Tantárgyfelosztás

Készítette: Szalayné Tahy Zsuzsanna
Neptun-azonosító: N0CD19
E-mail: sztzs@inf.elte.hu

BME-VIK mérnökinformatikus képzés
programozás alapjai 1 tantárgy
beadandó házi feladatának stílusában
Kurzuskód: BMEVIEEAA00

2020. február 02.

Előszó

A BME-VIK első szemeszterében tanult Programozás alapjai 1 tárgy teljesítéséhez tartozik egy szoftver tervezése, megvalósítása és dokumentációjának elkészítése. A megoldással kapcsolatos elvárások:

- C nyelvű program, amely a nyelv lehetőségeit kihasználja: strukturált felépítés, több modulra bontás, dinamikus memóriakezelés és fájlkezelés.
  - Dinamikus memóriakezelést kell használni a program lényegét adó adatszerkezetnél. Az adatszerkezetben az indirekciók száma legalább kettő kell legyen, azaz dinamikusan foglalt területen kell legyen dinamikusan foglalt terület címe.
  - A dinamikus memóriakezelést erre alkalmas beépülő eszközzel ellenőrizni kell.
  - A fájlkezelésben olyan adatokat kell tudni kezelni, amelyek száma változik, vagy változhatna.
- A program mellé el kell készüljön a programozói és a felhasználói dokumentáció.

Az elkészítendő szoftver kiválasztását számos példa segíti, de saját ötlet is megvalósítható. A választást a laborvezetővel kell egyeztetni. Több ajánlott feladat hasonlít a közismereti emelt szintű informatika érettségi 4. feladatához. A feladattípusok hasonlóak, de a nyelv választása, az elvárt adatstruktúrák (indirekciók alkalmazása) miatt az elvárt megoldás teljesen más.

Jelen esetben az összevetés alapja az érettségi feladatok közül a 2019. májusi idegennyelvű emelt szintű közismereti informatika érettségi 4., Tantárgyfelosztás nevű feladata, amely a „Nyilvántartás jellegű programok” kategóriájába tartozik. Az itt bemutatott megoldás a memóriakezelésének ellenőrzése megtörtént, de nem tartalmazza az ellenőrző eszközt.

Az összevetéshez tartozik egy harmadik megoldás is: az ELTE-IK programtervező informatikus képzésén, az első szemeszter végén Programozás tárgyból beadandó program és dokumentációja. Ott a feladat kötött, de az érettségi feladatok közül sok átírható az ott elvárt tudás mérésére is.

Tartalom

Felhasználói dokumentáció

Feladat

A tantárgyfelosztás a tanév tervezésének alapvető dokumentuma. A tantárgyfelosztás azt tartalmazza, hogy a tanárok a tantárgyaikat mely osztályokban, hány órában tanítják. A feladatban egy négy évfolyamos gimnázium tantárgyfelosztásának adatait kell elemezni. A tantárgyfelosztást ezúttal egy adatbáziskezelő programmal előállított, egyszerű szerkezetű szöveges állományból kapjuk az alábbi minta szerint. (Minden bejegyzést négy sor tárol.)

Albatrosz Aladin
biologia
9.a
2

A bejegyzés megadja, hogy Albatrosz Aladin tanár úr biológiát (biologia) fog tanítani a 9.a osztályban heti 2 órában. Ha az osztály betűjele x, akkor évfolyam szintű csoportról van szó. Például a 12. évfolyamon több német csoport van, de nem osztályok szerint, hanem előismeret alapján van bontva. Az osztályfőnököket arról ismerhetjük fel, hogy ők tartják az osztályfőnöki (osztalyfonoki) órát. Egyes osztályokban bizonyos tantárgyakat a tanulók csoportbontásban tanulnak: ekkor az adott tantárgyra és osztályra (vagy évfolyamra) két vagy több bejegyzést is tartalmaz a tantárgyfelosztás.

A megoldás során – az érettségi feladattól eltérően nem használható fel méretkorlát Az érettségi feladatban adottaktól eltérően, a heti óraszám nem negatív és maximum 40 óra lehet.

A program

- megadja a beolvasott bejegyzések számát;
- megadja, hogy az iskolában hetente összesen hány tanítási óra van;
- megadja, hogy egy adott tanár hetente hány órában tanít;
- listát készít, amelyben megadja az egyes osztályok osztályfőnökeinek a nevét;
- megvizsgálja, hogy egy adott osztály egy adott tárgyat csoportbontásban tanul-e (ekkor az adott tantárgyra és osztályra két bejegyzést is tartalmaz a tantárgyfelosztás);
- megadja, hogy hány tanár dolgozik az iskolában.

Futási környezet

A program konzol alkalmazás. Grafikus képernyőkezelést nem tartalmaz.

Használat

A program indítása

A program fordítás után `tanfel` nevű futtatható állománnyal indítható. A teljeskörű használathoz a fájlal azonos mappában kell legyen a `beosztas.txt`, adatokat tartalmazó állomány.

A program használata

A programindítást követően menü jelenik meg, a program használatát a konzol ablakban megjelenő utasítások segítik. Az egyes funkciókat a megjelenített szám, illetve betű beírásával lehet indítani.

A részfeladat választása az ’1’-’8’ karakterekkel történik. Ezen belül a 6. és 7. pont almenüt tartalmaz, amely kisbetű karakterek leütésével választhatók. A 6. pont esetében ’a’-’d’ hatására egy-egy lista jelenik

meg, más karakternek nincs hatása. A 7. pont 'a'-'f' pontja a feladat kérdéseire ad választ. A 'g', illetve minden más karakter is, a feladat teljes megoldását – az összes kérdésre sorban válaszolva – adja meg.

A választást Enterrel kell lezárni. A beírt első karakter jelenti a választ, ami ez után az Enterig következik, azt a program figyelmen kívül hagyja.

Az egyes részfeladatok után az Enter leütésével térhetünk vissza a menühöz.

