**Project Report**

**On**

**Sudoku Solver**

**Submitted in partial fulfillment for the requirement of the**

**award of TRAINING**

**IN**

**DATA STRUCTURES THROUGH C++**



***Submitted By***

**Anuj Kamath**

**(Manipal Institute of Technology, Manipal, Karnataka)**

***Under the guidance of***

**Akhilesh Sirohi**

**Acknowledgment**

I wish to express my sincere gratitude for the invaluable internship I undertook at Diginique Tech Labs. This experience offered a remarkable avenue for personal and professional growth, an opportunity I deeply cherish. Being chosen for this journey has left me feeling exceptionally fortunate. The experience not only allowed me to expand my knowledge but also enabled me to connect with a diverse group of outstanding individuals and experts who provided guidance throughout my time as an intern.

Considering the aforementioned, I would like to convey my heartfelt thanks and special recognition to my mentor, Mr. **Akhilesh Sirohi**. Despite his demanding responsibilities, he generously devoted time to offer guidance, unwavering support, and valuable insights. His involvement played a pivotal role in steering me in the right direction as I worked on my project within their esteemed organization and progressed through my training.

I view this experience as a significant marker in my professional journey, a stepping stone towards realizing my career aspirations. I am fully committed to applying the skills and knowledge I acquired in the most effective manner and am dedicated to their continuous refinement. With a positive outlook, I eagerly anticipate the possibility of future collaborations and partnerships with each of you.

Sincerely,

Anuj Kamath.

**Introduction**

The objective of the "C++ is Enjoyable" series is to demonstrate that creating code in C++ can be an enjoyable and productive experience, akin to well-established programming languages. In this second release, I will lead you through the journey of constructing a tic-tac-toe game entirely in C++. This documentation for the project, as well as the complete series, is designed for individuals seeking to learn C++ or those who are curious about the language's capabilities.

Many young individuals aim to learn programming in order to design video games. C++ is notably the prominent choice for game development, although a considerable amount of programming knowledge is necessary prior to attaining the ability to create games akin to Angry Birds. Commencing with a Sudoku Solver serves as a constructive starting point, nurturing a strong comprehension of recursion methodologies. I have confidence that this content can offer advantages to both beginners and adept developers striving to enhance their comprehension of C++, regardless of their existing level of familiarity.

**Abstract**

The Sudoku Solver project involves the design and implementation of an algorithmic solution to solve Sudoku puzzles efficiently. The primary goal of this project is to develop a program that can automatically solve Sudoku puzzles of varying difficulty levels using logical deduction and constraint propagation techniques.

1. **Objectives**

Our project is the Sudoku Solver program. Sudoku is a classic number placement puzzle where the objective is to fill a 9x9 grid with digits from 1 to 9, such that each column, each row, and each of the nine 3x3 subgrids (also called "regions" or "boxes") contains all the digits exactly once. This project aims to create a software application capable of solving Sudoku puzzles automatically.

1. **Logic of the code:**

We take the input Sudoku in an array called” table”. Then we check if there are any unassigned elements present in our table. If there is an unassigned element, then we run a loop for variable “num” from values 1 to 9. We test each value in the unassigned position.

To check for the conditions, we call the function “check” which checks whether the element “num”’s value is already present in the row, column or grid of the unassigned position. If it fails the condition, then we check for the next value of “num”. Once it satisfies the condition, then we assign the position with the satisfactory number and move to another unassigned position.

1. **Concept of Backtracking**

Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.

1. **Approach**

**Step 1**: The user makes the necessary changes in the table matrix so as to complete his question. He must assign 0 to the vacant spots of the Sudoku table.

**Step 2**: In the solveSudoku function, it first calls the search function, to get the position in the table where the element is unassigned. The position of the unassigned element is stored in variables row and col.

**Step 3**: A loop runs where the value of variable num ranges from 1 to 9. In each iteration it checks whether the value of “num” satisfies the condition i.e. the same value should not be present already in the same row, column or its subgrid. For that, a function “check” is called which returns true if “num” satisfies all the conditions, and false if it does not.

**Step 4**: Then the function solveSudoku is called again with the new table passed as the parameter. Then the whole process is repeated till the whole table is filled.

**Step 5**: If in case, in one of the loops, none of the values of num are possible, then the function is backtracked to the value from where we can check other values to be inserted instead.

