

College of Computer Studies

**COMPUTER PROGRAMMING 1**

**(CCS0006)**

**C++ Banking System**

**GRADE**

*Submitted by:*

1X1

PICTURE

**

|  |  |  |  |
| --- | --- | --- | --- |
| Babasa, Victoria Faye  1-BSCSSE  TN 01 - C | Padilla, Greg Allen  1-BSCSSE  TN 01 - C | Punzalan, John Christopher  1-BSCSSE  TN 01 - C | Signo, John  Benedict C.  1-BSCSSE  TN 01 - C |
| *Submitted to:*  **Doc. Hadji Javier Tejuco**  Professor  November 30, 2021 | | | |

1. Introduction

A banking transaction is a complicated process that would be difficult for a human to be responsible for it because it would be highly prone to errors – intentional or not – therefore, it is crucial to automate it with the use of programming. This project will serve as an exemplification of what a less error-free banking system would look like. This program will ensure to provide the means of a simple bank transaction with the use of C++ language.

1. Description of the Project

The project is a banking system that was programmed using C++ language. It is a simple system that enables a user to withdraw or deposit a specific number of credits (measured in dollars), or check their remaining credit points. In order for a user to test every feature of the system, they must have an existing account first, which the system offers too – A registration feature that gathers the user input and utilizes it to make an account. The program system also offers an administrator tool that enables an authorized user to delete and view every information of an account, which can be used in maintenance.

.

1. Objectives

The intended goal of this project is to execute the following:

1. To provide a user with an automated way of using and managing their bank account.
2. To simulate a scenario of real-life banking transactions.
3. To apply every concept and technique of C++ programming that the contributors of the project learned from Computer Programming 1 under Doc. Hadji Tejuco.
4. Significance of the Study

The outcome of this project has a potential contribution on the body of knowledge and a beneficial effect to the following areas:

**Starting Programmers.** Since the system project is not much complicated, it would benefit anyone who is trying to learn to code. The system’s source code contains comment lines that explain the purpose of a specific code block, it can be used in learning the structure of the C++ language.

**Banking Industry.** The system project may not be good as a system of a modern bank has but it still contains information that could contribute to the development of a bank system.

**System Project Developers.** The project is still in its early stages, its system is flawed, but undeniably has potential, especially for any programmer that searching for a program to develop. This project can be continued by anyone or even its contributors could use it in their future projects.

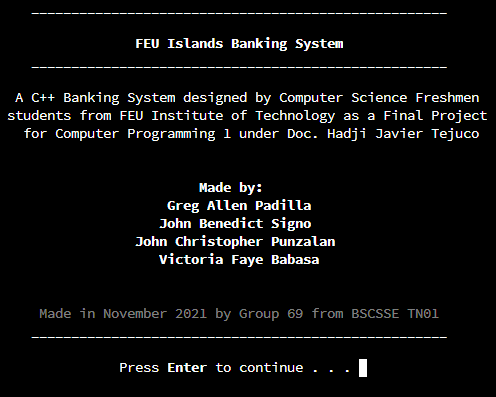
1. Scope and Delimitations

The project used C++ language and utilized its intermediate concepts to develop its banking system. The project was created in the first semester of Academic Year 2021-2022 under the FEU Institute of Technology’s blended online learning.

The contributors of this project concentrated on developing a banking system that contains major features such as withdrawal/deposit, registration/deletion of an account, and listing of existing accounts. The project considered providing a better user interface but was limited because the required IDE does not provide the desired library. The project is delimited in providing a unique identifier – used in finding a specific existing account – by the use of a random number generator. Any other form of unique identifiers such as username, password, or PIN is excluded.

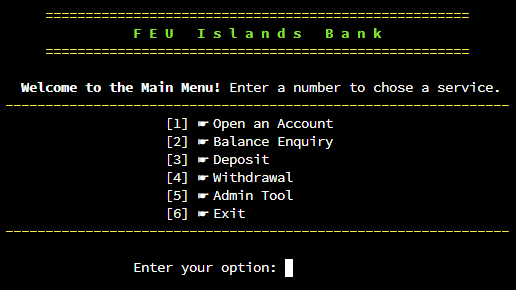
1. Screen Output

The following images are screenshots of every screen output of the system project:



**Figure 1. Introduction of the Project and its Contributors**

Figure 1 contains the introductory information about the project. It displays the Contributors, their course and section, date of completion, and the subject and its professor. The screen is paused until the user has pressed the enter key, which will clear the screen and proceed the user to the Main Menu.