A menüpontok funkciói

1 input fájlból

A programfájl mellett található `beosztas.txt` adatait olvassa be. Jelzi, ha a fájlolvasás nem sikeres, ilyenkor semmilyen módosítás nem történik. Amennyiben már van beolvasott adat, akkor az a beolvasás előtt törlődik. A program feltételezi, hogy a fájl helyes szerkezetű, így a beolvasás az első hibásan kezdődő bejegyzésig (a tanár neve üres) tart. A tanár neve legfeljebb 30, a tantárgy 20, a csoport neve 7 karakter lehet. Az ennél hosszabb adatsorokat a program levágja.

2 input konzolról

A menüpont kiválasztása után új képernyőn lehet megszakítás nélkül bejegyzéseket felvinni. A program kiírja az éppen várt adat megnevezését. A túl hosszú szöveges adatot a program levágja, az óraszámnál szöveg vagy 40-nél nagyobb vagy 0-nál kisebb érték beírásakor az adatot újra bekéri, kivéve -1 esetén. Amennyiben egy szöveges adat csupán egy Enter vagy a beírt óraszám -1, az a beírás végét jelenti és a megkezdett legutolsó rekord nem lesz tárolva.

3 egyedi bejegyzés hozzáadása

Amennyiben egyetlen rekordot szeretnénk beírni, ezt a menüpontot érdemes választani. A menü alatt beírhatók az adatok. Ha közben meggondoljuk magunkat, akkor szöveges adat nélkül Entert ütve vagy az óraszámhoz -1-et írva elkerülhetjük az adat rögzítését.

4 egyedi bejegyzés törlése

Bejegyzés törlésére csak mind a négy adat pontos megadásával van lehetőség. Amennyiben a beírt adatokkal van egyezés, akkor az első ilyen találat lesz törölve. Amennyiben nincs találat, a program jelzi, hogy nem törölt ki rekordot.

Adat módosítására nincs külön menüpont. Szükség esetén a hibás bejegyzés törlésével és újra felvitelével (3) oldható meg. Mivel a bejegyzések között lehetnek megegyezők – adott tanár adott tárgyat adott osztályban több csoportban azonos óraszámban tanít – a módosításnál figyelembe kell venni, hogy mindig csak az első találat törlődik és minden felvétel egy-egy külön bejegyzést eredményez

5 napló törlése

A naplóban található összes bejegyzés törlése.

6 listázás

Az adatok áttekintésére többféle listát kérhetünk. A listák megjelenítése minden esetben új képernyőre történik, ahonnan a végén Enter-t ütve térhetünk vissza a menühöz.

a bejegyzések listája

Az bejegyzéseket rendezetten, 1-1 sorban jeleníti meg. Minden 20. bejegyzésnél megáll, Entert ütve lehet folytatást kérni. Amennyiben nincs naplóbejegyzés, ezt üzenetben jeleníti meg.

b tanárok listája
c tantárgyak listája
d csoportok listája

A bejegyzések alapján a pillanatnyi állapotot tükröző ismétlésmentes rendezett listát jeleníti meg.

7 statisztika (érettségi feladat 2-7)

Az egyes statisztikai számítások, a feladatban megadott kérdésekre a válaszok. Az egyes részfeladatoknál az érettségi feladatban megadott részfeladat sorszámok is megjelennek (2—7).

a bejegyzések száma

A programban aktuálisan tárolt bejegyzések száma.

b összes óraszám

A programban aktuálisan tárolt bejegyzésekre az óraszámok összege.

c adott tanár óraszáma

A tanár nevét kell megadni, amelynek létét a program az ehhez készített tanár listában (mint 6b) ellenőriz. Három hibás névmegadást követően a programrész az óraszám összegzése nélkül befejeződik.

d osztályfőnökök listája

A bejegyzések között található „osztalyfonoki” tantárgyhoz tartozó osztályok és tanárok listába gyűjtése. Ez a lista is rendezett, az évfolyam számával kezdődő osztály jelölést számértékként veszi figyelembe. A feladat elvégzése során – az eredeti feladat kiírásnak megfelelően – a lista az `of.txt` fájlba is kiírásra kerül.

e csoportbontás lekérdezés

Az osztály és a tantárgy nevét kell megadni. Mindkettőnél ellenőrzi az adat létezését és bármelyiknél, ha háromszor olyan adatot írunk be, ami nem szerepel a bejegyzések között, akkor a részfeladat megoldása befejeződik. Amennyiben az adatok léteznek, de a megadott párosítás nem szerepel a bejegyzések között, akkor a program jelzi, hogy az adott osztály nem tanulja az adott tárgyat. A csoportbontásban tanult tárgy esetén a program nem vizsgálja a csoportok számát, többfelé bontottságát. A program nem vizsgálja, hogy létező osztály (abcd) bontásáról van-e szó, vagy a fakultációk jelzésére használt, évfolyamszintű ’x’ csoportról.

f tanárok száma

A bejegyzésekben szereplő tanárok listájának a számosságát adja meg.

g teljes statisztika

Új konzolablakban a statisztika eddigi részleteit egymás után kiírja. A megjelenítés és ellenőrzés minden tekintetben a részfeladatoknál olvashatókkal azonos. Teljes statisztika nem csak a ’g’ leütésével kérhető. Bármely, korábban nem szereplő választás ugyanezt eredményezi.

8 kilépés

A program bezárása. Az eredeti feladathoz alkalmazkodva, a program nem tartalmazza a bejegyzés adatainak fájlba mentését.

Minta bemenet és kimenet

A felhasználásra mutatnak példát az alábbi képernyőképek.

