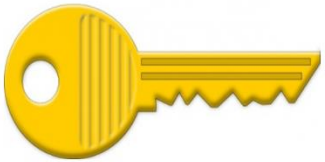
Modeling CDs

Objectives

You will be able to:

- Sketch a diagram showing the relationship between a class and classes that it contains a members.
- Define classes that have objects as members.
- Define classes that have a variable number of members.
- Describe and use Comma Separate Values (CSV) files.
- Initialize objects from a CSV file.

Classes and Models



Key Concept

- A class definition is based on a *model* of something to be represented in a program.
  - Captures what we care about.
  - Omits everything we don't care about.
- Object oriented design begins with the design of models.
  - Starts with models of real world things.
  - Evolves these models into class definitions.

CD Class

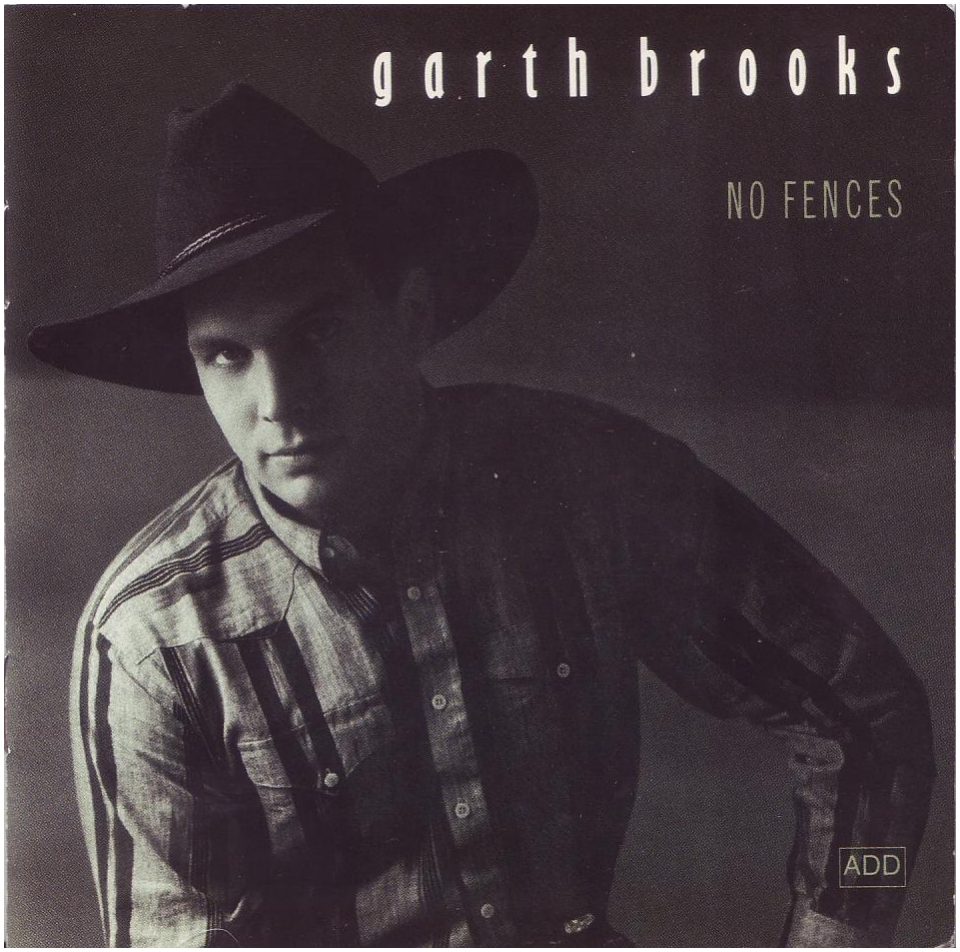
- Let's define a class to represent CDs.
  - Information *about* the CDs
  - Not the content.

- The main program will create a collection of information about CDs.
  - A *catalog*, not a CD player.

Object Oriented Design

- Attributes of a CD
  - What do we want to know about the CDs in our collection?
- Operations
  - What do we want the program to *do* with the information that we have about CDs?

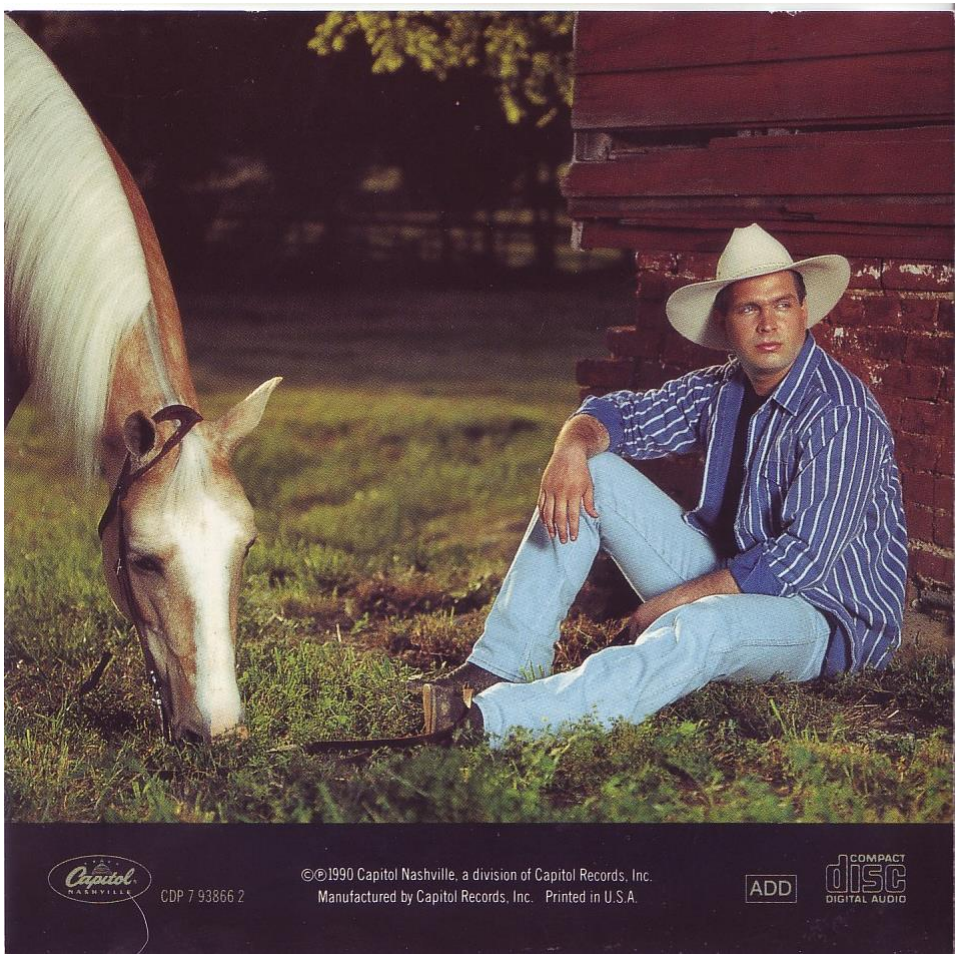
Attributes of a CD



Artist

Title

Attributes of a CD



Manufacturer

Copyright Date

Recording Technology

http://www.danielmcadam.com/AAD-ADD-DDD.html

A CD in Windows Media Player

Attributes and Operations of a CD

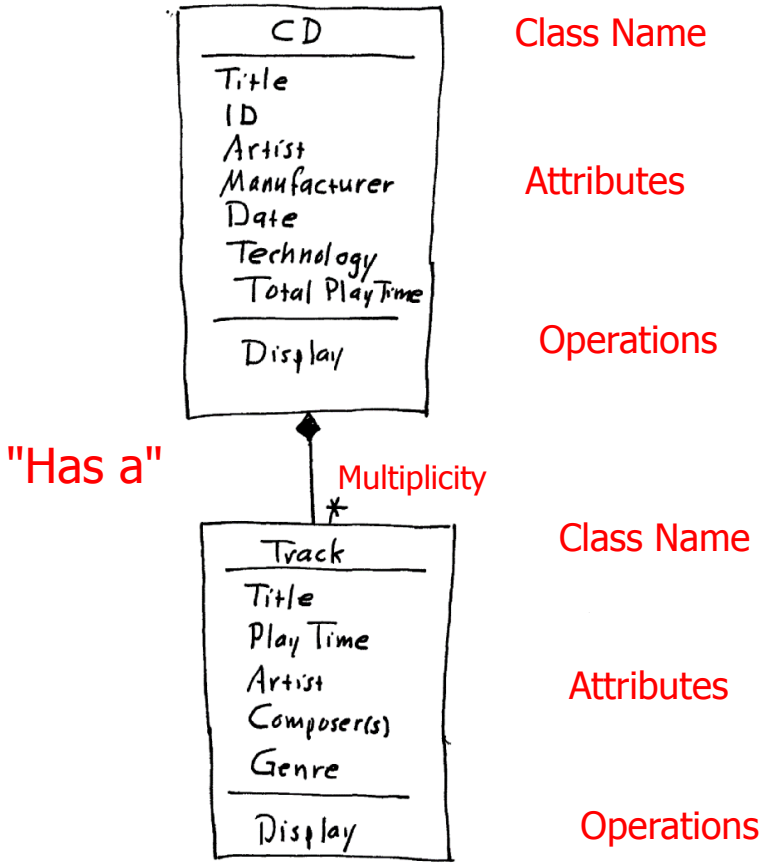
- Attributes
  - Title
  - ID (Usually ISBN)
  - Artist
  - Manufacturer
  - Date
  - Technology
  - Total play time
  - Collection of *Tracks*

- Operations
  - Display

Attributes of a Track

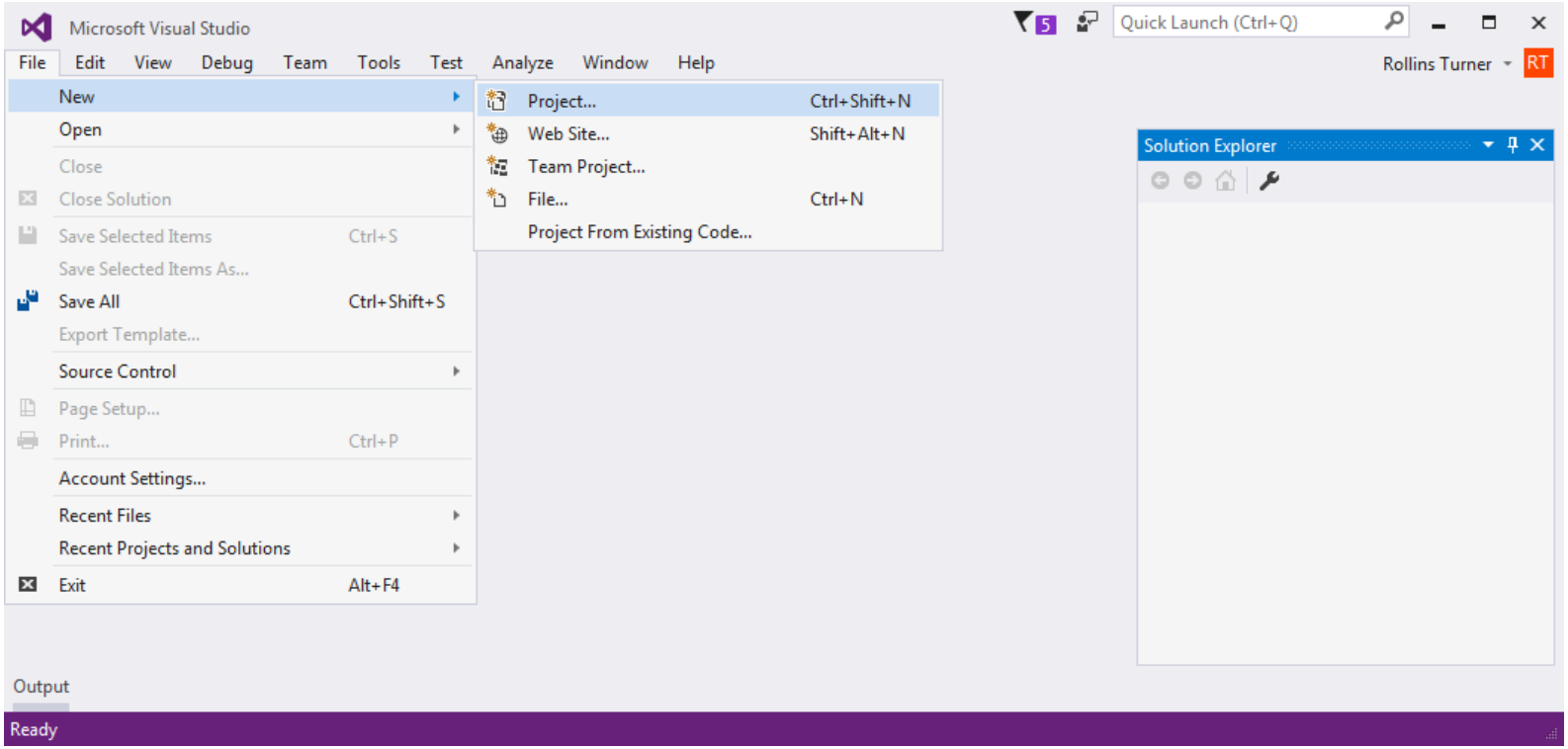
- Attributes
  - Title
  - Play time
  - Artist
  - Composer
  - Genre
- Operations
  - Display

