Elemi alkalmazások fejlesztése

4. előadás

Készítette: Szabóné Nacsa Rozália
nacsa@inf.elte.hu
2002.03.04.

MAX03

- Függvények

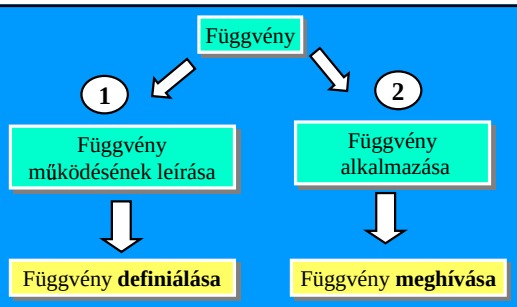
Feladat

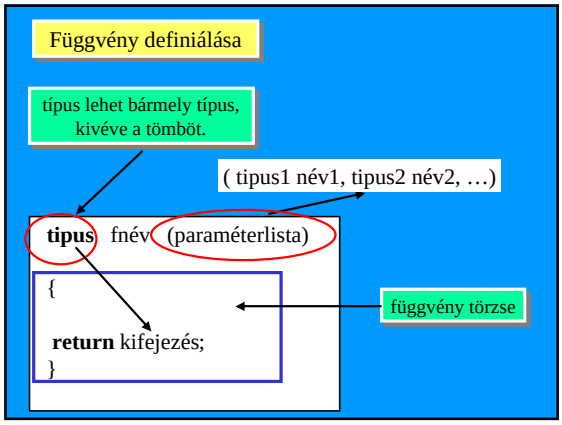
Írjunk egy olyan „univerzális programot” - **eljárást, függvényt** - amely alkalmas egy általunk megadott, tetszőleges vektor maximális elemének a meghatározására.

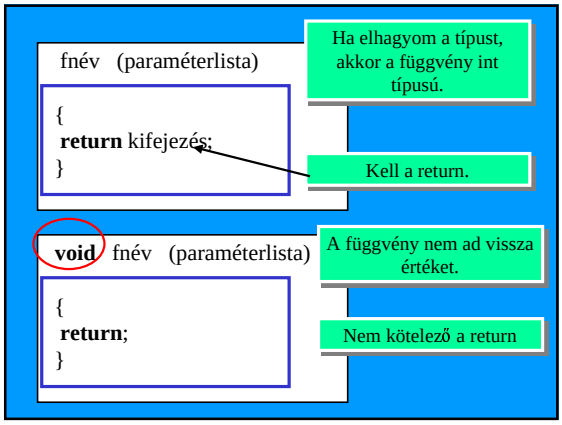
Megadjuk a vektort.

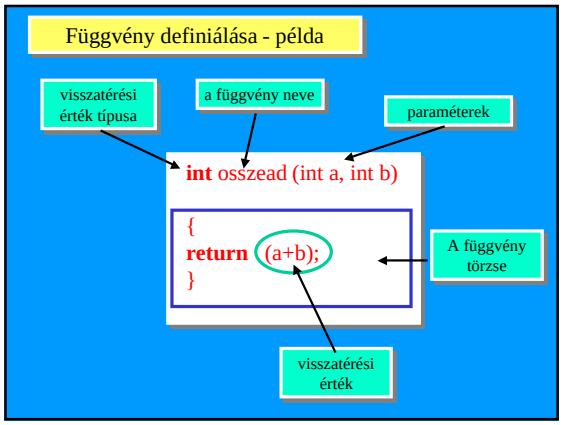
maxker(v, n, i, max)

