Yin-yang a programozásban

Szalayné Tahy Zsuzsa

SzIG Tudományos nap 2019

„ProgAlap” és ami mögötte van

Szalayné Tahy Zsuzsanna
ELTE IK
sztzs@caesar.elte.hu

Dr. Czirkos Zoltán
BME VIK
czirkos@eet.bme.hu

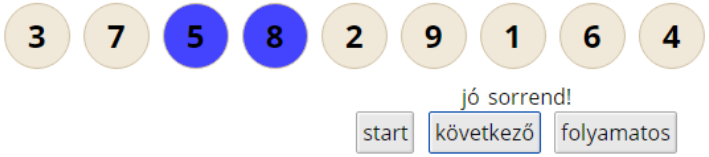
BME-VIK ProgAlap ea.

A ProgAlap oktatási célja

	BME-ProgAlap	ELTE-ProgAlap
Probléma-megoldás	készség szintű eszközhasználat	eszközfüggetlen stratégia
Algoritmus	megismerés	elemzés
Nyelv	C nyelv elsajátítása	C++ elemeinek megismerése
Adatstruk-túrák	(int*) gyakorlati, implementáció	($\mathbb{N}$) elméleti, matematikai definíció
Program-készítés	gyakorlata	folyamata
Tesztelés	debug	tesztadatok készítése
Feladat-megoldás	részekre bontás, adatok definiálása, részek implementálása	specifikáció, algoritmus, esetenként kódolás

InfoC - Buborékrendezés

11. Buborékrendezés (bubble sort)



Lényege: egymás melletti elemek összehasonlítása és cseréje.

Egy sor csere által a legnagyobb elem a végére kerül.

A buborékrendezés egymás melletti elemeket hasonlít össze. Lépései:

- Hasonlítsuk össze az első két elemet. Ha nincsenek jó sorrendben, cseréljük meg.
- Hasonlítsuk össze a második párt (második és harmadik elem). Esetleg csere.
- Folytassuk így a tömb végéig.
- A legnagyobb elem ezáltal a tömb végére kerül, még akkor is, ha legelöl volt. Az már a végleges helye.
- Csináljuk meg ugyanezt még egyszer, a tömb elejétől az utolsó előttiig. Az utolsóhoz már nem kell nyúlni, hiszen az a legnagyobb.
- Aztán ugyanezt megint, de az utolsó kettőhöz már nem nyúlunk stb.

Futás közben így a tömb két részre oszlik: egy már rendezett és egy még rendezetlen részletre. A rendezetlen részlet egyre csökken; azon belül kell összehasonlítani és esetleg cserélni a párokat. Ezért az algoritmus két ciklust tartalmaz. A külső ciklus az egyre kisebb rendezetlen részt határozza meg; a belsejében lévő pedig az egymás melletti párok összehasonlítását vezérli.

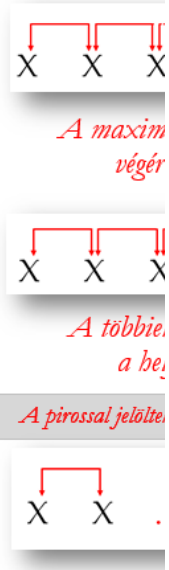
```
void buborek(double *t, int db) {
   int i, j;

   /* egyre rövidebb tömbrészletek ciklusa */
   for (i = db-1; i > 0; --i)
      /* egymás utáni párok ciklusa */
      for (j = 0; j < i; ++j)
         if (t[j+1] < t[j]) {        összehasonlítás
            double temp = t[j];
            t[j] = t[j+1];           csere
            t[j+1] = temp;
         }
}
```

ELTE PA - Buborékrendezés

Buborékos rendezés

A lényeg:

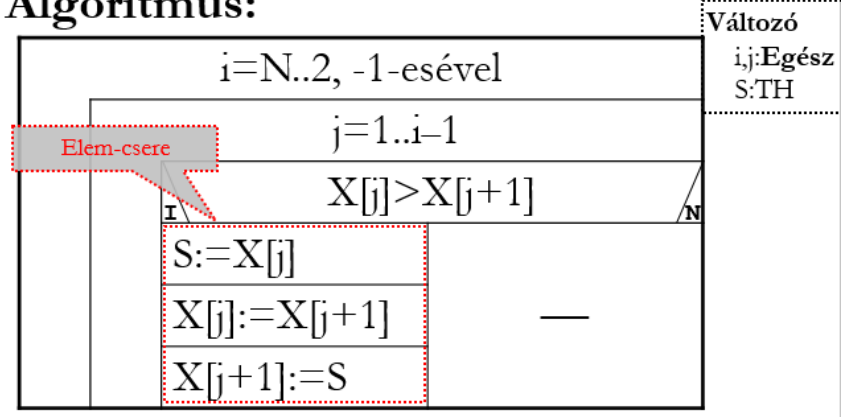
- Hasonlítsunk minden elemet a mögötte levővel, s ha kell, cseréljük meg!
- Ezután ugyanezt csináljuk az utolsó elem nélkül!
- …
- Végül az első két elemre!



Horváth-Papné-Szlávi-Zsakó: Programozási alapismeretek 11. előadás

Buborékos rendezés

Algoritmus:



- Hasonlítások száma: $1+2+..+N-1 = N \cdot \frac{N-1}{2}$
- Mozgatások száma: 0 … $3 \cdot N \cdot \frac{N-1}{2}$

Horváth-Papné-Szlávi-Zsakó: Programozási alapismeretek 11. előadás

11/36

Összegzés

BME:

1. informális specifikáció (feladat szövege)

↓ algoritmizálás

2. algoritmus fejben (informális)

↓ kódolás

3. Programkód (formális)

Kreativitás

ELTE:

1. informális specifikáció (feladat szövege)

↓ specifikáció

2. formális specifikáció (feladat görögül :))

↓ algoritmizálás

3. algoritmus struktogramban (formális)

↓ kódolás

4. Programkód (formális)

Gondolkodásmód

„Ötszáz” feladat
OH és a demonstrátorok

Feladat

Egy turkálóban minden póló darabja 500 Ft. Ha egy vásárlás során valaki több darabot is vesz, a második ára már csak 450 Ft, a harmadik pedig 400 Ft, de a negyedik és további darabok is ennyibe kerülnek, tehát az ár a harmadik vásárlása után már nem csökken tovább.

Írj C függvényt, amely a vásárolt pólók darabszámának ismeretében megmondja, hogy mennyit fizet a vásárló!

```
int fizetendo(int db) {
    int sum = 0;
    for (int i = 1; i <= db; ++i) {
        if (i == 1) sum += 500;
        else if (i == 2) sum += 450;
        else sum += 400;
    }
    return sum;
}
```

```
unsigned hulye_fv(unsigned n) {
   return n*400 + (n>=1 ? 100 : 0) + (n>=2 ? 50 : 0);
}
```

```
/*OH megoldása, hivatalos minta*/

int ertek(int db)
{
    if(db<=2)
     {
          return db * 500 - (db - 1) * 50;
     }
    else
     {
          return 1350 + (db - 3) * 400;
     }
}
```

/* Stílusgyakorlat jelleggel egész jól
újrafelhasználható kódot alkottam. */

```
int ar(int db)
{
    static const int arak[] = {0, 500, 950};
    static const int sokadik = 400;
    static const int limit = sizeof(arak) / sizeof(arak[0]);
    if(db < limit)
        return arak[db];
    return arak[limit - 1] + sokadik * (db - limit + 1);
}
```

```
/*Optimális megoldás*/

int ertek(int db)
{
    if (db == 1) return 500;
    if (db == 2) return 500 + 450;
    return 500 + 450 + 400*(db-2);
}
```

```
int ertek(int db)
{
    if (db == 1)
        return 500;
    if (db == 2)
        return 500 + 450;

    return 500 + 450
           + 400*(db-2);
}
```

```
ertek(int):
        cmp     edi, 1
        mov     eax, 500
        je      .L1
        cmp     edi, 2
        mov     eax, 950
        je      .L1
        sub     edi, 2
        imul    eax, edi, 400
        add     eax, 950
.L1:
        rep ret
```

Compiler Explorer: http://godbolt.org

Keresési stratégiák

A „keresés” kognitív szintjei

- Megkérdezem
- Kulcsszavas, tematikus (online)
- Alkalmazás függvénye, eljárása
- Programozásban metódus
- Algoritmikus
  - Közvetlen elérés
  - Lineáris
    - Explicit
    - Implicit
  - Logaritmikus, struktúra specifikus

Mikor játszották először?
Melyik kosárban van?