A Class Diagram

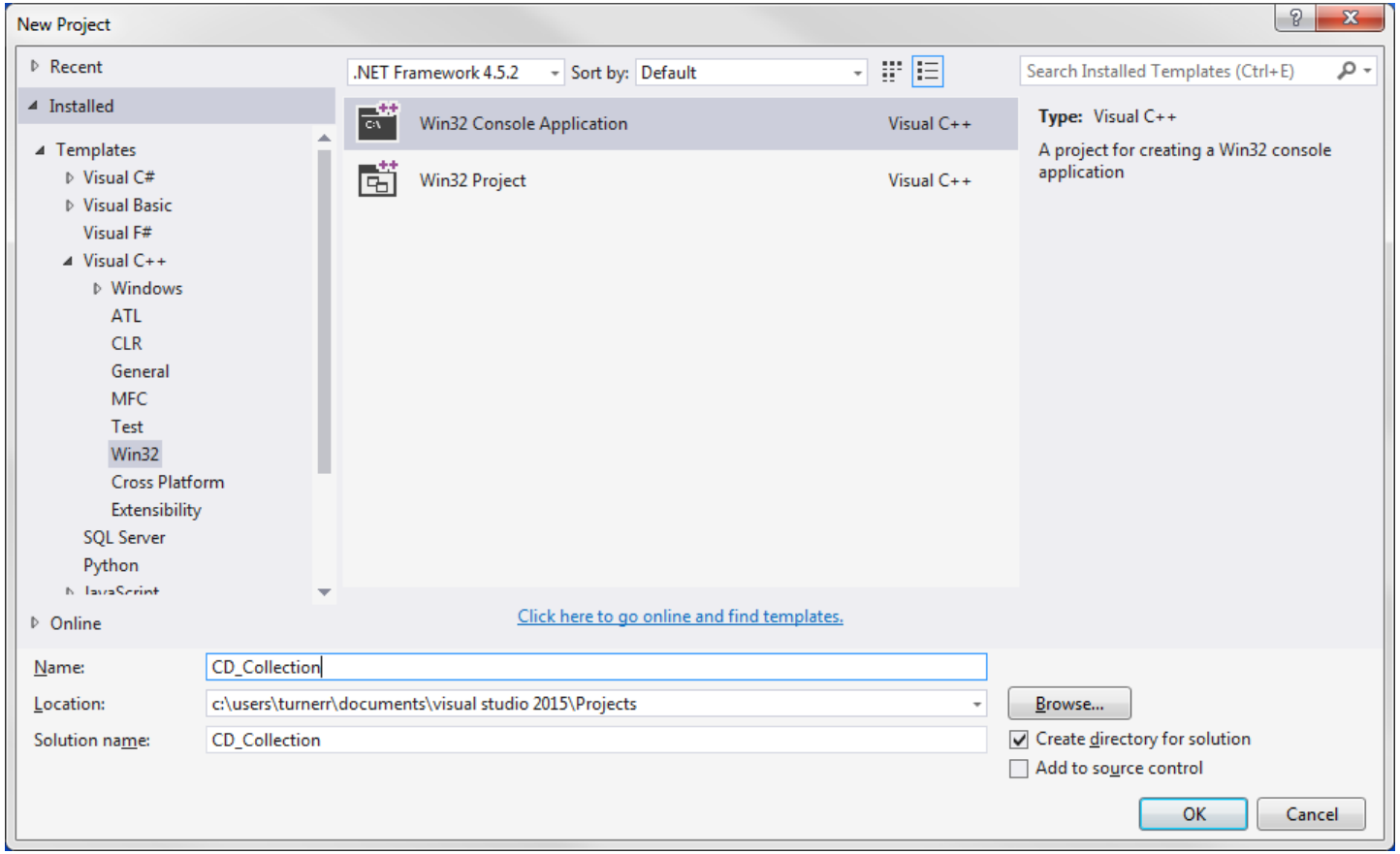


Let's write some code to implement the CD class.

