Programozás alapjai
5. gyakorlat
Egydimenziós tömbök kezelése, eldöntés és kiválasztás algoritmusok

1. feladat: Lottósorsolás. Írjon C programot, amely előállítja az ötöslottó heti nyerőszámait.
Megoldás: A program 1 és 90 között állít elő 5 különböző véletlenszámot.
Megoldható-e a feladat tömb használata nélkül?

```
#include <stdio.h>
#include <stdlib.h>                 // rand(), srand() függvényhíváshoz
#include <time.h>                   // time() függvényhíváshoz
#define MERET 5

int main()
{
   int i, j;
   int lotto[MERET], jelolt;

   srand(time(0));                  // véletlenszám generátor inicializálása

   for (i=0; i<MERET; ) {
      jelolt = rand()%90+1;         // rand()%x : [0...x) intervallumból ad vissza egy egész számot
      for(j=0; j<i; j++) {
         if (lotto[j] == jelolt)    // ismétlődés elkerülése
            break;
      }
      if (j==i) {
         lotto[i] = jelolt;
         i++;
      }
   }
   //tömb kiírása
   for(i=0; i<MERET; i++) {
      printf("%d. szám: %d\n", i+1, lotto[i]);
   }
   return 0;
}
```

2. feladat:
Az EURO árfolyamát egy negyedéven keresztül hetente nyilvántartjuk (HUF / EUR). Írjon C programokat az alábbi kérdések megválaszolására.
a) Hányszor volt a negyedévben 300 Ft alatt az árfolyam értéke? A tömböt inicializálja.
b) Monoton nőtt-e az árfolyam a negyedév során? Alkalmazzon ellenőrzött adatbeolvasást. A válasz kiírását feltételes operátorral valósítsa meg.
c) Melyik héten volt a legmagasabb, és melyiken a legalacsonyabb az árfolyam? Melyek voltak ezek a szélsőértékek? A tömböt véletlenszámokkal töltse fel.
d) Mennyi az adott negyedévre vonatkozó átlagos árfolyam érték? Előjelhelyesen írja ki, hogy az egyes adatok mennyivel térnek el az átlagtól. Ezt a feladatot pointer használatával is oldja meg.

a) Hányszor volt a negyedévben 300 Ft alatt az árfolyam értéke? A tömböt inicializálja.

```
int main() {
   int i, db=0;
   /* Tömb létrehozása inicializációs listával */
   double tomb[ ] = {289.5, 292.6, 290.2, 295.5, 300.4, 300.3, 302.5, 303.3, 303.5, 299.9};
   int meret =  sizeof(tomb)/sizeof(double);          //tömb méretének kiszámítása
   double limit=300.0;

   printf("HUF/EUR árfolyamok:\n");
   for (i=0; i<meret; i+=1) {
       printf("%d. érték: ", i+1, tomb[i]);
      /* Feltételt kielégítő elemek megszámlálása */
      if (tomb[i] < limit)  db++;
   }
   printf("Az árfolyam értéke %d-szer volt %.2f alatt.", db, limit);
   return 0;
}
```

b) Monoton nőtt-e az árfolyam a negyedév során? Alkalmazzon ellenőrzött adatbeolvasást. A válasz kiírását feltételes operátorral valósítsa meg.

```
#define N 12
int main() {
   int i, ok;
   double tomb[N];
   char c;

   /* Tömbelemek ellenőrzött beolvasása */
   printf("HUF/EUR árfolyamok:\n");
   for (i=0; i<N; i+=1) {
      do {
         ok = 1;
         printf("%d. érték: ", i+1);
         if (scanf("%lf", &tomb[i])!=1) {
                  printf("Hibás adat. Adja meg újra:\n");
                  ok = 0;
                  while ((c = getchar()) != '\n') ;
         }
      } while(!ok || tomb[i] < 270 || tomb[i] > 320);          //értékkorlátozás
  }

  //Eldöntés algoritmus
  //Feltesszük, h. monoton növő a sorozat, nincs a monotonitást elrontó érték
  int found=0;
   /* Monotonitás vizsgálat */
   for (i=1; i<N && !found; i+=1) {
        if (tomb[i] < tomb[i-1])   found=1;   //talált a monotonitást elrontó elemet
   }
   printf("A sorozat monoton növő: %s", !found ? "igaz" : "hamis");
   return 0;
}
```

