ATM SYSTEM

Anas Ahmed: 19220122

Nada Hamdy: 192200252

Jana Sherif: 122300004

Youssef Ahmed Moneir: 192300146

Hend Elhout: 192200264

Introduction

The provided code is a simple command-line program that simulates basic banking operations, including account creation, ATM transactions (withdrawal, deposit, and fund transfer), and account management. The program utilizes C++ and demonstrates the use of functions, structures, vectors, and basic input/output operations.

Overall Structure

1. Header Files:

- The code includes necessary header files like `<iostream>`, `<string>`, and `<vector>`.

- The `using namespace std;` directive is used for convenience.

2. Function Declarations:

- The program declares several functions at the beginning, including `ATM`, `new_account`, `account_no_creation`, `print`, `format_on`, `format_off`, `check_int_input`, and `check_passcode`.

- These functions are defined later in the code.

3. Global Variables:

- Global variables like `accounts_vector`, `choice`, `num`, `num2`, `pass`, `loop2`, `option`, and `cancel` are declared, and some are initialized.

Data Structures

1. Account Structure:

- The `Account` structure defines a blueprint for storing information about a bank account, including `name`, `type` (GOLD or SILVER), `acc_number`, `password`, and `balance`.

2. Vector of Accounts:

- The program uses a vector `accounts_vector` to store instances of the `Account` structure.

Main Function

1. Welcome Message:

- The `main` function begins by welcoming the user and presenting two options: (1) Create a new bank account and (2) Use the ATM.

2. User Choice Handling:

- The user is prompted to enter either '1' or '2'.
- If '1' is chosen, the `new_account` function is called.
- If '2' is chosen, the `ATM` function is called.

3. Account Creation:

- In the `new_account` function, the user is prompted to enter their name, create a password, and provide an initial balance.
- The account type (GOLD or SILVER) is determined based on the initial balance.
- The account is then added to the `accounts_vector`.

ATM Function

1. Account Verification:

- In the `ATM` function, the user is prompted to enter their account number.
- If the account number is not found, an error message is displayed.

2. Password Verification:

- The user is prompted to enter a PIN, and there are three attempts to enter the correct PIN.

3. ATM Transactions:

- Once authenticated, the user can perform cash withdrawal, deposit, fund transfer, or quit.

Utility Functions

1. Printing Functions:

- The `print`, `format_on`, and `format_off` functions are used for formatting and printing messages to the console.

2. Input Validation Functions:

- The `check_passcode` function ensures that the user's password is a four-digit number.
- The `check_int_input` function validates integer input within a specified range.

Code Styling

1. Code Formatting:

- The code utilizes indentation and consistent spacing for readability.
- Function names and comments follow a clear and descriptive style.

2. Use of Colors:

- The code includes color formatting for printed messages, enhancing visual appeal.

Conclusion

The provided code demonstrates a basic banking simulation with functionalities such as account creation, ATM transactions, and input validation. It follows good coding practices, including modularization through functions and the use of structures and vectors for data organization. Further improvements could include additional error handling, more detailed comments, and potential enhancements to user interactions.

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;


void ATM();
void new_account();
int account_no_creation();
```

```
10. void print(string text);
11. void format_on();
12. void format_off();
13. int check_int_input(int min, int max);
14. int check_passcode();
15.
16. struct Account {
17.     string name;
18.     string type;
19.     int acc_number;
20.     int password;
21.     int balance;
22. };
23.
24. vector<Account> accounts_vector;
25. int choice;
26. int num;
27. int num2 = 0;
28. int pass;
29. int loop2 = 1;
30. int option;
31. string cancel;
32.
33. /**
34. * Main function for the program. Prints welcome message and prompts user for choice
35. */
36. void main() {
37.     // This method is used to ask for a new bank account and ATM
38.     while (true) {
39.         print(
40.             "Welcome to our bank\n"
41.             "You have two choices: \n"
42.             "(1) NEW BANK ACCOUNT \n"
43.             "(2) ATM \n"
44.             "Type 1 or 2: ");
45.         choice = check_int_input(1, 2);
46.         cout << endl;
47.
48.         // This method is called by the user to choose the choice.
49.         if (choice == 1) {
50.             new_account();
51.         // This method is called by the user to choose the choice.
52.         } else if (choice == 2) {
53.             ATM();
54.         // This method is called when the choice is 3.
55.         } else if (choice == 3) {
56.             print("Thank you for using our bank\n");
57.             break;
58.         }
59.     }
60. }
61.
62.
63. int account_no_creation() {  // ACCOUNT NUMBER CREATION FUNCTION
64.
65.     return accounts_vector.size() + 1;
66. }
67.
68. /**
69. * NEW ACCOUNT FUNCTION Account creation and account number creation Function is called when
user presses enter in new
70. */
71. void new_account() {  // NEW ACCOUNT FUNCTION
72.     Account account;
73.     print(
```

