COS10029 Fundamentals of Programming TP 1 2023

Assignment 2 Report

Name: Nidula Mallikarachchi Student ID: 104756611

Lab class: Teacher: Ms. Lushaka Nisansala

Due Date: 10th September 2023 by 23:59

Date Submitted: 10th September 2023

Program 1: Data Encoder

Program Description:

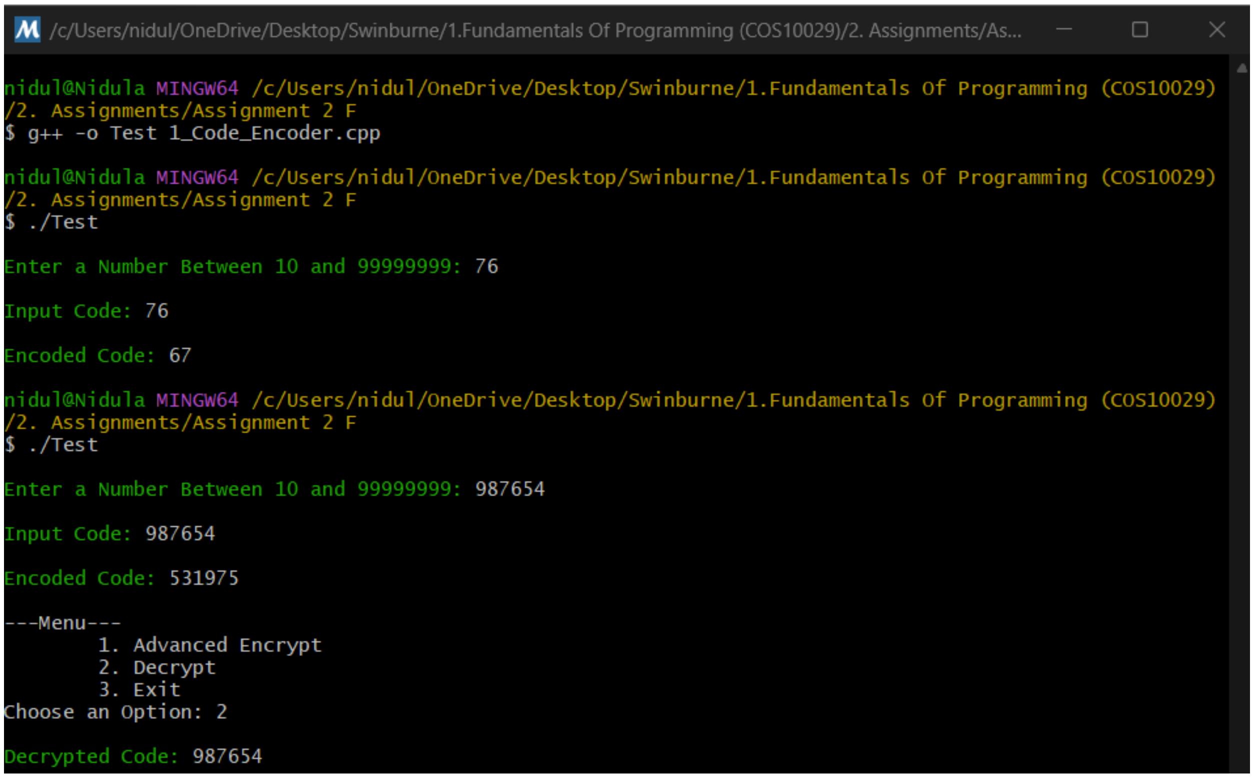
The program is made to encrypt any input code given by the user. The methodology behind the code is explained below.

- The program prompts the user to enter a Number between 10 and 99999999.
- If the user has entered a code which is less than 2 digits or more than 8 digits the program will keep asking for a correct input until inserted.
- After the user Enters the number. The program Stores this number in an array in the reverse order. Then a user defined function called Reverse () is called to order the elements in the array in the correct order.
- If the user entered a number which has 4 characters or more, the program will do the following→
    - Starting from the last index the position of the digit is added to the digit. (Last Position is 1 and First Position is 8)
    - Modulus of the addition is stored in the position of the current digit in the array.
- If the user entered number is less than 4 characters, the program simply reverses the order of the input number and terminates.
- For the numbers which has more than 4 digits, the program offers 3 other options as
    - Advanced Encrypt
    - Decrypt
    - Exit
- The advanced Encrypt Option Will Reverse the order of the already encrypted code.
- The Decrypt Option Will Decrypt the Code into its original Order.
- Exit Option Will terminate the program.

