Fundamentele programării
Curs 2

Șef lucr.dr.ing. GENGE Béla

Universitatea "Petru Maior", Departamentul de Informatică
Tîrgu Mureș, România
{bela.genge}@ing.upm.ro

Operatori C

- De atribuire simplă
- Aritmetici
- Relaționali
- Logici
- La nivel de bit
- De atribuire compusă
- De conversie explicită (cast)
- Sizeof
- De adresare
- Condițional
- Virgulă
- #

Operator	Priority	Description
()	1	Function call operator
[]	1	Subscript operator
−>	1	Element selector
!	2	Boolean NOT
~	2	Binary NOT
++	2	Post-/Preincrement
−−	2	Post-/Predecrement
−	2	Unary minus
(*type*)	2	Type cast
*	2	Derefence operator
&	2	Address operator
sizeof	2	Size-of operator
*	3	Multiplication operator
/	3	Division operator
%	3	Modulo operator
+	4	Addition operator
−	4	Subtraction operator
<<	5	Left shift operator
>>	5	Right shift operator
<	6	Lower-than operator
<=	6	Lower-or-equal operator
>	6	Greater-than operator
>=	6	Greater-or-equal operator
==	7	Equal operator
!=	7	Not-equal operator
&	8	Binary AND
^	9	Binary XOR
|	10	Binary OR
&&	11	Boolean AND
||	12	Boolean OR
?:	13	Conditional operator
=	14	Assignment operator
op=	14	Operator assignment operator
,	15	Comma operator

Operatori de atribuire simplă

- Notat cu '='
- Realizează memorarea valorii unei expresii într-o variabilă
- Pentru declarațiile:
  - `char c;`
  - `int i, j, k;`
  - `float x;`
- Avem atribuirile:
  - `c = 'a';`
  - `i = 2; j = 10; k = 50;`
  - `x = 34.989;`
- Atribuiri multiple:
  - `i = j = k = 50;`
- Atribuiri incorecte:
  - `i = j + 1 = 50;`

Operatori aritmetici

- Operatori aritmetici de bază: +, -, *, /
- Operatorul modulo: % (restul împărțirii a două numere întregi)
- Operatori unari: + (păstrarea semnului), -, ++ (incrementare), – (decrementare)

Exemple

Folosirea operatorilor aritmetici

```
int i, j, n;
float x;
n = 10 * 5 - 7 + 2;
i = 9 / 2;
j = 9 % 2;
x = n;
x = x % i; // Atenție !
```

Conversii

```
int i;
float x;
i = 11./2;
x = 11./2;
i = 5 / 2 + 7 / 2;
i = 5. / 2 + 7 / 2.;
```

Exemple

Folosirea operatorilor aritmetici

```
int i, j, n;
float x;
n = 10 * 5 - 7 + 2;
i = 9 / 2;
j = 9 % 2;
x = n;
x = x % i; // Atenție !
```

Conversii

```
int i;
float x;
i = 11./2;
x = 11./2;
i = 5 / 2 + 7 / 2;
i = 5. / 2 + 7 / 2.;
```

Exemple

- **++var**: preincrementare
- **−−var**: predecrementare
- **var++**: postincrementare
- **var−−**: postdecrementare

Pre/post- incrementare/decrementare

```
int i, j = 10;
i = j++;
// sau
i = ++j;
```

Pre/post- incrementare/decrementare

```
int i, j = 10;
i = 100 + j--;
// sau
i = 100 + --j;
```

Exemple

- **++var**: preincrementare
- **−−var**: predecrementare
- **var++**: postincrementare
- **var−−**: postdecrementare

Pre/post- incrementare/decrementare

```
int i, j = 10;
i = j++;
// sau
i = ++j;
```

Pre/post- incrementare/decrementare

```
int i, j = 10;
i = 100 + j--;
// sau
i = 100 + --j;
```

Operatori relaționali

- $>$: mai mare
- $<$: mai mic
- $>=$: mai mare sau egal
- $<=$: mai mic sau egal
- $==$: egal
- $!=$: diferit
- Rezultatul este 1 (relație îndeplinită) sau 0 (relație neîndeplinită)

Exemplu

```
int i = 2, j = 3; int egal, mare, mic, diferit;
egal = i == j;
mare = i > j;
mic = i < j;
diferit = i != j;
cout << egal << endl << mare << endl << mic << endl <<
diferit << endl;
```

Operatori relaționali

- $>$: mai mare
- $<$: mai mic
- $>=$: mai mare sau egal
- $<=$: mai mic sau egal
- $==$: egal
- $!=$: diferit
- Rezultatul este 1 (relație îndeplinită) sau 0 (relație neîndeplinită)

