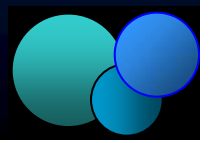
Основи програмирања 1
Лекција 3

Висока школа електротехнике и рачунарства струковних студија
Београд

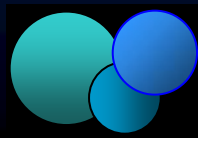
Садржај

- Оператори
- Контрола тока програма
- Анализа примера



Оператори

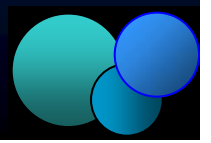
- Радње које се извршавају над операндима и које дају одређени резултат.
  - Аритметички оператори
  - Оператор доделе вредности
  - Релацијски оператори
  - Логички оператори
  - Оператори по битовима
  - Величина податка
  - Условни израз



Аритметички оператори

Бинарни оператори

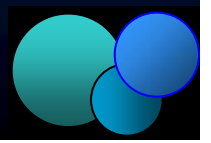
Име	Оператор	Пример
Сабирање	`+`	`num1 + num2`
Одузимање	`-`	`broj - cena`
Множење	`*`	`kolicna * 6`
Дељење	`/`	`suma / kolicina`
Модуо	`%`	`m % n`



Модуо - Остатак при дељењу

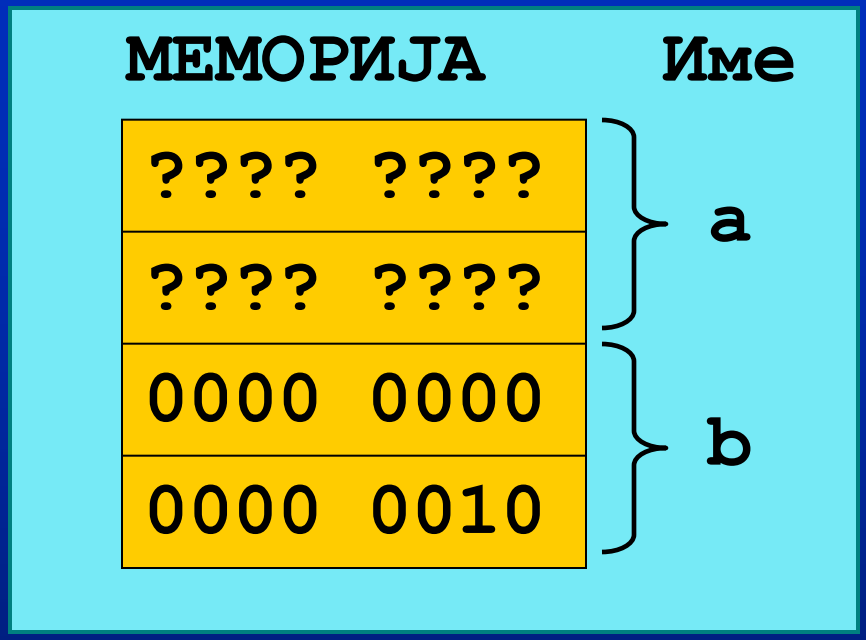
- Израз `m % n` даје целобројни остатак дељења `m` са `n`.
- Оба операнда МОРАЈУ бити целобројна
- Примери :

```
17 % 5   (резултат  2)
 6 % 3   (резултат  0)
 9 % 2   (резултат  1)
 5 % 8   (резултат  5)
```

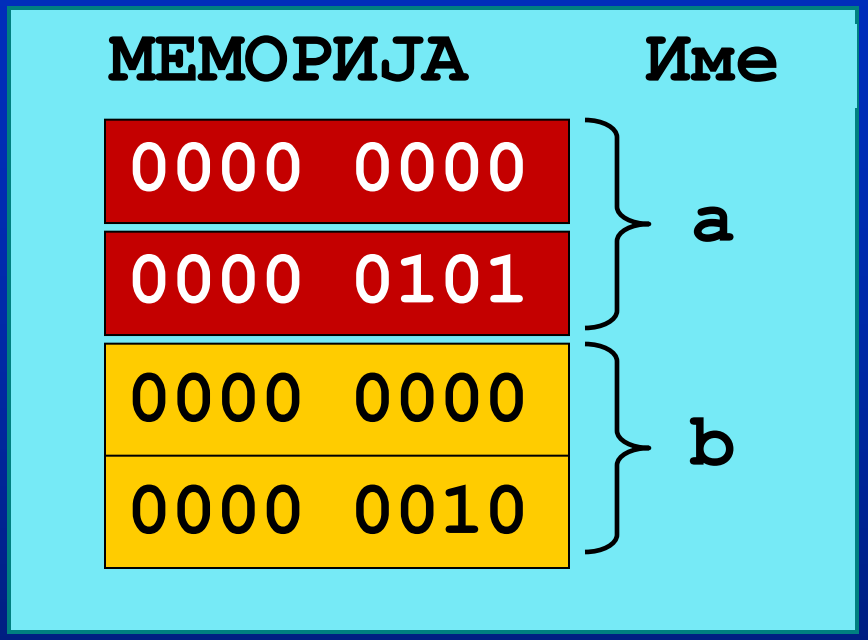
Пример

```
#include<stdio.h>
int main( )
{
  short int a;
  short int b = 2;

  a = b + 3;
  b = 14 - 5;
  a = b / 3;
  b = 5 % 4 ;

  return 0;
}
```



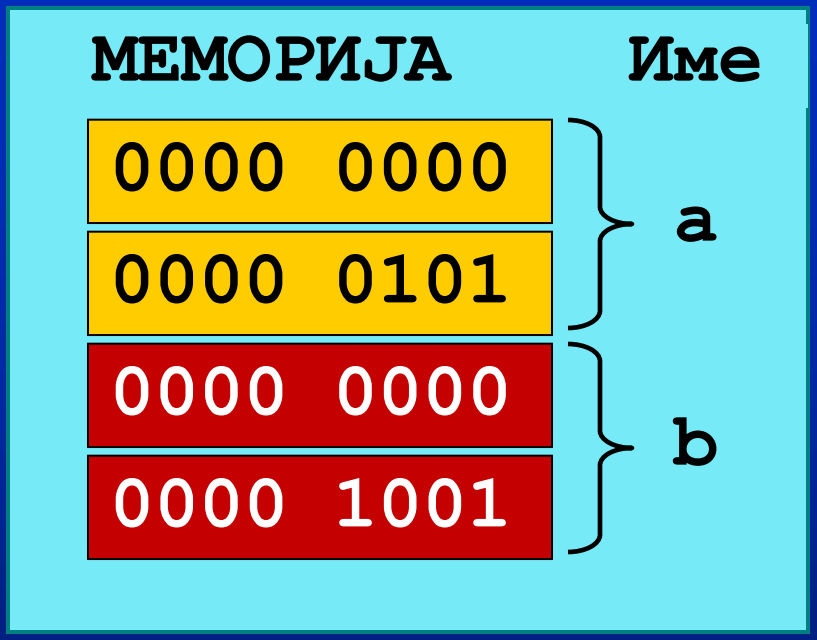
Пример

```
#include<stdio.h>
int main( )
{
   short int a;
   short int b = 2;

   a = b + 3;
   b = 14 - 5;
   a = b / 3;
   b = 5 % 4 ;

   return 0;
}
```



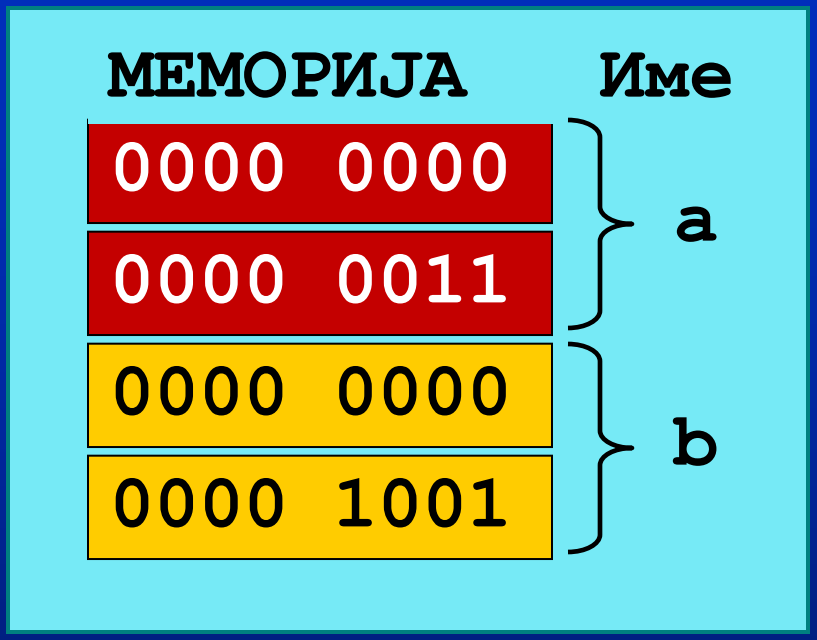
Пример

```
#include<stdio.h>
int main( )
{
   short int a;
   short int b = 2;

   a = b + 3;
   b = 14 - 5;
   a = b / 3;
   b = 5 % 4 ;

   return 0;
}
```



