

Programiranje i programsко инžенjerство

Predavanja
2014. / 2015.

**3. Tipovi podataka u
programskom jeziku C**

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	→	200000005
za n = -2 000 000 000	→	-1999999995
za n = 2 147 483 647	→	-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 + 0.125;
    printf("%15.5f", y);
    return 0;
}
```

za $x = 2\ 097\ 151.0$	→	2097151.12500	
za $x = -2\ 097\ 151.0$	→	-2097150.87500	
za $x = 2\ 097\ 152.0$	→	2097152.00000	?
za $x = 1\ 000\ 000.3$	→	1000000.43750	?

Osnovni tipovi podataka

int - cjelobrojni tip

char - znakovni tip (ili mali cijeli broj)

float - realni tip

double - realni tip u dvostruko preciznosti

Prikaz cijelih brojeva u računalu

Binarni brojevni sustav

- Znamenke su **0** i **1**, dakle baza brojanja **B=2** što određuje **binarni brojevni sustav**
- **BIT**: iz engleskog ***Bi*nary *digit***, najmanja količina informacije i znamenka binarnog brojevnog sustava.
- Broj s n znamenki u brojevnom sustavu s bazom 10:
$$z_{n-1} \cdot 10^{n-1} + z_{n-2} \cdot 10^{n-2} + \cdots + z_1 \cdot 10^1 + z_0 \cdot 10^0, \quad z_i \in \{0, 9\}$$
- Broj s n znamenki u brojevnom sustavu s bazom 2:
$$z_{n-1} \cdot 2^{n-1} + z_{n-2} \cdot 2^{n-2} + \cdots + z_1 \cdot 2^1 + z_0 \cdot 2^0, \quad z_i \in \{0, 1\}$$

Pretvorba dekadskog broja u binarni

$$N = z_{n-1} \cdot 2^{n-1} + z_{n-2} \cdot 2^{n-2} + \cdots + z_1 \cdot 2^1 + z_0 \cdot 2^0$$

Izluči li se iz svih pribrojnika, osim posljednjeg, zajednički faktor 2:

$$N = 2 \cdot (z_{n-1} \cdot 2^{n-2} + z_{n-2} \cdot 2^{n-3} + \cdots + z_1 \cdot 2^0) + z_0 \quad \Rightarrow$$

$$N = 2 \cdot q_1 + z_0$$

može se zaključiti da je z_0 ostatak dijeljenja N s 2

Pogledajmo sada količnik q_1

$$q_1 = z_{n-1} \cdot 2^{n-2} + z_{n-2} \cdot 2^{n-3} + \cdots + z_1 \cdot 2^0$$

Izluči li se iz svih pribrojnika, osim posljednjeg, zajednički faktor 2:

$$q_1 = 2 \cdot (z_{n-1} \cdot 2^{n-3} + z_{n-2} \cdot 2^{n-4} + \cdots) + z_1 \quad \Rightarrow$$

$$q_1 = 2 \cdot q_2 + z_1$$

može se zaključiti da je z_1 ostatak dijeljenja q_1 s 2 itd sve dok se uzastopnim dijeljenjem s 2 ne postigne 0.

Dakle, uzastopnim dijeljenjem cijelog broja N s 2 i zapisivanjem ostataka dijeljenja dobiju se znamenke ekvivalentnog binarnog broja.

Pretvorba dekadskog broja u binarni

$$13_{10} = ?_2$$

$$N = 13 = 2 \cdot q_1 + z_0 \cdot 2^0 = 2 \cdot 6 + 1 \cdot 2^0 \Rightarrow z_0 = 1, q_1 = 6$$

$$q_1 = 6 = 2 \cdot q_2 + z_1 \cdot 2^0 = 2 \cdot 3 + 0 \cdot 2^0 \Rightarrow z_1 = 0, q_2 = 3$$

$$q_2 = 3 = 2 \cdot q_3 + z_2 \cdot 2^0 = 2 \cdot 1 + 1 \cdot 2^0 \Rightarrow z_2 = 1, q_3 = 1$$

$$q_3 = 1 = 2 \cdot q_4 + z_3 \cdot 2^0 = 2 \cdot 0 + 1 \cdot 2^0 \Rightarrow z_3 = 1, q_4 = 0$$

$$13_{10} = 1101_2$$

Pretvorba dekadskog broja u binarni

$$13_{10} = ?_2$$

	13	1	$13 \% 2$
$13 : 2$	6	0	$6 \% 2$
$6 : 2$	3	1	$3 \% 2$
$3 : 2$	1	1	$1 \% 2$
$1 : 2$	0		

$13_{10} = 1101_2$

Raspon brojeva koji se mogu prikazati u binarnom brojevnom sustavu

- Općenito: najveći dekadski broj s d znamenaka iznosi $10^d - 1$
 - Primjer: $d=2$, najveći broj $99 = 10^2 - 1$
- Općenito: najveći **binarni** broj s b znamenaka iznosi $2^b - 1$
 - Primjer: $b=4$, najveći broj $1111 = 2^4 - 1$
- Koliko binarnih znamenaka treba za prikaz dekadskog broja?

$$10^d - 1 = 2^b - 1 \Rightarrow$$

$$10^d = 2^b \Rightarrow$$

$$d \cdot \log 10 = b \cdot \log 2 \Rightarrow$$

$$d = b \log 2 \Rightarrow$$

$$b = d : \log 2 \Rightarrow b \approx d \cdot 3,33$$

Primjer: za prikaz pozitivnih dekadskih brojeva s dvije znamenke (tj. brojeva iz intervala $[0, 99]$), potreban broj binarnih znamenki je $b \approx 2 \cdot 3,33 = 6,66$

Raspon brojeva koji se mogu prikazati u binarnom brojevnom sustavu

Broj binarnih znamenki $b = 6,66$?

- Koji se najveći broj može prikazati sa 6 binarnih znamenaka?
 - $b \approx d \cdot 3,33$
 - $111111_2 = 32+16+8+4+2+1 = 63 = 64-1 = 2^6-1$
- Koji se najveći broj može prikazati sa 7 binarnih znamenaka?
 - $1111111_2 = 64+32+16+8+4+2+1 = 127 = 128-1 = 2^7-1$
- Očito, broj znamenaka u prethodnim izrazima treba zaokružiti na viši cijeli broj tj:

$$b = \lceil d : \log_2 \rceil$$

$$b \approx \lceil d \cdot 3,33 \rceil$$

Negativni binarni brojevi

- U registar se može pohraniti samo 0 ili 1 \Rightarrow za pohranu negativnog predznaka potreban je dogovor (konvencija).
- Uobičajeno se negativni brojevi prikazuju tzv. tehnikom dvojnog komplementa, tj. nule pretvaramo u jedinice, jedinice u nule (dobije se jedinični komplement, *one's complement*), a zatim se jediničnom komplementu pribroji 1 (dobije se dvojni komplement, *two's complement*).
- Primjer: -37 u registru s 8 bita

37	<table border="1"><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	0	0	1	0	0	1	0	1	
0	0	1	0	0	1	0	1			
	<table border="1"><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table>	1	1	0	1	1	0	1	0	jedinični komplement
1	1	0	1	1	0	1	0			
	$+ \quad \quad \quad \quad \quad \quad \quad \quad \quad 1$									
-37	<table border="1"><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr></table>	1	1	0	1	1	0	1	1	dvojni komplement
1	1	0	1	1	0	1	1			
	$+ \quad \quad \quad \quad \quad \quad \quad \quad \quad$									
37	<table border="1"><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	0	0	1	0	0	1	0	1	
0	0	1	0	0	1	0	1			
	<table border="1"><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0			

Primjer svih sadržaja u registru od 3 bita (uz mogućnost prikaza negativnih brojeva)

U registru s $n=3$ bita, ako se koristi tehnika dvojnog komplementa, mogu se prikazati sljedeći brojevi:

Dekadski broj Binarni broj

0	000
1	001
2	010
3	011
-4	100
-3	101
-2	110
-1	111

Za $n=3$ dobije se interval $[-2^2, 2^2 - 1]$, općenito $[-2^{n-1}, 2^{n-1} - 1]$

Za $n=8$ dobije se interval $[-2^7, 2^7 - 1]$, tj. $[-128, 127]$

Zadatak

- Dekadski broj -11 prikazati u obliku binarnog broja, u tehnici dvojnog komplementa

?

Zadatak

- Prethodni zadatak nije ispravno zadan jer nije navedeno s pomoću koliko bitova treba prikazati zadani broj
- Ispravno zadani zadatak glasi:
 - Dekadski broj -11 prikazati u obliku binarnog broja, u tehnici dvojnog komplementa, u registru od 5 bitova

?

Oktalni brojevni sustav

- Baza sustava je **B=8** a znamenke su **0, 1, 2, 3, 4, 5, 6, 7**
 - Koristi se za skraćeno zapisivanje binarnih sadržaja kada je to spremno
 - Zapis se može dobiti iz dekadskog uzastopnim dijeljenjem s 8 i zapisivanjem ostataka s desna na lijevo, ali i izravno iz binarnog zapisa:

$$N = z_5 \cdot 2^5 + z_4 \cdot 2^4 + z_3 \cdot 2^3 + z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0$$

grupiramo li tri po tri pribrojnika i izlučimo zajednički faktor:

$$N = (z_5 \cdot 2^2 + z_4 \cdot 2^1 + z_3 \cdot 2^0) \cdot 2^3 + (z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0) \cdot 2^0$$

$$N = (z_5 \cdot 2^2 + z_4 \cdot 2^1 + z_3 \cdot 2^0) \cdot 8^1 + (z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0) \cdot 8^0$$

⇒

$$o_1 = (z_5 \cdot 2^2 + z_4 \cdot 2^1 + z_3 \cdot 2^0), o_0 = (z_2 \cdot 2^2 + z_1 \cdot 2^1 + z_0 \cdot 2^0)$$

Primjer:

32-bitni broj 10 101 111 001 010 011 111 000 100 001₂

oktalni ekvivalent **2 5 7 1 2 3 7 0 4 1**₈

Napomena: po tri binarne znamenke grupirati s desna na lijevo

Heksadekadski brojevni sustav

- Baza sustava je **B = 16**, a znamenke su **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**
 - Koristi se za skraćeno zapisivanje binarnog sadržaja.
 - Zapis se može dobiti iz dekadskog uzastopnim dijeljenjem sa 16 i zapisivanjem ostataka s desna na lijevo, ali i izravno iz binarnog zapisa.
- Primjer:

15-bitni broj

101 1011 0011 1110₂

heksadekadski ekvivalent

5 B 3 E₁₆

Napomena: po četiri binarne znamenke grupirati s desna na lijevo

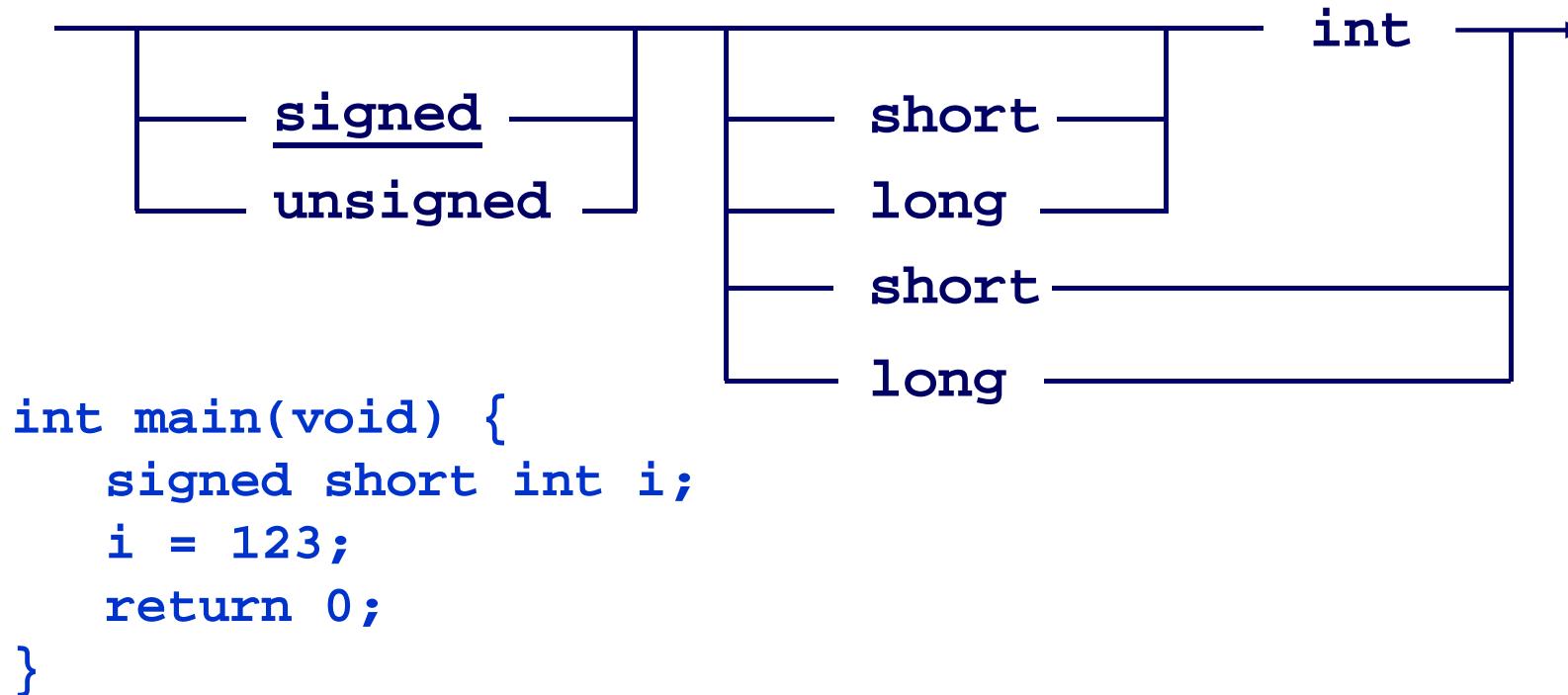
Cjelobrojni tip podatka i prefiksi u jeziku C

- **int** - cjelobrojni tip

- Prefiksi (ili kvalifikatori) za cjelobrojni tip podatka
 - short** - smanjuje raspon cjelobrojnih vrijednosti koje varijabla može sadržavati
 - long** - povećava raspon cjelobrojnih vrijednosti koje varijabla može sadržavati

- unsigned** - pohrana samo pozitivnih vrijednosti i nule
- signed** - pohrana pozitivnih vrijednosti, nule i negativnih vrijednosti (zato uz isti broj bitova dopušta manji raspon brojeva od *unsigned*)

Formalna definicija cjelobrojnog tipa



`signed short int i;` \Leftrightarrow `signed short i;` \Leftrightarrow `short int i;` \Leftrightarrow `short i;`
`signed int i;` \Leftrightarrow `int i;`
`signed long int i;` \Leftrightarrow `signed long i;` \Leftrightarrow `long int i;` \Leftrightarrow `long i;`
`unsigned short int i;` \Leftrightarrow `unsigned short i;`
`unsigned int i;`
`unsigned long int i;` \Leftrightarrow `unsigned long i;`

Cjelobrojni tip podatka i raspon brojeva

- Cjelobrojni tip podatka (`int`), s obzirom na raspon (određen brojem binarnih znamenki), može se definirati kao `short` ili `long`.
- U programskom jeziku C raspon tipova `short` i `long` nije propisan, ali vrijede sljedeća pravila:
 - `short` ne smije imati veći raspon od `int`
 - `int` ne smije imati veći raspon od `long`,
 - za raspone brojeva koji se mogu pohraniti može se napisati:
$$\text{short} \leq \text{int} \leq \text{long}$$
- U cjelobrojni tip podatka također pripada: `char` (kada se koristi kao brojevna, a ne znakovna vrijednost).

Zašto je to važno znati?

Primjer u programskom jeziku C:

- pretpostavka: **short int** koristi dva okteta