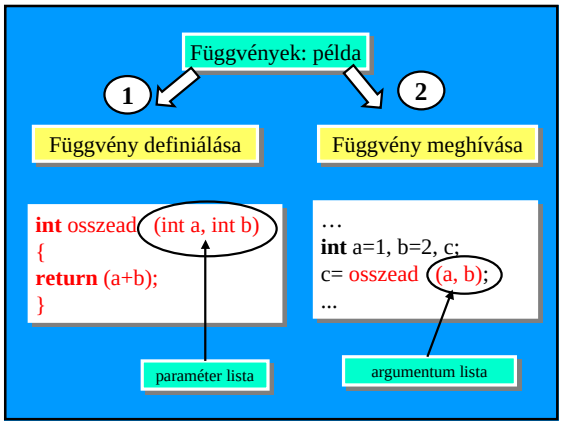
Megkapjuk a maximális elemet

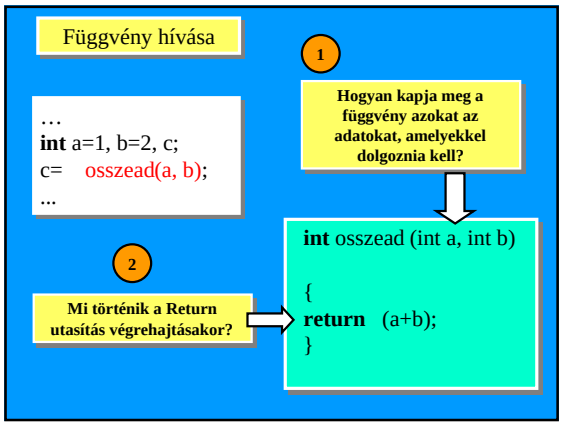


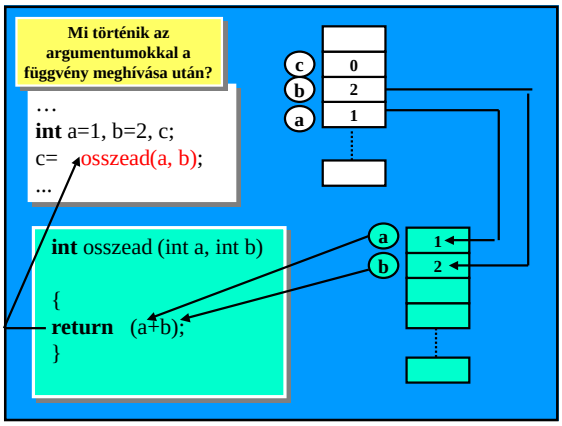


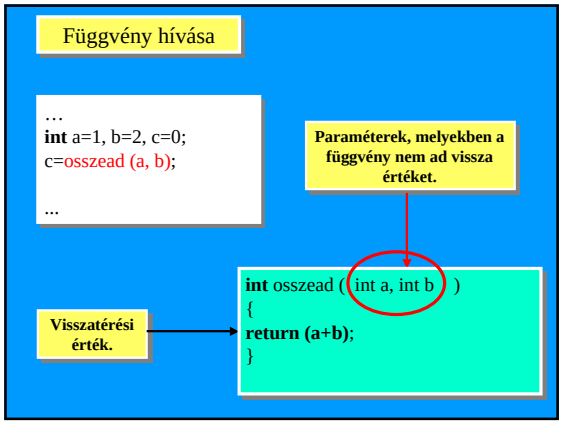


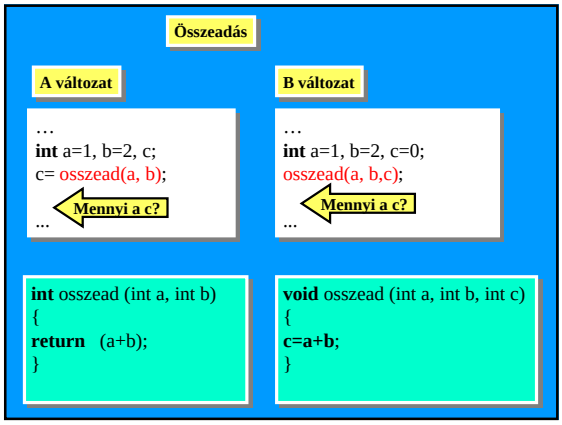

Függvény definiálása - példa
visszatérési érték típusa
a függvény neve
paraméterek
int osszead (int a, int b)
{
return (a+b);
}
A függvény törzse
visszatérési érték

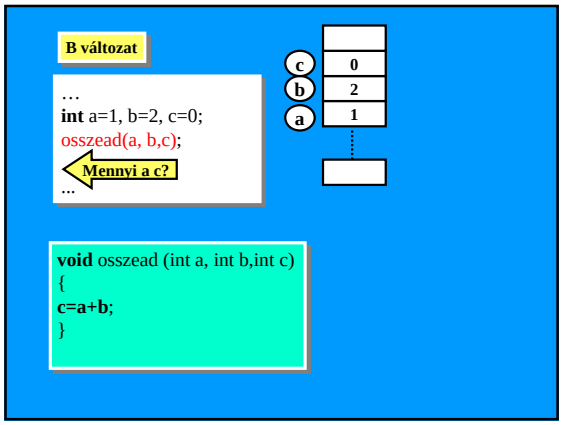

Függvények: példa
1
2
Függvény definiálása
Függvény meghívása
int osszead (int a, int b)
{
return (a+b);
}
paraméter lista
…
int a=1, b=2, c;
c= osszead (a, b);
…
argumentum lista

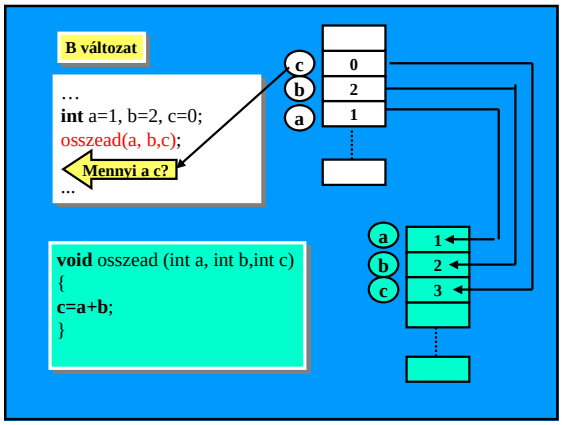

Függvény hívása
1
Hogyan kapja meg a függvény azokat az adatokat, amelyekkel dolgoznia kell?
…
int a=1, b=2, c;
c= osszead(a, b);
…
2
Mi történik a Return utasítás végrehajtásakor?
int osszead (int a, int b)
{
return (a+b);
}

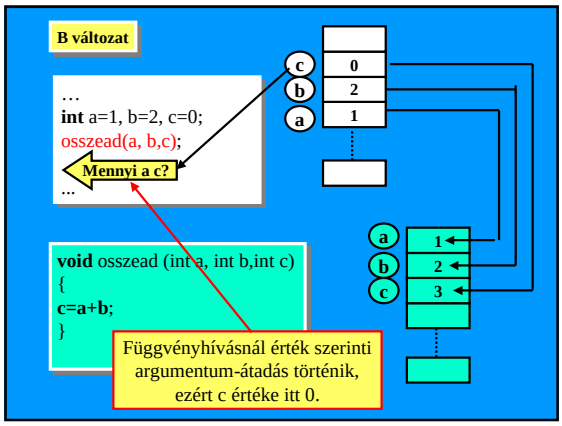

Mi történik az argumentumokkal a függvény meghívása után?
…
int a=1, b=2, c;
c= osszead(a, b);
…
int osszead (int a, int b)
{
return (a+b);
}
c
b
a
0
2
1
a
b
1
2

