De Arcade

Bernardus Jeffrey Darmawan

2001585474

Table of Contents

[Description 3](#_Toc466235727)

[Desing and Planning 5](#_Toc466235728)

[Charts 5](#_Toc466235729)

[Class and Functions Explanation 12](#_Toc466235730)

[Process 14](#_Toc466235731)

[Update Logs 14](#_Toc466235732)

[Things I have Learned 15](#_Toc466235733)

[Problems Encountered 16](#_Toc466235734)

[Coding 18](#_Toc466235735)

[About Writer 34](#_Toc466235736)

De Arcade

# Description

De Arcade is a program that is made using C++ programming language. De Arcade acts as a platform where games are held, just like common arcade center. The idea of this program is for entertainment purpose.

De Arcade consists of three games, those are: Guess the Number, Slots and Horse Race Betting. Mainly, the games use random function to generate random and unpredictable outcomes so that the players cannot win by predicting a fixed outcome, but purely by real life luck.

De Arcade also implements log-in system which lets the players to save their progress during the game and load the progress when they open the game again. The log-in system uses username specific identification so that each player may have different progresses. For example, person A uses a username called ‘personA’ and person B have another username called ‘personB’. personA and personB usernames record different game progress.

Imagine that De Arcade is a common arcade center, the username is your card, and your card has certain balance in it, which is the in-game credit. The players play by paying certain amount of required credit and, by winning, the players will get certain amount of credit as a reward, and also their progress, their balance, is saved in their card. In addition, users can add balance for their in-game currency by inputting certain combination of characters.

The Games

Guess the Number

In this game, the player is supposed to guess each digit of the randomly generated number. There are 3 digits, and so the player has to guess three times one at a time. When the player guessed a wrong number, the program will help them by telling whether the number they guessed exist in the combination or not and give the player a total of five chances to wrongly guess the numbers. For example, the combination is 1 2 3. Then, the player inputted 2 as their first guess. The program shall tell the player “You entered a correct number but at the wrong position.” If the player happens to successfully guessed all the three digits, the player will be granted certain reward, otherwise they will not.

Slots Game

This game does not involve player’s much. Basically, the player will only have to put their amount of bet on the game. Then, the program will generate a random combination of 3 digits between 1, 2, 3 or 7 (for example, 1 | 2 | 7). For certain results that come out, the player will be given certain amount of balance as a reward.

Horse Race Betting

This game is pretty much like usual horse betting, obviously, without real horses. The player can bet on 3 horses maximum out of 10 existing horses. Being without real horses, the program will generate random result as the winning horse. If the player bet on the winning horse, they will be given certain reward depending on how many horses they bet on.

Log in System

The Log in System lets players to keep record of their progress (their in-game currency balance) and also access it by entering their username. After entering username, the program will load their progress into a class that keeps the information of the player until the program is terminated.

Top Up System

This system is provided solely for the player to add more balance to their account. Player must enter a certain code that holds certain amount of credit balance to add. Then, that amount of credit will be added to player’s account. The code cannot be used more than one time. The voucher codes are generated by a program attached in Coding section.

Note:

The terms ‘user’ and ‘player’ are to have the same meaning -> the one running the program.

The terms ‘in-game currency’, ‘balance’ and ‘credit’ are to have the same meaning -> virtual money inside the game.

# Desing and Planning

## Charts

Figure 2.1 General Flowchart of How the Program Works

Start

Log In

Choose Game

Guess the Number

Slots

Horse Race Betting

Top Up Balance

Player Info

End

Back?

