Programozás alapjai
6. gyakorlat
Keresés, sztringkezelés

1. feladat: Karaktertömb kezelése
Deklaráljon és inicializáljon egy tetszőleges méretű és tartalmú karaktertömböt. Egy beolvasott karaktert lineáris kereső eljárással keressen meg ebben a tömbben.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define TRUE 1
#define FALSE 0

int main()
{
   char abc[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
                 'w', 'x', 'y', 'z'};        //nincs lezáró nulla a végén!
   int i, found = FALSE;
   char key;

   printf("Give the character you're searching for: ");
   scanf(" %c", &key);                  //VAGY   key = getchar();

   if (isalpha(key)) {
      for(i=0; i<sizeof(abc) && !found; i++) {
         if (abc[i] == key || abc[i] == tolower(key)) {
            found = TRUE;
         }
      }
      printf("This character is the %d. in the ABC.\n", i);
   }
   else
      printf("This character is not an ABC letter.\n");

   return 0;
}
```

2. feladat: Sztringkezelés
Inicializáljon, illetve Esc+Enter lenyomásáig 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.

Figyelem! Ha a szöveg beolvasását a *scanf("%s", szoveg);* utasítással valósítja meg, az nem tartalmazhat *space* karaktert.
Space-eket is tartalmazó szöveg beolvasása: ***scanf("%[^\n]", szoveg);***

Ahol lehet, próbálják ki a standard sztringkezelő függvények használatát!

a) A szöveg megfordítása.

```
#include <stdio.h>
#include <string.h>                 // sztringkezelő fv-ek
#include <ctype.h>                  // karakterkonverziós fv-ek
#define N 100
#define ESC 27                      // Esc billentyű ASCII kódja

int main() {
   char szoveg[N], forditott[N], nagy[N];
   int i, j;

   //szöveg beolvasása
   char ch;
   printf("Esc+Enter lenyomasaig olvassa a karaktereket:\n");
   i = 0;
   while ( (ch=getchar()) != ESC && i<N-1) {     // lehet benne szóköz
      szoveg[i] = ch;
      i++;
   }
   szoveg[i] = '\0';      // a sztring végét jelző karakter

   //szöveg kiírása
   printf("A beolvasott szöveg:\n%s\n", szoveg);      // nem kell a címképző operátor
   //fordított kiírás
   for(i=strlen(szoveg)-1; i>=0; i--) {
      printf("%c", szoveg[i]);
   }

   //szöveg megfordítása (fordítva másolás, pointeres megoldás)
   char *fp = forditott;
   i = strlen(szoveg)-1;
   while (i>=0) {
      *fp = szoveg[i];
      fp++;
      i--;
   }
   *fp= '\0';
   printf("\nA szöveg megfordítva:\n%s", forditott);

   //sztring megfordítása helyben (karakterek felcserélése)
   char seged;
   for(i=0, j=strlen(szoveg)-1; i<strlen(szoveg)/2; i++, j--) {
      seged = szoveg[i];
      szoveg[i] = szoveg[j];
      szoveg[j] = seged;
   }
   printf("%s\n", szoveg);

   return 0;
}
```

```
//Nem pointeres megoldás
for(i=strlen(szoveg)-1, j=0; i>=0; i--, j++) {
      forditott[j] = str[i];
}
forditott[j] = '\0';
```

b) A szöveg nagybetűssé konvertálása.
Az egyik megoldás az előzőhöz hasonló:

```
//nagybetűsen másol új tömbbe
   char *nf = nagy;
   i=0;
   while (i<strlen(szoveg)) {
      *nf = toupper(szoveg[i]);
      nf++;
      i++;
   }
   *nf = '\0';
   printf("\nA szöveg nagybetűsen:\n%s", nagy);
```

A feladat másik megoldása, ha magát a szöveget konvertálja nagybetűssé:

```
//eredetit nagybetűsít
   printf("Eredeti szöveg:\n%s", szoveg);
   char *sp = szoveg;

   while(*sp) {
      *sp=toupper(*sp);
      sp++;
   }
   *sp = '\0';
   printf("\nA szöveg nagybetűsen:\n%s", szoveg);
```

c) A szövegben az E betűk megszámolása.

```
//'E' betűk megszámolása
   int db = 0;
   sp = szoveg;
   i = 0;
   while(*sp) {
      if (*sp=='E') db++;
      sp++;
      i++;
   }
   printf("\nA szövegben az 'E' betűk száma: %d\n", db);
   printf("Ez %.2f%%-a a szöveg karaktereinek.\n", ((float)db/i)*100);
```

3. feladat:
Vegyük elő a 3. gyakorlat 6. feladatát.
Készítsen a négy alapművelet elvégzését megvalósító kalkulátort. Adjon meg két számot, és végezze el rajtuk a megadott aritmetikai műveletet. Az eredményt irassa ki a képernyőre. Az input alakja: “%d %c %d”. Most az inputot egyetlen sztringként olvassa be és az *sscanf()* függvény segítségével dolgozza fel.

