

Project #02: Chicago Crime Analysis

Complete By: Wednesday, February 7th @ 11:59pm

Assignment: C++ program to analyze Chicago Crime data

Policy: Individual work only, late work **is** accepted (see “Policy” section on last page for more details)

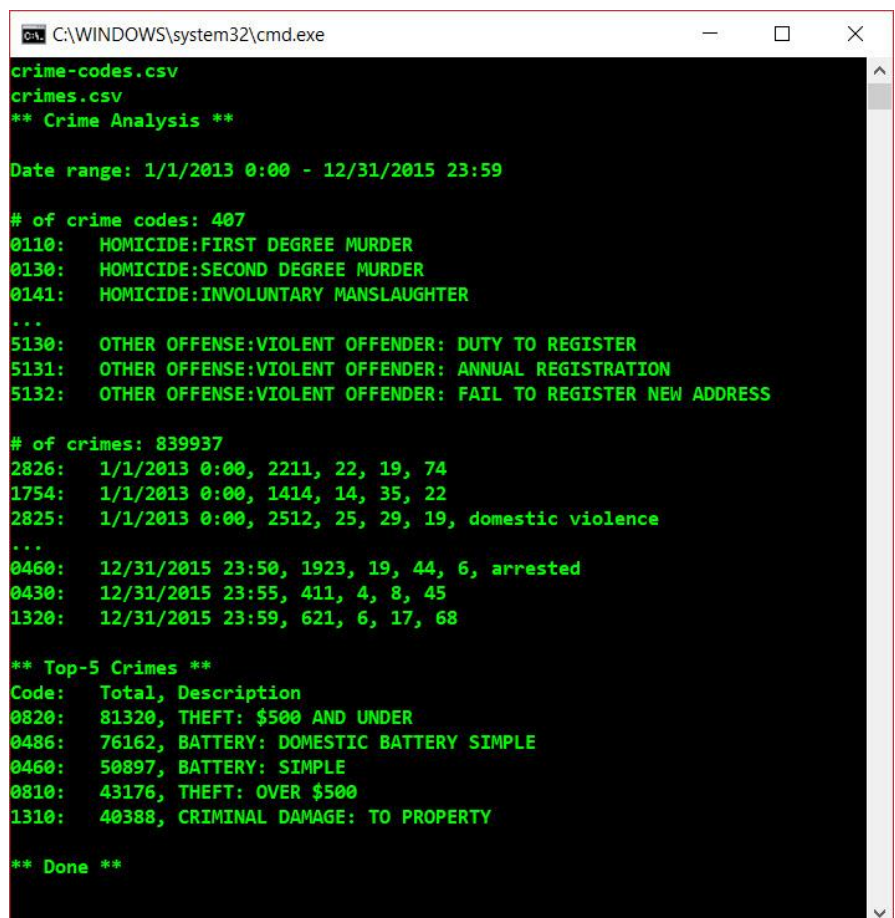
Submission: online via repl.it (exercise P02 Crime Analysis)

Assignment

The assignment is to input Chicago Crime data and perform a simple analysis of the top-5 crimes. There are 2 input files, both in CSV format: (1) “crime-codes.csv” defines a standard set of crime codes, and (2) “crimes.csv” contains the crimes reported in Chicago. Your job is to write a modern C++ program to input this data, organize it into a collection of objects, and compute the top-5 crimes using modern C++ techniques.

Here’s a screenshot for the input files “crime-codes.csv” and “crimes.csv”. Your program should match this output exactly.