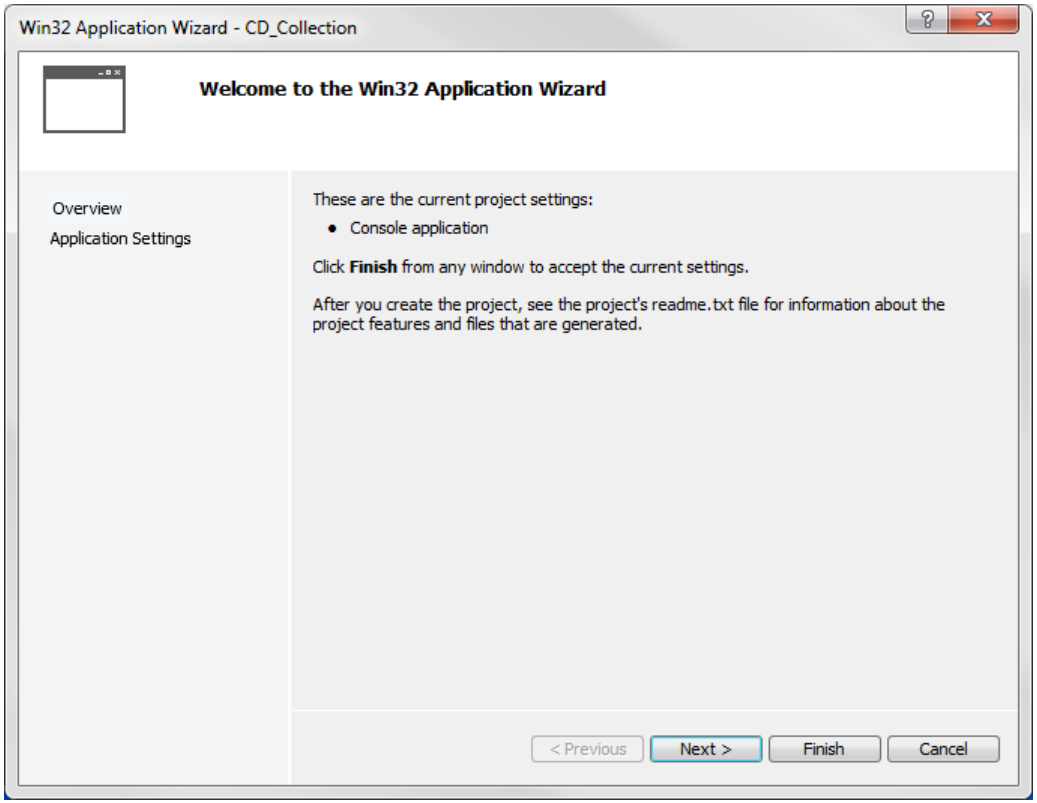
Create a Project



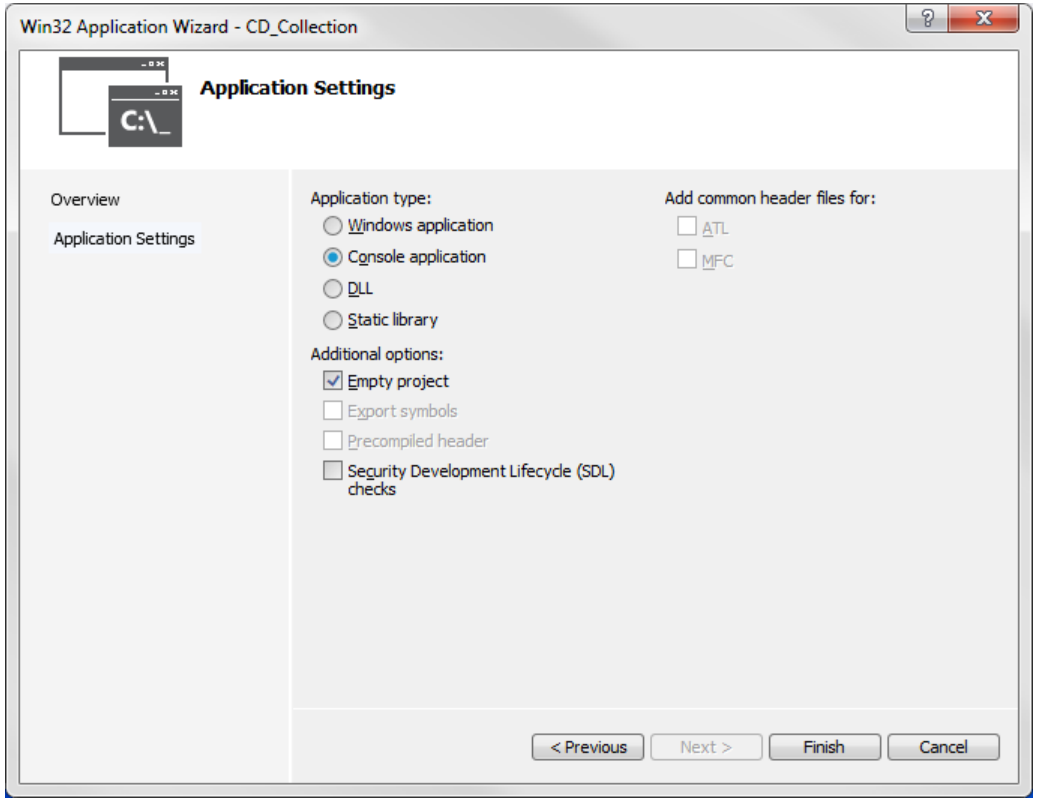
Creating a Project



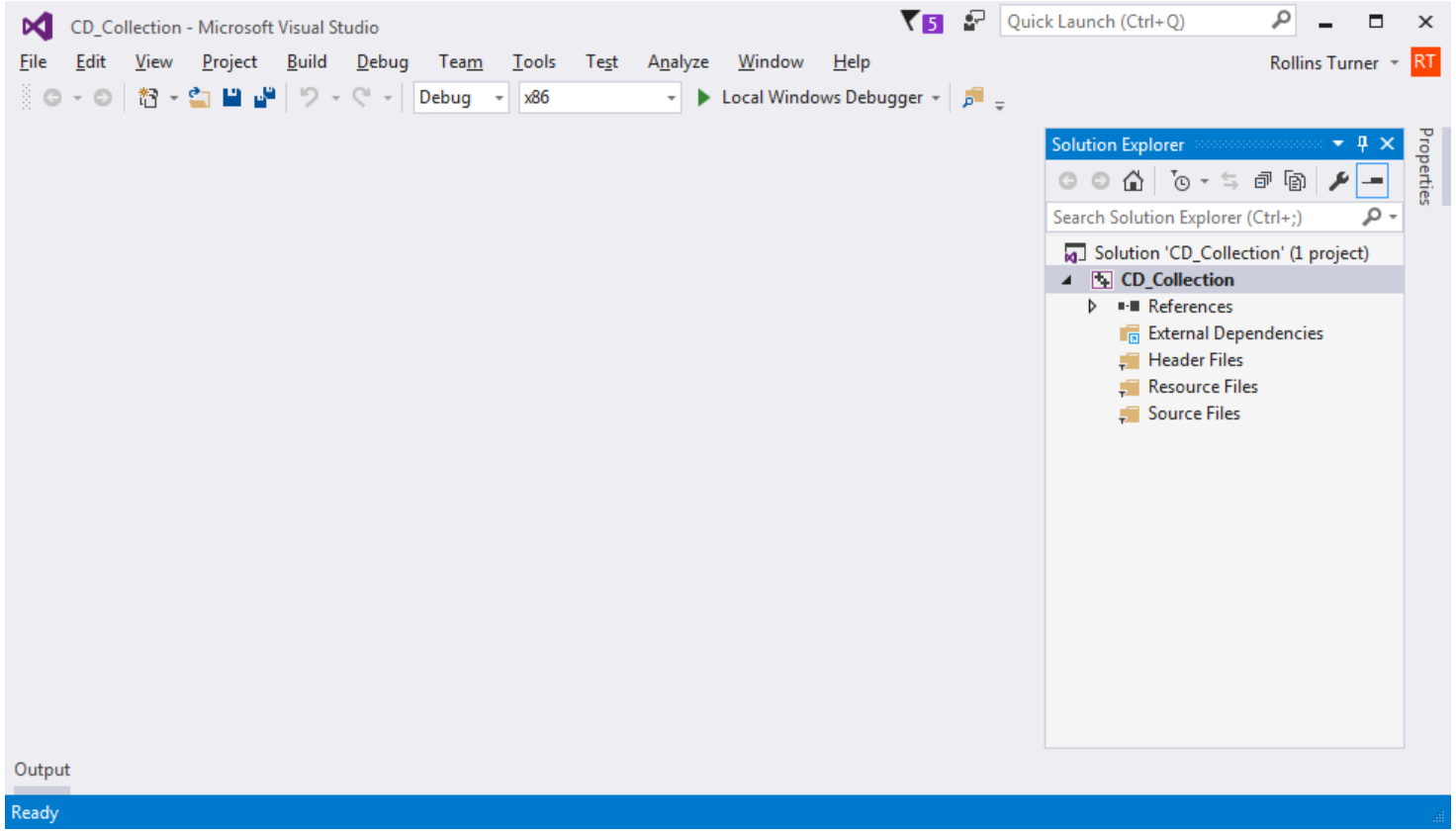
Creating a Project



Creating a Project



New Empty Project



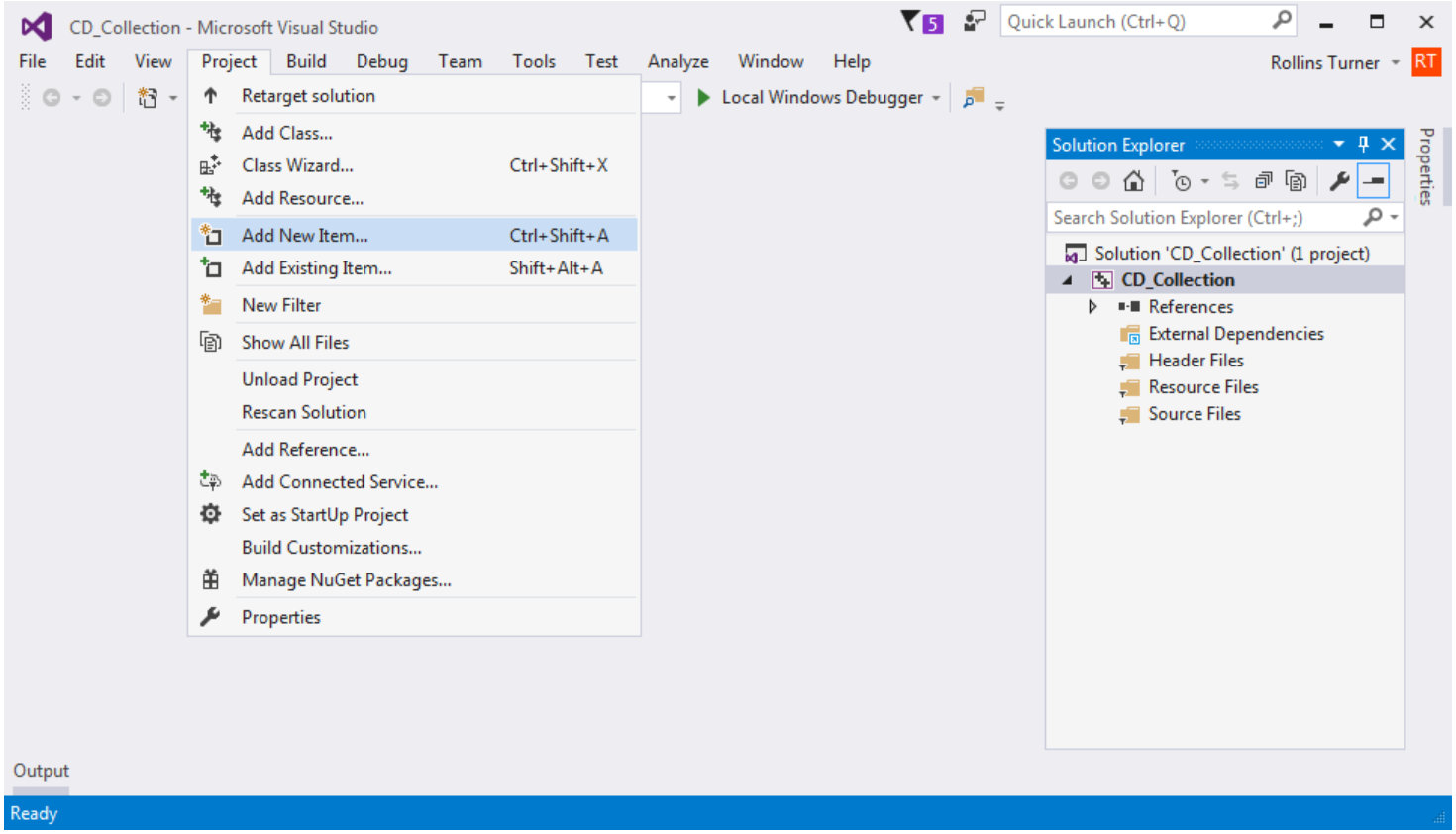
Test_CD

- Add file Test_CD.cpp to the project.
  - Project > Add New Item

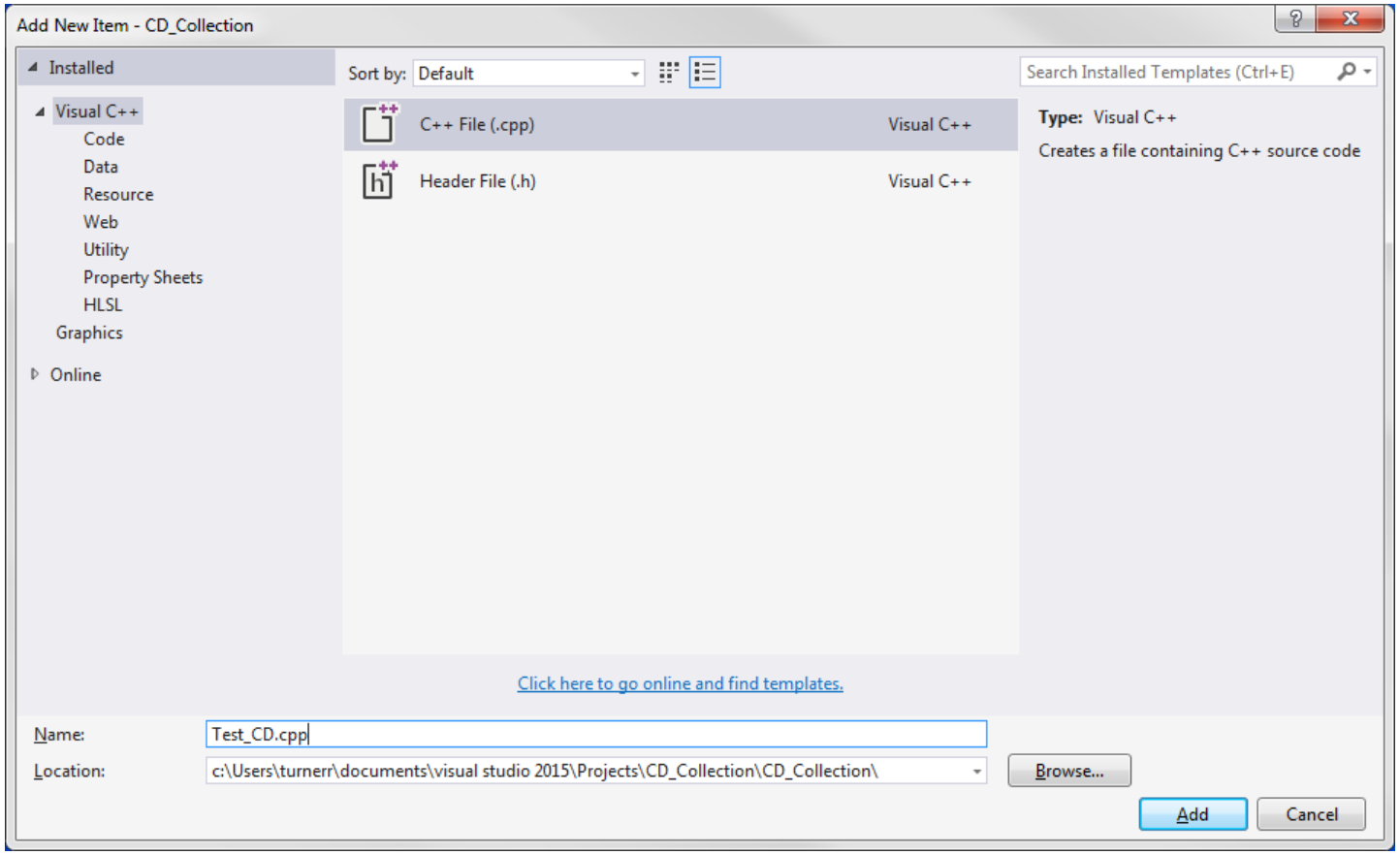
- This will be our “main()”
- Start with “Hello, World!”

- We will fill in the real test code after we have something to test.

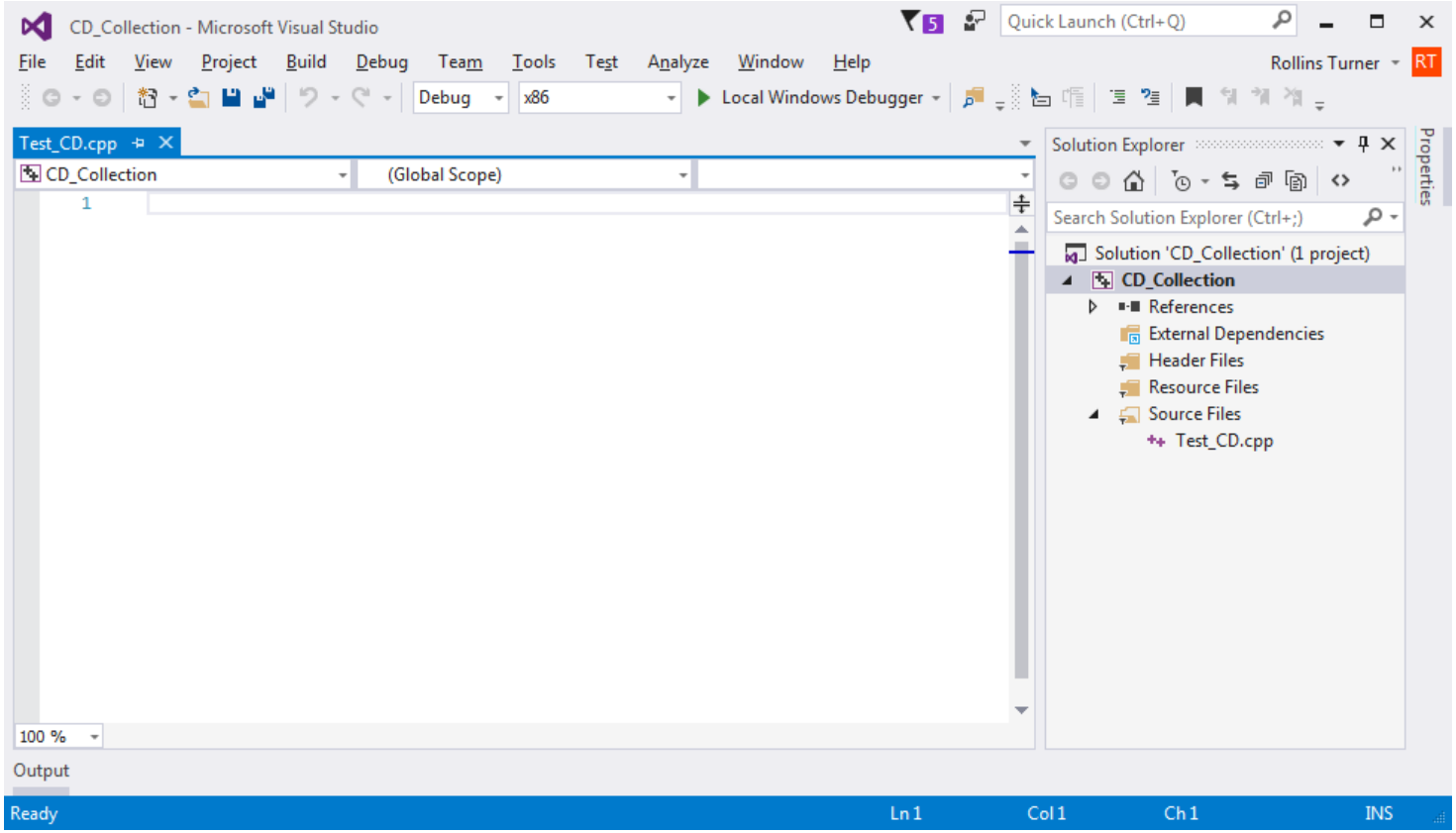
Adding Test_CD.cpp



Adding Test_CD.cpp



Initial File

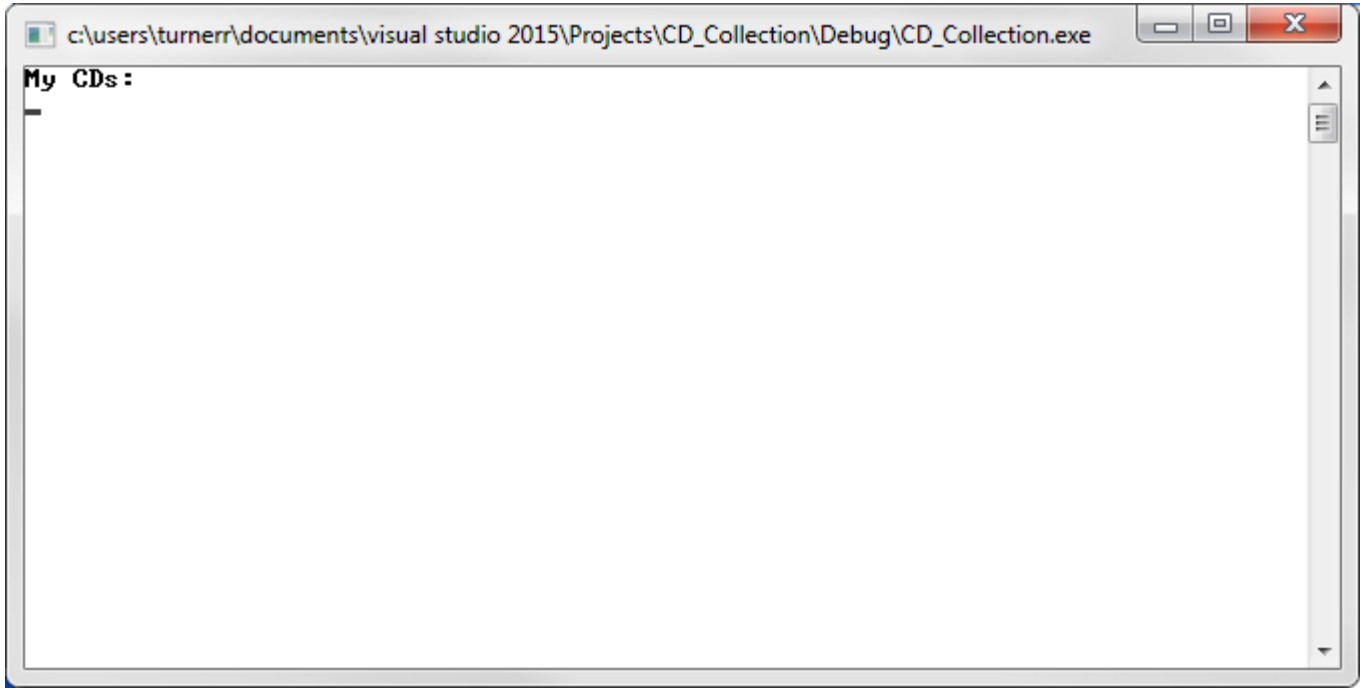


Test_CD

```
#include <iostream>

using namespace std;

int main()
{
    cout << "My CDs:\n";

    cin.get();
}
```

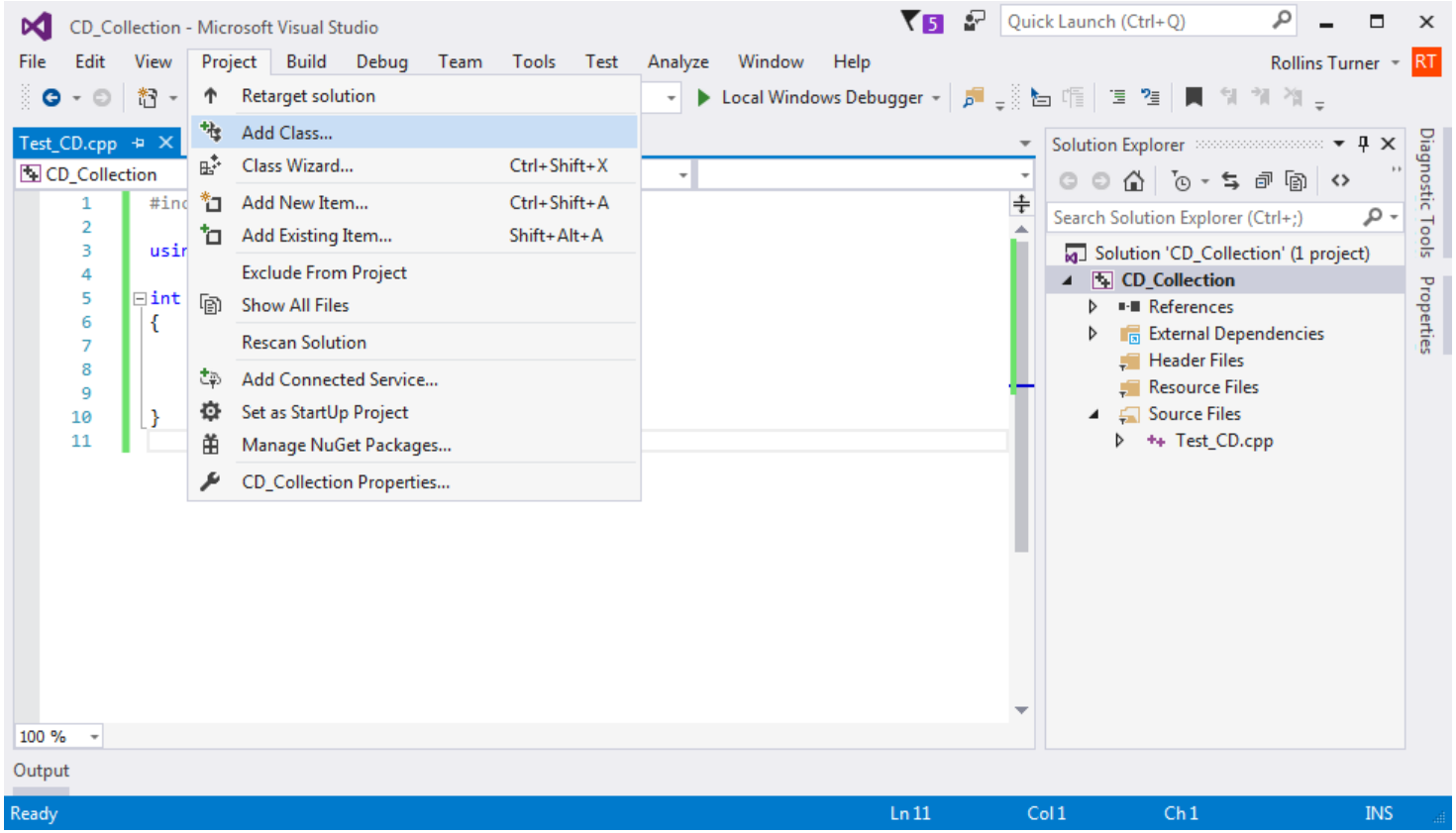
Build and run.

