Laborator 5-6. Operatori în limbajul C. Prioritatea operatorilor. Operatori aritmetici, relaţionali, logici, pe biţi, de atribuire, de incrementare/decrementare, condiţionali

5.1. Clasificarea operatorilor

După numărul operanzilor: unari, binari, ternari.
După ordinea operanzilor: prefixați, infixați, postfixați.
După tipul și destinația operanzilor: aritmetici, relaționali, logici, pe biți, de atribuire, de incrementare/ decrementare, condiționali, secvențial, de adresare, de conversie explicită (cast), de dimensiune (sizeof).

5.2 Operatorii aritmetici

În ordinea descrescătoare a priorității:

Nr. crt.	Categorie	Denumire	Sintaxa	Exemple
1	Operatori unari de păstrare / schimbare a semnului	Plus unar Minus unar	+opd -opd	+a -b
2	Operatori binari multiplicativi	De înmulțire De împărțire Modulo (restul împărțirii întregi)	opd1*opd2 opd1/opd2 opd1%opd2	a*b c/d n%i
3	Operatori binari aditivi	De adunare De scădere	opd1+opd2 opd1-opd2	a+b c+d

Ex1. Să se evalueze expresia: $z = \frac{15x^2 + 21x - 3}{y - 2}$, $y \neq 2$.

```
//program expresii aritmetice
#include <stdio.h>
void main(void)
{
float x,y,z;
printf("\nx=");
scanf("%f",&x);
printf("y=");
scanf("%f",&y);
z=(15*x*x+21*x-3)/(y-2);
printf("z=%f\n",z);
}
```

Ex2. Să se determine câtul `c` şi restul `r` a împărţirii întregi a două numere `p` şi `q`.

```
//program impartire intreaga
#include <stdio.h>
#include <conio.h>
void main(void)
{
int p,q,c,r;
printf("\np=");
scanf("%d",&p);
printf("q=");
scanf("%d",&q);
c=p/q; r=p%q;
printf("c=%-07dr=%-07d\n",c,r);
}
```

5.3 Operatorii relaționali

Prioritatea lor este egală. Sunt operatori binari. Apar în expresii cu rol de condiţie, valorile acestora fiind logice (0 – fals sau 1 – adevărat).

Nr. crt.	Categorie	Denumire	Sintaxa	Exemple
1	Operatori relaţionali mai mic / egal	Mai mic Mai mic sau egal	opd1 < opd2 opd1 <= opd2	a < b c <= d
2	Operatori relaţionali mai mare / egal	Mai mare Mai mare sau egal	opd1 > opd2 opd1 >= opd2	a > b c >= d
3	Operatori relaţionali egal / diferit	Egal (test de egalitate) Diferit	opd1 == opd2 opd1!= opd2	a==b c!=d

Ex3. Determinaţi valoarea de adevăr a expresiilor:
a mai mic decât b
c mai mare sau egal cu d
a diferit de d
a egal cu c
pentru valorile următoare: a = 3, b = 5, c = 3, d = 3.35.

5.4 Operatorii logici

Nr. crt.	Categorie	Denumire	Sintaxa	Exemple
1	Operatori logici unari	Negaţie (NU)	!opd	!a
2	Operatori logici binari	ŞI logic	opd1&&opd2	b&&c
3	Operatori logici binari	SAU logic	opd1||opd2	p||q

Obs: în C nu există în mod explicit valori de tip logic; de aceea, valoarea de tip false este reprezentată printr-o valoare egală cu zero, iar valoarea de tip true printr-o valoare diferită de zero.

Ex4. Ştiind că a=23 şi b=50, evaluaţi următoarele expresii de tip logic:
a. (a!=b) and (a>b); R= 0.
b. ((a+10) <b) or false; R= 1.
c. true and (a!=b); R=1.
d. b mod 10 > a div 7; R= 0.
e. not false and (a div 10 < b); R=1.
f. not (true or (a+b<10)); R=0.

5.5 Operatori pe biţi

Nr. crt.	Categorie	Denumire	Sintaxa	Exemple
1	Operatori logici pe biţi	complement faţă de 1 (unar)	~opd	~a
2	Operatori logici pe biţi	ŞI logic pe biţi	opd1&opd2	b&c
3	Operatori logici pe biţi	SAU EXCLUSIV pe biţi	opd1^opd2	c^d
4	Operatori logici pe biţi	SAU logic pe biţi	opd1|opd2	p|q
5	Operatori de deplasare pe biţi	deplasare spre stânga cu n poziţii	opd<<n	h<<2
6	Operatori de deplasare pe biţi	deplasare spre dreapta cu n poziţii	opd>>n	k>>3

Obs: operatorii pe biţi se aplică doar asupra operanzilor de tip întreg.

