Uvod u programiranje

- predavanja -

listopad 2020.

Tipovi podataka u programskom jeziku C

- 1. dio -

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
  - opcionalnim kvalifikatorima (short/long, signed/unsigned) opisuju se dodatne karakteristike nekih od navedenih tipova

Potrebno (i očekivano) predznanje

- dekadski, binarni, oktalni, heksadekadski brojevni sustavi
  - pretvorbe cijelih brojeva između bilo kojih od navedenih brojevnih sustava: $27_{10} = 11011_2 = 33_8 = 1B_{16}$
- način pohrane pozitivnih i negativnih cijelih brojeva u računalu
- zbrajanje pozitivnih i negativnih brojeva u binarnom brojevnom sustavu
- raspon brojeva koji se mogu pohraniti u registru od n bitova
  - ako se pohranjuju samo pozitivni brojevi

    $[0, 2^n - 1]$
  - ako se pohranjuju i pozitivni i negativni brojevi

    $[-(2^{n-1}), 2^{n-1} - 1]$

- Zašto je za programera važno poznavati način pohrane podataka?

Primjer

- Napisati program koji s tipkovnice učitava cijeli broj n, a zatim na zaslon ispisuje rezultat operacije n + 5

```
#include <stdio.h>

int main(void) {
   int n, m;
   scanf("%d", &n);
   m = n + 5;
   printf("%d", m);
   return 0;
}
```

za n = 2 000 000 000

```
2000000000↵
2000000005
```

✓

za n = -2 000 000 000

```
-2000000000↵
-1999999995
```

✓

za n = 2 147 483 647

```
·2147483647↵
-2147483644
```

?

Primjer

- Napisati program koji s tipkovnice učitava realni broj x, a zatim na zaslon ispisuje rezultat operacije x+0.125, u ukupnoj širini od 15 znakova, s pet znamenki iza decimalne točke

```
#include <stdio.h>

int main(void) {
   float x, y;
   scanf("%f", &x);
   y = x + 0.125f;
   printf("%15.5f", y);
   return 0;
}
```

za x = 2 097 151.0

```
··2097151.0↵
··2097151.12500
```

✓

za x = -2 097 151.0

```
·-2097151.0↵
·-2097150.87500
```

✓

za x = 2 097 152.0

```
··2097152.0↵
··2097152.00000
```

?

za x = 1 000 000.3

```
··1000000.3↵
··1000000.43750
```

?

