Bevezetés a programozásba

4. Előadás
Sorozatok, fájlok

ISMÉTLÉS

Specifikáció

- Előfeltétel: milyen körülmények között követelünk helyes működést
- Utófeltétel: mit várunk a kimenettől, mi az összefüggés a kimenet és a bemenet között
- Ezek **feltételek**, tehát vagy teljesülnek, vagy nem. Ha teljesülnek, akkor a program megoldja a feladatot.
- A specifikáció feltételekből áll, nem utasításokból, mert a feladatot írja le, és nem a programot.

ISMÉTLÉS

Elágazás

```
PROGRAM elágazás
  VÁLTOZÓK:
    a: EGÉSZ

  BE: a
  HA a > 0 AKKOR
    KI: "pozitív"
  KÜLÖNBEN
    KI: "nem pozitív"
  HA_VÉGE
PROGRAM_VÉGE
```

ISMÉTLÉS

Ciklus

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

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

ISMÉTLÉS

Összegzés tétele

```
Változók: összeg, a : T

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

ISMÉTLÉS

Számlálás tétele

```
VÁLTOZÓK: sz: EGÉSZ, a:T

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

ISMÉTLÉS

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
  i := i + 1
  HA feltétel(a) AKKOR
    van := IGAZ
    hol := i
  HA_VÉGE
CIKLUS_VÉGE
```

ISMÉTLÉS

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
  i := i + 1
  HA max < f(a) AKKOR
    max := f(a)
    hol := i
  HA_VÉGE
CIKLUS_VÉGE
```

Sorozatok

- A jellemző műveletek:
  - Következő elem olvasása
  - Sorozat végének felismerése/lekérdezése
  - Inicializálás (kezdeti értékek beállítása, olvasás lehetővé tétele)
    - Pl az előző példákban a sorozat hosszának beolvasása

Ismert hosszú sorozat

- A sorozat hossza ismert, vagy beolvasható
- Számoljuk az olvasások számát, és ha elérjük a sorozat hosszát, abbahagyjuk

```
Változók: i,n : egész, X : T

  n := Sorozat hossza, (pl: BE: n)
  i := 0
  CIKLUS AMÍG i<n
     BE: X
     X feldolgozása ...
     i:=i+1
  CIKLUS_VÉGE
```

Ismert hosszú sorozatra példa

- Számok intervalluma
- Táblázatoknál szokás először jelezni a méreteket
- Kép és hang formátumok sokszor ilyenek (.wav, .bmp)

Végjeles sorozat

- A sorozat értékkészletét megszorítva lehetővé válik, hogy speciális jelentésű értékeket használjunk
  - Például csupa nemnegatív elem van a sorozatban, és az első negatív elem jelzi a sorozat végét
- Előnye: könnyen bővíthető a sorozat
- Hátránya: nem használhatjuk az adott típus teljes készletét

Végjeles sorozat

- Jellegzetesség: a beolvasás után még el kell dönteni, hogy sorozatelemről van-e szó, vagy a végjelről, ami nem része a sorozatnak
- Tehát a beolvasás és a feldolgozás között kell lennie az ellenőrzésnek
- Az ellenőrzésnek a ciklusfeltételben kell lennie
- Következésképpen
  - A beolvasásnak a ciklus utolsó lépésének kell lennie
  - A ciklus előtt is kell olvasni

Végjeles sorozat

Változók: $X : T$

BE: X
CIKLUS AMÍG X *nem végjel*
X *feldolgozása ...*
BE:X
CIKLUS_VÉGE

Előreolvasás

- Általános technika: a ciklusfeltételhez szükséges adatokat a ciklus előtt elő kell állítani, különben „még nem kapott kezdeti értéket” hiba van
  - Ez akár az első néhány elem előreolvasását is jelentheti, ha a végjel úgy van megfogalmazva
- Hátránya, hogy a beolvasás többször szerepel a kódban

