OOP I. alkalom

Egyszerű algoritmusok és leírásuk

Készítette: Dr. Kotsis Domokos

Hallgatói tájékoztató

A jelen bemutatóban található adatok, tudnivalók és információk a számonkérendő anyag vázlatát képezik. Ismeretük szükséges, de nem elégséges feltétele a sikeres zárthelyinek, illetve vizsgának.

Sikeres zárthelyihez, illetve vizsgához a jelen bemutató tartalmán felül a kötelező irodalomként megjelölt anyag, a gyakorlatokon szóban, illetve a táblán átadott tudnivalók ismerete, valamint a gyakorlatokon megoldott példák és az otthoni feldolgozás céljából kiadott feladatok önálló megoldásának képessége is szükséges.

Az algoritmus

- **Fogalma:** Jól definiált utasítások véges sorozata
- **Algoritmus készítésének lépései:**
  - A folyamatot elemi lépésekre bontjuk
  - Figyelembe vesszük az összes felmerülő lehetőséget
  - Ügyelünk, hogy az algoritmus véges sok lépésben véget érjen

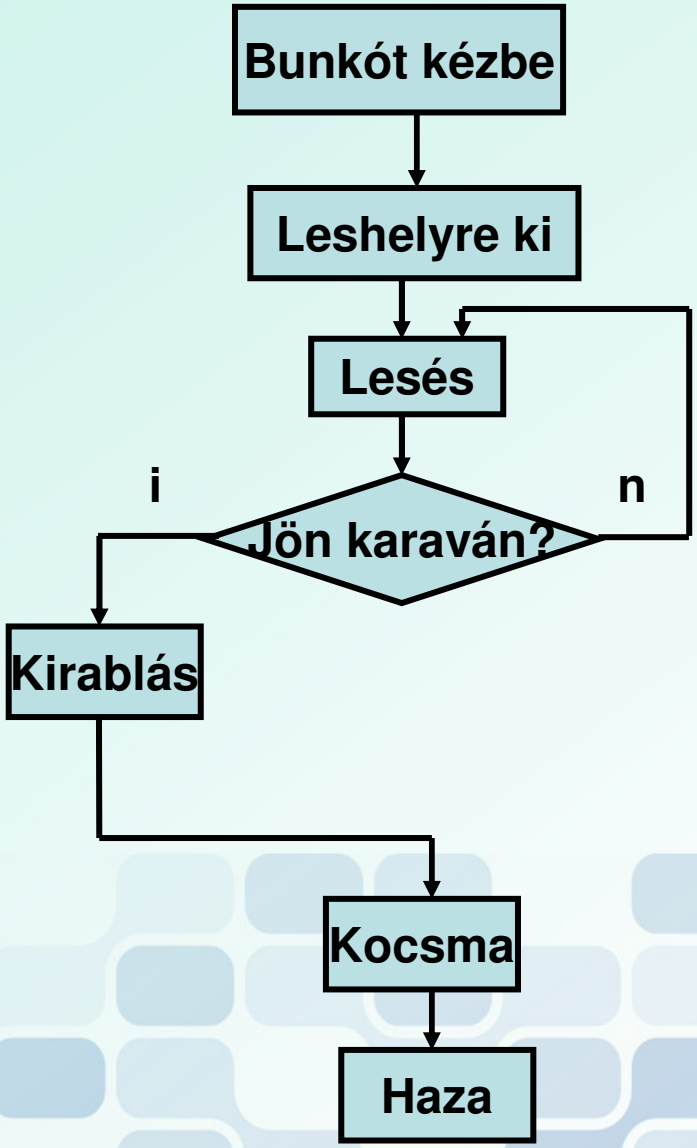
Blokkdiagram elemei

Teendő

Döntés

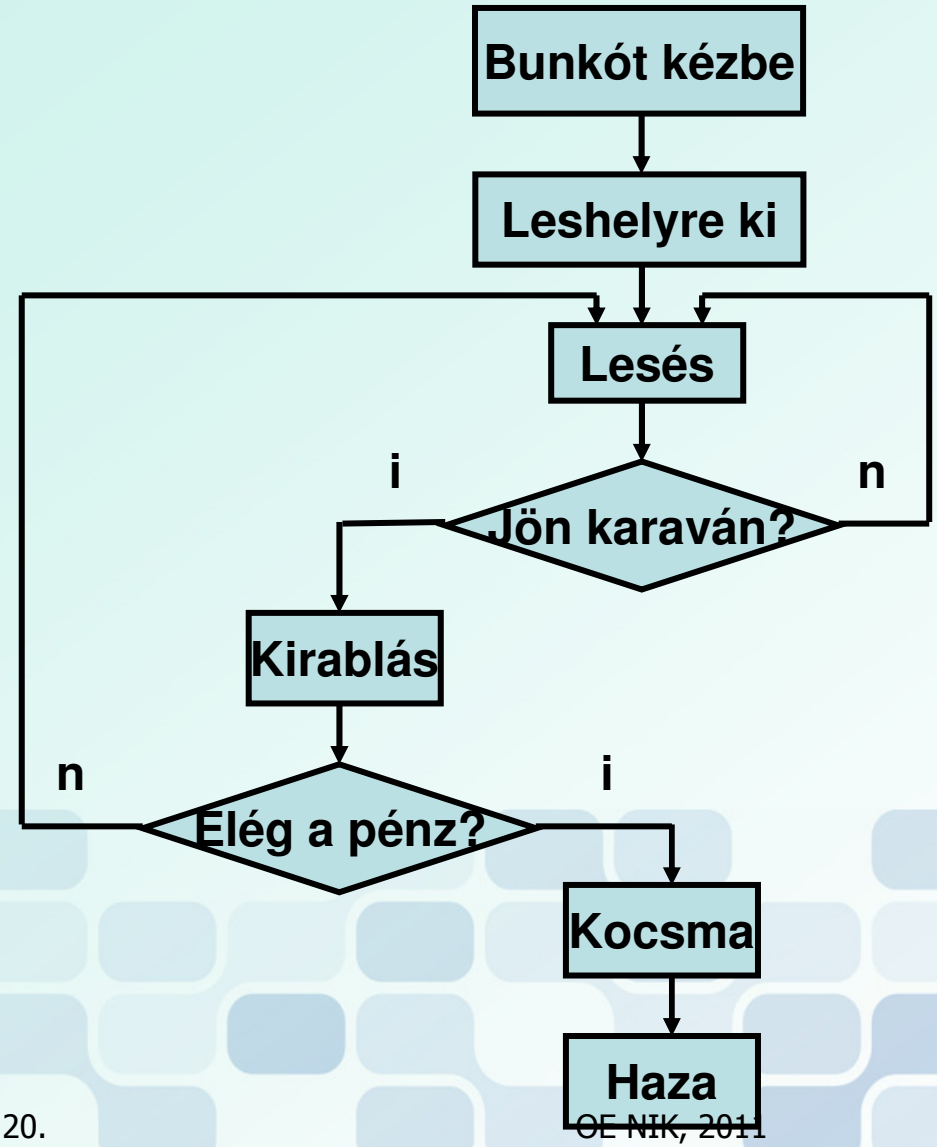
Folytatás iránya

Rablók I.


