Szabó Mária

Programozás – Programtervezés

NSZFI
NEMZETI SZAKKÉPZÉSI ÉS FELNŐTTKÉPZÉSI INTÉZET

A követelménymodul megnevezése:

Informatikai ismeretek

A követelménymodul száma: 1155-06 A tartalomelem azonosító száma és célcsoportja: SzT-017-50

Új Magyarország FEJLESZTÉSI TERV

PROGRAMOZÁS – PROGRAMTERVEZÉS

ESETFELVETÉS – MUNKAHELYZET

A munkahelyén megbízást kap egy program elkészítésére. Készítse el a feladat rendszertervét, határozza meg a fejlesztőfelületet. Készítse el az alkalmazandó algoritmusokat, optimalizálja azokat.

SZAKMAI INFORMÁCIÓTARTALOM

A FELADAT MEGHATÁROZÁSA

1. elemzés

Cél: a feladat helyes megoldás, ehhez az alkalmazandó rendszerterv felvázolása, algoritmusok, adatstruktúrák meghatározása, a megfelelő programnyelv, fejlesztői felület megválasztása.

A feladat pontos megfogalmazása: Az alkalmazandó módszer a felülről lefelé való tervezés. A top-down módszer lényege: lépésről lépésre finomítjuk programunkat. Ehhez pontosan meg kell ismerni a megoldandó feladatot. . A top-down technika során tehát először megfogalmazzuk, mit kell megoldani, majd a feladatot részfeladatokra osztjuk, így a finomítás lépései során eljutunk addig, hogy kialakuljon, hogyan kell a problémát megoldani.

Az alkalmazandó módszerek:

- konzultáció a program megrendelőjével, felhasználóival
- űrlapok kitöltetése
- Az esetlegesen meglévő információs rendszer megismerése
- az input adatok felmérése, azok kapcsolatának, beviteli formátumaik pontos meghatározása
- az output adatok elemzése: milyen adatok ezek, hogyan állítjuk elő, melyeket őrizzük meg ezekből, hogyan tudjuk ezeket előállítani.

Miután megismertük a feladat egészét elkezdhetjük a részek kidolgozását, az algoritmusok elkészítését.

2. Tervezés

A programtervezés feladata: az analízis során összegyűjtött információkat és adatokat alapul véve logikailag véglegesen kialakítsa az adatstruktúrákat és az adatokon manipuláló algoritmusokat. A program tervezése komoly, kreatív tevékenység, mely nagy szakértelmet igényel. Hogy milyen tervezési módszert választ az ember, az a következő dolgoktól függhet: Milyen számítógépre készül a program? Mekkora a megoldandó feladat? Milyen módszerek állnak rendelkezésre? Mik a tervező lehetőségei szoftverekben, felkészültségben. Nem kell egy feladat megoldása során szükségszerűen merőben új megoldásokat kitalálni, támaszkodhatunk már kidolgozott feladatokra. Felhasználhatunk már megoldott, hasonló feladatokat, meghatározva az azonosságokat és természetesen a különbözőségeket. Ezzel a megoldási móddal jelentősen lecsökkenthetjük:

- a feladat megoldására fordított időt
- elkerülhetünk klasszikusnak számító hibákat.

PROGRAMOZÁSI TÉTELEK

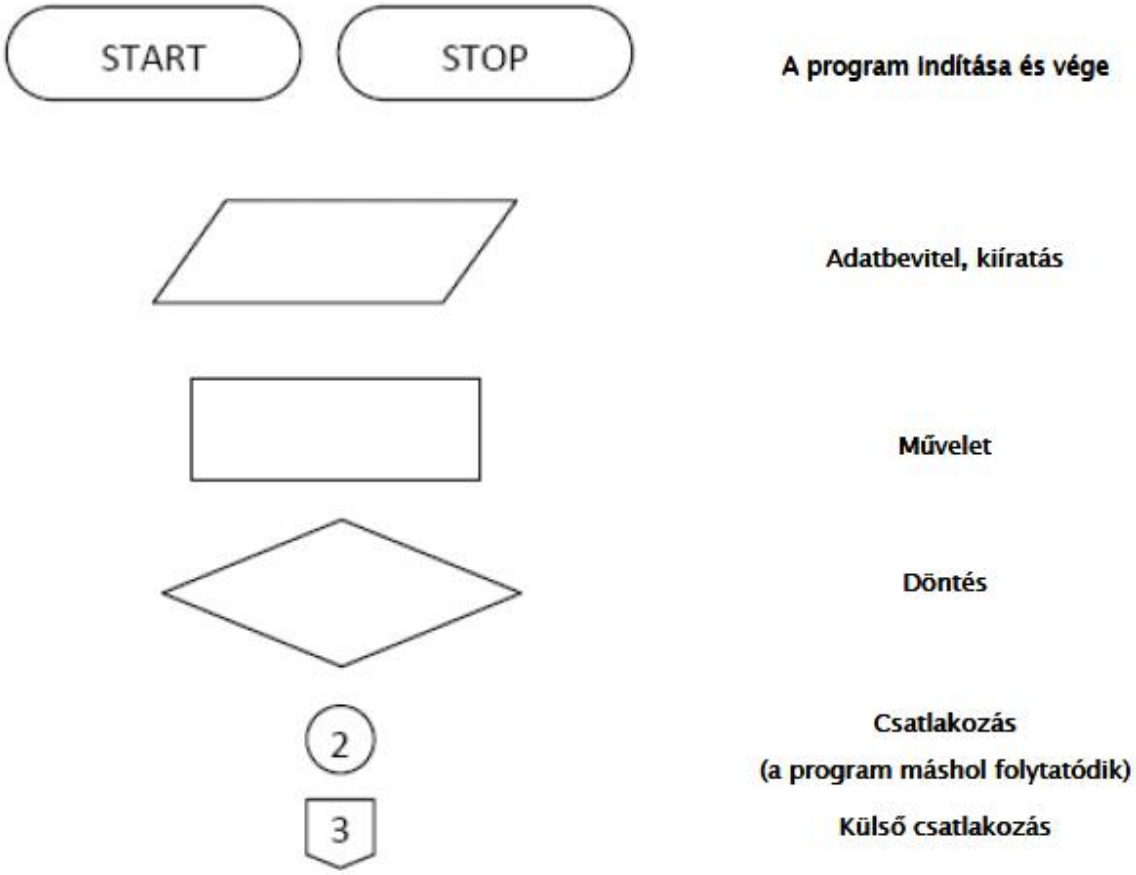
A programozási feladatok részei gyakran olyan problémák, melyeket gyakran alkalmazunk. Így fontos lehet a gyakori és viszonylagosan egyszerű algoritmusok ismerete. Ebben a fejezetben tehát a nyolc legfontosabb programozási tétellel ismerkedünk meg, melyek közös tulajdonsága, hogy tulajdonképpen valamennyi az egész számok egy részhalmazán értelmezett függvény. Ezek a tételek tulajdonképpen nem konkrét feladatok, hanem egy megoldandó problémát fogalmaznak meg, így tulajdonképpen igen sok feladat részei lehetnek. A programozási tételeket általában tömbökkel foglalkoznak, a következőkben az algoritmusok N elemű, tmb tömbökre vonatkoznak.

A tervezésben alkalmazzuk a különféle algoritmus-megadási módszereket. A legfontosabbak ezek közül a következők:

- **Pszeudokód**
  A pszeudokódok (szöveges leírás) az algoritmusok és általában az eljárások leírására használt olyan mesterséges formális nyelvek, melyek változókból és néhány konstansból illetve a folyamatok leírásából állnak.
- **Grafikus ábrázolási módok:**