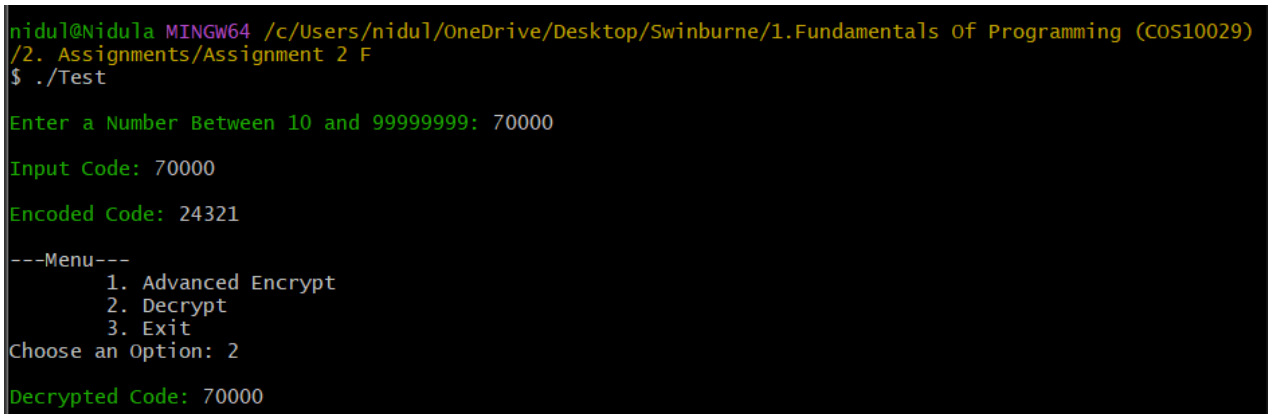
Source Code:

```cpp
#include <iostream>
using namespace std;

// Function Prototypes
void EncodeLargeNumber(int array[], int count);
void DecodeLargeNumber(int array[], int count);
void Reverse(int array[], int count);

int main() {
    int temp;
    int count = 0;
    int array[8];

    int num;
    do {
        cout << "\n\033[32mEnter a Number Between 10 and 99999999: \033[0m";
        cin >> num;
    } while (num < 10 || num > 99999999);

    // Append the number to the array in reverse order
    while (num > 0) {
        temp = num % 10;
        num = num / 10;
        array[count] = temp;
        count++;
    }

    // Reverse the array to get the digits in the correct order
    Reverse(array, count);

    // Display the initial array
    cout << "\n\033[32mInput Code: \033[0m";
    for (int i = 0; i < count; i++) {
        cout << array[i];
    }
    cout << endl;

    int option;

    // Check if count is greater than or equal to 4 (if the count is greater than 4, it is a
number with more than 4 digits and can be advanced encrypted)
    if (count >= 4) {
        EncodeLargeNumber(array, count); // Encrypted

        cout << "\n\033[32mEncoded Code: \033[0m"; // Display Encrypted Code
        for (int i = 0; i < count; i++) {
            cout << array[i];
        }

        cout << endl;

        // Display Menu
        cout << "\n---Menu---\n";
        cout << "\t1. Advanced Encrypt\n";
        cout << "\t2. Decrypt\n";
        cout << "\t3. Exit";

        do {
            cout << "\nChoose an Option: ";
            cin >> option;

            if (option < 1 || option > 3) {
                cout << "Invalid Option!\n";
            }
        } while (option < 1 || option > 3); // Option Validation

        switch (option) {
            case 1:
                Reverse(array, count);
                cout << "\n\033[32mAdvance Encrypted: \033[0m"; // Advanced Encrypted
                for (int i = 0; i < count; i++) { // Display Advanced Encrypted Code
                    cout << array[i];
                }
                cout << "\n\nDo You Want to Decrypt\n"; // Sub Menu 1
                cout << "\t1. Yes\n";
                cout << "\t2. No\n";

                do {
                    cout << "Option: ";
                    cin >> option;
                    if (option < 1 || option > 2) {
                        cout << "Invalid Option!\n"; // Option Validation
                    }
                } while (option < 1 || option > 2);

                switch (option) {
                    case 1:
                        Reverse(array, count); // Reverse Advanced Encrypted Code
                        DecodeLargeNumber(array, count); // Decode Encrypted Code

                        cout << "\n\033[32mAdvance Decrypted: \033[0m";
                        for (int i = 0; i < count; i++) {
                            cout << array[i];
                        }
                        cout << endl;
                        break;
                    case 2:
                        cout << "\n\033[31mProgram Terminated\033[0m\n";
                        return 0;
                }
                break;

            case 2:
                // Decrypt
                DecodeLargeNumber(array, count);
                cout << "\n\033[32mDecrypted Code: \033[0m";
                for (int i = 0; i < count; i++) {
                    cout << array[i];
                }

                cout << endl;
                break;
            case 3:
                cout << "\n\033[31mProgram Terminated\033[0m\n";
                return 0;
            default:
                cout << "\033[31mInvalid Option!\033[0m";
                break;
        }
    } else {
        Reverse(array, count);
        cout << "\n\033[32mEncoded Code: \033[0m";
        for (int i = 0; i < count; i++) {
            cout << array[i];
        }
        cout << endl;
    }

    return 0;
}

// Function to encode a large number
void EncodeLargeNumber(int array[], int count) {
    int digit = 1;

    for (int i = count - 1; i >= 0; i--) {
        array[i] = (array[i] + digit) % 10;
        digit++;
    }
}

// Function to decode a large number
void DecodeLargeNumber(int array[], int count) {
    int digit = 1;

    for (int i = count - 1; i >= 0; i--) {
        int temp = array[i] - digit;
        array[i] = (temp < 0) ? (array[i] + 10 - digit) : temp;
        // if short-hand property
        // (is_condition_true) ? (if_true_Do_This) : (If_Not_Do_This)
        digit++;
    }
}

// Function to reverse an array
void Reverse(int array[], int count) {
    int start = 0;
    int end = count - 1;

    while (start < end) {
        // Swap the elements at start and end positions
        int temp = array[start];
        array[start] = array[end];
        array[end] = temp;

        // Move the start and end pointers
        start++;
        end--;
    }
}
```

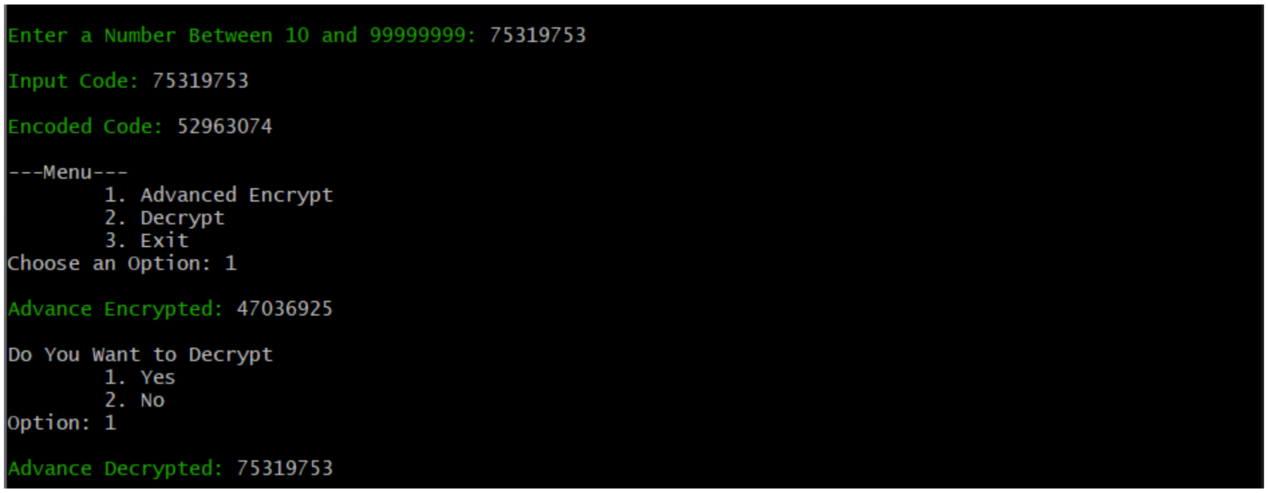
Outputs:



```
/c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)/2. Assignments/As...

nidul@Nidula MINGW64 /c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)
/2. Assignments/Assignment 2 F
$ g++ -o Test 1_Code_Encoder.cpp

nidul@Nidula MINGW64 /c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)
/2. Assignments/Assignment 2 F
$ ./Test

Enter a Number Between 10 and 99999999: 76

Input Code: 76

Encoded Code: 67

nidul@Nidula MINGW64 /c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)
/2. Assignments/Assignment 2 F
$ ./Test

Enter a Number Between 10 and 99999999: 987654

Input Code: 987654

Encoded Code: 531975

---Menu---
        1. Advanced Encrypt
        2. Decrypt
        3. Exit
Choose an Option: 2

Decrypted Code: 987654
```



```
nidul@Nidula MINGW64 /c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)
/2. Assignments/Assignment 2 F
$ ./Test

Enter a Number Between 10 and 99999999: 70000

Input Code: 70000

Encoded Code: 24321

---Menu---
        1. Advanced Encrypt
        2. Decrypt
        3. Exit
Choose an Option: 2

Decrypted Code: 70000
```



```
Enter a Number Between 10 and 99999999: 75319753

Input Code: 75319753

Encoded Code: 52963074

---Menu---
        1. Advanced Encrypt
        2. Decrypt
        3. Exit
Choose an Option: 1

Advance Encrypted: 47036925

Do You Want to Decrypt
        1. Yes
        2. No
Option: 1

Advance Decrypted: 75319753
```

