Teaching London Computing

Programming for GCSE Topic 10.2: Designing for File I/O

William Marsh
School of Electronic Engineering and Computer Science
Queen Mary University of London

Aims

- Review an example GCSE problem requiring file I/O
- Consider steps in problem solving

Weight Tracker Problem

The following problem is from the OCR GCSE text book (O'Byrne and Rouse, Hodder Education, 2012)

Alan is trying to lose weight and weighs himself every few days. He wants to keep track of these weights and the days they were taken is a file on his computer. Design, code, test and evaluate a system that allows Alan to input a date and a weight and record this in a suitable file on his computer. The system should automatically output previous data when the program is run. The program should advise if there is a weight loss or a weight gain since last time and should output the weight gain (or loss) with a suitable message. It should also output the overall weight change since the first entry in the file.

Weight Tracker: Suggested Steps

1. Start by designing a file and do this by using an editor to type an example (good for testing). Choose a design that is easy to read. How are weights represented? How is the user allowed to format the date (for example, can a date include spaces)?
2. Write a very simple program to read the file: show that you can distinguish the weights from the dates.
3. Time to decide in more detail what the final program is going to do. You could
   - Write the different messages that the program can output
   - Write an example input / output dialog of the program in use.
4. Now design variables to hold the data from the file in the program. Is it necessary to use arrays or can the data from the file be processed when reading the file (this question is quite hard)?
5. Compete the design of the program using pseudo code.

File Designs

- Each entry on one line

```
70.1,Friday 11th
72.2,Monday 14th
71.8,June 15th
```

- Date is just a string
  - May include spaces
  - **Assumes** no date arithmetic
- Weight is a floating point number

Reading the File

```
import io

wf = open("weights.txt", 'w')

line = wf.readline()
fields = line.split(",")

weight = fields[0]
date = fields[1]
```

Reading the File

```
import io

wf = open("weights.txt", 'w')

line = wf.readline()
fields = line.split(",")

weight = fields[0]
date = fields[1]
date = date.split("\n")[0]
```

Design the Dialog

Latest weight: 71Kg
Recorded on: 17th June

What is your current weight? 71.5
What is the date? 19th June

Oh dear!
You have gained 0.5 Kg since 17th June

Your total weight loss since May 1st is 2.3 Kg

File Handling

- How many files?
- Read and/or write?
- Idea 1
  - Read contents of file to an array. Close
  - … then rewrite, with new data
- Idea 2
  - Open file with mode 'a'
  - Read and then write **new** data only
- Idea 3
  - Rename file (how?); read old file, write to new file

Design The Data Structures

- Is it necessary to store all data?
- Without storing all data, variables for
  - First date and weight
  - Most recent date and weight
  - Latest date and weight
- Storing all data
  - 2 lists
  - List of dates
  - List of of weights

Other Issues

- File name
  - What is the file called?
- Errors
  - What errors are detected?
  - Important to include bad files for testing
- Extensions tasks
  - Multiple people – separate file for each
  - Calculate days between dates
  - Statistics and regression – predict future weight

Summary

- Problem solving for a program with data in files includes
  - Contents of the file: design for easy programming
  - User dialog or interface
  - File names and handling: read, write, delete
  - Program variables for holding data
- Pseudo code useful when higher-level decisions made
- Some code can be written before all code specified