05OGDLP - Algorithms & Programming

Cristiano **Pegoraro Chenet**, Riccardo **Smeriglio**

Department of Control and Computer Engineering (DAUIN),
Politecnico di Torino, Italy

October 24, 2024

Laboratory 5

Submission Instructions

For being able to request the additional points for the exam, the exercises of this laboratory must be submitted by Wednesday, 30 October 2024, at 23:59 UTC+2. The file timestamp associated with your submission (automatically provided by the teaching portal) is going to confirm your submission on time.

How to submit this lab

- What to deliver: All the exercises, including the HOMEWORK.
- Where to deliver: "Elaborati" Tab on the portal;
- In which format: A zip file with this name <studentID>_<lab_n>.zip and this structure:
  - *folder* "1_1"
    - cycling_performance_analysis.c
    - cyclist.txt
  - *folder* "1_2"
    - rectangle_sorting_1.c
    - rectangles.in
  - *folder* "2_1"
    - rectangle_sorting_2.c
    - rectangles.in

Learning Objectives

- Structured data type
- Sorting
- Dynamic memory allocation

1 Lab Exercises

1.1 Cycling performance analysis

During a training session, each athlete of a group of professional cyclists is checked during each lap. Each athlete's lap times are stored in a file with the following format. The first line of the file stores the number of cyclists in the group. Then, for each cyclist, the file stores:

- On the first line, their name (a string of 30 characters at most), identifier (integer value), and number of laps performed.
- On the second line, all lap times, time 1 time_2 ... time_N, stored as real values.

Write a program that, after reading the file and storing its content in a proper data structure, is able to reply to the following menu inquiry:

- *List:* The program prints out the number of athletes, their names, identifiers, and the number of laps performed.
- *detail <name>:* Given an athlete's name, the program prints out their identifier and all lap times.
- *best:* The program prints out the name, identifier, all lap times, and the average lap time for the athlete whose average lap time is smaller.
- *stop:* End the program.

Notice that all operations can be performed more than once till the stop command is issued.

Example

Let the following be the input file:

```
4
Rossi 100 3
1.30 1.38 1.29
Bianchi 101 5
1.46 1.43 1.42 1.51 1.28
Neri 117 2
```

```
1.26 1.34
Verdi 89 4
2.01 1.45 1.43 1.38
```

The following is a run example of the program:

```
Input file name: cyclist.txt
Command? list
Number of athletes: 4
Name: Rossi #Id:100 #Laps:3
Name: Bianchi #Id:101 #Laps:5
Name: Neri #Id:117 #Laps:2
Name: Verdi #Id:89 #Laps:4
Command? best
Name:Neri #Id number:117 Laps:2 Times: 1.26 1.34 (Average:1.30)
Command? details Bianchi
#Id:101 #Laps:5 Times: 1.46 1.43 1.42 1.51 1.28
Command? stop
Program ended.
```

1.2 Rectangle sorting

A file defines a set of rectangles with the following format:

- Each row of the file contains 1 string and 2 real numbers:
  - The string (4 characters long) is the rectangle identifier.
  - The two numbers specify the x and y coordinates, respectively, of one of its vertices.
- For each rectangle, there are two lines in the file, specifying the coordinates of two opposite vertices (top-right and bottom-left or top-left and bottomright).

Notice that, in general, the two rows defining a rectangle are not consecutive, and it is not known which vertex they specify. In any case, suppose the maximum number of rectangles is 100.

Write a C program that receives 3 file names on the command line:

- The first file is an input file, and it contains all rectangle specifications as previously indicated
- The second file is an output file, and it must contain the name of the rectangles ordered by descending area values.
- The third file is an output file, and it must contain the name of the rectangles ordered by descending perimeter values.

Examples

Let us suppose that the program receives the following three parameters:

```
rectangles.in area.out perimeter.out
```

Moreover, let us suppose that the content of *rectangles.in* is the following:

```
rct2 1.5 3.5
xxyy -0.5 3.0
xxyy 1.5 2.0
abcd 1.0 4.5
ktkr -2.5 1.5
abcd 2.0 2.0
rct2 3.5 -2.0
trya 2.5 -1.0
ktkr 1.5 3.5
trya 4.0 4.0
```

As are and perimeters of the rectangles are:

```
rct2 area=11.00 perimeter=15.00
xxyy area= 2.00 perimeter= 6.00
abcd area= 2.50 perimeter= 7.00
ktkr area= 8.00 perimeter=12.00
trya area= 7.50 perimeter=13.00
```

The program has to generate the following two files:
area.out:

```
rct2
ktkr
trya
abcd
xxyy
```

perimeter.out

```
rct2
trya
ktkr
abcd
xxyy
```

Implementation notes

Use an array of structures where each element of the array stores the name and the two extreme coordinates of a rectangle.

2 Homework

2.1 Taking it further: Rectangle sorting with dynamic memory

Extend the previous program with dynamic memory allocation, i.e. dynamically allocate an array of structures to store information on all rectangles **by reading the file one single time**. The input file remains the same as the previous exercise:

```
rct2 1.5 3.5
xxyy -0.5 3.0
xxyy 1.5 2.0
abcd 1.0 4.5
ktkr -2.5 1.5
abcd 2.0 2.0
rct2 3.5 -2.0
trya 2.5 -1.0
ktkr 1.5 3.5
trya 4.0 4.0
```

Implementation notes

Use the *realloc* function to dynamically allocate the needed amount of memory.

Acknowledgment

The teachers would like to thank Leonardo Giannantoni for writing the first version of some exercises contained in this laboratory text.