

OKTATÁSI MINISZTERIUM

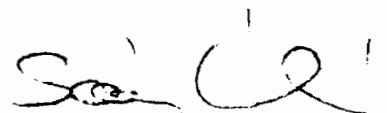
Szakmai írásbeli vizsgatétel megoldása

M

Szakképesítés: 54 4641 04 Számítástechnikai programozó
(azonosító száma, megnevezése)

Tantárgy: Komplex feladat

Jóváhagyta:



Soós László
osztályvezető

2003. 12.



NEMZETI SZAKKÉPZÉSI INTÉZET

1. A Pascal programozási nyelv

A nyelv jellemzői:

- Amatőr, Neumann-elvű.
- Program = 1 fordítási egység = több programegység, blokkstruktúra.
- Statikus deklaráció kiértékelés.
- Dinamikus memóriakezelés (statikus és véges élettartamú változók), van kézi lefoglalás, törlés is, sokféle élettartam.
- Hatáskör, láthatóság: statikus hatáskör kijelölés.
- Paraméterátadás: cím szerinti, érték szerinti (eljárásparaméter).
- Típuskompatibilitás név szerint.
- Azonosítók nem lapolhatók át.
- Nem erősen típusos (1973 szerint), erősen típusos (1983 szerint).
- Alapszavak védettek.
- Nincs sor=utasítás megfeleltetés.
- Utasítás elválasztó jel a pontosvessző.
- Mindent előre kell definiálni (alulról felfelé építkezés).

A program szerkezete:

```
PROGRAM <név>;
[<deklarációk>]
BEGIN
[<utasítások>]
END.
```

Deklarációk: (a szabvány szerint kötött sorrendben, TP szerint nem)

- címkedeklarációk,
- konstansdefiníciók,
- típusdefiníciók,
- változódeklarációk,
- eljárás- vagy függvénydeklarációk.

2.

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Bizonyítás

$$\frac{(n-1)(n-2)\dots 1}{(k-1)(n-1-k+1)!} = \frac{(n-1)!}{k!(n-1-k)!} = \frac{k(n-1)! + (n-k)(n-1)!}{k!(n-k)!} = \frac{(n-1)!(k+n-k)}{k!(n-k)!} = \frac{n!}{k!(n-k)!} = \binom{n}{k}$$

3.

Virtuális tárkezelés: A programvégrehajtás alapproblémája, hogy a mindenkor, végrehajtás alatt álló programrésznek és a feldolgozandó adatoknak a központi tárban kell lenniük. Ugyanakkor a főtár mérete nem teszi lehetővé, hogy a teljes program és az összes adat egyszerre a tárban legyen. A központi tár véges kapacitása miatt, az aktuálisan nem használt programokat, adatokat valamilyen háttértárolón kell tárolni, és szükség esetén betölteni a főtárba. A háttértárolón lévő programrészek központi tárba való betöltésére a programozók kedvelt módszere az ún. **overlay technika**, amelynek az újonnan betöltött programrész egy

másik, már nem használt rész helyére kerül. Ennél a módszernél a programozónak kell gondoskodnia az adott rész megfelelő időben történő betöltéséről és elhelyezéséről, és az átlapolásos program futtatása is igen körülményes. Ezért dolgozták ki az „automatikus overlay” technikát, a **lapozásos tárolókezelési módszert**. Ez eleinte az operációs rendszer része volt, később teljesen átkerült a processzor hatáskörébe. Az automatikus lapozás alkalmazásával lehetővé vált az is, hogy a teljes rendelkezésre álló tárolóterületet, vagy legalábbis annak nagy részét lefedő területet, látszólag közvetlenül címezhesük, azaz egy egységnek, látszólagos központi tárnak tekintsük. Ezt a címzési lehetőséget, módszert nevezik **virtuális címzésnek**, tárkezelésnek. Az így rendelkezésre álló címtartományt **virtuális címtartománynak** nevezik.

