**Lottery Simulation Code**

Technology & Visual Arts: Georgian College

COMP 2006: Introduction to C++

Instructor: Maziar Shajari

December 5th, 2023

This Project has been done by:

* Russo Vergilius Rillo Roldan #1178985
* Zabiullah Habibi #1205298
* Spencer Cloughley #1178551

Table of Contents

1. Introduction ……………..……………..……………..……………..……………..……………..……………..……………..………3

2. Game Rules and Explanation

2.1. Game Objective

2.2. Game Rules

2.2.1. Number Selection

2.2.2. Ticket Cost

2.2.3. Winning Numbers

2.2.4. Prizes ……………..……………..……………..……………..…………..……………..………………………………………..4

2.2.5. Play Again

2.3. How the Game Works ……………..……………..……………..……………..…………..……………..…………………….5

2.3.1. Number Selection

2.3.2. Ticket Cost

2.3.3. Winning Numbers

2.3.4. Prizes

2.3.5. Play Again

2.3.6. Play Again

3. Initial State and Identified Areas for Improvement ……………..……………..……………..……………..…………..6

4. Code Refactoring

4.1. Rules and Regulations Prompt

4.2. Output Formatting

4.3. Play-Again Option …………………………………………………………………………………………………………………….7

5. Explanation of the Game and Improvement of Gaming Experience

6. Impact of the Changes ……………..……………..……………..……………..……………..………………………………………8

7. Future Improvements

8. Conclusion

9. Code ……………..……………..……………..……………..……………..…………………………………………………………………9

1. Introduction

This report presents the refactoring of a lottery simulation code for the Lottario game. The objective was to improve the code's readability and functionality while preserving its core functionality.

2. Game Rules and Explanation

This section provides an explanation of the game rules for the Lottario game, including how the game works, the objective, and other relevant details based on the provided code.



**2.1. Game Objective**

The objective of the Lottario game is to match as many numbers as possible between the player's chosen numbers and the randomly generated winning numbers. The more numbers matched, the higher the potential prize.

**2.2. Game Rules**

**2.2.1. Number Selection:**

* In the Lottario game, players are required to pick 7 numbers between 1 and 45.
* Note: Lottario is 6 number + 1 bonus but has been simplified to 7 for our application
* Each number should be unique within the selection, and repetition is not allowed.

**2.2.2. Ticket Cost:**

* Each line of numbers selected for the Lottario game has a cost of 1 CAD.
* For each ticket bought the user is given a bonus play as well to increase odds

**2.2.3. Winning Numbers:**

* The winning numbers for the Lottario game are randomly generated.
* The code generates 7 winning numbers from a pool of numbers ranging from 1 to 45.

**2.2.4. Prizes:**

* The prizes in the Lottario game are determined based on the number of matches between the player's selection and the winning numbers.
* The code calculates the number of matches and determines the corresponding prize based on predefined prize tiers.

**2.2.5. Play Again:**

* After each round of the lottery simulation, the code prompts the user to play again.
* If the user chooses to play again, a new set of winning numbers is generated, and the process repeats.
* If the user chooses not to play again, the program terminates.



**2.3. How the Game Works**

To participate in the Lottario using the provided code, players follow these steps:



* **Number Selection:**
* Pick 7 unique numbers between 1 and 45
* Or get the computer to generate 7 unique number for you
* Ensure that the selected numbers do not repeat within the same selection.
* **Purchase the Ticket:**
* In the simulation, the cost of each line is 1 CAD, which is automatically deducted from the player's balance.
* **Winning Numbers Generation:**
* The code generates 7 winning numbers randomly from the range of 1 to 45
* **Match Calculation:**
* The code compares the player's chosen numbers with the generated winning numbers to determine the number of matches.
* It also determines the corresponding prize based on the number of matches and predefined prize tiers.
* **Results Display:**
* The code displays the generated winning numbers, the player's chosen numbers, the number of matches, and the corresponding prize amount.
* The output is formatted in a clear and organized manner to enhance readability.
* The output is also printed to a text file to view later after the program has finished
* **Play Again:**
* After displaying the results, the code prompts the user to play again.
* If the user chooses to play again, a new set of winning numbers is generated, and the process repeats from step 1.
* If the user chooses not to play again, the program terminates.
* The program then displays the grand total for winnings and the tickets that won over $20

