Курсов проект №10 Игра Судоку

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Изготвил: Георги Станчев, трети курс, първа група, факултетен номер: 71959  
Специалност: Информационни системи*

Съдържание

[1. Описание и идея на проекта 3](#_Toc96168251)

[2. Основни дефиниции, концепции и алгоритми 3](#_Toc96168252)

[3. Обща архитектура – ООП дизайн 3](#_Toc96168253)

[4. Реализация на класове 5](#_Toc96168254)

[5. Представяне на приложението чрез блок схема 7](#_Toc96168255)

[6. Планиране, описание и създаване на тестови сценарии 8](#_Toc96168256)

[7. Инструкции за компилиране 34](#_Toc96168257)

[8. Обобщение на изпълнението 34](#_Toc96168258)

[9. Насоки за бъдещо развитие и усъвършенстване 34](#_Toc96168259)

[10. Използвана литература 34](#_Toc96168260)

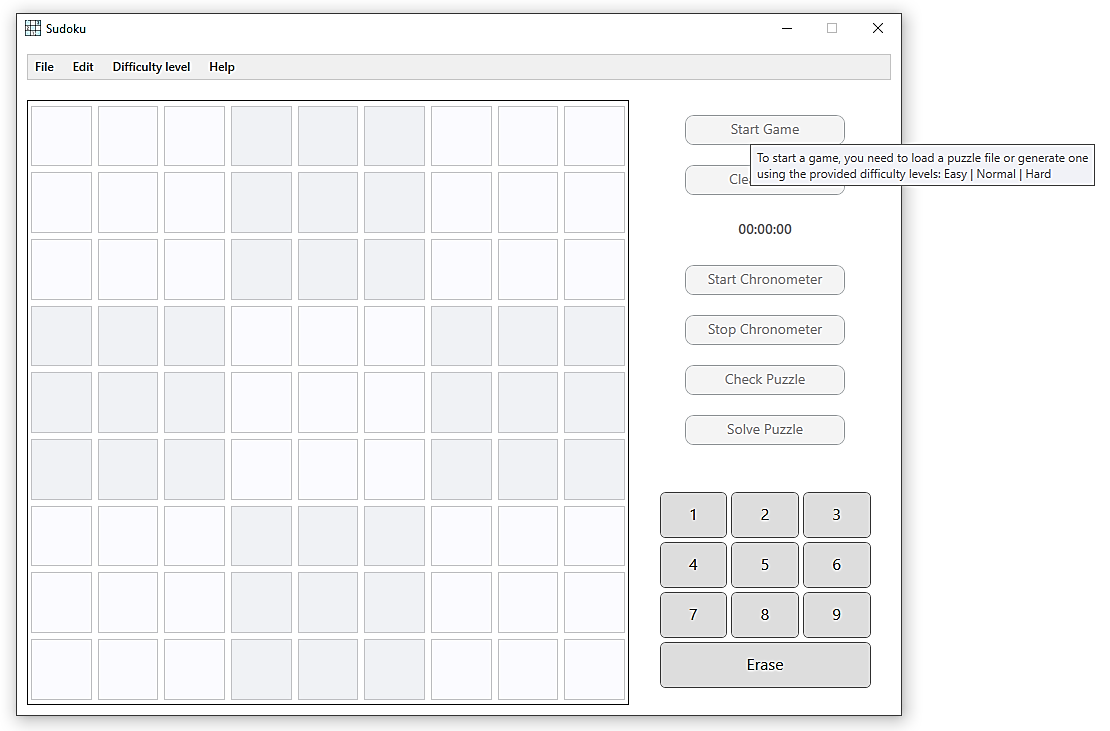
# 1. Описание и идея на проекта

Да се напише програма, по-конкретно WPF приложение, реализираща популярната японска игра Судоку. Судоку пъзелите са 9 x 9 мрежи, които се състоят от 9 по-малки квадрата, като всеки един от тях представлява подмрежа 3 x 3, наречена минимрежа. Целта е да се попълнят всички квадратчета, така че всяка колона, ред и минимрежа да съдържа цифрите от 1 до 9 точно веднъж. Това са и основните правила на играта.

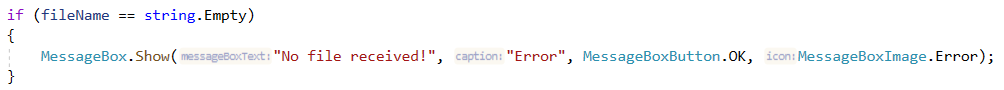
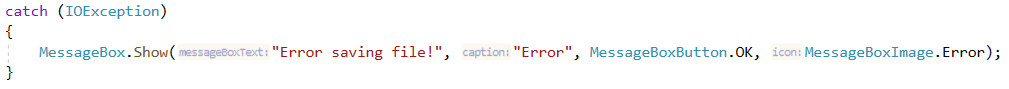
# 2. Основни дефиниции, концепции и алгоритми

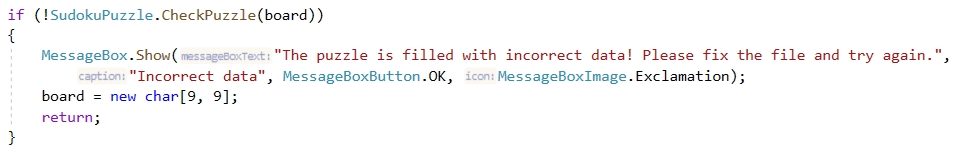
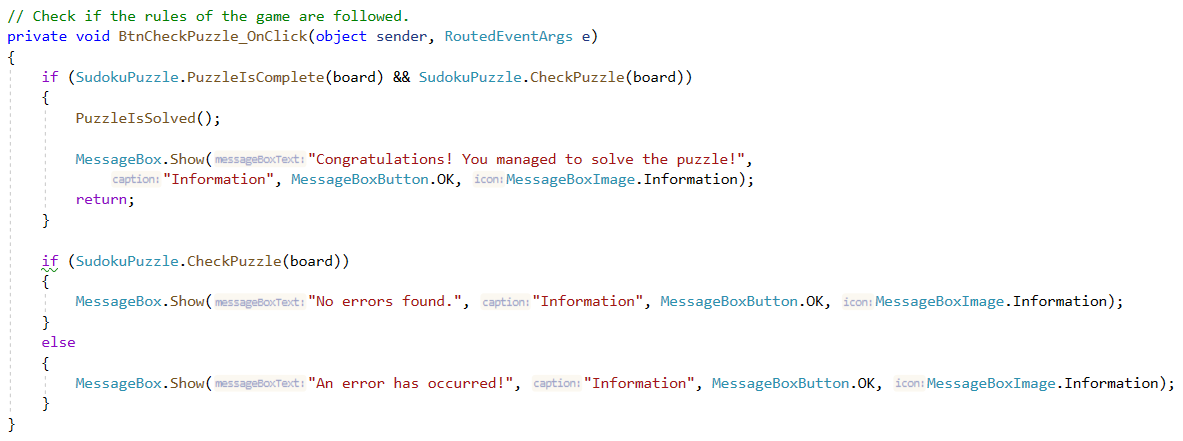
Програмата е реализирана на Visual Studio 2019 , C#.NET. Основната идея, по която е разработена, е играта Судоку. Разработен е подходящ потребителски интерфейс, благодарение на който потребителят може да си взаимодейства успешно с приложението. Изгледа се състои от меню бар в горната част на прозореца, мрежа, която представлява Судоку пъзелът, състояща се от 9 х 9 клетки, бутони за въвеждане на цифрите и изпълнение на различни операции, които ще бъдат разгледани по-подробно на по-късен етап. Имплементирани са алгоритми за решаване на пъзела и за неговото валидиране.

# 3. Обща архитектура – ООП дизайн

Архитектурата на програмата е внимателно моделирана, вземайки се предвид и най-малките детайли. Създадени са нужните класове за поддържането на функционалността, а изгледа е достатъчно улеснен с цел подсказване на потребителя по какъв начин да работи с приложението бързо и лесно.

Както се вижда от снимката, не всички бутони са активни. Приложението е изградено по следния начин с цел потребителя винаги да знае какво може да се натиска и какво не, тъй като за него трябва да бъде ясно кои бутони имат значение и по-конкретно смисъл да бъдат използвани по време на различните етапи от играта.

Също така са предвидени всички случаи, в които може да възникне грешка по време на работа с файлове, както и вариантите описващи невалидно съдържание в тях.

Осигурени са и подходящи съобщения в процес на играта, за да може играчът винаги да е наясно какво се случва и дали прогреса му в решението на пъзела е според правилата.

# 4. Реализация на класове

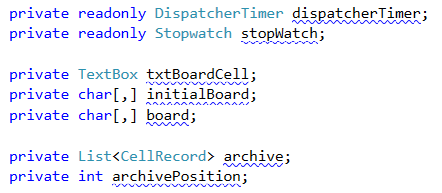
Дойде време да опишем с няколко думи имплементираните класове в приложението и нека първо започнем с класът DifficultyLevel. Както се подразбира от името, той представлява изброим тип и определя различните нива на трудност по време на игра, като самата трудност се изразява в броя на непопълнените клетки в мрежата. Възможностите са три на брой, а именно лесна, нормална и трудна игра, като съответно броя на празни клетки е 30, 45 и 60.

