

Bevezetés a programozásba I

8. gyakorlat

C++: szövegfolyamok, intelligens tömbök

© 2011.11.08. Giachetta Roberto
groberto@inf.elte.hu
<http://people.inf.elte.hu/groberto>

Adatfolyamok

Hibalehetőségek

- Adatfolyamnak nevezzük azt, ahol sorban tetszőleges számú adat megjelenhet a bemeneten, vagy a kimeneten, pl. konzol képernyő, fájl
- Bemenő adatok esetén sok esetben előfordulhat, hogy a bemenő adatfolyam nem a várt adatot, vagy adattípust tartalmazza
 - pl. pozitív helyett negatív szám, szám helyett szöveg, hibás fájlnev, stb.
 - bizonyos hibák kezelhetők a beolvasás után, ám amennyiben a beolvasandó elem típusa nem megfelelő, akkor a beolvasás sem történik meg (pl. szám helyett szöveget ad meg a felhasználó)

Adatfolyamok

Hibakezelése

- Minden adatfolyam esetén lehetőségünk van az állapot lekérdezésére
 - a `good()` művelet megadja, hogy az adatfolyam konzisztens állapotban van-e és elérhető
 - a `fail()` művelet megadja, hogy a legutolsó művelet sikertelen volt-e
 - az `eof()` művelet megadja, hogy az adatfolyamnak vége van-e
 - a `clear()` művelet helyreállítja az elromlott állapotot
 - továbbá minden beolvasás (`>>`, `getline`) esetén a művelet visszaadja, hogy sikeresen megtörtént-e a beolvasás az adott típusú változóba

Adatfolyamok

Hibakezelése

- Pl.:

```
int num;
cin >> num; // megpróbálunk számot beolvasni
if (cin.fail()) // ha sikertelen volt a művelet
    cout << "A megadott érték nem szám!";
else
    cout << "A beolvasás sikeres!";

// ugyanez:
if (cin >> num) // sikeres volt a beolvasás
    cout << "A beolvasás sikeres!";
else
    cout << "A megadott érték nem szám!";
```

Szövegfolyamok

Példa

Feladat: Írjuk ki egy számnak a rákövetkezőjét, de ellenőrizzük azt is, hogy számot adott-e meg a felhasználó.

- ellenőrizzük a bemeneten, hogy sikeres volt-e a beolvasás

Specifikáció:

- bemenet: egy szám (*szam*)
- kimenet: a szám rákövetkezője, vagy a hiba jelzése

Szövegfolyamok

Példa

Megoldás:

```
#include <iostream>

int main() {
    int szam;

    cout << "Kérek egy számot: ";
    if (cin >> szam) // ha be tudtuk olvasni
        cout << ++szam << endl;
    else // ha nem tudtuk beolvasni
        cout << "A megadott érték nem szám!";
    return 0;
}
```

Szövegfolyamok	
Használata	
<ul style="list-style-type: none"> A szövegfolyamok olyan folyamatok, ahol az írás és olvasás szövegen keresztül történik, így nem íródik ki fájlba, vagy a konzolra <ul style="list-style-type: none"> tetszőleges típusú változót írhatunk be és olvashatunk ki alkalmas olyan típuskonverziókat elvégezni, amik automatikusan nem történnek meg (pl. szöveg-szám) A szövegfolyam típusa a <code>stringstream</code>, használatához szükséges az <code>sstream</code> fájl és az <code>std</code> névtér <ul style="list-style-type: none"> pl. <pre>#include <sstream> ... std::stringstream sstr;</pre> 	8:7
PPKE ITK, Bevezetés a programozásba I	

Szövegfolyamok	
Műveletei	
<ul style="list-style-type: none"> A szövegfolyamba tetszőleges értéket a <code><<</code> operátorral, vagy a <code>getline</code> művelettel helyezhetünk, az egymás után bekerült értékek egymás után kerülnek A szövegfolyamból tetszőleges értéket a <code>>></code> operátorral vehetünk ki (a behelyezés sorrendjének megfelelően) <ul style="list-style-type: none"> amennyiben sikertelen a kivétel (az érték nem konvertálható az adott típusba, vagy üres a folyam), akkor a <code>fail()</code> művelettel lekérdezhető, hogy hiba történt-e a szövegfolyam végét az <code>eof()</code> művelettel ellenőrizhetjük Az <code>str()</code> művelet kiírja, vagy átalakítja <code>string</code> formátumúvá a teljes tartalmát, míg az <code>str(<szöveg>)</code> kitörli a korábbi tartalmat, és beírja az újat 	8:8
PPKE ITK, Bevezetés a programozásba I	

