Programozási tételek segédlet

Feladatbank • Megoldások • Hibavadászat • 45 perces ZH • C# kódsablonok

Készült: 2026-01-21

Tartalomjegyzék

Bevezetés

A programozási tételek tipikus tömbös/sorozatos feladatokra adnak sablon megoldásokat. Minden tételhez pszeudokód, C# sablon, 5 feladat megoldásvázlattal, házi ötletek és gyakori hibák tartoznak.

Konvenció: Indexelés 0-tól. A tömb elemszáma n. feltétel(x) logikai predikátum.

Elemi programozási tételek

1. Sorozatszámítás (általános)

Általános redukció: S0-ról S=f(S,elem) iteráció.

Pszeudokód (sablon)

```
S = S0

ciklus i = 0 .. n-1

    S = f(S, t[i])

ciklus vége

ki S
```

C# kódsablon

```
static T Sorozatszamitas<T>(IEnumerable<T> sor, T s0, Func<T,T,T> f)

{

    T s = s0;

    foreach (var x in sor) s = f(s, x);

    return s;

}
```

Gyakori hibák: Kezdőérték rossz • Típuskeverés • Nem asszociatív f esetén óvatosan

Feladatok (5 db) – megoldás vázlattal

1. Szorzat.

Megoldás vázlat: S0=1, f=szorzás.

C# tipp: Sorozatszamitas(t,1,(s,x)=>s*x)

2. Van-e páros?

Megoldás vázlat: S0=false, f=||.

C# tipp: Sorozatszamitas(t,false,(s,x)=>s || x%2==0)

3. Maximum.

Megoldás vázlat: S0=t[0], f=Math.Max.

C# tipp: Sorozatszamitas(t,t[0],Math.Max)

4. Negatívak száma.

Megoldás vázlat: S0=0, f=s+(x<0?1:0).

C# tipp: Sorozatszamitas(t,0,(s,x)=>s+(x<0?1:0))

5. Szavak összefűzése.

Megoldás vázlat: S0="", f konkatenáció.

C# tipp: Sorozatszamitas(words,"",(s,x)=>s==""?x:s+" "+x)

Házi feladat ötletek

- Mutasd meg: összegzés sorozatszámítás esete.
- Aggregate LINQ nélkül.
- Rekordok mezőinek összegzése.

. Sorozatszámítás (Összegzés)

Egy sorozat elemeiből aggregált értéket számolunk (összeg).

Pszeudokód (sablon)

```
osszeg = 0

ciklus i = 0 .. n-1

    osszeg = osszeg + t[i]

ciklus vége

ki osszeg
```

C# kódsablon

```
static int Osszegzes(int[] t)

{

    int osszeg = 0;

    for (int i = 0; i < t.Length; i++) osszeg += t[i];

    return osszeg;

}
```

Gyakori hibák: Kezdőérték rossz • Off-by-one (i<=Length) • Overflow nagy összegeknél

Feladatok (5 db) – megoldás vázlattal

1. Tömb elemeinek összege.

Megoldás vázlat: Sablon közvetlenül.

C# tipp: Osszegzes(t)

2. Páros elemek összege.

Megoldás vázlat: if (x%2==0) add.

C# tipp: if(t[i]%2==0) osszeg += t[i];

3. Pozitív elemek összege.

Megoldás vázlat: if (x>0) add.

C# tipp: if(t[i]>0) osszeg += t[i];

4. Abszolútértékek összege.

Megoldás vázlat: add Abs(x).

C# tipp: osszeg += Math.Abs(t[i]);

5. 0-k kihagyásával összegezni.

Megoldás vázlat: if (x!=0) add.

C# tipp: if(t[i]!=0) osszeg += t[i];

Házi feladat ötletek

• Átlag: összeg/n (double).

• Σ|t[i]-t[i-1]|.

• EOF-ig beolvasás és összeg.

2. Megszámlálás

Megszámoljuk, hány elem felel meg a feltételnek.