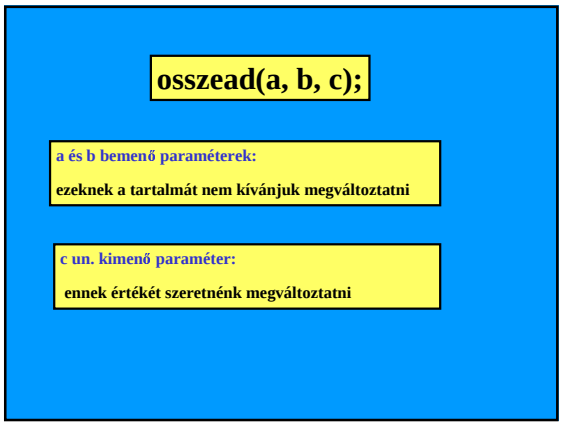

Függvény hívása
…
int a=1, b=2, c=0;
c=osszead (a, b);
…
Paraméterek, melyekben a függvény nem ad vissza értéket.
Visszatérési érték.
int osszead (int a, int b)
{
return (a+b);
}

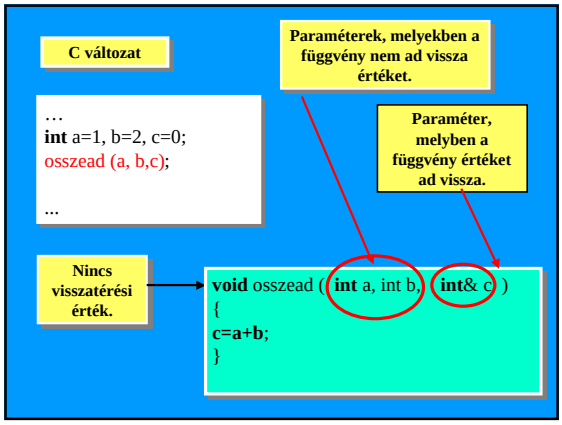

Összeadás
A változat
B változat
…
int a=1, b=2, c;
c= osszead(a, b);
Mennyi a c?
…
int a=1, b=2, c=0;
osszead(a, b,c);
Mennyi a c?
…
int osszead (int a, int b)
{
return (a+b);
}
void osszead (int a, int b, int c)
{
c=a+b;
}

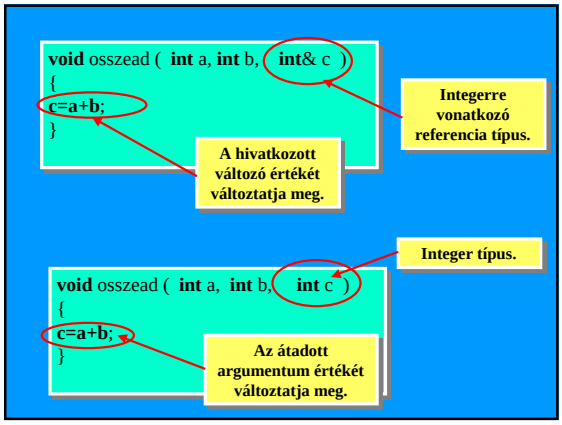

B változat
…
int a=1, b=2, c=0;
osszead(a, b,c);
Mennyi a c?
…
c
b
a
0
2
1
void osszead (int a, int b,int c)
{
c=a+b;
}

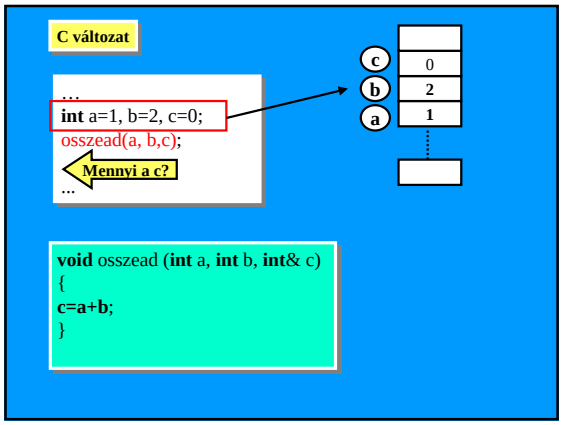

B változat
…
int a=1, b=2, c=0;
osszead(a, b,c);
Mennyi a c?
…
c
b
a
0
2
1
a
b
c
1
2
3
void osszead (int a, int b,int c)
{
c=a+b;
}

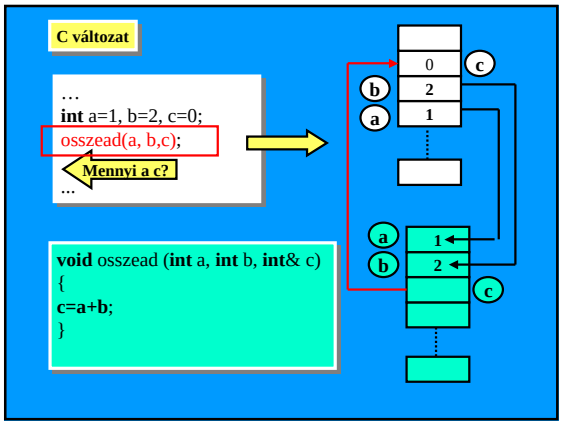

B változat
…
int a=1, b=2, c=0;
osszead(a, b,c);
Mennyi a c?
…
c
b
a
0
2
1
a
b
c
1
2
3
void osszead (int a, int b,int c)
{
c=a+b;
}
Függvényhívásnál érték szerinti argumentum-átadás történik, ezért c értéke itt 0.

