Elemi alkalmazások fejlesztése

3. előadás

MAX01

Témakörök

- konstans vektor deklarálása
- vektor hosszának meghatározása
- vektor elemek kiírása szabványos kimeneten

Feladat

Adjunk meg egy vektorban egész számokat, keressük meg a vektor valamely maximális elemét, és az eredményt írjuk ki a szabványos outputra.

4	7	0	9	6	7	9	4

Maximális elemek

Program

4	7	0	9	6	7	9	4

```
A vektor elemei: 4, 7, 0, 9, 6, 7, 9, 4.
A vektor egyik maximális eleme: 9.
Ez a vektor 7. eleme.
```

Megoldás

1. Adatok előkészítése
2. Absztrakt megoldóprogram
3. Eredmény megjelenítése

1 Adatok előkészítése

Itt biztosítjuk, hogy a feladat elvégzéséhez szükséges adatok megfelelő formában rendelkezésünkre álljanak.

2 Számítások elvégzése

A számítások elvégzéséhez felhasználunk valamilyen absztrakt megoldó programot.

Maximumkeresés

3 Eredmény megjelenítése

Írjuk ki a maximális elem értékét és azt, hogy ez a vektor hányadik eleme.

Időzzünk el egy kicsit a második pontnál!

Maximumkeresés

Maximumkeresés tétele

Állapottér | **Előfeltétel** | **Struktogram**

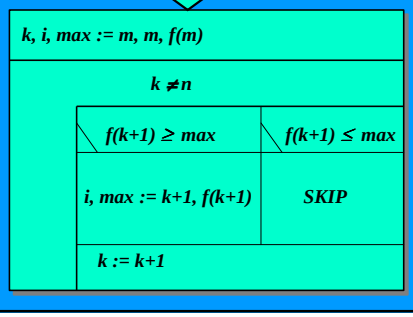
Maximumkeresés: állapottér

f:	függvény
m:	az értelmezési tartomány alsó határa
n:	az értelmezési tartomány felső határa
max:	a maximális elem értéke
i:	a maximális elem indexe
k:	segédváltozó

Maximumkeresés: előfeltétel

$\mathbf{m \leq n}$

Maximumkeresés: struktogram



Feleltessük meg a tételt a mi konkrét feladatunknak.

Mit kell tudnunk a c-beli vektorokról?

Az első elem indexe: 0.

Az utoló elem indexe: n-1.

4	7	0	9	6	7	9	4
v[0]			v[3]				v[n-1]

f: függvény
m: az értelmezési tartomány alsó határa
n: az értelmezési tartomány felső határa
max: a maximális elem értéke
i: a maximális elem indexe
k: segédváltozó

f → **v**
m → **0**
n → **n-1**
max → **max**
i → **i**
k → **k**

v: egész számokat tartalmazó vektor
n: a vektor elemeinek száma
0: a vektor első elemének indexe
n-1: a vektor utolsó elemének indexe
max: a maximális elem értéke
i: a maximális elem indexe
k: segédváltozó

"Megfeleltetés"

Absztrakt feladat

Maximumkeresés

Állapottér | **Előfeltétel** | **Struktogram**

Maximumkeresés: állapottér

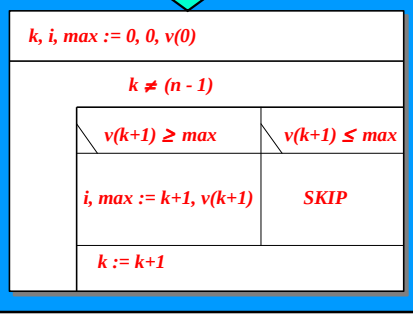
v: egész számokat tartalmazó vektor
n: a vektor elemeinek száma
max: a maximális elem értéke
i: a maximális elem indexe
k: segédváltozó

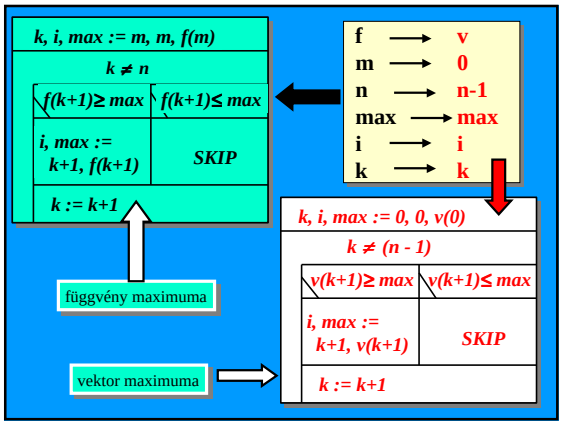
Maximumkeresés: előfeltétel

f →	v
m →	0
n →	n-1
max →	max
i →	i
k →	k

$m \le n$ ⇒ $n \ge 1$

Maximumkeresés vektorban: struktogram





Absztrakt megoldóprogram C++ - ban

Main program felépítése

```
int main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

visszatérési érték

Main program


```
int main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

Mit tudunk eddig a c-beli vektorokról?

Az első elem indexe: 0.

Az utoló elem indexe: n-1.

4	7	0	9	6	7	9	4

v[0]

v[n-1]

Mit kell tudnunk még a c-beli vektorokról?

?	4	7	0	9	6	7	9	4	?

v[-1]

v[n]

Ha a vektor **-1**-dik vagy az **n** - dik elemére hivatkozunk - **SAJNOS!!!!** - nincs hibajelzés, de természetesen ilyenkor akármi is történhet.

TILOS ILYET CSINÁLNI!!!!!