Pszeudokód (sablon)

```
db = 0

ciklus i = 0 .. n-1

    ha feltétel(t[i]) akkor db = db + 1

ciklus vége

ki db
```

C# kódsablon

```
static int Megszamlalas(int[] t, Func<int,bool> feltetel)

{

    int db = 0;

    for (int i = 0; i < t.Length; i++) if (feltetel(t[i])) db++;

    return db;

}
```

Gyakori hibák: db nincs 0-ra inicializálva • Rossz feltétel • Rossz bejárási határ

Feladatok (5 db) – megoldás vázlattal

1. Negatív számok száma.

Megoldás vázlat: db++ ha x<0.

C# tipp: Megszamlalas(t,x=>x<0)

2. Páros számok száma.

Megoldás vázlat: db++ ha x%2==0.

C# tipp: Megszamlalas(t,x=>x%2==0)

3. [5,10] intervallumban lévők száma.

Megoldás vázlat: 5<=x && x<=10.

C# tipp: Megszamlalas(t,x=>5<=x && x<=10)

4. Maximum előfordulásainak száma.

Megoldás vázlat: max után count x==max.

C# tipp: var m=Maximum(t); Megszamlalas(t,x=>x==m)

5. 3-mal vagy 5-tel oszthatók száma.

Megoldás vázlat: x%3==0 || x%5==0.

C# tipp: Megszamlalas(t,x=>x%3==0||x%5==0)

Házi feladat ötletek

- Különböző elemek száma (HashSet).
- Lokális maximumok száma.
- Feltételes megszámlálás fájlból olvasott adatokon.

3. Eldöntés

Eldöntjük, létezik-e feltételt kielégítő elem.

Pszeudokód (sablon)

```
i = 0

ciklus amíg i < n és nem feltétel(t[i])

    i = i + 1

ciklus vége

van = (i < n)

ki van
```

C# kódsablon

```
static bool Eldontes(int[] t, Func<int,bool> feltetel)

{

    int i = 0;

    while (i < t.Length && !feltetel(t[i])) i++;

    return i < t.Length;

}
```

Gyakori hibák: while feltétel sorrend hibás (OutOfRange) • Nincs korai kilépés • van értékét felülírja

Feladatok (5 db) – megoldás vázlattal

1. Van-e 0 a tömbben?

Megoldás vázlat: x==0.

C# tipp: Eldontes(t,x=>x==0)

2. Van-e 100-nál nagyobb?

Megoldás vázlat: x>100.

C# tipp: Eldontes(t,x=>x>100)

3. Van-e negatív páros?

Megoldás vázlat: x<0 && x%2==0.

C# tipp: Eldontes(t,x=>x<0 && x%2==0)

4. Van-e két azonos szomszéd?

Megoldás vázlat: keress i: t[i]==t[i-1].

C# tipp: for i=1.. if(t[i]==t[i-1]) return true

5. Minden elem pozitív?

Megoldás vázlat: negált: van-e x<=0?

C# tipp: !Eldontes(t,x=>x<=0)

Házi feladat ötletek

- Palindrom vizsgálat.
- Prím létezés.
- Rendezettség vizsgálat átvezetése.

4. Kiválasztás

Megadjuk a feltételt kielégítő elem indexét, ha garantáltan létezik ilyen elem.

Pszeudokód (sablon)

```
i = 0

ciklus amíg nem feltétel(t[i])

    i = i + 1

ciklus vége

ki i
```

C# kódsablon

```
static int Kivalasztas(int[] t, Func<int,bool> feltetel)

{

    int i = 0;

    while (!feltetel(t[i])) i++;

    return i; // előfeltétel: létezik

}
```

Gyakori hibák: Garancia nélkül kifut • Index/sorszám keverés • Végtelen ciklus rossz feltétellel

Feladatok (5 db) – megoldás vázlattal

1. Első negatív indexe (van).

Megoldás vázlat: x<0.

C# tipp: Kivalasztas(t,x=>x<0)

2. Első páros indexe (van).

Megoldás vázlat: x%2==0.

C# tipp: Kivalasztas(t,x=>x%2==0)

3. Első >10 indexe (van).

Megoldás vázlat: x>10.

C# tipp: Kivalasztas(t,x=>x>10)

4. Maximum indexe.

Megoldás vázlat: MaxIndex.

