Uvod u programiranje

- predavanja -

siječanj 2026.

21. Standardna biblioteka

Ponavljanje <string.h>

```
char *strcpy(char *s1, const char *s2);
```

```
char *strncpy(char *s1, const char *s2, size_t n);
```

```
char *strcat(char *s1, const char *s2);
```

```
char *strncat(char *s1, const char *s2, size_t n);
```

```
size_t strlen(const char *s);
```

```
int strcmp(const char *s1, const char *s2);
```

```
int strncmp(const char *s1, const char *s2, size_t n);
```

Primjer

- Programski zadatak
  - napisati definiciju funkcije `mystrcmp` koja obavlja isto što i funkcija `strcmp` iz `<string.h>`
  - u glavnom programu učitati dva niza znakova koji zajedno s eventualno učitanom oznakom novog retka sigurno neće biti dulji od 20 znakova. Pomoću funkcije `mystrcmp` usporediti nizove i ispisati odgovarajuću poruku

  - Primjeri izvršavanja programa

```
1. niz > kisik↵
2. niz > kiselo↵
1. niz je veci
```

```
1. niz > kiselo↵
2. niz > kisik↵
2. niz je veci
```

```
1. niz > kisik↵
2. niz > kisik↵
nizovi su jednaki
```

Rješenje (1. dio)

```
...
int mystrcmp(const char *s1, const char *s2) {
   while (*s1 == *s2 && *s1 != '\0' && *s2 != '\0') {
      ++s1;
      ++s2;
   }
   return *s1 - *s2;
}
```

Može se ispustiti

ili

```
int mystrcmp(const char *s1, const char *s2) {
   for (; *s1 == *s2 && *s1 != '\0'; ++s1, ++s2);
   return *s1 - *s2;
}
```

Rješenje (2. dio)

```
#define MAXNIZ 20

int main(void) {
   char niz1[MAXNIZ + 1];
   char niz2[MAXNIZ + 1];
   printf("1. niz > ");
   fgets(niz1, MAXNIZ + 1, stdin);
   printf("2. niz > ");
   fgets(niz2, MAXNIZ + 1, stdin);

   int rez = mystrcmp(niz1, niz2);
   if (rez == 0) {
      printf("nizovi su jednaki");
   } else if (rez > 0) {
      printf("1. niz je veci");
   } else {
      printf("2. niz je veci");
   }
   return 0;
}
```

strchr <string.h>

```
char *strchr(const char *s, int c);
```

- traženje prve pojave zadanog znaka `c` unutar niza `s`
- funkcija vraća
  - pokazivač na prvi znak unutar niza znakova koji ima vrijednost `c`
  - ako takav znak u nizu ne postoji, vraća `NULL`

```
char niz[] = "San Antonio";
char *p = "New Orleans";
printf("%c", *strchr(niz, 'a'));
printf("%s", strchr(p, 'e'));
printf("%s", strchr(p + 2, 'e'));

strchr(p, 'R');
```

```
a
ew Orleans
eans

vraća NULL
```

strrchr <string.h>

```
char *strrchr(const char *s, int c);
```

- traženje zadnje pojave zadanog znaka c unutar niza s
- funkcija vraća
  - pokazivač na zadnji znak unutar niza znakova koji ima vrijednost c
  - ako takav znak u nizu ne postoji, vraća NULL

```
char niz[] = "San Antonio";
char *p = "New Orleans";
printf("%c", *strrchr(niz, 'n'));
printf("%s", strrchr(p, 'e'));

strrchr(p + 1, 'N');
strrchr(p, 'R');
```

n
eans

vraća NULL
vraća NULL

strstr <string.h>

```
char *strstr(const char *s1, const char *s2);
```

- traženje prve pojave podniza u s1 koji je jednak nizu s2 (znak terminatora niza s2 se ne uzima u obzir kod usporedbe)
- funkcija vraća
  - pokazivač na prvi znak pronađenog podniza
  - ako takav podniz u nizu s1 ne postoji, vraća NULL

```
char niz[] = "Nigdar ni tak bilo da ni nekak bilo";
char *p = "tak";
printf("%s", strstr(niz, p));
printf("%s", strstr(niz, "ni"));
printf("%s", strstr(strstr(niz, "ni") + 1, "ni"));
strstr(niz, "Tak");
```