c) Melyik héten volt a legmagasabb, és melyiken a legalacsonyabb az árfolyam? Melyek voltak ezek a szélsőértékek? A tömböt véletlenszámokkal töltse fel.

```
#define N 12
int main() {
   int i, minindex, maxindex;
   double tomb[N], min, max;

   /* Tömb automatikus feltöltése */
   srand(time(0));                          // stdlib.h, time.h
   int upper = 320, lower = 270;
   double range = upper - lower;
   double div = RAND_MAX / range;
   double value;
   for (i=0; i<N; i+=1) {
      //tomb[i] = (double)(rand()%(upper-lower+1)+lower);      // pl.: 310.000000
      value = lower + (rand()/div);                            // pl.: 310.123456
      tomb[i] = round(value*100) / 100;                        // pl.: 310.120000
   }

   /* Tömbelemek kiírása */
   printf("HUF/EUR árfolyamok:\n");
   for (i=0; i<N; i+=1) {
      printf("%d. érték: %lf\n", i+1, tomb[i]);
   }

   /* Minimum kiválasztás */
   minindex=0;
   for(i=0; i<N; i++) {
      if(tomb[i]<tomb[minindex])
         minindex=i;
   }
   printf("\nA sorozat legkisebb eleme: %lf, sorszáma: %d", tomb[minindex], minindex+1);

  /* Maximum kiválasztás */
   maxindex=0;
   for(i=0; i<N; i++) {
      if(tomb[i]>tomb[maxindex])
         maxindex=i;
   }
   printf("\nA sorozat legnagyobb eleme: %lf, sorszáma: %d", tomb[maxindex],
         maxindex+1);

   return 0;
}
```

Megjegyzések:
- A sorozatból a legnagyobb és legkisebb elem kiválasztása megvalósítható egy cikluson belül.
- A *math.h round()* függvénye kerekítést valósít meg. A *floor()* függvény csak levágja a felesleges tizedesjegyeket. A *ceil()* függvény úgy adja vissza a számot, hogy az utolsó tizedesjegyet a felső szomszédjára cseréli.

d) Mennyi az adott negyedévre vonatkozó átlagos árfolyam érték? Előjelhelyesen írja ki, hogy az egyes adatok mennyivel térnek el az átlagtól. Ezt a feladatot pointer használatával is oldja meg. A tömböt véletlenszámokkal töltse fel.

```
#define N 12
int main() {
   int i;
   double tomb[N], *p, atlag=0.0;

   /* Tömb feltöltése véletlenszámokkal */
  /* lásd előző feladatrész */

   /* Tömbelemek összegzése */
   for (i=0; i<N; i++) {
         atlag += tomb[i];
   }
   /* Átlag kiszámítása */
   atlag /= N;
   printf("\nAz atlag: %.2f\n", atlag);
   /* Tömbelemek átlagtól való eltérésének kiírása */
   for (i=0; i<N; i++)
      printf("%d. \t %.2f \t %.2f\n", i+1, tomb[i], tomb[i]-atlag);
   return 0;
}
```

Tömbelemek átlagtól való eltérésének kiírása mutató használatával

1. verzió:

```
   // A p pointer mindvégig a tömb első elemére mutat; i a ciklusváltozó.
   for (p=tomb, i=0; i<N; i++)
      printf("%d. \t %.2f \t %.2f\n", i+1, *(p+i), p[i]-atlag);
```

2. verzió:

```
   // A p pointer mindig a tömb feldolgozás alatt álló elemére mutat;
   // az i sorszám csak a kiíratáshoz kell.
   for (p=tomb, i=0; p<=&tomb[N-1]; i++, p++)
      printf("%d. \t %.2f \t %.2f\n", i+1, *p, (*p)-atlag);
```

3. feladat:

-1000 és 1000 közé eső véletlenszámokkal töltsön fel egy 20 elemű integer tömböt, majd végezze el az alábbi feladatokat:

a) Válogassa szét a tömb elemeit pozitív és negatív számokra.

```
//* Tömbelemek szétválogatása */
int p_i, n_i;
p_i = n_i = 0;                          // A pozitív, ill. negatív tömb indexe
for (i=0; i<MERET; i+=1) {
   if (tomb[i]>=0) {     poz_tomb[p_i] = tomb[i];
                         p_i++;
   } else {     neg_tomb[n_i] = tomb[i];
                n_i++;
   }
}
```

b) Számítsa ki a résztömbök átlagát. Figyelem! Részhalmaz átlagolásakor a részhalmaz elemszámával kell osztani.

```
/* Résztömbök elemeinek átlaga */
double p_atl = 0, n_atl = 0;
for (i=0; i<p_i; i+=1) {
   p_atl += poz_tomb[i];
}
printf("Pozitív átlag: %lf\n", p_atl/p_i);
for (i=0; i<n_i; i+=1) {
   n_atl += neg_tomb[i];
}
printf("Negatív átlag: %lf\n", n_atl/n_i);
```

c) Eleme-e a pozitív tömbnek az 500? Ha igen, hanyadik eleme? (Eldöntés, kiválasztás)

```
/* Eldöntés, kiválasztás */
int searchkey = 500;
int found = 0;
for (i=0; i<p_i && !found; i++) {
   if(tomb[i]==searchkey) {
      found = 1;
   }
}
// Ha a keresett elem nincs a tömbben, a sorszám érvénytelen
//printf("%d a pozitív tömb eleme: %s, sorszáma: %d", searchkey, found ? "igen" : "nem", i);

if(found)
   printf("%d a pozitív tömb eleme, sorszáma: %d\n", searchkey, i);
else
   printf("%d a pozitív tömbnek nem eleme\n", searchkey);
```

4. feladat:

Megadott billentyű (Esc vagy q) leütéséig olvasson be karaktereket a billentyűzetről. Oldja meg az alábbi feladatokat. Elkerülhető-e a tömb adatszerkezet használata?

