Programozás (1. szemeszter) beadandó feladat

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

Kurzuskód: IP-18PROGEG

2020. február 02.

Előszó

Az ELTE-IK első szemeszterében tanult Programozás (régebben Programozási alapismeretek) tárgy teljesítéséhez tartozik egy adott feladathoz a program tervezése, megoldása és dokumentációjának elkészítése. A téma minden évben más, amihez 30-50 megoldandó feladat tartozik. A hallgatók véletlenszerűen kapják a feladatok egyikét.

A korábbi évek feladatait a mester.inf.elte.hu gyakorlófelületen a középhaladó szinten lehet megtalálni csak úgy, mint a közoktatási és szakképzési informatika érettségik programozás feladatait. A feladattípusok hasonlóak, leginkább a részfeladatok mennyiségében és az IO műveletek megvalósításában van eltérés. Az elvárt megoldás azonban teljesen más. Ezzel a megoldással azt szeretném megmutatni, hogy hogyan nézne ki egy érettségi feladat megoldása, ha a programtervező informatikus képzés első szemeszterének beadandó feladata lenne.

Az összevetéshez tartozik egy harmadik megoldás is: a BME-VIK mérnökinformatikus képzésén, az első szemeszter végén Programozás alapjai 1 tárgyból beadandó program és dokumentációja. Ott a feladat szabadon választható, de az érettségi feladatok többsége alapja lehet egy programnak.

Jelen esetben az összevetés alapja az érettségi feladatok közül a 2019. májusi idegennyelvű emelt szintű közismereti informatika érettségi 4. feladata.

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 adatbázis-kezelő 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 felhasználható, hogy a fájl maximum 1000 bejegyzést (azaz legfeljebb 4000 sort) tartalmaz; az iskolában legfeljebb 100 tanár és legfeljebb 50 osztály van, továbbá minden osztálynak pontosan egy osztályfőnöke van. Az érettségi feladatban adottaktól eltérően, a heti óraszám maximum 40 óra lehet.

Készítsen programot, amely

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

Futási környezet

IBM PC, exe futtatására alkalmas, 32 vagy 64-bites operációs rendszer (pl. Windows 10). Konzol alkalmazás.

Használat

A program indítása

A program `tanfel\main.cpp` fordítása után a létrejött `.exe` fájl kiválasztásával indítható.

A program bemenete

A program az adatokat a **billentyűzet**ről olvassa be a következő sorrendben:

#	Adat	Magyarázat
0.	futtatási mód	'n': nem ellenőrzött vagy 'y': ellenőrzött adatokkal
1.	line	Az adatsorok száma ($4 \leq line \leq 4000$).
2.	Bejegyzés$_1$.Tanár	A tantárgyfelosztás első bejegyzésében a tanár neve.
3.	Bejegyzés$_1$.Tantárgy	A tantárgyfelosztás első bejegyzésében a tantárgy neve.
4.	Bejegyzés$_1$.Osztály	A tantárgyfelosztás első bejegyzésében az osztály megnevezése.
5.	Bejegyzés$_1$.Óraszám	A tantárgyfelosztás első bejegyzésében a heti óraszám.
...		
line –2.	Bejegyzés$_{(line/4)}$.Tanár	A tantárgyfelosztás (line/4). bejegyzésében a tanár neve.
line – 1	Bejegyzés$_{(line/4)}$.Tantárgy	A tantárgyfelosztás (line/4). bejegyzésében a tantárgy neve.
line	Bejegyzés$_{(line/4)}$.Osztály	A tantárgyfelosztás (line/4). bejegyzésében az osztály megnevezése.
line + 1	Bejegyzés$_{(line/4)}$.Óraszám	A tantárgyfelosztás (line/4). bejegyzésében a heti óraszám.
line + 2	NÉV	Egy megadott tanár neve
line + 3	OSZTÁLY	Egy megadott osztály neve
line + 4	TANTÁRGY	Egy megadott tantárgy neve

A program kimenete

A program a standard kimenetre minden részfeladat eredménye előtt egy-egy, egyetlen # karaktert tartalmazó sort ír ki, amelyet a részfeladat eredményét tartalmazó egy vagy több sor követ. Felhasználóbarát üzenetek a standard hibaüzenet kimeneten jelennek meg.

1. részfeladat: kiírja, hogy hány bejegyzés található az állományban.
2. részfeladat: kiírja a heti tanítási órák összesített számát.
3. részfeladat: kiírja a NÉV-vel megadott tanár heti óraszámát.
4. részfeladat: listát készít az „osztály – osztályfőnök" párosításokról. Ellenőrzött futtatással az eredményt kiírja az of.txt fájlba
5. részfeladat: kiírja, hogy a megadott OSZTÁLY a megadott TANTÁRGY-at csoportbontásban vagy osztályszinten tanulja-e. (Feltételezhető, hogy a megadott osztály tanulja a megadott tantárgyat, de a program jelzi, ha nem talált megfelelő párt.)
6. részfeladat: kiírja az iskolában tanító tanárok számát.

Minta bemenet és kimenet

Bemenet	**Kimenet**
1316 Albatrosz Aladin biologia 9.a 2 Albatrosz Aladin osztalyfonoki 9.a ... Csincsilla Csilla matematika 9.x 2	# 329 # 1016 # 24 # (of.txt tartalma is) 9.a - Albatrosz Aladin 9.b - Hangya Hanna ... # Csoportbontásban tanulják. #

...	49

```
C:\Users\sztzs\Desktop\tanfel\bin\Release>tanfel
A tantargyfelosztas elemzese
Valasszon az alabbi beolvasasi lehetosegek kozul:

               Egy SZAM beirasaval (vegen enter) a 'biro' stilusu konzolos kiertekeles indul.

               n|N      konzolrol adott szamu rekord beolvasasa ellenorzes nelkul.
               y|Y      konzolrol adott szamu rekord beolvasasa adatonkent ellenorizve.
A fentiektol eltero kezdetu valaszokra a program futasa vegeter.

valasztas: y
*********************************************************************************************

Adja meg a beolvasando adatok (4-gyel oszthato) szamat: 1316
Tanar nev = Albatrosz Aladin
```

... bemenet.txt ...

```
Oraszam = 2
329 db vart bejegyzesbol sikeresen beolvasott rekordok szama: 329
Tanari oraszamhoz egy tanar neve = Szarvas Szamanta
Csoportbontas jellemzesehez
osztaly = 9.c
tantargy = informatika
#
1. feladat
rekordok szama: 329
#
2. feladat
az iskolaban a heti osszoraszam: 1016
#
3. feladat
A tanar heti oraszama: 18
#
4. feladat
osztalyfonokok:
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
#
5. feladat
Csoportbontasban tanuljak
#
6. feladat
tanarok szama: 49
Press any key to continue . . .
```

Hibalehetőségek

Az egyes bemeneti adatokat a fenti mintának megfelelően kell megadni. Hiba, ha a sorok száma nem egész szám, vagy nem esik a `[4, 4000]` intervallumba vagy nem többszöröse 4-nek. Hiba, ha az óraszám nem egész szám és tesztelt futtatáskor nem a [0, 40] intervallumban van. Tesztelt adatok esetén hiba, ha a NÉV, OSZTÁLY, illetve TANTÁRGY nem található meg a bejegyzések között.