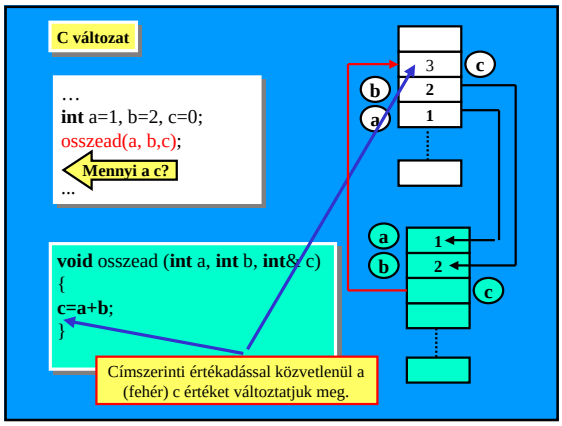

osszead(a, b, c);
a és b bemenő paraméterek:
ezeknek a tartalmát nem kívánjuk megváltoztatni
c un. kimenő paraméter:
ennek értékét szeretnénk megváltoztatni

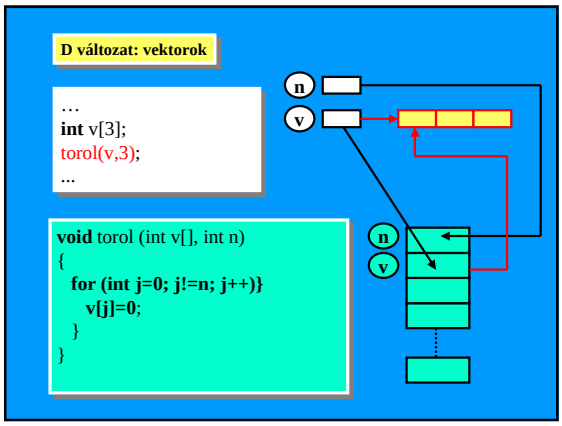

C változat
Paraméterek, melyekben a függvény nem ad vissza értéket.
Paraméter, melyben a függvény értéket ad vissza.
…
int a=1, b=2, c=0;
osszead (a, b,c);
…
Nincs visszatérési érték.
void osszead (int a, int b, int& c)
{
c=a+b;
}

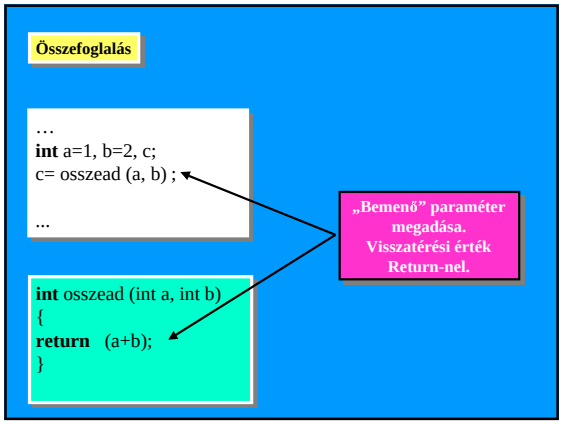

void osszead (int a, int b, int& c)
{
c=a+b;
}
Integerre vonatkozó referencia típus.
A hivatkozott változó értékét változtatja meg.
Integer típus.
void osszead (int a, int b, int c)
{
c=a+b;
}
Az átadott argumentum értékét változtatja meg.

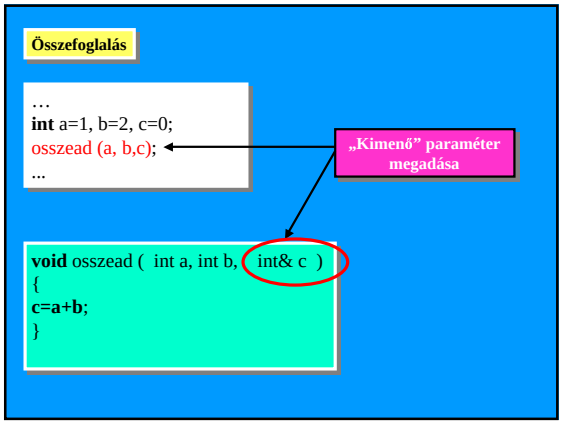

C változat
…
int a=1, b=2, c=0;
osszead(a, b,c);
Mennyi a c?
…
void osszead (int a, int b, int& c)
{
c=a+b;
}
c
b
a
0
2
1


C változat
…
int a=1, b=2, c=0;
osszead(a, b,c);
Mennyi a c?
…
void osszead (int a, int b, int& c)
{
c=a+b;
}
0
2
1
c
b
a
1
2
a
b
c


C változat
…
int a=1, b=2, c=0;
osszead(a, b,c);
Mennyi a c?
…
void osszead (int a, int b, int& c)
{
c=a+b;
}
Címszerinti értékadással közvetlenül a (fehér) c értéket változtatjuk meg.
3
2
1
c
b
a
1
2
a
b
c


D változat: vektorok
…
int v[3];
torol(v,3);
…
void torol (int v[], int n)
{
for (int j=0; j!=n; j++)}
v[j]=0;
}
}
n
v
n
v


Összefoglalás
…
int a=1, b=2, c;
c= osszead (a, b) ;
…
int osszead (int a, int b)
{
return (a+b);
}
„Bemenő” paraméter megadása. Visszatérési érték Return-nel.


