

**Universitatea Tehnică de Construcții din București**

**Facultatea de Hidrotehnică**

**Specializarea: Automatică și Informatică Aplicată**

**Proiect la LP2**

**Profesori coordonatori: Olteanu Gabriela, Neculoiu Georgian Studenti: Preda Paul Andrei, Dobre Sarrah Ștefania**

**București, 2025**



**Universitatea Tehnică de Construcții din București**

**Facultatea de Hidrotehnică**

**Specializarea: Automatică și Informatică Aplicată**

**Snake**

**Profesori coordonatori: Olteanu Gabriela, Neculoiu Georgian Studenti: Preda Paul Andrei, Dobre Sarrah Ștefania**

**București, 2025**

### ****CUPRINS****

**INTRODUCERE** ...................................................................................................7

Motivarea alegerii temei ..........................................................................................7

Obiectivele propuse în cadrul lucrării .....................................................................7

Structura lucrării .....................................................................................................8

**CAPITOLUL 1 – Conceptul aplicației „Snake” și scopul acesteia** ................. 8  
1.1 Conceptul aplicației „Snake” ............................................................................8  
1.2 Introducerea în joc ............................................................................................9  
1.3 Regulile jocului ................................................................................................9

**CAPITOLUL 2 – Aspecte teoretice ale aplicației** .............................................10  
2.1 Algoritmul jocului ...........................................................................................10  
2.2 Tehnologii utilizate ..........................................................................................11  
2.3 Funcționarea limbajului C++ în contextul aplicației .......................................13

**CAPITOLUL 3 – Structura jocului** ...................................................................13  
3.1 Componentele principale ale jocului ...............................................................13  
3.2 Fluxul principal al jocului ................................................................................14  
3.3 Descrierea detaliată a funcțiilor .......................................................................15  
3.4 Observații privind optimizarea jocului ............................................................16

**CAPITOLUL 4 – Concluzii și contribuții personale** ........................................16  
4.1 Concluzii ..........................................................................................................16  
4.2 Contribuții personale ........................................................................................17

**CAPITOLUL 5 – Perspective de viitor** .............................................................17  
5.1 Funcționalități suplimentare ............................................................................18  
5.2 Îmbunătățiri grafice și de interfață ..................................................................18  
5.3 Abordări avansate și extindere profesională ...................................................19

**BIBLIOGRAFIE** ................................................................................................20

**ANEXE** ................................................................................................................20

Anexa 1 – Codul sursă complet al jocului Snake .................................................20

Anexa 2 – Capturi de ecran ale aplicației în execuție ..........................................23

# ****INTRODUCERE****

## ****Motivarea alegerii temei****

Tema aleasă pentru acest proiect, dezvoltarea jocului **„Snake”**, este una cu o puternică încărcătură didactică și practică. Alegerea acestei teme nu este întâmplătoare, ci reflectă o decizie conștientă de a îmbina noțiunile fundamentale ale programării cu un scop aplicativ clar și ușor de înțeles. Jocul „Snake” este cunoscut și îndrăgit de generații întregi, reprezentând un exemplu clasic de logică simplă, dar provocatoare.

Pe plan educațional, implementarea acestui joc contribuie la consolidarea cunoștințelor de programare în limbajul **C++**, oferind o oportunitate concretă de a pune în aplicare noțiuni precum:

* lucrul cu structuri de date dinamice (vectori);
* controlul fluxului prin bucle și condiții;
* manipularea inputului în timp real;
* afișarea în consolă cu caractere ASCII.

Totodată, realizarea acestui proiect presupune o bună planificare, o structurare clară a codului și o abordare modulară, aspecte esențiale în dezvoltarea profesională a unui programator.

Am ales acest joc datorită caracterului său interactiv și intuitiv, care facilitează testarea și evaluarea performanței în orice etapă a dezvoltării. Având reguli simple și o logică de joc ușor de urmărit, „Snake” permite concentrarea pe implementare și optimizare fără a complica inutil procesul.

**În concluzie**, alegerea acestei teme îmbină utilul cu plăcutul: oferă provocări tehnice potrivite nivelului de pregătire, contribuie la dezvoltarea abilităților practice în C++ și permite, în același timp, crearea unui produs final vizual, testabil și cu potențial real de extindere.

## ****Obiectivele propuse în cadrul lucrării****

Scopul principal al lucrării este **dezvoltarea unei aplicații funcționale și interactive în C++**, care să simuleze un joc de tip „Snake”, respectând principiile de bază ale dezvoltării software.

