Programozás alapjai
9. gyakorlat
Tömb, mint függvény argumentum

1. feladat:
Az 5. gyakorlat 2. feladatát oldja meg a procedurális programtervezési alapelv betartásával, azaz minden végrehajtandó funkciót külön függvényben valósítson meg. Emlékeztető: A HUF/EUR árfolyamot kell nyilvántartani egy negyedéven keresztül hetente. Megvalósítandó függvények:

- A tömb elemeinek beolvasása ellenőrzött módon.
- Tömb elemeinek kiírása.
- Tömb feltöltése véletlenszámokkal.
- Megadott értéknél alacsonyabb árfolyamok megszámolása.
- Annak megállapítása, hogy a tömb elemei monoton növő sorozatot alkotnak-e? Eldöntendő kérdés.
- A tömbelemek átlagának kiszámítása.
- A tömbelemek, és azok átlagtól való eltérésének kiírása.
- A tömb legkisebb elemének meghatározása.
- A tömb legnagyobb elemének meghatározása.

a) A tömb elemeinek beolvasása ellenőrzött módon és kiírása.

```
#include <stdio.h>
/* Saját függvények prototípusa */
void beolvas(float *tomb, int meret);
void kiir(float *tomb, int meret);

int main() {
   const int n = 12;
   float tomb[12];
   beolvas(tomb, n);
   kiir(tomb, n);
   return 0;
}

void beolvas(float *tomb, int meret) {
   int i, ok;
   char ch;
   for(i=0; i<meret; i++) {
      do {
         ok = 1;
         printf("%d. érték: ", i+1);
         if (scanf("%f", &tomb[i])!=1) {
                  printf("Hibás input\n");
                  ok = 0;
         }
         while ((ch=getchar()) != '\n');
      } while ( !ok );
   }
   return ;
}
```

```
void kiir(float *tomb, int meret) {
    int i;
   for(i=0; i<meret; i++) {
         printf("%d. érték: %.2f \n", i+1, tomb[i]);
   }
   return ;
}
```

b) Tömb feltöltése véletlenszámokkal és megadott értéknél kisebb elemek megszámolása.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* Saját függvények prototípusa */
void float_feltolt(float *tomb, int meret);
int szamlal(float *tomb, int meret, float limit);

int main() {
   const int n = 12;
   int db;
   float tomb[12], limit = 300.0;
   float_feltolt(tomb, n);
   db = szamlal(tomb, n, limit);
   printf("Az árfolyam értéke %d-szer volt %.2f alatt.", db, limit);
   return 0;
}

void float_feltolt(float *tomb, int meret) {
   int i;
   srand(time(0));
   int upper = 280, lower = 311;
   double range = upper - lower;
   double div = RAND_MAX / range;
   double value;
   for (i=0; i<meret; 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
   }
   return ;
}

int szamlal(float *tomb, int meret, float limit) {
   int i, db=0;
   for (i=0; i<meret; i+=1) {
         if (tomb[i] < limit)  db++;
   }
   return db;
}
```

c) Annak megállapítása, hogy a tömb elemei monoton növő sorozatot alkotnak-e.

```
void beolvas(float *tomb, int meret);
int monoton_e(float *tomb, int meret);          //1-t (igaz) vagy 0-t (hamis) ad visssza
int main() {
   int i,  n = 12;
   float tomb[12];
   beolvas(tomb, n);
   //Monotonitás vizsgálata
   if ( monoton_e(tomb, n) ) printf("Az Euro árfolyama monoton nőtt.");
   else printf("A számsorozat nem monoton növő.");
   return 0;
}

int monoton_e(float *tomb, int meret) {
   int i;
   for (i=1; i<meret; i++)
      if (tomb[i-1]>tomb[i]) return 0;
   return 1;
}
```

d) A tömbelemek átlagának kiszámítása és az átlagtól való eltérés kiírása.