****

**Figure 2. Main Menu**

Figure 2 presents the main menu of the program. It contains a display of the header (FEU Islands Bank) and the different options that a user selects in order to use the program. The selected option is entered in the “Enter your option: “field.

The Main Menu includes the following:

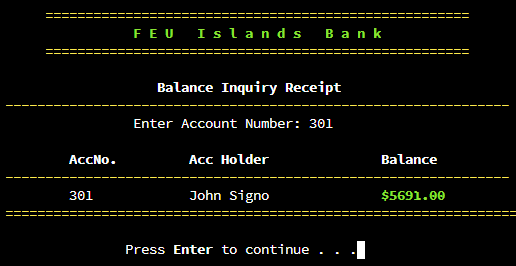
1. **Open an Account –** this menu creates an account for the user that includes a generated account number, account holder’s name, and initial deposit of the holder.
2. **Balance Enquiry** – this menu retrieves and displays the account information of an account holder. The program will ask the user to enter the account number to be searched.
3. **Deposit** – This menu deposits funds to a user’s account. The deposit menu will ask the user to enter the account number to deposit funds into.
4. **Withdrawal** – This menu withdraws funds from a user’s account. The withdrawal menu will ask the user to enter the account number from which to withdraw funds from.
5. **Admin tool** – This menu contains functions exclusive only to administrators. The admin functions include the deletion of accounts and showing the list of all account holders. The admin tool will ask the user to enter a security key to access the admin menu.
6. **Exit** – This option quits the program. Before quitting, it will display an ending message to the user that includes contact information of the program developers.

Text

Description automatically generated

**Figure 3. Registration of Account**

Figure 3 presents the screen output for account registration. After selecting ‘1’ in the main menu, the program will take you to the account creation page. In this section, the program will generate a random three-numbered combination that will serve as a unique identifier of an account. The program will then by ask for the user’s input on their desired Account Name and an Initial Deposit. Each of the inputs will be saved to the database that will serve as a reference



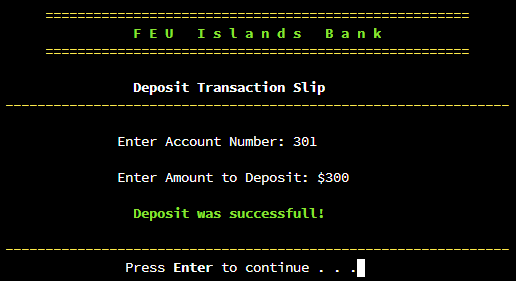
**Figure 4.1. Inquiry of Remaining Credits (Valid Input)**

Figure 4.1 presents the screen output for Balance Inquiry. After selecting ‘2’ in the main menu, the program will take you to the balance Inquiry form. This outcome can only be attained if the Account Number inputted is valid – it must exist on the database. It prints out information such as Account Number, Account Name, and the Remaining Balance.



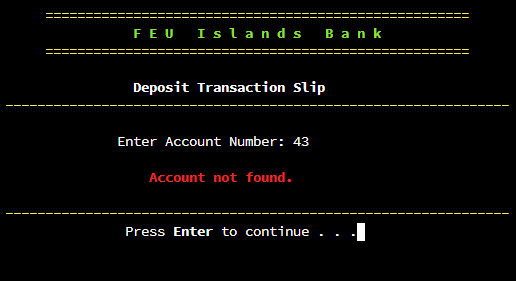
**Figure 4.2. Inquiry of Remaining Credits (Invalid Input)**

Figure 4.2 presents the alternate screen output of the Balance Inquiry page. This alternative screen output will only display if a user inputted an invalid Account Number – the account number doesn’t exist on the database.



