„Programozás”  
beadandó feladat

Készítette: *Tóth Botond*Neptun-azonosító: *MQH41V*E-mail: tothbotond00@gmail.com

Kurzuskód: IP-18PROGEGGyakorlatvezető neve: Veszprémi Anna

2021. január 1.

Tartalom

[Felhasználói dokumentáció 3](#_Toc410323328)

[Feladat 3](#_Toc410323329)

[Futási környezet 3](#_Toc410323330)

[Használat 3](#_Toc410323331)

[A program indítása 3](#_Toc410323332)

[A program bemenete 3](#_Toc410323333)

[A program kimenete 3](#_Toc410323334)

[Minta bemenet és kimenet 4](#_Toc410323335)

[Hibalehetőségek 4](#_Toc410323336)

[Fejlesztői dokumentáció 5](#_Toc410323337)

[Feladat 5](#_Toc410323338)

[Specifikáció 5](#_Toc410323339)

[Fejlesztői környezet 5](#_Toc410323340)

[Forráskód 6](#_Toc410323341)

[Megoldás 6](#_Toc410323342)

[Programparaméterek 6](#_Toc410323343)

[Programfelépítés 7](#_Toc410323344)

[Függvénystruktúra 7](#_Toc410323345)

[Algoritmus 7](#_Toc410323346)

[A kód 8](#_Toc410323347)

[Tesztelés 14](#_Toc410323348)

[Érvényes tesztesetek 14](#_Toc410323349)

[Érvénytelen tesztesetek 15](#_Toc410323350)

[Fejlesztési lehetőségek 16](#_Toc410323351)

2. Felhasználói dokumentáció
   1. Feladat

Az ország néhány helységében madármegfigyelést végeztünk. Mindegyikben megadtuk, hogy milyen fajú madárból hányat láttunk.

Készíts programot, amely megad egy madárfajt, amely legkevésbé kedveli mások társaságát – átlagosan minimális számú madárfajjal fordult elő!

* 1. Futási környezet

IBM PC, exe futtatására alkalmas, 32-bites operációs rendszer (pl. Windows 7, Windows 10). Nem igényel egeret.

* 1. Használat
     1. A program indítása

A program az MQH41V\bin\Release\MQH41V.exe néven található a tömörített állományban. A MQH41V.exe fájl kiválasztásával indítható.

* + 1. A program bemenete

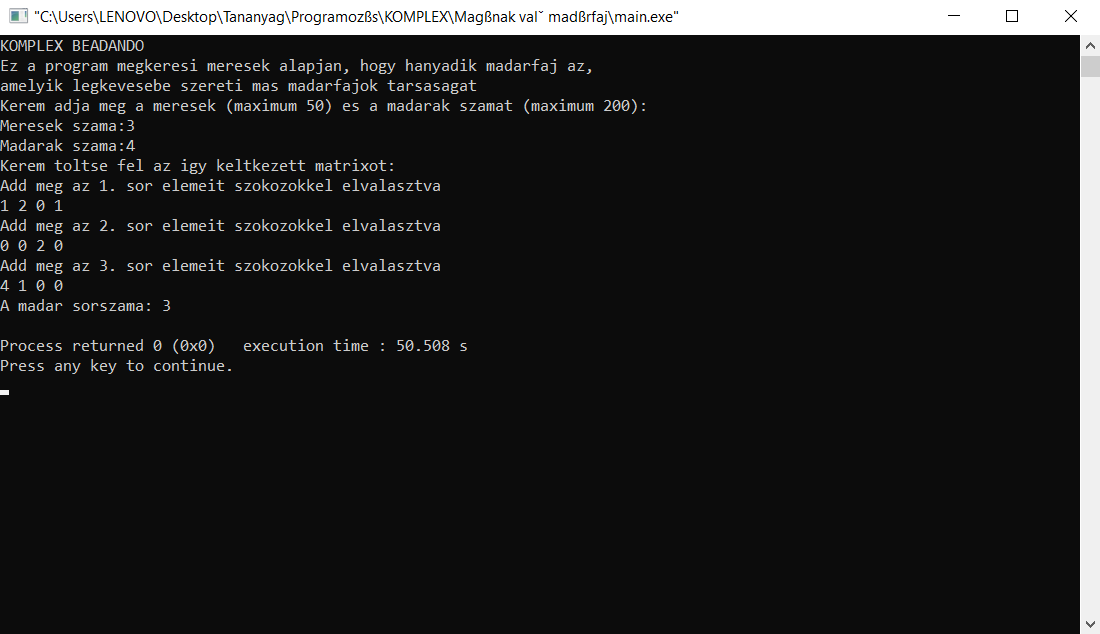
A program az adatokat a **billentyűzet**ről olvassa be a következő sorrendben:

|  |  |  |
| --- | --- | --- |
| # | Adat | Magyarázat |
| **1.** | *N* | A helységek száma (1≤N≤50). |
| **2.** | *M* | Madarak száma (1≤Magasság1≤200). |
| **3.** | *E11* | Első adat (0≤E11≤1000). |
| **4.** | *E12* | Második adat (0≤E12≤1000). |
| **…** | *…* |  |
| **(N\*M)+2.** | *ENM* | Az NM-edik adat (0≤ENM≤1000). |

* + 1. A program kimenete

A program kiírja annak a madárfajnak a sorszámát amelyik legkevésbé szereti mások társaságát, tehát átlagosan minimális számú madárfajjal fordult elő.

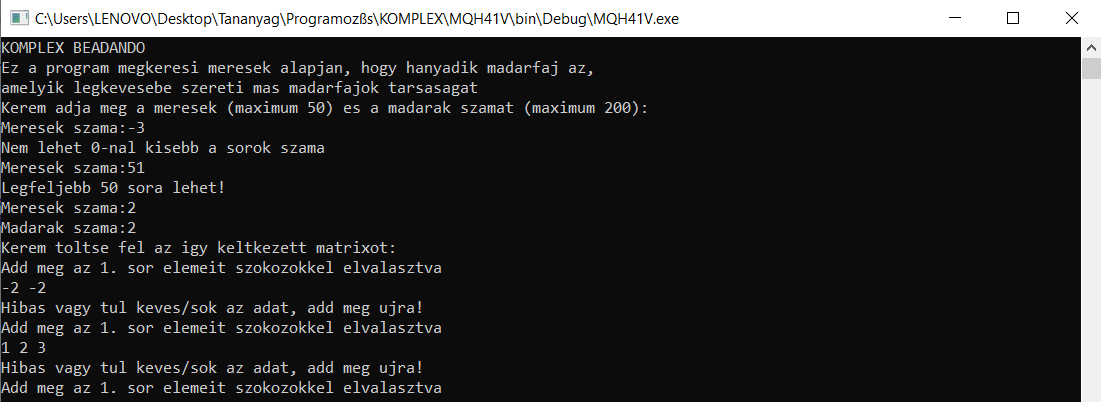
* + 1. Minta bemenet és kimenet



* + 1. Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha az N vagy az M értéke nem esik bele az annak fenntartott intervallumba, ha a mátrixunk értékei nem esnek bele a **0 .. 1000** intervallumba. Lehetséges hiba még, ha túl sok vagy túl kevés adatot írtunk be a mátrix valamelyik sorába. Érdemes arra is figyelni, hogy ne hagyjunk whitespace-t a sorunk megadása után. Az esetleges hibát a program minden esetben jelzi, és utána újra bekéri a várt adatot.

* + - 1. Mintafutás hibás bemeneti adatok esetén:



1. Fejlesztői dokumentáció
   1. Feladat

Az ország néhány helységében madármegfigyelést végeztünk. Mindegyikben megadtuk, hogy milyen fajú madárból hányat láttunk.

Készíts programot, amely megad egy madárfajt, amely legkevésbé kedveli mások társaságát – átlagosan minimális számú madárfajjal fordult elő!