Adatok beolvasása fájlból

Menüpont: 1

```
*****************   TANTÁRGYFELOSZTÁS   *****************
 MENÜ
        1  input fájlból
        2  input konzolról
        3  egyedi bejegyzés hozzáadása
        4  egyedi bejegyzés törlése
        5  napló törlése
        6  listázás
                a  bejegyzések listája
                b  tanárok listája
                c  tantárgyak listája
                d  csoportok listája
        7  statisztika (érettségi feladat 2-7)
                a  bejegyzések száma
                b  összes óraszám
                c  adott tanár óraszáma
                d  osztályfőnökök listája
                e  csoportbontás lekérdezés
                f  tanárok száma
                g  teljes statisztika
        8  kilépés


        Válasszon a menü sorszámok közül: 1
A napló nem volt üres. A korábbi adatok törlödtek
        Beolvasás kész.

Új menüpont választáshoz üssön Entert!
```

Folyamatos kézi adatbevitel

Menüpont: 2

Bevitel befejezése bejegyzés kezdéssel

```
*****************   FOLYAMATOS KÉZI ADATBEVITEL   *****************

Adatbevitel megszakítása: szöveg nélkül Enter vagy óraszám értéke -1

        Tanár neve [maxhossz 30]: Csincsilla Csilla
        Tantárgy [maxhossz 20]: informatika
        Csoport [maxhossz 7]: 9.a
        -1 | Óraszám [0;40]: 2

        Tanár neve [maxhossz 30]: Teve Teofil
        Tantárgy [maxhossz 20]: informatika
        Csoport [maxhossz 7]: 9.a
        -1 | Óraszám [0;40]: 2

        Tanár neve [maxhossz 30]:
                Ez a bejegyzés nem lesz mentve.
        Beolvasás vége

Új menüpont választáshoz üssön Entert!
```

Utolsó bejegyzés eldobása -1 óra megadásával

```
*****************   FOLYAMATOS KÉZI ADATBEVITEL   *****************

Adatbevitel megszakítása: szöveg nélkül Enter vagy óraszám értéke -1

A napló nem üres. Hozzáfűzés esetén duplikátum jöhet létre.
        a: append - jelenlegi bejegyzések megtartása
        c: clear  - jelenlegi bejegyzések törlése
válasszon (a|c): a
        Tanár neve [maxhossz 30]: Albatrosz Aladin
        Tantárgy [maxhossz 20]: osztalyfonoki
        Csoport [maxhossz 7]: 9.a
        -1 | Óraszám [0;40]: 1

        Tanár neve [maxhossz 30]: nincs
        Tantárgy [maxhossz 20]: nincs
        Csoport [maxhossz 7]: nincs
        -1 | Óraszám [0;40]: -1
                Ez a bejegyzés nem lesz mentve.
        Beolvasás vége

Új menüpont választáshoz üssön Entert!
```

Bejegyzések listájának kiegészítése

```
*****************   FOLYAMATOS KÉZI ADATBEVITEL   *****************

Adatbevitel megszakítása: szöveg nélkül Enter vagy óraszám értéke -1

A napló nem üres. Hozzáfűzés esetén duplikátum jöhet létre.
        a: append - jelenlegi bejegyzések megtartása
        c: clear  - jelenlegi bejegyzések törlése
válasszon (a|c): c
        Tanár neve [maxhossz 30]: SzTZs
        Tantárgy [maxhossz 20]: inf
        Csoport [maxhossz 7]: 9.c
        -1 | Óraszám [0;40]: 2

        Tanár neve [maxhossz 30]:
                Ez a bejegyzés nem lesz mentve.
        Beolvasás vége

Új menüpont választáshoz üssön Entert!
```

Egyedi adat törlése

Menüpont: 4

Bejegyzés törlése sikertelen

```
*****************   TANTÁRGYFELOSZTÁS   *****************
  MENÜ
        1  input fájlból
        2  input konzolról
        3  egyedi bejegyzés hozzáadása
        4  egyedi bejegyzés törlése
        5  napló törlése
        6  listázás
              a  bejegyzések listája
              b  tanárok listája
              c  tantárgyak listája
              d  csoportok listája
        7  statisztika (érettségi feladat 2-7)
              a  bejegyzések száma
              b  összes óraszám
              c  adott tanár óraszáma
              d  osztályfőnökök listája
              e  csoportbontás lekérdezés
              f  tanárok száma
              g  teljes statisztika
        8  kilépés


        Válasszon a menü sorszámok közül: 4

Bejegyzés törlése pontos egyezés esetén.
Adja meg a törlendö bejegyzés adatait
        Tanár   : SzTZs
        Tantárgy: inf
        Csoport : 9.c
        Óraszám : 1

Nincs megfelelő bejegyzés.

Új menüpont választáshoz üssön Entert!
```

Bejegyzés törlése sikeres

```
*****************   TANTÁRGYFELOSZTÁS   *****************
  MENÜ
        1  input fájlból
        2  input konzolról
        3  egyedi bejegyzés hozzáadása
        4  egyedi bejegyzés törlése
        5  napló törlése
        6  listázás
              a  bejegyzések listája
              b  tanárok listája
              c  tantárgyak listája
              d  csoportok listája
        7  statisztika (érettségi feladat 2-7)
              a  bejegyzések száma
              b  összes óraszám
              c  adott tanár óraszáma
              d  osztályfőnökök listája
              e  csoportbontás lekérdezés
              f  tanárok száma
              g  teljes statisztika
        8  kilépés


        Válasszon a menü sorszámok közül: 4

Bejegyzés törlése pontos egyezés esetén.
Adja meg a törlendö bejegyzés adatait
        Tanár   : SzTZs
        Tantárgy: inf
        Csoport : 9.c
        Óraszám : 2

        A megadott bejegyzés törölve.

Új menüpont választáshoz üssön Entert!
```

Listák megjelenítése

Menüpont: 6 a

```
*****************   TANTÁRGYFELOSZTÁS BEJEGYZÉSEK   *****************
| Csincsilla Csilla              | informatika          | 9.a      |    2 |
| Teve Teofil                    | informatika          | 9.a      |    2 |
| Albatrosz Aladin               | osztalyfonoki        | 9.a      |    1 |

Új menüpont választáshoz üssön Entert!
```

Menüpont: 6 c

```
*****************   TANTÁRGYAK LISTÁJA   *****************
angol
biologia
eletvitel
enek
etika
filozofia
fizika
foldrajz
informatika
kemia
magyar
matematika
nemet
osztalyfonoki
testneveles
tortenelem
vizualis

Új menüpont választáshoz üssön Entert!
```

