



Универзитет “Св. Кирил и Методиј” во Скопје  
Факултет за електротехника и информациски технологии



## ПРОГРАМИРАЊЕ И АЛГОРИТМИ

# АЛГОРИТМИ И ПОДАТОЧНИ ТИПОВИ

- Програмски јазик C -



Програмирање и алгоритми

# РАЗВОЈ НА ПРОГРАМА И АЛГОРИТАМ



# РАЗВОЈ НА ПРОГРАМА





Проблем кој треба да се реши

1. Идеја за постапка која го решава проблемот
2. Дизајн на алгоритамот
3. Пишување на изворниот код
4. Преведување на изворниот код
5. Анализа на грешките при преведувањето (опција)
6. Извршување на програмата и тестирање
7. Анализа на грешките при извршувањето (опција)
8. Почни одново (задолжително)

# БЛОК ДИЈАГРАМИ

- (Речник) Шематско претставување на низа од операции, како на пример во процес на производство или компјутерска програма
- (Технички) Графичко претставување на низа од операции во еден информациски систем или **програма**. Блок дијаграмите за **компјутерски програми** ја прикажуваат низата наредби во една програма или потпрограма. За секој тип блок дијаграми се користат различни симболи.
- Најчесто концептот за **компјутерскиот алгоритам** се реализира со помош на блок дијаграм.

# СИМБОЛИ ЗА БЛОК ДИЈАГРАМ

СИМБОЛ	ИМЕ	ОПИС
	TERMINAL	Почетна и крајна точка на блок дијаграм
	INITIALIZATION	Декларација и иницијализација на мемориски простор за податоци
	INPUT/OUTPUT	Влез на податоци за обработка и печатење на обработените податоци
	PROCESS	Обработка на податоци (доделување; математички пресметки)

# СИМБОЛИ ЗА БЛОК ДИЈАГРАМ

СИМБОЛ	ИМЕ	ОПИС
	FLOW LINES (ЛИНИИ НА ТЕК)	Дефинира логички редослед на програмата
	ON-PAGE CONNECTOR (КОНЕКТОР – во страница)	Врска кон дел од програмата на истата страница. Се избегнува преплетување на линиите.
	OFF-PAGE CONNECTOR (КОНЕКТОР – надвор од страница)	Врска кон дел од програмата на друга страница. Да се избегне преплетување на линиите.
	DECISION (ОДЛУКА)	Обработка на услови со користење релациони и логички оператори. Пронаоѓање и филтрирање податоци.



# ДЕФИНИЦИЈА НА ПРОБЛЕМ

Проблем: Да се пресмета средната вредност на броевите 5, 7, 11 и 13.

1. Постапка изразена со чекори:
2. Собери ги броевите 5 и 7
3. На нивната сума додај го бројот 11
4. На нивната сума додај го бројот 13
5. Сумата подели ја со 4
6. Добиената вредност прикажи ја на екранот



# ДЕФИНИЦИЈА НА ПРОБЛЕМ

Проблем: Да се пресмета средната вредност на броевите 5, 7, 11 и 13.

1. Постапка изразена со чекори:
2. На бројот 5 додај го бројот 7
3. На нивната сума додај го бројот 11
4. На нивната сума додај го бројот 13
5. Сумата подели ја со 4
6. Добиената вредност прикажи ја на екранот



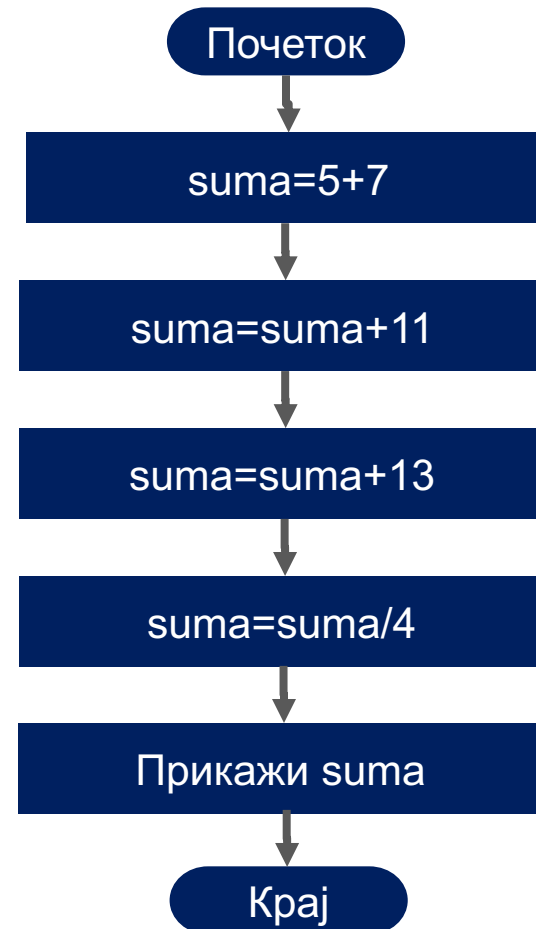
# АЛГОРИТАМ



## Постапка изразена со чекори:

1. На бројот 5 додај го 7
2. На нивната сума додај го бројот 11
3. На нивната сума додај го бројот 13
4. Сумата подели ја со 4
5. Прикажи ја вредноста на екранот

