C# programozási segédlet és feladatgyűjtemény

Adattípusok

Egyszerű típusok

Számtípusok: (12. oldal)

Típus neve	Értéktartomány
Egész számtípusok:	
byte	0 – 255
short	-32 768 – 32767
ushort	0 – 65535
int	-2 147 483 648 – 2 147 483 647
uint	0 – 4 294 967 295
long	-9 223 372 036 854 775 808 – 9 223 372 036 854 775 807
ulong	0 – 18 446 744 073 709 551 615
Valós számtípusok:	
float	$\pm 1{,}5*10^{-45}$ pontossággal $\pm 3{,}4*10^{38}$
double	$\pm 5*10^{-324}$ pontossággal $\pm 1{,}7*10^{308}$
decimal	$\pm 1*10^{-28}$ pontossággal $\pm 7{,}9*10^{28}$

Char típus: Karakter típus. Mérete 2 Byte. Értéke tetszőleges Unicode karakter lehet, amit ’ ’ jelek közé kell tenni. (43. oldal)

Bool típus: A C# logikai típusa, amelynek értéke TRUE (igaz, 1), vagy FALSE (hamis, 0) lehet. Ha egy logikai típusú változó neve önmagában – érték nélkül – szerepel a programban, akkor az értéke: IGAZ. (21., 29. oldal)

String típus: (31., 65., 54. oldal)

Tömb típus: lásd bővebben a jegyzet 2.13. fejezetében!

Rekord/Struktúra típus: (92. oldal)

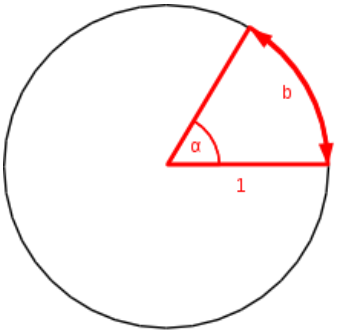
2.7. Matematikai függvények

π értéke:	Math.PI;
négyzetgyökvonás:	Math.Sqrt(szam);
hatványozás:	Math.Pow(alap, kitevo);
szinusz:	Math.Sin(x);
koszinusz:	Math.Cos(x);
kerekítés:	Math.Round(valos_szam); vagy Math.Round(valos_szam, tizedes_jegy);

Az első *kerekítés* a matematika szabályának megfelelően kerekít egész számra, a második pedig a megadott tizedesjegyig. Pl: Ha a=4,672834, akkor Math.Round(a) = 5, illetve Math.Round (a, 3) = 4,673

6. feladat:
Írjunk programot, amely bekér két egész számot, az alapot és a kitevőt a billentyűzetről és kiírja a hatványt, majd ennek a számnak a gyökét (a hatvány gyökét) 2 tizedes pontossággal! Írjuk ki a képernyőre, hogy az alap csak pozitív lehet! (Név: Matek1)

7. feladat:
Írjunk programot, amely bekér egy szöget fokban, és kiírja a szög szinuszát 4 tizedes pontossággal! Ne felejtsük el, hogy a szöget át kell váltani radiánba. Mivel 180º=π, ezért a π értékét meg kell szorozni a számítandó szög 180-ad részével. (Név: Matek2)



(A **radián** v. ívmérték a síkszögek egyik mértékegysége. Dimenzió nélküli mértékegység, mivel két hosszúság hányadosa. *Definíció:* Egy radián az a szög, amely alatt a sugárral megegyező nagyságú ívhossz a középpontból látszik. Másképp a radián a sugárnyi hosszúságú ívhosszhoz tartozó középponti szög. A radián definíciója egységkörben: α=b.
Radiánból fokba való *átszámítás:* a kör középponti szögei és e szögek ívhossza egyenesen arányosak, α~i. Ha α=360° ~ i=$K_{kör}$=2rπ, ekkor α=K/r=2rπ/r=2π. Tehát 360° éppen 2π≈6,28... radián. Innen **1°=2π/360 rad és 1 rad=360/(2π)°**)

8. feladat:
Kérjük be egy kör sugarát, és írjuk ki a kerületét és a területét 2 tizedes pontossággal! (Név: Matek3)

Gyakorló feladatok:

1. Kérje be egy kocka oldalának hosszát és számítsa ki a kocka felszínét és térfogatát! Az eredményt írja ki 3 tizedes pontossággal a képernyőre!
2. Kérje be egy henger sugarát és magasságát, majd számítsa ki a henger felszínét és térfogatát! Az eredményt írja ki 2 tizedes pontossággal a képernyőre!
3. Kérjen be egy számot, és írja ki a köbgyököt 4 tizedes pontossággal!
4. Írja be a táblázatba az első 3 feladat megoldásához szükséges változók nevét és típusát, valamint az alkalmazott matematikai függvényt!

Feladat száma	Változó neve	Változó típusa	Matematikai függvény
1			
2			
3			

5. Készítsen programot, amely segít a pénztárosnak a papírpénzek értékének megszámolásánál! Kérje be melyik bankjegyből (500, 1000, 2000, 5000, 10000, 20000) hány darab van, és a végén adja meg az összes bevételt.
6. Egy pénztáros a napi bevételének 5%-át megkapja jutalomként. Kérje be a napi bevételt, és írja a képernyőre mennyi a jutalom! A jutalmat kerekítse egész értékre!
7. Adott egy derékszögű háromszög 2 befogója. Határozza meg ennek ismeretében az átfogót! (Segítség: *Pitagorasz-tétel* $a^2+b^2=c^2$)

2.8. Elágazás (feltételes utasítás)

```
if (kifejezés) …
```

if utasítás: HA kulcsszóval kezdjük,
kifejezés: logikai kifejezés, relációs művelet, értéke igaz, vagy hamis lehet. Az összehasonlító művelet eredménye tehát *bool* típus (logikai adattípus) lehet.

Relációs műveletek:

a > b	*a* nagyobb, mint *b*,
a >= b	*a* nagyobb vagy egyenlő, mint *b*,
a < b	*a* kisebb, mint *b*,
a <= b	*a* kisebb vagy egyenlő, mint *b*,
a = = b	*a* egyenlő *b*-vel,
a != b	*a* nem egyenlő *b*-vel.

Az *a* és *b* változó, vagy kifejezés lehet.