```
 74.         "Welcome\n"
 75.         "Enter your name: ");
 76.     cin >> account.name;
 77.     cout << endl;
 78.     print("Create your password: ");
 79.     account.password = check_passcode();
 80.     cout << endl;
 81.     print("Enter your balance: ");
 82.     account.balance = check_int_input(0, 100000000);
 83.     cout << endl;
 84.     // Set account type to GOLD or SILVER
 85.     if (account.balance >= 1000000) {
 86.         account.type = "GOLD";
 87.     } else {
 88.         account.type = "SILVER";
 89.     }
 90.
 91.     account.acc_number = account_no_creation();
 92.     accounts_vector.push_back(account);
 93.
 94.     format_on();
 95.     cout << "Your account is created successfully\n";
 96.     cout << "Your account number is " << account.acc_number << endl;
 97.     cout << "Your account type is " << account.type << endl;
 98.     cout << "Your balance is " << account.balance << "$" << endl;
 99.     cout << endl;
100.     format_off();
101. }
102.
103. /**
104. * \ brief ATM This is the function that runs when the program starts
105. */
106. void ATM() {  // ATM
107.
108.     print(
109.         "Welcome\n"
110.         "Enter your account number: ");
111.     num = check_int_input(1, 100);
112.     cout << endl;
113.     bool not_found = false;
114.     for (auto account : accounts_vector) {
115.         // Check if the account is not found
116.         if ((num) != account.acc_number) {
117.             not_found = true;
118.         }
119.     }
120.     // This method is used to check the account number and check the password and check the
account number and check the account number and check the account type and check the account number
and check the account type and check the account number and check the account type and check the
account number and check the account number and check the account type and check the account number
and check the account number and check the account type and check the account number and check the
account number and check the account number and check the account number and check the account
number and check the account number and check the account number and check the account number and
check the account number and check the account
121.     if (not_found == true) {
122.         print(
123.             "Invalid account number\n"
124.             "Please try again Later\n\n\n");
125.     }
126.
127.     else {
128.         print("Please enter your PIN number and press ENTER: ");
129.         pass = check_int_input(0, 10000);
130.         cout << endl;
131.         // This method is used to send the cash withdrawal and deposit.
```

```
        for (int j = 1; j < 3; j++) {
            // This method is used to send the account to the server.
            if (pass == accounts_vector[num - 1].password) {
                // This method is used to send the action to the server.
                while (true) {
                    format_on();
                    cout << "Welcome " << accounts_vector[num - 1].name << endl;
                    cout << "Your account number is "
                         << accounts_vector[num - 1].acc_number << endl;
                    cout << "Your account type is "
                         << accounts_vector[num - 1].type << endl;
                    cout << "Your available balance is "
                         << accounts_vector[num - 1].balance << "$" << endl;
                    cout << endl;
                    // Set the type of account to GOLD or SILVER
                    if (accounts_vector[num - 1].balance >= 1000000) {
                        accounts_vector[num - 1].type = "GOLD";
                    } else {
                        accounts_vector[num - 1].type = "SILVER";
                    }
                    cout << endl;
                    int option;
                    int exchange;
                    int num3;      // to check acc num
                    int num4 = 0;  // other account to be transfered to
                    cout << "Choose the action\n";
                    cout << "(1) Cash Withdrawal\n";
                    cout << "(2) Deposit\n";
                    cout << "(3) Fund Transfer\n";
                    cout << "(4) quit\n\n";
                    cout << "Type 1 or 2 or 3 or 4: ";

                    format_off();
                    option = check_int_input(1, 4);
                    cout << endl;
                    // This method is used to generate the cash and deposit cashs
                    if (option == 1) {  // cash withdrawal
                        print("Your Limit is 2000\n");
                        print("Enter amount of cash to be withdrawn: ");
                        exchange = check_int_input(0, 2000);
                        // This method is used to calculate the amount of cash withdrawn
                        if (exchange > accounts_vector[num - 1].balance) {
                            print("You do not have enough cash\n\n");
                        } else {
                            accounts_vector[num - 1].balance -= exchange;
                            print("Cash withdrawn successfully\n");
                            format_on();
                            cout << "-" << exchange << endl;
                            cout << "Available balance = "
                                 << accounts_vector[num - 1].balance << "$"
                                 << endl;
                            format_off();
                        }
                    // This method is used to determine the amount of cash you want to deposit
put cash transfer cash to the other account
                    } else if (option == 2) {  // deposit (put cash)
                        print("Enter amount of cash you want to deposit: ");
                        format_on();
                        exchange = check_int_input(0, 100000);
                        accounts_vector[num - 1].balance += exchange;
                        cout << "+" << exchange << endl;
                        cout << "Available balance = "
                             << accounts_vector[num - 1].balance << "$" << endl;
                    // This method is called by the user to transfer cash to the other account.
                    } else if (option == 3) {
```

