Ez a wikioldal 2008 februárjától a BME alkalmazott matematikus hallgatói számára oktatott Informatika 2 tárgy honlapja. Kiindulási alapja a 2007-ben Szabó Péter által tartott kurzus. A tárgy oktatásához felhasznált, http://wiki.math.bme.hu/ -n belüli wikioldalak GNU FDL licenc vagy (választás szerint) CC-BY-SA-2.0 licenc szerint szabadon használhatók és terjeszthet?k.

Pótlás: 2008-05-23, péntek H45a 10:00

Az 1. ZH feladatai és megoldása, a 2. ZH feladatai, a 1. pótZH feladatai, a 2. pótZH feladatai.

1. ZH: 2008-04-03 csütörtök 16:00 Ka26

A C nyelvr?l szóló el?adások és gyakorlatok anyaga itt található.

A ruby nyelvr?l szóló el?adások és gyakorlatok anyaga majd itt van. És **ugyanitt az info a 2. ZH-ról!**

Tartalomjegyzék

A tárgyról

Oktatók

- Az el?adásokat és a gyakorlatokat Wettl Ferenc tartja <wettl kukac math p bme p hu>.
- T?z Csaba javítja a házi feladatokat és a ZH-kat, gondozza a SIO rendszert és konzultációkat tart.

Egyéb személyek:

- A Live CD-t készíti Er? Zsolt <zsero kukac math p bme p hu>.
- Az el?d tárgy utolsó honlapja Rácz Balázs oldalán található:
  http://www.ilab.sztaki.hu/~bracz/p/cimp2_2006tavasz.html

Hallgatók

A gyakorlati ismeretek kell? elsajátításához elengedhetetlen, hogy a gyakorlati kurzuson mindenki saját számítógéppel dolgozzon.

A Neptun beosztásához képest a hallgatók tankört cserélhetnek az egyenletes beosztás követelményét figyelembe véve.

Tárgykövetelmények

A szorgalmi id?szakban teljesítend?:

- A kiadott házi feladatok megoldása és beadása a SIO rendszeren keresztül. Csak azt a feladatot kell beadni, amelynek a beadási határideje ezen a wikioldalon szerepel. Addig kell próbálkozni, amíg a rendszer OK-val el nem fogadja a megoldást. Az el nem fogadott megoldás és a meg sem kisérelt megoldás is 0 pontot ér feladatonként. Az elfogadott megoldás 1 pontot ér.
- Az els? ZH megírása. A ZH valószín?leg papíralapú lesz. Az els? ZH témája: C programozási nyelv és egyszer? algoritmusok. Csak azt kell tudni, ami el?adáson vagy gyakorlaton elhangzott. Puskát lehet majd használni. Lásd még a #Az els? ZH c. szakaszt.
- A második ZH témája: Ruby programozási nyelv és objektum-orientált programozás. Csak azt kell tudni, ami el?adáson vagy gyakorlaton elhangzott. Puskát lehet majd használni.

A félévközi jegy a fentiek összesítésével lesz meghatározva. Elégtelen osztályzatot kap,

- aki a gyakorlatok több, mint 30%-áról hiányzott;
- aki a jegybeiratásig (de legkés?bb a szorgalmi id?szak végéig) nem adta be OK-val elfogadva az összes kötelez? házi feladatot;
- aki a két ZH egyikét sem teljesíti legalább elégségesre, vagy az elégtelenre sikerült, illetve meg nem írt ZH-t a pótlás alkalmával sem tudja megírni legalább elégségesre;

Aki nem elégtelen osztályzatot kap, annak az osztályzata a ZH-kra kapott pontszámok összegéb?l számítódik.

Ha a jegybeiratás el?tt bebizonyítódik, hogy valaki nem maga készítette a házi feladatát, vagy elkészítette valaki más házi feladatát, akkor az érintettek ellen fegyelmi eljárás indul a TVSz szerint.

Katalógus

A gyakorlatok látogatása kötelez?, minden gyakorlaton van katalógus.