Végjeles sorozatokra példa

- Szöveges állományokban részsorozatoknál bevett módszer üres sorral jelezni, hogy vége a sorozatnak, pl. .srt mozifelirat formátumban
- Bizonyos kódolásokban létezik „üzenet vége” karakter
- A Morse kód kiterjesztésében is van befejezést jelző kód
- Kisebb programoknál, saját formátumoknál kedvelt forma az egyszerűsége miatt

Fájlok

- A fájl névvel azonosított adattároló
- Általában
  - vagy olvasunk belőle, vagy írunk bele
  - az olvasott fájlok tartalma használat közben nem változik meg
  - hasonló a viselkedése, mint a „sima” kimenetnek és bemenetnek
  - lekérdezhető, hogy vége van-e
- Sokféle rendszer van, sokféle nyelv, sokféle kontextus mindegyikre van kivétel

Fájlok és a PLanG

- PlanGban a fájlok nem jelennek meg az operációs rendszer fájlrendszerében, virtuális fájlokról van szó
- Használat előtt meg kell nyitni a fájlt, megadva a nevét, utána pedig illik lezárni
- Ha a PlanG kódban megjelenik egy megnyitás, a bemenet és kimenet fülek bővülnek
- A „sima” BE: és KI: mintájára használható az olvasás és az írás, ugyanolyan működésűek

Fájlok és a PLanG

```
PROGRAM fájlos
  VÁLTOZÓK:
    fb: BEFÁJL,
    n: EGÉSZ

  MEGNYIT fb: "olvasnivalo"
  BE fb: n
  KI: n
  LEZÁR fb
PROGRAM_VÉGE
```

Fájlok

- A fájlok kezelése sokban hasonlít a végjeles sorozathoz a fájl végének kezelésében
- A nyelvek kétféle stratégiával dolgoznak,
  - vagy akkor ad igazat a „vége van a fájlnak?” kérdés, ha már nincs több olvasnivaló elem,
  - vagy akkor, ha már legalább egyszer próbáltunk olvasni sikertelenül
    - Nyilvánvalóan ez utóbbi helyzetben a sikertelenül olvasás mellékhatásaként a beolvasott változó tartalma nem a sorozat része, tehát nem szabad feldolgozni, ahogy a végjelet sem szabad

Fájlok és PLanG

```
PROGRAM fájlos-sorozatos
  VÁLTOZÓK:
    fb: BEFÁJL,
    n: EGÉSZ

  MEGNYIT fb: "olvasnivalo"
  BE fb: n
  CIKLUS AMÍG NEM VÉGE fb
    KI: n, SV
    BE fb: n
  CIKLUS_VÉGE
  LEZÁR fb
PROGRAM_VÉGE
```

Fájlok

- Lekérdezhető a fájl vége
- Egyszerre több fájlból olvashatunk, ezek egymástól függetlenül lépnek előre
- A „kifájl” és „befájl” típusú változók szerepe számon tartani, hogy hol tartunk a fájlban
  - ezért nem elég csak a fájl nevét írni olvasáskor
- Ipari nyelveknél fájl írásakor a lezárás elmulasztása veszteséget okozhat: mindig minden fájlt zárjunk le, még ha nem is fordítási vagy futásidejű hiba ennek elhagyása

Fájlok

- Ha egy fájlt lezárunk és újra megnyitunk, akkor elölről kezdődik az olvasás.
  - Ennek kihasználása nem szép dolog, de ha egy feladat nem elemenként feldolgozható, és túl sok adat van tömbhöz* akkor szükségmegoldásnak megteszi
  - A félév házi feladatai, és géptermi ZH feladatai nem ilyenek

* lásd jövő héten

Fájlok, műveletek összefoglalás

- **KIFÁJL, BEFÁJL**
  - Típusok fájlok kezeléséhez, ezeken keresztül érjük el az adatokat
- **Megnyit** *f*: ”*fájlnév*”
  - Fájlnév rendelése a KIFÁJL/BEFÁJL változóhoz
- **KI** *f*: X / **BE** *f*: X
  - Írás/olvasás a megadott fájlba/fájlból
- **VÉGE** *f*
  - Logikai kifejezés, értéke: „Olvastunk-e már sikertelenül f-ből?”