1 Adatok előkészítése és megjelenítése - I.

```
//Adatok előkészítése és megjelenítése
const int v [] = {4,7,0,9,6,7,9,4};
const int n = sizeof(v) / sizeof(v[0]);
```

A v vektor meghatározása

A v vektor meghatározása

A vektor elemeinek értékét a program futása közben nem változtatjuk meg.

A vektor egész számokat tartalmaz

```
const int v [ ] = { 4, 7, 0, 9, 6, 7, 9, 4 } ;

const int n = sizeof(v) / sizeof(v[0]);
```

A vektor elemszámát a program futása közben nem változtatjuk meg.

n a vektor elemeinek száma.

A v vektor elemszámának meghatározása - I.

4	7	0	9	6	7	9	4

```
const int v [] = {4, 7, 0, 9 6, 7, 9, 4};

const int n = sizeof(v) / sizeof(v[0]);
```

sizeof(v)

Megadja, hogy a v vektor hány byte-ból áll.

A v vektor elemszámának meghatározása - II.

4	7	0	9	6	7	9	4

```
const int v [] = {4,7,0,9,6,7,9,4};

const int n = sizeof(v) / sizeof(v[0]);
```

Megadja, hogy a v vektor nulladik indexű eleme hány byte-ból áll.

sizeof(v[0])

A v vektor elemszámának meghatározása - III.

4	7	0	9	6	7	9	4

```
const int v [] = {4,7,0,9,6,7,9,4};

const int n = sizeof(v)  /  sizeof(v[0]);
```

sizeof(v)

sizeof(v[0])

A két érték hányadosa megadja a vektorban szereplő elemek számát.

1 Adatok előkészítése és megjelenítése - II.

```
//Adatok előkészítése és megjelenítése
  const int v [] = {4,7,0,9,6,7,9,4};
  const int n = sizeof(v) / sizeof(v[0]);
  cout << "A vektor elemei: ";
  for (int j=0; j!=n; j++){
    cout << v[j];
    if (j != (n-1))
      cout << ", ";
    else
      cout << endl;
  }
```

A **v** vektor megjelenítése

Először kiírjuk „**A vektor elemei:** ” szöveget, soremelés nélkül.

A **v** vektor megjelenítése 1

1

```
cout << "A vektor elemei: ";
```

A vektor elemei:

Egy **for** ciklusban kiírjuk a vektor elemeit.

```
cout << "A vektor elemei: ";
for (int j=0; j!=n; j++){

}
```

for ciklus

j

cilusmag

4	7	0	9	6	7	9	4

```
cout << "A vektor elemei: ";
for (int j=0; j!=n; j++){

}
```

ciklus eleje

ciklus vége

A **for** ciklusban deklarált változó csak a cikluson belül létezik.

for (int j=0; j!=n; j++){

A **for** ciklusban szereplő elágazás **igaz** ágán **az első n-1 elemet** úgy írjuk ki, hogy mindegyik mögé írunk egy **vesszőt.**

A **v** vektor megjelenítése 2

2

```
cout << "A vektor elemei: ";
for (int j=0; j!=n; j++){
    cout << v[j];
    if (j != (n-1))
      cout << ", ";
    else
      cout << "." << endl;
  }
```

A vektor elemei: 4,7,0,9,6,7,9,

Az utolsó elemet a **for** ciklusban található elágazás **hamis** ágán egy **ponttal** és egy **soremeléssel (endl)** írjuk ki.

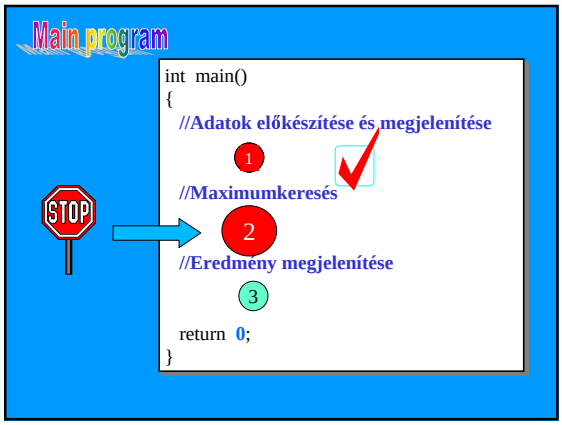
A **v** vektor megjelenítése

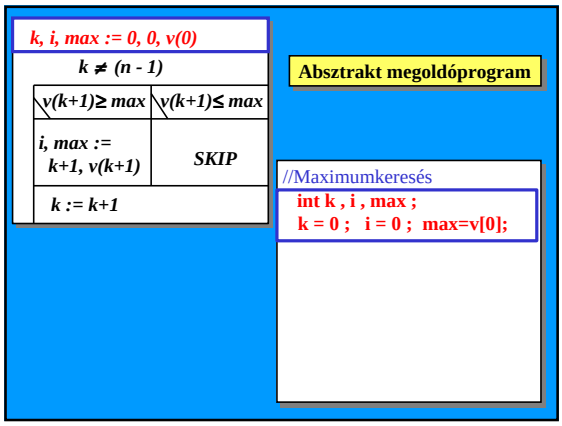
3

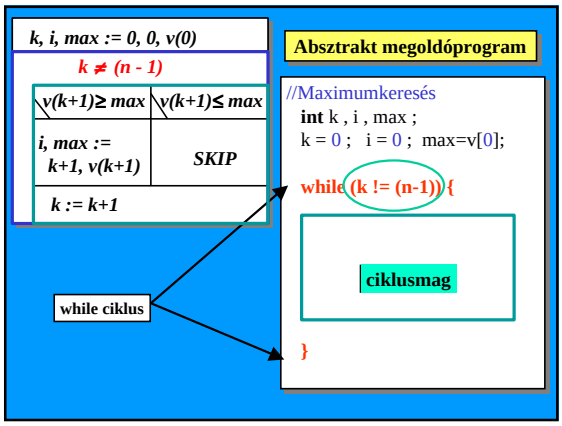
```
cout << "A vektor elemei: ";
  for (int j=0; j!=n; j++){
    cout << v[j];
    if (j != (n-1))
      cout << ", ";
    else
      cout << "." << endl;
  }
```