Mintafutás hibás bemeneti adatok esetén:

```
C:\Users\sztzs\Desktop\tanfel\bin\Release>tanfel
A tantargyfelosztas elemzese
Valasszon az alabbi beolvasasi lehetosegek kozul:

               Egy SZAM beirasaval (vegen enter) a 'biro' stilusu konzolos kiertekeles indul.

               n|N      konzolrol adott szamu rekord beolvasasa ellenorzes nelkul.
               y|Y      konzolrol adott szamu rekord beolvasasa adatonkent ellenorizve.
A fentiektol eltero kezdetu valaszokra a program futasa vegeter.

valasztas: YX
********************************************************************************************************

Adja meg a beolvasando adatok (4-gyel oszthato) szamat: i
Kerem az adatsorok szamat [4; 4000]: 9
Az adatok nem adnak ki teljes rekordokat. (9-nak nem osztoja a 4.)
Kerem az adatsorok szamat [4; 4000]: 8
Tanar nev = Fizy Kata Nora
Tantargy nev = osztalyfonoki
Osztaly nev= 11.d
Oraszam = 1
Tanar nev = Fizy Kata Nora
Tantargy nev = fizika
Osztaly nev= 11.d
Oraszam = 100
Irrealis oraszam
Hibas rekord.
               Kilepes a bevitelbol: x
               Rekord ismetlese:     ENTER

Tanar nev = Fizy Kata Nora
Tantargy nev = fizika
Osztaly nev= 11.d
Oraszam = 100
Irrealis oraszam
Hibas rekord.
               Kilepes a bevitelbol: x
               Rekord ismetlese:     ENTER
x
2 db vart bejegyzesbol sikeresen beolvasott rekordok szama: 1
FIGYELMEZTETES: A megadottnal kevesebb rekordot sikerult beolvasni.
Tanari oraszamhoz egy tanar neve = Fizy Kati
Ismeretlen tanar. Kerem, adja meg ujra: Fizy Kata Nora
Csoportbontas jellemzesehez
osztaly = 11d
Nincs ilyen nevu osztaly. Kerem, adja meg ujra: 11.d
tantargy = fizika
Nem letezo tantargy. Kerem, adja meg ujra: osztalyfonoki
#
1. feladat
rekordok szama: 1
#
2. feladat
az iskolaban a heti osszoraszam: 1
#
3. feladat
A tanar heti oraszama: 1
#
4. feladat
osztalyfonokok:
11.d - Fizy Kata Nora
#
5. feladat
Osztalyszinten tanuljak
#
6. feladat
tanarok szama: 1
Press any key to continue . . .
```

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 adatbázis-kezelő programmal előállított, egyszerű szerkezetű szöveges állományból kapjuk az alábbi minta szerint. (Minden bejegyzést négy sor tárol.)

Albatrosz Aladin
biologia
9.a
2

Az 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 felhasználható, hogy a fájl maximum 1000 bejegyzést (azaz legfeljebb 4000 sort) tartalmaz; az iskolában legfeljebb 100 tanár és legfeljebb 50 osztály van, továbbá minden osztálynak pontosan egy osztályfőnöke van. Az érettségi feladatban adottaktól eltérően, a heti óraszám maximum 40 óra lehet.

Készítsen programot, amely

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

Specifikáció

Definíció: $\mathbb{Edurec}$ = Teacher × Subject × Group × HoursNum;
Teacher, Subject, Group = $\mathbb{S}$, HoursNum = $\mathbb{N}$ [tantárgyfelosztás bejegyzés]

N ∈ $\mathbb{N}$ [bejegyzések száma]

line ∈ $\mathbb{N}$ line = 4 * N; [bejegyzések adatsorainak száma]

Bemenet: line, course ∈ $\mathbb{Edurec}^N$; name, subject, group ∈ $\mathbb{S}$

Előfeltétel: line ∈ [4, 4000], line/4 ∈ $\mathbb{N}$;
name: ∃ i (1≤ i≤ (line/4)): name = course.Teacher;
(subject, group): ∃ j (1≤ j≤ (line/4)): $course_j$.Subject = subject ∧ $course_j$.Group=group
∀i (1≤ i≤ N): 0 ≤ $course_j$.HoursNum ≤ 40

Kimenet_1: db ∈ $\mathbb{N}$

Kimenet_2: hours_sum ∈ $\mathbb{N}$

Kimenet_3: teacher_hours_sum ∈ $\mathbb{N}$

Kimenet_4: groups ∈ $\mathbb{S}^*$

Kimenet_5: grouping ∈ $\mathbb{S}$

Kimenet_6: teacher_count ∈ $\mathbb{N}$

Utófeltétel_1: N := db (db ≤ N): db := $\sum_{\substack{i=1..N \\ input(i)=true}} (1)$

Definíció_1: input: $\mathbb{N} \to \mathbb{L}$, input(i) := ∃ a, b, c ∈ ($\mathbb{S}$\{””} ∧ e ∈ $\mathbb{N}$, e(0≤ e ≤ 40): {a, b, c, e} ∈ $\mathbb{Edurec}$;

Utófeltétel_2: hours_sum := $\sum_{i=1..N} (course_i.HourseNum)$

Utófeltétel_3: teachers_hours_sum := $\sum_{\substack{i=1..N \\ course_i.Teacher=name}} (course_i.HourseNum)$

Utófeltétel_4: groups := $\text{Kigyujtes}_{\substack{i=1..N \\ course_i.Subject="osztalyfonoki"}} (course_i.Group\& " - "\&course_i.Teacher)$

Utófeltétel_5: grouping := grouptype_{gt}

Definíció_5: grouptype := {„Nem tanulják”, „Osztályszinten tanulják”, „Csoportbontásban tanulják”}
gt:=MIN(2; $\sum_{\substack{i=1..N \\ course_i.Subject=subject \wedge course_i.Group=group}} (1)$)

Megjegyzés: a „Nem tanulják” opció az eredeti feladatnak nem része.

Utófeltétel_6: teacher_count:= $\sum_{\substack{i=1..N \\ \forall j(1\leq j<i):\ course_j.Teacher \neq course_i.Teacher}} (1)$

Fejlesztői környezet

IBM PC, exe futtatására alkalmas operációs rendszer (pl. Windows 10). mingw32-g++.exe c++ fordítóprogram (v 5.1.0), Code::Blocks (v17.12) fejlesztőkörnyezet.

Forráskód

A teljes fejlesztői anyag – kicsomagolás után – a `tanfel` nevű könyvtárban található meg. A fejlesztés során használt könyvtár-struktúra:

Állomány	Magyarázat
tanfel\main.cpp	C++ forráskód (biro miatt 1 fájl)
tanfel\test\be1.txt	teszt-bemeneti fájl1
tanfel\test\be2.txt	teszt-bemeneti fájl2
tanfel\test\be3.txt	teszt-bemeneti fájl3
tanfel\test\be4.txt	teszt-bemeneti fájl4
tanfel\test\be5.txt	teszt-bemeneti fájl5
tanfel\test\be6.txt	teszt-bemeneti fájl6
tanfel\test\be7.txt	teszt-bemeneti fájl7
tanfel\test\be8.txt	teszt-bemeneti fájl8
tanfel\test\ki1.txt	teszt kimenet1
tanfel\test\ki2.txt	teszt kimenet2
tanfel\test\ki3.txt	teszt kimenet3
tanfel\test\ki4.txt	teszt kimenet4
tanfel\test\ki5.txt	teszt kimenet5
tanfel\test\ki6.png	képernyő kimenet
tanfel\test\ki7.png	képernyő kimenet
tanfel\test\ki8.png	képernyő kimenet
tanfel\test\of3.txt	3. teszt 4. részfeladat eredménye
tanfel\test\of5.txt	5. teszt 4. részfeladat eredménye
tanfel\test\of7.txt	7. teszt 4. részfeladat eredménye
tanfel\test\of8.txt	8. teszt 4. részfeladat eredménye
tanfel\test\beosztas.txt	eredeti teszt bemenet
tanfel\doc\feladat.txt	az érettségi feladat eredeti szövege
tanfel\doc\beosztas.txt	az érettségi feladathoz adott eredeti bemeneti fájl
tanfel\doc\of.txt	az érettségi feladat fájlba mentett adatai
tanfel\doc\Mintak\minta_eleje.png	minta futtatás teljes képernyőkép eleje
tanfel\doc\Mintak\minta_vege.png	minta futtatás - teljes képernyőkép véte
tanfel\doc\Mintak\minta_hibak_eleje.png	minta futtatás hibákkal - teljes képernyőkép eleje
tanfel\doc\Mintak\minta_hibak_vege.png	minta futtatás hibákkal - teljes képernyőkép vége
tanfel\doc\Mintak\of_minta.txt	minta futtatás hibákkal - fájlba mentett adatok
tanfel\doc\ELTE_IK_prog_tanfel.pdf	dokumentáció (ez a fájl)

A biro.inf.elte.hu-n tesztelt forráskód neve: `Még nem tesztelhető`

Megoldás

Programparaméterek

Konstans

MHours : **Egész**(40) [egy bejegyzésben a heti óraszám maximális értéke]