Összefoglalás
…
int a=1, b=2, c=0;
osszead (a, b,c);
…
void osszead (int a, int b, int& c)
{
c=a+b;
}
„Kimenő” paraméter megadása

Összefoglalás

```
…
int v[3];
torol(v,3);
...
```

Vektor paraméter megadása

```
void torol ( int v[],     int n)
{
  for (int j=0; j!=n; j++)}
    v[j]=0;
  }
}
```

Mit szeretnénk?

maxker (v, n, i, max)

v: „vektor” paraméter
n: „bemenő” paraméter
i: „kimenő” paraméter
max: „kimenő” paraméter

```
int main(){
  //Adatok előkészítése és megjelenítése
    1
  //Maximumkeresés
  maxker( v, n, i, max)
  //Eredmény megjelenítése
    3
  return 0;
}
```

```
void maxker(const int v[], const int n, int& i, int& max)
{
……
}
```

```
#include<iostream>
using namespace std;
void maxker(const int v[], const int n, int& i, int& max){
……..
}
int main(){
  //Adatok előkészítése és megjelenítése
  //Maximumkeresés
  maxker( v, n, i, max)
  //Eredmény megjelenítése
  return 0;
}
```

A függvény meghivásakor már definiáltuk a függvényt.

```
#include<iostream>
using namespace std;
int main(){
  //Adatok előkészítése és megjelenítése
  //Maximumkeresés
  maxker( v, n, i, max)
  //Eredmény megjelenítése
  return 0;
}
```

A függvény meghívásakor még nem definiáltuk a függvényt.

```
void maxker(const int v[], const int n, int& i, int& max){
……..
}
```

Függvény deklaráció

```
#include<iostream>
using namespace std;
void maxker(const int[],  const int, int&, int& );
int main(){
  //Adatok előkészítése és megjelenítése
  //Maximumkeresés
  maxker( v, n, i, max)
  //Eredmény megjelenítése
  return 0;
}
```

A függvény meghívásakor még nem definiáltuk , de már deklaráltuk a függvényt.

```
void maxker(const int v[], const int n, int& i, int& max){
……..
}
```

#include<iostream>
using namespace std;

Függvény definició

main……{

Függvény hivatkozás

}

A függvény definíciója megelőzi a függvény hivatkozást.

#include<iostream>
using namespace std;

Függvény deklaráció - **prototipus**

main……{

Függvény hivatkozás

}

Függvény definíció

A függvény definíciója később van, mint a függvény hivatkozás.

overloading

Függvénynév újrafelhasználása

Függvények neve megegyezik, de paraméterlistája eltérő.

A paraméterlista eltérhet
- **a típusokban**
- **a paraméterek számában**

Példa

1. int fv(int);
2. int fv(int,int);

… fv(10) … 1

… fv(1,2) ... 2

Példa

int fv(double);

int fv(float);

… fv(10) … 2
… fv(10.) … 1
… fv(10.f) ... 2

Ugyanazzal a névvel, de más típusú adatokkal definiált függvények.

1.
```
void maxker (const int v[], int n, int& i, int& max)
{
}
```

2.
```
void maxker (const double v[], int n, int& i, int& max)
{
}
```

3.
```
void maxker (const float v[], int n, int& i, int& max)
{
}
```

"Kódolás"

```
#include<iostream>
using namespace std;

void maxker (const int[],  int,  int&,  int& );

int main()
{
```

Függvény deklaráció

```
//Adatok előkészítése és megjelenítése
  char barmi;
  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;
  }
```

Megegyezik a max01 kódolással.

```
//Maximumkeresés
  int i, max;

  maxker (v, n, i, max) ;
```

Megkeressük a maximumot.

```
//Eredmény megjelenítése
  cout << "A vektor egyik legnagyobb eleme: " << v[i] << ".";
  cout << endl  << "Ez a vektor " << (i+1) << ". eleme."
       << endl;
  cin >> barmi;
  return 0;
}
```

Megegyezik a max01 kódolással.

```
void maxker (const int v[],  int n,  int& i,  int& max)
{
  int 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;
  }
}
```

A maxker nem ad vissza értéket.

maxker eljárás: 1. paraméter

```
void maxker(  const int v[], const int n,   int& i,    int&  max) {
  int 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;
  }
}
```

„vektor” paraméter - az eljárás a vektor címét kapja meg.

Hivatkozás a paraméterre.

maxker eljárás: 2. paraméter

```
void maxker(    const int v[], const int n, int& i,   int&  max) {
  int 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;
  }
}
```

bemenő paraméter - az eljárás az n aktuális értékét veszi át. n értékét a függvény nem változtatja meg.

Hivatkozás a paraméterre.

maxker eljárás: 3. és 4. paraméter

```
void maxker(    const int v[],  const int n, int& i,   int&  max  ){
  int 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;
  }
}
```

kimenő paraméter - az eljárás az i és max változó értékei megváltoztatja.

Hivatkozás a paraméterre.

```
void  maxker (const int v[],  const int n,  int& i,  int& max)
{
  int 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;
  }
}
```

Lokális változó: csak a maxker eljárásban létezik.

A maximum keresés kódolása megegyezik a max01-ben tárgyaltakkal.

MAX05

- Rekord típus

Feladat

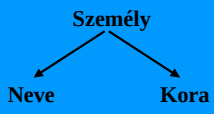
Írjunk egy olyan „univerzális programot” - **eljárást, függvényt** - amely alkalmas egy általunk megadott, tetszőleges vektor maximális elemének a meghatározására.