C# tipp: MaxIndex(t)

5. Első i, ahol t[i]!=i (van).

Megoldás vázlat: while(t[i]==i) i++;

C# tipp: int i=0; while(t[i]==i) i++;

Házi feladat ötletek

• Első 7-tel osztható (garancia).

• Első nem üres string (garancia).

• Első pozitív (garancia).

5. Keresés (lineáris)

Megkeressük a feltételt kielégítő első elemet; ha nincs, -1.

Pszeudokód (sablon)

```
i = 0

ciklus amíg i < n és nem feltétel(t[i])

    i = i + 1

ciklus vége

ha i < n akkor ki i különben ki -1
```

C# kódsablon

```
static int Kereses(int[] t, Func<int,bool> feltetel)

{

    for (int i = 0; i < t.Length; i++) if (feltetel(t[i])) return i;

    return -1;

}
```

Gyakori hibák: Nem return-öl azonnal • -1 jelzés hiánya • Üres tömb kezelése hiányzik

Feladatok (5 db) – megoldás vázlattal

1. Kulcs első indexe.

Megoldás vázlat: x==kulcs.

C# tipp: Kereses(t,x=>x==kulcs)

2. Első páros indexe.

Megoldás vázlat: x%2==0.

C# tipp: Kereses(t,x=>x%2==0)

3. Első |x|>50 indexe.

Megoldás vázlat: Abs(x)>50.

C# tipp: Kereses(t,x=>Math.Abs(x)>50)

4. Első lokális minimum.

Megoldás vázlat: t[i-1]>t[i]<t[i+1].

C# tipp: for (int i=1;i<Len-1;i++) ...

5. Első string, ami tartalmaz "a".

Megoldás vázlat: s.Contains("a").

C# tipp: Array.FindIndex(arr,s=>s.Contains("a"))

Házi feladat ötletek

• Utolsó előfordulás keresése.

• Első prím keresése.

• Első átlag feletti elem.

6. Maximum kiválasztás

A legnagyobb elem (és/vagy index) meghatározása.

Pszeudokód (sablon)

```
mi = 0

ciklus i = 1 .. n-1

    ha t[i] > t[mi] akkor mi = i

ciklus vége

ki mi
```

C# kódsablon

```
static int MaxIndex(int[] t)

{

    int mi = 0;

    for (int i = 1; i < t.Length; i++) if (t[i] > t[mi]) mi = i;

    return mi;

}
```

Gyakori hibák: Üres tömb (előfeltétel) • max kezdőérték 0-ra hibás • Index/érték keverése

Feladatok (5 db) – megoldás vázlattal

1. Maximum érték.

Megoldás vázlat: max változó frissítése.

C# tipp: var m = t[MaxIndex(t)];

2. Maximum index.

Megoldás vázlat: MaxIndex sablon.

C# tipp: MaxIndex(t)

3. Max abszolútérték szerint.

Megoldás vázlat: Abs összehasonlítás.

C# tipp: if (Abs(t[i]) > Abs(t[mi])) mi=i;

4. Max páros indexeken.

Megoldás vázlat: i+=2.

C# tipp: for(i=0;i<Len;i+=2)

5. Második maximum.

Megoldás vázlat: max1/max2.

C# tipp: tarts két változót

Házi feladat ötletek

- Max+min egy menetben.
- Max gyakoriság.
- Rekord max mező alapján.

. Minimum kiválasztás

A legkisebb elem (és/vagy index) meghatározása.

Pszeudokód (sablon)

```
mi = 0

ciklus i = 1 .. n-1

    ha t[i] < t[mi] akkor mi = i

ciklus vége

ki mi
```

C# kódsablon

```
static int MinIndex(int[] t)

{

    int mi = 0;

    for (int i = 1; i < t.Length; i++) if (t[i] < t[mi]) mi = i;

    return mi;

}
```

Gyakori hibák: Üres tömb • min kezdőérték 0-ra hibás • Index frissítés hiánya

Feladatok (5 db) – megoldás vázlattal

1. Minimum érték.

Megoldás vázlat: min frissítése.

C# tipp: var m = t[MinIndex(t)];

2. Minimum index.

Megoldás vázlat: MinIndex sablon.

C# tipp: MinIndex(t)

3. Legkisebb pozitív (ha van).

Megoldás vázlat: best=MaxValue; if(x>0&&x<best).

