Programozás I.

Szekeres György

Eddig tanultuk

- Kiírás képernyőre
- Ékezetes karakterek kezelése
- Változók használata
- Matematikai műveletek
- Beolvasás konzolról
- Float, double, char
- If-else elágazás
- For ciklus
- While, do-while ciklusok
- Switch-case elágazás
- Tömbök
- Karakterkezelő függvények
- Többdimneziós tömbök
- Struktúrák
- String típus
- String osztály tagfüggvények
- Kiíratás
- Véletlenszám generáltatása

Fájlkezelés

- szükséges az *fstream* heather állomány használata
- ofstream: adat írása fájlba
- ifstream: adat olvasása fájlból
- fstream: írás és olvasás egyszerre
- tagfüggvények használata az egyes műveletekhez

Tagfüggvények

- filenév.open("állomány", mód); - file megnyitás, ofstream és ifstreamnél is, ha nincs ofstream automatikusan létrehozza üresen
- Módok:
  - ios::in - megnyitás bevitelre
  - ios::out - megnyitás kivitelre
  - ios::binary - bináris módú megnyitás
  - ios::ate – a kezdeti pozíciót állítsd a file legvégére. Ha nincs értéke, akkor az aktuális pozíció a file legeleje lesz.
  - ios::app – minden kimeneti művelet a file végén történik, mellékelve a feldolgozott file tartalmát, csak a kimeneti műveletek esetén használható.
  - ios::truct – ha a file meg van nyitva a kimeneti műveletek számára mielőtt létezne, a jelenlegi tartalmát törölni kell és helyette ki kell írni az újat.

Tagfüggvények

- filenév.close(): - fájl lezárása
- filenév.is_open(): - igaz értékkel tér vissza, ha a fájl nyitva van
- filenév.eof(): - megvizsgálja, hogy a fájl végén járunk-e, igaz értéket ad a végén
- filenév.bad(): - igaz értékkel tér vissza, ha az olvasás vagy az írás sikertelen
- filenév.fail(): - igazzal tér vissza mindazon esetekben, mint a bad(), de akkor is, amikor formátumhibát vétünk. Például betű jött a bemenetre, holott egész számot vártunk.
- filenév.good(): - a leggyakrabban használt jelző. Hamissal tér vissza mindazon esetekben, amikor az előző jelzők igazzal térnek vissza.
- filenév.check(): - ez nem ad visszatérési értéket, csak visszatér az alapállapotra. Főleg tesztelésre.

1. feladat

- Készítsünk egy file-t, mely vizsgálja a sikeres megnyitást, majd két sort ír egy txt-be.

1. feladat

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ofstream myfile("proba01.txt");
    if (myfile.is_open())
    {
        myfile << "Ez egy sor.\n"; //Ez a file-ba ír
        myfile << "Ez meg egy ujabb.\n"; //Ez is
        myfile.close(); //Fájl lezárása
        cout << "A file megirasa befejezodott!"; //Ez már a képernyőre megy.
    }
    else cout << "A file megnyitasa sikertelen volt.";
    return 0;
}
```

2. feladat

- Olvassuk ki az előzőleg kiírt fájlt és írjuk ki. Vizsgáljuk a megnyitást!

2. feladat

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
    string sor;
    ifstream myfile("proba01.txt");
    if (myfile.is_open())
    {
        while (myfile.good()) //tesztelés
        {
            getline(myfile, sor); //kiolvasás a file-ból
            cout << sor << endl; //kiírás a képernyőre
        }
        myfile.close();
    }
    else cout << "A file megnyitasa sikertelen volt.";
    return 0;
}
```

3. feladat

- Egy 16 csapatos labdarúgó bajnokság egyik fordulójának eredményeit tároljuk a merkozes.txt állományban. Ebben a fordulóban nem biztos, hogy 8 mérkőzés volt, néhány meccs elmaradt a rossz időjárás miatt. Minden mérkőzés eredménye új sorban van és a gólok valamint csapatok száma szóközzel van elválasztva. (3 2 ute vac)
- Olvassa be a fájlt egy olyan adatszerkezetbe, amit az alábbi kérdésekhez fel tud használni és írassa ki hány mérkőzés volt a fordulóban!
  - Mennyi csapat nyert otthon?
  - Melyik csapat rúgta a legtöbb gólt?
  - Volt-e döntetlen?
  - Mennyi gólt lőtt az ute?

3. feladat

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
struct fordulo
{
    int lott, kapott;
    string nev1, nev2;
};
int main()
{
    int i, j, db = 0, nyert = 0;
    fordulo A[8];
    ifstream be("merkozes.txt"); // Adatbevitel
if (be.fail())
{
    cout << "Hiba a fájl megnyitasa soran." << endl;
    system("pause");
    exit(1);
}
for (i = 0; i < 8 && !be.eof(); i++) // while( ) is jó
{
    be >> A[i].lott;
    be >> A[i].kapott;
    be >> A[i].nev1;
    be >> A[i].nev2;
    db++;
    cout << A[i].lott << " " << A[i].kapott << " " << A[i].nev1 << " " << A[i].nev2
  << endl;
}
```

3. feladat

```
    cout << "\nA forduloban " << db << " merkozest jatszottak." <<
     endl;
    be.close();
    cout << endl;
// Mennyi csapat nyert otthon.
    for (i = 0; i < db; i++)
    if (A[i].lott > A[i].kapott) nyert++;
    cout << nyert << " csapat gyozott otthon." << endl;
// Melyik csapat rúgta a legtöbb gólt.
    string maxcsapat;
    int max = 0;
    for (i = 0; i < db; i++)
    {
        if (A[i].lott > max)
        {
            max = A[i].lott;
            maxcsapat = A[i].nev1;
        }
        if (A[i].kapott > max)
        {
            max = A[i].kapott;
            maxcsapat = A[i].nev2;
        }
    }
    cout << "\nA legtobb golt a " << maxcsapat << " lotte." << endl;
```

3. feladat

```
// Volt-e döntetlen.
    i = 0;
    while (i < db && A[i].lott != A[i].kapott)
    i++;
    if (i < db) cout << "\nVolt dontetlen merkozes" << endl;
    else cout<< "\nNem volt dontetlen merkozes." << endl;
```

```
// Mennyi gólt lőtt az ute
    i = 0;
    while (i < db && !(A[i].nev1 == "ute" || A[i].nev2 == "ute"))
    i++;
    if (A[i].nev1 == "ute")
    cout <<endl<< A[i].lott << " golt lott az UTE." <<endl<< endl;
    else cout <<endl<<A[i].kapott<<" golt lott az UTE" <<endl<<endl;
    system("PAUSE");
    return 0;
}
```