

6. Aplicații folosind operatori și expresii C/C++ (Applications using C/C++ operators and expressions)

1. Obiective:

- Înțelegerea noțiunilor de *expresie*, *operand* și *operator*
- Trecerea în revistă a principalilor operatori și reținerea celor mai importanți
- Înțelegerea modului în care se face evaluarea expresiilor
- Scrierea și rularea de programe simple care folosesc acești operatori

1. Objectives:

- Understanding the concept of *expression*, *operand* and *operator*
- Reviewing the main operators and remembering the most important ones
- Understanding the expression evaluation
- Writing and running some simple applications that use operators

2. Breviar teoretic

Operatorii sunt simboluri ce specifică operațiile de efectuat asupra operandilor. În urma aplicării unui operator se obține un rezultat.

Operatorii limbajului C se pot grupa pe clase astfel:

- aritmetici
 - unari : + (fără efect), - (negativare)
 - binari :
 - *multiplicativi*
 - *, *înmulțire* ;
 - /, *împărțire*, dacă sunt întregi rezultatul va fi partea întreagă a câtului;
 - %, *modulo*, doar pentru operandi întregi și are ca și rezultat restul împărțirii întregi;
 - *aditivi*
 - +, *adunare*
 - -, *scădere*
- de relație și de egalitate
 - *operatori de relație*, care sunt: <, <=, >, >=,
 - *operatori de egalitate*, care sunt:
 - ==, *egal*
 - !=, *diferit*.
- logici
 - *operatori pe operand* (normali) de următoarele tipuri:
 - !, negația logică, care se utilizează astfel:
 - !operand = 0 (False), dacă operand ≠ 0
 - !operand = 1 (True), dacă operand = 0.
 - &&, și logic,
 - E1 && E2, și are valoarea 1 (True) dacă E1 și E2 sunt ambele True (!=0), în rest are valoarea 0 (False).
 - ||, sau logic,
 - E1 || E2 care are valoarea 0 (False) dacă ambele expresii E1 și E2 au valoarea 0 (False). În rest expresia are valoarea 1 (True).
 - *operatori pe biți*
 - ~, complement față de unu, schimbă fiecare bit 1 al operandului în 0, respectiv fiecare bit 0 în 1;
 - <<, deplasare la stânga a valorii primului operand cu un număr de poziții egal cu valoarea celui de-al doilea operand, forma de utilizare fiind:
 - op1 << op2 este echivalent cu înmulțirea cu puteri ale lui 2;
 - >>, deplasare la dreapta a valorii primului operand cu un număr de poziții egal cu valoarea celui de-al doilea operand, forma de utilizare fiind:

- $op1 >> op2$ este echivalent cu împărțirea cu puteri ale lui 2;
 - **&, și logic pe biți**, se execută bit cu bit coform lui ȘI-LOGIC; este folosit la lucrul cu măști;
 - **^, sau exclusiv pe biți**, se execută bit cu bit conform lui SAU-EXCLUSIV; este folosit pentru a anula sau seta diferiți biți.
 - **|, sau logic pe biți**, se execută bit cu bit conform lui SAU-LOGIC; este folosit pentru a seta diferiți biți.
- de atribuire (asignare)
 - **asignare simplă**, care se notează prin caracterul = și are forma: $v=(expresie)$ unde v este o variabilă simplă, referință la un element de tablou sau de structură. Se *asociază de la dreapta la stânga* astfel: $v_n=...=v_1=v=expresie$; asignarea făcându-se mai întâi $v=expresie$ și apoi se asociază lui v_1 apoi v_2 și tot așa până la v_n .
 - **asignare compusă**, caz în care pentru atribuire se folosește forma: $operator=$, unde $operator$ poate fi unul binar aritmetic sau unul logic pe biți, și deci poate fi unul dintre următoarele: /, %, *, -, +, <<, >>, &, ^, |.
 - Structura: $var operator= expresie$ este echivalentă cu $var = var operator (expresie)$
 - **incrementare și decrementare**, sunt operatori unari. Aceștia sunt ++ și --, putând fi *postfixați* (după operand), respectiv *prefixați* (înaintea operandului).
- de forțare a tipului sau de conversie explicită (cast)
 - (*tip*) *operand* prin care valoarea operandului se convertește la tipul indicat.
- de determinare a dimensiunii
 - **sizeof data sau sizeof (tip)**, determinând dimensiunea în octeți a unei date sau a unui tip;
- de adresare (&, referențiere) și de indirectare (*, dereferențiere)
- paranteză
 - **parantezele rotunde**, ()
 - **paranteze pătrate**, []
- condiționali
 - $E1 ? E2 : E3$, unde $E1, E2, E3$ sunt expresii.
- virgulă
 - leagă două expresii în una singură astfel: $exp1, exp2$
- alți operatori (la structuri):
 - punct .
 - săgeată ->

