3. gyakorlat

Beolvasás, tömbök, beolvasás fájlból

Azt fogjuk rövid példákon kipróbálni, rövid előadás-emlékeztető magyarázattal, hogy hogyan lehet különböző módokon beolvasni adatokat, illetve kiíratni.

Az inputnak is, meg az outputnak is 3 módja van:

Input	Output
parancssori argumentumok (argv)	main visszatérési értéke (linuxban exit status érték)
consol input (cin)	consol output (cout)
file input (ifstream)	file output (ofstream)

Parancsargumentumok beolvasása

Gondolhatunk arra, hogy a múlt heti feladatban újabb és újabb kövek feliratának megfejtéséhez körülményes mindig újrafordítani a programot. Ezért praktikus, ha az adatokat valamelyik input módon adjuk meg.

Másik szempont, hogy ha olyan programokat akarunk írni, mint pl. a linux parancsok nagy része, amiknek egy vagy több argumentumot kell megadnunk. Pl. a head parancs esetében egy számot és egy sztringet (a fájlnevet).

Próbáljunk most ilyen, mondjuk 2 argumentumot váró programot írni! Codeblocks használata esetén kezdhetünk egy új projektet. Úgy, mint más függvénynek is, a main-nek is a zárójelei közt kell megadni, hogy milyen argumentumokat kap. Mivel az oprendszer adja át neki, a formátuma kötött: egy int típusú változó, és egy, sztringekből álló tömb. A sztringekről, és a *-os tömb definícióról még lesz szó, most fogadjuk el, hogy ezt kell beírnunk:

In [ ]:
```
int main(int argc, char* argv[])
```

Az argc, argv elnevezés megváltoztatható, de ezek a szokásos nevek: argc jelentése argumentum counter, ez az argumentumok száma, hozzá véve a parancsot; argv jelentése argumentum vektor, ez a parancsszóból (path+program) és az argumentumokból, mint sztringekből álló tömb.

Mielőtt kiolvassuk ezeket, meg kell bizonyosodnunk róla, az argc értéke alapján, hogy van-e elegendő argumentum. Enélkül könnyen érvénytelen indexet írnánk be. Ha nincs elegendő (azaz a paranccsal együtt nincs meg a 3 elem), akkor írjunk ki hibaüzenetet! Állítsuk is le a programot az exit utasítással, átadva vele egy 0-tól különböző hibakódot! Ez megoldható pl. az alábbi if utasítással:

In [ ]:
```cpp
#include <iostream>

using namespace std;

int main(int argc, char *argv[]) {

    if (argc<3) {
        cout << "Error: not enough number of arguments: " << argc-1 << endl;
        exit(1);
    }

    cout << "command line arguments: " << argv[1] << " " << argv[2] << endl;
    ...
    return 0;
}
```

Utána következhet az argumentumok kiolvasása és a feladat elvégzése. Ehhez nem is kell else ágat csinálni, hiszen ha a feltétel teljesült, akkor úgyis kiléptünk a programból.

Példánkban egyelőre csak próbaképp kiíratjuk a két argumentumot, stringben megadott megjegyzéssel együtt. A string végére ilyenkor spacet érdemes tenni, hogy ne follyon egybe az argumentumokkal, ill. hasonló okból a két argumentum közé is egy spacet tartalmazó stringet. Ezzel már ki tudjuk próbálni az argv működését.

Nézzük, hogyan kell beadni az argumentumokat! Linuxban parancssorban (pl. kooplex terminálban is) a linux parancsokhoz hasonlóan, a lefordított programot elindítva a neve mögé írjuk az argumentumokat (pl. ./main 12 file.txt). Windowsban is, megfelelő beállításokkal ez elérhető a command prompt v. powershell használata közben.

Ha CodeBlocksot használunk, akkor van egy kényelmesebb lehetőség: a Project / Set programs' arguments... menüpontban meg lehet adni az argumentumokat. (Ide ne írjuk be a parancs nevét, csak az argumentumokat!)

Amikor fel akarjuk használni az argumentumokat, akkor arra is kell gondolnunk, hogy a számokat is karaktersorozatként kapjuk meg (a rendszer nem tud különbséget tenni), így azt át kell alakítanunk pl. int vagy double típusú számmá. Ezt az stoi (azaz string to int, vagy atoi, azaz asci to int) függvénnyel tudjuk megtenni int esetén. Float változóba beolvasva stof (v. atof) kell, és double esetén stod (v. atof, mert atod nincs külön). Általában inkább az s betűs változatokat javasolják. Így tehát a ... részbe a következőképpen írhatjuk be az átalakító utasítást; majd utána mindjárt kiírathatjuk a már értelmezett érétékeket:

In [ ]:
```cpp
    int n=stoi(argv[1]);
    string filename=argv[2];
    cout << "command line arguments: " << argv[1] << " " << argv[2] << endl;
```

Próbáljuk ki azt is, hogy ezzel nem csak egész típusúvá alakítjuk a számértéket, de ha a parancssorban megadott szám nem egész, akkor a kiírt számérték is megváltozik a két kiíratás között!

