

Programiranje 2

10. predavanje — dodatak

Saša Singer

`singer@math.hr`

`web.math.pmf.unizg.hr/~singer`

PMF – Matematički odsjek, Zagreb

Sadržaj predavanja — dodatka

- Implementacija nekih funkcija:
 - Funkcija `my_gets`.

Implementacija funkcije `my_gets`

Specifikacija funkcije `my_gets`

Na 6. predavanju spomenuta je “moja” funkcija `my_gets`, kao zamjena za nesigurni `gets` i fleksibilna varijanta `gets_s`.

Svrha je čitanje stringova sa standardnog ulaza (`stdin`), po načelu “jedan string u jednom redu”. Prototip funkcije je

```
char *my_gets(char *s, int n)
```

Funkcija `my_gets` uvijek čita cijeli red ulaza, do (uključivo) kraja reda (`\n`), kraja datoteke ili greške. U string `s` sprema

- najviše $n - 1$ znak s ulaza (bez `\n`) i dodaje nul-znak,
- a eventualni višak znakova u redu se ignorira (preskače).

Izlaz je `s` ili `NULL`, kao kod `gets`. Ako su argumenti korektni, u `s` uvijek sprema korektan string, iako može vratiti `NULL`.

Funkcija `my_gets` — izlaz, implementacija

Funkcija `my_gets` vraća `NULL` u **tri** slučaja:

- ako su argumenti **pogrešni** (tad učitava red, **ne** sprema ga),
- ako na **početku** čitanja naiđe na **kraj datoteke** (**EOF**),
- ako dođe do **greške** prilikom čitanja, **prije** nego je učitano `\n` ili **EOF**.

Implementacija se nalazi u datoteci `my_gets.c`, koja sadrži

- **uključivanje** zaglavlja `<stdio.h>` (za `NULL`, `EOF`, funkcije `getchar`, `feof` i `ferror`) i cijeli **kôd** funkcije `my_gets`.

Ako nam treba funkcija `my_gets` u nekom programu, umjesto kopiranja kôda, samo **uključimo** ovu datoteku u taj program:

```
#include "my_gets.c"
```

Funkcija `my_gets`

Napomena. Većina komentara iz datoteke `my_gets.c` ovdje je napisana kao običan tekst, a cijeli radni dio kôda rastavljen je u manje smislene cjeline (blokove), tako da se lakše čita.

Prvo uključujemo `<stdio.h>`, a zatim ide kôd funkcije. Počinje prototipom i otvaranjem bloka (tijela funkcije).

```
#include <stdio.h>
```

```
char *my_gets(char *s, int n)  
{
```

Na vrhu bloka, odmah deklariramo sve potrebne (lokalne) varijable, uz detaljan opis značenja svake od njih.

Funkcija `my_gets` (nastavak)

Varijabla `izlaz` sadrži `izlaznu` (povratnu) vrijednost funkcije (pokazivač na znak = string). Inicijaliziramo ju na ulazni argument `s`, očekujući da će funkcija uspješno obaviti posao. Ako treba, kasnije postavljamo `izlaz = NULL`, kao signal za grešku (neuspjeh).

```
char *izlaz = s;
```

Cjelobrojne varijable: `i` je indeks prvog “slobodnog” mjesta u polju `s` (trenutno inicijaliziran na 0), `max_i` je maksimalni indeks za spremanje u polje `s`, a `c` je učitani znak s ulaza.

```
int i = 0;  
int max_i, c;
```

Funkcija `my_gets` (nastavak)

Prvi posao je provjera ograničenja na ulazne argumente `s` i `n`.

● Mora vrijediti `s != NULL` i `n > 0`.

Ako je `s == NULL`, onda ne možemo koristiti `s` kao polje za spremanje znakova. Ako je `n ≤ 0`, onda “dozvoljena” duljina polja `s` nije pozitivna \Rightarrow ne smijemo ništa spremiti u `s`.

U slučaju prekršaja, postavljamo `izlaz = NULL` i `n = 0` (to je zgodno za nastavak), ali se funkcija ne vraća odmah, jer mora učitati cijeli red teksta (svaki poziv čita jedan red ulaza)!

```
if (s == NULL || n <= 0) {
    izlaz = NULL;
    n = 0;    /* Koristimo u nastavku. */
}
```

Funkcija `my_gets` (nastavak)

Sad postavimo **maksimalni** indeks na $n - 1$. Ako su argumenti **pogrešni**, onda je $\text{max_i} = -1$, što će **spriječiti** spremanje u **s**!

```
max_i = n - 1;
```

Sljedeći blok je **petlja** za čitanje **jednog cijelog** reda (linije teksta) s ulaza (**stdin**),

• sve do (prvog) **kraja reda** (**\n**), **kraja datoteke** ili **greške**.

Petlja je **beskonačna**, a **izlaz** je s **break**, zbog izlaska u **sredini** petlje. Unutar te petlje,

• **učitamo** sljedeći **znak c** s ulaza (funkcijom **getchar**),

• **provjerimo kraj** čitanja = **izlaz** iz petlje,

• **spremimo** učitani znak u **s**, ako još **stane** u **s**.

Funkcija `my_gets` (nastavak)

Kod provjere **izlaza** iz petlje, koristimo da `getchar` vraća **EOF** (samo) u slučaju **kraja datoteke** ili **greške**. Kasnije ćemo provjeriti **što** od toga se dogodilo i odgovarajuće reagirati.

```
while (1) {
    c = getchar();
    if (c == '\n' || c == EOF) break;
    if (i < max_i) {
        s[i] = c;    /* Spremi znak. */
        ++i;        /* Povecaj i. */
    }
}
```

Za **pogrešne** argumente, **nema** ni **pokušaja** spremanja u **s**, jer je $i = 0 > \text{max_i} = -1$. Inače, na **kraju** je $i = \text{max_i} = n - 1$.

Funkcija `my_gets` (nastavak)

Ako su argumenti bili **korektni** ($\Leftrightarrow n > 0$), onda prvo **dodamo** nul-znak na **kraj** polja `s` (uvijek vratimo **string**), a zatim

- ako je **zadnji** učitani znak `c == EOF`, provjerimo je li došlo do **greške** ili smo **odmah** (za `i = 0`) naišli na **kraj datoteke** — tada postavimo `izlaz = NULL` (kao **signal**).

```
if (n > 0) {
    s[i] = '\0';
    if (c == EOF) {
        if (ferror(stdin)) izlaz = NULL;
        else if (i == 0 && feof(stdin))
            izlaz = NULL;
    }
}
```

Funkcija `my_gets` — kraj, testiranje

Na samom kraju, **vratimo** izlaznu vrijednost i **zatvorimo** blok tijela funkcije.

```
    return izlaz;  
}
```

Glavni program u `my_gets_t.c` radi skoro **potpuno** testiranje funkcije `my_gets`.

- 🔴 **Jedino** što je (gotovo) **nemoguće** provjeriti **običnim** testiranjem je simulacija **greške** prilikom čitanja!

Zgodne **ulazne** test-datoteke su `gets_1.in` i `gets_2.in` (standardni ulaz treba preusmjeriti na test-datoteku).