Exemplu

```
int i = 2, j = 3; int egal, mare, mic, diferit;
egal = i == j;
mare = i > j;
mic = i < j;
diferit = i != j;
cout << egal << endl << mare << endl << mic << endl <<
diferit << endl;
```

Operatori logici

- **&&**: ȘI (AND)
- ||: SAU (OR)
- **!**: NEGARE (NOT)

- În C **NU** există tipul **boolean**
- Orice operand cu valoarea 0 este considerat false
- Orice operand cu valoarea != 0 este considerat adevărat

Tabel de adevăr

x	y	x && y	x || y	!x
0	0	0	0	1
0	!=0	0	1	1
!=0	0	0	1	0
!=0	!=0	1	1	0

Operatori logici

- **&&**: ȘI (AND)
- ||: SAU (OR)
- **!**: NEGARE (NOT)

- În C **NU** există tipul **boolean**
- Orice operand cu valoarea 0 este considerat false
- Orice operand cu valoarea != 0 este considerat adevărat

Tabel de adevăr

x	**y**	**x && y**	**x || y**	**!x**
0	0	0	0	1
0	!=0	0	1	1
!=0	0	0	1	0
!=0	!=0	1	1	0

Operatori logici

Exemplu - verificare condiție ca x și y să fie 0

```
0 == x && 0 == y;
// sau
!x && !y;
// sau
!(0 != x && 0 != y);
```

- Observație: folosiți forma **0 == x** în loc de **x == 0**

Operatori logici

Exemplu - verificare condiție ca x și y să fie 0

```
0 == x && 0 == y;
// sau
!x && !y;
// sau
!(0 != x && 0 != y);
```

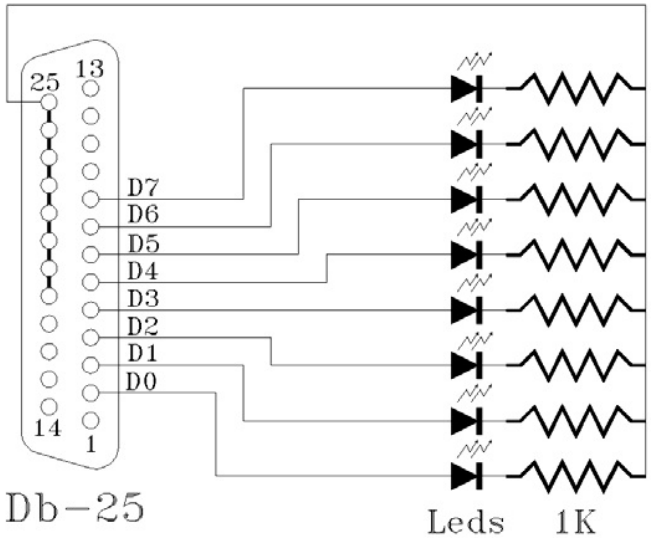
- Observație: folosiți forma **0 == x** în loc de **x == 0**

Operatori la nivel de bit

- &: ȘI
- |: SAU
- ^: SAU EXCLUSIV
- ~: NEGARE
- <<: DEPLASARE STÂNGA
- >>: DEPLASARE DREAPTA

- Exemple de utilizare:
  - Protocoale binare
  - Comunicare cu hardware (testare/setare flag-uri)
  - Controlul echipamentelor: LED-uri legate pe portul paralel

Exemple



Exemplu

```
unsigned char i = 1 << 1; //0 0 0 0 0 0 1 0
cout << i;
i = i << 1; //0 0 0 0 0 1 0 0
cout << i;
unsigned char j = i >> 2; //0 0 0 0 0 0 0 1
cout << j;
```

Operatori de atribuire compusă

- Combinarea operatorului de atribuire cu operatori aritmetici și operatori la nivel de bit
- += -= *= /= %=
- |= &= ^= <<= >>=

Exemplu

```
int i = 2;
// Incrementare cu 10
i = i + 10;
// sau
i += 10;  // Deplasare la stanga cu 1
i = i << 1;
// sau
i <<= 1;
```

Operatori de atribuire compusă

- Combinarea operatorului de atribuire cu operatori aritmetici și operatori la nivel de bit
- += -= *= /= %=
- |= &= ^= <<= >>=

Exemplu

```
int i = 2;
// Incrementare cu 10
i = i + 10;
// sau
i += 10;  // Deplasare la stanga cu 1
i = i << 1;
// sau
i <<= 1;
```

Operatorul de conversie explicită

- Sintaxa: `(tip_conv) expresie`

Exemplu: caractere

```
char c = 'A';
// Afisare caracter fără conversie cout << c; // Afișare cod
ASCII prin conversie cout << (int)c;
```

