Cursul 11

(*plan de curs*)

0. Exemple de cod ineficient

```
bool toateSuntPozitive5(int a[], int dim){
    int contor = 0;
    for (int i = 0; i < dim; i++){
        if (a[i]>0) contor++;
    }
    return contor == dim;
}

bool toateSuntPozitive10(int a[], int dim){
    for (int i = 0; i < dim; i++){
        if (a[i] <= 0) return false;
    }
    return true;
}
bool areNegativeApoiPozitive5(int a[], int dim){
    //cautam primul contraexemplu cu i<j
    for (int i = 0; i < dim; i++){
        for (int j = i + 1; j < dim; j++){
            if (a[i] > 0 && a[j] < 0) return false;
        }
    }
    return true;
}
bool areNegativeApoiPozitive10(int a[], int dim){
    //cautam primul Pozitiv
    int i;
    for (i = 0; i < dim && a[i] <= 0; i++){}        ;
    //if (i == dim) return true;//nu este necesara!
    //a[i] este pozitiv, cautam primul element negativ dupa el
    for (i++; i < dim; i++) {
        if (a[i] < 0) return false;
    }
    return true;
}
void afisareSirRecurent5(double x0, int n){
    n = (n>1000 ? 1000 : n);
    double x[1001];
    x[0] = x0;
    for (int i = 0; i < n; i++){
        cout << "i=" << i << " x=" << x[i] << endl;
        x[i + 1] = (1 - x[i]) / (1 + x[i] * x[i]);
    }
}
void afisareSirRecurent10(double x, int n){
    for (int i = 0; i < n; i++){
        cout << "i=" << i << " x=" << x << endl;
        x = (1 - x) / (1 + x*x);
    }
}
```

1. typedef

2. enumerări: program ilustrativ.

Să se definescă, în fiecare dintre cazurile următoare, câte o funcţie de forma

```
void afiseaza(int a[], int dim){...}
```

care parcurge tabloul ***a*** o singură dată şi îl afişează pe mai multe rânduri astfel încât

1) să nu existe două elemente consecutive asemenea pe acelaşi rând. Două elemente sunt *asemenea* dacă sunt ambele nule sau dacă au acelaşi semn şi aceeaşi paritate.
2) pe nici un rând să nu existe două elemente asemenea.
3) pe fiecare rând toate elementele nenule de acelaşi semn să aibă aceeaşi paritate.

Se vor forma cât mai puţine rânduri.

Exemplu:

```
Vectorul a:
0 1 -3 50 -2 3 0 0 61 51 0 -70 2 0 3 -2 7 -10

Cazul 1
0 1 -3 50 -2 3 0
0 61
51 0 -70 2 0 3 -2 7 -10

Cazul 2
0 1 -3 50 -2
3 0
0 61
51 0 -70 2
0 3 -2
7 -10

Cazul 3
0 1 -3
50 -2
3 0 0 61 51 0 -70
2 0
3 -2 7 -10
Press any key to continue . . .
```

Rezolvare:

```
#include<iostream>
using namespace std;
bool suntAsemenea(int a, int b){
    if (a == 0 && b == 0) return true;
    return a*b > 0 && (a - b) % 2 == 0;
}
void afiseaza1(int a[], int dim){
    cout << a[0] << " ";
    for (int i = 1; i < dim; i++){
        if (suntAsemenea(a[i - 1], a[i])) cout << endl;
        cout << a[i] << " ";
    }
    cout << endl;
}

enum ParSemn { zero, parPoz, parNeg, impPoz, impNeg };
ParSemn clasifica(int val){
    if (val == 0) return zero;
    return val % 2 == 0 ? (val > 0 ? parPoz : parNeg) :
        (val > 0 ? impPoz : impNeg);
```

```
}
void afiseaza2(int a[], int dim){
    bool exista[5] = {};                //vector martor de paritate & semn
    for (int i = 0; i < dim; i++){
        int val = a[i];
        //decidem pe ce rand afisam val
        ParSemn psVal = clasifica(val);
        if (exista[psVal]) {
            //trecem la linie noua si resetam vectorul martor
            cout << endl;
            for (int j = 0; j < 5; j++) exista[j] = false;
        }
        //afisam si consemnam
        cout << val << " ";
        exista[psVal] = true;
    }
    cout << endl;
}
void afiseaza3(int a[], int dim){
    bool exista[5] = {};                //vector martor de paritate & semn
    for (int i = 0; i < dim; i++){
        int val = a[i];
        //decidem pe ce rand afisam val
        ParSemn psVal = clasifica(val);
        exista[psVal] = true;
        if (exista[parPoz] && exista[impPoz] || exista[parNeg] && exista[impNeg])
        {
            cout << endl;               //trecem la linie noua si
            for (int j = 0; j < 5; j++) //resetam vectorul martor
                exista[j] = false;
            exista[psVal] = true;
        }
        cout << val << ' ';
    }
    cout << endl;
}
int main(void)
{
    int tab[] = { 0, 1, -3, 50, -2, 3, 0, 0,  61, 51, 0, -70, 2, 0, 3, -2, 7, -10 };
    int n = 18;
    cout << "Vectorul a:" << endl;
    for (int i = 0; i < n; i++) cout << tab[i] << " ";
    cout << endl;
    cout << "\nCazul 1" << endl;
    afiseaza1(tab, n);
    cout << "\nCazul 2" << endl;
    afiseaza2(tab, n);
    cout << "\nCazul 3" << endl;
    afiseaza3(tab, n);
    return 0;
}
```

3. structuri: program ilustrativ

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

struct Fractie{//  a/b
    int a = 0, b = 0;
    bool isNaN=true;

};

Fractie initFrac(int a, int b){
    Fractie q;
    if (b == 0) return q;
    q.a = a;
    q.b = b;
    q.isNaN =  false;
    return q;
}

void scrie(Fractie q){
    if (q.isNaN) cout << "Nu este valida: ";
    cout << q.a << " / " << q.b << endl;

}

Fractie catul(Fractie p, Fractie q){
    Fractie rez; // rez=p/q
    if (p.isNaN || q.isNaN || q.a == 0) return rez;
    rez.a = p.a*q.b;
    rez.b = p.b*q.a;
    rez.isNaN = false;
    return rez;
}
int main(){
    Fractie p = initFrac(5, 2), q = initFrac(1, 10);
    scrie(catul(p, q));
    return 0;
}
```

4. uniuni