Programozás I.

Matematikai lehetőségek
Műveletek tömbökkel
Egyszerű programozási tételek & gyakorlás

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.

Programozás I.

Matematikai lehetőségek

Műveletek tömbökkel

Egyszerű programozási tételek & gyakorlás

Matematikai függvények

- Elérésük: System.Math.Bármi();
  using System; esetén Math.Bármi();
- Math.PI (nem kell zárójel, mert ez nem eljárás, hanem konstans!)

Függvény	Bemenet	Kimenet
Sqrt() – négyzetgyök	double	double
Pow() – hatvány	double + double	double
Sin(), **Cos()**, **Tan()**	double (radián)	double
Asin(), **Acos()**, **Atan()**	double	double (radián)
Abs() – abszolút érték	tetszőleges	= bemenet típusa
Min(), **Max()**	2 tetszőleges	= bemenet típusa
Round(), **Ceiling()**, **Floor()** – (kerekítések)	decimal / double	= bemenet típusa

Gyakorló feladat

Készítsük el a következő feladat C# kódját:

A számítógép hozzon létre egy véletlen számot a [0..100] intervallumban!

A felhasználónak a feladata a szám kitalálása, a számítógép írja ki, hogy a tipp „Túl nagy”, vagy „Túl kicsi”

```
using System;
Random R = new Random();
int szam=R.Next(0, 101);
```

Gyakorló feladat

Készítsünk másodfokú egyenletet megoldó programot!

Programozás I.

Matematikai lehetőségek

Műveletek tömbökkel

Egyszerű programozási tételek & gyakorlás

Tömbbel végezhető tevékenységek C#-ban

1. Deklaráció
   int[] tomb**;**
2. Tömblétrehozás
   tomb = **new int**[10];
3. Értékadás
   tomb[5] = 25; vagy tomb[5] = 6 * 2 – 29;
4. Érték lekérdezése
   5 * 10 – tomb[5] + 2

A deklaráció és a tömblétrehozás összevonható:

int[] tomb = **new int**[10];

Tömbbel végezhető tevékenységek C#-ban (melyik kód helyes?)

```
int[] tomb;
Console.WriteLine(tomb[2]);
```

```
int[] tomb = new int[4];
Console.WriteLine(tomb[-1]);
```

```
int[] tomb = new int[4];
Console.WriteLine(tomb[4]);
```

```
int[] tomb = new int[4];
Console.WriteLine(tomb[0]);
```

```
int[] tomb = new int[4];
Console.WriteLine(
        tomb[tomb.Length]);
```

```
int[] tomb = new int[4];
tomb[3] = 22;
Console.WriteLine(tomb[3]);
```

A for utasítás

```
for (inicializátor ; feltétel ; iterátor )
   utasítás
```

- Az inicializátor és az iterátor tetszőleges utasítás lehet
- Működése:
  - Belépéskor egyszer végrehajtódik az inicializátor
  - Minden ciklusmenetben kiértékelődik a feltétel
  - Amennyiben a feltétel igaz, az utasítás ciklusmag egyszer lefut
  - A ciklusmag végeztével végrehajtódik az iterátor és ismét kiértékelődik a feltétel
  - A ciklus akkor ér véget, amikor a feltétel hamissá válik, ellenkező esetben újabb ciklusmenet következik
- Általában az inicializátor egy számlálót állít be, az iterátor pedig ezt a számlálót növeli vagy csökkenti
  - Legtöbbször akkor használjuk, ha előre ismert számú alkalommal szeretnénk végrehajtani egy utasítást

A foreach

```
foreach (típus változó in gyűjtemény)
    utasítás
```

- Lehetővé teszi egy utasítás végrehajtását egy adott gyűjtemény összes elemére
  - A „gyűjtemény” pontos fogalmát később részletesen tárgyaljuk
  - A tömbök gyűjtemények, tehát a foreach utasítás használható hozzájuk
- Működése:
  - Belépéskor létrejön egy „típus” típusú változó („iterációs változó”)
    - Ez a változó csak az utasításon belül használható
  - Az utasítás annyiszor hajtódik végre, ahány elemet tartalmaz a gyűjtemény
  - Az iterációs változó minden egyes végrehajtásnál felveszi a gyűjtemény soron következő elemének értékét
- Az iterációs változó az utasításban nem módosítható
  - Erre a célra a for utasítás használható

Break, continue

- Minden C#-os ciklusban alkalmazhatóak

```
foreach (int i in tomb)
{
    if (i < 0)
    {
        vanNegativ = true;
        break; //kilép a ciklusból
    }
}
```

```
for (int i = 0; i < tomb.Length; i++)
{
     if (tomb[i] < 0) continue; //kihagyja ezt a ciklusmenetet,
                                //a következő jön azonnal
     //… csinálunk valamit a nemnegatív számmal...
}
```

- Többnyire helyettesíthetők (continue helyett if v. if-else, break helyett pl. a for ciklus fejében feltétel)
- Elméleten (pl. vizsgán) és egyelőre gyakorlaton is TILOS őket használni!

