Uvod u programiranje
- predavanja -

studeni 2025.

8. Tipovi podataka u programskom jeziku C

Vodič „07 prije osmog predavanja”

- Očekuje se da ste savladali (na razini na kojoj je obrađeno u vodiču):
  - osnovni tipovi podataka - int

- Pitanja?

Osnovni tipovi podataka

Osnovni tipovi podataka

- Tip podatka varijable određuje karakteristike podatka koji u tu varijablu može biti pohranjen i operacije koje se nad podatkom mogu obaviti
  - U programskom jeziku C na raspolaganju su sljedeći osnovni tipovi podataka
    - cjelobrojni tipovi (*integer types*)
      - char
      - int
      - _Bool
    - brojevi s pomičnim zarezom (*floating point types*)
      - float
      - double
  - neobaveznim kvalifikatorima (short/long, signed/unsigned) opisuju se dodatne karakteristike nekih od navedenih tipova

Cjelobrojni tipovi podataka

Tip podatka char

Tip podatka char

- cjelobrojni tip podatka koji se koristi za pohranu malih brojeva
- izgovor za tip podatka *char*: tʃaː ili (rjeđe) 'kær
  - u engleskom govornom području koriste se oba oblika
  - kolokvijalno ćemo koristiti naziv *karakter* ili *znak*
- standardom se od tipa podatka char zahtijeva mogućnost pohrane **cijelih brojeva** u prostoru za pohranu od najmanje jednog bajta
  - obično to upravo jest jedan bajt (npr. za gcc i arhitekturu x86_64)
- dopušteni rasponi
  - signed: [-128, 127]
  - unsigned: [0, 255]

Upotreba tipa podatka char za prikaz znakova

- Znakovi se ne mogu pohraniti u računalu!
  - moguće je pohraniti binarne brojeve koji predstavljaju unaprijed dogovorene kodove znakova, npr.

Znak	binarno	dekadski
A	01000001	65
B	01000010	66

  - u upotrebi su razne kodne stranice
    - ASCII: sadrži 128 različitih znakova i upravljačkih znakova kodiranih vrijednostima 0-127
    - ISO-8859: kodovi 0-127 kao u 7-bitnom kodu, dok se kodovi 128-255 koriste za razne dodatne znakove, ovisno o kodnoj stranici, npr.
      - ISO 8859-1: dodatni znakovi zapadnoeuropskih jezika
      - ISO 8859-2: dodatni znakovi istočnoeuropskih jezika
        - npr. š = 185_{10}, Č = 200_{10}
    - Na ovom predmetu koristit će se kodna stranica ASCII

ASCII - American Standard Code for Information Interchange

ISO - International Organization for Standardization

Unicode

- Noviji industrijski standard za kodiranje znakova
  - Koristi više bajtova za kodiranje znakova i omogućuje predstavljanje gotovo svih pisama korištenjem jedinstvenog skupa znakova
  - The Unicode Consortium: www.unicode.org

  - Primjer: kod za znak LA iz pisma Tagalog (Filipini) jest $170E_{16}$

170E

ASCII tablica kodova znakova (1)

Dec. broj	C konst.	Znak
0	'\0'	Nul znak (NULL)
1		početak zaglavlja (SOH)
2		početak teksta (STX)
3		kraj teksta (ETX)
4		kraj prijenosa (EOT)
5		kraj upita (ENQ)
6		Potvrda (ACK)
7	'\a'	Alarm (BEL)
8	'\b'	Backspace (BS)
9	'\t'	vodoravni tabulator (HT)
10	'\n'	sljedeći red/novi red (LF)
11	'\v'	okomiti tabulator (VT)
12	'\f'	nova stranica (FF)
13	'\r'	skok na početak reda (CR)
14		pomak van (SO)
15		pomak unutra (SI)

Dec. broj	Znak
16	znak prekida veze (DLE)
17	provjera uređaja 1 (DC1)
18	provjera uređaja 2 (DC2)
19	provjera uređaja 3 (DC3)
20	provjera uređaja 4 (DC4)
21	negativna potvrda (NAK)
22	sinkrono mirovanje (SYN)
23	kraj prijenosnog bloka (ETB)
24	otkaži (CAN)
25	kraj medija (EM)
26	Zamjena (SUB)
27	Escape (ESC)
28	razdjelnik datoteka (FS)
29	razdjelnik grupe (GS)
30	razdjelnik zapisa (RS)
31	razdjelnik jedinice (US)