Exit

No

Yes

1

2

3

4

5

6

Figure 2.1.1 Flowchart for Log In Process

Start

Input Username

Check Exist

Create New Username

Write in File

Load Username

Not Exist

Exist

Display Username and Credit Balance

Choose Game

Figure 2.1.2 Flowchart of Guess the Number

Choose Game

Input Bet Amount

Correct

Add Reward to Balance

No

Show Reward

Choose Game

Reduce Balance by Bet Amount

Generate Random Digits

Input Guess

Yes

Trial < 5

All Digit Guessed

No

Guess Next Digit

Yes

Save Progress

No

No Reward

Figure 2.1.3 Flowchart of Slot

Choose Game

Input Bet Amount

Add Reward to Balance

Show Reward

Choose Game

Reduce Balance by Bet Amount

Generate Random Combination

Yes

Win Anything

No

Save Progress

No Reward

Figure 2.1.4 Flowchart of Horse Race Betting

Choose Game

Input Bet Amount for each Horse

Correct Prediction

Add Reward to Balance

Show Reward

Choose Game

Reduce Balance by Bet Amount

Generate Random Race Winner

Yes

No

Save Progress

No Reward

Input Number of Horses to Bet on

Input which Horse

Figure 2.1.5 Flowchart of Top Up Balance

Choose Game

Correct Code

Add Amount to Balance

Retrieve Amount

Choose Game

Yes

No

Save Progress

No Adding

Input Voucher Code

Figure 2.1.6 Flowchart of Player Info

Choose Game

Return?

Choose Game

Yes

No

Display Username & Balance

End Program

Figure 2.2 UML Diagram - UserData Class

|  |
| --- |
| **UserData** |
| * userName : string * igCurrency : int * counter : int |
| * UserData() * setUserName(string userName) : void * setCounter(int counter) : void * setIGCurrency(int igCurrency) : void * getUserName() : string * getIGCurrency() : int * getCounter() : int * saveGame() : void * checkIGC() : void * checkUserName() : void |

## Class and Functions Explanation

Class:

* UserData

his class temporarily stores the data of the player

Member Variables:

* + userName (string): stores current player’s userName
  + igCurrency (int): stores current balance of the account
  + counter (int): marks the position of igCurrency on the array

Member Functions:

* + UserData (constructor): initializes member variables as empty string and zero integers
  + saveGame (void): saves the current progress by rewriting the Database file.
  + checkIGC (void): prints current balance
  + checkUserName (void): prints current userName

Functions:

* void log\_In (UserData &)

This function asks the user to enter a username. The program will then look into its database whether the username entered already existed or not. If the username exists, the program will load user’s progress, otherwise, the program will assume that username is a new one then write it into the database and initialize in-game currency as 100. Most functions in this application accepts UserData object as an argument so that when there are changes that happen would automatically be recorded in the class.

* void defaultInterface ()

This function displays a welcoming header in chooseGame function.

* void chooseGame (UserData &)

This function, other than displaying what is in defaultInterface function, displays a series of options of the three games and other options such as Top Up Credits, Player Info, and exit which, then, asks user input of choice. This function also uses switch function to redirect user to their desired choice.

* void gameGuessTheNumber (UserData &)

This function contains the mechanism of the Guess the Number Game. This game asks user’s bet on the game, then generates 3 random 1 digit integers and asks the user to give a guess of what each digit is one at a time. When the user guessed correct series of numbers, the user will be granted a certain reward. However, after guessing wrong integers five times at any phases, the user will be redirected to chooseGame without receiving any reward. The user will be redirected to chooseGame after game ends.

* void gameSlots (UserData &)

This function contains the mechanism of the Slots Game. The game will ask the user to input their bet on the game, then generate random 3 random 1 digits out of the number 7, 1, 2 and 3. Each outcome grants different prizes. The user will be redirected to chooseGame after game ends.

* void gameHorseRace (UserData &)

This functions contains the mechanism of Horse Race Betting Game. The game will ask the user how many horses they want to bet on, which horse and how much they will bet, and determines whether any of the user’s prediction is correct or not. Correct prediction leads to reward giving. The amount of reward is dependent to how many horses the user bet on. The user will be redirected to chooseGame after game ends.

* void topUpCredits (UserData &)

This function contains the mechanism of adding balance to user by providing certain code to the program. If the code inputted is correct, user’s balance will be added, then the code will be ‘deleted’ from the database so that the player cannot enter the same code. The user will be redirected to chooseGame after the mechanism ends.

* int main ()

Main function is essential in C++ Programming in order for the program to run. This function runs some functions used for the application mechanism.