tak bilo da ni nekak bilo

ni tak bilo da ni nekak bilo

ni nekak bilo

vraća NULL

strpbrk <string.h>

```
char *strpbrk(const char *s1, const char *s2);
```

- traženje u nizu s1 prve pojave bilo kojeg od znakova navedenih u nizu s2
- funkcija vraća
  - pokazivač na prvi pronađeni znak
  - ako niti jedan od znakova iz s2 ne postoji u nizu s1, vraća NULL

```
char niz[] = "Gle, jedna duga u vodi se stvara,";
char *p = "aeiou";
printf("%s", strpbrk(niz, p));              e, jedna duga u vodi se stvara,
printf("%s", strpbrk(niz + 7, p + 1));      uga u vodi se stvara,

strpbrk(niz, "AEIOU");                      vraća NULL
```

Primjer

- Napišite program koji će učitavati redak po redak teksta sve dok se ne učita redak teksta koji sadrži niz znakova `KRAJ`.
- Nakon unosa zadnjeg retka, potrebno je ispisati sve retke, osim zadnjeg, redom kako ih je korisnik unosio. Redak teksta, uključujući oznaku novog retka, sigurno nije dulji od 80 znakova. Broj redaka teksta nije unaprijed poznat.
- Primjer izvršavanja programa

```
Upisite recenice >↲
Mali princ je neobicna knjiga: mozda za djecu, mozda za odrasle.↲
U njoj se mijesaju baobabi i ruze, ironija i pjesnicke slike.↲
I pisceve ilustracije.↲
Tesko je naci knjigu koja bi joj bila slicna.↲
No, i autor Maloga princa prilicno je neobicna pojava u svijetu knjizevnosti.↲
Roden 29. lipnja 1900. u plemickoj obitelji, u francuskom gradu.
KRAJ ↲
↲
Tekst:↲
↲
Mali princ je neobicna knjiga: mozda za djecu, mozda za odrasle.↲
U njoj se mijesaju baobabi i ruze, ironija i pjesnicke slike.↲
I pisceve ilustracije.↲
Tesko je naci knjigu koja bi joj bila slicna.↲
No, i autor Maloga princa prilicno je neobicna pojava u svijetu knjizevnosti.↲
Roden 29. lipnja 1900. u plemickoj obitelji, u francuskom gradu. ↲
```

Rješenje

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXNIZ 80

int main(void){
    char *tekst = malloc (1 * sizeof(char));    //za \0
    int duljinaTeksta = 1;
    char recenica[MAXNIZ + 1];
    *tekst = '\0';

    printf("Upisite recenice > \n");
    while (strstr(fgets(recenica, MAXNIZ + 1, stdin), "KRAJ") == NULL){
        char *pomTekst = realloc(tekst, (duljinaTeksta + strlen(recenica)+1) * sizeof(char));
        if (pomTekst == NULL) {
           printf("Nedovoljno memorije.");
           return 1;
        }
        tekst = pomTekst;
        strcat(tekst, recenica);
        duljinaTeksta = duljinaTeksta + strlen(recenica);
    }
    printf("\nTekst:\n\n");
    printf("%s", tekst);
    free(tekst);

