„Programozási alapismeretek”
beadandó feladat

Készítette: Gipsz Jakab [1]
Neptun-azonosító: A1B2C3
E-mail: gipszjakab@vilaghalo.hu

Kurzuskód: IT-13PA1EG
Gyakorlatvezető neve: ???

2018. január 24.

[1] Értelemszerűen töltendők itt ki a szerzőre vonatkozó adatok.
A lábjegyzet(ek) a végső dokumentációból törlendő(k)! Csak az Ön segítését szolgálja.

Tartalom

Felhasználói dokumentáció

Feladat

Egy repülőút során egyenlő távolságonként mértük a felszín tengerszint feletti magasságát. Zéró magasságot ott mértünk, ahol állóvíz volt, pozitív magasságot pedig ott, ahol szárazföld. Három egymást követő mérési eredményt jelöljön `A`, `B` és `C`. Ekkor `B`

- bal oldali partot jelez, ha `B>0 és A=0`;
- jobb oldali partot jelez, ha `B>0 és C=0`.

Készítsen programot, amely meghatároz két szigetet, melyeknél nincs egymáshoz közelebbi sziget-pár, ha nincs ilyen, akkor ezt egyetlen 0-val jelezze!

Futási környezet

IBM PC, exe futtatására alkalmas, 32-bites operációs rendszer (pl. Windows 7). Nem igényel egeret.

Használat

A program indítása

A program az `A1B2C3\bin\Release\A1B2C3.exe` néven található a tömörített állományban. A `A1B2C3.exe` fájl kiválasztásával indítható.

A program bemenete

A program az adatokat a **billentyűzet**ről olvassa be a következő sorrendben:

#	Adat	Magyarázat
1.	N	A magasságmérés hossza ($2 \le N \le 10000$).
2.	$Magasság_1$	Az első magasság ($0 \le Magasság_1 \le 9000$).
3.	$Magasság_2$	A második magasság ($0 \le Magasság_2 \le 9000$).
...	...	
N+1.	$Magasság_N$	Az N-edik magasság ($0 \le Magasság_N \le 9000$).

A program kimenete

A program kiírja az egymáshoz legközelebbi két sziget bal és jobb partját jelző mérés sorszámát. A kimenet első sorába az első szigetet, a második sorba a második szigetet azonosító két adat kerül. Ha nem lenne legalább két sziget, akkor egyetlen 0 a kimenet.

Minta bemenet és kimenet

```
                        Legközelebbi szigetek

Mérések száma [2..10000]:12
   1. mérés [0..9000]:3
   2. mérés [0..9000]:0
   3. mérés [0..9000]:2
   4. mérés [0..9000]:0
   5. mérés [0..9000]:4
   6. mérés [0..9000]:3
   7. mérés [0..9000]:0
   8. mérés [0..9000]:0
   9. mérés [0..9000]:3
  10. mérés [0..9000]:0
  11. mérés [0..9000]:2
  12. mérés [0..9000]:0
A legközelebbi szigetpár elsője: 3 3
... párja: 5 6
```

Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha a mérések száma nem egész szám, vagy nem esik a `2..10 000` intervallumba; vagy valamely magassági érték nem szám, vagy nem esik a `0..9 000` intervallumba. Hiba esetén a program azzal jelzi a hibát, hogy újra kérdezi azt.

Mintafutás hibás bemeneti adatok esetén:

```
                        Legközelebbi szigetek

Mérések száma [2..10000]:sok
Mérések száma [2..10000]:1
Mérések száma [2..10000]:1.1
Mérések száma [2..10000]:2
   1. mérés [0..9000]:kevés
   1. mérés [0..9000]:-1
   1. mérés [0..9000]:9001
   1. mérés [0..9000]:0
   2. mérés [0..9000]:
```

Fejlesztői dokumentáció

Feladat

Egy repülőút során egyenlő távolságonként mértük a felszín tengerszint feletti magasságát. Zéró magasságot ott mértünk, ahol állóvíz volt, pozitív magasságot pedig ott, ahol szárazföld. Három egymást követő mérési eredményt jelöljön `A`, `B` és `C`. Ekkor `B`

- bal oldali partot jelez, ha `B>0 és A=0`;
- jobb oldali partot jelez, ha `B>0 és C=0`.

