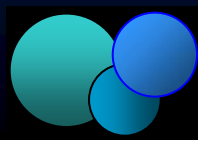
Основи програмирања 1
Лекција 2

Висока школа електротехнике и рачунарства струковних студија
Београд

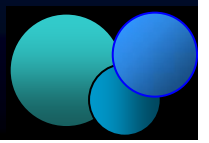
Садржај

- Лексички симболи
- Подаци и промењљиве
- Типови података
- Испис података на стандардни излаз
- Читање података са стандардног улаза



Лексички симболи

- Идентификати
- Службене речи
- Константе

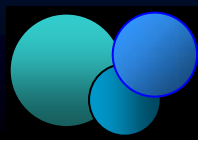


Идентификатори (имена)

- Служе за означавање елемената програма (података, симболичких константи, нових типова података, функција ...)

- Правила

1. Могу да се састоје од:
   - Слова (малих и великих)
   - Цифара
   - Знака подвучено (_)



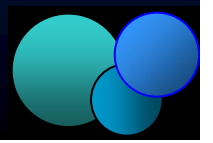
Идентификатори

2. Први знак у имену **мора** да буде слово.
   Може да буде и знак подвучено али...
3. Дужина имена идентификатора да не буде дужа од **31 карактера**

Пример:

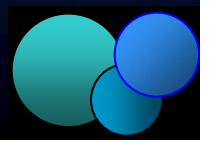
```
JednaPromenljivaKojaImaKarakteraJakoPuno = 1
JednaPromenljivaKojaImaKarakteraSamoJedanVise = 3
```

4. *C* разликује мала и велика слова (**`Rezultat`** и **`rezultat`**) *Case Sensitive*
5. Имена идентификатора не могу да буду из скупа службених речи.



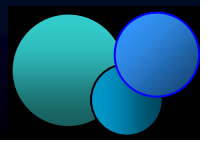
исправни идентификатори

```
suma
c5_4
JEDAN_BROJ
TX_protocol
TRUE

novoIme
novo_ime
Broj_studenata
Broj_Studenata
_split_name

UporediRezultat_sabiranja
dugoimesamnogokaraktera
```

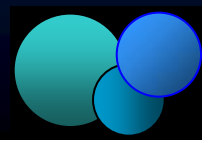
неисправни идентификатори

`70fc`	почиње бројем
`x-name`	знак - није дозвољен
`ime sa razmakom`	размак није дозвољен
`int`	резервисана реч
`axx&`	знак **&** није дозвољен
`Povrsina*kruga`	знак ***** није дозвољен
`jedan,dva`	знак **,** није дозвољен



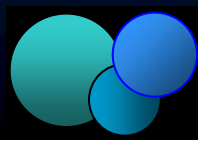
Службене речи

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	~~goto~~	sizeof	volatile
do	if	static	while



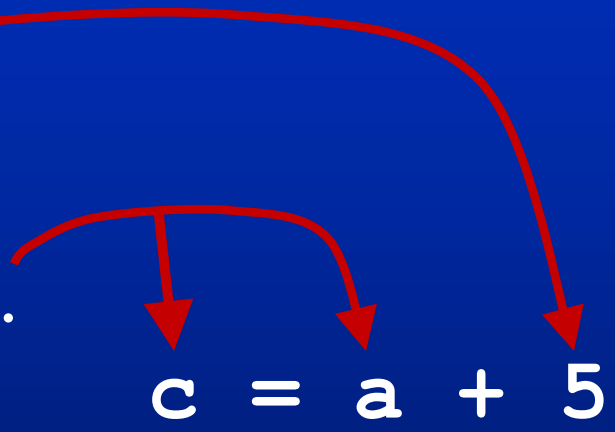
Подаци, Променљиве и типови података

- Подаци су предмет обраде у програмима.
- Сваки податак има одређене особине, а скуп свих особина једног податка: **тип податка**.
- Тип податка је одређен:
  - скупом могућих вредности које податак може да узме и
  - скупом могућих операција које је могуће извести над податком.



Подаци

- Подаци у програму могу да се представе помоћу:
  - вредности или
  - идентификатора.



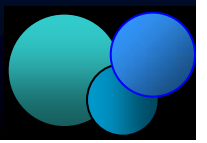
- **Константе:** подаци педстављени преко вредности
- **Промењљиве:** подаци педстављени преко идентификатора

Подаци

- Посебан случај представљају **симболичке константе**.

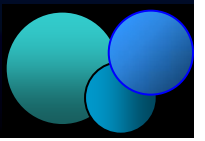
```
#define PI 3.14
#define MIN_RASTOJANJE 2
#define BROJ_SEMESTARA 8
```

Нема знака “ ; “ на крају линије!

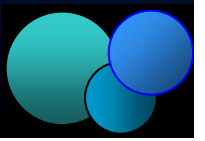


Подаци и меморија

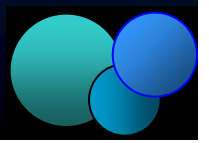
- Вредност променљиве се чува у меморији рачунара, па променљива може да се посматра као део меморије (скуп меморијских локација).
- Постоји једнозначна веза између променљиве и адресе меморијске локације на којој је записана тренутна вредност променљиве.
- Меморија је подељена на одговарајуће целине различитих величина (изражене у бајтовима) и свака та целина има своју адресу.
- Адреса представља број који одређује позицију било ког податка у меморији рачунара.