Test_CD

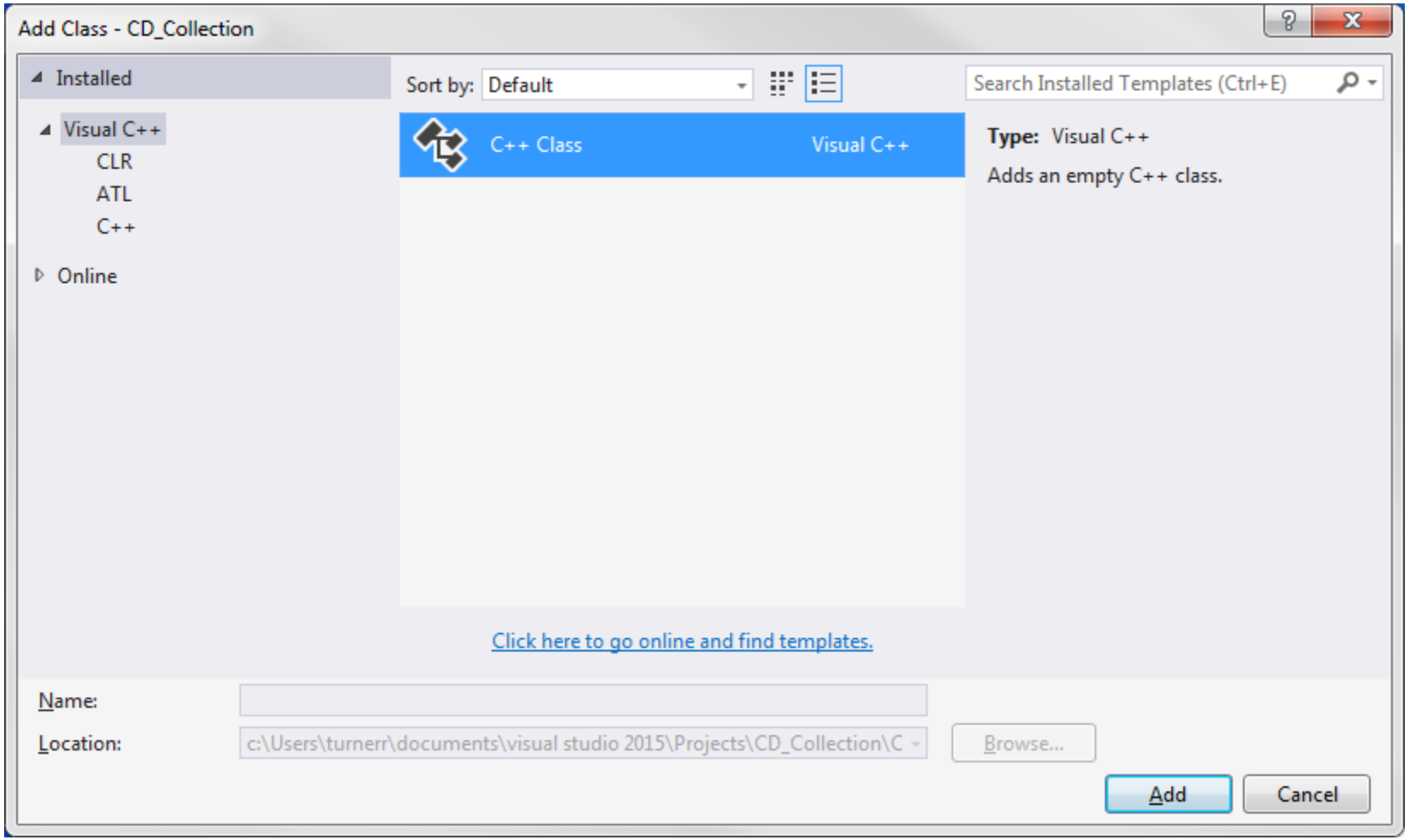


We have a working program!

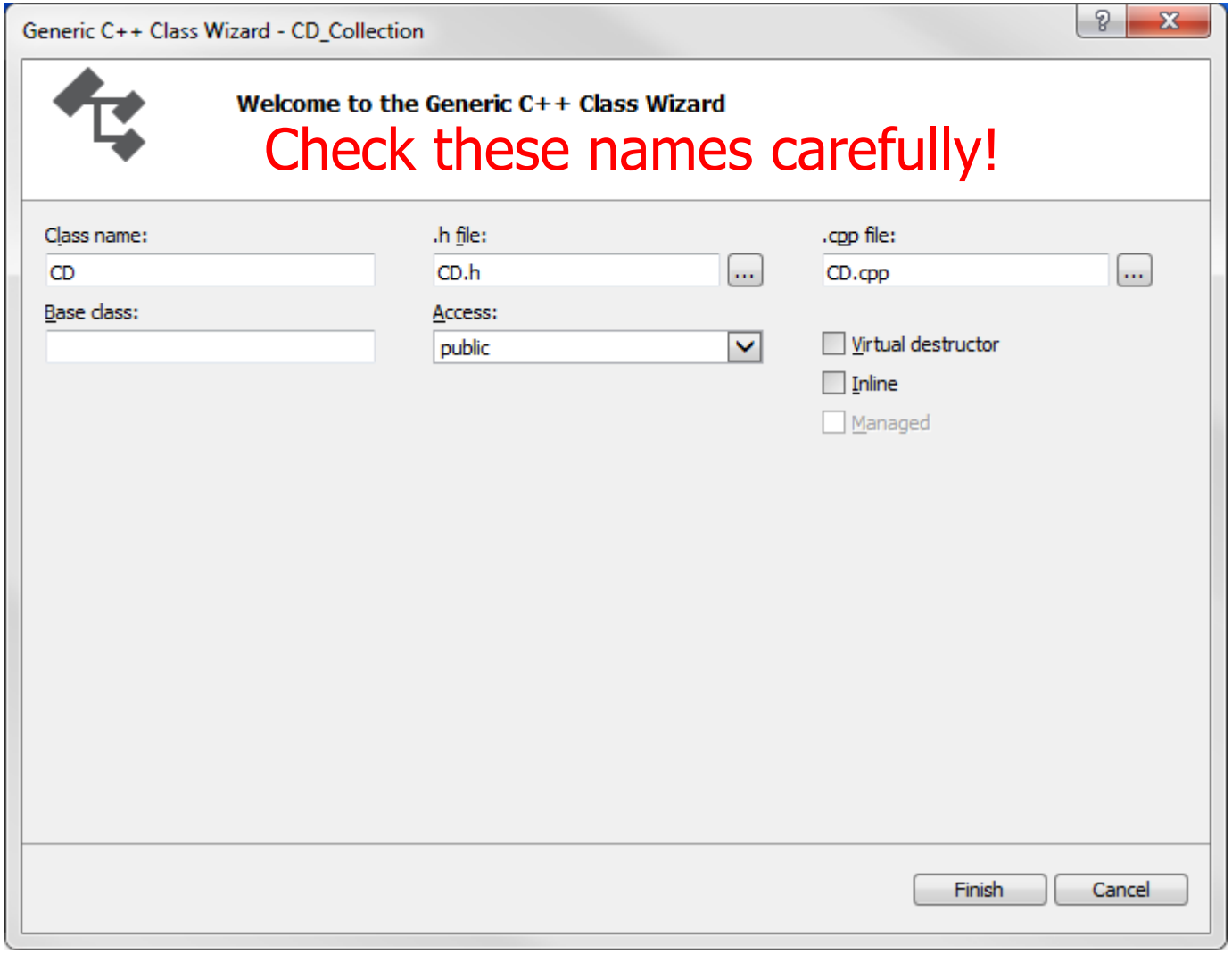
Add Class CD


CD_Collection - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File
Edit
View
Project
Build
Debug
Team
Tools
Test
Analyze
Window
Help
Rollins Turner
Retarget solution
Add Class...
Class Wizard... Ctrl+Shift+X
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A
Exclude From Project
Show All Files
Rescan Solution
Add Connected Service...
Set as StartUp Project
Manage NuGet Packages...
CD_Collection Properties...
Local Windows Debugger
Test_CD.cpp
CD_Collection
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CD_Collection' (1 project)
CD_Collection
References
External Dependencies
Header Files
Resource Files
Source Files
Test_CD.cpp
Diagnostic Tools
Properties
100 %
Output
Ready
Ln 11
Col 1
Ch 1
INS

Add Class CD


Add Class - CD_Collection
Installed
Visual C++
CLR
ATL
C++
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
C++ Class
Visual C++
Type: Visual C++
Adds an empty C++ class.
Click here to go online and find templates.
Name:
Location:
c:\Users\turnerr\documents\visual studio 2015\Projects\CD_Collection\C
Browse...
Add
Cancel

Add Class CD


Generic C++ Class Wizard - CD_Collection
Welcome to the Generic C++ Class Wizard
Check these names carefully!
Class name:
CD
.h file:
CD.h
.cpp file:
CD.cpp
Base class:
Access:
public
Virtual destructor
Inline
Managed
Finish
Cancel

Class CD

CD.h

```
#pragma once
#include <string>

using namespace std;

enum Recording_Technology {AAD, ADD, DDD, UNK};

class CD
…
```

CD.h (continued)

```
class CD
{

    private:
        string title;
        string id;
        string artist;
        string manufacturer;
        int year;
        Recording_Technology rec_tech;

    public:
        CD(void);
        CD(string title_, string id_, string artist_,
            string manufacturer_, int year_,
            rec_tech rec_tech_);
        ~CD(void);

        void Display() const;
};
```

Implement Class CD

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

using namespace std;
```

The Constructor

```
CD::CD(string title_, string id_, string artist_,
       string manufacturer_, int year_,
       Recording_Technology rec_tech_)  :
       title(title_), id(id_), artist(artist_),
       manufacturer(manufacturer_), year(year_),
       rec_tech(rec_tech_)
{}
```

Initialization List

Display

```
void CD::Display() const
{
    cout.fill('0');
    cout << "CD: " << title << endl;
    cout << "ID: " << id << endl;
    cout << "Artist: " << artist << endl;
    cout << "Mfgr: " << manufacturer << endl;
    cout << "Year: " << year << endl;
    cout << "Recording technology: " << rec_tech << endl;

    // Add code to display total play time

    // Add code to display each track
}
```

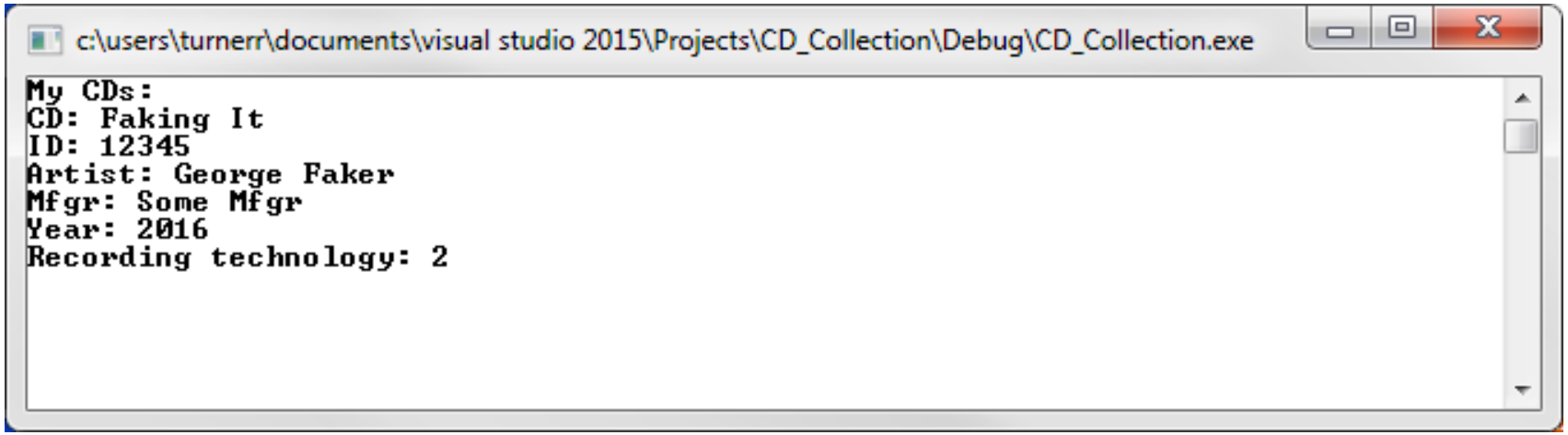
Initial Test Code

- Let's add code to create and display a single CD, using fictitious data.

Test_CD.cpp

```
#include <iostream>
#include "CD.H"

using namespace std;

int main()
{
    cout << "My CDs:\n";

    CD* cd1 = new CD("Faking It", "12345", "George Faker",
                     "Some Mfgr", 2016, DDD);

    cd1->Display();

    cin.get();
}
```

Test



Results look OK.

EXCEPT "2" is not what we want to see for Recording technology.

We really want to see "DDD".

Displaying Recording_Technology

- Let's add code to convert Recording_Technology values to a string.
- Overload the << operator for Recording_Technology to use this function.

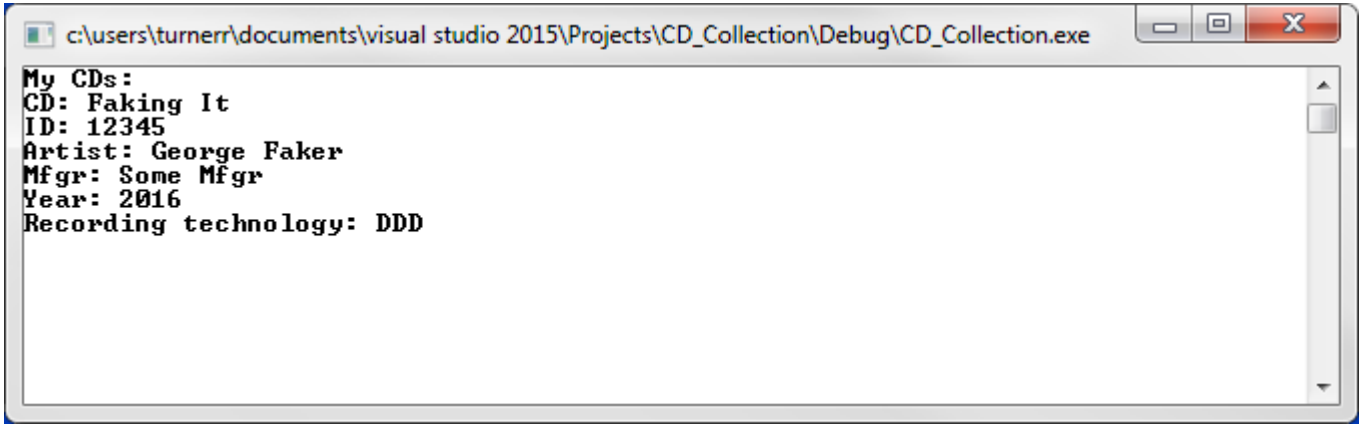
In CD.h

