Objektumorientált Programozás VII.

Összetett programozási tételek
Programozási tételek összeépítése
Feladatok

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.

Objektumorientált Programozás VII.

Összetett programozási tételek

Programozási tételek összeépítése

Feladatok

Másolás

- **Bemenet:**
  X: Feldolgozandó tömb
  N: Tömb elemeinek száma
- **Kimenet:**
  Y: Eredmény tömb

```
Eljárás Másolás(X, N, Y)
    Ciklus i := 1-től N-ig
        Y[i] := művelet X[i]
    Ciklus vége
Eljárás vége
```

Másolás: példa

- Állítsuk elő egy int tömb elemeinek négyzetét.

```
static int[] négyzet(int[] tömb)
{
    int[] négyzetek = new int[tömb.Length];
    for (int i = 0; i < tömb.Length; i++)
    {
        négyzetek[i] = tömb[i] * tömb[i];
    }
    return négyzetek;
}
```

Kiválogatás

- **Bemenet:**
  X: Feldolgozandó tömb
  N: Tömb elemeinek száma
  T: Tulajdonság függvény
- **Kimenet:**
  Y: Eredmény tömb
  DB: Tömb elemeinek száma

```
Eljárás Kiválogatás(X, N, T, Y, DB)
    DB := 0
    Ciklus i := 1-től N-ig
        Ha (X[i] teljesíti T-t) akkor
            DB := DB + 1
            Y[DB] := X[i]
        Elágazás vége
    Ciklus vége
Eljárás vége
```

Kiválogatás: példa

- Válogassuk ki egy szövegből a benne szereplő számjegyeket.

```
static string számjegyek(string szöveg)
{
    string számok = " ";
    for (int i = 0; i < szöveg.Length; i++)
    {
        if (Char.IsDigit(szöveg[i])) számok += szöveg[i];
    }
    return számok;
}
```

Szétválogatás

- **Bemenet:**
  X: Feldolgozandó tömb
  N: Tömb elemeinek száma
  T: Tulajdonság függvény
- **Kimenet:**
  Y: Egyik eredmény tömb
  Z: Másik eredmény tömb
  DBY: Y elemeinek száma
  DBZ: Z elemeinek száma

```
Eljárás Szétválogatás(X, N, T, Y, DBY, Z, DBZ)
    DBY := 0;  DBZ := 0
    Ciklus i := 1-től N-ig
        Ha (X[i] teljesíti T-t) akkor
            DBY := DBY + 1
            Y[DBY] := X[i]
        Különben
            DBZ := DBZ + 1
            Z[DBZ] := X[i]
        Elágazás vége
    Ciklus vége
Eljárás vége
```

Szétválogatás: példa

- Válogassuk szét egy int tömb elemeit egy új tömbbe úgy, hogy a párosak az elejére, a páratlanok a végére kerüljenek.

```
static int[] válogat(int[] tömb)
{
    int[] újtömb = new int[tömb.Length];
    int páros = 0;
    int páratlan = tömb.Length - 1;
    for (int i = 0; i < tömb.Length; i++) {
        if (tömb[i] % 2 == 0) {
            újtömb[páros] = tömb[i];
            páros++;
        } else {
            újtömb[páratlan] = tömb[i];
            páratlan- -;
        }
    }
    return újtömb;
}
```

Metszet

```
Eljárás Metszet(X, N, Y, M, Z, DB)
    DB := 0
    Ciklus i := 1-től M-ig
        j := 1
        Ciklus amíg (j ≤ N) és (X[i] ≠ Y[j])
            j := j+ 1
        Ciklus vége
        Ha j ≤ N akkor
            DB := DB + 1
            Z[DB] := X[i]
        Elágazás vége
    Ciklus vége
Eljárás vége
```

- **Bemenet:**
  X: Egyik feldolgozandó tömb
  Y: Másik feldolgozandó tömb
  M: X elemeinek száma
  N: Y elemeinek száma
- **Kimenet:**
  Z: Eredmény tömb
  DB: Z elemeinek száma