**Obiectivele specifice urmărite sunt:**

1. **Modelarea logicii de joc:**
   * Stabilirea regulilor de funcționare (deplasare, coliziuni, creșterea cozii);
   * Implementarea unui mecanism de generare aleatorie a fructelor.
2. **Gestionarea interacțiunii cu utilizatorul:**
   * Controlul direcției prin tastele W, A, S, D;
   * Utilizarea funcțiilor \_kbhit() și \_getch() pentru input în timp real.
3. **Afișarea grafică a jocului în consolă:**
   * Reprezentare vizuală cu caractere ASCII pentru pereți, șarpe și fruct;
   * Reîmprospătare constantă a consolei pentru a simula mișcarea.
4. **Gestionarea dinamică a cozii șarpelui:**
   * Utilizarea unui vector de coordonate pentru memorarea cozii;
   * Actualizarea pozițiilor în fiecare iterație a jocului.
5. **Calcularea și afișarea scorului:**
   * Creșterea scorului la fiecare fruct colectat;
   * Încheierea jocului în caz de coliziune cu marginea sau coada.
6. **Structurarea modulară a aplicației:**
   * Organizarea codului în funcții independente: Setup(), Draw(), Input(), Logic();
   * Cod clar, reutilizabil și ușor de extins/testat.

## ****Structura lucrării****

Pentru o prezentare coerentă și clară, lucrarea este împărțită în **cinci capitole**, completate de introducere, concluzii, contribuții personale, perspective de viitor și bibliografie.

* **Capitolul 1 – Conceptul aplicației „Snake” și scopul acesteia:**  
  Prezentarea generală a ideii jocului, obiectivele jucătorului, regulile și scopul educațional.
* **Capitolul 2 – Aspecte teoretice ale aplicației:**  
  Detalii despre algoritmul jocului, tehnologiile folosite și rolul limbajului C++.
* **Capitolul 3 – Structura jocului:**  
  Prezentarea codului sursă, a variabilelor, funcțiilor și a fluxului logic al aplicației.
* **Capitolul 4 – Concluzii și contribuții personale:**  
  Rezultatele obținute, ce s-a învățat și aportul propriu adus în realizarea jocului.
* **Capitolul 5 – Perspective de viitor:**  
  Propuneri de îmbunătățiri, extindere și transformarea proiectului într-o aplicație mai complexă.
* **Bibliografie** – enumeră sursele utilizate în realizarea lucrării.
* **Anexe** – includ materiale suplimentare relevante pentru proiect:

**Anexa 1** – Codul sursă complet al aplicației;

**Anexa 2** – Capturi de ecran cu aplicația în timpul execuției;

# ****CAPITOLUL 1 – Conceptul aplicației „Snake” și scopul acesteia****

## ****1.1 Conceptul aplicației „Snake”****

Aplicația „Snake” este o reinterpretare digitală a unui joc clasic, cu un concept simplu și captivant: un șarpe controlat de jucător se deplasează pe o hartă de dimensiuni fixe, colectând fructe pentru a-și crește lungimea. Scopul principal este evitarea coliziunilor cu pereții sau cu propriul corp, acumulând în același timp un scor cât mai mare.

În această implementare, jocul este dezvoltat în limbajul **C++**, utilizând doar interfața de **consolă**, fără elemente grafice externe. Astfel, jocul este ideal pentru consolidarea cunoștințelor de bază în programare.

Reprezentarea vizuală se face cu **caractere ASCII**:

* O – capul șarpelui;
* o – segmentele cozii;
* F – fructul;
* # – pereții hărții.

Această abordare minimalistă scoate în evidență logica jocului și oferă o bază didactică excelentă pentru începători. Fiind lipsit de grafică avansată, accentul este pus pe algoritmică, structură și interacțiune.

## ****1.2 Introducerea în joc****

La lansarea aplicației, șarpele este poziționat în centrul hărții, iar un fruct este generat aleatoriu în interiorul acesteia. Utilizatorul poate începe jocul imediat, controlând șarpele cu ajutorul tastelor:

* W – deplasare în sus;
* A – deplasare la stânga;
* S – deplasare în jos;
* D – deplasare la dreapta.

Pe măsură ce șarpele consumă fructe:

* lungimea cozii crește;
* scorul crește;
* spațiul de manevră devine tot mai restrâns;
* riscul coliziunilor cu propria coadă crește.

Jocul se încheie în următoarele cazuri:

* șarpele iese din hartă (coliziune cu marginea);
* șarpele se lovește de propriul corp;
* utilizatorul apasă X pentru a ieși voluntar.

Această introducere intuitivă face ca aplicația să fie ușor de înțeles și accesibilă pentru orice utilizator.

## ****1.3 Regulile jocului****

Regulile jocului sunt simple, dar creează un gameplay dinamic și captivant:

### ****Scopul jocului:****

* Colectarea fructelor (F) pentru a obține un **scor cât mai mare**;
* Evitarea coliziunilor care duc la **Game Over**.

### ****Mișcarea și controlul:****

* W – sus;
* A – stânga;
* S – jos;
* D – dreapta.

Direcția este preluată **în timp real**, fără a necesita confirmarea prin tasta Enter.

### ****Creșterea cozii:****

* La fiecare fruct colectat, șarpele se lungește cu **un segment suplimentar**;
* Coada este gestionată printr-un vector de coordonate.

