CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT
ON
SNAKE AND LADDER GAME

Program Name: BCA

Subject Name/Code: Data Structures(23CAT-201)

Submitted by:

Name: Anubhaw kumar

UID:23BCA10205

Section: BCA – 2 "A"

Submitted to:

Name: Mehak Bhatia

Designation: Assistant Professor

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Abstract

The code implements a simple console-based Snake and Ladder game in C++. It simulates a multiplayer game for 2-4 players on a basic board with defined positions for snakes and ladders. Each player takes turns to roll a virtual die, which generates a random number between 1 and 6 to advance their position. If a player lands on a snake, they slide down to a lower position; if they land on a ladder, they move up to a higher one. The game continues until one player reaches or surpasses the final position on the board, winning the game. The `Snake LadderGame` class handles core game logic, including board setup, player turns, dice rolls, and snake/ladder interactions. This implementation serves as a straightforward example of object-oriented programming in C++ with a focus on game mechanics, control flow, and user input/output in a text-based environment..

Introduction

The Snake and Ladder game is a popular board game traditionally played by two or more players on a grid-based board. The objective is simple: each player races to be the first to reach the end by advancing based on dice rolls. However, the board contains "snakes" and "ladders" that create surprises along the way. Ladders allow players to jump to higher positions, advancing their progress, while snakes send players back, setting them back several positions. This element of chance, combined with strategic moves, creates an engaging, suspenseful experience for players of all ages.

In this console-based C++ implementation, the game is designed for up to four players and runs entirely in a text-based format. The program uses basic object-oriented principles to encapsulate the game's functionality within a `Snake LadderGame` class, which handles player movements, dice rolls, and interactions with snakes and ladders. Each player's position is updated in turn, and the game tracks each move to determine if a player has landed on a snake, a ladder, or the winning position. Random number generation simulates dice rolls, ensuring that the game progresses unpredictably.

This project demonstrates fundamental programming concepts such as randomness, conditional logic, loops, and modular class design, making it a valuable example for learning C++ programming basics through a familiar game format.

Techniques

The C++ Snake and Ladder game implementation employs several programming techniques to handle game logic, randomness, and player interaction. Here's an overview of the key techniques:

1. Object-Oriented Programming (OOP)

- **Class Abstraction**: The `SnakeLadderGame` class encapsulates all game functionalities, including player moves, board setup, and interactions with snakes and ladders. This separation of concerns keeps the code modular and makes it easy to modify or extend.
- **Encapsulation**: By using private variables for game state (such as `numPlayers`, `playerPositions`, `snakes`, and `ladders`), the class protects these values from being modified outside its methods, preserving the integrity of the game logic.

2. Random Number Generation

- **Dice Simulation**: The game simulates dice rolls using the `rand()` function, which generates a random integer for each roll. By using `srand(time(0))`, the program initializes a seed based on the current time to ensure randomness each time the game runs.
- **Controlled Range**: By applying modulo arithmetic (`rand() % 6 + 1`), the program restricts the random values to the dice range (1–6).

3. Data Structures

- **Vectors for Dynamic Data**: A `std::vector<int>` is used to store each player's position on the board. Vectors allow for dynamic resizing, making them suitable for accommodating any number of players (within the allowed limits).
- **Maps for Snakes and Ladders**: `std::map<int, int>` is used to represent snakes and ladders, where the key is the start position and the value is the end position. Maps provide efficient lookups, so the program can quickly determine if a player's position corresponds to a snake or ladder.

4. Control Structures

- **Conditional Statements**: The program uses `if` and `else` statements to check for interactions with snakes and ladders. These conditionals allow the program to handle special cases where players encounter these obstacles.
- **Loops for Player Turns**: A `while` loop in the `play` method iterates until a player wins. The loop includes logic to alternate turns, making sure each player rolls the dice in sequence.

5. Input and Output Management

- **Console Interaction**: The program uses `std::cout` and `std::cin` to handle text-based interaction with the player, including displaying the game state, announcing moves, and printing the winning message.
- **Game State Display**: After each roll, the program outputs the current state, including the dice value, the player's new position, and any snake or ladder interactions, providing feedback to players.

6. Game Logic and Win Condition

- **Boundary Checks**: The program ensures that players do not exceed the board size. If a player's roll would take them beyond the end position, it allows the player to stay in place, making their next roll count.
- **Victory Detection**: After each move, the program checks if the player's position matches the board's end position, and if so, it declares the player as the winner and exits the game loop.

This collection of techniques makes the Snake and Ladder game an effective demonstration of core C++ concepts and provides a foundation for more complex game development or applications that require randomness, structured data handling, and control flow.

System Configuration:

- **OS:** Windows 10 or Linux

- **Processor:** Intel Core i3 (minimum); Core i5 or higher recommended
- **RAM:** 4 GB (minimum); 8 GB recommended
- **Development Environment:** Any C++ IDE (e.g., Visual Studio, Code: Blocks) or Visual Studio Code with a C++ compiler (GCC or Microsoft C++ Compiler)

Summary of Spam Mail Filter Project

This C++ program implements a simple, text-based Snake and Ladder game for multiple players. The objective is to reach the last position on the board first, with each player rolling a virtual dice to move forward. Snakes and ladders are strategically placed on the board: ladders advance players forward, while snakes move them back.