3. Initial State and Identified Areas for Improvement

The code from our Assignment 1 version was functional but lacked some key aspects. We needed an option to generate the numbers for the player, now all char input can no longer be an int and vice versa. Assignment 2 also calls for some new features such as; output to a text file, displaying the odds of winning, a ticket ID system, a class system for tickets, bonus lines for each play to increase odds, a vector of winning tickets and free tickets for 2/7 matches:

4. Code Refactoring

Many modifications have to been made to the code to add all new features, here is a list:



**4.1. Computer Generate Prompt**

* Added a prompt to ask the user if they would like to input their numbers manually or let the computer generate a set for them
* Accepts ‘Y,y,yes,Yes’ or ‘N,n,no,No’



**4.2. Output to Text File**

* All lines where we are displaying the tickets or the winnings of the ticket are printed to the console and then printed to a text file as well
* This allows the user to view the results even after closing the program
* As well as this at the end of the program we have added a printout of all the tickets that have won more than $20

**4.3. Verification of Int not Char**



* This section verifies that when the user is asked to input a number for how many lines they would like to play, they do not input a char
* It will tell them invalid input and prompt them to enter a valid input

**4.4. Probability of Winning**



* A short line of code to display to the user how small their chance of winning the lottery is even with bonus lines for every play

**4.5. Ticket Class And Ticket ID Systems**



* This is our Ticket class with valid properties and methods to access all the information stored in the class
* We then stored these tickets in vectors for how many lines were played each round
* This class also displays the Ticket ID system where every ticket has its own ID starting at 1
* Each ticket, bonus ticket, and winning numbers get the same ticket ID to ensure they all match accordingly

**4.6. Bonus Line For Every Ticket**



* This section of code within the auto generated tickets show that for every ticket the user has purchased they also receive a bonus ticket to increase odds
* This ticket has the same ID as the ticket that was purchased and will be tested against the same winning numbers

**4.7. Free Play Tickets When 2/7 Are Correct**



* In the Lottario rules there is a free play awarded for 0/6 + bonus correct that grants the user 1 free play
* This has been modified since we do not have a bonus number to just be if they get 2/7 correct, then next time they play they will receive a free ticket which also gets a bonus ticket

5. Explanation of the Game and Improvement of Gaming Experience

The Lottario game is a lottery simulation where users pick 7 numbers between 1 and 45 for a chance to win prizes. The objective is to match as many numbers as possible with the randomly generated winning numbers. The code refactoring aimed to improve the gaming experience by implementing the following changes:

* The addition of the rules and regulations prompt ensures that users are aware of the game requirements and costs. This improvement allows users to make informed decisions and enhances their understanding of the game.
* The improved output formatting provides a clear and organized presentation of the generated numbers, player's picked numbers, and game results. This enhancement enhances the readability of the game output, making it easier for users to interpret the results.
* The refined play-again option enables users to seamlessly continue playing without restarting the program. This improvement saves time and effort for the users, resulting in a smoother and more enjoyable gaming experience.
* The simplification of 6 + bonus to 7 numbers makes it easier for the user to understand what is needed and easier to account for all possibilities when coding the program
* Outputting the results to a text file enables the user to check their results even after they have closed the program and see how many times they won

6. Impact of the Changes

The implemented changes significantly enhanced the user experience by providing clear rules and regulations, improving output formatting, and refining the play-again option. These modifications contribute to a more user-friendly and enjoyable lottery simulation. Users now have easy access to the rules and regulations, and the output is presented in a clear and organized manner. The play-again option provides a seamless experience, allowing users to continue playing without restarting the program.

