„Programozási alapismeretek”
beadandó feladat:
„ProgAlap beadandó feladatok” téma: Árvízmentes szakaszok teljes hossza

Készítette: Prorok Ernest
Neptun-azonosító: FILSPA
E-mail: ernest.prorok@gmail.com

Kurzuskód: IP-08PAED
Gyakorlatvezető neve: Heizlerné Bakonyi Viktória

2016. május 8.

Tartalom

Felhasználói dokumentáció

Feladat

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. Elsőfokú árvízvédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, másodfokút, ha meghaladja a 900 centimétert és harmadfokút, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. Árvíznek nevezzük azt a szakaszt, ahol minden hely legalább elsőfokú készültségű.

Készíts programot, amely meghatározza az árvízmentes folyószakaszok teljes hosszát!

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\filspa.exe` néven található a tömörített állományban. A `filspa.exe` fájl kiválasztásával indítható.

A program bemenete

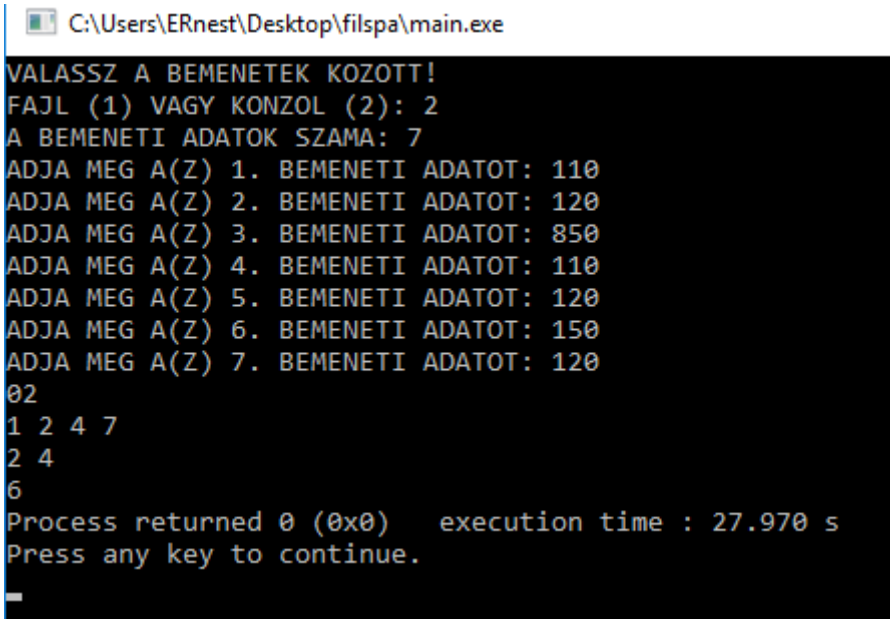
A program az adatokat a **billentyűzet**ről olvassa be a következő sorrendben:

A standard bemenet első sorában a mérési pontok száma szerepel (1<=N<=10 000), a következő N sor mindegyike egy-egy mérési eredményt tartalmaz (0<=A i <=3000).

A program kimenete

A standard kimenet első sorába az árvízmentes folyószakaszok K darabszámát kell írni (0, ha nincs ilyen folyószakasz, ebben az esetben nincsenek további sorok)! A második sorba ezen K folyószakasz kezdetének és végének a sorszáma kerüljön, növekvő sorrendben! A 3. sor tartalmazza a K szakasz hosszát, végül a 4. sor az árvízmentes folyószakaszok együttes hosszát!

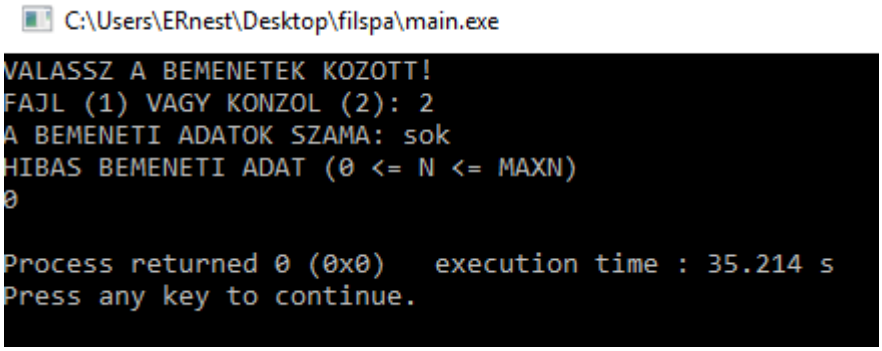
Minta bemenet és kimenet



Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha az N nem egész szám, vagy nem esik a `1..10 000` intervallumba; vagy valamely érték nem szám, vagy nem esik a `0..3 000` intervallumba. Hiba esetén a program azzal jelzi a hibát, hogy újra kérdezi azt.