Bunkót kézbe
Leshelyre ki
Lesés
i
n
Jön karaván?
Kirablás
Kocsma
Haza

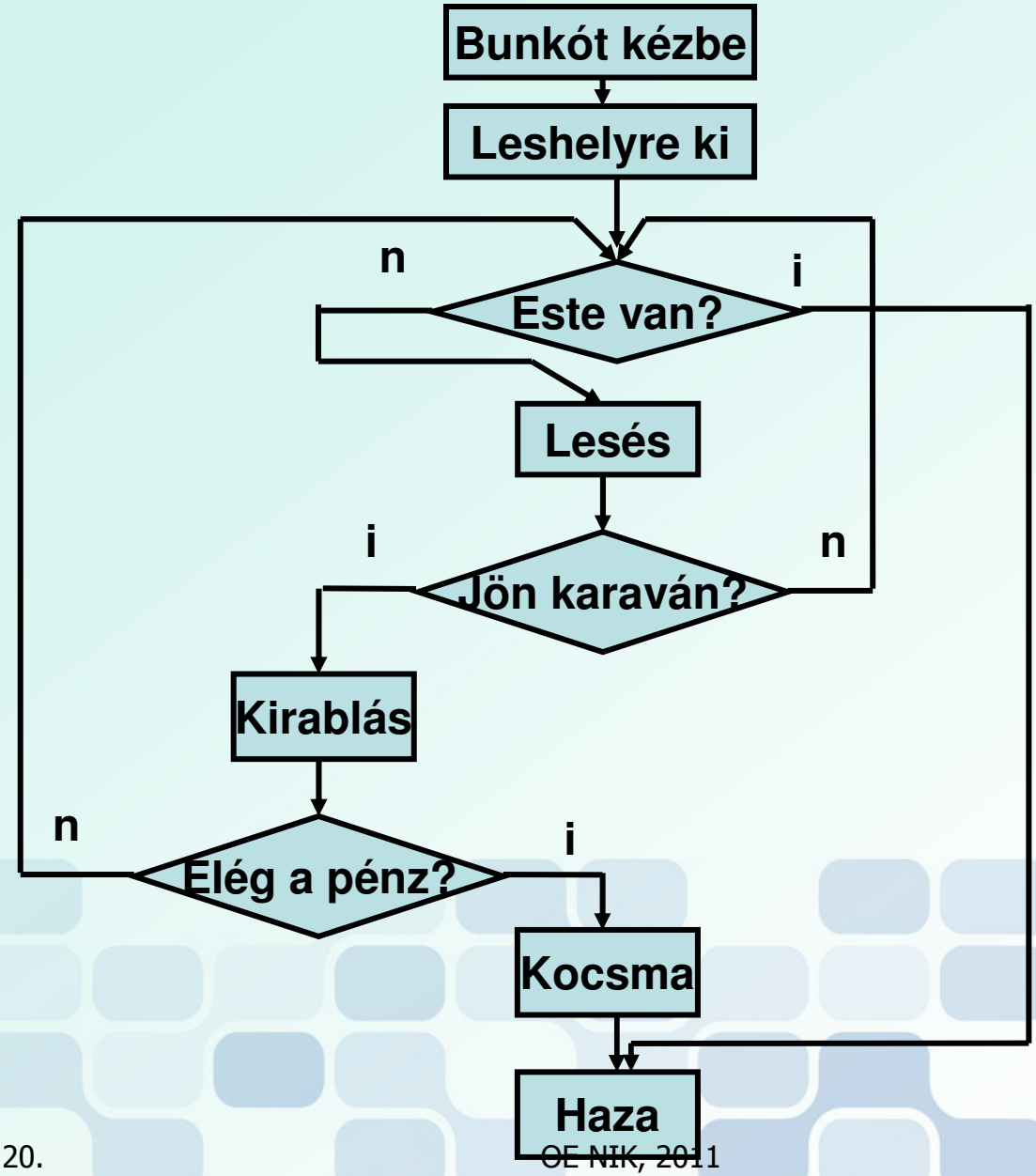

És ha nem volt
a
karavánnál
pénz?

Rablók II.


Bunkót kézbe
Leshelyre ki
Lesés
Jön karaván?
i
n
Kirablás
Elég a pénz?
n
i
Kocsma
Haza


És ha nem
is jött
karaván?

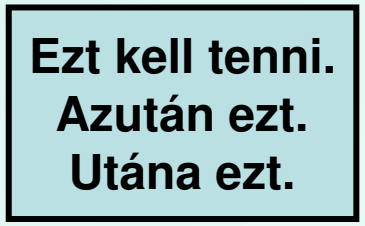
Rablók III.


Bunkót kézbe
Leshelyre ki
n
Este van?
i
Lesés
i
Jön karaván?
n
Kirablás
n
Elég a pénz?
i
Kocsma
Haza

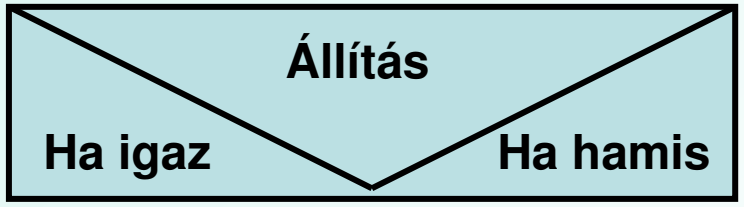

Ez már jó,
de kusza.

