### **Rezolvarea jocului Sudoku**

### **folosind algoritmul de verificare înainte (Forward Checking)**

*Florea Alexandra - 1408A*

*Pintilie Justinian - 1408A*

### **1. Introducere**

Proiectul nostru reprezintă implementarea unui program de rezolvare a jocului Sudoku utilizând algoritmul Forward Checking. Programul este dezvoltat în limbajul C# și are ca scop demonstrarea aplicării algoritmilor de inteligență artificială în probleme de căutare cu spațiu mare de soluții.

### **2. Descrierea problemei**

Sudoku este un joc logic popular, în care jucătorul trebuie să completeze un grid 9x9 astfel încât:

* Fiecare rând să conțină cifrele de la 1 la 9 fără repetări.
* Fiecare coloană să conțină cifrele de la 1 la 9 fără repetări.
* Fiecare dintre cele 9 sub-grile 3x3 să conțină cifrele de la 1 la 9 fără repetări.

Problema poate fi considerată o problemă de satisfacere a constrângerilor (CSP - *Constraint Satisfaction Problem*), ceea ce o face potrivită pentru aplicarea algoritmilor precum Backtracking sau Forward Checking.

### **3. Aspecte teoretice privind algoritmul Forward Checking**

Algoritmul Forward Checking este o tehnică de optimizare aplicată în problemele de CSP pentru reducerea spațiului de căutare.  
**Principiul de bază**:

După atribuirea unei valori unei variabile, se verifică imediat efectele acestei atribuiri asupra variabilelor neasignate. Dacă cel puțin o variabilă rămâne fără opțiuni valide, se efectuează backtracking.

**Pași principali**:

1. Se selectează o variabilă de asignat (euristică MRV - *Minimum Remaining Values* este des utilizată).
2. Se atribuie o valoare variabilei din domeniul său curent.
3. Se verifică impactul asupra variabilelor neasignate.
4. Dacă domeniul unei variabile devine gol, se revine la pasul anterior (backtracking).
5. Se continuă până când gridul este complet sau s-a determinat că soluția nu există.

Avantaje:

* Reducerea semnificativă a numărului de verificări inutile.
* Îmbunătățirea eficienței față de backtracking-ul clasic.

### **4. Modalitatea de rezolvare**

#### **Structura aplicației**

Aplicația a fost construită în C# folosit Windows Form. Pe parcursul dezvoltării s-a decis implementarea a două clase principale care includ:

* SudokuForm - reprezintă interfața care gestionează interacțiunea cu utilizatorul.
* TableGenerator - care este responsabilă cu generarea tabelelor Sudoku.

#### **Funcționalități principale**

##### Generarea unei tabele noi

Pentru a se crea o tablă nouă, se apelează funcția

**TableGenerator.SudokuTableGenerator(int difficulty)**.

În momentul apelării acesteia se poate seta dificultatea în momentul creării tabelei, în prezent doar de către dezvoltator. Algoritmul din spate presupune crearea unei matrice complet valide, create cu algoritmul Forward Checking din care se vor șterge un număr egal cu parametrul **difficulty** de valori.

Valorile generate vor fi afișate în celule gri cu scris verde, cu proprietatea ReadOnly.

##### Validarea valorilor introduse în celule

Funcția de validare este **IsMoveValid(int row, int column, int value)**.

Aceasta verifică, după ce utilizatorul a introdus câteva valori, dacă acestea sunt valide sau nu. După verificare, valorile valide vor fi de culoare verde, iar cele invalide de culoare roșie. De asemenea se va semnala și printr-un mesaj informativ stadiul jocului după validare (numărul de valori greșite, dacă valorile sunt în regulă, dacă nu mai sunt posibile mutări etc.).

##### Curățarea tabelei

În cazul în care utilizatorul dorește reluarea jocului, poate ori să genereze o nouă tabelă, ori să “curețe” valorile introduse de el.

##### Oferirea de hint-uri

În cazul în care utilizatorul nu știe cum să rezolve jocul sau are nevoie de ajutor, acesta poate apela la metoda care poate genera o valoare validă. Numărul de hint-uri poate fi setat doar de dezvoltator, în prezent acesta fiind 10. Această funcționalitate este oferită de funcția

**GetPossibleValuesFromForwardChecking(row, column)**

##### Funcție de rezolvare automată a jocului

Jocul are și o funcție automată de rezolvare, prin care jucătorul poate vedea rezolvarea unei table în mod automat, pas cu pas. Aceasta are implementat algoritmul de Forward Checking.

