

„Programming” Big Project

*Made by: Thomas Tiny¹
Neptun code: A1B2C3
E-mail: thomastiny@example.com*

*Course code: ?????
Teacher's name: ???*

2018. December 1.

Content

User documentation.....	4
Task.....	4
Runtime environment.....	4
Usage.....	4
Starting the program.....	4
Program input.....	4
Program output.....	4
Sample input and output.....	5
Possible errors.....	5
Developer documentation.....	6
Task.....	6
Specification.....	6
Developer environment.....	7
Source code.....	7
Solution.....	7
Program parameters.....	7
The structure of the program.....	7
Structure of functions.....	7
The algorithm of the program.....	8
The code.....	8
Testing.....	10
Valid test cases.....	10
Invalid test cases.....	11
Further development options.....	11

User documentation

Task

We measure the height above sea level at equal distances during a flight. If we measured 0, it means we were flying above the sea, if we measured a positive height, it means we were above land. Three consecutive measurements are labelled A, B and C. Then B

- means left side of land, if $B > 0$ and $A = 0$;
- right side of land, if $B > 0$ and $C = 0$.

Create a program that gives two islands that are the closest to each other among islands. If there are no such islands, the result should be 0.

Runtime environment

An IBM PC that is capable of running exe files, 32-bit operating system (eg. Windows 7). No mouse needed..

Usage

Starting the program

The program can be found in the archived file by the name `A1B2C3\bin\Release\A1B2C3.exe`. You can start the program by clicking the `A1B2C3.exe` file.

Program input

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

#	Data	Explanation
1.	N	The length of the measurements ($2 \leq N \leq 10000$).
2.	$height_1$	The first height ($0 \leq height_1 \leq 9000$).
3.	$height_2$	The second height ($0 \leq height_2 \leq 9000$).
...	...	
$N+1$.	$height_N$	The N^{th} height ($0 \leq height_N \leq 9000$).

Program output

The program writes out the measurement identifiers of the left and right sides of the two closest islands. The first line contains the identifiers of the first island, the second line contains the identifiers of the second island. If there are less than 2 islands, then 0 is the only output.

Sample input and output

```
===Closest islands===
# of measurements [2..10000]:12
1. measurement [0..9000]:3
2. measurement [0..9000]:0
3. measurement [0..9000]:2
4. measurement [0..9000]:0
5. measurement [0..9000]:4
6. measurement [0..9000]:3
7. measurement [0..9000]:0
8. measurement [0..9000]:0
9. measurement [0..9000]:3
10. measurement [0..9000]:0
11. measurement [0..9000]:2
12. measurement [0..9000]:0
The closest islands are:
First: 3 3
Second: 5 6
```

Possible errors

The input should be given according to the sample. If the number of measurements is not a whole number, or it is not in the range 2..10000, it will cause a problem. If one of the measurements is not a number, or it is not in the range 0..9000, it also will cause a problem. In the case of an error, the program displays an error message, or asks for the repetition of the input.

Sample of running in the case of invalid data:

```
===Closest islands===
# of measurements [2..10000]:lot
# of measurements [2..10000]:1
# of measurements [2..10000]:1.1
# of measurements [2..10000]:2
1. measurement [0..9000]:little
1. measurement [0..9000]:-1
1. measurement [0..9000]:9001
1. measurement [0..9000]:0
2. measurement [0..9000]:
```

Developer documentation

Task

We measure the height above sea level at equal distances during a flight. If we measured 0, it means we were flying above the sea, if we measured a positive height, it means we were above land.

Three consecutive measurements are labelled A, B and C. Then B means left side of land, if $B > 0$ and $A = 0$;

right side of land, if $B > 0$ and $C = 0$.

Create a program that gives two islands that are the closest to each other among islands. If there are no such islands, the result should be 0.

Specification

Input: $N \in \mathbb{N}$, $\text{Heights} \in \mathbb{N}^*$

Output: $\text{Exists} \in \mathbb{L}$, $\text{Is1}, \text{Is2} \in \text{Island}$, $\text{Island} = \text{Left} \times \text{Right}$, $\text{Left}, \text{Right} = \mathbb{N}$

Precondition: $N = \text{Length}(\text{Heights}) \wedge N \in [2..10000] \wedge \forall i \in [1..N]: \text{Heights}_i \in [0..9000]$

Postcondition: $\text{cnt} = \sum_{i=2}^{N-1} 1 \wedge$
 $\text{IslandBegin}(i)$

$\text{islands} \in \text{Island}^{\text{cnt}} \wedge$

$\forall i \in [1..\text{cnt}]: (\text{islands}_i.\text{Left} \in [2..N-1] \wedge \text{islands}_i.\text{Right} \in [2..N-1] \wedge$

$\text{islands}_i.\text{Left} \leq \text{islands}_i.\text{Right} \wedge \text{IslandBegin}(\text{islands}_i.\text{Left}) \wedge \text{IslandEnd}(\text{islands}_i.\text{Right}) \wedge$