Program 2: Credit Card Validator

Program Description:

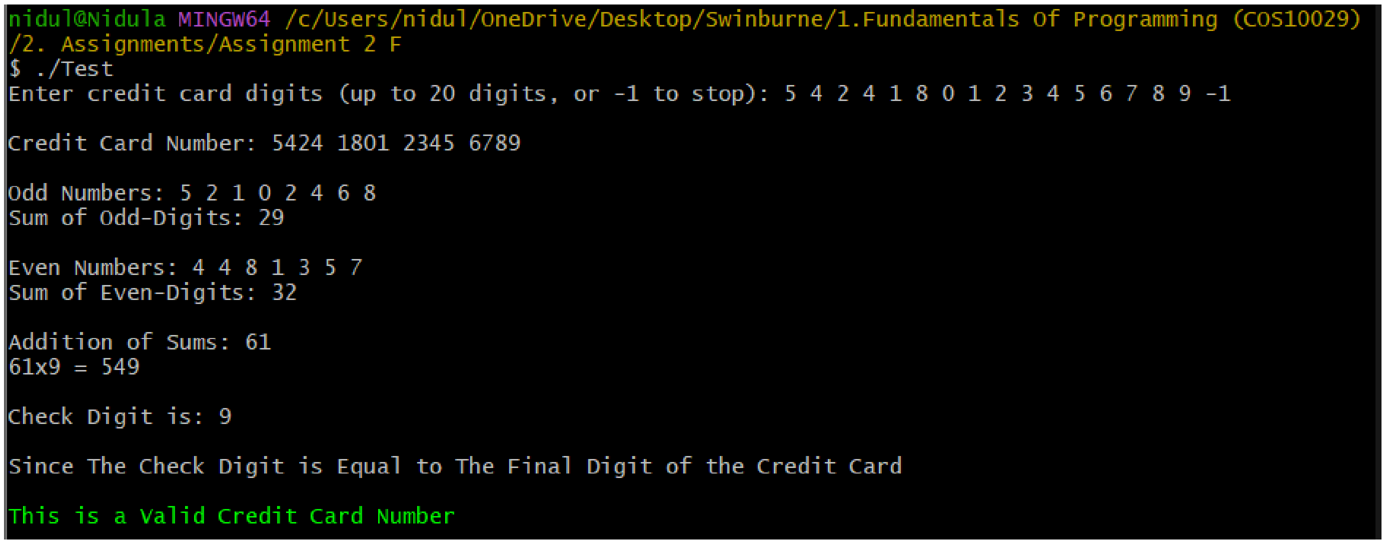
The purpose of the program is to make sure that the user enters a valid Credit Card Number. The process of the Program is Explained below.

- Program asks the user to enter a credit card number. Users can enter a credit card number that has a maximum of 20 digits. Users can enter digits separated with a space and then -1 in the end to terminate the program asking for inputs.
- Assume that the first digit of the Credit card is of index 1 (since in the real world 1st index is called the 1st position)
- Then the sum of the values in Position 2,4,6,8……… are added together and is assigned to a variable named Even_sum (The check digit(last Digit) is not added in this case)
- The values in the odd positions 1,3,5,7………… Are doubled first and then the modulus of those numbers are taken. After that they are added together and is assigned to a variable named Odd_sum (The check digit(last Digit) is not added in this case)
- Addition of these 2 variables are held in a variable called "addition". "addition" is multiplied by 9 and then the modulus of it is taken. This is referred to as the "Check Digit".
- If the Check Digit is equal to the last number of the array (credit card) the Program Verifies that this is a valid Credit Card. If the 2 numbers are different, then the program identifies it as a invalid Credit Card Number.