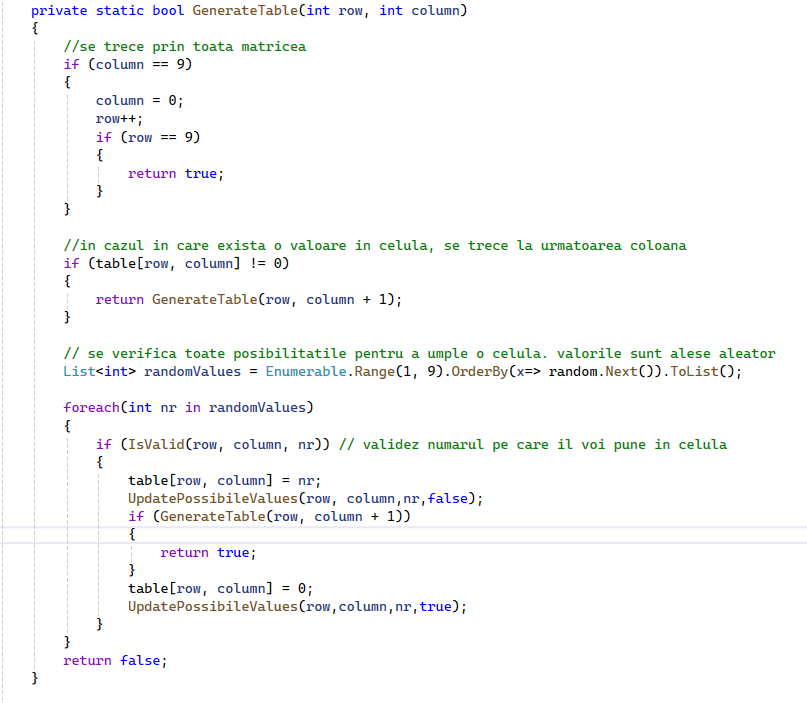
##### Ajutor pentru utilizator

Dacă utilizatorul nu știe regulile jocului, acesta poate beneficia de o funcție de help care explică regulile jocului.

#### 

### **5. Cod sursă și explicații**

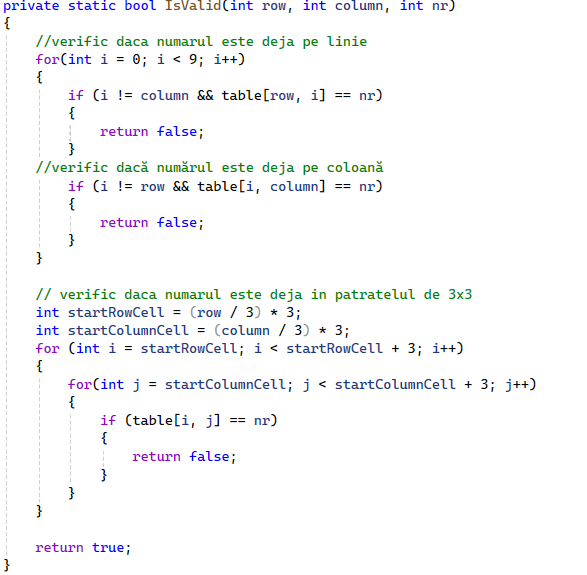
#### **Exemplu de cod 1: Generarea tabelei**



**Explicație:**

Această funcție stă la baza generării de tabele. Prin intermediul acesteia se fac validările necesare algoritmului Forward Checking, se verifică dacă celulele au deja o valoare, se verifică validarea mutărilor și se actualizează vectorul de situații posibile. Aici intervine și algoritmul de Backtracking, prin recursivitate, care revine la valorile anterioare pentru a putea valida intrarea curentă. O funcție asemănătoare este folosită și pentru interfață.