MaxRec : **Egész**(1000) [a bejegyzések maximális száma]

grouptype : **Tömb**(0..2:**Szöveg**)
(„Nem tanulják”, „Osztályszinten tanulják”, „Csoportbontásban tanulják”)

outfile : **Szöveg**(of.txt)

Típus

EduRec = **Rekord**(Teacher, Subject, Group : **Szöveg**, HoursNum : **Egész**)

Változó

N, line : **Egész**

course : **Tömb**(1..N: **EduRec**)

hours_sum, teacher_hours_sum, gt, teacher_count : **Egész**

groups : **Tömb**(*: **Szöveg**)

Programfelépítés

A program által használt modulok (és helyük):

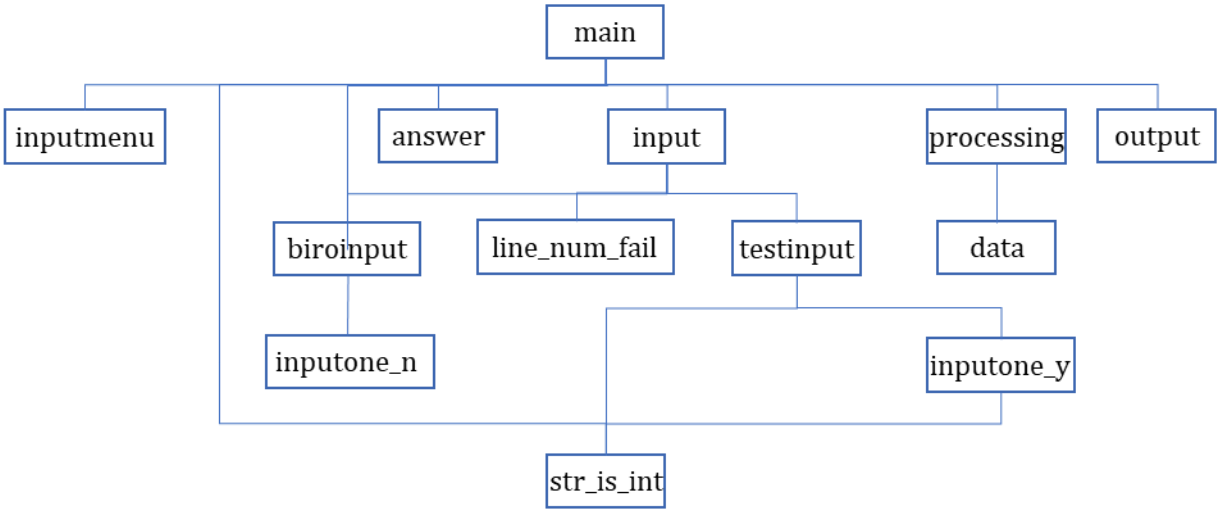
`main.cpp` – program, a forráskönyvtárban

`iostream` – képernyő-, és billentyűkezelés, a C++ része

`fsstream` – fájl output kezelése, a C++ része

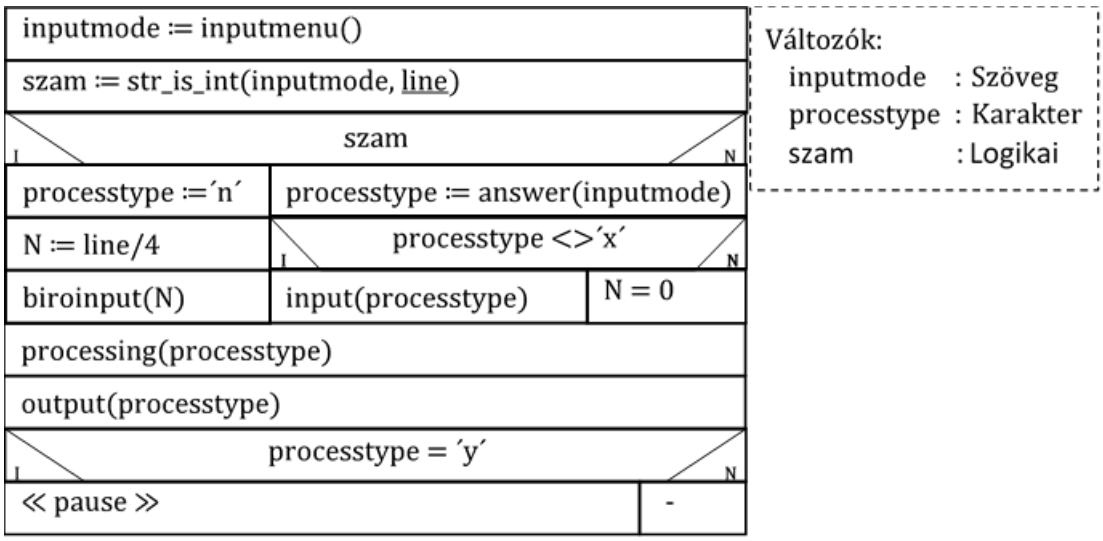
`vector` – dinamikusan változó méretű tömb, a C++ része