Mintafutás hibás bemeneti adatok esetén:



Fejlesztői dokumentáció

Feladat

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. Elsőfokú árvízvédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, másodfokút, ha meghaladja a 900 centimétert és harmadfokút, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. Árvíznek nevezzük azt a szakaszt, ahol minden hely legalább elsőfokú készültségű.

Készíts programot, amely meghatározza az árvízmentes folyószakaszok teljes hosszát!

Specifikáció

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

Kimenet: choice, a, b, j, k, $i \in \mathbb{N}$; b, c$\in$MAXN

Előfeltétel: $N=MAXN(szakasz) \wedge N \in [1..10000] \wedge \forall i \in [1..N]: szakasz_i \in [0..3000]$

Utófeltétel: $db=\sum_{i=1}^{N} 1 \wedge$
$szakasz(i) \in MAXN^N \wedge$
szakasz[i] <= 800

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 `filspa` nevű könyvtárban található meg. A fejlesztés során használt könyvtár-struktúra:

Állomány	Magyarázat
`filspa\bin\Release\filspa.exe`	futtatható kód
`filspa\obj\Release\main.o`	félig lefordított kód
`filspa\main.cpp`	C++ forráskód
`filspa\be1.txt`	teszt-bemeneti fájl$_1$
`filspa\be2.txt`	teszt-bemeneti fájl$_2$
`filspa\ki1.txt`	teszt-kimeneti fájl$_1$
`filspa\ki2.txt`	teszt-kimeneti fájl$_2$
`filspa\feladat.pdf`	a feladat szövege
`filspa\ filspa.depend`	

filspa\ filspa.layout filspa\ main.o filspa\ filspa.cbp	
filspa\dokumentáció.pdf	dokumentációk (ez a fájl)

Megoldás

Programparaméterek

Konstans

```
MaxN            : Egész(10000)[a maximál szám]
```

Típus

```
szakasz         = Tömb(1..MaxN:Egész)
result          = Rekord(bal,jobb:Egész)
choice          = Rekord (egész)
b               = Tömb(1..MaxN:Egész)
c               = Tömb(1..MaxN:Egész)
```

Változó

```
n               : Egész
j               : egész
k               : egész
i               : egész
```

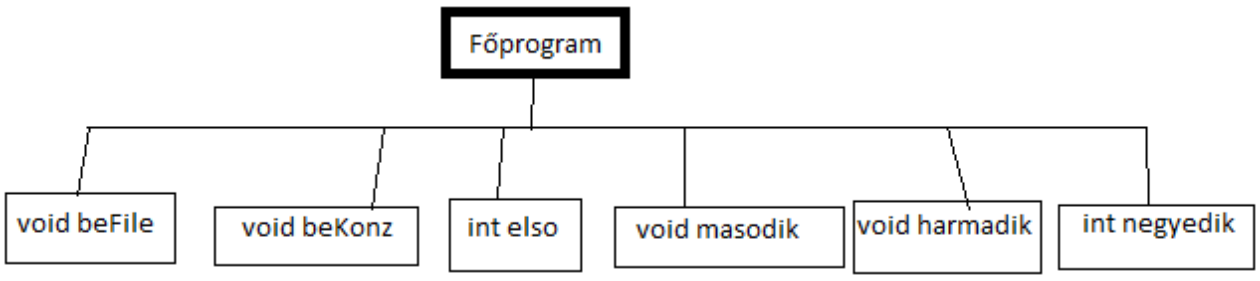
Programfelépítés

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

fstream - file

Függvénystruktúra



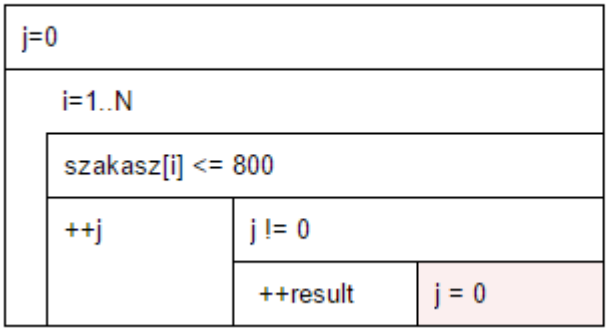
A teljes program algoritmusa

Főprogram:

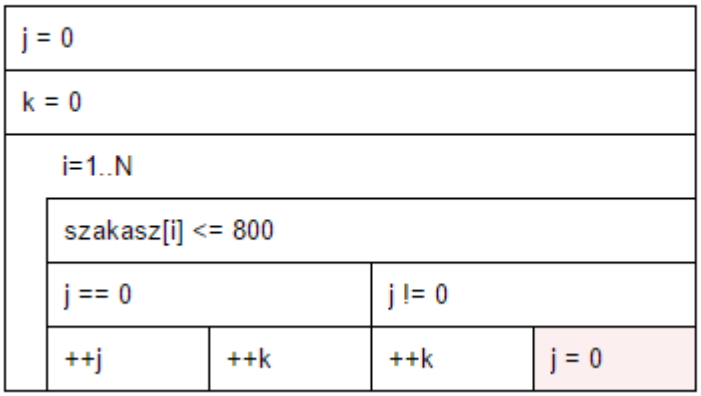
Szakaszok
beFile(n, szakasz)
beKonz(n, szakasz)
a = elso(n, szakasz)
masodik(n, szakasz, b)
harmadik(n, szakasz, c)
d = negyedik(n, c)

Alprogramok:

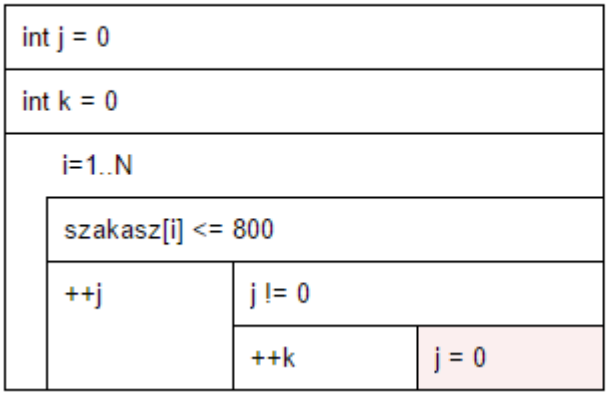
elso



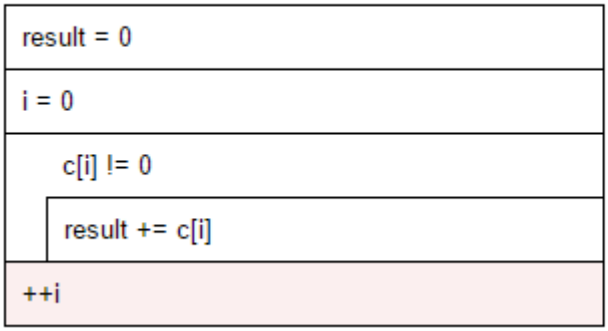
masodik



harmadik



negyedik



A kód

A `main.cpp` fájl tartalma:

```
/*
Prorok Ernest
filspa
ernest.prorok@gmail.com
„ProgAlap beadandó feladatok” téma: Árvízmentes szakaszok teljes hossza
*/
#include <iostream>
#include <fstream>
#include <stdlib.h>

using namespace std;

const int MAXN = 10000;

int menu();

void beFile(int &, int []);
void beKonz(int &, int []);

int elso(const int, const int []);
void masodik(const int, const int [], int []);
void harmadik(const int, const int [], int []);
int negyedik(const int, const int []);

int main() {
    int n, choice;
    int szakasz[MAXN];
    int a, d;
    int b[MAXN] = {0};
    int c[MAXN] = {0};


    choice = menu();
    switch(choice) {
        case 1:
            beFile(n, szakasz);
```

```
            break;

        case 2:
           beKonz(n, szakasz);
           break;
    }

    a = elso(n, szakasz);
    masodik(n, szakasz, b);
    harmadik(n, szakasz, c);
    d = negyedik(n, c);

    cout << a << endl;
    if (a != 0) {
        for (int i = 0; i < n; ++i) {
            if (b[i] != 0) cout << b[i] << " ";
        } cout << endl;
        for (int i = 0; i < n; ++i) {
            if (c[i] != 0) cout << c[i] << " ";
        } cout << endl;
        cout << d;
    }

    return 0;
}

int menu() {
    int result;

    cerr << "VALASSZ A BEMENETEK KOZOTT!\n";
    cerr << "FAJL (1) VAGY KONZOL (2): ";

    do {
        cin >> result;

        if (result > 2 || result < 1 || cin.fail()) {
            cerr << "HIBAS MENUPONT VALASZTAS! A KET OPCIO KOZUL VALASSZ!\n";

            cin.clear();
            cin.ignore(1000, '\n');
```

```
        }
    } while(result > 2 || result < 1 || cin.fail());

    return result;
}

void beFile(int &n, int szakasz[]) {
    string fname;
    ifstream infile;

    cerr << "ADD MEG A BEMENETI FAJLT: ";
    do {
        cin >> fname;

        infile.clear();
        infile.open(fname.c_str());

        if (infile.fail()) cerr << "HIBAS BEMENETI FAJL, A MEGADOTT FAJL NEM
TALALHATO.\n";
    } while(infile.fail());

    infile >> n;
    for (int i = 0; i < n; ++i) {
        infile >> szakasz[i];
    }
}

void beKonz(int &n, int szakasz[]) {
    cerr << "A BEMENETI ADATOK SZAMA: ";
    do {
        cin >> n;

        if (n > MAXN || n < 0 || cin.fail()) {
            cerr << "HIBAS BEMENETI ADAT (0 <= N <= MAXN)\n";

            cin.clear();
            cin.ignore(1000, '\n');
        }
    } while(n > MAXN || n < 0 || cin.fail());

    for (int i = 0; i < n; ++i) {
```

```
        cerr << "ADJA MEG A(Z) " << i+1 << ". BEMENETI ADATOT: ";

        do {
            cin >> szakasz[i];

            if (szakasz[i] > 3000 || szakasz[i] < 0 || cin.fail()) {