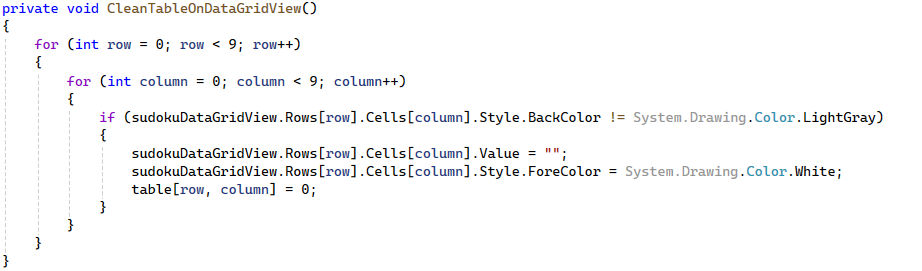
#### **Exemplu de cod 2: Funcție de validare a valorilor**



**Explicație:**

Această funcție este asemănătoare cu cea din clasa pentru interfață (**IsMoveValid**), rolul ei fiind de a face verificări pe colană, linie și în pătratul 3x3 din care face parte.

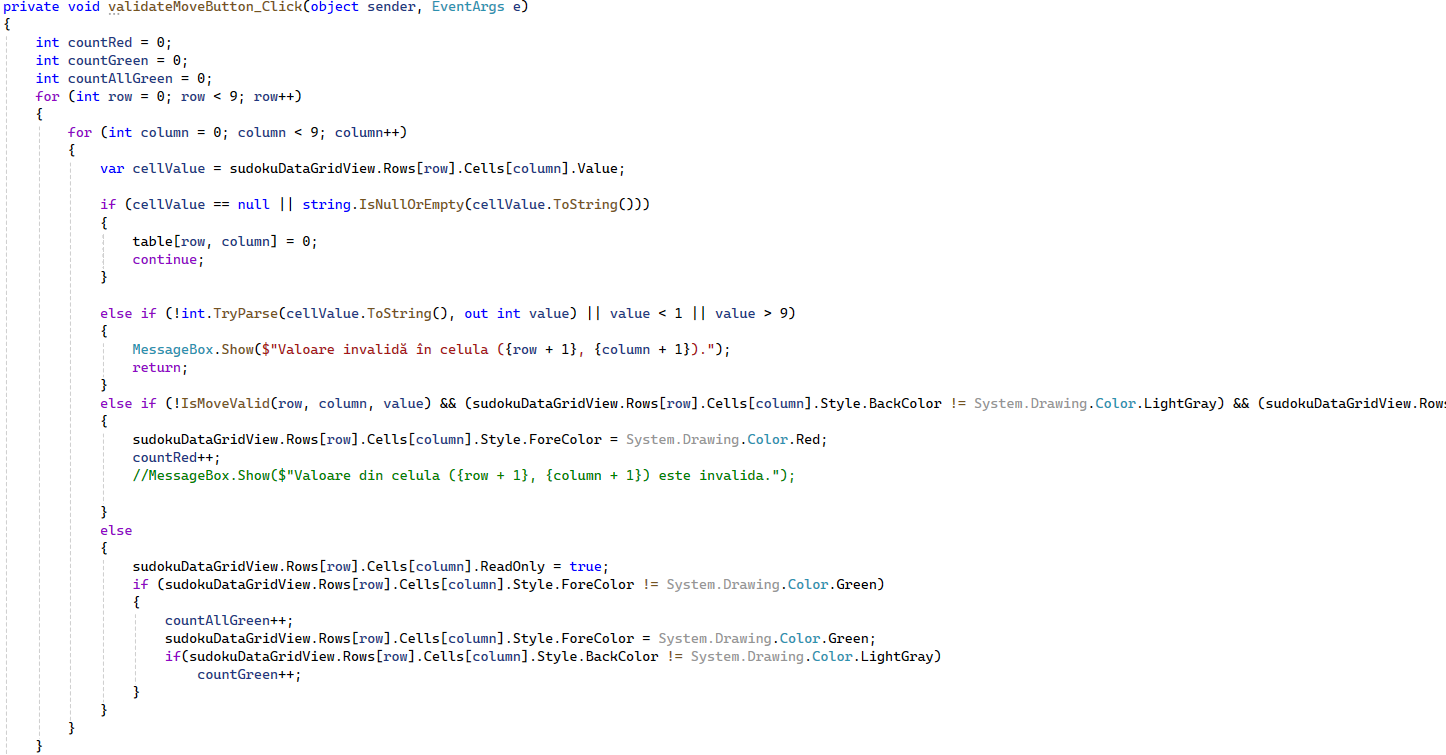
#### **Exemplu de cod 3: Funcție de ștergere a valorilor introduse de utilizator**



**Explicație:**

Funcția are rolul de a șterge valorile adăugate de utilizator, în cazul în care acesta dorește să înceapă din nou jocul sau ajunge într-un punct în care nu mai există valori valide.

