**Programozás alapjai**
**6. gyakorlat**
**Keresés, tömbkezelés**

**1. feladat: Keresés karaktertömbben**
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: Keresés rendezetlen numerikus tömbben**
Töltsünk fel egy 10 elemű tömböt 10 és 100 közé eső véletlenszámokkal. Keressük meg ebben a tömbben:
a) a legkisebb prímszámot,
b) a legnagyobb négyzetszámot.
Figyelem! Lehet hogy nincs a tömbben a feltételnek megfelelő elem.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
#include <math.h>
```

```
int main()
{
   int array[10];
   int i, size = sizeof(array)/sizeof(int);
   int divider, counter, minindex;
   bool found;

   srand(time(0));

   // tömb feltöltése 10 és 100 közé eső számokkal
   for (i=0; i<size; i++) {
      array[i] = rand()%91+10;
      printf("%d. szám: %d\n", i+1, array[i]);
   }

   // legkisebb prímszám keresése
   counter = 0;
   for (i=0; i<size; i++) {
      found = false;
      divider = 2;
      while (divider < sqrt(array[i]) && !found) {
         if (array[i]%divider==0)
            found = true;
         divider++;
      }
      if (!found) {   // ha prímszám
         printf("Prímszám: %d\n", array[i]);
         counter++;
         if (counter==1)                  // az első prímszám esetén
            minindex = i;
         else if (array[i] < array[minindex])  // nem az első prímszám esetén
            minindex = i;
      }
   }
  if (counter>0)
      printf("A legkisebb prímszám: %d\n", array[minindex]);
   else
      printf("Nincs prímszám a listában\n");
```

A b) rész ennek mintájára önálló feladat.

**3. feladat: Keresés rendezett numerikus tömbben**

Adott a férfi teniszezők világranglistája (csökkenően rendezett sorozat). Tárolja el a pontszámokat tömbben és implementálja a tanult kereső eljárásokat.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

```
/*  A bsearch() függvény működéséhez kell ez az összehasonlító függvény.
    Növekvő sorozat esetén a return utasításban fel kell cserélni
    a 'b' és 'a' változók megadásának sorrendjét. */
int cmpfunc(const void * a, const void * b) {
   return ( *(int*)b - *(int*)a );
}

int main()
{
    int array[] = {11260, 9850, 9125, 6630, 5890, 5385, 4650, 3030, 2860, 2665};
    int i, size = sizeof(array)/sizeof(int);
    bool found;
    int first, last, middle;

    int search = 4650;

    // lineáris keresés ciklusmegszakítással
    found = false;
    for (i=0; i<size && array[i]>=search; i++) {
        if (array[i]==search) {
            printf("A %d. versenyző pontszáma %d\n", i+1, array[i]);
            found = true;
            break;
        }
    }
    if (i==size || !found)   printf("Nem talált\n");

    // bináris keresés csökkenő sorozatban
    first = 0;
    last = size - 1;
    middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)    last = middle - 1;
        else if (array[middle] == search) {
            printf("A %d. versenyző pontszáma %d\n", middle+1, search);
            break;
        }   else  first = middle + 1;

        middle = (first + last)/2;
    }
    if (first > last)  printf("Nem talált\n");

    // bsearch() használata
    int* item = (int*)bsearch(&search, array, size, sizeof(int), cmpfunc);
    if( item != NULL )
        printf("A versenyző sorszáma: %d, pontszáma %d\n", (item-array)+1, *item);
    else
        printf("Nem talált\n");

    return 0;
}
```

Módosítsa úgy a programot, hogy annak a versenyzőnek a listabeli sorszámát keressük, akinek a pontszáma abszolút értékben legközelebb van az 5000-hez.

**Házi feladat:**

1. 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.

2. Adottak egy egyetemi hallgató eredményei 6 félévre vonatkozóan (féléves tanulmányi átlagok) időrendben. Keresse meg, hogy melyik félévben volt a hallgató tanulmányi eredménye 4,5 fölött. Figyelem! Azt az esetet is kezelje, ha nem volt ilyen félév; és azt is ha több ilyen félév volt.

3. Keresse meg az Interneten a Forma1 világranglistát (csökkenően rendezett sorozat). Tárolja el a pontszámokat tömbben, növekvő sorrendben és végezzen kereséseket. Implementálja a lineáris keresés és a logaritmikus keresés algoritmusait. Próbálja ki a standard C *bsearch()* függvényt. Hányadik a listában az a versenyző, akinek a pontszáma éppen fele annyi, mint a legmagasabb pontszám. Azt az esetet is kezelje, amikor nincs olyan pontszám, ami megfelel a feltételnek.

4. 1990 és 2020 között hányadik év volt az első szökőév.

5. Tekintsük a pitagoraszi számhármasokat 1-től 100-ig. Ezek azok a pozitív egész számokból álló számhármasok, amelyek lehetnek egy háromszög oldalai, azaz teljesül rájuk a háromszög egyenlőtlenség: $a < b < c$ és $c \leq 100$. Melyik az a számhármas, amelyikre igaz, hogy az a, b és c közötti különbség rendre 10. Megoldás: 52 db ilyen számhármas van és a 20. a keresett eset.