* 1. Specifikáció

**Bemenet**: n∈N, m ∈N , meresek∈Nnxm\*

**Kimenet**: minimum∈N,

**Előfeltétel**: 0<n<=maxsor, 0<m<=maxoszlop, minden ∀i és j∈ [1..n,1..m:Egész]: meresekij>0

**Utófeltétel**: Listázzuk ki egy segédtömbe annak az átlagát hogy egy madár hányszor találkozott más madárfajokkal, majd vegyük ennek a minimumát! Ha a madár volt jelen egyszer sem akkor ne vegyük figyelembe az eredményt (ezzel elkerüljük a nullázal való osztást).

**Tételek**: Minimum keresésben megszámlálás.

**Megjegyzés**: Abban az esetben ha mondjuk mindenhol 0 madár van vagy egyenlő számban fordulnak elő madarak akkor a kisebb indexű madarat fogjuk kiírni!

* 1. Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer ( pl. Windows 10). mingw32-g++.exe c++ fordítóprogram (v4.7), Code::Blocks (v20.03) fejlesztői környezet.

* 1. Forráskód

A teljes fejlesztői anyag –kicsomagolás után– az MQH41V nevű könyvtárban található meg. A fej­lesztés során használt könyvtár-struktúra:

|  |  |
| --- | --- |
| Állomány | Magyarázat |
| MQH41V\bin\Release\MQH41V.exe | futtatható kód |
| MQH41V\obj\Release\main.o | félig lefordított kód |
|  |  |
| MQH41V\main.cpp | C++ forráskód |
|  |  |
| MQH41V\teszt1.txt | teszt-bemeneti fájl1 |
| MQH41V\teszt2.txt | teszt-bemeneti fájl2 |
| MQH41V\teszt3.txt | teszt-bemeneti fájl3 |
| MQH41V\teszt4.txt | teszt-bemeneti fájl4 |
| MQH41V\teszt5.txt | teszt-bemeneti fájl5 |
|  |  |
| MQH41V\Dokumentáció\DokumentációMQH41V.docx | dokumentációk (ez a fájl) |

* 1. Megoldás
     1. Programparaméterek
        1. Konstans

1. maxsor : **Egész**(50) [A mátrix sorainak maximális száma]  
   maxoszlop : **Egész**(200) [A mátrix oszlopainak maximális száma]
   * + 1. Változó

n : **Egész**

m : **Egész**

meresek : **Tömb**(1..N,1..M:**Egész**)

madarak : **(**1..M:**Valós**)