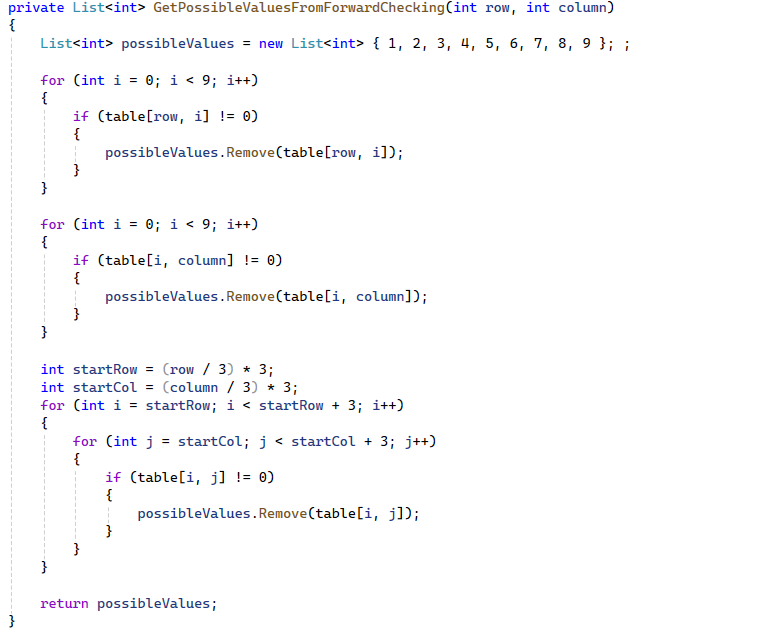
#### **Exemplu de cod 4: Funcție de validare a valorilor introduse de utilizator**



**Explicație:**

Această funcție utilizează funcția asemănătoare cu cea prezentată la exemplul 2, rolul ei fiind de a face verificări pe coloană, linie și în pătratul 3x3 din care face parte dar și de a semnala vizual utilizatorului stadiul valorilor introduse, atât prin culori, cât și prin mesaje informative sugestive.

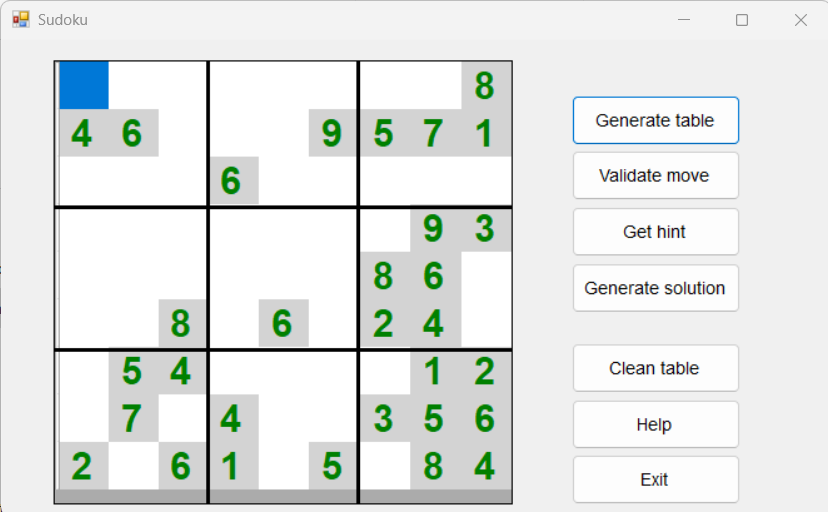
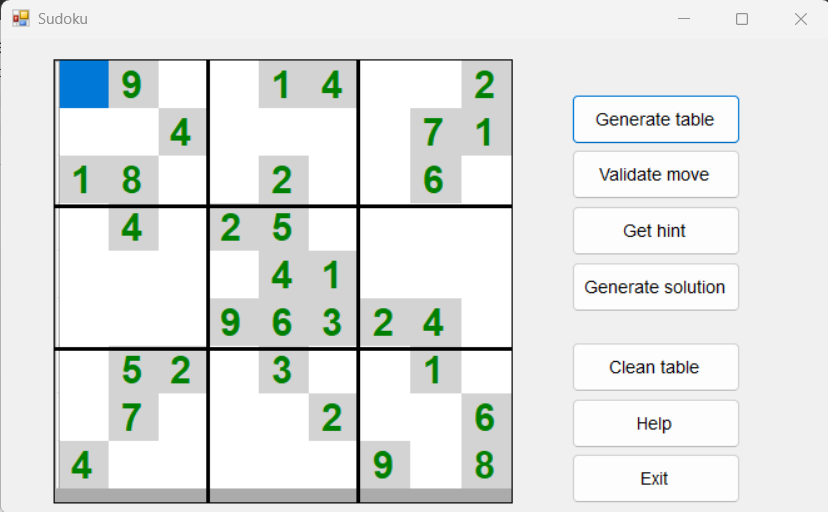
#### **Exemplu de cod 5: Funcție de actualizare a valorilor posibile ( Forward Checking)**



