

# INFORMATIKAI ÉS HÍRKÖZLÉSI MINISZTERIUM

Szakmai írásbeli vizsgatétel megoldása

# M

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

Tantárgy: Írásbeli feladat

Jóváhagyta:

*Borbély László*

2004 . 02 .



## NEMZETI SZAKKÉPZÉSI INTÉZET

A megoldási útmutatóban a feladatok egy lehetséges megoldásai vannak megadva. Az megoldási útmutatótól eltérő algoritmusleíró eszközt, programozási nyelvet stb. használó, a megoldási útmutatóban megadott megoldással egyenértékű (ugyanazt az eredményt, algoritmust stb. produkáló) megoldást is el kell fogadni.

## 1. Pascal programozási nyelvben

### Szekvencia megvalósítása a Pascal nyelvben

Az összetett utasítás nem más, mint utasítások csoportba fogása, mellyel lehetőség nyílik arra, hogy sok utasítást egy utasításként kezeljünk. A csoporton belül az utasítások szekvenciálisan követik egymást. Az utasítások a forrásprogramban a *Begin End* utasítások közé kell tenni.

pl.

```
Begin
  U1;
  U2;
  ...
  Un;
End;
```

### Szelekció megvalósítása a Pascal nyelvben

A szelekció válogatást jelent. Előre elgondoljuk, hogy milyen esetek következhetnek be, és minden esetre megadunk egy tevékenységet vagy tevékenység sorozatot, melyet a programnak végre kell hajtania.

#### Egyágú szelekció - IF ... THEN

A z egyágú szelekció azt jelent, hogy ha igaz a megadott feltétel, akkor a hozzá kapcsolódó tevékenységet végre kell hajtani, egyébként azt ki kell kerülni, és a programot az azt követő közös tevékenységgel kell folytatni.

pl.

```
If F1 Then
  Begin
    U1;
    U2;
    ...
    Un;
  End;
```

#### Kétágú szelekció - IF ... THEN ... ELSE

A kétágú szelekció azt jelenti, hogy ha igaz a megadott feltétel, akkor a hozzá kapcsolódó tevékenységet kell végrehajtani, egyébként a másikat.

pl.

```
If F1
Then
  U1
Else
  U2;
```

#### Egymásba ágyazott IF utasítás

Az IF utasítás bármely ágán egy utasítás áll. S ez lehet egy másik IF utasítás is. Ilyenkor azonban különösen figyelni kell a tiszta programszerkezetre, mert könnyen elkeveredhetünk.

#### Többágú szelekció -IF ... THEN ... ELSE IF

Vannak olyan esetek, amikor a feltételek egymástól függenek olyan értelemben, hogy a feltételek közül legfeljebb egy teljesülhet. Ekkor, ha az egyik feltétel teljesül, a többit nem kell vizsgálni. Az ilyen programelágazást nevezzük többágú szelekciónak.

**Többágú szelekció - CASE**

Az ELSE IF megoldással bármilyen többágú szelekció lekezelhető. Van azonban egy másik lehetőség a Pascalban, amely áttekinthető és elegáns. Hátránya, hogy csak bizonyos esetekben használható. A CASE utasítás segítségével egy adott kifejezés vagy változó értékéből válogathatunk.

pl.

```
Case Kor of
  0..13: write('Gyere');
  14..17: write('Fiatal');
  18..23: write('Ifju');
  24..59: write('Felnott');
  Else write('Oreg');
End;
```

**Iteráció megvalósítása a Pascal nyelvben**

Az iteráció ismétlést jelent, ilyenkor egy vagy több utasítás újra és újra végrehajtódik.

**Előtesztelő ciklus - WHILE**

Az előtesztelő ciklus esetén a program még a ciklusba való belépés előtt megvizsgál egy feltételt - ezt belépési feltételnek nevezzük -, és ha ez teljesül, akkor a ciklusmag végrehajtódik, egyébként nem. A ciklusmag ismételtlen végrehajtódik, amíg a belépési feltétel teljesül. Ha már nem teljesül, akkor a vezérlés a ciklus utáni utasításra kerül.