In the screenshot, the program starts by inputting the names of the input files (crime-codes first, then the crimes file). Then the crime codes are output: only the first 3 and last 3 crime codes are output. This gives us confidence that the data was input correctly, without flooding the console with output. Likewise, the first 3 and last 3 crimes are output. Finally, the top-5 crimes are output in descending order by #.



```
C:\WINDOWS\system32\cmd.exe
crime-codes.csv
crimes.csv
** Crime Analysis **

Date range: 1/1/2013 0:00 - 12/31/2015 23:59

# of crime codes: 407
0110:  HOMICIDE:FIRST DEGREE MURDER
0130:  HOMICIDE:SECOND DEGREE MURDER
0141:  HOMICIDE:INVOLUNTARY MANSLAUGHTER
...
5130:  OTHER OFFENSE:VIOLENT OFFENDER: DUTY TO REGISTER
5131:  OTHER OFFENSE:VIOLENT OFFENDER: ANNUAL REGISTRATION
5132:  OTHER OFFENSE:VIOLENT OFFENDER: FAIL TO REGISTER NEW ADDRESS

# of crimes: 839937
2826:  1/1/2013 0:00, 2211, 22, 19, 74
1754:  1/1/2013 0:00, 1414, 14, 35, 22
2825:  1/1/2013 0:00, 2512, 25, 29, 19, domestic violence
...
0460:  12/31/2015 23:50, 1923, 19, 44, 6, arrested
0430:  12/31/2015 23:55, 411, 4, 8, 45
1320:  12/31/2015 23:59, 621, 6, 17, 68

** Top-5 Crimes **
Code:  Total, Description
0820:  81320, THEFT: $500 AND UNDER
0486:  76162, BATTERY: DOMESTIC BATTERY SIMPLE
0460:  50897, BATTERY: SIMPLE
0810:  43176, THEFT: OVER $500
1310:  40388, CRIMINAL DAMAGE: TO PROPERTY

** Done **
```

Input Files

The input files are text files in **CSV** format — i.e. comma-separated values. The file “crime-codes.csv” defines a standard set of crime codes along with human-readable descriptions. The format is as follows:

```
IUCR,PRIMARY_DESCRIPTION,SECONDARY_DESCRIPTION
1025,ARSON,AGGRAVATED
1090,ARSON,ATTEMPT ARSON
1010,ARSON,BY EXPLOSIVE
.
.
```

The first line contains column headers, and should be ignored. The data starts on line 2, and each data line contains 3 values: an **IUCR** (Illinois Uniform Crime Reporting) code, a **primary** description of that code, along with a **secondary** description. For example, crime code “1025” denotes the crime of “Arson”, in particular “Aggravated Arson”. Note that the IUCR crime code is a 4-character string, not an integer code as you might expect; if you scroll through the file, you’ll see crime codes along the lines of “041B” and “501H”. Assume the file is ordered by primary and secondary description; the size of the file is unknown and may change.

The second input file “crimes.csv” contains info about crimes reported in Chicago. The format is as follows:

```
DateTime,IUCR,Arrest,Domestic,Beat,District,Ward,Community,Year
1/1/2013 0:00,2826,FALSE,FALSE,2211,22,19,74,2013
1/1/2013 0:00,1754,FALSE,FALSE,1414,14,35,22,2013
1/1/2013 0:00,2825,false,true,2512,25,29,19,2013
.
.
```

The first line contains column headers, and should be ignored. The data starts on line 2, and each data line contains 9 values: the **data & time** the crime occurred (string), the **IUCR** crime code (string), whether an **arrest** was made (TRUE or FALSE), whether the crime represents **domestic violence** (TRUE or FALSE, either all uppercase or all lowercase), where the crime occurred — the police **beat**, the city **district**, the voting **ward**, and Chicago **community** — and finally, the year. These last 5 values are all integers. Assume the file is ordered by date & time; the size of the file is unknown and may change. [**NOTE**: you do not have to store all data, parse and store only what you need.]

The input files are available on the course web page under “Projects”, then “[project02-files](#)”. When you download from dropbox, don’t click on the file but instead click “...” to the right of the filename and select “download”. Three files are being made available:

crime-codes.csv

crimes.csv

crimes-2.csv

The first 2 files — “crime-codes.csv” and “crimes.csv” — were used to produce the screenshot shown on page 1. The “crimes-2.csv” file is a larger version covering the 2001-2015 time period; this is being made available to test the efficiency of your approach on a larger dataset --- the goal is an execution time < 2 minutes (assuming optimized code). All three files are text files and can be opened in a text editor (e.g. Notepad) or a spreadsheet program (e.g. Excel); if you open in Excel, ****don’t save**** changes when you close.