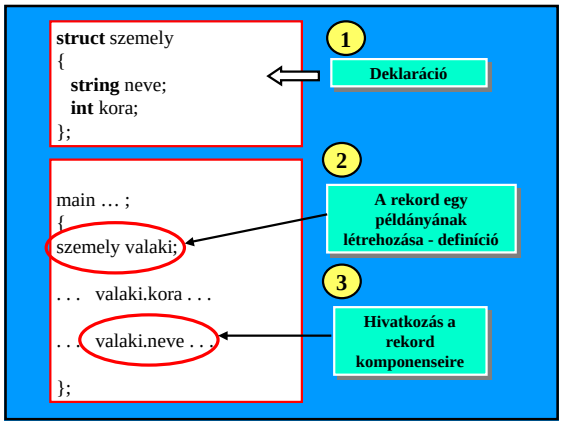
Megadjuk a vektort.

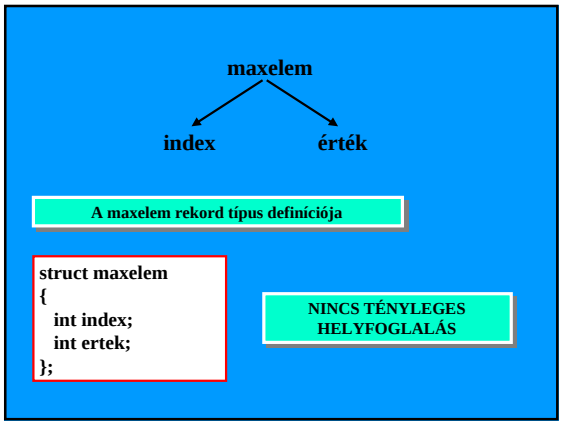
maxelem=maxker (v, n)

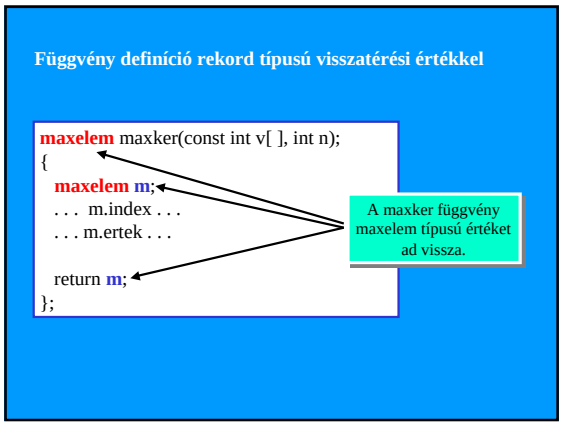
maxelem egy olyan összetett adat, mely tartalmazza a maximális elem indexét és értékét.

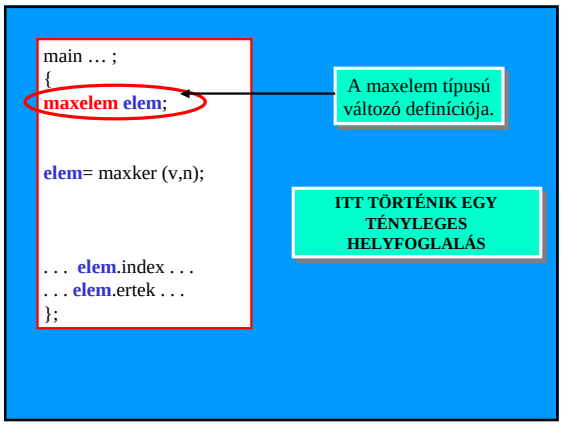
Rekord típus

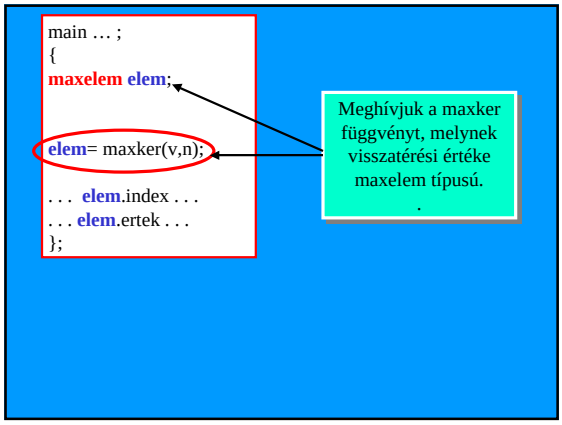


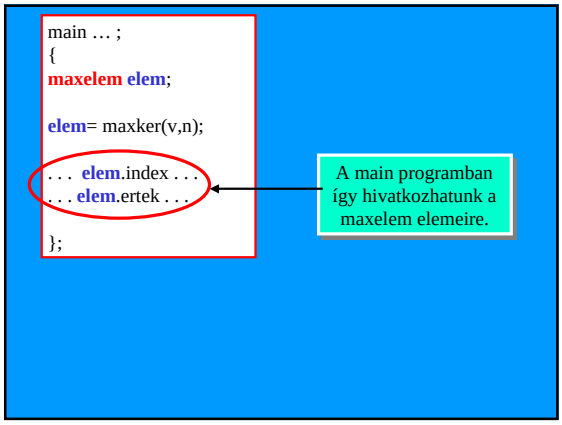

struct szemely
{
string neve;
int kora;
};
1
Deklaráció
main … ;
{
szemely valaki;
. . . valaki.kora . . .
. . . valaki.neve . . .
};
2
A rekord egy példányának létrehozása - definíció
3
Hivatkozás a rekord komponenseire


maxelem
index
érték
A maxelem rekord típus definíciója
struct maxelem
{
int index;
int ertek;
};
NINCS TÉNYLEGES HELYFOGLALÁS


Függvény definíció rekord típusú visszatérési értékkel
maxelem maxker(const int v[ ], int n);
{
maxelem m;
. . . m.index . . .
. . . m.ertek . . .
return m;
};
A maxker függvény maxelem típusú értéket ad vissza.


main … ;
{
maxelem elem;
elem= maxker (v,n);
. . . elem.index . . .
. . . elem.ertek . . .
};
A maxelem típusú változó definíciója.
ITT TÖRTÉNIK EGY TÉNYLEGES HELYFOGLALÁS


main … ;
{
maxelem elem;
elem= maxker(v,n);
. . . elem.index . . .
. . . elem.ertek . . .
};
Meghívjuk a maxker függvényt, melynek visszatérési értéke maxelem típusú.


main … ;
{
maxelem elem;
elem= maxker(v,n);
. . . elem.index . . .
. . . elem.ertek . . .
};
A main programban így hivatkozhatunk a maxelem elemeire.