```
int main (void) {  
    short int i;  
    i = 32600;  
    i = i + 100;  
    printf ("%d\n", i);  
    i = i + 100;  
    printf ("%d\n", i);  
    return 0;  
}
```

- Što se i zašto dogodilo?

Cjelobrojne konstante u jeziku C

- Konstante pisane u dekadskoj notaciji:

7 20 64 -110 8092 65535 34567821

34567821L -17698794ℓ (na kraju malo slovo L)

- slovo L označava da se radi o "long" konstanti

- Konstante pisane u oktalnoj notaciji:

07 024 0100 0156 017634 0177777

- Konstante pisane u heksadekadskoj notaciji

0x7 0x14 0x40 0x6E 0x1F9C 0xFFFF

0xFFFFFFFF

Primjer: Zadavanje cjelobrojnih konstanti

```
#include <stdio.h>
int main (void) {
    int x, y, rez;
    /* oktalno zadan cijeli broj */
    x = 010;
    /* heksadekadski zadan cijeli broj */
    y = 0xF;
    rez = 10 + y / x;
    printf("Rezultat = %d\n", rez);
    return 0;
}
```

Cjelobrojni tip s predznakom i bez predznaka

Primjer:

- Broj **-5** prikazan tehnikom dvojnog komplementa u 16-bitnom registru:

111111111111011

- Isti niz binarnih znamenaka pohranjen u varijablu cjelobrojnog tipa **bez** predznaka, predstavlja broj 65531

Cjelobrojni tip s predznakom i bez predznaka

- Vrijednost pohranjena u varijablu tipa `signed` smatra se pozitivnom ili negativnom, ovisno o najznačajnijem bitu.
- Vrijednost pohranjena u varijablu tipa `unsigned` uvijek se smatra pozitivnim, što udvostručuje (u odnosu na varijablu tipa `signed`) najveću pozitivnu vrijednost koja u varijabli može biti pohranjena.

Primjer:

- Najveći pozitivni broj u 16-bitnom registru: `signed short int i;`
 $0111111111111111_2 = 32767_{10}$
- Najveći pozitivni broj u 16-bitnom registru: `unsigned short int i;`
 $1111111111111111 = 65535_{10}$

Cjelobrojne konstante bez predznaka u C-u

Konstante bez predznaka pišu se s **U** ili **u** na kraju

- u dekadskoj notaciji:

7U **20u** **64u**

- u oktalnoj notaciji:

07u **024U** **0100u**

- u heksadekadskoj notaciji

0x7u **0x14u** **0xFFFFFFFFu**

Operacije nad podacima s predznakom i bez predznaka

```
int main (void) {
    unsigned int ub;
    signed int sb;
    ub = 0xF0000000U;      → ub = F000000016 = 402653184010
    sb = 0xF0000000;      → sb = F000000016 = -26843545610
    ub = ub / 2;          → ub = 7800000016 = 201326592010
    sb = sb / 2;          → sb = 4000000016 = -13421772810
    printf("ub=%u\nsb=%d\n", ub, sb);
    return 0;
}
```

ispis: **ub=2013265920
sb=-134217728**

Objašnjenje: sadržaj registara za **ub** i **sb** nakon dijeljenja s 2 je različit zato što su varijable **ub** i **sb** različitog tipa. Sadržaj **ub** se tijekom dijeljenja promatrao kao pozitivan, a (isti takav) sadržaj **sb** kao negativan broj.

Operacije nad podacima s predznakom i bez predznaka

```
int main (void) {
    unsigned int ub;
    signed int sb;
    ub = 2000000000U;
    sb = 2000000000;
    ub = ub * 2 / 2;
    sb = sb * 2 / 2;
    printf("ub=%u\nsb=%d\n", ub, sb);
    return 0;
}
```

ispis: **ub=2000000000**
sb=-147483648

Objašnjenje:

ub * 2 = 4000000000, 4000000000 / 2 = 2000000
sb * 2 = -294967296, -294967296 / 2 = -147483648

Formatske specifikacije za scanf i printf

- Formatska specifikacija za ispis cijelih brojeva jest **%d** za cijele brojeve s predznakom i **%u** za cijele brojeve bez predznaka