### ****Coliziuni și sfârșitul jocului:****

* Jocul se termină dacă:
  + șarpele se lovește de **pereții hărții** (#);
  + șarpele se lovește de **propriul corp** (o);
  + utilizatorul apasă X.

### ****Scor:****

* Se adaugă **10 puncte** pentru fiecare fruct consumat;
* Scorul este afișat constant în colțul din stânga jos al consolei.

### ****Reprezentare vizuală:****

| **Simbol** | **Element** |
| --- | --- |
| # | Marginea hărții |
| O | Capul șarpelui |
| o | Coada șarpelui |
| F | Fructul de colectat |

# ****CAPITOLUL 2 – Aspecte teoretice ale aplicației****

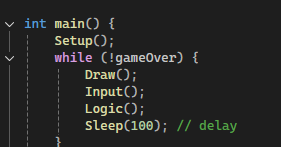
## ****2.1 Algoritmul jocului****

Algoritmul jocului „Snake” se bazează pe o **buclă principală** de tip while, care se execută în mod repetitiv, gestionând întreaga logică și afișare a jocului. Această structură este caracteristică jocurilor **în timp real**, unde fiecare iterație corespunde unui „cadru” sau „pas” din rularea aplicației.

### ****Etapele algoritmului sunt:****

1. **Inițializare (Setup())**
   * Se setează poziția inițială a șarpelui în centrul hărții.
   * Se generează aleator o poziție pentru fruct.
   * Se resetează scorul și coada șarpelui.
2. **Afișare (Draw())**
   * Se curăță consola cu system("cls").
   * Se afișează pereții hărții (#), capul șarpelui (O), coada (o) și fructul (F).
   * Se afișează scorul curent.
3. **Citirea inputului (Input())**
   * Se verifică apăsarea tastelor fără întreruperea buclei principale (cu kbhit() și getch()).
   * Se actualizează direcția de deplasare în funcție de tasta apăsată.
4. **Aplicarea logicii (Logic())**
   * Se mută capul șarpelui în direcția aleasă.
   * Se actualizează vectorul de coadă pentru a reflecta mișcarea.
   * Se verifică:
     + Coliziunea cu marginea hărții;
     + Coliziunea cu propria coadă;
     + Colectarea unui fruct (scorul crește, coada se prelungește, fruct nou este generat).
5. **Întârziere (Sleep(100))**
   * Se introduce o pauză de 100 ms pentru controlul vitezei de joc.

### ****Structura buclei principale:****



Această structură simplă, dar eficientă, permite actualizarea rapidă a jocului și poate fi extinsă cu funcții suplimentare precum niveluri, obstacole sau efecte vizuale.

### ****2.2 Tehnologii utilizate****

Pentru dezvoltarea aplicației „Snake”, a fost utilizat limbajul de programare **C++**, împreună cu biblioteci standard și specifice platformei Windows. Alegerea acestui limbaj se justifică atât din punct de vedere educațional, cât și tehnic, datorită caracteristicilor sale fundamentale.

#### ****Limbajul C++ – fundamente și avantaje****

C++ este un limbaj de programare de nivel mediu, care îmbină puterea limbajului de asamblare cu ușurința de utilizare a limbajelor de nivel înalt. Este un limbaj **compilat**, ceea ce înseamnă că programele scrise în C++ sunt traduse direct în cod mașină, asigurând astfel o viteză ridicată de execuție. Acest aspect este esențial în aplicații care necesită timp de răspuns rapid, cum este cazul jocurilor.

C++ oferă o **gestionare precisă a memoriei**, prin variabile, pointeri, vectori și structuri, permițând dezvoltatorului să controleze eficient resursele aplicației. Este, de asemenea, un limbaj **multi-paradigmă**, suportând programarea procedurală (utilizată în acest proiect), dar și programarea orientată pe obiect (OOP) și generică (prin șabloane/templates).

În cadrul acestui proiect, C++ a fost utilizat pentru:

* **Organizarea modulară a codului** prin funcții bine definite (ex. Setup(), Draw(), Input(), Logic());
* **Manipularea vectorilor** pentru gestionarea dinamică a cozii șarpelui;
* **Afișarea logică în consolă** prin caractere ASCII;
* **Prelucrarea inputului în timp real** folosind funcții din biblioteci suplimentare (\_kbhit(), \_getch()).

#### ****Biblioteci utilizate****

Pentru funcționalitățile suplimentare necesare în joc, au fost utilizate următoarele biblioteci:

* #include <iostream> – bibliotecă standard pentru operații de intrare/ieșire, esențială pentru afișarea elementelor în consolă și pentru interacțiunea cu utilizatorul;
* #include <conio.h> – bibliotecă specifică sistemului Windows, care permite citirea tastelor fără a fi nevoie de apăsarea Enter (input asincron), prin funcțiile \_kbhit() și \_getch();
* #include <windows.h> – oferă funcții utile pentru controlul fluxului aplicației, cum ar fi Sleep(), care introduce întârzieri între cadrele jocului;
* #include <vector> – parte din biblioteca STL (Standard Template Library), permite gestionarea unei liste dinamice de coordonate pentru coada șarpelui.