Az if-else utasítás alakja:

```
if (kifejezés)
        utasítás1(ok);
else
        utasítás2(ok);
```

Fajtái:

a) <u>Egyágú:</u> ha a feltétel igaz, akkor csináljunk valamit.

```
if ( a > b )
        {
        Console.WriteLine(”a nagyobb, mint b”);
        }
```

b) <u>Kétágú:</u> Ha a feltétel igaz, akkor csináljunk valamit, ha hamis, akkor valami mást.

```
if ( a > b )
        {
        Console.WriteLine(”a nagyobb, mint b”);
        }
else
        {
        Console.WriteLine(”a kisebb, vagy egyenlő, mint b”);
        }
```

9. feladat:

Kérjünk be egy egész számot a billentyűzetről, és ha a szám nagyobb, mint 0, akkor írjuk ki a képernyőre, hogy „A szám pozitív”! (Név: Felteteles1)

10. feladat:
Kérjünk be egy másodfokú egyenlet együtthatóit és számítsuk ki a gyököket 2 tizedes pontossággal! Ha a diszkrimináns negatív, írjuk ki a képernyőre: „Nincs megoldás”! (Név: Masodfoku)

c) Többirányú: lásd 2.12. fejezetben!

2.9. Logikai műveletek:

- A tagadás, vagy negáció jele: !
  Pl.: ! kifejezés – Az ellenkezőjére változtatja a kifejezés értékét.
- Az ÉS, vagy konjukció jele: &&
  Pl.: A && B – Csak akkor lesz igaz, ha mindkét állítás igaz.
- A VAGY, vagy diszjunkció jele: || (Alt Gr+W)
  Pl.: A || B – Akkor lesz igaz, ha bármelyik kifejezés igaz.

Gyakorló feladatok:
1. Kérjünk be 2 számot. Írjuk ki a képernyőre a nagyobbat!
2. Írjunk ki 2 tetszőleges szám hányadosát. (!!!Nullával nem lehet osztani!!!)
3. Három tetszőleges számból döntsük el, hogy lehetnek-e egy háromszög oldalai! (háromszög szerkeszthetőségének szabálya)
4. Adott egy tetszőleges pont koordinátáival. Határozza meg, melyik síknegyedben van!
5. A középszintű érettségin maximum 150 pont szerezhető. Kérje be egy tanuló elért pontszámát és írja ki, milyen érdemjegyet kapott (betűvel és számmal), ha a százalékos értékelés a következő:

Százalékérték:	Érdemjegy:
0-24%	Elégtelen (1)
25-39%	Elégséges (2)
40-59%	Közepes (3)
60-79%	Jó (4)
80-100%	Jeles (5)

6. Egy tetszőleges számról mondjuk meg, hogy osztható-e maradék nélkül 3-mal!
7. Ha ismerjük a víz hőmérsékletét, adjuk meg milyen halmazállapotú (szilárd, folyékony, gáz)!
8. Kérjen be 2 számot és a nagyobbat ossza el a kisebbel! Az eredményt 2 tizedesjegy pontossággal írja ki! (!!!Nullával nem lehet osztani!!!)

2.10. Véletlen számok előállítása

A C# a *Random utasítás*t (osztályt) használja a véletlen szám előállításra.

```
Random veletlen = new Random();
veletlenszam = veletlen.Next(100);
```

Ekkor a *veletlenszam* nevű változó csak egész számot vehet fel értékül, melynek legkisebb értéke 0, legnagyobb pedig 99 lehet. A 100 az előállítható számok darabszámát adja meg.

*Negatív számok előállítás*ának 2 módja:
1. veletlenszam = veletlen.Next(101)-50;
2. veletlenszam = veletlen.Next(-50,51);

Mindkét esetben a *veletlenszam* a [-50, 50] intervallumból vesz fel értéket. (Az első módszer lehetővé teszi nem egész számok véletlenszerű előállítását is, meghatározott tizedesjegyig. Ilyenkor vigyázni kell a változó deklarációjával.)

11. feladat:

Állítsunk elő véletlenszerűen két számot a [0,10] tartományból, és írjuk ki a számokat a képernyőre, majd azt, hogy melyik szám a nagyobb! Ha egyenlők írjuk ki: „A két szám egyenlő”! (Név: Veletlen1)

12. feladat:

Állítsunk elő véletlenszerűen egy egész számot az [1,10]-ból, és írjuk ki, hogy a szám páros-e vagy páratlan! (Név: Véletlen2)

2.11. Ciklusok

A ciklusok *feladata*, hogy a program valamely pontján ugyanazokat az utasításokat többször is végrehajtsa.

A ciklusok 3 fő *rész*ből állnak:

- fej,
- mag,
- vég.

A fejben vagy a magban lehetnek az ismétlődésre vonatkozó információk. A magban az ismétlődő utasítás(ok) szerepelnek.

A ciklusok 4 *fajtája*:

- elöltesztelő,
- hátultesztelő,
- előírt lépésszámú (számlálóvezérelt),
- **ForEach:**

Abban az esetben, ha például egy tömb, vagy egy kollekció minden elemén végig kell mennünk egy ciklussal, akkor felesleges egy *for* ciklust készítenünk ehhez. Sokkal egyszerűbben használható a *foreach*.

Példa: Egy tömbön megyünk végig, mely egész számokat tartalmaz. A ciklusnak az a feladata, hogy a tömb elemeinek értékét összegezze, majd megjelenítse az eredményt.

```
int[] array = new int [] {100, 200, 300};
int sum = 0;

foreach (int i in array)
    {
        sum += i;
    }
```

A *foreach* használatához a kulcsszó után zárójelben létrehozunk egy változót *i* névvel *int* típussal, majd az *in* kulcsszó megadása után rendelkezhetünk arról, hogy mely elemen kell végig menni. Ez jelen esetben az *array* nevű int értékeket tartalmazó tömb lesz. Ebben a példában az i változó azonban nem a ciklusváltozó lesz, hanem egy olyan változó, mely az *array* tömb aktuális elemének értékét tárolja!