O *expresie* poate folosi operatori din clase diferite, la *evaluarea* ei ținându-se cont de:

- **precedență** (prioritate): dă ordinea de efectuare a operațiilor;
- **asociativitate**: indică ordinea de efectuare a operațiilor care au aceeași precedență; poate fi stânga->dreapta (S->D) sau dreapta->stânga (D->S);
- **regula conversiilor implicite**: asigură stabilirea unui tip comun pentru ambii operanzi, la fiecare operație care solicită acest lucru și în care tipurile diferă; regula de bază: se promovează tipul cu domeniu de valori mai mic către tipul cu domeniul de valori mai mare;

Ordinea de evaluare dată de precedență și asociativitate poate fi modificată grupând operațiile cu ajutorul parantezelor.

3. Exemple

Operatori condiționali

Exemplul 1:

```
#include <iostream>
using namespace std;
```

```

int main (void)
{
double n1, n2, n3;
    cout << "Introduceti 3 numere: ";
    cin >> n1 >> n2 >> n3;
    cout << (n1 <= n2 && n2 <= n3 ? "Sunt in ordine" : "Nu sunt in ordine") << '\n';
    return 0;
} //end main

```

Operatori logici

Exemplul 2:

```

#include <iostream>
using namespace std;

int isAlpha (char ch);

int main (void)
{
char ch;
    cout << "Introduceti un caracter: ";
    cin >> ch;
    cout << (isAlpha(ch) != 0 ? "Este o litera" : "Nu este o litera") << '\n';
    return 0;
} //end main

int isAlpha (char ch)
{
    return ch >= 'a' && ch <= 'z' || ch >= 'A' && ch <= 'Z';
} // end isAlpha

```

Operatori pe biți

Exemplul 3: program care aplică operatorii pe biți unor valori după care le afișează în zecimal și octal.

```

#include <iostream>
using namespace std;

int main(void)
{
unsigned char x = '\011';
unsigned char y = '\027';
    cout << "x = " << oct << short(x) << '\n';
    cout << "y = " << oct << short(y) << '\n';
    cout << "~x = " << oct << (short)(~x) << '\n';
    cout << "x & y = " << oct << (short)(x & y) << '\n';
    cout << "x / y = " << oct << (short)(x / y) << '\n';
    cout << "x ^ y = " << oct << (short)(x ^ y) << '\n';
    cout << "x << 2 = " << oct << (short)(x << 2) << '\n';
    cout << "x >> 2 = " << oct << (short)(x >> 2) << '\n';
    return 0;
} //end main

```

Operatori de atribuire multiplă și de incrementare/decrementare

Exemplul 4: program care citește de la consolă un număr întreg și realizează cu el operații de: înmulțire, împărțire modulo, sau exclusiv pe biți, și logic pe biți.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <conio.h>

int main(void)
{
    int x;

    printf("Introduceți numărul x: ");
    scanf("%d", &x);

    x ^= 0x0f;
    printf("\n%x", x);

    x %= 2;
    printf("\n%d", x);

    x &= 0x01;
    printf("\n%x\n", x);

    x *= 4;
    printf("\n%d", x);
    return 0;
} //end main
```

Exemplul 5: program care citește un număr întreg și aplică operatorul de incrementare postfixat și operatorul de decrementare prefixat, afișând de fiecare dată rezultatul.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <conio.h>

int main(void)
{
    int x;

    printf("Introduceți numărul x: ");
    scanf("%d", &x);

    printf("\n%d", x++);
    printf("\n%d", x);

    printf("\n%d", --x);
    printf("\n%d\n", x);
    return 0;
} //end main
```

Operatori de determinare a dimensiunii și de forțare a tipului

Exemplul 6: program care afișează dimensiunea (în octeți) pentru unele tipuri de date și pentru niște constante.

```
#include <iostream>
using namespace std;
```

```

int main (void)
{
    cout << "dimensiunea tipului char = " << sizeof(char) << " octeti\n";
    cout << "dimensiunea tipului char* = " << sizeof(char*) << " octeti\n";
    cout << "dimensiunea tipului short = " << sizeof(short) << " octeti\n";
    cout << "dimensiunea tipului int = " << sizeof(int) << " octeti\n";
    cout << "dimensiunea tipului long = " << sizeof(long) << " octeti\n";
    cout << "dimensiunea tipului float = " << sizeof(float) << " octeti\n";
    cout << "dimensiunea tipului double = " << sizeof(double) << " octeti\n";
    cout << "dimensiunea pentru 1.55 = " << sizeof(1.55) << " octeti\n";
    cout << "dimensiunea pentru 1.55L = " << sizeof(1.55L) << " octeti\n";
    cout << "dimensiunea pentru SALUT = " << sizeof("SALUT") << " octeti\n";
    return 0;
} //end main