Struktogram elemei

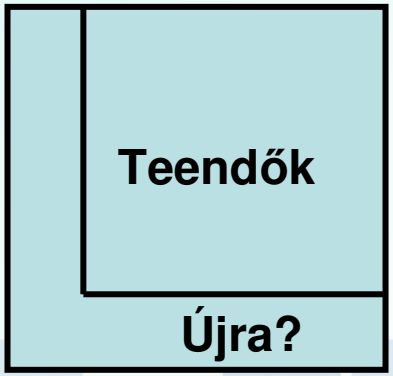
Szekvencia

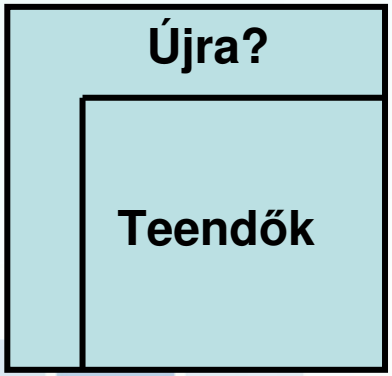


Elágaztatás



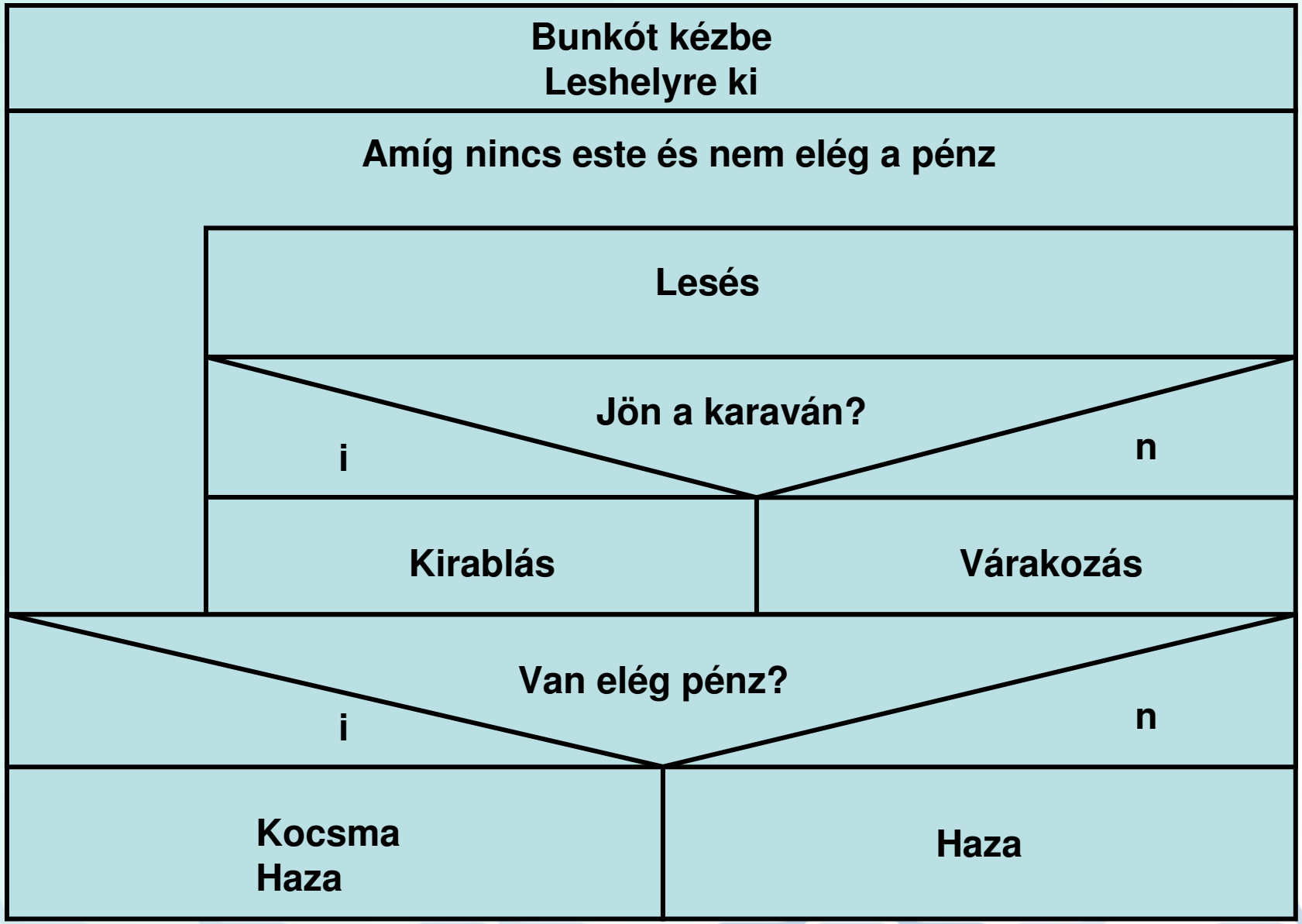
Ciklus





- **Nincsenek nyilak!**
- **Minden algoritmus leírható ezekkel az elemekkel**

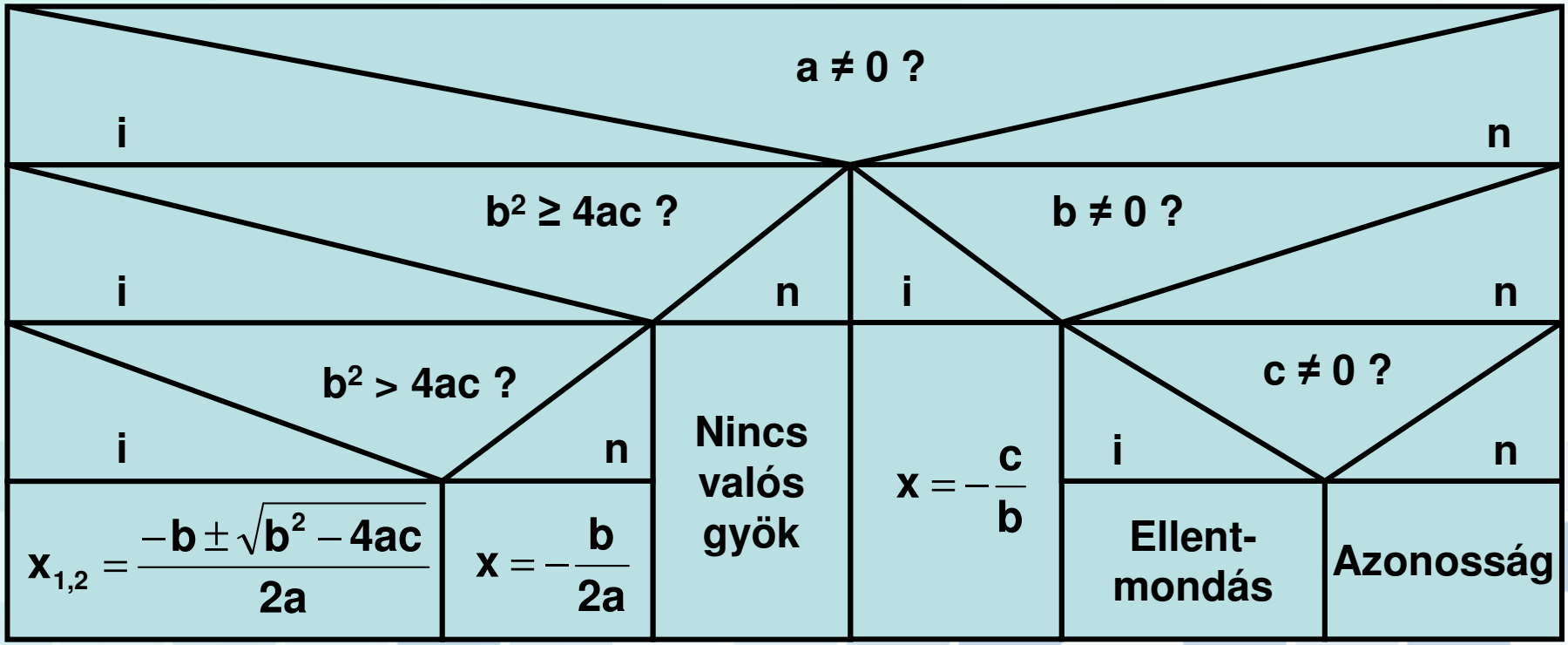
Rablók IV.


Bunkót kézbe
Leshelyre ki
Amíg nincs este és nem elég a pénz
Lesés
Jön a karaván?
i
n
Kirablás
Várakozás
Van elég pénz?
i
n
Kocsma
Haza
Haza

Feladat 1a.

Készítsen algoritmust a másodfokú egyenlet megoldására.

$ax^2 + bx + c = 0$



Szöveges leírás I.

Szekvencia

Az utasítás végén „;”
Az utasításokat kapcsos zárójelekkel fogjuk össze:

```
{  ez legyen;
   az legyen;  }
```

Elágaztatás

```
Ha (feltétel) { ez legyen; }
egyébként { az legyen; }
```

Az „egyébként” ág nem kötelező

Szöveges leírás II.