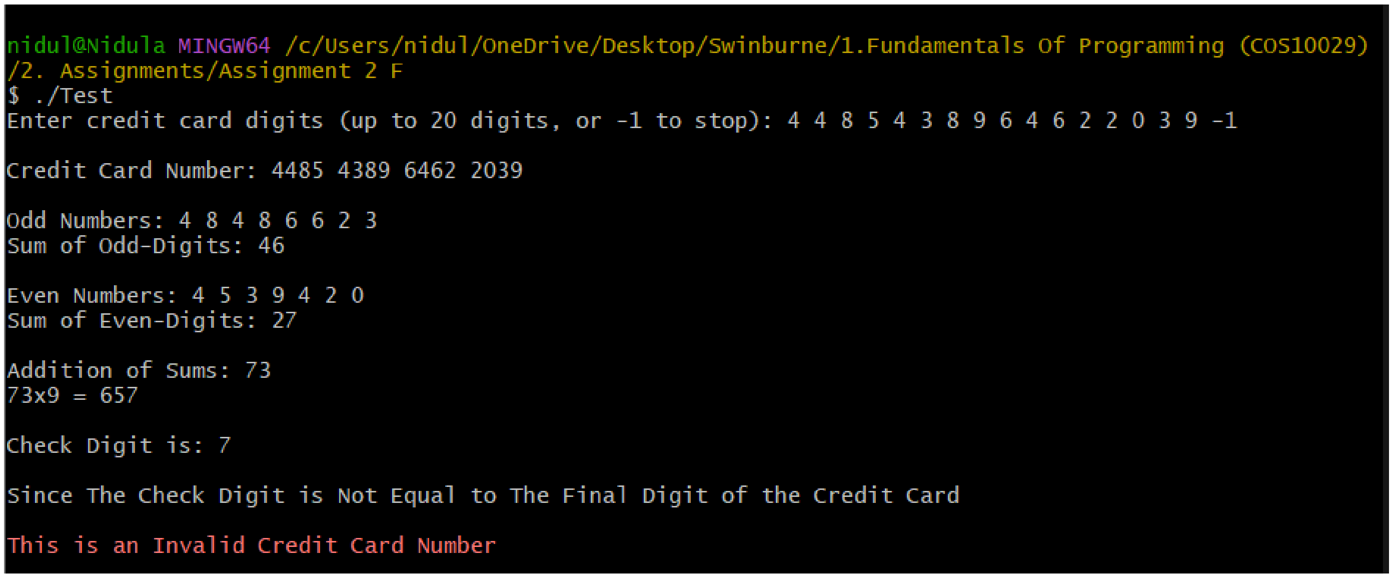
Source Code:

```cpp
#include <iostream>
using namespace std;

const int MAX_DIGITS = 20;

// Function prototypes
int OddNumber(int array[], int count);
int EvenNumber(int array[], int count);
void CheckDigit(int odd_sum, int even_sum, int count, int array[]);

int main() {
    int credit_card[MAX_DIGITS];
    int count = 0;
    int digit;

    // Input credit card digits
    cout << "Enter credit card digits (up to " << MAX_DIGITS << " digits, or -1 to stop): ";
    while (count < MAX_DIGITS) {
        cin >> digit;

        if (digit == -1) {
            break;
        } else {
            credit_card[count] = digit;
            count++;
        }
    }
    // Display the credit card number in the standard form "xxxx xxxx xxxx xxxx"
    cout << "\nCredit Card Number: ";
    for (int i = 0; i < count; i++) {
        cout << credit_card[i];
        if ((i + 1) % 4 == 0 && i < count - 1) {
            cout << " ";
        }
    }

    cout << "\n\n";

    int oddSum = OddNumber(credit_card, count);
    cout << endl;
    int evenSum = EvenNumber(credit_card, count);
    cout << endl;

    int addition = oddSum+evenSum;

    cout<<"Addition of Sums: "<<addition<<endl;

    int multiplication = addition*9;

    cout<<addition<<"x9 = "<< multiplication<<endl<<endl;

    cout<<"Check Digit is: "<<multiplication%10<<endl<<endl;

    CheckDigit(oddSum, evenSum, count, credit_card);

    return 0;
}

// Function to calculate the sum of odd-indexed digits and display them
int OddNumber(int array[], int count) {
    int odd_sum = 0;

    cout << "Odd Numbers: ";

    for (int i = 0; i < count-1; i += 2) {
        int temp = array[i] * 2;
        int quotient = temp / 10;
        int remainder = temp % 10;

        cout << array[i] << " ";

        odd_sum = odd_sum + remainder + quotient;
    }

    cout << endl;
    cout << "Sum of Odd-Digits: " << odd_sum << endl;
    return odd_sum;
}

// Function to calculate the sum of even-indexed digits and display them
int EvenNumber(int array[], int count) {
    int even_sum = 0;

    cout << "Even Numbers: ";

    for (int i = 1; i < count-1; i += 2) {
        cout << array[i] << " ";
        even_sum = even_sum + array[i];
    }

    cout << endl;
    cout << "Sum of Even-Digits: " << even_sum << endl;

    return even_sum;
}

// Function to check the validity of the credit card number
void CheckDigit(int odd_sum, int even_sum, int count, int array[]) {
    int sum = odd_sum + even_sum;

    int check_Digit = (sum * 9) % 10;

    if (check_Digit == array[count - 1]) {
        cout<<"Since The Check Digit is Equal to The Final Digit of the Credit Card\n\n";
        cout << "\033[1;32mThis is a Valid Credit Card Number\033[0m\n\n";
    } else {
        cout<<"Since The Check Digit is Not Equal to The Final Digit of the Credit Card\n\n";
        cout << "\033[1;31mThis is an Invalid Credit Card Number\033[0m\n\n";
    }
}
```

Outputs:



```
nidul@Nidula MINGW64 /c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)
/2. Assignments/Assignment 2 F
$ ./Test
Enter credit card digits (up to 20 digits, or -1 to stop): 5 4 2 4 1 8 0 1 2 3 4 5 6 7 8 9 -1

Credit Card Number: 5424 1801 2345 6789

Odd Numbers: 5 2 1 0 2 4 6 8
Sum of Odd-Digits: 29

Even Numbers: 4 4 8 1 3 5 7
Sum of Even-Digits: 32

Addition of Sums: 61
61x9 = 549

Check Digit is: 9

Since The Check Digit is Equal to The Final Digit of the Credit Card

This is a Valid Credit Card Number
```



```
nidul@Nidula MINGW64 /c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)
/2. Assignments/Assignment 2 F
$ ./Test
Enter credit card digits (up to 20 digits, or -1 to stop): 4 4 8 5 4 3 8 9 6 4 6 2 2 0 3 9 -1

Credit Card Number: 4485 4389 6462 2039

Odd Numbers: 4 8 4 8 6 6 2 3
Sum of Odd-Digits: 46

Even Numbers: 4 5 3 9 4 2 0
Sum of Even-Digits: 27

Addition of Sums: 73
73x9 = 657

Check Digit is: 7

Since The Check Digit is Not Equal to The Final Digit of the Credit Card

This is an Invalid Credit Card Number
```

Program 3: Splashkit Drawing Bars of Random Height Using Structures and Arrays

Program Description:

- The program draws vertical bars of random heights using the help of an array and a structure. The explanation is given below:
- The program defines a structure named "bar_porperties".
- An array of data type bar_properties is created inside the int main () as array_rectangles [].
- A user defined function called fill_array() is used to allocated values to the array. The data array is passed as a parameter for the fill_array ().

In the fill_array():

- 800 values are filled in to the array using the struct. The height of the bar is called "bar_height" and the color of the bar is called "bar_color".
- The height is taken from the rand() function. It is always less than 600px (Since it is divided by screen height - 100).
- A new function called get_random_color(); is called in order to assign random color values to the bars.