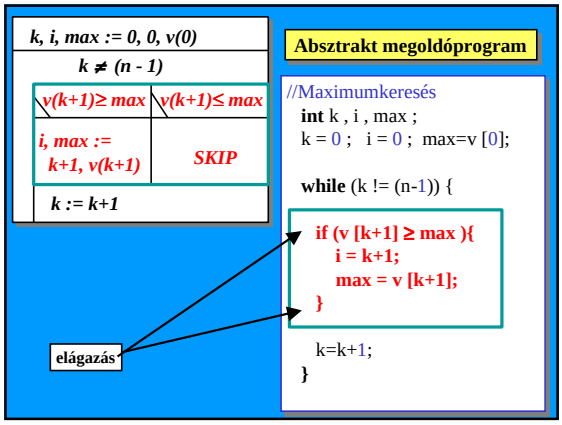
3

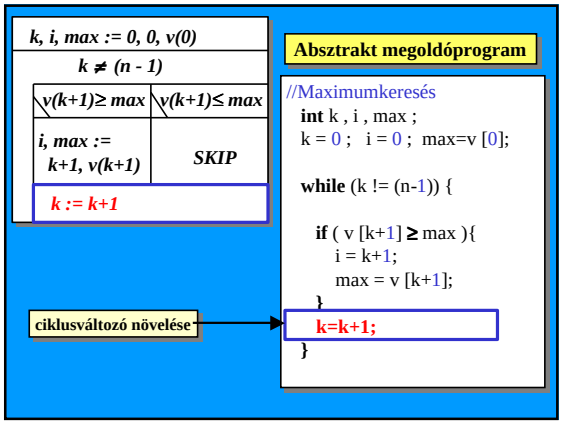
A vektor elemei: 4,7,0,9,6,7,9,4.











k, i, max := 0, 0, v(0)	
k ≠ (n - 1)	
v(k+1)≥ max	*v(k+1)≤ max*
i, max := k+1, v(k+1)	*SKIP*
k := k+1	

Absztrakt megoldóprogram

```
//Maximumkeresés
  int k , i , max ;
  k = 0 ;   i = 0 ;  max=v[0];
  while (k != (n-1)) {
    if ( f [k+1] ≥ max ){
       i = k+1;
       max = v [k+1];
    }
    k=k+1;
  }
```

Main program

```
int  main()
{
  //Adatok előkészítése és megjelenítése
      1
  //Maximumkeresés
      2
  //Eredmény megjelenítése
      3
  return  0;
}
```

STOP

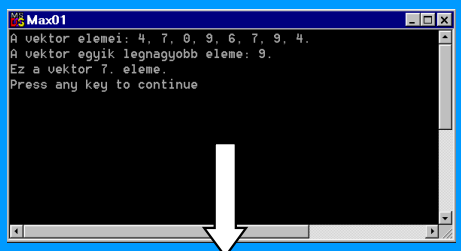
```
//Eredmény megjelenítése

cout << "A vektor egyik maximális eleme: "<< v[i] << "."<<endl ;
cout << "Ez a vektor " <<  (i+1)  << ". eleme." << endl;
```

```
A vektor egyik maximális eleme: 9.
Ez a vektor 7. eleme.
```

Main program

```
int  main()
{
  //Adatok előkészítése és megjelenítése
      1
  //Maximumkeresés
      2
  //Eredmény megjelenítése
      3
  return  0;
}
```

Hogyan biztosíthatjuk, hogy a futás eredménye „ne szaladjon el” a képernyőről?



Consol várakoztatása

Consol várakoztatása

```
#include <iostream>
using namespace std;
int  main()
{ . . .
```

```
//Consol várakoztatása
  char barmi ;
  cin >> barmi ;
```

A futtatás eredménye

Max01

```
A vektor elemei: 4, 7, 0, 9, 6, 7, 9, 4.
A vektor egyik legnagyobb eleme: 9.
Ez a vektor 7. eleme.
Press any key to continue.
```

MAX02a

Témakörök

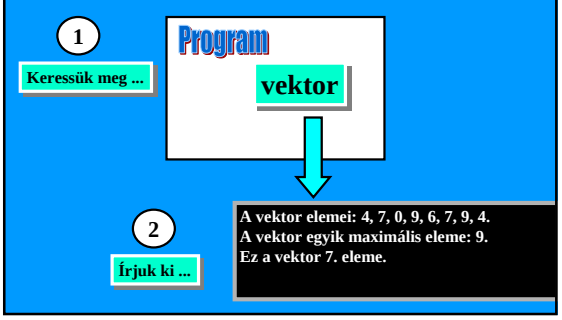
- fájlkezelés (*minimális hibaelhárítással*)
- „nyíl-nyíl” (<<, >>) operátorok alkalmazása adatok beolvasására és kiírására
- parancssorban megadott argumentumok bekérése