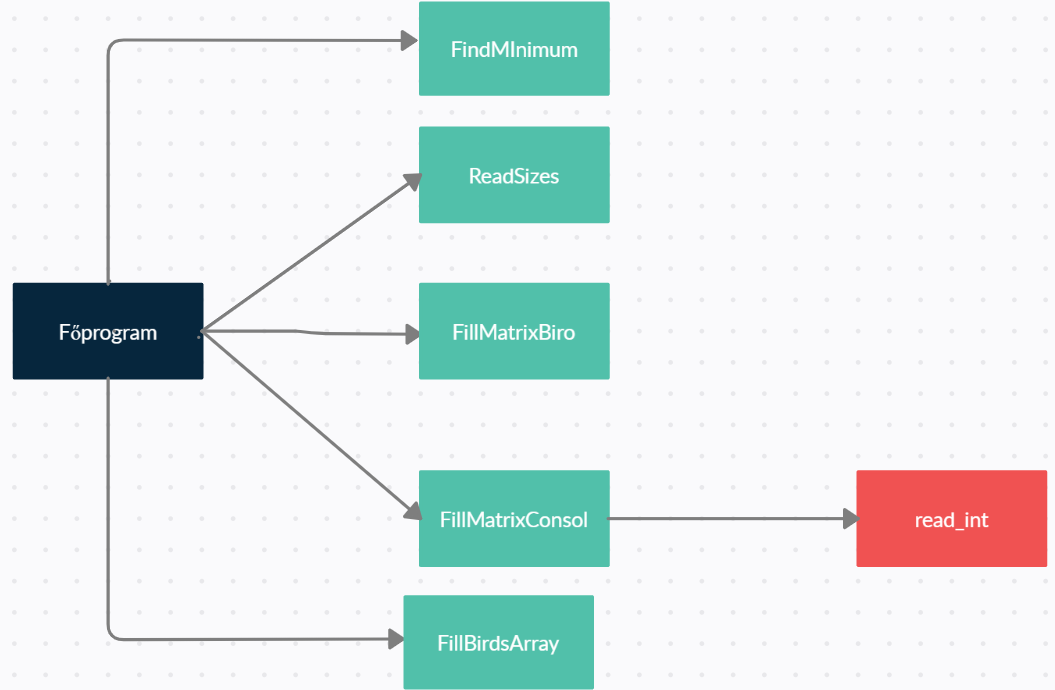
minimum : **Egész**

* + 1. Programfelépítés

A program által használt modulok (és helyük):

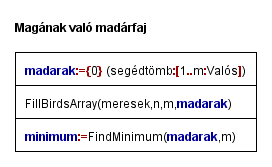
main.cpp – program, a forráskönyvtárban  
iostream – képernyő-, és billentyűkezelés, a C++ rendszer része  
sstream – stringek kezelése, a C++ rendszer része

* + 1. Függvénystruktúra

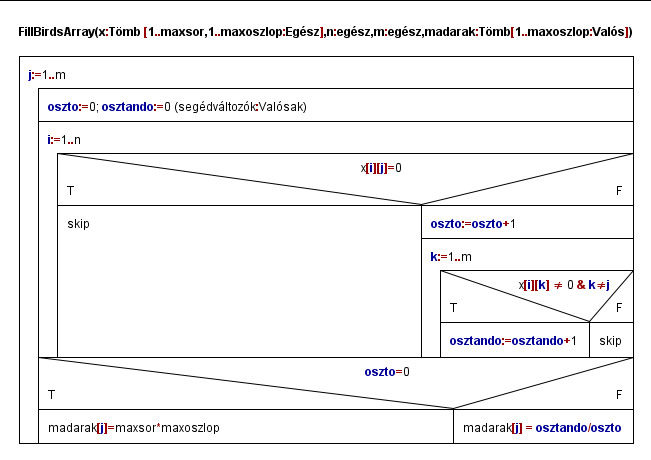
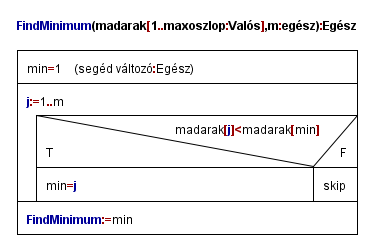


* + 1. A teljes program algoritmusa

Főprogram:



Alprogramok:

****

* + 1. A kód

/\*

Készítette: Tóth Botond

Neptun: MQH41V

E-mail: tothbotond00@gmail.com

Feladat: „Programozás komplex beadandó” téma „Magának való madárfaj” feladat

\*/

#include <iostream>

#include <sstream>

using namespace std;

const int maxsor = 50;

const int maxoszlop = 200;

//#define BIRO

///Egész ellenőrzött beolvasása

//msg - beolvasás előtti üzenet

//errmsg - hibaüzenet

//Szerző: Gregorics Tibor

int read\_int(const string &msg, const string &errmsg);

///Mátrix méreteit beolvasó függvény(Forrás:Matrix feladatsor)

//n - sorok száma (>0 és <maxsor

//m - oszlopok száma (>0 és <maxoszlop

void ReadSizes(int &n, int &m);

///Mátrix feltöltése BÍRÓ-ra feltöltés miatt

//n - sorok száma

//m - oszlopok száma

//x - feltöltendõ mátrix

void FillMatrixBiro (int x [] [maxoszlop],int &n, int &m);

