“Programming” Big Project

Made by: Xie Zong Pu
Neptun code: RLKNMA
Email: szumixie@gmail.com

Course code: IP-18fPROGEG
Teacher’s name: Menyhárt László Gábor

28th December 2018

Contents

User documentation

Task

We have information about the schedules of N trains from Budapest to Siófok.

Write a program that gives the train that is the fastest among the trains which took longer than 120 minutes to reach their destination.

Runtime environment

An IBM PC that can run exe files, 32-bit operating system (e.g. Windows 7). No mouse needed.

Usage

Starting the program

The program can be found in the archived file by the name `trains\bin\Release\trains.exe`. You can start the program by clicking the `trains.exe` file.

Program input

The program reads the input data from the keyboard in the following order:

#	Data	Explanation
1	N	The amount of trains ($1 \leq N \leq 100$)
2	T_1	Travel time of the first train in minutes ($1 \leq T_1 \leq 300$)
3	T_2	Travel time of the first train in minutes ($1 \leq T_2 \leq 300$)
...		
N+1	T_N	Travel time of the first train in minutes ($1 \leq T_N \leq 300$)

Program output

The program writes out the index of the fastest train among the trains whose travel time is more than 120 minutes, and its travel time separated by a space. If there are no such trains, the output is −1. If there is more than 1 solution, the output is the smallest index.

Sample input and output

```
Fastest among the slow
Please input the number of trains (between 1 and 100): 6
Please input the travel time of train #1 in minutes (between 1 and 300): 216
Please input the travel time of train #2 in minutes (between 1 and 300): 120
Please input the travel time of train #3 in minutes (between 1 and 300): 144
Please input the travel time of train #4 in minutes (between 1 and 300): 63
Please input the travel time of train #5 in minutes (between 1 and 300): 145
Please input the travel time of train #6 in minutes (between 1 and 300): 290
The fastest slow train is train #3, with a travel time of 144 minutes.
3 144
```

Possible errors

The input should be given according to the sample. If the number of trains is not a whole number, or it is not between 1 and 100, it will cause a error. If one of the travel times is not whole a number, or it is not between 1 and 300, it also will cause a error. In the case of an error, the program displays an error message and asks for the repetition of the input.

Sample of running in the case of invalid data:

```
Fastest among the slow
Please input the number of trains (between 1 and 100): lots
Input must be an integer, please try again: 123
The number of trains must be between 1 and 100, please try again: 3.1
Input must be an integer, please try again: 2
Please input the travel time of train #1 in minutes (between 1 and 300): short
Input must be an integer, please try again: -1
The travel time of a train must be between 1 and 300 minutes, please try again:
     123.4
Input must be an integer, please try again: 60
Please input the travel time of train #2 in minutes (between 1 and 300):
```

Developer documentation

Task

We have information about the schedules of N trains from Budapest to Siófok.

Write a program that gives the train that is the fastest among the trains which took longer than 120 minutes to reach their destination.

Specification

Input $N \in \mathbf{N},\ Times_{1..N} \in \mathbf{N}^N$

Output $Exists \in \mathbf{L},\ Index \in \mathbf{N},\ Minutes \in \mathbf{N}$

Precondition $1 \leq N \leq 100 \ \wedge\ \forall i (1 \leq i \leq N) : 1 \leq Times_i \leq 300$

Postcondition $Exists = \big(\exists i (1 \leq i \leq N) : Times_i > 120\big) \ \wedge$
$\Big(Exists \implies \big(\forall i (1 \leq i \leq N),\ Times_i > 120 : Times_{Index} \geq Times_i\big) \ \wedge$
$\big(\forall i (1 \leq i \leq N),\ Times_{Index} = Times_i : Index \leq i\big) \wedge Minutes = Times_{Index}\Big)$

Developer environment

IBM PC, an operating system capable of running exe files (e.g. Windows 7).
`mingw32-g++.exe` C++ compiler (v5.1), Code::Blocks (v17.12) developer tool.

Source code

All the sources can be found in the `trains` folder (after extraction). The folder structure used for development:

File	Explanation
`trains\bin\Release\trains.exe`	Executable code
`trains\obj\Release\main.o`	Semi-compiled code
`trains\main.cpp`	C++ source code
`trains\test1.txt`	input test file$_1$
`trains\test2.txt`	input test file$_2$
`trains\test3.txt`	input test file$_3$
`trains\test4.txt`	input test file$_4$
`trains\test5.txt`	input test file$_5$
`trains\test6.txt`	input test file$_6$
`trains\test7.txt`	input test file$_7$
`trains\test8.txt`	input test file$_8$
`trains\documentation.pdf`	documentation (this file)

