

COMPSCI 3307A 001 FW20 >

🔍 📄 Assignments and Projects

Assignments and Projects

Assignment - In progress

Complete the form, then choose the appropriate button at the bottom.

Title	CS3307 Individual Assignment
Due	Oct 7, 2020 11:55 pm
Number of resubmissions allowed	Unlimited
Accept Resubmission Until	Oct 7, 2020 11:55 pm
Status	In progress
Grade Scale	Points (max 100.00)
Modified by instructor	Sep 15, 2020 7:58 pm

Instructions

CS3307 Individual Assignment

Fall Session 2020

Purpose of the Assignment

The general purpose of this assignment is to develop some a simple C++ utility for the [Raspberry Pi Desktop](#), or a comparable Linux system, given a number of requirements, making use of the principles and techniques discussed throughout the course. This assignment is designed to give you experience in:

- object-oriented programming using C++, using basic language constructs, classes, and data types
- looking through Linux manual pages and documentation, as you will likely need to do this in your projects later
- getting acquainted with Linux-based programming and services, which will help in project development on this environment later in the course

The assignment is intended to give you some freedom in design and programming the explore the subject matter, while still providing a solid foundation and preparation for the type of work you will later be doing in the group project.

Assigned

Wednesday, September 16, 2020 (please check the main [course website](#) regularly for any updates or revisions)

Due

The assignment is due Wednesday, October 7, 2020 by 11:55pm (midnight-ish) through an electronic submission through the [OWL site](#). If you require assistance, help is available online through [OWL](#).

Late Penalty

Late assignments will be accepted for up to two days after the due date, with weekends counting as a single day; the late penalty is 20% of the available marks per day. Lateness is based on the time the assignment is submitted.

Individual Effort

Your assignment is expected to be an individual effort. Feel free to discuss ideas with others in the class; however, your assignment submission must be your own work. If it is determined that you are guilty of cheating on the assignment, you could receive a grade of zero with a notice of this offence submitted to the Dean of your home faculty for inclusion in your academic record.

What to Hand in

Your assignment submission, as noted above, will be electronically through [OWL](#). You are to submit all source code files, header files, and build files necessary to compile and execute your code. If any special instructions are required to build or run your submission, be sure to include a README file documenting details. (Keep in mind that if the TA cannot run your assignment, it becomes much harder to assign it a grade.)

Assignment Task

Your assignment task is to familiarize yourself with the [Raspberry Pi Desktop](#), or a comparable Linux system, and develop a simple C++ utility for working with COVID-19 data from <https://covid19api.com>. In essence, your utility will load a [CSV](#) file of COVID-19 data from countries around the world, do some sorting of the data and plot some simple terminal-based bar charts of the data. This should give you a good exposure to C++ classes and STL packages.

To complete this assignment, you will need access to [Raspberry Pi Desktop](#), including its C++ compiler and requisite supporting tools, libraries, and packages. (A comparable Linux system would also work.) Your program must run in this environment.

It is likely easiest to build yourself a virtual machine running this system; details on how to do so can be found under Useful Links in the OWL site side bar. You should do this as early as possible to make sure you are set up and ready to go for the assignment. If you have a computer that completely lacks virtualization support, Science Technology Services has a solution for remotely accessing something that is compatible for this work. They have created a cloud based Linux machine running [Raspberry Pi Desktop](#); this can be found at cs3307.gaul.csd.uwo.ca. To access this machine, you should be able ssh to log in from pretty much anywhere, using your Western credentials for access. You can scp/sftp files to and from this machine as necessary.

The Dataset

For this assignment, you will be using a summary dataset that can be accessed using an API here: <https://api.covid19api.com/summary>. From a command prompt, you should be able to pull the latest version of the dataset using the command:

```
wget https://api.covid19api.com/summary
```

This will download the summary data in [JSON format](#). JSON is a nice format but can be painful at times to work with. For this assignment, it will be much easier to convert this to CSV first, as it is more straightforward to parse on

your own. To get this data into a CSV format, a shell script called `json2csv` has been attached to this assignment for this JSON data. Amongst other things, this script uses `jq` to help with the actual conversion. Your [Raspberry Pi Desktop](#) system likely doesn't have it installed by default (a "`which jq`" should confirm this). In such a case, you can grab it using:

```
sudo apt install jq
```

You can then execute the `json2csv` script on the downloaded data to convert it to a CSV format. If you have any difficulties doing this, a sample converted CSV file is attached to this assignment that you can use instead. (Doesn't hurt to try getting fresh data first!)

