

”Programozási alapismeretek”
beadandó feladat:
”ProgAlap beadandó” téma 1. feladat

Készítette: Bárdosi Bence
Neptun-azonosító: VY9NJN
E-mail: bardosi.bence@gmail.com

Kurzuskód: IP-08PAEG
Gyakorlatvezető neve: Pap Gábor Sándorné

2016-11-31

Tartalom

1	Felhasználói dokumentáció	2
1.1	Feladat	2
1.2	Futási környezet	2
1.3	Használat	2
1.3.1	A program indítása	2
1.3.2	A program bemenete	2
1.3.3	A program kimenete	3
1.3.4	Minta bemenet és kimenet	3
1.3.5	Hibalehetőségek	3
2	Fejlesztői dokumentáció	4
2.1	Feladat	4
2.2	Specifikáció	4
2.3	Fejlesztői környezet	4
2.4	Forráskód	5
2.5	Megoldás	5
2.5.1	Programparaméterek	5
2.5.2	Programfelépítés	5
2.5.3	Függvénystruktúra	5
2.5.4	Algoritmus	6
2.5.5	A kód	7
2.6	Tesztelés	10
2.6.1	Érvényes tesztesetek	10
2.6.2	Érvénytelen tesztesetek	10
2.7	Fejlesztési lehetőségek	10

1 Felhasználói dokumentáció

1.1 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 az egyéni versenyek győzteseinek rangsorát!

1.2 Futási környezet

IBM PC, exe futtatására alkalmas, 32-bites operációs rendszer (pl. Windows 10). Nem igényel egeret.

1.3 Használat

1.3.1 A program indítása

A program a `.\VY9NJN\bin\Release\VY9NJN.exe` néven található a tömörített állományban. A `VY9NJN.exe` fájl kiválasztásával indítható.

1.3.2 A program bemenete

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

Table 1: Bemenet

#	Adat	Magyarázat
1	N	Tanulók száma ($1 \leq N \leq 100$)
2	M	Versenyszek száma ($1 \leq M \leq 100$)
3	Min_1	Az 1. verseny minimum ponthatára ($0 \leq Min_1 \leq 50$)
4	Min_2	A 2. verseny minimum ponthatára ($0 \leq Min_2 \leq 50$)
.		
.		
.		
$M + 2$	Min_M	Az M . verseny minimum ponthatára ($0 \leq Min_M \leq 50$)
$M + 2 + 1$	$Para_1$	Az 1. verseny paraméterei (lásd: table2)
$M + 2 + 2$	$Para_2$	A 2. verseny paraméterei (lásd: table2)
.		
.		
.		
$M + 2 + M$	$Para_M$	Az M . verseny paraméterei (lásd: table2)

Table 2: Egy adott verseny paramétereit

#	Adat	Magyarázat
1	Ind_i	A versenyen indulók száma ($1 \leq Ind_i \leq N$)
2	$S_{i,1}$	Az első tanuló sorszáma ($1 \leq S_{i,1} \leq N$)
3	$P_{i,1}$	Az első tanuló által elért pont ($1 \leq P_{i,1} \leq 100$)
4	$S_{i,2}$	A második tanuló sorszáma ($1 \leq S_{i,2} \leq N$)
5	$P_{i,2}$	A második tanuló által elért pont ($1 \leq P_{i,2} \leq 100$)
.	.	.
.	.	.
.	.	.
$2 * Ind_i$	S_{i,Ind_i}	Az Ind_i . tanuló sorszáma ($1 \leq S_{i,1} \leq N$)
$2 * Ind_i + 1$	P_{i,Ind_i}	Az Ind_i . tanuló által elért pont ($1 \leq P_{i,1} \leq 100$)

1.3.3 A program kimenete

A program kiírja az egyéni versenyek győzteseinek rangsorát. A kimenet első sorába az egyéni győzelmet elért tanulók számát, amelyet a győztesek sorszáma követi, győzelmek száma szerint csökkenő, azon belül sorszám szerint növekvő sorrendben.

1.3.4 Minta bemenet és kimenet

```

Versenyek győztese rangsora
Tanulok (N) es versenyek (M) szama szokozel elvalasztva (N=[1..100], M=[1..100]):5 3
A 3 verseny minimum pontthatarai (Mi) szokozel elvalasztva (Mi=[0..50]):10 10 10
Kerem soronkent adja be a 3 verseny parametereit szokozel elvalasztva a kovetkezo modon:
    indulok szama [1..5]
    versenyzo sorszama [1..5] versenyzo eredménye [1..100]
A(z) 1. verseny parameterei:3 1 10 2 30 3 10
A(z) 2. verseny parameterei:2 4 50 1 30
A(z) 3. verseny parameterei:5 1 10 2 20 3 30 4 50 5 50
Az egyeni gyozelmeket elertek szama, es sorszamaik gyozelmek szama szerint csokkeno, azon belul
sorszam szerint novekvo sorrendben:
3 4 2 5

```

1.3.5 Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha a bármely bemenő adat nem egész szám, nem esik az adott intervallumba, vagy ha nem szám. Hiba esetén a program "HIBA!"-t jelez és újrakérdezi az adott adatot.