  Folyamatábra

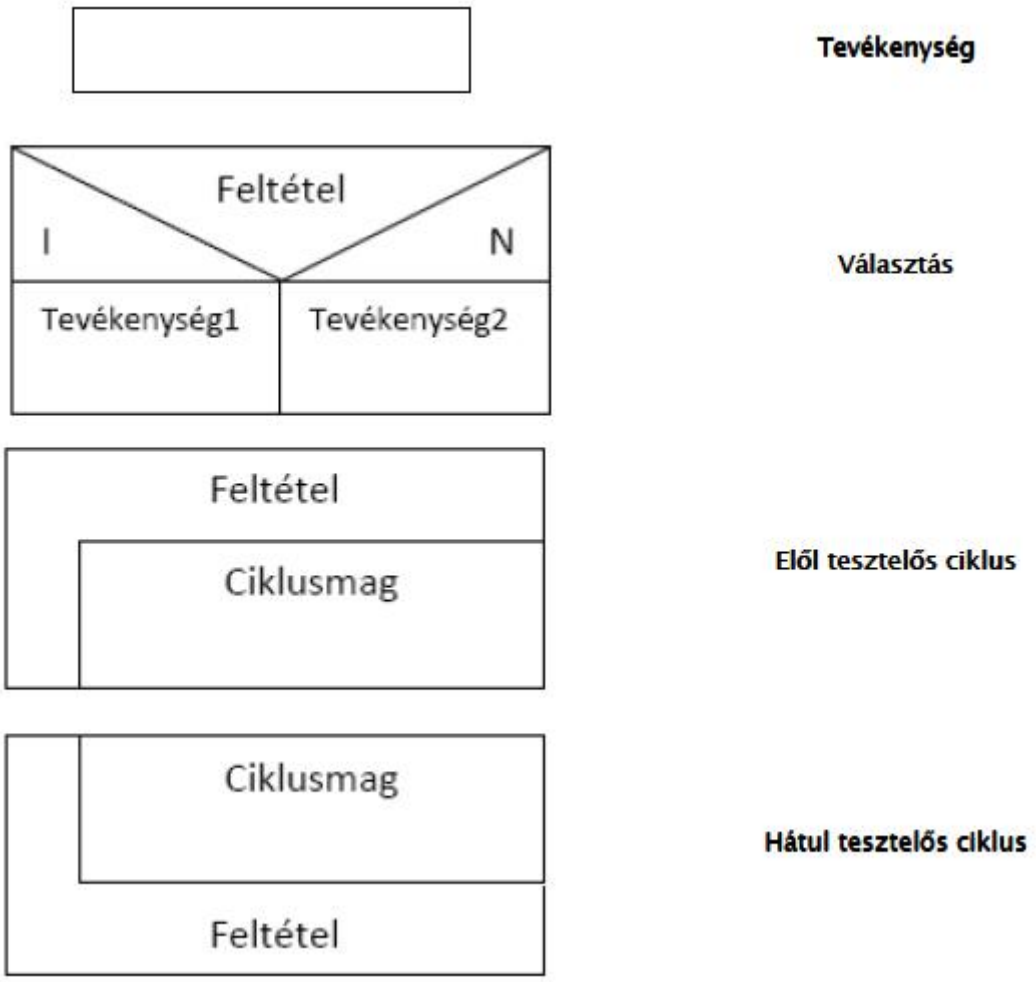
  Elemei a következők:



1. ábra.

Struktogram

A struktogram szintén az algoritmusok grafikus ábrázolására való. Kevésbé szemléletes, mint a folyamatábra, azonban segítségével könnyebb lekódolni az algoritmust. Elemei egymáshoz csatlakozó téglalapok.



2. ábra.

1. Összegzés

Feladat: Határozzuk meg egy N elemű tömb elemeinek összegét!

```
szum=0
Ciklus i=1-től N-ig ismétel
        szum=szum+tmb[i]
Ciklus vége
Ki:szum
```

2. Számlálás

Feladat:Határozza meg, hogy a tmb tömbnek hány darab adott tulajdonságú eleme van!

```
szum=0
Ciklus i=1-től N-ig ismétel
    Ha tmb[i] adott tulajdonságú
        akkor szum=szum+1
Ciklus vége
Ki: szum
```

3. Maximum (minimum) kiválasztás

Feladat: A tmb tömb legnagyobb elemének a kiválasztása.

```
Maximum=1
Ciklus i=2-től N-ig ismétel
    Ha tmb[i]>tmb[Maximum] akkor Maximum=i
Ciklus vége
Ki: Maximum, tmb[Maximum]
```

4. Eldöntés

Feladat: Az algoritmus eldönti, hogy van-e a tömbben adott tulajdonságú elem. Amint talál egyet, a ciklus leáll. Ha a ciklus azért állt le, mert túlléptünk a tömb utolsó elemén is, akkor nem volt benne keresett elem.

```
i=1
Ciklus amíg i<=N és tmb[i]<>keresett elem
    i:=i+1
Ciklus vége
Ha i<=N akkor
    ki:"volt a tömbben a keresett elem"
Különben
    ki:"nem volt a tömbben a keresett elem"
```

5. Kiválasztás

Feladat:Határozza meg, hogy a tömb egy elme (amely biztosan benne van a tömbben) hol, azaz hányadik helyen van a tömbben!

```
i:=1
Ciklus amíg tmb[i]<>keresett elem
    i=i+1
Ciklus vége
ki: i
```

6. Keresés

(a) Feladat: megadja, hogy van-e olyan elem amelyet keresünk a rendezetlen tömbben, és ha igen, hányadik. A keresést szokásos lineáris keresésnek is nevezni.

```
i=1
Ciklus amíg i<=N és tmb[i]<>keresett elem
    i=i+1
Ciklus vége
Ha i<=N akkor
    ki: i
különben
    ki: A tömbben nincs a keresett elem
```

(b) Feladat: megadja, hogy van-e olyan elem amelyet keresünk az N elemű tmb rendezett tömbben. A keresést szokásos logaritmusos keresésnek is nevezni.

```
A=1
F=N
Ciklus
    K:=INT((A+F)/2)
    Ha tmb(K)<keresett_elem akkor A=K+1
    Ha tmb(K)>X akkor F:=K-1
```

```
        Amíg A<=F és tmb(K)<>keresett_elem

        Ciklus vége

    VAN=A<=F

    Ha VAN akkor SORSZ=K
```

7. Kiválogatás

Feladat: Ez az algoritmus egy tömb bizonyos tulajdonságú elemeit teszi egy másik tömbbe.

```
        j=0

        Ciklus i=1-től N-ig ismétel

            Ha tmb[i] rendelkezik az adott tulajdonsággal akkor

                j=j+1

                tmbuj[j]:=tmb[i]

            Ha vége

        Ciklus vége
```

8. Szétválogatás

Feladat:Adott tulajdonsággal rendelkező elemeket a tmb1 tömbbe gyűjtsük ki, a többi elemet pedig a tmb2 tömbbe!

```
        j=0

        k=0

        Ciklus i=1-től N-ig ismétel

            Ha tmb[i] adott tulajdonságú akkor

                j=j+1, tmb1[j]=tmb[i]

            különben

                k=k+1, tmb2[k]=tmb[i]

            ha vége

        Ciklus vége
```

9. Metszet

Feladat: Adott két tömb (tmb1[1..N] és tmb2[1..M]). Válogassuk ki a közös elemeket a tmb tömbbe!

```
k=0
Ciklus i=1-től N-ig ismétel
    j=1
    Ciklus amíg j<=M és tmb2[j]<>tmb1[i]
        j=j+1
    Ciklus vége
Ha j<=M akkor k=k+1, tmb[k]=tmb1[i]
Ciklus  vége
```

10. Unió

Feladat:Adottak a tmb1[1…N] és tmb2[1…M] tömbök. Készítsük el elemeikből a tmb tömböt ügyelve arra, hogy az azonos elemeket csak egyszer használjuk fel.

```
Ciklus i=1-től N-ig ismétel
    tmb[i]:=tmb1[i]
Ciklus vége
k=N
Ciklus j=1-től M-ig ismétel
    i=1
    Ciklus amíg i<=N és tmb2[j]<>tmb1[i]
        i:=i+1
    Ciklus vége
    Ha i>N akkor
        k=k+1
        tmb[k]=tmb2[j]
```

Ha vége

Ciklus vége

11. Rendezési algoritmusok

Rendezés maximum kiválasztással

Ciklus i=N-től 2-ig ismétel

max_index=i

Ciklus j:=1-től i-ig ismétel

Ha tmb[j]>T[max_index] akkor max_index=j

Ciklus vége

Ha i<>max_index

Csere(i,maxindex)

Ha vége

Ciklus vége

Buborékos rendezés

Ciklus i=N-től 2-ig ismétel

Ciklus j=1-től (i-1)-ig ismétel

Ha tmb[j]>tmb[j+1] akkor Csere(j,j+1)

Ciklus vége

Ciklus vége

ADATBÁZISTERVEZÉS

1. Az adatbázis fogalma

Adatbázison az adatok valamely célszerűen elrendezett, valamilyen szisztéma szerinti tárolását értjük. Nem az adatok nagy számán van elsősorban a hangsúly, hanem azok célszerű rendezettségen. Az iskolába naponta igen sok levél érkezik, mégsem tekintjük ezek halmazát adatbázisnak, hiszen hiányzik ezek valamely szempont szerinti rendezettsége. **Az adathalmaz csak akkor válik adatbázissá, ha az valamilyen rend szerint épül fel, mely lehetővé teszi az adatok kezelését.** Természetesen ugyanazon adathalmazból többféle rendszerezés alapján alakíthatunk ki adatbázist. Az adatok tárolásába bevitt rendszernek alkalmasnak kell lennie a leggyakrabban előforduló igények hatékony kielégítésére. Az adatbázisok mellé egy adatbázis kezelő rendszer (DBMS) is járul, mely az adatbázis vagy adatbázisok üzemeltetését biztosítja.

Adatbázis-kezelő rendszer az a programrendszer, amelynek feladata az adatbázishoz való hozzáférések biztosítása és az adatbázis belső karbantartási funkcióinak végrehajtása. (Az adatbázis-kezelő rendszer az adatbázishoz történő mindennemű hozzáférés kezelésére szolgál.(Codd által megadott értelmezés)).

Az adatbázis kezelő rendszer főbb funkciói:

- adatbázisok létrehozása
- adatbázisok tartalmának definiálása
- adatok tárolása,
- adatok lekérdezése,
- adatok védelme, biztonsága
- adatok titkosítása,
- hozzáférési jogok kezelése,
- hozzáférések szinkronizációja
- fizikai adatszerkezet szervezése.