ASCII tablica kodova znakova (2)

Dec. broj	Znak
32	razmak
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/

Dec. broj	Znak
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@

Dec. broj	Znak
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O

Dec. broj	Znak
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	w
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_

ASCII tablica kodova znakova (3)

Dec. broj	Znak
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o

Dec. broj	Znak
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	DEL

Znakovi za upravljanje ulazno-izlaznim jedinicama računala (kontrolni znakovi, *non-printable*) nalaze se na pozicijama 0-31 i na poziciji 127 (DEL)

Znakovi koji se mogu tiskati (*printable*) nalaze se na pozicijama 32-126

Konverzijske specifikacije za printf i scanf

- u slučaju kada se podatak tipa char prikazuje kao broj

Tip	signed	unsigned
char	%hhd	%hhu

- **d** ili **u** se mogu zamijeniti s **x**, **X**, **o** radi ispisa u heksadekadskom ili oktalnom obliku

```
signed char a = -10;
unsigned char b = 255;
printf("%hhd %hhu", a, b);          -10 255
printf("%hho %hhx", a, b);          366 ff
```

- u većini primjena dopušteno je kod ispisa koristiti format %d jer se odgovarajuće konverzije char ↔ int obavljaju automatski

```
char a = 65;
printf("%d", a);                    65
```

Konverzijske specifikacije za printf i scanf

- kada se podatak tipa char prikazuje ili čita kao znak, koristi se konverzijska specifikacija %c

```
char a, b, c, d;

a = 65;
printf("%hhd %c", a, a);              65 A

b = a + 3;
printf("%hhd %c", b, b);              68 D

scanf("%c", &c);                      za ulaz  7
printf("%hhd %c", c, c);              55 7

scanf("%c", &d);                      za ulaz  E
printf("%hhd %c", d, d);              69 E
```

Konstante

- konstanta tipa char ne postoji. Koriste se konstante tipa int
  - prikladno je koristiti poseban oblik pisanja konstante tipa int
    - znak (koji se može ispisati) iz ASCII tablice napisan pod jednostrukim navodnicima, npr.

      `'B'`
    - ovako napisana konstanta tipa int zauzima 4 bajta i ima vrijednost 00000042_{16}, odnosno 66_{10}
    - sljedeća dva programska odsječka će dati isti rezultat:

```
char znak;
znak = 66;
```

```
char znak;
znak = 'B';
```

    - u oba slučaja sadržaj konstante veličine 4 bajta bit će upisan u sadržaj varijable znak veličine jednog bajta (pri čemu se odbacuju tri bajta sa značajnijim bitovima, koji ionako sadrži samo nule)

Primjeri cjelobrojnih konstanti

C program	Hex.	Dek.	U ASCII tablici odgovara kodu znaka:
`'A'`	`0x41`	65	veliko slovo A
`'0'`	`0x30`	48	znamenka nula

- znakove koji u sintaksi imaju posebno značenje ili se ne mogu jednostavno prikazati jednim znakom, prikazuju se pomoću tzv. "*escape sequence*"
  - niz od dva ili više znakova koji započinju znakom \ koji nagovještava "specijalno" značenje znakova koji slijede

C program	Hex.	Dek.	U ASCII tablici odgovara kodu znaka:
`'\a'`	`0x07`	7	alarm (*bell, beep*), aktivira zvuk na terminalu
`'\t'`	`0x09`	9	vodoravni tabulator
`'\n'`	`0x0A`	10	skok u novi red
`'\0'`	`0x00`	0	nul-znak, oznaka kraja niza
`'\\'`	`0x5C`	92	obrnuta kosa crta (*backslash*)
`'\''`	`0x27`	39	jednostruki navodnik
`'\"'`	`0x22`	34	dvostruki navodnik