Mintafutás hibás bemeneti adatok esetén:

```

Versenyek győztese rangsora
Tanulok (N) es versenyek (M) szama szokozel elvalasztva (N=[1..100], M=[1..100]):öt három
HIBA!
Tanulok (N) es versenyek (M) szama szokozel elvalasztva (N=[1..100], M=[1..100]):5 3
A 3 verseny minimum pontthatarai (Mi) szokozel elvalasztva (Mi=[0..50]):10 10 10
Kerem soronkent adja be a 3 verseny parametereit szokozel elvalasztva a kovetkezo modon:
    indulok szama [1..5]
    versenyzo sorszama [1..5] versenyzo eredménye [1..100]
A(z) 1. verseny parameterei:3 1 10 2 30 3 10
A(z) 2. verseny parameterei:2 4 50 1 30
A(z) 3. verseny parameterei:5 1 10 2 20 3 30 4 50 5 50
Az egyeni gyozelmeket elertek szama, es sorszamaik gyozelmek szama szerint csokkeno, azon belul
sorszam szerint novekvo sorrendben:
3 4 2 5

```

2 Fejlesztői dokumentáció

2.1 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 az egyéni versenyek győzteseinek rangsorát!

2.2 Specifikáció

Be : $N \in \mathbb{N}$

$M \in \mathbb{N}$

$Min \in \mathbb{N}^M$

$P \in \mathbb{N}^{N \times M}$

Ki : $T \in \mathbb{N}$

$Nyert \in \mathbb{N}^T$

EF : $N \in [1..100]$

$M \in [1..100]$

$\forall i \in [1..M] : Min_i \in [0..50]$

UF : $T = \sum_{i=1}^N 1 \wedge$

$Db(i) \geq 1$

$Nyert \subseteq [1..N] \wedge$

$\forall i \in [1..T] : Db(Nyert_i) \geq 1 \wedge$

$\forall i \in [1..(T-1)] : Nyert_i ' > ' Nyert_{i+1}$

Def : $Db(x) = \sum_{j=1}^M 1$

$P_{x,j} = Max(P_{[1..N],j})$

$x ' > ' y \Leftrightarrow \begin{cases} Db(x) \neq Db(y) : Db(x) > Db(y) \\ Db(x) = Db(y) : x < y \end{cases}$

$Max(P_{X,y}) = (k > 0 \mid \exists i \in X : k = P_{i,y} \wedge \forall i \in X : k \geq P_{i,y})$

2.3 Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 10).
mingw32-g++.exe c++ fordítóprogram (v4.9.2), Code::Blocks (v16.01) fejlesztői környezet.

2.4 Forráskód

A teljes fejlesztői anyag –kicsomagolás után– a **ProgAlap_bead** nevű könyvtárban található meg. A fejlesztés során használt könyvtár-struktúra:

Table 3: könyvtár-struktúra

Állomány	Magyarázat
.\VY9NJN\main.cpp	C++ forráskód
.\LaTeX\Dokumentacio.tex	Ezen dokumentáció LaTeX kódja
.\Dokumentacio.pdf	Ez a fájl
.\minta\	Mintabemeneteket tartalmazó könyvtár

2.5 Megoldás

2.5.1 Programparaméterek

Konstans

MERET : **Egész** (100) [tanulók és versenyek maximális száma]