Ciklus 1.

amíg (feltétel)
{ ezek ismétlődjenek; }

Ciklus 2.

tedd { ezek ismétlődjenek; }
amíg (feltétel)

Szöveges leírás III.

Szekvencia

Az utasítás végén „;”
Az utasításokat kapcsos zárójelekkel fogjuk össze:

```
{  ez legyen;
   az legyen;  }
```

Elágaztatás

```
if (feltétel) { ez legyen; }
else { az legyen; }
```

Az „else” ág nem kötelező

Szöveges leírás IV.

Ciklus 1.

```
while (feltétel)
{ ezek ismétlődjenek; }
```

Ciklus 2.

```
do { ezek ismétlődjenek; }
while (feltétel)
```

Gyakorló feladatok

Készítsük el az alábbi feladat megoldásának struktogramját és szöveges leírását:

Beolvasunk egész számokat. Ha negatív az éppen beolvasott szám, akkor számítsuk ki az eddig beolvasott számok átlagát!

Tegyük fel, hogy ismert egy **Beolvas** és egy **Kiír** parancs.

Gyakorló feladatok

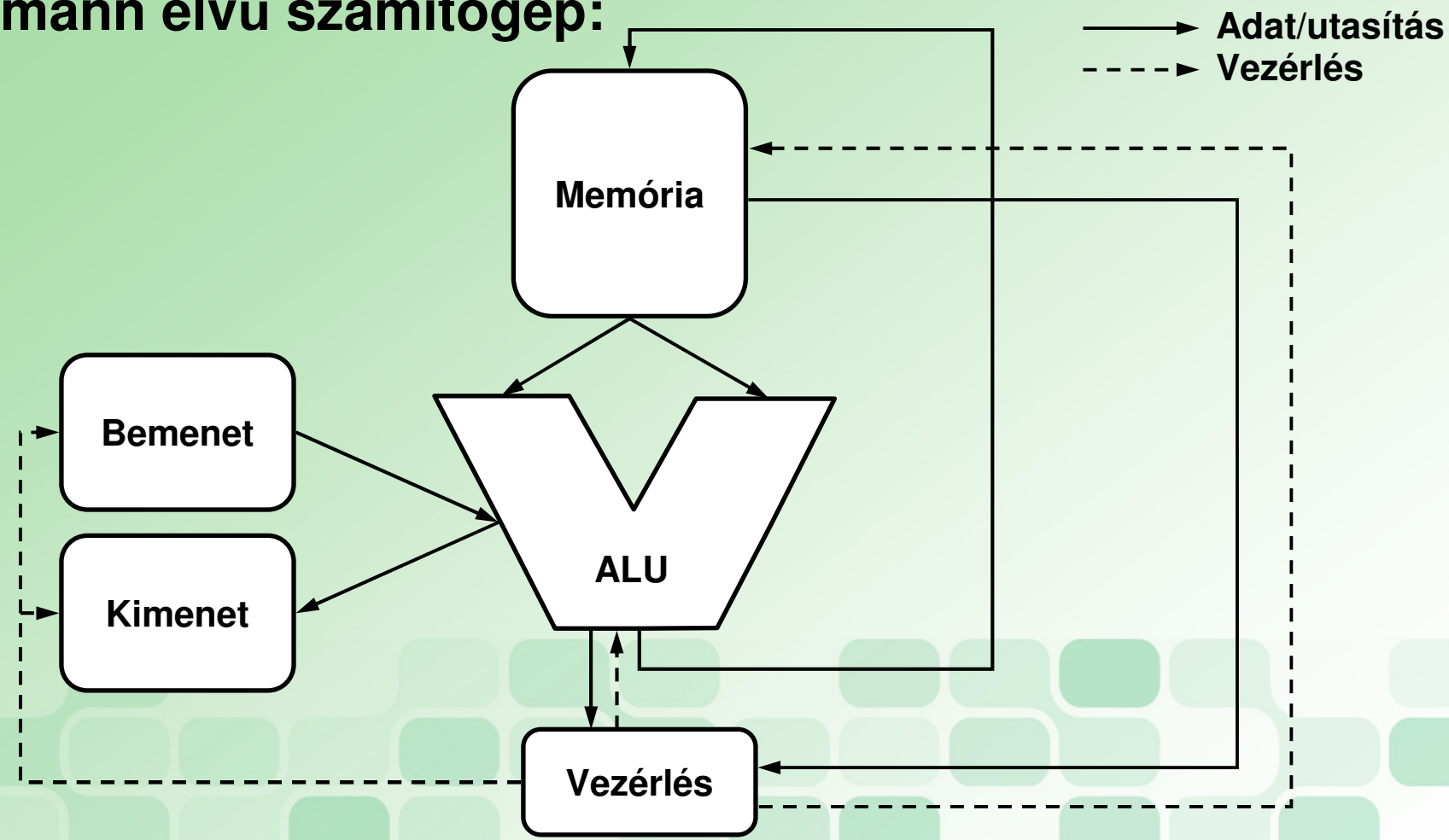
Készítsük el az alábbi feladat megoldásának struktogramját és szöveges leírását:

Beolvasunk egész számokat. Ha negatív az éppen beolvasott szám, akkor írjuk ki az eddig beolvasott számok közül a legnagyobbat és a legkisebbet!

Tegyük fel, hogy ismert egy **Beolvas** és egy **Kiír** parancs.

Algoritmus végrehajtása számítógépen

Neumann elvű számítógép:



Parancsok, adatok

- A számítógép minden adatot és utasítást bináris formában tárol a memóriában
- A tárolt bináris adat jelentése értelmezésfüggő
- Az adatokat változókban tároljuk
- A változó deklarációja határozza meg a tárolt adat méretét és értelmezését

Egyszerű adattípusok

- Számok
  - Egész
  - Valós
- Karakterek, karaktersorozatok (string-ek)
- Logikai értékek

Számok

- Bináris formában (kettes számrendszerben) vannak tárolva
- Mérete a byte (= 8 bit) többszöröse

Egész (fixpontos) számok

Lehet előjeles vagy előjel nélküli
Kis számtartomány ábrázolható, de pontos
Jellemző elnevezések:
Előjeles: integer, int
Előjel nélküli: u előtaggal vagy unsigned szóval

Valós (lebegőpontos) számok

$$\pm m * 10^{k}$$

- Normalizált szám formájában tároljuk (előjel, mantissza, karakterisztika)
- Nagy számtartomány, de nem pontos
- A karakterisztika mérete az ábrázolható számtartomány méretét, a mantissza mérete a pontosságot határozza meg
- **Jellemző elnevezések:** real, float, double
- A double kétszer annyi memóriát foglal, mint a float, cserében pontosabb

Karakterek, stringek

Karakter
Kódolásfüggő a mérete és a jelentése (pl. ASCII, UNICODE)
Jellemző elnevezés: char

Karaktersorozat (karakterfüzér)
Jellemző elnevezés: string

Logikai értékek

Két értéke lehet: igaz (true) vagy hamis (false)
Jellemző elnevezések: bool, boolean

Értékek használata I.

Az értékeket változókban tárolhatjuk
A legtöbb nyelvnél meg kell adni a változó típusát (deklarálás):

Pl. egy egész típusú változó:

```
int i;
```

Értékadás:

```
i = 3;
```

Mindezt később magyarázzuk részletesen.

Értékek használata II.

Pl. egy valós típusú változó:

```
double x;
```

Értékadás:

```
x = 3.14;
```

Pl. egy karakter típusú változó:

```
char c;
```

Értékadás:

```
c = 'c';
```

Mindezt később magyarázzuk részletesen.

Értékek használata III.

Pl. s egy string típusú változó:

```
string s;
```

Értékadás:

```
s = "kutya füle";
```

Pl. b egy logikai típusú változó:

```
bool b;
```

Értékadás:

```
b = true;
```

Mindezt később magyarázzuk részletesen.

Változók értékének beállítása

- Számokat egyszerűen
- Karaktereket ‘ ‘ között
- Stringeket " " között
- Logikai értékeket true, ill. false szavakkal

Mindezt később magyarázzuk részletesen.

Felületes segítség 1.

A programot az alábbi keretbe írjuk:

```
class Mindegy
{
        static void Main()
        {
                // ide jön a főprogram kódja
        }
}
```

Mindezt később magyarázzuk részletesen.

Felületes segítség 2.

A számítógép számára elemi utasítások többek között a matematikai alapműveletek, továbbá egyes elemi funkciók. Például a

```
string s = System.Console.ReadLine();
```

beolvas s-be egy karakter sorozatot, a

```
System.Console.Write(a);
System.Console.WriteLine(a);
```

kiírja az a változó értékét (az utóbbi sort is emel).