```
void beolvas(float *tomb, int meret);
float atlagol(float tomb[], int meret);
void eltereskiir(float tomb[], int meret, float atlag);

int main() {
   int i, n = 12;
   float tomb[12];
   beolvas(tomb, n);
   float atlag = atlagol(tomb, n);
   printf("\nÁtlag: %.2f\n", atlag);
   printf("Atlagtol valo elteresek:\n");
   eltereskiir(tomb, n, atlag);
   return 0;
}
float atlagol(float tomb[], int meret) {
   float eredmeny, sum=0.0;
   int i;
   for(i=0; i<meret; i++)
      sum += tomb[i];
   eredmeny = sum/meret;
   return eredmeny;
}
void eltereskiir(float tomb[], int meret, float atlag) {
   int i;
   for (i=0; i<meret; i++)
      printf("%d. \t %.2f HUF/EUR \t %.2f\n", i+1, tomb[i], tomb[i]-atlag);
   printf("\n");
   return ;
}
```

e) A tömb legkisebb elemének meghatározása. A legnagyobb elem meghatározása önálló feladat.

```
/* Saját függvények prototípusa */
void beolvas(float *tomb, int meret);
int minindex(float *tomb, int meret);

int main() {
   int i, N = 12;
   float tomb[12];
   beolvas(tomb, N);
   int min = minindex(tomb, N);
   printf("\nLegkisebb elem: %.2f", tomb[min]);
   return 0;
}

int minindex(float *tomb, int meret) {
   int i, mini = 0;                    //mini: a legkisebb tömbelem indexe
   for(i=0; i<meret; i++) {
      if(tomb[i]<tomb[mini])
         mini=i;
   }
   return mini;
}
```

Önálló feladatok:
1. A feladathoz készítsen lineáris keresést megvalósító függvényt.
2. A feladathoz készítse el a minimum kiválasztásos rendezés algoritmusát megvalósító függvényt.
3. Írjon függvényt, amely visszaadja a tömbben a legnagyobb páros szám indexét. Feltétellel bővített maximum kereső eljárás.

2. feladat: Sztringkezelő függvények írása
Inicializáljon, illetve olvasson be a szabványos bemenetről egy szöveget. Ezen a szövegen az alábbi műveleteket hajtsa végre.
a) A szöveg megfordítása, illetve fordított kiírása.
b) A szöveg nagybetűssé konvertálása.
c) A szövegben az E betűk megszámolása.
A feladatot a procedurális programtervezési alapelv betartásával oldja meg, azaz minden végrehajtandó funkciót külön függvényben valósítson meg.

Szöveg beolvasásának lehetőségei:
1. A *scanf("%s", szoveg);* utasítással beolvasott szöveg nem tartalmaz *whitespace* karaktereket.
2. A *scanf(“%[^\n]”, s);* utasítás Enterig minden karaktert beolvas.
3. A *ch=getchar();* utasítást hátultesztelő cikusban kiadva, a vezérlő feltételben megadott karakterig minden karaktert beolvas.

a) A szöveg megfordítása.

```
#define ESC 27                  // Esc billentyű ASCII kódja Unix op.rendszerben
#define N 100
void beolvas(char *szoveg);
char* forditvamasol(const char *forras, char *cel);
```

```
int main() {
   char szoveg[N], forditott[N];
   beolvas(szoveg);
   printf("A beolvasott szöveg:\n%s", szoveg);
   char *p;
   p = forditvamasol(szoveg, forditott);       //'p' a fordított karaktertömb elejére mutat
   printf("\nA szöveg megfordítva:\n%s", p);
   return 0;
}

void beolvas(char *szoveg) {
   int i=0;
   char ch;
   printf("Esc+Enter lenyomasaig olvassa a karaktereket:\n");
   while ((ch=getchar()) != ESC) {     // lehet benne szóköz
      szoveg[i] = ch;
      i++;
   }
   szoveg[i] = '\0';
   return ;
}

char* forditvamasol(const char *forras, char *cel) {
   char *seged = cel;
   int i = strlen(forras)-1;
   while (i>=0) {
      *cel = forras[i];
      cel++;
      i--;
   }
   *cel = '\0';
   return seged;
}
```

Szöveg megfordítása helyben:

```
char* megfordit(char *szoveg) {
        char *p = szoveg;
        char seged;
        int i, j;
        for(i=0, j=strlen(szoveg)-1; i<strlen(szoveg)/2; i++, j--) {
                seged = szoveg[i];
                szoveg[i] = szoveg[j];
                szoveg[j] = seged;
        }
        return p;
}
```

Szöveg fordított kiírása:

```
void forditvakiir(char *szoveg) {
        int i;
        for(i=strlen(szoveg)-1; i>=0; i--)
                printf("%c", szoveg[i]);
        printf("\n");
        return ;
}
```

b) A szöveg nagybetűssé konvertálása.