```
#include <stdio.h>
int main (void) {
    int m;
    unsigned int n;
    scanf ("%d %u", &m, &n);
    printf ("%d %u\n", m, n);
    return 0;
}
```

Rasponi različitih tipova cijelih brojeva

- Rasponi ovise o konkretnoj implementaciji i platformi
 - za prevodilac koji se koristi na vježbama vrijedi:

Definicija	Broj bita	Interval brojeva
signed int	32	[-2147483648, 2147483647]
signed short int	16	[-32768, 32767]
signed long int	32	[-2147483648, 2147483647]
unsigned int	32	[0U, 4294967295U]
unsigned short int	16	[0U, 65535U]
unsigned long int	32	[0U, 4294967295U]

Prikaz slova i ostalih znakova

Prikaz slova i ostalih znakova

- Znakovi se prikazuju kombinacijom jedinica i nula – kôdom
- Koliko ima znakova?
 - 26 velikih i 26 malih slova engleske abecede A - Z, a - z
 - 10 znamenaka 0 - 9
 - operatori, interpunkcije, upravljački znakovi
- Za prikaz jednog znaka dovoljan je 1 bajt
- ASCII (ISO-7 standard): 7 bita za informaciju + 1 bit za *paritet*
⇒ $2^7 = 128$ različitih znakova
 - ASCII - American Standard Code for Information Interchange
 - ISO - International Organization for Standardization.
- 8-bitni ASCII kod
0-127 kao u 7-bitnom kodu, 128-255 prijeglasi i grafički znakovi, ovisno o tzv. kodnoj stranici

Unicode

- Noviji industrijski standard za kodiranje znakova
- Korištenje više bajtova za kodiranje znakova omogućuje predstavljanje gotovo svih pisama korištenjem jednog skupa znakova
- The Unicode Consortium: www.unicode.org
- npr. kod za znak *LA* iz pisma Tagalog (Filipini) jest $170E_{16}$



170E

Tablica ASCII kontrolnih znakova koji se ne mogu ispisati

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)

Tablica ASCII znakova koji se mogu ispisati

Dec. broj	Znak	Dec. broj	Znak	Dec. broj	Znak	Dec. broj	Znak
32	razmak	48	0	65	A	80	P
33	!	49	1	66	B	81	Q
34	"	50	2	67	C	82	R
35	#	51	3	68	D	83	S
36	\$	52	4	69	E	84	T
37	%	53	5	70	F	85	U
38	&	54	6	71	G	86	V
39	'	55	7	72	H	87	w
40	(56	8	73	I	88	X
41)	57	9	74	J	89	Y
42	*	58	:	75	K	90	Z
43	+	59	;	76	L	91	[
44	,	60	<	77	M	92	\
45	-	61	=	78	N	93]
46	.	62	>	79	O	94	^
47	/	63	?			95	_
		64	@				

Tablica ASCII znakova koji se mogu ispisati

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, nonprintable*) nalaze se na pozicijama 0-31

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

Na poziciji 127 nalazi se još jedan od kontrolnih znakova, znak DEL

char - znakovni tip podataka

- Pohrana malih cijelih brojeva sa ili bez predznaka
- Pohrana slova, interpunkcija, posebnih znakova
- Zauzima 1 oktet
- Definicija varijabli u programskom jeziku C:

char [-128, 127]

unsigned char [0, 255]

Primjeri znakovnih konstanti

'A'	slovo A (redni broj znaka A u ASCII tablici)
'0'	znamenka nula (znak 0, a ne broj 0)
'\x41'	heksadekadski zapis slova A
'\101'	oktalni zapis slova A
'\a'	primjer simboličkog zapisa za znakove koji se ne mogu tiskati
'\0'	oznaka za kraj znakovnog niza (nul-karakter, ništični znak, znak praznoga)
'\n'	prijelaz u novi red
'\\'	znak \ ("backslash")
'\''	znak ' (jednostruki navodnik)

Primjeri sa znakovnim konstantama

Primjeri:

`int c;`

pridružuje 8-bitni ASCII kôd znaka **A** koji je $65_{10} = 41_{16} = 101_8$

`c = 'A';`

pridružuje (uz konverziju) 32-bitni cijeli broj $65_{10} = 41_{16} = 101_8$

`c = 65; ili c=0x41; ili c=0101;`

pridružuje 8-bitni ASCII kôd 41_{16}

`c = '\x41';`

pridružuje 8-bitni ASCII kôd 101_8

`c = '\101';`

pridružuje 8-bitni ASCII kôd znaka ' koji je $39_{10} = 27_{16}$

`c = '\'';`

pridružuje 8-bitni ASCII kôd znaka \ koji je $92_{10} = 5C_{16}$

`c = '\\\';`

Ispis znakovnog tipa podatka u različitim formatima

- za čitanje i ispis znakova koristi se formatska specifikacija %c
- ako se podatak koristi kao numerička vrijednost (mali cijeli broj), koristi se formatska specifikacija %d ili %u

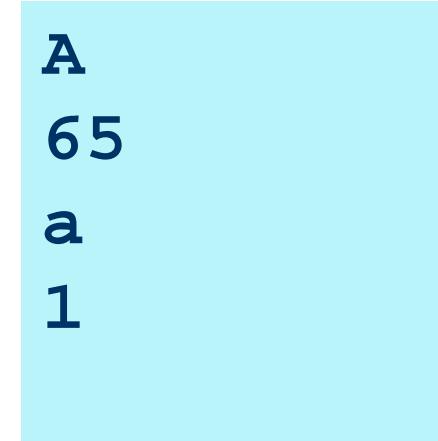
```
#include <stdio.h>
int main(void) {
    char x, y, p, q;
    x = 'A';
    y = x + 32;
    p = '\n';
    q = ' ';
    printf("%d %c %d %c %d %c\n", x, x, y, y, '0', '0'+1);
    printf("%d %c %d %c\n", p, p, q, q);
    return 0;
}
```

```
65_A_97_a_48_1_
10_ 
_32_
```

Primjer: Ispis znakovnog tipa podatka u različitim formatima

- Što će se ispisati naredbama:

```
char c;  
c ='A';  
printf ("%c\n", c);  
printf ("%d\n", c);  
printf ("%c\n", c + 32);  
printf ("%d\n", 'B' - 'A');
```



```
A  
65  
a  
1
```

Znamenke 0 do 9 u ASCII tablici

- Kada se znakovni tip koristi za pohranjivanje znamenki (0-9) treba obratiti pažnju na to da se u varijablu znakovnog tipa ne pohranjuje brojčana vrijednost te znamenke, nego ASCII vrijednost te znamenke, odnosno redni broj u ASCII tablici:

```
char a;
```

```
a = '1'; konačni rezultat isti kao nakon: a = 49;
```

- U varijabli **a** sada se nalazi brojčana vrijednost 49, odnosno redni broj znaka **'1'** u ASCII tablici.
- Ako se želi dobiti brojčana vrijednost znamenke, potrebno je od te vrijednosti oduzeti 48 , jer vrijednost 48 predstavlja ASCII vrijednost znaka **'0'**.

Pretvorba ASCII znamenke u cijeli broj

- Treba uočiti da pojedine znamenke prikazane kao znak ne odgovaraju po binarnom prikazu odgovarajućem cijelom broju. Npr. broj 7 prikazan kao cijeli broj u 1 oktetu iznosi $0000\ 0111_2$, a znamenka 7 prikazana kao ASCII znak $0011\ 0111_2$, odnosno 55_{10} .

Niz znakova (string)

- Konstantni znakovni niz se označava dvostrukim navodnicima, npr.
"Zagreb"
- U memoriji se kraj niza znakova označava nul-znakom ('\'0'). Na primjer, konstantni znakovni niz "Zagreb" u memoriji računala zauzima 7 okteta:

Z	a	g	r	e	b	\0
----------	----------	----------	----------	----------	----------	-----------

- primjer korištenja specijalnih znakova u konst. znakovnom nizu
"Znak \\ nazivamo \"backslash\""
- Definicija varijable u programskom jeziku C (objašnjeno kasnije):
char ime_niza[duljina_niza+1];
(kao polje znakova, paziti da se rezervira mjesto za '\0')

Nastavljanje konstantnog znakovnog niza

- "fakultet u Unskoj 3"
- "fakultet" " u Unskoj 3"
- "fakultet" "u Unskoj 3"
- "fakultet"
" u Unskoj 3"
- Prethodno prikazani konstantni znakovni nizovi u memoriji računala su pohranjeni na isti način, u 20 okteta

f	a	k	u	l	t	e	t		u		U	n	s	k	o	j		3\0
---	---	---	---	---	---	---	---	--	---	--	---	---	---	---	---	---	--	-----

- "fakultet
u Unskoj 3"

Pravopisna pogreška
(dojavljuje prevodilac)

Korištenje tipa podatka **char** za pohranu cijelog broja

```
#include <stdio.h>

int main (void) {

    char i;