4. Egy adott kapcsolatnál (kommunikációnál) használt szabályok és megállapodások összessége.
5. **Task fogalma:** Task-nak nevezzük azt a tevékenységet, mely a számítógépben más tevékenységekkel elvben párhuzamosan folyhat. Minden egyes task egy utasításokból és kiinduló adatokból álló programot futtat. Tipikus task-ok: file szerkesztése, forrás file fordítása stb. Természetesen ezek után ugyanazt a programot esetleg több task is futtathatja (például egy időosztásos rendszerben a számítógép egy meghatározott időtartamig, mondjuk 10 ms-ig az A felhasználó forrásprogramját, majd a B felhasználó forrásprogramját fordítja és í.t., azaz ugyanazt a compiler-t használja). A **virtuális processzor fogalma:** A rendszernek ugyanis az egyes task-ok számára átlátszónak kell lennie. Ez azt jelenti, hogy *minden task-nak (felhasználónak) úgy kell éreznie, hogy saját külön processzor áll rendelkezésére*. Más szavakkal: egy task elméletileg teljesen párhuzamosan fut a többi task-kal (eltekintve attól az esettől, amikor több task valamilyen közös cél érdekében együttműködik, mert ilyenkor az egyik task-nak olyan kiindulási adatra van szüksége, melyet egy másik task futtatásának eredményeként állít elő, azaz az egyik task működését egy másik task működéséhez kell szinkronizálni). A valóságban azonban az egyes task-ok ugyanazon processzoron (*általánosabban fogalmazva: ugyanazon erőforrásokon*) osztoznak. A multitasking operációs rendszer valamilyen időtartamra a fizikai processzort az egyik kiválasztott task számára rendelkezésre bocsátja. Más szavakkal: a multitasking operációs rendszer minden egyes időpillanatban a fizikai processzort hozzárendeli az egyik virtuális processzorhoz. Az egyes multitasking operációs rendszerek különböző módszerekkel választják ki, hogy a futásra váró task-ok közül melyik fusson egy adott időpillanatban. **Time-sharing fogalma:** Az ún. *time-sharing (időosztásos)* operációs rendszernél az operációs rendszer ütemezője periodikusan egy előre meghatározott rövid időtartamra az A, majd a B és í.t. task-ot futtatja.
6. **Relációs kulcs:** Az A attribútum halmaz egy K részhalmazát kulcsnak nevezzük, ha a K értékei az R reláció mindegyik sorát egyértelműen meghatározzák, de ha egyetlen attribútumot is elhagyunk K-ből, akkor ez már nem teljesül. Egy reláció kulcsa tehát olyan attribútum-csoport, amelyeken az attribútumok értékei egymástól különböznek, vagyis nincs két sor, amelyekben a kulcsban szereplő attribútumok értékei azonosak volnának. Ha K egyetlen attribútumból áll, akkor a kulcsot egyszerűnek, ha nem ilyen, akkor összetettnek nevezzük. Nyilvánvaló, hogy egy relációban mindig van kulcs. Az is világos, hogy egy relációnak több kulcsa is lehet. A reláció attribútumai közül azokat, amelyek legalább egy kulcsban szerepelnek, elsődleges attribútumoknak, a többieket másodlagosnak nevezzük. Külső kulcsnak (vagy idegen kulcsnak) nevezzük egy relációnak azokat az attribútumait, amelyek egy másik relációban kulcsot alkotnak. A külső kulcsot szaggatott vonallal húzzuk alá.

7. Bináris (logaritmikus) keresés algoritmus:

```

Eljárás BinárisKeresés (N:egész, A:tömb(1..N):<elemtípus>,
Elem:<elemtípus>, Változó IND:egész)
Változó E,U,K:egész
E:=1:U:=N:K:=INT((E+U)/2)
Ciklus amíg E<=U és A(K)<>Elem
  Ha Elem<A(K)
    akkor U:=K-1
    különben E:=K+1
  Elágazás vége
  K:=INT((E+U)/2)
Ciklus vége
Ha E>U akkor IND:=NULL különben IND:=K
Eljárás vége

```

8. Tesztípusok:

- Elfogadhatósági teszt: a specifikáció ellentmondásmentes-e, tartalmaz-e hiányosságokat
- Funkcióteszt: szerepel-e a programban az összes megvalósítandó funkció
- Biztonsági teszt: a program ellenőrzi a felhasználótól kapott adatokat, hibás adatok bevihetők-e
- Stressz teszt: nagy sebességgel érkező adatok esetén is megfelelő sebességű-e a működés
- Volumen teszt: nagy adathalmazzal is helyesen működik-e a program
- Hatékonysági teszt: végrehajtási idő, helyfoglalás
- Modulteszt: önálló modulok tesztelése a többi modultól függetlenül
- Összeépítési teszt: az önálló modulok összeépítésének helyessége

9. Mátrix determinánsa: -36

10. Pascal és C nyelven

Pascal	Turbo C
<pre> case <kifejezés> of <lista> : <utasítások>; <lista> : <utasítások>; <lista> : <utasítások>; else <utasítások> end; </pre>	<pre> switch (<kifejezés>) { case <tétel> : <utasítások> case <tétel> : <utasítások> case <tétel> : <utasítások> default : <utasítások> } </pre>

Értékelés:

0	-	60	pont	elégtelen
61	-	70	pont	elégséges
71	-	80	pont	közepes
81	-	90	pont	jó
91	-	100	pont	jeles

A megoldási útmutatóban a feladatok egy lehetséges megoldását adtuk meg. A megoldási útmutatótól eltérő jó megoldást is el kell fogadni!