Bevezetés a programozásba

3. Előadás

Algoritmusok, programozási tételek

Programkonstrukciók összefoglalás

- Elágazas kell, ha más kódra van szükség egyes esetekben
- Ciklus kell, ha ismételni kell lépéseket
- Okos megbízható matematikusok bebizonyították: Minden algoritmikusan megoldható probléma megoldható szekvencia, elágazás és ciklus segítségével.

- Tehát vége is a féléves anyagnak…

- Érdemes lesz azért bejárni …

Algoritmus

- Sokféle definíció van, pl.: *„Az algoritmus egy megengedett lépésekből álló módszer, utasítássorozat, részletes útmutatás, recept, amely alkalmas arra, hogy valamilyen felmerülő problémára megoldást adjon.”*
- Praktikusan elágazásból, ciklusból, értékadásból és I/O műveletekből álló helyes program elve, amely egy adott feladatot megold
  - A fogalom ennél absztraktabb
- Néhány algoritmus általános iskolai anyag, mint a „papíron szorzás”
- Néhány száz algoritmusnak neve is van
- Miért jó ismerni algoritmusokat? Ha valaki már talált megoldást egy problémára, nekünk ne kelljen újra kitalálni!

Program - Algoritmus

- A program **konkrét** nyelven írt, **konkrét** feladatot old meg, **konkrét** formátumban megadott bemenettel és kimenettel.
  - Valamint a programnak figyelembe kell vennie a rendelkezésre álló erőforrásokat.
- Az algoritmus a megoldás elve, formátumtól, típusoktól amennyire lehet, függetlenül.
  - Azaz az algoritmus egy általános megfogalmazás.

A továbbiakban megnézzünk konkrét programokat, majd azokból absztrakcióval kapott általános algoritmusokat

Sorozat

- A mai előadás minden sorozatának formátuma a következő:
  - Az első bemenet egy szám, amely az érkező sorozat hosszát jelenti (azaz a bemeneten érkező elemek számát)
  - Ezután következnek a sorozatelemek
  - Például:

BEMENET:

5 ** *azaz 5 elemű sorozatot fogok megadni*

1 3 5 7 9 ** *maga az 5 elemű sorozat*

Sorozat minden elemét „lemásoló”

```
PROGRAM sorozat_elemek_masolasa
    VÁLTOZÓK:
        n, i: EGÉSZ,
        a: VALÓS

    BE: n
    i := 0

    CIKLUS AMÍG i < n
        BE: a
        KI: a, SV
        i := i + 1
    CIKLUS_VÉGE

PROGRAM_VÉGE
```

Összegzés PLanG-ban

```
PROGRAM sorozatösszeadó
    VÁLTOZÓK:
        n, i: EGÉSZ,
        a, összeg: VALÓS

    BE: n
    i := 0
    összeg := 0
    CIKLUS AMÍG i < n
        BE: a
        összeg := összeg + a
        i := i + 1
    CIKLUS_VÉGE
    KI: összeg
PROGRAM_VÉGE
```

Mennyi a sorozatban levő számok összege?

Összegzés tétele

```
Változók:  összeg, a : T

  összeg := 0
  CIKLUS AMÍG nincs vége a sorozatnak
    a := következő elem
    összeg := összeg ⊕ f(a)
  CIKLUS_VÉGE
```

Összegzés tétele

- Mire jó?
  - Kumulatív (halmozó) feladatok sorozatokon
- Az algoritmus melyik része cserélhető?
  - Kezdőérték
  - Művelet / függvény
- Példák
  - Átlag
  - Faktoriális
  - Négyzetösszeg

Számlálás PLanG-ban

```
PROGRAM számlálás
    VÁLTOZÓK:
        n, i, számláló, a: EGÉSZ

    BE: n
    i := 0
    számláló := 0
    CIKLUS AMÍG i < n
        BE: a
        HA a MOD 2 = 0 AKKOR
            számláló := számláló + 1
        HA_VÉGE
        i := i + 1
    CIKLUS_VÉGE
    KI: számláló
PROGRAM_VÉGE
```

Hány páros szám van a sorozatban?

