Karakterláncok (sztringek)

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2022 október 11.

Karakterláncok (stringek)

Emlékeztető: egy bájton 256 különböző értéket lehet tárolni $\Rightarrow$ felhasználhato karakterkódolásra

- a szövegek karakterenként tárolódnak
- egyszerűbb eset: 1 bájt = 1 betű
- a bájtok értékeit betűkhöz kell rendelni
- az alapkaraktereket az ASCII szabvány definiálja
- az ékezetes betűket valamilyen kódlap szerint osztjuk ki
  pl. magyar nyelvhez: ISO 8859-2 (Latin-2), Windows-1250

Unicode karakterek (kitekintés)

- gyakorlatilag tetszőleges nemzetközi szöveg reprezentálására
- a szabvány majdnem 140 ezer karaktert definiál
- plusz speciális vezérlőjelek
- legalább 4 bájt kellene az egyedi karakterek tárolásához
- egy nyelv egyszerre sosem használja az összes karaktert
- ezért helyette általában kódolás: UTF-8, stb.

Alapvető speciális vezérlőkarakterek

Ezekhez nem tartozik betű, hanem valamilyen *hatásuk* van

- eredetileg a nyomtató vezérlésére használták
- a terminálok emulálják a nyomtató működését
- a C nyelv ehelyett \-sel kezdődő ún. escape-szekvenciákat is elfogad

Gyakran használt vezérlőkarakterek
Ezeket és a szóközt karaktert együtt *whitespace*-nek hívjuk

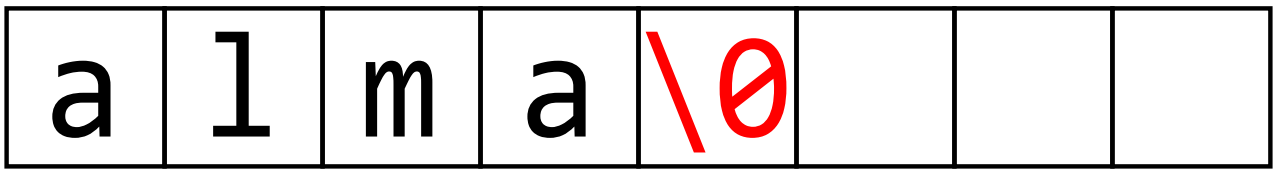
- `\t`: tabulátor (pl. fájlokban oszlopok között)
- `\n`: új sor
- `\r`: kocsi vissza (ld. nyomtatók)[1]

[1] Újsor jel Windows-on: `\r\n`, régi Mac-en: `\r`

Karakterláncok (stringek) a C nyelvben

A C nyelvben nincsen külön `string` típus!

- helyette: `char` típusú pointer mutat az első karakterre
- a karakterek egymás után folytonosan a memóriában
- a legutolsó karakter *után* kötelezően áll egy `\0`, ún. NULL karakter
- figyelem! emiatt mindig eggyel több bájtot kell allokálni, mint a szöveg maximális hossza



Szöveges konstansok

A C nyelvben ugyanakkor van *string konstans*!

- a stringkonstansokat dupla idézőjel jelzi, pl.: `"alma"`
- a stringkonstansok végére a fordító automatikusan kiteszi a lezáró `\0`-t
- létezik *üres string*: `""`: ez egyetlen bájtból áll, aminek 0 az értéke

```
1 int main() {
2   char *s = "alma";
3   printf("%s\n", s);
4   return 0;
5 }
```

Figyelem!

- a karakterkonstansokat szimpla idézőjel jelzi, pl.: `'a'`, `'\t'` stb.
- tehát az egyetlen karakterből álló sztring különbözik a karakterkonstantól! sztring esetén van egy lezáró `\0`
- üres karakter nincs, `''` hibás!

Műveletek stringekkel

Mivel nincsen `string` típus, ezért ezen ható operátorok sincsenek

- helyette függvények a szöveges adatok kezelésére
- külön könyvtárak különböző karakterkódolásokhoz

Néhány alapvető függvény

- `strlen`: string hossza, az utolsó `\0`-t nem beszámítva
- `strcmp`: két string összehasonlítása
- `strcpy`: egyik string másolása másikba
- `strcat`: egyik string hozzáfűzése egy másikhoz
- `sprintf`: formátumozott string készítése
- ezen függvények használatához a `<string.h>` header szükséges

Figyelem!

- a stringkezelő függvények mindig a `\0` karaktert várják a végén
- a bemenetük `char*`, és nem tudják, hogy mennyi memória lett foglalva a szövegnek

Szöveges fájlformátumok

Néhány alapvető adatfájl-formátum

- CSV: comma-separated values: oszlopok, vesszőkkel elválasztva
- tabular: oszlopok, általában `\t` karakterekkel elválasztva
- fix számú karakterből álló oszlopok
  itt a számok fix tizedesjeggyel, lehetnek jobbra igazítva

Szöveges fájlformátumok

Néhány alapvető adatfájl-formátum

- CSV: comma-separated values: oszlopok, vesszőkkel elválasztva
- tabular: oszlopok, általában `\t` karakterekkel elválasztva
- fix számú karakterből álló oszlopok
  itt a számok fix tizedesjeggyel, lehetnek jobbra igazítva

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

Hogyan kezeljük az ilyen adatfájlokat?