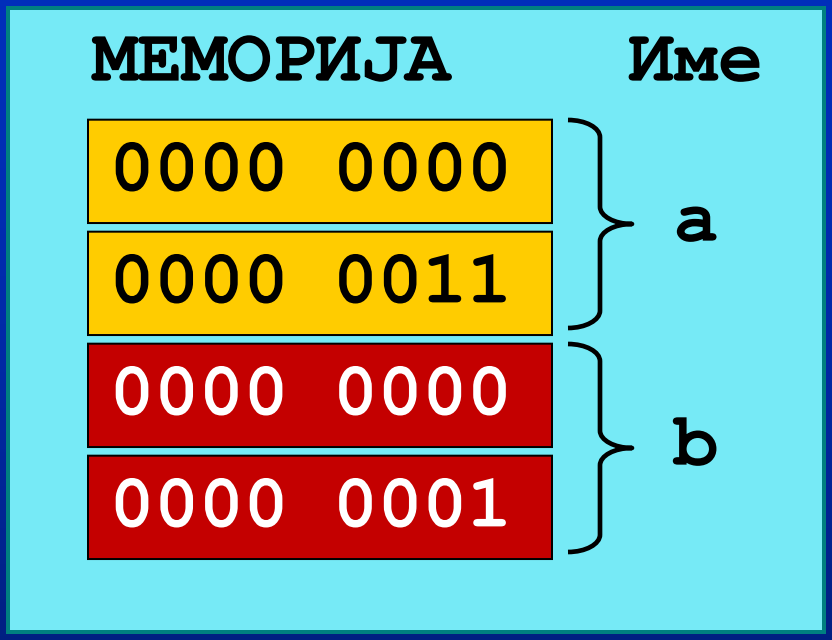
Пример

```
#include<stdio.h>
main( )
{
   short int a;
   short int b = 2;

   a = b + 3;
   b = 14 - 5;
   a = b / 3;
   b = 5 % 4 ;

   return 0;
}
```



Пример

```
#include<stdio.h>
main( )
{
   short int a;
   short int b = 2;

   a = b + 3;
   b = 14 - 5;
   a = b / 3;
   b = 5 % 4 ;

   return 0;
}
```



Дељење

- Ако су оба операнда при дељењу типа **`int`** онда ће и резултат бити типа **`int`**.

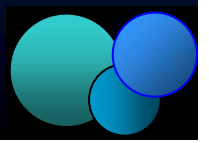
```
17 / 5 (резултат  3)
 4 / 3 (резултат  1)
```

```
#include <stdio.h>
main()
{
      int x = 5, y = 2;
      float z;
      z = x/y;                  /* z = 2.0 */
      print("rezultat je %.2f\n", z);
}
```

rezultat je 2.00

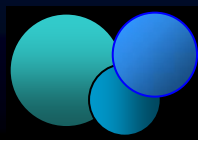
Дељење

- Ако је бар један операнд типа **`float`** резултат ће бити типа **`float`**.
- Примери: **`17.0/5`** **`(резултат 3.42)`**
  `4/3.2` **`(резултат 1.25)`**
  `35.2/9.1` **`(резултат 3.86813)`**
- Објашњење?
  Целобројни (**`int`**) операнд се привремено конвертује у реалну вредност (**`float`**), па се онда обави дељење.



Дељење нулом

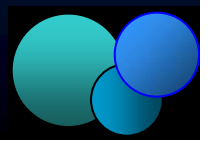
- Дељење нулом није математички дефинисано.
- Ако у току извршавања програма дође до дељења нулом то ће проузроковати прекид извшења програма ***fatal error***.



Аритметички оператори

- **Унарни оператори: `+` , `-`**

  Представљају предзнак операнда
- **Унарни оператори: `++` , `--`**
  - Оператор `++` увећава вредност операнда за 1 (*increment*).
  - Оператор `--` умањује вредност операнда за 1 (*decrement*).

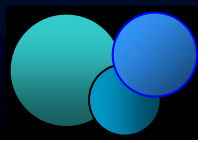


Унарни оператори: ++, --

```
a++;  /* odgovara naredbi a = a+1; */
++a; /* odgovara naredbi a = a+1; */
a--; /* odgovara naredbi a = a-1; */
--a; /* odgovara naredbi a = a-1; */
```