Paraméteres statisztikai adat lekérdezése

Menüpont: 7 e

Paraméter megadás javítással

```
*****************   TANTÁRGYFELOSZTÁS   *****************
  MENÜ
        1  input fájlból
        2  input konzolról
        3  egyedi bejegyzés hozzáadása
        4  egyedi bejegyzés törlése
        5  napló törlése
        6  listázás
                a  bejegyzések listája
                b  tanárok listája
                c  tantárgyak listája
                d  csoportok listája
        7  statisztika (érettségi feladat 2-7)
                a  bejegyzések száma
                b  összes óraszám
                c  adott tanár óraszáma
                d  osztályfőnökök listája
                e  csoportbontás lekérdezés
                f  tanárok száma
                g  teljes statisztika
        8  kilépés


        Válasszon a menü sorszámok közül: 7

                Adja meg a feladat betűjelét: e
6. feladat
        Osztály: 9.e
                Nincs ilyen jelű osztály. Adja meg újra.
        Osztály: 9.a
        Tantárgy: informatika
                csoportbontásban tanulják.

Új menüpont választáshoz üssön Entert!
```

Paraméter megadása sikertelen

```
*****************   TANTÁRGYFELOSZTÁS   *****************
  MENÜ
        1  input fájlból
        2  input konzolról
        3  egyedi bejegyzés hozzáadása
        4  egyedi bejegyzés törlése
        5  napló törlése
        6  listázás
                a  bejegyzések listája
                b  tanárok listája
                c  tantárgyak listája
                d  csoportok listája
        7  statisztika (érettségi feladat 2-7)
                a  bejegyzések száma
                b  összes óraszám
                c  adott tanár óraszáma
                d  osztályfőnökök listája
                e  csoportbontás lekérdezés
                f  tanárok száma
                g  teljes statisztika
        8  kilépés


        Válasszon a menü sorszámok közül: 7

                Adja meg a feladat betűjelét: e
6. feladat
        Osztály: inf
                Nincs ilyen jelű osztály. Adja meg újra.
        Osztály: 9.c
        Tantárgy: Inf
                Nincs ilyen nevű tantárgy. Adja meg újra.
        Tantárgy: informatika
                Nincs ilyen nevű tantárgy. Adja meg újra.
        Tantárgy: info
                Túl sok hibás próba. Nézze meg a tantárgyak listáját.

Új menüpont választáshoz üssön Entert!
```

Teljes statisztika

Menüpont: 7 g

Minimális bejegyzésre

```
*****************   STATISZTIKA   *****************
2. feladat
        A fájlban 1 bejegyzés van.
3. feladat
        Az iskolában a heti összóraszám: 1
4. feladat
        Tanár neve: Albatrosz Aladin
                A tanár heti óraszáma: 1
5. feladat
         9.a - Albatrosz Aladin
6. feladat
        Osztály: 9.a
        Tantárgy: osztalyfonoki
                osztályszinten tanulják.
7. feladat
        Az iskolában 1 tanár tanít

        Új menüpont választáshoz üssön Entert!
```

Eredeti forrásból beolvasott bejegyzésekre

```
*****************   STATISZTIKA   *****************
2. feladat
        A fájlban 329 bejegyzés van.
3. feladat
        Az iskolában a heti összóraszám: 1016
4. feladat
        Tanár neve: Puma
                Nincs ilyen nevű tanár. Adja meg újra.
        Tanár neve: Puma Pongor
                A tanár heti óraszáma: 23
5. feladat
         9.a - Albatrosz Aladin
         9.b - Hangya Hanna
         9.c - Zerge Zenina
         9.d - Medve Melani
        10.a - Farkas Farkas
        10.b - Bivaly Biti
        10.c - Giliszta Gilbert
        10.d - Borz Borka
        11.a - Pekry Petra
        11.b - Lemming Lea
        11.c - Cet Celina
        11.d - Panda Patrik
        12.a - Kaffer Kada
        12.b - Pulyka Pozsinka
        12.c - Vidra Viktor
        12.d - Puma Pongor
6. feladat
        Osztály: 9.a
        Tantárgy: fizika
                osztályszinten tanulják.
7. feladat
        Az iskolában 49 tanár tanít

        Új menüpont választáshoz üssön Entert!
```

Fejlesztői dokumentáció

Feladat

A tantárgyfelosztás a tanév tervezésének alapvető dokumentuma. A tantárgyfelosztás azt tartalmazza, hogy a tanárok a tantárgyaikat mely osztályokban, hány órában tanítják. A feladatban egy négy évfolyamos gimnázium tantárgyfelosztásának adatait kell elemezni. A tantárgyfelosztást ezúttal egy adatbáziskezelő programmal előállított, egyszerű szerkezetű szöveges állományból kapjuk az alábbi minta szerint. (Minden bejegyzést négy sor tárol.)

Albatrosz Aladin
biologia
9.a
2

A bejegyzés megadja, hogy Albatrosz Aladin tanár úr biológiát (biologia) fog tanítani a 9.a osztályban heti 2 órában. Ha az osztály betűjele x, akkor évfolyam szintű csoportról van szó. Például a 12. évfolyamon több német csoport van, de nem osztályok szerint, hanem előismeret alapján van bontva. Az osztályfőnököket arról ismerhetjük fel, hogy ők tartják az osztályfőnöki (osztalyfonoki) órát. Egyes osztályokban bizonyos tantárgyakat a tanulók csoportbontásban tanulnak: ekkor az adott tantárgyra és osztályra (vagy évfolyamra) két vagy több bejegyzést is tartalmaz a tantárgyfelosztás.

A megoldás során – az érettségi feladattól eltérően nem használható fel méretkorlát Az érettségi feladatban adottaktól eltérően, a heti óraszám nem negatív és maximum 40 óra lehet.