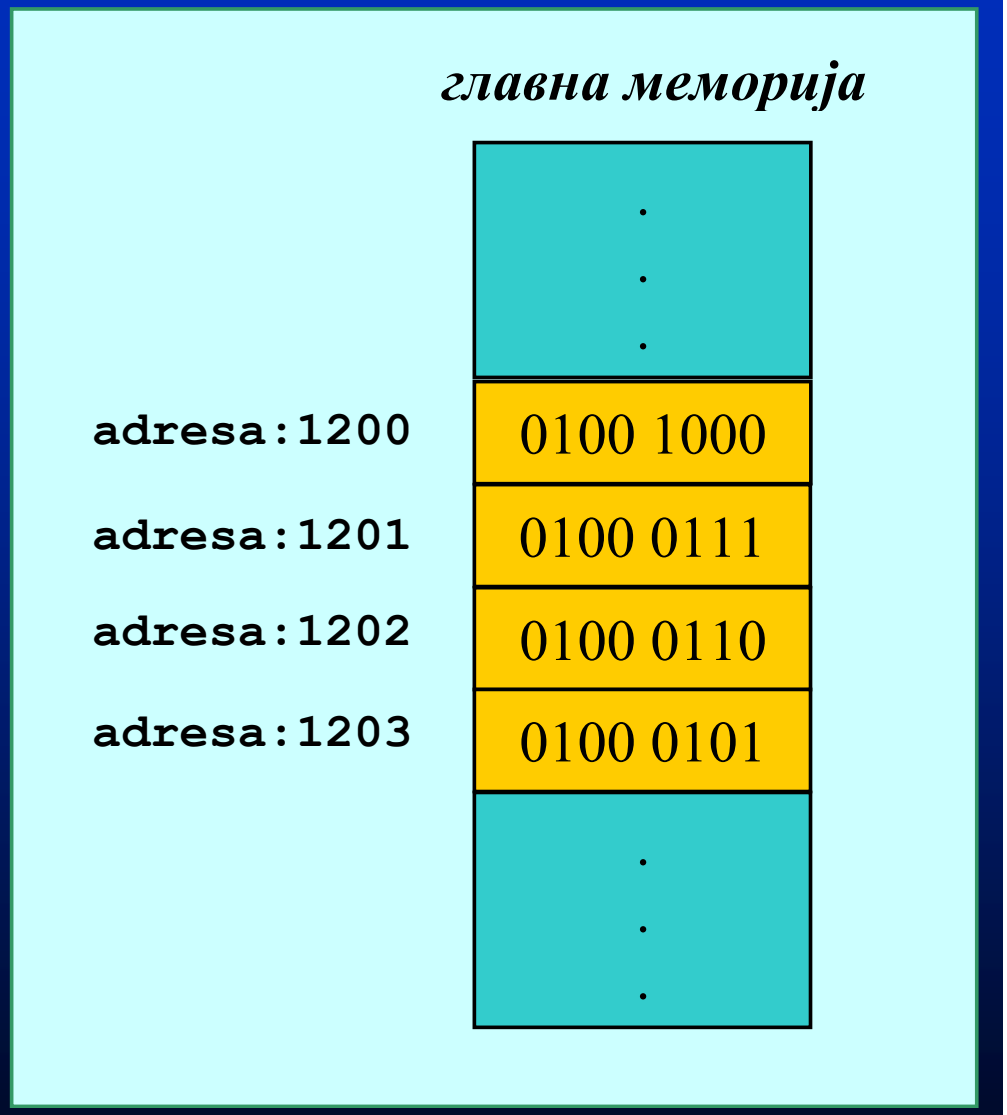
- Садржај дела меморије (бинарни кôд који је у њој записан) представља вредност променљиве.
- Када је програму потребана вредност променљиве, чита се бинарни кôд који је записан у том делу меморије и интерпретира на одговарајући начин.
- Када програм треба да промени вредност променљиве, прво се конвертује вредност променљиве у одговарајући бинарни кôд и потом се уписује у део меморије који припада променљивој.



- у меморију могу да се упишу само бинарне вредности.
- Програм може да обрађује различите врсте података, као што су цели бројеви, реални бројеви, слова ....
- Неопходно је да постоји посебна врста кодовања у бинарни облик за сваку врсту података
- Величина меморијске локације која се додељује једној променљивој одређује дужину бинарне вредности (број бита) која у њој може да се запише.
- Приликом превођења програма, име сваке променљиве се замењује са одговарајућом бројчаном вредношћу – **адресом променљиве**, која једнозначно одређује почетак дела меморије додељеног тој променљивој.
- Адреса променљиве не садржи податак о величини додељене меморије.

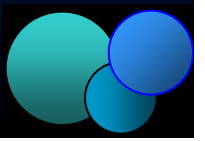


- $0100\ 1000_2$
  72_{10}
- $0100\ 0111\ 0100\ 1000_2$
  $18\ 248_{10}$
- $0100\ 0101\ 0100\ 0110\ 0100\ 0111\ 0100\ 1000_2$
  1162233672_{10}
- $0100\ 0101\ 0100\ 0110\ 0100\ 0111\ 0100\ 1000_2$
  $3.172,455078_{10}$



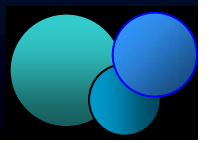
Типови података

- Тип неког податка одређен је:
  - скупом вредности који тај податак може да има и
  - дефинисаним скупом операција над њим.
- Програмски језик *C* познаје само нумеричке типове података
  - **целобројни** нумерички типови
  - **реални** нумерички типови.



