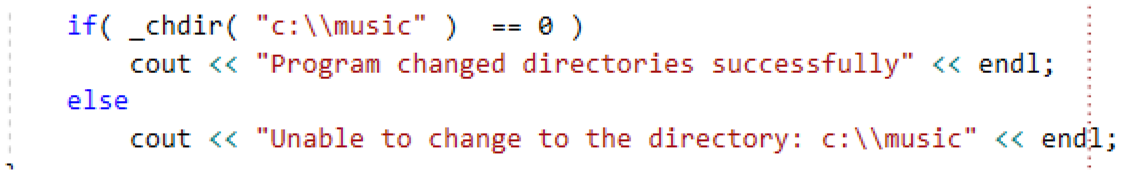
Your current directory:

Two functions that you will need are the _getcwd and the _chdir. Both of these functions have already been written for you.
Prototypes for _getcwd and _chdir, they are provided in <direct.h>

```
int _chdir( char *dirpath );
char *_getcwd(char *cstr, int size);
```

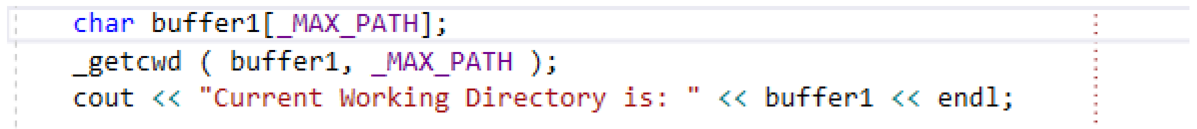
_chdir function – you will pass a c style string that contains the directory you wish to change to. If it is successful at changing to this directory, it will return a 0, any other value is an error and you should output a message to the listing.html file



```
    if( _chdir( "c:\\music" )  == 0 )
        cout << "Program changed directories successfully" << endl;
    else
        cout << "Unable to change to the directory: c:\\music" << endl;
```

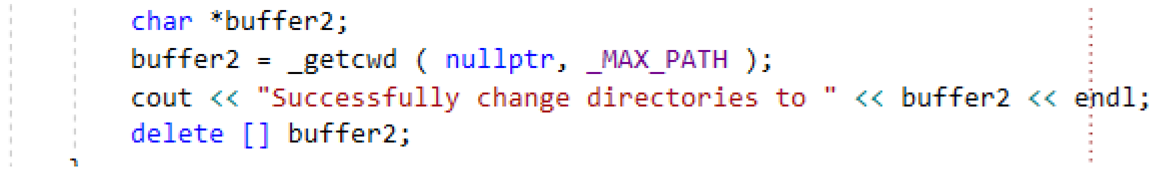
_getcwd function – will return the pointer to a c style array that contains the name of the directory you are in. This c style array is dynamically allocated and must be freed by you. Be smart, for each directory you process, call the function only once. By passing NULL and zero forces it to due dynamic memory allocation. If you pass in a char array, the size of the array must be passed and should be of size _MAX_PATH. It is safer to use the dynamic method.

Method 1:



```
    char buffer1[_MAX_PATH];
    _getcwd ( buffer1, _MAX_PATH );
    cout << "Current Working Directory is: " << buffer1 << endl;
```

Method 2:

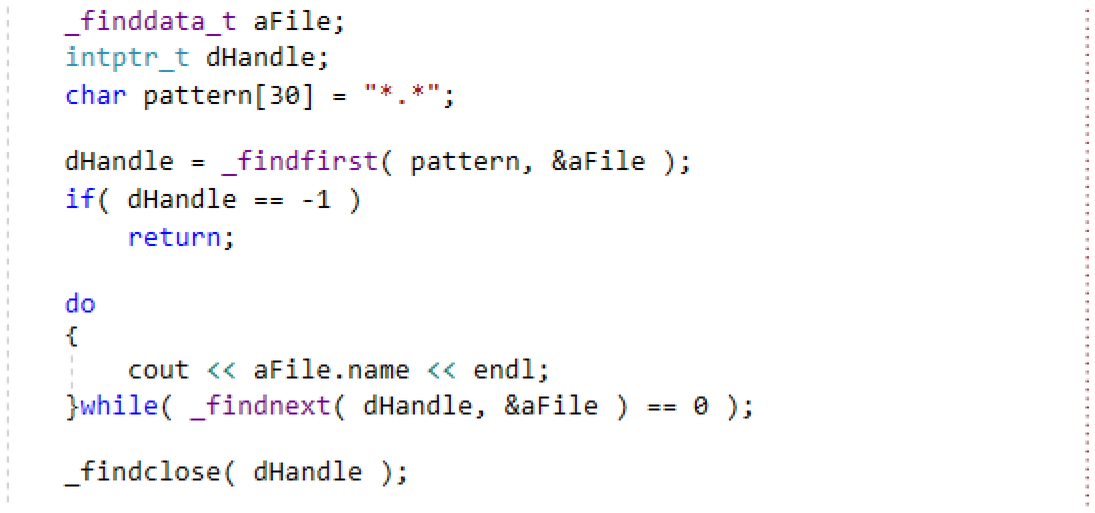


```
        char *buffer2;
        buffer2 = _getcwd ( nullptr, _MAX_PATH );
        cout << "Successfully change directories to " << buffer2 << endl;
        delete [] buffer2;
```

Getting a directory listing:

To get a directory listing, you will need to include two files that MS Visual Studio specific.
#include <direct.h>
#include <io.h>

The following code snippet will return a complete listing of all files and folders within a given directory.



```
_finddata_t aFile;
intptr_t dHandle;
char pattern[30] = "*.*";

dHandle = _findfirst( pattern, &aFile );
if( dHandle == -1 )
    return;

do
{
    cout << aFile.name << endl;
}while( _findnext( dHandle, &aFile ) == 0 );

_findclose( dHandle );
```

_finddata_t and intptr_t are system defined data types. You do not need to create them. Just use them. The _findfirst function's first parameter is the listing of files you wish to receive. * is a wildcard and says give my anything in that location. So *.* give you everything. *.jpg would give you a directory listing of just image files with the extension of .jpg.   M*.jpg would give you a directory list of all image files that start with a 'M' and have an extension of .jpg. ? is another wild card that matches any character in that spot. If I specified a pattern of Mom?.jpg, it would return a directory listing of all files that started with Mom and a single character following it with the extension of .jpg. It must have another character in that spot.   Mom.jpg would not match(no character for ?) and mom10.jpg also would not match (2 characters instead of 1), but mom1.jpg mom2.jpg and mom3.jpg are all possible matches. The pattern is a C string.

Structure _finddata_t

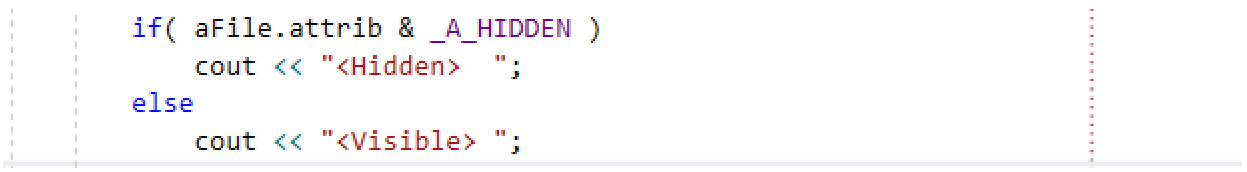
_finddata_t is a structure for a file / folder. From it you can access the fields (name, attrib, size, time_create, time_access, time_write).

The **name** field is a C style character array of _MAX_PATH characters. _MAX_PATH is a system defined variable for the maximum number of characters any file path can contain. In Windows XP and Vista it is defined to be 256. In windows 10 using vs2017 this is defined as 260.

The **size** file is an unsigned integer size_t (4 bytes) that provides the number of byes in the file.
The **time_create** is an unsigned integer time_t (8 bytes) that represents the data and time the file was create.
The **time_access** is an unsigned integer time_t (8 bytes) that represents the data and time the file was create.
The **time_write** is an unsigned integer time_t (8 bytes) that represents the data and time the file was create.
The **attrib** is an unsigned integer (4 bytes) that provides information about the file or folder.