Visszatérési érték: rekord - összefoglalás

1
```
struct maxelem
{
  int index;
  int ertek;
};
```

2
```
main … ;
{
maxelem elem;

elem= maxker(v,n);

};
```

3
```
maxelem maxker(const int v[ ], const int n);
{
  maxelem m;
  . . . m.index . . .
  . . . m.ertek . . .

  return m;
};
```

Rekord típusú paraméter, értékszerinti paraméterátadás

1
```
struct szemely
{
  string neve;
  int kora;
};
```

2
```
main … ;
{
szemely valaki;

fv(valaki);

};
```

3
```
void fv  ( szemely sz );
{
  sz.kora=21;

};
```

A teljes rekordot átmásolja.

Rekord típusú paraméter, címszerinti paraméterátadás

1
```
struct szemely
{
  string neve;
  int kora;
};
```

2
```
main … ;
{
szemely valaki;

fv(valaki);

};
```

3
```
void fv  (szemely &p);
{
  p.kora=21;

};
```

Csak a rekord címét másolja át.

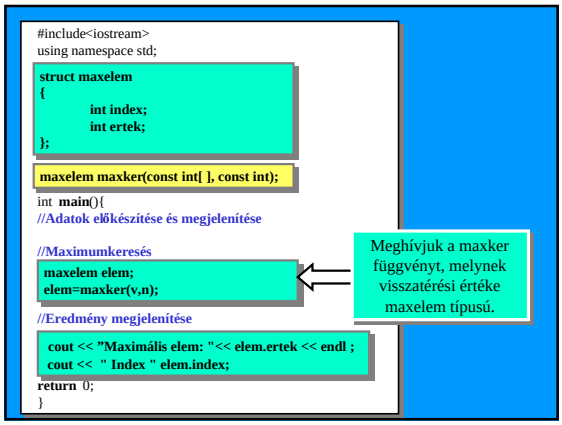
A feladat megoldása

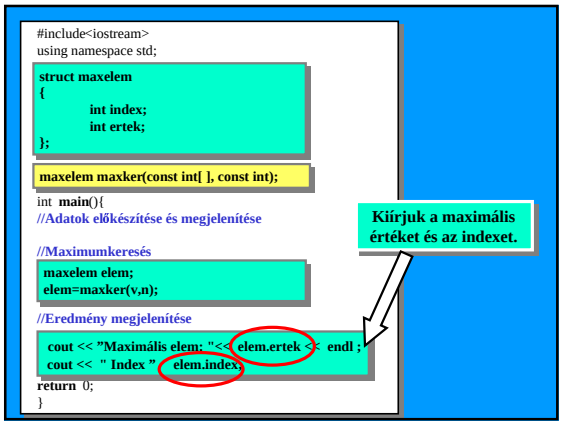
```
#include<iostream>
using namespace std;
struct maxelem
{
        int index;
        int ertek;
};
maxelem maxker(const int[ ], const int);
int  main(){
//Adatok előkészítése és megjelenítése

//Maximumkeresés
maxelem elem;
elem=maxker(v,n);
//Eredmény megjelenítése
cout << "Maximális elem: "<< elem.ertek << "\n" ;
cout <<  " Index " elem.index;
return  0;
}
```

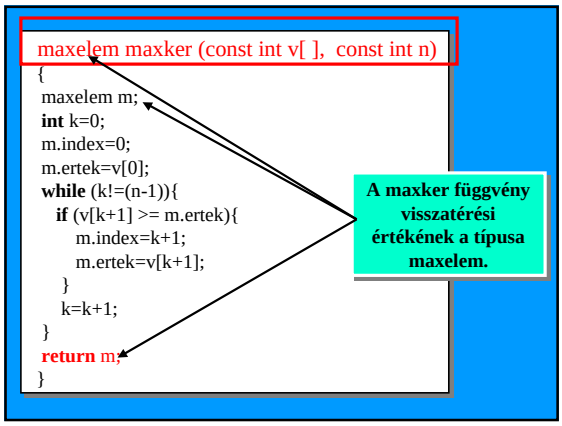
Definiálunk egy maxelem nevű rekordot. A definíciót úgy kell elhelyezni, hogy használatkor már ismert legyen.