#### ****Caracteristici relevante ale C++ în contextul jocului****

* **Tipuri de date**: în joc sunt utilizate tipuri fundamentale (int, bool), dar și tipuri compuse precum vectori de perechi (x, y), care ajută la stocarea pozițiilor segmentelor cozii.
* **Structuri de control**: jocul se bazează pe structuri precum while, if, switch, care controlează logica de joc, mișcarea șarpelui și verificarea coliziunilor.
* **Enumerări (enum)**: pentru reprezentarea direcției șarpelui s-a folosit enum eDirection { STOP, LEFT, RIGHT, UP, DOWN }, o soluție elegantă și clară pentru controlul stărilor.

#### ****Relevanța didactică a utilizării C++****

Utilizarea C++ în realizarea acestui joc nu a fost doar o alegere tehnică, ci și una cu valoare pedagogică. Prin scrierea jocului de la zero în consolă, fără motoare grafice sau frameworkuri externe, autorii au învățat:

* Să aplice concepte de bază din programare într-un proiect concret;
* Să înțeleagă modul în care logica, inputul și afișarea se îmbină într-un joc interactiv;
* Să scrie cod clar, modular și eficient, organizat în funcții și controlat prin variabile globale bine definite.

## ****2.3 Funcționarea limbajului C++ în contextul aplicației****

Limbajul **C++** este ideal pentru dezvoltarea unei aplicații precum Snake datorită multiplelor avantaje pe care le oferă:

### ****1. Controlul execuției****

C++ permite un control total asupra fluxului de execuție: se pot seta manual timpi de așteptare, se poate reacționa instant la input, iar memoria este gestionată eficient (vectori, alocare dinamică etc.).

### ****2. Viteză și performanță****

Fiind un limbaj compilat, C++ oferă o execuție rapidă, esențială pentru aplicațiile interactive. Jocul Snake rulează fluent chiar și pe sisteme cu resurse reduse.

### ****3. Modularitate și claritate****

Proiectul este organizat în funcții distincte (Setup, Draw, Input, Logic), ceea ce facilitează înțelegerea și extinderea codului. Această separare a responsabilităților este o bună practică în programare.

### ****4. Extensibilitate****

C++ permite cu ușurință extinderea aplicației:

* Se pot adăuga clase și obiecte (în OOP);
* Se pot integra biblioteci grafice (ex: SFML, SDL);
* Se poate compila codul și pentru alte platforme (Linux, macOS), cu mici ajustări.

**Concluzie:**  
Capitolul teoretic demonstrează că jocul „Snake”, deși simplu ca idee, este un exemplu excelent pentru aplicarea concretă a tehnicilor de bază în programare, în special în C++. Utilizarea eficientă a buclelor, vectorilor, inputului asincron și afișării în consolă transformă acest joc într-un proiect educațional complet și valoros.

# ****CAPITOLUL 3 – Structura jocului****

## ****3.1 Componentele principale ale jocului****

Aplicația „Snake” este structurată modular, fiecare componentă având un rol bine definit în logica și funcționarea jocului. Organizarea clară a codului facilitează înțelegerea, depanarea și eventualele extinderi.

### ****a) Variabile globale****

Acestea sunt utilizate pentru a reține starea generală a jocului:

* bool gameOver – determină dacă jocul este activ sau s-a încheiat;
* const int width, height – dimensiunile hărții de joc (ex. 20x20);
* int x, y – poziția curentă a capului șarpelui;
* int fruitX, fruitY – coordonatele fructului de pe hartă;
* int score – scorul acumulat de jucător;
* int tailLength – lungimea cozii șarpelui;
* vector<pair<int, int>> tail – vector de perechi ce reține pozițiile fiecărui segment din coada șarpelui;
* enum eDirection { STOP, LEFT, RIGHT, UP, DOWN } dir – direcția de deplasare a șarpelui.

### ****b) Funcțiile jocului****

Fiecare funcție are un scop precis:

* Setup() – inițializează toate variabilele: poziții, scor, direcție;
* Draw() – desenează în consolă harta, șarpele, fructul și scorul;
* Input() – gestionează comenzile tastaturii pentru controlul direcției;
* Logic() – actualizează poziția șarpelui, detectează coliziuni și fructe consumate.

Această abordare modulară asigură claritate, eficiență și posibilitate de testare individuală a fiecărei componente.

## ****3.2 Fluxul principal al jocului****

Funcția main() este punctul de intrare al aplicației și conține bucla principală a jocului. Aceasta este responsabilă cu rularea continuă a pașilor fundamentali până la terminarea jocului.

