Department of Computing and Information Systems

COMP20005 Engineering Computation
Semester 1, 2013
Project A

Learning Outcomes

In this project you will demonstrate your understanding of loops and `if` statements by writing a program that sequentially processes a file of text data. You are also expected to make some use of functions; but because you have not yet been shown how to store large volumes of data (arrays are covered in Chapter 7), your program will be hard to structure elegantly, and there will only be a limited number of ways in which functions can be used. Instead, you will need to declare a relatively large number of variables, so be sure to be systematic in your naming of them, so that you don't make errors.

Time-Series Data

Time-series data is very common in engineering and scientific applications. Measurements from sensors are annotated with time stamps, and then used in a variety of ways, including for mapping and other visualizations. For example, the file `earthquakes-all.txt`, available on the LMS, contains earthquake data from the US Geological Survey.[1] The first six lines of that file are duplicated into a smaller test file `earthquakes-006.txt`:

```
# Source: http://earthquake.usgs.gov/earthquakes/, 16 March 2013
# DateTime,Latitude,Longitude,Depth,Magnitude
2011-01-01T00:02:31+00:00,27.247,143.166,10,5.0
2011-01-01T09:56:58+00:00,-26.803,-63.136,576.8,7.0
2011-01-01T18:37:44+00:00,-49.155,121.555,10,5.0
2011-01-01T23:33:39+00:00,24.653,97.918,21.4,5.0
```

and represent two header lines, describing the fields in the remaining lines, followed by four data lines, describing four earthquakes. The thirteen variables on each of the data lines can be read using:

```
scanf("%4d-%2d-%2d%c%2d:%2d:%2d+%2d:%2d,%lf,%lf,%lf,%lf",
     &yyyy, &mm, &dd, &junkc, &hh, &min, &sec, &junki, &junki,
     &latit, &longi, &depth, &magnitude)
```

with `junkc` and `junki` indicating values of type `char` and `int` that can be immediately discarded; `latit` and `longi` being in degrees relative to the usual meridians (latitude an amount north/south of the equator, and longitude an amount east/west of the Greenwich baseline in England); `depth` being the depth of the earthquake in kilometers; and `magnitude` being the magnitude of the earthquake. The three `junk` fields are not used in this project, and are read and discarded. The file `earthquakes-all.txt` lists all earthquakes that occurred globally between 1 January 2011 and 16 March 2013 of magnitude 5 or greater[2], and contains 4,569 lines, including the header lines. Other shorter data files can be generated by taking prefixes of this data file, as listed on the LMS, including the file `earthquakes-006.txt` which contains the six lines listed above; or by filtering it in other ways.

You may assume that the data is error free in terms of how each row of data is formatted, and that the lines are given in non-decreasing date/time order. You may also assume that there is at least one data line in each file, but will not be penalized if your program checks for the presence of that first line and prints a message if it is not present.

[1] `http://earthquake.usgs.gov/earthquakes/`

[2] A magnitude five earthquake is felt by most people in the affected area, and will give rise to possibly significant damage in poorly constructed buildings, but only minor damage to well-constructed ones.

To get the data into your program you need to execute your program from a command-line, and pipe the file in using input redirection (using <). Instructions on how to do this appear on the LMS. Because the input data will come from a file, you should not print any prompts at all.

Stage 1 – Start Time (marks up to 3/10)

The energy released by an earthquake is (very approximately) given by

$$E = 10^{3M/2-7.2}$$

where M is the magnitude, and E is the energy, measured in TeraJoules, TJ.

Write a program that reads a file of earthquake records in the specified format and prints out the date, time, magnitude, and location, and energy release of the first earthquake described in the file. Use appropriate units for the energy (T = Tera = 10^{12}, P = Peta = 10^{15}, E = Exa = 10^{18}) in your output, and then report it to one decimal place if the number involved (after adjusting for units) is less than 10, and with no decimal places if the adjusted number is between 10 and 999.9. For example, 1.234 TJ would be printed as 1.2 TJ; 123.89 TJ would be printed as 124 TJ; but 12345.9 TJ would be printed as 12 PJ. (Hint – sounds like a specialized energy-printing function to me). For the six sample input lines shown above, your output should be:

```
Stage 1
=======
Records commence: 2011-01-01, 00:02:31
First earthquake: magnitude 5.0 at (27.2,143.2)
Energy released : 2.0 TJ
```

Stage 2 – Overall Statistics (marks up to 7/10)

Extend your program so that it reads right through the data file and prints a summary of the data in the file: the number of earthquakes, details of the strongest one, and the total amount of energy released. If there are two earthquakes with equal maximum magnitude, report the first one.