2.11.1. Elöltesztelő ciklus

Itt a ciklus változót előzőleg kell deklarálnunk és annak kezdő értéket adnunk. A **while**-nál csak az a feltétel adható meg, amely befolyásolja, hogy a ciklus meddig fusson. A ciklusváltozó értékének növeléséről a ciklusmagban kell rendelkeznünk.

```
while (kifejezés)
    {
       utasítás(ok);
    }
```

13. feladat:
Kérjünk be egy számot és írjuk ki, hogy az prímszám-e vagy sem! (Név: Eloltesztelo)

2.11.2. Hátultesztelő ciklus

A ciklusváltozó deklarációja és annak történő értékadása után a **do** kulcsszóval kezdeményezzük a ciklust. A ciklusváltozó értékének növeléséről szintén a ciklusmagban kell gondoskodnunk. A ciklus futása akkor ér véget, ha a ciklus végén lévő **while** kulcsszónál megadott feltétel teljesül.

```
do
    {
       utasítás(ok);
    }
while (kifejezés);
```

14. feladat:
Kérjünk be egy mobil telefonszámot, és csak akkor fogadjuk el, ha az valóban mobilszám! Írjuk ki, hogy a szám melyik szolgáltatóhoz tartozik! (Név: Mobilszamvizsgalo)

2.11.3. Előírt lépésszámú ciklus

A **for** esetében három dolgot kell megadnunk. Az első lesz a ciklusváltozó, melyet itt helyben deklarálunk és adunk kezdő értéket, a másodiknak azt a feltételt kell megadnunk, ameddig a ciklus futhat, végül arról kell rendelkeznünk, hogy a ciklusváltozó értéke miként növekedjen.

A for utasítás alakja:

```
for (kifejezés1; kifejezés2; kifejezés3)
    {
       utasítás (blokk)
    }
```

A *kifejezések* általában a következőket jelentik:

- kifejezés1: A ciklusváltozó kezdeti értékadása.
- kifejezés2: A lefutási feltétel(ek) megfogalmazása.
- kifejezés3: A ciklusváltozó növelése, vagy csökkentése.

Példa:

```
for ( i = 1; i <= 10; i++ )
{
    utasítás(ok);
}
```

15. feladat:
Állítsunk elő véletlenszerűen öt egész számot az [0,50]-ból, és írjuk ki a képernyőre egymás mellé 5 karakternyi helyet lefoglalva a számokat! (Név: Eloirtciklus1)

16. feladat:
Állítsunk elő véletlenszerűen 40 egész számot a [-100,100]-ból, és írjuk ki a képernyőre egymás mellé 6 karakternyi helyet lefoglalva, de egy sorban csak 8 szám legyen! A kiírás végén adjuk meg hány darab pozitív szám van! (Név: Eloirtciklus2)

2.12. Többirányú elágazás

A többirányú elágazást akkor használjuk, ha sok *if* utasításra lenne szükség, mivel egy változó értékétől függően több irányban is továbbhaladhatunk.

```
switch (kifejezés)
        {
        case érték :
            utasítás(ok);
            break;
        case érték :
            utasítás(ok);
            break;
        default :
            utasítás(ok);
            break;
        }
```

A **default** rész nem kötelező, csak akkor használjuk, ha van olyan tevékenység, amelynek alapesetben kell lefutnia, tehát ha a kifejezés egyik megadott értékkel se egyenlő.

14/a feladat:
Módosítsuk a 14. feladat megoldását, az *if* utasítások helyett alkalmazzunk *case* szerkezetet! (Név: Eloirtciklus1a)

17. feladat:
Írjunk oktatóprogramot, amely az összeadást és a kivonást gyakoroltatja! Az egész számokat véletlenszerűen állítsuk elő! A felhasználó csak -5 és +15 között képes a műveleteket elvégezni. Adjunk 10 feladatot, és a felhasználó minden helyes megoldása 1 pontot érjen! Osztályozzuk le a feladatsort, és az osztályzatot írjuk ki a képernyőre úgy, hogy 0-2-ig elégtelen, 3-4-ig elégséges, 5-6-ig közepes, 7-8-ig jó, e fölött pedig jeles legyen! (Név: Oktatoprogram)

Gyakorló feladatok:
1. Készítsen programot, ami számokat kér be mindaddig, amíg 0-át nem adunk meg!
2. Készítsen programot, ami számokat kér be addig, míg az utolsó két szám meg nem egyezik!
3. Készítsen számkitalálós programot! A gép véletlenszerűen találjon ki egy számot 1 és 100 között! A felhasználótól kérjen be tippeket, és mondja meg, hogy a gondolt szám a tipptől nagyobb-e vagy kisebb! A tippelés addig megy, amíg a felhasználó el nem találta a megadott számot.
4. Bővítse az előző feladatot úgy, hogy a szám eltalálásakor írja ki hány tippelésből sikerült eltalálni a számot!

5. Készítsünk programot, amely 15 darab *-ot ír ki a képernyőre!
6. Készítsünk programot, amely az első sorba kiír 15 *-ot, a másodikba 14 *-ot, …, a tizennegyedik sorba 2 *-ot, a tizenötödik sorba pedig 1*-ot! A csillagok kiírása a sor elején kezdődjön!
7. Készítse el az előző feladatot úgy, hogy a 2. sortól kezdődően a *-ok jobbra legyenek igazítva, az első sor utolsó csillagához!
8. Írassa ki a számokat 1-től 20-ig és mellé a négyzetüket is!
9. Írassa ki 99-től csökkenő sorrendben az összes pozitív, 3-al osztható egész számot!
10. Számítsa ki a gép 10 véletlen szám összegét, szorzatát, átlagát és írjuk ki a képernyőre!
11. Szimuláljon egy felhasználótól bekért számú kockadobást! A gép véletlenszerűen meghatározza a dobás értékét (1…6) és kiírja a dobás eredményét. A megfelelő szám elérése után a program készítsen statisztikát a dobások gyakoriságáról! Írjuk ki melyik szám hányszor fordult elő!
12. Készítsen programot, amely addig kér be számokat, amíg -30 és 40 közé nem esik egy szám!

2.13. A tömb

Egydimenziós tömbök (vektorok)

A tömb olyan adatszerkezet, amelynek elemei azonos típusúak lehetnek. A tömb elemeire a tömb nevével és az elem sorszámával hivatkozhatunk. A C# programnyelvben szögletes zárójelek közé kell írni az elem sorszámát és mindig 0-tól kezdődik az elemek sorszámozása. Pl.: tomb[2]

A tömb deklarációja:

típus [ ] tömbnév = new típus [elemszám];

Pl.: int [ ] a = new int [6];
A tömb elemei nem csak egész típusúak lehetnek, hanem *string*, *double*, *long* stb. is.