7. Future Improvements

While the code refactoring addressed the identified areas for improvement, there are potential avenues for further enhancements. These include:

* Visual Display better looks when generating numbers and when checking for wins
* Adding sounds to the game for each win/loss to enhance the experience

8. Conclusion

In conclusion, the refactoring of the lottery simulation code for the Lottario game resulted in a significant improvement in functionality and usability. The added rules and regulations prompt, improved output formatting, and refined play-again option contribute to a more engaging and user-friendly experience. The code now provides clear instructions and presents the lottery results in an organized manner. With further enhancements, the code can be extended to offer additional features and insights for an even more comprehensive lottery simulation experience.

9. Code

// C++Asssignment2.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <vector>

#include <string>

#include <unordered\_set>

#include <algorithm>

#include <fstream>

#include <limits> // for numeric\_limits

using namespace std;

int generateRandomNumber(int min, int max) {

return rand() % (max - min + 1) + min;

}

class Ticket {

private:

int ticketId;

vector<int> lineNumbers;

public:

int getTicketId() {

return ticketId;

}

vector<int> getLineNumbers() {

return lineNumbers;

}

void setTicketId(int id) {

ticketId = id;

}

void setLineNumbers(vector<int> nums) {

lineNumbers = nums;

}

};

//Lottario uses 6 + 1 bonus numbers in their tickets

//This has been simplified to just using 7 numbers for simplicity sake

int main()

{

srand(time(0)); // Seed the random number generator

cout << "Lottery Simulation - Lottario\n";

string playerName;

int studentNumber;

char playAgain = 'Y';

int grandTotalWinnings = 0; // Accumulated winnings across all replays

int totalSpent = 0;

int numberOfLines;

int totalCost;

int totalWinnings = 0;

int bonusLines;

int freeLines=0;

int prevFreeLines=0;

Ticket nextTicket;

unordered\_set<int> usedNumbers;

bool validNumber = false;

vector<Ticket> winningTickets;

int number;

string computerGenerate;

int ticketId=0;

ofstream myFile("outputGroup3.txt");

cout << "Enter your name: "; // User enters their name

cin >> playerName;

cout << "Enter your student number: "; // User enters their student number

while (!(cin >> studentNumber)) {

cout << "Invalid input. Please enter a valid student number: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

myFile << "Player Name: " << playerName << "\n";

myFile << "Student Number: " << studentNumber << "\n";

char knowRules;

while (true) {

cout << "Do you know the rules & regulations (Y/N)? "; // Asks if they know the rules

cin >> knowRules;

cin.ignore();

if (knowRules == 'Y' || knowRules == 'y' || knowRules == 'N' || knowRules == 'n') {

break;

}

else {

cout << "Invalid input. Please select 'Y' or 'N'.\n";

}

}

if (knowRules == 'N' || knowRules == 'n') {

cout << "Rules: You need to pick 7 numbers between 1 and 45.\n";

cout << "Each line costs 1 CAD.\n";

}

while (playAgain == 'Y' || playAgain == 'y') {

vector<Ticket> playerTickets; // Store player's numbers for each line

vector<Ticket> bonusTickets; // Store bonus numbers for each line

vector<Ticket> computerTickets; // Store computer's numbers for each line

cout << "How many lines do you want to play? "; // Asks user how many lines they want.

while (!(cin >> numberOfLines)) {

cout << "Invalid input. Please enter how many times you would like to play: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

//To make it more clear, if the player pays for one line to play, one more computer-generated line will also be giving to the player as a bonus line. - Taken from the Assignment Description

//In my understanding this means that we are doubling their plays, if they pay for 2 they get 2 bonusLines, if they pay for 10 they get 10 bonusLines

bonusLines = numberOfLines;

totalCost = numberOfLines\*1; //Each ticket costs $1

cout << "You are paying $" << totalCost << " CAD to play " << numberOfLines << " draws. You also get " << bonusLines << " bonus draws. \n";

cout << "You also have " << prevFreeLines << " free lines from the last play!\n";

totalSpent -= totalCost;