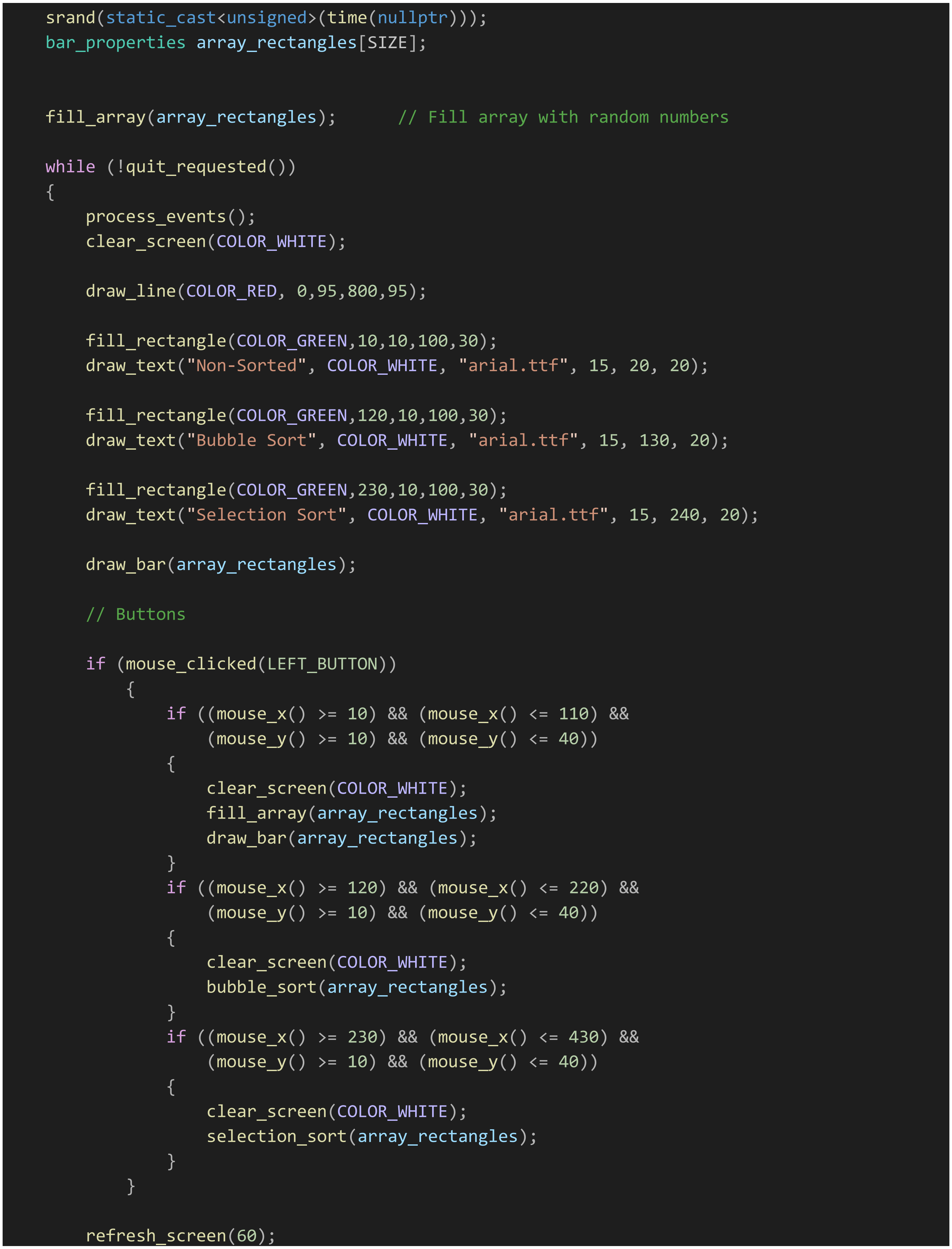
get_random_clr() function:

- Hue is calculated by taking the modulus of a random number (which is of data type int) converted in to float data type. And its value is divided by 360.0.
- In the assignment description it is said to get the hue value by dividing the random value by the screen height. But it will not result in the desired output. Hence, we must change it to the above logic.

Once the data has been filled, the 2 functions bubble sort and the selection sort are used to sort the bars according to the ascending order of their heights.

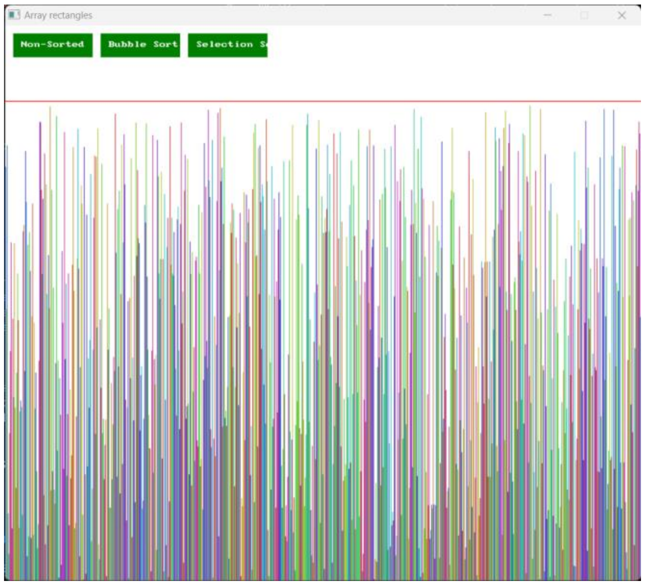
Source Code:

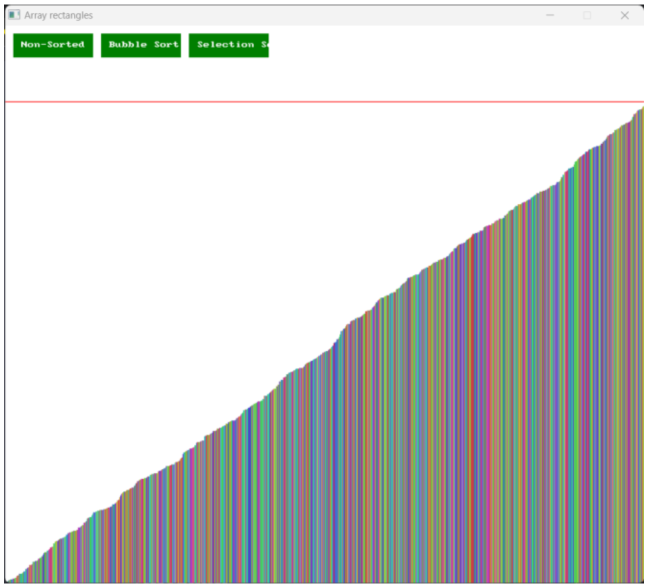
```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include "splashkit.h"

const int SIZE = 800;

// Define the struct for bar properties
struct bar_properties
{
    int bar_height;
    color bar_color;
};

//Function Prototypes
color get_random_color();
void fill_array(bar_properties array_rectangles[]);
void draw_bar(bar_properties array_rectangles[]);
void bubble_sort(bar_properties array_rectangles[]);
void selection_sort(bar_properties array_rectangles[]);

int main()
{
    open_window("Array rectangles", 800, 700);
```