Следващият клас, който ще опишем е CellRecord. Той представлява извършената промяна на клетка в мрежата и съхранява в себе си като данни редът и колоната на клетката, както и старият и новият символ. Сега се появява и въпросът каква стойност приема старият символ, ако клетката преди това е била празна. Отговорът на този въпрос е 0. По този начин се записват празните клетки в матрицата от символи, с която се работи извън полезрението на играчът. Вместо 0 в мрежата се изобразява непопълнено текстово поле. И като последни думи за класът, той е създаден с цел чрез него да се попълва списък от действията върху цялата мрежа, за да може да се реализира функционалност за връщане назад по стъпки.

SudokuGenerator класът отговаря за генерирането на пъзел на случен принцип според зададено ниво на трудност. По-конкретно на случаен принцип се запълват 3х3 минимрежите по главния диагонал на мрежата, а след това пъзелът се решава от програмата. След като приключи процеса на откриване на решение, отново на случаен принцип се премахват стойностите на толкова клетки, колкото нивото на трудност определя.

Не трябва да пропускаме да представим и един от най-важните класове, а именно класът SudokuPuzzle. Благодарение на него може да се проверява дали един пъзел е попълнен правилно, като се обхождат редовете, колоните и минимрежите. Чрез него още може и да се провери дали пъзелът е попълнен изцяло, а другата дефинирана функционалност, която предоставя е решаването на пъзела, което се случва с помощта на още една допълнително имплементирана функция, която проверява дали един символ е коректно да стои в дадена клетка, проверявайки дали нарушава правилата на играта.

Остава да се опише и последния клас в реализираната Судоку игра и това е класът MainWindow. Нека първо започнем с какви данни съхранява в себе си, а след това вече ще опишем и какво прави всеки един бутон от приложението.



Променливите dispatcherTimer и stopWatch служат за измерването на времето, през което потребителя решава пъзела, като неговото обновяване се случва на всяка една секунда.

Текстовата кутия txtBoardCell е създадена, за да може чрез нея да достъпваме всяко едно текстово поле в мрежата, която представлява пъзела. Тя приема стойността на последно кликнатата кутийка чрез свойството LostFocus и съответно TxtBoardCell\_OnLostFocus.

Матриците (9х9) initialBoard и board в началото пазят в себе си еднакви данни, като единствената разлика е, че каквито промени настъпват по време на решаване на пъзела, те се отразяват само в матрицата board. С други думи това копие на началната матрица се пази непокътнато, за да може ако играча има напредък, но не в правилната посока откъм достигане на коректното решение, пъзелът се решава благодарение на тази матрица.

Променливите archive и archivePosition са добавени, за да може човек, когато иска да се върне назад стъпка по стъпка, да може да го постигне безпроблемно.

Дойде и моментът, в който ще опишем същественото от действията на всеки един бутон, като за начало ще започнем с меню бара, разположен в горната част на приложението:

**File → Open:** Зарежда пъзел в приложението. Важно е да се отбележи, че могат да бъдат прочитани файлове с различни формати на пъзела, но по-подробно тази функционалност ще бъде показана при примерните тестове.

**File → Save:** Запазва пъзел във файл, като може или да създаде нов, или да пренапише, ако вече има създаден такъв със същото име, с което потребителя иска да съхрани играта си.

**File → Exit:** Изключване на приложението.

**Edit → Undo:** Връщане една стъпка назад, ако съществува някаква промяна на стойността на клетка в мрежата.

**Edit → Redo:** Връщане една стъпка напред, ако преди това се е състояла стъпка назад и съответно има някаква промяна, която да върнем обратно.

**Difficulty level → Easy:** Генериране на пъзел, който е лесен за решаване (30 празни клетки).

**Difficulty level → Normal:** Генериране на пъзел, който е на средно ниво относно трудността на решаване (45 празни клетки).

**Difficulty level → Hard:** Генериране на пъзел, който е труден за решаване (60 празни клетки)

**Help → About:** Предоставя информация за правилата на играта.

Остава да представим и бутоните, които са свързани с процеса на решаването на пъзела:

**Start Game:** Започване на играта, като едновременно с това се стартира и хронометър, за да се измерва времето, през което протича размишляването върху пъзела.

**Clear Board:** Изчиства съхраняваните данни заедно с дъската и рестартира хронометъра.

**Start Chronometer:** Стартира хронометъра, ако преди това е бил спрян на пауза.

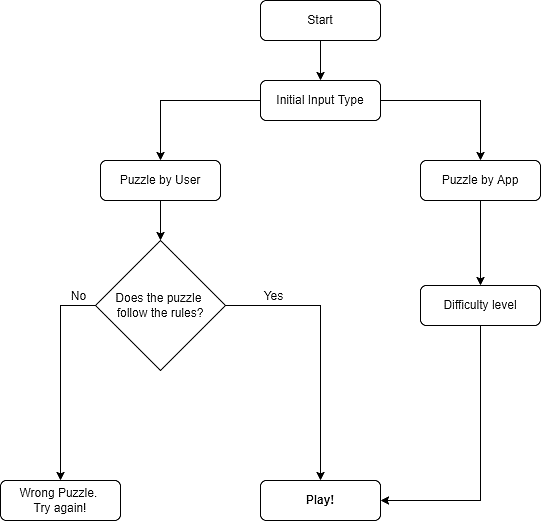
**Stop Chronometer:** Спира хронометъра на пауза, ако преди това е работил.

**Check Puzzle:** Проверява дали пъзела е валиден. При откриване на нарушение в правилата се изкарва грешка, а ако всичко е наред се съобщава, че няма проблем. При попълване на целия пъзел и при условие, че всички правила са спазени, се изразяват поздравления.

**Solve Puzzle:** Решава пъзела, дори и да са допуснати грешки от играча. При достигане на крайното състояние на пъзела, повече не могат да се правят никакви промени.

Чрез останалите бутони, които всъщност са цифрите от **1 до 9** и бутон за **изчистване на стойност** в клетка, се управлява съдържанието на текстовите кутийки в мрежата.

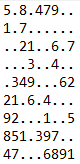
# 5. Представяне на приложението чрез блок схема

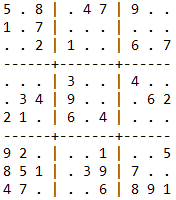
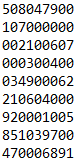
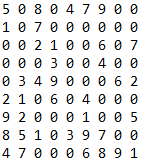


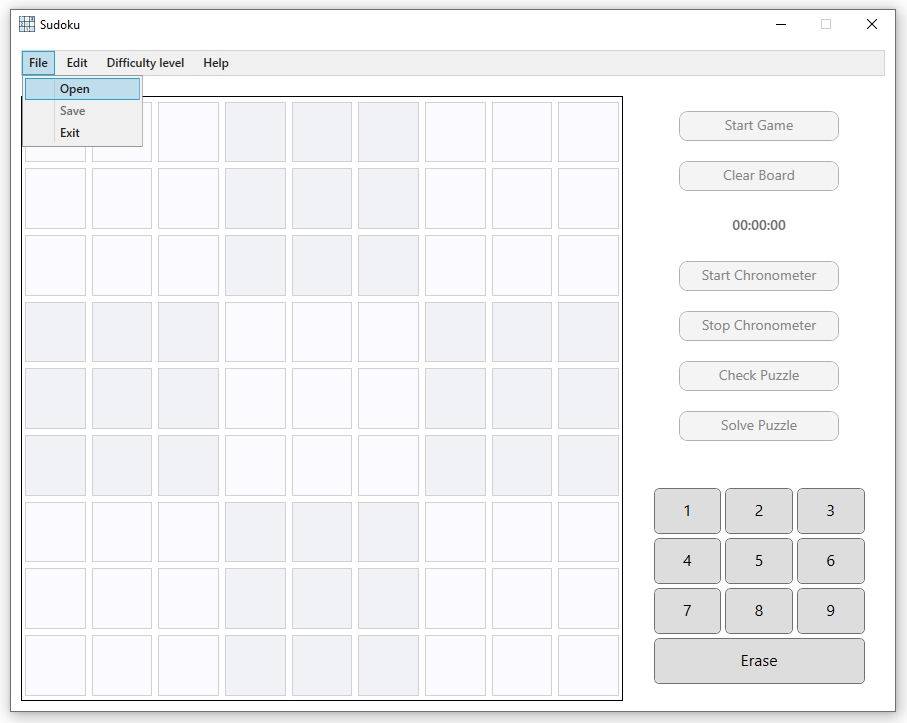
# 6. Планиране, описание и създаване на тестови сценарии

