Bevezetés a programozásba

5. Előadás: Tömbök, számábrázolás

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

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
```

ISMÉTLÉS

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
```

ISMÉTLÉS

Végjeles sorozat

Változók: $X : T$

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

ISMÉTLÉS

Fájl kezelése általában PLanG-ban

```
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
```

Mi a közös az eddigiekben?

- Tetszőlegesen hosszú sorozat feldolgozása
- Kevés (max 5-6) változóval
- Kizárólag elemenként feldolgozható feladatok
  - Minden elemet pontosan egyszer kezelünk
  - A sorrend kötött: csak abban a sorrendben tudunk olvasni, ahogy a sorozatban egymást követik
- Ezeknek a fő oka, hogy a feldolgoznivaló adatsor sokszor nem fér a memóriába, tehát spórolni kell a memóriaigénnyel

A tömb

- A tömb egy típuskonstrukció: egy egyszerű típust megsokszorozunk, adott hosszú sorozatát, mint típust kezeljük
- A tömb típusú változót közvetlenül ritkán, inkább a hordozott sorozat egy-egy tagját kezeljük
- **`VÁLTOZÓK: a: EGÉSZ[10]`**
  `a[2]:=3` **$a_2=3$**
- Lényegében (nullától) indexelt, egyforma típusú változók egységes kezeléséről van szó

Példa tömbre

```
PROGRAM tömb
  VÁLTOZÓK:
    a: EGÉSZ[10],
    i: EGÉSZ

  i := 0
  CIKLUS AMÍG i < 10
    a[i] := RND 5 + 1
    i := i + 1
  CIKLUS_VÉGE
PROGRAM_VÉGE
```

Tömbök

- A tömb szabadon címezhető, tetszőleges sorrendben bejárható, átírható, kezdeti értéket nem tartalmazó változósorozat, ellentétben a bemeneten kapott sorozatokkal
- Viszont előre, fordítási időben tudni kell a méretét.
  - Hallgatók rendszeresen abba a hibába esnek, hogy mindent tömbökkel akarnak megoldani. Ennek tipikus jele, hogy sorozatot akarnak beolvasni egy 1000 méretű tömbbe..

Tömbök

- A tételek tömbökön is alkalmazhatóak, mivel ismert hosszú sorozatokról van szó, ahol az adott elem a sorszámával hivatkozható

```
...
sum := 0
i := 0
CIKLUS AMÍG i < 10
  sum := sum + a[i]
  i := i + 1
CIKLUS_VÉGE
...
```

Többdimenziós tömb

- A tömb szintaxisa szerint T[méret] a tömb, ahol T tetszőleges típus
- A tömb is típus
- Tehát a T[méret1][méret2] is helyes, és tömbök tömbjét jelenti
- Ez kétdimenziós tömb, két független indexet lehet használni, mint koordinátákat
- Természetesen tetszőlegesen fokozható a dimenziószám – elméletben. Gyakorlatban kifogyunk a memóriából.

Tömb: vektor, mátrix

- Az egydimenziós tömböt szokás vektornak, kétdimenziósat mátrixnak nevezni

```
VÁLTOZÓK:
  a: EGÉSZ[10][10],
  i, j: EGÉSZ

i := 0
CIKLUS AMÍG i < 10
  j := 0
  CIKLUS AMÍG j < 10
    a[i][j] := RND 5 + 1
    j := j + 1
  CIKLUS_VÉGE
  i := i + 1
CIKLUS_VÉGE
```

Tömbök jelentősége

- Olyan feladatoknál, ahol
  - több adatra van szükség annál, mintsem külön változókban kényelmes lenne kezelni, fix számú, és még beleférünk a memóriába
  - többször kell kiértékelni ugyanazt az értéket
  - tetszőleges sorrendben kell hozzáférni az elemekhez
  - az adatokat módosítani is kell, nem csak olvasni

  .. rákényszerülünk a tömb használatára

Néhány példa

- Képkezelő szoftverek a képet mátrixként tárolják
  - konkrétan a FF szürkeárnyalatos képeket általában egész számok mátrixával ábrázolják
- Hagyományos mátrixműveletek, Gauss elimináció, bázistranszformáció
- Táblázatos adatok
- A szöveg típus néhány művelettől eltekintve felfogható karakter-vektornak

Tömbök és a PLanG

- A tömb típusú változók kiírathatóak, de nem beolvashatóak
- A változók értékeit mutató táblázatban az egész tömb követhető
- A tömbelemek kezdeti érték nélkül jönnek létre, erről gondoskodni kell

Összefoglaló

- Tömbök: fix hosszú homogén változósorozat
- Szintaxis: Típus[méret]
- Akkor és csak akkor használandó, ha ismert hosszú, többszöri írást vagy olvasást, vagy tetszőleges sorrendben feldolgozást igényel a feladat
- Lehet több dimenziós

Rövid kitérő a számábrázoláshoz

- Az „EGÉSZ” illetve „VALÓS” típusok nevei azt a látszatot keltik, hogy a típusértékhalmaz a teljes egész illetve valós számokat fedi
- Ez természetesen lehetetlen, véges hosszú memóriaszeletek állnak rendelkezésre a számok ábrázolásához
- A fenti típusok tehát nem mindenben viselkednek a várakozásoknak megfelelően

Rövid kitérő a számábrázoláshoz

- Például az EGÉSZ 32 bites szám, tehát legfeljebb 2^{32} féle értéket vehet fel. Ezt praktikusan a -2^{31} .. $+2^{31}-1$ tartományra tolták: -2147483648 .. 2147483647
- → kettes komplemens kódolás
- Ennek az a következménye, hogy 2147483647 + 1 = -2147483648
- Ezt a jelenséget túlcsordulásnak nevezik

Kettes komplemens kódolás

Túlcsordulás:

1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0

·Használjuk ki a túlcsordulást!
Melyik az a szám, amihez 1-et adva 0-t kapunk?
Jelentse tehát a csupa 1-es bit a -1-et.

Következmény: az első bit az előjel, és bármely pozitív szám negáltját a bitek megfordításához (komplemenséhez) egyet adva kapjuk

Rövid kitérő a számábrázoláshoz

- A valós számok véges ábrázolása miatt pontatlan
- Számológépről ismerős lehet: 3 gyökének a négyzete 2.99999999
- Ebből kifolyólag soha nem vizsgáljuk programban egy valós típusba tartozó változó egyenlőségét semmivel

Rövid kitérő a számábrázoláshoz

- A valós számokat $X * 2^Y$ alakban tárolják, visszavezetve az egész számokra
- Pl 32 bites esetben 23 bit X-re, 8 bit Y-ra, plusz előjel
- Ennek következménye, hogy nem mindenhol egyformán sűrű az ábrázolás, ha 23 kettedesjegynél nagyobb a nagyságrendi különbség, akkor előfordulhat, hogy $A \neq 0$, de $A+B=B$

Itt a vége a PLanG képességeinek, legközelebb a C++ alapoktól folytatjuk