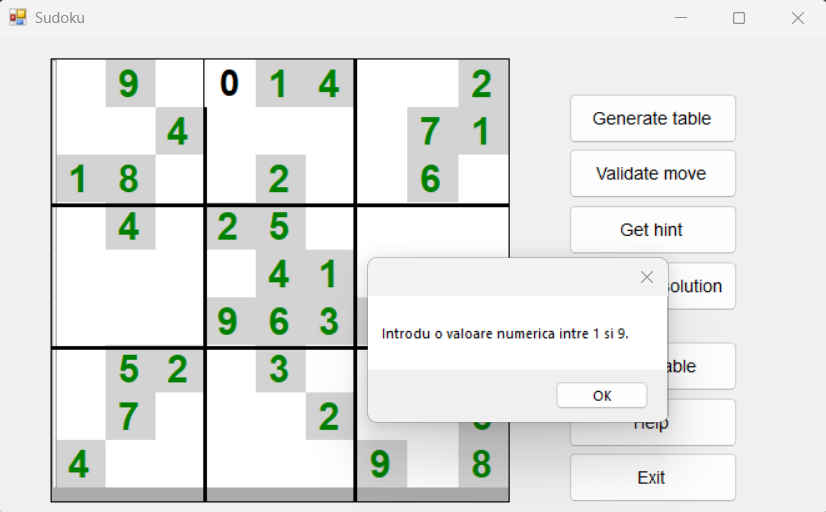
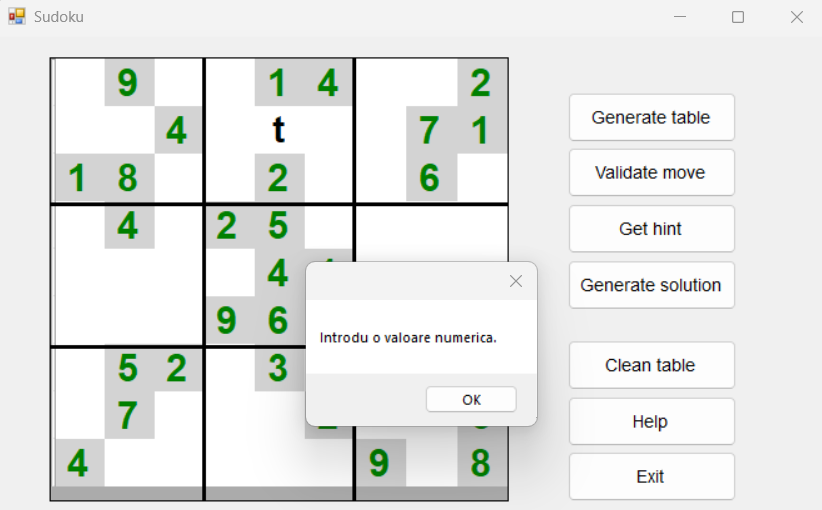
**Explicație:**

Prin intermediul funcției se actualizează valorile posibile pe care le poate avea o celulă. Această funcție cuprinde constrângerile jocului și crează o listă validă de valori. Este folosită în funcția pentru hint-uri.