Mindezt később magyarázzuk részletesen.

Felületes segítség 3.

A programkód egyszerű szöveges fájl (ezért akár a Notepad-ben is szerkeszthető)
A forrásprogram neve *valami*.cs legyen.
A programot parancssori értelmezőben
(pl. Visual Studio Command Prompt) a
csc *valami*.cs
paranccsal fordíthatjuk. Ha sikeresen lefutott, elkészül a
valami.exe

(Javasolt, hogy a Path rendszerváltozó tartalmazza a …\WINDOWS\Microsoft.NET\Framework\v2.0.50727\ hivatkozást.)

Mindezt később magyarázzuk részletesen.

Otthoni gyakorló feladatok

Készítsük el az alábbi feladatok megoldásának struktogramját és programját:

1. Olvassunk be egy sugár értéket és számítsuk ki kör kerületét és területét, valamint a gömb felszínét és térfogatát!

2. Olvasson be három számot, majd írassa ki őket csökkenő sorrendben!

3. Egy háromszög oldalainak (a, b, c) hosszát olvassa be a billentyűzetről, majd megmondja, hogy a háromszög szerkeszthető-e! (A háromszög szerkeszthető, ha az (a+b>c) és (a+c>b) és (b+c>a) feltétel teljesül.)

4. Olvassa be egy hónap nevét, majd írja ki, hogy melyik évszakban van az adott hónap!

Gyakorló feladatok

Készítsük el az alábbi feladat megoldásának struktogramját és programját:

Beolvasunk egész számokat. Ha negatív az éppen beolvasott szám, akkor számítsuk ki az eddig beolvasott számok átlagát!

A `string s = System.Console.ReadLine();`
beolvasott s stringet duplapontos valós számmá a
`double szám = double.Parse(s);`
utasítással konvertálhatjuk.

Deklarácók, adatbekérés

```
class Átlag
{
   static void Main()
  {
          double szumma = 0, utolsó;
          int darab = 0;
          System.Console.Write("Adjon nem nagatív egész adatokat: ");
          string s1 = System.Console.ReadLine();
          utolsó = double.Parse(s1); // konverzió valóssá
          while (utolsó >= 0)
          {
                  szumma = szumma + utolsó;
                  darab++;
                  s1 = System.Console.ReadLine();
                  utolsó = double.Parse(s1);
          }
```

...

Deklarációk, első érték adások

```
...
        if (darab > 0)
                System.Console.WriteLine("Átlaguk: " + szumma / darab);
        else
                System.Console.WriteLine("Nincs elég adat!");
        System.Console.ReadLine();
    } // Main vége
} // Program vége
```

Gyakorló feladatok

Készítsük el az alábbi feladat megoldásának struktogramját és programját:

Beolvasunk egész számokat. Ha negatív az éppen beolvasott szám, akkor számítsuk ki az eddig beolvasott számok átlagát, de a számításból hagyjuk ki a legnagyobbat és a legkisebbet!

Deklarácók, adatbekérés

```
class Programm
{
   static void Main()
  {
          double szumma = 0, max = 0, min, utolsó;
          int darab = 0;
          System.Console.Write(" Adjon nem nagatív egész adatokat: ");
          string s1 = System.Console.ReadLine();
          utolsó = double.Parse(s1); // konverzió valóssá
          min = utolsó;
  ...
```

Ciklus, eredmények

```
...
        while (utolsó >= 0)
        {
                szumma = szumma + utolsó;
                darab++;
                if (utolsó > max) max = utolsó;
                if (utolsó < min) min = utolsó;
                s1 = System.Console.ReadLine();
                utolsó = double.Parse(s1);
        }
        if (darab > 2)
                System.Console.WriteLine("Átlaguk: " +
                        + (szumma – max - min) / (darab - 2));
        else
                System.Console.WriteLine("Nincs elég adat!");
        System.Console.ReadLine();
    } // Main vége
} // Program vége
```

Irodalom, feladatok

- Kotsis-Légrádi-Nagy-Szénási: Többnyelvű programozástechnika, PANEM, Budapest, 2007

  1. fejezet (Az algoritmikus programozás alapjai), pp. 11-30.
- Faraz Rasheed: C# School, Synchron Data, 2006 http://www.programmersheaven.com/2/CSharpBook
- Reiter István: C# jegyzet, DevPortal, 2010, http://devportal.hu/content/CSharpjegyzet.aspx