C# tipp: int best=int.MaxValue;

4. Targethez legközelebbi.

Megoldás vázlat: Min |x-target|.

C# tipp: if (Abs(x-target) < best)

5. Minimum >5 között (van).

Megoldás vázlat: először válassz ki egy >5-öt.

C# tipp: find first >5

Házi feladat ötletek

• Min és max egy menetben.

• Min gyakoriság.

• Rekord min mező alapján.

Összetett programozási tételek

7. Másolás

Elemek átmásolása egy új tömbbe transzformációval.

Pszeudokód (sablon)

```
ciklus i = 0 .. n-1

    b[i] = művelet(a[i])

ciklus vége
```

C# kódsablon

```
static int[] Masolas(int[] a, Func<int,int> muvelet)

{

    int[] b = new int[a.Length];

    for (int i = 0; i < a.Length; i++) b[i] = muvelet(a[i]);

    return b;

}
```

Gyakori hibák: Cél tömb nincs lefoglalva • Rossz határ • Overflow műveletben

Feladatok (5 db) – megoldás vázlattal

1. Másolat készítése.

Megoldás vázlat: muvelet(x)=x.

C# tipp: Masolas(a,x=>x)

2. Duplázás.

Megoldás vázlat: muvelet(x)=2*x.

C# tipp: Masolas(a,x=>2*x)

3. Abszolútérték.

Megoldás vázlat: muvelet(x)=Abs(x).

C# tipp: Masolas(a,Math.Abs)

4. Negatívak nullázása.

Megoldás vázlat: muvelet(x)=max(0,x).

C# tipp: Masolas(a,x=>Math.Max(0,x))

5. Index hozzáadása: b[i]=a[i]+i.

Megoldás vázlat: muvelet indexfüggő.

C# tipp: for i: b[i]=a[i]+i

Házi feladat ötletek

• Celsius→Fahrenheit (double).

• Whitespace→_ (string).

• Négyzet (long).

8. Kiválogatás

A feltételt teljesítő elemeket új sorozatba gyűjtjük.

Pszeudokód (sablon)

```
k = 0

ciklus i = 0 .. n-1

    ha feltétel(a[i]) akkor

        b[k] = a[i]

        k = k + 1

ciklus vége
```

C# kódsablon

```
static int[] Kivalogatas(int[] a, Func<int,bool> feltetel)

{

    var lista = new List<int>();

    foreach (var x in a) if (feltetel(x)) lista.Add(x);

    return lista.ToArray();

}
```

Gyakori hibák: k++ kimarad • Fix b mérete kicsi • Határérték téves

Feladatok (5 db) – megoldás vázlattal

1. <5 elemek kiválogatása.

Megoldás vázlat: x<5.

C# tipp: Kivalogatas(a,x=>x<5)

2. Páros elemek.

Megoldás vázlat: x%2==0.

C# tipp: Kivalogatas(a,x=>x%2==0)

3. Nem nullák.

Megoldás vázlat: x!=0.

C# tipp: Kivalogatas(a,x=>x!=0)

4. [A,B] intervallumbeliek.

Megoldás vázlat: A<=x && x<=B.

C# tipp: Kivalogatas(a,x=>A<=x && x<=B)

5. Prímek kiválogatása.

Megoldás vázlat: IsPrime(x).

C# tipp: Kivalogatas(a,IsPrime)

Házi feladat ötletek

- Kiválogatás + összegzés.
- Kiválogatás + rendezés.
- Duplikátumok kiszűrése.

9. Szétválogatás

A sorozat elemeit két csoportra bontjuk feltétel szerint.

Pszeudokód (sablon)

```
ig = üres

ha = üres

ciklus x az a elemein

    ha feltétel(x) akkor ig-be, különben ha-ba
```

C# kódsablon

```
static (int[] igazak, int[] hamisak) Szetvalogatas(int[] a, Func<int,bool>
feltetel)

{

    var ig = new List<int>();

    var ha = new List<int>();

    foreach (var x in a) if (feltetel(x)) ig.Add(x); else ha.Add(x);

    return (ig.ToArray(), ha.ToArray());

}
```

Gyakori hibák: p/q keverése • else ág hiánya • stabilitás félreértése

Feladatok (5 db) – megoldás vázlattal

1. Páros/páratlan szétválasztása.

Megoldás vázlat: feltétel: páros.