18. feladat:
Állítsunk elő 20 egész számot a [-50,20] intervallumból és írjuk ki a képernyőre 2 sorba! Írjuk ki a pozitív páros számokat! Ha ilyen nincs, akkor írjuk ki, hogy „A számok között nincs pozitív páros szám”! (Név: Tomb1)

Két- vagy többdimenziós tömbök

Kétdimenziós tömb (táblázatok) deklarációja:

int [,] egesztomb = new int [5, 6];

vagy

string [,] szoveg = new string [12, 8];

Háromdimenziós tömb deklarációja:

double [, ,] tomb3d = new double [5, 6, 9];

További dimenziók használata a fentiek alapján történik: vesszőkkel választjuk el a dimenziók számát. A többdimenziós tömbök feltöltését több egymásba ágyazott ciklussal végezzük el.

19. feladat:

Töltsünk fel egy kétdimenziós tömböt véletlenszerűen a [-1,1] intervallumból 2 tizedes pontosságú számokkal! A tömbnek 4 sora és 7 oszlopa legyen! A sorok és oszlopok végére írjuk ki az adott sor, ill. az oszlop összegét! (Név: Tomb2d)

20. feladat:
Állítsunk elő ötös lottószámokat véletlenszerűen mindaddig, amíg azt a felhasználó szeretné! (Név: Lotto)

20/a. feladat:
Módosítsa az előző feladatot úgy, hogy a program egyetlen futtatás során is biztosítsa, hogy nem lehet 2 azonos szám a kihúzottak között! (Név: Lotto2)

2.14. A goto utasítás

A *goto*-val bárhová ugorhatunk, csak meg kell adni a címke nevét a *goto* után, és el kell helyezni a program megfelelő pontjára a címkét (kettősponttal lezárva) oda, ahova a vezérlésnek ugrani kell.

21. feladat:
Kérjük be a másodfokú egyenlet együtthatóit, és számítsuk ki a gyököket 2 tizedes pontossággal! Ha a diszkrimináns negatív, írjuk ki a képernyőre: „Nincs megoldás”! Ha az a értéke 0, akkor figyelmeztessük a felhasználót, és kérjük be újra az adatot! Ha a felhasználó 3-nál többször hibázik, akkor lépjünk ki a programból! (Név: Masodfoku2)

Használhatunk a *goto* utasítás helyett **return** utasítást, mivel a *return* utasítással kiléphetünk egy utasításcsoportból úgy, hogy a *return* után álló utasítások nem lesznek végrehajtva. Ekkor természetesen a címkére sincs szükség.

21/a feladat:
Módosítsuk az előző feladat megoldását! Alkalmazzunk a *goto* utasítás helyett *return* utasítást! (Név: Masodfoku3)

Gyakorló feladatok:
1. Készítsen programot, amely egy 10-es számrendszerbeli számot átvált 2-es számrendszerbe!
2. Készítsen programot, amely egy 2-es számrendszerbeli számot átvált 10-es számrendszerbe!
3. Egy tanuló 10 kérdésből álló tesztlapot tölt ki. Minden kérdésre a választ az abc első 4 betűje (a, b, c, d) jelöli. Ismerjük a 10 kérdés helyes megoldását is betűkkel megadva. Készítsünk programot, amely bekéri a tanulótól a feladatonkénti megoldásokat, majd meghatározza, hány helyes válasza (pontja) volt a tanulónak! Minden helyes válasz 1 pontot ér!
4. Bővítse az előző programot azzal, hogy csak a lehetséges tippeket fogadhatja el a program a tanulótól! (A válasz csak a, vagy b, vagy c, vagy d lehet.)
5. Bővítse úgy a 3. feladatot, hogy több tanuló tölti ki a tesztet! A program kérdezze meg a tanulók számát és utána kérje be egyesével a tanulók válaszait!
6. A 3. feladatot bővítse tovább úgy, hogy az eredményt osztályozza is le a program! Az osztályzáshoz szükséges százalékokat megtalálja a korábbi gyakorló feladatoknál.
7. A 3. feladatot oldja meg úgy is, hogy a feladatok különböző pontszámot érjenek! Számítsa ki az összpontszámot, a maximális összpontszámot, és az előző feladat alapján végezze el az osztályozást!
8. Kérjünk be egy számot 1 és 100 között, majd írassuk ki betűvel!

9. Írassuk ki n-től m-ig a 7-tel osztható számokat, n nem feltétlenül kisebb, mint m!
10. Írassuk ki *While* ciklussal 0-100-ig a 3-mal osztható számokat!

3. PROGRAMOZÁSI TÉTELEK

A programozási tételek olyan algoritmusok, amelyeket gyakran használunk a programírás során. A matematikai, logikai, adatkezelési feladatok megoldásánál általánosan alkalmazhatók ezek a módszerek.
Ezek a műveletek általában a tömbökhöz kötődnek és a tömb elemeivel végezzük a műveleteket.

3.1. Összegzés

Többnyire egy tömb elemeinek összegzését értjük rajta. Általános képlete:

összeg = összeg + aktuális elem, vagy összeg += aktuális elem.

(Értelmezése: az új összeg = a régi összeg + az aktuális elem.)

22. feladat:
Állítsunk elő véletlenszerűen 8 egész számot a [0,100]-ból, és írjuk ki a képernyőre 1 sorba! Számoljuk és írjuk ki a számok átlagát 3 tizedes pontossággal! (Név: Osszegzes)

3.2. Megszámlálás

Megszámoljuk, hogy hány adott tulajdonságú elem van a tömbben. Az összegzésnél tanult módszert alkalmazzuk a számláló értékének növelésére, értéke egyesével nőhet csak.

23. feladat:
Állítsunk elő véletlenszerűen 12 egész számot a [-50,50]-ból, és írjuk ki a képernyőre 1 sorba! Számoljuk meg és írjuk ki, hogy hány negatív és hány pozitív szám van köztük! (Név: Megszamolas)

3.3. Eldöntés

Az algoritmus eldönti, hogy van-e (logikai változó) a tömbben adott tulajdonságú elem. Amint talál egyet, a ciklus leáll. Feltételezve, hogy a megoldáshoz ciklusra van szükség. Ha a ciklus azért állt le, mert túlléptünk a tömb utolsó, vizsgált elemén is, akkor nem volt benne keresett elem. A tétel eredményként egy logikai változót ad meg, amely igaz, ha a sorozatnak volt adott tulajdonságú eleme, és hamis, ha nem volt.
Példa: 15. feladat megoldásában egy adott számról kellett eldönteni, hogy az prímszám e vagy sem. A döntés eredményét pedig kiírtuk a képernyőre.