    i = -5;
    printf ("%d\n", i);
    i = i + 17;
    printf ("%d\n", i);
    return 0;
}
```

Pohrana: 11111011

Ispis: -5

Pohrana: 00001100

Ispis: 12

Upamtite

Kod rješavanja zadataka, ako u zadatku nije drugačije navedeno, podrazumijevat će se:

- za tip podatka **char** koristi se jedan oktet
- za tip podatka **short int** koristi se dva okteta
- za tip podatka **int** koristi se četiri okteta
- za tip podatka **long** koristi se četiri okteta

Prikaz realnih brojeva u računalu

Decimalni brojevi u binarnom sustavu

- Decimalni binarni brojevi sadrže "binarnu točku", analogno decimalnom zarezu, odnosno točki u anglo-američkoj notaciji.
- Primjer prikaza decimalnih brojeva u binarnom sustavu:

$$\begin{aligned} \mathbf{5.75}_{10} &= \mathbf{5} * 10^0 + \mathbf{7} * 10^{-1} + \mathbf{5} * 10^{-2} = \\ &= \mathbf{1}*2^2 + \mathbf{0}*2^1 + \mathbf{1}*2^0 + \mathbf{1}*2^{-1} + \mathbf{1}*2^{-2} = \\ &= \mathbf{1 \ 0 \ 1 . \ 1 \ 1}_2 \end{aligned}$$

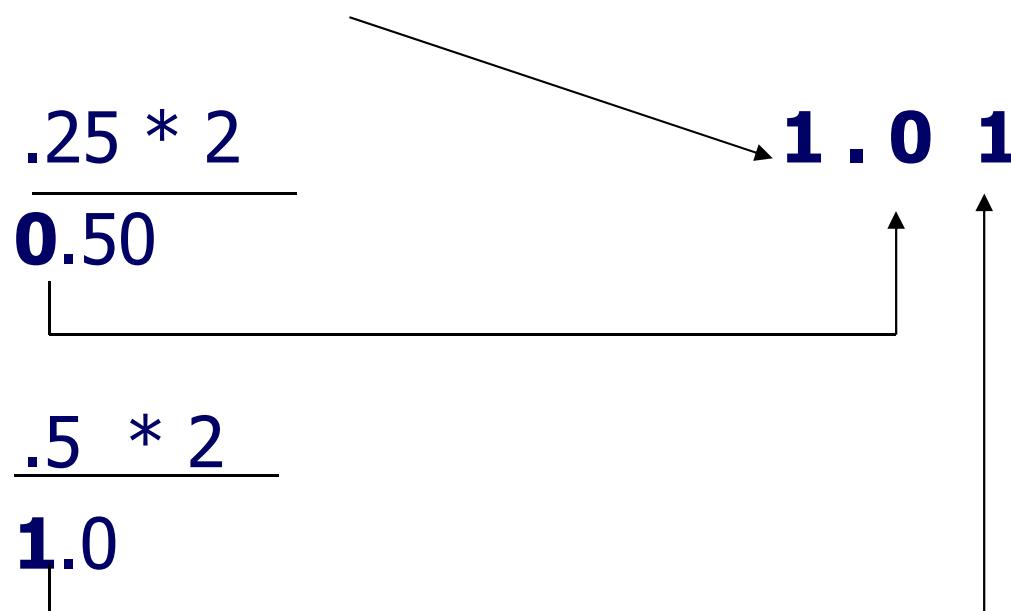
Općenito:

$$\begin{aligned} R &= z_{n-1} \cdot 2^{n-1} + z_{n-2} \cdot 2^{n-2} + \dots + z_1 \cdot 2^1 + z_0 \cdot 2^0 \\ &\quad + z_{n+1} \cdot 2^{-1} + z_{n+2} \cdot 2^{-2} + \dots \end{aligned}$$

Pretvaranje decimalnog broja iz dekadskog u binarni brojevni sustav

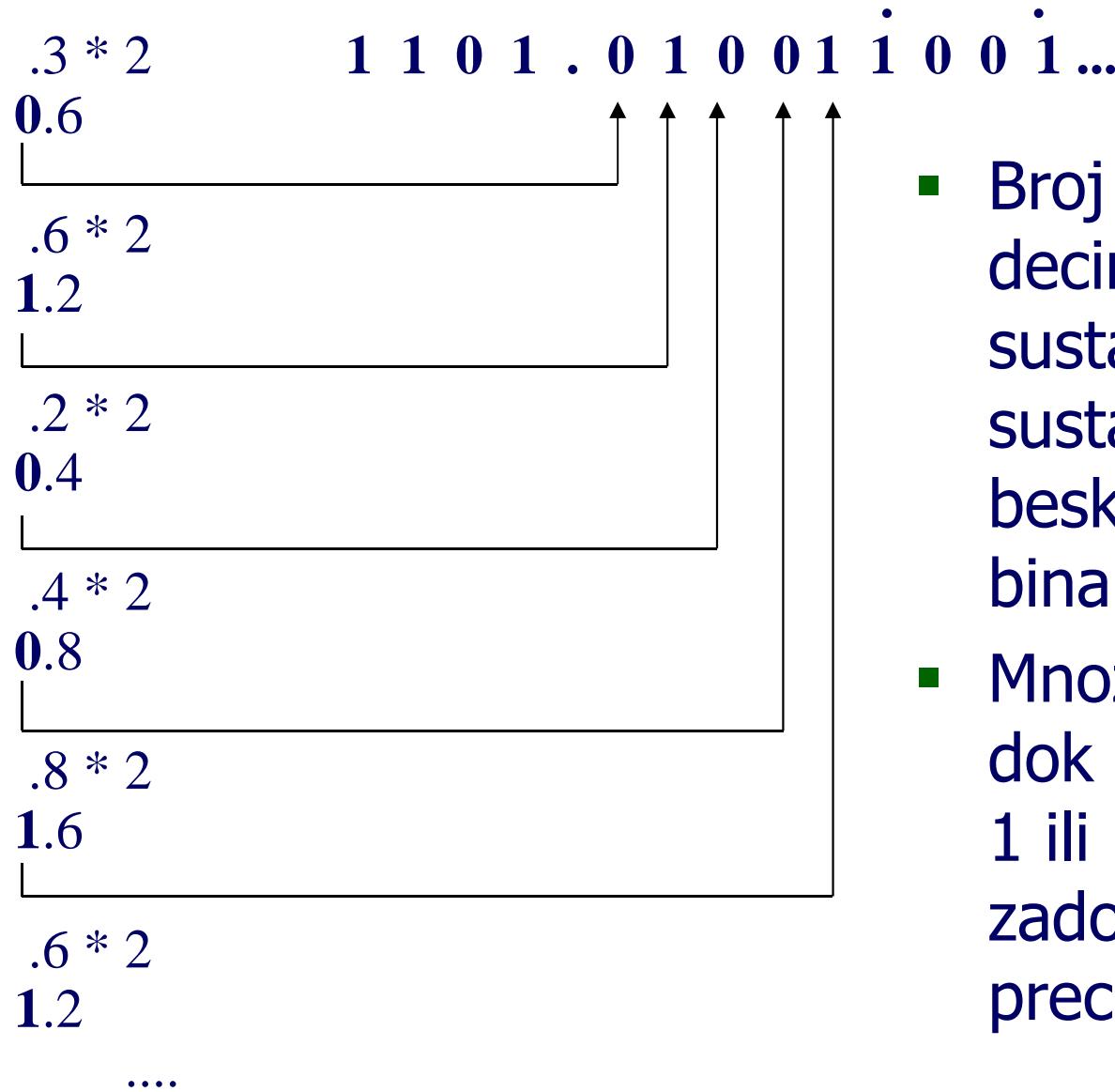
- Cjelobrojni dio dekadskog broja pretvara se u binarni uzastopnim dijeljenjem, a decimalni uzastopnim množenjem s 2, gdje cjelobrojni dio dobivenih produkata tvori decimalne znamenke binarnog broja.

$$1.25 = \mathbf{1} + .25$$



Pretvaranje decimalnog broja iz dekadskog u binarni brojevni sustav

$$13.3 = 13 + 0.3$$



- Broj s konačnim brojem decimala u dekadskom sustavu u binarnom sustavu može imati beskonačni periodički niz binarnih znamenaka.
- Množenje se nastavlja sve dok se ne dobije cijeli broj 1 ili bude dosegnuta zadovoljavajuća preciznost.

Množenje binarnog broja s 2^n i 2^{-n}

- Binarni broj se množi s potencijama baze 2 tako da se binarna točka pomakne odgovarajući broj mesta desno ili lijevo, ovisno o tome je li predznak potencije pozitivan ili negativan.

- Na primjer:

$$\begin{array}{r} 111 \\ \times 2^2 \\ \hline 11100 \end{array}$$

$$\begin{array}{r} 1.11 \\ \times 2^{-2} \\ \hline 0.0111 \end{array}$$

- Kako, međutim, u register pohraniti točku?

Prikaz realnih brojeva u računalu

Prikaz realnih brojeva u dekadskom obliku

- Kako inženjeri i znanstvenici prikazuju vrlo velike i vrlo male brojeve?
- Znanstvena notacija: decimalni broj s jednom znamenkom ispred decimalne točke (zareza), pomnožen odgovarajućom potencijom broja 10 (baza brojanja = 10).

- Kolika je prosječna udaljenost Neptuna i Sunca?

- 4503930000000 m

- $\underbrace{4.50393}_{\text{mantisa}} \cdot 10^{12} \text{ m}$

mantisa

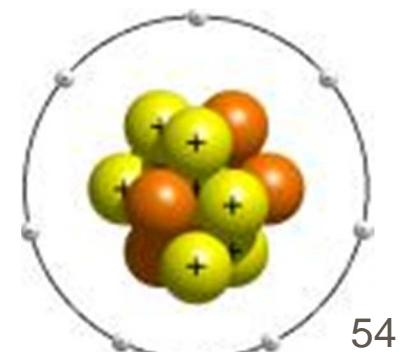
eksponent



- Koliko iznosi masa elektrona?

- $0.000000000000000000000000000000000910938188$ kg

- $9.10938188 \cdot 10^{-31}$ kg



Prikaz realnih brojeva u binarnom obliku

- Slično, realan broj u binarnom obliku, može se prikazati kao binarni decimalni broj s jednom binarnom znamenkom ispred binarne točke, pomnožen odgovarajućom potencijom broja 2 (baza brojanja = 2). Za broj u takvom obliku kaže se da je **normaliziran**.
- Na primjer:
$$101.11 = \underbrace{1.0111}_{\text{mantisa}} \cdot 2^2$$

$$0.00000000000010011 = 1.0011 \cdot 2^{-14}$$
- Normalizacija omogućava prikaz vrlo velikih i vrlo malih brojeva, bez korištenja velikog broja nula.

Prikaz realnih brojeva u računalu

- Realni brojevi u računalu se pohranjuju u normaliziranom obliku.
- Kako normalizirani broj pohraniti u registar?
- IEEE (Institute of Electrical and Electronics Engineers) standard 754 definira način pohrane realnih brojeva

Realni tip podatka u jeziku C

- Primjer definicije varijable: `float x;`
- Primjer realnih konstanti jednostrukе preciznosti:

`2.f` `2.34F` `-1.34e5f` `9.1093e-31f`

Zadnji znak realne konstante jednostrukе preciznosti treba biti f ili F

- Primjer C programa:

```
int main (void) {  
    float x;  
    x = 2.f * 5.0f * 3.14159F;  
    printf("Krug radijusa 5 ima opseg %f", x);  
    return 0;  
}
```

- Pored tipa `float`, u jeziku C postoje realni tipovi `double` i `long double`.

Realni brojevi jednostrukе preciznosti

- Najčešće prikazuje `float` tip podatka u jeziku C
- Koriste se 4 okteta (32 bita)
- Realni broj se pohranjuje u obliku:

31	30	23	22	0
P	Karakteristika	Mantisa bez skrivenog bita		

- 1 bit za pohranu predznaka
- 8 bitova za pohranu karakteristike
- 23 bita za pohranu mantise bez skrivenog bita

Realni brojevi jednostrukе preciznosti

31	30	23	22	0
P	Karakteristika	Mantisa bez skrivenog bita		

- P je oznaka za predznak
 - P = 1: negativan broj
 - P = 0: pozitivan broj
- BE je oznaka za binarni eksponent normaliziranog broja
 - raspon binarnog eksponenta $BE \in [-126, 127]$
- karakteristika K
 - $K = BE + 127$
 - raspon karakteristike: $K \in [0, 255]$
 - $K = 0$ i $K = 255$ se koriste za posebne slučajeve (objašnjeno kasnije)
- mantisa M
 - u registar se ne pohranjuje cijela mantisa M, već mantisa iz koje je uklonjen skriveni bit

Skriveni bit mantise

- U binarnom brojevnom sustavu, jedina znamenka koja se u normaliziranom broju (osim za broj 0) može pojaviti ispred binarne točke je 1

1.xxxxxxx

u registru od 8 bitova,
moguće je prikazati broj s ukupno 8 znamenaka

- Ta jedinica se ne pohranjuje i zato se naziva **skrivenim bitom (*hidden bit*)**. Na taj se način štedi jedan bit, a time povećava preciznost.

1.xxxxxxxx

u registru od 8 bitova,
moguće je prikazati broj s ukupno 9 znamenaka

1 ispred točke se podrazumijeva,
stoga ga ne treba pohraniti

Primjer: Prikazati broj 5.75 kao realni broj

- Realni dekadski broj prikazati u obliku realnog binarnog broja

$$5.75_{10} = 101.11_2$$

- Odrediti predznak: broj je pozitivan, stoga je $P = 0$

- Normalizirati binarni broj

$$101.11_2 \cdot 2^0 = 1.0111_2 \cdot 2^2$$

M

BE

- Izračunati karakteristiku i izraziti ju u binarnom obliku

$$K = 2_{10} + 127_{10} = 129_{10} = 1000\ 0001_2$$

- Izbaciti vodeću jedinicu iz mantise (skriveni bit)

$$M \text{ (bez skrivenog bita i binarne točke)} = 0111_2$$

- Prepisati predznak, karakteristiku i mantisu bez skrivenog bita u registar

0	10000001	011100000000000000000000
---	----------	--------------------------

P Karakteristika Mantisa bez skrivenog bita

0100 0000 1011 1000 0000 0000 0000 0000₂

4 0 B 8 0 0 0 0₁₆

Primjeri prikaza realnih brojeva

$$2 = 10_2 * 2^0 = 1_2 * 2^1 = 0100\ 0000\ 0000\ 0000\dots\ 0000\ 0000 = 4000\ 0000_{16}$$

P = 0, K = 1 + 127 = 128 (10000000), M = (1.) 000 0000 ... 0000 0000

$$-2 = -10_2 * 2^0 = -1_2 * 2^1 = 1100\ 0000\ 0000\ 0000\dots\ 0000\ 0000 = C000\ 0000_{16}$$

Jednako kao 2, ali P = 1

$$4 = 100_2 * 2^0 = 1_2 * 2^2 = 0100\ 0000\ 1000\ 0000\dots\ 0000\ 0000 = 4080\ 0000_{16}$$

Jednaka mantisa, BE = 2, K = 2 + 127 = 129 (10000001)

$$6 = 110_2 * 2^0 = 1.1_2 * 2^2 = 0100\ 0000\ 1100\ 0000\dots\ 0000\ 0000 = 40C0\ 0000_{16}$$

$$1 = 1_2 * 2^0 = 0011\ 1111\ 1000\ 0000\dots\ 0000\ 0000 = 3F80\ 0000_{16}$$

K = 0 + 127 (01111111).

$$.75 = 0.11_2 * 2^0 = 1.1_2 * 2^{-1} = 0011\ 1111\ 0100\ 0000\dots\ 0000\ 0000 = 3F40\ 0000_{16}$$

Kalkulator za vježbanje

- Internet stranica na kojoj se nalazi dobar kalkulator za uvježbavanje zadataka s prikazivanjem realnih brojeva

<http://babbage.cs.qc.cuny.edu/IEEE-754/>

Posebni slučajevi: prikaz broja 0

- Kada bi vodeća znamenka normaliziranog broja uvijek bila 1, ne bi bilo moguće prikazati broj 0
- Koristi se sljedeći dogovor: kada je $K=0$ i svi bitovi mantise postavljeni na 0, radi se o prikazu realnog broja 0
- U računalu se mogu prikazati brojevi "+0" i "-0"
0000 0000 0000 0000 0000 0000 0000 0000 → +0
1000 0000 0000 0000 0000 0000 0000 0000 → -0
- Međutim, pri usporedbi tih dviju vrijednosti, smatra se da su jednake.

Posebni slučajevi: denormalizirani broj

- Kada je $K=0$ i postoje bitovi mantise koji nisu 0, radi se o "denormaliziranom broju". Ne podrazumijeva se skriveni bit, te se smatra da je vodeći bit mantise 0. Vrijednost eksponenta je fiksirana na -126 (ne koristi se izraz $K=\text{binarni eksponent}+127$).

0000 0000 0110 0000 0000 0000 0000 0000

$$\rightarrow 0.11 \cdot 2^{-126}$$

0000 0000 0000 0000 0000 0000 0000 1101

$$\rightarrow 0.000\ 0000\ 0000\ 0000\ 0000\ 1101 \cdot 2^{-126}$$

Posebni slučajevi: prikaz $+\infty$ i $-\infty$

- Kada je K=255 i svi bitovi mantise su postavljeni na 0, radi se o prikazu $+\infty$ ili $-\infty$.
- Takvi brojevi se dobiju npr. prilikom dijeljenja s nulom:

```
float x, y, z, w;  
x = 5.f;  
y = -5.f;  
z = x / 0.f;  
w = y / 0.f;
```

0111 1111 1000 0000 0000 0000 0000 0000 $\rightarrow +\infty$

1111 1111 1000 0000 0000 0000 0000 0000 $\rightarrow -\infty$

Posebni slučajevi: prikaz NaN

- Ako je K=255 i postoje bitovi mantise koji nisu 0, radi se o NaN (*not a number*), tj. ne radi se o prikazu broja. NaN je posljedica obavljanja operacije čiji je rezultat nedefiniran ili se prilikom obavljanja operacije dogodila pogreška, npr.

