



SUDOKU GAME PRESENTATION

Μαρία Σιώπη, Μαρία Μιχαλέρη, Αφροδίτη Πιστικοπούλου

Περιεχόμενα Παρουσίασης

- Εισαγωγή & Motivation
- Objective & Scope
- System Architecture
- Τεχνολογίες που Χρησιμοποιήθηκαν
- Κώδικας & Υλοποίηση
- Αποτελέσματα & Demo
- Σύγκριση με AI-generated code
- Συμπεράσματα & Lessons Learned

Εισαγωγή & Motivation

- Το Sudoku είναι ένα παιχνίδι λογικής με αριθμούς από το 1 έως το 9.
- Ο στόχος είναι να γεμίσουν όλα τα κελιά χωρίς επαναλήψεις σε γραμμές, στήλες και περιοχές.
- Μέσα από τον προγραμματισμό, μπορούμε να δημιουργήσουμε μια λειτουργική εκδοχή του.
- Η ανάπτυξη Sudoku μάς βοηθά να εξασκηθούμε σε έννοιες όπως πίνακες, έλεγχοι και επαναληπτικές δομές.

2	3	9	8	7	5	1		
4	5	6		9	2			7
	7	8	6	4				
7		1				3		8
	4	2	5	3		7		1
3			7	8		4	2	9
9	8		3		6	5	1	4
5		4	9	1			8	
6		3	4	5		9		

Ιστορική Αναδρομή του Sudoku

Το Sudoku εμφανίστηκε για πρώτη φορά τον 18ο αιώνα, με ρίζες σε λογικά παζλ.

Η μοντέρνα μορφή του διαδόθηκε στην Ιαπωνία το 1986 από την εφημερίδα Nikoli.

Το 2005 έγινε παγκόσμιο φαινόμενο μέσω εφημερίδων και εφαρμογών.

Objective & Scope

- **Κύριοι Στόχοι:**
 - Δημιουργία λειτουργικού παιχνιδιού Sudoku σε C++
 - Βελτίωση δεξιοτήτων χρήστη μέσω λογικής σκέψης
- **Δυνατότητες και Λειτουργίες:**
 - Εισαγωγή αριθμών από τον χρήστη
 - Έλεγχος εγκυρότητας κάθε κίνησης
 - Σύστημα βοήθειας (Hints)
 - Περιορισμένος αριθμός προσπαθειών (3 λάθη)
- **Περιορισμοί:**
 - Δεν περιλαμβάνει γραφικό περιβάλλον.

Τρόπος λειτουργίας του συστήματος

Η εφαρμογή είναι ένα παιχνίδι Sudoku με επιλογή δυσκολίας (εύκολο, μεσαίο, δύσκολο) και αυτόματη δημιουργία πίνακα 9x9. Ο χρήστης κερδίζει πόντους για σωστές απαντήσεις και χάνει για λάθη. Υπάρχει δυνατότητα δωρεάν βοήθειας, ενώ αν εξαντληθούν οι προσπάθειες και ο παίκτης έχει ≥ 5 πόντους, μπορεί να αγοράσει τρεις νέες. Το παιχνίδι ολοκληρώνεται με επιτυχία, αποτυχία ή έξοδο. Το διάγραμμα ροής βρίσκεται αριστερά στο MENU.

Τεχνολογίες που Χρησιμοποιήθηκαν

- **Γλώσσα:** C++
- **Βιβλιοθήκες:** Standard C++ Library `<vector>`, `<iostream>`, `<cstdlib>` κ.ά.
- **Εργαλείο Παρουσίασης:** LaTeX Beamer

Η συνάρτηση `playSudoku` διαχειρίζεται ολόκληρη τη ροή του παιχνιδιού Sudoku από την πλευρά του παίκτη. Παρέχει τη δυνατότητα εισαγωγής αριθμών, χρήσης βοήθειας (hint), ή αυτόματης συμπλήρωσης ενός κελιού (auto-fill).


```

void playSudoku(int board[SIZE][SIZE], int solutionBoard[SIZE][SIZE]) {
    int row, col, num;
    int mistakes=0, hintsLeft=2;

    displayBoard(board, mistakes, hintsLeft);

    while (!isBoardFull(board)) {

        cout << "\nEnter row (1-9), column (1-9), and number (1-9), or type 'h' for hint, 'a' for auto-fill, or '0 0 0' to exit: ";
        string input;
        cin >> ws;
        getline(cin, input);

        if (input == "h") {
            if (hintsLeft > 0) {
                giveHint(board, solutionBoard);
                hintsLeft--;
            } else {
                cout << "No hints left.\n";
            }
            displayBoard(board, mistakes, hintsLeft);
            continue;
        }
        else if (input == "a") {
            if (hintsLeft > 0) {
                autoFillOneCell(board, solutionBoard);
                hintsLeft--;
            } else {
                cout << "No auto-fills left.\n";
            }
            displayBoard(board, mistakes, hintsLeft);
            continue;
        }
        else {
            istringstream iss(input);
            if (!(iss >> row >> col >> num)) {
                cout << "Invalid input. Please enter either 'h', 'a', or 3 numbers.\n";
                continue;
            }
        }
    }
}