///Mátrix feltöltése konzolról (Forrás:Olimpiai feladatsor)

//n - sorok száma

//m - oszlopok száma

//x - feltöltendõ mátrix

void FillMatrixFromConsol(int x[][maxoszlop],int &n, int &m);

///Egy segédtömb feltöltése az alapján hogy egy madárfaj átlagosan mennyit találkozott más fajokkal

//n - sorok száma

//m - oszlopok száma

//x - adatokkal teli tömb

//madarak - feltöltendő tömb

void FillBirdsArray(int x[][maxoszlop],int &n, int &m,float madarak []);

///A minimum megkeresése az átlagokat tartalmazó tömbben

//m - tömb hossza

//madarak - a tömb amiben a minimumot keressük

int FindMimimum(float madarak[],int &m);

int main()

{

#ifndef BIRO

cout<<"KOMPLEX BEADANDO\n";

cout<<"Ez a program megkeresi meresek alapjan, hogy hanyadik madarfaj az,\n";

cout<<"amelyik legkevesebe szereti mas madarfajok tarsasagat\n";

#endif // BIRO

///N és M bekérése (BÍRÓ/CONSOL)

int n;

int m;

#ifdef BIRO

//Bírós "n" és "m" bekérése

cin>>n;

cin>>m;

#else

//Ellenõrzéses bekérdezés

cout<<"Kerem adja meg a meresek (maximum "<<maxsor<<") es a madarak szamat (maximum "<<maxoszlop<<"):\n";

ReadSizes(n,m);

#endif

///Mátrix feltöltése (BÍRÓ/CONSOL)

int meresek[maxsor][maxoszlop];

#ifdef BIRO

//Bírós mátrix feltöltés

FillMatrixBiro(meresek,n,m);

#else

//Konzolos feltöltés

cout<<"Kerem toltse fel az igy keltkezett matrixot:\n";

FillMatrixFromConsol(meresek,n,m);

#endif // BIRO

///Segédtömbe átlagok kilistázása:

float madarak [m]={0};

FillBirdsArray(meresek,n,m,madarak);

///Minimum átlag megkeresése:

int minimum;

minimum = FindMimimum(madarak,m);

///Kiírás (BÍRÓ/CONSOL)

#ifndef BIRO

cout<<"A madar sorszama: ";

#endif // BIRO

cout<<minimum<<endl;

return 0;

}

int read\_int(const string &msg, const string &errmsg)

{

int n;

bool hiba;

do{

cout << msg;

cin >> n;

hiba=cin.fail();

if(hiba) cin.clear();

string tmp=""; getline(cin, tmp);

hiba = hiba || tmp.size()!=0 ;

if(hiba) cout << errmsg << endl;

}while(hiba);

return n;

}

void FillMatrixBiro (int x [] [maxoszlop],int &n, int &m)

{

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

{

for (int j=0; j<m;++j) cin>>x[i][j];

}

}

void ReadSizes(int &n, int &m)

{

bool hiba;

do{

n=read\_int("Meresek szama:","Pozitiv legyen!");

if(n>maxsor)

{

hiba = true;

cerr<<"Legfeljebb "<<maxsor<<" sora lehet!\n";

}

else if(n<=0)

{

hiba = true;

cerr<<"Nem lehet 0-nal kisebb a sorok szama\n";

}

else hiba =false;

}while(hiba);

do{

m=read\_int("Madarak szama:","Pozitiv legyen!");

if(m>maxoszlop)

{

hiba = true;

cerr<<"Legfeljebb "<<maxoszlop<<" sora lehet!\n";

}

else if(m<=0)

{

hiba = true;

cerr<<"Nem lehet 0-nal kisebb a sorok szama\n";

}

else hiba =false;

}while(hiba);

}

void FillMatrixFromConsol(int x[][maxoszlop],int &n, int &m)