Целобројни типови података

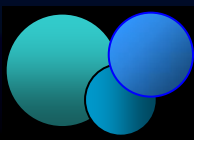
- Име типа:
  - `int`
  - `short int`
  - `long int`
- Опсег вредности: цели бројеви (унутар датог опсега) 5, -35, 401
- Операције: аритметичке (сабирање, одузимање, множење, …), друге



Целобројни типови података

- Сваки од ових типова може да буде неозначен (само позитивне вредности)
  - **`unsigned int`**
  - **`unsigned short int`**
  - **`unsigned long int`**

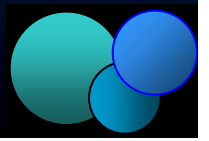
- Ако се не наведе **`unsigned`** сматра се да је означен



Пример опсега вредности

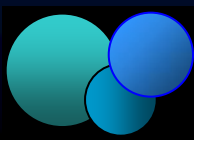
Тип	Бајтова	Бита	Мин вред.	Макс вред.
short int	2	16	-32768	32767
int	4(2)	32	-2147483648	2147483647
long int	4	32	-2147483648	2147483647

не заборавите **`signed`** **`unsigned`**



Зашто постоје ограничења?

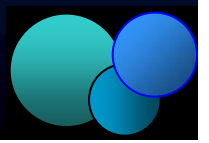
- Са одређеним бројем бита, може да се дефинише одређени број вредности
- 16 бита, 65 536 могућих вредности (2^{16})
  - 32768 негативних вредности
  - 1 нула
  - 32767 позитивних вредности
- Прекорачење (*Overflow*):
  - покушај да се променљивој додели сувише велика вредности (40 000 у **`short int`**)



Целобројни типови података

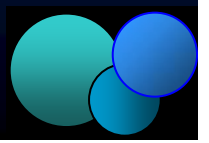
Тип **`char`**

- Мали означени целобројни податак
- величине 8 бита (1 бајт).
- Намењен за рад са карактерима
- може да послужи и за рад са малим бројевима вредности од –128 до 127.
- **`unsigned char`** може да има вредности од 0 до 255.
  (*ASCII* табела има 256 знакова).



Реални типови података

- Име типа:
  - `float (4 bajta)`
  - `double (8 bajtova)`
  - `long double (10? bajtova)`
- Могуће вредности: реални бројеви, 5.0, -3.5, 4.01
- Операције: аритметичке (сабирање, одузимање, множење, …), друге



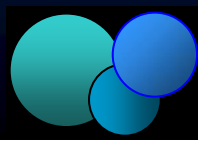
Реални типови података

`float`: 4 бајта, 32 бита

`double`: 8 bajтова, 64 бита

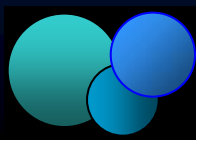
`long double`: 10 бајтова, 80 бита

- Представљање:
  - *m**E**e*
  - m магнитуда (одређени број бита)
    E ознака за степеновање са основом 10
  - *e* експонент



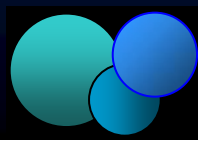
VOID тип података

- Користи се код функција које немају повратну вредност



Нумеричке константе

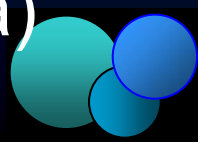
- Целобројне
- Реалне



Целобројне нум. константе

- Могу да буду написане у
  - декадном,
  - хексадецималном (основа 16) почиње са 0x или 0X (нула икс)
  - окталном (основа 8) бројевном систему почиње бројем 0 (нула)

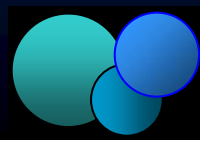
`974`	константа типа int (декадна)
`0x2fc`	константа типа int (хекса)
`0123`	константа типа int (октална)
`974l`	константа типа long int (декадна)
`01234L`	константа типа long int (октална)



Целобројне нум. константе

- `0xffff` негативна вредност
- `0xffffu` позитивна вредност
- `0723` негативна вредност
- `0723u` позитивна вредност

- $\texttt{0xffff}_{(16)}$
- $\texttt{1111 1111 1111 1111}_{(2)}$
- $\texttt{65536}_{(10)}$ **или** $\texttt{-32767}_{(10)}$



Реалне нум. константе

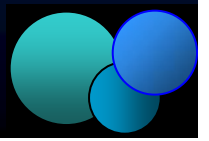
- Могу бити написане у само у декадним бројевном систему.

`2.54E-34` представља вредност $2.54\ 10^{-34}$

`1.23`	`3e5`	`125.`
`125.00`	`2.1E-45`	`.456`
`0.456`	`22.3F`	`2345.67L`

Подразумевани тип је `double`

Додавањем суфикса `F` или `f` постаје `float`

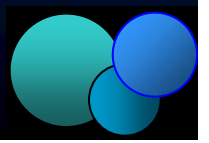


Дефинисање података

- Свака променљива мора да се дефинише пре употребе!

