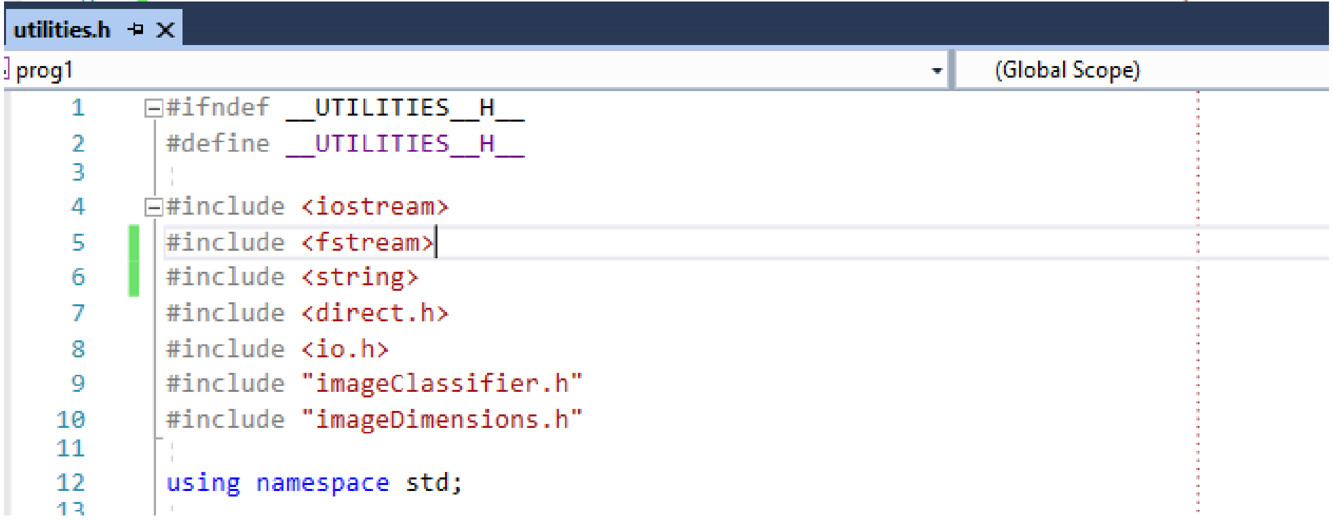
Header File Examples:



```
#ifndef __IMAGECLASSIFIER__H__
#define __IMAGECLASSIFIER__H__
#include <fstream>
using namespace std;

bool isBMP( ifstream &fin);
bool isGIF( ifstream &fin);
bool isJPG( ifstream &fin);
bool isPNG( ifstream &fin);


#endif
```

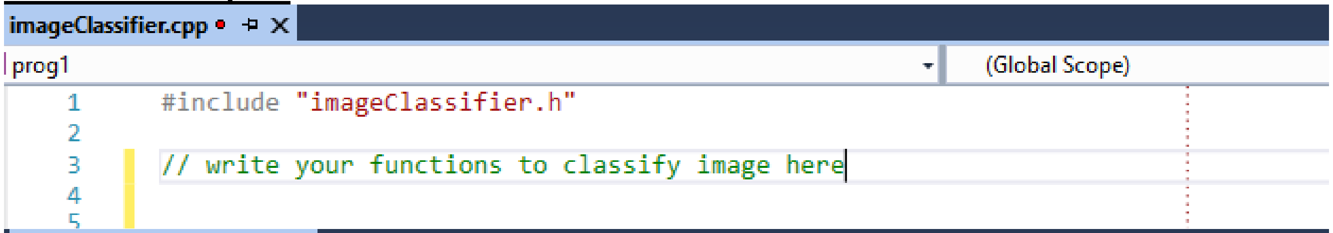


```
#ifndef __IMAGEDIMENSIONS__H__
#define __IMAGEDIMENSIONS__H__
#include <fstream>
using namespace std;

    // your prototypes here

#endif
```