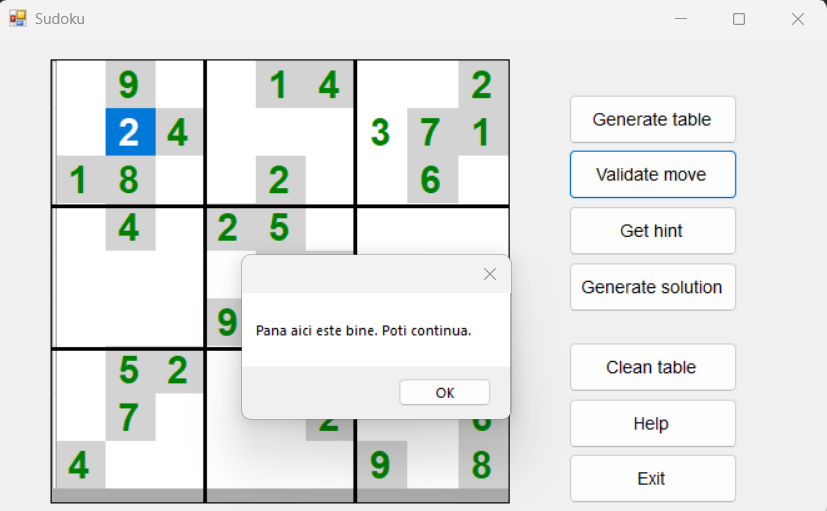
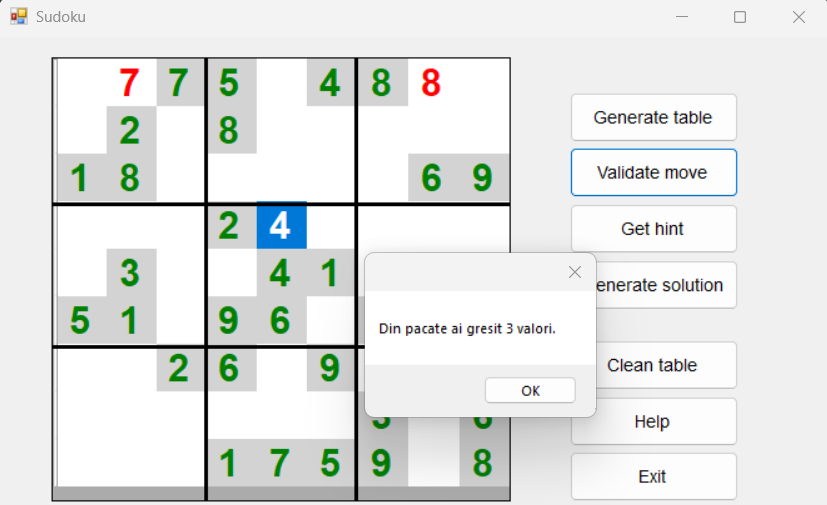
### **6. Rezultate obținute**

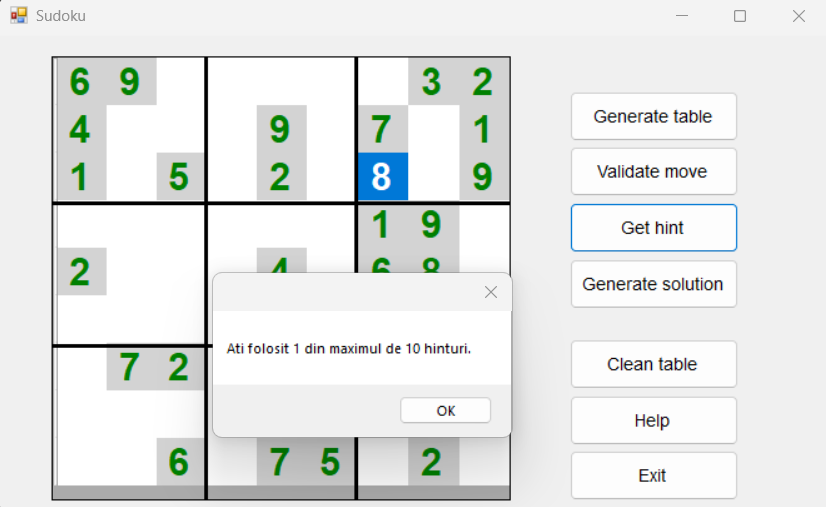
Aceasta este interfața cu utilizatorul după apăsarea butonului **Generate table.** La fiecare apăsare de buton se generează o altă matrice.

