Karakterláncok II

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2023 október 24.

Szöveges fájlformátumok

Karakterláncokat használhatunk szöveges adatfájlok feldolgozására

Szöveges fájlformátumok

Karakterláncokat használhatunk szöveges adatfájlok feldolgozására

Néhány egyszerő adatfájl-formátum:

- CSV: comma-separated values: oszlopok, vesszőkkel elválasztva
- oszlopok, üres karakterekkel elválasztva
- fix számú karakterből álló oszlopok

Általában előfordulnak speciális sorok

- fejléc, ami a bemenő paraméterek értékét, vagy az adatoszlopok nevét tartalmazza
- megjegyzés, amit többnyire speciális karakter vezet be
- pl üres sor, mátrix egyes sorait tartalmazó adatblokkok között

Szöveges fájlformátumok

A szöveges fájlok valamilyen karakterkódolással vannak tárolva

- feltesszük, hogy a karakterek egybájtos kódolással vannak tárolva
  - 1 bájt = 1 betű
  - a bájtok értékeit betűkhöz lehet rendelni
  - az alapkaraktereket az ASCII szabvány definiálja
- ha mégsem, akkor a legegyszerűbb valami külső programmal egybájtosra konvertálni
- linuxon pl. az `iconv` programmal

A szöveges fájlok sorokból állnak

- a sorok végén új sort jelölő karakter (`\r`, `\n` stb.)
- de egy sor elvileg *akármilyen hosszú lehet*!

`getline()` függvény

Ha soronként akarunk beolvasni adatokat: `getline()`

- egy bemeneti `stream`-t és egy `string` típusú változót használ
- az első újsor karakterig olvas a `stream`-ből és beleteszi a `string` változóba

getline() függvény

Ha soronként akarunk beolvasni adatokat: `getline()`

- egy bemeneti `stream`-t és egy `string` típusú változót használ
- az első újsor karakterig olvas a `stream`-ből és beleteszi a `string` változóba

```
int main()
{
ifstream inpfile("inp_data.dat");
std::string line;

while (getline(inpfile,line))
 { if (!line.empty())
  {cout << line << endl;}
 }
 retrun 0
}
```

- megnyitunk egy `stream`-t (most egy fájlt) olvasásra
- amíg nem érünk a fájl végére, soronként olvasunk
- lekérdezzük, hogy a `line` üres-e: `.empty()` operáció
- ha `line` nem üres, akkor `line.empty` értéke `false`
- csak akkor írjuk ki a sort, ha nem üres

getline() függvény

Egy sor (elvileg) nagyon hosszú is lehet

Hasznos operáció:

- `.size()`: a beolvasott sor hosszát adja meg

Figyeljük meg:

- a `line.size()` egy `size_t` típusú változót ad vissza (nem `int` vagy `unsigned` típusút)
- ha szükségünk van a beolvasott sor hosszára:
  - deklarálhatunk egy `size_t` típusú változót és abban eltárolhatjuk
  - használhatjuk a korábban már látott `auto` típust:

```
auto len = line.size();
```

Karaketerek feldolgozása egy beolvasott sorban

Szükség lehet arra, hogy egy beolvasott sort karakterenként feldolgozzunk

Két egyszerű módszer:

- ha minden karaktert fel kell dolgozni egy sorban: range based `for` ciklust

```
string line = "valami szoveg"
 for (auto c: line)
 {
// do something
 }
```

  - a ciklus változó most a c lesz
  - típusát megadhatnánk explicit is (`char`)

- ha nem minden karaktert kell feldolgozni: karakterek elérése az indexük segítségével

```
if (!line.empty())
 {
   line[0]=toupper(line[0]);
 }
```

  - az első karaktert nagybetűre változtatjuk a könyvtári `toupper()` függvénnyel

Karaketerek feldolgozása egy beolvasott sorban

További lehetőség

- könyvtári függvények, amely a `cctype` headerben vannak definiálva