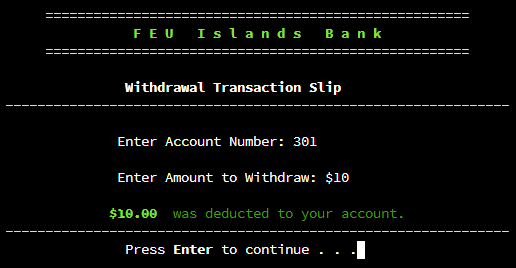
**Figure 5.1. Deposit (Input Exist)**

Figure 5.1 presents the screen output for the Deposit transaction page. In this area, a user can add money to their account by inputting their Account Number and amount of money to be deposited. The inputted amount to deposit will be added directly to the remaining balance of their account.

****

**Figure 5.2. Deposit (Input Does Not Exist)**

Figure 5.2 presents the alternative screen output for the deposit transaction page. This alternative screen output will only display if a user had inputted an invalid Account Number.

****

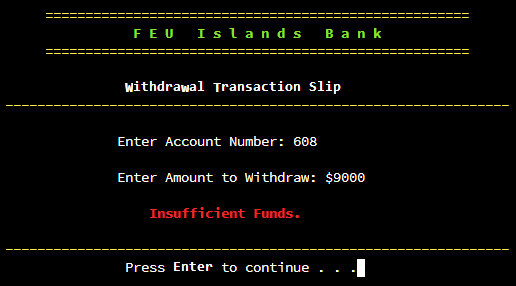
**Figure 6.1. Withdrawal (Account Number Input Exist)**

Figure 6.1 presents the screen output of the Withdrawal page. The 4th option of the main menu is a Withdraw section. Unlike the Deposit section, in this area, the Amount to Withdraw that a user input will be deducted directly from the addressed account by the inputted Account Number.

****

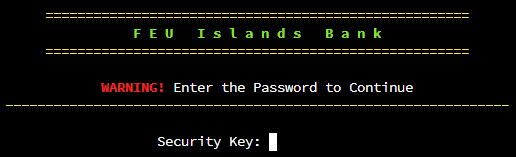
**Figure 6.2. Withdrawal (Account Number Input Does Not Exist)**

Figure 6.2 presents an alternative screen output of the Withdrawal transaction page. This alternative screen output will only display if the Account Number inputted is invalid.

****

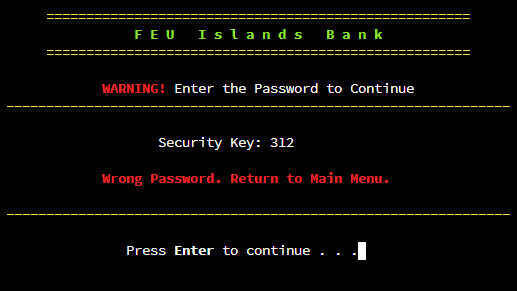
**Figure 6.3. Withdrawal (Withdraw Input is Invalid)**

Figure 6.3 presents another alternative screen output of the Withdrawal transaction page. This is the third alternative display of this section, which will only happen if the Amount to Withdraw input is higher than the remaining balance – the program uses the database as a reference for the information.

****

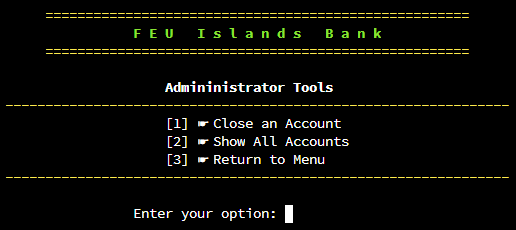
**Figure 7.1. Admin Tool Login (Input Screen)**

Figure 7.1 presents the main screen output of the admin tools login page. This output is the 5th option from the Main Menu. In this section, the user will be required to enter a security key – combination was set by the developers.



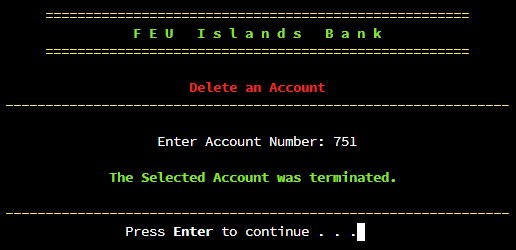
**Figure 7.2. Admin Tool Login (Failed Input)**

Figure 7.2 presents the alternative screen output of the admin tools login page. This alternative screen output only displays if the inputted security key is wrong. By pressing Enter, the program will return the user to the Main Menu.