Osnovni tipovi podataka

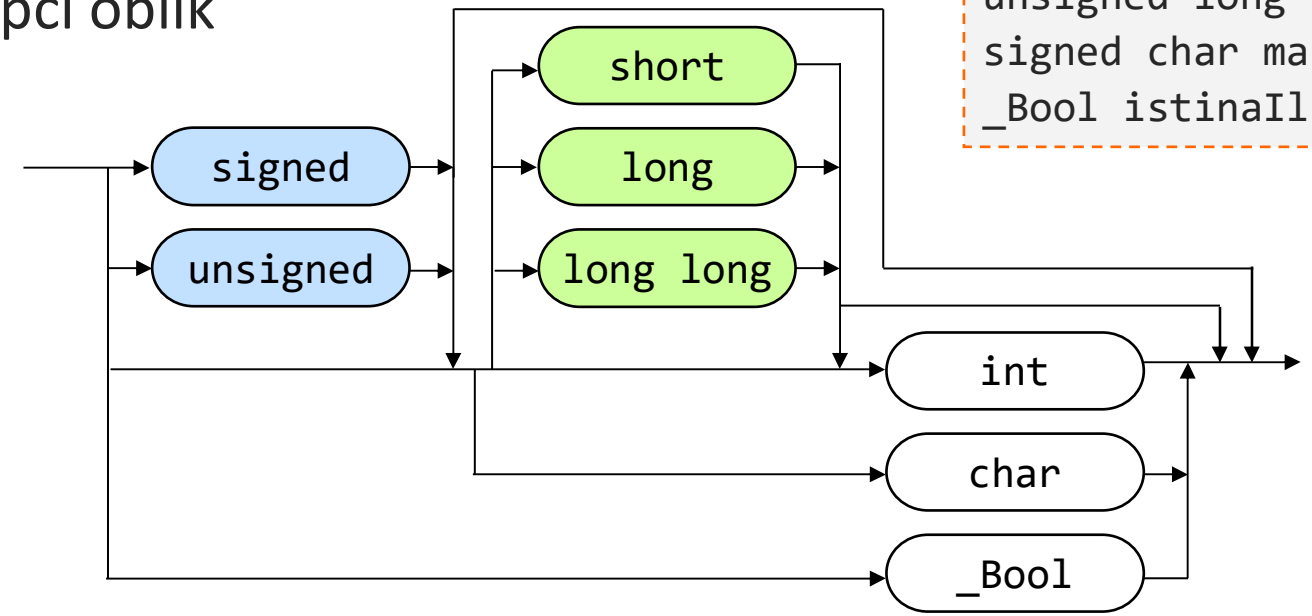
Cjelobrojni tipovi podataka

Cjelobrojni tipovi podataka

Tip podatka int

Cjelobrojni tipovi

- *integer types*
- opći oblik



Primjer definicije varijabli

```
unsigned long int velikiBroj;
signed char maliBroj;
_Bool istinaIliLaz;
```

- Prefiksi (kvalifikatori):
  - signed/unsigned: određuju mogu li se u varijablu tipa int ili char pohraniti samo pozitivni ili i negativni brojevi
  - short/long/long long: smanjuju/povećavaju raspon brojeva koji se mogu pohraniti u varijablu tipa int

Tip podatka int (*integer*) - varijante

- kvalifikator *signed* i ključna riječ *int* se podrazumijevaju (*default*), stoga se mogu ispustiti

Puni naziv tipa	Sinonimi	Preporučeni oblik u programima
signed int	int, signed	**int**
unsigned int	unsigned	**unsigned int**
signed short int	signed short, short int, short	**short**
unsigned short int	unsigned short	**unsigned short**
signed long int	signed long, long int, long	**long**
unsigned long int	unsigned long	**unsigned long**
signed long long int	signed long long, long long int, long long	**long long**
unsigned long long int	unsigned long long	**unsigned long long**

Kvalifikatori short, long i long long

- Tip podatka int, s obzirom na dopušteni raspon brojeva koji se mogu pohraniti (određen brojem binarnih znamenki), može se dodatno kvalificirati kao short, long ili long long
- Raspon brojeva za pojedine tipove nije propisan jezikom, ali vrijede pravila:
  - short: minimalno 2 bajta i ne smije imati raspon veći od int
  - int: minimalno 2 bajta i ne smije imati raspon veći od long
  - long: minimalno 4 bajta i ne smije imati raspon veći od long long
  - long long: minimalno 8 bajtova
- Stvarni rasponi brojeva ovisi o implementaciji prevodioca i arhitekturi računala

Primjer

- gcc i arhitektura x86_64

Tip	Bajtova	Raspon brojeva koje je moguće pohraniti
signed short int	2	-32 768, 32767
signed int	4	-2 147 483 648, 2 147 483 647
signed long int	4	-2 147 483 648, 2 147 483 647
signed long long int	8	-9 223 372 036 854 775 808, 9 223 372 036 854 775 807

Kvalifikatori signed i unsigned

- Vrijednost (binarne znamenke) pohranjena u varijablu tipa signed smatra se pozitivnom ili negativnom, ovisno o vrijednosti najznačajnijeg bita
- Vrijednost pohranjena u varijablu tipa unsigned, bez obzira na vrijednost najznačajnijeg bita, uvijek se smatra pozitivnom
  - time se udvostručuje najveća pozitivna vrijednost koja može biti pohranjena
- Primjer: ako varijabla tipa short koristi 2 bajta
  - raspon brojeva u signed varijabli: [-32 768, 32 767]
  - raspon brojeva u unsigned varijabli: [0, 65 535]

Primjer

- Rezultat operacije u kojoj se koristi signed ili unsigned int

```
unsigned int ua = 2147483647;
ua = ua + 1;
if (ua > 0)
   printf("veci od nule\n");
else
   printf("manji od nule");


signed int a = 2147483647;
a = a + 1;
if (a > 0)
   printf("veci od nule\n");
else
   printf("manji od nule\n");
```

01111111111111111111111111111111

10000000000000000000000000000000

promatra se isključivo kao pozitivni broj

ispis: veci od nule

01111111111111111111111111111111

10000000000000000000000000000000

promatra se kao pozitivni ili negativni broj

ispis: manji od nule

Primjer

- prevodilac gcc i arhitektura x86_64

Tip	Bajtova	Raspon brojeva koje je moguće pohraniti
unsigned short int	2	0, 65535
unsigned int	4	0, 4 294 967 295
unsigned long int	4	0, 4 294 967 295
unsigned long long int	8	0, 18 446 744 073 709 551 615