## Блок дијаграм:





# АЛГОРИТАМ



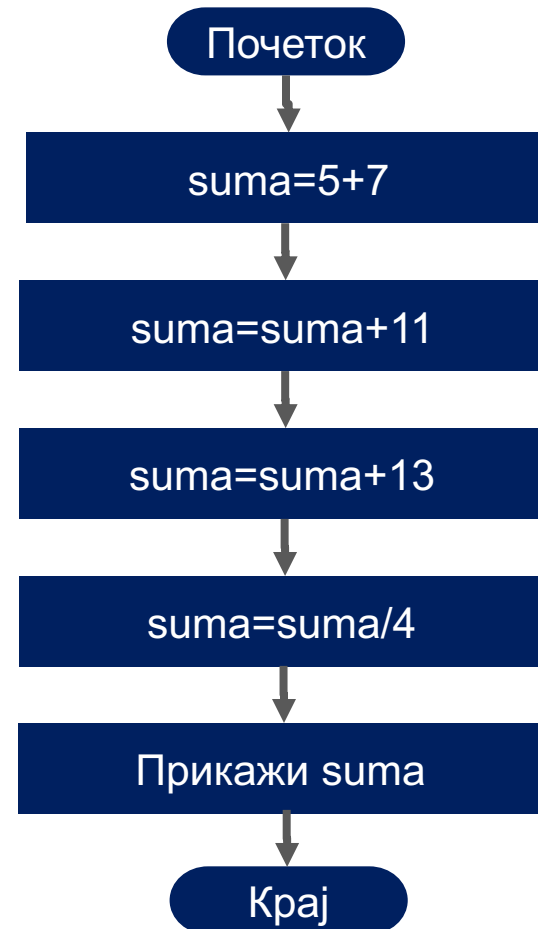
## Програма во програмски јазик C

```
/* A first program in C */
#include <stdio.h>
int main()
{
    int suma;
    suma = 5+7;
    suma = suma + 11;
    suma = suma + 13;
    suma = suma / 4;
    printf("Srednata vrednost e %d", suma);
    return 0;
}
```

Излез:

Srednata vrednost e 9

## Блок дијаграм:





## Програма во програмски јазик C

```
/* A first program in C */
#include <stdio.h>
int main()
{
    int suma;
    suma = 5+7;
    suma = suma + 11;
    suma = suma + 13;
    suma = suma / 4;
    printf("Srednata vrednost e %d", suma);
    return 0;
}
```

Излез:

Srednata vrednost e 9

### ■ Коментари

- текст ограничен со /\* и \*/ што компјутерот го игнорира
- се користи да ја опише програмата

### ■ #include <stdio.h>

- претпроцесорска директива – му кажува на компјутерот да ја вметне содржината на определена датотека
- <stdio.h> во датотеката се сместени програмските кодови на влезно-излезните операции
- директивата треба да се појави сама и да биде напишана во една линија
- директивите не завршуваат со ;



Програмирање и алгоритми

# СТРУКТУРА НА ПРОГРАМА ВО C



# СТРУКТУРА НА ПРОГРАМА ВО С

`int main()`

- секоја програма содржи една или повеќе функции, но точно една што се нарекува **main**
- оваа функција се повикува прва при извршување на С програма
- заградите означуваат дека станува збор за функција
  - оваа функција за да се изврши нема потреба од информации
- `int` означува дека функцијата „враќа“ целобројна вредност
- големите загради го означуваат телото на функцијата
- секој израз завршува со ;
- не постојат правила за уредување на програмата



# СТРУКТУРА НА ПРОГРАМА ВО C

`printf("Srednata vrednost e %d", suma/4);`

- дава инструкции компјутерот да изврши акција, да ја прикаже на компјутерскиот екран реченицата ограничена со наводниците
- целата линија се нарекува израз
  - сите изрази во C мора да завршат со ;
- `\` - специјален знак (escape character)
  - означува дека `printf()` треба да направи нешто различно од вообичаеното однесување
  - `\n` е ознака за знакот нов ред

`return 0;`

- по конвенција `return 0` означува дека програмата завршува успешно со својата работа

# СТРУКТУРА НА ПРОГРАМА ВО С

## Програма во програмски јазик С

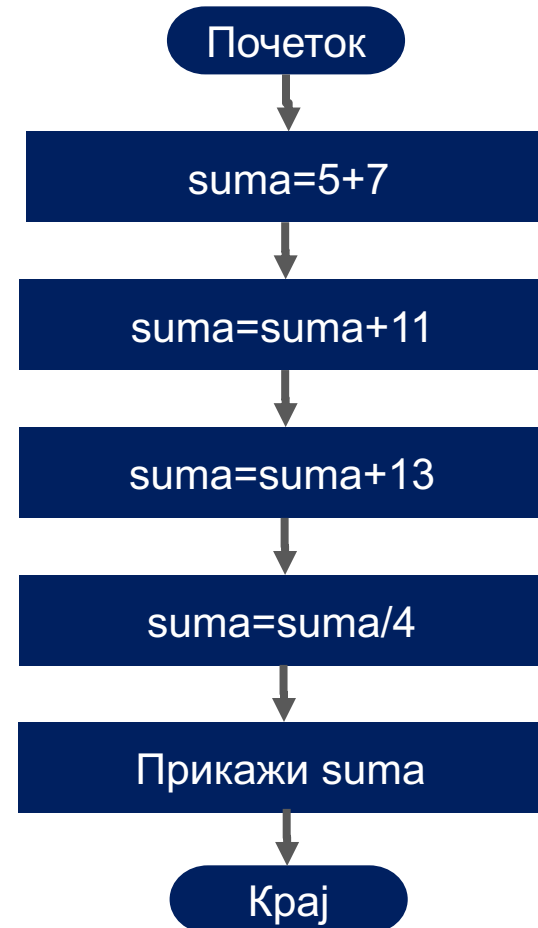
```
/* A first program in C */
#include <stdio.h>
int main()
{
    int suma;
    suma = 5+7;
    suma = suma + 11;
    suma = suma + 13;
    suma = suma / 4;
    printf("Srednata vrednost e %d", suma);
    return 0;
}
```

Податоци

Излез:

Srednata vrednost e 9

## Блок дијаграм:





Програмирање и алгоритми

# ПОДАТОЧНИ ТИПОВИ

# ПОДАТОЧНИ ТИПОВИ

## ■ Податочниот тип определува

- како вредностите за даден податок се сместуваат во меморијата и колку меморија зафаќаат (зависно од платформата),
- множеството вредности за податокот, и
- операциите што може да се извршат со или над неговите вредности.