Készítsen programot, amely meghatároz két szigetet, melyeknél nincs egymáshoz közelebbi sziget-pár, ha nincs ilyen, akkor ezt egyetlen 0-val jelezze!

Specifikáció

Bemenet: $N \in \mathbb{N}$, Magasságok$\in \mathbb{N}^*$

Kimenet: $VanE \in \mathbb{L}$, $Szig1, Szig2 \in Sziget$, $Sziget = Bal \times Jobb$, $Bal, Jobb = \mathbb{N}$

Előfeltétel: $N = Hossz(Magasságok) \wedge N \in [2..10000] \wedge \forall i \in [1..N]: Magasságok_i \in [0..9000]$

Utófeltétel: $db = \sum_{i=2}^{N-1}{}_{SzigetKezdet(i)} 1 \wedge$
$szigek \in Sziget^{db} \wedge$
$\forall i \in [1..db]: (szigek_i.Bal \in [2..N-1] \wedge szigek_i.Jobb \in [2..N-1] \wedge$
$szigek_i.Bal \leq szigek_i.Jobb \wedge SzigetKezdet(szigek_i.Bal) \wedge SzigetVég(szigek_i.Jobb) \wedge$
$\forall j \in [szigek_i.Bal..szigek_i.Jobb]: Magasságok_j > 0) \wedge$
$db < 2 \rightarrow VanE = Hamis \wedge$
$db \geq 2 \rightarrow VanE = Igaz \wedge$
$\exists i \in [1..db-1]: Szig1 = szigek_i \wedge Szig2 = szigek_{i+1} \wedge$
$\forall i \in [1..db-1]: szigek_{i+1}.Bal - szigek_i.Jobb \geq Szig2.Bal - Szig1.Jobb$

Definíció: $SzigetKezdet: \mathbb{N} \rightarrow \mathbb{L}$
$SzigetKezdet(i) := Magasságok_i > 0 \wedge Magasságok_{i-1} = 0$
$SzigetVég: \mathbb{N} \rightarrow \mathbb{L}$
$SzigetVég(i) := Magasságok_i > 0 \wedge Magasságok_{i+1} = 0$

Megjegyzés: a „ha nincs ilyen” kitételt (a VanE=Hamis esetben) a program egyetlen 0 kiírásával fogja jelezni, nem pedig a logikai érték megjelenítésével (hűen a feladat eredeti kiírásához).

Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 7). mingw32-g++.exe c++ fordítóprogram (v4.7), Code::Blocks (v13.12) fejlesztői környezet.

Forráskód

A teljes fejlesztői anyag –kicsomagolás után– az `A1B2C3` nevű könyvtárban található meg. A fejlesztés során használt könyvtár-struktúra:

Állomány	Magyarázat
`A1B2C3\bin\Release\A1B2C3.exe`	futtatható kód
`A1B2C3\obj\Release\main.o`	félig lefordított kód
`A1B2C3\main.cpp`	C++ forráskód
`A1B2C3\teszt1.txt`	teszt-bemeneti fájl$_1$
`A1B2C3\teszt2.txt`	teszt-bemeneti fájl$_2$
`A1B2C3\teszt3.txt`	teszt-bemeneti fájl$_3$
`A1B2C3\teszt4.txt`	teszt-bemeneti fájl$_4$
`A1B2C3\teszt5.txt`	teszt-bemeneti fájl$_5$
`A1B2C3\doksi\A1B2C3.docx`	dokumentációk (ez a fájl)

Megoldás

Programparaméterek

Konstans

```
MaxN          : Egész(10000) [a mérések maximális száma]
MaxMagasság   : Egész(9000)  [a maximális magasság]
```

Típus

```
TMaggasságok  = Tömb(1..MaxN:Egész)
TSziget       = Rekord(bal,jobb:Egész)
```