Kvalifikatori signed i unsigned

- savjet: ne miješati signed i unsigned tipove
  - usložnjavaju se konverzijska pravila
  - rezultat operacija može biti ovisan o implementaciji prevodioca (rezultat: neprenosivi programi)
- najčešće nije potrebno koristiti unsigned tip podatka. Na ovom predmetu će se koristiti u rijetkim prilikama, npr.
  - kada se ciljano želi povećati mogućnost pohrane (približno dvostruko) većih pozitivnih brojeva, na štetu mogućnosti pohrane negativnih brojeva
  - kada se obavljaju operacije nad bitovima, naročito u slučaju operacije posmaka bitova
    - operacije nad bitovima će biti objašnjene kasnije

Cjelobrojne konstante

- konstante se u memoriji računala pohranjuju na isti način kao i varijable
  - broj bajtova koji se koristi za pohranu i interpretacija konstante kao isključivo pozitivnog ili i negativnog broja ovisi o tipu konstante

Tip	Primjeri konstanti
signed short int	ne postoji konstanta tipa signed short, koristiti konstantu tipa int
signed int	2000000000, -2000000000
signed long int	2000000000**L**, -2000000000**L**
signed long long int	-5000000000**LL**, 5000000000**LL**
unsigned short int	ne postoji konstanta tipa unsigned short
unsigned int	2000000000**U**
unsigned long int	2000000000**UL**
unsigned long long int	5000000000**ULL**

  - umjesto sufiksa U, L ili LL može se koristiti u, l ili ll (mala slova U i L)

Cjelobrojne konstante

- cjelobrojne konstante se najčešće pišu u dekadskom brojevnom sustavu, ali u programu ih je moguće zapisati i u oktalnom ili heksadekadskom sustavu
  - oktalna konstanta započinje prefiksom 0 i sadrži znamenke 0-7
  - heksadekadska konstanta započinje prefiksom 0x ili 0X i sadrži znamenke 0-9 i A-F ili a-f

```
printf("%d", 17);              17
printf("%d", 017);             15
printf("%d", 019);             prevodilac dojavljuje pogrešku!
printf("%d", 0x17);            23
printf("%d", 0X1a);            26
printf("%d", 0x7FFFFFFF);      2147483647
printf("%d", 0xFFFFFFFF);      -1
```

Cjelobrojne konstante

- sufiksi U, L i LL mogu se dodati na konstante prikazane u bilo kojem brojevnom sustavu

`017LL`	signed long long konstanta, jednako 15LL
`0x17L`	signed long konstanta, jednako 23L
`0xFFFFFFFFu`	unsigned int konstanta, jednako 4294967295U
`0xFFFFFFFF`	signed int konstanta, jednako -1

Konverzijske specifikacije za printf i scanf

Tip	signed	unsigned
short int	%hd	%hu
int	%d	%u
long int	%ld	%lu
long long int	%lld	%llu

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

```
printf("%d", 0xFFFFFFFF);                 -1
printf("%u", 0xFFFFFFFF);                 4294967295
printf("%lld", 0xFFFFFFFFFFFFFFFF);       -1
printf("%llu", 0xFFFFFFFFFFFFFFFF);       18446744073709551615
printf("%o", 0xFFFFFFFF);                 37777777777
printf("%x", 0xFFFFFFFF);                 ffffffff
printf("%X", 0xFFFFFFFF);                 FFFFFFFF
printf("%llo", 0xFFFFFFFFFFFFFFFF);       1777777777777777777777
```

Primjer

```
#include <stdio.h>

int main(void) {
   unsigned int ubroj = 2000000000U;
   ubroj = ubroj * 2 / 2;
   printf("ubroj=%u\n", ubroj);

   signed int sbroj = 2000000000;
   sbroj = sbroj * 2 / 2;
   printf("sbroj=%d\n", sbroj);
   return 0;
}
```

```
ubroj=2000000000
```

```
sbroj=-147483648
```

- Objašnjenje:
  - `ubroj * 2 = 4000000000,  4000000000 / 2 = 2000000000`
  - `sbroj * 2 = -294967296, -294967296 / 2 = -147483648`

Konverzijske specifikacije - logičke pogreške

- Važno je koristiti odgovarajuće konverzijske specifikacije jer prevodilac neće prepoznati njihovu pogrešnu upotrebu
  - nastaju teško prepoznatljive logičke pogreške

```
printf("%lld", 10000000000LL);          10000000000
printf("%d", 10000000000LL);            1410065408

int a = 98304;
scanf("%hd", &a);                       za ulaz 10
printf("%d", a);                        65546
```

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'`	`0x61`	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
```