Szövegfolyamok	
Műveletei	
<ul style="list-style-type: none"> Pl átalakítás szöveggé: <pre>int num; cin >> num; stringstream converter; // az átalakítást szövegfolyammal végezzük converter << num; // behelyezzük a számot string output = converter.str(); // szöveggé kapjuk meg a tartalmat</pre> Pl. átalakítás számmá: <pre>stringstream converter; string input; cin >> input; converter << input; int num; converter >> num; // kiolvasás számként if (converter.fail()) // sikertelen átalakítás cout << converter.str() << " nem szám!";</pre> 	8:9
PPKE ITK, Bevezetés a programozásba I	

Szövegfolyamok	
Példa	
<p><i>Feladat:</i> Írjuk ki egy számnak a rákövetkezőjét, de ellenőrizzük azt is, hogy számot adott-e meg a felhasználó.</p> <ul style="list-style-type: none"> a bemenetet nem rögtön számként, hanem szöveggé olvassuk be használjunk szövegfolyamot a konverzió elvégzésére, csak akkor végezzük el a növelést, ha sikeres a konverzió <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> bemenet: egy szám szöveggé (<i>szöveg</i>) kimenet: a szám rákövetkezője, vagy a hiba jelzése 	8:10
PPKE ITK, Bevezetés a programozásba I	

Szövegfolyamok	
Példa	
<p><i>Megoldás:</i></p> <pre>#include <iostream> #include <string> #include <sstream> // kell a stringstream-hez int main(){ string szoveg; int szam; stringstream szfolyam; // szövegfolyam cout << "Kérek egy számot: "; cin >> szoveg; szfolyam << szoveg; // beírjuk a szöveget szfolyam >> szam; // kiolvasunk egy számot</pre>	8:11
PPKE ITK, Bevezetés a programozásba I	

Szövegfolyamok	
Példa	
<p><i>Megoldás:</i></p> <pre>if (szfolyam.fail()) // ha nem lehetett konvertálni cout << szfolyam.str() << " nem szám!"; else // ha lehetett konvertálni cout << ++szam << endl; return 0; }</pre>	8:12
PPKE ITK, Bevezetés a programozásba I	

<h3>Egyszerű tömbök</h3> <p>Használata</p> <ul style="list-style-type: none"> A C++ biztosít számunkra egy beépített tömb típust, amelyen az egyedüli értelmezett művelet az indexelő operátor használata <ul style="list-style-type: none"> pl.: <code>int t[10]; cin >> t[0];</code> lehetőségünk van ezzel a tömbbel mátrixot alkotni több index használatával, pl.: <pre>int m[10][5]; // 10*5-ös mátrix cin >> m[0][0]; // 1. sor 1. eleme</pre> ezzel a tömbbel több probléma is felmerül: nincs tisztában a saját méretével, csak konstans adható meg méretként, nem lehet a méretét futás közben megváltoztatni jó lenne, ha olyan tömböket is használhatnánk, amelyek a fenti tulajdonságokkal rendelkeznek 	8:13
PPKE ITK, Bevezetés a programozásba I	

<h3>Intelligens tömbök</h3> <p>Használata</p> <ul style="list-style-type: none"> A C++ ezért biztosít egy olyan tömb adatszerkezetet, amely a korábbi hiányosságokat pótolja, a neve <code>vector</code> <ul style="list-style-type: none"> használatahoz: <pre>#include <vector> using namespace std;</pre> létrehozásához speciálisan kell megadnunk a típust, illetve a méretet (amely lehet 0 is), pl.: <pre>vector<int> t1(10); // 10 elemű egész tömb vector<string> t2(5); // 5 elemű szövegtömb</pre> létrehozáskor a méretet nem kötelező megadni, ekkor egy 0 méretű tömböt hoz létre (ezt persze később módosíthatjuk), pl.: <code>vector<int> t3;</code> // 0 elemű egész tömb 	8:14
PPKE ITK, Bevezetés a programozásba I	

<h3>Intelligens tömbök</h3> <p>Műveletei</p> <ul style="list-style-type: none"> A <code>vector</code> leggyakoribb műveletei: <ul style="list-style-type: none"> elem lekérdezés, felülírás: <code><változónév>[<index>]</code> méret lekérdezése: <code><változónév>.size();</code> kiürítés: <code><változónév>.clear();</code> átméretezés: <code><változónév>.resize(<új méret>);</code> új elem behelyezése a tömb végére: <code><változónév>.push_back(<érték>);</code> utolsó elem kivétele: <code><változónév>.pop_back();</code> A tömb méretének tehát változó is megadható, pl.: <pre>int size; cout << "Tömb mérete: "; cin >> size; vector<double> v(size);</pre> 	8:15
PPKE ITK, Bevezetés a programozásba I	