Ötlet

Feladat: Megtalálni egy adott tulajdonságú elemet

Alkalmazás, pl. Excel

bool Megfelel()

Függvénykönyvtár

for () { if () { break;}}

Keresés függvény
return-ök

„~~break;~~” => ciklusfeltétel

Elvárt megoldás: „while()”

for() => while()

vadhajtások
hibás, pontatlan

Strukturált programozás

Ötlet

Feladat: Megtalálni egy adott tulajdonságú elemet

Függvénykönyvtár

Alkalmazás, pl. Excel

bool Megfelel()

for () { if () { break;}}

Keresés függvény
return-ök

„~~break;~~” => ciklusfeltétel

Elvárt megoldás: „while()”

for() => while()

vadhajtások
hibás, pontatlan

Strukturált programozás

Majdnem

Feladat: Megtalálni egy adott tulajdonságú elemet

Kiválogatás:

```
for (int i = 0; i < list.Count; i++ )
      if («logikai kif.») list2.Add(elem);
```

Megszámlálás:

```
for (int i = 0; i < list.Count; i++)
      if («logikai kif.») jo++;
```

Utolsó találat:

```
for (int i = 0; i < list.Count; i++)
      if («logical_expr») jo = i;
```

hibás, pontatlan

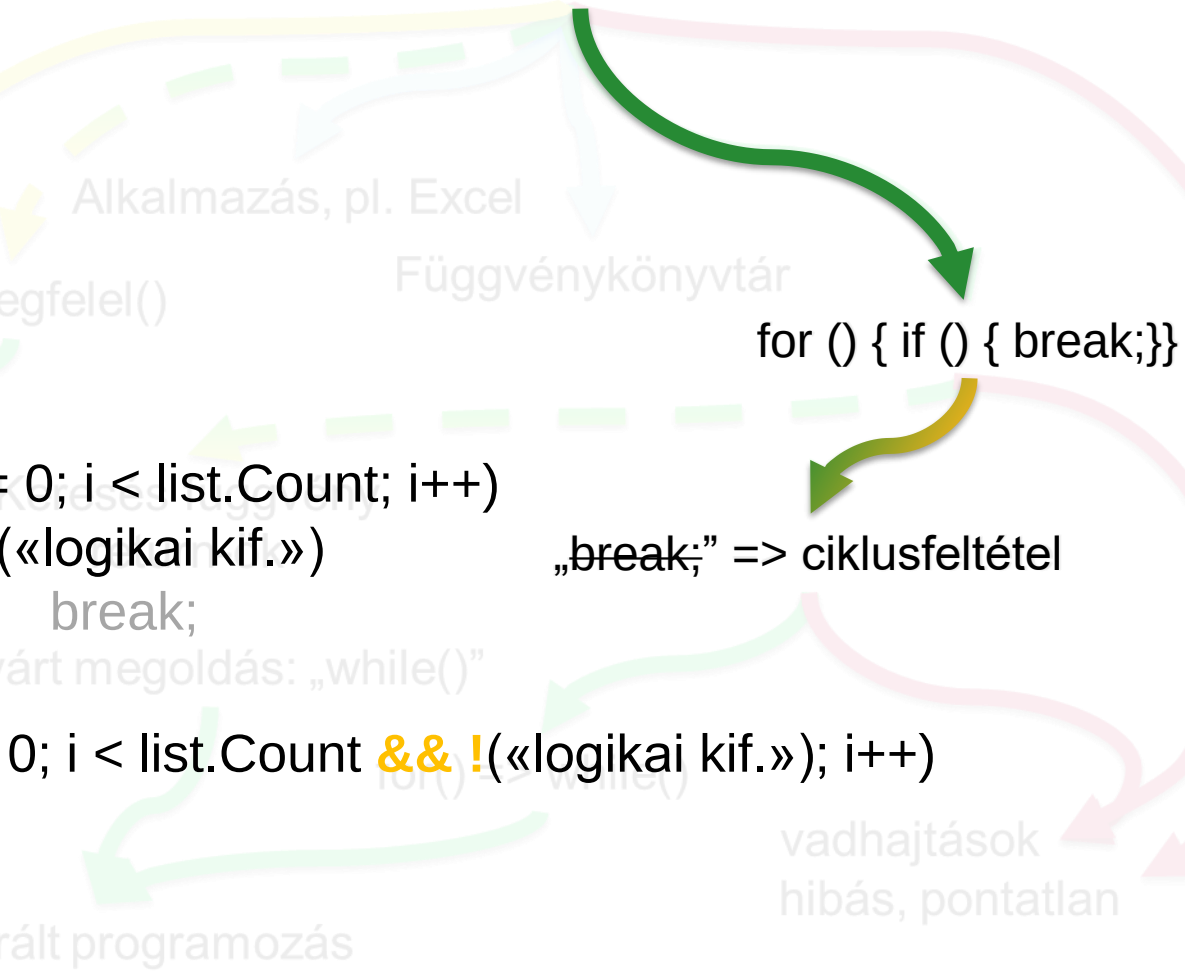
Működik

Feladat: Megtalálni egy adott tulajdonságú elemet

```
int i;
for (i = 0; i < list.Count; i++)
    if («logikai kif.»)
        break;
```

```
for () { if () { break;}}
```

És Nem Jó


Feladat: Megtalálni egy adott tulajdonságú elemet
Alkalmazás, pl. Excel
Függvénykönyvtár
bool Megfelel()
for () { if () { break;}}
int i;
for (i = 0; i < list.Count; i++)
if («logikai kif.»)