A list of system attribute constants:
_A_ARCH - Archive. Set whenever the file is changed and cleared by the **BACKUP** command. Value: 0x20.
_A_HIDDEN - Hidden file. Not normally seen with the DIR command, unless the **/AH** option is used. Returns information about normal files and files with this attribute. Value: 0x02.
_A_NORMAL - Normal. File has no other attributes set and can be read or written to without restriction. Value: 0x00.
_A_RDONLY - Read-only. File cannot be opened for writing and a file with the same name cannot be created. Value: 0x01.
_A_SUBDIR - Subdirectory. Value: 0x10.
_A_SYSTEM - System file. Not normally seen with the **DIR** command, unless the **/A** or **/A:S** option is used. Value: 0x04.

For the directory listing, you can do a bitwise and with any of the attribute constants to find out information. For instance, if you were interested in finding out if a file/folder was hidden, you would use the _A_HIDDEN attribute constant like so.



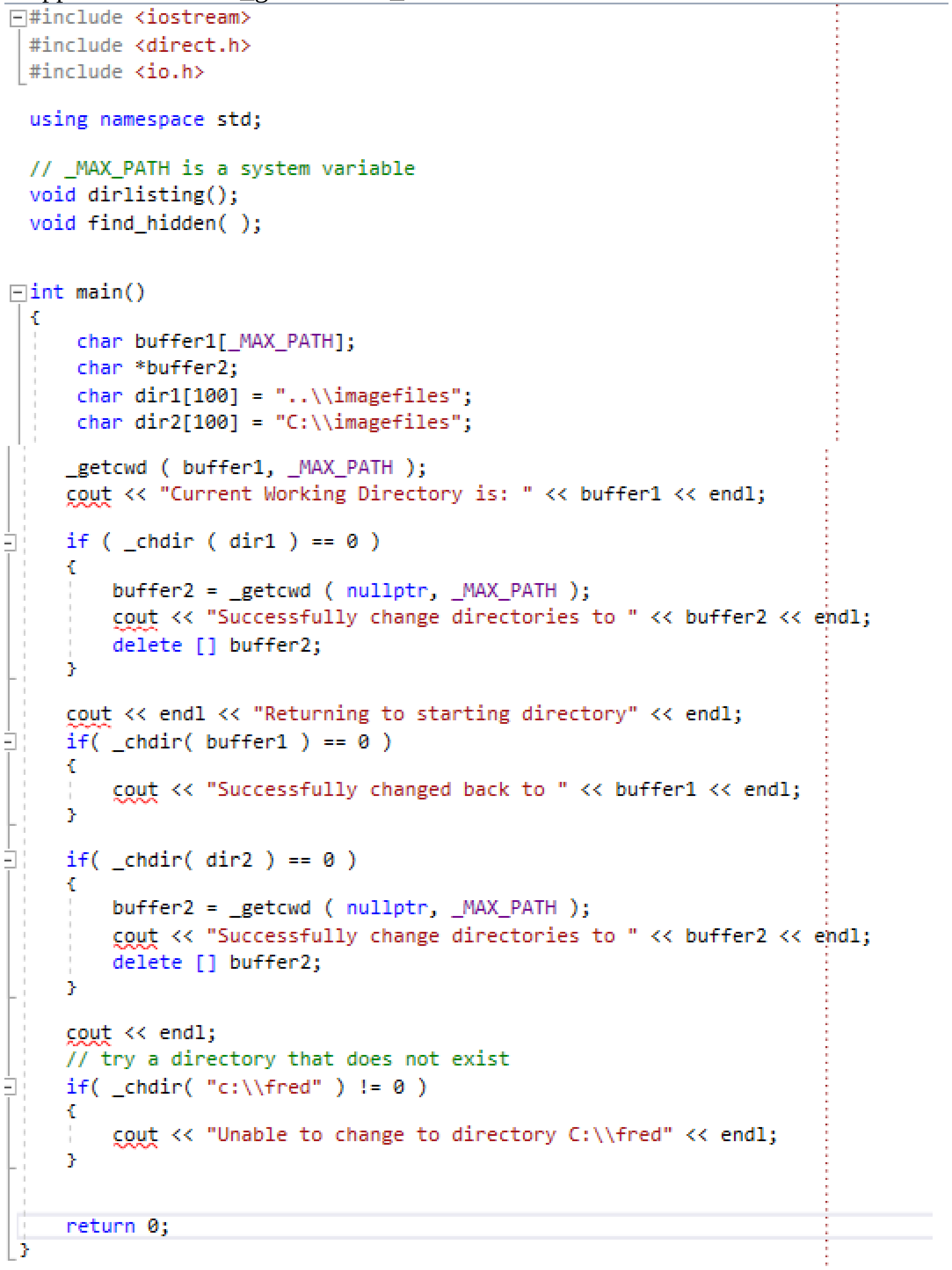
```
if( aFile.attrib & _A_HIDDEN )
    cout << "<Hidden>  ";
else
    cout << "<Visible> ";
```