**Figure 8. Administrator Tools Main Menu**

Figure 8 presents the admin menu screen output. After succeeding in inputting the right password in the admin login, the program will take you to the admin menu. It has a list of available options that only an administrator can use.



**Figure 9.1. Deletion of Account (Input Exist)**

Figure 9.1 presents the account deletion page. The first option available on the Admin Tool Menu has a Deleting feature. A user will be able to delete any account by inputting its respective Account Number. The program will search through the database for the addressed Account Number.

Text

Description automatically generated

**Figure 9.2. Deletion of Account (Input Does Not Exist)**

Figure 9.2 presents the alternative screen output for the account deletion page. The program will display this alternative screen output if the inputted Account Number is not valid, that is, it does not exist on the account list.

****

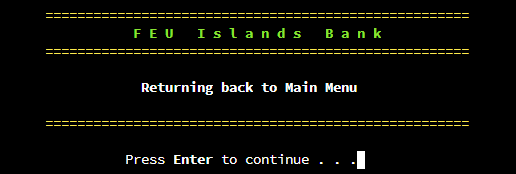
**Figure 10.1. List of Accounts (Exist)**

Figure 10.1 presents the main screen output of the show all accounts page. The 2nd option of the Admin Tool Menu is the show all accounts function. The program will display the complete list of all existing accounts from the database.



**Figure 10.2. List of Accounts (Does Not Exist)**

Figure 10.2 presents the alternative screen output of the show all accounts page. If the program did not detect any existing accounts from the database, an output like this will be displayed on the screen.

****

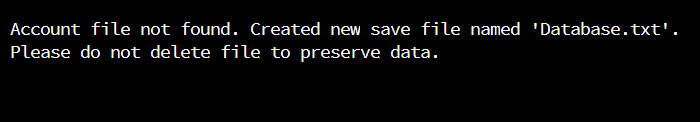
**Figure 11. Return to Main Menu Screen**

Figure 11 presents the return page of the Admin Tool menu. After selecting the 3rd option from the Admin Tool menu, the program will display this output before returning the user to the Main Menu.



**Figure 12. Exit Display**

Figure 12 presents the screen output of the Main Menu Exit option. The last option [6] is an exit button. If the user has selected it, the program would display an ending message containing the developers’ contact information on the screen before the program completely exits.



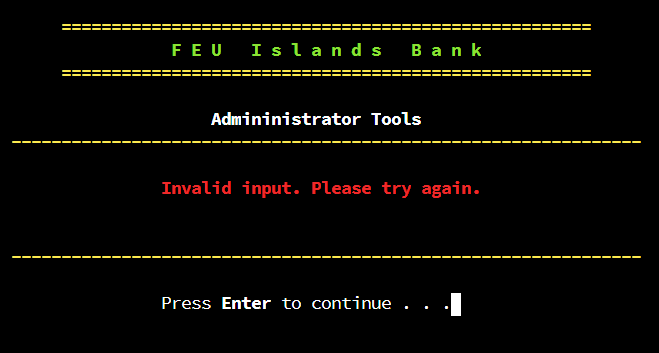
**Figure 13. Missing Database**

Figure 13 presents a screen output of a missing Database text file. This output will display if the program failed to detect the “database.txt” file near it.

****

**Figure 14.1 Invalid Input (Main Menu)**

Figure 14.1 presents the screen output of an invalid input on the Main Menu. If a user typed a number that is not on the list from the Main Menu – Negative integers, Numbers with Decimal Points, Alphabets, Special Characters, or Integers higher than 6 – an output like this will display on the screen.



**Figure 14.2 Invalid Input (Admin Tool Menu)**

Figure 14.2 presents the screen output of an invalid input on the Administrator Tools. The reason of this output is similar to what Figure 14.1 addresses.