C# tipp: Szetvalogatas(a,x=>x%2==0)

2. Pozitív/nem pozitív.

Megoldás vázlat: x>0.

C# tipp: Szetvalogatas(a,x=>x>0)

3. <10 és >=10.

Megoldás vázlat: x<10.

C# tipp: Szetvalogatas(a,x=>x<10)

4. Üres/nem üres stringek.

Megoldás vázlat: IsNullOrEmpty.

C# tipp: SzetvalogatasS(a,string.IsNullOrEmpty)

5. Prím/nem prím.

Megoldás vázlat: IsPrime.

C# tipp: Szetvalogatas(a,IsPrime)

Házi feladat ötletek

• 3 csoportba bontás (<0,=0,>0).

• Csoportonkénti statisztika.

• Szétválogatás után rendezés.

10. Metszet

Két sorozat közös elemei (halmazként).

Pszeudokód (sablon)

```
setB = B halmaz

C = üres

A elemein: ha benne van setB-ben, add
```

C# kódsablon

```
static int[] Metszet(int[] a, int[] b)

{

    var setB = new HashSet<int>(b);

    var c = new List<int>();

    foreach (var x in a) if (setB.Contains(x)) c.Add(x);

    return c.Distinct().ToArray();

}
```

Gyakori hibák: Duplikátum-kezelés tisztázatlan • O(n*m) kerülendő • Sorrend kérdés

Feladatok (5 db) – megoldás vázlattal

1. Metszet int tömbökre.

Megoldás vázlat: HashSet+Distinct.

C# tipp: Metszet(a,b)

2. Van-e közös elem?

Megoldás vázlat: Any + Contains.

C# tipp: a.Any(x=>setB.Contains(x))

3. Páros közös elemek.

Megoldás vázlat: Contains && x%2==0.

C# tipp: if(setB.Contains(x)&&x%2==0)

4. Metszet stringekre case-insensitive.

Megoldás vázlat: OrdinalIgnoreCase.

C# tipp: HashSet<string>(..., OrdinalIgnoreCase)

5. Rendezett metszet O(n+m).

Megoldás vázlat: Kétmutatós.

C# tipp: while(i<n&&j<m)

Házi feladat ötletek

• Multihalmaz metszet.

• Eredmény rendezése.

• Bináris kereséssel rendezett B esetén.

11. Unió

Két sorozat egyesítése (stabil unió).

Pszeudokód (sablon)

```
seen = üres

C = üres

A elemein: ha új, add

B elemein: ha új, add
```

C# kódsablon

```
static int[] UnioStabil(int[] a, int[] b)

{

    var seen = new HashSet<int>();

    var c = new List<int>();

    foreach (var x in a) if (seen.Add(x)) c.Add(x);

    foreach (var x in b) if (seen.Add(x)) c.Add(x);

    return c.ToArray();

}
```

Gyakori hibák: HashSet sorrendje nem garantált (stabilhoz lista kell) • Duplikátum-kezelés tisztázatlan • O(n*m) kerülendő

Feladatok (5 db) – megoldás vázlattal

1. Unió halmazként.

Megoldás vázlat: HashSet Add.

C# tipp: new HashSet<int>(a); set.UnionWith(b)

2. Stabil unió (A sorrendje megmarad).

Megoldás vázlat: seen.Add + lista.

C# tipp: UnioStabil(a,b)

3. Unió elemszám.

Megoldás vázlat: set.Count.

C# tipp: set.Count

4. String unió case-insensitive.

Megoldás vázlat: StringComparer.

C# tipp: HashSet<string>(..., OrdinalIgnoreCase)

5. Rendezett unió rendezetten.

Megoldás vázlat: Merge.

C# tipp: OsszefuttatasUnio(a,b)

Házi feladat ötletek

• Unió HashSet nélkül.

• Unió + metszet harmadik tömbbel.

• Multihalmaz unió (összefűzés).

Keresések

12. Keresés rendezett tömbben (bináris)

Rendezett tömbben logaritmikus keresés.