Az adatbázis kezelők három alapvető feladat elvégzésére alapozódnak, melyek mindegyike a számítógépes hardvertől és környezettől való függetlenséggel kapcsolatos. Az általános cél az, hogy inkább az emberi gondolkodáshoz, munkastílushoz hozza közelebb az információs rendszer kidolgozását, minthogy az embereket kényszerítse a számítógép stílusú gondolkozásra.

Függetlenség az aktuális hardver konfigurációtól

Az adatbázis kezelő rendszer rejtse el a felhasználó és a fejlesztő elől is a számítógépek és azok perifériái között jelentkező különbségeket. Az egyes megjelenítő eszközökön (képernyő, nyomtató), azok típusától függetlenül, az alkalmazás ugyanúgy használható legyen és azonos eredményt szolgáltasson. Ily módon az egyik géptípusra kifejlesztett alkalmazás bármelyik, az adatbázis kezelő által támogatott hardver illetve szoftver környezetben módosítás nélkül használható.

Függetlenség az adatelérés módjától

Az egyes operációs rendszerek a fájlokra többfajta adatelérési módot (szekvenciális, indexelt, véletlen) kínálnak az alkalmazások készítőinek. Ez azonban maga után vonja, hogy a fejlesztőnek ezt figyelembe kell vennie és az egyes adatállományokat tárolási módjuknak megfelelően kell kezelni. **Az adatbázis kezelőtől viszont elvárjuk, hogy az adatok tárolásáról és elérési módjáról maga rendelkezzék.** A felhasználó vagy a fejlesztő számára csak a kérdés megfogalmazása és nem az eredmény előállítási módja legyen a feladat. Egy példával ezt úgy szemléltethetnénk, hogy a főnök is csak azt mondja a titkárnőjének, hogy kéri a múlt havi jelentéseket a raktárkészletről, de nem ad útmutatást az adatok elérési módjáról, mivel azt a titkárnő ismeri. Az adatbázis kezelőtől nem csak azt várjuk el, hogy önállóan gondoskodjék az adatok eléréséről, hanem azt is hogy ha több alternatíva is létezik azok közül az optimálist válassza ki.

Függetlenség az adatstruktúráktól

Az adatbázisok szerkezetében beálló változások minél kevesebb módosítást okozzanak az alkalmazásokban. Például, ha az adatbázisokat új adatokkal kell bővíteni, akkor azokat a régebben elkészített alkalmazásokat, melyek ezeket az adatokat nem használják, változtatás nélkül tovább használhassuk.

A korszerű adatbázis-kezelő rendszerek az adatok mellett az adatokra vonatkozó konzisztencia szabályokat *(Egy adatbázis konzisztensnek nevezünk, ha ellentmondásmentes, következetes, azaz eleget tesz meghatározott szabályok halmazának. Ezeket a szabályokat nevezzük integritási szabályoknak.)* illetve az egyes felhasználók jogosultságait is tárolják. Az adatok elérése csak az adatbázis-kezelő rendszeren keresztül, szabványos nyelvet használva (SQL), a konzisztencia és jogosultsági szabályok figyelembevételével történhet meg. Biztosítják a felhasználóknak az adatok konkurens elérését és az ebből adódó konfliktusok kezelését.

2. Alapfogalmak

Redundancia

A redundancia fölösleges adatismétlést jelent. Ilyen módon hibázunk ha például a hallgatók nevét, címét és telefonszámát egyaránt tároljuk a **TAN_ER** (*a tanuló tanulmányi eredményeit tartalmazza*) és az **OSZTONDIJ** (*a tanuló ösztöndíjait tartalmazza*) táblákban is. Ez többszörösen foglalja a tárolóterületet, nehézkessé válik az adatbázis kezelése, lassabb lesz az adatelérés, és nehezebb a visszakeresés. A legnagyobb probléma azonban az adatok frissítésekor jelentkezik. Ha valamely adat – mondjuk egy hallgató címe – megváltozik, akkor azt mindenhol következetesen ki kell javítani, ami nem csak sok felesleges munkát jelent, hanem hibalehetőségeket is rejt magában, és így veszélyezteti az adatintegritást. Elképzelhető például, hogy a változást rögzítjük az **OSZTONDIJ** táblában, de már a **TAN_ER** táblában nem. Képzeljük el milyen problémákat okozhat ez a tanuló levélben való kiértesítése esetén.

Az egyetlen megoldás, ha minden adatot csak egy helyen tárolunk pontosan ott, ahová tartozik. Ezt megfelelő adatmodell felállításával lehet elérni. Ha harmadik normálformára hozott, kulcsmezőkkel ellátott adattáblákat hozunk létre, akkor más táblákban elég az adat egyedi azonosítójára hivatkozni ahelyett, hogy újra letárolnánk azt. A változásokat így csak a saját táblájában kell végrehajtani.

Adatintegritás

Az adatintegritás az adatok érvényességét, helyességét jelenti. Ez magában foglalja az adatok hitelességét, megbízhatóságát, pontosságát, időszerűségét és ellentmondás mentességét. Ha hibás adat kerül az adatbázisba, vagyis megsértjük az adatintegritást (például két azonos kulcsot alkalmazunk, elírunk értékeket, szám helyett szöveg típust alkalmazunk stb.) belső inkonzisztenciát okozunk. Ez téves információt eredményezhet.

Az adatintegritás ellenőrzött adatbevitellel, a hivatkozási integritás megőrzésének automatikus figyelésével (ez azt jelenti, hogy csak létező kulcsra hivatkozunk), és a javítások konzekvens végig vitelével biztosítható.

Adatfüggetlenség

A logikai és fizikai adatfüggetlenség esetén a logikai vagy fizikai adatszerkezet megváltoztatása nincs hatással a felhasználói programokra és megfordítva. Egy új adatmező felvétele miatt például nem kell megváltoztatni a programokat, de egy program módosítása sem vonja maga után az adatok megváltoztatását.

3. Adatbázis modellek

Az adatbázis kezelők fejlődése során többfajta logikai modell alakult ki, melyek az adatok közötti kapcsolatok tárolásában tértek el egymástól.

Ezek a következők:

- hierarchikus
- hálós
- relációs
- objektum relációs és az
- objektum orientált modell.

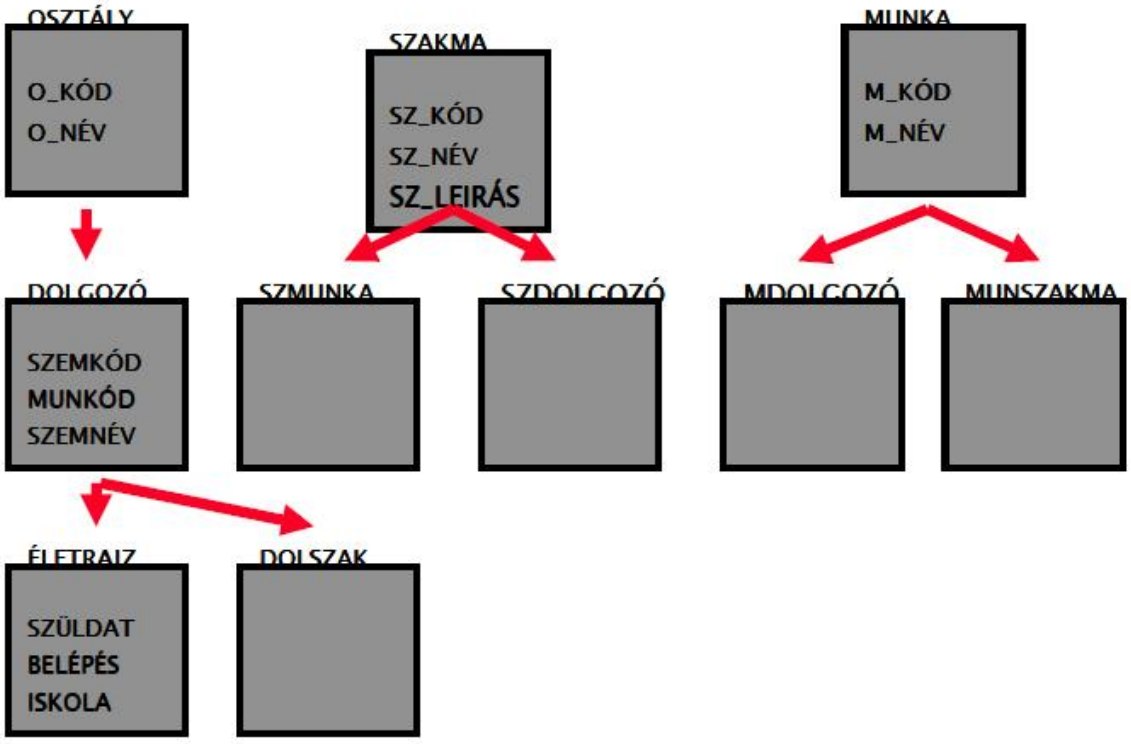
Ezek közül manapság döntően a relációs modellre épülő adatbázis kezelőket használnak.

Hierarchikus adatbázis modell

Ez a modell az 1960-as évek vége felé jelent meg. *Egy a sokhoz* jellegű kapcsolatok kifejezésére szolgált. *Egy az egyhez* jellegű kapcsolatokkal nem foglalkozott. Egy adatnak egy szülője és több leszármazottja lehet.