A hallgatók munkája

El?adáson a hallgatók figyelnek és jegyzetelnek. Gyakorlaton a gyakorlat ideje alatt a hallgatók megoldják a kit?zött feladatokat. Minden hallgatónak saját számítógépen dolgozik. Gyakorlaton praktikusabb füzet helyett fájlba jegyzetelni. A gyakorlatvezet? segít, és ellen?rzi is a megoldást. A gyorsan haladók (a) szorgalmi feladatot kapnak, vagy kitalálnak naguknak (b) segítik a mellettük ül? lemaradt diákok munkáját. Ennek érdekében a hallgatók úgy ülnek le, hogy minden kezd? mellé üljön egy profi. Profinak számít az, aki az alábbiakból sokat tud: Linux grafikus felületek haladó felhasználói szint? ismerete, Linux parancssor felhasználói szint? ismerete, hatékony munka Linux parancssorban, hatékony munka Midnight Commanderben, angol nyelvtudás, jó hibadiagnosztizálási és hibajavítási készség, az órán kívül is gyakran használ Linuxot, tetsz?leges programozási nyelv legalább hobbi-szint? ismerete, ismeretlen program m?ködésének megértése weben fellelhet? információk alapján, ismeretlen program m?ködésének megértése man page alapján, tetsz?leges program telepítése Windowsra, tetsz?leges program telepítése valamely Linux-disztribúcióra. A profinak nem kell ismernie se a C nyelvet, se a Ruby nyelvet.

A hallgatók rendszeresen tanulnak például az ajánlott irodalomból, esetleg az órai jegyzetükb?l, és saját, vagy a labor gépein gyakorolnak egy pl. adott feladatot adott módon megoldó program elkészítésével, vagy ilyen példaprogramok tanulmányozásával.

A hallgatók egyénileg (pl. otthon vagy a számítógép-laborban) megcsinálják a házi feladatokat, és beküldik a SIO rendszerbe. A beküldéshez internetelérés (web) szükséges. Ha valaki megakad a feladat megoldásában, kérdezzen a konzultáción, vagy évfolyamtársaitól, de *NE CSINÁLTASSA MEG MÁSSAL*!

Ajánlott szoftverek

C nyelvhez Linux alatt ajánlott:

- sima szövegszerkeszt?: kate
- fordítóprogram: gcc
- fejleszt?környezet: Kdevelop

C nyelvhez Windows alatt ajánlott:

- fejleszt?környezet: Dev-C++: http://www.bloodshed.net/dev/devcpp.html Ezt kell letölteni: Dev-C++ 5.0 (9.0 MB) with Mingw/GCC.

Ruby nyelvhez Linux alatt ajánlott:

- sima szövegszerkeszt?: kate
- interpreter: ruby
- fejleszt?környezet: FreeRIDE http://rubyforge.org/frs/?group_id=31 freeride-linux-installer-0.9.6.sh

Ruby nyelvhez Windows alatt ajánlott:

- fejleszt?környezet: FreeRIDE http://rubyforge.org/frs/?group_id=31
  freeride-win-installer-0.9.6-1.exe

Er? Zsolt egy Live CD-t készített, amin a fenti linuxos szoftverek megtalálhatók, és hosszadalmas telepítés nélkül kipróbálhatók. Barbalinux

Tananyag

- algoritmusok C nyelven
- objektum-orientált programozás Ruby nyelven

Ajánlott irodalom:

- Standard C reference (2 oldalas) http://www.math.bme.hu/~pts/szimp2/stdc2.pdf
- Brian W. Kernighan--Dennis M. Ritchie: A C programozási nyelv. Az ANSI szerint szabványosított változat. Kapható a jegyzetboltban.
- ruby.hu http://www.math.bme.hu/~pts/ruby.hu/1_eloszo.html (korábban innen volt elérhet?: http://segabor.web.elte.hu/ruby.hu/1_eloszo.html)
- Windowson interaktív, angol nyelv? Ruby tutorial: http://hacketyhack.net/

(Letölt, telepít, elindít.)

Egyéb irodalom:

- Ruby alaposztályok összes metódusa, angol nyelv? referencia: http://www.ruby-doc.org/core/
- egyéb leírás Rubyról: Ruby User's Guide (régi): http://www.math.hokudai.ac.jp/~gotoken/ruby/
- rövid összefoglaló Rubyról: Ruby Quick Reference (Ruby QuickRef): http://www.zenspider.com/Languages/Ruby/QuickRef.html

BarbaLinux

Azok számára, akik a C programozást a gyakorlatokon megszokott környezetben (Kate és GCC) szeretnék gyakorolni, Er? Zsolt készített egy CD-t, amely tartalmazza ezeket a programokat. Köszönet érte!

Részleteket a Barbalinux wiki oldalán találhattok itt: Barbalinux

Konvenciók

A konvenciók mind a hallgatók, mind az oktatók számára kötelez?en betartandók.