<h3>Intelligens tömbök</h3> <p>Példa</p> <p><i>Feladat:</i> Olvassunk be valós számokat a bemenetről amíg 0-t nem írunk, majd adjuk meg az átlagnál nagyobb elemek számát.</p> <ul style="list-style-type: none"> kénytelenek vagyunk eltárolni az elemeket, ahogy az összegzés és számlálás tételét is alkalmazni tudjuk mivel nem tudjuk az elemek számát, olyan adatszerkezet kell, amelynek mérete növelhető futás közben, tehát <code>vector</code>-t kell használnunk, és annak a <code>push_back</code> függvényét <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> bemenet: valós számok sorozata (<i>v</i>) kimenet: az átlagnál (<i>atlag</i>) nagyobb elemek száma (<i>c</i>) 	8:16
PPKE ITK, Bevezetés a programozásba I	

<h3>Intelligens tömbök</h3> <p>Példa</p> <p><i>Megoldás:</i></p> <pre>#include <vector> // használjuk a vector-t ... int main(){ vector<float> v; // float típusú vector, alaphól 0 hosszú lesz float akt; cin >> akt; // előreolvasás while (akt != 0) { v.push_back(akt); // berakjuk a végére cin >> akt; } /* ugyanez hibellenőrzéssel: while (cin >> akt && akt != 0) v.push_back(akt); */</pre>	8:17
PPKE ITK, Bevezetés a programozásba I	

<h3>Intelligens tömbök</h3> <p>Példa</p> <p><i>Megoldás:</i></p> <pre>float atlag = 0; // összegzés for (int i = 0; i < v.size(); i++) atlag += v[i]; // kiolvasunk minden elemet atlag /= v.size(); // mérettel osztunk int c = 0; // számlálás for (int i = 0; i < v.size(); i++) if (v[i] > atlag) c++; cout << "Az átlagnál " << c << " elem nagyobb." << endl; return 0; }</pre>	8:18
PPKE ITK, Bevezetés a programozásba I	

Intelligens tömbök	
Példa	
<p><i>Feladat:</i> Olvassunk be valós számokat tartalmazó sorokat egy fájlból és írjuk ki a sorok átlagának minimumát.</p> <ul style="list-style-type: none"> • mivel soronként akarunk feldolgozni, a beolvasást getline segítségével végezzük, amelyből a számokat egy szövegfolyam közbeiktatásával vesszük ki • az így kapott számokat átlagoljuk (ehhez összegzés és számlálás szükséges), az átlagot eltároljuk egy vektorban, amelyre lefuttatjuk a minimumkeresés <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> • bemenet: valós számok sorozata egy fájlban (f) • kimenet: a sorok átlagának minimuma (min) 	
PPKE ITK, Bevezetés a programozásba I	8:19

Intelligens tömbök	
Példa	
<p><i>Megoldás:</i></p> <pre>#include <iostream> #include <fstream> #include <sstream> #include <vector> using namespace std; int main(){ ifstream f; string fname, line; stringstream converter; // fájlnev bekérése és fájl megnyitása cout << "Kérem a beolvasandó fájl nevét: ";</pre>	
PPKE ITK, Bevezetés a programozásba I	8:20

Intelligens tömbök	
Példa	
<p><i>Megoldás:</i></p> <pre>cin >> fname; f.open(fname.c_str()); while (f.fail()){ cout << "Hibás fájlnev!" << endl; f.clear(); cout << "Kérem a beolvasandó fájl nevét: "; cin >> fname; f.open(fname.c_str()); } f.close(); double number; // számok, amiket kiolvasunk a sorból</pre>	
PPKE ITK, Bevezetés a programozásba I	8:21

Intelligens tömbök	
Példa	
<p><i>Megoldás:</i></p> <pre>double sum; // összegzéshez int count; // számláláshoz vector<double> avgs; // átlagokat tároló vektor // fájl kiolvasása getline(f, line); // előreolvasási technika while (!f.eof()) { sum = 0; count = 0; converter.str(line); // beállítjuk a tartalmat a sorra converter.clear(); // a korábbi hibát is kitöröljük</pre>	
PPKE ITK, Bevezetés a programozásba I	8:22

Intelligens tömbök	
Példa	
<p><i>Megoldás:</i></p> <pre>while (converter >> number) { // konvertálás és hibellenőrzés sum += number; count++; } if (count > 0) // ha sikerült beolvasnunk legalább egy // számot avgs.push_back(sum / count); // az átlagot behelyezzük a vektorba getline(f, line); }</pre>	
PPKE ITK, Bevezetés a programozásba I	8:23