Solution

Program parameters

Constants

```
maxN : Integer(100)
maxT : Integer(300)
```

Variables

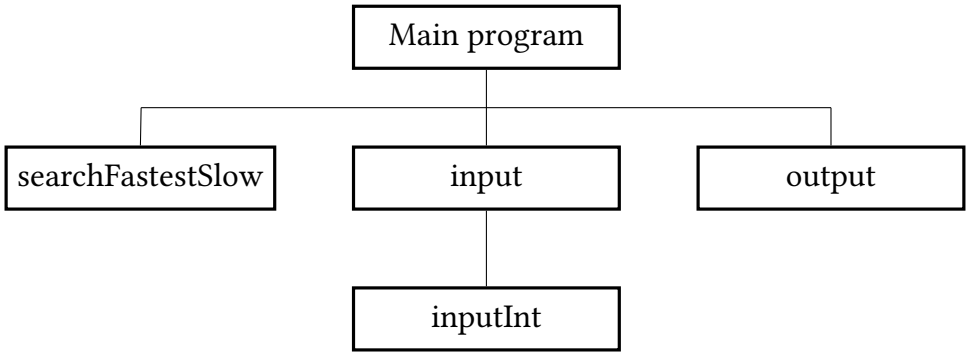
```
n    : Integer
ts   : Array(1..maxN : Integer)
ex   : Bool
idx  : Integer
mins : Integer
```

The structure of the program

The modules used by the program, and their locations:

`main.cpp` the program, in the source folder
`iostream` keyboard and console management, part of the C++ system

Structure of functions



The algorithm of the program

Main program:

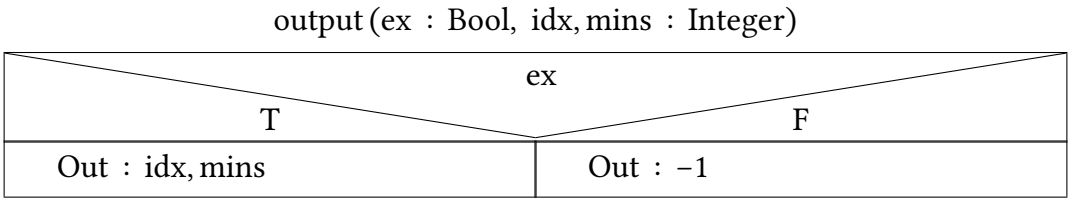
main

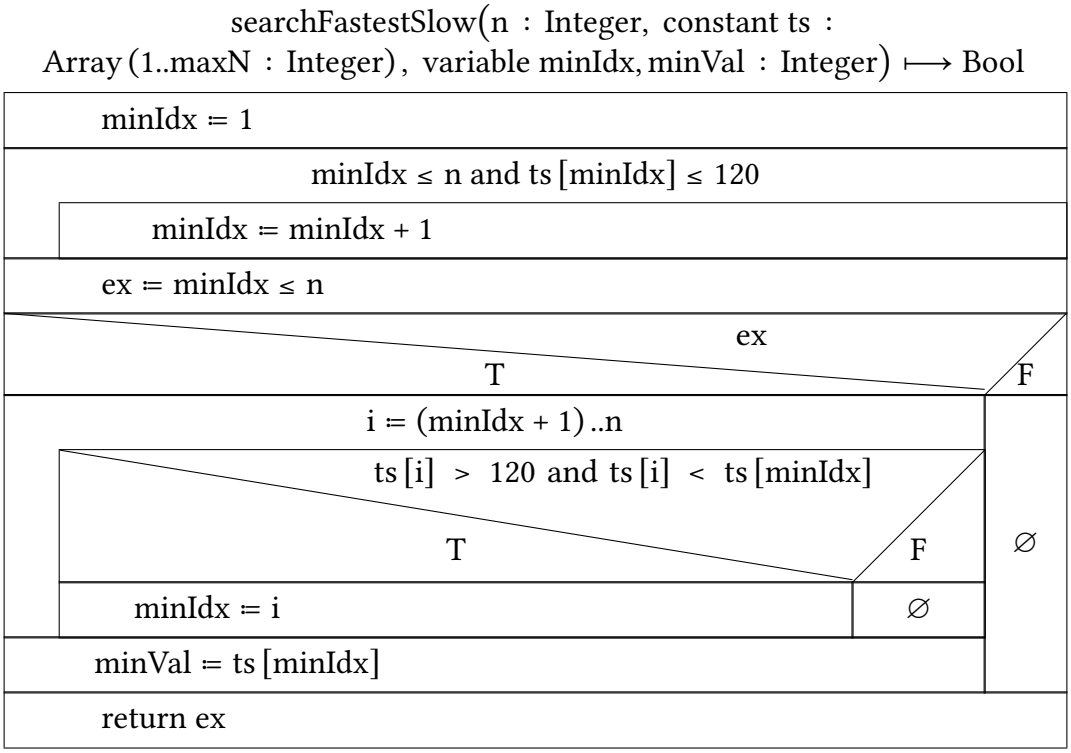
n ≔ input (ts)
ex ≔ searchFastestSlow (n, ts, idx, mins)
output (ex, idx, mins)