For this stage, the required output on `earthquakes-006.txt` is:

```
Stage 2
=======
Total earthquakes    : 4
Strongest earthquake : magnitude 7.0 at (-26.8,-63.1)
Took place on        : 2011-01-01, 09:56:58
Total energy released: 2.0 PJ
```

Stage 3 – Monthly Reporting (marks up to 9/10)

Now add in further logic (and another set of variables) so that a monthly report line is generated. That is, for each month, print out the number of earthquakes in that month, and the magnitude of the strongest one. For example, on `earthquakes-500.txt` you should produce something like:

```
Stage 3
=======
2011-01: 201, max 7.2
2011-02: 157, max 6.9
2011-03: 140, max 9.0
```

Your program should correctly handle cases where there are no earthquakes listed for a given month but there are records for months either side of the one(s) missing – in that case the missing months should be printed as having had zero earthquakes and with a maximum magnitude of "`---`". Be sure to test this aspect using your own test data (you can manually edit one of the supplied file to remove lines and/or alter dates, for example) before finalizing your submission; you should also check carefully that the last month in the input is handled correctly.

Unless you are ambitious and make use of arrays to store the data, your output for this stage will appear prior to the Stage 2 output (which will of necessity be the last thing printed by your program). Further (and complete) output examples are linked from the LMS.

Stage 4 – Visualization (marks up to 10/10)

Once you have Stage 3 working correctly, modify the output statements so that a graph of total monthly earthquake energy is also shown, scaled so that no bar is printed out if the monthly energy is 10 TJ or below, and logarithmic thereafter, with each ten characters representing a multiple of ten in terms of monthly total energy. This output will be integrated with the Stage 3 output in a single section; for `earthquakes-500.txt`:

```
Stage 3/4
=========
2011-01: 201, max 7.2 |---------+---------+---------+|
2011-02: 157, max 6.9 |---------+---------+------|
2011-03: 140, max 9.0 |---------+---------+---------+---------+---------+--|
```

Full example output listings for several test files are provided on the LMS page, so that you can confirm your understanding of what is required by this task.

If you want a (totally optional) challenge, further alter your program so that a scale for the graph is printed out after the Stage 3/4 heading; generated in such a manner that it is fully compatible with the `#defines` used to create the graph.

The boring stuff...

This project is worth 10% of your final mark.

You need to submit your program for assessment; detailed instructions on how to do that will be posted on the LMS once submissions are opened. Submission will *not* be done via the LMS; instead you will need to log in to a Unix server and submit your files to a software system known as `submit`. You can (and should) use `submit` **both early and often** – to get used to the way it works, and also to check that your program compiles correctly on our test system, which has some different characteristics to the lab machines. Only the last submission that you make before the deadline will be marked.

You may discuss your work during your workshop, and with others in the class, but what gets typed into your program must be individual work, not copied from anyone else. In particular, note that we have access to sophisticated similarity checking software that undertakes deep structural analysis of C programs, and routinely run an automated check over all student submissions. So, do **not** give hard copy or soft copy of your work to anyone else; do **not** "lend" your memory stick to others; and do **not** ask others to give you their programs "just so that I can take a look and get some ideas, I won't copy, honest". The best way to help your friends in this regard is to say a very firm "**no**" when they ask for a copy of, or to see, your program, pointing out that your "**no**", and their acceptance of that decision, is the only thing that will preserve your friendship.

Deadline: Programs not submitted by **9:00am on Monday 6 May** will lose penalty marks at the rate of two marks per day or part day late. Students seeking extensions for medical or other "outside my control" reasons should email Alistair, `ammoffat@unimelb.edu.au`.

And remember, programming is fun!