

**„Programozási alapismeretek”
beadandó feladat:
„Versenyess feladat”**

*Készítette: Skorka Bence
Neptun-azonosító: DOMJ1R
E-mail: skorka.bence@gmail.com*

*Kurzuskód: IP-08PAEG
Gyakorlatvezető neve: Menyhárt László*

2016. december 3.

Tartalom

| | |
|-----------------------------------|----|
| Felhasználói dokumentáció..... | 3 |
| Feladat..... | 3 |
| Futási környezet | 3 |
| Használat..... | 3 |
| A program indítása | 3 |
| A program bemenete | 3 |
| A program kimenete..... | 3 |
| Minta bemenet és kimenet | 4 |
| Hibalehetőségek | 4 |
| Fejlesztői dokumentáció | 5 |
| Feladat..... | 5 |
| Fejlesztői környezet | 5 |
| Forráskód | 5 |
| Megoldás..... | 6 |
| Programparaméterek | 6 |
| Programfelépítés | 6 |
| Függvénystruktúra | 6 |
| A teljes program algoritmus | 7 |
| A kód..... | 8 |
| Tesztelés | 11 |
| Érvényes tesztesetek | 11 |
| Fejlesztési lehetőségek..... | 12 |

Felhasználói dokumentáció

Feladat

Egy iskolában egyéni és összetett tanulmányi versenyt tartottak. A versenyekben összesen N tanuló vett részt. A versenyek száma M . Ismerjük versenyenként az induló tanulókat és elért pontszámukat. Az összetett versenyben csak azon tanulók eredményét értékelik, akik az összes egyéni versenyen indultak és elérték a versenyenként adott minimális pontszámot.

Készíts programot, amely megadja azon tanulókat, akik sehol sem érték el a minimális pontszámot!

Futási környezet

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

Használat

A program indítása

A program az `DOMJ1R\bin\Release\DOMJ1R.exe` néven található a tömörített állományban. A `DOMJ1R.exe` fájl kiválasztásával indítható.

A program bemenete

A program az adatokat a **billentyűzetről** olvassa be a következő sorrendben:

| # | Adat | Magyarázat |
|-----|---|---|
| 1. | $N M$ | A tanók száma ($1 \leq N \leq 100$). A versenyek száma ($1 \leq M \leq 100$). |
| 2. | $Min_1 Min_2 \dots Min_m$ | A versenyeken szükséges minimum pontszám ($0 \leq Min_i \leq 50$) |
| 3. | $Ind_1 S_1 P_1 S_2 P_2 \dots S_{ind1} P_{ind1}$ | Az 1. versenyen indulók pontszámai |
| 4. | $Ind_2 S_1 P_1 S_2 P_2 \dots S_{ind2} P_{ind2}$ | A 2. versenyen indulók pontszámai |
| ... | | |
| 6. | $Ind_m S_1 P_1 S_2 P_2 \dots S_{indm} P_{indm}$ | Az n . versenyen indulók pontszámai |

A program kimenete

A program kiírja azoknak a diákoknak a sorszámain, akik egy versenyen sem érték el a minimum pontszámot.

Minta bemenet és kimenet

```
Tanulok szama: 5
Versenyek szama: 3
1. verseny minimum pontszama: 10
2. verseny minimum pontszama: 20
3. verseny minimum pontszama: 20
Hanyan indultak az 1. versenyen? 3
1. verseny 1. indulojanak azonositoja: 1
Es pontszama: 10
1. verseny 2. indulojanak azonositoja: 2
Es pontszama: 30
1. verseny 3. indulojanak azonositoja: 3
Es pontszama: 10
Hanyan indultak az 2. versenyen? 2
2. verseny 1. indulojanak azonositoja: 2
Es pontszama: 10
2. verseny 2. indulojanak azonositoja: 1
Es pontszama: 10
Hanyan indultak az 3. versenyen? 4
3. verseny 1. indulojanak azonositoja: 1
Es pontszama: 10
3. verseny 2. indulojanak azonositoja: 2
Es pontszama: 20
3. verseny 3. indulojanak azonositoja: 3
Es pontszama: 30
3. verseny 4. indulojanak azonositoja: 5
Es pontszama: 50
```

Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha nem a mintának megfelelő adatot, vagy hibás számú adatokat viszünk be. A program hiba esetén újrakérdezi az adatot.

Fejlesztői dokumentáció

Feladat

Egy iskolában egyéni és összetett tanulmányi versenyt tartottak. A versenyekben összesen N tanuló vett részt. A versenyek száma M . Ismerjük versenyenként az induló tanulókat és elért pontszámukat. Az összetett versenyben csak azon tanulók eredményét értékelik, akik az összes egyéni versenyen indultak és elérték a versenyenként adott minimális pontszámot.