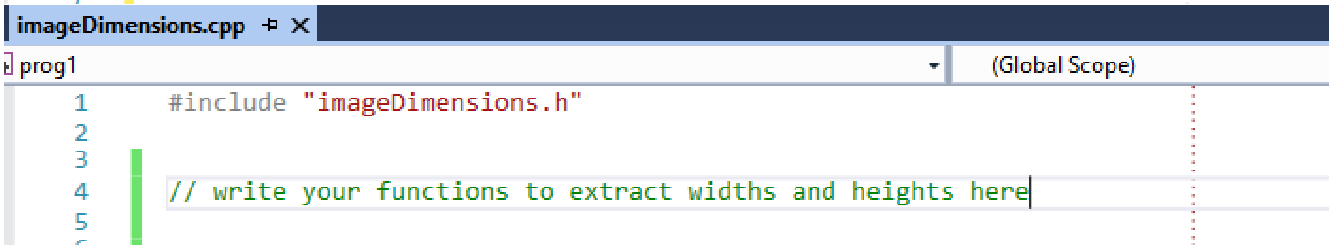


```
#ifndef __UTILITIES__H__
#define __UTILITIES__H__

#include <iostream>
#include <fstream>
#include <string>
#include <direct.h>
#include <io.h>
#include "imageClassifier.h"
#include "imageDimensions.h"

using namespace std;
```

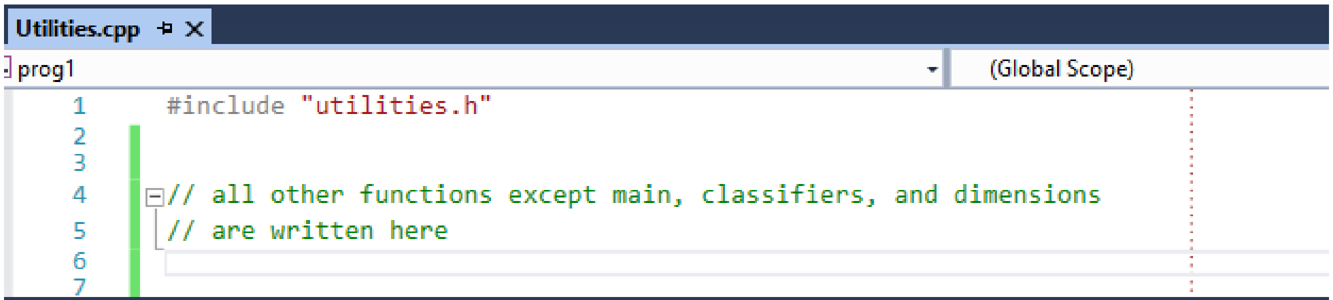
CPP Files Examples:



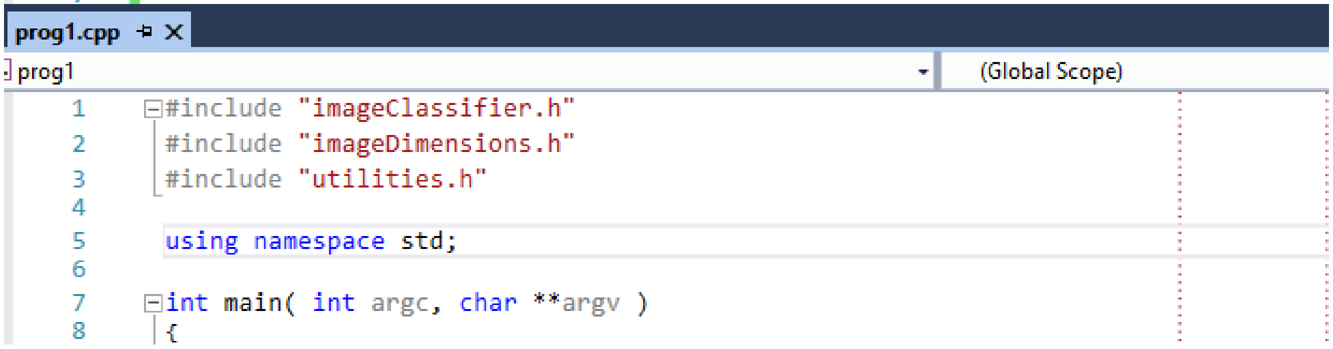
```
#include "imageClassifier.h"

// write your functions to classify image here


```



```
#include "imageDimensions.h"


// write your functions to extract widths and heights here

```



```
#include "utilities.h"


// all other functions except main, classifiers, and dimensions
// are written here


```



```
#include "imageClassifier.h"
#include "imageDimensions.h"
#include "utilities.h"

using namespace std;

int main( int argc, char **argv )
{
```

Your files may look different. You may include other libraries that I didn’t need or use in my program. A good rule of thumb is: only include what is necessary to compile the file the header was designed for.
Example: imageClassifier.h only includes the fstream library because that is the only thing that is needed for the functions to classify the images.