Modelling and Storing Data Records

To contain the data from the dataset in your program, you will first need to create a custom class to contain a country's record of data from this dataset. This class will need members to contain the country's name, two-letter country code, and six statistical counts (new confirmed cases, new deaths, new recovered cases, total confirmed cases, total deaths, and total recovered cases). The name and country code should be stored using strings and the statistical counts should be stored as numbers. You must provide at least one constructor for initializing objects of this class (either by taking a record as a string and parsing it internally or by taking individual values for each data member) and you must provide a destructor for this as well, even if it has nothing in particular to do.

You will then need some kind of container class like a [vector](#) to store the records. You can create a second class to model this collection of records and support functions; how you want to do this up to you. The records themselves must have a class, as noted above.

Ingesting and Processing the Data

To instantiate and populate the data records from your class, you will need to ingest the data from a CSV file. This will involve [file stream](#) operations as well as [string](#) manipulations, and you might find the [stream representation of strings](#) helpful too. You can either parse the file line by line using [getline](#) or character by character using [get](#). There are also a number of functions available to help tokenize strings. However you do it, please note that you will need to exercise some care in parsing the CSV data as some country names contain a comma; fortunately the text is enclosed in quotation marks (") and that should assist in parsing.

Processing the data is pretty minimal -- you just have to sort it. Your program must be able to sort the data by any of the six statistical counts noted above (new confirmed cases, new deaths, new recovered cases, total confirmed cases, total deaths, and total recovered cases) in either ascending or descending order. While this sounds like it could be a lot to do, I strongly recommend looking at the [sort](#) algorithm. It can do the heavy lifting for you, and you just need to provide methods of comparing things to one another. (This will help you appreciate what the STL has to offer!)

How is your program going to know what to do? What file should it use? How should it process/sort the data? That is somewhat up to you -- you have two choices here:

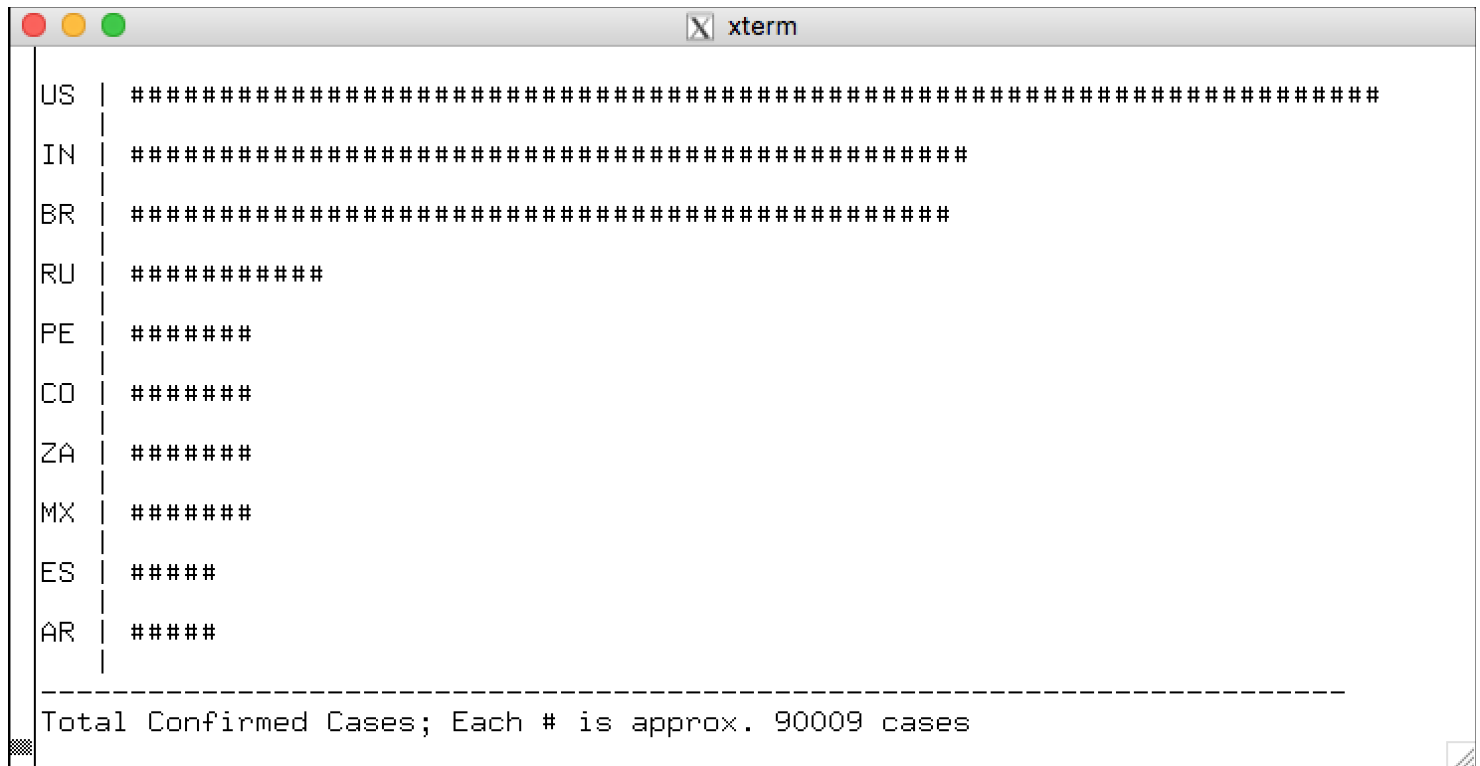
- Take in the file name, the data field to sort by, and the order of sorting (ascending or descending) as command line parameters to your program.
- Provide prompts and a simple menu for selecting options within your program.