    return 0;
}
```

Zadatak

Jedna od metoda za određivanje sličnosti dvaju nizova znakova - metoda n-grama rastavlja nizove na podnizove (n-grame) zadane duljine, a sličnosti određuje kao kvocijent broja podudarnih n-grama i ukupnog broja n-grama (bez ponavljanja).

Primjer

1. niz > "sreca"
2. niz > "srecemo"

i duljinu podniza jednaku 4, podnizovi su :

{"srec", "reca"}

{"srec", "rece", "ecem", "cemo"}

Podudarni: {"srec"}

Sličnost: 1/(2+4-1)=0,2.

Napisati funkcije:

- **void makeSubStr (const char *src, char *subStr, int startPosition, int subStrLen);**
  na temelju ulaznog niza znakova (**src**) generira podniz (**subStr**) zadane duljine (**subStrLen**) počevši od zadane pozicije (**startPosition**). Ne treba provjeravati jesu li ulazni parametri ispravni.
  Primjer: za ulazni niz "sreca", subStrLen = 3, startPosition = 0, funkcija niz znakova subStr treba napuniti sadržajem "sre", a za startPosition = 1, sadržajem "rec" itd.
- **_Bool containsSubStr (char *str, char *subStr);**
  koja vraća logičku vrijednost istina ako je podniz (**subStr**) sadržan u nizu (**src**), a inače vraća laž.

Zadatak

Napisati glavni program kojim je potrebno odrediti i ispisati sličnost dva niza znakova.
S tipkovnice učitati dva niza znakova ne dulja od 100 znakova i duljinu podniza (ništa nije potrebno provjeravati).
Sličnost dvaju nizova je definirana kao omjer broja podudarnih podnizova i ukupnog broja podnizova oba niza, ali podudarne podnizove treba uzeti samo jednom u obzir.