Metszet: példa

- Gyűjtsük ki a magánhangzókat egy megadott szövegből.

```
static string magánhangzók(string szöveg)
{
    string mgh = "aáeéiíoóöőuúüű";
    string magánhangzó = "";
    for (int i = 0; i < szöveg.Length; i++)
    {
        int j = 0;
        while (j < mgh.Length && szöveg[i] != mgh[j]) j++;
        if (j < mgh.Length) magánhangzó += szöveg[i];
    }
    return magánhangzó;
}
```

Egyesítés (unió)

```
Eljárás Egyesítés(X, N, Y, M, Z, DB)
    Z := X
    DB := M
    Ciklus j := 1-től N-ig
        i := 1
        Ciklus amíg (i ≤ M) és (X[i] ≠ Y[j])
            i := i+ 1
        Ciklus vége
        Ha i > M akkor
            DB := DB + 1
            Z[DB] := Y[j]
        Elágazás vége
    Ciklus vége
Eljárás vége
```

- **Bemenet:**
  X: Egyik feldolgozandó tömb
  Y: Másik feldolgozandó tömb
  M: X elemeinek száma
  N: Y elemeinek száma
- **Kimenet:**
  Z: Eredmény tömb
  DB: Z elemeinek száma

Unió: példa

```
static int[] egyesít(int[] tömb1, int[] tömb2)
{
    int[] újtömb=new int[tömb1.Length+tömb2.Length];
    tömb1.CopyTo(újtömb, 0);
    int db = tömb1.Length;
    for (int j = 0; j < tömb2.Length; j++)
    {
        int i = 0;
        while (i < tömb1.Length && tömb1[i] != tömb2[j]) i++;
        if (i == tömb1.Length)
        {
            újtömb[db] = tömb2[j];
            db++;
        }
    }
    int[] unió = new int[db];
    for (int i = 0; i < unió.Length; i++) unió[i] = újtömb[i];
    return unió;
}
```

Összefuttatás (rendezettek uniója)

```
Eljárás Összefuttatás(M, X, N, Y, DB, Z)
    i := 1; j := 1; DB := 0
    Ciklus amíg (i ≤ M) és (j ≤ N)
        DB := DB + 1
        Elágazás
            X[i ] < Y[j ] esetén Z[DB] := X[i ]; i := i + 1
            X[i ] = Y[j ] esetén Z[DB] := X[i ]; i := i + 1; j := j + 1
            X[i ] > Y[j ] esetén Z[DB] := Y[j ]; j := j + 1
        Elágazás vége
    Ciklus vége
    Ciklus amíg i ≤ M
        DB := DB + 1; Z[DB] := X[i ]; i := i + 1
    Ciklus vége
    Ciklus amíg j ≤ N
        DB := DB + 1; Z[DB] := Y[j ]; j := j + 1
    Ciklus vége
Eljárás vége
```

- **Bemenet:**
  X: Egyik feldolgozandó tömb
  Y: Másik feldolgozandó tömb
  M: X elemeinek száma
  N: Y elemeinek száma
- **Kimenet:**
  Z: Eredmény tömb
  DB: Z elemeinek száma

Összefuttatás: példa

```
static int[] összefuttat(int[] tömb1, int[] tömb2)
{
    int[] újtömb = new int[tömb1.Length + tömb2.Length];
    int i = 0, j = 0, db = 0;
    while (i < tömb1.Length && j < tömb2.Length) {
        if (tömb1[i] < tömb2[j]) {
            újtömb[db] = tömb1[i]; i++;
        } else if (tömb1[i] > tömb2[j]) {
            újtömb[db] = tömb2[j]; j++;
        } else {
            újtömb[db] = tömb1[i]; i++; j++;
        }
        db++;
    }
    while (i < tömb1.Length) {
        újtömb[db] = tömb1[i]; i++; db++;
    }
    while (j < tömb2.Length) {
        újtömb[db] = tömb2[j]; j++; db++;
    }
    int[] unio = new int[db];
    for (int x = 0; x < unio.Length; x++) unio[x] = újtömb[x];
    return unio;
}
```

Objektumorientált Programozás VII.