Pojednostavljenje načina izražavanja

```
char c1;
c1 = 'E';
```

- Radi pojednostavljenja, koristit će se kolokvijalni izrazi:
  - *'E' je znakovna konstanta*
    - iako je poznato da se radi o cjelobrojnoj (int) konstanti vrijednosti 69_{10}
  - *varijabla c1 je znakovnog tipa*
    - iako je poznato da je cjelobrojnog tipa (ali očito će se koristiti za pohranu ASCII koda znaka)
  - *u varijablu c1 upisan je znak E*
    - iako je poznato da je u varijablu upisan ASCII kod znaka E, odnosno cijeli broj 69_{10}

Primjer

```
#include <stdio.h>

int main(void) {
   char x = 'A', y = x + 32, z = '\n';
   printf("%hhd %c\n", x, x);
   printf("%hhd %c\n", y, y);
   printf("%hhd %c\n", z, z);
   printf("%d %c\n", '0' + 2, '0' + 2);
   printf("%d %c\n", '0' + '2', '0' + '2');

   return 0;
}
```

```
65 A↵
97 a↵
10 ↵
↵
50 2↵
98 b↵
```

Primjer

```
#include <stdio.h>

int main(void) {
   char c = 'D';
   printf ("%c %d\n", c, c);
   printf ("%c\n", c + 32);
   printf ("%d\n", 'C' - 'A');

   return 0;
}
```

```
D 68↵
d↵
2↵
```

Znamenke 0 - 9 u ASCII tablici

- Treba obratiti pažnju na to da ASCII kodovi znamenki 0-9 ne odgovaraju numeričkim vrijednostima znamenki

```
char a = '1';
```
u varijabli je pohranjena numerička vrijednost 49

- ako se na temelju koda znamenke želi dobiti njezina numerička vrijednost, potrebno je od te vrijednosti oduzeti 48, jer vrijednost 48 predstavlja kod znaka '0'

```
char c;
scanf("%c", &c);                          za ulaz  7
printf("%c %hhd %d", c, c, c - 48);       7 55 7
```
ili
```
printf("%c %hhd %d", c, c, c - '0');      7 55 7
```

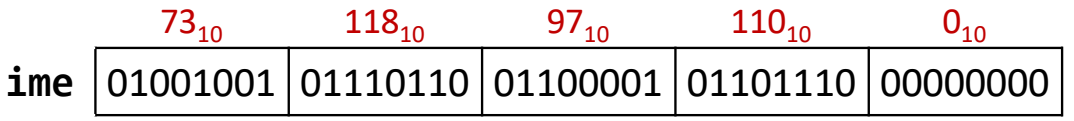
Nizovi znakova

Niz znakova - *string*

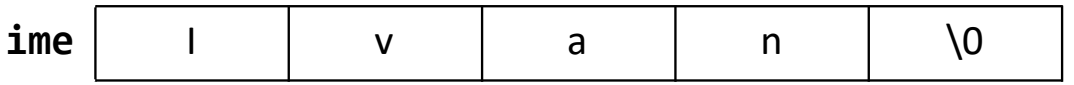
- U programskom jeziku C ne postoji osnovni tip podatka koji podržava rad s nizovima znakova
- za pohranu niza znakova koristi se jednodimenzijsko polje čiji su članovi tipa char, pri čemu se kraj niza obavezno označava članom polja koji sadrži nul-znak '\0' (kod iz ASCII tablice numeričke vrijednosti 0)

```
char ime[5];

ime[0] = 'I';
ime[1] = 'v';
ime[2] = 'a';
ime[3] = 'n';
ime[4] = '\0';


printf("%s", ime);
```

Ivan



- radi pojednostavljenja, na slikama će se umjesto kodova znakova prikazivati znakovi

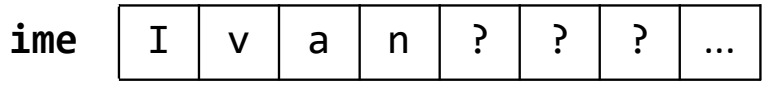


- kao konverzijsku specifikaciju koristiti %s
- kao vrijednost koju treba ispisati koristiti naziv polja u kojem je niz znakova pohranjen

Čemu služi terminator niza '\0' ?