//asking if they would like to generate their lines manually or the computer to generate them

while (true) {

cout << "Would you like the computer to generate your lines(Y/N)? ";

cin >> computerGenerate;

cin.ignore();

if (computerGenerate.compare("Y") || computerGenerate.compare("y") || computerGenerate.compare("N") || computerGenerate.compare("n") || computerGenerate.compare("yes") || computerGenerate.compare("no") || computerGenerate.compare("Yes") || computerGenerate.compare("No")) {

break;

}

else {

cout << "Invalid input. Please select 'Y' or 'N'.\n";

}

}

cout << "\n";

//Computer Generate

if (computerGenerate.compare("Y") || computerGenerate.compare("y")) {

for (int i = 0; i < numberOfLines + prevFreeLines; ++i) {

vector<int> playerLineNumbers;

for (int j = 0; j < 7; ++j) {

number = generateRandomNumber(1, 45);

playerLineNumbers.push\_back(number);

}

ticketId++;

nextTicket.setLineNumbers(playerLineNumbers);

nextTicket.setTicketId(ticketId);

playerTickets.push\_back(nextTicket);

cout << "Computer picked for you: ";

for (int num : playerLineNumbers) {

cout << " " << num;

}

cout << endl;

//generates bonus ticket

vector<int> bonusLineNumbers;

for (int j = 0; j < 7; ++j) {

number = generateRandomNumber(1, 45);

bonusLineNumbers.push\_back(number);

}

nextTicket.setLineNumbers(bonusLineNumbers);

bonusTickets.push\_back(nextTicket);

cout << "Computer picked bonus ticket for you: ";

for (int num : bonusLineNumbers) {

cout << " " << num;

}

cout << endl;

//generates winning ticket

vector<int> computerLineNumbers;

for (int j = 0; j < 7; ++j) {

number = generateRandomNumber(1, 45);

computerLineNumbers.push\_back(number);

}

nextTicket.setLineNumbers(computerLineNumbers);

computerTickets.push\_back(nextTicket);

cout << "\n";

}

}

//User generate

else {

for (int i = 0; i < numberOfLines+prevFreeLines; ++i) {

cout << "Enter 7 unique numbers for Line " << i + 1 << " (between 1 and 45):\n";

usedNumbers.clear(); // Emptys set of used numbers to start again

vector<int> playerLineNumbers;

//For loop to create all the number in the row

for (int j = 0; j < 7; ++j) {

validNumber = false;

while (!validNumber) {

cout << "Enter the " << j + 1 << "th number: ";

while (!(cin >> number)) {

cout << "Invalid input. Please a number between 1-45: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

if (number < 1 || number > 45) {

cout << "Invalid number. Number must be between 1 and 45.\n";

}

else if (usedNumbers.count(number) > 0) {

cout << "Number already used. Please enter a different number.\n";

}

else {

playerLineNumbers.push\_back(number);

usedNumbers.insert(number);

validNumber = true;

}

}

}

ticketId++;

nextTicket.setLineNumbers(playerLineNumbers);

playerTickets.push\_back(nextTicket);

//generates bonus ticket

vector<int> bonusLineNumbers;

for (int j = 0; j < 7; ++j) {

number = generateRandomNumber(1, 45);

bonusLineNumbers.push\_back(number);

}

nextTicket.setLineNumbers(bonusLineNumbers);

bonusTickets.push\_back(nextTicket);

//Print out user selected ticket and bonus ticket

//generates winning ticket

vector<int> computerLineNumbers;

for (int j = 0; j < 7; ++j) {

number = generateRandomNumber(1, 45);

computerLineNumbers.push\_back(number);