```
float x, y, z;  
x = 0.f;  
y = 0.f;  
z = x / y;
```

0111 1111 1110 0000 0000 0000 0000 0000 → NaN

Raspon realnih brojeva (za format IEEE 754 - jednostruka preciznost)

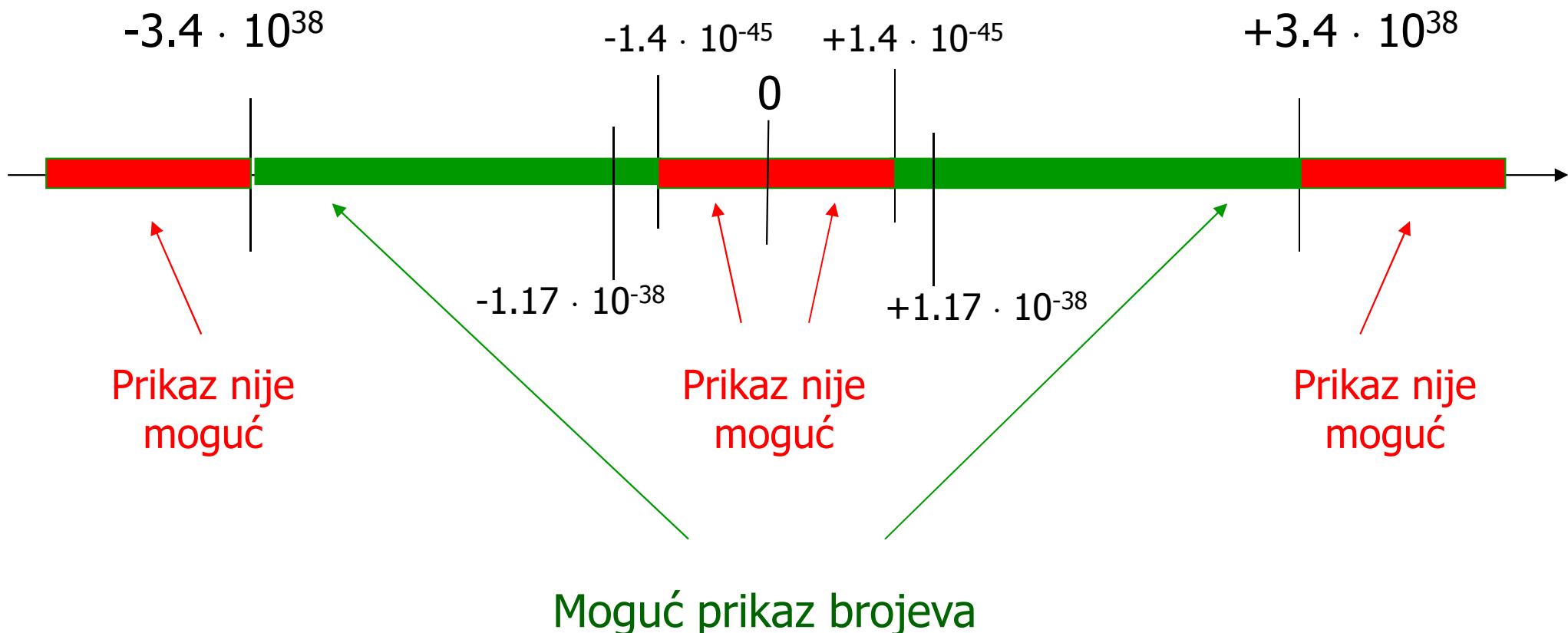
- Najmanji pozitivni broj koji se može prikazati je:

$$0.00000000000000000000000001_2 \cdot 2^{-126}$$
$$\approx 1.4 \cdot 10^{-45}$$

- Najveći pozitivni broj koji se može prikazati je:

$$1.111111111111111111111111111111_2 \cdot 2^{127} \approx 2^{128}$$
$$\approx 3.4 \cdot 10^{38}$$

Raspon realnih brojeva (za format IEEE 754 - jednostruka preciznost)



Prikaz cijelih brojeva u računalu (podsjetnik)

- U registru računala s konačnim brojem bitova moguće je prikazati konačan broj različitih brojeva
- Koliko je različitih **cijelih** brojeva moguće prikazati u registru od n bitova ?
 - 2^n različitih cijelih brojeva
- Skup cijelih brojeva Z je beskonačan - nije moguće prikazati **sve** brojeve iz skupa, ali
 - za prikaz **svih cijelih** brojeva iz intervala $[0, 2^n-1]$ ili iz intervala $[-2^{n-1}, 2^{n-1}-1]$ dovoljan je registar od n bitova

Prikaz realnih brojeva u računalu

- Koliko bitova bi trebao imati registar u kojem bi se mogli točno prikazati **svi** realni brojevi iz intervala [-1.0, 1.0] ?
 - beskonačno mnogo, jer $|[-1.0, 1.0]| \equiv |\mathbf{R}|$
- Samo konačan podskup realnih brojeva iz nekog intervala $[-a, a]$ moguće je **točno** (bez pogreške) prikazati u registru. Ostali realni brojevi iz tog intervala mogu se pohraniti samo kao njihove približne vrijednosti

Prikaz realnih brojeva u računalu

- Zašto se **svi** realni brojevi iz nekog intervala $[-a, a]$ ne mogu u registru prikazati bez pogreške?
 - realni broj može biti transcendentan - broj dekadskih znamenaka je beskonačan, stoga je i broj binarnih znamenaka beskonačan, npr.
 - $\pi, \ln 2$
 - realni broj s konačnim brojem dekadskih znamenaka može imati mantisu s beskonačnim periodičkim nizom binarnih znamenaka, npr.
 - $1.3 = 1.01001100110011001100\textcolor{red}{01100110011001} \dots$
 - realni broj može imati konačan, ali "prevelik" broj binarnih znamenaka mantise, npr.
 - $4194304.125 = 1.000000000000000000000001 \cdot 2^{22}$
 - previše znamenaka mantise da bi se mogao bez pogreške prikazati u IEEE 754 formatu jednostrukе preciznosti
 - ovaj broj se može bez pogreške prikazati u IEEE 754 formatu dvostrukе preciznosti (objašnjeno kasnije)

Koliko se različitih realnih brojeva može prikazati (za format IEEE 754 - jednostruka preciznost)

- za svaki $K \in [0, 254]$
 - moguće su 2^{23} različite mantise
 - moguća su dva predznaka
 - ukupno $255 * 2^{23} * 2 = 4,278,190,080$ različitih realnih brojeva
- uz $K=255$, moguće je prikazati $+\infty$, $-\infty$ i NaN
- bez pogreške je moguće prikazati približno $4 \cdot 3 \cdot 10^9$ različitih realnih brojeva iz intervala $[-3 \cdot 4 \cdot 10^{38}, -1 \cdot 4 \cdot 10^{-45}] \cup [1 \cdot 4 \cdot 10^{-45}, 3 \cdot 4 \cdot 10^{38}]$
- za ostale realne brojeve (njih beskonačno mnogo) iz navedenih intervala moguće je prikazati samo približne vrijednosti (**uz veću ili manju pogrešku**)
- realni brojevi izvan navedenih intervala se uopće ne mogu prikazati

Preciznost realnih brojeva

- Preciznost je svojstvo koje ovisi o količini informacije korištene za prikaz broja. Veća preciznost znači:
 - moguće je prikazati više različitih brojeva
 - brojevi su na brojevnom pravcu međusobno "bliži" (veća rezolucija)
 - veličina pogreške pri prikazu broja je manja

Pogreške pri prikazu realnih brojeva

- x - realni broj kojeg treba pohraniti u registar
- x^* - približna vrijednost broja x koja je zaista pohranjena u registar
- Apsolutna pogreška α
$$\alpha = x^* - x$$
- Relativna pogreška ρ
$$\rho = \alpha / x$$
- Primjer: ako je u registru umjesto potrebne vrijednosti $x = 1.57$ pohranjena vrijednost $x^* = 1.625$
$$\alpha = 1.625 - 1.57 = 0.055$$
$$\rho = 0.055 / 1.57 = 0.035$$

Pogreške pri prikazu realnih brojeva

- Najveća moguća relativna pogreška ovisi o broju bitova mantise. Za IEEE 754 jednostruke preciznosti:

$$|\rho| \leq 2^{-24} \approx 6 \cdot 10^{-8}$$

- Najveća moguća absolutna pogreška ovisi o broju bitova mantise i konkretnom broju x koji se prikazuje. Za IEEE 754 jednostruke preciznosti:

$$|\alpha| \leq 2^{-24} \cdot |x| \approx 6 \cdot 10^{-8} \cdot |x|$$

Primjer: pogreške pri prikazu realnih brojeva (za format IEEE 754 - jednostruka preciznost)

Najveća absolutna pogreška koja se uz jednostruku preciznost može očekivati pri pohrani realnog broja **332.3452**:

$$|\alpha| \leq 6 \cdot 10^{-8} \cdot 332.3452 \approx 2 \cdot 10^{-5}$$

```
float f1;  
f1 = 332.3452f;
```

očekuje se da će biti pohranjen broj **332.3452 ± 2 · 10⁻⁵**

```
printf("%19.15f", f1); → 332.345214843750000
```

Zaista, absolutna pogreška je $1.484375 \cdot 10^{-5}$, što je po absolutnoj vrijednosti manje od $2 \cdot 10^{-5}$.

Primjer: pogreške pri prikazu realnih brojeva (za format IEEE 754 - jednostruka preciznost)

Najveća absolutna pogreška koja se uz jednostruku preciznost može očekivati pri pohrani realnog broja 0.7 :

$$|\alpha| \leq 6 \cdot 10^{-8} \cdot 0.7 \approx 4.2 \cdot 10^{-8}$$

```
float f2;  
f2 = 0.7f;
```

očekuje se da će biti pohranjen broj $0.7 \pm 4.2 \cdot 10^{-8}$