- Pomoću znaka '\0' može se zaključiti gdje se nalazi kraj niza znakova. Npr.
  - funkcija printf prema konverzijskoj specifikaciji %s ispisuje znakove iz zadanog niza znakova, znak po znak, sve dok ne dođe do člana polja u kojem se nalazi vrijednost '\0'
  - ako kraj niza nije ispravno označen znakom '\0', funkcija će nastaviti ispisivati znakove čiji ASCII kodovi odgovaraju vrijednostima koji se nalaze u memoriji iza kraja niza

```
char ime[4];

ime[0] = 'I';
ime[1] = 'v';
ime[2] = 'a';
ime[3] = 'n';

printf("%s", ime);
```



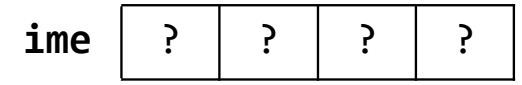
- ispis će se nastaviti sve dok se negdje u memoriji ne naiđe na bajt u kojem je upisana vrijednost nula

```
Ivan*)%&/!)=()Z)(B#DW=)$/")#*'@!"/...
```

Definicija niza znakova uz inicijalizaciju

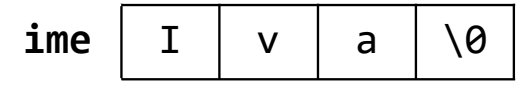
- Jednako kao kod polja ostalih tipova podataka:
  - bez inicijalizacije, članovi polja sadrže nedefinirane vrijednosti

```
char ime[4];
```



  - početne vrijednosti se mogu redom navesti u tzv. inicijalizatoru

```
char ime[4] = {'I', 'v', 'a', '\0'};
```



  - ili

```
char ime[] = {'I', 'v', 'a', '\0'};
```

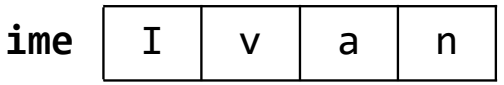
  - ili

```
char ime[4] = {'I', 'v', 'a'};
```

Definicija niza znakova uz inicijalizaciju

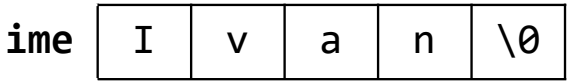
- Obratiti pažnju
  - ovakav način inicijalizacije jednodimenzijskog polja nije ispravan ako se njegov sadržaj namjerava koristiti kao niz znakova (string)

```
char ime[4] = {'I', 'v', 'a', 'n'};
```



  - ispravno:

```
char ime[4+1] = {'I', 'v', 'a', 'n'};
```



Jednostavniji način inicijalizacije niza znakova

- Umjesto:

```
char ime[4] = {'I', 'v', 'a', '\0'};
```

  - može se (i bolje je) koristiti drugačiji oblik inicijalizatora

```
char ime[4] = "Iva";
```

znak **\0** će biti dodan, programer **mora** osigurati prostor za barem jedan znak više!