# Process

## Update Logs

1. 6 October 2016

Planning of the project. Created files (text file) of descriptions about how the project would be.

1. 10 October 2016

Continue on planning of the project. Hierarchy chart and rough planning was created.

1. 13 October 2016

Created first draft. The draft contains non-runnable codes (main function was actually empty). However, base algorithm of how the ‘Guess the Number’ game works was created.

1. 21 October 2016

Tried using split files. Created header file and a file where the function body lies. However, I made a mistake. I included the ‘function body file’ rather than the header file which leads to no difference with one file coding.

Also, the program was already able to run. Added input validation for game choice using do-while loop.

1. 28 October 2016

Tried various ways to make header file work but failed. Created a one file code. All three games were already able to be run. But still need improvement. Log In system was not running yet.

1. 3 November 2016

Log In system was able to run. The code had been implemented with class. Class helps when storing and saving player’s data.

1. 6 November 2016

Final changes. Added Top Up Balance system. Split files successfully implemented. Also, created a program to generate random code for Top Up Balance system.

## Things I have Learned

1. Header file is not supposed to have the whole function body in it. As the title says, header file should only have the function headers and/or class declaration. After succeeding on using separate files, compiling time of the project is much faster than before.
2. Tracing errors by printing something even line by line worth the result I get. I could have used for existing algorithm in the internet, but some of them just did not fit the purpose I was expecting. In the end, I had to make my own algorithm which causes a lot of errors in my program. However, after that, I could run the program the way I wanted.
3. Along the process of making this project, I learned that there are a lot of ways that we can use to solve each problem. We can write all the codes in main function or in separate functions. We can use some arrays to store some interconnecting values (like username and balance) and we can use class for that.
4. Using Exceptions is very useful to skip codes after throwing the ‘error’ rather than using multiple **if** functions(if(true) else skip, if(true) else skip, if(true) else skip, continue).
5. Switch function is rather useful when it comes to integer cases, but as for string, we still need to use if function.
6. I learned how to make a flowchart.

## Problems Encountered

1. Using arrays as storing place. I planned on using arrays to store ‘username’ and ‘in-game currency’. The plan was to have those two arrays the same index in order for the player’s information to move around. However, after learning about classes, I found that using a class is more practical than using ‘same index arrays’.
2. Using fstream library.

It was not easy to use the functions in fstream library. The errors I encountered were everywhere; when reading the file (>>), when writing the file (append, <<), also when rewriting the file (<<). For example, when I tried to use append when writing file, it appeared that the text I wanted was written on the first line of the file. It turned out that the program needs to close the file first after being read. And there were more problems regarding read, write, and rewriting file that I solved by tracing the errors.

1. The Slots Game plan was to print first result first, then waits the player to press enter and then print second result, and so the third result. However, if the player presses enter, the program will go to one line below, which is not the result I expected. I tried using cin.get().

Expectation example:

7 | <enter> 7 | <enter> 7

Problem encountered:

7 | <enter>

7 | <enter>

7

I looked for ‘how to go back one line for c++’, but mostly the results are going back one line while also replacing the line printed before. Problem remained unsolved.

1. Splitting files. I tried splitting files into main.cpp, declare.hpp and declare.cpp, and failed. I consult with Sir Bagus, followed his instruction after he solved the problem in his computer, yet it still fails in my computer. Then I continued making my project in one file. I tried doing the same thing as before, however, this time I tried making new project and created new source codes in the project. This method worked.
2. There was a time when I did not declared the function playerInfo() in the header file, but the project still recognized the function. My guess is that it was because the function was not called in main function, but in chooseGame() function. Its position was below playerInfo() in declare.cpp file, so the function chooseGame() was able to recognize the existence of playerInfo() function.
3. This project report is the first time for me to make a flow chart. That being said, I might make some mistakes on those charts above; maybe I was not detailed enough, or I used wrong shapes.