### ****Structura buclei principale:****

cpp

CopyEdit

int main() {

Setup();

while (!gameOver) {

Draw();

Input();

Logic();

Sleep(100);

}

cout << "Game Over!" << endl;

return 0;

}

### ****Etapele buclei:****

1. **Inițializare (Setup())**  
   Se setează valorile inițiale: șarpele apare în centru, fructul este poziționat aleator, direcția este STOP, scorul este 0.
2. **Afișare (Draw())**  
   Se redesenează întreaga hartă la fiecare iterație, incluzând pereții, șarpele, fructul și scorul.
3. **Preluarea inputului (Input())**  
   Se verifică dacă jucătorul apasă o tastă și se actualizează direcția șarpelui.
4. **Aplicarea logicii (Logic())**  
   Se actualizează pozițiile, se detectează coliziuni, se mărește coada dacă s-a colectat un fruct.
5. **Întârziere (Sleep(100))**  
   Se introduce o pauză între iterații pentru a controla viteza jocului.

Această buclă este nucleul jocului și se repetă până când gameOver devine true.

## ****3.3 Descrierea detaliată a funcțiilor****

### Setup()

* Inițializează variabilele globale;
* Plasează șarpele în centrul hărții;
* Generează o poziție aleatoare pentru fruct;
* Resetează scorul și coada;
* Direcția este setată pe STOP.

### Draw()

* Curăță consola cu system("cls");
* Desenează marginea hărții cu #;
* Plasează șarpele (O pentru cap, o pentru coadă);
* Plasează fructul (F);
* Afișează scorul în partea de jos a consolei.

### Input()

* Verifică dacă a fost apăsată o tastă;
* Folosește kbhit() și getch() pentru input fără enter;
* Actualizează direcția în funcție de tasta apăsată:
  + W – sus,
  + A – stânga,
  + S – jos,
  + D – dreapta,
  + X – închide jocul.

### Logic()

* Actualizează poziția capului șarpelui în funcție de direcția setată;
* Vectorul tail este actualizat: capul devine primul segment, celelalte segmente se deplasează;
* Se verifică coliziunile:
  + cu pereții hărții → gameOver = true;
  + cu propria coadă → gameOver = true;
* Dacă șarpele ajunge pe fruct:
  + scorul crește cu 10;
  + coada se lungește;
  + se generează un nou fruct în poziție aleatorie.

## ****3.4 Observații privind optimizarea jocului****

Deși aplicația funcționează eficient, există câteva observații care pot duce la optimizări și îmbunătățiri:

* **Eficiența redării:**  
  Utilizarea system("cls") este simplă, dar lentă și dependentă de sistemul de operare. Poate fi înlocuită cu metode de manipulare directă a cursorului (ex: SetConsoleCursorPosition() în Windows) sau cu interfețe grafice dedicate.
* **Structura de date pentru coadă:**  
  Vectorul este eficient pentru început, dar pentru un joc cu coadă mare, o listă dublu înlănțuită ar putea reduce costurile de inserare/eliminare.
* **Controlul vitezei jocului:**  
  Pauza introdusă cu Sleep(100) poate fi ajustată în funcție de scor sau nivel, pentru a oferi un grad de dificultate dinamic.
* **Coliziuni și detecție:**  
  Verificarea coliziunilor cu coada se face prin parcurgerea întregului vector. O soluție mai eficientă poate fi folosirea unei hărți (unordered\_set) pentru detecție mai rapidă.

# ****CAPITOLUL 4 – Concluzii și contribuții personale****

## ****4.1 Concluzii****

Proiectul „Snake” a fost conceput și dezvoltat ca un exercițiu practic în cadrul studiului programării în limbajul C++, având ca scop aprofundarea unor concepte fundamentale prin aplicarea lor într-un context real, interactiv și intuitiv.

Prin realizarea acestui joc, s-a dovedit că o idee simplă poate fi transpusă într-o aplicație complet funcțională, organizată clar și capabilă să ofere o experiență coerentă utilizatorului. Întregul proces de dezvoltare a fost structurat pe etape: inițializare, proiectarea logicii, implementarea funcțiilor, testarea și validarea rezultatelor.

Pe parcursul lucrării, am atins o serie de obiective importante:

* **Aplicarea noțiunilor de bază**: am utilizat variabile, bucle, condiții logice, vectori, funcții și enumerații pentru a construi logica jocului;
* **Gestionarea inputului în timp real**: cu ajutorul funcțiilor kbhit() și getch(), am realizat controlul continuu al șarpelui fără a întrerupe bucla jocului;
* **Modelarea dinamică a cozii**: coada șarpelui a fost reprezentată eficient cu ajutorul unui vector de perechi de coordonate, fiind actualizată la fiecare iterație;
* **Afișare în consolă**: folosind caractere ASCII, am reușit să simulăm o interfață minimalistă, dar clară și funcțională;
* **Verificări logice**: am implementat detecția coliziunilor, actualizarea scorului și oprirea jocului în caz de Game Over;
* **Structurare modulară**: fiecare funcționalitate a fost organizată în funcții independente, ceea ce a permis testarea și extinderea ușoară a codului.