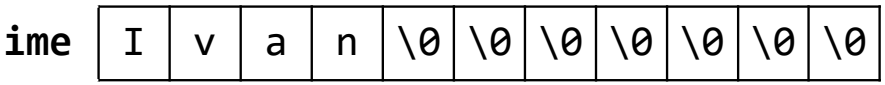
  - ili

```
char ime[] = "Iva";
```

potrebnu veličinu polja odredit će prevodilac, znak **\0** će biti dodan

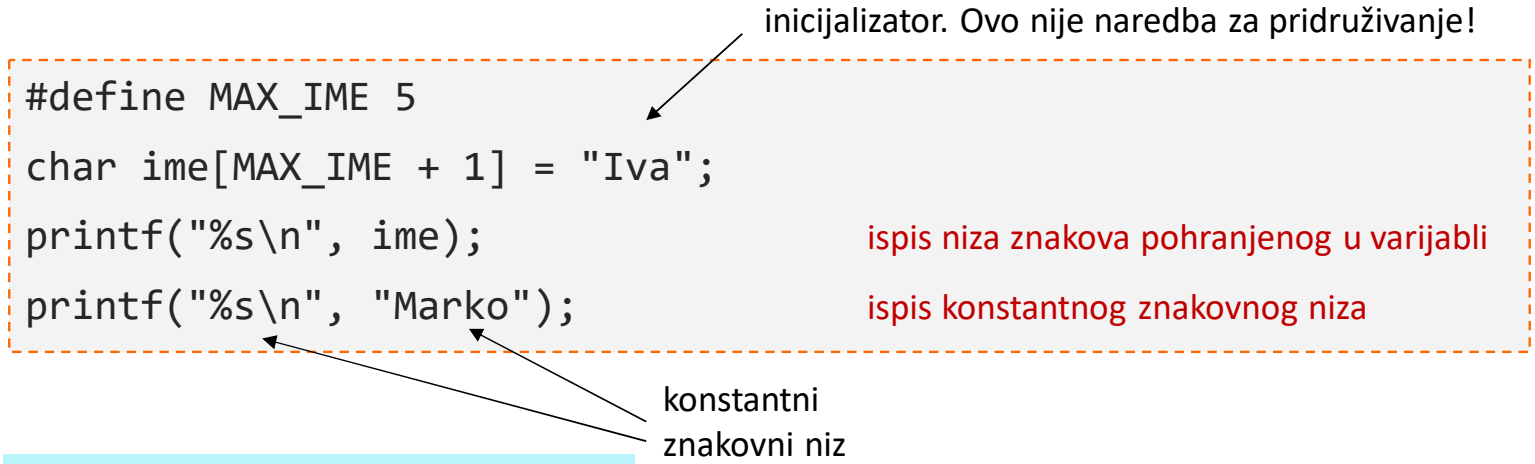
  - uobičajeno se nizovi znakova definiraju uz pomoć simboličkih konstanti. Pri tome, dobra je praksa koristiti notaciju "+ 1". Time se naglašava da nije zaboravljen prostor za znak kojim se terminira niz

```
#define MAX_IME 10
char ime[MAX_IME + 1] = "Ivan";
```



Konstantni znakovni niz

- U programu se konstantni znakovni niz označava dvostrukim navodnicima



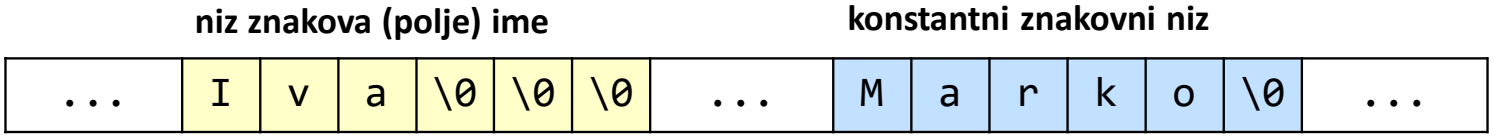

```
char ime[10 + 1];
ime = "Iva";
```

Nije dopušteno! Polje je ***non-modifiable*** *lvalue*!

Konstantni znakovni niz

```
#define MAX_IME 5
char ime[MAX_IME + 1] = "Iva";
printf("%s\n", ime);
printf("%s\n", "Marko");
```

- konstantni znakovni niz "Marko" u memoriji je pohranjen slično nizu znakova ime



- ali postoji bitna razlika: sadržaj niza znakova može se mijenjati

```
ime[1] = 'd';
printf("%s\n", ime);
```

```
Ida↵
```

Konstantni znakovni niz

- Konstantni znakovni niz se u programima ne smije lomiti kroz više redaka

```
printf("%s", "Fakultet
 u Unskoj 3");
```

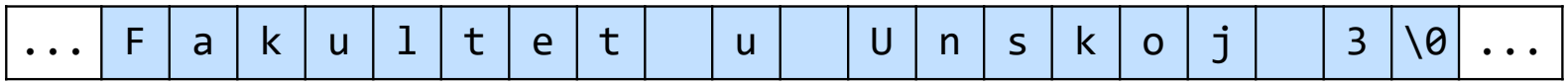
prevodilac dojavljuje pogrešku

- ako je niz predugačak da bi se mogao pregledno napisati u jednom retku, treba koristiti jednostavno nadovezivanje

```
printf("%s", "Fakultet"
            " u Unskoj 3");
```