To find out if it is a directory within the directory being processed, you would use _A_SUBDIR.

If the result is a number, it is a directory otherwise it is a file.

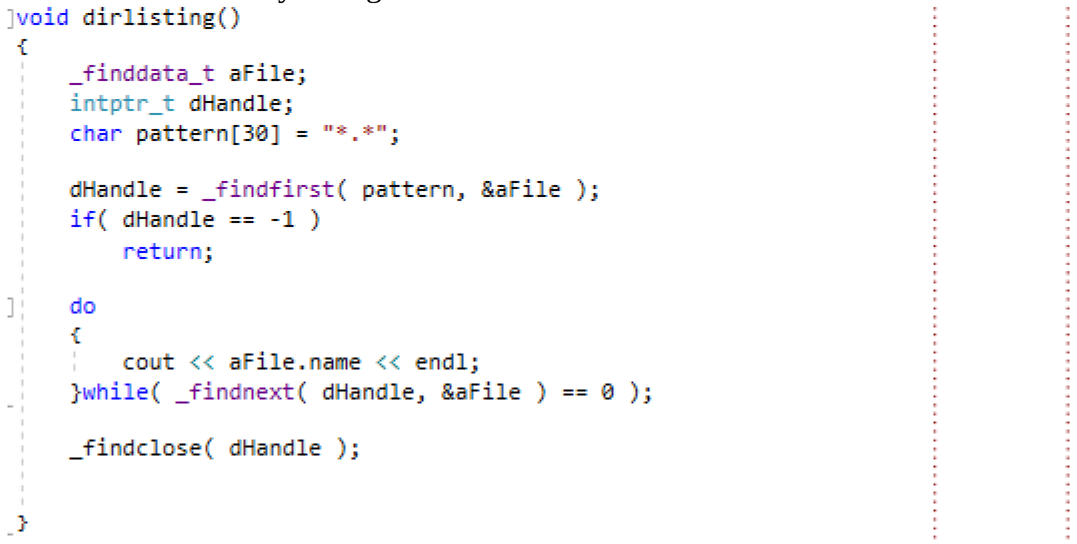
Notes: _A_Normal does not define that it is a file. Use the logic of “if it is not a folder, it is a file”