a) A leütött karakterek hány százaléka magánhangzó?

b) Írja vissza a képernyőre a karaktereket fordított sorrendben.

c) A leütött karakterek között szerepel-e ‘E’ és utoljára hányadik helyen?

a) A leütött karakterek hány százaléka magánhangzó? Megoldható anélkül, hogy a karaktereket eltárolnánk tömbben.

```
char ch;
int db = 0, mgh_db = 0, i;
char mgh[] = {'a', 'e', 'i', 'o', 'u'};
int meret = sizeof(mgh);                    // a char típus mérete 1 byte

printf("Adj meg karaktereket:\n");
i = 0;
do {
   ch = getchar();
   for(i=0; i<meret; i++) {
```

```
        if( mgh[i] == tolower(ch) ) {          // ctype.h
            mgh_db++;
        }
    }
    db++;
} while(ch != 'q' && ch != 'Q');
printf("A magánhangzók aránya (%d/%d): %.2lf %%", mgh_db, db, ((double)mgh_db/db)*100);
```

b) Írja vissza a képernyőre a karaktereket fordított sorrendben. Az összes karakter eltárolása nélkül nem oldható meg.

```
char ch;
int db = 0, mgh_db = 0, i, j;
char mgh[] = {'a', 'e', 'i', 'o', 'u'};
int meret = sizeof(mgh);
char ch_array[255];                    // elég nagy méretű legyen

printf("Adj meg karaktereket:\n");
i = 0;
do {
    ch = getchar();                    // A getchar() int-et ad vissza, nem char-t !
    ch_array[db] = ch;
    for(i=0; i<meret; i++) {
        if( mgh[i] == tolower(ch_array[db]) ) {
            mgh_db++;
        }
    }
    db++;
} while(ch != 'q' && ch != 'Q');

/* Tömb fordított kiírása */
printf("\nA leültött karakterek fordítva: \n");
for(i=db-1; i>=0; i--)
    printf("%c", ch_array[i]);
printf("\n");
```

c) A leütött karakterek között szerepel-e ‘E’ és <u>utoljára</u> hányadik helyen?

<u>1. megoldás:</u> Tömbös tárolás nélkül is megoldható. Ha nincs E a karakterek között, az output -1.

```
int index=-1;                          // érvénytelen index
for(i=0; i<db; i++) {
    if (toupper(ch_array[i]) == 'E')   // ctype.h
        index = i;
}
printf("E utolsó helye: %d\n", index);
```

2. megoldás: Csak tömbös tárolás esetén. Az utolsó előfordulás tulajdonképpen hátulról az első. Ha nincs E a karakterek között, nem ír ki semmit.

```
for(i=db-1; i>=0; i--) {
    if (toupper(ch_array[i]) == 'E') {
        printf("E utolsó helye: %d\n", i);
        break;
    }
}
```

Házi feladat:

1. A második feladathoz készítsen olyan kiírást, amelyben az első adatot követően a többi listaelem a megelőzőhöz képest számított relatív érték. Például: 301.1, 2.3, -0.4, 0.6, -0.2, stb. Használja fel ezt a listát a monotonitás vizsgálathoz.

2. Az egyetemi hallgatóknak félévente 6 vizsgájuk van. Félév végén az elért tanulmányi átlag alapján ki szeretnénk számolni a hallgató ösztöndíját a következő félévre: 3,5 alatt 0 Ft/hó, 3,6-4,0 között 5eFt/hó, 4,1-4,5 között 10eFt/hó és 4,6-5,0 között 15eFt/hó. Írjon C programot, amely beolvassa egy hallgató vizsgajegyeit és eltárolja egy tömbben, kiszámítja a tanulmányi átlagot és kiírja az ösztöndíj havi összegét.

3. Implementálja a “Gondoltam egy számot” játék algoritmusát.

CIKLUSBAN
1. A számítógép sorsol egy 1-100 közötti számot.
CIKLUSBAN
2. A felhasználó mond egy tippet.
3. A program kiírja, hogy a gondolt szám egyenlő-e, ill. kisebb vagy nagyobb-e mint a felhasználó tippje.
AMÍG a felhasználó tippje != a gondolt számmal
4. A felhasználó megmondja, akarja-e folytatni a játékot.
AMÍG a felhasználó akarja folytatni a játékot

4. Számrendszerek közötti konverzió. Írjon C programot, amely kiírja egy tizes számrendszerben megadott szám kettes számrendszerbeli megfelelőjét. A feladatot a bitléptető operátor használatával is oldja meg, ill. úgy is hogy 2 hatványaival osztja a számot. Oldja meg a konverziót fordítva is. A kettes számrendszerbeli számot tömbben tárolja.

5. Írjon C programot, amely a felhasználó által megadott karaktersorozatról eldönti, hogy érvényes C azonosító-e. Az input max. 10 hosszú legyen.

6. Írjon C programot, amely az angol ABC betűiből, számokból és az aláhúzás karakterből véletlenszerűen előállít egy 6 karakter hosszú érvényes C azonosítót.