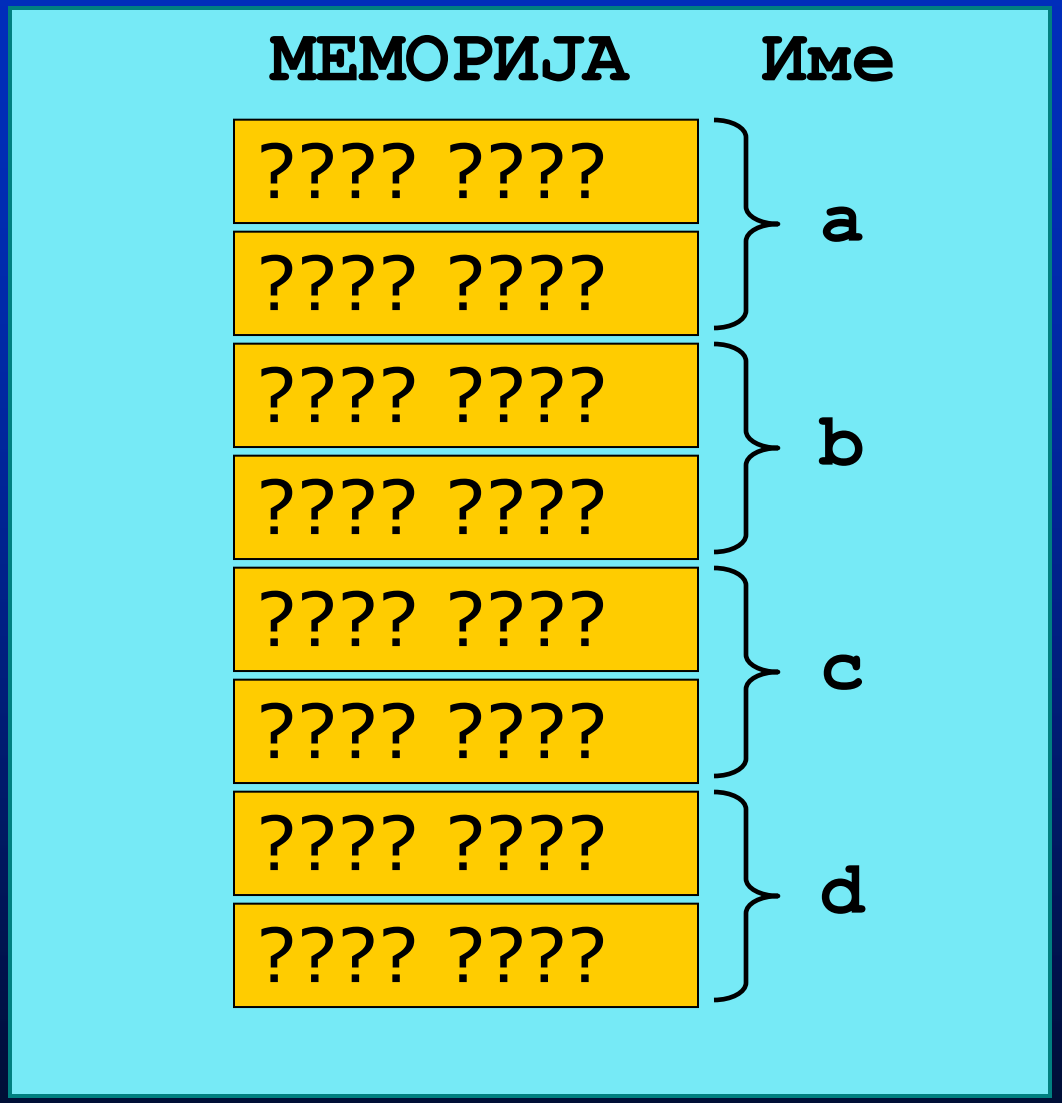
a++ не мора да буде исто са **++a**

a-- не мора да буде исто са **--a**



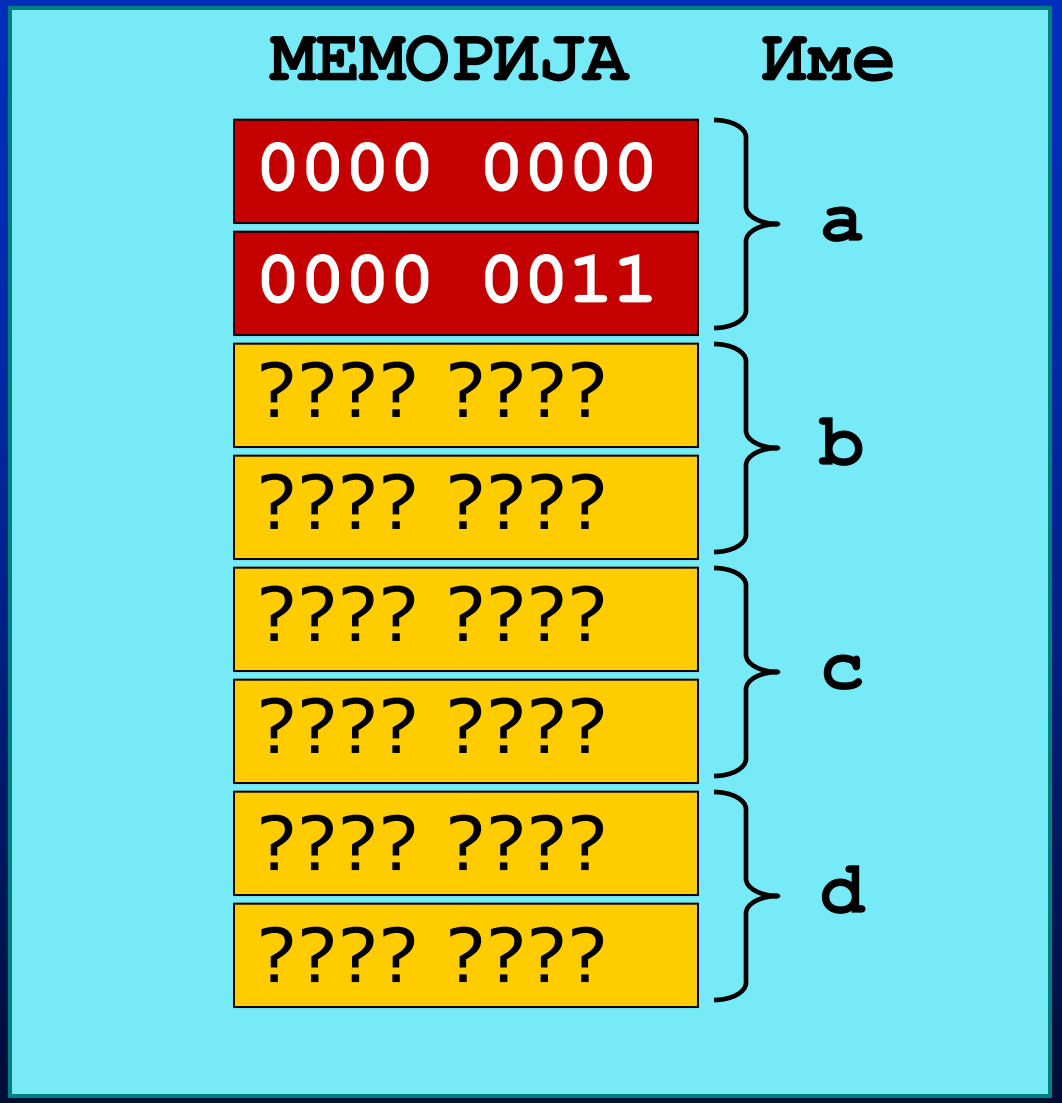
Унарни оператори: ++, --

```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```



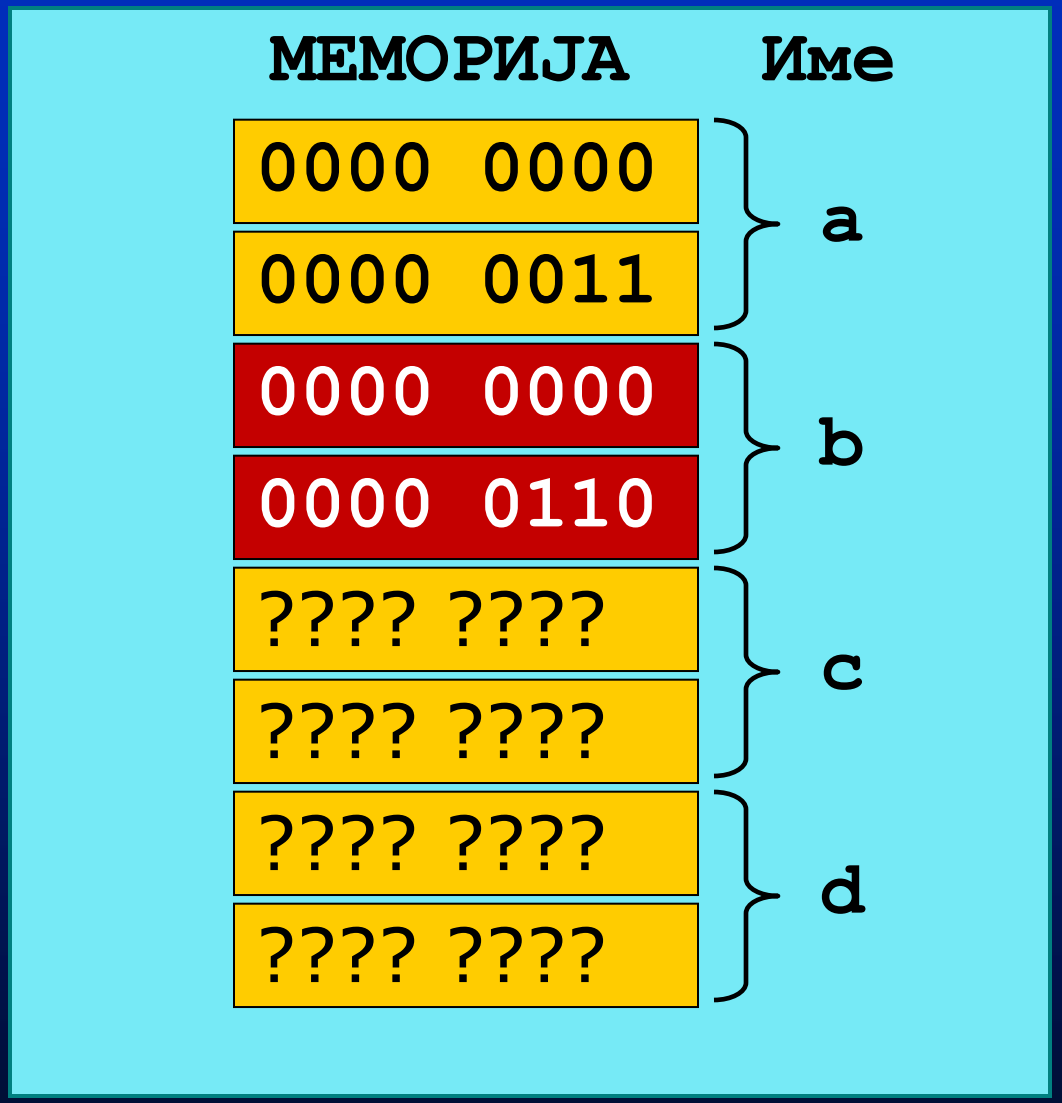
Унарни оператори: ++, --

```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```



Унарни оператори: ++, --

```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```

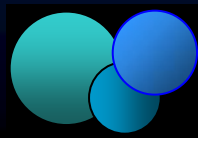


Унарни оператори: ++, --

```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```

```
c = b;
b = b + 1;
```