Pe lângă rezultatul concret – un joc Snake complet funcțional – această lucrare a reprezentat o oportunitate valoroasă de dezvoltare personală, perfecționare în programare și dobândire a unei gândiri algoritmice clare.

Jocul s-a dovedit a fi stabil, scalabil și ușor de modificat. Aceste calități îl recomandă ca bază pentru proiecte viitoare mai complexe, precum jocuri cu interfață grafică sau aplicații interactive cu funcționalități avansate.

## ****4.2 Contribuții personale****

Proiectul a fost realizat integral de către autori, fără utilizarea unor template-uri externe sau a unor motoare de joc. Acest lucru reflectă un angajament serios față de procesul de învățare și o dorință reală de a înțelege fiecare componentă a aplicației.

Principalele contribuții personale sunt evidențiate în următoarele aspecte:

* **Planificarea logicii jocului**: am conceput structura funcțională a aplicației înainte de scrierea codului, definind clar ce responsabilități are fiecare funcție;
* **Scrierea codului de la zero**: toate funcțiile (Setup, Draw, Input, Logic) au fost scrise de către autori, pe baza înțelegerii directe a cerințelor și a comportamentului dorit al jocului;
* **Gestionarea cozii în mod dinamic**: coada este o structură variabilă care trebuie actualizată cu precizie la fiecare cadru. Soluția aleasă (vector de perechi) s-a dovedit eficientă, clară și extensibilă;
* **Controlul în timp real**: am integrat un sistem de input care permite mișcarea fluidă a șarpelui fără blocaje, esențial pentru un gameplay plăcut;
* **Testarea și depanarea aplicației**: am rulat jocul în mai multe scenarii, am identificat și corectat bug-uri, asigurând o experiență stabilă și coerentă;
* **Comentarierea și organizarea codului**: codul sursă a fost documentat intern, iar funcțiile au fost ordonate logic, pentru a permite înțelegerea rapidă de către alți utilizatori;
* **Structurarea documentației**: această lucrare nu doar descrie implementarea jocului, ci o face într-un mod structurat, academic, complet, cu toate capitolele necesare.

Toate aceste contribuții reflectă munca efectivă depusă și gradul de implicare în proiect. Mai important decât realizarea jocului în sine este învățarea profundă și dezvoltarea abilităților dobândite pe parcurs.

# ****CAPITOLUL 5 – Perspective de viitor****

Deși aplicația „Snake” în forma actuală oferă o experiență completă și funcțională în mediul consolă, există o multitudine de direcții în care proiectul poate fi îmbunătățit și extins. Acest capitol prezintă o serie de idei și propuneri pentru dezvoltarea viitoare a aplicației, menite să îmbogățească atât interactivitatea, cât și complexitatea tehnică a jocului.

## ****5.1 Funcționalități suplimentare****

Pentru a crește gradul de atractivitate și provocare al jocului, pot fi implementate următoarele îmbunătățiri:

### 🔹 Sistem de niveluri și dificultăți

* Introducerea unor niveluri de joc, cu viteze de deplasare diferite sau cu obstacole plasate strategic;
* Fiecare nivel ar putea include reguli sau hărți proprii, crescând progresiv dificultatea și complexitatea jocului.

### 🔹 Obstacole statice și dinamice

* Pe hartă pot fi introduse obiecte (ziduri, mine, bariere) care trebuie evitate;
* Obstacolele mobile pot adăuga un element suplimentar de strategie și imprevizibilitate.

### 🔹 Sistem de scoruri maxime (High Score)

* Salvarea automată a celui mai mare scor într-un fișier text;
* Afișarea scorurilor maxime într-un clasament la începutul jocului.

### 🔹 Adaptarea vitezei în funcție de scor

* Viteza jocului poate fi mărită automat pe măsură ce scorul crește;
* Acest mecanism face jocul mai dinamic și îl transformă într-o adevărată provocare pentru utilizator.

### 🔹 Pauză și reluare

* Posibilitatea de a întrerupe temporar jocul (ex: apăsând tasta P);
* Reluarea jocului fără a pierde progresul înregistrat anterior.

## ****5.2 Îmbunătățiri grafice și de interfață****

Pentru a transforma jocul într-o experiență vizuală modernă, se pot aplica următoarele modificări:

### 🔹 Interfață grafică (GUI)

* Utilizarea bibliotecii **SFML**, **SDL** sau **OpenGL** pentru a înlocui consola cu o fereastră grafică;
* Desenarea șarpelui, fructelor și pereților folosind imagini și culori atractive.

