Project 2

<7 Up 7 Down>

CIS-5 45744

NAME: MARICLA RODRIGUEZ

DATE: 07/21/2024

# Introduction

Title: 7 Up 7 Down

This is the 7 Up 7 Down classic casino game that utilizes dice. The dealer will roll a pair of dice and the player will make a bet whether the number rolled is lower, higher, or equal to seven. Once the dealer rolls the dice, the player will be asked how much they are willing to bet that they can predict the outcome. The player’s balance will reflect on the right side of the text, once they’ve won or lost the game. If the player guesses right, they will keep their money and earn the amount they bet. If they bet that the outcome is seven then it quadruples. If they win twice, the bet amount is squared. After every game, the player can decide to keep playing or exit the game. If the player has less than one dollar, they cannot continue to play the game unless they restart.

# Summary

Project size: 328 Lines of codes

The number of variables: 10+

The number of method: 12

This project encompasses many concepts that we learned during lectures and the book. I found this project challenging, but I took it one step at a time. This project took me a week and I think I already see how I can better it for Project 2. I am already planning how I am going to utilize void and arrays.

This code implements a console-based dice game called "7 Up 7 Down."

* **Welcome and Setup**:
  + The program greets the player, asks for their name, and explains the rules of the game.
  + The player is prompted to input the initial amount of money they want to start with.
* **Gameplay**:
  + Players can place multiple bets in a single session.
  + For each bet, the player chooses one of three options:
    - Bet that the total dice roll is less than 7.
    - Bet that the total dice roll is more than 7.
    - Bet that the total dice roll is exactly 7.
  + The player then inputs their bet amount and the dice are rolled.
* **Processing Bets**:
  + Depending on the outcome of the dice roll, the player's cash is adjusted based on their bet and the result.
  + There are different versions of the processBet function to handle cases with or without a player's name and a jackpot feature.
  + A jackpot (extra $100) is awarded if the player correctly bets on rolling a 7.
* **Game Continuation**:
  + After each round, the player is asked if they want to play again. If they run out of money, they are given an option to add more money.
* **End of Game**:
  + The game displays a summary of the player's start and end money and thanks the player for playing.

**Additional Features**:

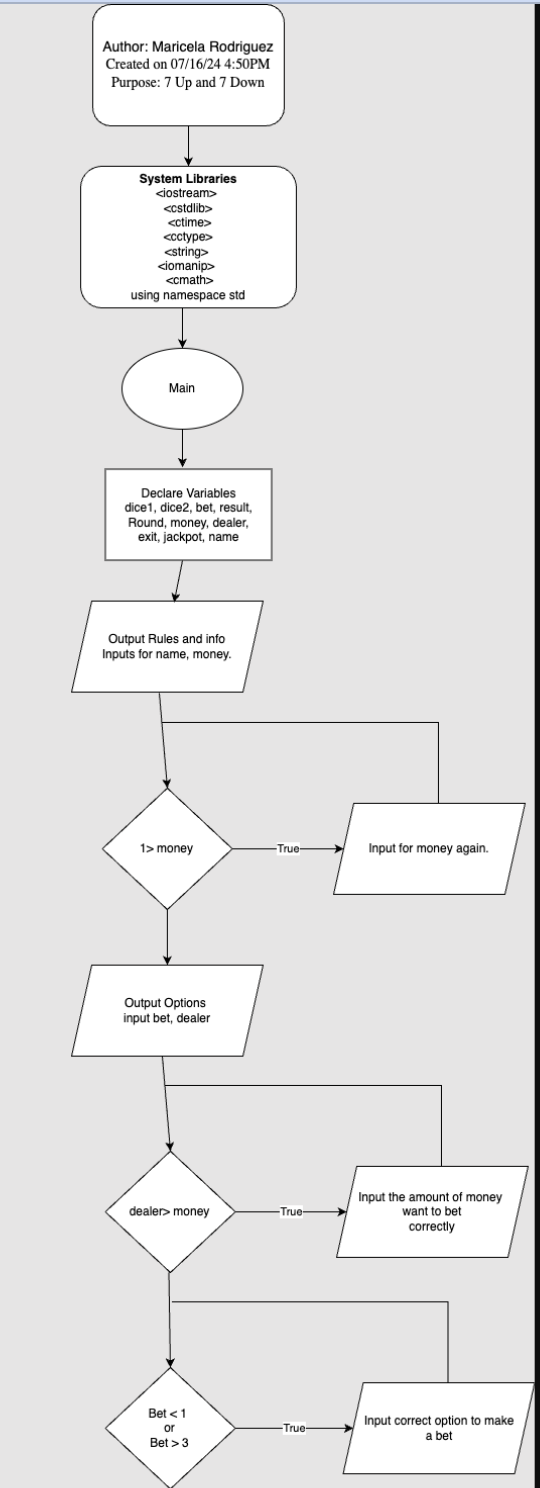
* The game tracks the number of games played.
* The checkJack function determines if a jackpot should be awarded.