Számlálás tétele

```
Változók: számláló: EGÉSZ,
      a : T

  számláló := 0
  CIKLUS AMÍG nincs vége a sorozatnak
     a := következő elem
     HA feltétel(a) AKKOR
        számláló := számláló + 1
     HA_VÉGE
  CIKLUS_VÉGE
```

Számlálás tétele

- Mire jó?
  - „mennyi?”, „hány?” feladatok sorozatokon
- Az algoritmus melyik része cserélhető?
  - Feltétel
  - Növelő függvény
- Példák
  - Osztók száma
  - Szavak száma egy szövegben

Lineáris keresés PLanG-ban

```
PROGRAM lineáris_keresés
     VÁLTOZÓK:
          n, i, a, hol: EGÉSZ,
          van: LOGIKAI

     BE: n
     i := 0
     van := HAMIS
     hol := 0
     CIKLUS AMÍG i < n ÉS NEM van
          BE: a
          HA a MOD 2 = 0 AKKOR
               van := IGAZ
               hol := i
          HA_VÉGE
          i := i + 1
     CIKLUS_VÉGE
     KI: van
     HA van AKKOR
          KI: hol
     HA_VÉGE
PROGRAM_VÉGE
```

Melyik az első páros szám a sorozatban?
Van-e egyáltalán páros szám?

Adjuk meg a “van” változóban, hogy van-e páros szám, és a “hol” változóban az első páros számot.

Lineáris keresés tétele

```
Változók: hol, i: EGÉSZ, van: LOGIKAI, a : T

   van := HAMIS
      hol := 0
   i := 0
   CIKLUS AMÍG nincs vége a sorozatnak ÉS NEM van
      a := következő elem
      HA feltétel(a) AKKOR
         van := IGAZ
         hol := i
      HA_VÉGE
      i := i + 1
   CIKLUS_VÉGE
```

Lineáris keresés tétele

- Mire jó?
  - „van-e?”, „és hányadik, ha van?” feladatok sorozatokon
- Az algoritmus melyik része cserélhető?
  - Feltétel
- Példák
  - Szerepel-e egy bizonyos érték egy sorozatban és hol
  - Prím-e egy szám (van-e osztója?)

Így is ismerheted

- A Lineáris keresés tételének részfeladatai:
  - **Eldöntés:**
    - Van-e a sorozatnak olyan eleme amelyre igaz a *feltétel*?
    - Tagadott változat: Igaz-e a *feltétel* a sorozat összes elemére?
  - **Kiválasztás:**
    - Hányadik az a sorozat elem amelyre igaz a *feltétel*?

Maximum keresés PLanG-ban

```
PROGRAM maximumkeresés
    VÁLTOZÓK:
        n, i, hol: EGÉSZ,
        a, max: VALÓS

    BE: n
    i := 1
    BE: a
    max := a
    hol := 1
    CIKLUS AMÍG i < n
        BE: a
        HA max < a AKKOR
            max := a
            hol := i
        HA_VÉGE
        i := i + 1
    CIKLUS_VÉGE
    KI: max, SV, hol
PROGRAM_VÉGE
```

Melyik a legnagyobb szám a sorozatban? A pozíciója és az értéke is érdekes, mindkettőt szeretnénk a “max” és a “hol” változókban megkapni.

Az előzőektől különböző módon nincs jó válasz üres sorozatra, ezért feltesszük, hogy legalább egy elem van

Maximum keresés tétele

```
Változók:  i, hol: EGÉSZ,
        a, max : T

   i := 1
   a := első elem
   max := f(a)
   hol := 1
   CIKLUS AMÍG nincs vége a sorozatnak
       a := következő elem
       HA max < f(a) AKKOR
           max := f(a)
           hol := i
       HA_VÉGE
      i := i + 1
   CIKLUS_VÉGE
```

Maximum keresés tétele

- Mire jó?
  - „Mi / Mennyi a leg… ?” feladatok sorozatokon
- Az algoritmus melyik része cserélhető?
  - Feltétel, reláció
  - Függvény
- Példák
  - Maximum, minimum keresés
  - Zárójelek mélysége szövegben