### 🔹 Efecte sonore și muzică de fundal

* Adăugarea de sunete pentru evenimente precum: colectarea fructului, coliziuni, final de joc;
* Integrarea unei coloane sonore ambientale pentru imersiune sporită.

### 🔹 Meniu principal și ecran de opțiuni

* Crearea unui meniu de start în care utilizatorul să aleagă dificultatea, să vizualizeze scorurile sau să configureze sunetul;
* Oferirea de opțiuni de personalizare (teme vizuale, tipuri de fundal).

### 🔹 Tematizare vizuală

* Implementarea unor „skin-uri” pentru șarpe și fundal (ex: clasic, futurist, pixel-art);
* Permisiunea utilizatorului de a alege tema preferată înainte de începerea jocului.

## ****5.3 Abordări avansate și extindere profesională****

Pentru utilizatori avansați sau pentru transformarea aplicației într-un produs de tip portofoliu profesional, se pot lua în considerare următoarele direcții:

### 🔹 Programare orientată pe obiect (OOP)

* Refactorizarea codului actual în clase: Game, Snake, Fruit, Map;
* Această abordare îmbunătățește organizarea codului, testarea și extinderea sa.

### 🔹 Compatibilitate multiplatformă

* Eliminarea dependenței de conio.h și windows.h pentru a permite rularea pe Linux sau macOS;
* Utilizarea funcțiilor standard C++ și a bibliotecilor portabile (ex: ncurses pentru Linux).

### 🔹 Multiplayer local

* Adăugarea unui mod în care doi jucători pot controla șerpi separați (ex: tastele W/A/S/D pentru jucătorul 1, săgețile pentru jucătorul 2);
* Competiție pentru fructe sau mod de joc cooperativ cu un singur scor comun.

### 🔹 Inteligență artificială (AI)

* Crearea unui bot care joacă automat, urmărind cel mai apropiat fruct și evitând coliziunile;
* Util pentru testarea jocului sau pentru un mod de „jucător vs. AI”.

## ****Concluzie****

Proiectul „Snake” reprezintă mai mult decât un simplu joc de consolă – el este o platformă de învățare, testare și extindere a cunoștințelor în programare. Pe baza acestei versiuni de bază, pot fi construite versiuni avansate, cu grafică, sunet, interactivitate sporită și funcționalități moderne.

Prin parcurgerea etapelor de extindere propuse, proiectul poate deveni o aplicație completă, profesională, utilă nu doar în mediul academic, ci și ca piesă de portofoliu pentru interviuri tehnice sau prezentări personale.