```
1. niz > sreca je bljesak
2. niz > trenutak srece
Duljina podniza > 4
Slicnost: 1/(13+11-1) = 0.043478
```

Standardna biblioteka

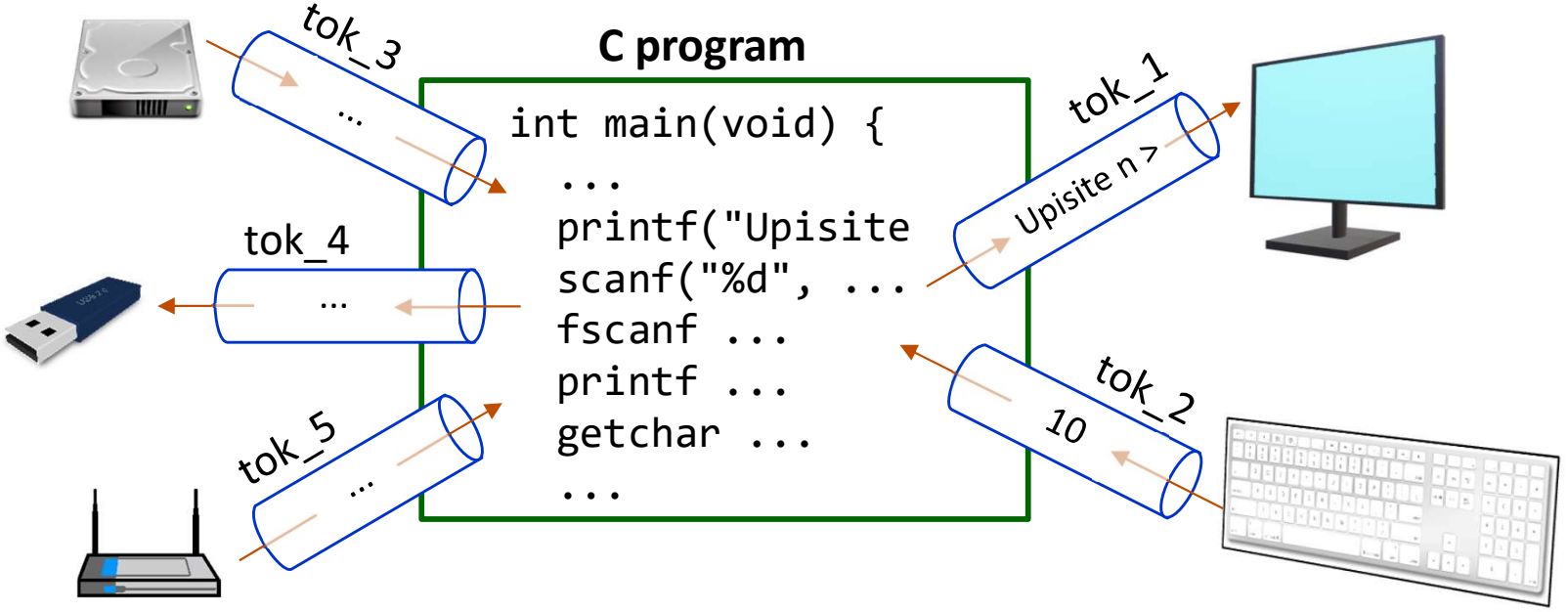
<stdio.h>

Input and Output

Tok (*stream*)

- aplikacijsko programsko sučelje za obavljanje ulaznih i izlaznih operacija (čitanje i pisanje podataka) temelji se na pojmu *tok* (*stream*)
- *tok* je apstraktni pojam: kanal ili veza programa s jednim od izvora podataka (*input source*) ili odredištem podataka (*output sink*)
  - program se može povezati (kolokvijalno: *otvoriti ulazni tok iz, otvoriti izlazni tok prema*) npr. sa sljedećim izvorima ili odredištima podataka
    - tipkovnica
    - zaslon
    - datoteka
    - računalna mreža (*network socket*)
    - drugi programi
  - dakle, čitanje iz izvora podataka obavlja se čitanjem iz ulaznog toka koji povezuje program i dotični izvor podataka (npr. datoteku)
    - simetrično vrijedi i za pisanje u odredište podataka

Tok (*stream*)



- velika prednost korištenja mehanizama *toka* je u tome što programera (uglavnom) oslobađa brige o tome koji se konkretni izvor ili odredište podataka, na kojem ulazno/izlaznom uređaju, koristi za čitanje ili pisanje podataka. Aplikacijsko programsko sučelje omogućuje
  - otvaranje i zatvaranje *toka* iz/prema nekom izvoru/odredištu podataka
  - korištenje funkcija za čitanje i pisanje (i ostalih operacija) na način koji (u principu) ne ovisi o vrsti izvora/odredišta podataka za koji je *tok* otvoren

Tok u programskom jeziku C

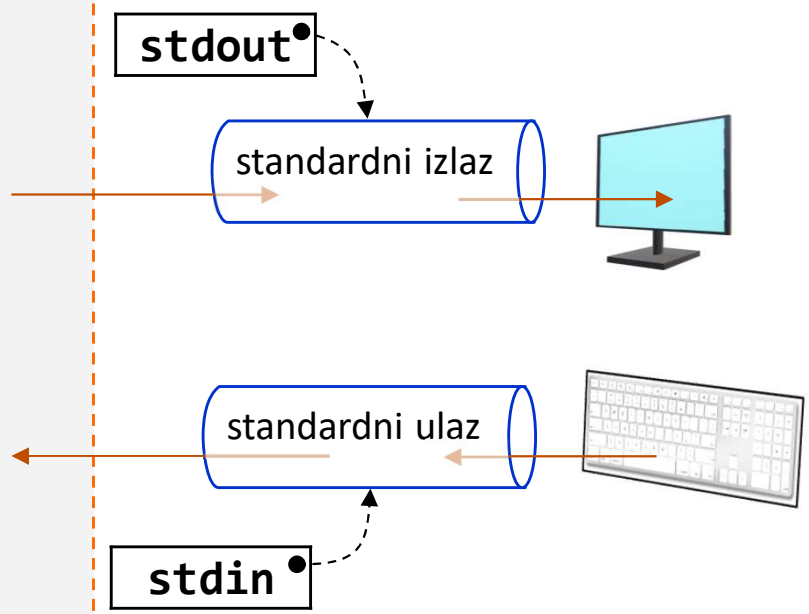
- U programskom jeziku C, *tok* je objekt tipa `FILE`
  - tip FILE je definiran u `<stdio.h>`. Konkretna implementacija tipa nije propisana standardom i za programera je potpuno nevažna
    - u programskom sučelju će se koristiti *pokazivači* na objekte tipa `FILE`
    - naziv tipa (pogrešno) asocira na datoteku (*file*) iz povijesnih razloga
  - za sada će se koristiti samo dva toka
    - *tok* iz tipkovnice (standardni ulaz) i *tok* prema zaslonu (standardni izlaz)
    - oba *toka* otvaraju se automatski, odmah po pokretanju programa
    - tokovi prema drugim izvorima/odredištima podataka (npr. datotekama) bit će objašnjeni kasnije
  - sljedeće globalne (eksterne) varijable tipa pokazivača na `FILE` mogu se koristiti u svim modulima s uključenom datotekom zaglavlja `<stdio.h>`
    - `stdin` (pokazivač na *tok* standardni ulaz)
    - `stdout` (pokazivač na *tok* standardni izlaz)

Primjer

