mMONTAGEe's
Workshop

Name: Bank ATM Program

Start Date: 18/12/2023

End Date: 31/12/2023

Grade: 95/100, A+

Project: #10

Assigned: mMONTAGEe

Supervisor: Gerardagh

Language: C++

Documents: Allowed **AI: Allowed for Assistance**

Tabel of Content:

INTRODUCTION:

The purpose of this project is to simulate bank ATM's operations and functionality, The simulated ATM project aims to replicate the fundamental transactions associated with traditional ATMs, such as cash withdrawals.

DESIGN:

ATM Simulation replicates standart ATM functionality. Architecture is built to follow step by step procedure from User's authorization, cash withdrawal to recipient printing.

Simulation creates a transactinfo file which stores required information which is meant to be sent to bank for approval, information itself inside file is encrypted (in my case its simple linear encryption), after receiving in bank, information is decrypted.

IMPLEMENTATION:

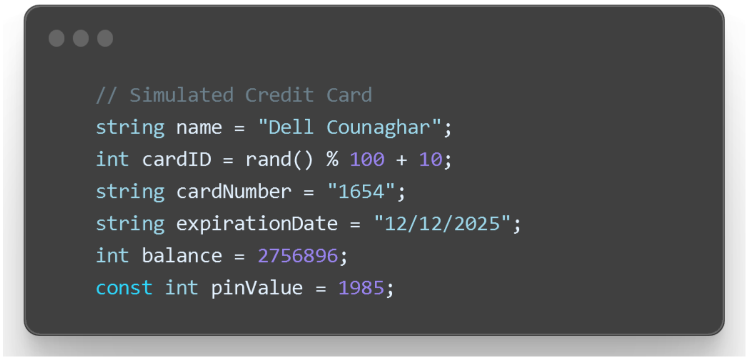
Algorithm: ATM Simulation | *Flowchart Form*

```
1.  welcomeMessage()
2.  chooseService()

if chosenService -> WITHDRAW {

3.  pin input;
4.  pin check;
        a.  If {Rejected} – Program asks user for pin again
        b.  If {Approved} – user moves to next step
5.  show withdrawal options
6.  user inputs which option he wishes to choose
7.  program remembers its choice and selects the amount from Array
8.  Program makes the check to make sure the chosen option is in range from 1 to 6
        a.  If { from 1 to 6} – user moves to next step
        b.  If (<1 || 6>} – error message, user asked to to choose again
9.  Program creates encrypted file, which is simulated to be sent to bank, bank approves/rejected request
10. Program checks if chosen amount does not exceed balance
        a.  If{WithdrawalAmount > balance} – transaction request is rejected
        b.  If{WithdrawalAmount < balance} – transaction request is approved
11. If it does not exceed balance, print approval, minus the chosen amount from balance to get finalbalance
12. Print recipe.
```

}

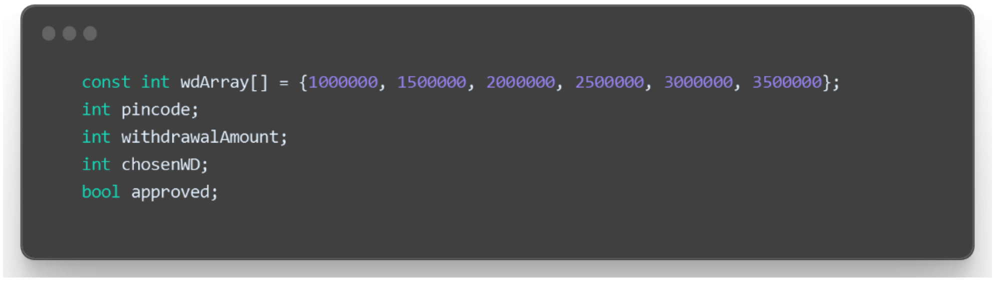
Please note that last options made purely cosmetical.

First Step, was to create a virtual credit card from which information would be taken, Credit card’s variables are declared in very beginning of main().



```
// Simulated Credit Card
string name = "Dell Counaghar";
int cardID = rand() % 100 + 10;
string cardNumber = "1654";
string expirationDate = "12/12/2025";
int balance = 2756896;
const int pinValue = 1985;
```