}

nextTicket.setLineNumbers(computerLineNumbers);

computerTickets.push\_back(nextTicket);

cout << "\n";

}//End of for loop running over number of Lines wanted

}//End of else for player selected numbers

cout << "Chance to win jackpot is : " << numberOfLines\*2+prevFreeLines\*2 << "/4,072,530\n\n";

//Test to see if there are any matches and if so how many. What do they win if there are matches?

for (int i = 0; i < numberOfLines+prevFreeLines; ++i) {

int matches = 0;

const vector<int> lineNumbers = playerTickets[i].getLineNumbers();

const vector<int> bonusLineNumbers = bonusTickets[i].getLineNumbers();

const vector<int> computerLineNumbers = computerTickets[i].getLineNumbers();

cout << "Ticket ID: " << playerTickets[i].getTicketId() << "\n";

cout << "Generated numbers:";

myFile << "Ticket ID: " << playerTickets[i].getTicketId() << "\n";

myFile << "Generated numbers:";

for (int number : computerLineNumbers) {

cout << " " << number;

myFile << " " << number;

}

cout << "\n";

myFile << "\n";

//Checks player tickets against the computer tickets

for (int number : computerLineNumbers) {

if (find(lineNumbers.begin(), lineNumbers.end(), number) != lineNumbers.end()) {

++matches;

}

}

cout << "You picked: ";

myFile << "You picked: ";

for (int num : lineNumbers) {

cout << " " << num;

myFile << " " << num;

}

if (matches > 0) {

cout << "\nCongratulations! You have " << matches << " match(es).";

myFile << "\nCongratulations! You have " << matches << " match(es).";

switch (matches) {

case 1:

cout << "\nYou picked 1 number correctly but you won nothing.\n";

myFile << "\nYou picked 1 number correctly but you won nothing.\n";

totalWinnings += 0;

break;

case 2:

cout << "\nYou picked 2 numbers correctly and won 1 free play.\n";

myFile << "\nYou picked 2 numbers correctly and won 1 free play.\n";

freeLines += 1;

totalWinnings += 0;

break;

case 3:

cout << "\nYou picked 3 numbers correctly and won 10 CAD!\n";

myFile << "\nYou picked 3 numbers correctly and won 10 CAD!\n";

totalWinnings += 10;

break;

case 4:

cout << "\nYou picked 4 numbers correctly and won 30 CAD!\n";

myFile << "\nYou picked 4 numbers correctly and won 30 CAD!\n";

winningTickets.push\_back(playerTickets[i]);

totalWinnings += 30;

break;

case 5:

cout << "\nYou picked 5 numbers correctly and won 500 CAD!\n";

myFile << "\nYou picked 5 numbers correctly and won 500 CAD!\n";

winningTickets.push\_back(playerTickets[i]);

totalWinnings += 500;

break;

case 6:

cout << "\nYou picked 6 numbers correctly and won 10,000 CAD!\n";

myFile << "\nYou picked 6 numbers correctly and won 10,000 CAD!\n";

winningTickets.push\_back(playerTickets[i]);

totalWinnings += 1000;

break;

case 7:

cout << "\nYou picked all 7 numbers correctly and won 40,000,000 CAD!\n";

myFile << "\nYou picked all 7 numbers correctly and won 40,000,000 CAD!\n";

winningTickets.push\_back(playerTickets[i]);

totalWinnings += 40000000;

break;

}

}

else {

cout << "\nSorry, you didn't win any matches.\n";

myFile << "\nSorry, you didn't win any matches.\n";

}

cout << "\n";

myFile << "\n";

matches = 0;

//Checks bonus tickets against the computer tickets

for (int number : computerLineNumbers) {

if (find(bonusLineNumbers.begin(), bonusLineNumbers.end(), number) != bonusLineNumbers.end()) {

++matches;

}

}

cout << "Bonus Ticket picked: ";

myFile << "Bonus Ticket picked: ";

for (int num : bonusLineNumbers) {

cout << " " << num;

myFile << " " << num;