Each method has its advantages and disadvantages. Either way you should have at least some minimal error handling to prevent the user from going too far astray. Nothing too fancy, but you should point out issues if the

indicated file does not exist, if they try to pick an invalid data field to sort on or an invalid sort ordering, and so on.

Producing Graphs

After the data has been processed, you need to print out a simple graph and then it's done. For this assignment, you should print out a simple horizontal bar chart for the first 10 records according to the sorted data. (So top 10 or bottom 10, depending on the sort order chosen by the user.) For the chart, you should report the country code of the country in question, as well as a bar of characters (you can choose which one) representing the data field selected as the sorting criteria for the user. For example, if the user asked to sort the data set by total confirmed cases in descending order, you would get something like this:




You should also have a status line/legend as shown in the above image. How do you know how wide to make the bars? Well, first you need to figure out how big to make the longest bar. In the example above, the longest bar was 70 characters and so that was how long the first data bar needed to be for whatever its total confirmed cases value was. All the other bars can be scaled relative to that first bar, using the appropriate data values. (For example, if the second country's data value is half of the first country's data value, the bar should be half as long, right?) Doing this, it shouldn't be too hard to print out a simple horizontal bar chart. (I know, because that's my implementation printing out the graph in the image!)

Implementation Notes

For this assignment, every class should be captured in two files: one header and one code file. The main function should be in a file separate from the classes. (So you will have at least three files in this assignment: a header and a code file for your data record class, and one containing the main function. You may have other files for other classes and other support functions and such.)

Your code should also be written in adherence to the [Coding Guidelines available here](#). Deviations will result in deductions from your assignment grade up to 10% of its overall value.

Additional resources for assignment

- [json2csv](#) (1 KB; Sep 13, 2020 11:00 pm)
-  [summary.csv](#) (15 KB; Sep 13, 2020 11:06 pm)

Submission

Assignment Text

This assignment allows submissions using both the text box below and attached documents. Type your submission in the box below and/or use the Browse button or the "select files" button to include other documents. **Save frequently while working.**

Source			
Styles ▾	Format ▾	Font ▾	Size ▾ ▾

Words: 0, Characters (with HTML): 0/1000000 ▲

Attachments

No attachments yet

Select a file from computer No file chosen

or select files from workspace or site

Don't forget to save or submit!

- [Gateway](#)
- [Help & Support](#)
- [Western University](#)

OWL is the learning management system of Western University. It is a customized version of Sakai.
Copyright 2003-2020 The Apereo Foundation. All rights reserved.



•

OWL - OWL - Sakai 11.3-owl5.1 - Server azuki24