Feladat: MAX02a

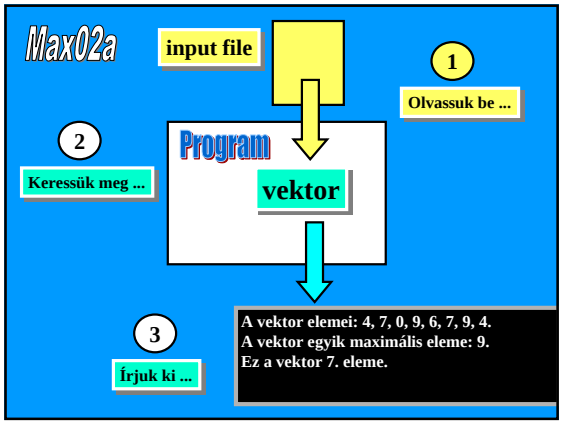
Olvassunk be egy fájlból egész számokat egy vektorba, majd keressük meg a vektor valamely maximális elemét és az eredményt írjuk ki a szabványos outputra.

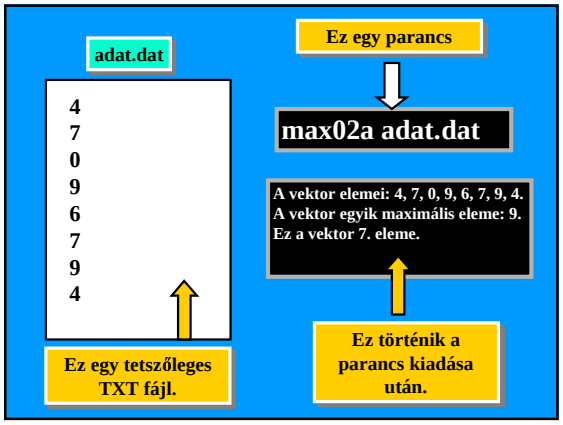
A **Max02a** feladat abban különbözik a **Max01** feladattól, hogy a vektort egy fájlból kell feltölteni..

Max01



Max02a





Kérdések

- Hol adjuk meg a fájl nevét?
- Milyen a fájl hossza?
- Milyen a vektor hossza?
- Ez a két hossz hogy viszonyul egymáshoz?
- Létezik-e a fájl?
- Mit tegyünk, ha nem létezik?
- Lehet-e a fájlban lévő elemek száma nagyobb, mint a vektor hossza?
- ...

Feladat: MAX02a

Keressük meg egy vektor valamely maximális elemét és az eredményt írjuk ki a szabványos outputra.

Megkötések:
A vektor feltöltése **fájlból** történik.
A fájl nevét **parancssorban** adjuk meg.
A fájl **pontosan 8** elemet tartalmaz.
A vektor hossza **pontosan 8**.

Megoldás

1. Adatok előkészítése
2. Absztrakt megoldóprogram
3. Eredmény megjelenítése

Adatok előkészítése

Itt biztosítjuk, hogy a feladat elvégzéséhez szükséges adatok megfelelő formában rendelkezésünkre álljanak.

Ez más, mint a Max01-ben.

Számítások elvégzése

A számítások elvégzéséhez felhasználunk valamilyen absztrakt megoldó programot.

Maximumkeresés

Ez ugyanaz, mint a Max01-ben.

Eredmények megjelenítése

Írjuk ki a maximális elem értékét, valamint azt, hogy ez a vektor hányadik eleme,

Ez ugyanaz, mint a Max01-ben.

Absztrakt megoldóprogram C++ - ban


```
int main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

Ez más lesz, mint a Max01-ben.

```
int main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

Ez más, mint a Max01-ben.

```
int main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

Ez ugyanaz, mint a Max01-ben.

Hogyan módosul a main eljárás deklarációja, ha a fájl nevét parancssorban adjuk meg?

A fájl nevét **<u>parancssorban</u>** adjuk meg.

parancssor

```
max02a     adat.dat
```

A fájl nevét **parancssorban** adjuk meg.

fájl neve

```
max02a    adat.dat
```

A main függvény deklarációja

int **main**(int argc, char *argv[ ])

```
int  main( int argc, char *argv[])
{
  //Adatok előkészítése és megjelenítése

  //Maximumkeresés

  //Eredmény megjelenítése

  return  0;
}
```

Milyen egy parancssor?

Hogyan hivatkozhatunk a parancssorban megadott argumentumokra?

int **main**(int argc, char *argv[ ])

parancssorban megadott argumentumok száma

2 db

```
max02a      adat.dat
```

int **main**(int argc, char *argv[ ])

argumentumok listája

arg[0] arg[1]

```
max02a    adat.dat
```

```
int  main( int argc, char *argv[])
{
  //Adatok előkészítése és megjelenítése
      … argv[1] …
  //Maximumkeresés

  //Eredmény megjelenítése

  return  0;
}
```

hivatkozás a fájl nevére

```
max02a    adat.dat
```

Hogyan kezelhetjük a fájlokat?

Fájlkezelés

Fájlok kezelése

bemeneti osztály

kimeneti osztály

Az **ifstream** bemeneti osztályt és az **ofstream** kimeneti osztályt az **fstream** deklarációs fájl tartalmazza.

Adatok beolvasása fájlból

#include <fstream>

Fájl **deklarációja:**	**ifstream inp;**
Fájl **megnyitása:**	**inp.open(fájl neve);**
Fájl **bezárása:**	**inp.close();**
Hibakezelés:	**inp.fail();** **(logikai érték)**
Következő adat **beolvasása**:	**inp >> ahova_beolvas;**

Adatok kiírása fájlba

#include <fstream>

Fájl **deklarációja:**	**ofstream out;**
Fájl **megnyitása:**	**out.open(fájl neve);**
Fájl **bezárása:**	**out.close();**
Hibakezelés:	**out.fail();** **(logikai érték)**
Következő adat **kiírása:**	**out << amit_kiír;**

Return

Kezeljünk le néhány hibaesetet.

1 **A visszatérési érték legyen 1, ha a fájl nevét nem adták meg a parancssorban.**

2 **A visszatérési érték legyen 2, ha a megadott fájlt nem sikerült megnyitni.**

```
#include <fstream>
using namespace std;
...