24. feladat:
Állítsunk elő véletlenszerűen 6 egész számot a [0,50]-ból, és írjuk ki a képernyőre 1 sorba! Írjuk ki, hogy van-e 3-al osztható szám! (Név: Eldontes)

3.4. Kiválasztás

Kiválasztás esetén az adatok közül kiválasztunk valamilyen szempont alapján egy olyan elemet, amely a megadott szempontnak megfelel. Csak akkor működik, ha biztosan van ilyen elem. Az algoritmus megadja, hogy a tömbben egy bizonyos elem hol (hányadik helyen) van.

25. feladat:

Állítsunk elő véletlenszerűen 10 egész számot a [-50,50]-ból, és írjuk ki a képernyőre 1 sorba! Írjuk ki az 1. pozitív szám értékét és indexét, ha van ilyen! Ha nincs, írjuk ki, hogy „Nincs pozitív szám”! (Név: Kivalasztas)

3.5. Maximum- (minimum) kiválasztás

Rendezetlen elemek közül kiválasztja a legnagyobbat, vagy a legkisebbet. Karakter és szöveg típusú változókra is érvényes. Az algoritmus lényege: a *for* ciklus megkezdése előtt kinevezzük az 1. elemet maximumnak (vagy minimumnak), majd belépünk a ciklusba, ahol i=2-től megyünk az utolsó elemig. A ciklusban a 2. elemtől kezdve minden elemet összehasonlítunk a maximummal, és ha az aktuális elem nagyobb, mint a maximum, akkor az aktuális elem lesz az új maximum.

26. feladat:
Állítsunk elő véletlenszerűen 5 egész számot a [0,500]-ból, és írjuk ki a képernyőre 1 sorba! Írjuk ki a legnagyobb számot a képernyőre! (Név: Maximumkivalasztas)

26/a. feladat:
Írjuk át a programot úgy, hogy a legkisebb számot válaszuk és írjuk ki! (Név: Minimumkivalasztas1)

27. feladat:
Kérjünk be 5 karaktert és írjuk ki az angol ABC szerinti elsőt! (Név: Minimumkivalasztas)

Gyakorló feladatok: (54. oldal)
1. Egy futóversenyre egyesületenként jelentkeztek a versenyzők. Az egyesületek megadják hány versenyzőt neveznek. Határozzuk meg az egyesületek számának ismeretében hány induló lesz a versenyen!
2. 12 hónapon keresztül spóroltunk. Minden hónapban változó összeget félreteszünk. Az összegek ismeretében adja meg, mennyi pénzünk van! Mekkora összegünk lesz, ha a megtakarításunkat 3 hónapra lekötjük, évi 10,5%-os kamat mellett?
3. Adott egy tekéző sorozata (melyik fordulóban hány fát ütött). Írjunk programot, amely meghatározza a versenyző összesített eredményét!
4. Egy horgászverseny adatait egy táblázatban tároljuk. A horgászok száma 6, a fogható halfajok száma 8. A táblázat 1 cellája M(i,j) azt jelenti, hogy az i horgász a j halfajból mennyit fogott.
   a) Írjunk programot, amely kiszámítja, hogy a horgászok összesen hány halat fogtak az egyes halfajokból!
   b) Írjunk programot, amely meghatározza, hogy az egyes horgászok hány halat fogtak összesen!
5. 12 napig megmértük a napi csapadékmennyiséget. Határozzuk meg mennyi eső esett összesen!
6. Egy labdarúgó bajnokságban tudjuk minden csapatról, hogy hányszor győzött és hányszor játszott döntetlent. Adjuk meg, melyik csapatnak hány pontja van (győzelem 3 pont, döntetlen 1 pont)!
7. Ismerjük 9 autó fogyasztását. Döntsük el, hogy minden autó 10 liter alatt fogyaszt-e!
8. Egy halgazdaság próbafogást végez. Minden hal súlyát és hosszát tárolják. A vizsgált halak száma 14. Az adatokat beírhatjuk a programba, bekérhetjük a billentyűzetről, vagy előállíthatjuk véletlenszerűen. Készítsen programot:
   a) Megadja a halak összsúlyát, és az átlagos hosszát!
   b) Megadja azon halak számát, amelyek 70 dkg-nál nagyobbak!

c) Megadja azon halak számát, amelyek 50 dkg-nál nagyobbak és 28 cm-nél hosszabbak!
d) Van-e 2 kg-nál nagyobb hal?
e) Van-e a kifogott halak között olyan, amelyik 40 dkg alatti?
f) Mekkora a mérete a legkisebb súlyú halnak?
g) Mekkora a súlya a legnagyobb méretű halnak?