** **

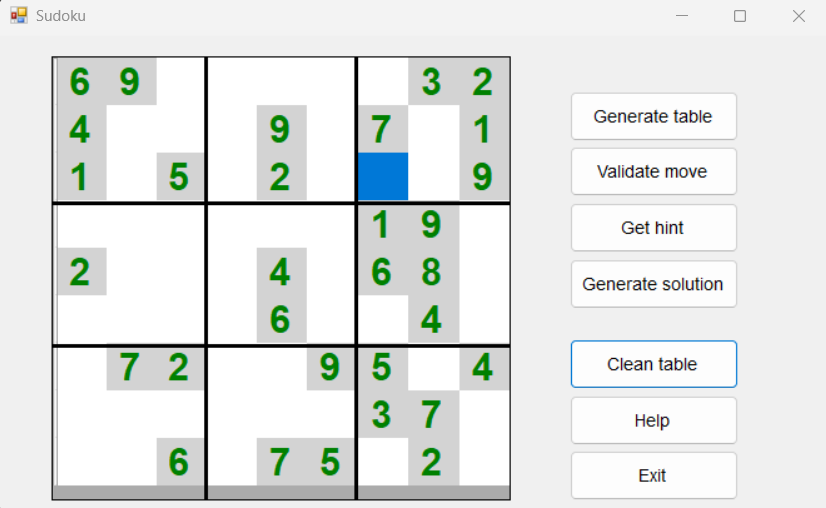
Eroare primită după adăugarea unui caracter non-numeric, respectiv din afara intervalului [1,9] într-o celulă.

** **

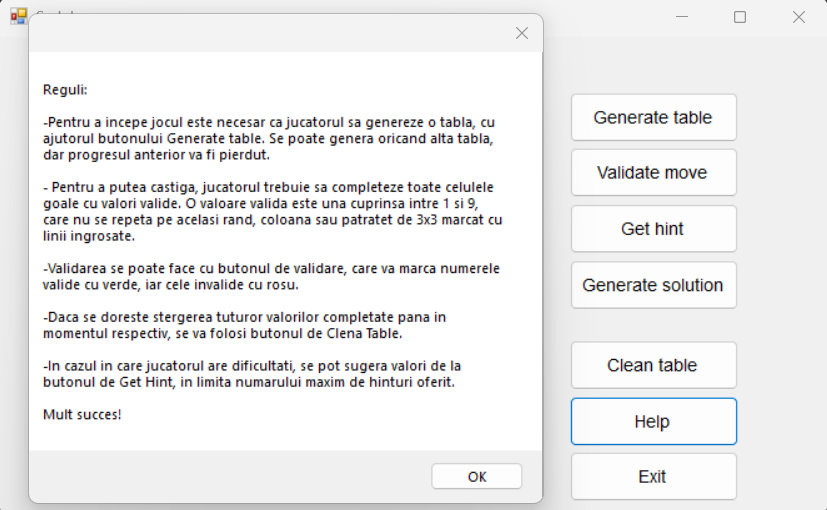
Exemple de mesaje primite după apăsarea butonului **Validate move.** Se observă că celulele cu valori valide au textul verde, iar cele invalide au textul roșu.