```
enum Recording_Technology { AAD, ADD, DDD, UNK };

string& ToString(const Recording_Technology& rec_tech);

ostream& operator<<(ostream& os,
          const Recording_Technology& rec_tech);
```

In CD.cpp

```
string& ToString(const Recording_Technology& rec_tech)
{
    string* str;
    switch (rec_tech)
    {
        case AAD: str = new string("AAD");  break;
        case ADD: str = new string("ADD");  break;
        case DDD: str = new string("DDD");  break;
        default: str = new string("UNK");
    }
    return *str;
}

ostream& operator<<(ostream& os, const Recording_Technology& rec_tech)
{
    os << ToString(rec_tech);
    return os;
}
```

Try it!

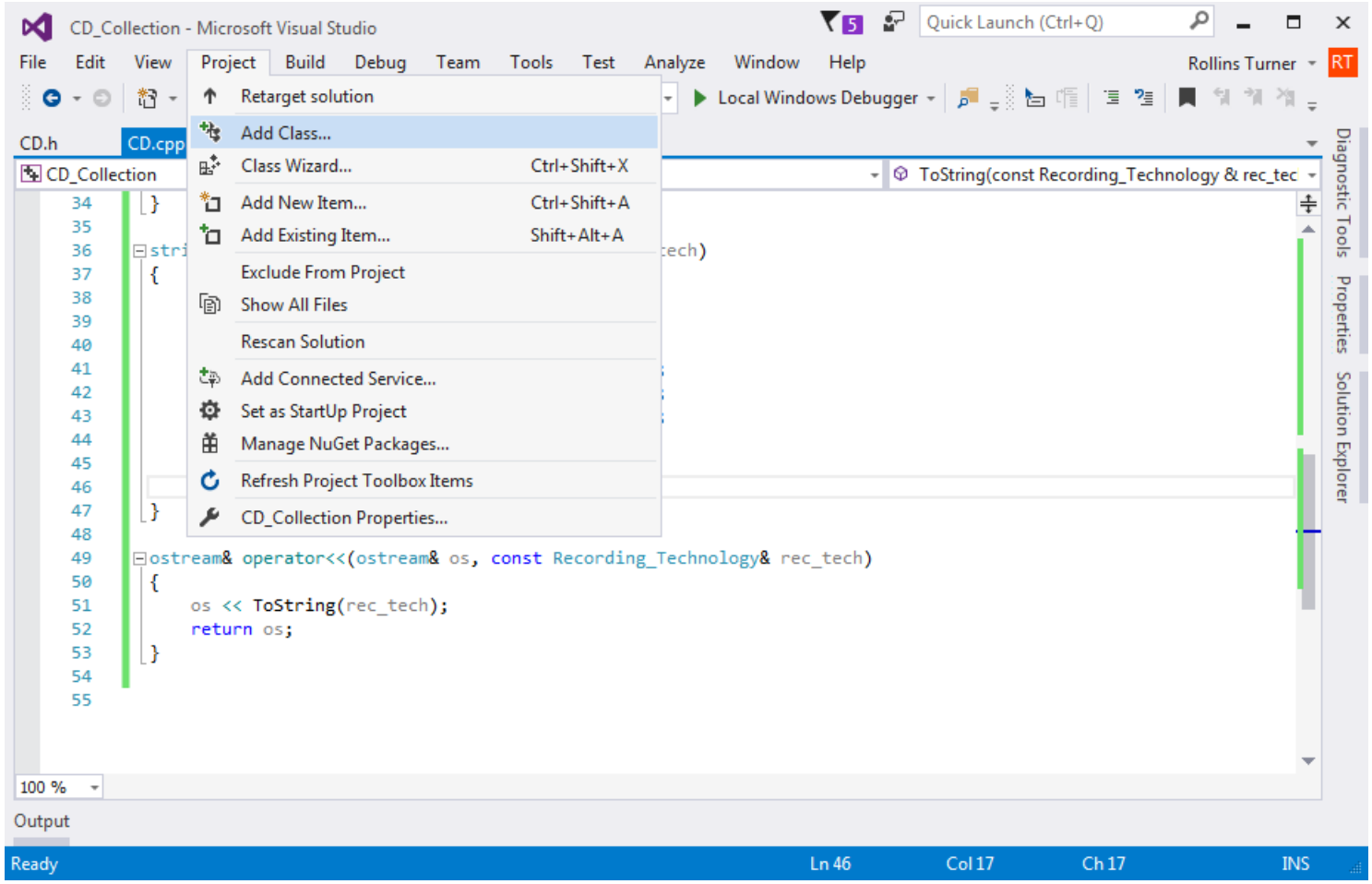


End of Section

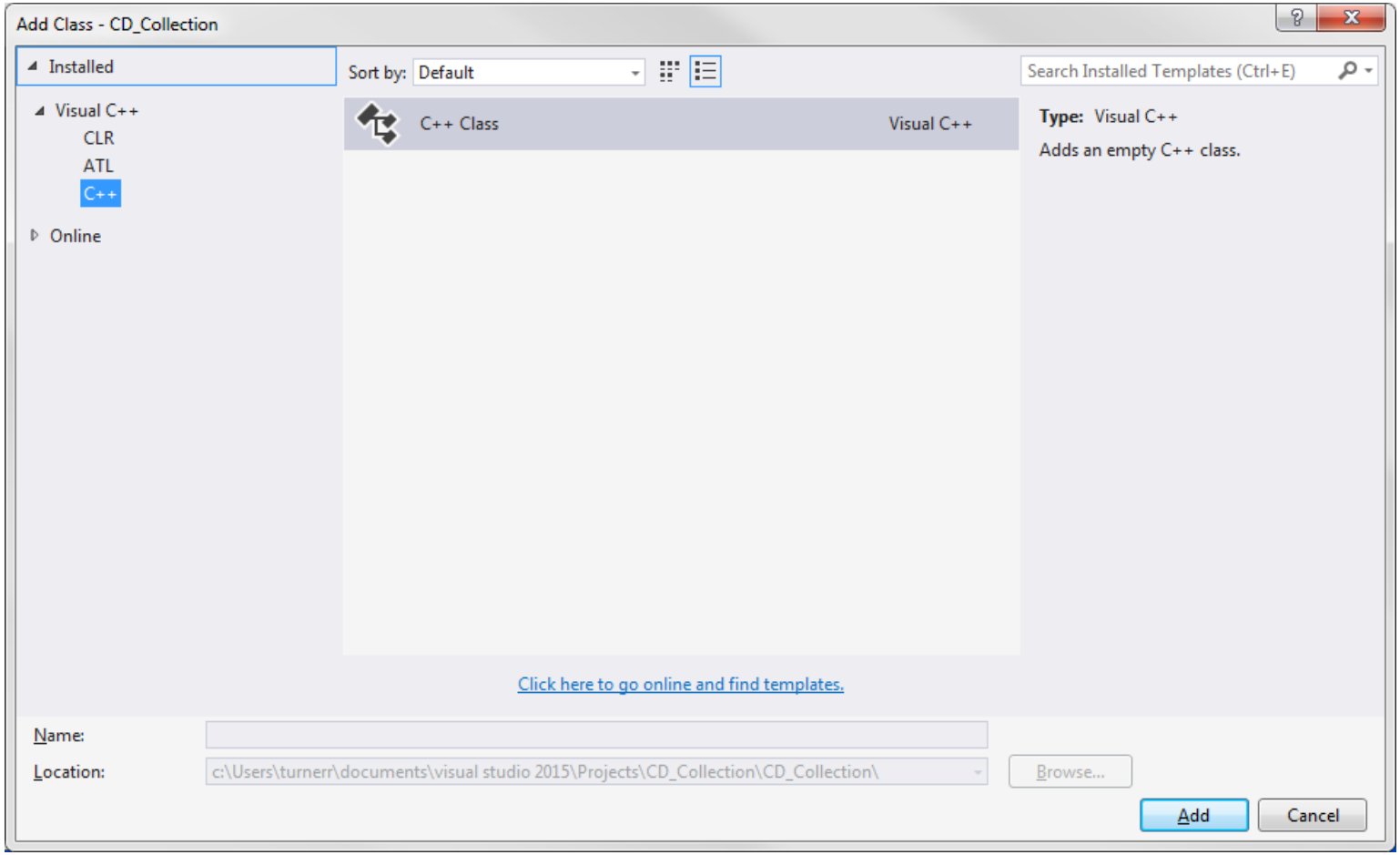
Class Track

- We need another class to hold information about each track.
  - Class Track
- A CD object will contain a variable number of track objects.
  - We will need to add code to the CD class to account for the tracks.

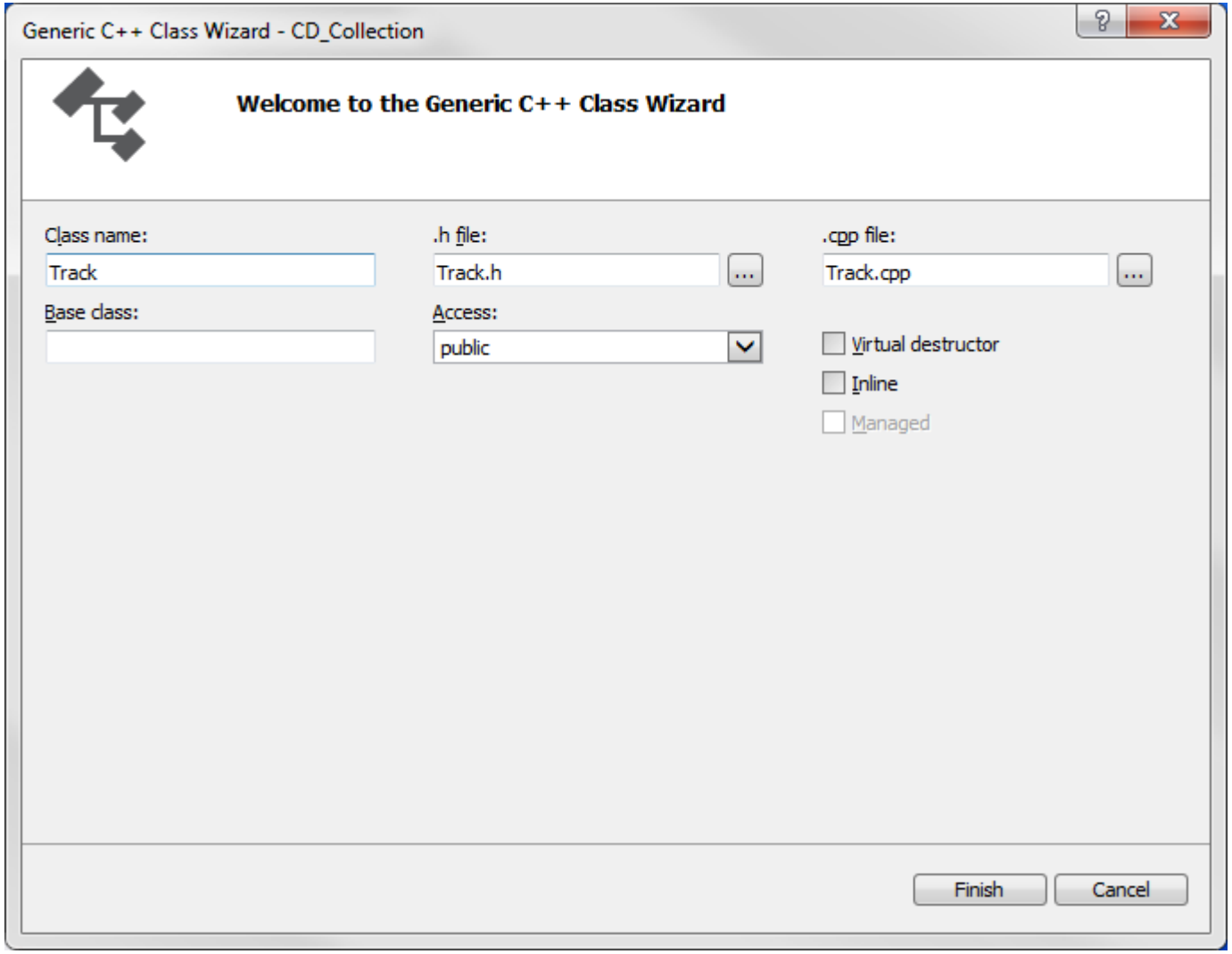
Add Class Track



Adding Class Track



Adding Class Track



Initial File

CD_Collection - Microsoft Visual Studio

Track.h* | Track.cpp* | CD.h* | CD.cpp* | Test_CD.cpp

CD_Collection | Track

```
1  #pragma once
2  class Track
3  {
4  public:
5      Track();
6      ~Track();
7  };
8
9
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'CD_Collection' (1 project)
  - CD_Collection
    - References
    - External Dependencies
    - Header Files
      - CD.h
      - Track.h
    - Resource Files
    - Source Files
      - CD.cpp
      - Test_CD.cpp
      - Track.cpp

Ready | Ln 1 | Col 1 | Ch 1 | INS

Class Track

Track.h

```
#pragma once
#include <string>
using namespace std;

enum Genre {Classical, Pop, Country, Folk, Rap, Hip_Hop, Unknown};

class Track
{
private:
    string title;
    int play_time;      // Seconds
    string artist;
    Genre genre;
```

Track.h (continued)

```
public:
    Track(void);
    Track(string title_, int play_time_,
          string artist_, Genre genre_);
    ~Track(void);

    int Play_Time() const {return play_time;};
    void Display() const;
};
```

Implement Class Track

Track.cpp

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

Track::Track(string title_, int play_time_,
             string artist_, Genre genre_) :
  title(title_), play_time(play_time_), artist(artist_),
  genre(genre_)
{}
```

initialization list

```
Track::~Track(void)
{}
```

Track.cpp

```
void Track::Display() const
{
    cout.fill('0');
    cout << title << endl;
    cout << "\tPlay time: " << (play_time / 60) << ":" ;
    cout.width(2);
    cout << (play_time%60) << endl;
    cout << "\tArtist: " << artist << endl;
    cout << "\tGenre: " << genre << endl;
}
```

Collections as Members

- We often need a variable number of objects as members of a class.
  - Example: Tracks of a CD.
- How do we get these members into the object?
  - Constructor?
- "Add" method.
  - Can be called multiple times.
  - A supplement to the constructor.

In CD.h