break;
„break;” => ciklusfeltétel
Elvárt megoldás: „while()”
int i;
for (i = 0; i < list.Count && !(«logikai kif.»); i++)
;
vadhajtások
hibás, pontatlan
Strukturált programozás

break ➔return

```
int i;
for (i = 0; i < list.Count; i++)
        if («logikai kif.»)
            break;
```

```
for () { if () { break;}}
```

Keresés függvény
return-ök

```
int Holvan(…) {
    for (int i = 0; i < list.Count; i++)
        if («logikai kif.»)
            return i;
    return -1;
}
```

Eredmény meghatározása

```
…
    for (int i = 0; i < list.Count; i++)
        if («logikai kif.») return i;
        else return -1;
…
    for (int i = 0; i < list.Count; i++)
        if («logikai kif.») ez = i;
    return ez;
```

for () { if () { break;}}

Eredmény elemzése

Ha van, akkor …	Ha van, akkor …
egyébként	Ha nincs, akkor

vadhajtások
hibás, pontatlan

Üres ciklus?

```
int i;
for (i = 0; i < list.Count && !(«logikai kif.»); i++)
;
<<eredmény>>
```

„~~break;~~” => ciklusfeltétel

for() => while()

```
int i=0;
while (i < list.Count && !(«logikai kif.»))
      i++;
<<eredmény>>
```

Mindent bele egy ciklusba…

```
for (int i = 0; i < items.size()
        or items.push_back(make_pair(item, 1)),   false); i++){
    if (items[i].first == item){
        items[i].second++;
        break;
    }
}
```

„~~break;~~” => ciklusfeltétel

vadhajtások

Tervezés

Keresés

Keresés tétel algoritmusa

Fejben:

rövidzár I≤N
T(X[I])… öööö
és nem T(X[I])…???
T(X[I]) ⇒ I≤N

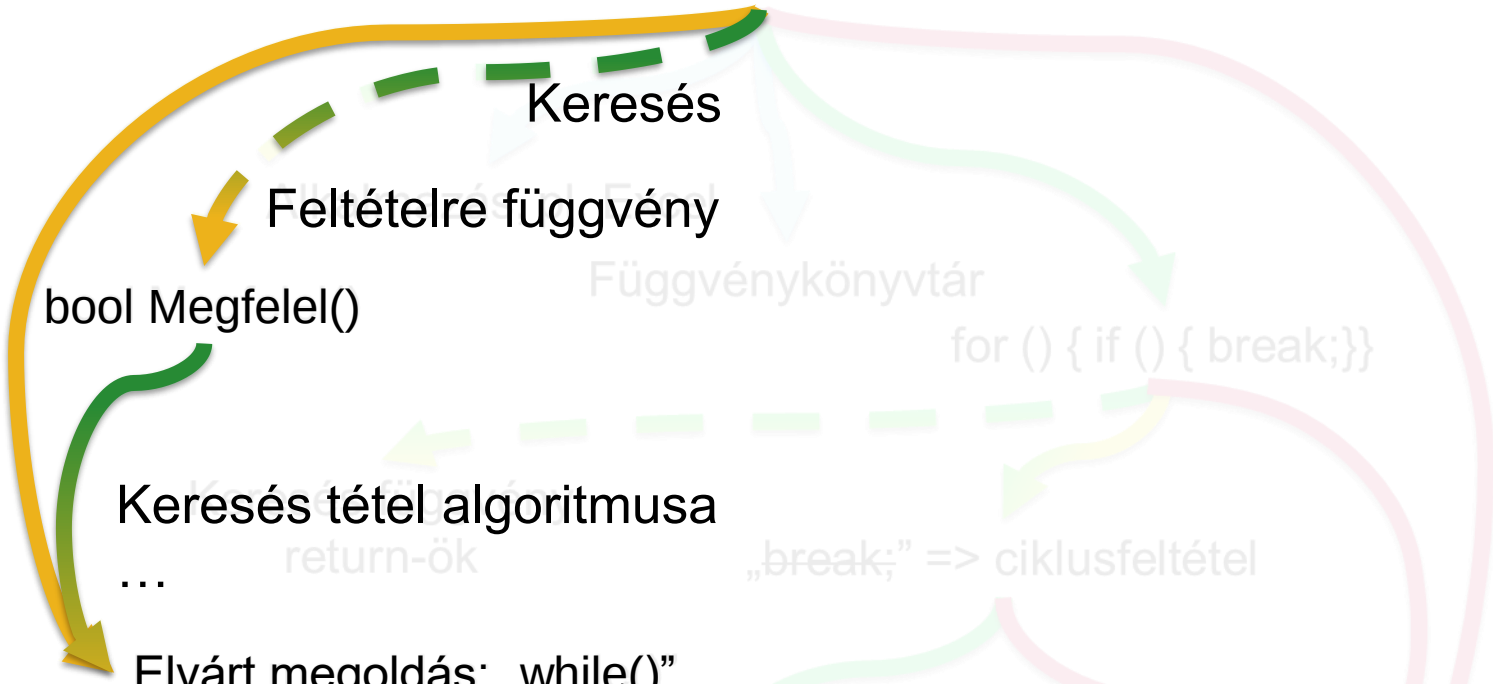
Elvárt megoldás: „while()”

```
I:=1
Ciklus amíg I≤N és nem T(X[I])
    I:=I+1
Ciklus vége
VAN:=(I≤N)
Ha VAN akkor SORSZ:=I
Eljárás vége.
```