9. Ismerjük 11 ember magasságát. Készítsen programot, amely megadja, hogy van-e olyan ember, aki alacsonyabb, mint a mögötte állók valamelyike!
10. Határozzuk meg, hogy egy adott hónap melyik évszakba esik!
11. Olvassunk be neveket addig, míg nem írtunk egymás után két azonos nevet!
12. A kosaras csapat nyilvántartásában tároljuk minden játékos nevét és magasságát. A játékosok száma 18. Határozzuk meg van-e a csapatnak 210 cm-nél magasabb játékosa. Ha van, írjuk ki a nevét (egy személy elég, de ha van több, akkor kiírathatjuk mindet is)!
13. Egy repülőgépről a szárazföldön kezdve 500 méterenként megmértük a felszín tengerszint feletti magasságát. A mérések száma 500. A méréssorozatot szárazföld felett fejeztük be. Ahol a mérés eredménye 0, ott tenger van, mindenhol máshol föld.
    a) Határozza meg, hogy van-e sziget a mérési helyen!
    b) Hány sziget található a mérési helyen?
    c) Mekkora a hossza az egyes szigeteknek?
    d) Milyen távolságba vagyunk a kiinduláskor a tengertől?
    e) Melyik a legrövidebb, ill. a leghosszabb sziget?
    f) Melyik a szárazföld legmagasabb pontja?
    g) Melyik a sziget legmagasabb pontja?
    h) Milyen távolságban vagyunk a mérés végeztekor a tengertől?
14. Nyelvvizsgán a 20 tanuló pontszámait ülési sorrendben jegyezték föl. Adja meg azoknak a vizsgázóknak a sorszámát, akik ugyanannyi pontot értek el, mint a szomszédjuk!
15. Állítson elő véletlenszerűen 28 számot [-10,10] tartományból.
    a) Állapítsa meg, hogy pozitív vagy negatív számból van-e több! A választ és a darabszámokat írja a képernyőre!
    b) Hányszor fordul elő a 7 szám?
    c) Van-e a számok között 0, ha igen hányadik volt?
    d) Adja meg a negatív számok átlagát!
    e) Melyik volt a pozitív számok közül a legkisebb?
    f) Melyik volt a negatív számok közül a legnagyobb?
16. A műkorcsolyában a versenyző teljesítményét 7 bíró pontozza. Az összpontszámot úgy számítjuk ki, hogy a leggyengébb és a legjobb pontszámot nem veszik figyelembe, az összes többit pedig átlagolják. Van 6 versenyzőnk és az eredményük. Írjunk programot amely:
    a) Meghatározza az egyes versenyzők összes pontszámát!
    b) Meghatározza a legnagyobb pontszámot, és azt, hogy hányadik versenyző kapta!
    c) Az összes pontszámok alapján meghatározza a versenyzők átlagát és megmondja hányan voltak átlag alatt!
    d) Meghatározza a legkisebb összpontszámot!
17. Adott a síkban 7 db pont a koordinátáival. Határozzuk meg, melyik esik legtávolabb az origótól!
18. Kérjünk be 0 végjelig számokat, írassuk ki a legnagyobbat, legkisebbet, és hogy hány szám volt összesen!

3.6. Kiválogatás

Kiválogatásról beszélünk, ha egy tömb elemei közül kiválogatunk egy vagy több feltételnek megfelelő elemet. Ezeket az elemeket kiírjuk a képernyőre, vagy elhelyezhetjük egy másik tömbben.

28. feladat:
Állítsunk elő véletlenszerűen 15 egész számot a [0,100]-ban, és írjuk ki a képernyőre 1 sorba! Írjuk ki és tegyük egy másik tömbbe az 5-el osztható számokat! (Név: Kivalogatas)

3.7. Szétválogatás

Szétválogatáskor egy tömb minden elemét elhelyezzük adott feltételtől függően más tömbökben. Ekkor az eredeti tömb elemszáma egyenlő szétválogatás után létrejött tömbök elemeinek összegével.

29. feladat:
Állítsunk elő véletlenszerűen 15 egész számot [0,100]-ban, és írjuk ki a képernyőre 1 sorba! Válogassuk szét 3 különböző tömbbe a kettővel osztható, a kettővel nem osztható, de hárommal osztható és az egyéb számokat, majd írjuk ki a tömbök tartalmát egymás alá! (Név: Szetvalogatas)

3.8. Rendezés

Rendezéskor az elemeket valamilyen tulajdonságuk alapján sorba rendezzük. Általában növekvő vagy csökkenő sorrendbe állítjuk az adatokat. Szám típusú adatokat nagyságuk szerint rendezzük, szöveg vagy karakter típusúakat pedig az angol ABC sorrend vagy ASCII kód szerint.

Rendezésnél több módszer közül választhatunk. Az elemek száma és rendezettségük foka dönti el melyik a jobb módszer, azaz melyik a gyorsabb. A két leggyakrabban használt módszer:

1. Minimum (maximum) kiválasztásos módszer,
2. Szomszédos elemek cseréjén alapuló módszer.

3.8. Minimum (maximum) kiválasztásos módszer

Két egymásba ágyazott *for* ciklus segítségével rendezzük az elemeket. A külső ciklus 1-től az utolsó előtti elemig, a belső az aktuális elem+1 indexétől az utolsóig megy. A ciklusban összehasonlítja az elemeket, és ha a sorrend rossz, akkor megcseréli őket.

```
Ciklus i := 1-től n-1-ig
    Ciklus j := i + 1-től n-ig
        Ha a[i] > a[j] akkor csere
```

A csere (segédváltozó):

```
s := a[i]
a[i] := a[j]
a[j] := s
```

30. feladat:
Állítsunk elő véletlenszerűen 5 egész számot a [0,100]-ból, és írjuk ki a képernyőre 1 sorba! Rendezzük növekvő sorrendbe a számokat, és írjuk ki a képernyőre! (Név: Rendezes)

31. feladat:
Kérjünk be 5 betűket (vagy nevet) és rendezzük ABC szerint sorrendbe! (Név: Rendezes2)

3.8.2. Szomszédos elemek cseréjén alapuló módszer

A módszer lényege, hogy egy hátul tesztelő ciklusban addig végezzük a szomszédos elemek cseréjét, amíg minden elem a helyére nem kerül.

```
Ciklus
   Ciklus i := 1-től n-1-ig
        Ha a[i] > a[i+1] akkor csere
amíg volt csere
```

A hátul tesztelő ciklusból csak akkor léphetünk ki, ha már nem volt szükség cserére, tehát az elemek sorrendje megfelelő. Természetesen a relációs jel megfordításával csökkenő sorrendbe állítatjuk az elemeket.

32. feladat:
Módosítsuk a 30. feladat megoldását ezzel a módszerrel! Állítsunk elő véletlenszerűen 5 egész számot a [0,100]-ból, és írjuk ki a képernyőre 1 sorba! Rendezzük növekvő sorrendbe a számokat és írjuk ki a képernyőre! (Név: Rendezes3)

Gyakorló feladatok: (64. oldal)