```
                        print(
                            "Enter account number you want to transfer cash "
                            "to: ");
                        num2 = check_int_input(1, 100);
                        for (auto account : accounts_vector) {
                            // check if the account number is valid
                            if ((num2 - 1) != account.acc_number) {
                                print("This account number is invalid\n");
                                not_found = true;
                            }
                        }
                        // This method is called by the user to transfer the cash to the other
account.
                        if (not_found) {
                            print(
                                "Invalid account number\n"
                                "Please try again Later\n");
                        } else {
                            print("Enter amount of cash to be transfered: ");
                            exchange = check_int_input(0, 100000);
                            // This method is called by the user to transfer the cash to
another account.
                            if (exchange > accounts_vector[num - 1].balance) {
                                print(
                                    "You do not have enough cash to transfer\n"
                                    "Please try again later\n");
                            } else {
                                accounts_vector[num - 1].balance -= exchange;
                                accounts_vector[num2 - 1].balance += exchange;
                                format_on();
                                cout << "-" << exchange
                                     << " are transfered successfully from "
                                        "your account\n";
                                cout << "+" << exchange
                                     << " are transfered successfully to the "
                                        "other account\n";
                                cout << "Cash transfered successfully\n";
                                cout << "Available balance = "
                                     << accounts_vector[num - 1].balance << "$"
                                     << endl;
                                format_off();
                            }
                        }
                    // This method is called by the user when the user clicks on the bank.
                    } else if (option == 4) {
                        print("Thank you for using our bank\n");
                        break;
                    } else {
                        print("Enter a valid number.");
                    }
                }
            } else {
                format_on();
                cout << "Incorrect password you have " << 3 - j
                     << " times left: ";
                format_off();
                pass = check_int_input(0, 10000);
            }
            cout << endl;
        }
    }
}

/**
```

```
258. * Prints a string to the screen. This is a debugging function to be used in conjunction with
print_string
259. *
260. * Args:
261. *     text: The string to print
262. */
263. void print(string text) {
264.     cout << "\033[1m\033[34;3m" << text;
265.     cout << "\033[0m";
266. }
267.
268. /**
269. * \ fn format_on \ brief Display format on screen and start formatting
270. */
271. void format_on() { cout << "\033[1m\033[34;3m"; }
272.
273. /**
274. * \ brief Turn off formatting \ ingroup QGIS_QUICK
275. */
276. void format_off() { cout << "\033[0m"; }
277. /**
278. * Check passcode for correctness and convert to integer for efficient comparison This function
is called by main. cpp
279. *
280. *
281. * Returns:
282. *     int The integer representation of the passcode or 0 if
283. */
284. int check_passcode() {
285.     string passcode;
286.     cin >> passcode;
287.     // Ensure passcode is a string with exactly four characters
288.     // Check if passcode is four characters long
289.     if (passcode.length() != 4) {
290.         print(
291.             "Passcode must be exactly four characters long.\n"
292.             "Please Create a Valid Passcode: ");
293.         return check_passcode();
294.     }
295.
296.     // Check if each character is a digit
297.     for (char c : passcode) {
298.         // Check if the passcode is four digit number
299.         if (!isdigit(c)) {
300.             print("Passcode must be a four-digit number.\n");
301.             return check_passcode();  // Add the missing argument
302.         }
303.     }
304.
305.     // Convert the passcode string to an integer for efficient comparison
306.     int passcode_int = stoi(passcode);
307.     return passcode_int;
308. }
309. /**
310. * Check if user input is between min and max. If not the user is prompted to enter a valid
number
311. *
312. * Args:
313. *     min: Minimum value of the input
314. *     max: Maximum value of the input ( inclusive ).
315. *
316. * Returns:
317. *     Integer that was entered or min if input isn't
318. */
319. int check_int_input(int min, int max) {
```

```
320.     int input;
321.     cin >> input;
322.
323.     // Check if the input operation with cin was successful
324.     // Check if the input is valid.
325.     if (cin.fail()) {
326.         cin.clear();  // Clear the error flag
327.         cin.ignore(numeric_limits<streamsize>::max(),
328.                    '\n');  // Discard invalid input
329.         print("Invalid input. Please enter a valid number: ");
330.         return check_int_input(min, max);
331.     // Check if the input is within the range min max.
332.     } else if (input < min || input > max) {
333.         format_on();
334.         cout << "Invalid input. Please try again:";
335.         format_off();
336.         return check_int_input(min, max);
337.     } else {
338.         return input;
339.     }
340. }
341.
```