**Metoda bulelor** presupune parcurgerea iterativă a tabloului şi, la fiecare parcurgere, ori de câte ori se întâlnesc două elemente consecutive care nu se află în ordinea care trebuie, valorile lor se interschimbă. La prima parcurgere elementul cu valoarea maximă/minimă (în funcţie de cum dorim să sortăm şirul, crescător/descrescător) va ajunge pe ultima poziţie din şir, acolo unde îi şi este locul final. La următoarea parcurgere al doilea element maxim/minim va ajunge pe penultima poziţie, şi aşa mai departe. La fel ca şi bulele de aer dintr-un pahar cu apă, valorile mari/mici vor "urca" la capătul şirului.

**Metoda inserţiei** este asemanătoare cu metoda bulelor; comparăm două valori consecutive şi dacă nu se află în ordinea care trebuie, le interschimbăm. După interschimbare deplasăm la stânga elementul actual până găsim un predecesor mai mic/mare (în funcţie de cum dorim să sortăm şirul, crescător/descrescător) sau până ajungem la începutul şirului.

**Metoda selecţiei** presupune aflarea poziţiei pe care se află cel mai mic/mare element din şir (în funcţie de cum dorim să sortăm şirul, crescător/descrescător) şi interschimbăm primul element din şir cu acesta. În continuare căutam următorul cel mai mic/mare element, îl interschimbăm cu elemntul aflat pe a doua poziţie din şir, şi aşa mai departe.

**Metoda numărarii** presupune parcurgerea şirului şi determinarea numărului de elemente mai mici/mari (în funcţie de cum dorim să sortăm şirul, crescător/descrescător) decât elementul curent. Într-un şir auxiliar se memorează elementul actual pe poziţia corespunzătoare cu numărul de elemente mai mici/mari decât acesta. La sfârşit în şirul auxiliar se vor afla elementele sortate; le vom copia înapoi în şirul iniţial. (Obs: această metodă presupune că şirul este format din valori distincte)

Algoritmul de sortare prin **interclasare** presupune utilzarea metodei *Divide et Impera*. Şirul de numere ce trebuie sortat este împărţit în două subşiruri; acest pas se repetă până când toate subşirurile sunt formate dintr-un singur element (astfel putem spune ca aceste subşiruri sunt gata ordonate). După ce s-a făcut această diviziune se trece la procesul de interclasare al subşirurilor.

Metoda **Divide et Impera** este folosită pentru rezolvarea problemelor complexe ce pot fi împărţite în subprobleme mai uşoare. După ce o problemă a fost împărţită în mai multe subprobleme, se rezolvă fiecare dintre acestea(în general recursiv), iar în final soluţile parţiale sunt combinate într-o soluţie finală.

Un **subalgoritm recursiv** este un subalgoritm ce se apelează pe el însuşi, ceea ce permite implementarea unei funcţii matematice recursive. La intrarea într-un subalgoritm recursiv se alocă spaţiu pe stivă pentru parametri transmişi şi pentru variabilele locale ce au fost declarate în subalgoritm. La ieşirea din apel stiva se eliberează de informaţile depuse la intrarea în apel.

Metoda **Backtracking** poate fi folosită la rezolvarea problemelor a căror soluţie este de forma unui vector ($x_1$, $x_2$, ... $x_n$). Pentru fiecare $x_i$ se cunoaşte mulţimea în care poate să ia valori($x_i \in M_i$), iar elementele soluţiei trebuie să verifice anumite condiţii.

Metoda **Greedy** este o metodă de rezolvare a problemelor, ce construieşte soluţia componentă cu componentă. La fiecare pas se alege cel mai bun candidat posibil, după evaluarea tuturor acestora, ce îndeplineşte toate condiţile date. Metoda determină o singură soluţie, asigurând un optim local, dar nu întotdeauna şi global.

**Programarea dinamică** este o metodă rezolvare a problemelor în care se cere determinarea unuei soluţi optime referitor la un anumit criteriu. O problemă este descompusă în subprobleme de dimensiuni mai mici şi soluţia optimă a problemei depinde de soluţiile optime ale subproblemelor sale.

**Identificatorii** reprezintă nume de constante, variabile, funcţii, etc. În denumire pot fi folosite numere, cifre şi caracterul underscore, dar primul caracter nu poate fi un număr. *Obs:* C++ este case-sensitive.

Un **tip de data** reprezintă mulţimea valorilor pe care le pot lua datele de tipul respectiv, modul de repartizare a acestora în memorie, precum şi operaţiile care se pot efectua cu datele respective. (ex de tipuri de date: int, double, char, bool)

**Variabila** este un mod de a face referire la o locaţie de memorie folosită de către program pentru a memora o valoare. Pe parcursul execuţiei programului, valoarea înmagazinată de către o variabila poate fi modificată.

**Variabila globală** este o variabilă ce a fost declarată în afara tuturor funcţiilor. Ea este iniţializata cu valoarea 0, este vizibilă şi poate fi modificată în orice subprogram al programului.