5.6 Operatori de incrementare şi decrementare (unari)

Nr. crt.	Categorie	Denumire	Sintaxa	Exemple
1	Operatori prefixaţi	Incrementare prefixată	++opd	++a
2	Operatori prefixaţi	Decrementare prefixată	--opd	--b
3	Operatori postfixaţi	Incrementare postfixată	opd++	c++
4	Operatori postfixaţi	Decrementare postfixată	opd--	d--

Obs: în cazul operatorilor prefixaţi, mai întâi are loc incrementarea/decrementarea operandului, valoarea rezultată fiind mai mare/mai mică cu o unitate faţă de valoarea iniţială a operandului. În cazul operatorilor postfixaţi, incrementarea/decrementarea operandului se realizează după utilizarea valorilor operandului, valoarea rezultată fiind chiar valoarea iniţială a operandului.

Ex5. Să se studieze comportamentul operatorilor de incrementare/decrementare.

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
int a = 1, b = 2, i_pre, i_post, d_pre, d_post;
i_pre = ++a + b;        //rezultat 4, a = 2
i_post = a++ + b;       //rezultat 4, a = 3
d_pre = --a + b;        //rezultat 4, a = 2
d_post = a-- +b;        //rezultat 4, a = 1
printf("Incrementare prefixata: %d\n", i_pre);
printf("Incrementare postfixata: %d\n", i_post);
printf("Decrementare prefixata: %d\n", d_pre);
printf("Decrementare prefixata: %d\n", d_post);
getch();
}
```

5.7 Operatorul de atribuire

Expresie de atribuire: v = expresie;
Expresie de atribuire multiplă: vn = ... = v2 = v1 = v = expresie;
Expresie de atribuire combinată: v op = expresie;
(echivalentă cu: v = v op expresie;
unde op poate fi oricare din operatorii: +,-,*,/,%,&,|,^,<<,>>)

5.8 Operatorul condiţional ternar (?:)

Se utilizează în expresii de forma:

exp1?exp2:exp3

Efect: se evaluează exp1; dacă este adevărată (valoare diferită de zero), se va evalua exp2, care va da valoarea expresiei finale; dacă nu (valoare zero), se evaluează exp3 , care va determina valoarea expresiei condiţionale.

Ex6. Să se determine maximul şi minimul dintre două numere date:

```
//program expresii condiţionale
#include <stdio.h>
void main(void)
{
int a,b,max,min;
printf("\na=");
scanf("%d",&a);
printf("b=");
scanf("%d",&b);
max=a>b?a:b;
```

```
min=a<b?a:b;
printf("max=%d\tmin=%d\n",max,min);
}
```

5.9 Operatorul secvenţial (virgulă)

Operatorul virgulă permite gruparea mai multor expresii, astfel încât să fie tratate din punct de vedere sintactic, ca o singură expresie. Se evaluează expresiile separate prin virgulă, în ordinea de la stânga la dreapta, valoarea întregii expresii fiind egală cu valoarea *expresiei_n* (ultima expresie).
Sintaxa: expresie_1, expresie_2, ..., expresie_n

Ex7. Să se calculeze valoarea expresiei: a=4, b=3, c=a+b, d=2*c

```
#include <stdio.h>
void main(void)
{
int a, b, c, d;
a=4, b=3, c=a+b, d=2*c;
printf("d = %d\n", d);                    //raspuns:  d=14
}
```

5.10 Operatorul de conversie explicită (cast)

Operatorul converteste o variabilă dintr-un tip în altul.
Sintaxa: (tip_conversie) expresie

Ex8. Convertiti o variabila de tip double in int.

```
#include <stdio.h>
void main(void)
{
double pi = 3.1415;
printf("Valoarea convertita = %d", (int)pi);                    // raspuns=3
}
```

5.11 Operatorul de dimensiune (sizeof)

Permite aflarea dimensiunii in octeti a unei date sau expresii in functie de tipul declarat.
Sintaxa:

sizeof (expresie)
sizeof (tip)

Dimensiunea in octeti a tipurilor standard:

DENUMIRE TIP		CUVÂNT REZERVAT	VALORI POSIBILE	LUNGIME
PREDEFINITE (STANDARD SAU FUNDAMENTALE)	ÎNTREGI	Char	-128.....127	1octet
		Unsigned char	0.....255	
		Signed char	-128.....127	
		Int	-32768...32767	2octeți
		Unsigned int	0...65535	
		Signed int	-32768...32767	
		Short int	-32768...32767	
		Long int	-2147483648...2147483647	4octeți
		Signed long int	-2147483648...2147483647	
		Unsigned long int	0...4294967295	
	REALE	Float	10-37...1037	
		Double	1.7e-308...1.7e308	8octeți
		Long double	3.4e-4932...3.4e4932	10octeți

Ex9. Determinaţi lungimea tipurilor: short int, long double.