## ■ Поделба

- Стандардни: `int`, `char`, `float`, ...
- Изведени (корисничко дефинирани, композитни):  
набројувачки, полиња, записи итн.

## ■ Операции:

- Аритметички, логички и релациски



# ПОДАТОЧНИ ТИПОВИ

32-bit процесор

■ `int` – 4B

$-2147483648 \div 2147483647$

■ `long` – 8B

■ `short` – 2B

$-32768 \div 32767$

■ `unsigned int` – 2B



# ПОДАТОЧНИ ТИПОВИ

## ■ Типови за реални вредности

□ `float` – 4B (32 bit)

6 децимали

□ `double` – 8B (64 bit)

15 децимали



# ПОДАТОЧНИ ТИПОВИ

## ■ Типови за знакови вредности

- елементи на ASCII кодната шема
- во програмите се користат со 'a', '!'
- `char` – 1B (8 bit)      -128 ÷ 127

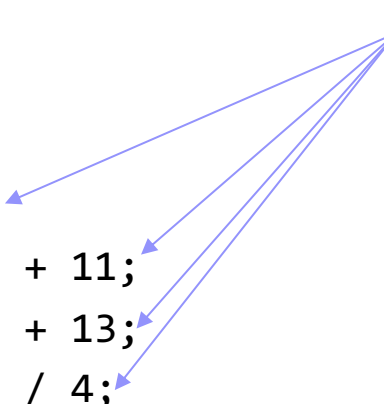


# СТРУКТУРА НА ПРОГРАМА ВО C

## Програма во програмски јазик C

```
/* A first program in C */
#include <stdio.h>
int main()
{
    int suma;
    suma = 5+7;
    suma = suma + 11;
    suma = suma + 13;
    suma = suma / 4;
    printf("Srednata vrednost e %d", suma);
    return 0;
}
```

Константи



Излез:

Srednata vrednost e 9



# КОНСТАНТИ

- Означуваат податоци што во текот на извршување на програмата не ја менуваат вредноста
- За секоја константа во програмскиот јазик C се одредува податочниот тип
- Нумерички константи
- Знакови константи

# КОНСТАНТИ

## ■ Нумерички константи

### □ Целобројни константи

### □ Декадни константи - 1, 0, 2, 123

- дозволени цифри, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

### □ октални константи – 012

- дозволени цифри, 0, 1, 2, 3, 4, 5, 6, 7

### □ хексадецимални константи – 0x23

- дозволени цифри, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

### □ константи од типот long 890L



# КОНСТАНТИ

## ■ Нумерички константи

- Реални константи
- Декадна и експоненцијална нотација
- Броевите што содржат (.) или (e) се третираат како константи од видот **double**
- да се дефинира **float** константа потребно е да се додаде F на крајот од бројот
- 3.5F, 1e-7F
- **ВНИМАНИЕ:** 12.0L е **long double**, но 12L е **long int**



# КОНСТАНТИ

## ■ Знакови константи

- 'a', '\n', '%'

## ■ Текстуални низи

- "ana", "", "\n", "a"



# СТРУКТУРА НА ПРОГРАМА ВО C

## Програма во програмски јазик C

```
/* A first program in C */
```

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

```
int main()
```

```
{
```

```
    int suma;
```

```
    suma = 5+7;
```

```
    suma = suma + 11;
```

```
    suma = suma + 13;
```

```
    suma = suma / 4;
```

```
    printf("Srednata vrednost e %d", suma);
```

```
    return 0;
```

```
}
```

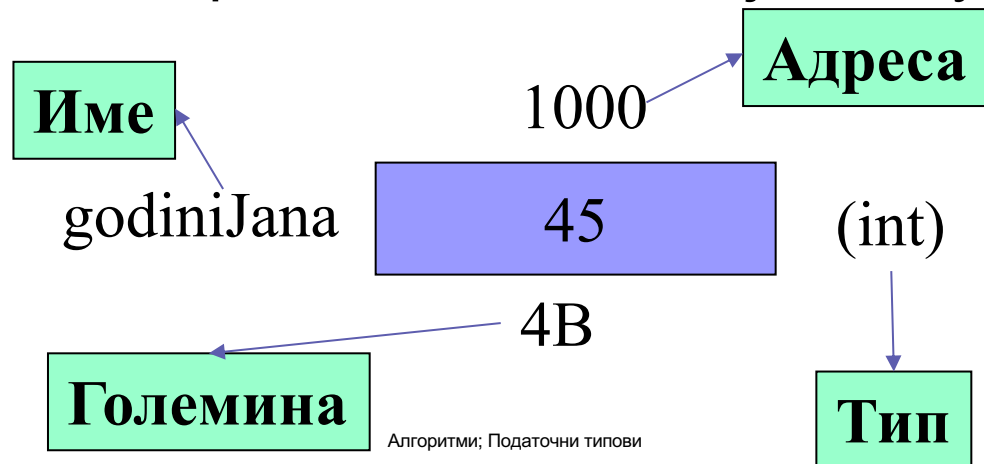
Променлива

Излез:

Srednata vrednost e 9

# ПРОМЕНЛИВИ

- Имињата на променливите соодветствуваат на локација во компјутерската меморија
- Секоја променлива има **име**, **тип**, **големина** (колку меморија зафаќа) и **вредност**
- Секогаш кога нова вредност ќе биде сместена во контејнерот (променливата), претходната вредност се брише
- Читањето вредност од променливата не ја менува истата





# ПРОМЕНЛИВИ

## ■ Декларирање

- го кажува на компјутерот името и типот на променливата

## ■ Формат

- `tipPromenliva ImePromenliva;`
- `tipPromenliva Ime1, Ime2, Ime3;`
- `tipPromenliva Ime=Vrednost;`
- `tipPromenliva Ime1=vrednost1, Ime2=vrednost2;`