```
printf("%17.15f", f2); → 0.699999988079071
```

Zaista, absolutna pogreška je $-1.1920929 \cdot 10^{-8}$, što je po absolutnoj vrijednosti manje od $4.2 \cdot 10^{-8}$.

Numeričke pogreške

- Neki dekadski brojevi se ne mogu prikazati pomoću konačnog broja binarnih znamenaka. Primjer:

```
float f = 0.3f;  
printf("%18.16f ", f); → 0.300000119209290
```

- Za prikaz nekih realnih brojeva potrebno je "previše" binarnih znamenaka. Primjer:

```
float f = 4194304.125f;  
1.00000000000000000000000001 · 222  
printf("%14.6f ", f); → 4194304.000000
```

Numeričke pogreške

- Računanje s brojevima bitno različitog reda veličine može dovesti do numeričke pogreške
- **Primjer (uz jednostruku preciznost):**

1000000.0_{10} : (1.) $11101000010010000000000 * 2^{19}$

0.015625_{10} : (1.) $0000000000000000000000000 * 2^{-6}$

Kod zbrajanja, binarne točke moraju biti poravnate:

$$\begin{array}{r} 1.1101000010010000000000 * 2^{19} \\ + 0.0000000000000000000000000 * 2^{-6} \\ \hline = 1.1101000010010000000000 * 2^{19} = 1000000.0_{10} \end{array}$$

Numeričke pogreške

Računanje s brojevima bitno različitog reda veličine

- primjer u programskom jeziku C:

```
float f = 6000000.0f;  
float malif = 0.25f;  
f = f + malif;  
printf("%f ", f);
```

- očekuje se ispis

6000001.000000

- međutim, ispisat će se:

6000000.000000

Formatske specifikacije za scanf i printf

- Formatska specifikacija za realne brojeve jednostrukog preciznosti jest %f

```
#include <stdio.h>
int main (void) {
    float x, y;
    scanf ("%f %f", &x, &y);
    printf ("%f %12.3f\n", x, y);
    return 0;
}
```

Realni brojevi dvostrukе preciznosti

- Najčešće prikazuje **double** tip podatka u jeziku C
- Koristi se 8 okteta (64 bita)
- Realni broj se pohranjuje u obliku

63 62

52 51

0

P	Karakteristika	Mantisa

- P je predznak ($P = 1$: negativan broj; $P = 0$: pozitivan broj)
- $K = BE + 1023$ (11 bita)
Raspon karakteristike: $K \in [0,2047]$.
Raspon binarnog eksponenta $BE \in [-1022,1023]$
- Mantisa (52+1 bit).

Realni brojevi dvostrukе preciznosti

▪ **Posebni slučajevi**

- Kada je $K = 0$ i svi bitovi mantise su nula, radi se o broju nula
- Kada je $K = 0$ i postoje bitovi mantise koji nisu 0, tada se radi o denormaliziranom broju
- Kada je $K = 2047$ i svi bitovi mantise su 0, radi se o $+\infty$ ili $-\infty$
- Kada je $K = 2047$ i postoje bitovi mantise koji nisu 0, tada se radi o prikazu broja (NaN)

Raspon realnih brojeva (za format IEEE 754 - dvostruka preciznost)

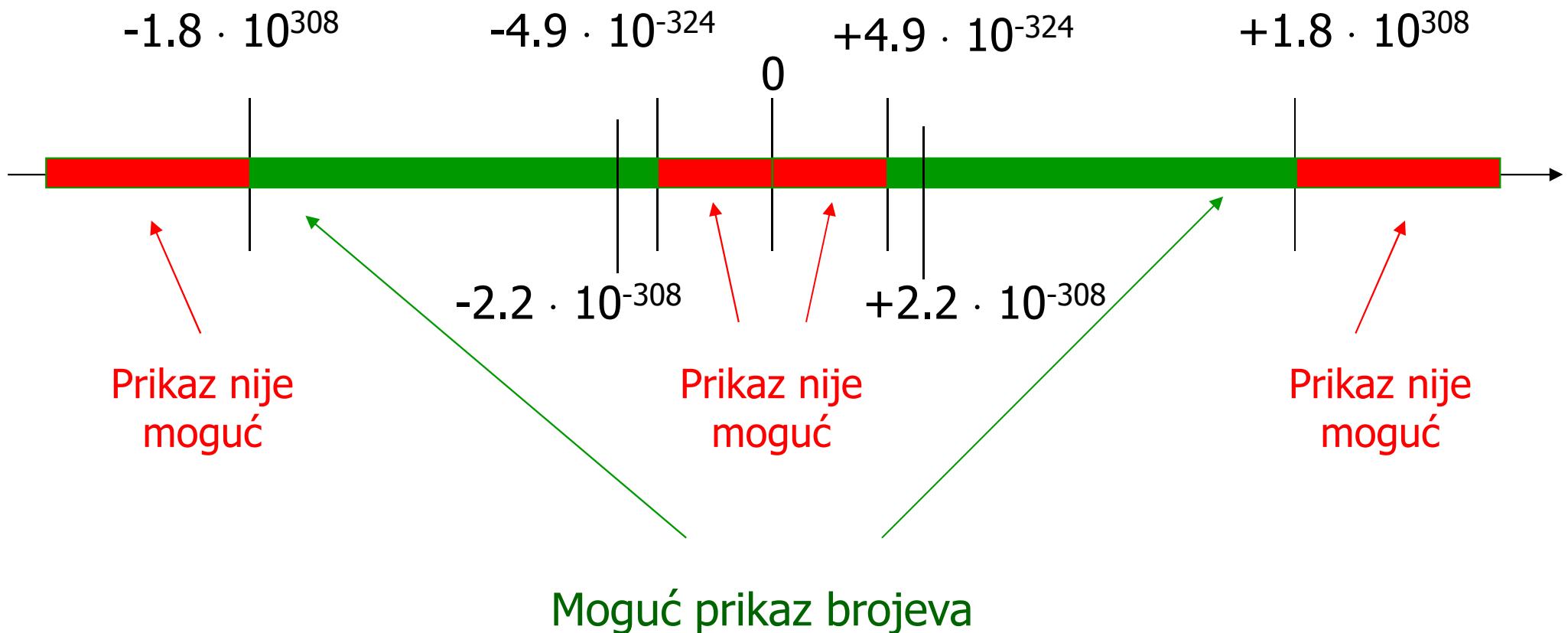
- Najmanji pozitivni broj koji se može prikazati je:

$$\begin{aligned}0.00\dots001_2 \cdot 2^{-1022} \\ \approx 4.9 * 10^{-324}\end{aligned}$$

- Najveći pozitivni broj koji se može prikazati je:

$$\begin{aligned}1.11\dots11_2 \cdot 2^{1023} \approx 2^{1024} \\ \approx 1.8 * 10^{308}\end{aligned}$$

Raspon realnih brojeva (za format IEEE 754 - dvostruka preciznost)



Pogreške pri prikazu realnih brojeva (za format IEEE 754 - dvostruka preciznost)

- Najveća moguća relativna pogreška ovisi o broju bitova mantise. Za IEEE 754 dvostrukе preciznosti:

$$|\rho| \leq 2^{-53} \approx 1.1 \cdot 10^{-16}$$

- Najveća moguća absolutna pogreška ovisi o broju bitova mantise i konkretnom broju x koji se prikazuje. Za IEEE 754 dvostrukе preciznosti:

$$|\alpha| \leq 2^{-53} \cdot |x| \approx 1.1 \cdot 10^{-16} \cdot |x|$$

Primjer: pogreške pri prikazu realnih brojeva (za format IEEE 754 - dvostruka preciznost)

Najveća absolutna pogreška koja se uz dvostruku preciznost može očekivati pri pohrani realnog broja 0.7:

$$|\alpha| \leq 1.1 \cdot 10^{-16} \cdot 0.7 \approx 7.7 \cdot 10^{-17}$$

```
double f4;  
f4 = 0.7;
```

očekuje se da će biti pohranjen broj $0.7 \pm 7.7 \cdot 10^{-17}$

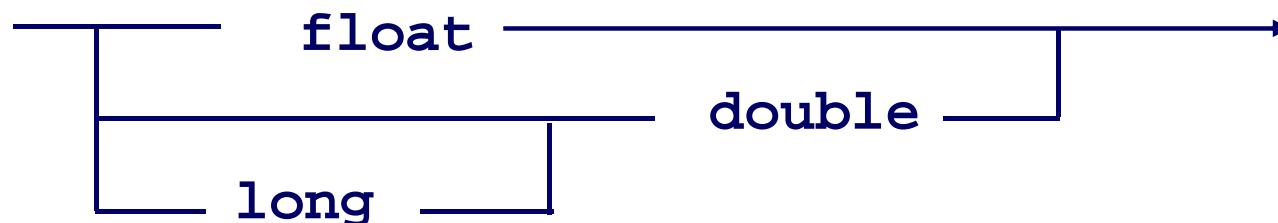
```
printf("%19.17f", f4); → 0.6999999999999996
```

Zaista, absolutna pogreška je $-4.0 \cdot 10^{-17}$, što je po absolutnoj vrijednosti manje od $7.7 \cdot 10^{-17}$.

Realni tip podatka u jeziku C

- Primjer definicije varijabli u programskom jeziku C:

```
float x;  
double y;  
long double z;
```



- Programske jezike C ne propisuju preciznost tipova `float`, `double` i `long double`, ali `float` ne smije biti precizniji od `double`, te `double` ne smije biti precizniji od `long double`:

`float ≤ double ≤ long double`

Tip `long double`

- Prostor za pohranu ovisi o platformi, pa se tako mogu naći implementacije u kojima je njegova veličina 64, 80, 96 ili 128 bita.
- ANSI standard ne propisuje preciznost, ali zahtijeva da nije manje precizan od **double** tipa.
- Primjer raspodjele ako je njegova veličina 80 bita:

Karakteristika: 15 bita

Binarni eksponent: Karakteristika – 16383

Realne konstante

- Primjer realnih konstanti za različite tipove realnih brojeva:

2.f 2.34F -1.34e5f float

1. 2.34 9e-8 8.345e+25 double

1.L 2.34L -2.5e-37L long double

Formatske specifikacije za scanf i printf

- Formatska specifikacija za realne brojeve dvostrukе preciznosti је %lf kod čitanja, te %lf ili %f kod pisanja