Fájl kezelése általában PLanGban

```
VÁLTOZÓK:
   fb: BEFÁJL,
   a: T

MEGNYIT fb: "fájlnév"
BE fb: a
CIKLUS AMÍG NEM VÉGE fb
   a feldolgozása
   BE fb: a
CIKLUS_VÉGE
LEZÁR fb
```

Példa: Maximumkeresés fájlra

```
PROGRAM fajlos
  VÁLTOZÓK:
    fb: BEFÁJL,
    a, max: VALÓS,
    i, hol: EGÉSZ

  MEGNYIT fb: "olvasnivalo"
  i := 1
  BE fb: a
  max := a
  hol := 1
  CIKLUS AMÍG NEM VÉGE fb
    HA a > max AKKOR
      max := a
      hol := i
    HA_VÉGE
    i := i + 1
    BE fb: a
  CIKLUS_VÉGE
  KI: hol, ".: ", max
  LEZÁR fb
PROGRAM_VÉGE
```

Példa: Maximumkeresés fájlra

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

i := 1
a := első elem
max := f(a)
hol := 1
CIKLUS AMÍG tart még a sorozat
  a := következő elem
  i := i + 1
  HA max < f(a) AKKOR
    max := f(a)
    hol := i
  HA_VÉGE
CIKLUS_VÉGE
```

```
PROGRAM fajlos
  VÁLTOZÓK:
    fb: BEFÁJL,
    a, max: VALÓS,
    i, hol: EGÉSZ

  MEGNYIT fb: "olvasnivalo"
  i := 1
  BE fb: a
  max := a
  hol := 1
  CIKLUS AMÍG NEM VÉGE fb
    HA a > max AKKOR
      max := a
      hol := i
    HA_VÉGE
    i := i + 1
    BE fb: a
  CIKLUS_VÉGE
  KI: hol, ".: ", max
  LEZÁR fb
PROGRAM_VÉGE
```

Formátumok

- Beolvasás egyértelműsége
- Technikai megkötések
  - A PLanG szám beolvasásakor nem tud különbséget tenni szóközzel elválasztás és sorvégével elválasztás között, tehát az a forma, hogy az összetartozó számok egy sorban vannak, és a következő sor már megkülönböztetendő, nem jó formátum PLanG-ban
  - A szöveg beolvasása soronként történik, tehát két megkülönböztetendő szöveget ne írjunk egy sorba

Példa formátumra

- Feladat kezelni egy középiskolai osztályt egyik tanár szemszögéből: van sok diák, mindegyiknek valahány jegye
- Fájlban két sor ír le egy diákot
  - az első sor a neve
    - Mivel ha egy sorban lennének a jegyek is, a név beolvasásakor az is belekerülne a szövegbe, és az lenne a név, hogy „Gipsz Jakab 5 3 4 5 ...”
  - a következő sorban vannak a jegyek, és a végén -1
    - Ez megtehető, mert a jegy eleve szűk értékkészletű, így az általánosság megszorítása nélkül vehetünk fel végjelet az egész számok közül
    - Az is jó lenne, ha a jegyek számával kezdődne a sor

Néhány elterjedt egyszerű formátum

- .sub mozifelirat: elválasztó karakterek között a kezdeti képkocka és a befejező képkocka sorszáma, aztán a sor végéig az itt kiírandó szöveg
- .ini: [] jelek között külön sorban fejezetcím, majd kulcs „=” érték párok, sem a kulcs, sem az érték nem tartalmazhatja a '=' karaktert
- .csv: táblázatformátum, melyben vesszővel elválasztva vannak a mezők, a szöveges mezők „” jelek között szerepelnek

Összehasonlítás

	Ismert hosszú	Végjeles	Fájl
Teljes értékkészlet	☺		☺
Beszúrással bővíthető		☺	☺
Előre olvasást igényel		☺	☺
Nyelvtől független	☺	☺	