# Coding

main.cpp

|  |
| --- |
| //main program  #include "declare.hpp"  int main()  {  UserData player;    log\_In(player);    chooseGame(player);  return 0;  } |

declare.hpp

|  |
| --- |
| //This is header file  #ifndef DECLARE\_HPP  #define DECLARE\_HPP  #include<iostream>  #include<string>  #include<fstream>  #include<ctime>  using namespace std;  class UserData{  private:  string userName;  int counter = 0;  int igCurrency=0;  public:  UserData();  void setUserName(string);  void setCounter(int );  void setIGCurrency(int );  string getUserName();  int getCounter();  int getIGCurrency();  void saveGame();  void checkIGC();  void checkUserName();  };  void log\_In(UserData &);  void defaultInterface();  void chooseGame(UserData &);  void gameGuessTheNumber(UserData &);  void gameSlots(UserData &);  void gameHorseRace(UserData &);  void topUpCredits(UserData &);  void playerInfo(UserData &);  #endif |

declare.cpp

|  |
| --- |
| //  //File where functions' body are written  //  //include header file  #include"declare.hpp"  string temporaryAllFile[999]; //stores all content of Database.txt file  int i = 0; //total number of lines on the txt file  //member functions in UserData class  UserData::UserData(){  this->userName = "";  this->counter = 0;  this->igCurrency = 0;  }    void UserData::setUserName(string userName)  {  this->userName = userName;  }  void UserData::setCounter(int counter)  {  this->counter = counter;  }  void UserData::setIGCurrency(int igCurrency)  {  this->igCurrency = igCurrency;  }    string UserData::getUserName()  {  return this->userName;  }  int UserData::getCounter()  {  return this->counter;  }  int UserData::getIGCurrency()  {  return this->igCurrency;  }    void UserData::saveGame()  {  //save game by rewriting the file  fstream fileOp;  fileOp.open("Database.txt", ios\_base::in | ios\_base::out);  temporaryAllFile[counter-1] = userName; //in the array, userName is located before the igCurrency  temporaryAllFile[counter] = to\_string(igCurrency);    for(int j = 0; j < i; j++)  {  fileOp << temporaryAllFile[j] << "\n";  j++;  fileOp << temporaryAllFile[j] << "\n\n";  }  fileOp.close();  }  void UserData::checkIGC()  {  cout << "In-game currency balance: " << igCurrency << endl;  }  void UserData::checkUserName()  {  cout << "UserName: " << userName << endl;  }  //global functions  void defaultInterface()  {  cout << "--------Welcome to DE ARCADE---------" << endl;  cout << "You will experience tons of happiness" << endl;  cout << " by playing fun games here " << endl;  cout << "This game is made by Jeffrey Darmawan" << endl;  cout << "-------------------------------------" << endl;  cout << endl;  }  void playerInfo(UserData &player)  {  int userInput;  system("CLS");  cout << "--- Your Data ------------------------" << endl;  player.checkUserName();  player.checkIGC();  cout << endl << "1. Return to choose game" << endl;  cout << "0. Exit" << endl;  cin >> userInput;  switch(userInput)  {  case 1: chooseGame(player);  break;  case 0: break;  default: break;  }  }  //I haven't decided the functions types  void chooseGame (UserData &player)  {  bool flag;  int userInput;  system("CLS");  do{  defaultInterface();  flag = true;  cout << "Please choose one of the following games: " << endl;  cout << "1. Guess the Number " << endl;  cout << "2. Slots " << endl;  cout << "3. Horse Race Betting " << endl;  cout << "8. Top Up Credits" << endl;  cout << "9. Player Info" << endl;  cout << "0. Exit game " << endl;  cout << "Your choice : "; cin >> userInput;    switch(userInput)  {  case 1: gameGuessTheNumber(player);  break;  case 2: gameSlots(player);  break;  case 3: gameHorseRace(player);  break;  case 8: topUpCredits(player);  break;  case 9: {  playerInfo(player);  break;  }  case 0: userInput = 0;  break;  