Principii logice

1) ***Când căutăm ceva, ne oprim la primul găsit bun, dacă există, altfel căutăm până la capăt.***

Căutarea într-o *funcţie de decizie* vs. căutarea *in line*:

```
#include<iostream>
using namespace std;
const int dimMax = 30;

bool esteBun(int a){
    return a % 2 == 0;
}
bool amGasitCeCaut(int tab[dimMax], int n){
    for (int i = 0; i < n; i++){
        if (esteBun(tab[i])) return true;
    }
    return false;
}
int main(){
    int n = 6;
    int tab[dimMax] = { 1, 3, 5, 7, 9, 11 };

    //cautare cu functie de decizie
    cout << (amGasitCeCaut(tab, n) ? "da, " : "nu ") << "am gasit" << endl;

    //cautare "in line"
    bool amGasit = false; // variabila martor
    for (int i = 0; i < n && !amGasit; i++){
        if (esteBun(tab[i])) amGasit = true;
    }
    cout << (amGasit ? "da, " : "nu ") << "am gasit" << endl;
    return 0;
}
```

2) Când avem de stabilit valoarea de adevăr a unei *afirmaţii existenţiale*, căutăm *primul exemplu*:

```
#include<iostream>
using namespace std;
const int dimMax = 30;
//stabilim daca in tab exista macar un element pozitiv

bool arePozitive(int tab[dimMax], int n){
    //cautam primul exemplu
    //cu iesire din functie
    for (int i = 0; i < n; i++){
        if (tab[i]>0) return true;
    }
    return false;
}
```

```
int main(){
    int n = 6;
    int tab[] = { -1, -2, -3, 4, -5, -6 };

    // varianta cu  functie de decizie
    cout << (arePozitive(tab, n) ? "da, " : "nu ") << "are pozitive" << endl;

    // varianta "in line"
    // cautam primul exemplu
    bool amGasitUnPozitiv = false;
    for (int i = 0; i < n && !amGasitUnPozitiv; i++){
        if (tab[i]>0) amGasitUnPozitiv = true;
    }
    cout << (amGasitUnPozitiv ? "da, " : "nu ") << "are pozitive" << endl;
    return 0;
}
```

3) Când avem de stabilit valoarea de adevăr a unei *afirmaţii universale*, căutăm *primul contraexemplu*:

```
#include<iostream>
using namespace std;
const int dimMax = 30;

//stabilim daca toate elementele lui tab sunt pozitive

bool toateSuntPozitive(int tab[dimMax], int n){
    //cautam primul contraexemplu
    //cu iesire din functie
    for (int i = 0; i < n; i++){
        if (tab[i] <= 0) return false;
    }
    return true;
}

int main(){
    int n = 6;
    int tab[dimMax] = { 1, 2, 3, 4, 5, 6 };

    //varianta cu  functie de decizie
    cout << (toateSuntPozitive(tab, n) ? "da, " : "nu ") << "toate sunt  pozitive"
<< endl;

    //varianta "in line"
    //cautam primul contraexemplu
    bool amGasitUnNegativ = false;
    for (int i = 0; i < n && !amGasitUnNegativ; i++){
        if (tab[i] <= 0) amGasitUnNegativ = true;
    }
    cout << (!amGasitUnNegativ ? "da, " : "nu ") << "toate sunt  pozitive" <<
endl;
    return 0;
}
```

Exemple compuse:

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

const int dimMax = 30;

bool toateLiniileAu10(int A[dimMax][dimMax], int n){
    //decide daca in matricea A de tip n x n
    //fiecare linie are macar un element egal cu 10;
    //cautam cu iesire din functie
    //prima linie contra-exemplu (o linie fara nici nu 10)
    for (int i = 0; i < n; i++){
        //stabilim "in line" daca pe linia i exista macar un 10
        //cautam primul exemplu
        bool amGasit10 = false;
        for (int j = 0; j < n && !amGasit10; j++){
            if (A[i][j] == 10) amGasit10 = true;
        }
        if (!amGasit10) return false;
    }
    //nu am gasit nici o linie contra-exemplu
    return true;
}

bool existaLinieDe10(int A[dimMax][dimMax], int n){
    //decide daca in matricea A de tip n x n
    //exista macar o liniile cu toate  elementele egale cu 10
    //cautam cu iesire din functtie
    //prima linie buna
    for (int i = 0; i < n; i++){
        //stabilim "in line" daca pe linia i toate elementele sunt egale cu 10
        //cautam primul contraexemplu
        bool amGasitDiferit10 = false;
        for (int j = 0; j < n && !amGasitDiferit10; j++){
            if (A[i][j] != 10) amGasitDiferit10 = true;
        }
        if (!amGasitDiferit10) return true;
    }
    //nu am gasit nici o linie buna
    return false;
}

int main(){
    int n = 3;
    int A[dimMax][dimMax] = { { 1, 10, 3 }, { 10, 1, 10 }, { 10, 0, 3 } };
    cout << (toateLiniileAu10(A, n) ? "da, " : "nu ") << "pe fiecare linie exista
macar un 10" << endl;
    cout << (existaLinieDe10(A, n) ? "da, " : "nu ") << "exista o linie de 10" <<
endl;
    return 0;
}
```