- A tárggyal kapcsolatos fájlokat mindenki a ~/info2 mappába helyezze.
- Minden fájlnév és mappanév kisbet?s.
- A fájlnevek ékezetes bet?ket és szóközöket nem tartalmaznak.
- A forrásfájlok (*.c és *.rb) ékezetes bet?ket nem tartalmaznak.
- Minden program forráskódja külön (*.c vagy *.rb) fájlba kerül. Tehát nem írunk felül egyetlen korábban létrehozott fájlt sem, hanem lemásoljuk a fájlt, és más néven mentjük el. A más néven mentést még a fájl tartalmának módosítása el?tt végrehajtjuk.
- A sor elején lev? szóközök számának megállapításánál betartjuk a *beljebb kezdési szabályt* (C nyelvhez lásd az 1. gyakorlat anyagában).

Gyakorló feladatok

Elméleti kérdések

- Tekintsük a `struct nagy { int i; char c1, *c2; short s; char c3; double d; }` struktúratípust.
  - Mekkora a `struct nagy a[10][20][30], *b;` változók mérete igazítás (alignment) nélkül? Segítség: a char 1 bájt, a short 2 bájt, a double 8 bájt, a többi 4 bájt.
  - Mekkora a `struct nagy a[10][20][30], *b;` változók mérete igazítással? Segítség: a char igazítása 1 bájt, a short igazítása 2 bájt, a többi igazítása 4 bájt.
  - Csökkentse minimálisra a struktúra igazításos méretét pusztán a mez?k áthelyezésével. Mennyi az új méret?
- Milyen (geometriai) transzformációt végez a derékszög? koordinátarendszerbeli (x,y) ponton az `x-=y+=x-=y;` utasítás?
- Tekintsük a törtszámokat összeadó *osszead* függvényt:

```
... ...(...) {
  egyszerusit(as,an);
  egyszerusit(bs,bn);
  en=lkkt(an,bn);
  es=(en/an)*as+(en/bn)*bs;
  egyszerusit(es,en);
}
```

- Egészítse ki a ...-os részeket. A paraméterek sorrendje tetsz?leges.
- Pótolja a kihagyott `&` és `*` jeleket.
- Változtat-e valamit az eredményen, ha az utolsó utasítást (`egyszerusit(es,en);`) kihagyjuk? A válasz csak indoklással együtt fogadható el.

- Tekintsük az alábbi függvényt egy am?bajátékot játszó programból. Az aktuális játékállás a *t* változóban található. A tábla 19x19 mez?b?l áll.

```
int feladvany1(int n, int m) {
  int x, y;
  char c;
  for (y=0;y<n;y++) {
    for (x=4;x<m;x++) {
      if (0==(c=t[y][x])) {
      } else if (c==t[y][x-1] && c==t[y][x-2] && c==t[y][x-3] && c==t[y][x-4]) {
        return c;
      }
    }
  }
  return 0;
}
```

- Deklarálja a *t* változót.
- Mit számol ki (pontosabban: mit ad vissza) a függvény?
- Mit tartalmaz a *t* változó a játék kezdetekor?
- Miért lenne rossz, ha a kódban `x=4` helyett `x=0` szerepelne?

- Tekintsük az alábbi függvényt. Minden kett?spontra végz?d? sorhoz adja meg, hogy mit tartalmaz a *t* tömb.

```
void feladvany2(void) {
  int t[9], *p, i;
  t[0]=0; t[1]=1;
 januar:
```

```
  for (i=2;i<9;i++) t[i]=t[i-1]+t[i-2];
 februar:
  p=&t[3];
 marcius:
  t[0]=*p++;
 aprilis:
  t[1]=*++p;
 majus:
  t[2]=(*p)++;
 junius:
  t[3]=++(*p);
 julius:
  t[4]=++*p;
 augusztus:
  *p+=*p;
 szeptember:
  ;
}
```

- Tekintsük az alábbi ciklust:

```
for(i=0;i<=100&&t[i]!=42;i++){}
```

- ♦ Deklarálja a változókat úgy, hogy a lehet? legkevesebb memóriát használja.
- ♦ Írja át a ciklust *for*-b?l *while*-ba.
- ♦ Valósítsa meg a ciklust *goto*-val.
- ♦ Rajzolja le a ciklus m?ködését folyamatábrán.
- ♦ A ciklus befejeztével mi az *i* változó értékének jelentése?
- ♦ A ciklus befejeztével az *i* változó milyen esetben veszi fel a maximumát?