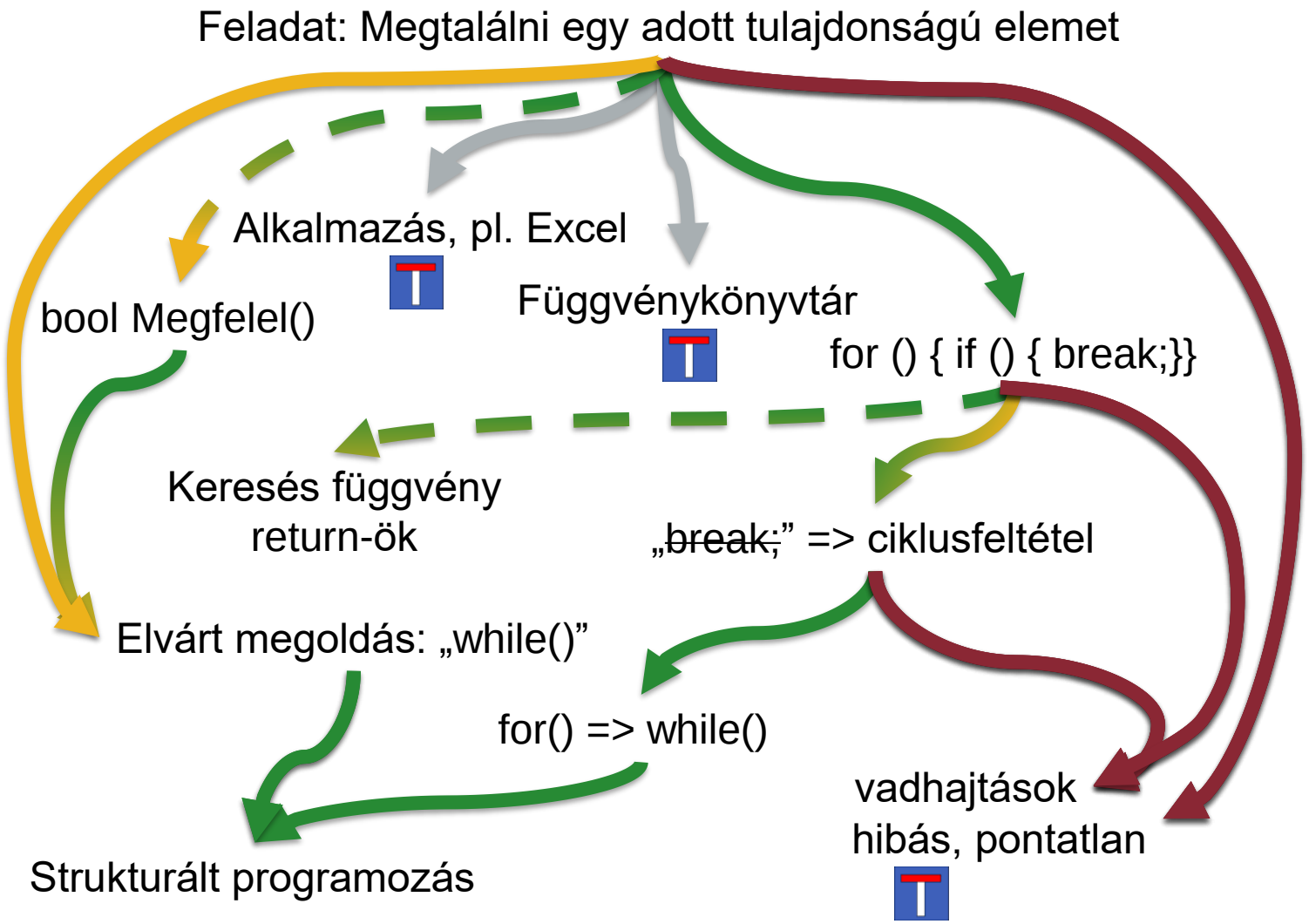
Matematikai logika:

ciklus amíg A ∧ ¬B
kilépett, mert ¬A ∨ B
B ⇒ A, „A”-t ellenőrzöm

Probléma részekre bontása



Keresésre adott megoldások



Így Írtok Ti…

Feladat

A csatolmányban található érettségi feladat … megoldását szeretném kérni tőled, speciális dokumentációval:
A feladat megoldását tetszőleges nyelven meg lehet adni, de a megoldásba részletesen bele kellene írni, hogy közben mire gondoltál, miért, hogyan választottál nyelvet, mikor volt "hogy is van", hol gépelted el, mikor tesztelted, ismerted-e...
Mindezt nem úgy, ahogy tanítanád, hanem úgy, ahogy gyorsan, ösztönösen csinálnád.
…

Kérem, hogy a feladatot ne nézd meg előre, csak ha rögtön meg is oldod. Örülnék, ha megírnád, mennyi időt foglalkoztál vele. (Lehet mérgelődni is, minősíteni is... és ha közben feladod, azt is írd meg, küldd el)

4. Zenei adók

A rádióhallgatás ma már egyre inkább zene vagy hírek hallgatására korlátozódik. Ez a feladat három, folyamatosan zenét sugárzó adóról szól, azok egyetlen napi műsorát feldolgozva. A reklám elkerülése érdekében az adókat nevük helyett egyetlen számmal azonosítottuk.

A *musor.txt* állomány első sorában az olvasható, hogy hány zeneszám (z<=1000) szólt aznap a rádiókban, majd ezt z darab sor követi. Minden sor négy, egymástól egyetlen szóközzel elválasztott adatot tartalmaz: a rádió sorszámát, amit a szám hossza követ két egész szám (perc és másodperc) formában, majd a játszott szám azonosítója szerepel, ami a szám előadójából és címéből áll. A rádió sorszáma az 1, 2, 3 számok egyike. Az adás minden adón 0 óra 0 perckor kezdődik. Egyik szám sem hosszabb 30 percnél, tehát a perc értéke legfeljebb 30, a másodperc pedig legfeljebb 59 lehet. A szám azonosítója legfeljebb 50 karakter hosszú, benne legfeljebb egy kettőspont szerepel, ami az előadó és a cím között található. A számok az elhangzás sorrendjében szerepelnek az állományban, tehát a később kezdődő szám későbbi sorban található. Az állományban minden zeneszám legfeljebb egyszer szerepel.