## ■ Правила:

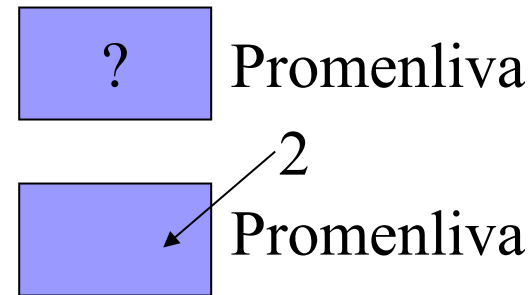
- секогаш се прави на почетокот на функцијата или
- на почеток на кој и да е блок од изрази или
- надвор од функцијата
- и тоа секогаш пред да биде употребена



# ЗАДАВАЊЕ ВРЕДНОСТ НА ПРОМЕНЛИВА

```
int Promenliva;  
Promenliva = 2;
```

- формат  
Promenliva = vrednost;





# ПОДАТОЧНИ ВИДОВИ - ПРИМЕР

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

```
int main() {
```

```
    int suma;
```

```
    float pari;
```

```
    char bukva;
```

```
    double pi;
```

```
    suma = 10;
```

```
    pari = 2.21;
```

```
    bukva= 'A';
```

```
    pi = 2.01E6;
```

```
    printf("Vrednost na suma = %d\n", suma );
```

```
    printf("Vrednost na pari = %f\n", pari );
```

```
    printf("Vrednost na bukva = %c\n", bukva );
```

```
    printf("Vrednost na pi = %e\n", pi );
```

```
    return 0;
```

```
}
```

**Излез:**

Vrednost na suma = 10

Vrednost na pari = 2.210000

Vrednost na bukva = A

Vrednost na pi = 2.010000e+06

/\* pridruzi celobrojna vrednost \*/

/\* pridruzi realna float vrednost \*/

/\* pridruzi znak \*/

/\* pridruzi realna double vrednost\*/

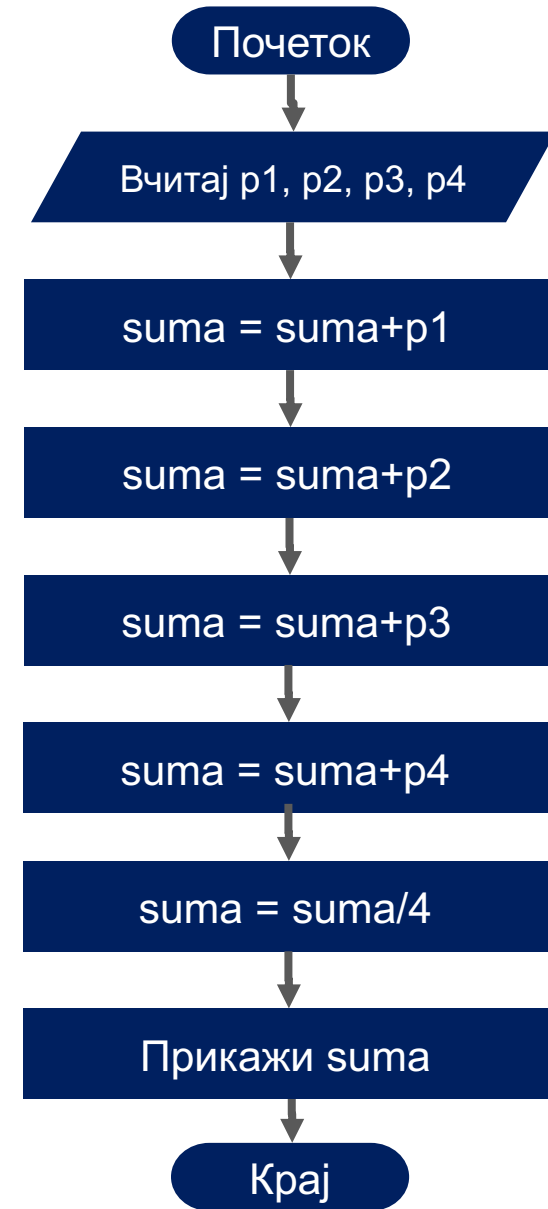


# ПРОГРАМА ВО C



```
/* A first program in C */  
#include <stdio.h>
```

```
int main()  
{  
    int p1, p2, p3, p4, suma=0;  
    scanf("%d%d%d%d",&p1, &p2, &p3, &p4);  
    suma = suma + p1;  
    suma = suma + p2;  
    suma = suma + p3;  
    suma = suma + p4;  
    suma = suma / 4;  
    printf("Srednata vrednost e %d", suma);  
    return 0;  
}
```





