Snakes and Ladders project

Game overview/requirements

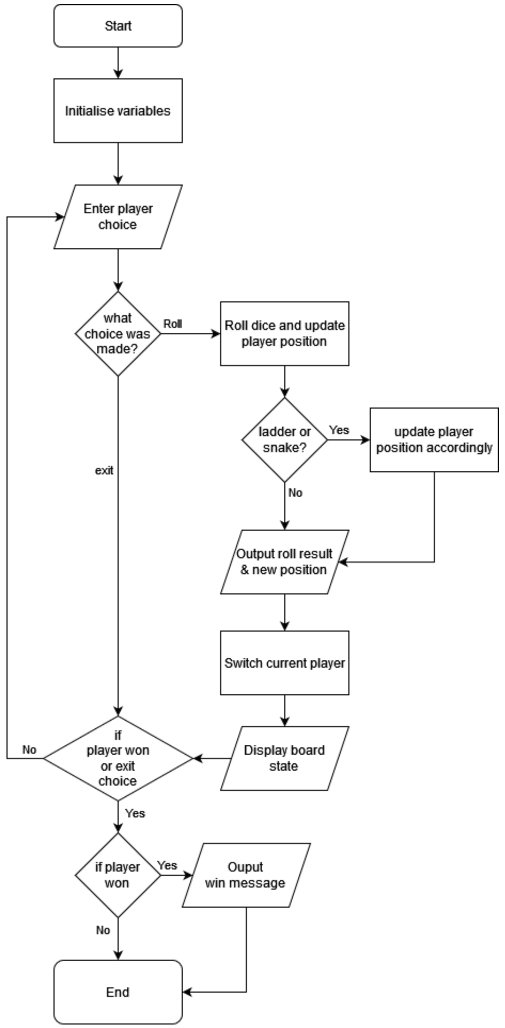
Overview of the classic board game rules:

- Players take turns to roll a dice
- First player that reaches the end wins
- A player landing on the bottom of a ladder moves to the top
- A player landing on the top of a snake moves to the bottom

- Need C++ code that implements these rules

Program flowchart

- Basic overview of step-by-step program flow

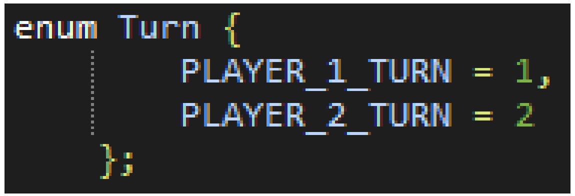


Initializing Variables

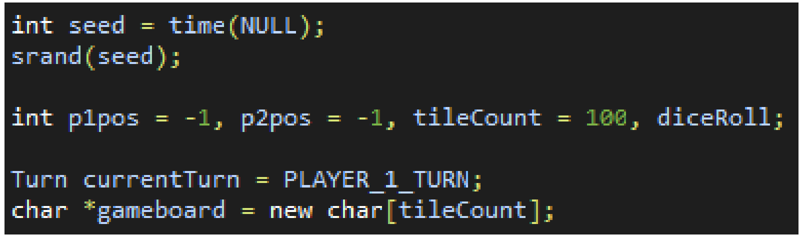
- enum Turn - whose turn it is
- int p1pos and int p2pos – player positions
- int tileCount – size of the board
- char* gameboard – holds the state of each tile

- srand(seed) - for variance in games

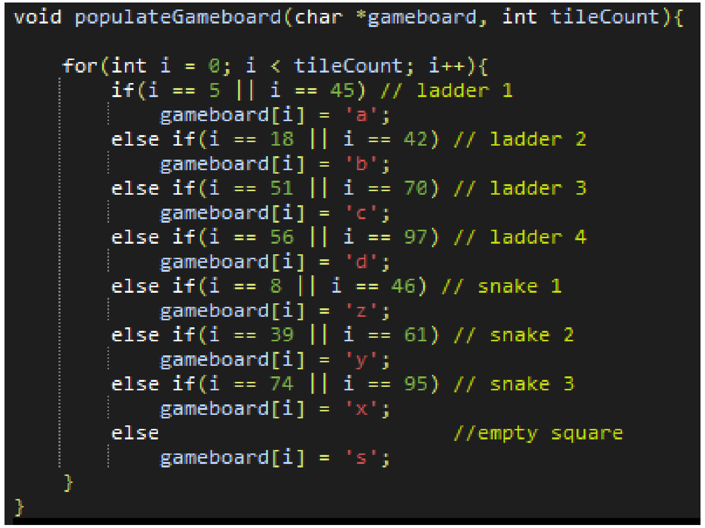
- PopulateGameboard function – fills the game board with different tile types



```
enum Turn {
        PLAYER_1_TURN = 1,
        PLAYER_2_TURN = 2
    };
```



```
int seed = time(NULL);
srand(seed);

int p1pos = -1, p2pos = -1, tileCount = 100, diceRoll;

Turn currentTurn = PLAYER_1_TURN;
char *gameboard = new char[tileCount];
```



```
void populateGameboard(char *gameboard, int tileCount){

    for(int i = 0; i < tileCount; i++){
        if(i == 5 || i == 45) // ladder 1
            gameboard[i] = 'a';
        else if(i == 18 || i == 42) // ladder 2
            gameboard[i] = 'b';
        else if(i == 51 || i == 70) // ladder 3
            gameboard[i] = 'c';
        else if(i == 56 || i == 97) // ladder 4
            gameboard[i] = 'd';
        else if(i == 8 || i == 46) // snake 1
            gameboard[i] = 'z';
        else if(i == 39 || i == 61) // snake 2
            gameboard[i] = 'y';
        else if(i == 74 || i == 95) // snake 3
            gameboard[i] = 'x';
        else                         //empty square
            gameboard[i] = 's';
    }
}
```