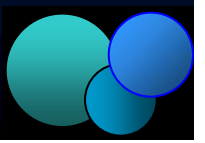
- Општи облик наредбе је

```
tip_podatka ime_podatka;
```

Примери

```
main()
{
   short int broj1;
   char slovo;
   float a;
   int visina;
   float povrsina, pom, obim;
   int kolicina, redni_broj;
   …
}
```

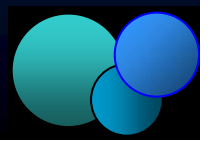
Додела почетне вредности

- Приликом дефинисања података МОЖЕ да се додели почетна (иницијална вредност) податку.

```
tip_podatka ime_podatka = pocetna_vrednost;
```

- Општи облик:

```
tip_podatka ime_podatka[= pocetna_vrednost];
```

Примери

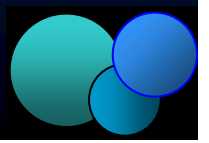
```
main()
{
   short int broj2 = 34, d, gr = 5;
   char slovo;
   float povrsina = a*b;
   float a = 5.43;
   float b = 12.65;
   int visina = 34/2;
   float povrsina = a*b;
}
```

Иницијализација

- је додељивање почетне вредности (*initial value*) променљивој
- Променљива се не мора иницијализовати (вредност је у том случају случајна?)

Пример:

```
int broj=2, prisutni=0, godina=2004, br_sati;
```

Оператор доделе вредности

оператор `=`

Пример:

```
broj = 2;
```

- **На меморијску локацију која је именована као `broj` упиши вредност 2**
- **При додели вредности са ЛЕВЕ стране мора да се налази променљива**
- **Увек се додељује вредност која се налази на ДЕСНОЈ страни**

```
2 = broj;
```

Вредност променљиве `broj` увећај за 2

```
broj = broj + 2;
```

Пример

```
int main()
{   /* pocetak funkcije main() */

    /* deklaracija i inicijalizacija */
    int umanjenik, razlika;
    int umanjilac = 12;

    /* dodela vrednosti */
    umanjenik = 34;

    /* dodela vrednosti */
    razlika = umanjenik - umanjilac;

    return 0;
}   /* kraj funkcije main() */
```

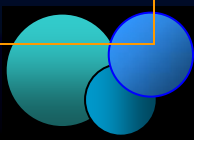
Прво се израчуна израз на десној страни једнакости, потом се израчуната вредност додели промењливој на левој страни

Овде дописати део програма за приказивање вредности промењљиве **razlika** на монитор.

Дефинисање непроменљивих података

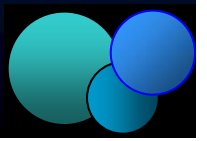
- Непроменљиви подаци су именовани подаци чија се вредност не мења.
- Дефинишу се на сличан начин с тим да:
  - дефинисање почиње са кључном речи **`const`**
  - мора бити додељена почетна вредност
- Општи облик дефинисања непроменљивих података

```
const tip_podatka ime_podatka =
                              pocetna_vrednost;
```

Дефинисање непроменљивих података

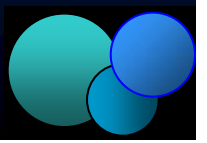
```
const double e = 2.71828182845905;

const int br_radnih_dana = 26;

const float radijus =1 4;

const float cena = 14.2, porez = 0.22;
  /* i porez je konstanta*/

const float obim = 2*radijus*3.14;
```

Величина податка

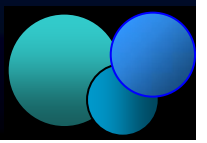
- Величина меморијске локације коју заузима неки податак се може израчунати употребом оператора **`sizeof`**.
- Примери:

```
sizeof( short int);
sizeof( long int);
sizeof( broj_a );
sizeof( broj_a*broj_b-3 );
```

Читање и писање података

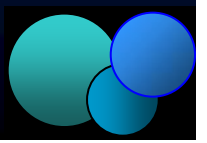
- Сви подаци који се уносе са тастатуре се прихватају као знакови (*character*).
- Улазна конверзија се у програмском језику *C* обавља преко уграђене функције **`scanf()`**
- Приликом исписа података на монитор потребно је да се обави излазна конверзија,
- Ова конверзија се обавља применом уграђене функције **`printf()`**



Читање и писање података

- Информације о функцији **`printf()`** налази се у фајлу **`stdio.h`**.
- Име *stdio* потиче од ***st**and**d**ard **i**nput **o**utput* а екстензија .h од ***header*** (заглавље).

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

Приказ вредности података

```
printf("Ovo je jednostavna poruka")
```

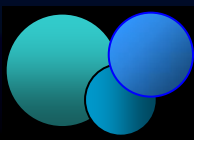
- Резултат: `Ovo je jednostavna poruka`

```
printf( format,izraz,...,izraz)
```

- **`izraz`** може да представља име променљиве, вредност константе или аритметички израз

`format` има облик:

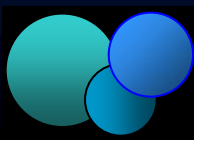
`"%[Sirina][.Prec]Tip_kon"`



Примери исписа и анализа

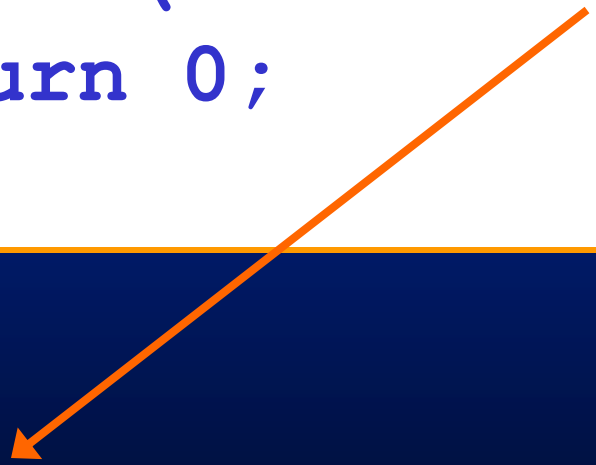
```
#include <stdio.h>
/* prikaz jedne linije teksta */
int main( )
{
   printf("Zdravo!");
   return 0;
}
```

```
Zdravo!Press any key to continue
```

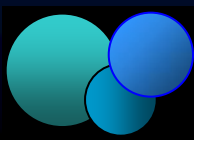



Примери исписа и анализа

```
#include <stdio.h>
/* prikaz jedne linije teksta */
int main( )
{
   printf("Zdravo! ");
   return 0;
}
```



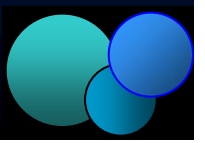
```
Zdravo! Press any key to continue
```

Примери исписа и анализа

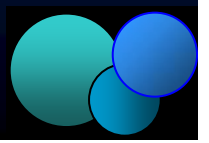
```
#include <stdio.h>
/* prikaz jedne linije teksta */
int main( )
{
   printf("\nZdravo!\n\n");
   return 0;
}
```

```
1
2 Zdravo!
3
4 Press any key to continue
```



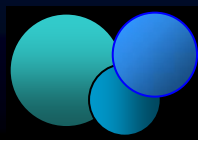
"%[Sirina][.Prec]Tip_kon"

Tip_kon	Тип	Излазни формат
i	int	цео број (декадни, октални, хекса)
d	int	декадни цео број
u	int	неозначени цео број
o	int	означени октални број
x	int	означени хексадецимални „abcdef”
X	int	означени хексадецимални, „ABCDEF”



"%[Sirina][.Prec]Tip_kon"

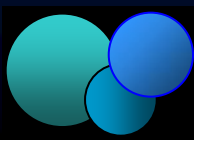
Tip_kon	Тип	Излазни формат
c	int	карактер дефинисан једним бајтом (ASCII)
e	float	реални број једноструке тачности, експ. облик
f	float	реални број једноструке тачносити стд. облик
g	float	реални број једноструке тач. облика f ,e у зависности од тога који је погоднији
lf	double	реални број двоструке тачности.
s		низ карактера (стринг)
p		вредност показивача (адреса)



Пример

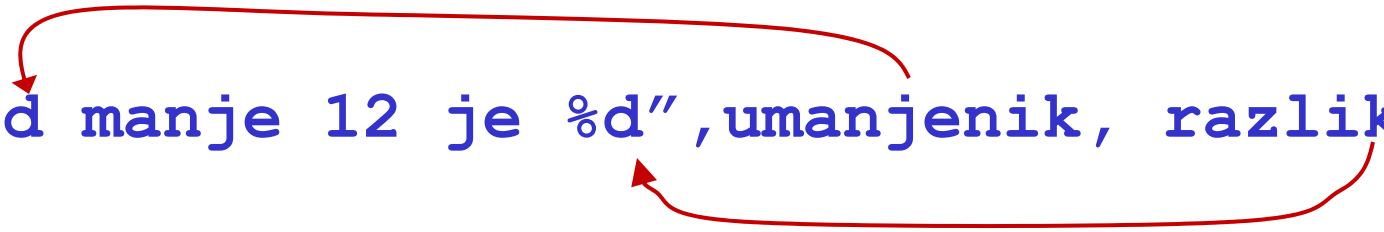
```
int main()
{
   int prom = 10;

   printf("a) Najmanji dvocifren broj je 10 \n");
   printf("b) najmanji dvocifren broj je %d ", 10);
   printf("c) najmanji dvocifren broj je %d ", prom);


   return 0;
}  /* kraj funkcije main() */
```

```
a) Najmanji dvocifren broj je 10
b) Najmanji dvocifren broj je 10
c) Najmanji dvocifren broj je 10
```

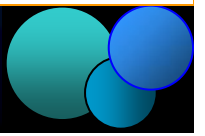



Пример

```
#include <stdio.h> /* zbog printf() */
int main()
{
  /* deklaracija i inicijalizacija */
  int umanjenik, razlika;
  int umanjilac = 12;

  umanjenik = 34;

  razlika = umanjenik - umanjilac;

  printf("%d manje 12 je %d",umanjenik, razlika );

  return 0;
}  /* kraj funkcije main() */
```

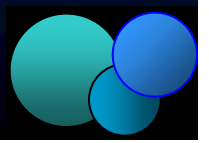


34 manje 12 je 22



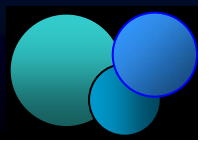
Приказивање података

- `printf("%c ", 'A');`
  резултат је: **A**
- `printf("%c %d", 'A', 35);`
  резултат је: **A 35** размак између **%c** и **%d**
- `printf("%c%d", 'A', 35);`
  резултат је: **A35** овде нема размака.
- `printf("%c %d %f", 'A', 35, 4.5);`
  резултат је: **A 35 4.500000**
- `printf("%c %d %e", 'A', 35, 42.123);`
  резултат је: **A 35 4.212300e+001**



"%[Sirina][.Prec]Tip_kon"

- **.Prec** представља прецизност исписа значај само за реалне бројеве: (**e, f, g**)
- одређује број цифара које ће бити исписане након децималне тачке.
- Подразумевана вредност је 6.
- Уколико је наведена прецизност мања од броја децимала: исписује се онолико цифара колико прецизност дефинише а последња цифра се **заокржује.**
- За тип конверзије **g**: одређује укупан број цифара који ће се исписати (целобројни +децимални део).