//Adatok előkészítése és megjelenítése
```

Megadjuk az **fstream** deklarációs fájlt.

```
//Adatok előkészítése és megjelenítése
if(argc <=1){
    cout << "Hiányzó paraméter!" << endl;
    return 1;
 }
 int v[8];
 int n=8;
 ifstream inp;
 inp.open(argv[1]);
 if(inp.fail()){
   cerr << "A megadott fájlt nem találom!" <<
   return 2;
 }
 for (int j=0; j!=n; j++){
   inp >> v[j];
 }
 inp.close();
```

A visszatérési érték legyen 1, ha nincs fájlnév.

```
//Adatok előkészítése és megjelenítése
 if(argc <=1){
    cout << "Hiányzó paraméter!" << endl";
    return 1;
 }
 int v[8];
 int n=8;
 ifstream inp;
 inp.open(argv[1]);
 if(inp.fail()){
   cerr << "A megadott fájlt nem találom!" << endl;
   return 2;
 }
 for (int j=0; j!=n; j++){
   inp >> v[j];
 }
 inp.close();
```

Deklarálunk egy 8 elemű vektort.

```
//Adatok előkészítése és megjelenítése
if(argc <=1){
   cout << "Hiányzó parameter!" << endl;
   return 1;
 }
 int v[8];
 int n=8;
 ifstream inp;
 inp.open(argv[1]);
 if(inp.fail()){
   cerr << "A megadott fájlt nem találom!" << endl;
   return 2;
 }
 for (int j=0; j!=n; j++){
   inp >> v[j];
 }
 inp.close();
```

Definiáljuk az **ifstream** típusú **inp** változót (objektumot).

```
//Adatok előkészítése és megjelenítése
if(argc <=1){
   cout << "Hiányzó paraméter!" << endl;
   return 1;
 }
 int v[8];
 int n=8;
 ifstream inp;
 inp.open(argv[1]);
 if(inp.fail()){
   cerr << "A megadott fájlt nem találom!" << endl;
   return 2;
 }
 for (int j=0; j!=n; j++){
   inp >> v[j];
 }
 inp.close();
```

Alkalmazzuk az inp változóra az open műveletet. A művelet egyetlen paramétere: argv[1].

```
//Adatok előkészítése és megjelenítése
if(argc <=1){
   cout << "Hiányzó paraméter!" << e
   return 1;
}
int v[8];
int n=8;
ifstream inp;
inp.open(argv[1]);
if(inp.fail()){
   cerr << "A megadott fájlt nem találom!" << endl;
   return 2;
}
for (int j=0; j!=n; j++){
   inp >> v[j];
}
inp.close();
```

Magyarul: Megnyitjuk azt a fájlt, melynek nevét a parancssorban adtuk meg.

```
//Adatok előkészítése és megjelenítése
if(argc <=1){
   cout << "Hiányzó paraméter!" << endl
   return 1;
}
int v[8];
int n=8;
ifstream inp;
inp.open(argv[1]);
if(inp.fail()){
   cerr << "A megadott fájlt nem találom!" << endl;
   return 2;
}
for (int j=0; j!=n; j++){
   inp >> v[j];
}
inp.close();
```

A visszatérési érték legyen 2 sikertelen fájlnyitásnál.

```
//Adatok előkészítése és megjelenítése
if(argc <=1){
   cout << "Hiányzó paraméter!" << endl;
   return 1;
}
int v[8];
int n=8;
ifstream inp;
inp.open(argv[1]);
if(inp.fail()){
   cerr << "A megadott fájlt nem találom!" << endl;
   return 2;
}
for (int j = 0; j != n; j++){
   inp >> v[j];
}
inp.close();
```

Egy „belső változós” for ciklussal beolvasunk **pontosan(!)** 10 adatot.

```
//Adatok előkészítése és megjelenítése
if(argc <=1){
   cout << "Hiányzó paraméter!" << endl;
   return 1;
}
int v[8];
int n=8;
ifstream inp;
inp.open(argv[1]);
if(inp.fail()){
   cerr << "A megadott fájlt nem talá
   return 2;
}
for (int j=0; j!=n; j++){
   inp >> v[j];
}
inp.close();
```

Az adatok beolvasása után bezárjuk a fájlt.

Main

```
int main( int argc, char *argv[])
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

A futtatás eredménye

```
max02a adat.dat
A vektor elemei: 4, 7, 0, 9, 6, 7, 9, 4.
A vektor egyik legnagyobb eleme: 9.
Ez a vektor 7. eleme.
Press any key to continue.
```

MAX02B

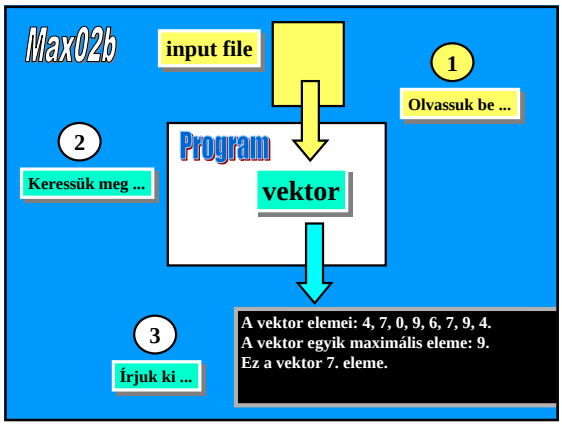
Témakörök

- string
- Adat (string) bekérése szabványos bemenetről

Feladat: MAX02B

Olvassunk be egy fájlból egész számokat egy vektorba, majd keressük meg a vektor valamely maximális elemét, és az eredményt írjuk ki a szabványos outputra.

