



Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Операции

Структурно програмирање

ФИНКИ 2014

Содржина

- **Аритметички операции - продолжение**
- Промена на вид на вредност – **cast** операција
- Логички оператори
- Релациски оператори

Аритметички операции - Пример за операции со цели и реални броеви

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

Решете сега!

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() */
```

Содржина

- Аритметички операции - продолжение
- **Промена на вид на вредност – `cast` операција**
- Логички оператори
- Релациски оператори

Промена на вид 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 чиј код е 70 ('F')

(unsigned short) 3.14

3.14 во 3 (unsigned short)

- Со следнава програма се илустрира примената на операторите 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

- Со следнава програма се илустрира примената на операторите за директни претворби

```
int main() {
    int i=5, j=4;
    double f;
    f = (double) i/j; // f=1.25
    f = i / (double) j; // f=1.25
    f = (double) i / (double) j;
        // f=1.25
    f = (double) (i/j); // f=1.0
    return 0;
}
```

```
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 флексибилен:

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

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

- која променлива прва ќе добие вредност?

n

- 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 ;

```
#include<stdio.h>
int 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);
}
```

За променливите

a = 5, b= 5, c = 7, d = 8,
излезот од програмата
ќе гласи:



КОЛКУ Е а?

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

Оператори ++ 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:

За $i=1$ за следниот програмски сегмент

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

- излезот е :

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

- пример:

- Нека $i=1$

- за следниот програмски сегмент

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

- излезот е:

```
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;  
}
```

Ќе се отпечати:

i=6 j=6

i=5 j=6

Содржина

- Аритметички операции - продолжение
- Промена на вид на вредност – **cast** операција
- **Релациски оператори**
- Логички оператори

Релациски операции

Оператори	Синтакса	Пример	Значење
Релациски оператори			
>	>	$x > y$	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
Оператори за еднаквост			
=/=	==	$x == y$	x е еднакво на y
	!=	$x != y$	x не е еднакво на y

Правила за логичките типови

- **НЕ** постои логички тип во C
- секоја ненулева вредност има значење **вистина**
- вредноста **0** има значење **невистина** (0.0 , -0 , $+0$)
- при пресметување на сите операции од табелата, резултатот ќе биде **1** ако условот е исполнет, односно **0** ако условот не е исполнет

Релациски операции (2)

■ Примери

- $(7 == 5)$ враќа невестина (false)
- $(5 > 4)$ враќа вистина (true)
- $(3 != 2)$ враќа вистина (true)
- $(6 \geq 6)$ враќа вистина (true)
- $(5 < 5)$ враќа невестина (false)

Задача:

Нека е $a=2, b=3, c=6$

$(a == 5)$ враќа:
невестина (false)

$(a*b \geq c)$ враќа:
вистина (true)

$(b+4 > a*c)$ враќа:
невестина (false)

$((b=2) == a)$ враќа:
вистина (true)

Содржина

- Аритметички операции - продолжение
- Промена на вид на вредност – **cast** операција
- Релациски оператори
- **Логички оператори**

Логички оператори (1)

■ 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 - ^

- вредноста на изразот ќе биде вистина (различна од 0) ако и само ако едниот операнд е вистина а другиот неистина

Израз	Резултат
<code>true && false</code>	<code>false</code>
<code>true false</code>	<code>true</code>
<code>!false</code>	<code>true</code>

Логички оператори (2)

■ Приоритетот на операторот ! е

- ☐ повисок од множење и делење,
- ☐ ист е со операторите за инкрементирање и декрементирање
- ☐ понизок е од заградите.

■ Операторот && има

- ☐ повисок приоритет од операторот ||
- ☐ и двата имаат понизок приоритет од релациските оператори.

■ Така 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?**

Пресметување на логички изрази (2)

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

Што ќе се отпечати на
компјутерскиот екран?

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

- ☐ 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” и „оддесно налево”

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	Одлево надесно

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++ */
```

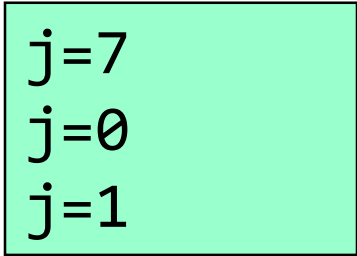
```
    printf("j= %d\n", j);
```

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

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

```
    return 0;
```

```
}
```



```
j=7  
j=0  
j=1
```


Прашања?