Програмирање и алгоритми

# АРИТМЕТИЧКИ ОПЕРАЦИИ

# АРИТМЕТИЧКИ ОПЕРАЦИИ

## ■ Аритметички оператори

## ■ Изрази

## ■ Правила

- ☐ int op int
- ☐ float op float
- ☐ int op float
- ☐ char op char

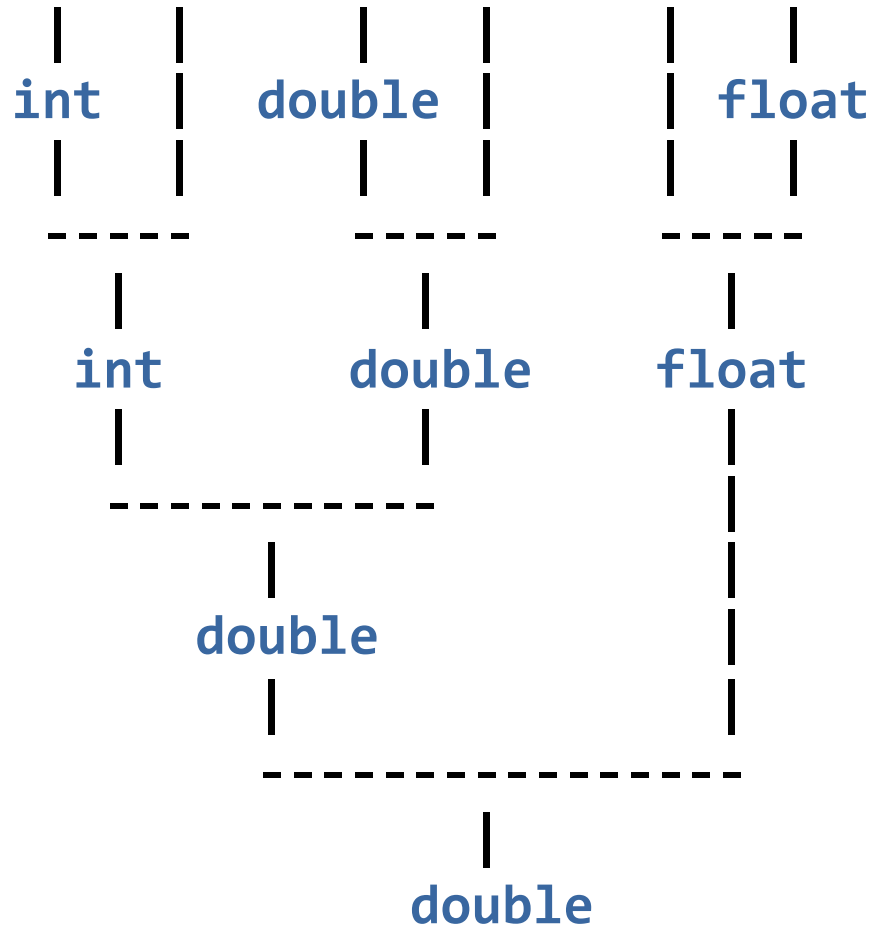
С операции	Оператор	Аритметички Израз	Израз во С
собирање	+	$f + 7$	<code>f + 7</code>
одземање	-	$p - c$	<code>p - c</code>
множење	*	$bm$	<code>b * m</code>
делење	/	$x / y$	<code>x / y</code>
модул	%	$r \text{ mod } s$	<code>r % s</code>

Оператор(и)	Операции	Приоритет
( )	загради	Се пресметува на почеток. Ако заградите се вгнездени, изразот во највнатрешните загради се пресметува прв. Ако постојат повеќе загради на исто ниво во изразот тие се пресметуваат одлево надесно.
*, /, или %	множење, делење, модул	Се пресметува втора. Ако во изразот има повеќе операции од овој вид тие се пресметуваат одлево надесно.
+ или -	собирање одземање	Се пресметува последен. Ако во изразот има повеќе операции од овој вид тие се пресметуваат одлево надесно.



# АРИТМЕТИЧКИ ОПЕРАЦИИ

```
char ch; int i; float f; double d, rezultat;  
rezultat = (ch / i) + (f * d) - (f + i);
```





# ОПЕРАЦИИ СО ЦЕЛИ И РЕАЛНИ БРОЕВИ

```
int main() {  
    int celo;                /* cel broj */  
    float realno;            /* realna vrednost */  
    realno = 1.0 / 2.0;      /* postavi (float)0.5*/  
    celo = 1 / 3;            /* postavi (int)0 */  
    realno = (1/2) + (1/2);  /* postavi (float)0.0 */  
    realno = 3.0 / 2.0;      /* postavi (float)1.5 */  
    celo = realno;           /* postavi (int)1 */  
    celo = celo + 1;  
    return 0;  
}
```

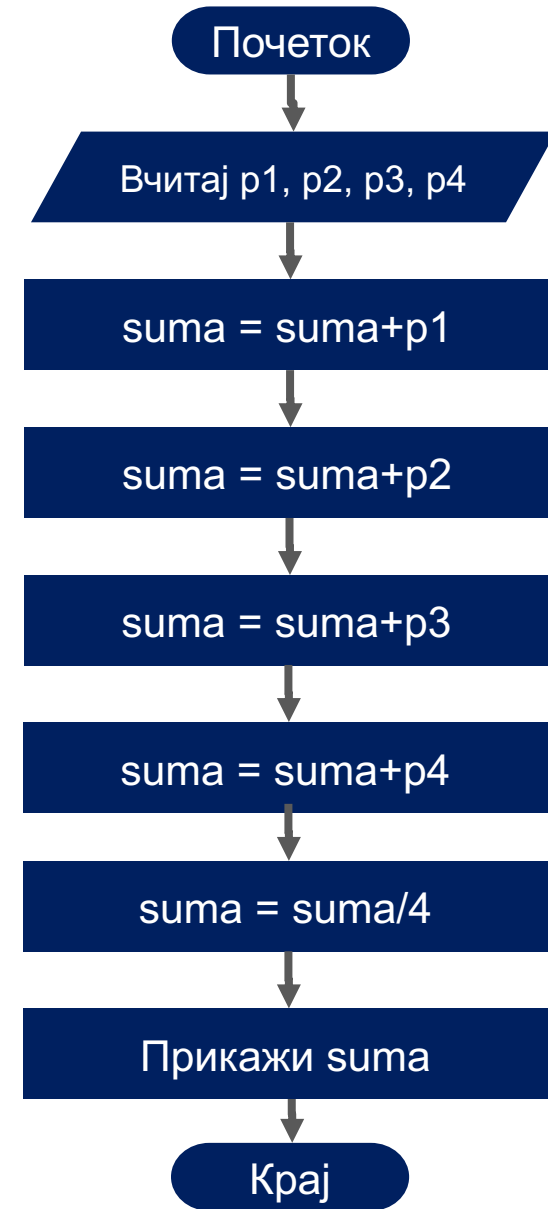


# ПРОГРАМА ВО C



```
/* A first program in C */  
#include <stdio.h>
```

```
int main()  
{  
    int p1, p2, p3, p4;  
    float suma=0;  
    scanf("%d%d%d%d",&p1, &p2, &p3, &p4);  
    suma = suma + p1;  
    suma = suma + p2;  
    suma = suma + p3;  
    suma = suma + p4;  
    suma = suma / 4;  
    printf("Srednata vrednost e %f", suma);  
    return 0;  
}
```





