IMSC: POINTEREK ALKALMAZÁSAI

Általános információk

Az 1-5. feladatokból 4 teljesítésével teljesítésével 1 IMSC pont szerezhető.

Feladatok

A feladatok a mellékelt Qsort solutionre vonatkoznak.

1. Pointer, mint ismeretlen törzsű függvény hívásának eszköze

Helyezzünk el egy töréspontot a *compare* függvénybe, és nézzük a kapott **da* ill. **db* változók értékeit.

- Tényleg úgy működik, ahogy vártuk?
- Miért kell függvénypointert használnunk?
- Hogyan állítunk elő egy adott függvényre mutató pointert?
- Miért két *void** típusú paramétert kap az összehasonlító függvény?

2. Pointer, mint dinamikus memóriakezelés eszköze

Írd át a programot úgy, hogy a *felhasználótól kérje be* a tömb méretét! A feladat megoldásához használj *dinamikus memóriakezelés*t! Ne felejtsd fel *felszabadítani* a lefoglalt területet!

3. Pointer, mint cím szerinti paraméterátadás eszköze

- Az előző órai feladatot felidézve, miért vár pointert a *scanf* függvény?
- Valósítsd meg a tömbelemek bekérését (scanf) háromféleképpen!
    - Tipp
        - hogyan kell egy változó címét képezni?
        - mire fordítja át a compiler a d[i]-t?

4. Biztonságos szövegbevitel

Oldd meg, hogy a program védve legyen buffer túlcsordulás támadással (buffer overflow attack) szemben!

- Tipp: a megoldáshoz használd a scanf_s(…) függvényt

5. Saját rendezőfüggvény

- Készíts saját rendezőfüggvényt (például buborékrendezést) az előző feladatokban is használt, beépített qsort() interfészét (paraméterlistáját) felhasználva.
- Írd át a programot úgy, hogy qsort() helyett a Te függvényedet használja a program
    - predikátumként használd fel a már megadott compare() függvényt

Tipp: a qsort paraméterlistája a következő:

```
void qsort(void *base, size_t nmemb, size_t size, int(*compar)(const void *, const void *));
```

base: a rendezendő tömb címére mutató pointer
nmemb: tömb elemeinek száma
size: a tömb egy elemének memóriában foglalt mérete (byte-okban)
compar: függvénypointer a predikátumra