```

```

if (row == 0 && col == 0 && num == 0) {
    cout << "You exited the game.\n";
    break;
}

row--;
col--;

if (row >= 0 && row < SIZE && col >= 0 && col < SIZE && num >= 1 && num <= 9) {
    if (board[row][col] == 0) {
        GamePoints(board, solutionBoard, row, col, num);
        if (solutionBoard[row][col] == num) {
            board[row][col] = num;
            displayBoard(board, mistakes, hintsLeft);
        }
        else {
            mistakes++;
            cout << "Mistakes:" << mistakes << "/3\n";
            displayBoard(board, mistakes, hintsLeft);
            if (mistakes == 3) {
                bool bought = Tries(mistakes);
                cout << "Remaining chances: " << mistakes << "/3\n";
                displayBoard(board, mistakes, hintsLeft);
                if (!bought) { break; }
            }
        }
    }
    else {
        cout << "Cell already filled.\n";
    }
}
else {
    cout << "Invalid input. Try again.\n";
}
}

if (isBoardFull(board))
    cout << "Congratulations! You've completed the Sudoku!\n";
cout << "\nSolution Board:\n";
displayBoard(solutionBoard, mistakes, hintsLeft);

```

Δείτε το βίντεο παρουσίασης του Sudoku demo εδώ: [Πατήστε εδώ για το demo](#)

Το demo περιλαμβάνει τα εξής βίντεο (αριθμημένα από 1 έως 7):

- 1) Βίντεο με το Μενού
- 2) Επίπεδα δυσκολίας: *Easy, Medium, Hard*
- 3) Χρήση της λειτουργίας *Hint*
- 4) Χρήση της λειτουργίας *Auto-fill*
- 5) Αντιμετώπιση *mistakes* (με αρκετούς πόντους για αγορά ευκαιριών)
- 6) Αντιμετώπιση *mistakes* (χωρίς αρκετούς πόντους)
- 7) Έξοδος από το παιχνίδι

Σύγκριση με AI-generated Code

- Κατά την ανάπτυξη του έργου, αξιοποιήθηκε η τεχνητή νοημοσύνη (π.χ. ChatGPT) για την κατανόηση εργαλείων όπως το GitHub και το Overleaf.
- Παρείχε υποστήριξη στην υλοποίηση ιδεών, όπως η κατανόηση συναρτήσεων και η συγγραφή παραδειγμάτων κώδικα.
- Η συμβολή της AI υπήρξε υποστηρικτική και ενίσχυσε τη διαδικασία επίλυσης προβλημάτων, χωρίς να υποκαταστήσει την ανθρώπινη σκέψη και δημιουργικότητα.

- **Τι μάθαμε από το project;**
 - Πώς να σχεδιάζουμε μια διαδραστική εφαρμογή που ανταποκρίνεται σε επιλογές του χρήστη σε πραγματικό χρόνο.
 - Πώς να συνδυάζουμε πολλαπλές λειτουργίες (π.χ. hints, σύστημα πόντων, έλεγχος δυσκολίας) σε ένα ενιαίο και λειτουργικό παιχνίδι.
 - Εξοικείωση με δομές δεδομένων όπως πίνακες και vector, καθώς και αλγόριθμους όπως το `random_shuffle`.

- **Τι θα μπορούσε να βελτιωθεί;**
 - Καλύτερος έλεγχος εγκυρότητας εισόδου από τον χρήστη, με λιγότερο επαναλαμβανόμενο κώδικα.
 - Ο διαχωρισμός της λογικής του παιχνιδιού από την παρουσίαση (modularization) για πιο καθαρό και επαναχρησιμοποιήσιμο κώδικα.
 - Εμφάνιση του πίνακα σε πιο γραφικό περιβάλλον (π.χ. GUI με SFML ή Qt).