default:  {  cout << "Please re-enter your choice" << endl;  cout << "Press enter to continue " << endl;  cin.ignore();  cin.get();  flag = false;  system("CLS");  }  }  }while(!flag);  }  void log\_In(UserData &player)  {  string userName;  cout << "Enter username ";  cin >> userName;    string temp;  int igCurrency;  fstream fileOp;  fileOp.open("Database.txt", ios\_base::in | ios\_base::out | ios\_base::app );  bool flag = false;  while(fileOp >> temp)  {  temporaryAllFile[i] = temp;  i++;  //if userName exists load igCurrency  if(temp == userName)  {  fileOp >> temp;  player.setCounter(i);  igCurrency = atoi (temp.c\_str());  temporaryAllFile[i] = temp;  i++;  flag = true;  cout << "Loading data..." << endl;  }  }  fileOp.close();    //if userName does not exist,  //create new userName, write it on the file,  //and initialize igCurrency as 100  if (flag == false)  {  fileOp.open("Database.txt", ios\_base::in | ios\_base::out | ios\_base::app);  fileOp << userName << "\n";  igCurrency = 100;  fileOp << igCurrency << "\n\n";  fileOp.close();    player.setCounter(i+1);  i+=2;  cout << "Creating New Username..." << endl;  }    //assign userName and igCurrency to player class  player.setUserName(userName);  player.setIGCurrency(igCurrency);    cout << "--- Your Data ------------------------" << endl;  player.checkUserName();  player.checkIGC();  cin.ignore();  cin.get();  }  void gameGuessTheNumber(UserData &player)  {  system("CLS");  cout << "---------------------------------------" << endl;  cout << "---------- Guess the Number -----------" << endl;  cout << "---------------------------------------" << endl;  try{  //declaration of necessary variables  srand(time(NULL));  int num1 = 0, num2 = 0, num3 = 0, digit;  bool condition = 1;  int chances = 0;  int bid, reward = 0, multiplier = 3;    cout << "How much do you want to bet? "; cin >> bid;  player.setIGCurrency(player.getIGCurrency() - bid);  //each digit can't be same as the other  while (num1 == num2 || num2 == num3 || num1 == num3)  {  num1 = rand() % 10; //one digit random number  num2 = rand() % 10;  num3 = rand() % 10;  }    while (condition)  {  cout << "Please guess the first digit : "; cin >> digit;  if (digit == num1)  {  condition = 0; //exit loop  }  else if(digit == num2 || digit == num3)  {  cout << "You entered a correct number but at the wrong position" << endl;  cout << "PLEASE TRY AGAIN!" << endl;  chances++;  if(chances >= 5)  {  cout << "You failed guessing the number " << endl;  string failGuess;  throw failGuess;  }  }  else  {  cout << "You entered wrong number!! PLEASE TRY AGAIN!" << endl;  chances++;  if(chances >= 5)  {  cout << "You failed guessing the number " << endl;  string failGuess;  throw failGuess;  }  }  }  condition = 1;  while (condition)  {  cout << "Please guess the second digit : "; cin >> digit;  if (digit == num2)  {  condition = 0; //exit loop  }  else if(digit == num1 || digit == num3)  {  cout << "You entered a correct number but at the wrong position" << endl;  cout << "PLEASE TRY AGAIN!" << endl;  chances++;  if(chances >= 5)  {  cout << "You failed guessing the number " << endl;  string failGuess;  throw failGuess;  }  }  else  {  cout << "You entered wrong number!! PLEASE TRY AGAIN!" << endl;  chances++;  if(chances >= 5)  {  cout << "You failed guessing the number " << endl;  string failGuess;  throw failGuess;  }  }  }    condition = 1;  while (condition)  {  cout << "Please guess the third digit : "; cin >> digit;  if (digit == num3)  {  condition = 0; //exit loop  }  else if(digit == num2 || digit == num1)  {  cout << "You entered a correct number but at the wrong position" << endl;  cout << "PLEASE TRY AGAIN!" << endl;  chances++;  if(chances >= 5)  {  cout << "You failed guessing the number " << endl;  string failGuess;  