Egy a sokhoz jellegű kapcsolatban két rekord-típust kapcsolhatunk össze. Az első rekordtípus (amelyikhez a másik rekordtípus több előfordulása tartozhat) a kapcsolat szülője, míg a másik rekordtípus a kapcsolat gyermeke lesz.

Korlátozás: már szó esett arról a tényről, hogy adat sem lehet több szülőnek gyermeke. Sok a sokhoz jellegű kapcsolatok kifejtése igen problémás, a feladat megoldása jelentős redundanciával jár.

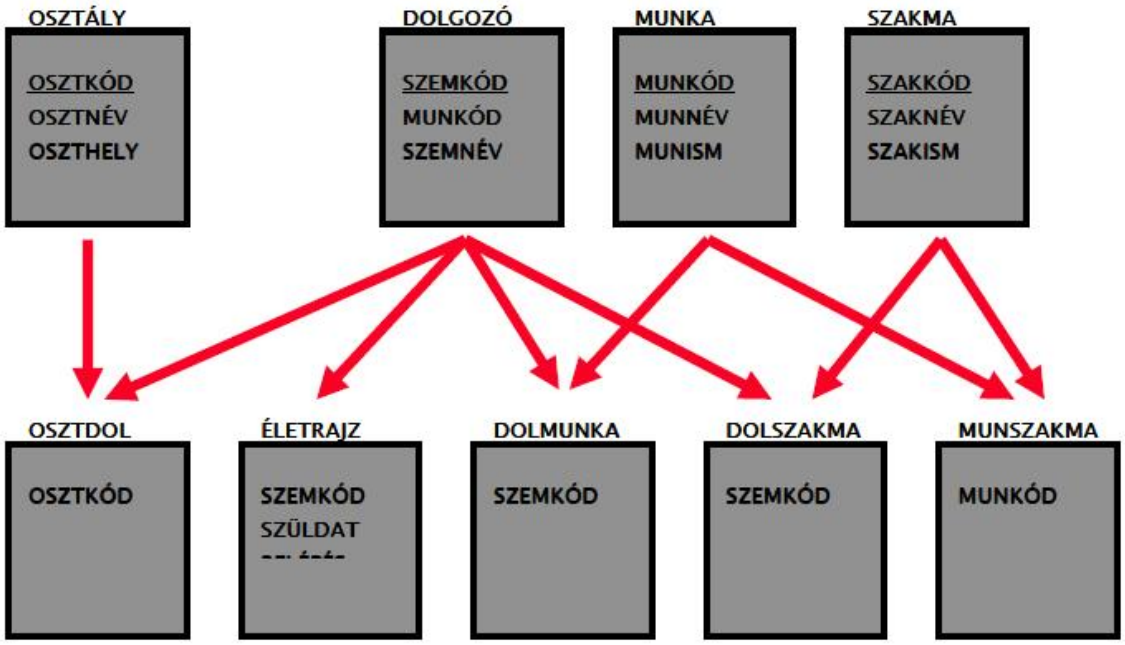


Hálós adatmodell

1960-as évek vége felé jelenik meg. Egy a sokhoz jellegű kapcsolatok közvetlen megjelenítésére alkalmas. Egy egy-a-sokhoz jellegű kapcsolatban az első rekordtípus a kapcsolat főrekordja, míg a másik rekordtípus a kapcsolat alrekordja lesz. A főrekordtípusnak egyedi azonosítóként használható adatelemeket kell tartalmaznia. Ez a rekordtípus kulcsa, amelyet az alrekordokban is meg kell jeleníteni. A főrekordtípustól egy alrekord típusig láncolással jutunk el. Ez **a láncolási útvonal**. Tehát egy alrekordnak több szülője és több leszármazottja is lehet.

Korlátozás: egy láncolási útvonalon található rekordtípus sem lehet ennek az útvonalnak a főrekordja.

Sok a sokhoz jellegű kapcsolatok esetén nem tudunk elkerülni a kevés redundanciát, de ez általában nem okoz problémát az adatbázis használatában.



3. ábra. Példa hálós adatmodellre

Relációs adatmodell

A relációs az egyik legáttekinthetőbb és egyben a 80-as évektől kezdve a legelterjedtebb adatmodell is. Kidolgozása E. F. Codd (1923-2003) nevéhez fűződik. 1970-ben jelent meg alapvető műve a "*A Relational Model Data Large Shared Data Banks*". A relációs modellben az adatokat táblázatok soraiba képezzük le. A legfontosabb eltérés az előzőekben bemutatott két modellhez képest az, hogy itt nincsenek előre definiált kapcsolatok az egyes adategységek között, hanem a kapcsolatok létrehozásához szükséges adatokat tároljuk többszörösen. Ezzel egy sokkal rugalmasabb és általánosabb szerkezetet kapunk.

A modell alapja a matematikai (relációk). Egy rekordtípus előfordulásai táblázatokban jelenik meg. Az adatkezelés ezeken a táblázatokon végzett műveletekként hajtható végre.

4. ábra. Példa relációs adatmodellre

Objektum-relációs adatbázis modell

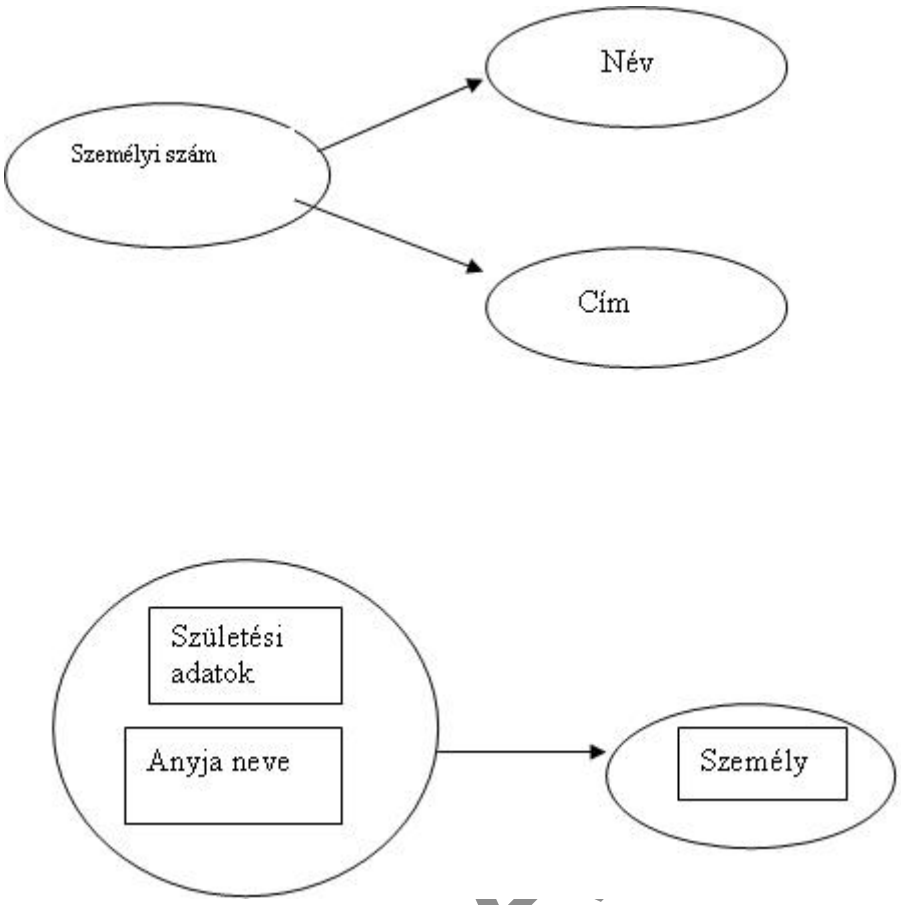
Az objektum relációs adatmodell a relációs adatmodell bővítésével állt elő. Egyrészt az objektum orientált megközelítésben használt osztály, objektum, öröklődés fogalmakat alkalmazza a relációs adatbázis táblákra és a lekérdező nyelvet is ez irányba bővíti. Másrészt pedig támogatja az adatmodell bővítését saját adattípusokkal és azokat kezelő beépített függvényekkel.

4. A relációs adatbázissal kapcsolatos alapfogalmak

Adatok közötti funkcionális kapcsolat

Adatok között akkor áll fenn funkcionális kapcsolat, ha egy vagy több adat konkrét értékéből más adatok egyértelműen következnek. Például a személyi szám és a név között funkcionális kapcsolat áll fenn, mivel minden embernek különböző személyi száma van. Ha ismerünk egy személyi számot, úgy egyértelműen a személyt is azonosítani tudjuk. Tehát a személyi szám és a személy között funkcionális függés van.

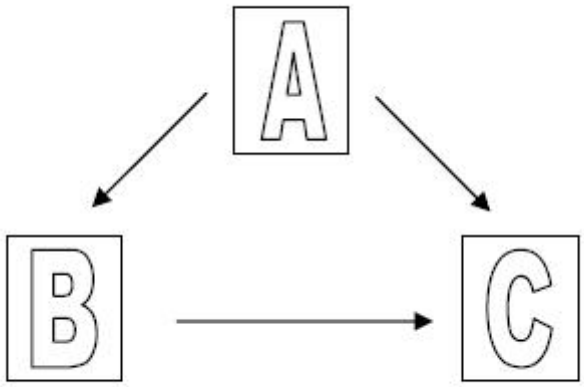
Az adatok közötti funkcionális függőségek az adatok természetéből következnek, nekünk csak fel kell ismerni ezeket a törvényszerűségeket. A tervezés során nagyon fontos, hogy ezeket pontosan felismerjük és figyelembe vegyük.