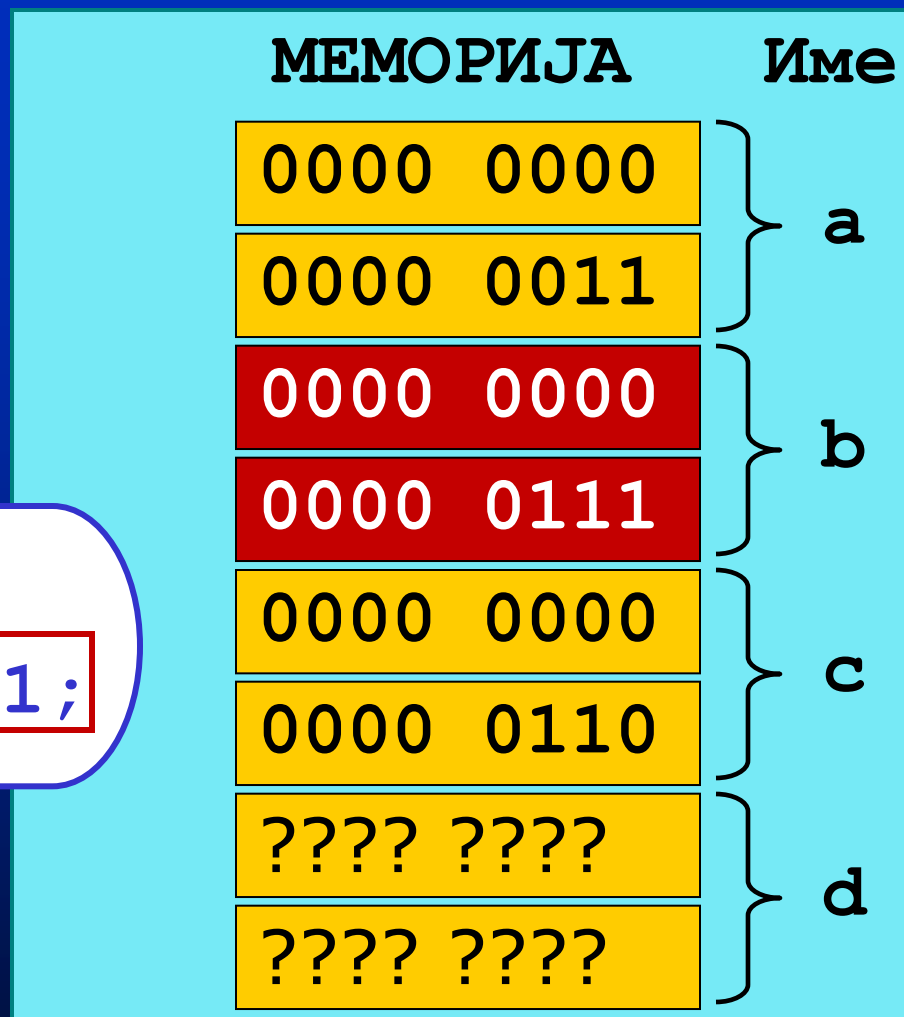
МЕМОРИЈА	Име
0000 0000	a
0000 0011	a
0000 0000	b
0000 0110	b
0000 0000	c
0000 0110	c
???? ????	d
???? ????	d



Унарни оператори: ++, --

```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```

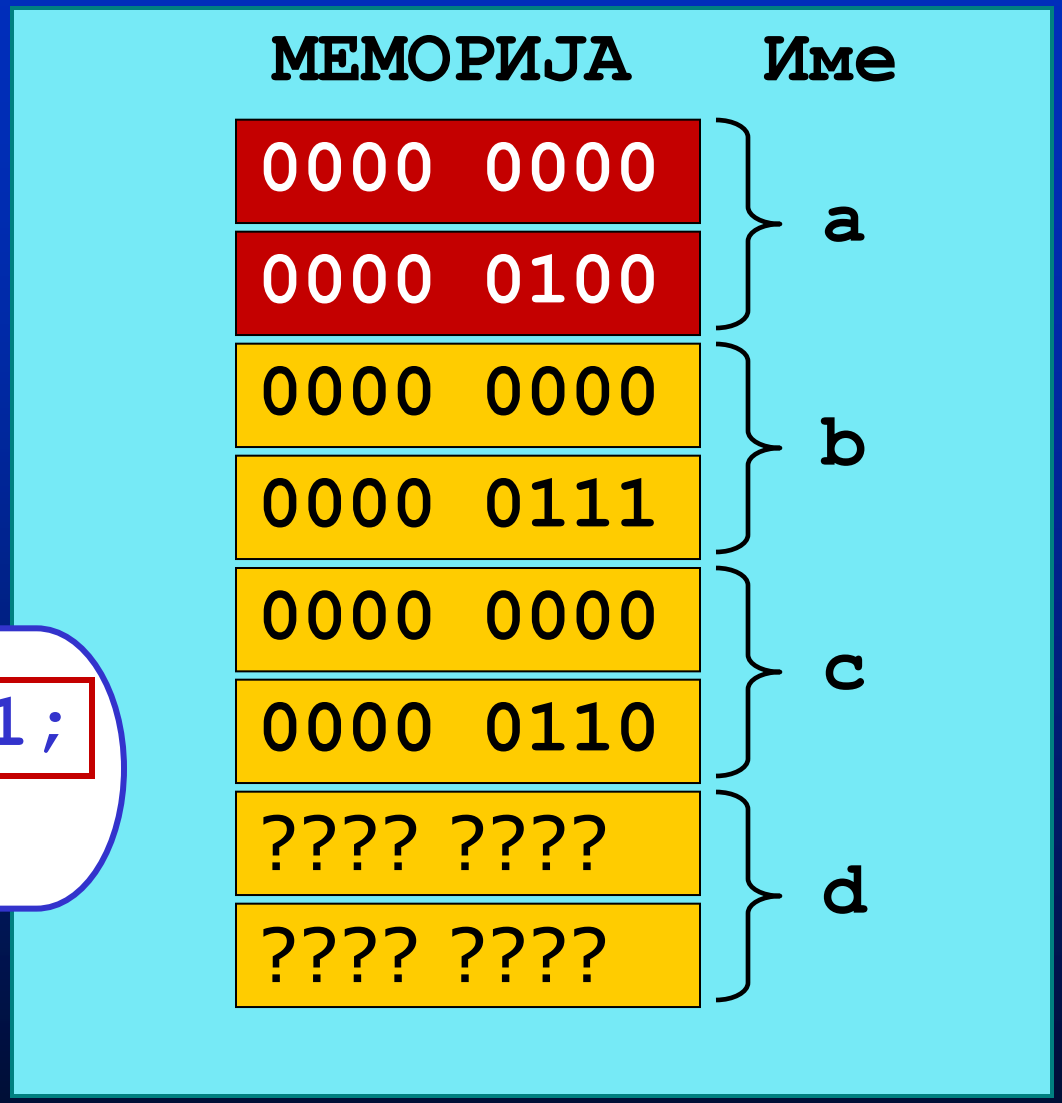
```
c = b;
b = b + 1;
```



Унарни оператори: ++, --