pl.

```
While F Do
  Begin
    U1;
    U2;
    ...
    Un;
  End;
```

**Hátulatesztelő ciklus - REPEAT**

Hátulatesztelő ciklus esetén a ciklus magja egyszer mindenképpen végrehajtódik, majd a ciklus végén, hátul történik egy feltételvizsgálat, ami eldönti, kiléphetünk-e a ciklusból vagy sem. Ha nem, akkor újból végrehajtjuk.

pl.

```
Repeat
  U
Until F;
```

**Növekményes ciklus -FOR**

Növekményes (vagy számláló) ciklus a ciklusmagot egy előre meghatározott számszor hajtja végre. Nem közvetlenül a ciklusok számát adjuk meg: Pascal-ban a ciklus egy bizonyos változó egymás utáni értékeire hajtódik végre. A ciklust "kísérő" változót *ciklusváltozónak* nevezzük.

pl.

```
For CV := Kezd TO Veg Do
  U;
```

## 2. Függvény és eljárás

Az *eljárások és a függvények* - közös nevükön alprogramok- a program önálló részei. Ezeket egy külső blokkból meg lehet hívni, eljárás utasítással. Ekkor az alprogram végrehajtásra kerül, majd a végrehajtás folytatódik, annál az utasításnál, amely az alprogram meghívását követi. A függvényeljárást az különbözteti meg az eljárástól, hogy van visszatérési értéke, melyet végrehajtása után visszaad a blokknak, amelyből meghívtuk. Az alprogramok *paramétereken* keresztül kommunikálhatnak környezetükkel, melyeken keresztül bizonyos értékeket kaphatnak, majd a végrehajtás után bizonyos paraméterein keresztül eredményeket adhatnak vissza. Az eljárásfej deklarációjakor a paraméterlistában résztvevő változókat *formális paramétereknek* nevezzük. Az *aktuális paraméterek* pedig, azok amelyekkel meghívjuk az eljárást. Kétféle paraméterátadás létezik a Turbo Pascalban, a *cím szerinti és az érték szerinti*. A *cím szerinti* paraméterátadáson azt értjük, hogy a külső blokk aktuális paraméterei és az alprogram nekik megfelelő formális paraméterei a memóriában páronként azonos helyen találhatók. Ez azt jelenti, hogy a paraméterek a külső ill. a belső blokkokban fizikailag ugyanazokat a változókat

takarják, a tárban is ugyanazon a címen található. A paraméterátadás másik módja az *érték szerinti* paraméterátadás. Ez szemléletesen azt jelenti, hogy az aktuális és a formális paraméterek külön helyet foglalnak a tárban, más változókat jelentenek, amelyeknek azonban darabszámra és típusra szigorúan egyezni kell.

### 3. Állományok kezelésének alaptevékenységei

*Fizikai állománynak* nevezzük a másodlagos tárolón elhelyezett adatok önálló névvel ellátott halmazát.

Ez az önálló név lemezen az állomány specifikáció.

*Másodlagos tárnak* nevezzük azokat a számítógéphez kapcsolt egységeket, amelyek alkalmasak adatok és programok hosszú távú tárolására. A leggyakrabban használt címezhető külső tároló a mágneslemez.

*Logikai állománynak* az egyed előfordulások önálló névvel ellátott halmazát nevezzük, mely olyan tulajdonságtípusok előfordulásait tartalmazza, mely egy adott feladat szempontjából lényeges.

#### A logikai állomány leképezése fizikaira

A fizikai állományra való leképezést egy adott hardver illetve szoftver környezetben végezzük.

A leképezés eredményeként nem biztos, hogy egy logikai állománynak pontosan egy fizikai állomány feleltethető meg. A gyors és ésszerű feldolgozás érdekében lehet, hogy a logikailag összetartozó dolgokat fizikailag szét kell választani vagy épp fordítva.

#### Általános fájl-kezelés

A fájlhoz annak deklarációját egy azonosítót rendelünk. Ezt fájl-változónak nevezzük. Az e módon definiált *logikai fájl*-t egy *fizikai*, a háttértárban létező fájlhoz kell hozzárendelni az Assign eljárással. A fájl-változóval végrehajtott minden művelet ezután erre a fizikai fájlra fog vonatkozni. A fájl megnyitását Reset vagy Rewrite végzi, lezárását Close.