- konstantni znakovni niz napisan u dva ili više dijelova, u memoriji će biti pohranjen na isti način kao da je napisan u jednom dijelu

```
printf("%s", "Fakultet u Unskoj 3");
```



Konverzijska specifikacija %s za printf

- Za ispis niza znakova (također i konstantnog znakovnog niza) koristi se konverzijska specifikacija %s

```
printf("Ime: %s\nPrezime: %s\nAdresa:\n%s\n",
       "Ana",
       "Novak",
       "Ilica 1\n10000 Zagreb\\Centar");
```

```
Ime: Ana↵
Prezime: Novak↵
Adresa:↵
Ilica 1↵
10000 Zagreb\Centar↵
```

Konverzijska specifikacija %s za scanf

- Konverzijska specifikacija %s omogućuje čitanje niza znakova do pojave prve praznine, oznake novog retka ili tabulatora

```
char ime[10 + 1], prezime[10 + 1];
scanf("%s %s", ime, prezime);
printf("%s, %s", prezime, ime);
```

```
Anamarija Horvat Novak↵
Horvat, Anamarija
```

- Čitanje konverzijskom specifikacijom %s na kraj niza ugrađuje '\0'
- Čitanje niza znakova pomoću funkcije scanf i konverzijske specifikacije %s je potencijalno opasno
  - što će se dogoditi ako korisnik upiše predugo ime ili prezime (u odnosu na duljinu niza znakova navedenu u definiciji)

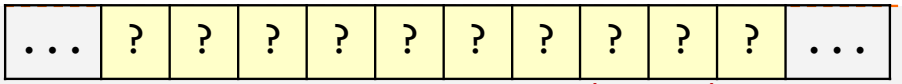
```
Anamarija Horvat-Novak↵
```

Učitavanje niza znakova na siguran način

- Ako je potrebno onemogućiti unos predugog niza znakova ili je potrebno omogućiti čitanje niza znakova koji sadrži praznine ili tabulatore, tada za čitanje niza znakova s tipkovnice umjesto funkcije scanf treba koristiti funkciju **fgets**
  - funkcija fgets će detaljnije biti objašnjena kasnije. Ovdje je naveden skraćeni opis, koji će biti dovoljan za njeno ispravno korištenje
  - funkcija fgets prima tri argumenta
    - ime polja (varijabla tipa niza znakova) `niz` u koje s tipkovnice treba učitati niz znakova
    - najveći dopušteni broj znakova n koje funkcija fgets smije upisati u to polje (za vrijeme dok učitava znakove s tipkovnice)
    - *tok podataka* iz kojeg se učitavaju znakovi. Za sada uvijek napisati `stdin` (skraćenica dolazi od *standard input* - vrlo pojednostavljeno: tipkovnica). Detaljnije objašnjenje slijedi u poglavlju o datotekama

Učitavanje niza znakova

```
char tekst[10];
fgets(tekst, 10, stdin);
```



- u zadani niz učitava znakove s tipkovnice sve dok ne pročita oznaku novog retka ili učita n-1 znakova (u ovom primjeru 9). Iza zadnjeg učitanog znaka u niz dodaje znak '\0'
  - zašto se ovoj funkciji mora zadati najveći dopušteni broj znakova?

↵	... \n \0 ? ? ? ? ? ? ? ? ...
Pas↵	... P a s \n \0 ? ? ? ? ? ...
Pet pasa↵	... P e t ␣ p a s a \n \0 ...
Dva macka↵	... D v a ␣ m a c k a \0 ...
Jedanaest pasa↵	... J e d a n a e s t \0 ...