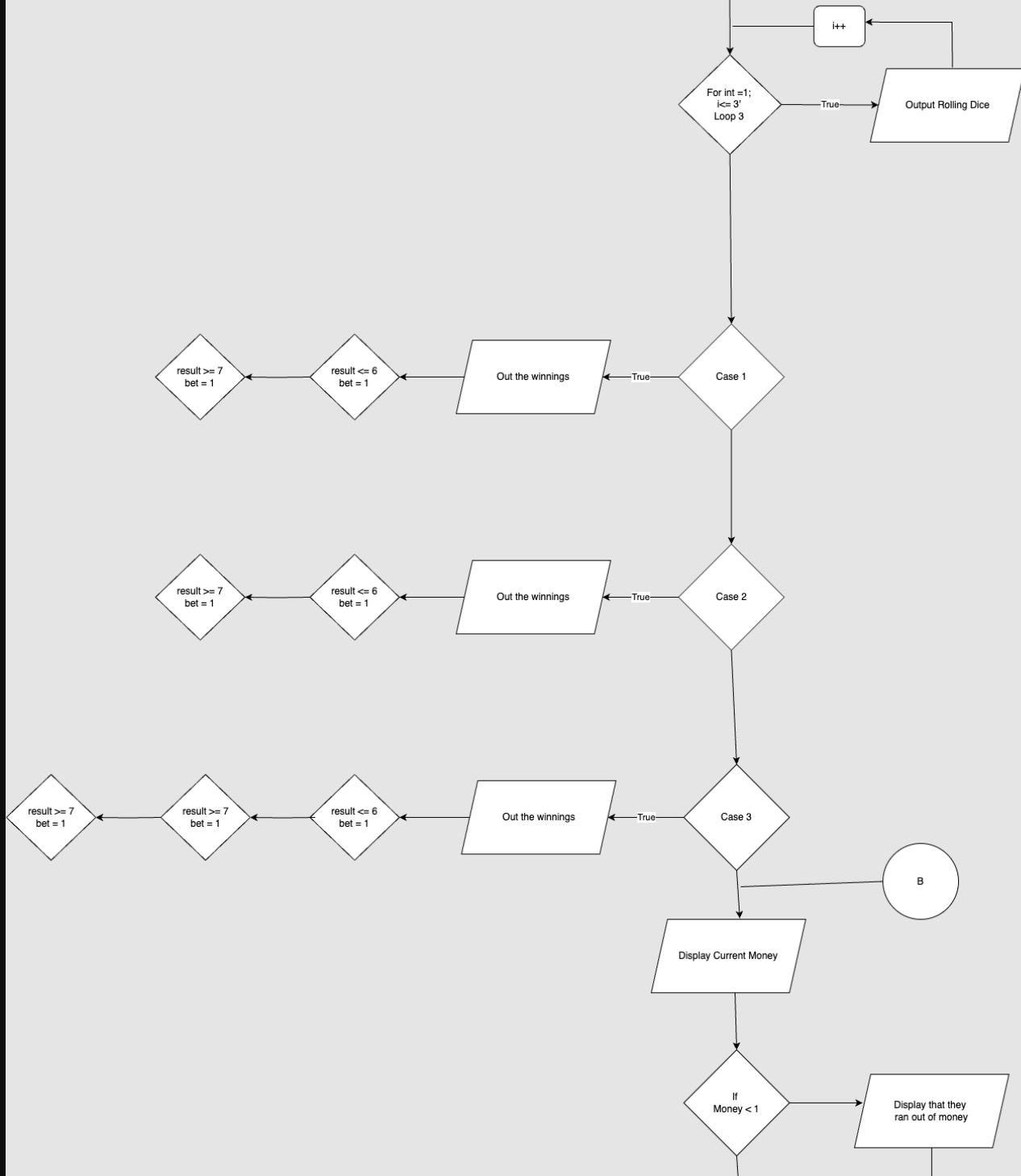
Overall, the game uses dynamic memory allocation for storing bets and dice results, and includes functionality for both interactive gameplay and jackpot features.