                cerr << "HIBAS BEMENETI ADAT! A VIZALLAS 0 ES 3000 KOZE KELL
ESSEN!\n";

                cin.clear();
                cin.ignore(1000, '\n');
            }
        } while(szakasz[i] > 3000 || szakasz[i] < 0 || cin.fail());
    }
}

int elso (const int n, const int szakasz[]) {
    int result = 0;
    int j = 0;

    for (int i = 0; i < n; ++i) {
        if (szakasz[i] <= 800) ++j;
        else {
            if (j != 0) ++result;

            j = 0;
        }
    }

    if (j != 0) {
                cerr << szakasz[n];
                ++result;
    }

    return result;
}

void masodik(const int n, const int szakasz[], int b[]) {
    int j = 0;
    int k = 0;
```

```
    for (int i = 0; i < n; ++i) {
        if (szakasz[i] <= 800) {
            if (j == 0) {
                b[k] = i+1;
                ++k;
            }

            ++j;
        } else {
            if (j != 0) {
                b[k] = i;
                ++k;
            }

            j = 0;
        }
    }

    if (j != 0) b[k] = n;
}

void harmadik(const int n, const int szakasz[], int c[]) {
    int j = 0;
    int k = 0;

    for (int i = 0; i < n; ++i) {
        if (szakasz[i] <= 800) ++j;
        else {
            if (j != 0) {
                c[k] = j;
                ++k;
            }

            j = 0;
        }
    }

    if (j != 0) c[k] = j;
}
```

```
int negyedik(const int n, const int c[]) {
    int result = 0;

    int i = 0;
    while (c[i] != 0) {
        result += c[i];
        ++i;
    }

    return result;
}
```

Tesztelés

Érvényes tesztesetek

1. teszteset: be1.txt

Bemenet
30 888 45 1082 170 42 1864 1813 666 1143 1363 42 53 662 1375 1166 954 1554 1726 1769 1383 1361 2000 374 258 291 259 1841 1792 1585 1517
Kimenet
5 2 2 4 5 8 8 11 13 23 26 1 2 1 3 4 11

2. teszteset: be2.txt

Bemenet

100
286
346
1986
1899
1914
1360
1511
865
321
708
1298
1574
597
1162
1372
620
450
1558
1135
147
587
916
1280
370
1593
1197
901
613
1279
1642
1101
739
1037
1726
36
1568
738
834
1767
1060
86
1710
1167
931
1359
451
216
1539

1627
729
1464
497
936
397
542
1513
1055
1135
327
1256
129
1498
1155
77
395
1193
841
1188
137
1380
1789
51
1868
569
1615
1489
1634
457
1709
1604
771
1787
492
691
1700
588
1556
1502
458
1572
1843
1077
1825
1650
1936
1443
1237
76

61 1152 662
Kimenet
27 1 2 9 10 13 13 16 17 20 21 24 24 28 28 32 32 35 35 37 37 41 41 46 47 50 50 52 52 54 55 59 59 61 61 64 65 69 69 72 72 74 74 78 78 81 81 83 84 86 86 89 89 98 99 2 2 1 2 2 1 1 1 1 1 1 2 1 1 2 1 1 2 1 1 1 1 1 2 1 1 2 36

3. teszteset: be3.txt

Bemenet
N = 7 mérési eredmény1 = 110 mérési eredmény2 = 120 mérési eredmény3 = 850 mérési eredmény4 = 110 mérési eredmény5 = 120 mérési eredmény6 = 150
Kimenet
2 1 2 4 7 2 4 6

Érvénytelen tesztesetek

4. teszteset

Bemenet
N = 11tizenegy
Kimenet
Újrakérdezés: N =

5. teszteset

Bemenet
N = 11 mérési eredmény1 = -1
Kimenet
Újrakérdezés: mérési eredmény1 =

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.