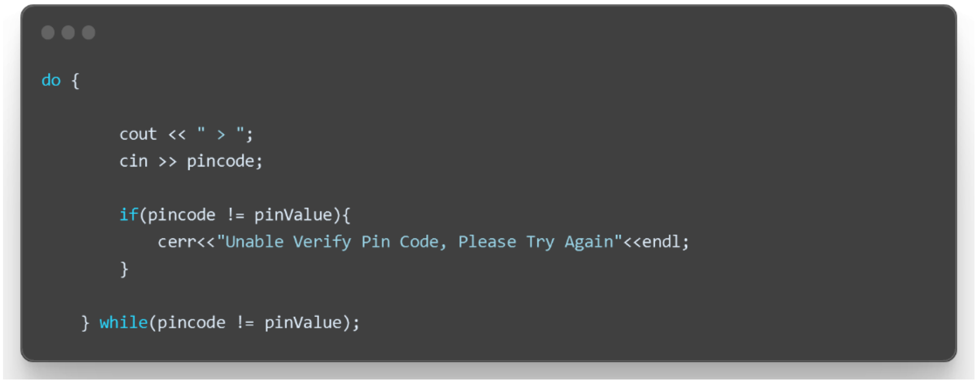
After which created variables identifying ATM, For the code itself I declared variables which are used in processes.



```
const int wdArray[] = {1000000, 1500000, 2000000, 2500000, 3000000, 3500000};
int pincode;
int withdrawalAmount;
int chosenWD;
bool approved;
```

During Authentication process, First is shown Welcome Message. Welcome Message consists of Welcoming ‘name’ (‘name’ is taken from simulated credit card) after which user is asked to choose a service, if chosen service is withdrawal, then user is asked to input their Credit Card’s Pin Code.

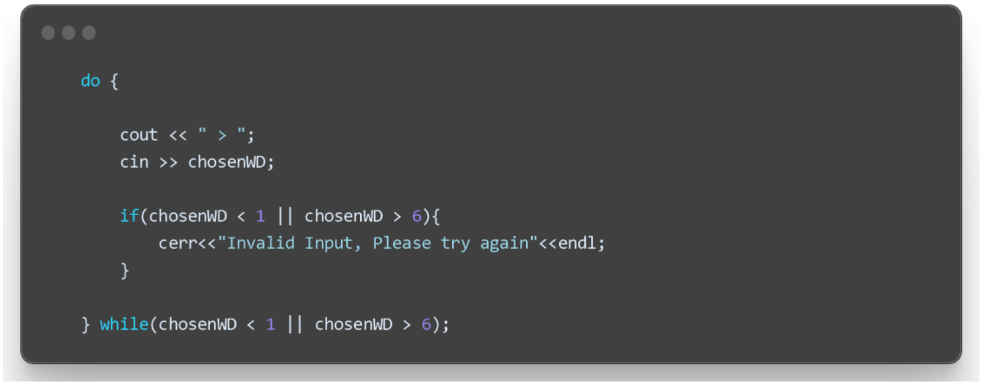
After user input its Pin Code, PinCode is compared with Stored Pin Code value in Simulated Credit Card. Incase of wrong PinCode, Program shows error message and prompts user to input PinCode again.



```
do {

        cout << " > ";
        cin >> pincode;

        if(pincode != pinValue){
            cerr<<"Unable Verify Pin Code, Please Try Again"<<endl;
        }

    } while(pincode != pinValue);
```

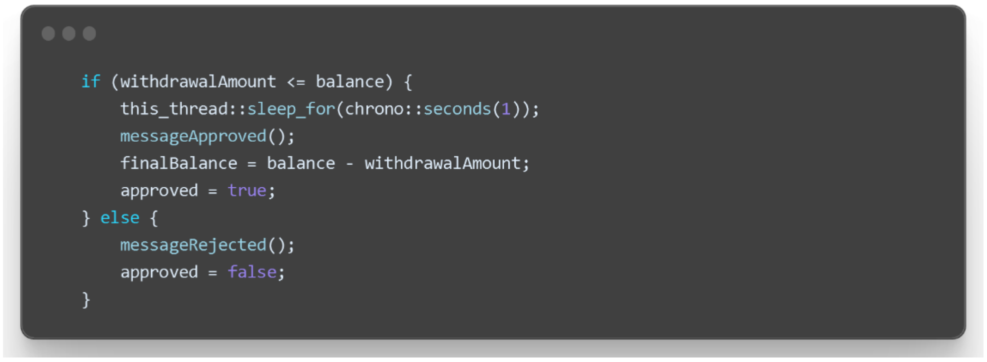
After check has passed succsesfully, user proceeds to withdrawal process. User gets 6 options, the 6 options are set by Bank and the values are stored in Array mentioned before.