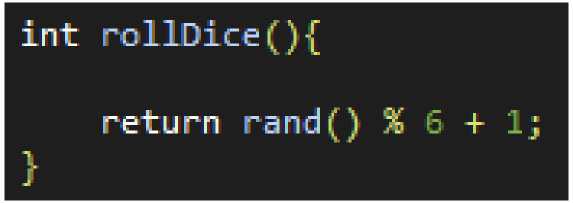
Utilizing Arrays

- 1 dimensional character array to store the board

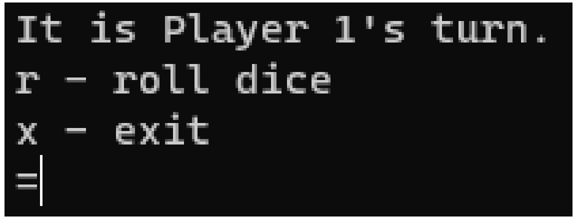
Prompting the player

- Prompting the current player to roll the dice or exit

- Utilizing srand and rand functions to generate random numbers



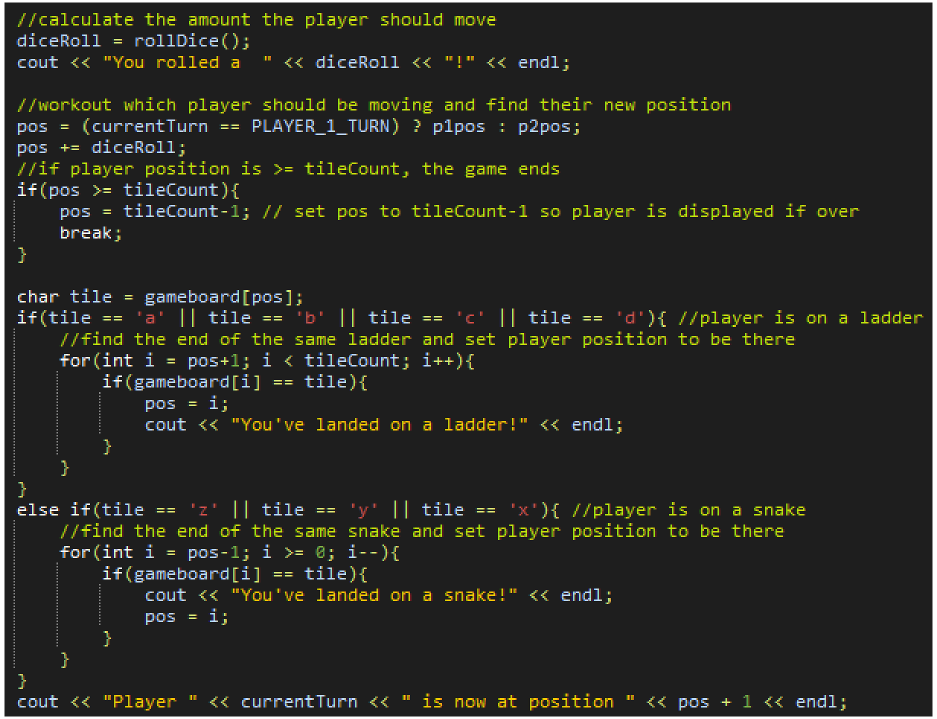
```
int rollDice(){

    return rand() % 6 + 1;
}
```



```
It is Player 1's turn.
r - roll dice
x - exit
=
```