- Tekintsük az alábbi programrészletet:

```
void helyben_transzponal(int t[9][9])
int x, y
double d, *p
for (y=0;y<9;y++)
p=&t[y][0]
for (x=y+1;x<9;x++)
d=p[x]
p[x]=t[x][y]
p[x]=d
```

- ♦ Pótolja a hiányzó pontosvessz?ket és kapcsos zárójeleket.
- ♦ Pótolja a tanult módon a hiányzó soreleji szóközöket (indentation).
- ♦ Lassít vagy gyorsít a függvény futásán, ha `y+1` helyett `y`-t írunk?
- ♦ Változtat-e a függvény futásának eredményén, ha `y+1` helyett `y`-t írunk?

- Tekintsük az alábbi függvényt:

```
int fib(int n) {
  if (n<1) return 0;
  return fib(n-1)+fib(n-2);
}
```

- ♦ A függvény az *n*-edik Fibonacci-számot adná vissza, ha helyesen m?ködne. Mi a hiba?
- ♦ Javítsa ki a hibát 2 karakter megváltoztatásával.
- ♦ A Fibonacci-sorozatot értelmezhetjük negatív *n*-ekre is (a rekurzív képlet ugyanaz, mint pozitívakra). Módosítsa a programot, hogy negatív *n*-ekre is m?ködjön. Háromsoros

megoldást adjon.

- Tekintsük az alábbi *keres* függvényt, amely az *n* elemb?l álló, nagyság szerint rendezett *t* tömbben keresi meg az *e* értéket, és tekintsük az ?t hívó *feladvany3* függvényt.

```
int keres(int t[], int n, int e) {
  int a=0, b=n-1, f;
  while (a<=b) {
    f=(a+b)/2;
    if (e==t[f]) return f;
    else if (e<t[f]) b=f-1;
    else a=f+1;
  }
  return n;
}

int feladvany3(int t[], int n, int e) {
  int a=keres(t, n, e), b=a;
  while (a>0 && t[a-1]==e) a--;
  while (b<n && t[b]==e) b++;
  return b-a;
}
```

- ♦ M?ködik-e a *keres* függvény minden monoton csökken? tömbre? A válasz csak bizonyítással együtt fogadható el.
- ♦ M?ködik-e a *keres* függvény minden monoton növekv? tömbre? A válasz csak bizonyítással együtt fogadható el.
- ♦ Mit ad vissza a *keres* függvény, ha a keresett érték szerepel a tömbben?
- ♦ Megtalálja-e a *keres* függvény a keresett értéket, ha az többször is szerepel? Ha igen, melyik el?fordulását találja meg?
- ♦ Mit ad vissza a *keres* függvény, ha a keresett érték nem szerepel a tömbben?
- ♦ Mit számol ki (pontosabban: mit ad vissza) a *feladvany3* függvény, ha a keresett érték szerepel a tömbben?
- ♦ Mit ad vissza a *feladvany3* függvény, ha a keresett érték nem szerepel a tömbben?

- Tekintsük az alábbi deklarációt és függvényt.

```
struct csucs {
  int ertek;
  struct csucs bal , jobb;
};

int feladvany4 struct csucs gyoker {
  int bal , jobb;
  if gyoker == NULL return 0;
  return feladvany4 gyoker . bal + feladvany4 gyoker . jobb + gyoker . ertek;
}
```

- ♦ Pótolja a hiányzó `*`-okat és zárójeleket (a lehet? legkevesebbet).
- ♦ Mit számol ki (pontosabban: mit ad vissza) a függvény?
- ♦ Hányszor hívja meg önmagát a függvény egy *n* csúcsból álló fa esetén? Teljesen pontos választ adjon.

- Tekintsük az alábbi függvényt, ami 3 egész szám összesített eltérését határozza meg a közép? elemt?l:

```
int medianelteres(int a, int b, int c) {
  a=abs(a-kozepso(a, b, c));
  b=abs(b-kozepso(a, b, c));
  c=abs(c-kozepso(a, b, c));
  return a+b+c;
```

```
}
```

- ♦ A függvényben van egy alapvet? hiba, például a `medianelteres(7, 6, 5)` hívás eredménye 6, noha azt szeretnénk, hogy 2 legyen. Javítsa a hibát új változó bevezetésével.
  ♦ Javítsa a hibát új változó bevezetése nélkül. (Át kell írni az egész kapcsos zárójelen belüli részt esetszétválasztásosra.)
- Tekintsük az alábbi függvényt:

```
int feladvany5 int n {
int c = 0 int d = 2
if n < 0 n *= - 1
while n > 1{
if n % d == 0 {
n /= d c ++
} else {
++ d
}
}
return c
}
```