```

5. Teme:

1. Să se scrie un program care afișează valoarea polinomului de gradul 3 pentru o anumită valoare a variabilei reale x.
2. Să se scrie un program care citește lungimile laturilor unui triunghi (folosind variabile întregi) și afișează aria triunghiului ca valoare reală.
3. Să se scrie un program care afișează valorile biților unei variabile de tip *unsigned char* aplicând succesiv operatorul de deplasare dreapta și operatorul %.
4. Să se scrie un program care monitorizează un canal de 16/32/64 biți. Pentru aceasta citiți de la tastatură o valoare întregă fără semn a care va fi afișată în zecimal, binar, octal și hexazecimal. Folosiți o funcție pentru conversia numerelor din baza 10 în baza 2. Implementați o altă funcție numită *getsets()* care primește trei valori ca parametri:

- x: valoarea citită de la tastatură
- p: poziția unui bit din cei 16/32 sau 64 de biți (numărând de la dreapta)
- n: numărul de biți care vor fi extrași din tabel.

Funcția returnează, aranjați spre dreapta, acei n biți ai valorii x, pornind de la poziția p, unde $p < 8 * \text{sizeof}(x)$ și $p > n$. Afișați rezultatul în binar, octal și hexazecimal.

5. Să se scrie un program care citește de la intrarea standard un număr întreg și afișează numărul de zerouri din reprezentarea sa binară.
6. Se citește de la intrarea standard o valoare întreagă. Să se afișeze în format zecimal valoarea fiecărui octet al întregului citit.
7. Se citesc de la tastatură 2 numere reale. Să se realizeze operațiile de adunare, scădere, înmulțire și împărțire cu valorile date. Să se afișeze rezultatele obținute, apoi să se rotunjească valorile obținute la valori întregi, folosind operatorul cast și fără a folosi funcții specifice. Să se afișeze apoi valoarea minimă dintre numerele citite folosind operatorul condițional.
8. Citiți de la tastatură mai multe caractere reprezentând litere mici. Să se transforme caracterele citite în litere mari în 2 moduri: printr-o operație aritmetică, respectiv folosind o operație logică și o mască adecvată.
9. Citiți de la tastatura 2 numere întregi. Dacă sunt egale, determinați aria cercului cu raza egală cu valoarea citită, iar dacă sunt diferite calculați aria dreptunghiului cu laturile egale cu valorile date. Afișați aria calculată specificând forma geometrică pentru care s-a făcut calculul.

5'. Homework:

1. Write a program that displays the value of a 3-rd degree polynom, calculated in a certain point x.
2. Write a program that reads from the keyboard 3 values representing the lengths of a triangle's sides and then displays the triangle's area.
3. Write a program that reads an *unsigned char* value and displays the binary values resulting after recursively applying the shift right and the % operators.
4. Write a program that monitors a communications channel on 16/32/64 bits. In order to do that, read from the keyboard an unsigned int value a, that will be displayed in decimal, binary, octal and hexadecimal counting bases. Use a function for converting the number from base 10 in base 2.

Implement another function called *getsets()* that receives 3 parameters:

- x: the value read from the keyboard
- p: the position of a bit of those 16/32 or 64 bits (counting from the right).
- n: the number of bits that will be extracted from the array.

The function returns, adjusted to the right, those n bits from the value x, starting with the position p, where $p < 8 * \text{sizeof}(x)$ and $p > n$. Display the result in binary, octal and hexadecimal

5. Write a program that reads from the standard input an integer number and displays the number of zeroes from its binary representation.

6. Read an integer value from the standard input. Write, in decimal format, the value of each byte of the value.

7. Read 2 float numbers from the keyboard. Calculate and display the results obtained by applying the main arithmetic operations (+, -, *, /) upon them. Cast the results into integer values, without using any specific rounding functions. Display the minimum input value using the conditional operator.

8. Read from the keyboard several lowercase characters. Transform them into uppercase letters using:

- an arithmetic operation;
- a logical operation, combined with a specific mask;

9. Read 2 integer values from the keyboard. If they are equal, determine the area of a circle that has the radius equal with the read value. If the values are not equal, determine the area of the rectangle that has as sides the read values. In both cases, display the name of the geometrical shape that corresponds to the calculated area.