```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```

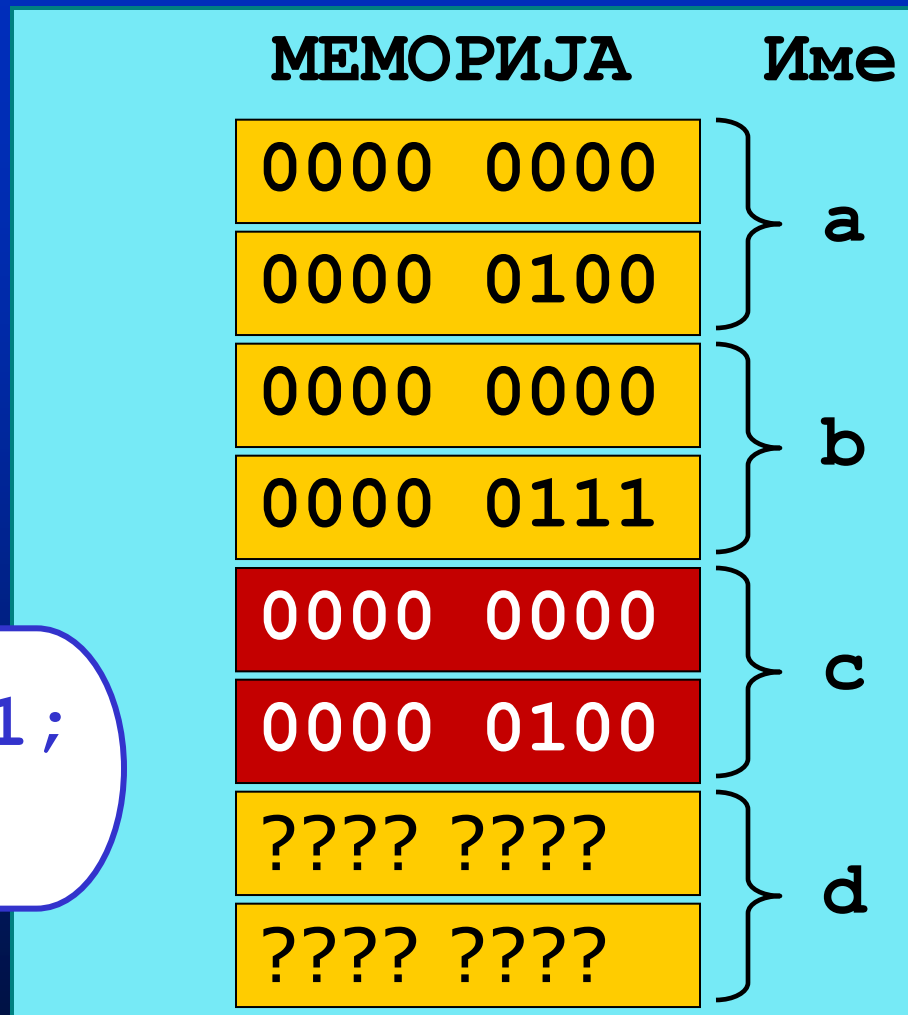
```
a = a + 1;
c = a;
```



Унарни оператори: ++, --

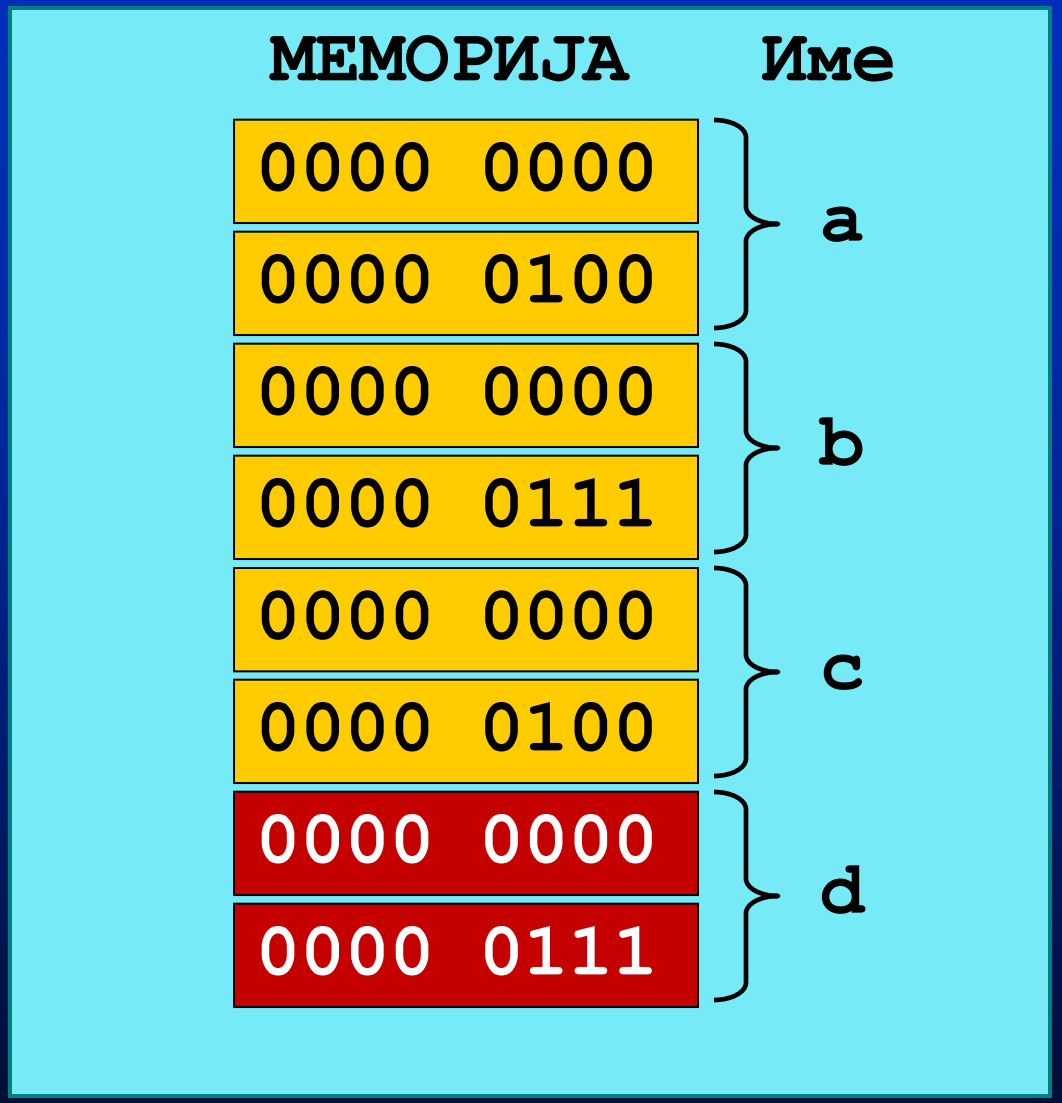
```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```

```
a = a + 1;
c = a;
```



Унарни оператори: ++, --

```
int main()
{
 short int a, b;
 short int c, d;
 a = 3;
 b = 6;
 c = b++;
 c = ++a;
 d = b;

 return 0;
}
```



Оператор доделе вредности

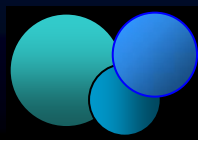
operatori

= += -= *= /= %=

^= &= |= &&= ||=

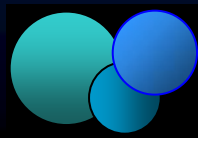
Општи облик

```
ime_promenljive [operator]= izraz
```

Оператор доделе, пример

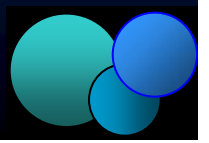
```
broj_a = 3;
broj_c = broj_b;
a = a + 1;
a += b;    /* isto je sto i a = a + b; */
a -= b;    /* isto je sto i a = a - b; */
a *= b;    /* isto je sto i a = a * b; */
a /= b;    /* isto je sto i a = a / b; */
a %= b;    /* isto je sto i a = a % b; */
a = b = c = 27;
b += c+2; /* isto je sto i b=b+(c+2); */
d *= e-5; /* isto je sto i d=d*(e-5); */
```

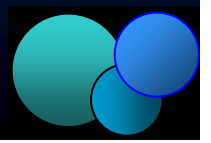
Релацијски оператори

`>`	јесте веће од	`a > b`
`<`	јесте мање од	`a < b`
`>=`	јесте веће или једнако	`a >= b`
`<=`	јесте мање или једнако	`a <= b`
`==`	јесте једнако	`a == b`
`!=`	није једнако	`a != b`

- Резултати поређења су целобројног типа (`int`) и могу имати вредност:
  - **1 ако релација важи или**
  - **0 уколико релација не важи.**

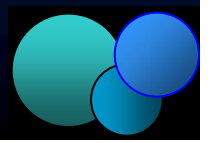


```
int main()
{
 int a, b = 7, c = 0;

 a = (b > c);        /* Tacno, a=1 */
 a = b < 3;          /* Netacno a=0 */
 a = (c >= (-4));    /* Tacno, a=1 */
 a = (b <= 3);       /* Netacno a=0 */
 a = (b == c);       /* Netacno a=0 */
 a = (c != 5);       /* Tacno, a=1 */

 return 0;
}
```

Логички оператори

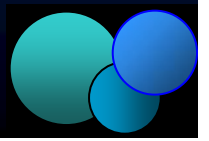
!	негација
&&	логичко И
||	логичко ИЛИ



Логичка негација

- 0 је логичка неистина
- Било који број различит од нуле је логичка истина
- Негација логичке неистине је логичка истина (1)
- Негација логичке истине је логичка неистина (0)

```
int main()
{
 int a, b = 7, c = 0;

 a = !b; /* a = 0 */
 a = !c; /* a = 1 */

 return 0;
}
```

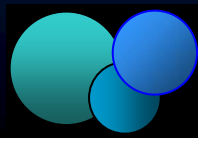
Логички оператори, пример

- да ли се вредност промењљиве налази између 10 и 100

```
b = ( (a>10) && (a<100) )
/* Ako je a>10 i a<100 vrednost izraza
   je 1, u suprotnom 0*/
```

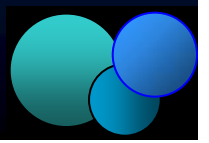
- да ли је апс. вредност промењљиве већа од 15

```
b = ( (a < -15) || (a > 15) )
/* ako je a < -15 ili a > 15
   vrednost izraza je 1,
   u suprotnom 0  */
```

Оператори над битовима

~	потпуни комплемент на битском нивоу
<<	померање у лево на битском нивоу
>>	померање у десно на битском нивоу
&	логичко И на битском нивоу
|	логичко ИЛИ на битском нивоу
^	логичко ексклузивно или (*exor*)



Оператори над битовима, пример