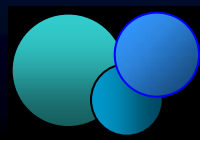


"%[Sirina][.Prec]Tip_kon"

- `printf("%e", 42.126);`
  резултат је: 4.212600e+001

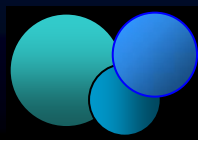
  Подразумевана вредност је 6 децимала
- `printf("%.4e" 42.126);`
  резултат је: 4.2126e+001

  Исписује онолико децимала колико је наведено
- `printf("%.3e", 42.126);`
  резултат је: 4.213e+001
  Заокруживање последње децимале



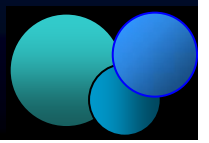
"%[Sirina][.Prec]Tip_kon"

- **`printf("%f", 42.126);`**
  резултат је: 42.126000
  Подразумевана вредност је 6 децимала
- **`printf("%.7f", 42.12345679);`**
  резултат је: 42.1234568
  Заокруживање
- **`printf("%.2f", 42.126);`**
  резултат је: 42.13
  Заокруживање



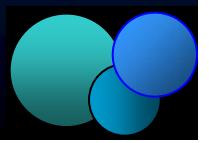
"%[Sirina][.Prec]Tip_kon"

- `printf("%g", 42.126);`
  rezultat je: 42.126
- `printf("%.7g", 42.3456789);`
  rezultat je: 42.34568
- `printf("%.2g", 42.126);`
  rezultat je: 42
- `printf("%.2g", 142.126);`
  rezultat je: 1.4e+002



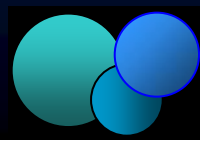
"%[Sirina][.Prec]Tip_kon"

- **Sirina** Одређује минимални број карактера који се резервише за испис неког броја. То је позитивна вредност. Уколико је за испис потребно мање карактера од броја који је одређен параметром **Sirina,** испис се допуњава бленковима (размацима) и поравнава се у десно.
- Уколико се испред параметра **Sirina** наведе 0 бленкови се, уколико постоје, замењују нулама.



"%[Sirina][.Prec]Tip_kon"

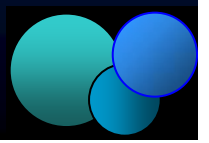
- `printf("%5c%4d", 'A', 35);`
  резултат је: A 35 четири бленка испред A и два бленка испред 35
- `printf("%04d", 35);`
- резултат је: 0035 нуле замењују празна места
- `printf("%07.3f", 42.123);` /*042.123*/ укупно 7 цифара од тога три цифре након децим. тачке и једна за децим. тачку */



Додатно форматирање

escape sequence

```
\a – bell                \\ -- backslash
\b – backspace           \? – question mark
\f – formfeed            \’ – single quote
\n – new line            \” – quote mark
\r – CR                  \o# – # in octal
\t – horiz. tab          \x# – # in hex
```

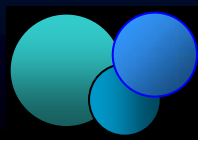
```
printf("%c\n\t%d\n%f\n", 'A',35, 42.126);
```

резултат је:

A

  35

42.126000



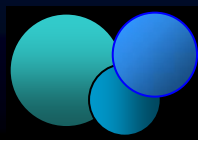
Читање података

- Општи облик наредбе је следећи:

```
scanf(format,&podatak,...,&podatak)
```

- Оператор & испред имена променљиве означава да се функцији **`scanf()`** додељује меморијска адреса променљиве (податка) на коју се уписује резултат конверзије.