**Implementation of my Sudoku Solver Project source code**

A computer code with many text

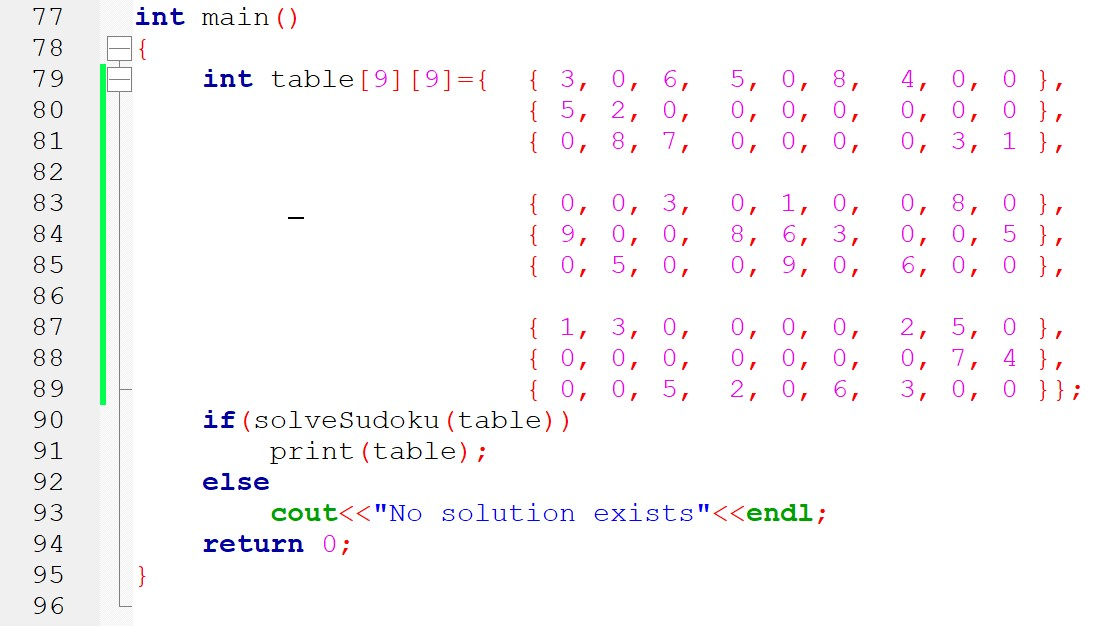
Description automatically generated with medium confidence

A screenshot of a computer code

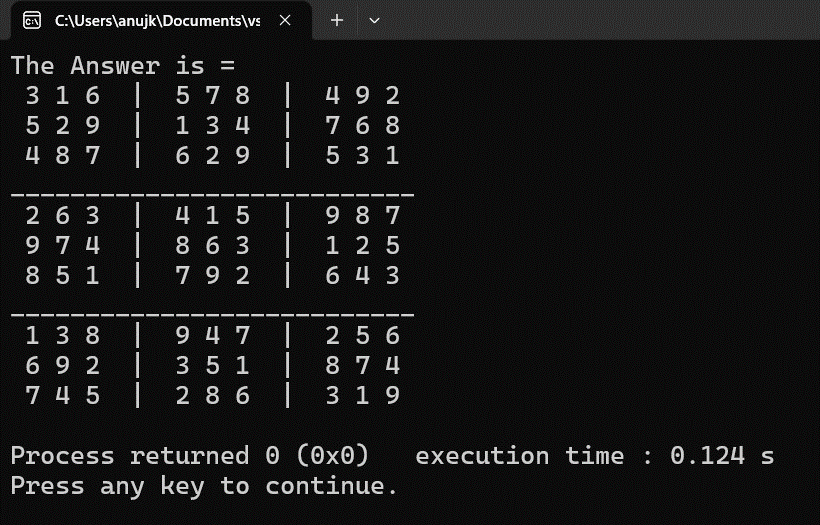
Description automatically generated

A computer code with text

Description automatically generated with medium confidence



**Solution :**



**Conclusion**

The Sudoku game is familiar among all age groups. Using the concept of Backtracking, we are able to get the solution of most of the Sudoku problems. An algorithm of using the Sudoku Solver has been presented and tested that works in an efficient way. Overall, the system works without any bugs. Each developer has his/ her own style of algorithm.

**Bibliography**

* Internship Lectures
* geeksforgeeks.org
* Youtube