Updating Player Positions

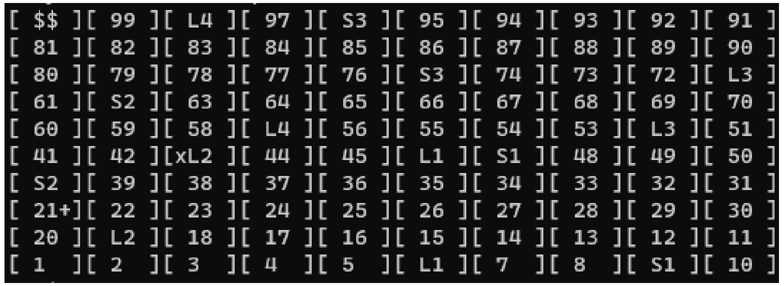
- Updating current player's position based on diceRoll result
- Check if the player has won
- Check if the tile landed on is a ladder or snake
- Move player accordingly
- Inform player of new position



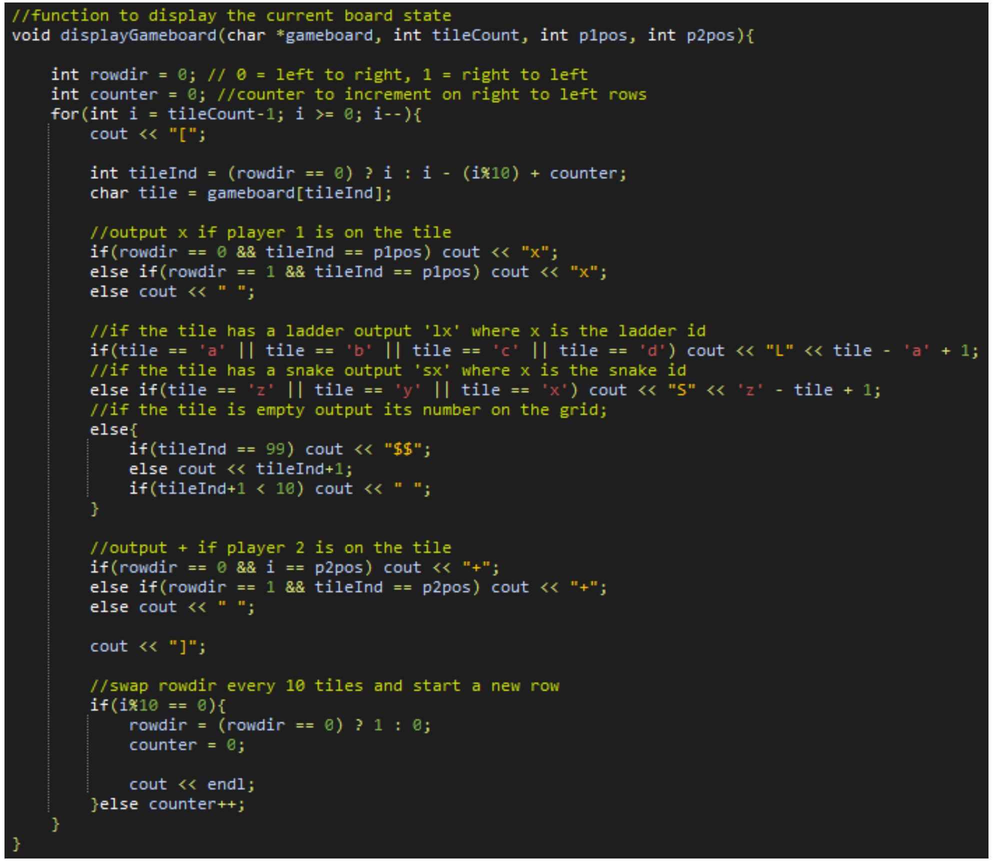
```
//calculate the amount the player should move
diceRoll = rollDice();
cout << "You rolled a  " << diceRoll << "!" << endl;

//workout which player should be moving and find their new position
pos = (currentTurn == PLAYER_1_TURN) ? p1pos : p2pos;
pos += diceRoll;
//if player position is >= tileCount, the game ends
if(pos >= tileCount){
    pos = tileCount-1; // set pos to tileCount-1 so player is displayed if over
    break;
}

char tile = gameboard[pos];
if(tile == 'a' || tile == 'b' || tile == 'c' || tile == 'd'){ //player is on a ladder
    //find the end of the same ladder and set player position to be there
    for(int i = pos+1; i < tileCount; i++){
        if(gameboard[i] == tile){
            pos = i;
            cout << "You've landed on a ladder!" << endl;
        }
    }
}
else if(tile == 'z' || tile == 'y' || tile == 'x'){ //player is on a snake
    //find the end of the same snake and set player position to be there
    for(int i = pos-1; i >= 0; i--){
        if(gameboard[i] == tile){
            cout << "You've landed on a snake!" << endl;
            pos = i;
        }
    }
}
cout << "Player " << currentTurn << " is now at position " << pos + 1 << endl;
```