Változó

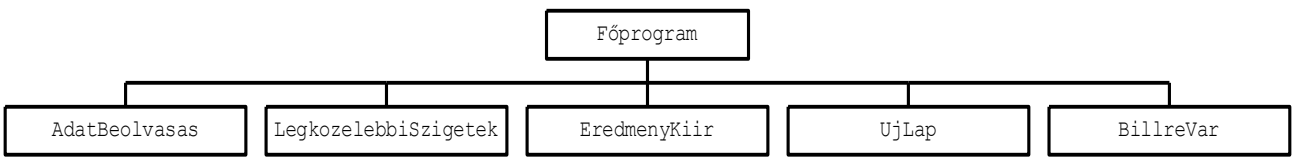
```
N             : Egész
Magasságok    : TMagasságok
Szig1,Szig2   : TSziget
```

Programfelépítés

A program által használt modulok (és helyük):

`main.cpp` – program, a forráskönyvtárban
`iostream` – képernyő-, és billentyűkezelés, a C++ rendszer része
`stdlib.h` – általános rutinok, a C++ rendszer része

Függvénystruktúra



A teljes program algoritmusa

```
Program Szigetek:
  [programparaméterek:]
  Konstans
    MaxN:Egész(10000)   [a mérések maximális száma, a bekéréshez]
    MaxMagasság:Egész(9000)   [a maximális magasság, a bekéréshez]
  Típus
    TMagasságok=Tömb(1..MaxN:Egész)
    TSziget=Rekord(bal,jobb:Egész)
  Változó
    [Bemenet:]
    N:Egész
    Magasságok:TMagasságok
    [Kimenet:]
    VanE:Logikai
    Szig1,Szig2:TSziget
  [főprogram:]
  AdatBeolvasás(N,Magasságok)
  LegközelebbiSzigetek(N,Magasságok,VanE,Szig1,Szig2)
  EredményKiírás(VanE,Szig1,Szig2)
Program vége.

[alprogramok:]
Eljárás AdatBeolvasás(Változó n:Egész, magok:TMagasságok):
  Be:n [n∈[2..MaxN]]
  Be:magok(1..n) [magok(1..n)∈[0..MaxMagasság]]
Eljárás vége.

Eljárás LegkozelebbiSzigetek(Konstans n:Egész, magok:TMagasságok,
                             Változó vanE:Logikai, sz1,sz2:TSziget):
  [Lokális konstansok, típusok, változók:]
  Kosntans
    MaxSz:Egész(MaxN Div 2)
    sziget0:TSziget=(bal:0,jobb:0)
  Típus
    TSzigetek=Tömb(0..MaxSz:TSziget)
  Változó
    db,i,
    minI,minT:Egész
    szigek:TSzigetek
```

```
  [szigetek inicializálása:]
  szigek(0..MaxSz):=sziget0 [2]
  [szigetek kiválogatása:]
  db:=0
  Ciklus i=2-től n-1-ig
    Ha magok(i)>0 és magok(i-1)=0 akkor
      db:+1
      szigek(db).bal:=i
    Elágazás vége
    Ha magok(i)>0 és magok(i+1)=0 akkor
      szigek(db).jobb:=i
    Elágazás vége
  Ciklus vége
  Ha db>0 és szigek(db).jobb=0 akkor db:-1 [a túlpart bal partja, amely nem sziget]
  Ha db<2 akkor [nincs két legközelebbi sziget:]
    vanE:=Hamis
  különben [van legalább két sziget:]
    [minimális távolság:]
    minI:=1; minT:=szigek(2).bal-szigek(1).jobb
    Ciklus i=2-től db-1-ig
      Ha szigek(i+1).bal-szigek(i).jobb<minT akkor
        minI:=i; minT:=szigek(i+1).bal-szigek(i).jobb
      Elágazás vége
    Ciklus vége
    sz1.bal:=szigek(minI).bal;   sz1.jobb:=szigek(minI).jobb
    sz2.bal:=szigek(minI+1).bal; sz2.jobb:=szigek(minI+1).jobb
    vanE:=Igaz
  Elágazás vége
Eljárás vége.

Eljárás EredményKiirás(Konstans vanE:Logikai, sz1,sz2:TSziget):
  Ha vanE akkor
    Ki: 0
  különben
    Ki: sz1.bal,sz1.jobb,sz2.bal,sz2.jobb [a megkívánt formátum szerint]
  Elágazás vége
Eljárás vége.
```