Például:

```
677
1 5 3 Deep Purple:Bad Attitude
2 3 36 Eric Clapton:Terraplane Blues
3 2 46 Eric Clapton:Crazy Country Hop
3 3 25 Omega:Ablakok
…
```

Készítsen programot *zene* néven, amely az alábbi kérdésekre válaszol! Ügyeljen arra, hogy a program forráskódját a megadott helyre mentse!

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: 3. feladat:). Ha a billentyűzetről olvas be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár.

Az adatszerkezet készítése során vegye figyelembe az Ön által használt programozási környezetben az adatok tárfoglalási igényét!

1. Olvassa be a *musor.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, akkor a forrás első 10 sorának adatait jegyezze be a programba, s úgy oldja meg a következő feladatokat!
2. Írja a képernyőre, hogy melyik csatornán hány számot lehetett meghallgatni!
3. Adja meg, mennyi idő telt el az első Eric Clapton szám kezdete és az utolsó Eric Clapton szám vége között az 1. adón! Az eredményt *óra:perc:másodperc* formában írja a képernyőre!
4. Amikor az „Omega:Legenda” című száma elkezdődött, Eszter rögtön csatornát váltott. Írja a képernyőre, hogy a szám melyik adón volt hallható, és azt, hogy a másik két adón milyen számok szóltak ekkor. Mivel a számok a kezdés időpontja szerint növekvő sor-

Felkérést kapták

BME MIT

Tanszéki tagok

ELTE IK

ProgAlap oktatók

BME VIK

ProgAlap oktatók

Informatikatanárok

(Sulinet lista, Ötszáz feladat)

Kérdések, döntések

Bocs, ha félre értettem a kérést
Módszerem az lesz, feltéve, hogy egy nem túl bonyodalmas feladatról van szó, olvasom a feladatot, és rögvest kódoljam, azt amit abban a pillanatban lehet.
(Tehát valóban nem úgy teszek, ahogy tanítanám! kicsit rosszul is érzem magamat emiatt!)

vajon kell működnie a programnak, ha egyáltalán nincs eric clapton szám? most nem figyelek erre

mi is volt a feladat? :D ja hogy mikor kezdődött, meddig tartott

A lényegi részéig nem jutottam el, mert elkezdtem szépen megcsinálni (ellenőrizni a megadott kényszereket, teszteket írni hozzá, szép kódot írni).

… Azt hiszem ebből látszik hogy dilemmázok ezen. Azon legfőképpen, hogy listákat kellene inkább mutatni tömbök helyett vagy gyorsan utána.
Ezzel azt akartam kifejteni, hogy a struktúrás, fix hosszú tömb jelenleg még az iskolai stílusom, amin a program írás közben nem nagyon gondolkoztam.

Programozási nyelv

Nem ismertem ezt a feladatot. Mielőtt olvastam volna, már eldöntöttem, hogy C#.

0. indextől kezdem, nehogy az legyen a kifogás, mégha nem is szeretem!

```
st_one <- dat[dat$station==1,]
claptonon1 <- which(st_one$performer == "Eric Clapton")
cl_first <- min(claptonon1)
cl_last <- max(claptonon1)
```

```
# Klasszikus Python megoldás:
cx = '|'.join(cikkek)
```

```
var egyesadó =  műsorok.Where(x => x.adó == 1).ToList();
//linq-t ki akarom használni, a sum tulajdonságot
```

```
adomusor[m[n].ado-1][adodb[m[n].ado-1]]=n;
//ha lenne pacal compilerem, abban irtam volna
```

```
Console.WriteLine(v.Where(x => x.Ssz == iSsz).Aggregate("\n7.
feladat:\n", (c, n) => c += n.Darab + " " + n.Árucikk + "\n"));
```

```
return (from aru in arucikk where aru.Equals("F") select aru).Count();
// Megoldás LINQ segítségével
```

Bemenet

...Kell egyáltalán ellenőrizni a bemenetet és tesztelni?

az elején nyilván reguláris kifejezésekkel illett volna boncolni a sztringeket, de ehhez nem igazán volt kedvem

tesztelést, bemenet ellenőrzést el kell felejteni:)

gonosz már az input formátum is...