The program is structured around a `SnakeLadderGame` class that encapsulates the core functionalities:

- **Random Dice Rolls** simulate the gameplay, with `rand()` generating values between 1 and 6.
- **Player Positions** are updated in turn, using vectors and maps to track positions and handle encounters with snakes or ladders.
- **Game Logic** checks for a winning condition after each move, declaring a winner when the end position is reached.

The code uses object-oriented programming principles, including encapsulation and modular design, to handle game interactions efficiently. This project demonstrates key C++ programming concepts, including randomness, control flow, and data structures, in a fun, interactive format.

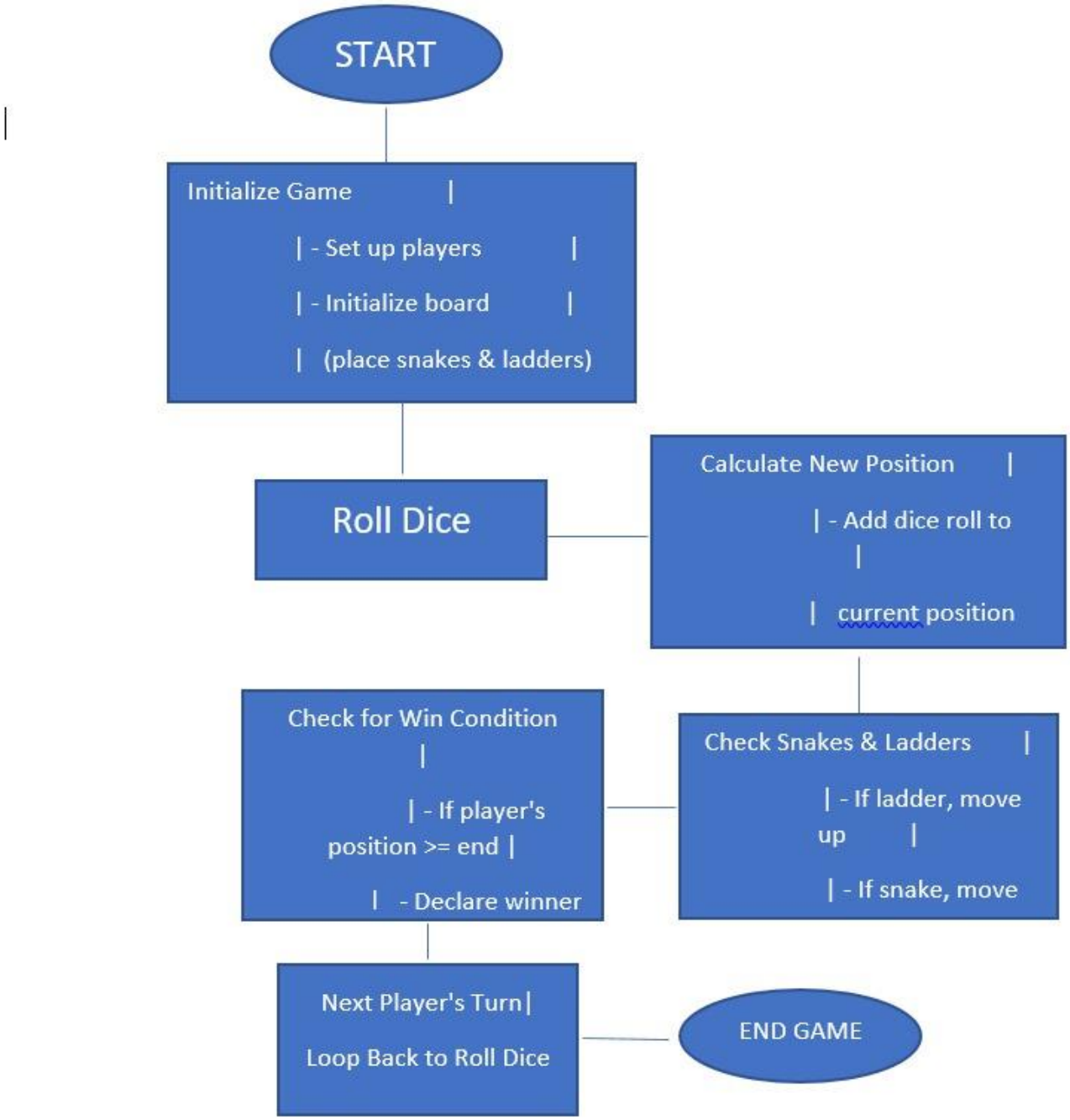
Conclusion

In conclusion, this C++ implementation of the Snake and Ladder game successfully recreates a classic board game experience in a console format. By using object-oriented design, the program encapsulates game logic in a structured, maintainable way, demonstrating the effective use of classes, vectors, and maps for game state management. The project also leverages random number generation to simulate dice rolls, adding an element of unpredictability that mirrors the real game's excitement and suspense.

This game not only highlights core C++ concepts such as control flow, data handling, and modular code design but also provides a practical example of programming logic in a real-world context. It's a solid starting point for understanding game development basics and offers a foundation for expanding into more complex applications. Overall, this project is an effective exercise for learning or reinforcing essential C++ programming skills.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Process



OUTPUT

```
Welcome to Snake and Ladder Game!
Enter the number of players (2-4): 4

Player 1's turn.
Rolled a 4.
Player 1 is now at position 5.

Player 2's turn.
Rolled a 3.
Player 2 is now at position 4.

Player 3's turn.
Rolled a 2.
Player 3 is now at position 3.

Player 4's turn.
Rolled a 3.
Player 4 is now at position 4.
```