Függvénystruktúra



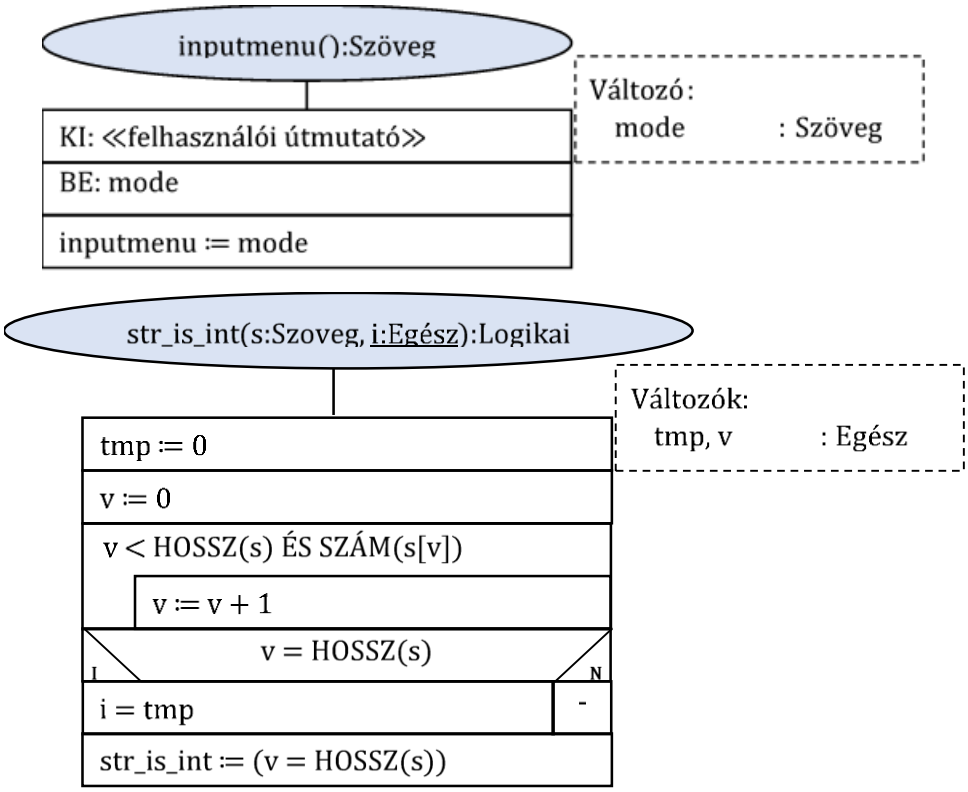
A teljes program algoritmusa

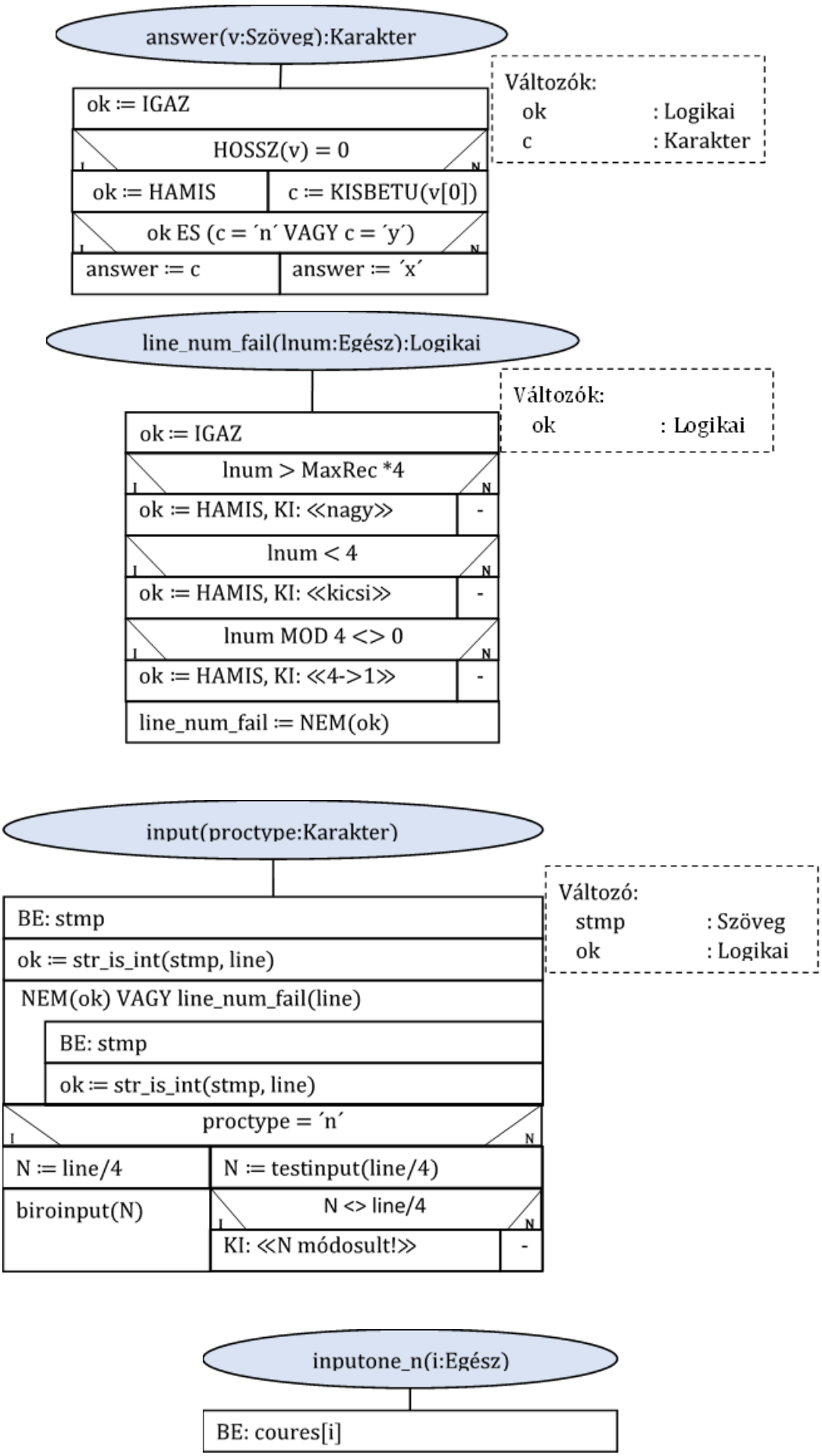
Főprogram

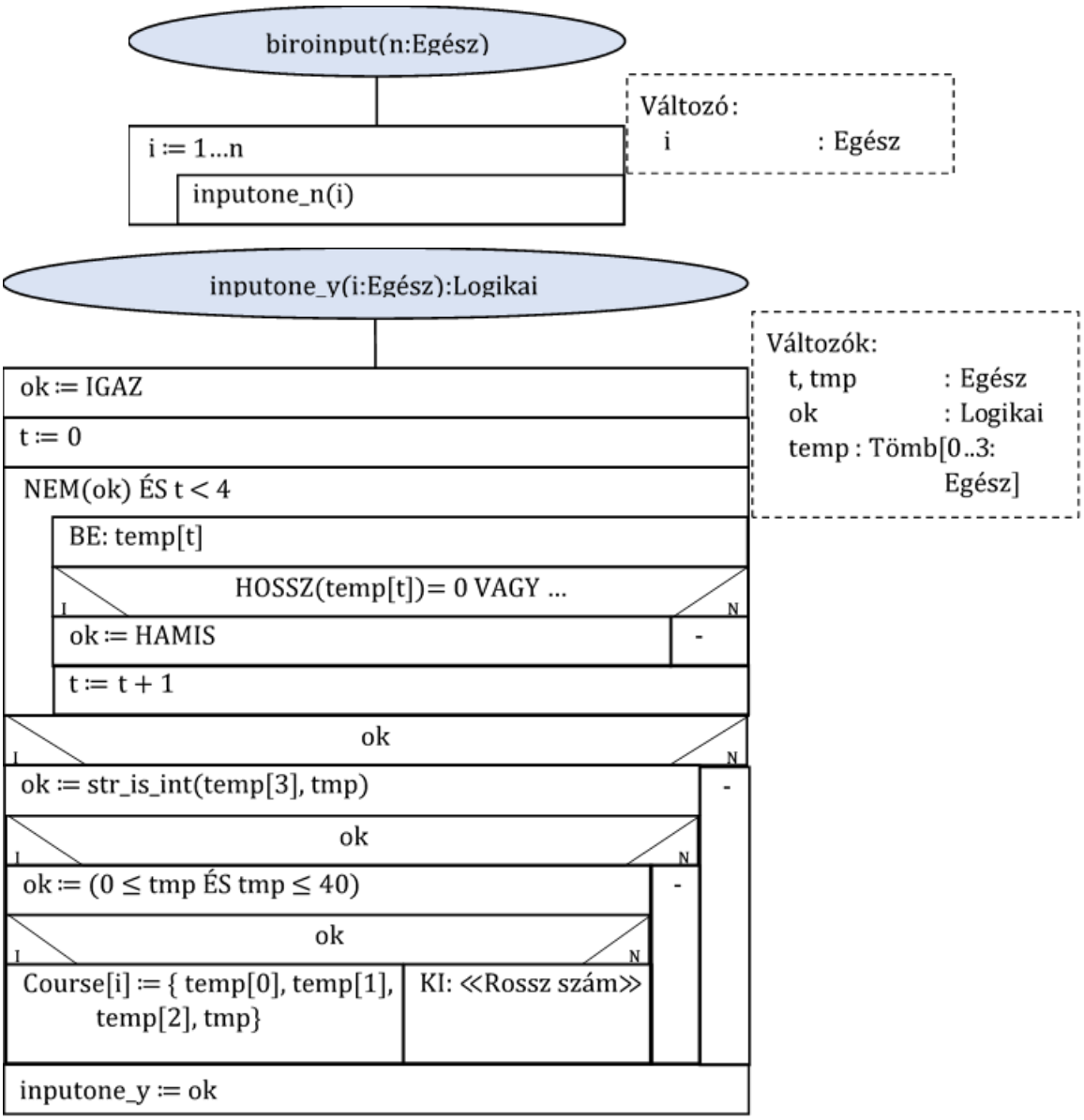


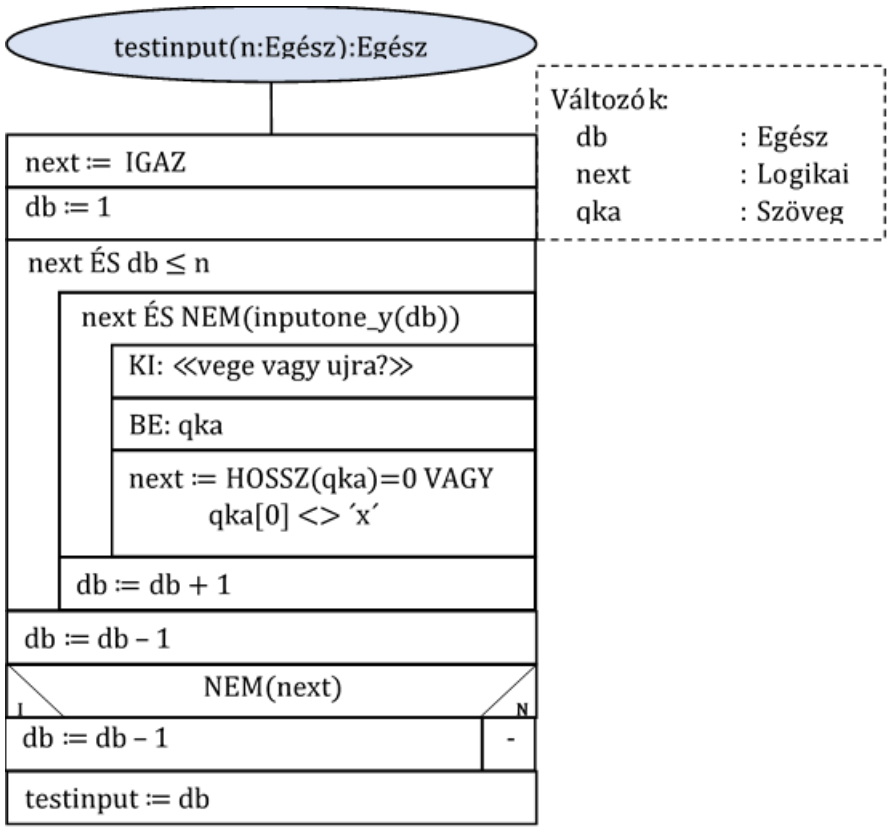
Megjegyzés: a „szam” hasznalata helyett fordítási direktríva használható.