A program

- megadja a beolvasott bejegyzések számát;
- megadja, hogy az iskolában hetente összesen hány tanítási óra van;
- megadja, hogy egy adott tanár hetente hány órában tanít;
- listát készít, amelyben megadja az egyes osztályok osztályfőnökeinek a nevét;
- megvizsgálja, hogy egy adott osztály egy adott tárgyat csoportbontásban tanul-e (ekkor az adott tantárgyra és osztályra két bejegyzést is tartalmaz a tantárgyfelosztás);
- megadja, hogy hány tanár dolgozik az iskolában.

Specifikáció

A forráskód és dokumentálás elemeinek helye

A programkód egyes részeinek definícióit a C standard könyvtárai tartalmazzák. A programkód ezeken túlmenően a funkcionalitás mentén több fordítási egységre van bontva. Ezen modulok és dokumentálásához szükséges fájlok a következők:

Modul	Magyarázat
Szabványos C könyvtárak	
stdio.h	Konzol és fájlműveletek
stdlib.h	NULL érték kezelése
stdbool.h	Logikai (bool) adattípus
string.h	Szövegkezelő függvények
Fejléc és forrásfájlok (tanfel\)	
main.c	C forráskód
datamanager.h datamanager.c	Adatbeolvasás és lista készítés
edulist.h edulist.c	Bejegyzések listájának műveletei
edurec.h edurec.c	Bejegyzés rekord műveletei
statistics.h statistics.c	Lekérdezések, statisztikák
stringset.h, stringset.c	Szöveges adatot tartalmazó felsorolás műveletei
Egyéb fájlok a fejlesztéshez és dokumentáláshoz	
doc\	Mappa: dokumentáció fájljai, feladat forrása
test\	Mappa: a tesztelés dokumentálása

Programparaméterek

Konstans

A program előfordításra megadott konstansai a szöveges adatok maximális mérete, mely az `edurec.h` fejlécben található:

```
#define TMAX 30
#define SMAX 20
#define GMAX 7
```

Valamint az adatbevitel megszakításának jelzése az óraszámnál a `datamanager.h` fejlécben:

```
#define STORNO -1
```

Típus

A feladatban megadott „bejegyzés” adatstruktúrát az `edurec.h` fejlécfájl tartalmazza. Programozói döntés, hogy az óraszám lehet 0, továbbá a -1 értéket hibakezeléshez használom. Másik lehetséges megoldás, hogy a 0 nem elfogadott érték, ebben az esetben ez lehet jelzése a hibás adatnak. További lehetőség egy logikai típusú adattag hozzáadása.

```
typedef struct edurec
{
    char teacher[TMAX+1];
    char subject[SMAX+1];
    char group[GMAX+1];
    int ora;
}EduRec;
```

A bejegyzések listájának definícióját az `edulist.h` fejlécfájl tartalmazza. A lista egyirányban láncolt, az újabb bejegyzések mindig a lista végéhez fűződnek. A lista elemei `EduNode` csomópontok, amely tartalmazza az `EduRec` típusú adatot. A lista kezelését egy pointereket és hosszt tároló `EduList` típus valósítja meg:

```
typedef struct edunode{
    EduRec adat;
    struct edunode* next;
}EduNode;
typedef struct edulist{
    EduNode* first;
    EduNode* last;
    int length;
}EduList;
```

A feladat szerint csak az osztály-osztályfőnök adatokat kell kilistázni, azonban a paraméterek megadásánál is fontos a szöveges adatok típusonkénti áttekintése, ezért a szövegeket tartalmazó felsorolások készítésére külön típus van, melyhez a definíciókat `stringset.h` tartalmazza. Az adatok ismétlődésmentesen kerülnek be egy bináris fába. Az adatlista jellemzője, hogy melyik adatokat rendezi listába, ezeknek pedig a megengedett maximális hossza közös tulajdonsága. Ezért az dinamikus adatstruktúra adata - a `StringItem` – a \0-val lezárt, adott méretre foglalt szövegen kívül tartalmazza a felhasználható méretet (kapacitást) is. A listák használatánál két elv érvényesülhet: egyrészt a bejegyzések felvételével együtt lehet építeni az egyes adatmezők bináris fáit. Ebben az esetben az összes módosítást az összes listában érvényesíteni kell, ami az ismétlődések miatt a lista törlését és újraépítését jelenti. Másik módszer, hogy a bináris fát csak a lista kérésekor, illetve az adatbekérés ellenőrzésekor építjük fel és a feladat befejezésével töröljük. Ez utóbbi került megvalósításra.

```
typedef struct stringitem
{
    char* str;
    unsigned length;
}StringItem;
```

Az adatstruktúra elemei, csomópontjai – `StringNode` – az adaton kívül tartalmazza a szövegre jellemző kapacitást, a bejáráshoz a *right* és *left* pointert, a két-irányú bejárhatósághoz a *root* pointert. A binárisfa építését az ismétlések elkerülése és az adat gyors kereshetősége indokolja.

```
typedef struct stringnode{
    StringItem* strit;
    unsigned strlength;
    struct stringnode* root;
    struct stringnode* left;
    struct stringnode* right;
}StringNode;
```

A lista kezelését segítő `StringList` a bináris fa elejére mutató pointeren kívül a szövegre jellemző kapacitást is tartalmazza.

```
typedef struct stringlist{
    StringNode* first;
    unsigned strlength;
}StringList;
```

A program függvényei, szerepük

Főprogram: main.c

A program indításakor betöltődő modul feladata:

– program menüjének kiírása;

– a menüpont választás értelmezése;
– az egyes részfeladatok meghívása;
– konzolablak tartalmának törlése, menü ismételt megjelenítése.