$\forall j \in [\text{islands}_i.\text{Left}..\text{islands}_i.\text{Right}]: \text{Heights}_j > 0) \wedge$

$\text{cnt} < 2 \rightarrow \text{Exists} = \text{False} \wedge$

$\text{cnt} \geq 2 \rightarrow \text{Exists} = \text{True} \wedge$

$\exists i \in [1..\text{cnt}-1]: \text{Is1} = \text{islands}_i \wedge \text{Is2} = \text{islands}_{i+1} \wedge$

$\forall i \in [1..\text{cnt}-1]: \text{islands}_{i+1}.\text{Left} - \text{islands}_i.\text{Right} \geq \text{Is2}.\text{Left} - \text{Is1}.\text{Right}$

Definitions: $\text{IslandBegin}: \mathbb{N} \rightarrow \mathbb{L}$

$\text{IslandBegin}(i) := \text{Heights}_i > 0 \wedge \text{Heights}_{i-1} = 0$

$\text{IslandEnd}: \mathbb{N} \rightarrow \mathbb{L}$

$\text{IslandEnd}(i) := \text{Heights}_i > 0 \wedge \text{Heights}_{i+1} = 0$

Comment: If there are less than 2 islands ($\text{Exists} = \text{False}$ case) the program will write out a 0, and not the logical value (as it was required by the task).

Developer environment

IBM PC, an operating system capable of running exe files (eg. Windows 7). mingw32-g++.exe c++ compiler (v4.7), Code::Blocks (v13.12) developer tool.

Source code

All the sources can be found in the *A1B2C3* folder (after extraction). The folder structure used for development:

File	Explanation
<i>A1B2C3\bin\Release\A1B2C3.exe</i>	Executable code
<i>A1B2C3\obj\Release\main.o</i>	Semi-compiled code
<i>A1B2C3\main.cpp</i>	C++ source code
<i>A1B2C3\test1.txt</i>	input test file ₁
<i>A1B2C3\test2.txt</i>	input test file ₂
<i>A1B2C3\test3.txt</i>	input test file ₃
<i>A1B2C3\test4.txt</i>	input test file ₄
<i>A1B2C3\test5.txt</i>	input test file ₅
<i>A1B2C3\doksi\A1B2C3.docx</i>	documentation (this file)

Solution

Program parameters

Contants

```
MaxN           : Integer(10000)      [the max of measurements]
MaxHeight      : Integer(9000) [the max height]
```

Types

```
THeights       = Array(1..MaxN:Integer)
TIsland        = Record(left, right:Integer)
```

Variables

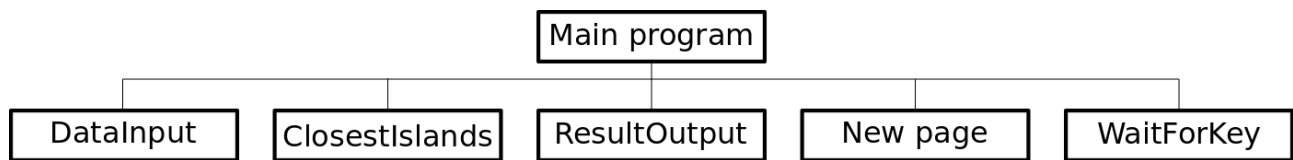
```
N              : Integer
Heights        : THeights
Is1, Is2       : TIsland
```

The structure of the program

The modules used by the program, and their locations:

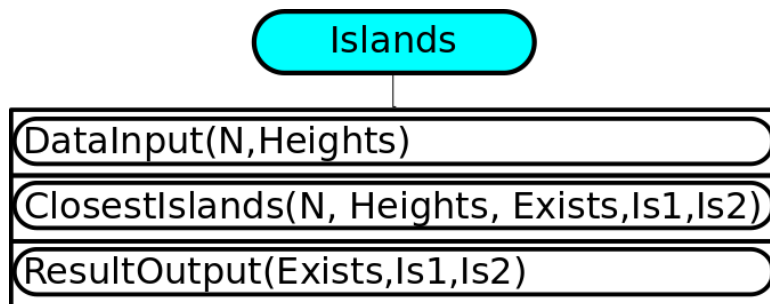
```
main.cpp      – the program, in the source folder
iostream      – keyboard and console management, part of the C++ system
stdlib.h      – general routines, part of the C++ system
```