A kód

A `main.cpp` fájl tartalma:

```
/*
  Készítette: Gipsz Jakab
  Neptun: A1B2C3
  E-mail: gipszjakab@vilaghalo.hu
  Feladat: „ProgAlap beadandó feladatok” téma „Legközelebbi szigetek” feladat
*/
#include <iostream>
#include <stdlib.h>

using namespace std;
```

[2] A fenti az alábbival ekvivalens tömör leírás:

```
  Ciklus i=0-tól MaxSz-ig
    szigek(0.MaxSz):=sziget0
  Ciklus vége
```

```
const string Cim="Legközelebbi szigetek";
const int MaxN=10000;
const int MaxMagassag=9000;
typedef int TMagassagok[MaxN];
typedef struct{int bal,jobb;} TSziget;
//Bemenet:
int N;
TMagassagok Magassagok;
//Kimenet:
bool VanE;
TSziget Szig1,Szig2;

void AdatBeolvasas(int& n, TMagassagok magok);
void LegkozelebbiSzigetek(int n, const TMagassagok magok,
                          bool& vanE, TSziget& sz1, TSziget& sz2);
void EredmenyKiir(bool VanE, TSziget sz1, TSziget sz2);
void BillreVar();

int main()
{
    clog << Cim << endl << endl;
    AdatBeolvasas(N,Magassagok);
    LegkozelebbiSzigetek(N,Magassagok,VanE,Szig1,Szig2);
    EredmenyKiir(VanE,Szig1,Szig2);
    BillreVar();//ez a biro-s változatban kommentálva szerepel!!!
    return 0;
}

void AdatBeolvasas(int& n, TMagassagok magok)
{
    do{
      clog << "…"; cin >> …;
      …
    }While (…);
    …
}

void LegkozelebbiSzigetek(int n, const TMagassagok magok,
                          bool& vanE, TSziget& sz1, TSziget& sz2)
{
    …
}

void EredmenyKiir(bool VanE, TSziget sz1, TSziget sz2)
{
    clog << "…"; cout << …;
    …
}

void BillreVar()
{
    …
}
```

Tesztelés

Érvényes tesztesetek

1. teszteset: be1.txt

Bemenet – *nincs sziget; minimális hossz*
N = 2 Magasság$_1$ = 0 Magasság$_2$ = 0
Kimenet
0

2. teszteset: be2.txt

Bemenet – *kontinenssel kezdődik, van legalább 2 sziget*
N = 12 Magasság$_1$ = 3 Magasság$_2$ = 0 Magasság$_3$ = 2 Magasság$_4$ = 0 Magasság$_5$ = 4 Magasság$_6$ = 3 Magasság$_7$ = 0 Magasság$_8$ = 0 Magasság$_9$ = 3 Magasság$_{10}$ = 0 Magasság$_{11}$ = 2 Magasság$_{12}$ = 0
Kimenet
Szig1 = 3 3 Szig1 = 5 6

3. teszteset: be3.txt

Bemenet – *kontinenssel végződik, van legalább 2 sziget*
N = …
Magasság$_1$ = …
…
Kimenet
…

4. teszteset: be4.txt

Bemenet – *nincs kontinens, egy sziget van*
N = …
Magasság$_1$ = …
…
Kimenet
…

5. teszteset: be5.txt

Bemenet – *csak kontinens van*
N = …
Magasság$_1$ = …
…
Kimenet
…

Érvénytelen tesztesetek

6. teszteset

Bemenet – *Rossz hossz*
N = 11tizenegy
Kimenet
Újrakérdezés: N =

7. teszteset

Bemenet – *Rossz magasság*
N = 11 Magasság$_1$ = -1
Kimenet
Újrakérdezés: Magasság$_1$ =

…

8. teszteset

…

Fejlesztési lehetőségek

1. Adatok –a felhasználó igénye szerint– akár fájlból is fogadása.
2. Hibás fájl-bemenetek felismerése, és a hiba helyének (sor sorszámának) kiírása.
3. Többszöri futtatás megszervezése
4. A bemeneti sorozat grafikus megjelenítése, s az eredmény-szigetek elütő színű kijelzése.