Requirements

As will be the case for all homework assignments in CS 341, how you solve the problem is just as important as simply getting the correct answers. You are required to solve the problem the “proper” way, which in this case means using modern C++ techniques: classes, objects, built-in generic algorithms and data structures, foreach-based iteration, etc. It’s too easy to fall back on existing habits to just “get the homework done.”

In this assignment, your solution is required to do the following:

- Use `ifstream` to open / close the input files
- Use classes to store the input — at the very least a class for each crime code (IUCR, descriptions), and a class for each crime (code, date & time, where it occurred, etc.). These classes don’t need to be in separate .h files; it may be easier to just place them at the top of `main.cpp`.
- Use `std::vector<T>` or any of the built-in containers to store the objects; note that `std::vector<T>` is sufficient for this assignment if you think carefully about your solution
- If you need to sort, use the built-in `std::sort` algorithm
- If you need to search, use a built-in search function such as `std::find_if`
- Loop using range-based for whenever possible, though index-based loops can (and should) be used where it makes sense (e.g. outputting top-5 crimes)
- No explicit pointers of any kind — no `char *`, no `NULL`, no C-style pointers
- Execution time < 2 minutes on the larger “crimes-2.csv” input file; this assumes optimized code, e.g. “-O” option with g++ or “Release Mode” in Visual Studio

Writing code that is not modern C++ will result in a score of 0. Using an array or a linked-list instead of a modern C++ container? 0. No classes? 0. Writing your own sort function or search function? 0.

Programming Environment

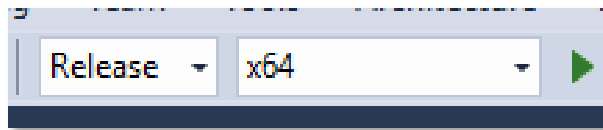
To facilitate testing and program submission, we’ll be using the repl.it system; see the exercise titled “**P02 Crime Analysis**”. You can work entirely within repl.it, or in the C++ environment of your choice; Visual Studio is not required for this assignment. Any environment with a modern (C++11 or newer) compiler will suffice; note that **bert** and **ernie** do not support modern C++, while **bertvm** does. If using g++, you may need to supply the compiler option “-std=c++11” or “-std=c++14”.

If you would like to start using Visual Studio 2017, the course web site contains a “Getting Started with Visual Studio” [handout](#) (or see course web [page](#), under “Miscellaneous”).

Working in Visual Studio?

When compiling, Visual Studio defaults to “debug” mode, which is un-optimized code. The input files are large enough that the difference between optimized and un-optimized code is significant — from minutes of execution time to just seconds when running an optimized program. The good news is that optimization

requires just a simple switch from “Debug” mode to “Release” mode via a drop-down on the toolbar:



Notice the other drop-down shown above is set to “x64” — this configures Visual Studio to generate a 64-bit program. You may need to switch from “x86” (32-bit) to “x64” (64-bit) mode in order to process the larger “crimes-2.csv” file.

Getting Started

Recall that the lecture [notes](#) from the first day of class discussed modern C++ techniques for basic data processing: classes, objects, file input, parsing CSV files, sorting, and iteration. You should be using **all** of these techniques in your solution. The lecture notes are available on the course web page, and the lecture recordings are available on Blackboard.

The best advice I can give — especially when learning a new language — is to start slow. Build the program one step at a time, from a working state to the next working state. This assignment is designed to be solved in steps:

1. Input the file names.
2. Input data from crime codes file.
3. Output total # of crime codes — correct?
4. Output first 3 and last 3 crime codes — does it match input file?
5. Sort by crime code
6. Output first 3 and last 3 crime codes — does it match screenshot?
7. Input data from crimes file.
8. Output total # of crimes — does it match screenshot?
9. Output date range (recall crimes are in order by date), so just output first and last date
10. Output first 3 and last 3 crimes — does it match screenshot? For each crime, note that the screenshot is outputting the date & time, district, beat, ward, and community, followed by “arrested” if that field was TRUE, followed by “domestic violence” if that field was TRUE
11. Compute and output top-5 crimes...

```
C:\WINDOWS\system32\cmd.exe
crime-codes.csv
crimes.csv
** Crime Analysis **

Date range: 1/1/2013 0:00 - 12/31/2015 23:59

# of crime codes: 407
0110:  HOMICIDE:FIRST DEGREE MURDER
0130:  HOMICIDE:SECOND DEGREE MURDER
0141:  HOMICIDE:INVOLUNTARY MANSLAUGHTER
...
5130:  OTHER OFFENSE:VIOLENT OFFENDER: DUTY TO REGISTER
5131:  OTHER OFFENSE:VIOLENT OFFENDER: ANNUAL REGISTRATION
5132:  OTHER OFFENSE:VIOLENT OFFENDER: FAIL TO REGISTER NEW ADDRESS

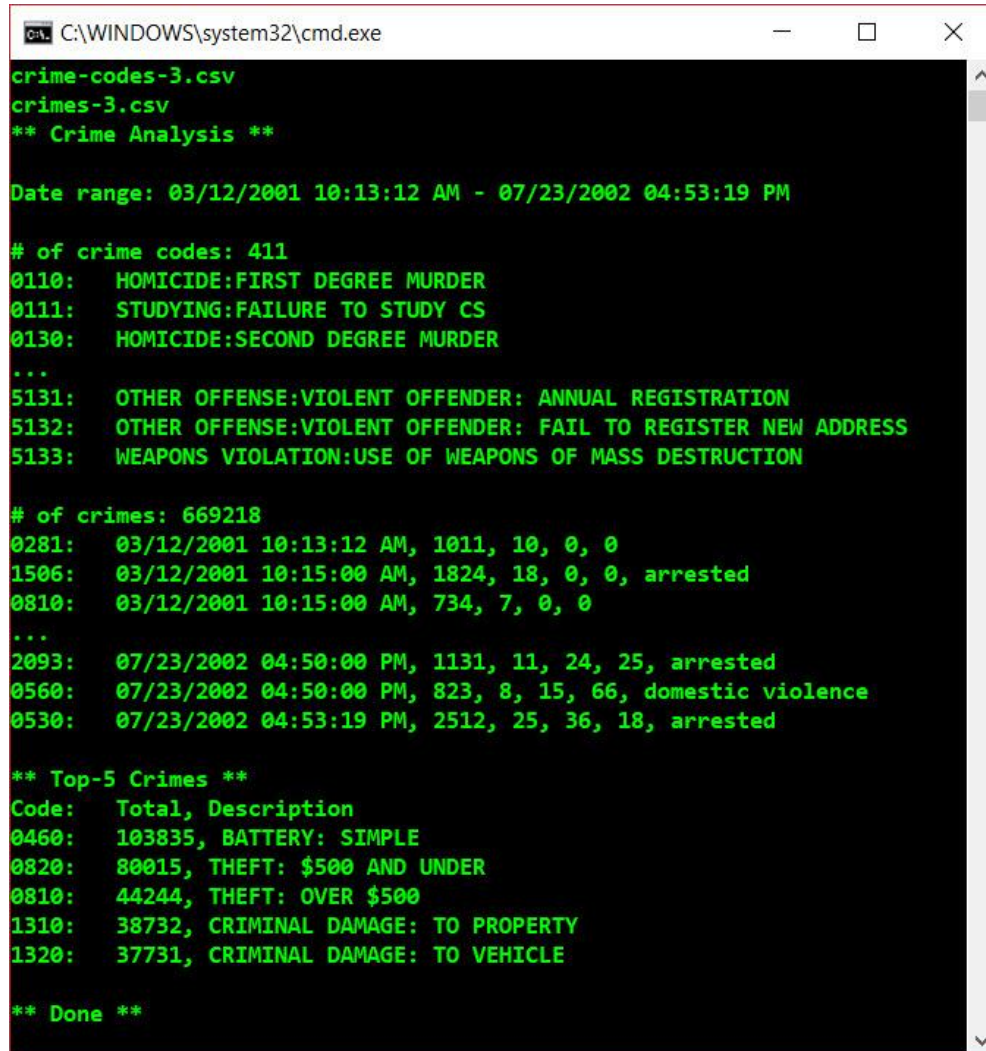
# of crimes: 839937
2826:  1/1/2013 0:00, 2211, 22, 19, 74
1754:  1/1/2013 0:00, 1414, 14, 35, 22
2825:  1/1/2013 0:00, 2512, 25, 29, 19, domestic violence
...
0460:  12/31/2015 23:50, 1923, 19, 44, 6, arrested
0430:  12/31/2015 23:55, 411, 4, 8, 45
1320:  12/31/2015 23:59, 621, 6, 17, 68

** Top-5 Crimes **
Code:  Total, Description
0820:  81320, THEFT: $500 AND UNDER
0486:  76162, BATTERY: DOMESTIC BATTERY SIMPLE
0460:  50897, BATTERY: SIMPLE
0810:  43176, THEFT: OVER $500
1310:  40388, CRIMINAL DAMAGE: TO PROPERTY

** Done **
```