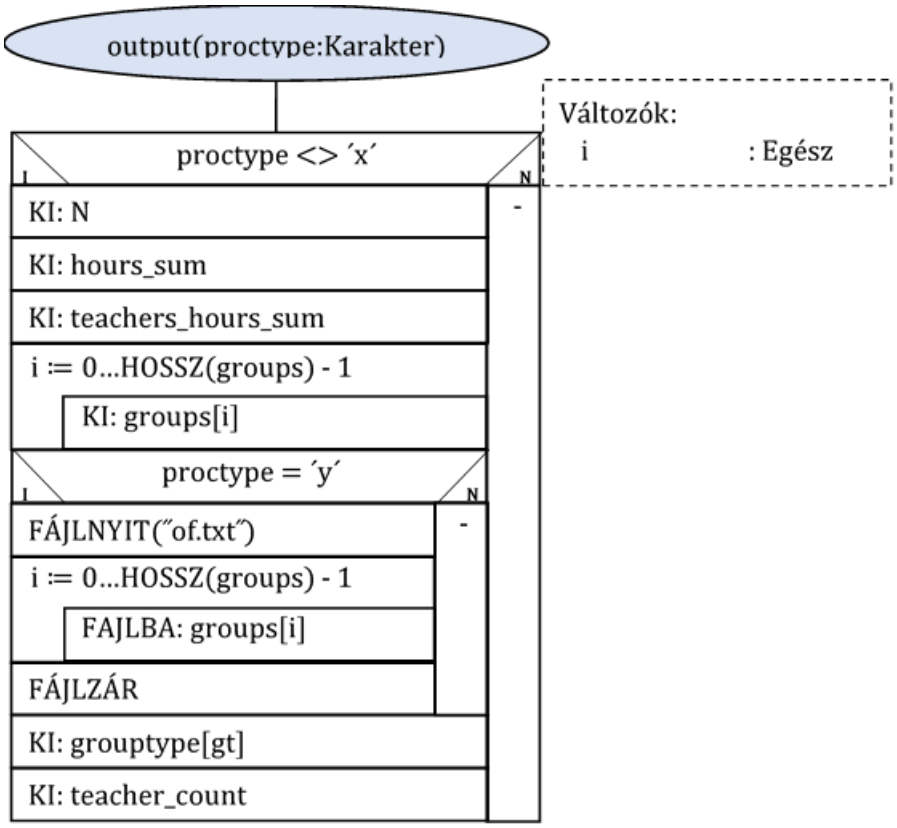
Alprogramok

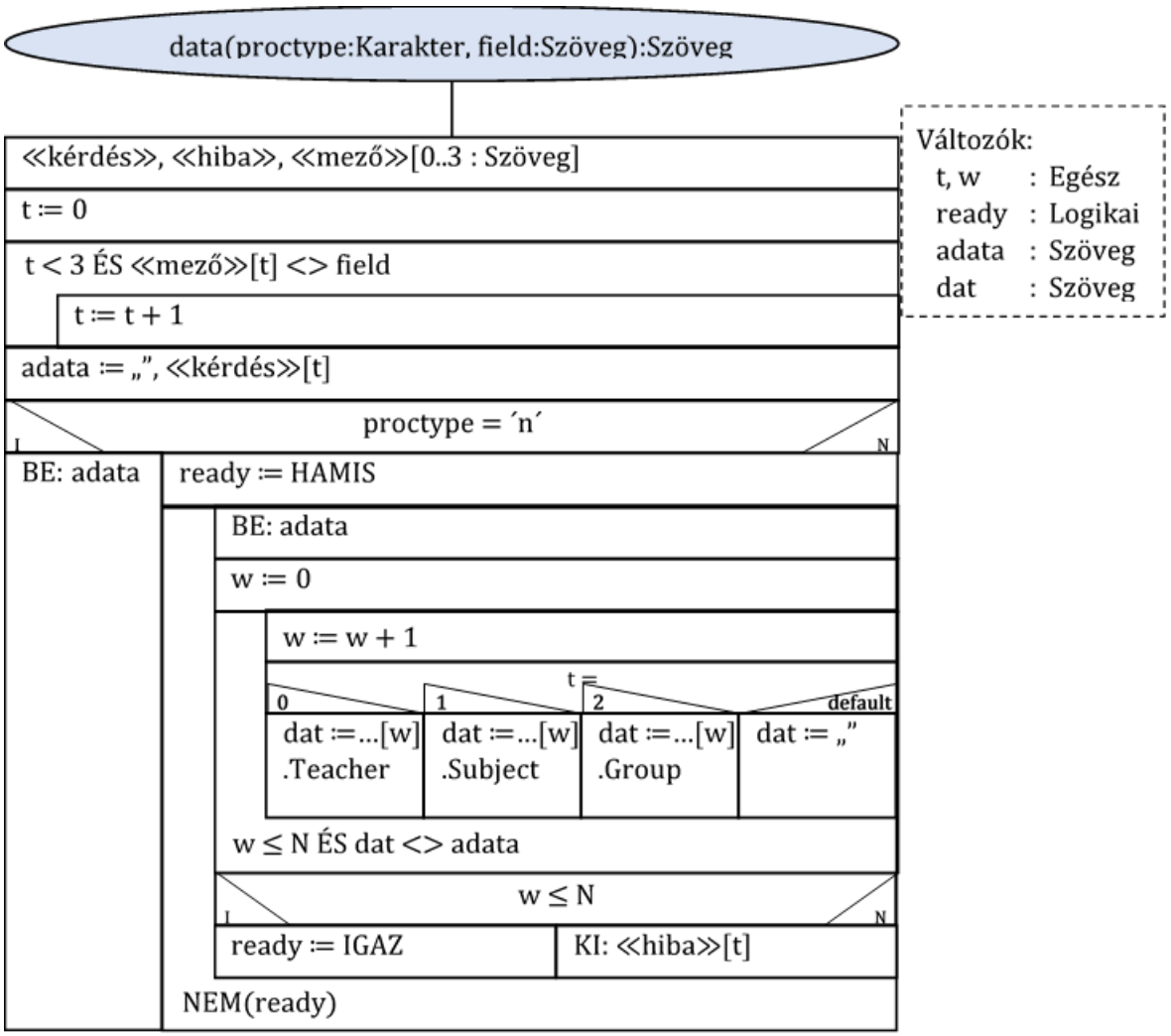


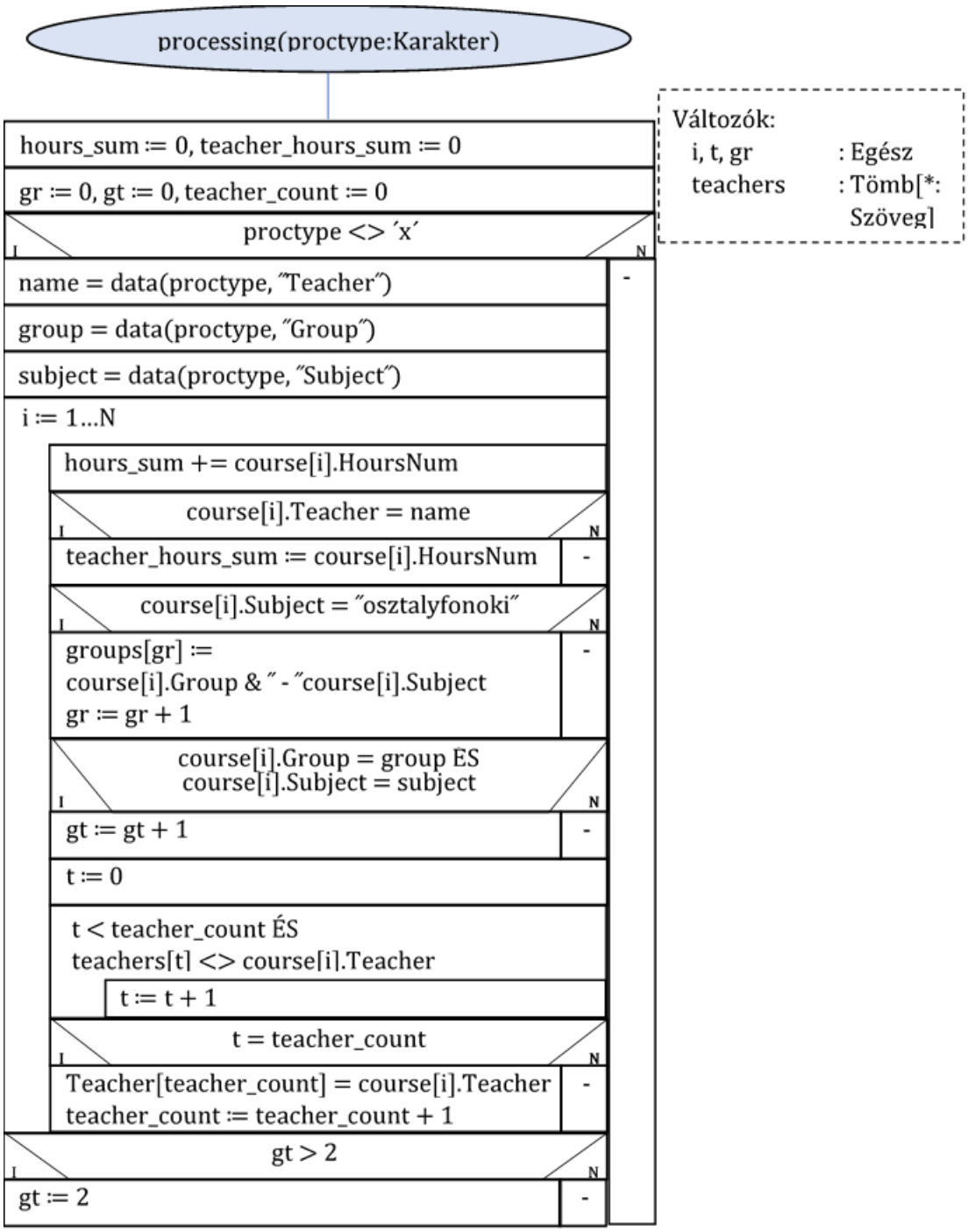

answer(v:Szöveg):Karakter
Változók:
ok : Logikai
c : Karakter
ok := IGAZ
HOSSZ(v) = 0
I
N
ok := HAMIS
c := KISBETU(v[0])
ok ES (c = ´n´ VAGY c = ´y´)
I
N
answer := c
answer := ´x´
line_num_fail(lnum:Egész):Logikai
Változók:
ok : Logikai
ok := IGAZ
lnum > MaxRec *4
I
N
ok := HAMIS, KI: ≪nagy≫
-
lnum < 4
I
N
ok := HAMIS, KI: ≪kicsi≫
-
lnum MOD 4 <> 0
I
N
ok := HAMIS, KI: ≪4->1≫
-
line_num_fail := NEM(ok)
input(proctype:Karakter)
Változó:
stmp : Szöveg
ok : Logikai
BE: stmp
ok := str_is_int(stmp, line)
NEM(ok) VAGY line_num_fail(line)
BE: stmp
ok := str_is_int(stmp, line)
proctype = ´n´
I
N
N := line/4
N := testinput(line/4)
biroinput(N)
N <> line/4
I
N
KI: ≪N módosult!≫
-
inputone_n(i:Egész)
BE: coures[i]


biroinput(n:Egész)
Változó:
i : Egész
i := 1...n
inputone_n(i)
inputone_y(i:Egész):Logikai
Változók:
t, tmp : Egész
ok : Logikai
temp : Tömb[0..3: Egész]
ok := IGAZ
t := 0
NEM(ok) ÉS t < 4
BE: temp[t]
HOSSZ(temp[t])= 0 VAGY ...
I
N
ok := HAMIS
-
t := t + 1
ok
I
N
ok := str_is_int(temp[3], tmp)
-
ok
I
N
ok := (0 ≤ tmp ÉS tmp ≤ 40)
-
ok
I
N
Course[i] := { temp[0], temp[1], temp[2], tmp}
KI: ≪Rossz szám≫
inputone_y := ok


testinput(n:Egész):Egész
Változók:
db : Egész
next : Logikai
qka : Szöveg
next := IGAZ
db := 1
next ÉS db ≤ n
next ÉS NEM(inputone_y(db))
KI: «vege vagy ujra?»
BE: qka
next := HOSSZ(qka)=0 VAGY
qka[0] <> ´x´
db := db + 1
db := db – 1
NEM(next)
I