Változó

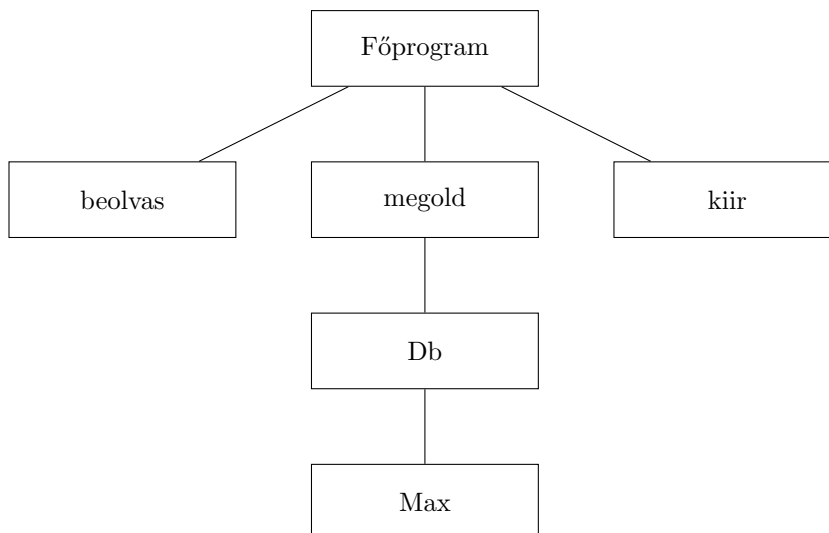
N : **Egész**
M : **Egész**
Min : M hosszú **vektor**
P : NxM méretű **mátrix**

2.5.2 Programfelépítés

A program által használt modulok (és helyük):

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

2.5.3 Függvénystruktúra



2.5.4 Algoritmus

Főprogram:

FOPROGRAM

$beolvas(N, M, Min, P)$
$megold(N, M, T, P, Nyert)$
$kiir(T, Nyert)$

Alprogramok:

$beolvas(N[Egész], M[Egész], Min[M \text{ Vektor}], P[N \times M \text{ Mátrix}])$

$Be : N - [1..MERET]$
$Be : M - [1..MERET]$
$Be : Min[1..M] - [0..50]$
$Be : P[1..N][1..M] - [1..100]$

$megold(N[Egész], M[Egész], T[Egész], P[N \times M \text{ Mátrix}], Nyert[T \text{ Vektor}])$

$T := 0$					
$i := 1..N$					
<table> <tr> <td colspan="2">$Db(i, N, M, P, db) \geq 1$</td></tr> <tr> <td>i</td><td>n</td></tr> </table>		$Db(i, N, M, P, db) \geq 1$		i	n
$Db(i, N, M, P, db) \geq 1$					
i	n				
$T := T + 1$					
$i := 1..N$					
$Nyert[i] := i$					
$i := N..2 \text{ } (-1)esével$					
$j := 1..i$					
<table> <tr> <td colspan="2">$db[j] < db[j + 1]$</td></tr> <tr> <td>i</td><td>n</td></tr> </table>		$db[j] < db[j + 1]$		i	n
$db[j] < db[j + 1]$					
i	n				
$S := db[j]$	\emptyset				
$db[j] := db[j + 1]$					
$db[j + 1] := S$					
$S := Nyert[j]$					
$Nyert[j] := Nyert[j + 1]$					
$Nyert[j + 1] := S$					

Db(i[Egész], N[Egész], M[Egész], P[NxM Mátrix], db[N Vektor])

$s := 0$	
$j := 1..M$	
$P[i][j] = \text{Max}(j, N, P) \text{ és nem } P[i][j] = 0$	
i	n
$s := s + 1$	
$db[i] = s$	
$Db := s$	

Max(j[Egész], N[Egész], P[NxM Mátrix])

$s := P[1][j]$	
$1 := 1..N$	
$P[i][j] > s$	
i	n
$s := P[i][j]$	\emptyset
$Max := s$	

2.5.5 A kód

```

1  #include <iostream>
2
3  using namespace std;
4
5  #define MERET 100
6
7  void beolvas(int &N, int &M, int Min[MERET], int P[MERET][MERET]);
8  void megold(int N, int M, int &T, int P[MERET][MERET], int Nyert[MERET],
9             int db[MERET]);
10 int Max(int j, int N, int P[MERET][MERET]);
11 int Db(int i, int N, int M, int P[MERET][MERET], int db[MERET]);
12 void kiir(int T, int Gyozt[MERET]);
13
14 int main()
15 {
16     //Be
17     int N, M, Min[MERET]= {}, P[MERET][MERET]= {};
18
19     //Ki
20     int T, Nyert[MERET]= {};
21
22     beolvas(N, M, Min, P);
23
24     //Db fv memorizacioja
25     int db[MERET];
26     for(int i=0; i<N; i++)
27         db[i]=-1;
28
29     megold(N, M, T, P, Nyert, db);
30     kiir(T, Nyert);
31
32     return 0;