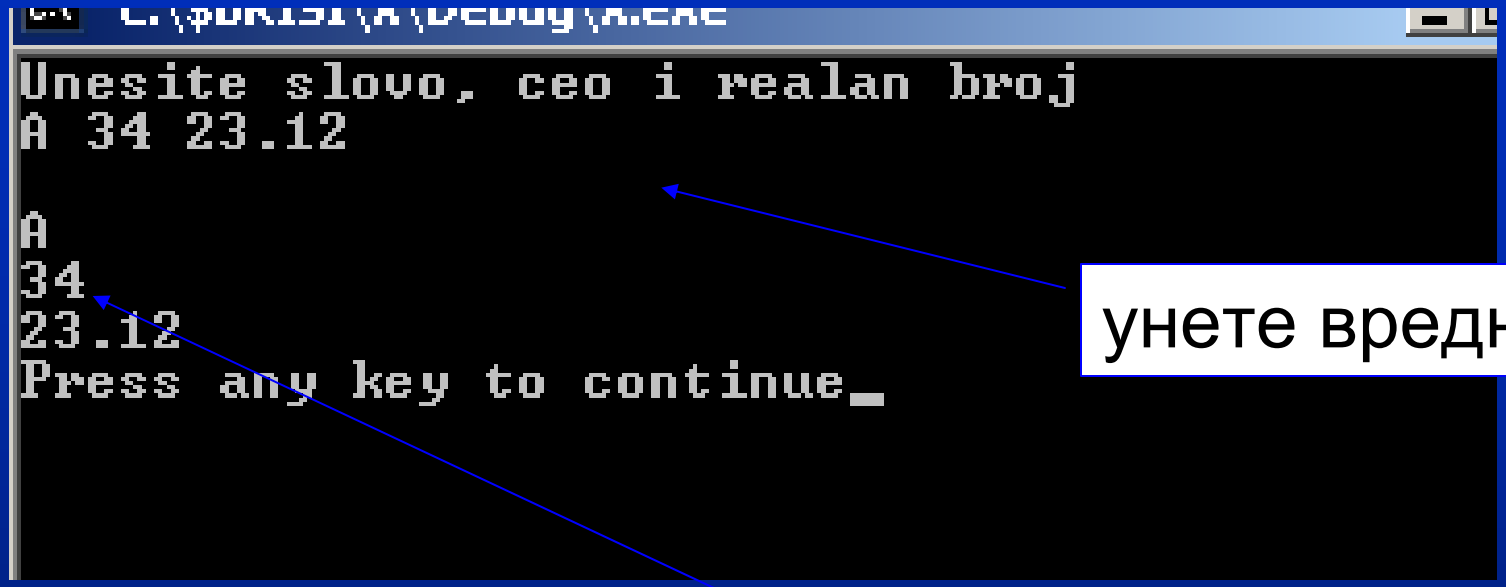
- **`format`** има исти облик као и код функције **`printf()`**.



пример 1

```
#include <stdio.h>
int main()
{
   char  cVar;
   int   dVar;
   float fVar;

   printf("Unesite slovo, ceo i realan broj\n");

   scanf("%c %d %f", &cVar,&dVar, &fVar);
   printf("\n%c \n%d \n%f\n",cVar, dVar, fVar);

   return 0;
}
```