Primjer

- Programski zadatak
  - s tipkovnice učitati niz znakova iz jednog retka. Niz znakova, uključujući oznaku novog retka (ako bude učitana), ne smije biti dulji od 10 znakova.
  - ako učitani niz sadrži znak za prelazak u novi redak, izbaciti ga iz niza. Sva mala slova u nizu pretvoriti u velika. Ispisati novi sadržaj niza i odmah iza kraja niza uskličnik.

```
Upisite niz znakova > Kvaka 22↵
KVAKA 22!
```

```
Upisite niz znakova > Put u svemir↵
PUT U SVEM!
```

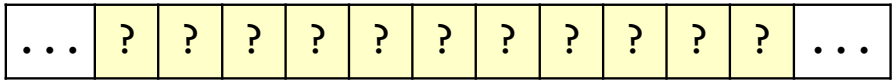
```
Upisite niz znakova > ↵
!
```

Rješenje

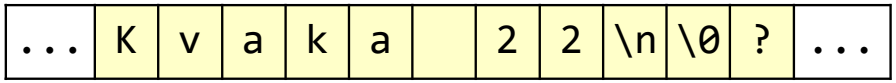
```
#include <stdio.h>
#define MAX_NIZ 10

int main(void) {
   char niz[MAX_NIZ + 1];
   int i = 0;
   printf("Upisite niz znakova > ");
   fgets(niz, MAX_NIZ + 1, stdin);
   while (niz[i] != '\0') {
      if (niz[i] >= 'a' && niz[i] <= 'z') {
         niz[i] = niz[i] - ('a' - 'A');
      } else if (niz[i] == '\n') {
         niz[i] = '\0';
      }
      i = i + 1;
   }
   printf("%s!", niz);
   return 0;
}
```

Objašnjenje

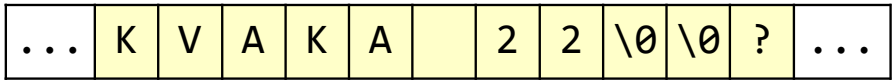
```
char niz[10 + 1];
```



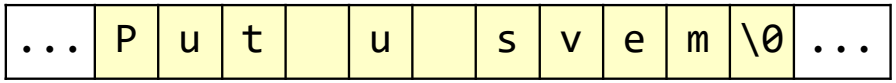

```
fgets(niz, 11, stdin);
```



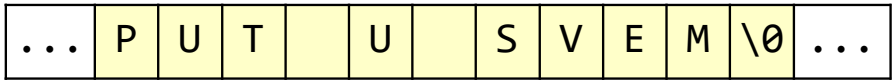
nakon petlje while



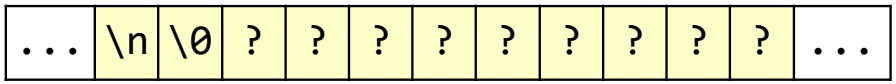

```
fgets(niz, 11, stdin);
```



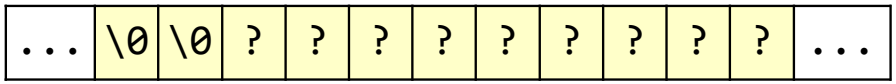
nakon petlje while




```
fgets(niz, 11, stdin);
```



nakon petlje while



Zadatak

- **Zašto QWERTY?**

The keyboard we use today was created by Christopher Latham Sholes, an American inventor, in the 19th century. Initially, he designed the keyboard alphabetically for typists' convenience in accessing keys. However, typists frequently used letter combinations, leading to key clashes and jams, prompting Christopher Sholes to reorganize the keyboard. He separated commonly used letter pairings, ultimately resulting in the QWERTY keyboard design. The name is derived from the order of the six keys on the top row of the keyboard (Q W E R T Y). This layout change facilitated faster typing and reduced jamming errors on typewriters. The design was incorporated into typewriters by 1874 and has since become the ubiquitous standard.

https://www.microsoft.com/en-us/microsoft-365-life-hacks/writing/why-our-keyboard-layouts-are-the-way-they-are#:~:text=The%20name%20is%20derived%20from,since%20become%20the%20ubiquitous%20standard.

Procijenite učestalost slova u engleskom jeziku tako što ćete prebrojiti koliko puta se koje slovo pojavljuje u svim Shakespearovim djelima.

Prikažite grafički, s pomoću stupčastog grafa (histogram).

Prije sljedećeg predavanja

- Edgar:
  - Tutorial: **08. Prije devetog predavanja**
  - **8. vježbe uz predavanja**