```
#include "Track.h"

...
public:
    static const int MAX_TRACKS = 50;
...
private:
...
    Track* tracks[MAX_TRACKS];
    int nr_tracks;
...
public:
...
    void Add_Track(Track* track);
    int Total_Play_Time() const;
    void Display() const;
};
```

In CD.cpp

```
#include <iostream>
#include "CD.h"
#include "Track.h"
#include <cassert>

...
    title(title_), id(id_), artist(artist_),
    manufacturer(manufacturer_), year(year_),
    rec_tech(rec_tech_),
    nr_tracks(0)
...

void CD::Add_Track(Track* track)
{
    assert(nr_tracks < MAX_TRACKS);
    tracks[nr_tracks++] = track;
}
```

Add to initializer list

Build and run

Add to CD.cpp

```
int CD::Total_Play_Time() const
{
    int total = 0;
    for (int i = 0; i < nr_tracks; ++i)
    {
        total += tracks[i]->Play_Time();
    }
    return total;
}
```

Add to CD::Display

```
cout << "Total play time: " << (Total_Play_Time()/ 60) << ":";
   cout.width(2);
   cout << (Total_Play_Time()%60) << endl;
   cout << endl;
```

Add to CD::Display

```
void CD::Display() const
{
    cout.fill('0');
    cout << "CD: " << title << endl;
    cout << "ID: " << id << endl;
    cout << "Artist: " << artist << endl;
    cout << "Mfgr: " << manufacturer << endl;
    cout << "Year: " << year << endl;
    cout << "Recording technology: " << technology << endl;
    cout << "Total play time: " << (Total_Play_Time()/ 60) << ":";
    cout.width(2);
    cout << (Total_Play_Time()%60) << endl;
    cout << endl;

    for (int i = 0; i < nr_tracks; ++i)
    {
        cout << "\tTrack " << i+1 << ": ";
        tracks[i]->Display();
        cout << endl;
    }
}
```

Test the Track Code

Add to Test_CD.cpp:

```
...
Track* track1 = new Track("A Fake Track", 130, "Fake Artist",
                                Pop);
    cd1->Add_Track(track1);
    cd1->Display();

    cin.get();
}
```

Test Result

c:\users\turnerr\documents\visual studio 2015\Projects\CD_Collection\Debug\CD_Collection.exe

```
My CDs:
CD: Faking It
ID: 12345
Artist: George Faker
Mfgr: Some Mfgr
Year: 2016
Recording technology: DDD
Total play time: 2:10

        Track 1: A Fake Track
        Play time: 2:10
        Artist: Fake Artist
        Genre: 1
```

Looks good, except for Genre: 1

Add a ToString function and << operator for Genre.

Add to Track.h

```
enum Genre { Classical, Pop, Country, Folk, Rap, Hip_Hop, Unknown };
string& ToString(const Genre& genre);
ostream& operator<<(ostream& os, const Genre& genre);
```

Add to Track.cpp

```
string& ToString(const Genre& genre)
{
    string* str;
    switch (genre)
    {
        case Classical: str = new string("Classical"); break;
        case Pop: str = new string("Pop"); break;
        case Country: str = new string("Country"); break;
        case Folk: str = new string("Folk"); break;
        case Rap: str = new string("Rap"); break;
        case Hip_Hop: str = new string("Hip_Hop"); break;
        default: str = new string("Unknown"); break;
    }
    return *str;
}

ostream& operator<<(ostream& os, const Genre& genre)
{
    os << ToString(genre);
    return os;
}
```

Test Result

c:\users\turnerr\documents\visual studio 2015\Projects\CD_Collection\Debug\CD_Collection.exe

```
My CDs:
CD: Faking It
ID: 12345
Artist: George Faker
Mfgr: Some Mfgr
Year: 2016
Recording technology: DDD
Total play time: 2:10

        Track 1: A Fake Track
        Play time: 2:10
        Artist: Fake Artist
        Genre: Pop
```

End of Section

Capturing Real Data

- Media players have all of the information that we need for our CD catalog.
  - Some comes from the CD.
  - Some comes from an on-line database.
- We can't get the information directly from the CD.
  - An audio CD does not have a file system
- But we can copy the information from iTunes.

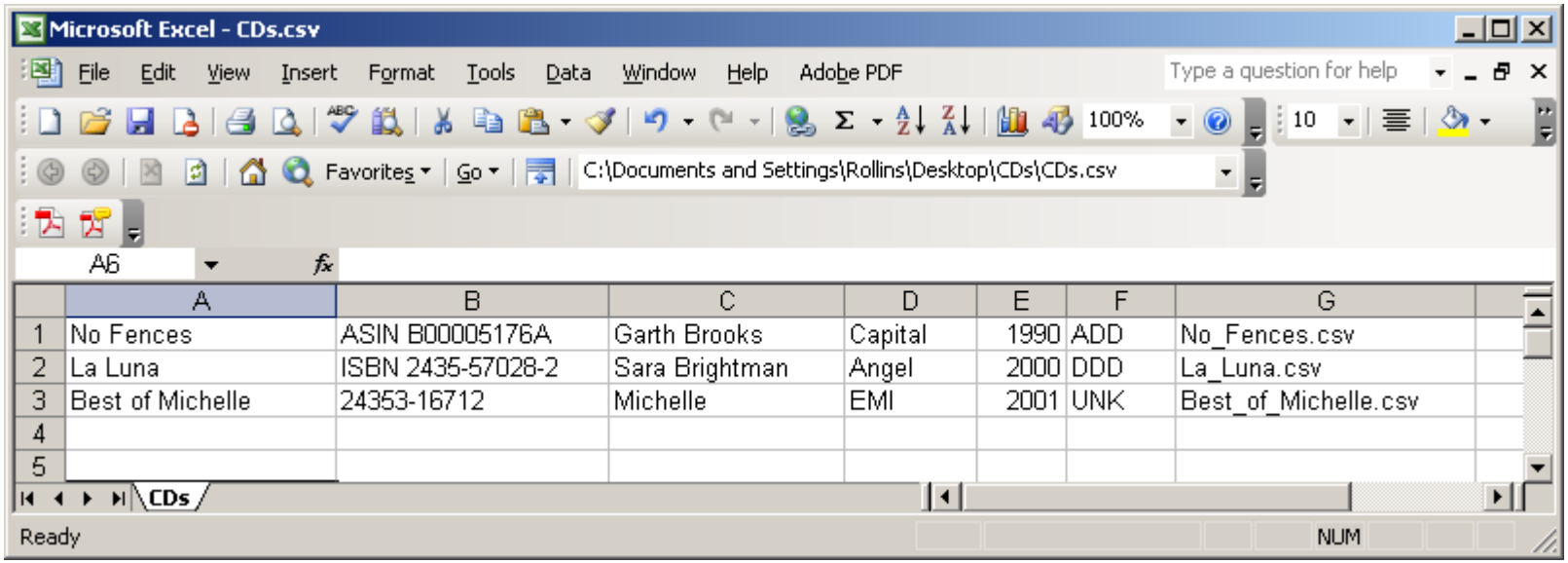
A CD in iTunes

CD Data

- The data for three CDs is available in the Downloads area of the class web site:
- http://www.cse.usf.edu/~turnerr/Object_Oriented_Design/Downloads/2016_02_05_CD_Collection/

- Comma Separated Values
  - Widely used format for structured text files.
  - Read and written by Excel

CDs.csv



	A	B	C	D	E	F	G
1	No Fences	ASIN B00005176A	Garth Brooks	Capital	1990	ADD	No_Fences.csv
2	La Luna	ISBN 2435-57028-2	Sara Brightman	Angel	2000	DDD	La_Luna.csv
3	Best of Michelle	24353-16712	Michelle	EMI	2001	UNK	Best_of_Michelle.csv
4							
5							

- We will need code to read a Recording Technology as a string and create a Recording_Technology enum value.

- Call it Parse_Rec_Tech, and add it to the CD class files (but not in the class.)

CD.h

Add to CD.h

```
enum Recording_Technology { AAD, ADD, DDD, UNK };
string& ToString(const Recording_Technology& rec_tech);
ostream& operator<<(ostream& os, const Recording_Technology& rec_tech);
Recording_Technology Parse_Rec_Tech(const string& str);
```

Add to CD.cpp:

```
Recording_Technology Parse_Rec_Tech(const string& str)
{
    if (str.compare("AAD") == 0) return AAD;
    if (str.compare("ADD") == 0) return ADD;
    if (str.compare("DDD") == 0) return DDD;
    return UNK;
}
```

Read CDs.csv

- Let's add code to read CDs.csv and create a collection of CD objects.
  - Without the track info initially.

- Create new files, CD_Info.h and CD_Info.cpp
  - Not a class

Adding CD_Info.h

CD_Collection - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

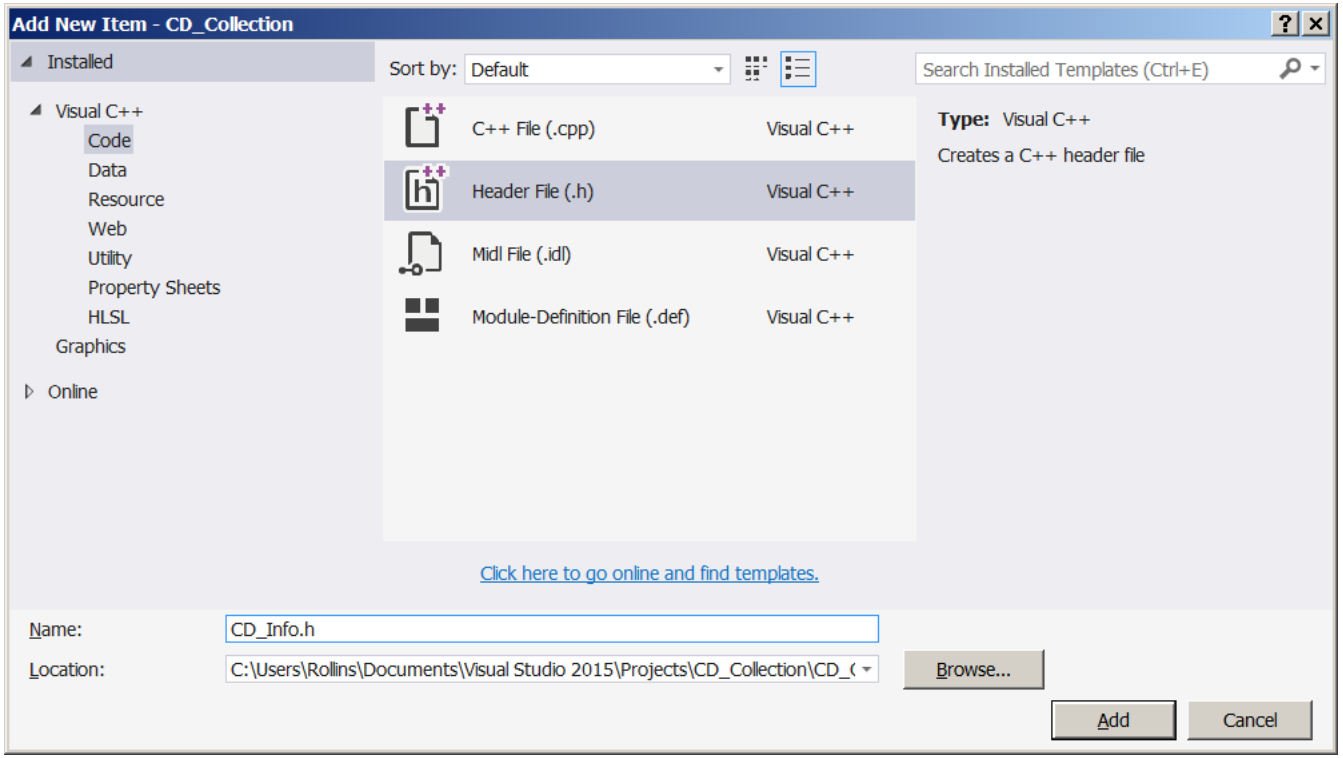
Retarget solution
Add Class...
Class Wizard... Ctrl+Shift+X
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A
Exclude From Project
Show All Files
Rescan Solution
Add Connected Service...
Set as StartUp Project
Manage NuGet Packages...
CD_Collection Properties...

```
    //cd1->Add_Track(track1);
    //cd1->Display();

    CD* my_cds[100];
    int nr_cds = 0;
//  int get_CDs(CD** cds, int max)

    nr_cds = get_CDs(&my_cds, 100);