1. Source Code

/\*

C++ Banking System | FEU Institute of Technology (2021)

Made by: Greg Allen Padilla, John Benedict Signo, John Christopher Punzalan, Victoria Faye Babasa

CCS006L Final Project

 \*/

#include <iostream>

#include <unistd.h>

#include <iomanip>

#include <fstream>

#define max 75

using namespace std;

// this block of code is for tab spacing

string tab = "\n\t ";

// this block of code is an array that is used to store user data

string accountNo[max] = {};

string name[max]   = {};

double money[max]  = {};

double inimoney[max]  = {};

// these codes are the function prototypes

void headerTitle();

void openAccount();

void deposit();

void withdraw();

void adminLog();

void adminMenu();

void showAccounts();

void searchAccount();

void closeAccount();

void saveToFile();

void loadFile();

void initialize();

void loadingscreen();

void generateAccNo(string &var);

// this block of code is the Main Function

int main() {

   system("clear||cls");

   initialize();

   loadingscreen();

   loadFile();

   int option;

   cout << fixed << setprecision(2);

    do { // Main Menu

        headerTitle();

        cout << tab << "  \e[1mWelcome to the Main Menu!\e[0m Enter a number to choose a service."

             << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m"

             << tab << "\t             [1] ☛ Open an Account"

             << tab << "\t             [2] ☛ Balance Inquiry"

             << tab << "\t             [3] ☛ Deposit"

             << tab << "\t             [4] ☛ Withdrawal"

             << tab << "\t             [5] ☛ Admin Tool"

             << tab << "\t             [6] ☛ Exit"

             << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

        cout << tab << "\t\t Enter your option: "; cin >> option;

        system("clear||cls");

        switch(option) {

            case 1: openAccount();      break;

            case 2: searchAccount();    break;

            case 3: deposit();          break;

            case 4: withdraw();         break;

            case 5: adminLog();         break;

            case 6: // this function Exits the Display

                    headerTitle();

                    cout << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m\n"

                         << tab << "\t   Thank you for trusting the FEU Islands bank!"

                         << tab << "\t If you have any concerns reguarding the program"

                         << tab << "\t    feel free to reach us out in the following: \n"

                         << tab << "\t\t \033[1;31m\e[1mG-Mail:\033[0m GregKalbo69@gmail.com"

                         << tab << "\t      \033[1;35mContact Number:\033[0m (+63)0932-8752-354"

                         << tab << "\t\t  \033[1;34mFacebook Page:\e[0m\033[0m HadjinaticsTM \n"

                         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl; break;

            default: // this function displays invalid input (7,…)

headerTitle();

cout << tab << " \e[1mWelcome to the Main Menu!\e[0m Enter a number to choose a service."

<< tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m"

<< tab << "\t"

<< tab << "\t"

<< tab << "\t"

<< tab << "\t \033[1;31mInvalid input. Please try again.\033[0m\n"

<< tab << "\t"

<< tab << "\t"

<< tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

break;

        }

        // this block of code pauses the screen then clears it right after

        cout << tab << "\t\tPress \e[1mEnter\e[0m to continue . . .";

        cin.ignore();

        cin.get();

        system("clear||cls");

    } while (option != 6);

    return 0;

}

// this block of code adds the header

void headerTitle() {

        cout << tab << "     \e[1m\033[1;33m====================================================="

             << tab << "\t         \033[1;32mF E U   I s l a n d s   B a n k \033[0m    "

             << tab << "     \033[1;33m=====================================================\e[0m\033[0m" << endl;

}

// this block of code calls the function adminLog() which allows the admin to login in the admin tools

void adminLog() {

    int sk;

    headerTitle();

    cout << tab << "\t     \033[1;31m\e[1mWARNING!\e[0m\033[0m Enter the Password to Continue"

         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

    cout << tab << "\t\t    Security Key: "; cin >> sk;

    // this block of code is responsible for verifying the password input for the admin login.

    if(sk == 6132) {

        system("clear||cls");

        adminMenu();

    }

    else {

        cout << tab << "\t     \033[1;31mWrong Password. Return to Main Menu.\033[0m\n"

             << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

    }

}

// this block of code is the function adminMenu() which is used for the Admin Tools menu

void adminMenu(){

    int option;

    // this block of code is the Admin Tools Menu

    headerTitle();

    cout << tab << "\t\t     \e[1mAdmininistrator Tools\e[0m "

         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m"

         << tab << "\t             [1] ☛ Close an Account"

         << tab << "\t             [2] ☛ Show All Accounts"

         << tab << "\t             [3] ☛ Return to Menu"

         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

    cout << tab << "\t\t Enter your option: "; cin >> option;

    system("clear||cls");

    switch(option) {

            case 1: closeAccount();    break;

            case 2: showAccounts();    break;

            case 3: // this block of code returns the admin to the main menu

                    headerTitle();

                    cout << tab << "\t\t  \e[1mReturning back to Main Menu\e[0m "

                         << tab

                         << tab << "     \033[1;33m=====================================================\e[0m\033[0m" << endl;

                    break;

            default: // this function displays when the input is invalid

headerTitle();

cout << tab << "\t\t \e[1mAdmininistrator Tools\e[0m "

<< tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m"

<< tab << "\t"

<< tab << "\t \033[1;31mInvalid input. Please try again.\033[0m\n"

<< tab << "\t"

<< tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

break;

    }

}

// this block of code creates the accounts and stores it in the array

void openAccount() {

    string createID;

    string createName;

    double createMoney;

    headerTitle();

    cout << tab << "\t\t\e[1mRegistration Information Receipt\e[0m"

         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

    generateAccNo(createID);

    cout << "\n\t\t Account number generated: " << createID << endl;

    cin.ignore();

    cout << tab << "\t Enter an Account Name: "; getline(cin, createName);

    cout << tab << "\t Enter an Initial Deposit: $"; cin >> createMoney;

    if(cin.fail()) // this prevents saving the file when istream fails

        return;

    cout << tab << "\t\t\033[1m\033[32mAccount successfully created.\033[0m\n"

         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m"

         << tab << " \e[1mREMINDER!\e[0m Always remember every information in this receipt"

         << tab << "   because it is required in Managment or the retrieval of your"

         << tab << "   account in an event of a loss. Never share it to anyone." << endl;

    int i = 0;

    // this block of code finds an empty array to store

    while(accountNo[i] != "\0") {

        i++;

    }

    accountNo[i] = createID;

    name[i] = createName;

    money[i] = createMoney;

    inimoney[i] = createMoney;

    saveToFile();

}

// this block of code shows all the accounts inside the Banking System

void showAccounts() {

    int i = 0;

    const char separator    = ' ';

    const int accnWidth     = 10;

    const int nameWidth     = 12;

    const int nameWidth2    = 25;

    const int numWidth      = 3;

    // this block of code displays all the accounts in a tabular format

    headerTitle();

    cout << tab << "\t\t       \e[1mList of Accounts\e[0m"

         << tab << "  \e[1m\033[1;33m-----------------------------------------------------------\033[0m"

         << tab << "  no.    Account No.    Account Holder             Balance   "

         << tab << "  \e[1m\033[1;33m-----------------------------------------------------------\033[0m" << endl;

    if (accountNo[0] == "\0")

        cout << tab << "\t\t\033[1;31m There Isn't Any Existing Account.\n"

             << tab << "  \e[1m\033[1;33m-----------------------------------------------------------\033[0m" << endl;

    while(accountNo[i] != "\0") {

        cout << " ";

        cout << left << "\t   " << setw(accnWidth) << setfill(separator) << i+1;

        cout << left << setw(nameWidth) << setfill(separator) << accountNo[i];

        cout << left << setw(nameWidth2) << setfill(separator) << name[i];

        cout << left << setw(numWidth) << setfill(separator) << "$" << money[i] << endl;

        i++;

        cout << tab << "  \e[1m\033[1;33m-----------------------------------------------------------\033[0m" << endl;

    }

}

// this block of code searches the registered accounts from the array

void searchAccount() {

    string search;

    int i = 0, counter = 0;

    headerTitle();

    cout << tab << "\t\t    \e[1mBalance Inquiry Receipt\e[0m "