```
#include <stdio.h>
int main(void) {
   ...
   printf("Upis n > ");

   ...

   scanf("%d", &n);
   ...
```



- `stdin` i `stdout` su globalne varijable
- inicijaliziraju se automatski u trenutku pokretanja programa
- funkcije `printf` i `scanf` po definiciji koriste tokove podataka standardni izlaz i standardni ulaz na koje pokazuju `stdin` i `stdout`

Varijante funkcija u `<stdio.h>`

- `printf` zadani sadržaj uvijek piše na *standardni izlaz*
  - `fprintf` ima dodatni parametar: pokazivač na *tok* u kojeg treba ispisati zadani sadržaj (može biti pokazivač na *standardni izlaz* ili na neki drugi *tok*)
  - `fprintf(stdout, "format", ...)` ≡ `printf("format", ...)`
- `scanf` sadržaj uvijek čita iz *standardnog ulaza*
  - `fscanf` ima dodatni parametar: pokazivač na *tok* iz kojeg treba čitati sadržaj (može biti pokazivač na *standardni ulaz* ili na neki drugi *tok*)
  - `fscanf(stdin, "format", ...)` ≡ `scanf("format", ...)`

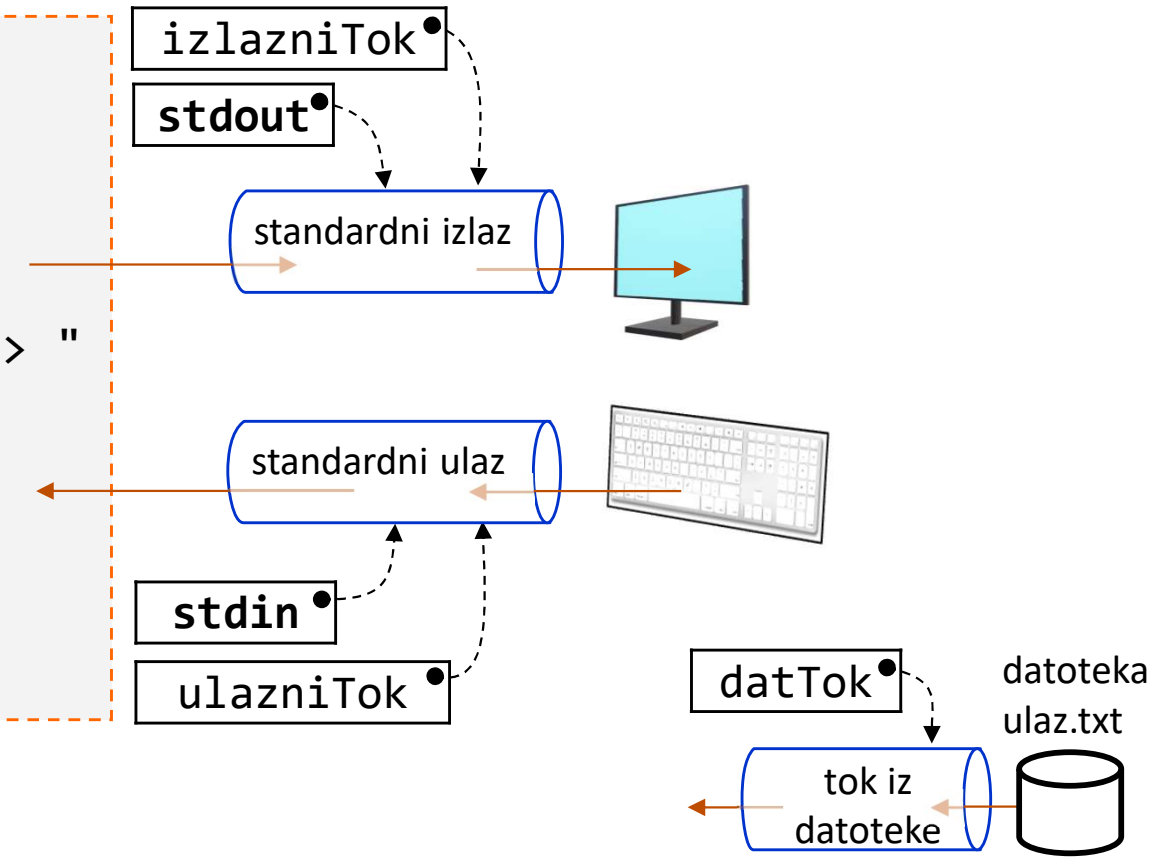
- i ostale funkcije iz `<stdio.h>` često se pojavljuju "u paru"
  - jedna funkcija koja po definiciji čita/piše na standardni ulaz/izlaz
  - jedna funkcija kojoj se pokazivač na *tok* zadaje kao argument/parametar

Primjer