5. ábra. Példa funkcionális függésre

Tranzitív függés

Ha az "A" attribútum funkcionálisan meghatározza "B" attribútumot, "B" pedig meghatározza "C"-t, továbbá "A" funkcionálisan meghatározza "C"-t, akkor azt mondjuk, hogy az attribútumok között tranzitív függés van.



6. ábra. Példa tranzitív függésre

Reláció kulcs fogalma

A reláció kulcs a reláció egy rekordját (a táblázat egy sorát) azonosítja egyértelműen. A reláció nem tartalmazhat két azonos sort, ezért minden relációban létezni kell kulcsnak.

A kulcs tehát attribútumok azon legszűkebb részhalmaza, mely a reláció rekordjait egyértelműen és egyedileg azonosítja.

Kulcsok fajtái:

Egyszerű kulcs: ez a típusú kulcs egyetlen attribútumból áll.

Összetett kulcs: ez a kulcs típus legalább két attribútumból áll. Előfordulhat az is, hogy az összes oszlop (összes attribútum) szerepel a kulcsban.

Minimális kulcs: egy kulcsot minimálisnak nevezünk, ha az azt alkotó attribútumok közül bármelyiket elhagyva, az attribútumok kapott halmaza már nem rendelkezik a kulcs tulajdonságával. Az egyszerű kulcs egyben minimális is.

Kulcsjelöltek: mindazok a kulcsok, melyek megfelelnek a minimális kulcs definíciójának.

Elsődleges kulcs (primary key): az a kulcs, melyet a kulcsjelöltek közül választunk ki, és kulcsként használjuk. A ki nem választott kulcsjelölteket alternatív kulcsnak nevezzük. Az elsődleges kulcsnak nem lehet NULL az értéke. Az összetett kulcs egyben minimális kulcs is.

Idegen kulcs: olyan attribútum kombináció egy relációban, mely egy másik relációban elsődleges kulcsként szerepel. Az idegen kulcsot tartalmazó relációt *hivatkozó relációnak*, a másikat, melyben ez a kulcs elsődleges, *hivatkozott relációnak* nevezzük.

Külső kulcs: a relációban külső kulcsot vagy kulcsokat is megkülönböztetünk. Ezek az attribútumok nem az adott relációban, hanem az adatbázis másik relációjában alkotnak kulcsot.

Hivatkozási integritás: a hivatkozási integritás azt jelenti, hogy csak olyan kulcsra hivatkozunk, ami létezik a másik táblában.??????????????????????

Hivatkozási integritás megőrzése: két tábla összekapcsolásakor bekapcsolhatjuk a hivatkozási integritás megőrzése opciót, aminek hatására automatikusan figyelni fogja a rendszer, hogy ne vihessünk be rossz hivatkozást, vagy ne módosíthassunk illetve törölhessünk ki olyan kulcsot, amire egy másik táblában hivatkozunk. Amennyiben olyan műveletet kezdeményezünk, ami megsértené a hivatkozási integritást, a rendszer figyelmeztetést ad erről, és nem hagyja elvégezni azt. További lehetőség a hivatkozási integritás biztosítására, ha egy kulcs módosításának vagy törlésének következményeit konzekvensen kijavítjuk a többi táblában is, ahol hivatkoztak rá.

Kaszkádolt frissítés: ha megváltoztatunk egy kulcsot, akkor mindenhol javítja azt, ahol hivatkozunk rá.

Kaszkádolt törlés: ha kitörlünk egy kulcsot, akkor minden olyan rekordot töröl, ami hivatkozik rá.

5. Kapcsolatok típusai

Egy az egyhez (One To One) kapcsolat: ez kölcsönösen egyértelmű megfeleltetés, ami azt jelenti, hogy az egyik tábla egy rekordjához a másik táblából csak egy rekord tartozhat, vagy esetleg egy sem. Ez megfordítva ugyanúgy igaz.

Például mondjuk egy dolgozó legfeljebb egy egységnek lehet a főnöke, és egy egységnek is csak egy főnöke lehet. Persze van olyan alkalmazott, aki egyik egységnek sem főnöke.

Egy a többhöz (One To Many) kapcsolat: csak az egyik irányban egyértelmű a hozzárendelés, azaz a másik oldalról nézve egy rekordhoz több rekord is tartozhat a másik táblából.

Például egy munkás egyértelmű, hogy melyik egységnél dolgozik, egy egységnél viszont többen is dolgoznak.

Több a többhöz (Many To Many) kapcsolat: egyik irányban sem egyértelmű a hozzárendelés, tehát egy rekordhoz több rekord is tartozhat a másik táblából, ugyanakkor a másik tábla egy rekordja tartozhat többhöz is. Például egy filmnek több szereplője van, egy színész pedig általában több filmben is játszik.

6. Normálformák:

1. normálforma (1NF): az R reláció 1. normálformában van, ha a relációban szereplő minden érték elemi, minden attribútum csak egy értéket vesz fel az értelmezési tartományából. Tehát a tábla minden cellájában pontosan egy érték van, ha esetleg valamelyik cellájába nem kerülne érték, úgy oda automatikusan NULL érték kerül.

2. normálforma (2NF): az R reláció 2. normálforma van akkor és csak akkor, ha 1. normálformában van és minden olyan attribútuma, mely nem része az elsődleges kulcsnak, funkcionálisan teljesen függ az elsődleges kulcstól.

> Egy másodlagos attribútum teljesen függ a kulcstól, ha funkcionálisan nem függ a kulcsnak semmilyen valódi részhalmazától.

3. normálforma (3NF): az R reláció 3. normálformában van akkor és csak akkor, ha 2. normálformában van és minden olyan attribútuma, mely nem része az elsődleges kulcsnak funkcionálisan teljesen függ az elsődleges kulcstól és csak attól. Ezt az állítást úgy is megfogalmazhatnánk, hogy a relációnak nincs két olyan másodlagos attribútuma melyek funkcionálisan meghatároznák egymást.

> *Például* az a tábla, amely másodlagos attribútumként tartalmazza egy város irányítószámát és nevét is, nincs 3. normálformában, mert az irányítószám funkcionálisan meghatározza a város nevét.

A 3. normálforma definícióját meghatározhatjuk úgy is, hogy a reláció 3. normálformában van, ha nem tartalmaz tranzitív függőségeket.

A normálformák használatával az adatmodellt redundancia mentessé tehetjük, valamint kiküszöbölhetjük az úgynevezett anomáliákat:

Hozzáadási anomália: például a táblába egy rekordot nem tudunk teljesen feltölteni, mert még van olyan adat (oszlop), aminek az értékét nem ismerjük.

Módosítási anomália: több helyen is tárolhatjuk ugyanazt az adatot. Módosításnál mindet át kellene írni, ha valamelyik kimarad, az hibát okozhat.

Törlési anomália: Előzőhöz hasonlóan több helyen történő tárolásnál például az egyik táblából kitörlünk egy sort, mert az egyik oszlopában lévő adat már nem szükséges, ekkor olyan információt is elveszíthetünk, ami csak itt volt tárolva.

7. Normalizálás

Ha a reláció 1. normálformában van és a kulcs egyszerű , akkor a reláció 2. normálformában is van. Összetett kulcs esetén a kívánt második alak eléréséhez a *táblánkat sokszor fel kell bontanunk több kisebb táblára.* Ezt a felbontást úgy kell elvégezni, hogy az új relációk kulcsai az eredeti reláció elsődleges kulcsa, vagy annak egy részhalmaza legyen, oszlopai pedig azok az attribútumok legyenek, melyek az új kulcsoktól funkcionálisan teljesen függnek. A 3. normálformára hozásnál a tranzitív függőséget tartalmazó táblát fel kell bontani több táblára úgy, hogy a tranzitív függőségben lévő oszlopok különböző táblákba kerüljenek.

8. Adatbázis életciklusa:

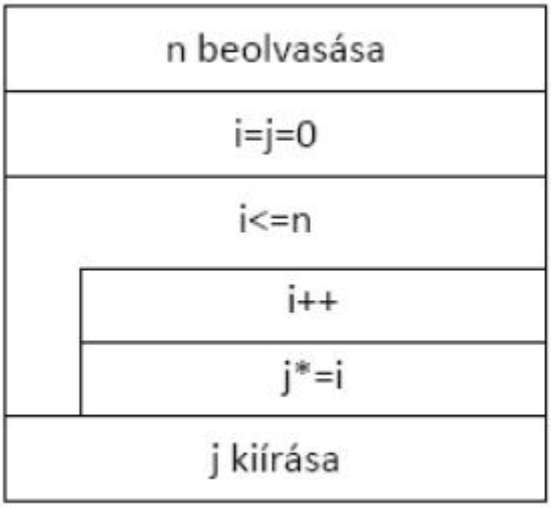
- Elemzés
- Tervezés
- Az adatbázis létrehozása
- Az adatbázis feltöltése
- Az adatbázis használata és karbantartása
- Az adatbázis módosítása

TANULÁSIRÁNYÍTÓ

1. Ön egy cég informatikai szakembereként dolgozik, ahol a rendszer egyik programjában, valahol az egydimenziós tömbben tárolt adatok közötti keresés során probléma merült fel. Az adatok rendezetlenül vannak, a tömb tartalma nem változik sűrűn. Azt a feladatot kapta, hogy adjon megoldást a keresési algoritmus-problémára. Mutassa be a vázolt lehetőség előnyeit és hátrányait egyaránt!