Programozás I.

Matematikai lehetőségek

Műveletek tömbökkel

Egyszerű programozási tételek & gyakorlás

Egyszerű programozási tételek

- A következő feladatokban az A tömb adott számú egész számot tartalmaz. A tömb elemei 0 és 100 között véletlen számok. A tömb feltöltése érdekében készítsen önálló metódust!
- Az A tömb mérete: A.Length segítségével határozható meg
- Készítsen metódust, mely kilistázza tömb elemeit!
- Minden egyes feladatot külön metódussal valósítson meg!
- (Érdemes egy olyan metódust is készíteni, amely egy szám öttel való oszthatóságát vizsgálja.)

Metódus szintaktikája

```
static Típus MetódusNév(ParTípus1 par1, ParTípus2 par2)
{
    //metódustörzs
    return ...;
}
```

Példa1:

```
static bool OszthatóÖttel(int szám)
{
    return szám % 5 == 0;
}
```

Példa2:

```
static int[] BeolvasTömb(int[] a)
{
    Console.Write("Elemszám = ");
    a = new int[int.Parse(Console.ReadLine())];
    for (int i = 0; i < a.Length; i++)
    {
        Console.Write(i + ". elem = ");
        a[i] = int.Parse(Console.ReadLine();
    }
    return a;
}
```

Feladatok

1. Határozza meg a tömb elemeinek összegét! (sorozatszámítás tétel)
2. Szerepel-e a tömbben öttel osztható szám? (eldöntés tétel)
3. Ha tudjuk, hogy szerepel a tömbben öttel osztható szám, akkor mi az indexe? (kiválasztás tétel)
4. Szerepel-e a tömbben öttel osztható szám, és ha igen, akkor mi az indexe? (A metódus visszatérési értéke -1 legyen, ha nincs öttel osztható szám, egyébként pedig a megfelelő elem indexe.) (lineáris keresés tétel)

Feladatok

5. Hány darab öttel osztható szám van a tömbben? (megszámlálás tétel)
6. Határozza meg a tömb legkisebb értékének indexét! (maximumkiválasztás tétel)

Gyakorló feladatok

Készítsünk algoritmust, majd programot, amely elvégzi egy egydimenziós tömb feltöltését a konzolról beolvasott adatokkal!

Tömb indexe legyen 0

Amíg a tömb indexe nem haladja meg a maximumot

- Következő elem beolvasása
- Tömb indexének növelése

Gyakorló feladatok

Készítsünk algoritmust, majd programot, amely a konzolról beolvassa egy kétdimenziós, 3x3-as tömb minden elemét, majd kiírja a tömb teljes tartalmát!

Tömb sorindexe legyen 0
Tömb oszlopindexe legyen 0

Amíg a tömb sorindexe nem haladja meg a maximumot

Amíg a tömb oszlopindexe nem haladja meg a maximumot

Következő elem beolvasása

Tömb oszlopindexének növelése

Tömb sorindexének növelése
Tömb oszlopindexe legyen 0

Gyakorló feladatok

Olvasson be maximum n db. egész számot egy tömbbe, ezután döntse el a negatívok összegét, a pozitívok átlagát és a zérusok darabszámát!

Az előző feladat tömbjéből másolja ki egy másik, megfelelő elemszámú tömbbe a nemnegatív számokat.

Kérjen be a felhasználótól hónapnevet! Egészen addig ne fogadja el a bemenetet, amíg tényleg hónapnevet nem ír be. (Ellenőrizze: a hónapneveket tömbben tárolja, ha a felhasználó bemenete nem szerepel ebben, akkor kérje újra.)

Töltsön fel egy mátrixot FOR ciklusok segítségével, majd állítsa elő a transzponáltját (a sorokat fel kell cserélni az oszlopokkal), és írassa ki mindkét mátrixot!

Irodalom, feladatok

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