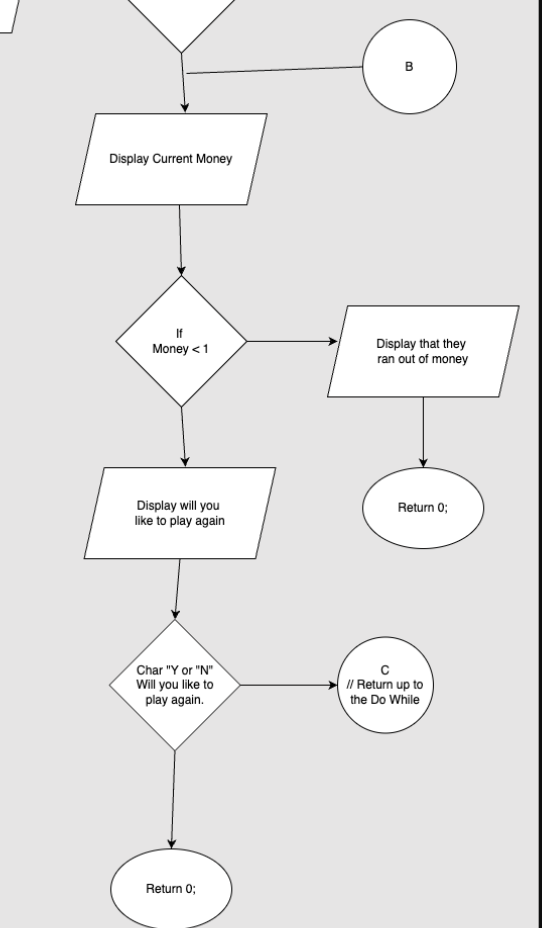
# Description

This program's main point is to Play 7 Up 7 Down and bet on the outcome, only if they have the money for it. It is a simple betting game that allows users to bet on the 7 Up and 7 Down.

# Flowchart

****

****

****

# Pseudo Code

Initialize Variables:

string name

float money, startingMoney

bool jackpot = false

Display Welcome Message:

cout "Welcome! What's your name?"

cin name

Set first letter of name to uppercase

cout "Nice to meet you, " + name + ". This game is called 7 Up 7 Down..."

Get Starting Money:

cout "You must start off with $1 or more to play."

do:

cout "How much money do you want to start with?"

cin money

while money < 1

Set startingMoney to money

Display Game Instructions:

cout "Now that you have some money to bet with. Let's start."

do:

Play Game:

int numBets

cout "How many bets would you like to place?"

cin numBets

// Initialize 2D array to store results and bets

Initialize 2D array results with numBets rows and 2 columns

Initialize array bets with numBets elements

Initialize array dealers with numBets elements

for i from 0 to numBets-1:

if money < 1:

cout "You don't have enough money to continue betting. Please add more money."

money = getStartingMoney()

cout "Bet " + (i+1) + ":"

cout "Choose one of the following options, either 1 to 3."

cout "1: Number is lower than 7."

cout "2: Number is higher than 7."

cout "3: Number is equal to 7."

cin bets[i]

while bets[i] < 1 or bets[i] > 3:

cout "Invalid bet. Please choose a number from 1-3:"

cin bets[i]

cout "How much would you like to bet? Current Balance $" + money

placeBet(dealers[i], money)

for j from 1 to 3:

cout "Loading..."