2. a) Mit csinál a következő algoritmus?

b) Írja át a struktogramot pszeudokódokra, majd folyamatábrára!



7. ábra.

3. Írjon algoritmust egy N elemű tömb elemei szorzatának meghatározására! (használja az összegzés tételét)

4. Határozza meg a logaritmusos keresésben szereplő változók típusait!

5.feladat

(a) Írja meg a minimum kiválasztás algoritmusát!

(b) Módosítsa az algoritmust úgy, hogy a tömb 2. legkisebb elemét határozza meg!

(c) Módosítsa az algoritmust úgy, hogy a tömb adott tulajdonságú (pl. páros) elemei közül határozza meg a legkisebbet!

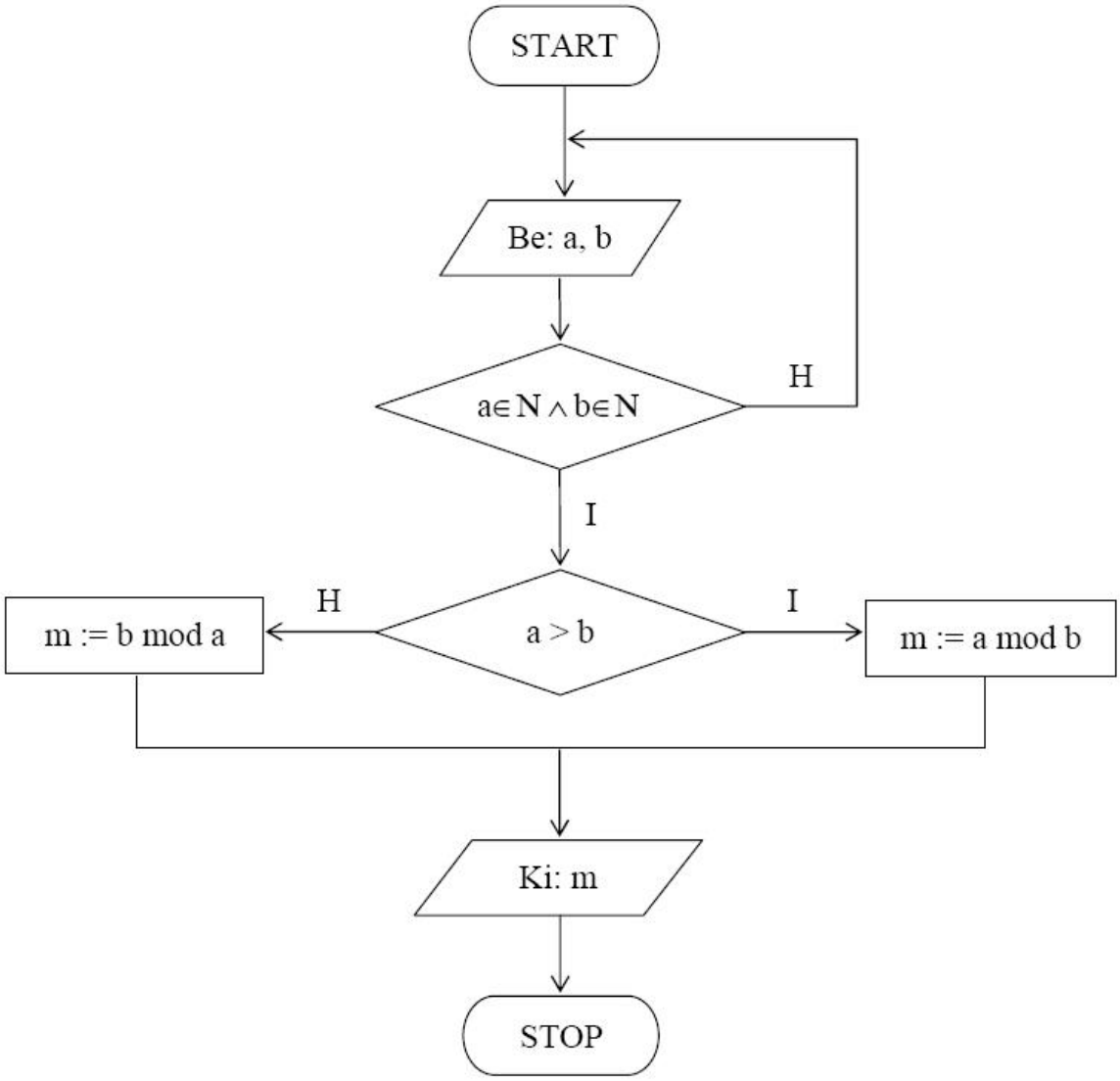
6.Adott egy N elemű, egész számokat tartalmazó tömb. Válogassa ki a POZITIV nevű tömbbe az adott tömb pozitív elemeit, a többi elem kerüljön a NEMPOZITIV tömbbe. Az algoritmus határozza meg mindkét tömb

- maximális és minimális elemét, illetve a
- tömbök elemeinek az átlagát (ügyeljen arra, hogy a tömb üres is lehet)

7. Írja át az UNIÓ algoritmust rendezett tömbökre (összefésülés algoritmusa)

8.Írja meg a minimum kiválasztásos rendezés algoritmusát!

9.Mit csinál a következő algoritmus?



8. ábra.

10. A Fekete Macska Kft. főleg kereskedelemmel foglalkozik. Ennek megfelelően rendelések futnak be hozzá vásárlóktól, amelyeket a szállítók révén teljesít. Egy terméket csak egy szállító szállít, és egy szállító csak egy terméket szállít. Nyilvántartjuk a vásárlók pillanatnyi egyenlegét. Egy rendelésen több tétel is szerepelhet, a tétel tartalma a termék megnevezése és a mennyiség. A szállítók, termékek és városok neve egyedi, de a vásárlók neve csak egy városon belül egyedi. Feltehetjük, hogy minden városnak csak egy irányítószáma van.

Készítse el az interjú tervét, kérdőíveket stb., amely segítségével pontosan feltárja, hogy milyen követelményeket támaszt a felhasználó az elkészített adatbázissal szemben.

Ennek alapján készítsen szerződéstervezetet.

Tanárával pontosítsa az elkészített szerződés mintát!

Segítség a feladat megoldásához:

A tárolandó adatok:

- SzállítóIrányítóSzám,
- VásárlóIrányítóSzám
- SzállítóNév, RendelésSzáma
- Mennyiség
- RendelésDátuma
- VásárlóVárosnév
- EgyenlegEgységár
- Terméknév
- SzállítóVárosnév
- Vásárlónév
- Terméknév.

Feladatok:

Készítsük el a Fekete Macska Kft. adatbázis-modelljét (a 3NF alakig)!

Valósítsuk meg az adatmodellt az Access segítségével, és néhány tesztadatot írjunk be az adatbázis tábláiba!

Valósítsuk meg a következő lekérdezéseket:

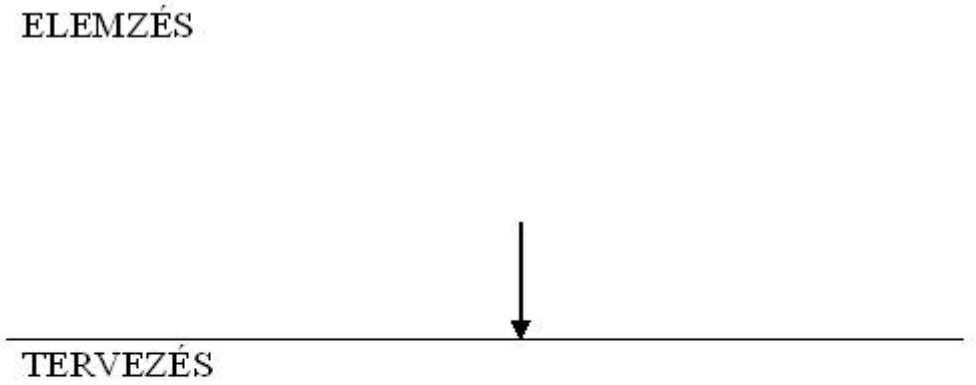
- Az "XXX" nevű terméket szállító szállító városának az irányítószáma
- 2010 évi vásárlóink neve
- YYY város vásárlóink egyenlege
- a legnagyobb egyenlegű vásárló neve és városa
- a leghűségesebb vásárlónk melyik szállítótól vásárolt a legtöbbször?

Amennyiben problémája van az adatbázis tervének elkészítésével, tanulmányozza át az ÖNELLENŐRZÉS feladatsor 12. feladatát, és az ott található megoldási javaslatot!

ÖNELLENŐRZŐ FELADATOK

1. feladat

Egészítse ki az ábrát a programtervezés során megvalósított elemzés és tervezés lépéseivel!



9. ábra.