Intelligens tömbök	
Példa	
<p><i>Megoldás:</i></p> <pre>// minimum megállapítása if (avgs.size() > 0) { double min = avgs[0]; for (int i = 1; i < avgs.size(); i++) if (avgs[i] < min) min = avgs[i]; cout << "A legkisebb sorátlag: " << min << endl; } else { cout << "Nem sikerült egy számot se kiolvasni a fájlból." << endl; } return 0; }</pre>	
PPKE ITK, Bevezetés a programozásba I	8:24

<p>Intelligens tömbök Mátrixok kezelése</p> <hr/> <ul style="list-style-type: none"> Lehetőségünk van mátrixot létrehozni a tömbök tömbje koncepció alapján <ul style="list-style-type: none"> akkor a mátrix minden sora maga is egy vector a típus helyére egy tömbtípust adunk meg ilyenkor csak a külső tömb méretét adhatjuk meg, a belső tömbök 0 méretűek lesznek, így át kell őket később méretezni pl. egy 10*5-ös egész mátrix létrehozása: <pre>vector<vector<int> > m(10); for (int i = 0; i < m.size(); i++) m[i].resize(5);</pre> ezután az indexelés a megszokott módon, pl.: <code>m[0][0]</code> <hr/>	8:25
PPKE ITK, Bevezetés a programozásba I	

<p>Intelligens tömbök Példa</p> <hr/> <p><i>Feladat:</i> Adott 10 tanuló egyenként 5 jeggyel, állapítsuk meg a legjobb átlaggal rendelkező tanulót.</p> <ul style="list-style-type: none"> ehhez használjunk egy mátrixot, kérjük be a jegyeket egyenként határozzuk meg minden hallgatóra az átlagot, majd abból keressük meg a maximumot <p><i>Specifikáció:</i></p> <ul style="list-style-type: none"> bemenet: hallgatók jegyei egy mátrixban (<i>m</i>) kimenet: a legjobb átlagú hallgató indexe (<i>max</i>) <hr/>	8:26
PPKE ITK, Bevezetés a programozásba I	

<p>Intelligens tömbök Példa</p> <hr/> <p><i>Megoldás:</i></p> <pre>... int main() { vector<vector<int> > m(10); // sorok létrehozása for (int i = 0; i < m.size(); i++) { m[i].resize(5); // oszlopok létrehozása for (int j = 0; j < m[i].size(); j++) { cout << i+1 << ". tanuló " << j+1 << ". jegye: "; cin >> m[i][j]; // jegyek beolvasása } } }</pre> <hr/>	8:27
PPKE ITK, Bevezetés a programozásba I	

<p>Intelligens tömbök Példa</p> <hr/> <p><i>Megoldás:</i></p> <pre>vector<float> avg(m.size()); // az átlagokat egy vektorba helyezzük for (int i = 0; i < m.size(); i++) { // összegzés végrehajtása minden sorra a // mátrixban avg[i] = 0; for (int j = 0; j < m[i].size(); j++) avg[i] += m[i][j]; avg[i] /= m[i].size(); // átlagot számítunk }</pre> <hr/>	8:28
PPKE ITK, Bevezetés a programozásba I	

<p>Intelligens tömbök Példa</p> <hr/> <p><i>Megoldás:</i></p> <pre>int max = 0; // maximumkeresés az átlagok vektorán for (int i = 1; i < avg.size(); i++) { if (avg[i] > avg[max]) max = i; } cout << "Legjobban a(z) " << max+1 << ". tanuló teljesített." << endl; return 0; }</pre> <hr/>	8:29
PPKE ITK, Bevezetés a programozásba I	

<p>Intelligens tömbök Feladatok</p> <hr/> <p>Az alábbi feladatok megoldásához használjunk hibellenőrzést, valamint intelligens tömböket!</p> <p><i>III. Programozási tételek:</i></p> <ol style="list-style-type: none"> Add meg egy tetszőleges egész számsorról, hogy hány eleme nagyobb ill. kisebb az átlagánál. (A számsorozat végét q jelöli.) Számítsd ki két N dimenziós vektor skaláris szorzatát. (A felhasználótól kérjük be N értékét.) Add meg egy tetszőleges egész számsorban a szomszédos elemek átlagos különbségét. <hr/>	8:30
PPKE ITK, Bevezetés a programozásba I	

Intelligens tömbök

Feladatok

IV. Tömbös feladatok:

6. (*) Vektor permutálása, hogy végül monoton növekedő sorrendben legyenek az elemek a vektorban.
7. c) Mátrixban tároljuk egy osztály adatait, minden sora egy diák, minden oszlopa egy tantárgy, a mátrix értékei a jegyek. Melyik a legnehezebb tantárgy (legtöbb bukás).
8. a) Adott méretű mátrix feltöltése fájlból, majd (megfelelő méretek esetén) dönts el, hogy szimmetrikus-e. (A fájl első két adata a mátrix mérete, majd utána jönnek az értékek sorfolytonosan).