пример 1



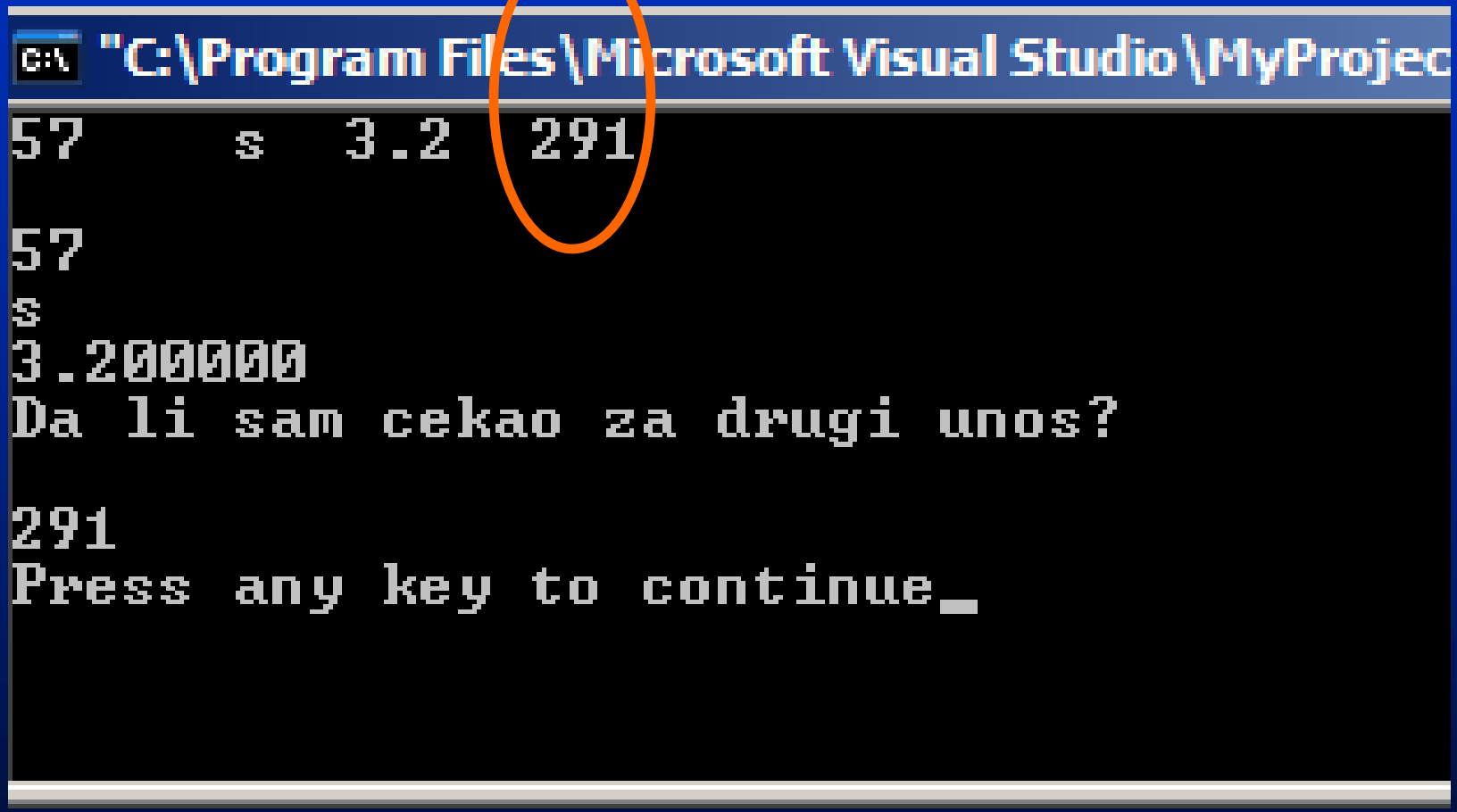
унете вредности

излазне вредности

пример 2

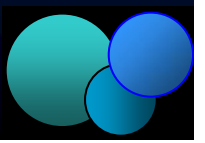
```
#include <stdio.h>
main()
{
  int x;
  char c;
  float y;
  printf(“Unesi ceo br. Slovo, realan br.\n);
  scanf("%d %c %f", &x, &c, &y);
  printf("\n%d \n%c \n%f\n", x, c, y);
  scanf("%d", &x);
  printf("Da li sam cekao za drugi unos?\n");
  printf("\n%d\n",x);
}
```

пример 2


"C:\Program Files\Microsoft Visual Studio\MyProjec
57 s 3.2 291
57
s
3.200000
Da li sam cekao za drugi unos?
291
Press any key to continue_

пример

```
#include <stdio.h>
main()
{
  int x;
  char c;
  float y;

  scanf("%d %c %f", &x, &c, &y);
 fflush(stdin);
  ...
}
```

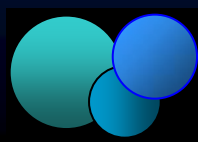
Конверзија типа

- Експлицитном наредба за промену типа применом тзв. ***cast*-оператора**.
- Његов општи облик је:

```
(novi_tip) izraz
```

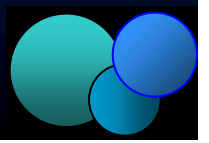
```
int a;
(char)a;
```

```
float x;
double y;
y=sin((double)x);
```

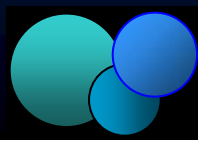
Анализа примера

```
#include  <stdio.h>

int main( )
{
   char vel;

   vel = sizeof(char);
   printf("\nchar zauzima %d B memorije", vel);

   return 0;
}
```

Анализа примера

```
#include  <stdio.h>

int main( )
{
  printf("\nchar zauzima %d B   memorije\n",
                                  sizeof(char));
  return 0;
}
```

Анализа примера

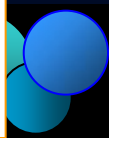
```
#include <stdio.h>

int main( )
{   int x;

    printf("\nUnesite ceo broj:\t");
    scanf("%d", &x);
    fflush(stdin);
    printf("\nDecimalni oblik:\t%d",x);
    printf("\nOktalni oblik:\t\t%o",x);
    printf("\nHeksa oblik:\t%x (%X)", x,x);

    return 0;
}
```

Анализа примера

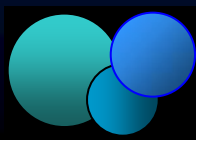
```
#include <stdio.h>

  int main( )
  {
     double x;

      printf("\nUnesite jedan realan broj: ");
      scanf("%lf", &x);

      printf("\nOblik sa decimalnom tackom:\t%f",x);
      printf("\nOblik sa eksponentom:\t\t%e",x);
      printf("\nJednostavniji oblik:\t\t%g\n\n",x);

     return 0;
  }
```

Анализа примера

```
#include  <stdio.h>
  main( )
  {
     const int x = 999;
     const double y = 1.2345;

     printf("\nCeo broj:\n\n");
      printf("%10d\n", x);
      printf("%-10d\n", x);
      printf("\nRealan broj:\n\n");
      printf("%10f%\n", y);
      printf("%-10f\n", y);
      printf("%10.2f\n", y);
      printf("%.2f\n", y);
  }
```



```
Ceo broj:

        999
999

Realan broj:

   1.234500
1.234500
      1.23
1.23
Press any key to continue
```

Анализа примера

```
#include  <stdio.h>
main( )
{
   int a, b, temp;
  printf("\nUnesite dva cela broja: ");
  scanf("%d%d", &a, &b);
  printf("\nPre zamene:\ta=%d, b=%d", a, b);

  temp = a;
  a = b;
  b = temp;

  printf("Posle zamene:a=%d, b=%d\n", a, b);
}
```

Анализа примера

```
#include  <stdio.h>
#define  PI  3.14159f

int main( )
{
  float r, obim, povrs;

  printf("\nUnesite poluprecnik:  ");
  scanf("%f", &r);
  obim = 2 * r * PI;
  povrs = r * r * PI;
  printf("\nObim kruga je %.2f", obim);
  printf("\nPovrsina je %.2f\n\n", povrs);

  return 0;
}
```

Анализа примера

```
#include  <stdio.h>
int main( )
{
  double fahr, cel;

  printf("\nUnesite T u Far. ");
  scanf("%lf", &fahr);

  cel = (5.0 / 9.0) * (fahr - 32.0);
  printf("\nTemperatura od %.2lf F ", fahr);
  printf("je isto sto i %.2lf C", cel);

  return 0;
}
```

Хвала на пажњи

Питања?