Subprograms:

input(ts : Array (1..maxN : Integer)) ⟼ Integer

In : n [1 ≤ n ≤ maxN]
In : ts [1..n] [1 ≤ ts [1..n] ≤ maxT]
return n





The code

The content of the `main.cpp` file:

```
/*********************************************\
 * Created by: Xie Zong Pu                     *
 * Neptun: RLKNMA                              *
 * Email: szumixie@gmail.com                   *
 * Task: "Big Project" - Fastest among the slow *
\*********************************************/

#include <iostream>

#define maxN 100
#define maxT 300
```

```
; bool searchFastestSlow(int n, const int ts[maxN], int &minIdx, int &minVal)
  { minIdx = 0
  ; while (minIdx < n && ts[minIdx] <= 120)
      ++minIdx
  ; bool ex = minIdx < n

  ; if (ex)
    { for (int i = minIdx + 1; i < n; ++i)
        if (ts[i] > 120 && ts[i] < ts[minIdx])
          minIdx = i
    ; minVal = ts[minIdx++]
    ;}

  ; return ex
  ;}

; void inputInt(int &n)
  { while (!(std::cin >> n))
    { std::cin.clear()
    ; std::cin.sync()
    ; std::cerr << "Input must be an integer, please try again: "
    ;}
  ;}

; int input(int ts[maxN])
  { int n

  ; std::clog << "Please input the number of trains (between 1 and 100): "
  ; inputInt(n)
  ; while (n < 1 || n > maxN)
    { std::cerr << "The number of trains must be between 1 and 100, please try
        again: "
    ; std::cin.sync()
    ; inputInt(n)
    ;}

  ; for (int i = 0; i < n; ++i)
    { std::clog << "Please input the travel time of train #" << i + 1 << " in
        minutes (between 1 and 300): "
    ; inputInt(ts[i])
    ; while (ts[i] < 1 || ts[i] > maxT)
      { std::cerr << "The travel time of a train must be between 1 and 300
          minutes, please try again: "
      ; std::cin.sync()
      ; inputInt(ts[i])
      ;}
    ;}

  ; return n
  ;}

; void output(bool ex, int idx, int mins)
  { if (ex)
    { std::clog << "The fastest slow train is train #" << idx << ", with a
        travel time of " << mins << " minutes." << std::endl
    ; std::cout << idx << ' ' << mins << std::endl
```

```
    ;}
    else
    { std::clog << "There aren't any slow trains." << std::endl
    ; std::cout << -1 << std::endl
    ;}
  ;}

; int main()
  { std::clog << "Fastest among the slow" << std::endl

  ; int n, ts[maxN]
  ; bool ex
  ; int idx, mins

  ; n = input(ts)
  ; ex = searchFastestSlow(n, ts, idx, mins)
  ; output(ex, idx, mins)

  ; return 0
  ;}
```

Testing

Valid test cases

1st test case: `test1.txt`

Input – only fast trains
$N = 3$ $Times_1 = 60$ $Times_2 = 21$ $Times_3 = 109$
Output
-1

2nd test case: `test2.txt`

Input – only one slow train
$N = 5$ $Times_1 = 100$ $Times_2 = 96$ $Times_3 = 9$ $Times_4 = 150$ $Times_5 = 75$
Output
4 150

3rd test case: `test3.txt`

Input – only slow trains
$N = 5$ $Times_1 = 210$ $Times_2 = 165$ $Times_3 = 165$ $Times_4 = 288$ $Times_5 = 121$
Output
5 121

4th test case: `test4.txt`

Input – all slow trains are different
$N = 6$ $Times_1 = 216$ $Times_2 = 120$ $Times_3 = 144$ $Times_4 = 63$ $Times_5 = 145$ $Times_6 = 290$
Output
3 144

5th test case: `test5.txt`

Input – all slow trains are equal
$N = 4$ $Times_1 = 132$ $Times_2 = 132$ $Times_3 = 66$ $Times_4 = 132$
Output
1 132

Invalid test cases

6th test case: `test6.txt`

Input – wrong amount
N = lots
Output
Asking again: N =

7th test case: `test7.txt`

Input – less than one minute
N = 6 $Times_1$ = −3
Output
Asking again: $Times_1$ =

8th test case: `test8.txt`

Input – more than `maxN` minutes
N = 20 $Times_1$ = 432
Output
Asking again: $Times_1$ =

Further development options

1. Reading data from a file
2. Detection of wrong file input, writing out the location and ID# of error
3. Capability of running multiple times one after another