N
db := db – 1
-
testinput := db


output(proctype:Karakter)
Változók:
i : Egész
proctype <> ´x´
I
N
KI: N
KI: hours_sum
KI: teachers_hours_sum
i := 0...HOSSZ(groups) - 1
KI: groups[i]
proctype = ´y´
I
N
FÁJLNYIT(˝of.txt˝)
i := 0...HOSSZ(groups) - 1
FAJLBA: groups[i]
FÁJLZÁR
-
KI: grouptype[gt]
KI: teacher_count
-


data(proctype:Karakter, field:Szöveg):Szöveg
≪kérdés≫, ≪hiba≫, ≪mező≫[0..3 : Szöveg]
t := 0
t < 3 ÉS ≪mező≫[t] <> field
t := t + 1
adata := „”, ≪kérdés≫[t]
proctype = ´n´
I
N
BE: adata
ready := HAMIS
BE: adata
w := 0
w := w + 1
t =
0
1
2
default
dat :=...[w] .Teacher
dat :=...[w] .Subject
dat :=...[w] .Group
dat := „”
w ≤ N ÉS dat <> adata
w ≤ N
I
N
ready := IGAZ
KI: ≪hiba≫[t]
NEM(ready)
Változók:
t, w : Egész
ready : Logikai
adata : Szöveg
dat : Szöveg


processing(proctype:Karakter)
Változók:
i, t, gr : Egész
teachers : Tömb[*: Szöveg]
hours_sum := 0, teacher_hours_sum := 0
gr := 0, gt := 0, teacher_count := 0
proctype <> 'x'
I
N
name = data(proctype, "Teacher")
group = data(proctype, "Group")
subject = data(proctype, "Subject")
i := 1...N
hours_sum += course[i].HoursNum
course[i].Teacher = name
teacher_hours_sum := course[i].HoursNum
course[i].Subject = "osztalyfonoki"
groups[gr] :=
course[i].Group & " - "course[i].Subject
gr := gr + 1
course[i].Group = group ÉS
course[i].Subject = subject
gt := gt + 1
t := 0
t < teacher_count ÉS
teachers[t] <> course[i].Teacher
t := t + 1
t = teacher_count
Teacher[teacher_count] = course[i].Teacher
teacher_count := teacher_count + 1
gt > 2
gt := 2
-