Another test case

Here's the correct output for the input files "crime-codes-3.csv" and "crimes-3.csv":



```
C:\WINDOWS\system32\cmd.exe
crime-codes-3.csv
crimes-3.csv
** Crime Analysis **

Date range: 03/12/2001 10:13:12 AM - 07/23/2002 04:53:19 PM

# of crime codes: 411
0110:  HOMICIDE:FIRST DEGREE MURDER
0111:  STUDYING:FAILURE TO STUDY CS
0130:  HOMICIDE:SECOND DEGREE MURDER
...
5131:  OTHER OFFENSE:VIOLENT OFFENDER: ANNUAL REGISTRATION
5132:  OTHER OFFENSE:VIOLENT OFFENDER: FAIL TO REGISTER NEW ADDRESS
5133:  WEAPONS VIOLATION:USE OF WEAPONS OF MASS DESTRUCTION

# of crimes: 669218
0281:  03/12/2001 10:13:12 AM, 1011, 10, 0, 0
1506:  03/12/2001 10:15:00 AM, 1824, 18, 0, 0, arrested
0810:  03/12/2001 10:15:00 AM, 734, 7, 0, 0
...
2093:  07/23/2002 04:50:00 PM, 1131, 11, 24, 25, arrested
0560:  07/23/2002 04:50:00 PM, 823, 8, 15, 66, domestic violence
0530:  07/23/2002 04:53:19 PM, 2512, 25, 36, 18, arrested

** Top-5 Crimes **
Code:  Total, Description
0460:  103835, BATTERY: SIMPLE
0820:  80015, THEFT: $500 AND UNDER
0810:  44244, THEFT: OVER $500
1310:  38732, CRIMINAL DAMAGE: TO PROPERTY
1320:  37731, CRIMINAL DAMAGE: TO VEHICLE

** Done **
```

Submission

The program is to be submitted via <http://repl.it>: see the exercise "P02 Crime Analysis". The grading scheme is 90% correctness and 10% style. You must pass all test cases to obtain a score of 90/100; if you fail the test cases we will still award partial credit as appropriate. We will likely perform additional testing by running your program outside of repl.it (e.g. to test against the larger "crimes-2.csv" file). Your program's execution time should be < 2 minutes assuming optimized code (e.g. "-O" option with g++ or "Release mode" in Visual Studio).

For style, we expect basic commenting, indentation, and readability. You should also be using functions, parameter passing, and good naming conventions. Add your name to the header comment at the top of each

.cpp and .h file. Here's an example of what we consider an appropriate header comment (at least in "main.cpp"):

```
//  
// Chicago Crime Analysis  
//  
// <<YOUR NAME HERE>>  
// U. of Illinois, Chicago  
// CS341, Spring 2018  
// Project 02  
//
```

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 25%. After 24 hours, no submissions will be accepted.

All work is to be done individually — group work is not allowed. While I encourage you to talk to your peers and learn from them (e.g. via Piazza), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is described here:

<http://www.uic.edu/depts/dos/docs/Student%20Disciplinary%20Policy.pdf> .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at <http://www.uic.edu/depts/dos/studentconductprocess.shtml> .