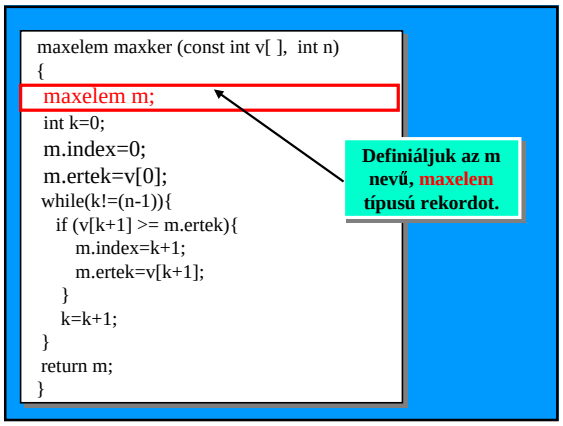
```
#include<iostream>
using namespace std;
struct maxelem
{
        int index;
        int ertek;
};
maxelem maxker(const int[ ], const int);
int  main(){
//Adatok előkészítése és megjelenítése

//Maximumkeresés
maxelem elem;
elem=maxker(v,n);
//Eredmény megjelenítése
cout << "Maximális elem: "<< elem.ertek << endl ;
cout <<  " Index " elem.index;
return  0;
}
```

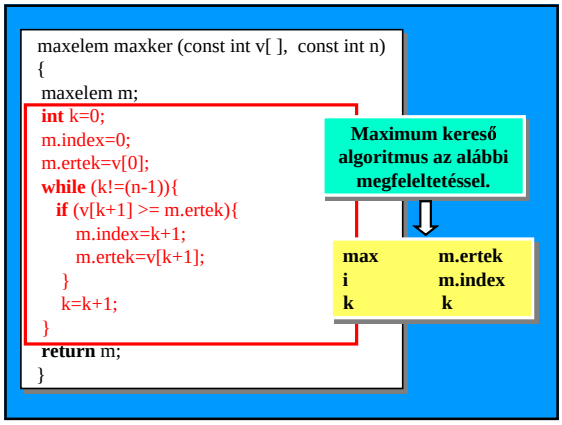
Megadjuk a maxker függvény prototípusát, mert azt később fogjuk definiálni.

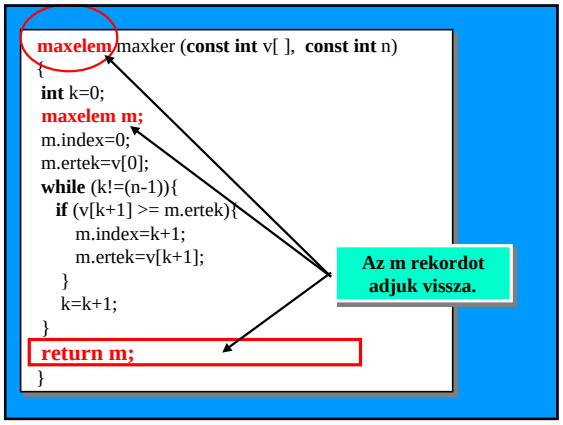

#include<iostream>
using namespace std;
struct maxelem
{
int index;
int ertek;
};
maxelem maxker(const int[ ], const int);
int main(){
//Adatok előkészítése és megjelenítése
//Maximumkeresés
maxelem elem;
elem=maxker(v,n);
//Eredmény megjelenítése
cout << "Maximális elem: "<< elem.ertek << endl ;
cout << " Index " elem.index;
return 0;
}
Meghívjuk a maxker függvényt, melynek visszatérési értéke maxelem típusú.

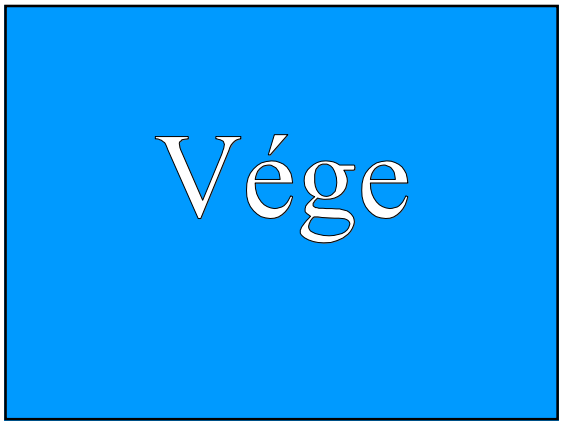

#include<iostream>
using namespace std;
struct maxelem
{
int index;
int ertek;
};
maxelem maxker(const int[ ], const int);
int main(){
//Adatok előkészítése és megjelenítése
//Maximumkeresés
maxelem elem;
elem=maxker(v,n);
//Eredmény megjelenítése
cout << "Maximális elem: "<< elem.ertek << endl ;
cout << " Index " elem.index;
return 0;
}
Kiírjuk a maximális értéket és az indexet.


maxelem maxker (const int v[ ], const int n)
{
maxelem m;
int k=0;
m.index=0;
m.ertek=v[0];
while (k!=(n-1)){
if (v[k+1] >= m.ertek){
m.index=k+1;
m.ertek=v[k+1];
}
k=k+1;
}
return m;
}
A maxker függvény visszatérési értékének a típusa maxelem.


maxelem maxker (const int v[ ], int n)
{
maxelem m;
int k=0;
m.index=0;
m.ertek=v[0];
while(k!=(n-1)){
if (v[k+1] >= m.ertek){
m.index=k+1;
m.ertek=v[k+1];
}
k=k+1;
}
return m;
}
Definiáljuk az m nevű, maxelem típusú rekordot.


maxelem maxker (const int v[ ], const int n)
{
maxelem m;
int k=0;
m.index=0;
m.ertek=v[0];
while (k!=(n-1)){
if (v[k+1] >= m.ertek){
m.index=k+1;
m.ertek=v[k+1];
}
k=k+1;
}
return m;
}
Maximum kereső algoritmus az alábbi megfeleltetéssel.
max m.ertek
i m.index
k k


maxelem maxker (const int v[ ], const int n)
{
int k=0;
maxelem m;
m.index=0;
m.ertek=v[0];
while (k!=(n-1)){
if (v[k+1] >= m.ertek){
m.index=k+1;
m.ertek=v[k+1];
}
k=k+1;
}
return m;
}
Az m rekordot adjuk vissza.


Vége