throw failGuess;  }  }  else  {  cout << "You entered wrong number!! PLEASE TRY AGAIN!" << endl;  chances++;  if(chances >= 5)  {  cout << "You failed guessing the number " << endl;  string failGuess;  throw failGuess;  }  }  } //loop ends and earn reward    reward = multiplier \* bid;  cout << "Congratulations, you guessed the right numbers! " << endl;  cout << "Here is your reward : " << reward << endl;    player.setIGCurrency(player.getIGCurrency() + reward);  player.saveGame();  chooseGame(player);  }  catch(string failGuess)  {  cout << "You failed this game, please proceed to return to choose game." << endl;  player.saveGame();  cin.ignore();  cin.get();  chooseGame(player);  }  }  void gameSlots(UserData &player)  {  system("CLS");  cout << "----------------------------------------" << endl;  cout << "---------------- SLOTS -----------------" << endl;  cout << "----------------------------------------" << endl;  srand(time(NULL));  int slot[3] = {rand() % 100, rand() % 100, rand() % 100};  int result[3];    int bid, reward = 0, multiplier = 2;    cout << "Welcome to the slots game!" << endl;  do{  cout << "How much do you want to bet(min. 10)? "; cin >> bid;  if (bid < 10)  cout << "Please re-enter bet======================" << endl;  }while (bid<10);  player.setIGCurrency(player.getIGCurrency() - bid);  cout << player.getIGCurrency();    //assume the common chances of the slots are: 10%, 30%, 30%, and 30%  for (int i = 0; i <=2; i++)  {  if (slot[i] >=0 && slot[i] <= 9)  {  result[i] = 7;  cout << " " << result[i]<< " | ";  }  else if (slot[i] > 9 && slot[i] <=39)  {  result[i] = 1;  cout << " " << result[i]<< " | ";  }  else if (slot[i] > 39 && slot[i] <=69)  {  result[i] = 2;  cout << " " << result[i]<< " | ";  }  else if (slot[i] > 69 && slot[i] <=99)  {  result[i] = 3;  cout << " " << result[i]<< " | ";  }  }    if(result[0] == result[1] && result[1] == result[2])  {  if (result[0] == 7)  {  reward = bid \* 10;  cout << "Congratulations! You won: " << reward << endl;  player.setIGCurrency(player.getIGCurrency() + reward);  player.saveGame();  cout << "Please proceed to return to Choose Game Menu..";  cin.ignore();  cin.get();  chooseGame(player);  }  else if (result[0] == 1 || result[0] == 2 || result[0] == 3)  {  reward = bid \* 3;  cout << "Congratulations! You won: " << reward << endl;  player.setIGCurrency(player.getIGCurrency() + reward);  player.saveGame();  cout << "Please proceed to return to Choose Game Menu..";  cin.ignore();  cin.get();  chooseGame(player);  }  }  else if (result[0] == result[1] || result[0] == result[2] || result[1] == result[2])  {  if ((result[0] == 7 && result[1] == 7) || (result[0] == 7 && result[2] == 7) || (result[2] == 7 && result[1] == 7))  {  reward = 0;  cout << "Sorry you did not win anything" << endl;  player.setIGCurrency(player.getIGCurrency() + reward);  player.saveGame();  cout << "Please proceed to return to Choose Game Menu..";  cin.ignore();  cin.get();  chooseGame(player);  }  else  {  reward = bid + 10;  cout << "Congratulations you won " << reward << endl;  player.setIGCurrency(player.getIGCurrency() + reward);  player.saveGame();  cout << "Please proceed to return to Choose Game Menu..";  cin.ignore();  cin.get();  chooseGame(player);  }  }  else  {  reward = 0;  cout << "Sorry you did not win anything" << endl;  player.setIGCurrency(player.getIGCurrency() + reward);  player.saveGame();  cout << "Please proceed to return to Choose Game Menu..";  cin.ignore();  cin.get();  chooseGame(player);  }  }  void gameHorseRace(UserData &player)  {  system("CLS");  cout << "---------------------------------------" << endl;  cout << "--------- Horse Race Betting ----------" << endl;  cout << "---------------------------------------" << endl;  int numHorses, input, multiplier;  bool flag = 0;    cout << "Welcome to the Fast-Forwarded Horcerace betting!"