1. Adott egy 10 elemű számsorozat. Rendezzük a számokat növekvő sorrendbe, majd kérjünk be egyenként további számokat, és döntsük el róluk megtalálhatóak-e az eredeti 10 szám között! Amennyiben van ilyen szám, mondja meg hogy hányadik, különben írja ki, hogy nincs. Az új számokat addig kérje be, míg egymás után kétszer 0-át adunk meg.
2. Az iskolában egyéni és összetett versenyt tartottak. A versenyben összesen 20 tanuló vett részt. A versenyek száma 8. Ismerjük versenyenként a diák nevét és pontszámát. Összetett versenybe csak az indulhat, a ki az összes egyéni versenyben indult, és elérte a versenyenként megadott minimális pontszámot.
   a) Adjuk meg az egyéni versenyek rangsorát!
   b) Adjuk meg az összetett versenyben értékelendő tanulók számát!
   c) Adjuk meg az összetett versenyben értékelendő tanulók nevét!
   d) Adjuk meg az összetett verseny rangsorát!
   e) Adjuk meg versenyenként a tanulók névsorát!
   f) Adjuk meg azok nevét, akik csak 1 versenyen indultak!
   g) Adjuk meg azon tanulók névsorát, akik valamilyen versenyen indultak és elérték a versenyenkénti minimális pontszámot!
   h) Adjuk meg azon tanulók névsorát, akik valamilyen versenyen indultak és nem érték el a versenyenkénti minimális pontszámot!
   i) Adjuk meg a versenyenkénti első 3 helyezettet!
   j) Döntsük el, hogy az összetett verseny győztese minden számban győztes volt-e!
   k) Döntsük el, hogy volt-e valamelyik versenyben holtverseny!
   l) Döntsük el, hogy az összetett verseny győztese holtversenyben győzött-e!
   m) Állapítsuk meg, hogy volt-e olyan tanuló, aki minden versenyen elindult, de nem érte el a minimális pontszámot!
3. Írjunk programot, amely mértékegységeket kezel! A program a standard bemenetről sorokat olvas be, amelyekben tömegmennyiségek vannak megadva különböző mértékegységben. A program írja a szabványos outputra ezek összegét grammban, majd a következő sorba ugyanezt tonna, mázsa, kilogramm, dekagramm, gramm formában! Mindig a lehető legnagyobb mértékegységekkel legyenek a mennyiségek kifejezve! A bemeneti sorok mindig egy nem negatív egész számból (mennyiség) és egy karakterből (t – tonna, m – mázsa, k – kilogramm, d – dekagramm, g – gramm) állnak, ezeket egy szóköz választja el egymástól. A bemenet végét egy „0 0\n” tartalmú sor jelzi. A

bemenetről érkező sorok közül több is tartalmazhatja ugyanazt a mértékegységet, és a mértékegységek sorrendje is tetszőleges. A program azokat a mértékegységeket is írja ki amelyekhez 0 mennyisé tartozik! A mértékegység nem maradhat le: az összegnél egy szóközzel elválasztva a „gramm” szót, a felbontásnál pedig szintén egy szóközzel elválasztva a mértékegység betűjelét is ki kell írni!

Példa:	**bemenet**	**kimenet**
	1 k	3000 gramm
	2 k	0 t
	0 0	0 m
		3 k
		0 d
		0 g

3.11. Műveletek stringekkel

A *string* típusú változók néhány tulajdonságát már megismertük a mobilszám-ellenőrző programban. A C#-ban, mint objektumorientált nyelvben a *string* nem csak típus, hanem *osztály* is. Az egy típusba tartozó változókat, a hozzájuk tartozó műveleteket és operátorokat **osztály**nak nevezzük.

A string osztályhoz tartozó műveletek:

Szöveghossz meghatározása:

A *szam.Length* megadja a *szam* változó string hosszát, a *szam[0].ToString()* pedig a szöveg 1. karakterét.

33/a. feladat:
Írassuk ki a képernyőre, hogy a „Karaktersorozat” szót karakterenként, majd alá írassuk ki a szó 7. karakterét és hogy hány karakterből áll a szó! (Név: Szoveghossz)

Módosítsuk a feladatot, hogy a felhasználó adhassa meg a kiírandó szöveget!

Összeadás (Összefűzés, összekapcsolás):

Az összeadás művelet értelmezhető *string*eken, de eredménye nem azonos a számtípusoknál megismerttel. Itt a szövegeket egymás után fűzi a + operátor.
Pl.: ”objektum”+”orientált”=”objektumorientált”, számokból álló stringek esetén ”77”+”23”=”7723”.

33/b. feladat:
Kérjünk be két szöveget és írjuk ki a képernyőre a két szót egymás mellé! (Név: Szovegrendezes)

Összehasonlítás:

A *string*ek egy karakterére értelmezhető a < és a > jel ebben a formában: szoveg[4] > szoveg[1]. A <, > relációs jeleket a *CompareTo* segítségével tudjuk a teljes *string*re alkalmazni. Viszont csak a teljes *string*re értelmezhetők a == és a != relációs jelek.

*Két string összehasonlítása*kor 3 eset lehetséges:

1. Ha s1 < s2, vagyis s1 előbb van az ABC-ben, akkor az *s1.CompareTo(s2)=-1*
2. Ha s1 > s2, vagyis s1 később van az ABC-ben, akkor az *s1.CompareTo(s2)=1*
3. Ha s1 = s2, akkor az *s1.CompareTo(s2)=0*

Két *string* egyenlő akkor és csak akkor, ha hosszuk egyenlő, és minden karaktere rendre egyenlő.

Két *karakter* egyenlő, ha mindkét karakternek egyforma a kódtábla szerinti kódja.

Pl.: Az „almafa” és „alma” stringek nem egyenlők (nem egyforma hosszúak), „almafa” és „Almafa” sem egyenlők (a nullás pozíción kis „a”, illetve nagy „A” szerepel).

33. feladat:
Kérjünk be két szöveget és írjuk ki a képernyőre az ABC-ben előbb lévőt, majd alá az ABC-ben hátrébb lévőt! (Név: Szovegrendezes)

Levágás:

A levágás a **Trim()**, a **TrimStart()** és a **TrimEnd()** segítségével jön létre. Ha a zárójelek közé nem írunk paramétereket, akkor a **Trim()** levágja a szöveg elejéről és végéről, a **TrimStart()** csak az elejéről, a **TrimEnd()** csak a végéről a szóközöket.

Pl.: Ha a *szoveg* egy string típusú változó, akkor a *szoveg.Trim('b');* levágja a *szoveg* elejéről és végéről az összes *'b'* karaktert. A *szoveg.TrimStart('b');* csak az elejéről, a *szoveg.TrimEnd('b');* csak a végéről vágja le az összes 'b'-t.

Átalakítás (kisbetű – nagybetű):

Átalakítani nagybetűkké a **ToUpper(),** kisbetűkké a **ToLower()** segítségével lehet.

Pl.: Ha sz = ”félév”, akkor az *sz.ToUpper();* után az sz = ”FÉLÉV” lesz.