Pszeudokód (sablon)

```
elso = 0

utolso = n-1

ciklus amíg elso <= utolso

    kozep = elso + (utolso-elso) div 2

    ha t[kozep] = kulcs akkor ki kozep és kilép

    különben ha kulcs < t[kozep] akkor utolso = kozep - 1

    különben elso = kozep + 1

ciklus vége

ki -1
```

C# kódsablon

```
static int BinarisKereses(int[] t, int kulcs)

{

    int elso = 0, utolso = t.Length - 1;

    while (elso <= utolso)

    {

        int kozep = elso + (utolso - elso) / 2;

        if (t[kozep] == kulcs) return kozep;

        if (kulcs < t[kozep]) utolso = kozep - 1;

        else elso = kozep + 1;

    }

    return -1;

}
```

Gyakori hibák: Bemenet nincs rendezve • Határfrissítés hibás • Közép számítás hibás

Feladatok (5 db) – megoldás vázlattal

1. Kulcs indexe.

Megoldás vázlat: Bináris keresés.

C# tipp: BinarisKereses(t,k)

2. Van-e benne?

Megoldás vázlat: index!=-1.

C# tipp: BinarisKereses(t,k)!=-1

3. Első előfordulás (lower_bound).

Megoldás vázlat: Találatkor menj balra.

C# tipp: lower_bound minta

4. Előfordulások száma.

Megoldás vázlat: upper-lower.

C# tipp: ub-lb

5. Beszúrási hely.

Megoldás vázlat: lower_bound.

C# tipp: return elso

Házi feladat ötletek

• Generikus bináris keresés.

• Legközelebbi elem.

• Több kulcs keresése.

Programozási tételek egymásra építése

13. Összefuttatás (merge)

Két rendezett sorozat összeolvasztása rendezetten (unió/összefésülés).

Pszeudokód (sablon)

```
i = 0; j = 0

C = üres

amíg i < n és j < m

    a[i] és b[j] közül a kisebbet C-be, mutató lép

maradék elemeket C-be
```

C# kódsablon

```
static int[] OsszefuttatasUnio(int[] a, int[] b)

{

    int i = 0, j = 0;

    var c = new List<int>();

    while (i < a.Length && j < b.Length)

    {

        if (a[i] < b[j]) c.Add(a[i++]);

        else if (a[i] == b[j]) { c.Add(a[i]); i++; j++; }

        else c.Add(b[j++]);

    }

    while (i < a.Length) c.Add(a[i++]);

    while (j < b.Length) c.Add(b[j++]);

    return c.ToArray();

}
```

Gyakori hibák: Egyenlő eset kezelése hiányzik • Maradék kimarad • Bemenet nem rendezett

Feladatok (5 db) – megoldás vázlattal

1. Rendezett unió duplikátum nélkül.

Megoldás vázlat: Merge unió.

C# tipp: OsszefuttatasUnio(a,b)

2. Rendezett összefűzés duplikátumokkal.

Megoldás vázlat: Egyenlőnél mindkettő.

C# tipp: if(a[i]==b[j]) add both

3. Rendezett metszet O(n+m).

Megoldás vázlat: Egyenlőnél add, különben kisebbik lép.

C# tipp: MergeIntersect

4. Különböző elemek száma merge közben.

Megoldás vázlat: last követése.

C# tipp: track last

5. Három rendezett tömb merge.

Megoldás vázlat: Kétszer merge.

C# tipp: Merge(Merge(a,b),c)

Házi feladat ötletek

- Mergesort alapfeladat.
- Időmérés.
- String merge IComparer-rel.

Rendezések

14. Rendezések

Rendezési algoritmusok (buborék, kiválasztásos, beszúrásos; kitekintés: quick/merge).

Pszeudokód (sablon)

```
// Buborékrendezés

ciklus i = n-1 .. 1

    ciklus j = 0 .. i-1

        ha t[j] > t[j+1] akkor csere

    ciklus vége

ciklus vége
```

C# kódsablon

```
static void BuborekRendezes(int[] t)

{

    for (int i = t.Length - 1; i >= 1; i--)

        for (int j = 0; j < i; j++)

            if (t[j] > t[j + 1])

            {

                int tmp = t[j];

                t[j] = t[j + 1];

                t[j + 1] = tmp;

            }

}
```

Gyakori hibák: Csere tmp nélkül • Rossz ciklushatárok • Rendezés iránya téves

Feladatok (5 db) – megoldás vázlattal

1. Buborékrendezés implementálása.

Megoldás vázlat: Sablon.