Készíts programot, amely megadja azon tanulókat, akik sehol sem érték el a minimális pontszámot!
Specifikáció

Bemenet: $N \in \mathbb{N}, M \in \mathbb{N}, \text{Min}_i \in \mathbb{N}, S_i \in \mathbb{N}, P_i \in \mathbb{N}$

Kimenet: $\text{db} \in \mathbb{N}, \text{ID}_1 \in \mathbb{N}, \text{ID}_2 \in \mathbb{N}, \text{ID}_3 \in \mathbb{N} \dots, \text{ID}_{\text{db}} \in \mathbb{N}$

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

Utófeltétel: $DB = \sum_{i=1}^{\text{tanulokSzama}} \text{NemNyert}(i)$
 $\forall i \ (1 \leq i \leq \text{tanulokSzama}): \text{NemNyert}(i)$

Definíció: $\text{NemNyert}(n): \forall i \ (1 \leq i \leq \text{versenyekSzama}) \ \text{verseny}[i].\text{pontszam}[n] < \text{verseny}[i].\text{minimum}$

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 `DOMJ1R` nevű könyvtárban található meg. A fejlesztés során használt könyvtár-struktúra:

| Állomány | Magyarázat |
|---------------------------------|----------------------------------|
| DOMJ1R\bin\Release\Solution.exe | futtatható kód |
| DOMJ1R\obj\Release\main.o | félig lefordított kód |
| DOMJ1R\main.cpp | C++ forráskód |
| DOMJ1R\be1.txt | teszt-bemeneti fájl ₁ |
| DOMJ1R\be2.txt | teszt-bemeneti fájl ₂ |
| DOMJ1R\DOMJ1R.pdf | dokumentációk (ez a fájl) |

Megoldás

Programparaméterek

Konstans

MaxN : **Egész** (256) [a tanulók és a versenyek maximális száma]

Struktúrák

Tanulo = **Rekord**(sikerultEgyVerseny:**Logikai**)
Verseny = **Rekord**(minimumPontszam:**Egész**)

Változó

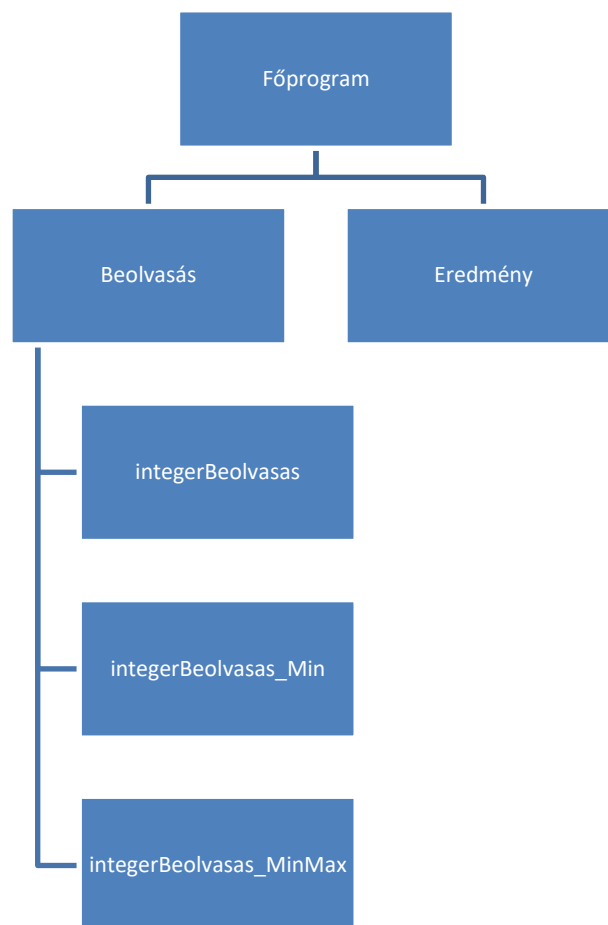
versenyekSzama : **Egész**
tanulokSzama : **Egész**
versenyek : Verseny^N
tanulok : Tanulo^N

Programfelépítés

A program által használt modulok (és helyük):

