

5. konzultáció (gyakorlat)

Bináris állományok kezelése, dinamikus tömbök

1. feladat (*binfajl.cpp*)

int fajlIr(char fnev[]); A megadott nevű bináris állományba írja ki a billentyűzetről beolvasott adatokat. (beolvasás ctrl+Z -ig: `while(!cin.eof()) {...}`)

Bináris állomány írásához objektum létrehozása és megnyitása ellenőrzöten, új állományt hoz létre:

```
ofstream ki(fnev, ios::binary);  
if(!ki)  
{  
    cerr<<"Hiba: fajliras"<<endl;  
    return 1;  
}
```

Bináris állományt kezelő objektum létrehozása és megnyitása bővítésre

```
ofstream ki(fnev, ios::binary|ios::app);
```

Bináris állományba írás

```
ki.write((char *)&sz, sizeof(sz));
```

Bináris állomány lezárása

```
ki.close();
```

int fajlOlvas(char fnev[]); A megadott nevű bináris állományból beolvassa az adatokat, és megjeleníti a képernyőn, beolvasás a fájlból fájlvége (eof) jelig:

Fájlolvasáshoz objektum létrehozása

```
ifstream be;
```

Bináris állomány megnyitása

```
be.open(fnev, ios::binary);
```

Ciklus fájlvége jelig

```
while(!be.eof()) {...}
```

vigyázni kell az utolsó adatnál ne történjen kétszer beolvasás (`if(!be.eof()) {...}`)

Bináris állományból olvasás

```
be.read((char*)&sz, sizeof(int));
```

int fajlOlvas_tombbe(char fnev[]); A megadott nevű bináris állományból beolvassa az adatokat dinamikus tömbbe, és megjeleníti a képernyőn.

1. lépés: Állomány végére pozicionálás

```
be.seekg(0, ios::end);
```

2. lépés: Adatok számának meghatározása (teljes méret bájtokban osztva egy adat méretével)

```
db=be.tellg()/sizeof(int);
```

3. lépés: dinamikus tömb létrehozása

```
int *p=new int[db];
```

4. lépés: visszapozicionálás az állomány elejére

```
be.seekg(0, ios::beg);
```

5. lépés: a tömb beolvasása

```
be.read((char*)p, db*sizeof(int));
```

```
Tárfelszabadítás: delete[]p;
```

2. feladat (*ruha.cpp*)

A **PELDATAR:\ADATOK** könyvtárban található a **ruha.dat** nevű **bináris állomány**, amely ruhák adatait a következő struktúrában tartalmazza:

```
struct ruha{
    int kod;
    char nev[15];
    int ar;};
```

Készítsen **programot**, amely **függvényhívásokkal**, és a **visszatérési érték kiíratásával** megvalósítja a következőket:

Main függvény

Határozza meg az állományban lévő adatok számát, majd dinamikus tömbbe olvassa be őket! Írassa ki az adatok számát, majd a függvény hívások elvégzése után szabadítsa fel a tömböt.

Kiír függvény

A függvény bemenő paramétere a **ruhák adatait tartalmazó tömb** és az **adatok száma** legyen, visszatérési értéke **nincs!** A függvény **minden adatot sorszámozva** az adatokat kiíratja a képernyőre (A sorszám 1-ről indul).

Legdrágabb függvény

A függvény bemenő paramétere a **ruhák adatait tartalmazó tömb** és az **adatok száma** legyen, visszatérési értéke a **legdrágább ruha ára** legyen!

Lista függvény

A függvény listázza ki a képernyőre az 5 000 Ft-nál drágább, de 10 000 Ft-nál olcsóbb ruhák adatait táblázatos formában! A függvény bemenő paramétere a **ruhák adatait tartalmazó tömb** és az **adatok száma** legyen, visszatérési értéke a **listázott adatok száma** legyen!

Keres függvény

A függvény bemenő paramétere a **ruhák adatait tartalmazó tömb**, az **adatok száma** és egy **kód** legyen, visszatérési értéke az **adott kódu ruhát tartalmazó struktúra** legyen!

3. feladat (*eredmeny.cpp*)

Az **eredmeny.dat** nevű bináris fájl, egy futóverseny adatait a következő struktúrában tartalmazza.

```
struct versenyzo{
    char nev[20];
    short int kor;
    short int helyezés;};
```

Készítsen programot, amely paraméteres függvényekkel valósítja meg a következőket:

1. függvény (FileOlvas): Ellenőrzöttén olvassa be a bináris fájl tartalmát és írassa ki a képernyőre táblázatos formában. A függvénynek át kell adni a megnyitandó fájl nevét, visszatérési értéként pedig az adatok számát kapjuk vissza, amit szintén ki kell írni.

2. függvény (Legfiatalabb): Keresse meg a legfiatalabb versenyzőt. A függvénynek át kell adni a megnyitandó fájl nevét, visszatérési értéke a legfiatalabb versenyző adatait tartalmazó struktúra legyen, amelyet a főfüggvényben kell kiírni.

3. függvény (Dobogos): Számolja meg hány dobogós helyezett versenyző szerepel az állományban, és írassa is ki az adataikat. A függvénynek át kell adni a megnyitandó fájl nevét, visszatérési értéke a dobogósok száma legyen.

4. függvény (Pozicional): A függvénynek át kell adni a megnyitandó fájl nevét és egy p egész számot, visszatérési értéke a p-edik versenyző adatait tartalmazó struktúra legyen.

5. függvény (Modosit): A függvénynek át kell adni a megnyitandó fájl nevét és egy versenyző nevét. Keresse meg a paraméterként kapott nevű versenyzőt, majd billentyűzetről kapott adatokkal írja felül az állományban az adatait. A függvény visszatérési értéke 1, ha van, 0 ha nincs a paraméterként kapott nevű versenyző az állományban.

6. függvény (Atlag): A függvénynek át kell adni a megnyitandó fájl nevét, visszatérési értéke a versenyzők átlagéletkora legyen.

7. függvény (AdatokSzama): A függvénynek át kell adni a megnyitandó fájl nevét, visszatérési értéke az állományban lévő adatok száma legyen. Használja az ftell() függvényt.

A függvények visszatérési értékeinek kiírásáról a főfüggvényben gondoskodjon!

Házi feladat

Kockadobások kiírása bináris állományba, majd beolvasása és a képernyőn táblázatos formában való megjelenítése! A kockadobások előállítása véletlenszerűen történik.

Beolvas() fv. átvesz cím szerinti paraméterátadással egy egész tömböt és hivatkozási típusként az elemszámot, visszatérési értéke nincs. A fv. bekéri, hogy hány dobás legyen, dinamikusan lefoglalja a memóriát ennyi elemre, majd véletlenszerűen feltölti a tömböt kockadobásokkal.

FileIr() fv. kiírja az állományba a kockadobásokat, a fv. átveszi a fájl nevét, cím szerinti paraméterátadással az egész tömböt és érték szerint az elemszámot, visszatérési értéke nincs.

FileOlvas() fv. kiírja a képernyőre a kockadobásokat az állományból, a fv. átveszi a fájl nevét, visszatérési értéke az adatok száma.

Atlag() fv. átveszi a fájl nevét, visszatérési értéke az adatok átlaga.

Gyakorisag() fv. kiírja a képernyőre a gyakoriságokat (hány egyes, kettes, stb. dobás volt). A fv. átveszi paraméterként a fájl nevét, visszatérési értéke nincsen.

Készítse el a tesztelő főfüggvényt!