A részfeladat választása az ’1’–’8’ karakterekkel történik. Ezen belül a ’6’ után az ’a’-’d’ karakterrel lehet a listát kiválasztani, valamint a ’7’ után az ’a’–’f’ pontokkal egy-egy kérdés választható, a ’g’ (és minden más betű is) a feladat teljes megoldását – az összes kérdésre sorban a válaszokat – adja meg.

```
int main()
{…
   EduList naplo = edulist_inic();
   int valasztas, feladat;
   do {
      valasztas = feladat = ' ';
      menu_print();
      printf("\n\n\tVálasszon a menü sorszámok közül: ");
      valasztas = getchar();
      while(getc(stdin) != '\n');
      switch(valasztas) {
         case '1':
            naplo = file_input(naplo);
            break;
/*…*/
         case '6':
            printf("\n\t\tAdja meg a lista betüjelét: ");
            feladat = getchar();
            while(getc(stdin) != '\n');
            switch(feladat) {
               case 'a':
                  system("cls");
                  printf(
"\n*****************   TANTÁRGYFELOSZTÁS BEJEGYZÉSEK   *****************\n");
                  edulist_print(naplo);
                  break;
/*…*/
            }
            break;
         case '7':
            printf("\n\t\tAdja meg a feladat betüjelét: ");
            feladat = getchar();
            while(getc(stdin) != '\n');
            switch(feladat) {
               case 'a':
                  F2(naplo);
/*…*/
               case 'g':
               default:
                  system("cls");
                  printf("\n*****************   STATISZTIKA   *****************\n");
                  F2(naplo);
/*…*/
                  F7(naplo);
                  break;
            }
            break;
         default:
            break;
      }
      if(valasztas != '8') {
         printf("\nÚj menüpont választáshoz üssön Entert!");
         while(getc(stdin) != '\n');
      }
   } while(valasztas != '8');
   if(naplo.length != 0)
      edulist_free(&naplo);
   return 0;
}
```

Alprogramok

edurec.h fájlban deklarált függvények

```
EduRec create_edurec(FILE* f);
```

Egy bejegyzés létrehozása fájlban tárolt adatokból. Ha a szöveges adat hosszabb, mint az adatra megadott maximális méret, akkor az enterig fennmaradó részt eldobja. Az óraszám értékét nem ellenőrzi, de sor végéig kiolvassa az adatokat. Ha bármely szöveges adat üres vagy véget ér a fájl vagy sikertelen az óraszám beolvasása, akkor az óraszám -1 értéke jelzi, hogy érvénytelen a bejegyzés.

```
EduRec input_edurec();
```

A `create_edurec` függvényhez hasonló, annak konzolos, kommunikatív változata, amely az `infput_stringfield` és `input_intfield` függvényet használja.

```
void lineprint_edurec(EduRec e);
```

Egy bejegyzés egysoros kiírása konzolra. Könnyen továbbfejleszthető más kimenetre írásra, de a feladat megoldása ezt nem igényelte.

edulist.h fájlban deklarált függvények

```
EduNode* create_edunode(EduRec er);
```

Egy bejegyzést tartalmazó listaelemnek memóriafoglalás és értékadás.

```
void delete_edunode(EduNode* en);
```

Egy listaelem törlése (memória felszabadítása).

```
EduList edulist_inic();
```

`EduNode`-okból álló lista inicializálása, üres lista.

```
EduList edulist_append(EduList el, EduRec er);
```

Az `EduNode`-okból álló *el* lista végére teszi az *er* adatot tartalmazó `EduNode`-ot.

```
EduNode* edulist_find(EduList el, EduRec er);
```

Az `EduNode`-okból álló *el* listában megkeresi az *er* adatot tartalmazó `EduNode`-ot. Ha megtalálja, akkor az erre mutató pointert, egyébként NULL pointert ad vissza.

```
void edulist_delete(EduList* el, EduRec er);
```

Az *el* listában az `edulist_find` függvény megkeresi *er*-t. Amennyiben megtalálja, akkor kiveszi a listából és a `delete_edunode` függvénnyel törli az `EduNode`-ot.

```
void edulist_free(EduList* el);
```

Az *el* lista minden `EduNode`-ját törli és üresre állítja a listát.

```
void edulist_print(EduList el);
```

A `lineprint_edurec` függvényt használva az *el* lista minden elemét képernyőre írja. A kiírás olvashatatlan futását megakadályozza, hogy 20 elemenként Enter leütésére vár a „lapozáshoz”.

stringset.h fájlban deklarált függvények

```
StringItem* createstring(char* s, unsigned maxlength);
```

Lezárónullával tárolt szöveget és az adattípusra jellemző maximális adathosszt tartalmazó adat létrehozása. Amennyiben a szöveg hosszabb a megadott *maxlength* méretnél, az adat nem jön létre. (A felhasználás jellege miatt ez a tulajdonság nem érvényesül.) Rövidebb szöveg esetén is *maxlength* méretű a szövegnek foglalt terület, a maradék terület \0 értékkel van feltöltve.

```
void deletestring(StringItem* strit);
```

A *strit* törlése, az általa lefoglalt memóriaterület felszabadítása.

```
StringSet stringset_inic(unsigned strlength);
```

`StringNode`-okat tartalmazó bináris fa inicializálása, üres binfa.

```
StringNode* stringnode_find(StringNode* sn, char* s);
StringNode* stringset_find(StringSet ss, char* s);
```

A `stringnode_find` a megadott helytől keresi a bináris fában az *s* szöveget. A `stringset_find` a bináris fa gyökerére meghívja a `stringnode_find` függvényt.

```
StringNode* stringnode_insertstr(StringNode* sn, char* s, unsigned maxlength);
void stringset_insert(StringSet* ss, char* s);
```

Amennyiben a `stringnode_find` függvény nem talál egyezést, akkor új `StringNode` létrehozása és beillesztése a bináris fában az ábécé rendezés szerinti helyre. A `stringset_insert` ellenőrzi a szöveg hosszát, a hibát kijelzi. Megfelelő adat esetén, amennyiben még nincs benne (`stringnode_find`), a `stringnode_insertstr` függvény beilleszti a stringet.

```
StringNode* stringnode_dropstr(StringNode* sn, char* s);
void stringset_delete(StringSet* ss, char* s);
```

Adott *s* szöveg megtalálása (`stringnode_find`) esetén a kapott `StringNode` törlése a bináris fából. A `stringset_delete` a `stringnode_dropstr` függvényt hívja meg a bináris fa gyökerére

```
void stringnode_fprint(FILE* f, StringNode sn);
void stringset_fprint(FILE* f, StringSet ss);
```