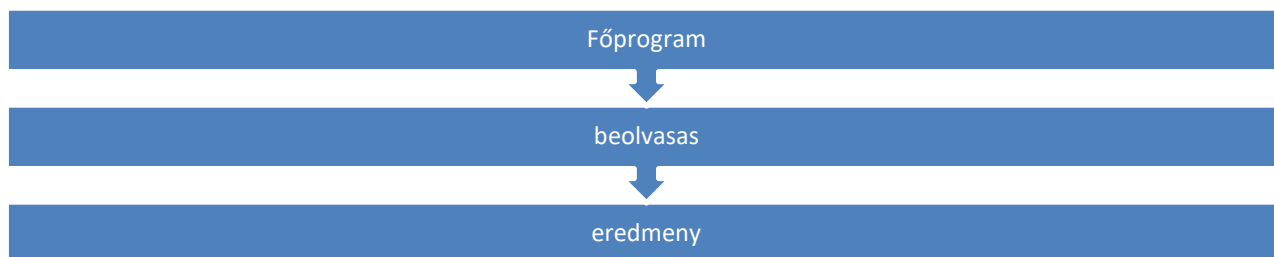
main.cpp – program, a forráskönyvtárban
iostream – képernyő-, és billentyűkezelés, a C++ rendszer része

Függvénystruktúra



A teljes program algoritmusa

Főprogram:



Alprogramok:

| | | | | | |
|--|--|--|--|--|---|
| Beolvas(versenyek: Verseny ^N , Tanulok[maxN]: Tanulo ^N , versenyekSzama: egesz, tanulokSzama: egesz) | | | | | |
| Be: tanulokSzama | | | | | |
| Be: versenyekSzama | | | | | |
| I = 1..versenyekSzama | | | | | |
| | Be: versenyek[I].minimumPontszam | | | | |
| I = 1..versenyekSzama | | | | | |
| | Be: indultTanulokSzama | | | | |
| | J = 1..indultTanulokSzama | | | | |
| | | Be: tanuloid | | | |
| | | Be: pontszam | | | |
| | I | Ha (pontszam > versenyek[I].minimumPontszam) | | | H |
| | tanulok[tanuloid].sikerultEgyVerseny := igaz | | | | |

| | | | |
|--|--------------------------|---|---|
| eredmeny(konstans versenyek: Verseny ^N , konstans tanulok: Tanulok ^N , versenyekSzama: egesz, tanulokSzama: egesz) | | | |
| Szamolas := 0 | | | |
| I = 1..tanulokSzama | | | |
| | I | Ha (tanulok[I].sikerultEgyVerseny == HAMIS) | H |
| | Szamolas := Szamolas + 1 | | |
| Kiír: Szamolas | | | |
| I = 1.. tanulokSzama | | | |
| | I | Ha (tanulok[I].sikerultEgyVerseny == HAMIS) | H |
| | Kiír: I | | |