Displaying the Board

- Displaying the board after each player's turn
- Indicating the position of each player with player 1 is 'x' and player 2 is '+'
- Place ladders and snakes as Lx and Sx respectively (where x is an identifier)
- Empty tiles display their number



```
[ $$ ][ 99 ][ L4 ][ 97 ][ S3 ][ 95 ][ 94 ][ 93 ][ 92 ][ 91 ]
[ 81 ][ 82 ][ 83 ][ 84 ][ 85 ][ 86 ][ 87 ][ 88 ][ 89 ][ 90 ]
[ 80 ][ 79 ][ 78 ][ 77 ][ 76 ][ S3 ][ 74 ][ 73 ][ 72 ][ L3 ]
[ 61 ][ S2 ][ 63 ][ 64 ][ 65 ][ 66 ][ 67 ][ 68 ][ 69 ][ 70 ]
[ 60 ][ 59 ][ 58 ][ L4 ][ 56 ][ 55 ][ 54 ][ 53 ][ L3 ][ 51 ]
[ 41 ][ 42 ][xL2 ][ 44 ][ 45 ][ L1 ][ S1 ][ 48 ][ 49 ][ 50 ]
[ S2 ][ 39 ][ 38 ][ 37 ][ 36 ][ 35 ][ 34 ][ 33 ][ 32 ][ 31 ]
[ 21+][ 22 ][ 23 ][ 24 ][ 25 ][ 26 ][ 27 ][ 28 ][ 29 ][ 30 ]
[ 20 ][ L2 ][ 18 ][ 17 ][ 16 ][ 15 ][ 14 ][ 13 ][ 12 ][ 11 ]
[ 1  ][ 2  ][ 3  ][ 4  ][ 5  ][ L1 ][ 7  ][ 8  ][ S1 ][ 10 ]
```



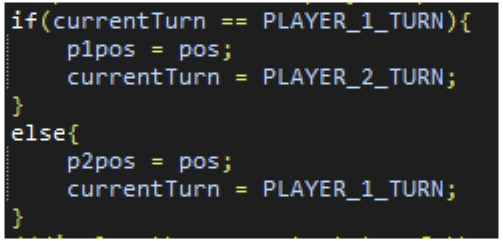
```cpp
//function to display the current board state
void displayGameboard(char *gameboard, int tileCount, int p1pos, int p2pos){

    int rowdir = 0; // 0 = left to right, 1 = right to left
    int counter = 0; //counter to increment on right to left rows
    for(int i = tileCount-1; i >= 0; i--){
        cout << "[";

        int tileInd = (rowdir == 0) ? i : i - (i%10) + counter;
        char tile = gameboard[tileInd];

        //output x if player 1 is on the tile
        if(rowdir == 0 && tileInd == p1pos) cout << "x";
        else if(rowdir == 1 && tileInd == p1pos) cout << "x";
        else cout << " ";

        //if the tile has a ladder output 'lx' where x is the ladder id
        if(tile == 'a' || tile == 'b' || tile == 'c' || tile == 'd') cout << "L" << tile - 'a' + 1;
        //if the tile has a snake output 'sx' where x is the snake id
        else if(tile == 'z' || tile == 'y' || tile == 'x') cout << "S" << 'z' - tile + 1;
        //if the tile is empty output its number on the grid;
        else{
            if(tileInd == 99) cout << "$$";
            else cout << tileInd+1;
            if(tileInd+1 < 10) cout << " ";
        }

        //output + if player 2 is on the tile
        if(rowdir == 0 && i == p2pos) cout << "+";
        else if(rowdir == 1 && tileInd == p2pos) cout << "+";
        else cout << " ";

        cout << "]";

        //swap rowdir every 10 tiles and start a new row
        if(i%10 == 0){
            rowdir = (rowdir == 0) ? 1 : 0;
            counter = 0;

            cout << endl;
        }else counter++;
    }
}
```

Switching Players

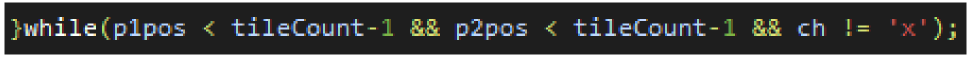
- Updating the currentPlayer variable

- Continuing the loop if the player has entered 'r' to roll the dice

- Exit the loop if a player wins or presses 'x' to exit



```
if(currentTurn == PLAYER_1_TURN){
    p1pos = pos;
    currentTurn = PLAYER_2_TURN;
}
else{
    p2pos = pos;
    currentTurn = PLAYER_1_TURN;
}
```



```
}while(p1pos < tileCount-1 && p2pos < tileCount-1 && ch != 'x');
```

Conclusion

- Exploring the fun and classic game of Snakes and Ladders

- Implementing the game using C++

Thank you for your time 😊