Keresés tömbben:

1. Hozzunk létre egy 100 elemű tömböt, majd töltsük fel 1-1000 közötti véletlen számokkal.
2. Kérjünk be egy 1-1000 közötti egész számot, majd keressük meg és írjuk ki, hogy hányadik helyen szerepel a tömbben (első előfordulását keressük).

Lineáris keresés megértéséhez animáció:

- http://liveexample.pearsoncmg.com/liang/animation/web/LinearSearch.html

Bináris keresés tömbben:

1. Hozzunk létre egy 100 elemű tömböt, majd töltsük fel 1-1000 közötti véletlen számokkal.
2. Rendezzük a tömböt tetszőleges rendezési algoritmussal.
3. Kérjünk be egy 1-1000 közötti egész számot, majd BINÁRIS KERESÉSSEL keressük meg és írjuk ki, hogy hányadik helyen szerepel a tömbben.

Bináris keresés megértéséhez segédanyag és animáció:

- https://wiki.prog.hu/wiki/Logaritmikus_keres%C3%A9s_(algoritmus)
- https://liveexample.pearsoncmg.com/liang/animation/web/BinarySearch.html
- https://yongdanielliang.github.io/animation/web/BinarySearchNew.html

Keresés mátrixban:

1. Hozzunk létre egy 10x10-es mátrixot és töltsük fel 1-1000 közötti véletlen számokkal.
2. Kérjünk be egy 1-1000 közötti egész számot, majd keressük meg és írjuk ki, hogy melyik sorban és oszlopban szerepel a mátrixban.

Mintapélda - mátrix használata:

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

main() {

	// matrix letrehozasa - a matrix 3 sorbol es 5 oszlopbol all
	int mat[3][5], i, j;

	// matrix generalasa
	srand(time(NULL));
	for (i = 0; i < 3; i++) {
		for (j = 0; j < 5; j++) {
			// 1..250 kozotti veletlen szamok
			mat[i][j] = rand() % 250 + 1;
		}
	}

	// matrix kiirasa
	srand(time(NULL));
	for (i = 0; i < 3; i++) {
		for (j = 0; j < 5; j++) {
			// egy elem kiirasa 4 helyre
			printf("%4d", mat[i][j]);
		}
		printf("\n");
	}

}
```