Problémamegoldás programozással

3. laborgyakorlat

Szoftvertervezés és –fejlesztés Intézet

Neumann János Informatikai Kar

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

A gyakorlat témakörei

- Egydimenziós tömbök, listák, a foreach utasítás
- Többdimenziós tömbök
- Hibakeresés (debug)
- További gyakorló feladatok

Egydimenziós tömbök, listák, a foreach utasítás

1. feladat

Készítsünk programot, amely ciklusok használatával felsorolja a francia kártya lapjait egy tömbbe . A lehetséges színek: `Kőr`, `Káró`, `Treff` és `Pikk`. A lapoknak 13 féle magassága lehet: számok `2`-től `10`-ig, majd `Jumbó`, `Dáma`, `Király` és `Ász`. Az 52 elemű tömb néhány eleme tehát:

```
{ “Kőr 2”, “Kőr 3”, ..., “Kőr Király”, “Kőr Ász”, “Káró 2”,
  “Káró 3”, ..., “Pikk Dáma”, “Pikk Király”, “Pikk Ász” }
```

Egydimenziós tömbök, listák, a foreach utasítás

2. feladat

Keverjük meg a korábban készített kártyapaklit a Fisher–Yates keveréssel. A módszer lényege, hogy a tömb elemein végighaladva mindegyikhez kiválaszt egy véletlen helyen lévő elemet a korábban még nem vizsgáltak közül, amelyeket utána megcserél. Az algoritmus pszeudokóddal az alábbi formában adható meg (1-alapú indexelést használva).

```
ciklus i ← 1-től n–1-ig

    j ← véletlen egész, úgy hogy i ≤ j ≤ n

    x[i] ⇔ x[j]
```

Egydimenziós tömbök, listák, a foreach utasítás

3. feladat

Kérjünk el a felhasználótól előre megadott darabszámú szót, amelyeket tároljunk el egy tömbben. Ezután kérjünk el a felhasználótól egy további szót, és válaszoljuk meg az alábbiakat.

- Benne van-e a gyűjteményben a megadott szó?
- Ha benne van, hol található először?

Egydimenziós tömbök, listák, a foreach utasítás

4. feladat

Módosítsuk az előző feladat megoldását úgy, hogy a felhasználótól bekért szavakat egy listában tároljuk el, és a bekérést a `STOP` kulcsszó megadásakor fejezzük be. Ha szükséges, módosítsuk a két előbbi lekérdezést is. Milyen hasonlóságokat és különbségeket tapasztalunk a tömbök és listák használatában?

Egydimenziós tömbök, listák, a foreach utasítás

5. feladat

Felmérést végzünk barátaink programozói ismereteiről. Kérjük el az adott személy nevét (string), életkorát (int) és hogy rendelkezik-e programozói tapasztalattal (bool). A neveket, életkorokat és tapasztalatokat tároljuk három külön listában, amelyeket az kapcsol össze, hogy egy adott indexen egy konkrét személy adatait találjuk. A bekérést egy üres név megadásáig folytassuk. Ezt követően határozzuk meg az alábbiakat.

- Mi az átlagéletkor a teljes adathalmazban? (Használjuk a `foreach` utasítást a bejáráshoz.)
- Mi az átlagéletkor a programozói tapasztalat nélküli személyek között?
- Hány éves a legidősebb, programozó tapasztalattal rendelkező személy és mi a neve?

Többdimenziós tömbök

6. feladat

Hozzunk létre egy *N*×*M*-es kétdimenziós tömböt (1 < *N*, *M* < 10), amit töltsünk fel véletlenszerűen –9 és 9 közötti értékekkel. Jelenítsük meg a képernyőn ennek a mátrixnak az elemeit. Állítsuk elő a mátrix transzponáltját*, vagyis tükrözzük azt a főátlójára.

```
1 2 3        1 4 7
4 5 6   →    2 5 8
7 8 9        3 6 9
```

```
a b c d        a e i
e f g h   →    b f j
i j k l        c g k
               d h l
```

** https://hu.wikipedia.org/wiki/M%C3%A1trix_(matematika)#Transzpon%C3%A1l%C3%A1s*

Többdimenziós tömbök

7. feladat

Egy horgászverseny fogási adatait egy *F* táblázatban (kétdimenziós tömbben) tároljuk. *F*(*i*,*j*) azt jelenti, hogy az *i*-edik horgász a *j*-edik halfajtából hány darabot fogott.