**Variabile locală** este o variabilă ce a fost declarată în interiorul unei funcţii sau al unui bloc de instrucţiuni. Ea este vizibilă şi poate fi modificată doar pe plan local.

Procesul de **declararea al unei variabile** constituie specificarea numelui acesteia, tipul ei şi, opţional, o valoare iniţială.

**Constanta** reţine o valoare ce nu se poate modifica pe parcursul execuţiei programului.

O **expresie** este alcatuită din unul sau mai mulţi operanzi legaţi între ei prin operatori.
**operanzi** = constante, variabile sau o expresie între paranteze rotunde
**operatori** = reprezintă operaţiile care se execută asupra operanzilor.
**o.** aritmetici (+, -, *, /, %), **o.** logici(!, &&, ||), **o.** relaţionali(<, >, <=, >=, ==, !=)

Un **tablou** este o colecţie de date de acelaşi tip, situate într-o zona de memorie continuă (două elemente consecutive dintr-un tablou au adrese de memorie succesive).

Un **şir de caractere** este o succesiune de caractere terminată cu caracterul NULL. Caracterul NULL este caracterul ce are codul ASCII 0.

**Structura de date** reprezintă un ansamblu de date simple (int, double, char) organizate după anumite reguli care depind de tipul de structură (ex de SD: lista, coada stiva, arbori binari)

**Structurile de control** reprezintă componentele programarii structurate.

1. Structura liniară (a = 5; b = a+3; tipareste a)
2. Structura alternativă (if...else, swithc)
3. Structura repetitivă
   3.1. cu test iniţial (Cat timp expresie executa instructiuni) se executa *cat timp* valoarea expresiei este diferita de **0**.
   3.2. cu test final(repeta instructiuni pana cand expresie) se executa *pana cand* valoarea expresiei **va deveni 1**.
   3.3. cu nr cunoscut de paşi(pentru initializare, conditie, pas executa instructiuni)

**Tipul înregistrat** reprezintă o colecţie de un număr fix de componente care pot fi tipuri diferite (caracter neomogen), ce constituie o unitate de prelucrare.

**Algoritmul** este o noţiune primară fără definiţie, însa intuitiv putem spune că un algoritm este un ansamblu de instrucţiuni corelate, care au ca scop rezolvarea unei probleme.

***Caracteristicile unui algoritm***

- **Generalitatea:** trebuie să rezolve o clasă întreagă de probleme de acelaşi tip
- **Finititudinea:** se termină după un număr finit de paşi
- **Claritatea:** trebuie descris clar, fără ambiguităţi
- **Eficienţa:** timpul de execuţie şi spaţiul de memorie utilizat trebuie să fie optime

Un **subprogram** este un ansamblu de instrucţiuni corelate ce rezolvă o anumită sarcină. Subprogramul poate fi executat doar dacă este apelat de către un program sau alt subprogram.

(*declarare, apel, definiţie*)

**Parametri formali** sunt scrişi în *declararea şi antetul* funcţiei (subprogramului). Parametri formali sunt variabile locale ce sunt iniţializate cu valorile argumentelor cu care este apelat subprogramul.

**Parametri actuali** reprezintă valorile cu care este apelat un subprogram. Între parametri actuali şi parametri formali există o relaţie de corespondenţă.

**Parametri transmişi prin valoare =** atunci când parametri formali primesc o copie a parametrilor actuali. La întoarcerea în program, variabilele cu care a fost apelată funcţia rămân nemodificate.

**Parametri transmişi prin referinţă** = atunci când variabielele parametrilor formali indica catre aceeaşi zonă de memorie ca şi variabilele parametrilor actuali. Acele variabile nu sunt interpretate ca şi poinetri, ci ca sinonime ale variabilelor parametrilor actuali. La întoarcerea în program, variabilele cu care a fost apelată funcţia vor fi modificate de către subprogram.

**Stiva** este o structură de date abstractă pentru care operaţia de inserare şi operaţia de extragere a unui element din structură se realizează la un singur capăt, denumit vârful stivei. Stiva funcţionează după principiul LIFO.

**Coada** este o structură de date abstractă pentru care operaţia de inserare a unui element se realizează la un capăt, în timp ce operaţia de extragere se realizează la celălalt capăt. Coada funcţionează după principiul FIFO.

O **listă** este o colecție de elemente(noduri), legate între ele prin referință, realizându-se astfel o stocare necontinuă a datelor în memorie. Lungimea unei liste este dată de numărul de noduri din listă. Ea permită determinarea atât a primului, ultimului nod din structură cât și care este predecesorul sau succesorul unui nod dat.

Un **arbore binar** este un arbore în care orice nod are 0, 1 sau 2 descendenţi.

**Variabila dinamică** este o variabilă creată şi, opţional, dealocată în timpul execuţiei programului.

**Variabila statică** este o variabilă creată și, opţional, inițializată, înainte ca funcția main să fie executată și nu este dealocată decât când se termină programul.