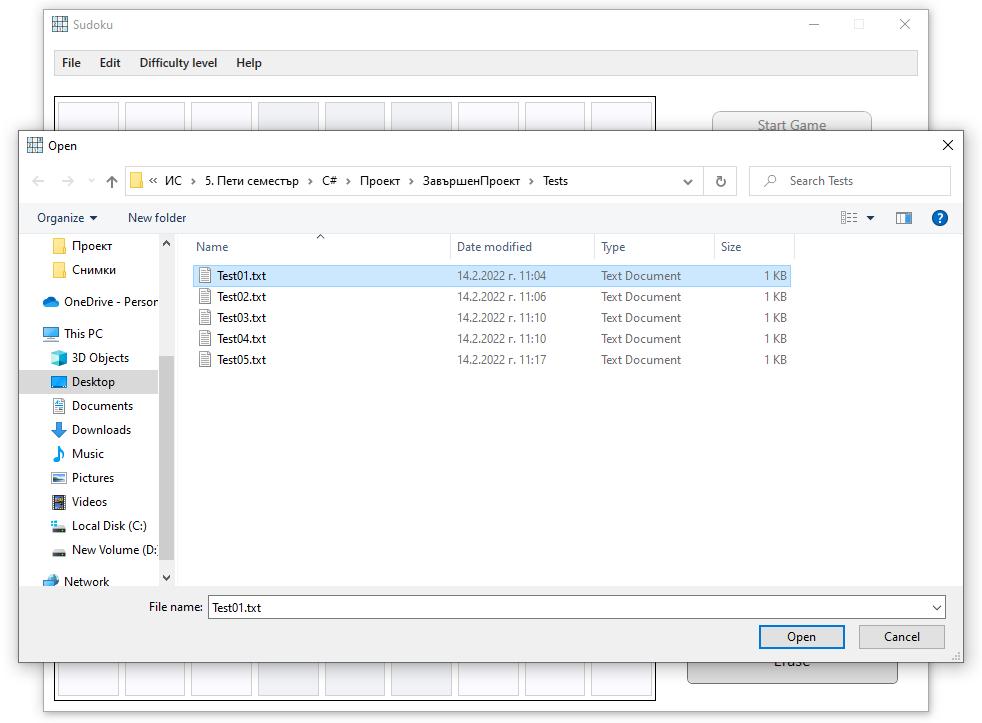
За постигането на успешна реализация на играта е необходимо много добре да се планират стъпките и дейностите в етапите на тестване. С цел проверка на постигнатата разработка на приложението с възложените критерии по заданието, е необходимо да се изпълняват минимум следните функционални тестови сценарии, които са представени в таблицата малко по-долу. Още по време на създаването на всяка една функционалност са направени нужните тестове, за да се потвърди тяхното коректно изпълнение. Не винаги се получаваше от първия път, но за сметка на това се раждаха нови идеи, благодарение на които решението се усъвършенстваше и ставаше все по-добро.

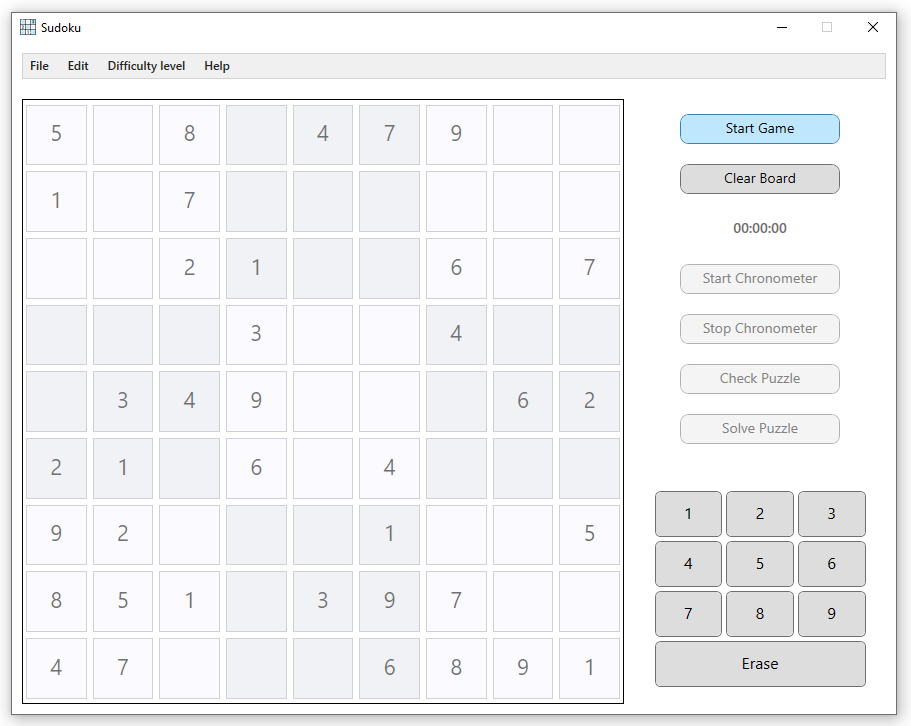
|  |  |
| --- | --- |
| Изискване | Реализация |
| Load and save Sudoku puzzles | Да |
| Provide a menu option to solve the current Sudoku puzzle | Да |
| Don’t allow invalid numbers to be placed in a cell | Да |
| Check whether any loaded Sudoku puzzle has been solved | Не |
| Keep track of the time needed to solve a Sudoku puzzle | Да |
| Undo and redo unlimited previous moves | Да |
| Allow different modes of difficulty for playing the game | Да |

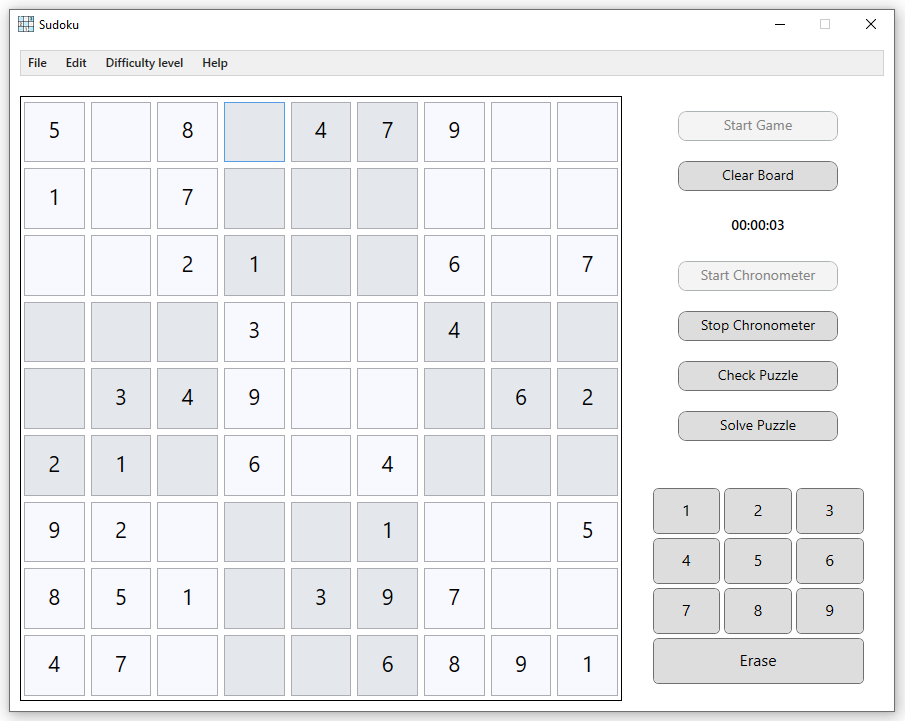
Нека сега покажем и какви различни версии на пъзели може да зарежда приложението, а след това вече ще преминем към демонстрация на основните му функционалности. Трябва и да се отбележи, че в тестовете точката (.) и нулата (0) са взаимнозаменяеми и по този начин се декларира, че на тази позиция няма цифра и съответно се показва, че това е празна клетка.