2. feladat

Keresse meg például az internet lehetőségeit felhasználva, mi a felülről lefele tervezés fordított művelete. Fejtse ki a módszer lényegét! Mikor alkalmazzuk ezt a tervezési technikát?

3. feladat

Mit érdemes leírni a felhasználóval kötött szerződésben?

4. feladat

Határozza meg az algoritmus fogalmát! Írja le a tengerparti vízesárok építés algoritmusát!

5. feladat

Írja át a lineáris keresés algoritmusát rendezett tömbre! Használja a már megismert lineáris rendezési algoritmust! Miben tér el a módszer rendezett tömbök esetén?

Ha a megoldás nem egyértelmű, gondoljon végig egy konkrét példát!

Legyen adott a következő rendezett halmaz: 5 12 18 20 23 25

Keresett elem: 15

Meddig érdemes vizsgálni a tömb elemeit? Ha a tömb eleme nagyobb mint a keresett elem, esetünkben a 15, mi történjen?

Mi a helyzet, ha a keresett elem 3 (kisebb, mint a tömb első eleme)?

Mi a helyzet, ha a keresett elem 30 (nagyobb, mint a tömb utolsó eleme)?

A fenti két problémát végig gondolva, hogyan optimalizálná az Ön által elkészített algoritmust?

6. feladat

Állítsa sorrendbe a programozás lépéseit!

- Dokumentálás
- Tervezés
- A probléma meghatározása, felmérése (specifikáció)
- Tesztelés
- Kódolás
- A megvalósítás (implementáció)

1. _____

2. _____

3. _____

4. _____

5. _____

6 _____

7. feladat

Írja meg az "a" és "b" elemek cseréjének az algoritmusát!

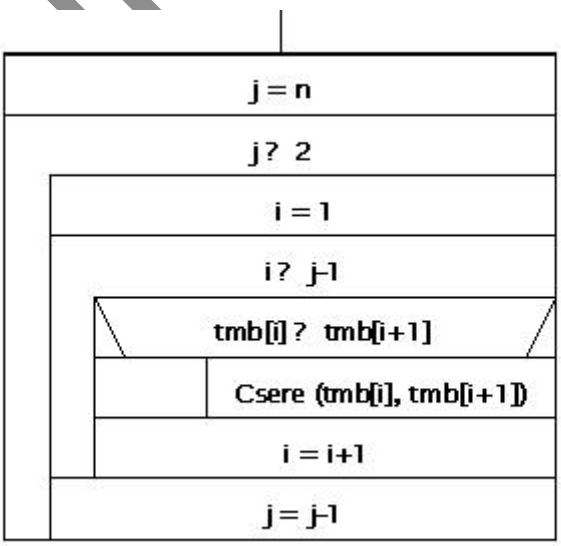
_____

_____

_____

8. feladat

Melyik megismert algoritmus struktogramját látja?



10. ábra.

9. feladat

Miért fontosak számunkra az adatbázisok?

10. feladat

Sorolja fel az adatbázis-kezelőkkel szembeni követelményeket!

11. feladat

Sorolja fel az adatbázis kezelők előnyeit!

12. feladat

Egy videó kölcsönzőtől azt a feladatot kapja, hogy készítse el egyszerűen használható módon adatbázisuk modelljét. Ehhez a feladathoz a következő adatok állnak rendelkezésére:

- Kazetta száma
- Kazetta típusa
- Film címe
- Film száma
- Rendelés száma
- Rendelés dátuma
- Kölcsönzés száma
- Tag sorszáma
- Tag neve
- Tag címe
- Kölcsönzés dátuma
- Visszahozás dátuma
- Film főszereplője
- Film rendezője

A filmeknek egyedi azonosító száma van, a cím több különböző filmhez is tartozhat (ezért nem lehet kulcsnak választani). Egy film több kazettán is szerepelhet, a kölcsönző által készített másolatokban. A kazettáknak szintén egyedi azonosítójuk van, valamint a típusát is nyilvántartják. A kölcsönzőből kizárólag tagok kölcsönözhetnek.

Gondolja végig, hogy elegendő ismerete van-e a feladatról, ha nemmilyen probléma megoldása miatt nem elegendő az ismeretanyaga

- mit kérdezne
- kitől kérdezné és milyen formában

- milyen módon dokumentálná az új ismereteit

Határozza meg a következő fogalmakat! Ahol az aktuális tananyagban nem találja a megfelelő választ, ott például használja az internet nyújtotta segítséget, vagy lapozzon fel egy aktuális témával foglalkozó könyvet.

- egyedtípus, -előfordulás, -halmaz
- az egyed tulajdonságai, tulajdonságtípus, -előfordulás, -értékkészlet
- az egyedtípus azonosítója
- az adatbázis (adatmodell) fogalmi, logikai és fizikai szintjei
- az elsődleges kulcs, közös kulcs és a külső kulcs, egy-sok és sok-sok kapcsolatok ábrázolása
- redudancia
- funkcionális függés
- 1. normálforma
- 2. normálforma
- 3. normálforma
- adatintegritás
- adatfüggetlenség
- relációs adatszerkezetek jellemzői

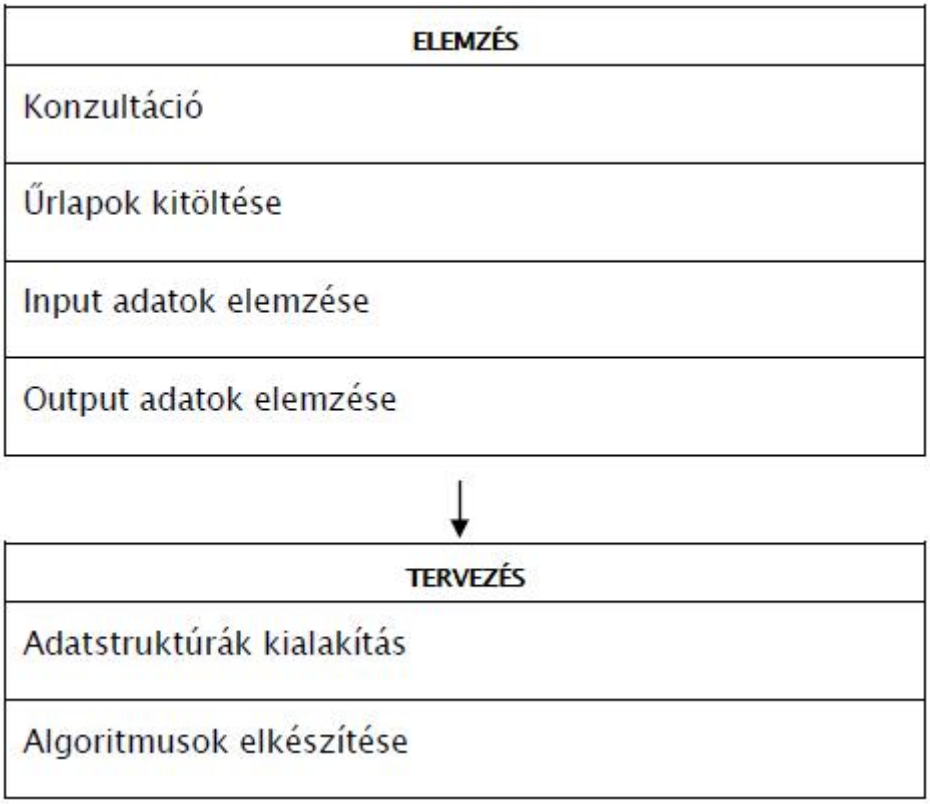
Fogalmazza meg, milyen problémákat vetne fel, ha valamennyi adatot egy táblában tárolná!

Lekérdezések készítése az adatbázishoz formához

- Tudjuk meg azoknak a tagoknak az adatait, akik bár be vannak iratkozva, de még nem kölcsönöztek egyetlen filmet sem (lekérdezés neve: Nem_kolcsonzott)!
- A word körlevél készítő funkcióját felhasználva készítsen reklámanyagot a lekérdezésben megjelenő tagok számára!
- Tudjuk meg az összes rendelkezésre álló filmünk főszereplőjét (lekérdezés neve: Foszereplok)!
- Állapítsuk meg az összes rendelkezésre álló film címét és rendezőjét a film címe szerint növekvő rendben (lekérdezés neve: Filmek)!
- Vizsgáljuk meg, hogy van-e esetleg két azonos nevű, különböző lakcímű tagunk. (lekérdezés neve:Azonos_nevek)!
- Jelenítsük meg azon filmek és kazetták számát és a film címét, amelyeket még soha nem kölcsönöztek ki (lekérdezés neve: Nem_nezettek)!

MEGOLDÁSOK

1. FELADAT



11. ábra.

2. feladat

A kérdéses módszer a Bottom-up technika. Ha egy adott blokkban világos a feladat, tehát szétbontással nem jutunk már sehová, viszont a feladat megvalósításának módját nem látjuk át, akkor elemi részekből rakjuk összerakni a blokkot. Az elemi részek lehetnek utasítások (ciklus, feltételes elágazás stb.), függvények, objektumok, más által fejlesztett blokkok. A bottom-up tervezés nem is mindig válik el a kódolástól, hiszen a kódolás során általában ugyanazok az alapegységek, mint a bottom-up tervezésnél.