```



```

32 }
33
34 void beolvas(int &N, int &M, int Min[MERET], int P[MERET][MERET])
35 {
36     bool hiba=false;
37     int s=0, si=0;
38
39     do {
40         cerr << "Tanulok (N) es versenyek (M) szama szokozzel
41             elvalasztva (N=[1..100], M=[1..100]):";
42         cin >> N;
43         cin >> M;
44         hiba= cin.fail() || (cin.peek() != '\n') || (N<1 || N>100) || (M
45             <1 || M>100);
46         if(hiba) {
47             cerr << "HIBA!" << endl;
48             cin.clear();
49             cin.ignore(100, '\n');
50         }
51     } while(hiba);
52
53     do {
54         hiba=false;
55         cerr << "A " << M << " verseny minimum pontthatarai (Mi)
56             szokozzel elvalasztva (Mi=[0..50]):";
57         for(int i=0; i<M; i++) {
58             cin >> Min[i];
59             hiba=hiba || (Min[i]<0 || Min[i]>50) || cin.fail() || ((cin.
60                 peek() != '\n') && cin.peek() != ' ');
61         }
62         if(hiba) {
63             cerr << "HIBA!" << endl;
64             cin.clear();
65             cin.ignore(100, '\n');
66         }
67     } while (hiba);
68
69     cerr << "Kerem soronkent adj be a " << M << " verseny parametereit
70         szokozzel elvalasztva a kovetkezo modon:" << endl;
71     cerr << "\t indulok szama [1.." << N << "]" << endl;
72     cerr << "\t versenyzo sorszama [1.." << N << "]" << " versenyzo
73         eredmenye [1..100]" << endl;
74
75     for(int i=0; i<M; i++) {
76         do {
77             hiba=false;
78             cerr << "A(z) " << i+1 << ". verseny parameterei:";
79             cin >> s;
80             for(int j=0; j<s; j++) {
81                 cin >> si;
82                 hiba = hiba || si<1 || si>N || cin.fail() || ((cin.peek
83                     () != '\n') && cin.peek() != ' ');
84                 cin >> P[si-1][i];

```

```

81         hiba = hiba || P[si-1][i]<0 || P[si-1][i]>100 || cin.
            fail() || ((cin.peek() != '\n') && cin.peek() != ' ');
82         if(P[si-1][i]<Min[i])
83             P[si-1][i]=0;
84     }
85     if(hiba) {
86         cerr << "HIBA!" << endl;
87         cin.clear();
88         cin.ignore(100, '\n');
89     }
90     } while(hiba);
91 }
92 }
93
94 void megold(int N, int M, int &T, int P[MERET][MERET], int Nyert[MERET],
            int db[MERET])
95 {
96     //megszámolás
97     T=0;
98     for(int i=0; i<N; i++)
99         if(Db(i,N,M,P,db)>=1)
100             T++;
101
102     //Nyert vektor inic
103     for(int i=0; i<N; i++)
104         Nyert[i]=i+1;
105
106     //buborekrendezés
107     //Egyszerre rendezi a nyert es a db halmazokat, így a nyert
        halmazban a tanulok indexei is "helyükre kerülnek"
108     for(int i=N-1; i>0; i--)
109         for(int j=0; j<i; j++)
110             if(db[j]<db[j+1]) {
111                 int S=db[j];
112                 db[j]=db[j+1];
113                 db[j+1]=S;
114                 S=Nyert[j];
115                 Nyert[j]=Nyert[j+1];
116                 Nyert[j+1]=S;
117             }
118 }
119
120 int Max(int j, int N, int P[MERET][MERET])
121 {
122     int s=P[0][j];
123     for(int i=1; i<N; i++)
124         if(P[i][j]>s)
125             s=P[i][j];
126     return s;
127 }
128
129 int Db(int i, int N, int M, int P[MERET][MERET], int db[MERET])
130 {
131     if(db[i]!=-1)
132         return db[i];
133     int s=0;

```

```

134     for(int j=0; j<M; j++)
135         if(P[i][j]==Max(j, N, P) && P[i][j]!=0)
136             s++;
137     db[i]=s;
138     return s;
139 }
140
141 void kiir(int T, int Nyert[MERET])
142 {
143     cerr << "Az egyeni gyozelmeket elertek szama, es sorszamaik
           gyozelmek szama szerint csokkeno, azon belül sorszam szerint
           novekvő sorrendben:" << endl;
144     cout << T;
145     if(T>0)
146         cout << " " << Nyert[0];
147     for(int i=1; i<T; i++)
148         cout << " " << Nyert[i];
149 }

```

2.6 Tesztelés

2.6.1 Érvényes tesztesetek

2.6.2 Érvénytelen tesztesetek

2.7 Fejlesztési lehetőségek

- Adatok - a felhasználó igénye szerint - akár fájlból is fogadása.
- Hibás fájl-bemenetek felismerése, és a hiba helyének (sor számának) kiírása.
- Többszöri futtatás megszervezése.
- Külső adatbázis alapján tanulók sorszámának megfeleltetése neveikkel és neveik kiírása.