Exemplu: numere reale

```
float x = 7.23;
int i;
i = (int)x;
```

Operatorul de conversie explicită

- Sintaxa: `(tip_conv) expresie`

Exemplu: caractere

```
char c = 'A';
// Afisare caracter fără conversie cout << c; // Afișare cod
ASCII prin conversie cout << (int)c;
```

Exemplu: numere reale

```
float x = 7.23;
int i;
i = (int)x;
```

Operatorul de conversie explicită

- Sintaxa: `(tip_conv) expresie`

Exemplu: caractere

```
char c = 'A';
// Afisare caracter fără conversie cout << c; // Afișare cod
ASCII prin conversie cout << (int)c;
```

Exemplu: numere reale

```
float x = 7.23;
int i;
i = (int)x;
```

Operatorul sizeof

- Sintaxa: `sizeof(expresie)`
- Returnează dimensiunea în octeți a unui tip de date sau a unei expresii

Exemplu

```
int i;
float x;
// Afisare dimensiune
cout << "Dimensiunea unui intreg este: " << sizeof(i) <<
endl;
cout << "Dimensiunea unui numar real este: " << sizeof(x) <<
endl;
// Stocare dimensiune
int j = sizeof(x);
```

Operatorul sizeof

- Sintaxa: `sizeof(expresie)`
- Returnează dimensiunea în octeți a unui tip de date sau a unei expresii

Exemplu

```
int i;
float x;
// Afisare dimensiune
cout << "Dimensiunea unui intreg este: " << sizeof(i) <<
endl;
cout << "Dimensiunea unui numar real este: " << sizeof(x) <<
endl;
// Stocare dimensiune
int j = sizeof(x);
```

Operatorul de adresare

- Operatorii de adresare sunt:
  - **[]**: indexare
  - **.**: selecție directă
  - **->**: selecție indirectă
  - **&**: determinare adresă
  - ***:** adresare indirectă

Exemplu: afișarea adresei

```
int i;
cout << &i;
unsigned int adr = (unsigned int)&i;
cout << adr;
```

Operatorul de adresare

- Operatorii de adresare sunt:
  - **[]**: indexare
  - **.**: selecție directă
  - **->**: selecție indirectă
  - **&**: determinare adresă
  - *: adresare indirectă

Exemplu: afișarea adresei

```
int i;
cout << &i;
unsigned int adr = (unsigned int)&i;
cout << adr;
```

Operatorul condițional

- Sintaxa: `expr_cond ? rezultat_1:rezultat_2;`
- Semnificație: dacă expr_cond atunci rezultat = rezultat_1, altfel rezultat = rezultat_1

Exemplu: maximul

```
double a, b;
double max;
max = (a > b)? a:b;
cout << max;
```

Exemplu: mesaj

```
int i = 3;
cout << (0 == i)? "este egal cu 0":"este diferit de 0";
```

Operatorul condițional

- Sintaxa: `expr_cond ? rezultat_1:rezultat_2;`
- Semnificație: dacă expr_cond atunci rezultat = rezultat_1, altfel rezultat = rezultat_1

Exemplu: maximul

```
double a, b;
double max;
max = (a > b)? a:b;
cout << max;
```

Exemplu: mesaj

```
int i = 3;
cout << (0 == i)? "este egal cu 0":"este diferit de 0";
```

Operatorul condițional

- Sintaxa: `expr_cond ? rezultat_1:rezultat_2;`
- Semnificație: dacă expr_cond atunci rezultat = rezultat_1, altfel rezultat = rezultat_1

Exemplu: maximul

```
double a, b;
double max;
max = (a > b)? a:b;
cout << max;
```

Exemplu: mesaj

```
int i = 3;
cout << (0 == i)? "este egal cu 0":"este diferit de 0";
```

Operatorul

- # este utilizat pentru preprocesare

Exemplu

```
#include <stdio.h>
#define ok 1
#define nok 0
...
test = (0 == i)? ok:nok;
cout << test;
```

Operatorul

- # este utilizat pentru preprocesare

Exemplu

```
#include <stdio.h>
#define ok 1
#define nok 0
...
test = (0 == i)? ok:nok;
cout << test;
```

Problemă

- S-a măsurat grosimea gheții pe râul Mureș (în metrii) în ziua de ieri și astăzi. Patinoarul poate fi deschis doar dacă în cele două zile consecutive grosimea gheții a fost cel puțin M centimetrii. Scrieți un program care determină:
  - Când a fost mai groasă gheața, ieri sau astăzi
  - Diferența absolută (fără semn) de grosime între ziua de ieri și astăzi în centimetrii și metrii
  - Dacă se poate deschide patinoarul