A **Max02b** feladat abban különbözik a Max01feladattól, hogy a vektort egy fájlból kell feltölteni..



Feladat: MAX02b

Keressük meg egy vektor valamely maximális elemét és az eredményt írjuk ki a szabványos outputra.

Megkötések:

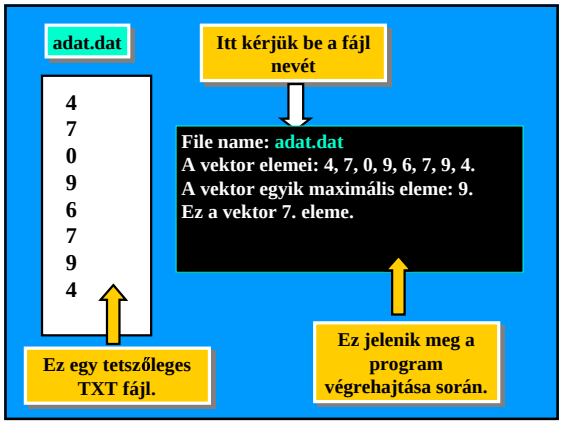
A vektor feltöltése **fájlból** történik.

A fájl nevét **a szabványos bemenetről** kérjük be

A fájl **pontosan 8** elemet tartalmaz.

A vektor hossza **pontosan 8**.

Ez más, mint a Max02a-ban.



A fájl nevét egy string típusú változóban szeretnénk elhelyezni,

#include <string>

Ha string típusú változókat szeretnénk használni, akkor meg kell adnunk azt a fájlt, amely deklarálja a különböző string kezelő rutinokat.

A fájl nevét egy string típusú változóban szeretnénk elhelyezni,

```
#include <string>

string      InpFileName;
```

Az InpFileName legyen egy string típusú változó.

A fájl nevét egy string típusú változóban szeretnénk elhelyezni,

```
#include <string>

string      InpFileName;

inp.open(InpFileName.c_str());
```

A c_str() művelettel az InpFileName változót az open számára elfogadható típusra alakítjuk át.

Igy nyitjuk meg a fájlt.

Absztrakt megoldóprogram C++ - ban

"Kódolás"

```
int  main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return  0;
}
```

Ez a rész változik.

```
//Adatok előkészítése
…

ifstream inp;
string InpFileName;
cout << "Fájl neve:  ";
cin >> InpFileName;
inp.open(InpFileName.c_str());

...
```

Definiáljuk az **ifstream** típusú **inp** változót (objektumot).

```
//Adatok előkészítése
…

ifstream inp;
string InpFileName;
cout << "Fájl neve:  ";
cin >> InpFileName;
inp.open(InpFileName.c_str());

...
```

Legyen az InpFileName egy string típusú változó

```
//Adatok előkészítése
…

ifstream inp;
string InpFileName;
cout << "Fájl neve:  ";
cin >> InpFileName;
inp.open(InpFileName.c_str());

...
```

Írjuk ki a szabványos outputra a „Fájl neve: „ üzenetet.

Fájl neve:

```
//Adatok előkészítése
…

ifstream inp;
string InpFileName;
cout << "Fájl neve:  ";
cin >> InpFileName;
inp.open(InpFileName.c_str());

...
```

A felhasználó megadja a fájl nevét.

Fájl neve: adat.dat

```
//Adatok előkészítése
…

ifstream inp;
string InpFileName;
cout << "Fájl neve:  ";
cin >> InpFileName;
inp.open(InpFileName.c_str());

...
```

Nyissuk meg az adott nevű fájlt.

A c_str() művelettel az InpFileName változót az open számára elfogadható típusra alakítjuk át.

```
#include <string>
```

```
//Adatok előkészítése
…

ifstream inp;
string InpFileName;
cout << "Fájl neve:  ";
cin >> InpFileName;
inp.open(InpFileName.c_str());

...
```

Main

```
int  main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return  0;
}
```

A futtatás eredménye

Max01

```
Fájl neve: adat.dat
A vektor elemei: 4, 7, 0, 9, 6, 7, 9, 4.
A vektor egyik legnagyobb eleme: 9.
Ez a vektor 7. eleme.
Press any key to continue.
```