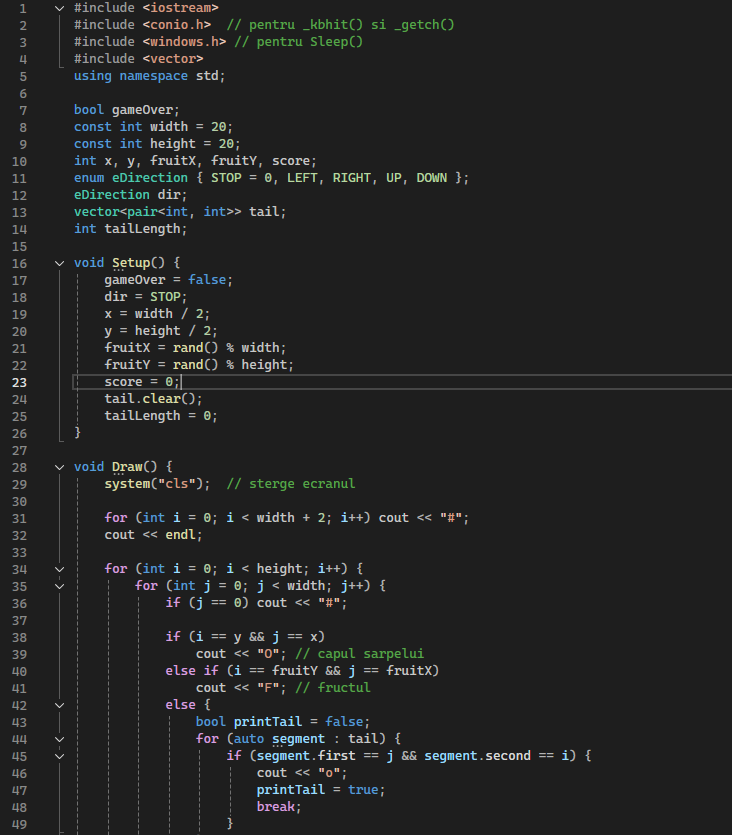
# ****BIBLIOGRAFIE****

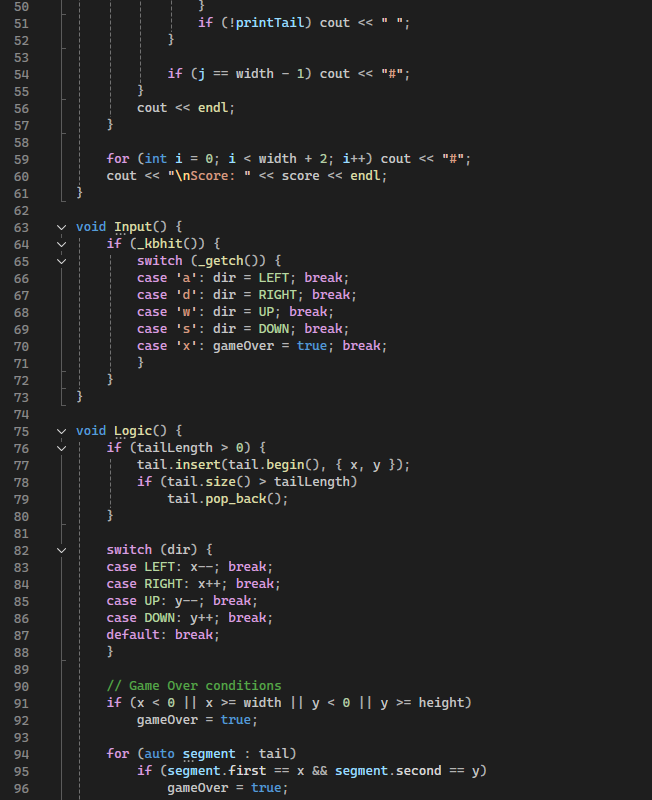
1. [cplusplus.com](https://www.cplusplus.com/) – Documentație oficială C++  
    Utilizată pentru înțelegerea funcțiilor de bază, a sintaxei limbajului C++ și a manipulării vectorilor.
2. [geeksforgeeks.org](https://www.geeksforgeeks.org/) – Tutoriale și exemple C++  
    Resurse utile pentru aprofundarea conceptelor de jocuri, algoritmi și structuri de date.
3. [stackoverflow.com](https://stackoverflow.com/) – Forum pentru programatori  
    Folosit pentru soluționarea problemelor legate de \_kbhit(), \_getch(), structuri de date dinamice și logica de joc.
4. YouTube – „How to Make Snake Game in C++ Console”  
    Surse vizuale și practice pentru implementarea jocului Snake pas cu pas.
5. Documentația Visual Studio Code și MinGW  
    Folosite pentru configurarea mediului de dezvoltare și compilatorului pentru rularea aplicației.

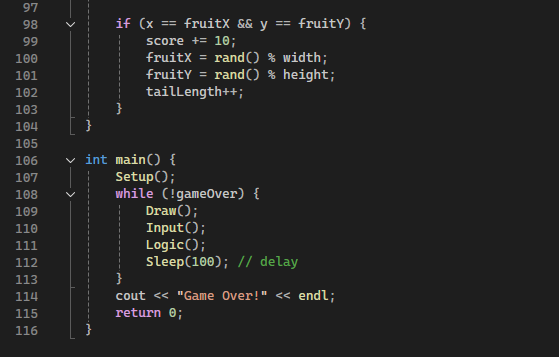
# ****ANEXE****

## ****Anexa 1 – Codul sursă complet al jocului Snake****

Această anexă conține codul sursă complet al aplicației „Snake”, realizată în limbajul C++. Codul este organizat modular și respectă principiile de programare procedurală. Toate funcțiile necesare funcționării jocului (inițializare, desenare, input, logică de joc) sunt incluse, iar aplicația poate fi compilată și rulată într-un mediu de dezvoltare compatibil cu MinGW sau Visual Studio Code.





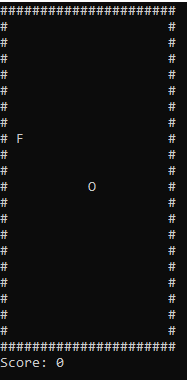


## ****Anexa 2 – Capturi de ecran ale aplicației în execuție****

Această anexă prezintă o serie de capturi de ecran realizate în timpul rulării aplicației, pentru a evidenția funcționalitatea jocului și comportamentul acestuia în diferite etape.

### ****Figura 1 – Lansarea aplicației (stare inițială)****

Imaginea prezintă starea inițială a jocului după lansare. Șarpele este poziționat în centrul hărții, iar primul fruct apare într-o poziție aleatoare.



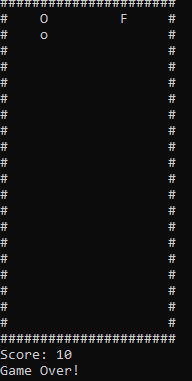
### ****Figura 2 – Șarpele cu coadă lungă****

Imaginea evidențiază cum se extinde coada șarpelui pe măsură ce sunt colectate mai multe fructe, iar spațiul de mișcare devine tot mai redus.



### ****Figura 3 – Game Over prin coliziune cu peretele/coada****

Această captură arată sfârșitul jocului în momentul în care șarpele a atins limita hărții/coada.

****