A kód

A `main.cpp` fájl tartalma:

```
/*
Készítette: Szalayné Tahy Zsuzsanna
Neptun kód: N0CD19
E-mail: sztzs@inf.elte.hu
Feladat: Tantárgyfelosztás (érettségi 2019-05i emelt 4.)
*/
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
#define biro

struct EduRec {
   string Teacher;
   string Subject;
   string Group;
   unsigned HoursNum;
};

//input
const unsigned MHours = 40;
const unsigned MaxRec = 1000;
EduRec course[MaxRec + 1];
string name;    //Q3: name of a teacher
string subject; //Q5:
string group;   //Q5

//output
unsigned line;              //number of data lines (N*4)
unsigned N;                 //Q1: number of EduRec records
int hours_sum;              //Q2: weekly sum of hours
int teacher_hours_sum;      //Q3: weekly sum of hours of the named teacher
string outfile = "of.txt";  //Q4: file for the form-master's list
vector<string> groups;      //Q4: list of form-master
const string grouptype[3]   //Q5: expected answers
    = {"Nem tanuljak", "Osztalyszinten tanuljak","Csoportbontasban tanuljak"};
int gt;                     //Q5: index of grouptype
int teacher_count;          //Q6: number of teachers

/*                      Input functions                     */
//menu for the input mode
string inputmenu();

//if all characters are numbers then convert to int
bool str_is_int(string, unsigned&);

//check the answer of the input mode
char answer(string v);

//check the number of line (number of records = lines / 4)
bool line_num_fail(unsigned);

//read 1 record without testing
void inputone_n(int);

//read and test 1 record
bool inputone_y(int);

//input a list of records - from console, given number of items, nontested
//basic "biro" input style
void biroinput(unsigned);

//input a list of records - from console, given number of items, tested
//"biro + test" returns the number of success reading
unsigned testinput(unsigned);

//choosed input process
void input(char);
```

```
/*                      Processor functions                      */
//checked parameter input
string data(char, string);
void processing(char);

/*                        Output functions                        */
void output(char);

/*                              MAIN                              */

int main()
{
   string inputmode = inputmenu();
   char processtype; /*in {y, n, x}*/
   /*bool szam = */
   str_is_int(inputmode, line);
#ifdef biro     //   if(szam) {
   processtype = 'n';
   N = line / 4;
   biroinput(N);
#else           //   } else {
   processtype = answer(inputmode);
   if(processtype != 'x')
      input(processtype);
   else
      N = 0;
#endif // biro  //   }
   processing(processtype);

   output(processtype);

   if(processtype == 'y')
      system("pause");

   return 0;
}

/*                      output function's definition                      */

void output(char proctype)
{
   if(proctype != 'x') {
      cout << "#"<< endl;
      cerr << "1. feladat" << endl;
      cout << N << endl;
      cout << "#"<< endl;
      cerr << "2. feladat" << endl;
      cerr << "az iskolaban a heti osszoraszam: ";
      cout << hours_sum << endl;
      cout << "#" << endl;
      cerr << "3. feladat" << endl;
      cerr << "A tanar heti oraszama: ";
      cout << teacher_hours_sum << endl;
      cout << "#" << endl;
      cerr << "4. feladat" << endl;
      for(unsigned i = 0; i < groups.size(); i++) {
         cout << groups[i] << endl;
      }
      if(proctype == 'y') {
         ofstream fout(outfile.c_str());
         for(unsigned i = 0; i < groups.size(); i++)
            fout << groups[i] << endl;
         fout.close();
      }
      cout << "#" << endl;
      cerr << "5. feladat"<< endl;
      cout << grouptype[gt] << endl;
      cout << "#" << endl;
      cerr << "6. feladat" << endl;
      cout << teacher_count << endl;
   }
}
```

```
/*                      processor function's definition                      */

//checked parameter input
string data(char proctype, string field)
{
   string question[4] = {"egy tanar neve = ", "tantargy = ", "osztaly = ", "!!! programhiba !!!"};
   string err[4] = { "Ismeretlen tanar. Kerem, adja meg ujra: ",
                     "Nem letezo tantargy. Kerem, adja meg ujra: ",
                     "Nincs ilyen nevu osztaly. Kerem, adja meg ujra: ",
                     " "};
   string prop[4] = {"Teacher", "Subject", "Group", " "};
   int t;
   for(t = 0; t < 3 && prop[t] != field; t++)
      ;
   string adata = "";
   cerr << question[t];
   if(proctype == 'n') {
      getline(cin, adata);
   } else { /* if(proctype == 'y') */
      bool ready = false;
      unsigned w;
      string dat;
      do {
         getline(cin, adata);
         w = 0;
         do {
            w++;
            switch (t) {
               case 0:
                  dat = course[w].Teacher;
                  break;
               case 1:
                  dat = course[w].Subject;
                  break;
               case 2:
                  dat = course[w].Group;
                  break;
               default:
                  dat = "";
                  break;
            }
         } while (w <= N && dat != adata);
         if(w <= N)
            ready = true;
         else
            cerr << err[t];
      } while(!ready);
   }
   return adata;
}

void processing(char proctype)
{
   hours_sum = 0;                                          /*Q2*/
   teacher_hours_sum = 0;                                  /*Q3*/
   gt = 0;                                                 /*Q5*/
   teacher_count = 0;                                      /*Q6*/

   if(proctype != 'x') {
      cerr << "Tanari oraszamhoz ";
      name = data(proctype, "Teacher");

      cerr << "Csoportbontas jellemzesehez" << endl;
      group = data(proctype, "Group");
      subject = data(proctype, "Subject");

      vector<string> teachers;                            /*Q6*/

      for(unsigned i = 1; i <= N; i++) {
         hours_sum += course[i].HoursNum;                 /*Q2*/

         if(course[i].Teacher == name)                    /*Q3*/
```

```
            teacher_hours_sum += course[i].HoursNum;

         if(course[i].Subject == "osztalyfonoki")        /*Q4*/
            groups.push_back(course[i].Group + " - " + course[i].Teacher);

         if(course[i].Group == group && course[i].Subject == subject) /*Q5*/
            gt++;

         unsigned t = 0;                                 /*Q6*/
         while(t < teachers.size() && teachers[t] != course[i].Teacher)
            t++;
         if(t == teachers.size())
            teachers.push_back(course[i].Teacher);
      }

      if(gt > 2)
         gt = 2;
      teacher_count = teachers.size();
   }
}

/*                      input function's definitions                      */

//menu for the input mode
string inputmenu()
{
   cerr << "A tantargyfelosztas elemzese" << endl;
   cerr << "Valasszon az alabbi beolvasasi lehetosegek kozul:" << endl << endl;
   cerr << "\t\tEgy SZAM beirasaval (vegen enter) a 'biro' stilusu konzolos kiertekeles indul." <<
endl << endl;
   cerr << "\t\tn|N\t konzolrol adott szamu rekord beolvasasa ellenorzes nelkul." << endl;
   cerr << "\t\ty|Y\t konzolrol adott szamu rekord beolvasasa adatonkent ellenorizve." << endl;

   cerr << "A fentiektol eltero kezdetu valaszokra a program futasa vegeter." << endl << endl;

   cerr << "valasztas: ";
   string mode;
   getline(cin, mode);
   cerr <<
"*********************************************************************************************"
<<endl<<endl;
   return mode;
}

//if all characters are numbers then convert to int
bool str_is_int(string s, unsigned& i)
{
   unsigned tmp=0;
   unsigned v;
   for(v = 0; v < s.size() && isdigit(s[v]); v++) {
      tmp = tmp * 10 + s[v]-'0';
   }
   if(v == s.size())
      i = tmp;
   return (v == s.size());
}

//check the answer of the input mode
char answer(string v)
{
   char c;
   bool ok = true;
   if(v.size() == 0)
      ok = false;
   else
      c = tolower(v[0]);
   return (ok && (c == 'n' || c == 'y'))? c : 'x';
}

//check the number of line: number of record <- line / 4
bool line_num_fail(unsigned lnum)
{
   bool ok = true;
```

```
   if(lnum > MaxRec * 4) {
      ok = false;
      cerr << "Tul sok a beolvasando adat. (" << lnum << " > " << MaxRec << ")" << endl;
   }
   if(lnum < 4) {
      ok = false;
      cerr << "Tul keves a beolvasando adat. (" << lnum << " < 4)" << endl;
   }
   if(lnum %4 != 0) {
      ok = false;
      cerr << "Az adatok nem adnak ki teljes rekordokat. (" << lnum << "-nak nem osztoja a 4.)"<<
endl;
   }
   return !ok;
}

//read 1 record without testing
void inputone_n(int i)
{
   string qka;
   getline(cin, course[i].Teacher);
   getline(cin, course[i].Subject);
   getline(cin, course[i].Group);
   cin >> course[i].HoursNum;
   getline(cin, qka); //read to endline
}

//unchecked database input
void biroinput(unsigned n)
{
   for(unsigned i = 1; i <= n; i++)
      inputone_n(i);
   cerr << "A megaodtt " << n << " bejegyzes teszteles nelkul beolvasva" << endl;
}

//read and test 1 record
bool inputone_y(int i)
{
   bool ok = true;
   string text[4] = {"Tanar nev = ", "Tantargy nev = ", "Osztaly nev= ", "Oraszam = "};
   string temp[4];
   unsigned tmp;
   for(unsigned t = 0; ok && t < 4; t++) {
      cerr << text[t];
      getline(cin, temp[t]);
      if(cin.fail() || temp[t].size() == 0) {
         ok = false;
         cin.clear();
      }
   }
   if(ok) {
      ok = str_is_int(temp[3], tmp);
      if (ok) {
         //between distance learning and weekly working time
         ok = (0 <= tmp && tmp <= MHours);
         if(ok)
            course[i] = {temp[0], temp[1], temp[2], tmp};
         else
            cerr << "Irrealis oraszam" << endl;
      }
   }
   return ok;
}

unsigned testinput(unsigned n)
{
   bool next = true;
   unsigned db;
   for (db = 1; next && db<=n; db++)
      while (next && !inputone_y(db)) {
         cerr << "Hibas rekord." << endl;
         cerr << "\t\tKilepes a bevitelbol: x" << endl;
         cerr << "\t\tRekord ismetlese:     ENTER" << endl;
```

```
            string qka;
            getline(cin, qka);
            next = (qka.size() == 0 || qka[0] != 'x');
        }
    db--; //for(...db++)
    if(!next)
        db--; //drop the last, not finalized record
    cerr << n << " db vart bejegyzesbol sikeresen beolvasott rekordok szama: ";
    cerr << db << endl;
    return db;
}

//choosed special input process
void input(char proctype)
{
    string stmp;
    cerr << "Adja meg a beolvasando adatok (4-gyel oszthato) szamat: ";
    getline(cin, stmp);
    bool ok = str_is_int(stmp, line);
    while(!ok || line_num_fail(line)) {
        cerr << "Kerem az adatsorok szamat [4; " << MaxRec * 4 << "]: ";
        getline(cin, stmp);
        ok = str_is_int(stmp, line);
    }
    if (proctype == 'n') {
        N = line / 4;
        biroinput(N);
    } else { /*if(proctype == 'y')*/
        N = testinput(line/4);
        if(N != line/4)
            cerr<< "FIGYELMEZTETES: A megadottnal kevesebb rekordot sikerult beolvasni." << endl;
    }
}
```