Elemenként feldolgozás

- Sok adat, kevés változó
- Egy feladat elemenként feldolgozható, ha
  - Meg lehet úgy oldani, hogy a hosszú adatsorból egyszerre csak kevésre van szükség, és ez a kevés nem függ a sorozat hosszától
  - Csak egyszer kell végignézni mindegyiket
- Sok hasznos algoritmus tartozik ide
  - Az előbb látottak
  - Válogatás (pl. sorozatból a párosakat írjuk ki)
  - Összefésülés (két sorozat egyesítése)
  - Metszetképzés, unióképzés

Elemenként feldolgozás

- Példa egyéb elemenként feldolgozható feladatra:
  - Sorozat értékkészletét befoglaló intervallum (min – max)
  - Sorozat második legnagyobb értéke
  - Van-e még egy olyan elem, amely megegyezik az első elemmel?
  - Van-e két egymást követő egyforma elem a sorozatban?
- Példa feladatra, amely **NEM** elemenként feldolgozható:
  - Sorozat rendezése (növekvő/csökkenő sorrendbe)
  - Sorozat mediánja (középértéke)
  - Van-e két egyforma elem a sorozatban

Tételek kombinációi

- Feladat: egy intervallumban számoljuk meg a prímszámokat!
- Számlálás tétel kell hozzá
- De ez még nem elég: nincsen „prím-e?” műveletünk
- Ezért a prímdöntést lineáris kereséssel (van-e valódi osztója?) dönthetjük el a számlálás belsejében
- -> „állapottér transzformáció”, úgy teszünk a számlálásban, mintha logikai értékeket olvasnánk, amelyeket egy másik tétellel állítunk elő

Prímek száma n elemű sorozatban

```
Változók:  n, i, a, sz: EGÉSZ

  BE: n
  i := 0
  sz := 0
  CIKLUS AMÍG i < n
    BE: a
    HA prím(a) AKKOR
   sz:= sz + 1
    HA_VÉGE
    i := i + 1
  CIKLUS_VÉGE
  KI: sz
```

Prímek száma n elemű sorozatban

```
Változók:  n, i,

  BE: n
  i := 0
  sz := 0
  CIKLUS AMÍG i
    BE: a
    HA prím(a) A
    sz:= sz + 1
    HA_VÉGE
    i := i + 1
  CIKLUS_VÉGE
  KI: sz
```

```
PROGRAM van_e_valódi_osztó
    VÁLTOZÓK: osztók, a: EGÉSZ,
              van: LOGIKAI

    BE: a
    van := HAMIS
    osztók := 2
    CIKLUS AMÍG osztók < a ÉS NEM van
      HA a MOD osztók = 0 AKKOR
       van := igaz
      HA_VÉGE
      osztók := osztók + 1
    CIKLUS_VÉGE
    KI: van
PROGRAM_VÉGE
```

```
PROGRAM tétel_kombináció
  Változók: n, i, a, sz, osztók: EGÉSZ,
        van: LOGIKAI

  BE: n
  i := 0
  sz := 0
  CIKLUS AMÍG i < n
    BE: a
    van := HAMIS
    osztók := 2
    CIKLUS AMÍG osztók < a ÉS NEM van
        HA a MOD osztók = 0 AKKOR
          van := igaz
        HA_VÉGE
        osztók := osztók + 1
    CIKLUS_VÉGE
    HA NEM VAN ÉS a /= 1 AKKOR
     sz:= sz + 1
    HA_VÉGE
    i := i + 1
  CIKLUS_VÉGE
  KI: sz

PROGRAM_VÉGE
```

```
Változók:   n, i, a, sz: EGÉSZ

  BE: n
  i := 0
  sz := 0
  CIKLUS AMÍG i < n
    BE: a
    HA prím(a) AKKOR
      sz:= sz + 1
    HA_VÉGE
    i := i + 1
  CIKLUS_VÉGE
  KI: sz
```

```
VÁLTOZÓK:   osztók, a: EGÉSZ,
       van: LOGIKAI

BE := a
van := HAMIS
osztók := 2
CIKLUS AMÍG osztók < a ÉS NEM van
     HA a MOD osztók = 0 AKKOR
        van := igaz
     HA_VÉGE
     osztók := osztók + 1
CIKLUS_VÉGE
KI: van
```