```
#include <stdio.h>

int main(void) {
   FILE *izlazniTok = NULL;
   izlazniTok = stdout;
   fprintf(izlazniTok, "Upis n > "

   FILE *ulazniTok = NULL;
   ulazniTok = stdin;
   fscanf(ulazniTok, "%d", &n);
   ...
```



- zgodno je da se jednostavnom zamjenom vrijednosti pokazivača može promijeniti izvor ili odredište podataka
  - npr. u varijablu `ulazniTok` upisati pokazivač na *tok* iz datoteke "ulaz.txt"
  - kako otvoriti *tok* (osim standardnog ulaza i izlaza) i dobiti pokazivač na *tok* npr. iz datoteke ulaz.txt bit će objašnjeno u poglavlju o datotekama

getchar, getc <stdio.h>

```
int getchar(void);
int getc(FILE *stream);
```

- čitanje jednog znaka iz standardnog ulaza (`getchar`) ili zadanog ulaznog *toka* (`getc`)
  - `getchar()` ≡ `getc(stdin)`
- funkcija prvo čeka da se u ulaznom *toku* pojavi jedan ili više znakova
- zatim čita prvi po redu znak u ulaznom *toku*. Znak koji je pročitala smatra se *konzumiranim* i uklanja se iz ulaznog *toka*
  - ako je čitanje znaka iz ulaznog *toka* uspjelo, funkcija vraća ASCII vrijednost pročitanog znaka
  - ako čitanje znaka iz ulaznog *toka* nije uspjelo ili je pročitan znak koji označava kraj datoteke, funkcija vraća cjelobrojnu vrijednost `EOF`
    - `EOF` je makro definiran u `<stdio.h>`
    - znak koji označava kraj datoteke: `0x04` (`Ctrl-D`) na Unix , odnosno `0x1A` (`Ctrl-Z`) na Windows operacijskim sustavima

Primjer

- Programski zadatak
  - s tipkovnice čitati znak po znak (koristiti funkciju `getchar`) i za svaki znak, odmah po učitavanju znaka, na zaslon ispisati njegovu ASCII vrijednost po konverzijskoj specifikaciji %4d. Učitavanje znakova ponavljati sve dok se ne učita znak koji predstavlja oznaku kraja datoteke. Tada ispisati poruku "Kraj".
  - Primjeri izvršavanja programa

Windows

```
aAB↲
  97   65   66   10/↲
  47   10<Ctrl-Z>↲
Kraj
```

Unix/Linux

```
aAB↲
  97   65   66   10/↲
  47   10<Ctrl-D>Kraj
```

- Oznake `<Ctrl-Z>` i `<Ctrl-D>` znače da su na tipkovnici istovremeno pritisnute tipke `Ctrl` i `Z`, odnosno `Ctrl` i `D`
- na taj se način preko tipkovnice unose kontrolni znakovi koji predstavljaju oznaku kraja datoteke, `0x1A`, odnosno `0x04`

Rješenje