Tesztelés

Érvényes tesztesetek

teszteset: < be1.txt > ki1.txt

Bemenet – *egy rekord, ami osztalyfonoki*
4 Albatrosz Aladin osztalyfonoki 9.a 1 Albatrosz Aladin 9.a osztalyfonoki
Kimenet
1 # 1 # 1 # 9.a - Albatrosz Aladin # Osztalyszinten tanuljak # 1

teszteset: < be2.txt > ki2.txt

Bemenet – *2 rekord, csoportbontás*
8 Csincsilla Csilla informatika 9.a 2 Teve Teofil informatika 9.a 2 Csincsilla Csilla 9.a informatika
Kimenet
2 # 4 # 2 # # Csoportbontasban tanuljak # 2

teszteset: < be3.txt > ki3.txt, fájlba: of3.txt

Bemenet – *adatellenőrzéssel 1 rekord, osztalyfonoki*	
y 4 Albatrosz Aladin osztalyfonoki 9.a 1 Albatrosz Aladin 9.a osztalyfonoki	
Kimenet	
konzol	**fájl**
# 1 # 1 # 1 # 9.a - Albatrosz Aladin # Osztalyszinten tanuljak # 1 Press any key to continue . . .	9.a - Albatrosz Aladin

teszteset: < be4.txt > ki4.txt

Bemenet – *adatellenőrzés nélkül a kapott adatokkal*
n 1316 ≪bemenet.txt adatai≫ Teve Teofil 11.a informatika
Kimenet
329 # 1016 # 18 # 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 # Nem tanuljak # 49

teszteset: < be5.txt > ki5.txt, fájl: of5.txt

Bemenet – *ellenőrzött bevitel, kapott adatokkal*	
y 1316 ≪bemenet.txt adatai≫ Farkas Farkas 9.b kemia	
Kimenet	
konzol	**fájl**
# 329 # 1016 # 18	9.a - Albatrosz Aladin 9.b - Hangya Hanna 9.c - Zerge Zenina 9.d - Medve Melani 10.a - Farkas Farkas 10.b - Bivaly Biti

# 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 # Csoportbontasban tanuljak # 49 Press any key to continue . . .	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

Érvénytelen tesztesetek

teszteset: be6.txt, ki6.png

Bemenet – *Hibás futtatási mód megadása*
alma
Kimenet
A program futása befejeződik

```
C:\Users\sztzs\Desktop\tanfel\bin\Release>tanfel
A tantargyfelosztas elemzese
Valasszon az alabbi beolvasasi lehetosegek kozul:

                Egy SZAM beirasaval (vegen enter) a 'biro' stilusu konzolos kiertekeles indul.

                n|N      konzolrol adott szamu rekord beolvasasa ellenorzes nelkul.
                y|Y      konzolrol adott szamu rekord beolvasasa adatonkent ellenorizve.
A fentiektol eltero kezdetu valaszokra a program futasa vegeter.

valasztas: alma
********************************************************************************************************


C:\Users\sztzs\Desktop\tanfel\bin\Release>
```

teszteset: be7.txt, ki7.png

Bemenet – *Adatsorok száma hibás*
y **alma** **1** 4 SzTZs inf 9.c 2

SzTZs
9.c
inf

Kimenet

Tájékoztatás a hibáról, az adat újra bekérése

...

```
valasztas: y
*****************************************************************************************************

Adja meg a beolvasando adatok (4-gyel oszthato) szamat: alma
Kerem az adatsorok szamat [4; 4000]: 1
Tul keves a beolvasando adat. (1 < 4)
Az adatok nem adnak ki teljes rekordokat. (1-nak nem osztoja a 4.)
Kerem az adatsorok szamat [4; 4000]: 4
Tanar nev = SzTZs
```

...

teszteset: be8.txt, ki8.png

Bemenet – *A kiválasztandó adat nem létezik (név, osztály, tantárgy)s*

y
4
SzTZs
inf
9.c
2
SzTZs
inf
9.c
inf

Kimenet

Ha nem találja az adatot (pl. osztály helyett tantárgynév lett beírva), akkor újra bekéri.

...

```
1 db vart bejegyzesbol sikeresen beolvasott rekordok szama: 1
Tanari oraszamhoz egy tanar neve = SzTZs
Csoportbontas jellemzesehez
osztaly = inf
Nincs ilyen nevu osztaly. Kerem, adja meg ujra: 9.c
tantargy = inf
#
1. feladat
rekordok szama: 1
```

...

Fejlesztési lehetőségek

- Kiírások jobb tördelése, hibajelzések szofisztikáltabb megfogalmazása
- A bejegyzések –a felhasználó igénye szerint– fájlból beolvashatók, fájlvégéig olvasással (eredeti érettségi feladatnak megfelelően.
- Bevitt adatok keresése, módosítása, törlése.
- A tantárgy és a csoport adatok rendezett listába (halmazba) gyűjtése, a kérdésekben megadott adatok ellenőrzése logaritmikus kereséssel
- Menü vezérelt program: a kérdések egyenként, akár többször történő kiválasztásának lehetősége
- További kérdések beépítése. (pl. az ´a´ tagozat összes informatikaórájának száma heti vetületben).