Az *ss* `StringSet` minden elemének soronkénti kiírása. A `stringnode_fprint` rekurzív bal-közép-jobb bejárással megy végig az adatokon, a `stringset_fprint` ellenőrzi, hogy van-e adat és meghívja a bináris fa gyökerére a rekurzív függvényt.

```
unsigned stringnode_length(StringNode sn);
unsigned stringset_length(StringSet ss);
```

Az *ss* `StringSet` elemeinek száma. A `stringnode_length` rekurzív közép-bal-jobb bejárással megy végig az adatokon, számlálja a „közép” állapotot a `stringset_length` üras binfára 0-t ad, és meghívja a bináris fa gyökerére a `stringnode_length` függvényt.

```
void stringnode_free(StringNode* sn);
void stringset_free(StringSet* ss);
```

Az *ss* `StringSet` összes elemének eltávolítása törlése (memóriaterület felszabadítása) a bináris fából. A `stringnode_free` egy node részfáját bal-jobb-közép sorrendben rekurzívan törli, a szülő rámutató pointerét NULL-ra állítja és a `deletestring` hívással törli a node-ot. A `stringset_free` a bináris fa létezését ellenőrzi, ha nem üres, akkor meghívja a gyökérre a `stringnode_free` függvényt.

```
bool stringset_isitem(StringSet ss, char* s, unsigned maxlength, const char*
        question, const char* message);
```

Ellenőrzött adatbevitelhez (`input_stringfield`), a bekért adat hosszát ellenőrzi és megnézi, hogy az adatnak megfelelő kigyűjtésben benne van-e. Hiba esetén az adatot újra bekéri, de összesen csak 3 próbát engedélyez. Sikeres adat-megnevezés vagy három hiba után az eredményt logikai értékkel jelzi.

statistics.h fájlban deklarált függvények

Az egyes – érettségi feladatként is szereplő – feladatok függvényei. A feladatok számozása az érettségi feladatbeli sorszámnak felelnek meg, ezért 2-7 feladatról van szó. A megoldásoknál, kiírások szövegénél az érettségi feladat elvárásai érvényesülnek.

```
void F2(EduList el);
```

Az *el* `EduList`-be bejegyzett elemek számát adja meg.

```
void F3(EduList el);
```

Az *el* `EduList`-be bejegyzett elemek óraszámainak összegét számolja ki.

```
void F4(EduList el);
```

Az *el* `EduList` alapján `StringSet`-et készít a tanárok neveiből. A bekért nevet ebben ellenőrzi (`stringset_isitem`). Ha sikeres az ellenőrzés, akkor kiszámolja a tanár heti óraszámát. Ha az *el* lista üres vagy nem sikerült egy, a kigyűjtésben megtalálható tanár nevét beírni, akkor nem számol eredményt, csak kiírja a hiba jellegét. Végül törli a tanár nevek bináris fáját.

```
void F5(EduList el);
```

Az *el* `EduList`-ből kigyűjti az „osztalyfonoki" bejegyzésekhez tartozó csoport és tanárneveket, az adatpárokat a megadott formátumban összefűzi és `StringSet`-et készít belőle. A felsorolást a konzolra és az `of.txt`-be kiírja, majd törli.

```
void F6(EduList el);
```

Az el `EduList`-be bejegyzett elemekből létrehozza az osztályok és tantárgyak felsorolását a bináris fákban, ellenőrzi (három próbát engedve), hogy a beírt adatok léteznek-e. Az ellenőrzés után a két `StringSet` törlésre kerül. Amennyiben az osztály és csoport létezik, megszámlálja (2-ig) az együttes előfordulásukat. Az eredmény háromféle válasz lehet: a 0 (nem tanított), 1 (osztályszintű) illetve 2 (csoportbontás) találatok esetére.

```
void F7(EduList el);
```

Az *el* `EduList`-be bejegyzett adatok alapján `StringSet`-et készít a tanárok neveiből. A felsorolás elemszámának megadása után a bináris fát törli.

datamanager.h fájlban deklarált függvények

```
void menu_print();
```

Konzolon megjeleníti a menü szövegét.

```
void input_stringfield(const char * label, char* field, unsigned capacity);
```

Szöveges adat bekérése konzolon. A *label* változóban megadható a kért adathoz tartozó szöveg. A szöveget legfeljebb a *capacity* paraméterként megadott hosszúságig tárolja, a többit – Enter-ig – eldobja.

```
void input_intfield(const char* label, int* field, int minvalue, int maxvalue);
```

Egész bekérése konzolon. A *label* változóban megadható a kért adathoz tartozó szöveg. Az értékhatáron (zárt intervallumon) belül található egész értéket, illetve a STORNO értéket fogadja el, egyéb esetben újra kéri az adatot.

```
EduList file_input(EduList el);
```

Amennyiben a megadott fájl megnyitása sikeres, az adatokat beolvassa, végül bezárja a fájlt.

```
EduList stdin_input(EduList el);
```

A konzolon beírt bejegyzéseket – az első érvénytelen bejegyzésig – beteszi az *el* `EduList`-be. Amennyiben az adatfelvitel elején el nem üres, akkor dönthetünk arról, hogy az adatokat hozzáfűzzük vagy előbb a régi bejegyzéseket töröljük.