Structure of functions

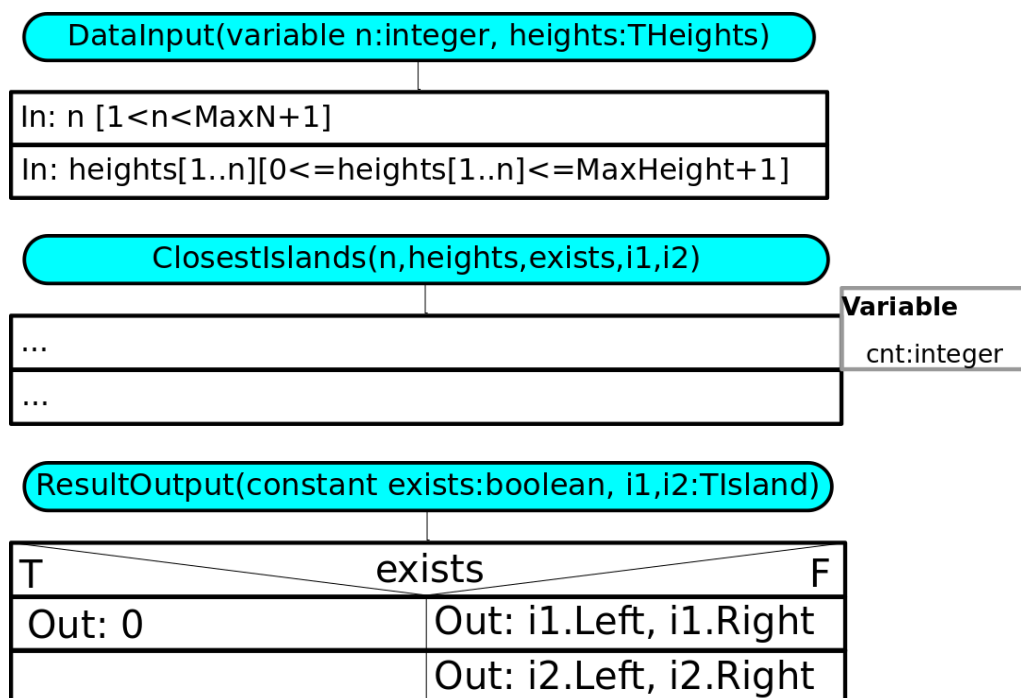


The algorithm of the program

Main program:



Subprograms:



The code

The content of the `main.cpp` file:

```

/*
  Created by: Thomas Tiny
  Neptun: AlB2C3
  E-mail: thomastiny@example.com
  Task: „Big Project” - Closest islands
*/
#include <iostream>
#include <stdlib.h>
  
```



```

using namespace std;

const string Title="Closest islands";
const int MaxN=10000;
const int MaxHeight=9000;
typedef int THeights[MaxN];
typedef struct{int left,right;} TIsland;
//Input:
int N;
THeights Heights;
//Output:
bool Exists;
TIsland Isl,Is2;

void DataInput(int& n, THeights heights);
void ClosestIslands(int n, const THeights heights,
                    bool& exists, TIsland& i1, TIsland& i2);
void ResultOutput(bool exists, TIsland i1, TIsland i2);
void WaitForKey();

int main()
{
    clog << Title << endl << endl;
    DataInput(N,Heights);
    ClosestIslands(N,Heights,Exists,Isl,Is2);
    ResultOutput(Exists,Isl,Is2);
    WaitForKey();//should be commented in Biro version!!!
    return 0;
}

void DataInput(int& n, THeights heights)
{
    do{
        clog << "..."; cin >> ...;
        ...
    }While (...);
    ...
}

void ClosestIslands(int n, const THeights heights,
                    bool& exists, TIsland & i1, TIsland& i2)
{
    ...
}

void ResultOutput(bool exists, TIsland i1, TIsland i2)
{
    clog << "..."; cout << ...;
    ...
}

void WaitForKey()
{
    ...
}

```

Testing

Valid test cases

1. test case: in1.txt

Input – no island, lenght is minimal	
N = 2	
Height ₁ = 0	
Height ₂ = 0	
Output	
0	

2. test case: in2.txt

Input – starts with continent, there are at least 2 islands	
N = 12	
Height ₁ = 3	
Height ₂ = 0	
Height ₃ = 2	
Height ₄ = 0	
Height ₅ = 4	
Height ₆ = 3	
Height ₇ = 0	
Height ₈ = 0	
Height ₉ = 3	
Height ₁₀ = 0	
Height ₁₁ = 2	
Height ₁₂ = 0	
Output	
Is1 = 3 3	
Is2 = 5 6	

3. test case: in3.txt

Input – ends with continent, there are at least 2 islands	
N = ...	
Height ₁ = ...	
...	
Output	
...	

4. test case: in4.txt

Input – no continent, only one island	
N = ...	
Height ₁ = ...	
...	
Output	
...	

5. test case: in5.txt

Input – there are only continents	
-----------------------------------	--

N = ...
Height ₁ = ...
...
Output
...

Invalid test cases

6. test case

Input – wrong length
N = eleven11
Output
Asking again: N =

7. test case

Input – wrong height
N = 11 Height ₁ = -1
Output
Asking again: Height ₁ =

...

8. test case

...

Further development options

1. Data to be read from file
2. Detection of wrong file input, writing out the location and ID# of error
3. Capability to run multiple times after each other
4. Visual representation of input data, and emphasizing the result islands with different colors