```
#include <stdio.h>
int main (void) {
    double x, y;
    scanf ("%lf %lf", &x, &y);
    printf ("%lf %12.3lf\n", x, y);
    printf ("%f %12.3f\n", x, y);
    return 0;
}
```

Upamtite

Kod rješavanja zadataka, ako u zadatku nije drugačije navedeno, podrazumijevat će se:

- za tip podatka **float** koristi se četiri okteta
- za tip podatka **double** koristi se osam okteta

Određivanje sjecišta dvaju pravaca

- Odrediti sjecište dvaju pravaca (ili riješite sustav od dvije jednadžbe s dvije nepoznanice!?). Parametre pravaca učitati programom. Ako sjecište ne postoji, ispisati odgovarajuću poruku.

Postupak:

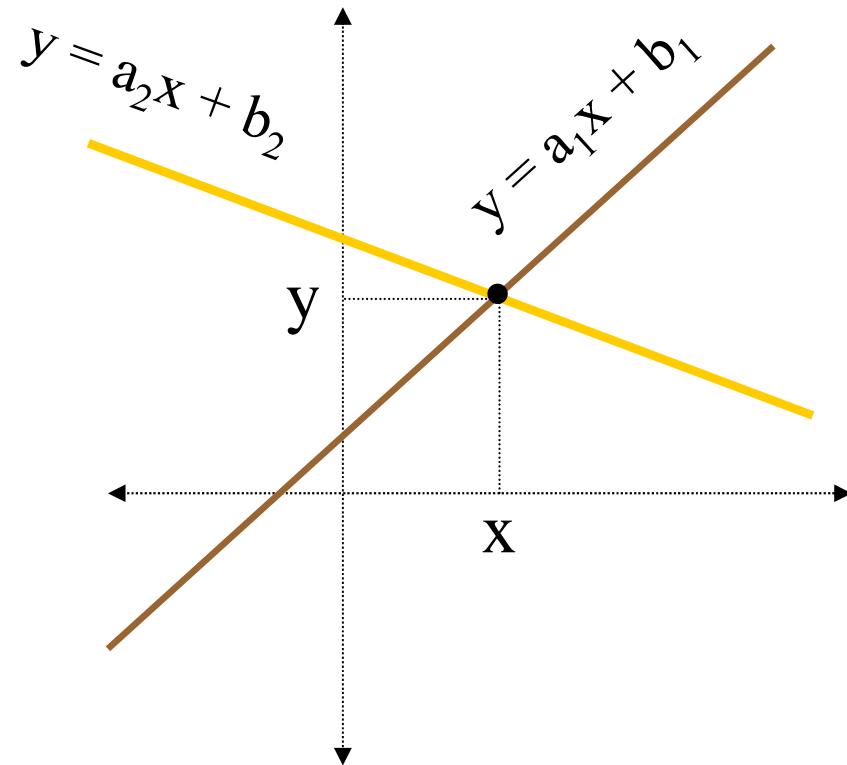
$$y = a_1 \cdot x + b_1$$

$$y = a_2 \cdot x + b_2$$

$$a_1 \cdot x + b_1 = a_2 \cdot x + b_2$$

$$x = (b_2 - b_1) / (a_1 - a_2)$$

$$y = a_1 \cdot x + b_1$$



Određivanje sjecišta dvaju pravaca

Rješenje u pseudokodu

```
učitaj (a1,b1,a2,b2)
ispiši (a1,b1,a2,b2)
izračunaj nazivnik izraza (b2-b1)/(a1-a2)
ako je nazivnik jednak nuli onda
|   ispiši poruku da su pravci paralelni
inače
|   izračunaj x,y
|   ispiši (x,y)
kraj
```

Određivanje sjecišta dvaju pravaca

Rješenje u C-u (I dio):

```
#include <stdio.h>
int main (void) {

    float a1, a2, b1, b2, x, y, anaz;
    scanf ("%f %f %f %f", &a1, &b1, &a2, &b2);
    printf("a1=%f b1=%f a2=%f b2=%f\n"
           , a1, b1, a2, b2);

    /* izracunaj nazivnik izraza (b2-b1)/(a1-a2)*/
    anaz = a1 - a2;
```

Određivanje sjecišta dvaju pravaca

Rješenje u C-u (II dio)

```
if (anaz == 0.f) {
    printf ("Pravci su paralelni\n");
} else {
    /* pravci se sijeku */
    printf ("Nazivnik izraza: %f\n", anaz);
    x = (b2 - b1) / anaz;
    y = a1 * x + b1;
    printf("Koordinate sjecista su(%f,%f)\n",
           x, y);
}
return 0;
}
```

Određivanje sjecišta dvaju pravaca

Prvo testiranje programa

- Ulazni podaci:

1 2 3 4

- Ispis na zaslonu:

```
a1=1.000000 b1=2.000000 a2=3.000000 b2=4.000000
Nazivnik izraza: -2.000000
Koordinate sjecišta su (-1.000000, 1.000000)
```

Određivanje sjecišta dvaju pravaca

Drugo testiranje programa

- Ulazni podaci:

```
1 2 1.000001 3
```

- Ispis na zaslonu:

```
a1=1.000000 b1=2.000000 a2=1.000001 b2=3.000000  
Nazivnik izraza: -0.000001  
Koordinate sjecista su (-1048576.000000, -1048574.000000)
```

Točan rezultat: (-1000000, -999998)

Provesti eksperiment: povećati preciznost varijabli

Veličina osnovnih tipova podataka

- Veličina osnovnih tipova podataka, odnosno količina memorije koju zauzima jedna varijabla osnovnog tipa ovisi o konkretnoj implementaciji prevodioca.
- za GCC:

Tip	Veličina
<code>char, unsigned char</code>	1 oktet
<code>short, unsigned short</code>	2 okteta
<code>int, unsigned int</code>	4 okteta
<code>long, unsigned long</code>	4 okteta
<code>float</code>	4 okteta
<code>double</code>	8 okteta
<code>long double</code>	12 okteta

Osobitosti C prevodilaca

- Veličina prostora za pohranu podatka određenog tipa nije propisana standardom. Stoga je programeru na raspolaganju operator `sizeof`, koji za zadani operand izračunava veličinu prostora za pohranu izraženu u oktetima.
- Operand može biti naziv tipa podatka ili naziv varijable
- Primjer:

```
int var;  
printf("%d", sizeof(char)); /* ispisuje 1 */  
printf("%d", sizeof(var)); /* ispisuje 4 */
```

Primjeri pohrane različitih tipova konstanti

- Na koji će način prevodilac interpretirati i pretvoriti u binarni oblik sljedeće konstante:

5 00000000000000000000000000000000101 (cjelobrojna konstanta u 32 bita)
'5' 00110101 (znakovna konstanta u 8 bita)
5. 01000000010100000....0 (realna konstanta tipa double u 64 bita)
5.f 010000010100000....0 (realna konstanta tipa float u 32 bita)
05 00000000000000000000000000000000101 (oktalna konstanta u 32 bita)
0x5 00000000000000000000000000000000101 (heksadekadska konst. u 32 bita)
"5" 00110101 00000000 (konstantni znakovni niz 2 x 8 bita)

- Poseban problem studenti imaju u predočavanju raznih "nula"

0 00000000000000000000000000000000 (cjelobrojna konstanta u 32 bita)
'0' 00110000 (znakovna konstanta u 8 bita)
'\0' 00000000 (znakovna konstanta "nul-karakter", u 8 bita)
"0" 00110000000000 (konstantni znakovni niz, 2 puta po 8 bita)

Zamjena za logički tip podataka

Matematička logika: sud

- Osnovni pojam: logički sud
- Može biti istinit ili lažan
- Primjeri:
 - sud $1 < 2$ je istinit
 - sud $3 > 4$ je lažan
- Osnovni ili atomni sud (atom): istinitost ili lažnost utvrđuje se neposrednim zaključivanjem
- Složeniji sud tvori se formulama koje se sastoje od:
 - atoma
 - logičkih operatora
 - zagrada

Matematička logika: osnovni operatori

- Negacija \neg

$$A \quad \neg A$$

$$\begin{matrix} 1 & 0 \end{matrix}$$

$$\begin{matrix} 0 & 1 \end{matrix}$$

- Konjunkcija \wedge

$$A \quad B \quad A \wedge B$$

$$\begin{matrix} 0 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 1 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 0 & 1 & 0 \end{matrix}$$

$$\begin{matrix} 1 & 1 & 1 \end{matrix}$$

- Disjunkcija \vee

$$A \quad B \quad A \vee B$$

$$\begin{matrix} 0 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 1 & 0 & 1 \end{matrix}$$

$$\begin{matrix} 0 & 1 & 1 \end{matrix}$$

$$\begin{matrix} 1 & 1 & 1 \end{matrix}$$

- Ekskluzivna disjunkcija \otimes

$$A \quad B \quad A \otimes B$$

$$\begin{matrix} 0 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 1 & 0 & 1 \end{matrix}$$

$$\begin{matrix} 0 & 1 & 1 \end{matrix}$$

$$\begin{matrix} 1 & 1 & 0 \end{matrix}$$

Matematička logika: korisne ekvivalencije

- S jednom varijablu i konstantama

$\neg(\neg A)$	= A	zakon dvostrukog negacije
$A \wedge A$	= A	idempotentnost konjunkcije
$A \wedge 1$	= A	
$A \wedge 0$	= 0	
$A \wedge \neg A$	= 0	
$A \vee A$	= A	idempotentnost disjunkcije
$A \vee 1$	= 1	
$A \vee 0$	= A	
$A \vee \neg A$	= 1	

Matematička logika: korisne ekvivalencije

- S dvije varijable

$$A \vee B = B \vee A$$

$$A \wedge B = B \wedge A$$

$$\neg(A \wedge B) = \neg B \vee \neg A$$

$$\neg(A \vee B) = \neg B \wedge \neg A$$

$$A \wedge (A \vee B) = A$$

$$A \vee (A \wedge B) = A$$

komutativnost disjunkcije

komutativnost konjunkcije

de Morganov zakon

de Morganov zakon

apsorpcija

apsorpcija

- S tri varijable

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C$$

$$A \vee (B \vee C) = (A \vee B) \vee C$$

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

asocijativnost konjunkcije

asocijativnost disjunkcije

distributivnost konjunkcije

distributivnost disjunkcije

Logička vrijednost u C-u

- U C-u ne postoji ključna riječ koja bi označavala podatak logičkog tipa, dok u nekim jezicima postoji poseban tip podataka (logical, boolean)
- Logička vrijednost odgovara vrijednosti logičkog suda (prosudbe)

DA	ili	NE
YES	ili	NO
TRUE	ili	FALSE
T	ili	F

- Svaki tip podatka u C-u je ujedno logički podatak, i to:

istina	ako je vrijednost	$\neq 0$
laž	ako je vrijednost	$= 0$

Logička vrijednost u C-u

- Ponekad se razumljivost programskog koda može povećati korištenjem simboličkih konstanti

```
#define TRUE 1
#define FALSE 0
...
int kisaPada, sunceSija;
sunceSija = TRUE;
kisaPada= FALSE;
```

Relacijski (usporedbeni) operatori - podsjetnik

- Uspoređuju dva operanda. Tvore atomne sudove:

<i>Operator</i>	<i>Značenje</i>	<i>Logički izraz</i>	<i>Rezultat</i>
<code>==</code>	jednako	<code>1 == 1</code>	1 (Istina)
<code>!=</code>	različito	<code>2 != 2 + 2</code>	1 (Istina)
<code>></code>	veće	<code>5 > 6</code>	0 (Laž)
<code>>=</code>	veće ili jednako	<code>6 >= 6</code>	1 (Istina)
<code><</code>	manje	<code>7 < 10</code>	1 (Istina)
<code><=</code>	manje ili jednako	<code>7 <= 6</code>	0 (Laž)

Logički operatori

- Jednostavni relacijski izrazi mogu se kombinirati u složene pomoću logičkih operatora. C uključuje 3 logička operatorka:

Operator	Značenje
&&	logičko I
 	logičko ILI
!	logičko NE

- U programskom jeziku C postoje i logički operatori nad bitovima. Obrađeni u jednom od narednih predavanja!

Primjeri:

```
if ((x > 20) && (x < 100))
    printf("x se nalazi u otvorenom intervalu 20-100");
if ((x < 5) || (x > 20))
    printf("x se ne nalazi u zatvorenom intervalu 5-20");
if (!(x > 20))
    printf("x je manji ili jednak 20");
```

Poteškoće sa složenim logičkim uvjetima

- Složeni izrazi često se pogrešno napišu zbog "doslovног prepisivanja" izrečenog uvjeta
- Izraz "ako je x veći od 20 i manji od 100" (uočiti da se pod "i manji od 100" podrazumijeva "i x je manji od 100"), često se "doslovno prepiše" u:

```
if ( x > 20 && < 100 )
```

što dovodi do pogreške pri prevodenju.

Ispravno:

```
if ( x > 20 && x < 100 )
```

Rezultat primjene logičkih operatora

- logička vrijednost FALSE

0 0.0 0.0f 0L '\0'

- logička vrijednost TRUE

1 1.F 1.0L 0.15 148.9f -512 'a' '\n'

- rezultat logičkog izraza je uvijek 0 ili 1 (tip int)

7 > 8 → 0

7.5 <= 8.5 → 1

! 1 → 0

! 15.75F → 0

! 0 → 1

! 0.0F → 1

! '\0' → 1

Logički operatori - prioritet

OPERATORI	
	!
	< <= > >=
	== !=
	&&

Primjeri:

```
if (x > 20 && x < 100)
    printf("x se nalazi u otvorenom intervalu 20-100");

if (!x > 20)
    printf("x je manji ili jednak 20");

if (!(x > 20))
    printf("x je manji ili jednak 20");
```

Logički operatori - vježba

1. Ispisati tekst "istina je" ako je učitani realni broj u intervalu [3,5] ili je u intervalu [7,9]
2. Ispisati tekst "istina je" ako je učitani cijeli broj pozitivan i ima 2 ili 4 znamenke
3. Ispisati tekst "istina je" ako uvjet iz 1. zadatka nije zadovoljen (riješiti sa i bez korištenja operatora negacije)
4. Ispisati tekst "istina je" ako uvjet iz 2. zadatka nije zadovoljen (riješiti sa i bez korištenja operatora negacije)
5. U char varijable c1 i c2 učitana su neka od velikih slova abecede (A-Z). Ispisati tekst "istina je" ako se u c1 i c2 (dakle u obje varijable) nalaze samoglasnici.
6. Ispisati tekst "istina je" ako uvjet iz 5. zadatka nije zadovoljen (riješiti sa i bez korištenja operatora negacije)

Izrazi s različitim tipovima podataka

Pretvorba (konverzija) tipova podataka

Aritmetika s različitim tipovima operanada

```
int i;  
float f;  
i = 2;  
f = 2.99f;
```

Što je rezultat operacije `i + f`

4 ili 4.99

Kada su operandi različitog tipa, prije obavljanja operacije obavlja se implicitna (automatska) pretvorba tipa rezultata u "veći (važniji)" od tipova operanada koji sudjeluju u operaciji.

U prikazanom primjeru, prije nego se obavi operacija zbrajanja, vrijednost koja se nalazi u varijabli `i` pretvara se u 2.0 (pri tome sadržaj varijable `i` ostaje nepromijenjen).

Implicitna (automatska) pretvorba tipova podataka

Pretvorba tipova podataka u izrazima obavlja se prema jednom od sljedećih 5 pravila. Treba iskoristiti prvo po redu pravilo koje se može primijeniti na konkretan slučaj!

1. Ako je jedan od operanada tipa **long double**, preostali operand se pretvara u tip **long double**
2. Ako je jedan od operanada tipa **double**, preostali operand se pretvara u tip **double**
3. Ako je jedan od operanada tipa **float**, preostali operand se pretvara u tip **float**
4. Ako je jedan od operanada tipa **long**, preostali operand se pretvara u tip **long**
5. Operande tipa **short** i **char** pretvoriti u tip **int**

Kada se u izrazima pojavljuju *unsigned* tipovi, pravila pretvorbe su složenija i ovise o implementaciji. Zato se ovdje neće razmatrati.

Primjeri: implicitna pretvorba tipova podataka

```
char c;           int i;           float f;           long double ld;  
short s;          long li;          double d;
```

Operacija	Pretvorba tipa prije obavljanja operacije	Prema pravilu
$ld * i$	sadržaj od $i \rightarrow \text{long double}$	1.
$c \% i$	sadržaj od $c \rightarrow \text{int}$	5.
s / f	sadržaj od $s \rightarrow \text{float}$	3.
$f - ld$	sadržaj od $f \rightarrow \text{long double}$	1.
$li \% i$	sadržaj od $i \rightarrow \text{long}$	4.
$i * f$	sadržaj od $i \rightarrow \text{float}$	3.
$d + c$	sadržaj od $c \rightarrow \text{double}$	2.
$s - c$	sadržaj od $s \rightarrow \text{int}$ sadržaj od $c \rightarrow \text{int}$	5.

Primjeri: implicitna pretvorba tipova podataka

```
float f1, f2;  
int i1, i2; /* pretpostavka int: 4 okteta */  
char c1, c2;  
  
f1 = 32000.5f; f2 = 1.0f;  
i1 = 2147483647; i2 = 1;  
c1 = 127; c2 = 1;
```

<u>Izraz</u>	<u>Rezultat</u>	<u>tip rezultata</u>
f1 + f2	32001.5	float
f1 + i2	32001.5	float
i1 + i2	-2147483648	int
i1 + f2	2147483648.0	float
c1 + c2	128	int

Pretvorba tipova kod pridruživanja

- Vrijednost s desne strane pretvara se u tip podatka varijable ili izraza s lijeve strane znaka za pridruživanje

- Primjer:

```
int i;  
float f;  
i = 2.75;      /* 2.75 se pretvara u int */  
f = 2147483638;
```

- Potrebno je obratiti pozornost da se pri promjeni tipa podatka može "izgubiti" manji ili veći dio informacije. U gornjem primjeru:
 - "izgubljene" su decimale 0.75 kod prvog pridruživanja, tj. ispisom vrijednosti varijable `i` dobilo bi se `2`
 - `f` sadrži vrijednost za 10 veću od one koja je pridružena, tj. ispisom vrijednosti varijable `f` dobilo bi se `2147483648`.

Što kada varijabli pridružimo "prevelik" broj

```
char c1, c2, c3;  
float f1, f2;  
c1 = 80; c2 = 150; c3 = 300;  
f1 = 332.345282347f; ← "previše binarnih znamenaka"  
f2 = 1.0e40f;           ← izvan dopuštenog raspona  
printf("%d\n", c1);      → 80  
printf("%d\n", c2);      → -106  
printf("%d\n", c3);      → 44  
printf("%20.15f\n", f1); → 332.345275878906250  
printf("%f\n", f2);       → inf
```

Eksplicitna (zadana) pretvorba tipa podataka

- Opći oblik zadane (eksplicitne) pretvorbe (eng. *cast operator*) glasi:

(tip_podatka) operand

- Operand može biti varijabla, konstanta ili izraz.
- Zadana pretvorba podataka ima viši prioritet od automatske.

- Primjer:

```
int i;  
double d1, d2;  
i = 2000000000;  
d1 = 2 * i;           → -294967296.000000  
d2 = 2 * (double)i; → 4000000000.000000
```

Cjelobrojno dijeljenje

- Potrebno je obratiti pozornost na "neželjene" rezultate kod cjelobrojnog dijeljenja. Ako se na primjer u realnu varijablu `a` želi pridružiti vrijednost $\frac{1}{2}$, sljedeća naredba pridruživanja **neće** varijabli `a` pridružiti vrijednost 0.5:

`a = 1 / 2;`

- U izrazu s desne strane jednakosti **oba su operanda cjelobrojnog tipa**, pa će se obaviti cjelobrojno dijeljenje. Rezultat tog izraza je 0 (uz ostatak 1).
- Za izbjegavanje cjelobrojnog dijeljenja potrebno je koristiti zadalu pretvorbu tipa (dovoljno samo na jednom operandu) ili zadati konstante tako da je barem jedna realna:

`a = (float) 1 / 2;` ili

`a = 1.f / 2;` ili

`a = 1 / 2.f;`

Napomena: U prvom slučaju korištena je zadana pretvorba tipa operanda (mogla se uporabiti i nad drugim operandom). U drugom i trećem slučaju, dodavanjem točke (ili točke i slova `f`), konstanta je postala realna te se dijeljenje obavlja u realnoj domeni.

Cjelobrojno dijeljenje

- Primjer: Koliko iznosi:

a) $9 \text{ / } 4$ Rješenje: 2 (cjelobrojno dijeljenje)

b) $9 \% 4$ Rješenje: 1 (ostatak cjelobrojnog dijeljenja $9 : 4 = 2$ i ostatak 1)

Prioritet osnovnih aritmetičkih operatora

1. * / %

2. + -

Ako u izrazu ima više operatora jednakog prioriteta, izračunavaju se *sljeva nadesno*. Na primjer:

$$2 + \underbrace{3 / 2}_{\text{1. red}} * 4 - \underbrace{5 * 6}_{\text{2. red}} \% 8$$

$$2 + \underbrace{1 * 4}_{\text{1. red}} - \underbrace{30 \% 8}_{\text{2. red}}$$

$$\underbrace{2 + 4}_{\text{1. red}} - 6$$

$$\underbrace{6 - 6}_{\text{2. red}} = 0$$

Korištenje okruglih zagrade

Kada treba koristiti okrugle zgrade ?

- a) ako se želi promijeniti ugrađeni redoslijed izvođenja operacija
- b) u slučaju vlastite nesigurnosti
- c) radi bolje čitljivosti programa

$$2 + 3 / (2 * 4) - 5 * (6 \% 8) \rightarrow -28$$

Primjeri

- Koliki je rezultat sljedećeg izraza:

$$5 + 10 / 3 * (8 - 6)$$

Rješenje:

$$5 + 10 / 3 * 2$$

$$5 + 3 * 2$$

$$5 + 6$$

$$\underline{11}$$

- Koliki je rezultat svakog od sljedećih izraza:

$$5 + 10 / 3 * (8 - 6.)$$

$$5. + 10 / 3 * (8 - 6)$$

$$5 + 10 / 3. * (8 - 6)$$

Primjeri

- Koju će vrijednost poprimiti varijable i, x, c

```
int i;  
double x, c, d;
```

nakon naredbi:

```
d = 6.0;  
i = (int)(d + 1.73);  
x = i / 2;  
c = (double)i / 2;
```

- Rješenje:

```
i = 7, x = 3.0, c = 3.5
```

Primjeri

Koje su vrijednosti i tipovi rezultata evaluacije sljedećih izraza:

1. `3 / 2 * 2`
2. `3 / 2 * 2.`
3. `3 / 2. * 2`
4. `3 / (float)2 * 2`
5. `(double)3 / 2`
6. `(double)(3 / 2)`
7. `2+0.5 * 4`
8. `(2 + 0.5) * 4`
9. `(int)(0.5 + 2) * 4`
10. `(int)1.6 + (int)1.6`
11. `(int)(1.6 + 1.6)`

Primjer: S tipkovnica učitati tri cijela broja, na zaslon ispisati učitane brojeve i njihovu aritmetičku sredinu.

```
#include <stdio.h>
int main (void) {
    int i, j, k;
    float sredin;
    scanf ("%d %d %d", &i, &j, &k);
    sredin = (i + j + k) / 3.f;
    printf ("Brojevi su %d, %d, %d\n", i, j, k);
    printf ("Aritm. sredina je %f", sredin);
    return 0;
}
```

Rezultat izvođenja programa

Ulazni podaci:

1 3 4

Ispis na zaslonu:

Brojevi su 1, 3, 4

Aritm. sredina je 2.666667