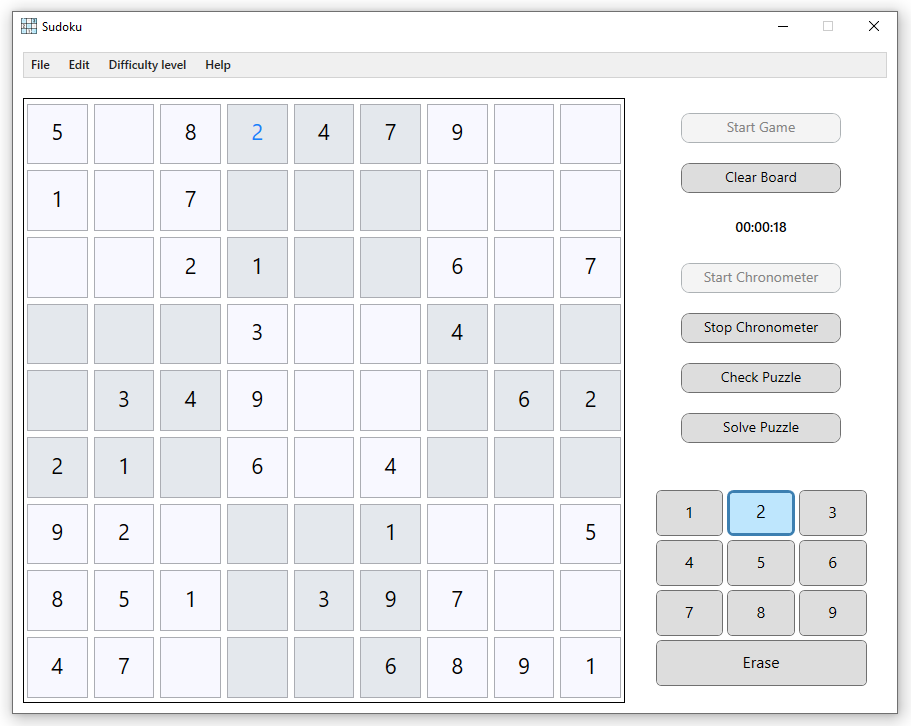


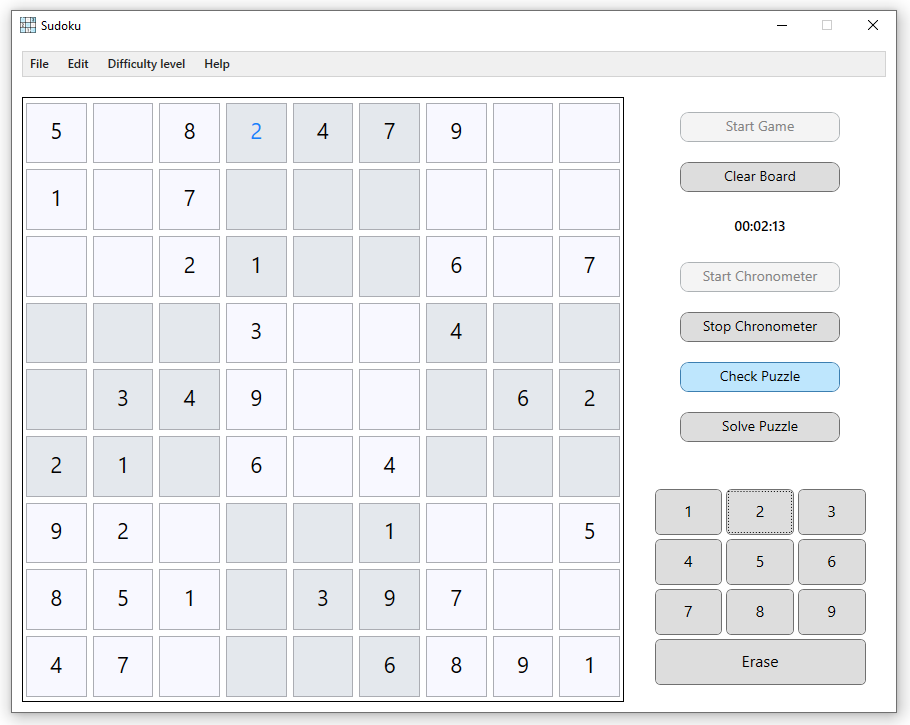
Като изложение на свършената работа върху заданието, ще започнем първо с демонстрация на програмата при зареждане на пъзел от файл, а след това при генериране на пъзел от приложението в зависимост от избраното ниво на трудност.