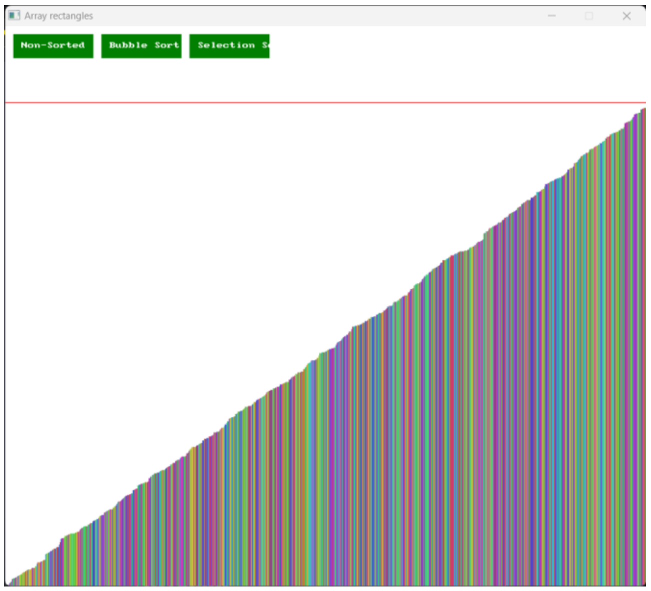


```
    srand(static_cast<unsigned>(time(nullptr)));
    bar_properties array_rectangles[SIZE];


    fill_array(array_rectangles);      // Fill array with random numbers

    while (!quit_requested())
    {
        process_events();
        clear_screen(COLOR_WHITE);

        draw_line(COLOR_RED, 0,95,800,95);

        fill_rectangle(COLOR_GREEN,10,10,100,30);
        draw_text("Non-Sorted", COLOR_WHITE, "arial.ttf", 15, 20, 20);

        fill_rectangle(COLOR_GREEN,120,10,100,30);
        draw_text("Bubble Sort", COLOR_WHITE, "arial.ttf", 15, 130, 20);

        fill_rectangle(COLOR_GREEN,230,10,100,30);
        draw_text("Selection Sort", COLOR_WHITE, "arial.ttf", 15, 240, 20);

        draw_bar(array_rectangles);

        // Buttons

        if (mouse_clicked(LEFT_BUTTON))
            {
                if ((mouse_x() >= 10) && (mouse_x() <= 110) &&
                    (mouse_y() >= 10) && (mouse_y() <= 40))
                {
                    clear_screen(COLOR_WHITE);
                    fill_array(array_rectangles);
                    draw_bar(array_rectangles);
                }
                if ((mouse_x() >= 120) && (mouse_x() <= 220) &&
                    (mouse_y() >= 10) && (mouse_y() <= 40))
                {
                    clear_screen(COLOR_WHITE);
                    bubble_sort(array_rectangles);
                }
                if ((mouse_x() >= 230) && (mouse_x() <= 430) &&
                    (mouse_y() >= 10) && (mouse_y() <= 40))
                {
                    clear_screen(COLOR_WHITE);
                    selection_sort(array_rectangles);
                }
            }

        refresh_screen(60);
    }
    return 0;
}

// Function to get a random color
color get_random_color()
{
    // Generate a random hue bar_height between 0 and 1
    float hue = static_cast<float>(rand() % 360) / 360.0;

    // The When the hue is divided by the Screen Height, The desired output will not come.
Hence divided by 360
    color bar_color = hsb_color(hue, 0.7, 0.8);
    return bar_color;
}

//Fill Array with random Numbers
void fill_array(bar_properties array_rectangles[])
{
    for (int i = 0; i < SIZE; ++i)
    {
        array_rectangles[i].bar_height = rand() % (screen_height() - 100);

        array_rectangles[i].bar_color = get_random_color();
    }
}

// Draw bars
void draw_bar(bar_properties array_rectangles[])
{
    float rectangle_width = 1;
    for (int i = 0; i < SIZE; ++i)
    {
        float x = rectangle_width * i;
        float y = screen_height() - array_rectangles[i].bar_height;

        //Drawing random hue rectangels with the help of struct variable
        fill_rectangle(array_rectangles[i].bar_color, x, y, rectangle_width,
array_rectangles[i].bar_height);
    }
}

// Bubble Sort
void bubble_sort(bar_properties array_rectangles[])
{
    for (int i = 0; i < SIZE - 1; ++i)
    {
        for (int j = 0; j < SIZE - i - 1; ++j)
        {
            if (array_rectangles[j].bar_height > array_rectangles[j + 1].bar_height)
            {
                bar_properties temp = array_rectangles[j];
                array_rectangles[j] = array_rectangles[j + 1];
                array_rectangles[j + 1] = temp;
            }
        }
    }
}

// Selection Sort
void selection_sort(bar_properties array_rectangles[])
{
    for (int i = 0; i < SIZE - 1; ++i)
    {
        int minimum_index = i;
        for (int j = i + 1; j < SIZE; ++j)
        {
            if (array_rectangles[j].bar_height < array_rectangles[minimum_index].bar_height)
            {
                minimum_index = j;
            }
        }
        bar_properties temp = array_rectangles[i];
        array_rectangles[i] = array_rectangles[minimum_index];
        array_rectangles[minimum_index] = temp;
    }
}
```

Output:


Array rectangles
Non-Sorted
Bubble Sort
Selection S


Array rectangles
Non-Sorted
Bubble Sort
Selection S


Array rectangles
Non-Sorted
Bubble Sort
Selection S

Program 4: Music Player

Program Description:

This program acts as a music player. The functionalities of the player are defined below.