MAX02C

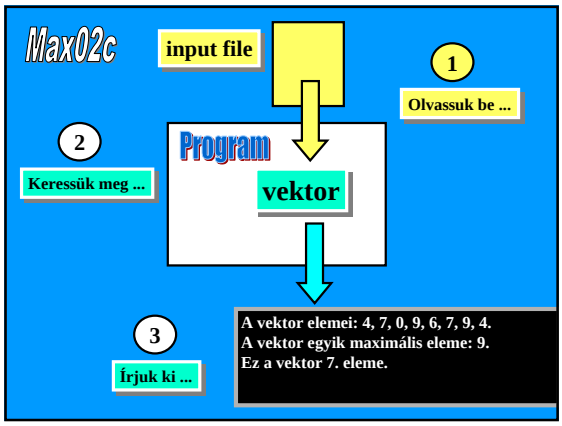
Témakörök

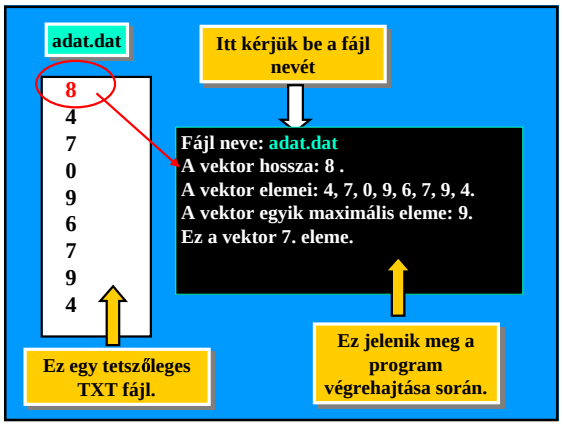
- Dinamikus helyfoglalás(**new** operátor)
- Dinamikus memória felszabadítás (**delete** operátor)
- Tömbök létrehozása **new** operátorral

Feladat: MAX02C

- A fájl első elemeként olvassuk be a létrehozandó vektor hosszát.
- Foglaljuk le dinamikusan a szükséges méretű vektort.
- Olvassuk be a fájlból az adott számú egész számokat a dinamikusan lefoglalt vektorba.
- Keressük meg a vektor valamely maximális elemét és az eredményt írjuk ki a szabványos outputra.

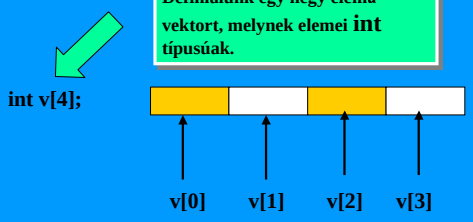
A Max02c feladat abban különbözik a Max02b feladattól, hogy a vektort hosszát is a fájlban adjuk meg, és a vektort ennek függvényében kell megadni.





Statikus helyfoglalás

Definiálunk egy négy elemű vektort, melynek elemei **int** típusúak.



A definiálás pillanatában lefoglalunk egy tárterületet, mely a változó teljes élettartamára foglalt marad.

Dinamikus helyfoglalás

new

A **new** operátorral igény szerint lefoglalhatunk, majd felszabadíthatunk a típusnak megfelelő méretű területet a szabad memóriában.

Dinamikus helyfoglalás

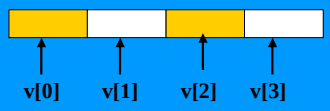
int *v;

Ezen a ponton még csak jelezzük, hogy majd lesz egy adatunk, de helyet még nem foglalunk neki.

Dinamikus helyfoglalás

int *v;

v =new int[4];

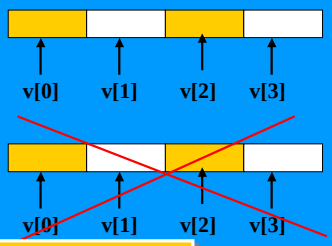


Itt foglalunk helyet.

Dinamikus helyfoglalás

int *v;

v =new int[4];



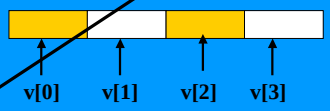
delete[ ] v;

Amikor nincs rá tovább szükségünk, felszabadítjuk a lefoglalt területet.

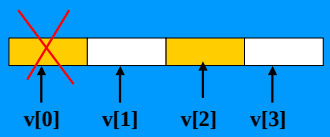
Dinamikus helyfoglalás HIBA!!!

int *v;

v =new int[4];



delete v ;



!

Dinamikus helyfoglalás egy változó esetén. 1

int *x;	Először deklaráljuk az x változót, de nem foglalunk neki helyet.
x=new int;	A new hívásával int típusú adatból 1 darabot lefoglalunk.
*x=10;	Igy adunk értéket az x változónak.
… x ...;	Igy hivatkozunk az x változóra.
delete x;	A lefoglalt területet felszabadítása.

Dinamikus helyfoglalás tömbök esetén 2

int *v;	Deklarálunk egy int típusra mutató v változót.
v=new int[4];	A new hívásával int típusú adatból 4 darabot lefoglalunk.
… v[i] … ;	Így adunk értéket a vektor i-dik elemének
delete[ ] v;	A vektorként lefoglalt területet így szabadíthatjuk fel.

Dinamikus helyfoglalás kezdeti értékadással. 3

int *a;	Deklarálunk egy int típusra mutató változót.
a=new int(4);	A new hívásával lefoglalunk egy int típusú változót, melynek értéke 4.
… a …	Így hivatkozunk az elemre.
delete a;	A lefoglalt területet felszabadítása.