# ПРОМЕНА НА ТИП НА ВРЕДНОСТ

## - cast операција -

### ■ формат

□ (podatocen tip) vrednost

### ■ Пример:

```
int i;  
double d = 6.28;  
i = (int) d;
```

### ■ Пример:

(int) 3.56	/* 3.56 во 3 */
(long) 6.28	/* 6.28 во 6L */
(double) 2	/* 2 во 2.0 */
(char) 70	/* 70 во char код е ('F') */
(unsigned short) 3.14	/* 3.14 во 3 (unsigned short) */



# ПРИМЕНА НА ОПЕРАТОРИ ЗА ДИРЕКТНИ ПРЕТВОРБИ



```
#include <stdio.h>

int main() {
    int a = 50;
    float b = 20.2;
    char c;
    c = (char)a + (char)b;
    printf("c = %c \n",c);
    return 0;
}
```

■ Програмата ќе испише  
c = F

```
void main() {
    int i=5, j=4;
    double f;
    f = (double) i/j;
    f = i / (double) j;
    f = (double)i /(double)j;
    f = (double) (i/j);
}
```

```
void main() {
    double d; float f;
    long l; int i;
    i=l=f=d= 10/3;
    d=f=l=i= 10/3;
    i=l=f=d= 10/3.;
    d=f=l=i= (double) 10/3;
    d=i=l=f= 10/3.0;
}
```



# ОПЕРАТОР ЗА ДОДЕЛУВАЊЕ

- Самата операција на доделување враќа вредност и може да се вметне во друг израз.
- Операторот за доделување е пофлексибилен отколку што тоа на прв поглед се чини, и истиот може да се употреби и на следниот начин:

```
int i, j, k, l, m, n;
```

```
i = j = k = l = m = n = 22;
```

- `n=22` е првата операција што се извршува, и тоа прави вредноста 22 да биде расположива за следната операција додели ја вредноста 22 на променливата `m`, итн.

```
printf("%d\n", j=93);
```

```
a=(b=c)+(d=e+2);
```

# КОМПРЕСИЈА НА ОПЕРАТОРИ

- Секој израз во следниот облик

*Promenliva* = *Promenliva* **op** *Vrednost*

- Може да се напише во следниот облик

*Promenliva* **op=** *Vrednost*

оператор	израз	еквивалентен израз
+=	x+=2;	x=x+2;
-=	x-=2;	x=x-2;
*=	x*=2; x*=a+b;	x=x*2; x=x*(a+b);
/=	x/=2; x/=j+2;	x=x/2; x=x/(j+2);
%=	x%=2;	x=x%2;



# ПРОГРАМА ВО С

## - пример -

```
#include<stdio.h>
void main( ) {
    int a,b,c,d;
    printf("Vnesi gi vrednostite za a,b,c,d\n");
    scanf("%d%d%d%d",&a, &b, &c, &d);
    a += b*c+d;
    printf("\n a = %d",a);
}
```

Ако корисникот ги внесе следните вредности за променливите  
a = 5, b= 5, c = 7, d = 8,  
излезот од програмата ќе гласи:

```
Vnesi gi vrednostite za
a,b,c,d
5
5
7
8
a = 48
```

# ОПЕРАТОРИТЕ ++ и --

- $i=i+1$  може да се напише и како  $++i$  или  $i++$
- $i=i-1$  може да се напише и како  $--i$  или  $i--$
- пример: Ако  $i=1$  излезот на следниот програмски сегмент

```
printf (" i = %d\n", i);  
printf (" i = % d\n", ++i);  
printf (" i = %d\n", i);
```

```
i = 1  
i = 2  
i = 2
```

- пример: Ако  $i=1$  излезот на следниот програмски сегмент

```
printf (" i = %d\n", i);  
printf (" i = %d\n", i++);  
printf (" i = %d\n", i );
```

```
i = 1  
i = 1  
i = 2
```



# ПРЕФИКС И ПОСТФИКС ВЕРЗИИ НА ++

## - пример -

```
#include <stdio.h>
int main() {
    int i, j=5;
    i = ++j;                /* ova e isto so j++; i=j; */
    printf("i=%d, j=%d\n", i, j);
    j = 5;
    i = j++;                /* ova e isto so i=j; j++; */
    printf("i=%d, j=%d\n", i, j);
    return 0;
}
```

# РЕЛАЦИСКИ ОПЕРАЦИИ

При пресметување на сите операции од табелата, резултатот ќе биде 1 ако условот е исполнет, односно 0 ако условот не е исполнет.

Оператори	Синтакса	Пример	Значење
<i>Релациски оператори</i>			
>	>	$x > y$	$x$ е поголемо од $y$
<	<	$x < y$	$x$ е помало од $y$
	>=	$x \geq y$	$x$ е поголемо или еднакво на $y$
	<=	$x \leq y$	$x$ е помало или еднакво на $y$
<i>Оператори за еднаквост</i>			
==	==	$x == y$	$x$ е еднакво на $y$
	!=	$x != y$	$x$ не е еднакво на $y$

# РЕЛАЦИСКИ ОПЕРАЦИИ

## ■ Примери

☐  $(7 == 5)$  враќа

НЕВИСТИНА (false)

☐  $(5 > 4)$  враќа

ВИСТИНА (true)

☐  $(3 != 2)$  враќа

ВИСТИНА (true)

☐  $(6 >= 6)$  враќа

ВИСТИНА (true)

☐  $(5 < 5)$  враќа

НЕВИСТИНА (false)

## ■ Нека е $a=2$ , $b=3$ $c=6$

☐  $(a == 5)$  враќа

НЕВИСТИНА (false)

☐  $(a*b >= c)$  враќа

ВИСТИНА (true)

☐  $(b+4 > a*c)$  враќа

НЕВИСТИНА (false)

☐  $((b=2) == a)$  враќа

ВИСТИНА (true)



# РЕЛАЦИСКИ ОПЕРАЦИИ

- Што ќе биде отпечатено со следнава програма?

