ProgAlap

11. gyakorlat

Bevezetés

Mivel már megtanultuk, hogy hogyan kell függvényeket és tömböket kezelni, és már unhatjuk, hogy vagy statikus adatokkal dolgozunk, vagy minden egyes programfuttatáskor billentyűzetről kell beolvasni az adatokat. Jogos igényként merülhet fel, hogy háttérrendszeren tároljuk az adatok, és azokkal dolgozzunk. Persze eddig is át tudtunk irányítani a standard input-tot és output-ot, de abban az esetben a fájlnak pontosan a program beolvasási sorrendjében kellett soronként az adatokat tartalmaznia és a kiírás is a program futásának megfelelő sorrendben ömlesztve történik, és egy program csak egy fájlt tud kezelni. De nem tekinthetünk el attól, hogy be- és kimenti fájlokban szereplő adatokat strukturált formában tároljuk, hiszen így könnyebben kezelhetőek. Például, ha az a feladat, hogy olvassuk be két tömb elemszámát és adatit, akkor a bemenet átirányításával a következő fájlt lehet megadni, ami 2 elemű tömbbe Anna és Béla adatokat tölti, míg a 3 elemű tömbbe a 1986, 1990, 1985 értékeket.

A fájl: (progi.exe <adatok.be)

```
2
Anna
Béla
3
1986
1990
1985
<<vége>>
```

Sokkal szebb lenne, ha az adatokat két külön fájlban kezelnénk a következő formában:

(nevek.be)

```
2
Anna Béla
```

(evek.be)

```
3
1986 1990 1985
```

De lássuk, hogyan is lehet ezt megvalósítani!

C++ függvények

Először nézzük végig, hogy a fájlok kezeléséhez mire van szükség a C++-ban, melyek a legfontosabb elemek (változók, függvény, …).

1. Szükségünk van az fstream-re a fájlok kezeléséhez, ezért ezt használatba kell vennünk.

```
#include <fstream>
```

2. A fájlra hivatkoznunk kell egy stringgel, ami tartalmazza a fájl nevét esetleg az elérési útjával együtt.

```
string filename="befile.txt";
```

3. Az ifstream típust kell használnunk a fájl kezeléséhez. Az infile változó fog a fájlra mutatni, amin keresztül érjük el a háttértárolón lévő adatokat. A változó deklarásakor rögtön meg is adhatjuk a fájlt és a megnyitás módját, hogy automatikusan megnyíljon a kapcsolat. A példában az előző filename-t használjuk és bináris módban nyitjuk meg a fájlt. A c_str() azt jelenti, hogy C típusú karakterláncként kell átadni a fájl nyitásához az első paramétert.

```
ifstream infile(filename.c_str(),ios::binary);
```

4. Még a következő módok lehetségesek, amiket a vagy („||”) kapcsolóval lehet összekötni.

```
ios::in      //Bement
ios::out     //Kimenet
ios::binary  //Bináris mód
ios::ate     //A fájl végére teszi a mutatót. Alapértelmezetten a
fájl elején áll.
ios::app     //Hozzáfűzés, csak kimenet
ios::trunc   //Felülír
```

5. Ha végeztünk, nem szabad megfeledkezni a fájl bezárásáról!!!

```
infile.close();
```

A fájl változó, pozíciók alaphelyzetbe állítása

```
infile.clear();
```

6. A fájl megnyitása, ha már bezártuk, vagy nem nyitottuk meg a deklaráskor.

```
infile.open(filename);
```

7. Egy adat beolvasása a fájlból. Ami a példában szöveg, de lehet más is, mint a standard inputról való olvasáskor. Whitespace-ig, elválasztójelig (space, tab, sorvégjel) olvas.

```
string strbe;
infile >> strbe;
```

8. Ha minden karaktert egyesével be akarunk olvasni, akkor a get() függvényt kell használnunk.

```
char ch;
infile.get(ch);
```

9. Egy sor beolvasása a fájlból

```
getline(infile,myline);
```

10. Egy függvény, mely igaz értékkel tér vissza, ha a fájl nyitva van, különben hamis.

```
infile.is_open()
```

11. Egy függvény, mely igaz értékkel tér vissza, ha a fájl végén áll a mutató, különben hamis.

```
infile.eof()
```

12. A fájl olvasásának pozícióját tudjuk befolyásolni, például vissza tudunk ugrani a fájl elejére. Meg lehet adni pontos pozíció értéket vagy eltérést az elejéről (ios::beg), az aktuális pozíciótól (ios:cur) vagy a végéről (ios::end). Lásd még seekp; tellg; tellp.

```
infile.seekg(0);
```

13. Kimeneti fájlba ugyanúgy kell írni, mint a standard kimenetre.

```
string strki=”kiir”;
outfile << strki << endl;
```

Feladatok és megvalósításuk

1. Készítsünk függvényt, mely kap egy fájlnevet bemenetként, és beolvas egy szöveget a fájlból. (File_kezeles projekt; getStringFromFile függvény)
2. Készítsünk függvényt, mely kap egy fájlnevet bemenetként, és beolvassa a megadott fájlból egy tömb elemszámát az első és a tömb elemeit a második sorból. (File_kezeles projekt; getArrayFromFileN függvény)
3. Készítsünk függvényt, mely kap egy fájlnevet bemenetként, és beolvas egy tömböt a fájlból, ahol ugyan nem ismert a tömb mérete, de csak a tömb elemit tartalmazza a fájl. (File_kezeles projekt; getArrayFromFile függvény)
4. Készítsünk függvényt, mely kap egy fájlnevet bemenetként, és beolvas egy több dimenziós tömböt (mátrixot) a fájlból. Az első sor tartalmazza a sorok illetve oszlopok (sorokban szereplő elemek) számát. Amit az adott számú sor követ, melyben az elemek találhatóak. (File_kezeles projekt; getArraysFromFileNM függvény)
5. Készítsünk függvényt, mely kap egy fájlnevet bemenetként, és beolvas egy többdimenziós tömböt a fájlból. Automatikusan veszi fel a sorok és a sorokban szereplő elemek max. számát a két méretet tartalmazó változó. (File_kezeles projekt; getArraysFromFile függvény)
6. Készítsük el a Linux-os wc prgramot. (wc projekt)

Összefoglalás

Tehát a mai órán megtanultuk, hogy hogyan kell kezelni szöveges állományokat C++ programból.

Segédanyag: http://www.cplusplus.com/doc/tutorial/files.html