3. feladat

Szerződésben kell rögzíti:

- a megrendelő és a programozó adatait

- a program által megoldandó feladatot
- a megrendelő részére járó egyéb termékeket és szolgáltatásokat (dokumentáció, felhasználói kézikönyv, betanítás, tanácsadás, karbantartás, stb.)
- a megrendelőnek a program forgalmazására való jogát,
- a vállalási határidőt
- mindezek árát, stb.

4. feladat

Valamely probléma megoldására bevezetett, véges számú lépéssorozat, amelyet véges számú alkalommal, mechanikusan megismételve a probléma megoldását kapjuk.

A probléma megoldásának egy lehetséges algoritmusa:

Vedd a kezedbe a kisvödröt

Ismételd az alábbiakat mindaddig, amíg tele nem lesz a vizesárok:

Merítsd tele a vödröt tengervízzel.

Töltsd a vizesárokba.

Tedd le a kisvödröt.

5. feladat

```
i=1
Ha keresett_elem<tmb[1] akkor
        kiír:  "Nincs az elem a tömbben"
Különben
        Ciklus amíg i<=N és tmb[i]<keresett_elem

                i=i+1

        Ciklus vége

        Ha tmb[i]=keresett_elem akkor

                kiír:"A keresett elem az i-dik"

        Különben

                kiír: "A tömbben nincs a keresett elem"

        Ha vége
```

6. feladat

1. A probléma meghatározása, felmérése (specifikáció)
2. A megvalósítás (implementáció)
3. Tervezés
4. Kódolás
5. Tesztelés
6. Dokumentálás

7. feladat

csere = b

b = a

a = csere

8. feladat

Buborék rendezés

9. feladat

- Az adatok széles körét adatbázisokban tároljuk
- Pontosan el kell tudni magyarázni az adatbázist tervező, programozó szakembereknek, hogy a számunkra szükséges adatbázis környezet hogyan álljon elő
- Tudják, mit várhatnak el egy adatbázis-kezelő rendszertől

10. feladat

- Függetlenség az aktuális hardver konfigurációtól
- Függetlenség az adatelérés módjától
- Függetlenség az adatstruktúráktól

11. feladat

- Elfedi az adatok fizikai tárolási szerkezetét, a felhasználóknak, programoknak csak a logikai adatszerkezetet kell ismernie
- Hatékony adatelérést biztosítanak
- Adat integritás ellenőrzése, jogosultságok kezelése
- Konkurens adatelérés és adatvesztés elleni védelem (hardver hiba esetén is)

- Lerövidíti a programfejlesztés idejét

12. feladat

Táblák kialakítása, első normálformára hozás (1NF)

Kazetta

- Kazetta száma
- Kazetta típusa
- Film címe
- Film száma
- Rendelés száma
- Rendelés dátuma
- Film főszereplője
- Film rendezője

Kölcsönzés

- Kölcsönzés száma
- Kazetta száma
- Tag sorszáma
- Tag neve
- Tag címe
- Kölcsönzés dátuma
- Visszahozás dátuma

A kazetta száma mindkét adatlapon egyszer-egyszer szerepel, mivel minden kazettának egyedi azonosítója van, de egy kazettát többen is kikölcsönözhetnek, ezért a kölcsönzés száma és a tag adatai ismétlődnek egy adatlapon.

Második normál forma kialakítása (2NF)

Kiválasztjuk az elsődleges attribútumokat (kulcs).

Meghatározzuk azokat az attribútumokat, amelyek nem a teljes kulcstól függenek azoktól, illetve melyek teljes függőségben vannak a teljes kulccsal. Mivel egyik kulcsunk sem összetett kulcs (egyszerű kulcs), ezért mindkét reláció változatlan marad.

Természetesen, ha lenne olyan másodlagos attribútum, melynek függése a kulcstól nem lenne teljes (van olyan másodlagos attribútum funkcionálisan függ a kulcs egy részétől), az aktuális táblát részekre bontanánk.

A feladat tükrében válaszolja meg, miért törekszünk adattábláinkban az egyszerű kulcs használatára!

Kazetta

- Kazetta száma

- Kazetta típusa
- Film címe
- Film száma
- Rendelés száma
- Rendelés dátuma
- Film főszereplője
- Film rendezője

Kölcsönzés

- Kölcsönzés száma
- Kazetta száma
- Tag sorszáma
- Tag neve
- Tag címe
- Kölcsönzés dátuma
- Visszahozás dátuma

Harmadik normál forma kialakítása (3NF)

Elkülönítjük azokat az attribútumokat, melyek a reláció kulcsától csak közvetetten függnek. Pl. a film rendezője nem függ közvetlenül a kazetta számától, stb.

Kazetta tábla --> kiválik a film számától függő három mező (Film tábla), és a rendelés számától függő egy mező (Rendelés tábla).

Kölcsönzés tábla --> kiválik a tag sorszámától függő két mező.

- Film
- Film száma
- Rendelés száma (külső kulcs)
- Film címe
- Film rendezője
- Film főszereplője

Kazetta

- Kazetta száma
- Film száma (külső kulcs)
- Kazetta típusa

Rendelés

- Rendelés száma
- Rendelés dátuma

Kölcsönzés

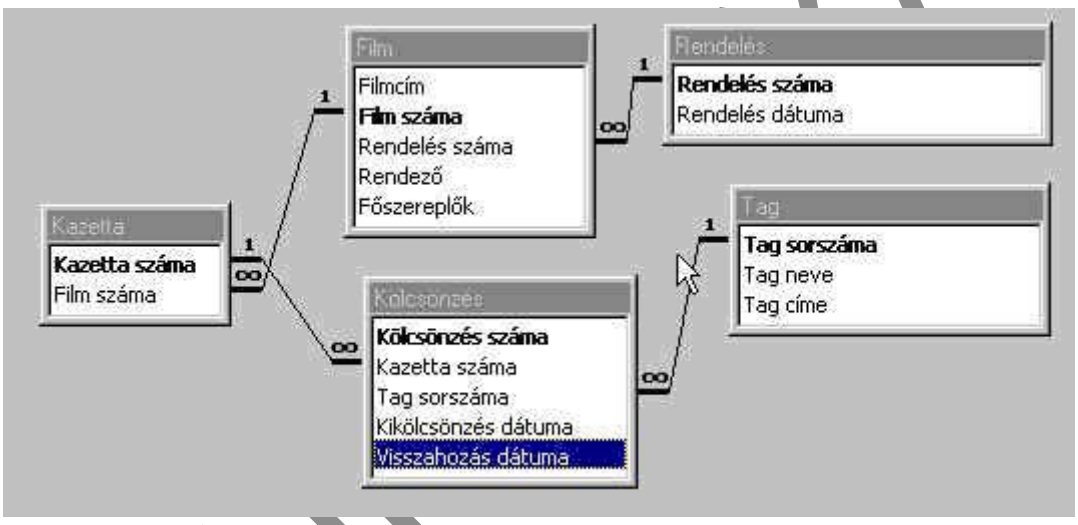
- Kölcsönzés száma
- Kazetta száma (külső kulcs)
- Tag sorszáma (külső kulcs)
- Kölcsönzés dátuma
- Visszahozás dátuma

Tag

- Tag sorszáma
- Tag neve
- Tag címe

Ez az adatszerkezet már megfelelő arra, hogy adatbázis kezelő programmal dolgozzuk fel.

Kapcsolatok a táblák között



12. ábra. Táblák kapcsolata, kapcsolatok típusai

Készítsük el az adatbázist, majd töltsük fel adatokkal!

IRODALOMJEGYZÉK

FELHASZNÁLT IRODALOM

B. W. Kernighan –D.M. Ritchie: A C programozási nyelv; Műszaki Könyvkiadó, Budapest, 1988

Peter Norton – John Socha: Az IBM PC assembly nyelvű programozása; Novotrade Kiadó – Prentice– Hall, 1991

Molnár Bálint: Bevezetés a rendszerelemzésbe A rendszerszervezés alapjai; Műszaki Könyvkiadó 2004

A(z) 1155-06 modul 017-es szakmai tankönyvi tartalomeleme felhasználható az alábbi szakképesítésekhez:

A szakképesítés OKJ azonosító száma:	A szakképesítés megnevezése
54 481 01 1000 00 00	CAD-CAM informatikus
54 481 04 0010 54 01	Gazdasági informatikus
54 481 04 0010 54 02	Infostruktúra menedzser
54 481 04 0010 54 03	Ipari informatikai technikus
54 481 04 0010 54 04	Műszaki informatikus
54 481 04 0010 54 05	Távközlési informatikus
54 481 04 0010 54 06	Telekommunikációs informatikus
54 481 04 0010 54 07	Térinformatikus

A szakmai tankönyvi tartalomelem feldolgozásához ajánlott óraszám:
20 óra

A kiadvány az Új Magyarország Fejlesztési Terv
TÁMOP 2.2.1 08/1-2008-0002 „A képzés minőségének és tartalmának fejlesztése” keretében készült.
A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Kiadja a Nemzeti Szakképzési és Felnőttképzési Intézet
1085 Budapest, Baross u. 52.
Telefon: (1) 210-1065, Fax: (1) 210-1063

Felelős kiadó:
Nagy László főigazgató