**Proiect PROGRAMAREA CALCULATOARELOR SI LIMBAJE DE PROGRAMARE**

**-C++-**

**Profesori Coordonatori:** **Studenți:**

NECULOIU GIORGIAN VLAICU ADRIANA TEODORA

Gabriela OLTEANU

**București, 2024**

**JOCUL SUDOKU 4X4**

**Documentație pentru Jocul Sudoku 4x4 în C++**

**VLAICU ADRIANA TEODORA**

**1. Introducere generală**

Această documentație descrie în detaliu implementarea unui joc simplificat de Sudoku scris în C++. Scopul acestui program este de a familiariza programatorii începători cu:

* Structuri de date dinamice (vector<vector<int>>)
* Manipularea inputului utilizatorului
* Algoritmica validării regulilor Sudoku
* Fluxul de control și structurarea codului

Sudoku este un joc de logică care poate fi modelat foarte eficient în limbaje precum C++ datorită expresivității acestuia și suportului său robust pentru structuri de date.

**2. Prezentare generală a Sudoku**

În forma sa clasică, Sudoku este un joc logic în care scopul este de a umple o grilă 9x9 împărțită în 9 subgrile 3x3 cu cifre de la 1 la 9. În cazul nostru, folosim o variantă 4x4, împărțită în 4 subgrile 2x2.

**Reguli Sudoku 4x4**

* Fiecare rând trebuie să conțină numerele 1-4 fără repetări
* Fiecare coloană trebuie să conțină numerele 1-4 fără repetări
* Fiecare subgrilă 2x2 trebuie să conțină numerele 1-4 fără repetări

**3. Designul aplicației**

Structura programului este modulară și separă clar responsabilitățile:

* Funcția main() pornește jocul.
* joc\_sudoku() conține bucla principală și logica de joc.
* Funcții ajutătoare: afișare, verificare validitate, verificare completitudine.

Grila este reprezentată ca un vector bidimensional de întregi:

cpp

CopyEdit

vector<vector<int>> tabla;

Această alegere permite o manipulare ușoară a rândurilor și coloanelor.

**4. Explicații detaliate pe componente**

**4.1. afiseaza\_tabla()**

cpp

CopyEdit

void afiseaza\_tabla(const vector<vector<int>>& tabla)

**Rol:**  
Afișează conținutul grilei în consolă. Fiecare linie este afișată pe un rând nou, fiecare valoare fiind separată de un spațiu.

**De ce referință constantă?**

* **Eficiență**: nu copiem grila.
* **Securitate**: nu permitem modificarea ei accidentală.

**Output exemplu:**

CopyEdit

1 0 0 4

0 0 2 0

0 3 0 0

4 0 0 0

**4.2. este\_valida()**

cpp

CopyEdit

bool este\_valida(const vector<vector<int>>& tabla, int linie, int coloana, int valoare)

**Rol:**  
Determină dacă o valoare poate fi plasată într-o celulă dată fără a încălca regulile Sudoku.

**Verificări efectuate:**

1. **Rând**: nu trebuie să existe valoare în tabla[linie][\*]
2. **Coloană**: nu trebuie să existe valoare în tabla[\*][coloana]
3. **Subgrilă 2x2**:
   * Se calculează colțul stânga-sus al subgrilei:

cpp

CopyEdit

int start\_linie = (linie / 2) \* 2;

int start\_coloana = (coloana / 2) \* 2;

* + Se verifică toate cele 4 celule din pătratul 2x2 respectiv.

**Exemple:**

* Dacă în rândul 0 există deja 1, inserarea unui alt 1 va returna false.

**4.3. este\_complet()**

cpp

CopyEdit

bool este\_complet(const vector<vector<int>>& tabla)

**Rol:**  
Verifică dacă există celule goale (0) în grilă.

* Parcurge toate valorile.
* Dacă găsește cel puțin una egală cu 0, returnează false.

**Utilitate:**  
Determină oprirea jocului atunci când tabla este completă.

**4.4. joc\_sudoku()**

Aceasta este **funcția principală** care gestionează logica jocului.

**Inițializare:**

cpp

CopyEdit

vector<vector<int>> tabla = {

{1, 0, 0, 4},

{0, 0, 2, 0},

{0, 3, 0, 0},

{4, 0, 0, 0}

};

**Bucla de joc:**

cpp

CopyEdit

while (!este\_complet(tabla)) { ... }

Aceasta:

* Afișează tabla
* Solicită input de la utilizator (linie, coloană, valoare)
* Realizează validări multiple:
  + Indici în interval (1-4)
  + Celula este liberă
  + Valoarea respectă regulile Sudoku

Dacă mutarea este validă, se actualizează grila:

cpp

CopyEdit

tabla[linie][coloana] = valoare;

La final:

cpp

CopyEdit

cout << "Felicitari! Ai completat tabla Sudoku:";

afiseaza\_tabla(tabla);

**4.5. main(**

cpp

CopyEdit

int main() {

joc\_sudoku();

return 0;

}

Punctul de intrare în program.

**5. Fluxul de execuție pas cu pas**

1. Se pornește programul.
2. Se afișează tabla inițială.
3. Utilizatorul introduce o mutare.
4. Se verifică validitatea mutării.
5. Dacă este validă, se actualizează tabla.
6. Se verifică dacă jocul s-a terminat.
7. Se reia pasul 2 până la completarea grilei.