{

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

{

bool hiba;

do{

hiba=false;

cout<<"Add meg az "<<i+1<<". sor elemeit szokozokkel elvalasztva\n";

string sz;

getline(cin,sz);

istringstream inp(sz);

for (int j=0;!hiba && j<m; ++j)

{

inp>>x[i][j];

hiba=inp.fail() || (x [i][j]<0 || x[i][j] >1000);

}

hiba=hiba || !inp.eof();

if(hiba)

{

cerr<<"Hibas vagy tul keves/sok az adat, add meg ujra!\n";

}

}while(hiba);

}

}

void FillBirdsArray(int x[][maxoszlop],int &n, int &m,float madarak [])

{

for (int j=0;j<m;++j)

{

int oszto=0;

int osztando=0;

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

{

if (x[i][j]==0) continue;

else

{

++oszto;

for (int k=0;k<m;++k)

{

if (x[i][k]!=0 &&k!=j) ++osztando;

}

}

}

if (oszto==0)

{

madarak[j]=maxsor\*maxoszlop;

}

else

{

madarak[j]=(float)osztando/(float)oszto;

}

}

}

int FindMimimum(float madarak[],int &m)

{

int min=1;

for (int j=1;j<m;++j)

{

if (madarak[j]<madarak[min-1])

{

min=j+1;

}

}

return min;

}

* 1. Tesztelés
     1. Érvényes tesztesetek
        1. teszteset: teszt1.txt

|  |
| --- |
| Bemenet – *egy madár van csak* |
| n=1  m=1  meresek11=2 |
| Kimenet |
| minimum=1 |

* + - 1. teszteset: teszt2.txt

|  |
| --- |
| Bemenet – *két madár, a második egyszer magában van* |
| n = 2  m = 2  meresek11=1  meresek12=1  meresek21=0  meresek22=1 |
| Kimenet |
| minimum=2 |

* + - 1. teszteset: teszt3.txt

|  |
| --- |
| Bemenet – *nincs sehol egy madár sem (ilyenkor az első kell)* |
| n = 2  m = 2  meresek11=0  meresek12=0  meresek21=0 |
| meresek22=0 |
| Kimenet |
| minimum = 1 |

* + - 1. teszteset: teszt4.txt

|  |
| --- |
| Bemenet – *Egy véletlenszerű madáreloszlás* |
| n = 3  m = 4 |
| meresek11=1  meresek12=2  meresek13=0  meresek14=1  meresek21=0  meresek22=0  meresek23=2  meresek24=0  meresek31=4  meresek32=1  meresek33=0  meresek34=0 |
| Kimenet |
| minimum = 3 |

* + - 1. teszteset: teszt5.txt

|  |
| --- |
| Bemenet – *minden madár el van szigetelve (ilyenkor is az elsőt adja vissza)* |
| n = 3  m = 3 |
| meresek11=1  meresek12=0  meresek13=0  meresek21=0  meresek22=1  meresek23=0  meresek31=0  meresek32=0  meresek33=1 |
| Kimenet |
| minimum = 1 |

* + 1. Érvénytelen tesztesetek
       1. teszteset

|  |
| --- |
| Bemenet – *nem számformátumba írás* |
| n = kettő |
| Kimenet |
| Újrakérdezés: n = |

* + - 1. teszteset

|  |
| --- |
| Bemenet – *negatív madárszám* |
| n = 1  m = 1  meresek11= -10 |
| Kimenet |
| Újrakérdezés:  meresek11= |

* + - 1. teszteset

|  |
| --- |
| Bemenet – *túl sok madárszám* |
| n = 1  m = 1  meresek11= 1 1 |
| Kimenet |
| Újrakérdezés:  meresek11= |

Fejlesztési lehetőségek

1. Többszöri futtatás lehetővé tevése
2. Madarak nevük alapján felismerése (nem sorszámok használata)
3. A meg sem jelent madarak elkülönítése más módon
4. Nagyobb adatok feldolgozása (maxsor és maxoszlop növelése)