```
int c;
do {
   c = getchar();
   if (c != EOF) {
      printf("%4d", c);
   }
} while (c != EOF);
printf("Kraj");
```

```
int c;
while ((c = getchar()) != EOF) {
   printf("%4d", c);
}
printf("Kraj");
```

```
aAB↵
  97  65  66  10/↵
  47  10<Ctrl-Z>↵
Kraj
```

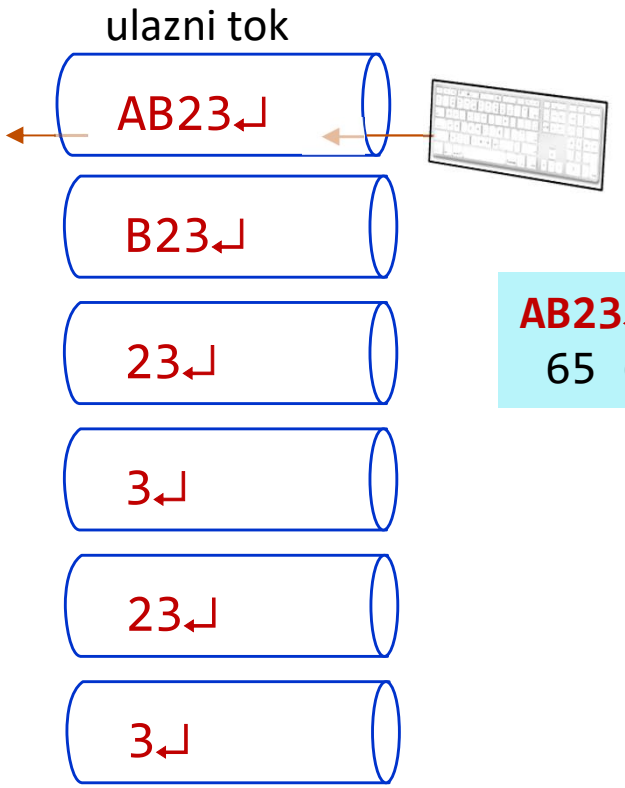
- zašto funkcija `getchar` ne pročita znak a istog trenutka kada je utipkan, nego tek kada je utipkan cijeli redak aAB↵?
  - znakovi koji se unose preko tipkovnice prvo se pohranjuju u *međuspremnik tipkovnice* (*buffer*). Tek kad se na tipkovnici pritisne tipka <Enter> sadržaj međuspremnika tipkovnice se dostavlja u *tok* standardni ulaz
  - taj mehanizam se naziva *line buffering*

ungetc <stdio.h>

```
int ungetc(int c, FILE *stream);
```

- vraćanje (*push back*) jednog upravo pročitanog znaka iz ulaznog toka natrag u ulazni tok
  - ako je vraćanje znaka uspjelo, funkcija vraća vrijednost parametra c
  - ako vraćanje znaka nije uspjelo, funkcija vraća EOF

```
printf("%3d", c = getchar());    65
printf("%3d", c = getchar());    66
printf("%3d", c = getchar());    50
ungetc(c, stdin);
printf("%3d", c = getchar());    50
```




```
AB23↵
 65 66 50 50
```

putchar, putc <stdio.h>

```
int putchar(int c);
int putc(int c, FILE *stream);
```

- ispis jednog znaka na standardni izlaz (`putchar`) ili zadani izlazni *tok* (`putc`)
  - `putchar(c)` ≡ `putc(c, stdout)`
  - ako je ispis uspio, funkcija vraća vrijednost parametra `c`
  - ako ispis nije uspio, funkcija vraća `EOF`

```
for (int i = 'A'; i <= 'Z'; ++i)
   putchar(i);
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

fgets <stdio.h>

```
char *fgets(char *s, int n, FILE *stream);
```

- u niz na kojeg pokazuje `s` čitaju se znakovi do kraja retka (ali ne više od `n - 1` znak) iz zadanog ulaznog toka. Iza zadnjeg učitanog znaka u niz se upisuje terminator niza `'\0'`.
  - ako je čitanje uspjelo, funkcija vraća vrijednost parametra `s`
  - ako čitanje nije uspjelo (zbog greške ili pokušaja čitanja znaka koji predstavlja kraj datoteke) funkcija vraća `NULL`