A kód

A `main.cpp` fájl tartalma:

```
/*
GitHub: http://github.com/SkorkaBence/ProgalapBeadando
*/
#include <iostream>

using namespace std;

const int maxN = 256;

struct Tanulo {
public:
    bool sikerultEgyVerseny = false;
};

struct Verseny {
public:
    int minimumPontszam;
};

// beolvasó algoritmusok

void integerBeolvasas(int &N);
void integerBeolvasas_Min(int &N, int minimum);
void integerBeolvasas_MinMax(int &N, int minimum, int maximum);

// feladatmegoldás lépéseit tartalmazó algoritmusok
void beolvasas(Verseny versenyek[], Tanulo tanulok[maxN], int &versenyekSzama, int &tanulokSzama);
void eredmény(const Verseny versenyek[], const Tanulo tanulok[maxN], int versenyekSzama, int tanulokSzama);

int main()
{
    // adatokat tároló változók
    int versenyekSzama;
    int tanulokSzama;
    Verseny versenyek[maxN];
    Tanulo tanulok[maxN];

    beolvasas(versenyek, tanulok, versenyekSzama, tanulokSzama);
    eredmény(versenyek, tanulok, versenyekSzama, tanulokSzama);

    return 0;
}

void integerBeolvasas(int &N) {
    bool hiba = false;
    do {
        cin >> N;
        hiba = cin.fail();
        if (hiba) {
            cin.clear();
            cin.ignore(1000, '\n');
            cerr << "Hibas bemenet, egész szám szükséges!" << endl;
        }
    } while (hiba);
}

void integerBeolvasas_Min(int &N, int minimum) {
    bool hiba = false;
    do {
        cin >> N;
        hiba = cin.fail();
        if (!hiba) {
            if (N < minimum) {
                hiba = true;
            }
        }
    }
    if (hiba) {
        cin.clear();
        cin.ignore(1000, '\n');
        cerr << "Hibas bemenet, egész szám szükséges, amely minimum " << minimum << "!" << endl;
    }
}
```



```

    } while (hiba);
}

void integerBeolvasas_MinMax(int &N, int minimum, int maximum) {
    bool hiba = false;
    do {
        cin >> N;
        hiba = cin.fail();
        if (!hiba) {
            if ((N < minimum) || (N > maximum)) {
                hiba = true;
            }
        }
        if (hiba) {
            cin.clear();
            cin.ignore(1000, '\n');
            cerr << "Hibas bemenet, egész szám szükséges, amely minimum " << minimum << " és maximum " << maximum << "!" << endl;
        }
    } while (hiba);
}

void beolvasas(Verseny versenyek[], Tanulo tanulok[maxN], int &versenyekSzama, int &tanulokSzama) {
    // egyéb változók
    int minimumPontszam;
    int indultTanulokSzama;
    int tanuloid;
    int pontszam;

    // tanulók számának bekérdezése, és a változókat tároló tömb előkészítése
    cerr << "Tanulok száma: ";
    integerBeolvasas_MinMax(tanulokSzama, 0, maxN);
    for (int i = 0; i < tanulokSzama; i++) {
        tanulok[i] = Tanulo();
    }

    // versenyek számának bekérdezése, és a versenyeket tároló tömb előkészítése, majd feltöltése a minimum pontszámokkal
    cerr << "Versenyek száma: ";
    integerBeolvasas_MinMax(versenyekSzama, 0, maxN);
    for (int i = 0; i < versenyekSzama; i++) {
        cerr << (i+1) << ". verseny minimum pontszama: ";
        integerBeolvasas_Min(minimumPontszam, 0);
        versenyek[i] = Verseny();
        versenyek[i].minimumPontszam = minimumPontszam;
    }

    // a versenyek adatainak bekérdezése
    for (int i = 0; i < versenyekSzama; i++) {
        // hányan indultak a versenyen
        cerr << "Hányan indultak az " << (i+1) << ". versenyen? ";
        integerBeolvasas_MinMax(indultTanulokSzama, 0, tanulokSzama);
        for (int j = 0; j < indultTanulokSzama; j++) {
            // a versenyen induló pontszámainak bekérdezése
            cerr << (i+1) << ". verseny " << (j+1) << ". indulójának azonosítója: ";
            integerBeolvasas_MinMax(tanuloid, 1, tanulokSzama + 1);
            cerr << "Es pontszama: ";
            integerBeolvasas_Min(pontszam, 0);

            if (pontszam >= versenyek[i].minimumPontszam) { // elérte a minimum pontszámot?
                // sikerült neki!
                tanulok[tanuloid-1].sikerultEgyVerseny = true;
            }
        }
    }
}

void eredmény(const Verseny versenyek[], const Tanulo tanulok[maxN], int versenyekSzama, int tanulokSzama) {
    int szamolas = 0;
    // hány tanuló nem érte el a minimum pontot?
    for (int i = 0; i < tanulokSzama; i++) {
        if (!tanulok[i].sikerultEgyVerseny) {
            szamolas++;
        }
    }

    // eredmény kiírása
    cerr << endl << "Minimum pontszámot sehol el nem erok szama: ";
}

```

```
cout << szamolas;
cerr << endl << "Es az azonositoik:";
for (int i = 0; i < tanulokSzama; i++) {
    if (!tanulok[i].sikerultEgyVerseny) {
        cout << " " << i + 1;
    }
}
}
```

Tesztelés

Érvényes tesztesetek

1. *teszteset: be1.txt*

| Bemenet |
|---|
| tanulokSzama = 5 versenyekSzama = 3 minimum[1] = 10 minimum[2] = 20 minimum[3] = 20 IndulokSzama[3] = 5 InduloSorszam[0,1] = 1 InduloPontszama[0,1] = 10 InduloSorszam[0,2] = 2 InduloPontszama[0,2] = 30 InduloSorszam[0,3] = 3 InduloPontszama[0,3] = 10 IndulokSzama[2] = 5 InduloSorszam[1,1] = 2 InduloPontszama[1,1] = 10 InduloSorszam[1,2] = 1 InduloPontszama[1,2] = 10 IndulokSzama[4] = 5 InduloSorszam[2,1] = 1 InduloPontszama[2,1] = 10 InduloSorszam[2,2] = 2 InduloPontszama[2,2] = 20 InduloSorszam[2,3] = 3 InduloPontszama[2,3] = 30 InduloSorszam[2,4] = 5 InduloPontszama[2,4] = 50 |
| Kimenet |
| 2 1 4 |

2. *teszteset: be2.txt*

| Bemenet - Részlet |
|--|
| tanulokSzama = 100 versenyekSzama = 100 minimum[1] = 8 minimum[2] = 9 minimum[3] = 7 minimum[4] = 9 minimum[5] = 11 Teljes bemenet: https://github.com/SkorkaBence/ProgalapBeadando/blob/master/tesztesetek/be2_leirva.txt |
| Kiment |
| 5 22 40 45 50 76 |

Fejlesztési lehetőségek

1. Adatok –a felhasználó igénye szerint– akár fájlból is fogadása.
2. Hibás fájl-bemenetek felismerése, és a hiba helyének (sor sorszámanak) kiírása.
3. Többszöri futtatás megszervezése