cout "...Dice Rolling..."

rollDice(dice1, dice2, results[i][0])

results[i][1] = dice1 + dice2

processBet(bets[i], results[i][1], money, dealers[i], name, &jackpot)

Clean up 2D array results

Clean up array bets

Clean up array dealers

Display Game Count:

Increment game count

cout "Games played so far: " + gameCount

if money <= 0:

cout "You have run out of money. Would you like to add more money to continue playing? (y/n):"

char choice

cin choice

if choice == 'y' or choice == 'Y':

money = getStartingMoney()

startingMoney += money

else:

cout "Thanks for playing!"

exitGame(startingMoney, money, name)

return 0

while money > 0 and playAgain()

Exit Game:

exitGame(startingMoney, money, name)

return 0

Function Definitions:

displayWelcomeMessage(name):

cout "Welcome! What's your name?"

cin name

Set first letter of name to uppercase

cout "Nice to meet you, " + name + ". This game is called 7 Up 7 Down..."

getStartingMoney():

cout "You must start off with $1 or more to play."

do:

cout "How much money do you want to start with?"

cin money

while money < 1

return money

gameInstructions():

cout "Now that you have some money to bet with. Let's start."

playGame(name, money, jackpot):

int numBets

cout "How many bets would you like to place?"

cin numBets

// Initialize 2D array to store results and bets

Initialize 2D array results with numBets rows and 2 columns

Initialize array bets with numBets elements

Initialize array dealers with numBets elements

for i from 0 to numBets-1:

if money < 1:

cout "You don't have enough money to continue betting. Please add more money."

money = getStartingMoney()

cout "Bet " + (i+1) + ":"

cout "Choose one of the following options, either 1 to 3."

cout "1: Number is lower than 7."

cout "2: Number is higher than 7."

cout "3: Number is equal to 7."

cin bets[i]

while bets[i] < 1 or bets[i] > 3:

cout "Invalid bet. Please choose a number from 1-3:"

cin bets[i]

cout "How much would you like to bet? Current Balance $" + money

placeBet(dealers[i], money)

for j from 1 to 3:

cout "Loading..."

cout "...Dice Rolling..."

rollDice(dice1, dice2, results[i][0])

results[i][1] = dice1 + dice2

processBet(bets[i], results[i][1], money, dealers[i], name, jackpot)

Clean up 2D array results

Clean up array bets

Clean up array dealers

placeBet(dealer, money):

cin dealer

while dealer > money or dealer <= 0:

cout "Invalid bet. Please enter an amount between 1 and " + money + ": $"

cin dealer

rollDice(dice1, dice2, result):

dice1 = rand() % 6 + 1

dice2 = rand() % 6 + 1

result = dice1 + dice2

cout "Dice 1: " + dice1 + ", Dice 2: " + dice2 + ", Total: " + result

processBet(bet, result, money, dealer, name, jackpot):

switch bet:

case 1:

cout "Your bet is that the total you rolled is smaller than 7. Let's find out!"

cout "You rolled a " + result + "!"

if result <= 6:

cout "Congratulations " + name + ", you win!"

money += dealer \* 2

if checkJackpot(result, bet, jackpot):

cout "Jackpot! Extra $100 added!"

money += 100

else:

cout "Sorry, you lost!"

money -= dealer

break

case 2:

cout "Your bet is that the number rolled is bigger than 7."

cout "You rolled a " + result + "!"

if result >= 8:

cout "Congratulations " + name + ", you win!"

money += dealer \* 2

if checkJackpot(result, bet, jackpot):

cout "Jackpot! Extra $100 added!"

money += 100

else:

cout "Sorry " + name + ", you lost!"

money -= dealer

break

case 3:

cout "Your bet is that you rolled a 7, let's see."

cout "You rolled a " + result + "!"

if result == 7:

cout "Wow " + name + ", you're super lucky! You won!"

money += dealer \* 4

if checkJackpot(result, bet, jackpot):

cout "Jackpot! Extra $100 added!"

money += 100

else:

cout "Sorry " + name + ", you lost."