```
0x1234 & 0x5678   (4660₁₀ & 22136₁₀)
   0001 0010 0011 0100     0x1234
 & 0101 0110 0111 1000     0x5678
 = 0001 0010 0011 0000=0x1230(4656₁₀)
```

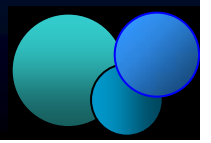
```
0x1234 | 0x5678
   0001 0010 0011 0100     0x1234
| 0101 0110 0111 1000     0x5678
= 0101 0110 0111 1100 =0x567c(22140₁₀)
```

Оператори над битовима, пример

```
0x1234 ^ 0x5678
   0001 0010 0011 0100
^  0101 0110 0111 1000
=  0100 0100 0100 1100 = 0x444c (17484_10)
```

```
~0x1234
 ~0001 0010 0011 0100
=  1110 1101 1100 1011 = 0xedcb (60875_10)
```

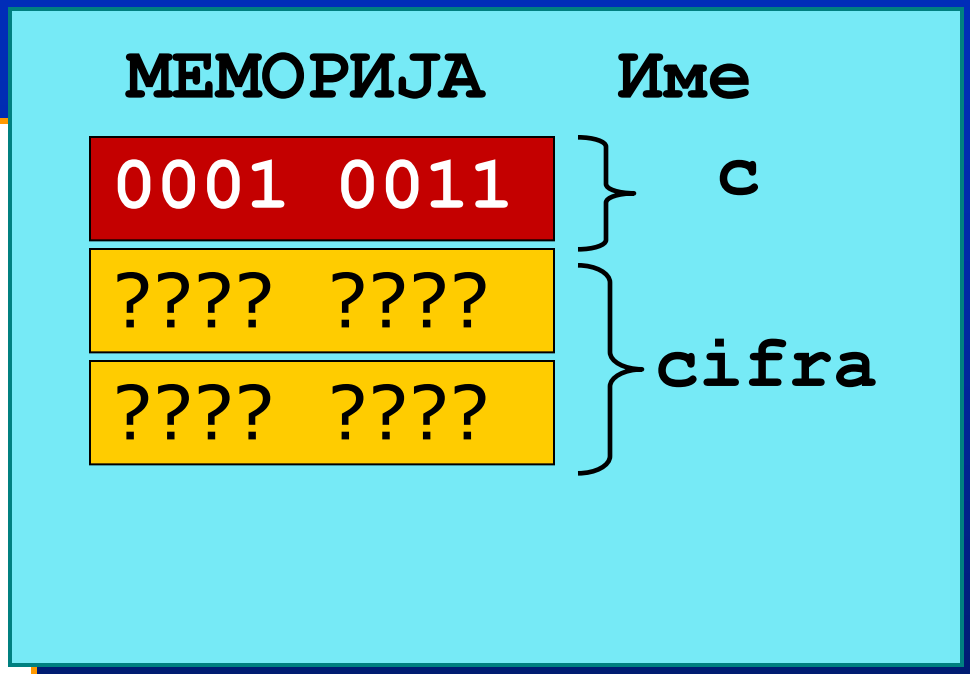
```
0x0001 << 5
0000 0000 0000 0001 << 5 =
0000 0000 0010 0000      = 0x0020 (32_10)
```

Оператори над битовима, пример

& - битско AND

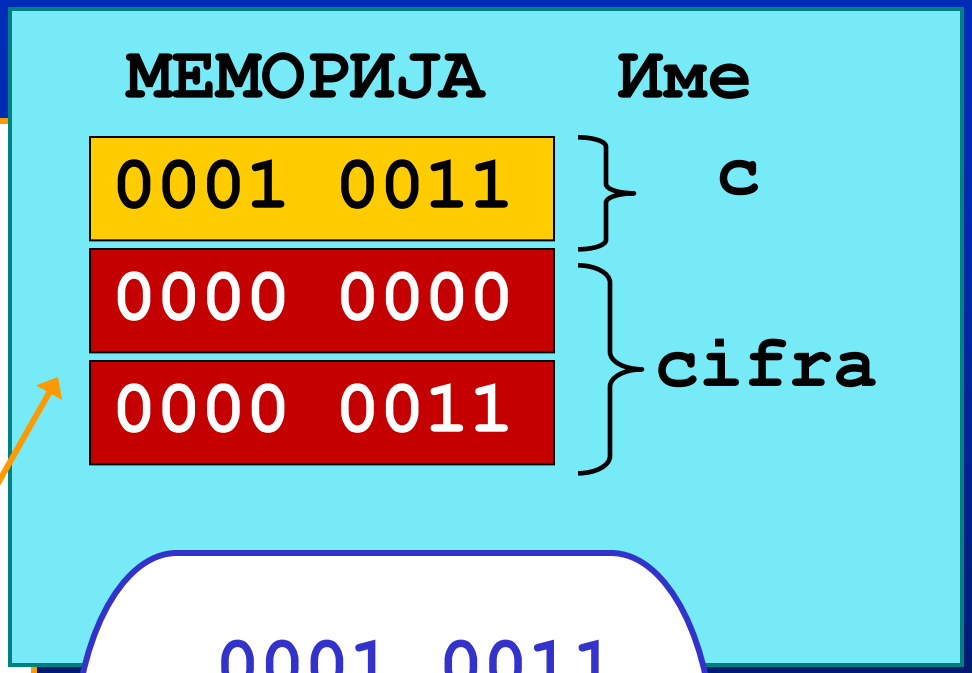
```
#include <stdio.h>
int main()
{
 char c = 19;
 short int cifra;
 cifra = c & 0x0F;
 printf("%d\n",cifra);

 return 0;
}
```



Оператори над битовима, пример

& - битско AND

```
#include <stdio.h>
int main()
{
 char c = 19;
 short int cifra;
 cifra = c & 0x0F;
 printf("%d\n",cifra);