1	2	3
int *x;	int *v;	int *a;
x=new int;	v=new int[4];	a=new int(4);
*x=10;	… v[i] … ;	… a …
delete x;	delete[ ] v;	delete a;

Dinamikus helyfoglalás - összefoglalás

1. new tipus;
2. new tipus[méret];
3. new tipus(méret);

A feladat megoldása a fenti eszközökkel.

```
//Adatok előkészítése és megjelenítése
ifstream inp;
string InpFileName;
cout << "File name:";
cin >> InpFileName;
inp.open(InpFileName.c_str());
if(inp.fail()){
  cerr << "A megadott fájlt nem találom!" << endl;
  return 1;
}
```

Folytatás

Ez a rész ugyanaz, mint a max02b-ben.

```
//Adatok előkészítése és megjelenítése
...
int n;
inp >> n;
cout << "A vektor hossza: ";
cout << n;
cout << "." << endl";
//Előfeltétel ellenőrzése
if(n<1){
  cout << "A vektor üres!" << endl;
  return 2;
}
int *v;
v=new int [n];
```

Folytatás

Deklarálunk egy int típusú változót, majd a fájlból beolvassuk az első adatot. Ez a vektor hossza.

```
//Adatok előkészítése és megjelenítése
...
int n;
inp >> n;
cout << "A vektor hossza: ";
cout << n;
cout << "." << endl;
//Előfeltétel ellenőrzése
if(n<1){
  cout << "A vektor üres!" << endl;
  return 2;
}
int *v;
v=new int [n];
```

Folytatás

Kiírjuk a vektor hosszát a szabványos kimenetre.

```
//Adatok előkészítése és megjelenítése
...
int n;
inp >> n;
cout << "A vektor hossza: "
cout << n;
cout << "." << endl;
//Előfeltétel ellenőrzése
if(n<1){
   cout << "A vektor üres!" << endl;
   return 2;
 }
int *v;
v=new int [n];
```

Folytatás

Ha nem teljesül az m≥1 előfeltétel, akkor befejezzük a programot.
A visszatérési érték: 2.

```
//Adatok előkészítése és megjelenítése
...
int n;
inp >> n;

//Előfeltétel ellenőrzése
if(n<1){
  cout << "A vektor üres!" << endl;
  return 2;
}
cout << "A vektor hossza: ";
cout << n;
cout << "." << endl;
int *v;
v=new int [n];
```

Folytatás

Deklarálunk egy int típusra mutató v változót.

```
//Adatok előkészítése és megjelenítése
...
int n;
inp >> n;
//Előfeltétel ellenőrzése
if(n<1){
  cout << "A vektor üres!" << endl;
  return 2;
}
cout << "A vektor hossza: ";
cout << n;
cout << "." << endl;
int *v;
v=new int [n];
```

Folytatás

A new hívásával int típusból n darabot lefoglalunk.

```
//Adatok előkészítése és megjelenítése
...

for (int j=0; j!=n; j++){
  inp >> v[j];
}
inp.close();
cout << "A vektor elemei: ";
for (int j=0; j!=n; j++){
  cout << v[j];

  if (j < n-1)
    cout << ", ";
  else
    cout << "." << endl;
}
```

Folytatás

Így hivatkozunk a vektor elemeire.

Main

```
int main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

Ez ugyanaz, mint a **Max02b**-ben.

Main

```
int main()
{
  //Adatok előkészítése és megjelenítése
        1
  //Maximumkeresés
        2
  //Eredmény megjelenítése
        3
  return 0;
}
```

Ez majdnem ugyanaz, mint a **Max02b**-ben.

Visszatérés előtt a dinamikus tárterületet fel kell szabadítani.

```
//Eredmény megjelenítése
cout << "A vektor egyik legnagyobb eleme: " << v[i] << ".";
cout << endl << "Ez a vektor " << (i+1) << ". eleme." << endl;

//Dinamikusan lefoglalt tárterület felszabadítása
delete[ ] v;
return 0;
}
```

Visszatérés előtt a dinamikus tárterületet fel kell szabadítani.

Összefoglalás

```
//Adatok előkészítése és megjelenítése
                                        1
ifstream inp;
string InpFileName;
cout << "File name:";
cin >> InpFileName;
inp.open(InpFileName.c_str());
if(inp.fail()){
  cerr << "A megadott fájlt nem találom!" << endl;
  return 1;
}
```

```
//Adatok előkészítése és megjelenítése
...   Folytatás                         1
int n;
inp >> n;
cout << "A vektor hossza: ";
cout << n;
cout << "." << endl;
//Előfeltétel ellenőrzése
if(n<1){
  cout << "A vektor üres!" << endl;
  return 2;
}
int *v;
v=new int [n];
```

2

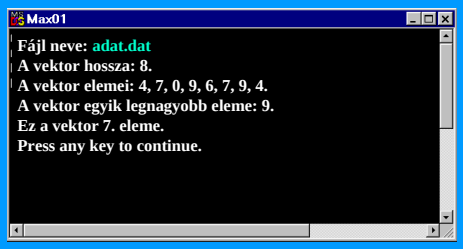
```
//Maximumkeresés
  int k , i , max ;
  k = 0 ;   i = 0 ;  max = v [0];
  while (k != (n-1)) {
    if ( v [k+1] ≥ max ){
      i = k+1;
      max = v [k+1];
    }
    k=k+1;
  }
```

Ez ugyanaz, mint a Max01-ben.

```
//Eredmény megjelenítése

cout << "A vektor egyik maximális eleme: " << v[i] << "." ;
cout << endl << "Ez a vektor " << (i+1) << ". eleme." << endl;

delete [ ] v;

return 0;
```

3

A futtatás eredménye



Összefoglalás

- Vektorok deklarálása
- Vektor hosszának meghatározása
- Statikus és dinamikus helyfoglalás/felszabadítás
- Parancssorban megadott argumentumok kezelése
- Fájlkezelés minimális hibaelhárítással
- String használata
- String konvertálása char* típusra

Feladatok

Levezetéses feladatok:
14, 16, 17, 18, 19, 20, 21, 22, 23, 26.

Visszavezetéses feladatok:
17, 21, 28, 30, 31,

Vége az előadásnak