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

```
int main() {
```

```
    printf("%d \n", "Malo" > "Golemo");
```

```
    printf("%d \n", "Golemo" > "Malo");
```

```
    return 0;
```

```
}
```

```
0
```

```
1
```

(но може да се случи да отпечати и 1 0 – **споредува покажувачи!!!**)

- Кај знаците важи

□ 'A' < 'F'

исто со 65 < 70

# ЛОГИЧКИ ОПЕРАТОРИ

- НЕ постои логички тип во C
- секоја ненулта вредност има значење вистина
- вредноста 0 има значење неистина (0.0, -0, +0)
  
- логичко **И** - **&&**
  - ☐ вредноста на изразот ќе биде вистина (различна од 0) ако и само ако двата операнди се вистина
- логичко **ИЛИ** - **||**
  - ☐ вредноста на изразот ќе биде вистина (различна од 0) ако барем еден од двата операнди е вистина
- логичко **НЕ** - **!**
  - ☐ вредноста на изразот ќе биде вистина (различна од 0) ако операндот има вредност неистина
- логичко **Исклучително ИЛИ (EXOR)** - **^**
  - ☐ вредноста на изразот ќе биде вистина (различна од 0) ако и само ако едниот операнд е вистина а другиот неистина



# ЛОГИЧКИ ОПЕРАТОРИ

- Приоритетот на операторот !
  - повисок од множење и делење,
  - ист е со операторите за инкрементирање и декрементирање
  - понизок е од заградите.
- Приоритетот на операторот &&
  - повисок од ||,
- Приоритетот на операторите && и ||
  - понизок од релациски оператори

Изразот  $a > b \ \&\& \ b > c \ || \ b > d$

ќе биде развиен како  $((a > b) \ \&\& \ (b > c)) \ || \ (b > d)$

- Примери со проблеми
  - $c == ' ' \ || \ c == '\t' \ || \ c == '\n'$  има секогаш вредност вистина. Зошто?
  - $1 < i < 10$  секогаш ќе биде вистина. Зошто?  
 $(1 < i) < 10$

# ПРЕСМЕТУВАЊЕ НА ЛОГИЧКИ ИЗРАЗИ

- Сите логички изрази се пресметуваат одлево надесно
- Пресметувањето се врши се додека не сме сигурни за вредноста на изразот
- Пример:
  - За  $i=11$  при пресметување на изразот  $(i < 10) \&\& (i > 5)$  ќе се пресмета само вредноста на изразот  $(i < 10)$  и бидејќи истата е 0, пресметувањето на целиот израз ќе прекине
- Пример:
  - Каква ќе биде вредноста на изразот  $!i == 5$ ?



# СТРАНИЧНИ ЕФЕКТИ

```
#include <stdio.h>

int main()
{
    int x,y,z,w;
    y=(x=4,z=2,w=1);
    w=(--x<2)&&(z=y);
    printf("x=%d y=%d z=%d w=%d\n",x,y,z,w)
    z=(y-=w)||(w++!=1);
    printf("x=%d y=%d z=%d w=%d\n",x,y,z,w)
    y=(--x>2)&&(--z==++w);
    printf("x=%d y=%d z=%d w=%d\n",x,y,z,w)
    w=(y||(z=(y+w||--x)&&(w==0)));
    printf("x=%d y=%d z=%d w=%d\n",x,y,z,w)
    return 0;
}
```

x=3	y=1	z=2	w=0
x=3	y=1	z=1	w=0
x=2	y=0	z=1	w=0
x=1	y=0	z=1	w=1



# ПРИОРИТЕТ И АСОЦИЈАТИВНОСТ НА ОПЕРАТОРИТЕ

## ■ Приоритет

- ☐ Во принцип сите унарни оператори имаат повисок приоритет од бинарните
- ☐ Употребата на загради го менува приоритетот
- ☐ Во C се дефинирани 15 нивоа

## ■ Асоцијативност

- ☐ За два оператора со ист приоритет, операцијата што треба да се изврши се избира на основа на правилата за асоцијативност на операторите
- ☐ Дефинирани се
  - „одлево надесно”
  - „оддесно налево”

# ПРИОРИТЕТ И АСОЦИЈАТИВНОСТ НА ОПЕРАТОРИТЕ

Ниво	Оператор						Вид	Редослед
Високо								
	( )	[ ]	->	.			Бинарен	Одлево надесно
	+ -	++ --	! ~	* &	sizeof		Унарен	Оддесно налево
	->*	.*					Бинарен	Одлево надесно
	*	/	%				Бинарен	Одлево надесно
	+	-					Бинарен	Одлево надесно
	<<	>>					Бинарен	Одлево надесно
	<	<=	>	>=			Бинарен	Одлево надесно
	==	!=					Бинарен	Одлево надесно
	&						Бинарен	Одлево надесно
	^						Бинарен	Одлево надесно
							Бинарен	Одлево надесно
	&&						Бинарен	Одлево надесно
							Бинарен	Одлево надесно
	? :						Тернарен	Одлево надесно
	=	+= -=	*= /=	^= %=	&=  =	<<= >>=	Бинарен	Оддесно налево
	Ниско	,						Бинарен



# ПРОГРАМА ВО С

## - пример -

Што ќе прикаже на компјутерскиот екран следната програма?