- ♦ Pótolja a hiányzó zárójeleket és pontosvessz?ket.
  ♦ Pótolja a tanult módon a soreleji szóközöket (indentation).
  ♦ Mit számol ki (pontosabban: mit ad vissza) a függvény?
- Deklaráljon egy függvényt, ami kiszámolja három egész szám legnagyobb közös osztóját.
- Deklaráljon egy függvényt, ami b?vít egy törtet (számláló,nevez?) egy adott számmal.
- Deklaráljon egy függvényt, ami összehasonlít két bináris keres?fát, hogy egymás tükörképei-e. El?bb definiálja a használt struktúratípust.
- Deklaráljon egy függvényt, ami lemásol egy bináris keres?fát, és visszaadja a másolatot. El?bb definiálja a használt struktúratípust.
- Deklaráljon egy változót, ami egy int-ekb?l álló háromdimenziós tömb.
- Deklaráljon egy változót, ami egy short-okra mutató mutatókból álló kétdimenziós tömb.

Az alábbiakhoz hasonló, tisztán elméleti feladatok nem lesznek:

- Mi a különbség a predekrementálás és a posztdekrementálás között?
- Mik a C nyelv kulcsszavai?
- Hogyan kell paraméterezni a scanf függvényt?
- Mi a strúktúratípus-definíció formális szintaxisa?
- Hogyan lehet egy függvényb?l két értéket visszaadni?

Programozási feladatok

- Írjon olyan függvényt C nyelven, ami visszadja egy bináris fában található csúcsok összegét. Deklaráció: `int ertekeket_osszead(struct csucs *gyoker);`
- Írjon olyan függvényt C nyelven, ami összehasonlít két bináris fát, és 0-t ad vissza, ha megegyeznek, és 1-et, ha különböznek. Az egyezéshez nem elég, hogy ugyanazokat az értékeket tartalmazzák, a gráfnak is meg kell egyeznie. Deklaráció: `int osszehasonlit(struct csucs *gyoker1, struct csucs *gyoker2);`
- Írjon olyan függvényt C nyelven, ami visszadja egy 1-nél nagyobb egész szám legkisebb, 1-nél nagyobb (prím)osztóját. Deklaráció: `int minoszto(int n);`. Az osztó keresését 2-t?l négyzetgyök n-ig végezze.
- Írjon olyan függvényt C nyelven, ami visszadja az n karakterb?l álló t tömbben a leghosszabb szó hosszát (ami 0 is lehet, ha egy szó sincs a tömbben). Szónak számít egy szóközt nem tartalmazó, tovább nem b?víthet?, összefügg? részsorozat. Deklaráció: `int maxszohossz(char t[],`

`int n);`

- Írjon egy olyan rekurzív függvényt C nyelven, ami kiszámolja az n alatt a k binomiális együtthatót modulo m. Deklaracio: `int binom(int n, int k, int m);` Használja ki, hogy 0 alatt a 0 egyenl? 1-gyel, egyébként 0 alatt az n egyenl? 0-val, egyébként pedig n alatt a k egyenl? (n-1) alatt a k plusz (n-1) alatt a (k-1).
- Írjon olyan függvényt C nyelven, ami keres a 100-szor 100-as m mátrixban két azonos sort, és visszadja azok sorszámát (0 <= i < j < 100). Ha nincs két azonos sor, akkor i=j=100-at ad vissza. Deklaráció: void azonos_sort_keres(int m[100][100], int *i, int *j);</code>
- Írjon olyan függvényt C nyelven, ami kiszámolja három szám legnagyobb közös osztóját. Írhat segédfüggvényt is. Deklaráció: `int lnko3(int a, int b, int c);`
- Írjon olyan függvényt C nyelven, ami meghatározza, hogy egy háromszög derékszög?-e, és 0-et ad vissza, ha nem, 1-et, ha az A csúcsnál van derékszög, 2-t, ha a B csúcsnál, és 3-at, ha a C csúcsnál. A háromszög 3 csúcspontjával van megadva, minden koordináta egész szám. A derékszög?séget vektorok skalárszorzatával határozza meg. Deklaráció: `int derekszogu_e(int ax, int ay, int bx, int by, int cx, int cy);`.