}

if (matches > 0) {

cout << "\nCongratulations! You have " << matches << " match(es).";

myFile << "\nCongratulations! You have " << matches << " match(es).";

switch (matches) {

case 1:

cout << "\nYou picked 1 number correctly but you won nothing.\n";

myFile << "\nYou picked 1 number correctly but you won nothing.\n";

totalWinnings += 0;

break;

case 2:

cout << "\nYou picked 2 numbers correctly and won 1 free play.\n";

myFile << "\nYou picked 2 numbers correctly and won 1 free play.\n";

freeLines += 1;

totalWinnings += 0;

break;

case 3:

cout << "\nYou picked 3 numbers correctly and won 10 CAD!\n";

myFile << "\nYou picked 3 numbers correctly and won 10 CAD!\n";

totalWinnings += 10;

break;

case 4:

cout << "\nYou picked 4 numbers correctly and won 30 CAD!\n";

myFile << "\nYou picked 4 numbers correctly and won 30 CAD!\n";

winningTickets.push\_back(bonusTickets[i]);

totalWinnings += 30;

break;

case 5:

cout << "\nYou picked 5 numbers correctly and won 500 CAD!\n";

myFile << "\nYou picked 5 numbers correctly and won 500 CAD!\n";

winningTickets.push\_back(bonusTickets[i]);

totalWinnings += 500;

break;

case 6:

cout << "\nYou picked 6 numbers correctly and won 10,000 CAD!\n";

myFile << "\nYou picked 6 numbers correctly and won 10,000 CAD!\n";

winningTickets.push\_back(bonusTickets[i]);

totalWinnings += 1000;

break;

case 7:

cout << "\nYou picked all 7 numbers correctly and won 40,000,000 CAD!\n";

myFile << "\nYou picked all 7 numbers correctly and won 40,000,000 CAD!\n";

winningTickets.push\_back(bonusTickets[i]);

totalWinnings += 40000000;

break;

}

}

else {

cout << "\nSorry, you didn't win any matches.\n";

myFile << "\nSorry, you didn't win any matches.\n";

}

cout << "\n";

myFile << "\n";

}

cout << "Your total winnings for this draw: $" << totalWinnings << " CAD.\n";

myFile << "Your total winnings for this draw: $" << totalWinnings << " CAD.\n";

grandTotalWinnings += totalWinnings; // Accumulate the winnings for this replay

//After each play cycle save all the data to word document/text file

prevFreeLines = freeLines;

freeLines = 0;

//Play again feature

cout << "Do you want to play again (Y/N)? ";

cin >> playAgain;

cin.ignore();

if (playAgain != 'Y' && playAgain != 'y') {

cout << "Thank you for playing!\n";

cout << "Your grand total winnings: $" << grandTotalWinnings + totalSpent << " CAD.\n"; // Display the accumulated winnings

myFile << "Thank you for playing!\n";

myFile << "Your grand total winnings: $" << grandTotalWinnings + totalSpent << " CAD.\n";

//Print out winning tickets

cout << "Winning tickets! \n";

myFile << "Winning tickets! \n";

for (Ticket ticket : winningTickets) {

cout << "Ticket Id:" << ticket.getTicketId() << ":";

myFile << "Ticket Id:" << ticket.getTicketId() << ":";

for (int num : ticket.getLineNumbers()) {

cout << " " << num;

myFile << " " << num;

}

cout << "\n";

myFile << "\n";

}

}

}

myFile.close();

}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu

// Debug program: F5 or Debug > Start Debugging menu

// Tips for Getting Started:

// 1. Use the Solution Explorer window to add/manage files

// 2. Use the Team Explorer window to connect to source control

// 3. Use the Output window to see build output and other messages

// 4. Use the Error List window to view errors

// 5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project

// 6. In the future, to open this project again, go to File > Open > Project and select the .sln file