Snipped to show the _getcwd and _chdir



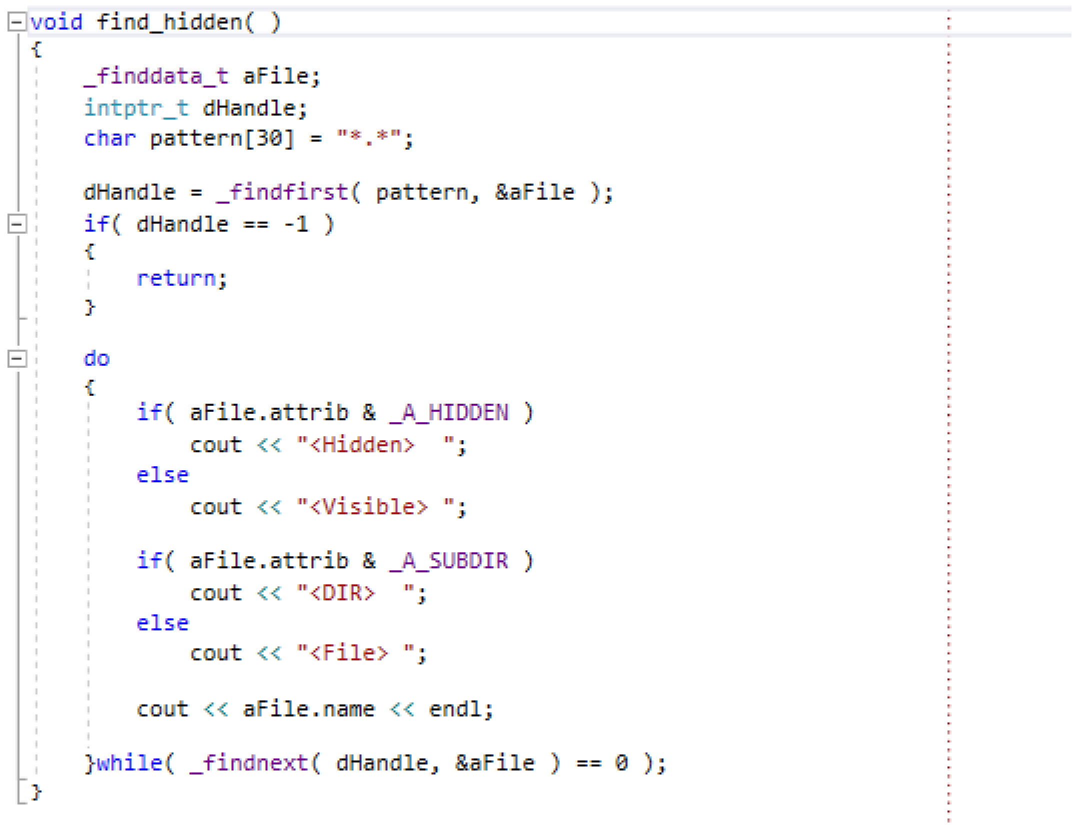
```
#include <iostream>
#include <direct.h>
#include <io.h>

using namespace std;

// _MAX_PATH is a system variable
void dirlisting();
void find_hidden( );


int main()
{
    char buffer1[_MAX_PATH];
    char *buffer2;
    char dir1[100] = "..\\imagefiles";
    char dir2[100] = "C:\\imagefiles";

   _getcwd ( buffer1, _MAX_PATH );
   cout << "Current Working Directory is: " << buffer1 << endl;

   if ( _chdir ( dir1 ) == 0 )
   {
       buffer2 = _getcwd ( nullptr, _MAX_PATH );
       cout << "Successfully change directories to " << buffer2 << endl;
       delete [] buffer2;
   }

   cout << endl << "Returning to starting directory" << endl;
   if( _chdir( buffer1 ) == 0 )
   {
       cout << "Successfully changed back to " << buffer1 << endl;
   }

   if( _chdir( dir2 ) == 0 )
   {
       buffer2 = _getcwd ( nullptr, _MAX_PATH );
       cout << "Successfully change directories to " << buffer2 << endl;
       delete [] buffer2;
   }

   cout << endl;
   // try a directory that does not exist
   if( _chdir( "c:\\fred" ) != 0 )
   {
       cout << "Unable to change to directory C:\\fred" << endl;
   }


   return 0;
}
```

Code to do a directory listing



```
void dirlisting()
{
    _finddata_t aFile;
    intptr_t dHandle;
    char pattern[30] = "*.*";

    dHandle = _findfirst( pattern, &aFile );
    if( dHandle == -1 )
        return;

    do
    {
        cout << aFile.name << endl;
    }while( _findnext( dHandle, &aFile ) == 0 );

    _findclose( dHandle );


}
```

Code to find hidden files and directories



```
void find_hidden( )
{
    _finddata_t aFile;
    intptr_t dHandle;
    char pattern[30] = "*.*";

    dHandle = _findfirst( pattern, &aFile );
    if( dHandle == -1 )
    {
        return;
    }

    do
    {
        if( aFile.attrib & _A_HIDDEN )
            cout << "<Hidden>  ";
        else
            cout << "<Visible> ";

        if( aFile.attrib & _A_SUBDIR )
            cout << "<DIR>  ";
        else
            cout << "<File> ";

        cout << aFile.name << endl;

    }while( _findnext( dHandle, &aFile ) == 0 );
}
```