#### A Turbo Pascal által kezelt adatállományok fajtái.

- tipizált
- tipizálatlan
- szöveg fájl

### 4. Az adatmodell fogalma

Az **adatmodell** **egyedek**, **tulajdonságok** és **kapcsolatok** halmaza, amely absztrakt módon tükrözi a valós objektumoknak, azok jellemzőinek (tulajdonságainak) és viszonyainak (kapcsolatainak) elvont kategóriáit.

#### A hálós adatmodell

A hálós adatmodell szerkezetét **gráffal** adjuk meg. Ebben a gráfban a csomópontok az egyedek, az élek pedig a kapcsolatot fejezik ki. Az egyedeket tulajdonságaival írjuk le.

#### A hierarchikus adatmodell

A **hierarchikus** adatmodell szerkezetét szintén gráffal adjuk meg, de a gráf speciális, nevezetesen **fa**.

#### A relációs adatmodell

Ez a modell más filozófiára épül, mint az előbbi kettő. Ennél az adatmodellnél a három adatmodell-elem közül a kapcsolat nem játszik szerepet, pontosabban szólva a kapcsolat nem, csak a lehetőség épül be az adatmodellbe. A relációs adatmodellnél a tulajdonságok kapják a fő hangsúlyt, a tulajdonságokkal definiáljuk az adatmodell szerkezetét.

Az egyedeket egy **táblázattal** adjuk meg, a táblázat oszlopai a tulajdonságok. A táblázat sorai az egyed értékei.

A relációs adatmodellben létrehozott adatbázisokat több táblázattal adjuk meg (ne felejtjük el, minden tábla egy egyedhalmaz), de a táblázatok közötti kapcsolatokat nem definiáljuk az adatmodell felírásakor. Ez nem azt jelenti, hogy nincsen közöttük kapcsolat, de ezeket a kapcsolatokat például egyszerűen az fejezi ki, hogy a két táblának van közös oszlopa.

### 5. A SELECT parancs végrehajtásának eredményeként egy új tábla keletkezik, a továbbiakban ezt **eredménytáblának (E-táblának)** hívjuk.

Általános szerkezete a következő:

SELECT...	oszlop kiválasztása = projekció
[INTO...]	az E-tábla 1. sorának tárolása
FROM...	táblák Descartes-szorzata
[WHERE...]	sorok kiválasztása = szelekció

[GROUP BY...]	csoportosítás
[HAVING...]	csoportok közötti választás
[UNION...]	E-táblák összefűzése = az unió művelete
[ORDER BY/FOR UPDATE OF.]	E-tábla rendezése/tábla módosítása
[SAVE TO TEMP...];	E-tábla megőrzése, elmentése

### A SELECT...FROM alparancs

*Formátuma:*

```
SELECT [ALL/DISTINCT] oszlopnév-lista | *
FROM táblalista;
```

*Hatása:* A parancs a FROM után szereplő táblákból kiemeli az oszloplistában szereplő oszlopokat, illetve \* esetén az összes oszlopot.

### WHERE alparancs

A WHERE alparancs operandusa egy logikai kifejezés; ennek igazi értéke esetén kiválasztja a rendszer a megfelelő rekordokat.

*Formátuma:*

WHERE feltétel

*Hatása:* Az E-táblába a SELECT után felsorolt oszlopokba azoknak a rokonak az értékei kerülnek, amelyekre teljesül a feltétel.

### GROUP BY alparancs

Sokszor van szükségünk arra, hogy a rekordokat valamelyik oszlop azonos értékei szerint csoportosítsuk, és például összegezzük valamely más oszlop értékeit a csoport soraira.

*Formátuma:*

GROUP BY oszlopnév [,oszlopnév]...

*Hatása:* A megadott oszlop azonos értékei szerint csoportosítja a rekordokat.

### HAVING alparancs

A GROUP BY *oszlopnév* parancs mindegyik oszlopnév értékre elvégzi a csoportosítást. Ha csak bizonyos csoportokat akarunk betenni az E-táblába, akkor a HAVING paranccsal toldjuk meg a GROUP BY parancsot.

*Formátuma:*

HAVING feltétel

*Hatása:* A GROUP BY által kialakított E-táblából kiválasztja azon sorokat, amelyek eleget tesznek a feltételnek.

### ORDER BY alparancs

Sokszor kívánatos, hogy az eredménytábla rendezve legyen egy, vagy esetleg több oszlopa szerint. Ezt az ORDER BY paranccsal tehetjük meg.