A main függvény visszatérési értéke és console output

Kis programunkkal a parancsargumentumon kívül a main visszatérési értékének és a console outputnak a használatát is gyakoroltuk.

Console input és output

A konzolról való beolvasás hasonlóan végezhető a kiíráshoz, csak a "cin" helyett a "cout" objektumot kell használni, és természetesen fordítottak a "<<" jelek. Pl. így olvashatunk be 3 számot és írhatunk ki mindjárt utána a képernyőre:

In [ ]:
```
    double a,b,c;
    cin >> a >> b >> c;
    cout << "a " << a << "\nb " << b << "\nc " << c << endl;
```

Érdemes kipróbálni, hogy amikor a program futása megáll, és várja az inputot, akkor mindegy, hogy a 3 számból hányat adunk meg egy sorban. Ha nem adunk meg 3 számot, akkor enter után tovább vár egy következő sorban a további számokra. A változókba való bekerülés szempontjából is mindegy, hogy a 3 számot hogyan tagoljuk 1 vagy több sorba. A kiírásnál viszont az elrendezés a parancsban megadottak szerint szabályozható. Példánkban az egyes változók kiírása előtt stringként megadva kiírattuk a nevüket. Itt vigyázni kell, hogy a szövegkaraterek és a számok ne follyanak egybe, 1--2 spacet vagy soremelést érdemes tenni közéjük. A soremelést a "\n"-nel (newline) jelölt kontrollkarakterrel végezhetjük.

Fájl input és output

Fájlból való beolvasáshoz létre kell hozni az adatok áramlatát biztosító ifstream típusú objektumot (input-file-stream), aminek megadjuk, hogy melyik fájlhoz kapcsolódjon. A streamnek lényegében tetszőleges nevet adhatunk, pl. filein. Ezt a következő deklarációval tehetjük meg, utána a cin-hez hasonlóan használhatjuk az adatok beolvasására:

In [ ]:
```
    ifstream filein("in.dat");
    double a,b,c;
    filein >> a >> b >> c;
    cout << "a " << a << "\nb " << b << "\nc " << c << endl;
                    // for now, we will just write out what we have read
```

Hozzunk létre egy 3 számot tartalmazó in.dat fájlt, és próbáljuk ki programunk! Persze hibajelzést kapunk, mert a fájl stream deklarálásához be kell töltenünk a megfelelő csomagot. Ehhez írjuk az include sorhoz:

In [ ]:
```
#include <fstream>
```

Annyiban is a cin-hez hasonlóan működik, hogy itt is mindegy, hogyan taguljuk a fájl soraiba a számokat. Próbáljuk ki ezt is!

Fájlba írás lépései hasonlóak:

In [ ]:
```
    ofstream fileout("out.dat");
    fileout << sqrt(a) << "  " << sqrt(b) << "  " << sqrt(c) << endl;
```

Itt a négyzetgyök függvény eléréséhez még a cmath csomag betöltése szükséges. Így a teljes program:

In [ ]:
```
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

int main(int argc, char *argv[]) {

    ifstream filein("in.dat");
    double a,b,c;

    filein >> a >> b >> c;
    cout << "a " << a << "\nb " << b << "\nc " << c << endl;

    ofstream fileout("out.dat");
    fileout << sqrt(a) << "  " << sqrt(b) << "  " << sqrt(c) << endl;

    return 0;
}
```

Néhány gyakorlati tanács, amit menetközben, vagy akár előre kipróbálhatunk:

- egy függvény nevének beírása közben a codeblocks felajánlja a már beírt szórészlettel kezdődő függvényneveket, egyúttal mutatja a visszatérési érték típusát (nem minden függvényre működik, de saját függvényeket is kiad)
- a függvény neve utáni zárójelet beírva a függvény szintaxisa jelenik meg (sajnos ez sem mindig működik)
- bővebb leírást könnyen találunk a neten (pl. google: c scanf), a c függvényekről pl. a tutorialspoint.com ad jó leírást egyszerű példákkal
- linux terminálban is lekérdezhetjük a c függvényeket, pl.: man scanf
- globális változókat nem tanácsos használni, ezért a beadandó feladatokban elvárjuk, hogy ne legyenek ilyenek (kivételes esetek vannak, amikor nehéz másként megoldani, pl. ha a quick sort algoritmus esetében az összehasonlítások számát akarjuk meghatározni)
- ha ékezetes karaktereket is használunk (akár csak a kommentekben) és el akarjuk kerülni a program és outputjának megjelenítési problémáit más gépeken (pl. kooplex), állítsuk át UTF-8-ra a karakterkódolást a codeblocksban a Settings - Editor - Encoding settings menűpontban!