    cin.get();
}
```

Solution Explorer
Solution 'CD_Collection' (
CD_Collection
References
External Depende
Header Files
CD.h
Track.h
Resource Files
Source Files
CD.cpp
CD_Info.cpp
Test_CD.cpp
Track.cpp

Ready Ln 20 Col 5 Ch 5 INS

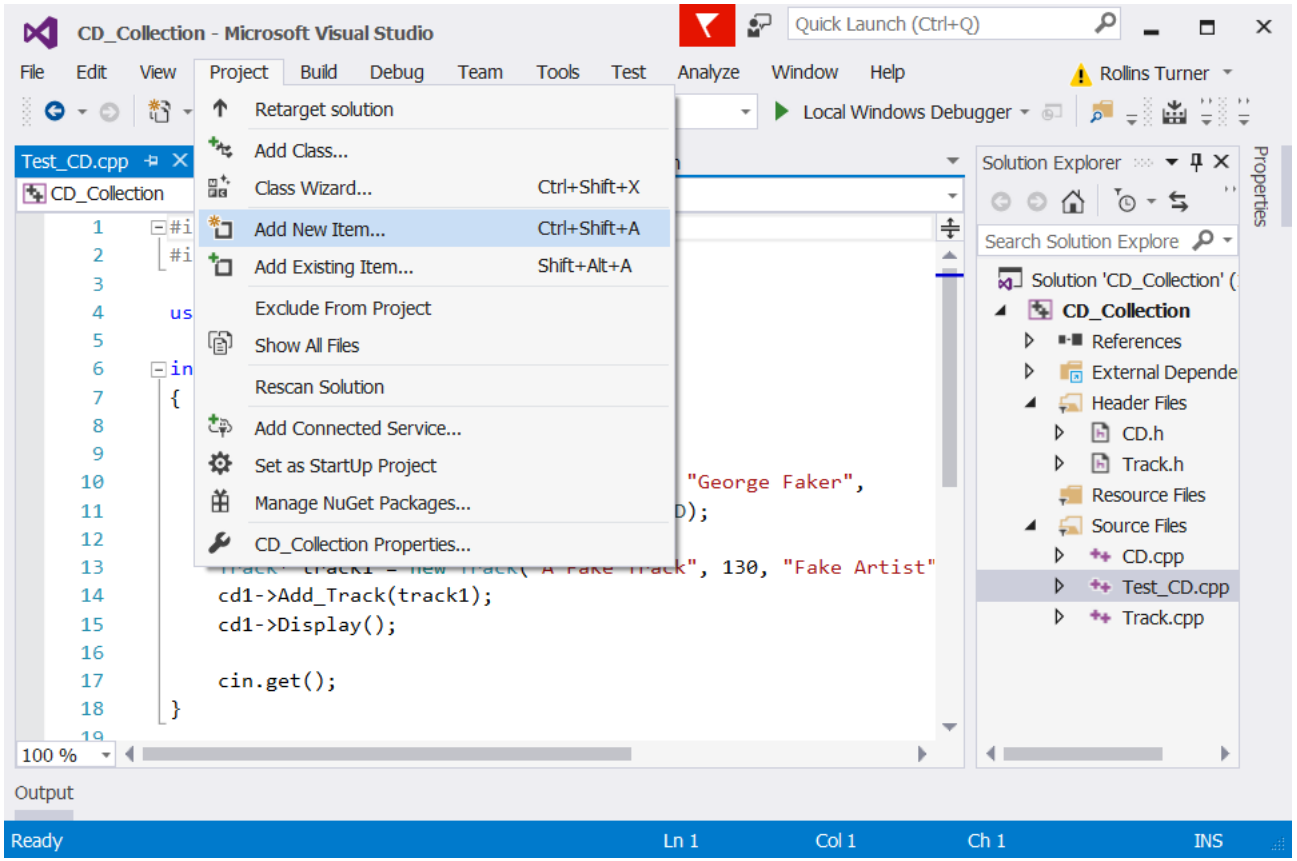
Adding CD_Info.h



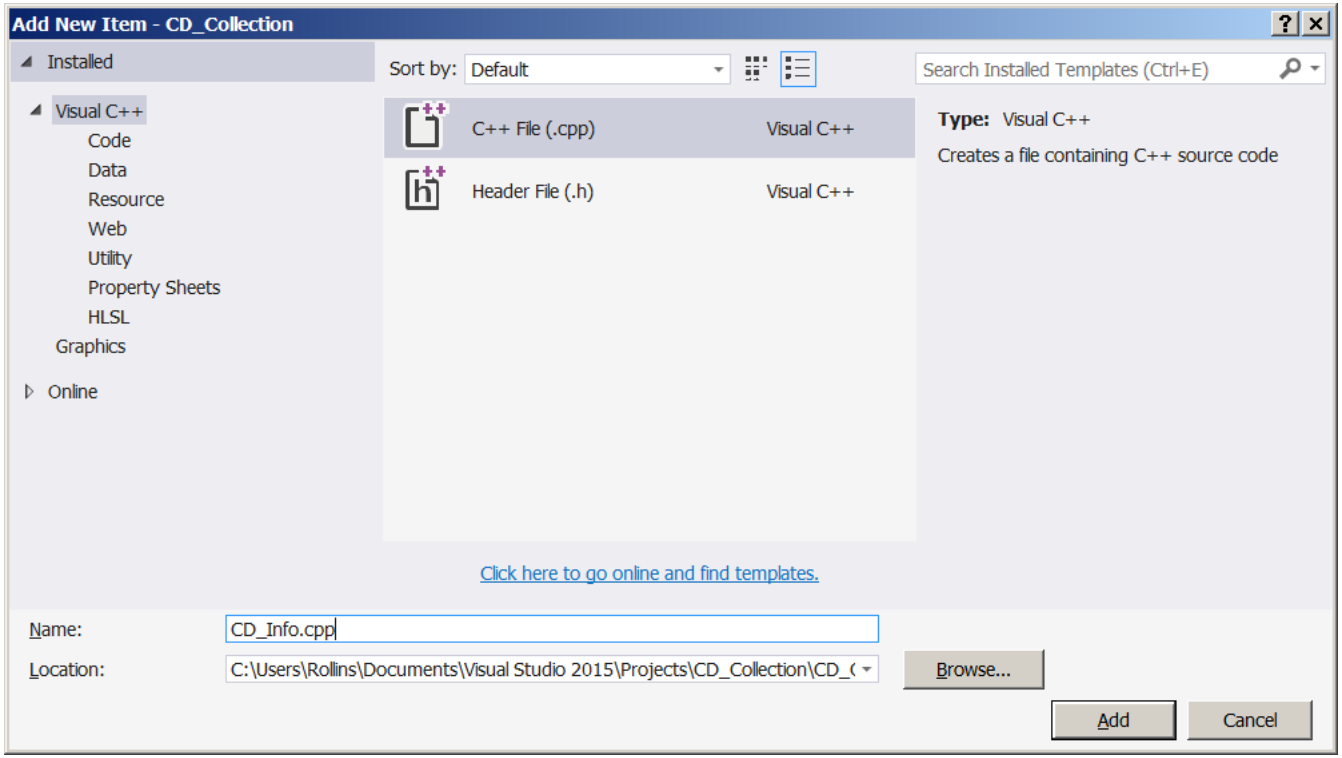
CD_Info.h

```
#pragma once
#include "CD.h"

// Read a CSV file of CD info and create the corresponding CDs
// at location specified by the caller.
// Return count of CDs created.
int get_CDs(CD** cds, int max);
```

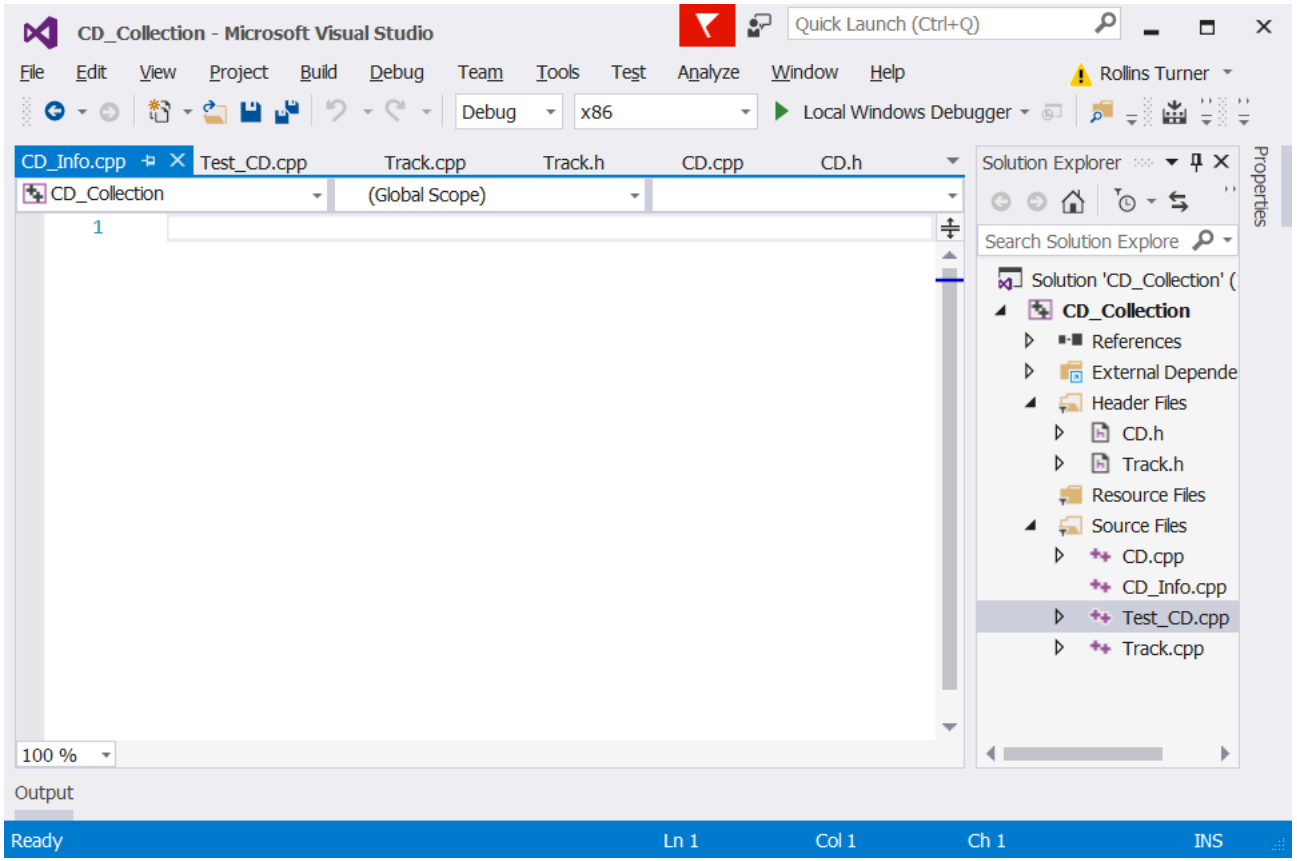
Adding CD_Info.cpp



Adding CD_Info.cpp



CD_Info.cpp Initial File



CD_Info.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include "CD_Info.h"
using namespace std;

// Create a CD object from one line of a CSV file
// containing CD info.
CD* create_CD(string* info)
{
    string title = info[0];

    // Skip over any leading spaces
    while (isspace(title[0]))
    {
        title = title.substr(1);
    }
```

CD_Info.cpp (continued)

```
string id = info[1];
string artist = info[2];
string manufacturer = info[3];
int year;

istringstream(info[4]) >> year;
Recording_Technology recording_technology;

recording_technology = Parse_Rec_Tech(info[5]);

CD* cd = new CD(title, id, artist, manufacturer,
                year, recording_technology);

// Add code to get track info
return cd;
}
```

CD_Info.cpp (continued)

```
// Read a CSV file of CD info and create the corresponding CDs
// at location specified by the caller.
// Return count of CDs created.
int get_CDs(CD** cds, int max)
{
    ifstream cds_file;
    string cds_filename;
    int count = 0;

    // Get filename for CD Info file
    cout << "Collection Filename: ";
    getline(cin, cds_filename);
    cds_file.open(cds_filename.c_str());

    if (!cds_file.is_open())
    {
        cout << "Failed to open file " << cds_filename << endl;
        return -1;// Error
    }
```

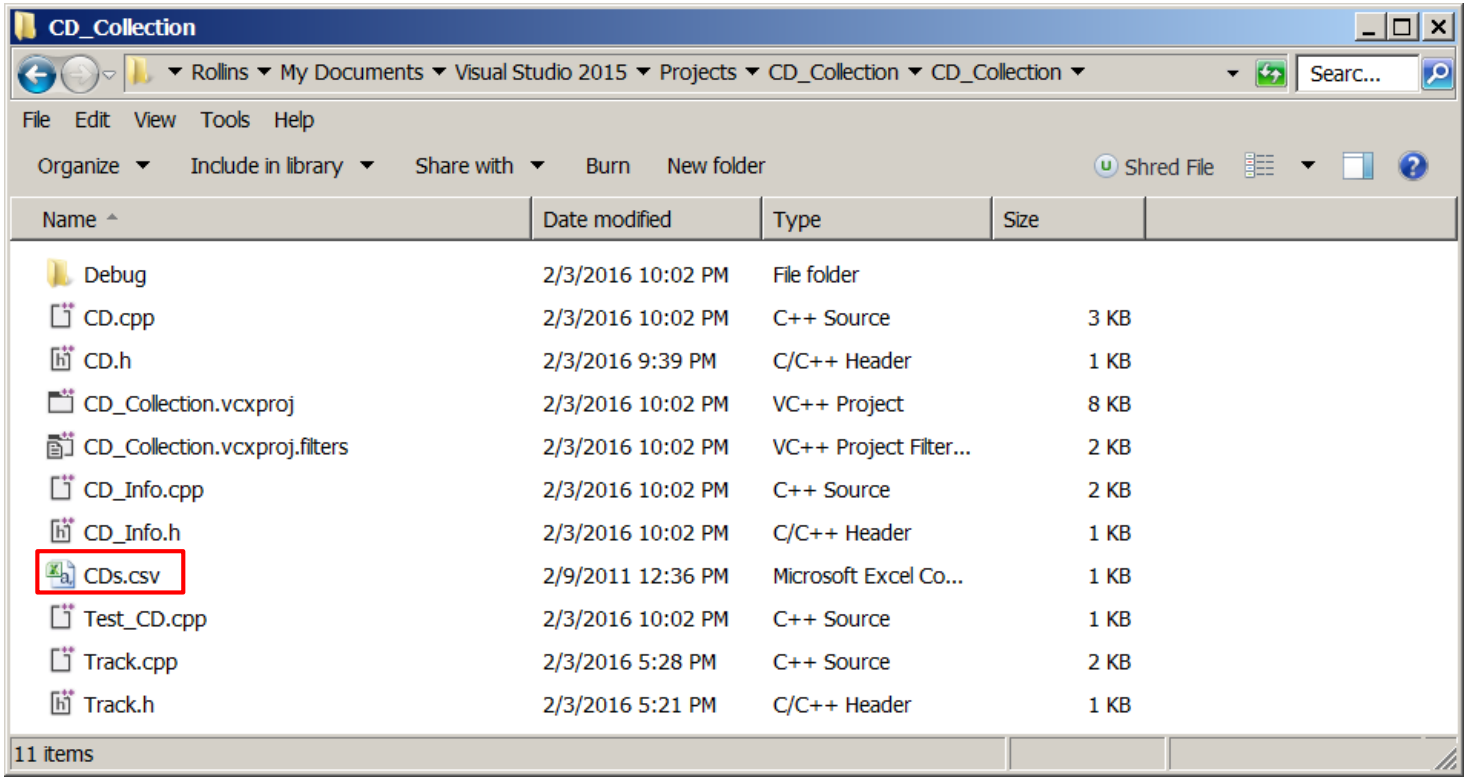
```
// Input file is open
while (cds_file.good() && (count < max))
{
    int i;
    string info[7];
    for (i = 0; i < 7; ++i)
    {
        getline(cds_file, info[i], ',');
        if (!cds_file.good())
        {
            break;
        }
    }

    if (i == 7)
    {
        CD* next_cd = create_CD(info);
        next_cd->Display();     // Temporary test code
        cds[count] = next_cd;
        ++count;
    }
}
```

CD_Info.cpp (continued)

```
    if (cds_file.eof())
    {
        cout << endl << "End of file \n";
    }
    else
    {
        cout << endl << "Error reading file\n";
    }
    cds_file.close();
    return count;
}
```

Testing CD_Info

- In order to test CD_Info we need a CSV file with CD information in the default directory.
- Copy file CDs.csv (from the Downloads area) into the project directory
  - along with the source files