<< endl;  cout << "There are 10 horses out there and you are allowed to bet at the maximum of 3 horses" << endl <<  "How many horses do you want to bet?"; cin >> numHorses;    //different reward based on how many horses the player bet on  if(numHorses == 1)  multiplier = 7;  else if(numHorses == 2)  multiplier = 3;  else if(numHorses == 3)  multiplier = 2;    int tempWhichHorse[numHorses];  int tempHowMuch[numHorses];    //input amount the player bid on each horse  for (int i = 0; i < numHorses; i++)  {  cout <<"Which horse do you want to bet on? (1-10) "; cin >> input;  tempWhichHorse[i] = input;  cout << "How much do you want to bet? "; cin >> input;  tempHowMuch[i] = input;    player.setIGCurrency(player.getIGCurrency() - input);  cout << "Thank you" << endl;  }    int winnerHorse = rand() % 10+1;  int reward;    cout << "Winner is horse: " << winnerHorse << endl;  for(int i = 0; i < numHorses; i++)  {  if(tempWhichHorse[i] == winnerHorse)  {  reward = tempHowMuch[i] \* multiplier;  flag = 1;  break;  }  }    if(flag)  {  cout << "Congratulations, you won: " << reward << endl;  player.setIGCurrency(player.getIGCurrency() + reward);  }  else  cout << "Sorry, you didn't win anything" << endl;    player.saveGame();  cout << "Please proceed to return to Choose Game Menu..";  cin.ignore();  cin.get();  chooseGame(player);  }  void topUpCredits(UserData &player)  {  system("CLS");  string userInput;  int value, nContents = 0; //nContents stores number of contents the file  string allInVoucher[600]; //number of codes exist when it was created was 300 + amount (300) = 600  cout << "-----------TOP UP CREDITS-----------" << endl;  cout << "Insert code: "; cin >> userInput;    fstream fileOp;  fileOp.open("Voucher.txt", ios\_base::in | ios\_base::out);  bool flag = false;  string temp;    while(fileOp >> temp)  {  if(userInput == temp) //if the voucher code is correct  {  fileOp >> temp; //retrieve the value below the code  value = atoi (temp.c\_str());  flag = true;  }  else//else, put the content in allInVoucher  {  allInVoucher[nContents] = temp;  nContents++;  }  }  fileOp.close();    cout << "Processing ------------" << endl;    if(flag){  //rewrite the file  fstream fileOp;  fileOp.open("Voucher.txt", ios\_base::in | ios\_base::out);    for(int j = 0; j < nContents; j++)  {  fileOp << allInVoucher[j] << "\n";  j++;  fileOp << allInVoucher[j] << "\n\n";  }  fileOp.close();    player.setIGCurrency(player.getIGCurrency() + value);  cout << "Congratulations your In-game currency balance is now " << player.getIGCurrency() << endl;  }  else  {  cout << "Sorry the code you inserted is wrong" << endl;  }  player.saveGame();  cout << "Returning to Choose Game" << endl;  cin.ignore();  cin.get();  chooseGame(player);  } |

voucher\_code\_generator.cpp

|  |
| --- |
| #include<iostream>  #include<ctime>  #include<string>  #include<cstring>  #include<fstream>  using namespace std;  int main(){  string code;  int amount = 10;  static const char insideRandom[] =  "0123456789"  "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  "abcdefghijklmnopqrstuvwxyz";    srand(time(0));  int codeLength = 12; //12 random characters  int possRandom = strlen(insideRandom);    fstream fileOp;  fileOp.open("Voucher.txt");  for(int x = 0; x < 300; x++)  {  for(int i = 0; i < codeLength; ++i)  {  code += insideRandom[rand() % possRandom];  }  fileOp << code << "\n";  code = "";  fileOp << amount << "\n\n";  if(x%5 == 0)  amount+=10;  }  fileOp.close();  cout << "Generate voucher codes successful!" << endl;  return 0;  } |

# About Writer

Name: Bernardus Jeffrey Darmawan

Student of: BINUS University International

Major: Computer Science

Batch: 2020

Github address: github.com/BJeffreyDarmawan