Értelmes kimenetet csak -DNDEBUG-al fordítva látsz, ami arra utal, hogy az "az input file helyes, a feladatoknak van megoldása" állítással én korlátozottan értek egyet

```
zenek=fopen("musor.txt","rt"); //nem kell hibakezeles :-) :-)
```

Debug - önismeret

a bugfixelésekkel ment el idő, de nagyjából egyenletesen - az R gyengén típusos és a listák, vektorok és adatkeretek között oda-vissza zsonglőrködve az ember tud hibázni

most itt bő tíz percet debugoltam amiatt, mert a szövegfájlban windowsos fájlvégek vannak, én pedig linuxon vagyok.

```
const int  MaxZ=1000;
//persze, hogy az 'int'-et kihagytam, nagy sietségemben
```

azért írtam így, mert elfelejtettem a feladat kitételét, hogy csak 3 adó van összesen

{65. sor, a beolvasás után} lefordítom, mert már nagyon szeretnék egy kis sikerélményhez jutni! :))

```
record.name = tmp.substr(pos+1, tmp.size()-pos);
// muszaj volt kirpobalni, mert biztos voltam, hogy ugyis elrontom
```

Változónév

int z;
//hát nem igazán tetszik ez az azonosító, de megpróbálom nem elfelejteni, hogy ez lesz a zenék száma

size_t omega_start_time;
// ezeknek kene valami ertelmesebb nev, de ez van

typedef TZene TZenek[MaxZ];
//francba! MaxN-et írtam mert mindig az szokott a méret maximuma lenni

Struktúra

```
std::map<int, int> melyiken_hanyszam;
//rengetegen megbuknának az érettségin, akik átmennek prog1-ből.
```

```
private Dictionary<string, int> termekek = new Dictionary<string, int>();
```

```
protected struct Zeneszám
//egy osztály létrehozását itt feleslegesnek tartottam, elég a struct
```

```
szam *tomb = (szam*)malloc(n*sizeof(szam));
```

Tárkezelés

Mivel nem volt kikotes, a harom reszfeladathoz ujra es ujra olvasom az adatokat, igy oldom meg azt, hogy gondoljak a tar optimalis kihasznalasara.

char eloado[50], cim[50];
// A : miatt a '\0' biztos befér, nincs kedvem jobban kicentizni a max. méreteket

szam *tomb = (szam*)malloc(n*sizeof(szam));
// lehetett volna nem dinamikus 1000 eleműt használni, de ügyelek a tárfoglalási igényre

frászt tökölök azzal -- legalábbis elsőre --, h. csökkentsem a méretet; … Ami 1000-es darabszámot illeti: gyanítom, h. nem lesz értelme.

int N; //erre nincs is szükségem később, nem kell adattagként

Keresés

foreach (Zeneszám z in műsorok)…
//kilépjek-e ha megvan mind a három adóra vagy ne tegyek be még egy vizsgálatot - melyik hatékonyabb ...

while (vi >-1 && !van)… végigiterálok, csak „eric claptonokat” nézek

```
const string keresendo="Eric Clapton"; const int kHossz=keresendo.size();
int i=0;
while (/*i<z &&*/ !(A_Zenek[i].ado==1 && A_Zenek[i].szam.substr(0,kHossz)==keresendo)){
        ++i;
}
        while (index < zarak.Count && !zarak[index].Imsetlodo()) index++;

if (omega_channel != -1) {/* erre eloszor egy kulon bool found flaget hasznaltam*/
        break;
}                       for(n=0;strncmp(m[n].szam,"Omega:Legenda",13)!=0;++n);
for(i=0;i<n;i++){…
        if(!strcmp(tunes[x-1],"Omega:Legenda"))
                break;
}                                                       ez is menjen előrefelé, és break
```

Jellemző

Legfontosabb a hibakezelés

Stuktúrált program

beszédes kód, minimális tárhely

Többféle megoldás, dilemma

Köszönöm minden névtelen forrásomnak a fáradozást és őszinte megjegyzéseit!