2. Gyakorlat

Áttekintő:

- Tudnivalók ismertetése (szabályok, osztályozás)
- Header és c fájl struktúra
- Egyszerű feladatok implementálása
- Algoritmusok bonyolultsága

Kitűzött feladatok:

1. Feladat:

Futtasd az előző labor feladatait CMD-ből (CLion terminal).

2. Feladat:

Vizsgáld a **lineáris keresés** bonyolultságát. Írd ki a képernyőre, hogy mennyi a **végrehajtási idő** (mp-ben), illetve a **műveletek száma**. Teszteld n = 10, 100, 1000 ….. 1000000000 elemű sorozatra is. A sorozat elemei és a keresett érték is legyen véletlenszerűen generálva. Alkalmazd a **.h** és **.c** fájl struktúrát.

Segítség:

```
linear_search (list, value)

   for each item in the list
      if match item == value
         return the item's location
```

3. Feladat:

Vizsgáld a **bináris keresés** (rekurzív változat) bonyolultságát. Írd ki a képernyőre, hogy mennyi a **végrehajtási idő** (mp-ben), illetve a **műveletek száma**. Teszteld n = 10, 100, 1000 ….. 1000000000 elemű sorozatra is. A sorozat elemei és a keresett érték is legyen véletlenszerűen generálva. Biztosítsd a sorozat elemeinek rendezettségét mielőtt alkalmaznád a bináris keresést. Használj **qsort**-ot vagy más általad választott **rendező algoritmust**. Alkalmazd a **.h** és **.c** fájl struktúrát.

2. GYAKORLAT

Segítség:

```
binarySearch(arr, x, low, high)
    if low > high
        return False
    else
        mid = (low + high) / 2
        if x == arr[mid]
            return mid
        else if x < data[mid]
            //x is on the right side
           return binarySearch(arr, x, mid + 1, high)
        else
            //x is on the right side
           return binarySearch(arr, x, low, mid - 1)
```

4. Feladat:

Hasonlítsd össze a **lineáris-** és **bináris** keresés bonyolultságát (mindkettőt ugyanazokon a számsorozatokon teszteld n = 10, 100, 1000 ….. 1000000000 esetén). Mit veszel észre?

Gyakorló feladatok:

A következő feladatok mindegyikéhez alkalmazd a .c és .h struktúrát. A feladatokat írhatod egy projekt-be, de rendezd a megoldásod külön függvényekbe. Teszteld a függvényeket a Main.c main függvényében.

1. Feladat:

Írj egy eljárást, mely paraméterként megkapja az éjfél óta eltelt másodpercek számát, és kiírja az időpontot óra:perc:másodperc formában!

2. Feladat:

Írj egy függvényt, mely paraméterként megkapja két pont (X,Y) koordinátáit és visszaadja a két pont távolságát!

3. Feladat:

Írj egy programot, mely egy $ax2 + bx + c = 0$ alakú másodfokú egyenletnek meghatározza a megoldását az a, b és c tényezők ismeretében!

2. Gyakorlat

4. Feladat:

Határozd meg a következő algoritmus komplexitását:

```
for( int i = n; i > 0; i /= 2 )
{
      for( int j = 1; j < n; j *= 2 )
      {
            for( int k = 0; k < n; k += 2 )
            {
                  ... // constant number of operations
            }
      }
}
```

5. Feladat:

Határozd meg a következő algoritmus komplexitását:

```
for( int i = n; i > 0; i-- )
{
       for( int j = 1; j < n; j *= 2 )
            {
                 for( int k = 0; k < j; k++ )
                 {
                 ... // constant number of operations
                 }
       }
}
```

6. Feladat:

Találjunk és adjunk vissza egy „csúcsot” egy számsorozatból, ha létezik! Ellenőrizd a szélsőséges eseteket (pl. nincs eredmény, üres a tömb stb.), működjön a program bármilyen bemenetre. Törekedj többféle megoldásra, és ellenőrizd melyik hatékonyabb (pl. iteratív megoldás vs. rekurzív megoldás).

2. Gyakorlat

Extra feladatok:

1. Extra:

Az előző alpontokban megírt lineáris- és bináris keresés esetén teszteld az algoritmusok bonyolultságát és hasonlítsd össze őket úgy, hogy minden n – re keresd a sorozat legelső, középső, legutolsó, valamely belső elemét, illetve egy olyan értéket, ami nincs benne. Mit tapasztalsz? Vezesd táblázatba az algoritmusok bonyolultságára vonatkozó észrevételeid. Töltsd fel a kódod, amivel tesztelted GitHub-ra.

Segítség a táblázathoz:

n	Első elem				Középső elem				Utolsó elem				Belső elem				Hiányzó elem			
	Idő		Műveletek száma		Idő		Műveletek száma		Idő		Műveletek száma		Idő		Műveletek száma		Idő		Műveletek száma	
	lin	bin	lin	bin	lin	bin	lin	bin	lin	bin	lin	bin	lin	bin	lin	bin	lin	bin	lin	bin
10																				
100																				
1000																				
10000																				
100000																				
1000000																				
10000000																				
100000000																				
1000000000																				

Hasznos linkek:

- https://randerson112358.medium.com/how-to-run-c-program-in-command-prompt-e435186cd162
- https://www.programiz.com/dsa/algorithm
- https://www.geeksforgeeks.org/linear-search/
- https://www.geeksforgeeks.org/binary-search/