Add Song:

- The program lets the user add a Song Album and store it in a file called “Music.txt”.
- The Song information is added to the Album Structure defined in the header of the code.
- An Enum is defined to get the genre of the album.
- Users can Store up to 5 songs maximum in 1 album.

View Album Details:

- When The user enters the name of the album the program retrieves the information stored in the “Music.txt” file and displays it on the terminal about the related album.

Play Song:

When The user Types the Name of The Album and Name of The Song the Correct song will play.

Important:

- If you want to test out this code, I have included a Folder Called Music inside the zip File with 2 Albums named “Classics” and “Pop”.
- The “Classics” album is already added to the “Music.txt” file.

- If you want to check out the add new song functionality do the following steps
  - Select Add New Song from the menu.
  - Enter “Pop” as the album name.
  - Enter “2” for the number of songs in the album.
  - Select Genre as “Pop”
  - Enter Track 1 name as “Bad Habits”.
  - Enter Track 2 name as “New Rules”.

- Since I have included these songs in the folder “Albums” The Program will be able to play them without an issue.

```
-----MENU----
1. Add a New Song
2. View Details of a Selected Album
3. Play a Song
4. Exit
Select an option: 1

Enter Album Name: Pop
How Many Songs does this Album Consist of: 2

Select Genre:
        1. Pop
        2. Jazz
        3. Classic
Option: 1
Enter Track 1 : Bad Habits
Enter Track 2 : New Rules
```

If you want to know what songs are in an album you can do the following steps:

- Select “View Details of a Selected Album” from the menu.
- Enter the Name of the Album you want to display. “Classics” or “Pop”

```
-----MENU----
1. Add a New Song
2. View Details of a Selected Album
3. Play a Song
4. Exit

Select an option: 2
Please Enter the Name of The Album You Want To Find: Classics
```

If you want to play a song do the following steps:

- Select “Play a Song” from the menu.
- Enter the album name “Classics” or “Pop”.
- Enter the name of the song you want to play (Make sure the spellings are correct)
- You don’t have to enter the location because I’ve programmed it to automatically generate the Location.

```
-----MENU----
1. Add a New Song
2. View Details of a Selected Album
3. Play a Song
4. Exit
Select an option: 3
Please Enter the Name of The Album You Want To Find: Classics
Please Enter The Name of the Song You Want To Play: Sleeping Child
```

Please be careful when entering the names of the songs and albums since, if the album names and song names are wrong the program won’t be able to detect the song properly from its location

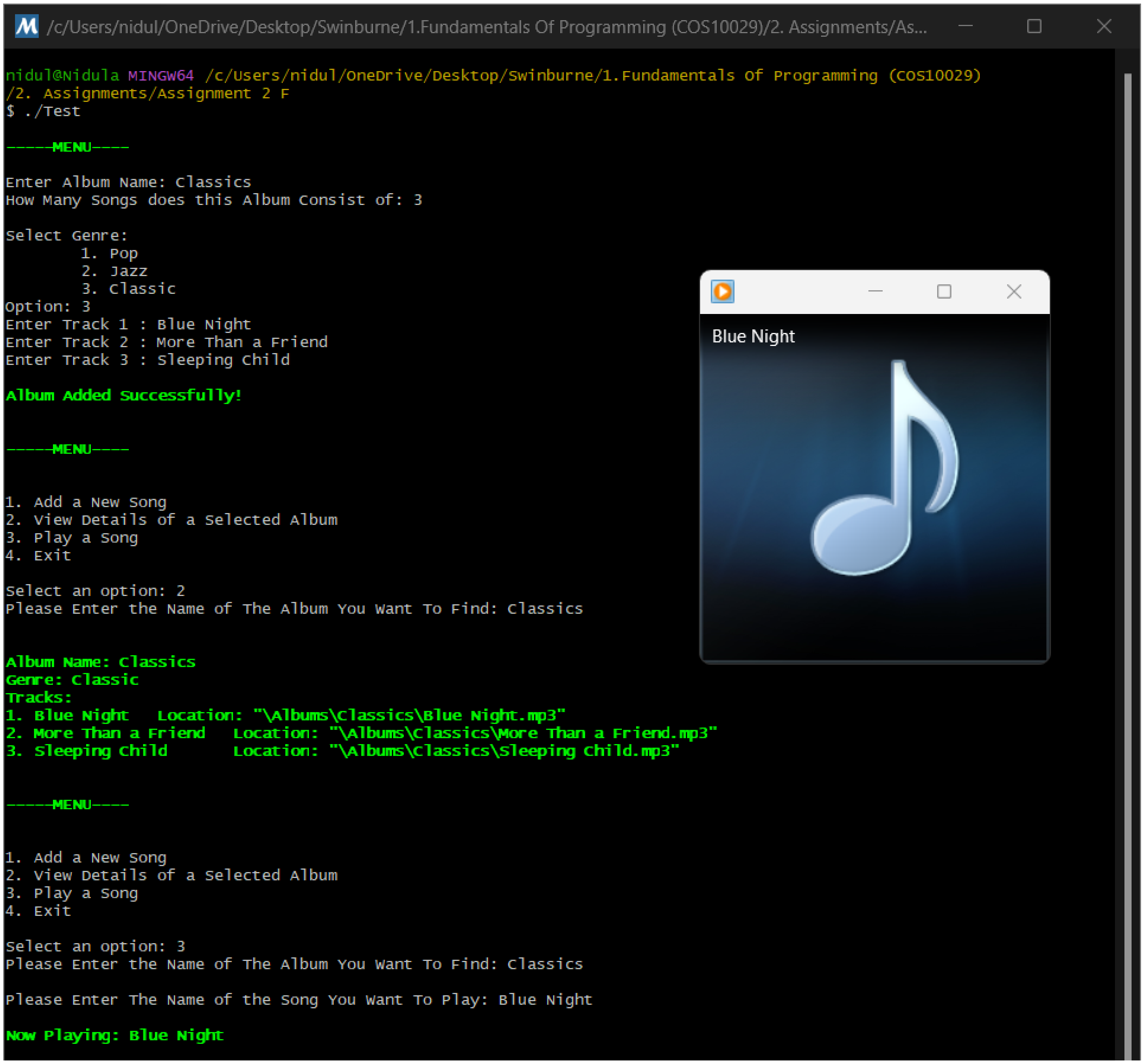
Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>

using namespace std;