*Formátuma:*

```
ORDER BY oszlopnév / oszlop-sorszám [ASC/DESC]
[,oszlopnév / oszlop-sorszám [ASC/DESC]]...
```

*Hatása:* A parancs megadott oszlop vagy oszlopok szerint rendezi az E-táblát, mégpedig növekedőleg, ha ASC-t írunk és csökkenőleg DESC esetén.

6. Ebben az esetben a vonalat nem konkrétan egy adóhoz és egy vevőhöz rendeljük hozzá, hanem az állomások a kommunikáció szükséglete alapján jutnak hozzá. Ezt a megoldást nevezik vonalkapcsolásnak. Abban az esetben, ha adatátvitelre van szükség, kialakítanak egy olyan vonalat, amely a vevő és az adó pont-pont kapcsolatnak érzékel. A vonal kialakítása kapcsolóközpontok által hajtódik végre. A kommunikáció végén a vonal bontásra kerül. Tegyük fel, hogy az A és a V hoszt között kell információtovábbítást elvégezni. Az A állomás jelzi az 1-es vagy a 2-es állomásnak, hogy kapcsolatot szeretne kialakítani a V jelzésével. Ha az 1-es állomás szabad, akkor az létrehoz egy kapcsolatot az A hoszttal. Ezt követően a következő szinten történik meg a kérelem. Látható az ábrán, hogy az 1-es csak a 4-essel áll kapcsolatban. Amikor képes ez a két állomás kapcsolatot létrehozni, akkor elvégzik ezt a műveletet. Gyakorlatilag ebben az esetben már az A és a 4-es állomás között van pont-pont kapcsolatot. A 4-es már közvetlen kapcsolatban áll a V végponttal, amikor az kész a kapcsolat

kialakítására, létrejön a kapcsolat, de ezzel tulajdonképpen már az A és a V hoszt között alakult ki a kapcsolat. Miután ez létrejött, a V állomás nyugtajelet küld a közbelső hosztok segítségével az A-nak, amire az megkezdí az adatátvitelt. Ennek a műveletnek a befejezése után a vonalat bontani kell annak érdekében, hogy a hosztok közötti vonalak más számára is hozzáférhetők legyenek.

7. **Lézernyomtató:** A lézernyomtatók egy gyenge lézersugárral az elektromosan feltöltött félvezető henger felületére rajzolják a jeleket és a grafikát, majd onnan juttatják a papírra. *A lézersugár eltérítése egy mozgó tükörrel történik, és igen gyorsan végigpásztázza a forgó henger aktuális alkotóját. Mivel a lézersugár átmérője igen kicsire fókuszálható, az elérhető felbontás nagy (tipikusan 600x600 DPI). A fény hatására a henger a megfelelő helyen kiszil, majd a szintén töltéssel rendelkező szilárd festék (toner) ezekről a helyekről lelökődik. A hengeren csak a nyomtatandó helyen marad festék, mely egy ellentétes tér hatására a hengerről a papírra tapad. A porfestékeknek a papírra olvasztását a papír felfűtött hengerek közötti átvezetésével érik el. A hengeren maradt festéket kefék és elektromos erőtér segítségével távolítják el. A henger következő körbefordulásával újabb lap nyomtatása kezdődik.*
8. **Egységbezárás:** Az egybezártság elve azt mondja ki, hogy egy tetszőleges objektum tulajdonságadatai csak az adott objektum metódusaiban láthatók, kívülről (függvényből, eljárásból vagy más osztály metódusaiból) csak az adott osztály metódusai érhetők el.
9. **Holtpont megelőzési módszerek:**
- Egyetlen foglalási lehetőség (One-shot allocation): csak az a folyamat foglalhat erőforrást, amelyik még egyetlen eggyel sem rendelkezik.
  - Rangsor szerinti foglalás (Hierarchical allocation): egy folyamat csak olyan osztályból igényelhet erőforrást, melynek sorszáma magasabb, mint a már birtokolt erőforrások sorszáma.
  - Bankár algoritmus (Banker's algorithm): sohase elégítsünk ki egy igényt, ha az bizonytalan állapotot eredményez.
10. 1111111011100010<sub>2</sub>