C# tipp: BuborekRendezes(t)

2. Kiválasztásos rendezés.

Megoldás vázlat: MinIndex + csere.

C# tipp: SelectionSort

3. Beszúrásos rendezés.

Megoldás vázlat: Kulcs + tolás.

C# tipp: InsertionSort

4. Csökkenő sorrend.

Megoldás vázlat: Fordítsd meg a relációt.

C# tipp: if(t[j]<t[j+1]) swap

5. Rekord rendezés kulcs szerint.

Megoldás vázlat: IComparer/Array.Sort.

C# tipp: Array.Sort(arr, comparer)

Házi feladat ötletek

• Mérj időt: buborék vs Array.Sort.

• Mergesort implementálása.

• Stabilitás vizsgálata.

Mellékeletek / 1

16. Hibavadászat (C#)

Javítsd a kódrészletek hibáit és röviden indokold a javítást!

1) Off-by-one összegzés

```
int sum=0;

for(int i=0;i<=t.Length;i++) sum+=t[i];
```

Javítás: i < t.Length

2) While sorrend

```
int i=0;

while(!pred(t[i]) && i<t.Length) i++;
```

Javítás: i<t.Length && !pred(t[i])

3) db inicializálás hiányzik

```
int db;

for(int i=0;i<t.Length;i++) if(t[i]<0) db++;
```

Javítás: int db=0;

4) Bináris határfrissítés

```
if(k<ul) elso=kozep-1; else utolso=kozep+1;
```

Javítás: utolso=kozep-1; elso=kozep+1

5) Csere tmp nélkül

```
t[i]=t[j]; t[j]=t[i];
```

Javítás: tmp=t[i]; t[i]=t[j]; t[j]=tmp;

6) Kiválogatásnál k++ hiányzik

```
if(a[i]<5) b[k]=a[i];
```

Javítás: b[k++]=a[i];

7) Minimum kezdőérték 0

```
int min=0; for(i=0;i<n;i++) if(t[i]<min) min=t[i];
```

Javítás: min=t[0]; i=1-től

8) Eldöntés felülír

```
bool van=false; for(i=0;i<n;i++) van=(t[i]==0);
```

Javítás: van |= (t[i]==0); vagy break

17. 45 perces dolgozat minta

Időtartam: 45 perc. Összesen 50 pont. Segédeszköz: papír/ceruza.

Feladatok

1. feladat (10p) Számold ki a pozitív elemek összegét és darabszámát egy menetben!

2. feladat (12p) Keresd meg egy kulcs első indexét; ha nincs, -1.

3. feladat (14p) Két rendezett tömb rendezett uniója összefuttatással (duplikátum nélkül).

4. feladat (14p) Írj C# metódust: legnagyobb abszolútértékű elem indexe.

Megoldókulcs (vázlat)

1. feladat (10p): osszeg=0; db=0; ha t[i]>0: osszeg+=t[i]; db++;

2. feladat (12p): Lineáris keresés sablon.

3. feladat (14p): Merge unió sablon, utóciklusok.

4. feladat (14p): mi=0; for i=1..: if Abs(t[i])>Abs(t[mi]) mi=i; return mi;

18. Melléklet: Tétel → C# kódsablonok (kivonat)

Összegzés

```
static int Osszegzes(int[] t)

{

    int osszeg = 0;

    for (int i = 0; i < t.Length; i++) osszeg += t[i];

    return osszeg;

}
```

Megszámlálás

```
static int Megszamlalas(int[] t, Func<int,bool> feltetel)

{

    int db = 0;

    for (int i = 0; i < t.Length; i++) if (feltetel(t[i])) db++;

    return db;

}
```

Eldöntés

```
static bool Eldontes(int[] t, Func<int,bool> feltetel)

{

    int i = 0;

    while (i < t.Length && !feltetel(t[i])) i++;

    return i < t.Length;

}
```