money -= dealer

break

cout "Current Money $" + money

checkJackpot(result, bet, jackpot):

if result == 7 and bet == 3:

jackpot = true

return true

return false

playAgain():

char answer

cout "Would you like to play again? (y/n):"

cin answer

return (answer == 'y' or answer == 'Y')

displayGameCount():

static int gameCount = 0

gameCount++

cout "Games played so far: " + gameCount

displayFinalSummary(startingMoney, finalMoney):

cout "Starting Money: $" + startingMoney

cout "Final Money: $" + finalMoney

exitGame(startingMoney, finalMoney, name):

displayFinalSummary(startingMoney, finalMoney)

cout "Thank you for playing, " + name + "! See you next time!"

**Proof that code works**

| **Type** | **Variable Name** | **Description** | **Location** |  |
| --- | --- | --- | --- | --- |
| Integer | Dice1 | Dice one | Line 53 |  |
|  | Dice2 | Dice 2 | Line54 |  |
|  | NumBets | Amount of bets want to do | Line 83 |  |
|  | Results | Both dice added | Line 57 |  |
|  | bet | Which bet they chose | Line 61,64,72 |  |
|  | gameCount | Tell you how many times you played | 175,177 |  |
|  |  |  |  |  |
| Float | money | Money you have | Line 15,17,,21,  26,35,41,43 |  |
|  | startingMoney | Used to track initial money and is updated when more money is added. | Line 16,22,28,30,, etc |  |
|  | dealer | handling of bets and the dice-rolling process. | Line 62, ,64,74 |  |
|  | Final Money | The amount of money had at the  End of the game | line194,198 |  |
| String | name | Get user name | Line 14,17,18  26,28,29,35,41,43 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| char | answer | Leave code | Line 145 |  |
|  |  |  |  |  |
| bool | Jackpot | Hit the jackpot | Line 17,26,35,37 |  |
|  |  |  |  |  |
| Libraries | Iostream |  |  |  |
|  | cstdlib | Rand() |  |  |
|  | ctime | time() |  |  |
|  | cctype | toupper(c) |  |  |
|  | string |  |  |  |
|  | iomanip | set precision |  |  |
|  | cmath | pow() |  |  |
|  |  |  |  |  |

**Cross Reference for Project 2**

**You are to fill-in with where located in code**

| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| --- | --- | --- | --- | --- | --- |
| 6 |  | Functions |  |  |  |
| x | 3 | Function Prototypes | 9-21 | 4 | Always use prototypes |
| x | 5 | Pass by Value | 15,21,22,27,30 | 4 |  |
| x | 8 | return | 66,85 | 4 | A value from a function |
| x | 9 | returning boolean | 86, 106 | 4 |  |
|  | 10 | Global Variables |  | XXX | Do not use global variables -100 pts |
| x | 11 | static variables | 104 | 4 |  |
| x | 12 | defaulted arguments | 15 | 4 |  |
| x | 13 | pass by reference | 5,11,15,21,30,106 | 4 |  |
| x | 14 | overloading | 22-31 | 5 |  |
| x | 15 | exit() function | 60 | 4 |  |
| 7 |  | Arrays |  |  |  |
| x | 1 to 6 | Single Dimensioned Arrays | 47,48 | 3 |  |
| x | 7 | Parallel Arrays | 47-49 | 2 |  |
|  | 8 | Single Dimensioned as Function Arguments | g | 2 |  |
|  | 9 | 2 Dimensioned Arrays | 46-49 | 2 | Emulate style in book/in class repositiory |
|  | 12 | STL Vectors |  | 2 |  |
| x |  | Passing Arrays to and from Functions |  | 5 |  |
|  |  | Passing Vectors to and from Functions |  | 5 |  |
|  |  |  |  |  |  |
| 8 |  | Searching and Sorting Arrays |  |  |  |
|  | 3 | Bubble Sort |  | 4 |  |
|  | 3 | Selection Sort |  | 4 |  |
|  | 1 | Linear or Binary Search |  | 4 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not | required to | show | Total | 70 | Other 30 points from Proj 1 first sheet tab |

Page 2

# Cross Reference For Project 2