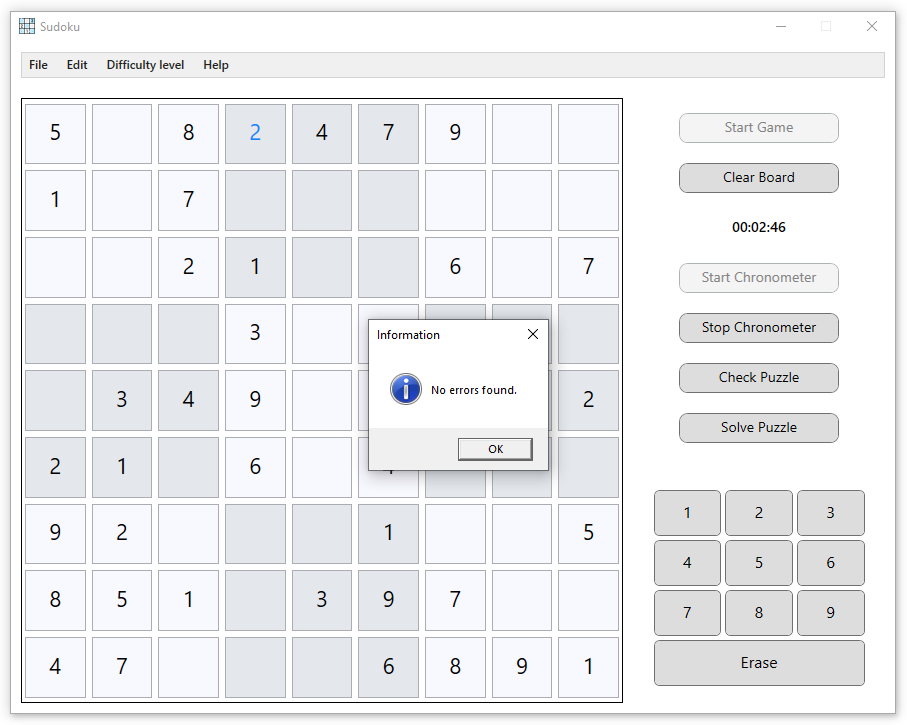


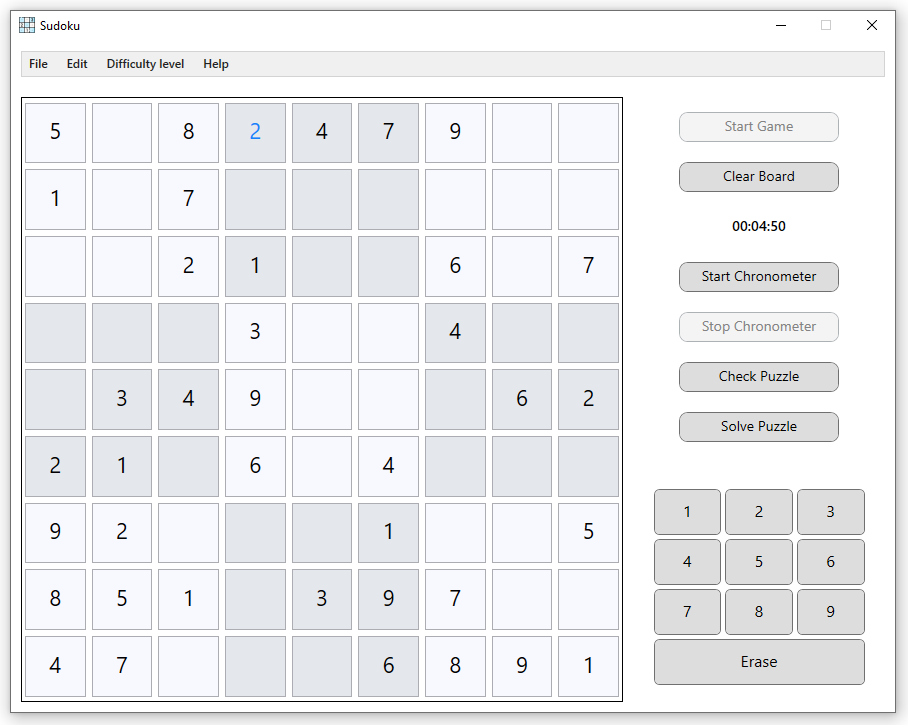


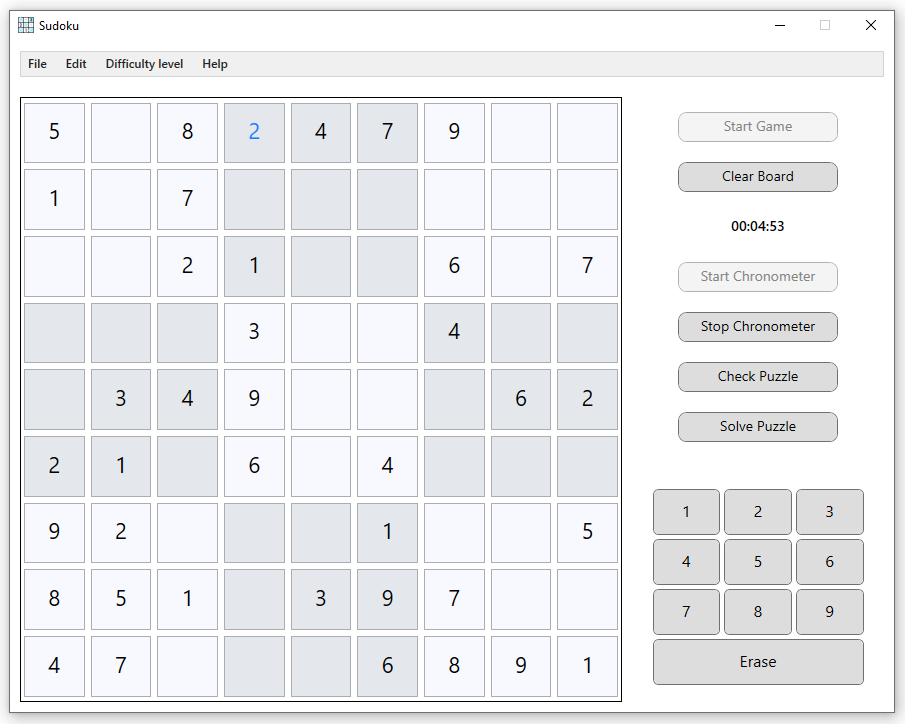


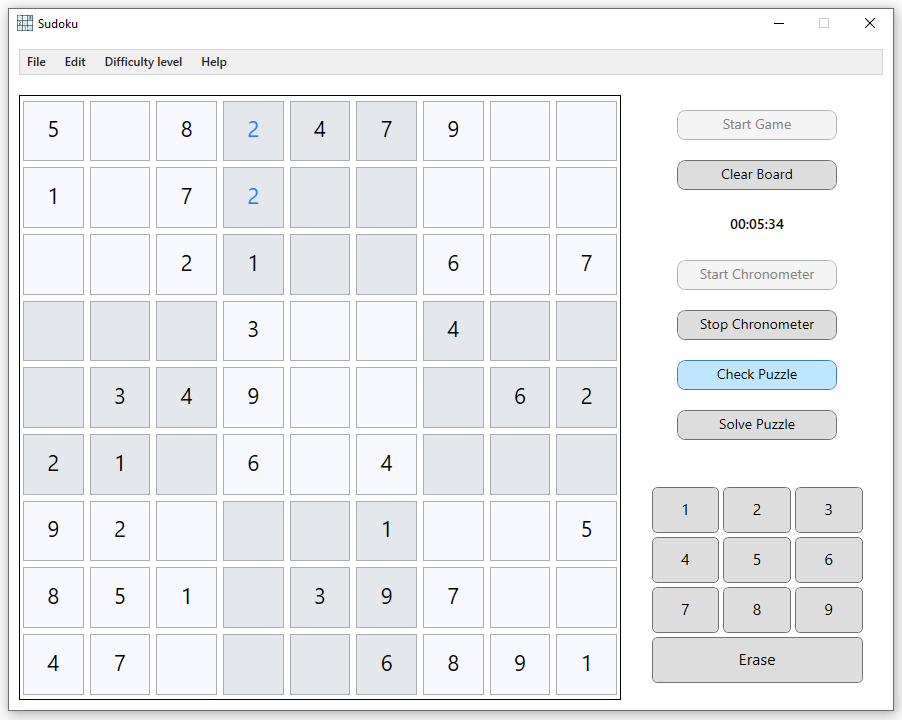


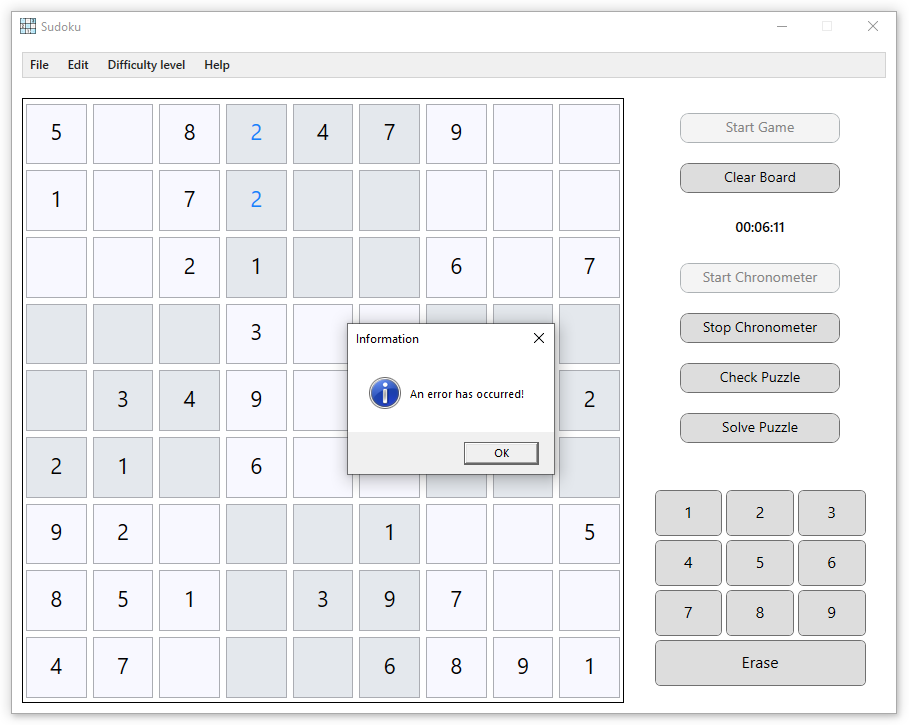


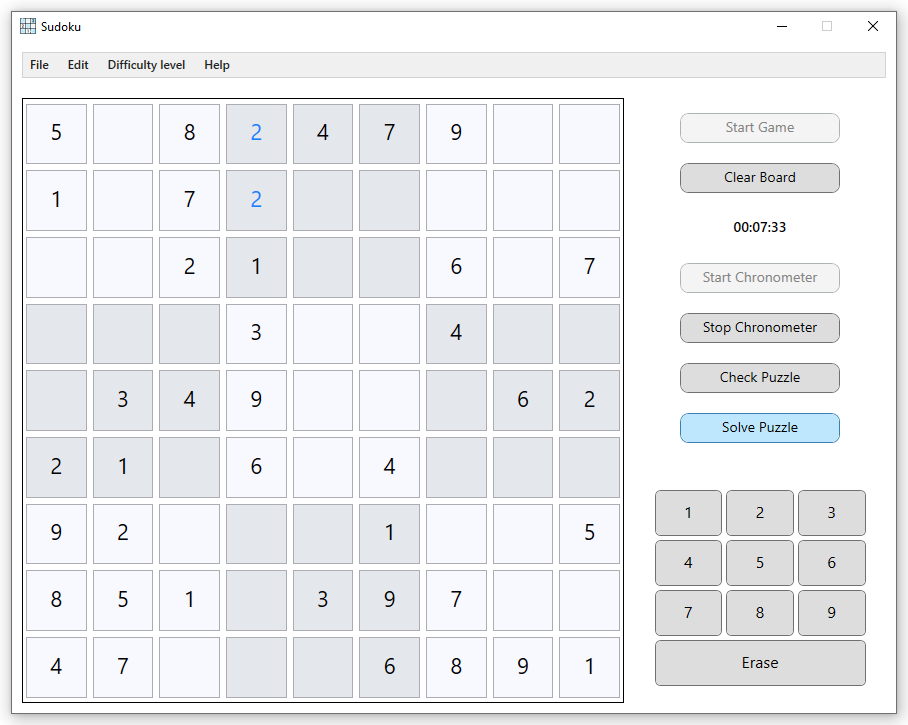


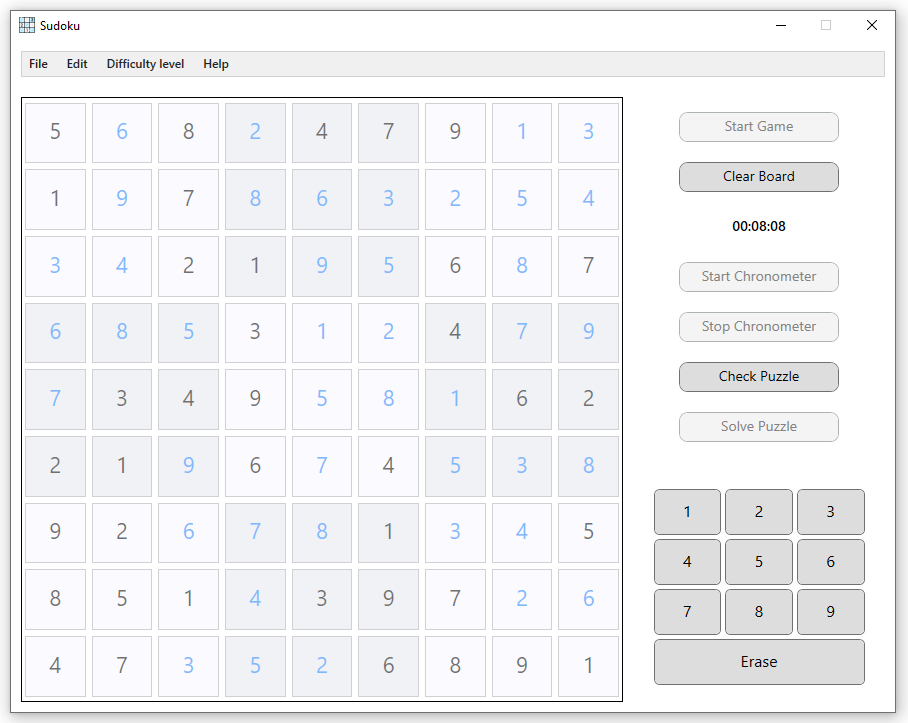


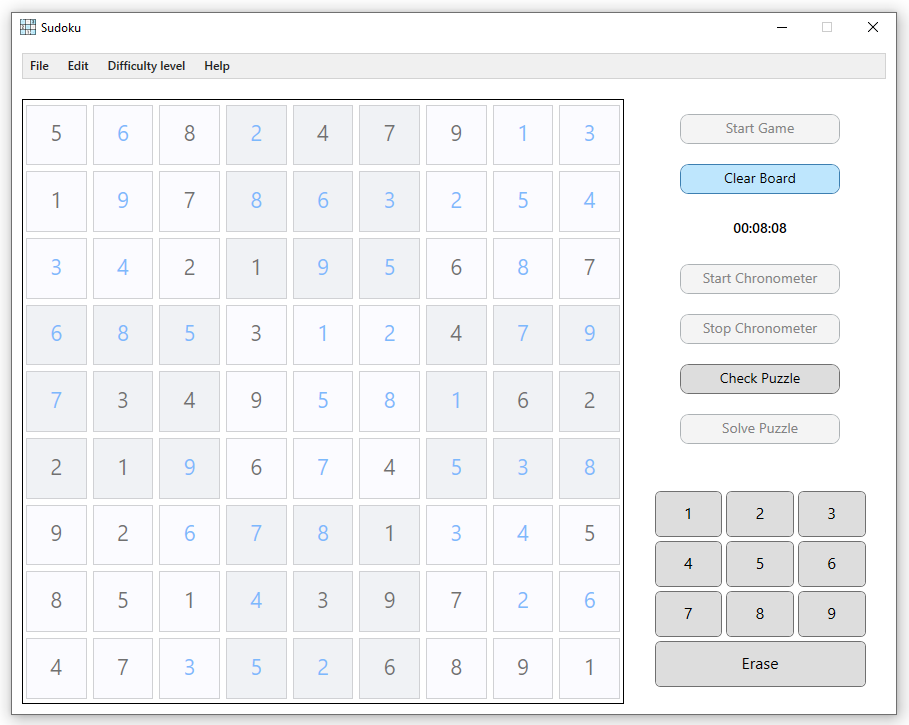


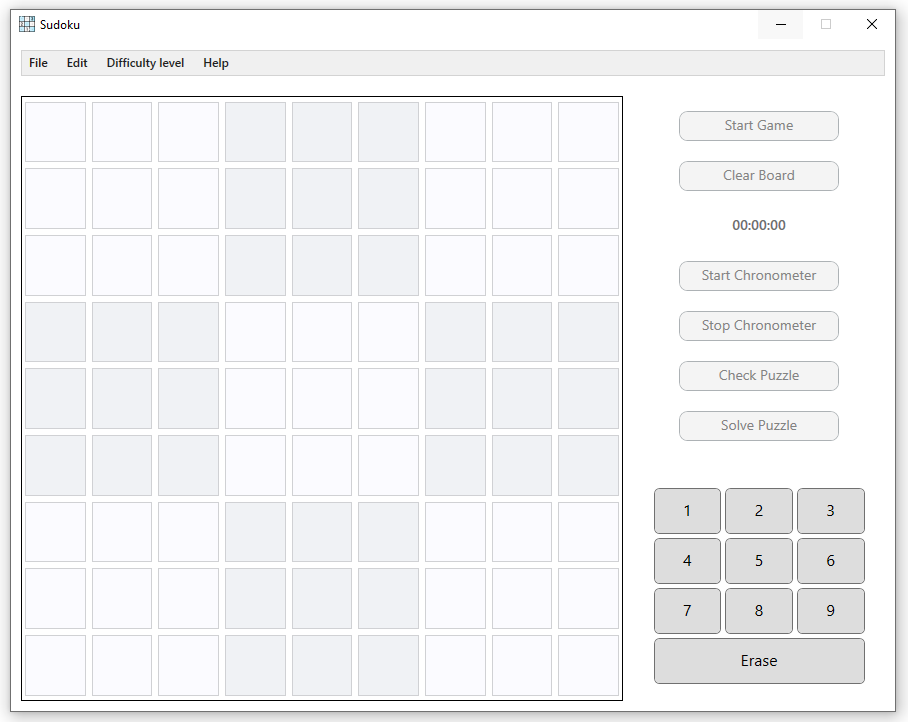


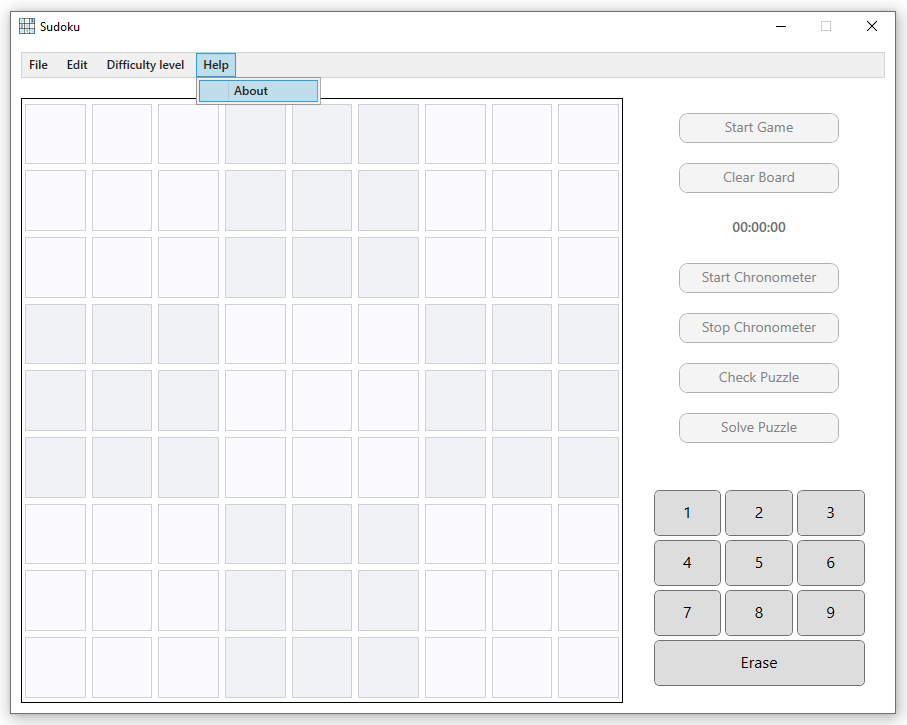


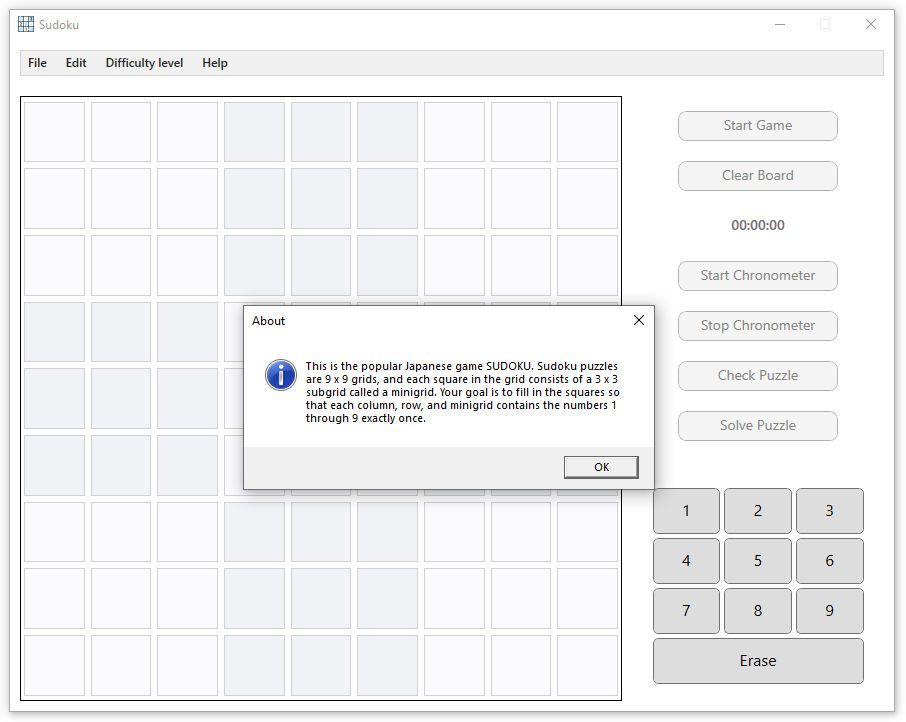


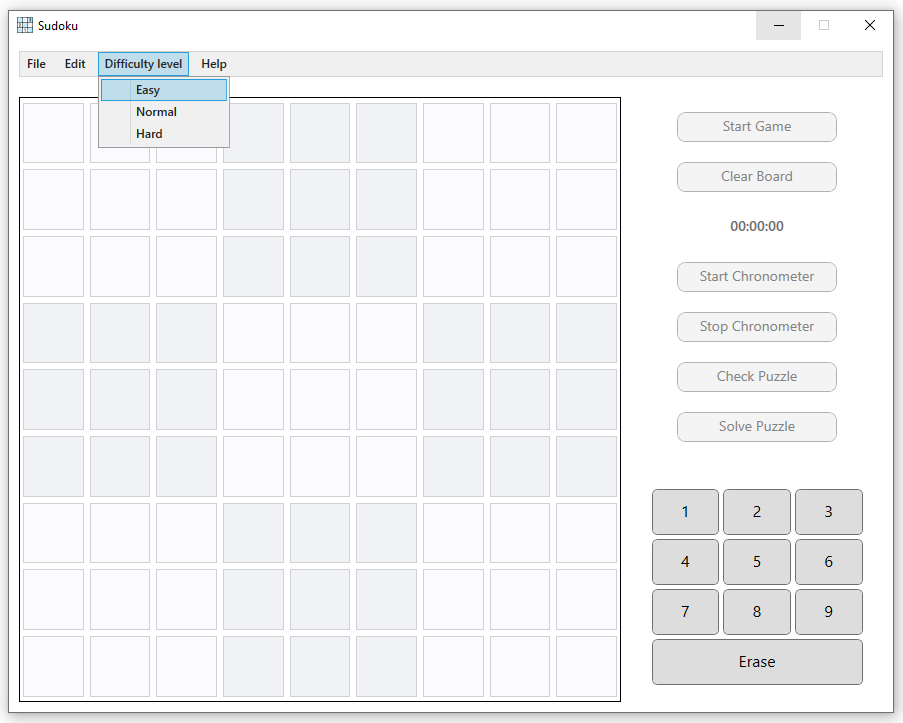


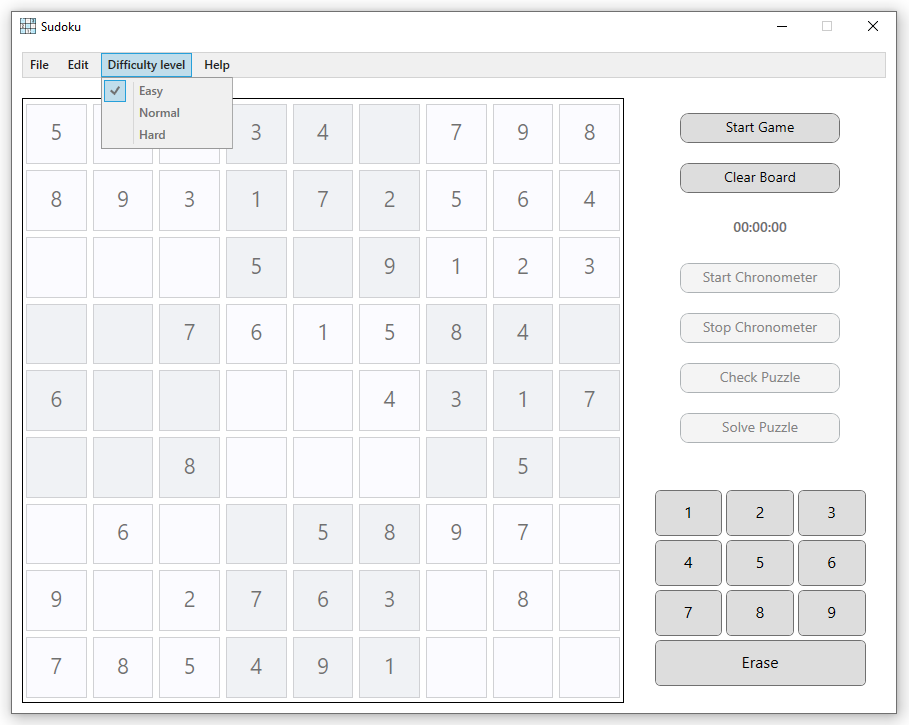


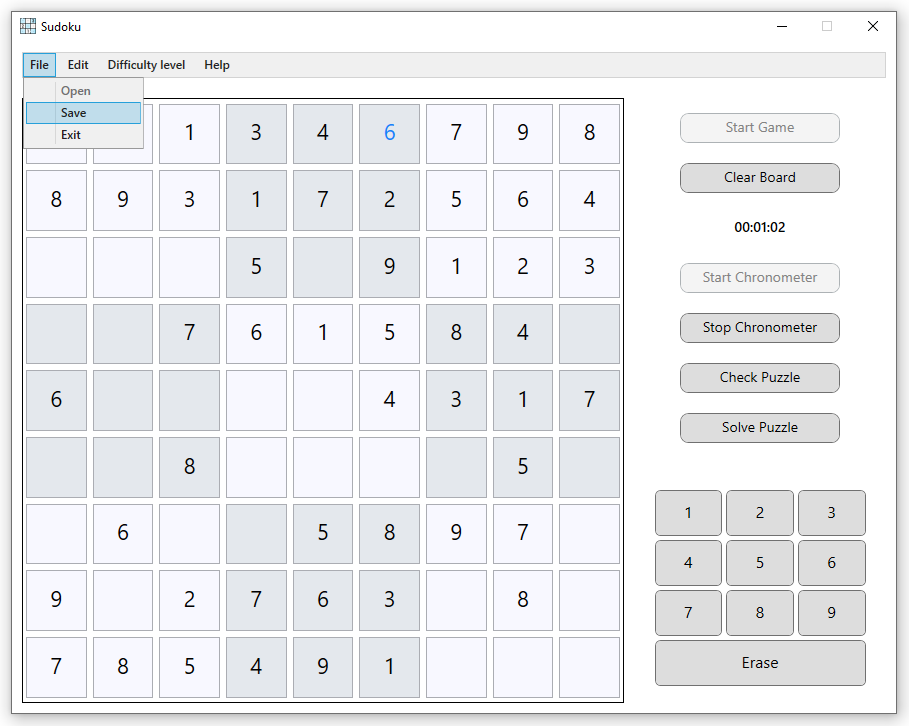


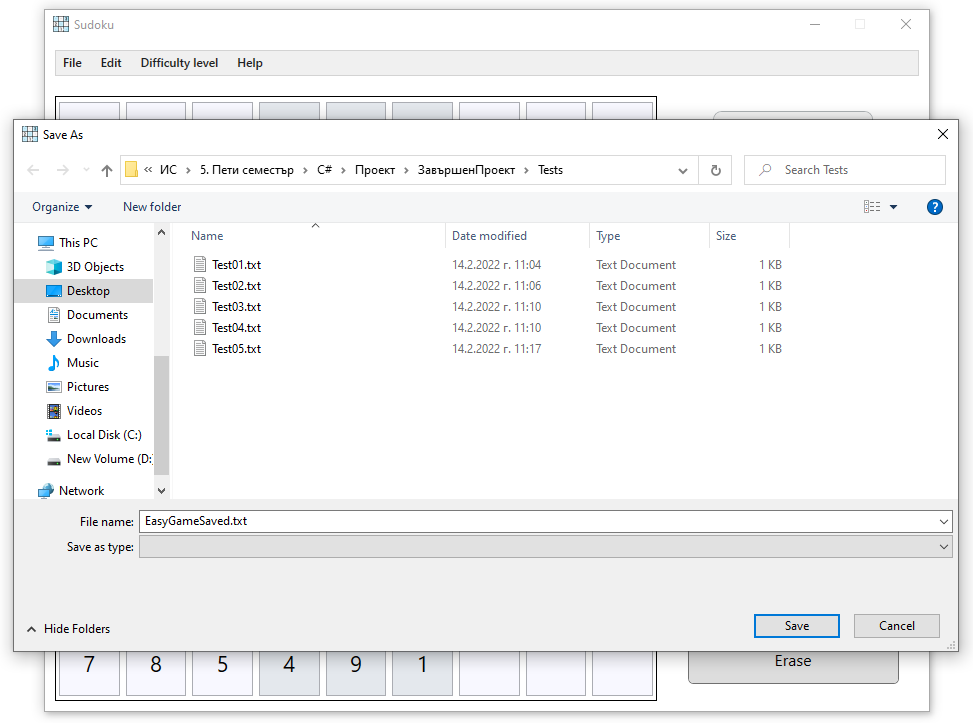


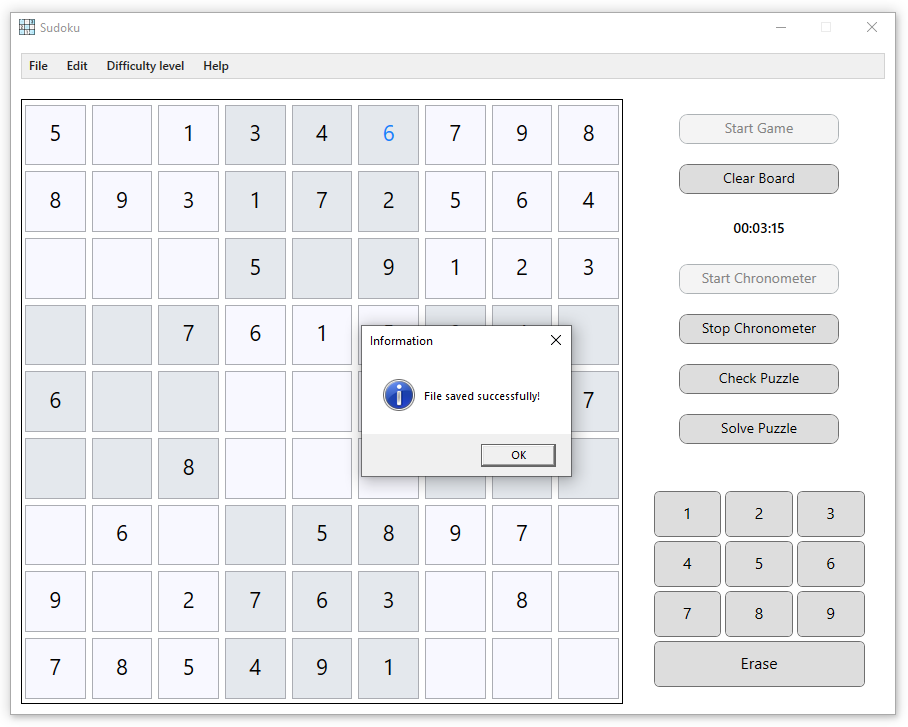


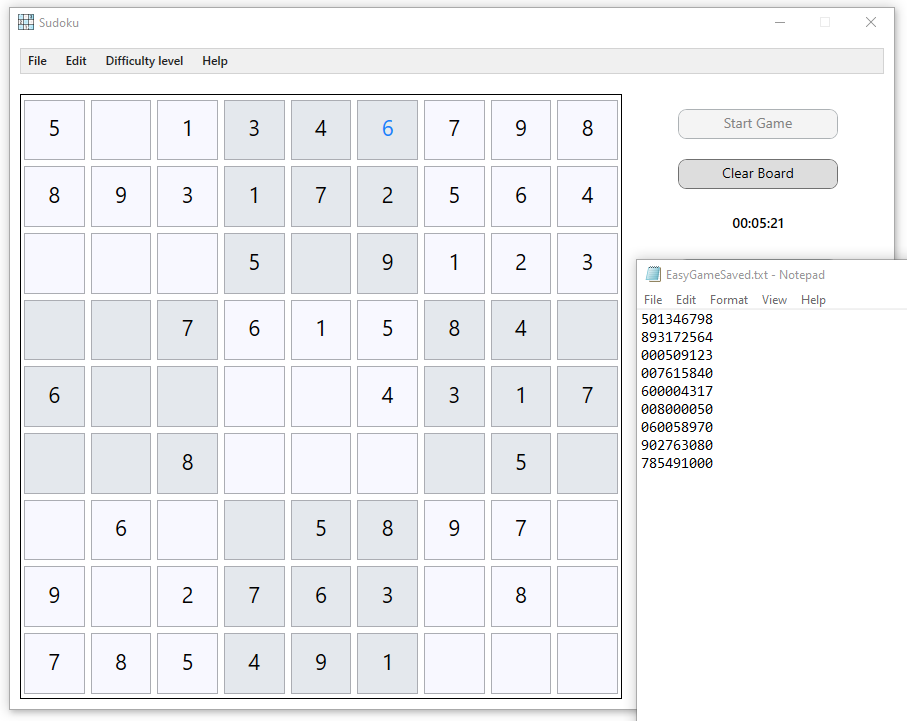


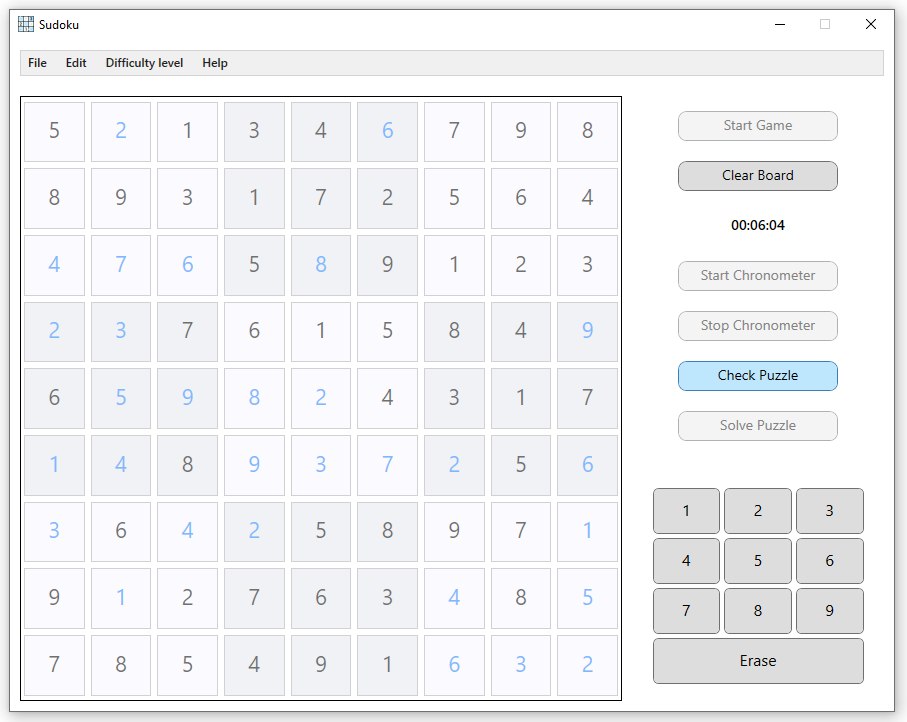


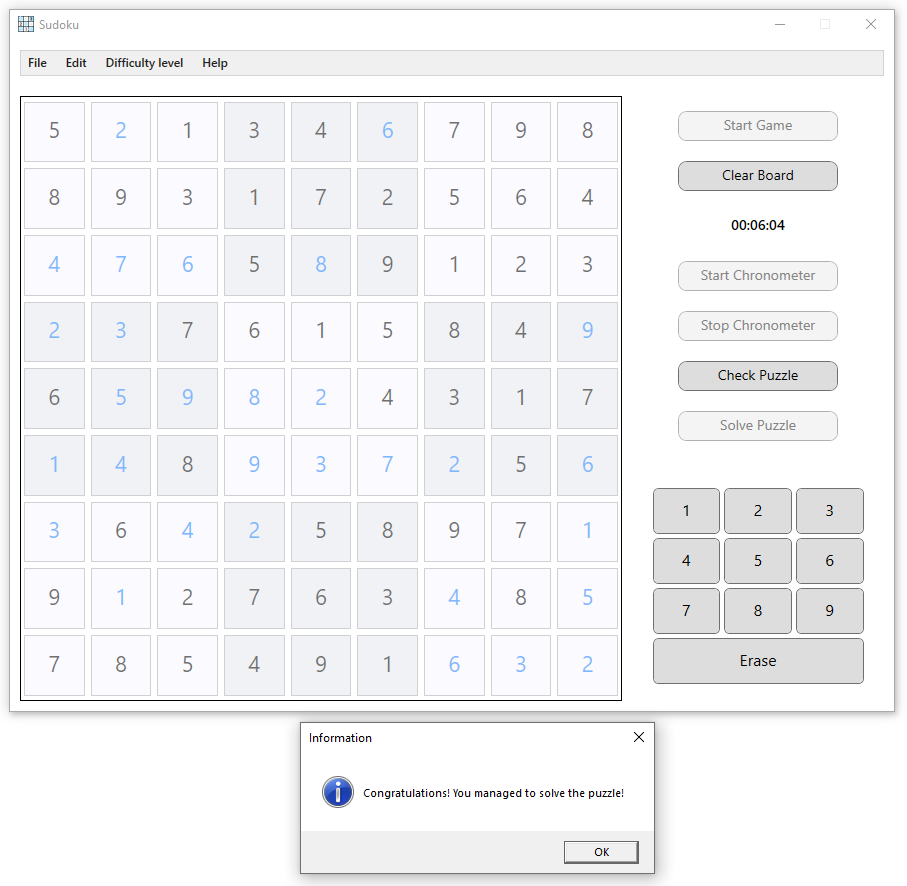