**6. Verificări și validări**

**Validări de input:**

cpp

CopyEdit

if (linie < 0 || linie > 3 || coloana < 0 || coloana > 3 || valoare < 1 || valoare > 4)

**Validare stare celulă:**

cpp

CopyEdit

if (tabla[linie][coloana] != 0)

**Validare logică Sudoku:**

cpp

CopyEdit

if (este\_valida(tabla, linie, coloana, valoare))

**7. Analiză algoritmică**

| **Funcție** | **Complexitate** | **Explicație** |
| --- | --- | --- |
| afiseaza\_tabla | O(n²) | Parcurgere a fiecărei celule |
| este\_valida | O(n) | Rând + coloană + subgrilă |
| este\_complet | O(n²) | Verificare a fiecărei celule |

Pentru o grilă 4x4, aceste complexități sunt nesemnificative, dar pentru 9x9 devin mai importante.

**8. Îmbunătățiri posibile**

1. **Suport pentru 9x9**
2. **Hint-uri (sugestii pentru mutări valide)**
3. **Salvarea și încărcarea jocului**
4. **Interfață grafică (ex: Qt)**
5. **Validare finală a soluției**
6. **Algoritm de generare automată a grilelor**
7. **Scor și cronometru**

**9. Aplicații educaționale**

* Exersarea structurii vector<vector<int>>
* Înțelegerea programării orientate pe reguli
* Gestionarea erorilor și a inputului utilizatorului
* Algoritmica validării și parcurgerii matricelor

**11. Motivare alegere temă**

Tema aleasă — implementarea unui joc Sudoku 4x4 în C++ — a fost selectată pentru a atinge mai multe obiective educaționale și practice:

* Sudoku este un joc logic, accesibil și cunoscut, ceea ce îl face o temă atractivă și intuitivă.
* Varianta 4x4 este suficient de simplă pentru a putea fi implementată și testată rapid, dar suficient de interesantă pentru a evidenția concepte precum:
  + manipularea structurilor bidimensionale (vector<vector<int>>),
  + controlul fluxului de execuție,
  + validarea inputului,
  + și implementarea unui joc interactiv în consolă.
* C++ oferă atât performanță, cât și expresivitate, ceea ce îl face ideal pentru acest tip de aplicație logică.

**12. Obiective**

Obiectivele propuse pentru această aplicație sunt:

1. **Didactice**:
   * Învățarea utilizării vectorilor 2D în C++.
   * Practicarea lucrului cu bucle, condiții, funcții.
   * Consolidarea noțiunilor de validare a datelor introduse de utilizator.
2. **Funcționale**:
   * Implementarea unui joc Sudoku funcțional în consolă.
   * Verificarea regulilor de joc pentru fiecare mutare.
   * Oprirea jocului când tabla este completă.
3. **Tehnice**:
   * Separarea codului în funcții modulare.
   * Respectarea principiilor de programare clară și eficientă.
   * Posibilitatea extinderii aplicației pentru a suporta versiuni mai mari.

**13. Tehnologii utilizate**

| **Tehnologie / Limbaj** | **Rol** |
| --- | --- |
| C++ (standard C++11+) | Limbajul folosit pentru dezvoltarea aplicației |
| STL (<vector>) | Pentru manipularea dinamică a matricei de joc |
| <iostream> | Pentru input/output în consolă |
| Terminal/Consolă C++ | Mediu de rulare și testare |
| IDE (ex: Code::Blocks, VS Code, Dev-C++) | Editare și compilare cod |

**14. Studiu de caz – Sudoku 4x4**

Am ales să implementăm un joc **Sudoku de dimensiune 4x4**, în locul versiunii clasice 9x9, din următoarele motive:

* Permite testarea rapidă a jocului.
* Este ideal pentru începători, datorită numărului redus de celule.
* Poate fi ușor extins la versiuni mai complexe.
* Permite punerea în aplicare a regulilor de Sudoku într-un mod simplificat.

**Date de intrare:**

* Grila inițială cu unele celule completate.
* Input de la utilizator: linie, coloană, valoare.

**Reguli:**

* Fiecare rând, coloană și subgrilă 2x2 trebuie să conțină cifrele 1-4 fără repetări.

**15. Implementare pas cu pas**

1. **Structura grilei:**
   * Se declară un vector<vector<int>> pentru a simula grila 4x4.
   * Celulele goale sunt marcate cu 0.
2. **Funcția afiseaza\_tabla:**
   * Afișează grila în forma unui tabel în consolă.
3. **Funcția este\_valida:**
   * Verifică rândul, coloana și subgrila corespunzătoare dacă permit valoarea introdusă.
4. **Funcția este\_complet:**
   * Verifică dacă toate celulele sunt completate.
5. **Funcția joc\_sudoku:**
   * Inițializează tabla.
   * Afișează instrucțiuni și primește mutări de la utilizator.
   * Verifică validitatea mutării și actualizează grila dacă este corectă.
6. **Funcția main:**
   * Punctul de pornire al aplicației. Apelează joc\_sudoku().

**16. Bibliografie**

1. Bjarne Stroustrup – *The C++ Programming Language*
2. cppreference.com – STL Containers (vector)
3. geeksforgeeks.org – C++ Vector 2D Usage
4. Wikipedia – [Sudoku](https://en.wikipedia.org/wiki/Sudoku)
5. openai.com – Asistență generare documentație

**17. Anexe**

**Anexa A: Exemplu de input valid**

java

Copy code

Introduceti linia (1-4): 2

Introduceti coloana (1-4): 2

Introduceti valoarea (1-4): 1

**Anexa B: Output grilă intermediară**

Copy code

1 0 0 4

0 1 2 0

0 3 0 0

4 0 0 0

***18. Concluzie***

Acest joc Sudoku 4x4 implementat în C++ este un exemplu excelent de aplicare a logicii, structurilor de date bidimensionale și manipulare de input în limbajul C++. Deși simplu, codul poate fi extins și rafinat pentru a deveni o aplicație completă. Este ideal pentru cei care doresc să îmbine noțiunile de bază de C++ cu gândirea algoritmică.