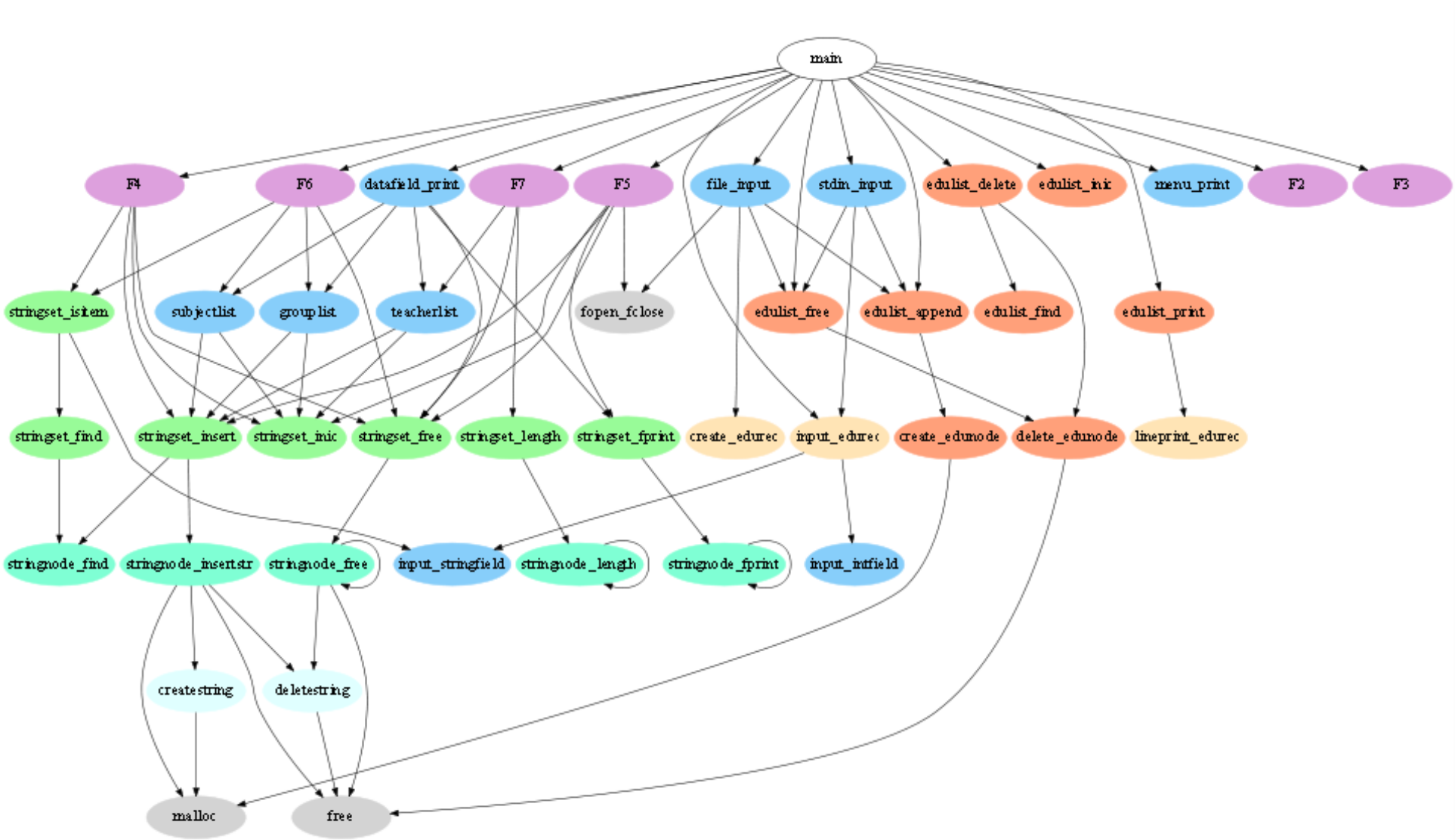
```
StringSet teacherlist(EduList el);
StringSet subjectlist(EduList el);
StringSet grouplist(EduList el);
```

Ez a három függvény létrehozza az adattagokból az egyes `StringSet`-eket.

```
void datafield_print(EduList el, StringSet (*fieldlist)(EduList));
```

A függvénypointerben megadott `StringSet` bináris fát elkészíti, tartalmát kiírja konzolra majd törli.

Programfelépítés[*]



[*] Nem kötelező része a beadandónak. A diagram a Graphviz 2.38.0 programmal készült.

Tesztelés

A teszteléshez használt adatok, műveletek és eredményfájlok. (A képfájlok a Felhasználói dokumentációban is elérhetők, de nem az itt feltüntetett sorrendben.)

Állományok	Műveletek: menüpont választás és beírt adatok
be1.txt, ki1.png, of1.txt	**3** ≪be1.txt≫ **7 g** (Albatrosz Aladin 9.a osztalyfönöki) **8**
be2.txt, ki2.png ki2v.png	**2** ≪be2.txt≫ **7 e** (9.e 9.a informatika)
be3.txt, ki3.png, ki3v.png	**2** (a) ≪be3.txt≫ (nincs nincs nincs -1) **6 a**
be4.txt, ki4.png	**5 2** ≪be4.txt (1319 sor)≫ (-1) **7 a 6 c**
be5.txt, ki5.png, ki5v.png, of5.txt	**1** ≪be5.txt≫ **7 g** (Puma Pongor 11.a kemia)
be6.txt, ki6.png, ki6v.png	**2** (c) ≪be6.txt≫ **7 e** (inf 9.c Inf informatika info)
be7.txt (üres), ki7.txt, ki7v.png	**4** (SzTZs inf 9.c 1) **4** (SzTZs inf 9.c 2)
be8.txt ki8.png, of8.txt	**1** ≪be8.txt (beosztas.txt)≫ **7 g** (Puma Pongor 9.a fizika) **8**

Fejlesztési lehetőségek

A program funkcionalitása a felhasználói igényeknek megfelelően bővíthető. A feldolgozott adatstruktúra kiegészíthető az évfolyam megjelölésével (mely szakkör esetén „nem meghatározott” lehet), az osztály is lehet „nem meghatározott” az „x” jelölés helyett és új adattagként az egyes csoportok nevét is tárolhatjuk. Ezzel a feltehető statisztikai kérdések száma is jelentősen bővülhet.

A feldolgozás alapadata – az eredeti feladat követése miatt – a bejegyzés, de ez átszervezhető relációs adatbázis kapcsolatokra. Ekkor érdemben lehet foglalkozni a tanár, tantárgy, illetve csoport neveknek a kezelésével (felvétel, törlés, módosítás, listába szervezés, tantervvel való kapcsolatok kezelése).