Házi feladatok

A házi feladatokat a SIO rendszer segítségével kell beadni. Minden SIO-ban kit?zött feladatot be kell adni, kivéve amelyekhez explicit módon oda van írva, hogy szorgalmi feladat.

Technikai részletek

A C fordításhoz használt szoftver a *gcc* (a pontos verzió: `gcc 4.1.2 Debian 4.1.1-21`). A házi feladatokat a következ? módon fordítjuk:

```
$ gcc -x c -O2 -std=c99 -pedantic-errors -static -o <prog> <forrásfájl>.c -lm
```

A feladat megoldásakor el?bb nem a fenti paranccsal, hanem az alábbival érdemes el?bb kipróbálni a fordítást:

```
$ gcc -W -Wall -s -O2 -std=c99 -pedantic-errors -lm -o <program> <program>.c
```

Ez azért jó, mert a `-W -Wall` sok hasznos figyelmeztet? üzenetet megjelenít, és ily módon a hibákra hamar felhívja a figyelmet.

A házi feladatok listája elérhet? a SIO rendszerben is: https://sioweb.math.bme.hu/user.phtml?op=zadania

Típushibák a házikban

- A *Compile error* oka lehet az is, hogy a beadott fájl kiterjesztése nem `.c` . Tipikus rossz kiterjesztések: `.cc`, `.cpp`, `.cxx`.
- A *runtime error*-nak lehet oka, hogy a `return 0;` hiányzik a *main* függvény végér?l.
- A forrásfájl utolsó sorának végén nincs soremelés (emiatt *compile error*).
- Nagy, 8 MB-nál nagyobb tömb van a main függvényen belül (emiatt *runtime error* és *segmentation fault*). Megoldás: a tömb deklarációját a függvényen kívülre tenni. Például így jobb:

```
char nagytomb[10000000];
int main(void) {
  ...
  return 0;
}
```

- A getchar() eredménye *char* típusú változóba kerül. Ez azért nem jó, mert a char csak 256 különböz? értéket tud tárolni, de nekünk 257 kéne (a 256 karakter, és a fájl végét jelz? -1). Megoldás: *int* típusú változóba tenni.
- Függvényen belül függvény van definiálva. Ez nem jó, a függvényeket csak egymás után szabad definiálni.
- A program fölösleges szóközöket ír ki (pl. a sor végére), és emiatt *wrong answer*-ös a megoldás. Ha azt kell kiírni, hogy hello, majd egy soremelést, akkor jó megoldás a `printf("http://wiki.math.bme.huhello\n"http://wiki.math.bme.hu);`, rossz megoldás a `printf("http://wiki.math.bme.huhello \n"http://wiki.math.bme.hu);`
- Az eratoszthenészi szitát while cikluson belül az összes feladványra végigszámolja, pedig elég lenne összesen egyszer. Emiatt *time limit exceeded*-es a megoldás.
- Tetsz?leges karaktert nem lehet beolvasni gets()-sel vagy fgets()-sel, mert elnyelik a 0-s kódú karaktereket. A helyes megoldás karakter beolvasásásra getchar().
- A program beszélgetni próbál a felhasználóval, pl. kiírja, hogy *Most kérem a mátrix sorainak számát*. Épp amiatt rossz (*wrong answer*) a kimenet, hogy ez az üzenet is szerepel benne.
- A program egymás alá próbálja igazítani a számokat. Épp amiatt rossz (*wrong answer*) a kimenet, hogy az igazítás miatt fölösleges szóközöket ír ki.
- A program túl sok memóriát használ, például a szitáláshoz használt tömb elemeit *int*-ekb?l építi, noha *char*-okba is beleférne. i386 architektúrán az int 32 bites (ebb?l 1 el?jel), a char 8 bites (ebb?l 1 el?jel), tehát egy charban a [-128,127] zárt intervallumból lehet szám.
- Túl nagynak veszi a tömböt (és amiatt *memory limit exceeded*-et kap). A feladatkiírásban pontosan meg van adva a bemeneti adatok értéktartománya. Pont akkora (vagy egy hajszálnyival nagyobb) tömböt kell deklarálni, amekkorába az adat belefér.
- A tömb mérete nem konstans. Például hibás az `int tomb[n];` deklaráció, és helyes az `int tomb[500];` deklaráció, ha *n* értéke legfeljebb 500 lehet (az értéktartományt lásd a feladat szövegében).