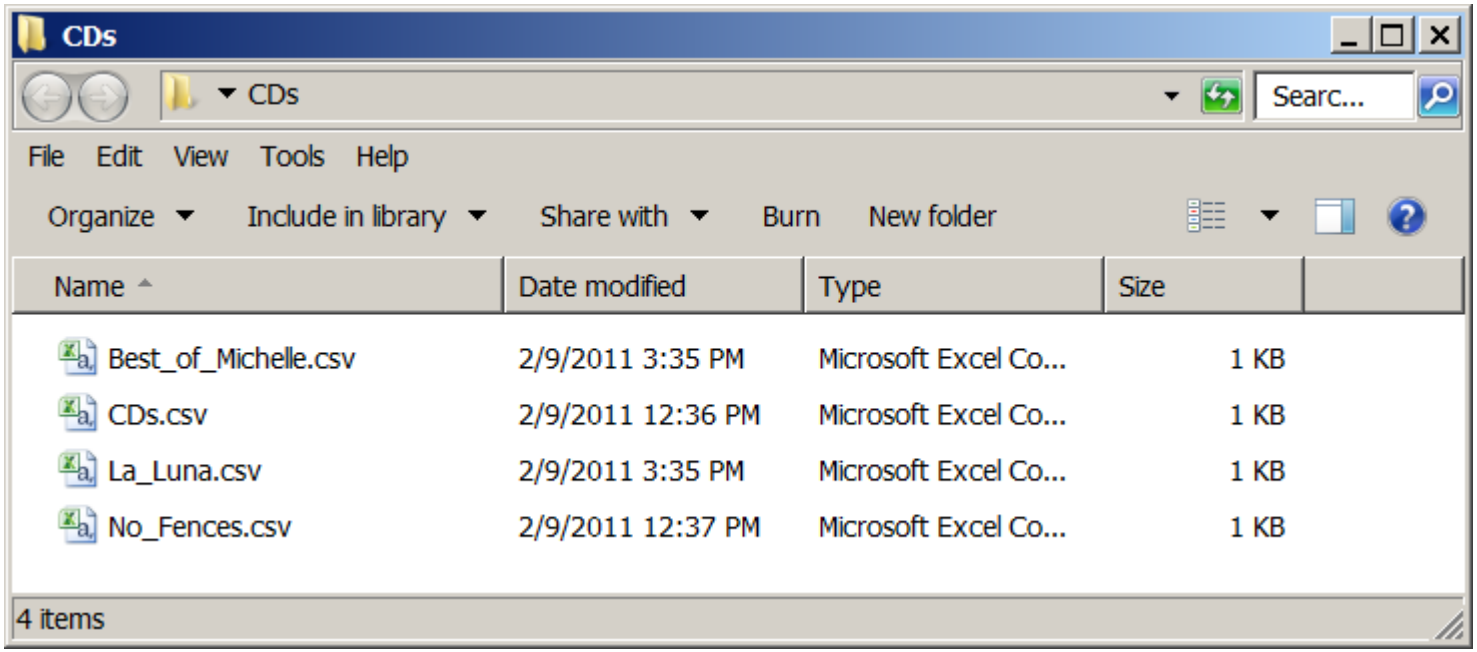
Build and Run

C:\Users\Rollins\Documents\Visual Studio 2015\Projects\CD_Collection\Debug\CD...

```
My CDs:
Collection Filename: CDs.csv
CD: No Fences
ID: ASIN B00005176A
Artist: Garth Brooks
Mfgr: Capital
Year: 1990
Recording technology: ADD
Total play time: 0:00

CD: La Luna
ID: ISBN 2435-57028-2
Artist: Sara Brightman
Mfgr: Angel
Year: 2000
Recording technology: DDD
Total play time: 0:00

CD: Best of Michelle
ID: 24353-16712
Artist: Michelle
Mfgr: EMI
Year: 2001
Recording technology: UNK
Total play time: 0:00


End of file
```

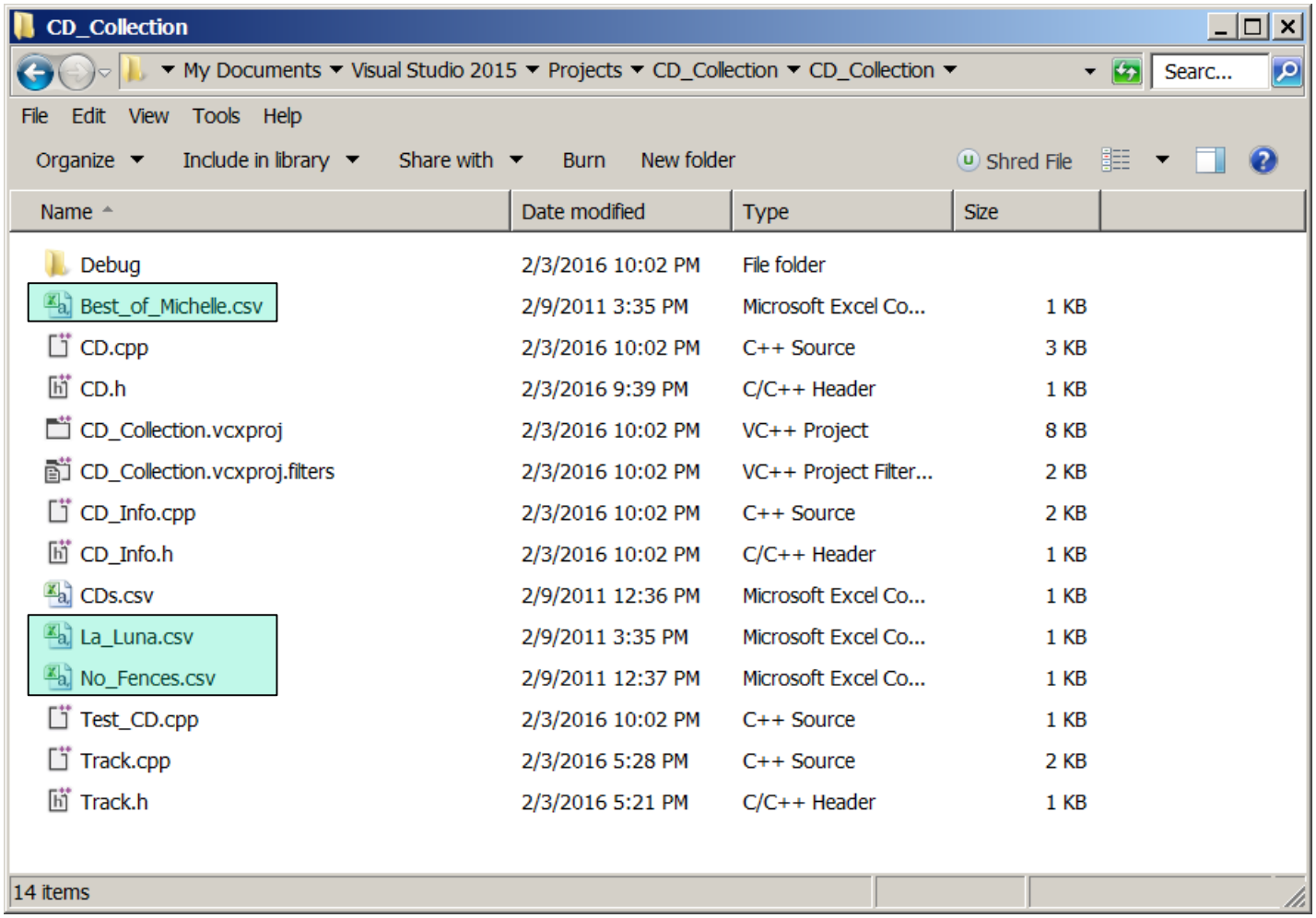
CD Info Files

- We have three CD Information files from the Downloads site.



Copy these into the project directory.

Project Directory with CD Info Files



La_Luna.csv in Excel

Microsoft Excel - La_Luna.csv

File Edit View Insert Format Tools Data Window Help Adobe PDF

C:\Documents and Settings\Rollins\Desktop\CDs\La_Luna.csv

A17

	A	B	C	D	E	F	G
1	La Lune	2:53	Sarah Brightman	La Luna	Classical		
2	Winter In July	4:32	Sarah Brightman	La Luna	Classical		
3	Scarborough Fair	4:11	Sarah Brightman	La Luna	Classical		
4	Beethoven: Figlo Perduto	4:37	Maria Callas: Orchestra	La Luna	Classical		
5	A Whiter Shade Of Pale	3:39	Sarah Brightman	La Luna	Classical		
6	He Doesn't See Me	4:29	Sarah Brightman	La Luna	Classical		
7	Serenade	1:16	Sarah Brightman	La Luna	Classical		
8	How Fair This Place	2:10	Sarah Brightman	La Luna	Classical		
9	Hijo De La Luna	4:28	Sarah Brightman	La Luna	Classical		
10	Here With Me	5:24	Sarah Brightman	La Luna	Classical		
11	La Califfa	2:47	Sarah Brightman	La Luna	Classical		
12	This Love	6:12	Sarah Brightman	La Luna	Classical		
13	Solo Con Te	3:07	Sarah Brightman	La Luna	Classical		
14	Gloomy Sunday	3:48	Sarah Brightman	La Luna	Classical		
15	La Luna	5:00	Sarah Brightman	La Luna	Classical		
16	Heaven is Here	6:24	Sarah Brightman	La Luna	Classical		
17							
18							

La_Luna

Ready NUM

La_Luna.csv in Notepad

C:\Users\Rollins\Documents\Visual Studio 2015\Projects\CD_Collection\CD_Collection\La_Luna.csv - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

La_Luna.csv

```
La Lune,2:53,Sarah Brightman,La Luna,Classical,
Winter In July,4:32,Sarah Brightman,La Luna,Classical,
Scarborough Fair,4:11,Sarah Brightman,La Luna,Classical,
Beethoven: Figlo Perduto,4:37,Maria Callas: Orchestra Del Teatro All Scala,I
A Whiter Shade Of Pale,3:39,Sarah Brightman,La Luna,Classical,
He Doesn't See Me,4:29,Sarah Brightman,La Luna,Classical,
Serenade,1:16,Sarah Brightman,La Luna,Classical,
How Fair This Place,2:10,Sarah Brightman,La Luna,Classical,
Hijo De La Luna,4:28,Sarah Brightman,La Luna,Classical,
Here With Me,5:24,Sarah Brightman,La Luna,Classical,
La Califfa,2:47,Sarah Brightman,La Luna,Classical,
This Love,6:12,Sarah Brightman,La Luna,Classical,
Solo Con Te,3:07,Sarah Brightman,La Luna,Classical,
Gloomy Sunday,3:48,Sarah Brightman,La Luna,Classical,
La Luna,5:00,Sarah Brightman,La Luna,Classical,
Heaven is Here,6:24,Sarah Brightman,La Luna,Classical,
```

Normal text file | length : 921 lines : 18 | Ln : 1 Col : 1 Sel : 0 | 0 | Dos\Windows | ANSI | INS

Genre

- We will need a function to convert a string representing a genre into a Genre enum.
  - Just as we did for Recording_Technology

- Add to Track.h

```
enum Genre { Classical, Pop, Country, Folk, Rap, Hip_Hop, Unknown };
string& ToString(const Genre& genre);
ostream& operator<<(ostream& os, const Genre& genre);
Genre Parse_Genre(const string& str);
```

Add to Track.cpp

```
Genre Parse_Genre(const string& str)
{
        if (str == "Classical") return Classical;
        if (str == "Pop") return Pop;
        if (str == "Country") return Country;
        if (str == "Folk") return Folk;
        if (str == "Rap") return Rap;
        if (str == "Hip_Hop") return Hip_Hop;
        return Unknown;
}
```

Track Infomation

- Add files to get track info and create Track object.
  - Track_Info.h
  - Track_Info.cpp

Track_Info.h

```
#pragma once
#include <string>
#include "CD.h"

// Read track info from specified CSV file.
// Create a Track object and add it to the CD
// specified by the first parameter.
void get_tracks(CD& cd, string& track_file_name);
```

Track_Info.cpp

```
#include <iostream>
#include <sstream>
#include <fstream>
#include "Track.h"
#include "CD.h"

using namespace std;
```

Track_Info.cpp

```
// Read track info from specified CSV file.
// Create a Track object and add it to the CD
// specified by the first parameter.
void get_tracks(CD& cd, string& track_file_name)
{
    ifstream track_file;
    int count = 0;
    track_file.open(track_file_name.c_str());

    if (!track_file.is_open())
    {
        cout << "Failed to open track file "
             << track_file_name << endl;
        return;
    }
```

Track_Info.cpp (Continued)

```
    // Input file is open
    while (track_file.good())
    {
        int i;
        string info[5];
        for (i = 0; i < 5; ++i)
        {
            getline(track_file, info[i], ',');
            if (!track_file.good())
            {
                break;
             }
        }

        if (i == 5)
        {
            Track* track = create_track(info);
            cd.Add_Track(track);
        }
    }
}
```

Add to Track_Info.cpp above get_tracks()

```
// Create a Track object from the strings in an array
// passed by the caller
Track* create_track(string* info)
{
    string title = info[0];
    // Skip over any leading white space.
    while (isspace(title[0]))
    {
        title = title.substr(1);
    }
    int play_time = get_seconds(info[1]);
    string artist = info[2];
    string genre_str = info[4];
    Genre genre = Parse_Genre(genre_str);

    Track* track = new Track(title, play_time, artist, genre);
    return track;
}
```

Add to Track_Info.cpp above create_track

```
// Convert a string representing minutes and seconds
// into an integer representing total seconds.
// Parameter time_str will be of the form 3:45
int get_seconds(string& time_str)
{
    size_t pos = time_str.find(':');
    string minutes_str = time_str.substr(0, pos);
    string seconds_str = time_str.substr(pos + 1);
    int minutes = 0;
    int seconds = 0;
    istringstream(minutes_str) >> minutes;
    istringstream(seconds_str) >> seconds;
    return minutes * 60 + seconds;
}
```

Add to CD_Info.cpp

```
#include "Track_Info.h"
```

At end of Function create_CD()

```
    // Add code to get track info
    string track_file_name = info[6];
    get_tracks(*cd, track_file_name);
    return cd;
}
```

Build and Run

C:\Users\Rollins\Documents\Visual Studio 2015\Projects\CD_Collection...

```
My CDs:
Collection Filename: CDs.csv
CD: No Fences
ID: ASIN B00005176A
Artist: Garth Brooks
Mfgr: Capital
Year: 1990
Recording technology: ADD
Total play time: 34:35

        Track 1: The Thunder Rolls
        Play time: 3:45
        Artist: Garth Brooks
        Genre: Country

        Track 2: New Way To Fly
        Play time: 3:55
        Artist: Garth Brooks
        Genre: Country

        Track 3: Two Of A Kind Workin On A Full House
        Play time: 2:33
        Artist: Garth Brooks
        Genre: Country

        Track 4: Victim Of The Game
        Play time: 3:07
        Artist: Garth Brooks
        Genre: Country

        Track 5: Friends In Low Places
        Play time: 4:20
        Artist: Garth Brooks
        Genre: Country

        Track 6: Wild Horses
        Play time: 3:10
        Artist: Garth Brooks
        Genre: Country

        Track 7: Unanswered Prayers
        Play time: 3:25
        Artist: Garth Brooks
        Genre: Country
```