// Enum to store music genres
enum Genre { Pop, Jazz, Classic };

// Structure to store album information
struct Album {
    string album_name;
    int number_of_songs;
```

```cpp
    Genre genre;
    int track_number;
    string tracks[5];
    string track_location[5];
};

void AddNewSong(Album& album);
void ViewAlbumDetails();
void PlaySong();

int main() {
    Album album;


    while (true) {
        cout << "\n\033[1;92m-----MENU----\033[0m\n\n\n";

        cout << "1. Add a New Song\n";
        cout << "2. View Details of a Selected Album\n";
        cout << "3. Play a Song\n";
        cout << "4. Exit\n\n";

        cout << "Select an option: ";

        int choice;
        cin >> choice;
        cin.ignore(); // Consume newline character

        switch (choice) {
            case 1:
                AddNewSong(album);
                break;
            case 2:
                ViewAlbumDetails();
                break;
            case 3:
                PlaySong();
                break;
            case 4:
                cout<<"\n\033[1;92mProgram Terminated. Thanks For Using The Music 
Player!\033[0m\n";
                return 0;
            default:
                cout << "Invalid option. Please try again.\n";
                break;
        }
    }

    return 0;
}
```

```cpp
//Album information Input
void AddNewSong(Album& album) {

    // Get user input for album details
    cout << "\nEnter Album Name: ";
    getline(cin, album.album_name);

    cout<<"How Many Songs does this Album Consist of: ";
    cin>>album.number_of_songs;

    // Display genre selection menu
    cout << "\nSelect Genre:\n";
    cout << "\t1. Pop\n\t2. Jazz\n\t3. Classic\n";
    cout<<"Option: ";
    int genre_choice;
    cin >> genre_choice;
    cin.ignore();
    album.genre = static_cast<Genre>(genre_choice - 1);

    // Get user input for song names and locations
    string song_name;
    for (int i = 0; i < album.number_of_songs ; i++) {
        cout << "Enter Track " << i + 1 << " : ";
        getline(cin, song_name);
        album.tracks[i] = song_name;
        album.track_location[i] = "\\Albums\\" + album.album_name + "\\" + song_name;
    }

    // Append album details to Music.txt
    ofstream music_file("Music.txt", ios::app);
    if (music_file.is_open()) {
        music_file << "Album Name: " << album.album_name << "\n";
        music_file << "Genre: ";

        if (album.genre == Pop) music_file << "Pop";
        else if (album.genre == Jazz) music_file << "Jazz";
        else music_file << "Classic";

        music_file << "\n";

        music_file << "Tracks:\n";
        for (int i = 0; i < album.number_of_songs ; i++) {
            music_file << i + 1 << ". " << album.tracks[i] << "\tLocation: \"" <<
album.track_location[i] << ".mp3\"\n";
        }
        music_file << "\n";
        music_file.close();
        cout << "\n\033[1;92mAlbum Added Successfully!\033[0m\n\n";
    }
    else {
        cout << "Unable to open Music.txt for appending.\n";
    }
}

//View Album Inforamation
void ViewAlbumDetails() {

    string album_name;

    cout<<"Please Enter the Name of The Album You Want To Find: ";
    getline(cin,album_name);
    ifstream music_file("Music.txt");
    if (music_file.is_open()) {
        string line;
        while (getline(music_file, line)) {
            if (line.find("Album Name: "+album_name) != string::npos) {
                cout <<"\n\n\033[1;92m"<< line << "\033[0m\n";
                while (getline(music_file, line) && !line.empty()) {
                    cout << "\033[1;92m"<<line <<"\033[0m\n";
                }
                cout << "\n";
            }
        }
        music_file.close();
    }
    else {
        cout << "Unable to open Music.txt for reading.\n";
    }
}

//Play Song from music file
void PlaySong() {
    ifstream music_file("Music.txt");
    if (music_file.is_open()) {

        string album_name;
        cout << "Please Enter the Name of The Album You Want To Find: ";
        getline(cin, album_name);

        cout << "\n";


        string song_name;
        cout << "Please Enter The Name of the Song You Want To Play: ";
        getline(cin,song_name);


        cout << "\n";
        string line;

        while (getline(music_file, line)) {
            if (line.find("Album Name: " + album_name) != string::npos) {
                while (getline(music_file, line) && !line.empty()) {
                    if (line.find(song_name) != string::npos) {
                        // Find the start and end positions of the location within double
quotes
                        int start = line.find("Location: \"");
                        int end = line.find_last_of("\"");

                        if (start != string::npos && end != string::npos) {
                            string track_location = line.substr(start + 12, end - (start
+12));


                            cout<<"\033[1;92mNow Playing: "<<song_name<<"\033[0m\n\n";

                            // You can use track_location with ShellExecute here
                            ShellExecute(NULL, NULL, track_location.c_str(), NULL, NULL,
SW_SHOWNORMAL);
                        }
                        else {
                            cout << "Location not found for the selected track.\n\n";
                        }
                    }
                }
            }
        }
        music_file.close();
    } else {
        cout << "Unable to open Music.txt for reading.\n";
    }
}
```

Outputs:



/c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)/2. Assignments/As...

```
nidul@Nidula MINGW64 /c/Users/nidul/OneDrive/Desktop/Swinburne/1.Fundamentals Of Programming (COS10029)
/2. Assignments/Assignment 2 F
$ ./Test

-----MENU-----

Enter Album Name: Classics
How Many Songs does this Album Consist of: 3

Select Genre:
        1. Pop
        2. Jazz
        3. Classic
Option: 3
Enter Track 1 : Blue Night
Enter Track 2 : More Than a Friend
Enter Track 3 : Sleeping Child

Album Added Successfully!


-----MENU-----


1. Add a New Song
2. View Details of a Selected Album
3. Play a Song
4. Exit

Select an option: 2
Please Enter the Name of The Album You Want To Find: Classics


Album Name: Classics
Genre: Classic
Tracks:
1. Blue Night   Location: "\Albums\Classics\Blue Night.mp3"
2. More Than a Friend   Location: "\Albums\Classics\More Than a Friend.mp3"
3. Sleeping Child       Location: "\Albums\Classics\Sleeping Child.mp3"


-----MENU-----


1. Add a New Song
2. View Details of a Selected Album
3. Play a Song
4. Exit

Select an option: 3
Please Enter the Name of The Album You Want To Find: Classics

Please Enter The Name of the Song You Want To Play: Blue Night

Now Playing: Blue Night
```