****

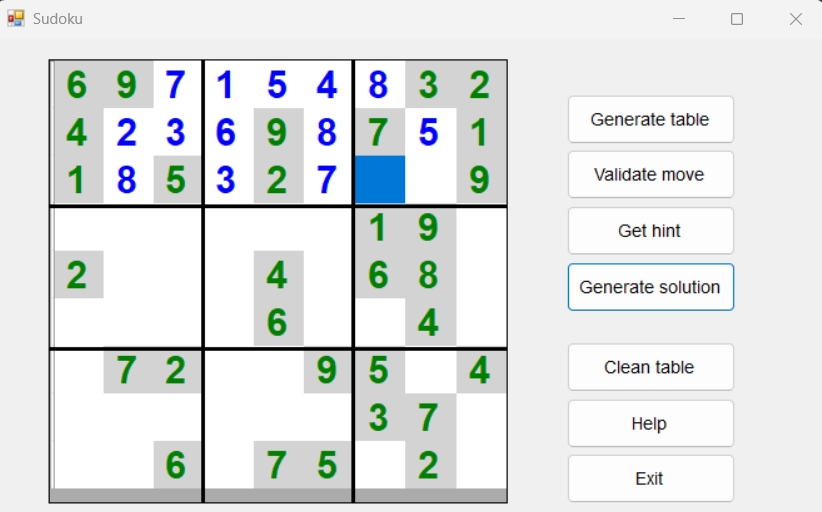
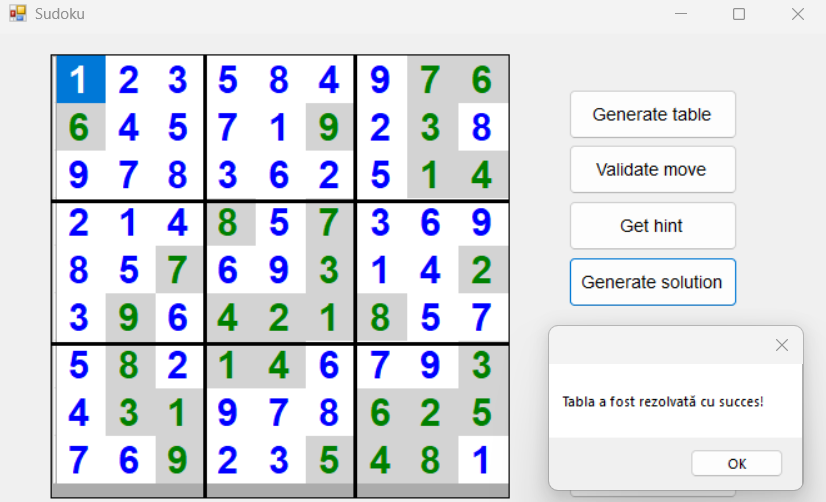
Exemplu de mesaj informativ după apăsarea butonului **Get hint**. Fiecare hint se contorizează și se oferă utilizatorului acces prin astfel de mesaje informative la numărul de hint-uri consumate.



După apăsarea butonului **Clean table**  se revine la stadiul tabelei din momentul generării.



Butonul **Help** oferă utilizatorului un mesaj informativ cu regulile jocului și o scurtă prezentare a funcționalităților butoanelor.

Butonul **Generate solution** este cel mai important buton și el îndeplinește scopul acestui proiect deoarece prin intermediul lui se generează, cu ajutorul algoritmului Forward Checking rezolvarea jocului. Algoritmul a fost implementat astfel încât să existe întârzieri și să se evidențieze fiecare mișcare pentru a se pune în evidență Backtracking-ul.

### **7. Concluzii**

Implementarea algoritmului Forward Checking în rezolvarea Sudoku a demonstrat o performanță superioară metodei de backtracking simplu, reducând semnificativ timpul de căutare. Cu toate acestea, complexitatea problemei crește exponențial pentru grid-uri de dimensiuni mai mari.

### **8. Bibliografie**

* <https://sandipanweb.wordpress.com/2017/03/17/solving-sudoku-as-a-constraint-satisfaction-problem-using-constraint-propagation-with-arc-consistency-checking-and-then-backtracking-with-minimum-remaning-value-heuristic-and-forward-checking/>
* <https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/>
* <https://www.youtube.com/watch?v=ejhmLHd3n_4&list=LL&index=11&t=70s&ab_channel=BrandonAdame>
* <https://cs188ai.fandom.com/wiki/Forward_Checking>
* https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/

### **9. Contribuții individuale ale membrilor echipei**

* Creare proiect Github - **Florea Alexandra**
* Creare schelet proiect - **Justinian Pintilie**
* Adăugare funcții de generare a matricei pentru Sudoku - **Florea Alexandra**
* Adăugare constrângeri pentru utilizator și validări pentru celule - **Justinian Pintilie**
* Crearea unei schițe a documentației. Scrierea primei părți a documentației - **Justinian Pintilie**
* Corectarea erorilor. Crearea funcției GetHint - **Florea Alexandra**
* Redactare capitole 4,5,6,9 documentație - **Florea Alexandra**
* Redactare capitole 1,2,3,8 documentație - **Pintilie Justinian**
* Adăugare funcție de generare soluții cu backtracking - **Florea Alexandra**
* Update documentație cu noile funcționalități - **Florea Alexandra**