COMP1551 Core Programming

Coursework 3

Introduction

Your task is to write a C++ program that analyzes meteorological data. You have three different options for your solution, offering progressively greater difficulty (and higher marks). The assignment is worth 10% of your overall module grade.

Preparation

1. Download `cwk3files.zip` from the VLE and put it into a directory that you have created for this coursework.
2. Unpack the Zip archive, e.g., by using `unzip cwk3files.zip` on the command line. This will give you a sample data file called `sheffield.data`, plus two other files that will help with the graph plotting in the advanced solution (see below).
3. Examine the contents of `sheffield.data`. This is part of a Met Office historical dataset from a meteorological station in Sheffield. It records minimum and maximum temperature (in °C), days of air frost, rainfall (in mm) and number of hours of sunshine for each month in each year from 1930 up to 2012. (The data were obtained from `data.gov.uk`.)

   You code will need to read and process the data in this file. Make sure that you are familiar with file I/O and with storing data in containers by doing the relevant exercises *before* you attempt this coursework.

Basic Solution

This is worth **22 marks**, out of a possible 40 marks.

1. For 10 marks, create a file called `analyze.cpp` containing a program that
   - Opens `sheffield.data` and skips the header lines
   - Uses a loop to read records from the file
   - Stores the rainfall measurements in a suitable container
2. For 6 marks, add code that uses the data stored in the container to find the wettest and driest years in the dataset. In each case, your program should display not only the year but also the total rainfall recorded during that year.

Note that `getline` will be useful for skipping the header lines. The easiest way of reading a data record is to have a set of variables capable of holding each of the values in the record. You can chain together stream extraction operations to read the values from one record into these variables.

The features above are worth a total of 16 marks. There are a further 3 marks for correct program execution and the remaining 3 marks are for coding style and commenting.

Intermediate Solution

An intermediate solution should extend the basic solution by having the following additional features, worth a further **10 marks**:

- The program should find the year and month in which the highest number of days of air frost were recorded. Year, month and number of days should all be displayed. Month should be displayed as a word rather than as a number—i.e., `July` rather than `7`.

- The program should handle different data files by having the filename specified as a command line argument. It should terminate with an appropriate usage hint if no filename has been given.
- Functions should be used to give more structure to the solution.

Advanced Solution

An advanced solution should extend the intermediate solution by adding functions to generate the following fully-labelled data plots:

- A bar chart showing monthly rainfall averaged over all years in the dataset
- A line plot of maximum temperature as a function of time
- A scatter plot of maximum temperature (y axis) against hours of sunshine (x axis)

The main program should prompt the user to choose one of these three options and then should call the appropriate function in order to generate the plot. The functions and the enhancements to the main program are worth a total of **8 marks**.

To help you with the plotting, we have provided you with `gnuplot_i.hpp`. This header file defines a class `Gnuplot`, providing a C++ interface to the popular plotting program gnuplot. The example program in `gnuplot-example.cpp` shows you how to use the `Gnuplot` class. You can compile and run this example in the normal way on School of Computing machines. If you are using your own PC, note that you will need to install the gnuplot program first.

Note: we recommend doing this on Linux or Mac OS X; we cannot provide advice or support on getting it working with MS Windows.

Further information on gnuplot itself is available at `http://www.gnuplot.info`.

Submission

Submit your code as a **single Zip archive**, via the link provided in the VLE. This Zip archive should include the following items:

- The main program file `analyze.cpp`
- Any other source files that you created as part of your solution
- A makefile that will compile your solution, if you implemented it in more than one source file
- A text file called `README.txt` that identifies all the features that you have implemented

Do not include any object code files or executables! Also, make sure that you generate a proper Zip archive, not an archive of any other type (RAR, 7-Zip, tar, gzipped tar, etc).

The deadline for submissions is **10 am on Thursday 5 March 2015**. The standard university penalty of 5% of available marks per day will apply to late work, unless an extension has been arranged due to genuine extenuating circumstances.