         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

    cout << tab << "\t\t Enter Account Number: "; cin >> search;

    cout << tab << "\t \e[1mAccNo.\t        Acc Holder      \tBalance\e[0m"

         << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

    for(int i = 0; accountNo[i] != "\0" ; i++) {

        if(accountNo[i] == search) {

            cout << tab << "\t " << accountNo[i] << "      \t" << name[i] << "\t\t\033[1;32m$" << money[i]

                 << tab << "\033[0m\033[1;33m================================================================\e[0m\033[0m\n";

            counter++;

        }

    }

    if (counter == 0)

        cout << tab << "\t\t     \033[1;31mAccount not found.\033[0m\n"

             << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

}

// this block of code is the one responsible for the Deposit Funds Function

void deposit() {

   string search;

   double deposit;

   int counter = 0;

   headerTitle();

   cout << tab << "     \t\t\e[1m Deposit Transaction Slip\e[0m"

        << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

   cout << tab << "\t       Enter Account Number: "; cin >> search;

   // this block of code loops through arrays and find account

   for(int i=0; i <= max; i++) {

       if (accountNo[i] == search) {

           cout << tab << "\t       Enter Amount to Deposit: $"; cin >> deposit;

           money[i] += deposit;

           counter++;

           cout << tab << "\t\t \033[1m\033[32mDeposit was successfull!\033[0m\n"

                << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

           break;

       }

   }

   if(counter == 0) {

       cout << tab << "\t\t   \033[1;31mAccount not found.\033[0m\n"

            << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

       return;

   }

    // this block of code prevents saving the file when istream fails

    if (cin.fail())

        return;

   saveToFile();

}

// this block of code is the one responsible for Withdrawing Funds Function

void withdraw() {

   string search;

   double withdraw;

   int counter = 0;

   headerTitle();

   cout << tab << "     \t\t\e[1mWithdrawal Transaction Slip\e[0m"

        << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

   cout << tab << "\t       Enter Account Number: "; cin >> search;

   // this block of code loops through arrays and find account

   for(int i=0; i <= max; i++) {

       if (accountNo[i] == search) {

           cout << tab << "\t       Enter Amount to Withdraw: $"; cin >> withdraw;

           if (withdraw > money[i]) {

                cout << tab << "\t\t   \033[1;31mInsufficient Funds.\033[0m\n"

                     << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

            } else {

                money[i] -= withdraw;

                cout << tab << "\t      \033[1m\033[32m$" << withdraw << "\033[0m \033[0m\033[32m was deducted to your account.\033[0m"

                << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

            }

           counter++;

           break;

       }

       // this block of code ends the function when input fails and prevents saving the file

       if (cin.fail())

        return;

   }

   if(counter == 0) {

       cout << tab << "\t\t    \033[1;31mAccount not found.\033[0m\n"

            << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

       return;

   }

    saveToFile();

}

// This block of code Deletes the Account

void closeAccount() {

    string search;

    int counter = 0;

    headerTitle();

    cout << tab << "\t\t\t\e[1m\033[1;31mDelete an Account\033[0m\e[0m"

        << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m" << endl;

    cout << tab << "\t\t    Enter Account Number: "; cin >> search;

    if(cin.fail())

        return;

    for(int i = 0; accountNo[i] != "\0"; i++) {

        // this block of code removes the account from the array if the account exists

        if (accountNo[i] == search)

        {

            // this block of code removes the account by shifting array elements

            for (int j=i; accountNo[j] != "\0"; j++)

            {

                accountNo[j] = accountNo[j + 1];

                name[j] = name[j + 1];

                money[j] = money[j + 1];

            }

            // this block of code updates the counter and stops the search

            counter++;

            break;

        }

        }

    // this block of code displays appropriate end prompts

    if (counter == 0)  {

        cout << tab << "\t\t   \033[1;31mThe Account does not exist.\033[0m\n"

             << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

        return;

    } else {

        cout << tab << "\t      \033[1;32mThe Selected Account was terminated.\033[0m\n"

             << tab << "\e[1m\033[1;33m---------------------------------------------------------------\e[0m\033[0m";