Összetett programozási tételek

Programozási tételek összeépítése

Feladatok

Programozási tételek összeépítése

- **Összetettebb programok esetén szintén használhatjuk a programozási tételeket, ilyenkor gyakran szükség van az egymásba építésükre.**
- **Gyakori megvalósítások**
  - tételek egymás után
  - tételek egymásba ágyazva
  - egyéb (optimalizált) megoldások

Egy példa az összeépítésre

- **Megszámlálás-eldöntés**
  - Egy A tömbben van-e legalább K darab T tulajdonságú elem?

```
Eljárás SzámolásÉsEldöntés(A, N, T, K, VAN)
    i := 1; DB := 0
    Ciklus amíg (i ≤ N) és (DB < K)
        Ha (A[i] teljesíti T-t) akkor
            DB := DB + 1
        Elágazás vége
        i := i + 1
    Ciklus vége
    VAN := (DB = K)
Eljárás vége
```

További példák

- **Megszámolás-kiválasztás**
  - pl. hol van a K-adik T tulajdonságú elem?
- **Megszámolás-keresés**
  - pl. van-e K darab T tulajdonságú elem, és ha igen, hol található a sorozatban a K-adik?
- **Maximumkiválasztás-megszámlálás**
  - pl. hány darab maximális elem van a listában?
- **Maximumkiválasztás-kiválogatás**
  - pl. melyek a maximális elemek a listában? (értelemszerűen az indexeket várjuk)
- **Kiválogatás-sorozatszámítás**
  - pl. adjuk össze az összes T tulajdonságú elemet!

További példák

- **Kiválogatás-maximumkiválasztás**
  - pl. keressük meg a T tulajdonságú elemek közül a maximálisat!
- **Kiválogatás-másolás**
  - pl. másoljuk le a sorozat T tulajdonságú elemeit, úgy, hogy végezzünk rajtuk valamilyen elemi műveletet is!
- **Másolás-sorozatszámítás**
  - pl. adjuk meg a sorozat elemeinek négyzetösszegét!
- **Másolás-maximumkiválasztás**
  - pl. adjuk meg a sorozat elemei közül azt, amelyiknek maximális az abszolút értéke!
- **Egyéb összeépítések**

Objektumorientált Programozás VII.

Összetett programozási tételek
Programozási tételek összeépítése
Feladatok

Gyakorló feladatok

Egy felhasználó által megadott, tetszőleges szövegre oldjuk meg az alábbi feladatokat!

Határozza meg a szöveg szótagszámát!
(Tipp: ahány magánhangzó, annyi szótag.)

Válogassa ki a szövegből a nagybetűket!
(Tipp: Char. IsUpper() metódus)

Van-e a szövegben legalább három 'a' betű?
Ha igen, hol van a harmadik?

Másolja le a szöveget úgy, hogy minden benne található kis 'x'-et nagy 'X'-re cserél!

Gyakorló feladatok

Egy véletlenszámokkal feltöltött int tömbre oldjuk meg az alábbi feladatokat!

Határozza meg, hány db van a minimális, és hány db a maximális elemből!

Összegezze a tömbben található páros számokat!

Válogassa ki a tömbben található prímeket!

Válogassa szét a tömb elemeit aszerint, hogy kisebbek vagy nagyobbak a tömbátlagnál.

Objektumorientált Programozás VII.

- ✓Összetett programozási tételek
- ✓Programozási tételek összeépítése
- ✓Feladatok

Irodalom, feladatok

- **Kotsis-Légrádi-Nagy-Szénási: Többnyelvű programozástechnika, PANEM, Budapest, 2007**
- **Zsakó-Szlávi: Mikrológia 19.**
- **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**