 return 0:
}
```



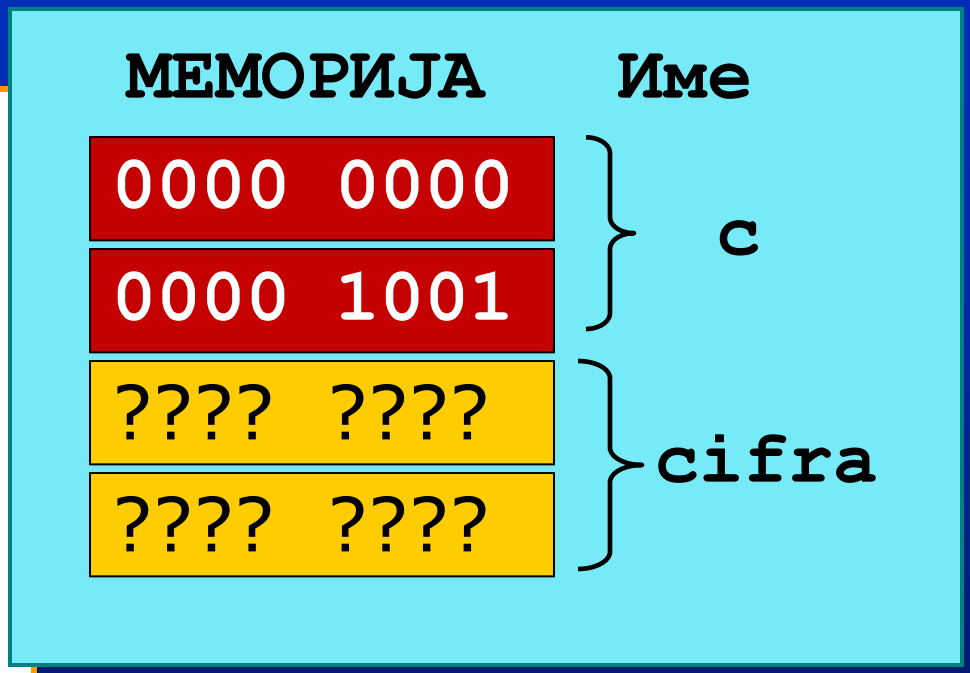

```
  0001 0011
& 0000 1111
  0000 0011
```

3 (0x03)

Оператори над битовима, пример

| - битско OR

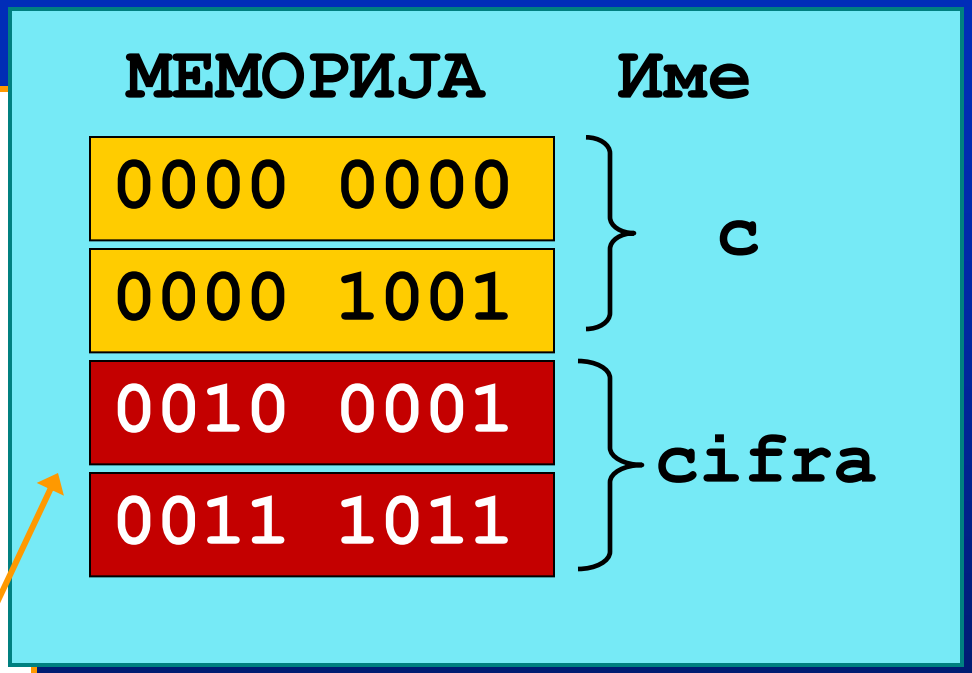
```
#include <stdio.h>
int main()
{
 short int c = 9;
 short int cifra;
 cifra = c | 0x213a;

 printf("%d\n",cifra);

 return 0;
}
```



Оператори над битовима, пример

| - битско OR

```
#include <stdio.h>

int main()

{

 short int c = 9;

 short int cifra;

 cifra = c | 0x213a;

 printf("%d\n",cifra);


 return 0;

}
```




```
  0000 0000 0000 1001
| 0010 0001 0011 1010
-------------------
  0010 0001 0011 1011
```


8507 (0x213B)

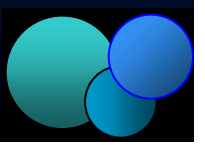
Оператори над битовима, пример

^ - битско EXOR

```
#include <stdio.h>
int main()
{
 short int x=0x73, y;
 y = x ^ 0x32;

 printf("%d", y):

 return 0;
}
```

```
  0000 0000 0111 0011
^ 0000 0000 0011 0010
  0000 0000 0100 0001
```

65 (0x41)



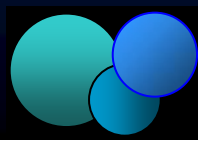
Померање у лево

```
#include <stdio.h>
int main()
{
 char x = 3, y;

 y = x << 1;

 printf( "%d\n", y );

 return 0;
}
```

x = 0000 0011

y = 0000 0110

6 (0x06)



Померање у лево

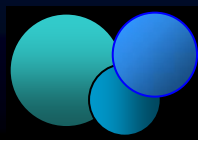
```
#include <stdio.h>
int main()
{
 char x = 3, y;

 y = x << 4;

 printf( "%d\n", y );

 return 0;
}
```

x = 0000 0011

y = 0011 0000

48 (0x30)

```
y = x << 5;   /*  y = 96 */
y = x << 6;   /*  y = -64 */
```

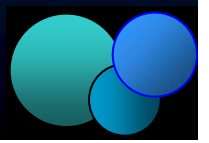



Померање у десно

```
#include <stdio.h>
int main()
{
   short int x = 12, y;

   y = x >> 1;
   printf( "%d\n", y );

   return 0;
}
```

6 (0x00 06)

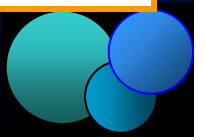
```
x = 0000 0000 0000 1100
y = 0000 0000 0000 0110
```

Условни израз

```
uslov ? izraz1 : izraz2
```

```
c = (a > b) ? 5 : 12;
/* Ako je a > b onda c = 5;
    inace c = 12; */

c =(x<=y && y>=z) ? y : (x+z)/2
```

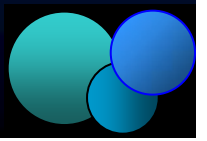
Пример

```
#include <stdio.h>
int main()
{
   int x , y, z;
   printf( "Unesite dva cela broja: ");
   scanf("%d %d",&x, &y);

   z=(x > y) ? x-y : y-x;

   printf("Vrednost apsolutne razlike je:");
   printf(" %d\n", z);

   return 0;
}
```

Конверзија типа

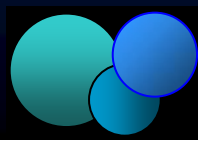
- Бинарни оператори захтевају да два оператора имају исти тип.
- Када то није испуњено долази до аутоматске конверије типова.

```
5 + 6.0    /* isto sto i 5.0+6.0 */
5/4 + 3    /* rezultat je 1+3=4 */
3.0 * 5/4  /* (3.0*5)/4=(3.0*5.0)/4
              15.0/4 =15.0/4.0=3.75  */
```

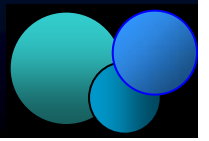
Редослед извршавања оператора

- Редослед извршавања оператора је унапред дефинисан одговарајућим приоритетом.
- Заграде ( ) имају највиши приоритет.



Област важења идентификатора

- Досег или област важења идентификатора (*scope*)
  - означава део програма у ком идентификатор може да се користи, тј. део програма у ком је идентификатор „видљив”.
- област важења идентификатора је ***од места где је дефинисан до краја блока***!
- Оваква област важења се назива **блоковска област важења.**

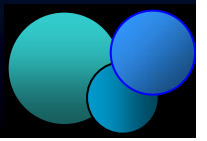


Досег

```
int main()
{
  int prom1 = 1;
  double prom2 = 3.22;
  double prom3 =  prom2 + 5.92;

  prom1 += 32;
  prom3 = (double)prom1 * 54.76;
  printf("%d %lf\n", prom1, prom3);

 return 0;
}
```

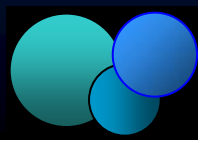
Досег

```
int main( )
{
  int prom1 = 1;
  double prom3 = prom2 + 5.92; /* GRESKA */
  /*error C2065: 'prom2': undeclared
                                identifier*/
  double prom2 = 3.22;
  ...
   return 0;
}
```

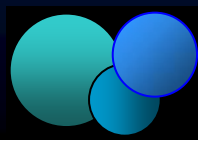
Наредбе

- Сваки програм написан на неком програмском језику је састављен од низа појединачних наредби.
- Редослед извршавања наредби је секвенцијалан
- Измена редоследа може постићи помоћу **управљачких наредби**.



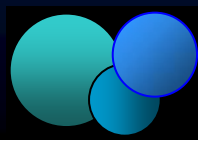
Управљачке наредбе

- У управљачке наредбе спадају:
  - наредбе гранања
    `if-else` и **`switch`** наредбе.
  - наредбе понављања (петље, циклуси или итерације) у које спадају
    `for`, **`while`** и **`do-while`** наредбе.



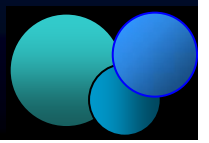
Наредбе гранања

- Наредба **`if else`**
- Наредба **`switch`**



Наредба if else

```
if (izraz1)
{
  naredbe
}
else if (izraz2)
{
  naredbe
}
...
else
{
  naredbe
}
```

- Програм за дељење два броја

```
#include <stdio.h>
int main()
{
   float x , y, z;
   printf( "Unesite dva realna broja: ");
   scanf("%f %f",&x, &y);

   z= x/y;

   printf("Kolicnik je %.2f", z);

   return 0;
}
```

- Ако је y=0?