    }

    saveToFile();

}

// this function displays info on startup

void initialize() {

    // this function fixes the text color for the windows operating system

    #if defined \_WIN32

        system("Color 0F");

    #endif

    cout << "\n\t     \e[1m\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

         << "\n\t\t                                        "

         << "\n\t\t          FEU Islands Banking System     "

         << "\n\t     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\e[0m" << endl;

    cout << tab << "  A C++ Banking System designed by Computer Science Freshmen"

         << tab << " students from FEU Institute of Technology as a Final Project"

         << tab << "   for Computer Programming 1 under Doc. Hadji Javier Tejuco"

         << tab

         << tab

         << tab << "\t                  \e[1mMade by:"

         << tab << "\t              Greg Allen Padilla"

         << tab << "\t             John Benedict Signo"

         << tab << "\t          John Christopher Punzalan"

         << tab << "\t             Victoria Faye Babasa\e[0m"

         << tab

         << tab

         << tab << "     \e[2mMade in November 2021 by Group 69 from BSCSSE TN01\e[0m";

    cout << "\n\t     \e[1m\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\e[0m" << endl

         << "\n\t\t        Press \e[1mEnter\e[0m to continue . . . ";

    cin.get();

    system("clear||cls");

}

// this block of code creates a loading screen after the intro page

void loadingscreen() {

    int timer = 2;

    cout << "\n\t Loading please wait ";

    // this block of code is the loading dots

    while(timer != 0) {

        cout << ". ";

        sleep(1);

        timer--;

    }

    cout << endl;

    system("clear||cls");

}

// this block of code saves array data to file

void saveToFile() {

    ofstream masterFile("accounts.txt");

    // this block of code writes array elements into file

    for(int i=0; accountNo[i] != "\0"; i++) {

        masterFile << accountNo[i] << ' ' << name[i] << "$ " << money[i] << endl;

    }

    masterFile.close();

}

// this block of code loads preserved data from account file

void loadFile() {

    ifstream loadData;

    fstream accountFile("accounts.txt", ios::in);

    // this block of code creates file if it does not exist

    if(!accountFile) {

        cout << "\n\t Account file not found. Created new save file named 'accounts.txt'."

             << "\n\t Please do not delete file to preserve data. \n";

        accountFile.open("accounts.txt", ios::in | ios::out | ios::trunc);

        accountFile << "";

        accountFile.close();

        cin.get();

    }

    accountFile.close();

    loadData.open("accounts.txt");

    // this block of code loads file contents in arrays

    int i = 0;

    while(loadData >> accountNo[i]) {

        loadData >> ws;

        getline(loadData, name[i], '$');

        loadData >> money[i];

        i++;

    }

    loadData.close();

}

// this block of code generates the uid for account no

void generateAccNo(string &var) {

    string uid;

    int randonum;

    bool not\_unique = true, found = false;

    // this block of code uses bool variables to check validity of generated uid

    while(not\_unique) {

        uid = "";

        // this block of code generates random uid using rand function

        srand(time(NULL));

        for(int i = 1; i <= 3; i++) {

            randonum = rand()%9;

            uid += to\_string(randonum);

        }

        // this block of code checks whether generated id exists

        for(int i = 0; accountNo[i] != "\0"; i++) {

            if (accountNo[i] == uid) {

                found = true;

                break;

            } else found = false;

        }

        // this block of code quits the loop if generated uid is valid

        if(found == false)

            not\_unique = false;

    }

    // this block of code assigns uid to passed (by ref) variable

    var = uid;

}

1. Conclusion

The complexity of coding a bank system ranges from middle to high level – depending on the number of features that the system will have – because it mirrors an underlying set of problems that need to find solutions for. Examples are, developing a database that serves as the reference of accounts, providing an automated system of utilizing every input of a user (including the errors), and making a user-friendly interface. Therefore, it is crucial for a developer, not only to be familiarized but also to master the structure of a C++ program, because it will be an advantage in debugging the errors and glitches that they will encounter. C++ also features a thousand sets of libraries that serve as a pre-processor directive which will provide another set of commands and codes that will significantly make the system better, interface and integrity wise.