```
char muvelet[10];
int a, b; char op;
printf("Elvégzendő művelet: ");
```

```
  // Előzőleg:  scanf("%d %c %d", &a, &op, &b);
 // Most egyetlen sztring az input, amit később bontunk fel elemeire
   scanf("%s", muvelet);
   if(sscanf(muvelet,"%d %c %d", &a, &op, &b) == 3) {
      // helyes működés kódja
   }
   else {
      printf("Hibás input\n");
   }
```

Ezzel a módszerrel megoldható, hogy az input buffer teljes tartalmát feldolgozzuk a programból.

4. feladat: Sztringek tömbje. Akasztófa játék

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>

#define  MERET   5

int main() {
    char betu;
    int talalat=0, i, kerdes=0;

    //mutatótömb, mérete nem számítható ki a sizeof operátorral
   char *szavak[MERET] = {"kertmozi", "balettruha", "zuhanykabin", "hangyaboly",
                                  "sajttorta"};
    srand(time(0));
    char *gondolt = szavak[rand()%MERET];         //az egyik szó kiválasztása

    char kitalalt[strlen(gondolt)+1];
    //a kitalalt tömb inicializálása helypótló karakterekkel, pointeres megoldás
    char *init = kitalalt;
    char *gp = gondolt;
    while (*gp) { *init = '_'; init++; gp++; }
    *init = '\0';

    /* //tömbös megoldás
       for (i=0; i<strlen(gondolt); i++) { kitalalt[i] = '_'; }
       kitalalt[i] = '\0';
   */
    printf("Felírtam egy szót. Találd ki! Kérdezz betűket!\n");
    printf("A gondolt szó: %s\n", kitalalt);     // annyi '_' ahány karakterből áll a gondolt szó

   do {                                  // amíg ki nem találjuk a szót, kéri a karaktereket
       printf("Van benne? ");
       scanf(" %c", &betu);
       kerdes++;                         // számoljuk a felhasználó tippjeit
```

```
      for (i=0; i<strlen(gondolt); i++) {      // a megadott karaktert keresi a gondolt szóban
         if (gondolt[i]==tolower(betu)) {
            kitalalt[i]=tolower(betu);
            talalat++;                        // számoljuk hányszor fordul elő a karakter a szóban
         }
      }
      printf("%s\n", kitalalt);               // a játék aktuális állása
   } while (strlen(gondolt)!=talalat);
   printf("Gratulálok, %d kérdésből kitaláltad!", kerdes);

   return 0;
}
```

Házi feladat:

1. Inicializáljon egy sztringet, egy másikat pedig olvasson be a scanf() megengedett karaktereket felsoroló változatával. Próbálja ki a string.h standard függvényeit (sztring hosszának meghatározása, sztring másolás, sztring hozzáfűzés, karakter keresése sztringben, rész-sztring keresés, két sztring összehasonlítása).
2. Olvasson be egy sztringet, majd írja ki virágnyelven; azaz minden magánhangzót ‘v’ előtaggal megismételve. Például: “alma” → “a**va**lma**va**”. A beolvasáshoz használja a scanf() függvényt.
3. Olvasson be egy sztringet, és állapítsa meg, hogy palindróma-e; azaz visszafelé olvasva ugyanazt a szót kapjuk-e. Próbálja meg ugyanezt a feladatot space-t is tartalmazó szövegre is megoldani. Palindrómára példa: “Géza kék az ég”.
4. Az akasztófa játékot (4. feladat) tegye folyamatos működésűvé.
5. Szókitaláló játék.
Inicializáljon egy sztringtömböt. Ebből véletlenszerűen kiválasztva egyet, keverje össze a betűit és mutassa meg a felhasználónak. A feladat kitalálni az összekevert betűkből az eredeti szót. A felhasználó mondjon tippeket, a program válaszoljon: hanyadik karakter helyét találta el (a Mesterlogika játék mintájára). A játék végén írja ki a helyes szót és azt, hogy hány tipp után találta ki a felhasználó.
6. Olvasson be egy mondatot (az utolsó karakter ‘.’, ‘?’, vagy ‘!’). Keresse meg, hogy a mondatban:
a) a mondatkezdő karakteren kívül van-e nagybetű és az hanyadik,
b) van-e speciális karakter vagy szám, és az hanyadik.
7. Töltsön fel egy 10 elemű tömböt 10 és 100 közé eső véletlenszámokkal. Keresse meg ebben a tömbben:
a) a legkisebb prímszámot,
b) a legnagyobb négyzetszámot.
8. Keresse meg az Interneten a férfi teniszezők világranglistáját (csökkenően rendezett sorozat). Tárolja el a pontszámokat tömbben és végezzen kereséseket. Implementálja a lineáris keresés és a logaritmikus keresés algoritmusait. Próbálja ki a standard C *bsearch()* függvényt.

```
 // Összehasonlító függvény növekvő int sorozat esetén
int cmpfunc(const void * a, const void * b) {
   return ( *(int*)a - *(int*)b );
}
// bsearch() használata int sorozat esetén
 int* item = (int*)bsearch(&searchkey, array, size, sizeof(int), cmpfunc);
    if( item != NULL )
       printf("Keresett elem: %d, sorszáma: %d\n", *item, (item-array)+1);
    else
       printf("Nem talált\n");
```