```
char* nagybetusenmasol(const char *forras, char *cel) {
   char *seged = cel;
   while (*forras) {
      *cel = toupper(*forras);          //ctype.h függvénye
      cel++;
     forras++;
   }
   *cel = '\0';
   return seged;
}
```

Hívása:

```
char *p;
p = nagybetusenmasol(szoveg, nagybetus);
printf("\nA szöveg nagybetűsen:\n%s", p);
```

Szöveg nagybetűssé konvertálása helyben:

```
char* nagybetusit(char *szoveg) {
   char *seged = szoveg;
   while(*szoveg) {
      *szoveg=toupper(*szoveg);
      szoveg++;
   }
   *szoveg = '\0';
   return seged;
}
```

Hívása:

```
char *p;
p = nagybetusit(szoveg);
printf("\nA szöveg nagybetűsen:\n%s", p);
```

Szöveg nagybetűs kiírása:

```
void nagybetuvelkiir(char *szoveg) {
        int i;
        for(i=0; i<strlen(szoveg); i++)   printf("%c", szoveg[i]);
        printf("\n");
}
```

c) A szövegben az E betűk megszámolása.

```
int Ebetuk(char *szoveg) {
   int db=0;
   while(*szoveg) {
      if (*szoveg=='E') db++;
      szoveg++;
   }
   return db;
}
```

Hívása:

```
int db = ebetuk(szoveg);
printf("\nA szövegben %d db 'E' betű van.", db);
```

Önálló feladatok:

1. Írjon függvényt, amely visszaadja, hogy a szöveg karaktereinek hány százaléka magánhangzó.
2. Lineáris kereső eljárással keressen meg a szövegben egy megadott karaktert.
3. Írjon függvényt, amely a szöveg karaktereit véletlenszerűen összekeveri (segédtömb felhasználása nélkül). Keverés algoritmus (shuffling) – 9. előadás.

Házi feladatok:

1. Deklaráljon egy 10 elemű integer tömböt. Írjon függvényt, amely feltölti a tömb elemeit 1 és 5 közé eső véletlenszámokkal.

a) Írjon függvényt, amely visszaadja a sorozat móduszát (a legtöbbször előforduló elemét).

- Hogyan kezelné azt az esetet, amikor többmóduszú a sorozat? Például az 1, 1, 1, 2, 3, 4, 4, 5, 5, 5 sorozatban a leggyakrabban előforduló elemek az 1 és az 5, mindkettőnek azonos az előfordulási gyakorisága.
- Vannak olyan sorozatok, ahol minden elem azonos gyakorisággal fordul elő (pl. egyszer, ha nincs az elemek között ismétlődés). Ekkor azt mondjuk, hogy a sorozat móduszát nem lehet meghatározni. Ezt az esetet is kezelje a program.

b) Írjon függvényt, amely kiírja a rendezett sorozatot.

c) Írjon függvényt, amely kiírja a sorozat statisztikáját táblázatos formában: melyik elem hányszor fordul elő. Például ha a sorozat elemei 1, 1, 1, 2, 3, 4, 4, 5, 5, 5, a statisztika így néz ki:

1 - 3 db
2 - 1 db
3 - 1 db
4 - 2 db
5 - 3 db

2. Olvasson be egy szöveget. Írjon függvényt a szöveg kódolására (titkosítására), majd dekódolására. A feladatnak a szöveg nagybetűssé konvertálásához hasonlóan 3 lehetséges megközelítése van.

3. Implementálja a 6. előadás sztringkezelő algoritmusait saját függvényként. Minden esetben próbálja ki a megfelelő standard függvényhívást is (sztring hosszának meghatározása, sztring másolás, sztring hozzáfűzés, karakter keresése sztringben, rész-sztring keresés).