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

```
int main(){
```

```
    int i=0, j, k=7, m=5, n;
```

```
    j = m+= 2;
```

```
/* j = m = m+2; */
```

```
    printf("j= %d\n", j);
```

```
    j = k++ > 7;
```

```
/* j = k>7, k++ */
```

```
    printf("j= %d\n", j);
```

```
    j = i == 0 && k;
```

```
/* i==0, j = (1 && k) */
```

```
    printf("j= %d\n", j);
```

```
    return 0;
```

```
}
```



Програмирање и алгоритми

# ВНЕСУВАЊЕ И ПРИКАЗ НА ПОДАТОЦИ



# ПРИКАЗ НА ПОДАТОЦИ НА КОМПЈУТЕРСКИ ЕКРАН

## ■ Формат:

```
printf(format, izraz1, izraz2, ...);
```

□ format – текстуална низа

□ izraz1, izraz2, ... се индивидуални вредности што треба да се прикажат

## ■ Пример:

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

```
int main() {
```

```
    int term;
```

```
    term = 3 * 5;
```

```
    printf("Twice %d is %d\n", term, 2*term);
```

```
    printf("Three times %d is %d\n", term, 3*term);
```

```
    return 0;
```

```
}
```



# ПРИКАЗ НА ПОДАТОЦИ НА КОМПЈУТЕРСКИ ЕКРАН (II)

## ■ Специјални знаци во текстуалната низа за форматирање во `printf`:

% -> после овој знак се очекува спецификација за типот на податок кој ќе се прикаже на таа позиција (освен во случај кога после него следи знакот % - тогаш се испишува % на екран = **%%**)

## ■ Знаци на позиција помеѓу % и знакот за спецификација на тип на податок

. -> служи за вметнување на цифри за прецизност при печатење, а превзема вредност од индивидуалните вредности после контролната низа

број (пример 5) -> број на места за запис на податокот

.број (пример .5) -> број на децимални места при запис на бројот

+ -> испишување на знакот на бројот (дури и ако има +)

- -> порамнување на записот во лево



# ПРИМЕРИ СО ПРОБЛЕМИ

```
#include <stdio.h>

int main() {
    int zbir; /* zbir*/
    zbir = 2 + 2;
    printf("Zbirot e %d\n");
    return 0;
}
```

Случајна вредност од меморија

```
#include <stdio.h>

int main() {
    float kolicnik; /* kolicnik */
    kolicnik = 7.0 / 22.0;
    printf("Kolicnikot e %d\n", kolicnik);
    return 0;
}
```



# ВНЕСУВАЊЕ ПОДАТОЦИ ОД ТАСТАТУРА

- формат `scanf(format, &prom1, &prom2, ...);`
  - `format` – текстуална низа што ги содржи информациите за редоследот и распоредот на податоците што се внесуваат
    - ознаките за внесување на податоци
      - `%c` податокот е знак
      - `%d` податокот е цел број
      - `%f` податокот е реален број
      - `%h` податокот е цел број од типот `short int`
      - `%o` податокот е цел број во октална нотација
      - `%u` податокот е од типот `unsigned decimal integer`
      - `%x` податокот е цел број во хексадецимална нотација
  - `&` означува дека вредноста на променливите `prom1, prom2, ...` треба да се измени со сместување на вредноста што се внесува од тастатура
  - `prom1, prom2, ...` ги означуваат променливите во кои се внесуваат податоците



# ВНЕСУВАЊЕ ПОДАТОЦИ ОД ТАСТАТУРА

Пример:

```
#include <stdio.h>

int main() {
    int a, b;
    /*kreira dve celobrojni promenlivi*/

    printf("Vnesi dva broja: ");

    scanf("%d%d", &a, &b);
    /*smestuva celobrojni vrednosti vo promenlivate a i b*/

    printf("%d - %d = %d\n ", a, b, a-b);
    return 0;
}
```



# ПРОГРАМА ВО С

## - пример -

Да се напише програма за конверзија на Целзиусови во Фаренхајтови степени. Целзиусовите степени се внесуваат од тастатура, а формулата за претворање гласи:  $f = 1.8C + 32$

```
#include <stdio.h>
int main ( ) {
    float fahrenheit, celsius;
    scanf("%f", &celsius);
    fahrenheit = 1.8*celsius + 32.0;
    printf ("fahrenheit = %f\n", fahrenheit);
    return 0;
} /* end-main() */
```



# ВНЕСУВАЊЕ И ПРИКАЗ НА ЕДЕН ЗНАК

## ■ Формат

- внесување на знак од тастатура
  - `char getchar();`
- приказ на знак на компјутерски екран
  - `int putchar(char c);`

## ■ Пример:

```
#include <stdio.h>
int main ( ) {
    char bukva;
    bukva = getchar();
    putchar(bukva);
    printf ("\n vnesena e bukvata = %c", bukva);
    return 0;
} /* end-main() */
```



# ДЕКЛАРИРАЊЕ КОНСТАНТИ

- Формат

- `const tipPromenliva Ime=Vrednost;`

- Пример:

- `const float PI = 3.1415;    /* vrednost na PI*/`

- Константата се иницијализира само при нејзината декларација и понатаму нејзината вредност не може да се измени.



# ТЕРНАРЕН УСЛОВЕН ОПЕРАТОР (:?)

## ■ Има три операнди:

- услов
- вредност ако условот е исполнет (vrednost\_T)
- вредност ако условот не е исполнет (vrednost\_F)

(uslov ? vrednost\_T : vrednost\_F)

## ■ Пример:

```
a = b>5 ? c : d;
```

```
printf("%s\n", grade >= 60 ? "Polozil" : "Padna" );
```



# КРАЈ