# 7. Инструкции за компилиране

На Visual Studio проекта може да се стартира чрез натискане на клавишната комбинация: Ctrl + F5. Друг възможен вариант за стартиране е с помощта на .exe файла, чиято локация е: CourseProject\SudokuGame\SudokuGameApp\bin\Debug\SudokuGameApp.exe

Както може да се подразбере от тестовите примери, има създадена папка Tests, в която се намират няколко текстови файла, всеки един от които с различна версия на пъзел. На по-късен етап са добавени и файлове, които не са валидни и приложението ще ги отхвърли.

# 8. Обобщение на изпълнението

Като цяло запознаването отново с тази дългогодишна игра и размишляването върху подходящи алгоритми за нейното правилно функциониране беше доста интересно. Съществуват различни начини за намиране на решение на един Судоку пъзел, както и разнообразни начини за генериране на такъв, но се надявам, че съм успял да реализирам достатъчно добър, оптимизиран и лесен за разбиране. По време на реализирането на програмата се появяваха малки пречки, но за сметка на това решението се подобряваше. Един от проблемите, с които се сблъсках беше още в началото и се изявяваше в това по какъв начин да реализирам функционалността чрез бутон да може да се вписва определен символ в последно кликнатото текстово поле от курсора на мишката, но благодарение на документацията на Майкрософт бързо се намери решение. И в заключение трябва да се отбележи, че процеса на реализация беше динамичен, но все пак и много полезен.

# 9. Насоки за бъдещо развитие и усъвършенстване

Като първо усъвършенстване може да се реализира при всяко въвеждане и съответно промяна на стойност в непопълнена клетка да се проверява дали всички правила са в сила. Едновременно с това може да се добави и функционалността за всяка клетка програмата да изчислява възможните цифри, които могат да се поставят, но това вече е като голямо улеснение за играча. Предполагам, че и още едно подобрение няма да се отрази лошо, а по-конкретно добавяне на бутон „Подсказка“, благодарение на който потребителя ще може след като е кликнал на произволна празна клетка да достигне бързо до търсената стойност.

# 10. Използвана литература

Проекта е разработен с помощта на материалите, които са публикувани в курса „Обектно-ориентирано програмиране с C#.NET (2021)“. Използвана на моменти е и документацията на Майкрософт с цел припомняне по какъв начин се използват някои свойства, свързани с имплементацията на графичния потребителски интерфейс на играта.