- Generáljuk le véletlenszerűen a táblázat adatait.
- Jelenítsük meg formázottan a fogási adatokat a képernyőn.
- Adjuk meg, hogy a horgászok mennyit fogtak az egyes halfajtákból.
- Melyik horgász fogta a legtöbb halat összesen?
- Volt-e olyan horgász, aki egyetlen halat sem fogott?

Hibakeresés (debug)

8. feladat

Kérjünk el a felhasználótól egy *N* pozitív egész értéket, és adjuk hozzá egy listához első elemként. Vegyük a lista utoljára hozzáadott elemét, legyen ez *K*. Ha *K* páros, adjuk hozzá a listához *K* felét, ha páratlan, akkor 3*K*+1-et. Addig ismételjük az előbbieket, amíg 1-et nem kapunk eredményül*.

Kövessük nyomon a kiszámított érték és a lista állapotának változását hibakereső módban.

Próbáljuk meg hibakeresés közben módosítani az aktuálisan kiszámított értéket.

** A Collatz-sejtés szerint akármilyen pozitív számmal is kezdünk, a végén mindig elérjük az 1-et.*

Hibakeresés (debug)

9. feladat

Az alábbi algoritmussal szeretnénk az x tömb elemeit fordított sorrendben megkapni. Használjuk a hibakereső üzemmódot a hibák felderítésére és javítására.

```
int[] x = { 1, 2, 3, 4, 5, 6, 7, 8};
for (int i = 0; i < x.Length; i++)
{
    int tmp = x[i];
    x[i] = x[x.Length – i – 1];
    x[x.Length - i] = tmp;
}
```

További gyakorló feladatok

10. feladat

Töltsünk fel egy egydimenziós tömböt megadott számú véletlen értékkel, majd valósítsuk meg az alábbi műveleteket, majd oldjuk meg a feladatot listával is.

- Válogassuk ki a gyűjtemény minden második elemét egy új gyűjteménybe
- Fordítsuk meg a gyűjtemény elemeinek sorrendjét.
- Rendezzük a lehető legkisebb négyzetes mátrixba a gyűjtemény elemeit (az esetlegesen üresen maradó értékek helyére nulla kerüljön).

További gyakorló feladatok

11. feladat

Készítsünk algoritmust, amely egy *N*×*M*-es mátrix elemeit az óramutató járásának megfelelően *K*×90°-kal “elforgatja”, ahol *K* egész szám. Két példa a *K*=1 esetre:

```
1 2 3        4 1 2
4 5 6   →    7 5 3
7 8 9        8 9 6
```

```
 1  2  3  4          5  1  2  3
 5  6  7  8    →     9 10  6  4
 9 10 11 12         13 11  7  8
13 14 15 16         14 15 16 12
```

További gyakorló feladatok

12. feladat

Töltsünk fel egy kétdimenziós tömböt véletlenszerűen `true` és `false` értékekkel. Adjunk meg egy kezdő koordinátát, majd határozzuk meg, hogy onnan eljuthatunk-e bármilyen úton a jobb alsó sarokba mindig csak szomszédos* `true` mezőkre lépve.

X = 2 és Y = 1 esetén egy lehetséges út például a pirossal jelölt. A feltételeknek eleget tevő út nem minden esetben létezik.

T F T T T F T T T T
F F T T F F T F T T
F T T T T T T F F T
F F T F F F F T T T

**Egy adott elem szomszédai legyenek a tőle balra és jobbra, valamint felette és alatta lévő elemek.*