- zavisno od parametra `n` i duljine retka na ulazu, funkcija će pročitati oznaku novog retka
  - kako iz učitanog niza ukloniti eventualno učitani znak `'\n'`

```
fgets(niz, n, stdin);
char *nr = NULL;
if ((nr = strchr(niz, '\n')) != NULL)
   *nr = '\0';
```

Primjer

- Programski zadatak
  - napisati funkciju `void citajRedak(char *niz, int n, FILE *tok);`
  - u niz na kojeg pokazuje `niz` čitaju se svi znakovi prije kraja retka (ali ne više od `n - 1` znak) iz zadanog ulaznog toka te se niz terminira s `'\0'`. Znak za oznaku kraja retka, ako ga je bilo, treba ostati nepročitan u ulaznom toku. Zanemariti mogućnost pojave znaka koji označava kraj datoteke
  - Napisati glavni program za testiranje funkcije
  - Primjeri izvršavanja programa (za n = 10)

```
Duljina 10↵
Ucitan niz: |Duljina 1|↵
Na ulazu je znak: |0|
```

```
Duljina 9↵
Ucitan niz: |Duljina 9|↵
Na ulazu je znak: |↵
|
```

```
D↵
Ucitan niz: |D|↵
Na ulazu je znak: |↵
|
```

```
↵
Ucitan niz: ||↵
Na ulazu je znak: |↵
|
```

Rješenje (1. dio)

```
#include <stdio.h>

void citajRedak(char *niz, int n, FILE *tok) {
   int c;
   while (n > 1 && (c = getc(tok)) != '\n') {
      *niz++ = c;
      --n;
   }
   if (c == '\n') {
      ungetc(c, tok);
   }
   *niz = '\0';
}
```

Rješenje (2. dio)

```
#include <stdio.h>
#define N 10

void citajRedak(char *niz, int n, FILE *tok);

int main(void) {
   char niz[N];

   citajRedak(niz, N, stdin);

   printf("Ucitan niz: |%s|\n", niz);
   printf("Na ulazu je znak: |%c|", getc(stdin));

   return 0;
}
```

puts <stdio.h>

```
int puts(const char *s);
int fputs(const char *s, FILE *stream);
```

- ispis niza znakova na standardni izlaz (`puts`) ili zadani izlazni *tok* (`fputs`)
  - `puts(s)` ≠ `fputs(s, stdout)`
    - `puts` (za razliku od `fputs`) nakon ispisa niza dodatno ispisuje znak za novi red
  - ako je ispis uspio, funkcija vraća nenegativni broj
  - ako ispis nije uspio, funkcija vraća `EOF`

Primjer

- Programski zadatak
  - uzastopno učitavati i ispisivati retke teksta (učitati redak teksta iz standardnog ulaza, ispisati redak teksta na standardni izlaz). Učitavanje i ispis ponavljati dok god se ne upiše redak teksta u kojem se pojavljuje tekst `KRAJ`.
  - niti jedan redak teksta (uključujući oznaku novog retka) sigurno neće biti dulji od 20 znakova

```
The quick↵
The quick↵
brown fox jumps↵
brown fox jumps↵
nigdje KRAJA↵
```

Rješenje

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

#define MAXNIZ 20

int main(void) {
   char niz[MAXNIZ + 1];
   while (strstr(fgets(niz, MAXNIZ + 1, stdin), "KRAJ") == NULL) {
      fputs(niz, stdout);
   }

   return 0;
}
```

- Za vježbu analizirati:
  - koji bi se rezultat dobio kada bi se makro `MAXNIZ` promijenio na `10`?
  - zašto se program uz `MAXNIZ=10` neće zaustaviti ako se upiše redak `nigdje KRAJA↵`, a zaustavit će se ako se umjesto tog retka upiše redak `ima KRAJA↵`

Prije sljedećeg predavanja

- Edgar tutorial: **nema**
- **21. vježbe uz predavanja**