```
#include <stdio.h>
int main()
{
   float x , y, z;
   printf( "Unesite dva realna broja: ");
   scanf("%f %f",&x, &y);
   if(y != 0 )
   {
      z= x/y;
      printf("Kolicnik je %.2f", z);
   }
   return 0;
}
```

Порука кориснику?

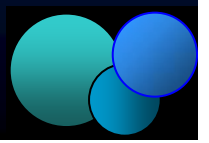
```c
#include <stdio.h>
int main()
{
  float x , y, z;
  printf( "Unesite dva realna broja: ");
  scanf("%f %f",&x, &y);
  if(y !=0)
  {
     z= x/y;
     printf("Kolicnik je %.2f", z);
  }
  else
  {
    printf("Delilac ne moze biti 0!");
  }
  return 0;
}
```

Наредба if else

```
if(a > 5) b = 3;
```

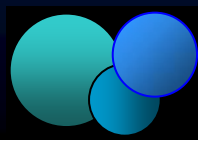
```
if(a > 5)
   b = 3;
```

```
if(b >= 3)      /* НЕМА ; */
{
  b = 1;
  count = 45.32;
  slovo = 'A';
}
```

Наредба `if else`

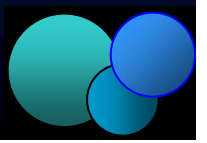
- `if(broj != 0)` је исто што **и** `if(broj)`
- `if(broj == 0)` је исто што и `if(!broj)`



Наредба if else

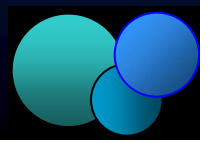
```
if(broj_a == 0)
  printf("broj a je jednak nuli\n");
else
  printf("broj a je razlicit od nule\n");
```

```
if(a > 5)
{
   b = a + 6;
   c = 6;
}
else
{
   b = a * 2;
   d = 136;
}
```

```
if(a > 5)
{
    b = a + 6;

   if(b > 15)
   {
     pom = 0;
     b = b*2;
   }
}
else
{
    b = a * 2;
    d = 136;
}
```

```
if(a > 5)
{
    b = a + 6;

    if(b > 15)
      b = b*2;
}
else
{
    b = a * 2;
    d = 136;
}
```

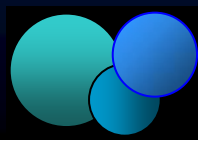
```
if(a > 5)
{
   b = a + 6;
    if(b > 15)
    {
      pom = 0;
      b = b*2;
    }
    else
    {
      b = 5;
    }
}
else
{
   b = a * 2;
   d = 136;
}
```

```
if(a > 5)
{
 b = a + 6;
if(b > 15)
{
   pom = 0;
   b = b*2;
}

}
else
{
   b = a * 2;
   d = 136;
}
```

```
if(a > 5)
{
   b = a + 6;
if(b > 15)
   pom = 0;

   b = b*2;


}
else
{
    b = a * 2;
    d = 136;
}
```

Наредба if elseif else

```
if(izraz1)
{
  naredbe
}
else if(izraz2)
{
  naredbe
}
...
else
{
  naredbe
}
```

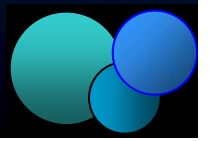



```c
#include <stdio.h>
int main()
{
   int x;
   printf( "Unesite ceo broj od 101 do 200: ");
   scanf("%d",&x);
   if(x < 101)
     printf("Uneti broj  je manji od 101\n");
   else if (x >100 && x <=200)
   {
     x = x-100;
     printf("Novi broj je %d\n", x);
   }
   else
     printf("Uneti broj  je veci od 200\n");

   return 0;
}
```

Наредба if-elseif-else

```
if(a > 90)
  ocena = 10;
else if(a > 80)
  ocena = 9;
else if(a > 70)
  ocena = 8;
else if(a > 60)
  ocena = 7;
else if(a > 50)
  ocena = 6;
else
  ocena = 5;
```

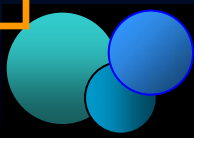
```
ocena=5;
if(a>90)
  ocena = 10;
else if(a>80)
  ocena = 9;
else if(a>70)
  ocena =8;
else if(a>60)
  ocena =7;
else if(a>50)
  ocena =6;
```

Наредба switch

- Наредба switch представља уопштење наредбе **if** . Општи облик наредбе је дат са:

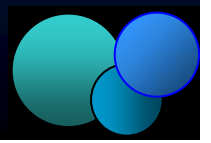
```
switch(izraz)
{
  case vrednost_1: niz_naredbi_1
  case vrednost_2: niz_naredbi_2
  ...
  case vrednost_N: niz_naredbi_N
  [default:  niz_naredbi]
}
```

Наредба `switch`

- Прво се израчуна вредност **`izraz`.**
- Израчуната вредност се пореди са vrednost_1, vrednost_2,... vrednost_N.

- Уколико је иста вредност пронађена биће **извршене СВЕ наредбе** које припадају тој **`case`** наредби.



Наредба switch

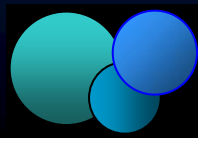
```
#include<stdio.h>
int main()
{
   int a = 1;
   int b = 2;
   switch(a+b)
   {
    case 12:
        printf("Zbir je 12\n");
        break;
    case 3:
        printf("Zbir je 3\n");
        break;
    case 41:
        printf("Zbir je 41\n");
        break;
    default:
        printf("Zbir je nepoznat\n");
        break;
   }
}
```

```
Zbir je 3
```

Анализа примера

```
#include  <stdio.h>

  int main( )
  {
     int a = 3, b = 3;

     printf("\n a     b \n");
     printf("\n %d    %d", a--, --b);
     printf("\n %d    %d", a--, --b);
     printf("\n %d    %d \n\n", a--, --b);

     return 0;
  }
```

```
 a  b
 3  2
 2  1
 1  0
```

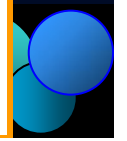
Анализа примера

```
#include  <stdio.h>
int main( )
{
  int x, a = 5,  b = 6,  c = 5,  d = 1;

  x = a<b  ||  a<c  &&  c<d;
  printf("\nx = a<b || a<c && c<d");
  /* && ima   veci prioritet od || */
  printf("\nBez zagrada rezultat=%d\n",x);

  x = (a<b || a<c) && c<d;
  printf("\nx = (a<b || a<c) && c<d");
  printf("\nSa zagradama rezultat=%d\n",x);

  return 0;
}
```

```
x = a<b || a<c && c<d
Bez zagrada rezultat=1
x = (a<b || a<c) && c<d
Sa zagradama rezultat=0
```

Анализа примера

```
#include<stdio.h>
int main( )
{
 unsigned short broj, rb;

 printf("\nUnesite heksa broj: ");
 scanf("%x",&broj);
 printf("\nUnesite redni broj bita: ");
 scanf("%d",&rb);

 if(broj & (1 << (rb - 1)))
   printf("\n%d. bit ima vrednost 1\n",rb);
 else
   printf("\n%d. bit ima vrednost 0\n",rb);

  broj ^= (1 << (rb - 1));
  printf("\nNova vrednost je: %#x\n\n",broj);
  return 0;
}
```



бит на *n* тој позицији је 1 или 0?

КАКО ОДРЕДИТИ ВРЕДНОСТ БИТА НА *n*-тој ПОЗИЦИЈИ?

broj = 0x1234 (4660_{10})

(0001 0010 0011 0100)$_2$

rb = 5_{10}

if(broj & (1 << (rb-1)))

(rb-1) je 4

1 << 4

0000 0000 0000 0001 << 4

0000 0000 0001 0000

broj & (1 << 4)

```
   0001 0010 0011 0100
 & 0000 0000 0001 0000
 ---------------------
   0000 0000 0001 0000
```

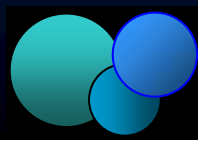
Инверзија бита са *n* те позиције

КАКО ИНВЕРТОВАТИ ВРЕДНОСТ БИТА?

```
broj ^= (1 << (rb-1));
```

```
broj = broj ^ (1 << (rb-1));
broj = broj ^ (1 << 4);
broj = broj ^ (0000 0000 0001 0000);
```

```
  0001 0010 0011 0100
^ 0000 0000 0001 0000
  0001 0010 0010 0100
```

Анализа примера

```
#include<stdio.h>
int main( )
{
   int a, b, max;

   printf("\nUnesite dva cela broja: ");
   scanf("%d%d", &a, &b);

   max = a;
   if( b > max)
       max = b;

   printf("\nVeci je: %d\n", max);

   return 0;
}
```

Анализа примера

```
#include<stdio.h>
int main( )
{
  int a, b;

  printf("\nUnesite dva broja: ");
  scanf("%d%d", &a, &b);

  if(a == b)
     printf("\na je jednako b\n");
  else  if(a > b)
     printf("\na je vece  od b\n");
  else
     printf("\na je manje  od b\n");

  return 0;
}
```

Анализа примера

```
#include  <stdio.h>
int main( )
{
  int a, b;
  printf("\nUnesite dva broja, a i b);
  scanf("%d%d", &a, &b);

  if(a == b)
    printf("\n a je jednako b \n");
  else
    printf("\n%d je veci broj\n",(a > b) ? a : b);

  return 0;
}
```

Хвала на пажњи

Питања?