Másolás:

Egy szövegből tetszőleges részt másolhatunk a **Substring(kezd,db)** segítségével, ahol a *kezd* és a *db* egész típusú változók és a stringből a *kezd* karaktertől *db* darab karaktert másolhatunk ki. Ha zárójelek között csak egy paraméter van, akkor a *kezd* értékétől egész a szöveg végéig másolhatunk ki részsztringeket. A *string* karaktereinek számozását 0-tól kezdjük, mint a tömböknél.

Pl.: Ha sz = ”Paróka”, akkor *sz.Substring(2);* = ”róka”.

Keresés:

Az **IndexOf()** függvénnyel megadhatjuk, hogy az adott karakter vagy karaktersorozat szerepel-e a stringben, és ha igen, hol. Ha benne van, akkor a függvény értéke 0 vagy pozitív, ha nincs benne, akkor -1 lesz. Ha a függvény értéke nem -1, akkor a kapott érték (egész szám lesz) azt mutatja meg, hogy hányadik karaktertől találjuk meg az adott karaktersorozatot a szövegben.

Pl.: sz ="almafa"; *sz.IndexOf("fa");* = 4. Ha a szoveg = ”M á t é s z a l k a”; és a szo = ”é s z”; akkor a *szoveg.IndexOf(szo)*; = 3 (a *szoveg* szovegben a *szo* szó a 4. pozíciótól található meg).

34. feladat:
Kérjünk be egy szöveget és egy karaktert majd írjuk ki, hogy a szövegben hányszor fordul elő a megadott karakter! (Név: Karakterszamlalas)

35. feladat:
Kérjünk be egy szöveget és egy szót, majd írjuk ki, hogy a szövegben hányszor fordul elő a megadott szó! (Név: Szoszamlalas)

36. feladat:
Kérjünk be egy szöveget, és írjuk ki visszafelé! A fordított szöveg csak nagybetűből álljon! (Név: Szovegmegfordito)

37. feladat:
Kérjük be egy személy nevét, amely vagy 2, vagy 3 részből áll! Írjuk ki a vezetéknevet, majd alá a keresztnevet és ez alá a második keresztnevet, ha van! (Név: Nevkiiras)

Gyakorló feladatok:

Mi lesz az alábbi szövegfüggvényekkel végzett műveletek eredménye?

1. Ha sz = ”süveges”; , akkor sz.Trim(’s’); = …….;
2. Ha sz = ”Messi a messiás”; , akkor sz.IndexOf(”si”); = …….;
3. Ha sz = ”Paróka”; , akkor sz.Substring(2, 2); = …….;
4. Ha sz = ”Ouagadougou”; , akkor sz.Substring(8); = …….;

40. feladat:

Jók-e az alábbi paraméterlisták:

Aktuális paraméterlista	Formális paraméterlista
int a = 0, f = 0; (a, f)	(int d, int b)
int x = 0; double y = 0; (x, y)	(int x, int y)
string a = ” ”; int y = 0; (a, y)	(string nev, int y)
char k = ’ ’; bool v = true; (k, v)	(string k, bool v)
double x = 0; byte y = 0; (x, y)	(int x, byte y)
string x = ” ”; double y = 0; (x, y)	(char x, double y)
int x = 0; string szo = 0; (x, szo)	(int a, string szoveg)
bool x = 0; double y = 0; (x, y)	(bool x, double z)

44. feladat:

Állítsunk elő véletlenszerűen 5 egész számot [0,100]-ból, majd írjuk ki a számokat a képernyőre és a háttértárolóra a program BIN/DEBUG könyvtárba a *szamok.txt* nevű fájlba! (Név: Fajlbair)

45. feladat:

Hozzunk létre egy szövegfájlt az alábbi adatokkal, és mentsük a programunk BIN/DEBUG könyvtárba *fajl.txt* (UTF-8 kódolás) néven!

Barna Barbara
Kiss István
Nagy Róbert
Pót Csaba
Nagy Evelin
Kovács Virág

Írjunk programot, amely beolvassa az adatokat egy 6 elemű szövegtömbbe soronként és kiírja a képernyőre! (Név: Beolvas)

46. feladat:

Kérjünk be 6 nevet és a hozzá tartozó átlagot, majd írjuk ki a háttértárolóra a program BIN/DEBUG könyvtárába *nevatlag.txt* néven! A nevek két részből álljanak, az átlagok 2 tized pontossággal legyenek megadva! A név és a hozzátartozó átlag egy sorban legyen! A sorokat tömbben tároljuk! (Név: Nevatlag)

47. feladat:

Másoljuk az előző feladatban létrehozott *nevatlag.txt* fájlt a programunk BIN/DEBUG könyvtárba! Olvassuk be az adatokat, és írjuk ki a képernyőre! Számoljuk ki az átlagok átlagát és írjuk ki a képernyőre a következő szöveg után: ”A csoport átlaga: ”. (Név: Nevatlagbeolvas)

48. feladat:

Másoljuk a 45-ös feladatban létrehozott *nevatlag.txt* fájlt a programunk BIN/DEBUG könyvtárába! A megfelelő adatokhoz írjunk még újabbakat! Az adatbevitel ENTER végjelig tartson! (Név: Hozzafuz)

49. feladat:

Hozzunk létre egy adattáblát, amelynek max. 500 sora lehet, az oszlopok neve pedig *név, osztály és átlag* legyen! Kérjük be az adatsorok számát és ennek megfelelően az adatokat, majd írjuk ki a háttértárolóra a programunk BIN/DEBUG könyvtárába *adatok.txt* néven! (Név: Rekord)

AJÁNLOTT IRODALOM:

Czygléczky Gábor: Bevezetés a C# programozásba (Petrik TISZK TÁMOP projekt tananyagfejlesztés)
http://itszp.hu/SitePages/foglalkozasok.aspx?elagid=11&foglalkid=16

Reiter István: C# programozás lépésről lépésre
http://reiteristvan.wordpress.com/2012/10/17/c-programozas-lepesrol-lepesre-letoltheto/

Illés Zoltán: Programozás C# nyelven
http://compalg.inf.elte.hu/~tony/KedvencKonyvek/InfoKonyvtar/09-Programozas%20C-sharp%20nyelven/Programozas-Csharp-nyelven-Konyv.pdf

http://tudasbazis.sulinet.hu/hu/informatika/informatika/informatika-9-12-evfolyam/programozas-c-kornyezetben/

Jónás Katalin: A C# nyelv és a programozás alapjai