Kiválasztás

```
static int Kivalasztas(int[] t, Func<int,bool> feltetel)

{

    int i = 0;

    while (!feltetel(t[i])) i++;

    return i; // előfeltétel: létezik

}
```

Keresés (lineáris)

```
static int Kereses(int[] t, Func<int,bool> feltetel)

{

    for (int i = 0; i < t.Length; i++) if (feltetel(t[i])) return i;

    return -1;

}
```

Másolás

```
static int[] Masolas(int[] a, Func<int,int> muvelet)

{

    int[] b = new int[a.Length];

    for (int i = 0; i < a.Length; i++) b[i] = muvelet(a[i]);

    return b;

}
```

Kiválogatás

```
static int[] Kivalogatas(int[] a, Func<int,bool> feltetel)

{

    var lista = new List<int>();

    foreach (var x in a) if (feltetel(x)) lista.Add(x);

    return lista.ToArray();

}
```

Szétválogatás

```
static (int[] igazak, int[] hamisak) Szetvalogatas(int[] a, Func<int,bool>
feltetel)

{

    var ig = new List<int>();

    var ha = new List<int>();

    foreach (var x in a) if (feltetel(x)) ig.Add(x); else ha.Add(x);

    return (ig.ToArray(), ha.ToArray());

}
```

Metszet

```
static int[] Metszet(int[] a, int[] b)

{

    var setB = new HashSet<int>(b);

    var c = new List<int>();

    foreach (var x in a) if (setB.Contains(x)) c.Add(x);

    return c.Distinct().ToArray();

}
```

Unió

```
static int[] UnioStabil(int[] a, int[] b)

{

    var seen = new HashSet<int>();

    var c = new List<int>();

    foreach (var x in a) if (seen.Add(x)) c.Add(x);

    foreach (var x in b) if (seen.Add(x)) c.Add(x);

    return c.ToArray();

}
```

Maximum kiválasztás

```
static int MaxIndex(int[] t)

{

    int mi = 0;

    for (int i = 1; i < t.Length; i++) if (t[i] > t[mi]) mi = i;

    return mi;

}
```

Minimum kiválasztás

```
static int MinIndex(int[] t)

{

    int mi = 0;

    for (int i = 1; i < t.Length; i++) if (t[i] < t[mi]) mi = i;

    return mi;

}
```

Sorozatszámítás (általános)

```
static T Sorozatszamitas<T>(IEnumerable<T> sor, T s0, Func<T,T,T> f)

{

    T s = s0;

    foreach (var x in sor) s = f(s, x);

    return s;

}
```

Keresés rendezett tömbben (bináris)

```
static int BinarisKereses(int[] t, int kulcs)

{

    int elso = 0, utolso = t.Length - 1;

    while (elso <= utolso)

    {

        int kozep = elso + (utolso - elso) / 2;

        if (t[kozep] == kulcs) return kozep;

        if (kulcs < t[kozep]) utolso = kozep - 1;

        else elso = kozep + 1;

    }

    return -1;

}
```

Összefuttatás (merge)

```
static int[] OsszefuttatasUnio(int[] a, int[] b)

{

    int i = 0, j = 0;

    var c = new List<int>();

    while (i < a.Length && j < b.Length)

    {

        if (a[i] < b[j]) c.Add(a[i++]);

        else if (a[i] == b[j]) { c.Add(a[i]); i++; j++; }

        else c.Add(b[j++]);

    }

    while (i < a.Length) c.Add(a[i++]);

    while (j < b.Length) c.Add(b[j++]);
```

Rendezések

```
static void BuborekRendezes(int[] t)

{

    for (int i = t.Length - 1; i >= 1; i--)

        for (int j = 0; j < i; j++)

            if (t[j] > t[j + 1])

            {

                int tmp = t[j];

                t[j] = t[j + 1];

                t[j + 1] = tmp;

            }

}
```

19. Források

• szit.hu – Programozási tételek (mondatszerű leírás): https://szit.hu/doku.php?id=oktatas:programozas:programozasi_tetelek:mondatszeru_leiras

• ELTE IK – Számítógépes problémamegoldás I. tematika: http://szprob1.elte.hu/