

1. Algoritmi elementari

A. Tipuri simple de date și structuri de control

```
#include <iostream>
using namespace std;
int main() {
    int a = 10;    // Tip întreg (4 octeți)
    bool ok = true; // Tip logic

    // Structură alternativă (if-else)
    if (a > 5)
        cout << "Mai mare decat 5\n";
    else
        cout << "Mai mic sau egal cu 5\n";

    // Structură repetitivă cu test inițial (while)
    int i = 1;
    while (i <= 3)
    {
        cout << i << " ";
        i++;
    }
    return 0;
}
```

B. Prelucrarea cifrelor unui număr

```
// Inversarea cifrelor unui număr n
int n = 123, invers = 0;
while (n > 0) {
    invers = invers * 10 + n % 10;
    n /= 10;
}
```

C. Divizibilitate și CMMDC (algoritmul lui Euclid)

```
// CMMDC prin împărțiri repetate
int a = 24, b = 36;
while (b != 0)
{
    int r = a % b;
    a = b;
    b = r;
}
cout<<"CMMDC = "<<a;
// a memorează CMMDC-ul (12)
```

D. Expresii: Factorial unui număr n (ex: dacă $n=4$ atunci $4! = 1 * 2 * 3 * 4$)

```
int n = 5, fact = 1;
for (int i = 1; i <= n; i++) {
    fact = fact * i; // 5! = 120
}
```

2. Generări de șiruri (Șirul lui Fibonacci)

// Generarea primilor **n** termeni din șirul lui Fibonacci

```
int n = 6;
int a = 0, b = 1;
cout << a << " " << b << " ";
for (int i = 3; i <= n; i++) {
    int c = a + b;
    cout << c << " ";
    a = b;
    b = c;
}
// Afîșează: 0 1 1 2 3 5
```

3. Fișiere text

```
#include <fstream>
using namespace std;
ifstream f ("date.in");
ofstream g("date.out");
int main() {
    int x;
    f >> x; //citesc un element din f si il salvez in x (date.in)
    g << x; //scriu valoarea lui x in g (date.out)

    f.close();
    g.close();
}
```

4. A. Determinarea maximului dintr-un șir de numere fără a memora tot șirul

```
int n; // numărul de elemente
cin >> n;
int maxim = -1000000;
for (int i = 0; i < n; i++)
{
    int x;
    cin >> x;
    if (x > maxim)
        maxim = x; // stocăm elementul maxim curent
} //for
cout << maxim;
```

4.B.1 Se citește un șir format de n numere naturale. Să se afișeze de câte ori apar două numere egale consecutive (pe poziții alăturate).

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int a, b;
    int nr_perechi = 0;
    if (n > 0) {
        cin >> a; // citim primul element
        for (int i = 2; i <= n; i++) {
            cin >> b; // citim elementul curent (ultimele 2 elemente sunt 'a' și 'b')
            if (a == b) {
                nr_perechi++;
            }
            a = b; // Elementul curent devine precedentul pentru pasul următor
        }
    }
    cout << "Numar perechi egale: " << nr_perechi << "\n";
    return 0;
}
```

4.B.2. Se citește un șir format din n numere. Să se determine lungimea celei mai lungi subsecvențe în care fiecare element este strict mai mare decât cel din stânga lui.

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int a, b;
    int lungime_curenta = 1, lungime_maxima = 1;

    if (n > 0) {
        cin >> a;
        for (int i = 2; i <= n; i++)
        {
            cin >> b;
            if (b > a)
                lungime_curenta++; // Continuă creșterea
            else
                lungime_curenta = 1; // Se resetează subsecvența

            if (lungime_curenta > lungime_maxima) {
                lungime_maxima = lungime_curenta;
            }
        }
    }
}
```

```

        a = b; // Trecem la pasul următor
    }
}
cout << "Lungimea maxima: " << lungime_maxima << "\n";
return 0;
}

```

4.B.3. Se citește un șir format din n numere întregi (nenule). Să se numere de câte ori un număr pozitiv este urmat imediat de unul negativ sau invers (schimbări de semn între elemente alăturate).

```

#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    int a, b;
    int schimbari = 0;

    if (n > 0) {
        cin >> a;
        for (int i = 2; i <= n; i++) {
            cin >> b;
            // Verificăm dacă produsul dintre elementul precedent și cel curent este negativ
            if ((a > 0 && b < 0) || (a < 0 && b > 0)) {
                schimbari++;
            }
            a = b;
        }
    }
    cout << "Numar schimbari de semn: " << schimbari << "\n";
    return 0;
}

```

Aplicații:

1.) Se citește un număr natural din fișierul *numere.in*. Determinați și scrieți în fișierul *numere.out* cel mai mare număr care se poate scrie cu cifra unităților și cifra zecilor numărului inițial.

Ex: dacă *numere.in* conține **4367** atunci fișierul *numere.out* va conține numărul **76**

2.) Se citește un număr natural **n** din fișierul *file.in* de pe prima linie. De pe a doua linie se citește un număr natural **c** (care urmează să fie căutat). Pe a treia linie a fișierului se află **n** numere naturale separate între ele cu spațiu. Să se scrie în *file.out* pe a câta poziție se află **c** între cele **n** numere. Dacă **c** nu se găsește printre cele **n** numere atunci se va scrie în fișier valoarea -1.

Ex: dacă *file.in* are conținutul:

```

6
35
2 17 82 35 437 8765

```

atunci *file.out* va conține 4.