Coding Tomorrow Cup Döntős Feladat

Rendezés

Tartalomjegyzék

1. FELADAT KIÍRÁS 3
1.1 Implementáció 3
1.2 Fejlesztő és teszt környezet 3
1.2.1 Teszt környezet elemei 4
2. A FELADAT PONTOZÁSA 4

1. Feladat kiírás

Ebben a feladatban egy olyan algoritmust kell implementálnotok, mely egy rendezetlen, viszont számotokra közvetlenül nem elérhető listát kell, sorba rendezzen.

A lista azért nem látható, mert „titkosított módon” egy „külső tárolóban” foglal helyet, így az közvetlenül nem írható és olvasható. A tárolóban lévő lista kezelésére (írás, olvasás) már létezik a megfelelő titkosító, visszafejtő (kititkosító) alkalmazás, viszont a listát nem tudja rendezni, ezért 3 segédfüggvényét felhasználva egy külső alkalmazás segítségével oldható meg a rendezés. A „külső tároló” alapvető tulajdonsága még az, hogy gyorsan olvasható, viszont lassan írható, tehát költségesebb az írása, mint az olvasása.

A feladat megoldásához a fenti kezelő szoftver (mely ebben a feladatban, a megoldást is tesztelő szoftver környezet része lesz) három függvényt biztosít, melyekkel (i) lekérdezhető a tárolt lista elemeinek száma, (ii) a lista bármely két eleme összehasonlítható, illetve (iii) ugyanezen lista bármely két eleme felcserélhető. A rendezendő lista fontos tulajdonsága még az, hogy nem tartalmaz két egyforma elemet.

1.1 Implementáció

Az implementálandó algoritmus célja (a „külső tároló” tulajdonságaiból kiindulva) tehát az, hogy a lehető legkevesebb összehasonlítással és elemcserével valósítsa meg a lista rendezését, tehát ezen műveletek számának különböző arányú súlyozásával képzett érték (költség) lesz a feladat pontozásánál a mérvadó. A feladatban tehát az a csapat (, vagy csapatok) éri el a maximálisan adható pontszámot, amelyik a legkisebb költséggel valósítja meg a rendezést. A költség képzése a következő számítás alapján történik:

$$költség = \frac{összehasonlítások\ száma}{1000} + cserék\ száma$$

A feladatban tehát nem számít költségnek sem a futásidő, sem a memória-használat, csak a két API függvény hívásainak száma. A megírandó algoritmusnak egy előkészített függvény vázban **void SortList(void)** kell helyet foglalnia (természetesen az algoritmus egyes eljárásai más hivatkozott függvényekben is megvalósíthatók). Az implementálandó algoritmus pedig felhasználhatja az

- **uint32_t ModuleTest_GetListSize(void)**
- **int ModuleTest_CompareListItems(uint32_t aIndex1, uint32_t aIndex2)**
- **void ModuleTest_SwapListItems(uint32_t aIndex1, uint32_t aIndex2)**

nevű függvényeket. Ezen függvények dokumentációja a feladat kódjában megtalálható.

1.2 Fejlesztő és teszt környezet

A fejlesztő és tesztelő környezet egy Eclipse projekt, mely C nyelven írt modulokat, header-eket és egy előre fordított függvény könyvtárat tartalmaz. A kapott Eclipse projektet nem szükséges módosítani, viszont a megfelelő fordítót (GCC) biztosítani kell. Saját fordító környezet is használható, de ebben az esetben nektek kell gondoskodni a projekt megfelelő fordításáról és a felmerülő problémák kiküszöböléséről, ami a feladat kidolgozásának idejéből vehet el értékes perceket. A kapott projekt módosítások nélkül is fordítható.

Az implementációt egy előre megírt modul teszt (libSortTest.a, vagy libSortTestLinux.a) segítségével tesztelhetjük le, mely ellenőrizni fogja az algoritmus által rendezett lista helyességét. Ez az automatikus tesztelés már elő van készítve a kapott környezetben (main.c) és a **ModuleTest_Start()** függvény meghívásával indítható.

A modul teszt végeztével a konzol kimeneten megjelenik a teszt eredménye, mely a „**PASSED**, vagy a „**FAILED**” üzenetet tartalmazhatja a lista rendezettségétől függően. „**PASSED**” esetén a feladat megoldása sikeres, a konzolon kiírásra kerül még (i) a „**COST**” mely a rendezési művelet költségét mutatja, ill. (ii) az összehasonlítások és (iii) a cserék száma is. „**FAILED**” esetén nem sikerült rendezett listát előállítani az implementált algoritmussal.

1.2.1 Teszt környezet elemei

- **Sort.h:** Ebben a C header-ben található a megvalósítandó API (Application Programming Interface) függvény deklarációja.
- **ModuleTest.h**: Ebben a C header-ben találhatók a teszt környezet függvény deklarációi.
- **Types.h**: Ebben a C header-ben az alaptípus definiciókon felül a *boolean* típus, és a *NULL_PTR* típus definíció is megtalálható és természetesen bárhol az implementációban fel is használható.
- **Sort.c**: Ebben a C modulban található az a függvény váz, melybe az algoritmus implementációt el kell készíteni.
- **main.c**: Ebben a C modulban található a tesztet futtató függvény.
- **libSortTest.a**: Ez a szoftver könyvtár tartalmazza az algoritmust tesztelő eljárásokat.

2. A Feladat pontozása

- Sikeres rendezés esetén (az algoritmus megvalósításától függetlenül): **25 pont**
- A legkisebb költséggel megvalósított rendezés esetén: **+35 pont**.
- A második legkisebb költséggel megvalósított rendezés esetén: **+25 pont**.
- A harmadik legkisebb költséggel megvalósított rendezés esetén: **+15 pont**.