During withdrawal process user is asked to choose amount of withdrawal, Chosen withdrawal itself passes through several checks, first check is is to make sure chosen option is in range of from 1 to 6.



```
do {

    cout << " > ";
    cin >> chosenWD;

    if(chosenWD < 1 || chosenWD > 6){
        cerr<<"Invalid Input, Please try again"<<endl;
    }

} while(chosenWD < 1 || chosenWD > 6);
```

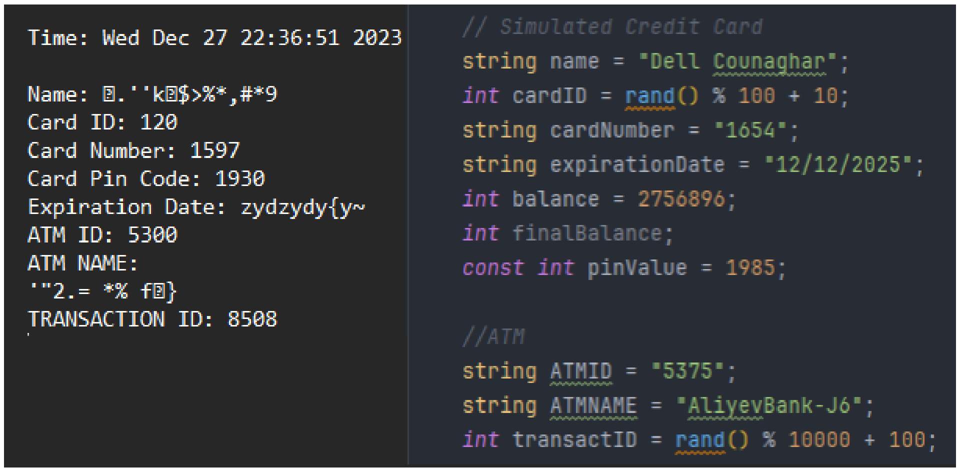
Second check makes sure that chosen withdrawal does not exceed user's balance on the credit card.



```
if (withdrawalAmount <= balance) {
    this_thread::sleep_for(chrono::seconds(1));
    messageApproved();
    finalBalance = balance - withdrawalAmount;
    approved = true;
} else {
    messageRejected();
    approved = false;
}
```

A lot of Cout's are inside different Functions not to clutter main() code block, which also improved code readability.

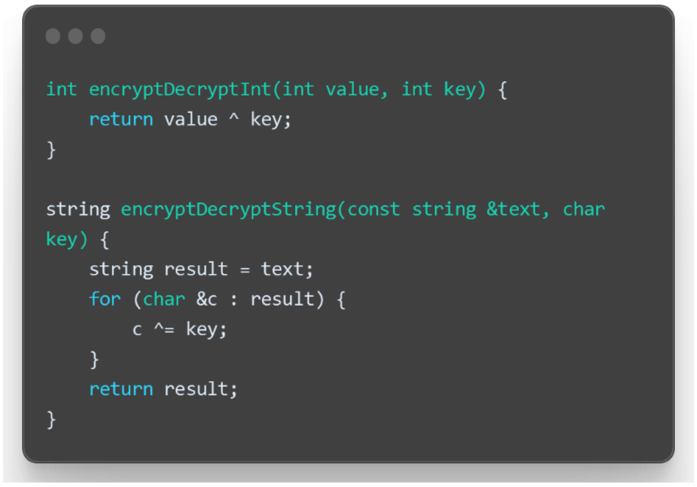
Right before balance check, an transaction file is created which stores required information which is sent to Bank for approval, Information inside text file is encrypted and after receiving by bank its decrypted.



```
Time: Wed Dec 27 22:36:51 2023

Name: ▯.''k▯$>%*,#*9
Card ID: 120
Card Number: 1597
Card Pin Code: 1930
Expiration Date: zydzydy{y~
ATM ID: 5300
ATM NAME:
'"2.= *% f▯}
TRANSACTION ID: 8508
```

```
// Simulated Credit Card
string name = "Dell Counaghar";
int cardID = rand() % 100 + 10;
string cardNumber = "1654";
string expirationDate = "12/12/2025";
int balance = 2756896;
int finalBalance;
const int pinValue = 1985;

//ATM
string ATMID = "5375";
string ATMNAME = "AliyevBank-J6";
int transactID = rand() % 10000 + 100;
```

Encryption and decryption are done by a Functions, since there are integers and strings, encryption and decryption are done separately.



```cpp
int encryptDecryptInt(int value, int key) {
    return value ^ key;
}

string encryptDecryptString(const string &text, char
key) {
    string result = text;
    for (char &c : result) {
        c ^= key;
    }
    return result;
}
```

Please note that provided encryption and decryption is very simple and is done to demonstrate as an example.

Recepient printing itself is done by a Function not to clutter main() code block itself

The deposit option in this simulator is straightforward. When the user selects option 2, they are prompted to enter their PIN. Following the PIN entry, the user inputs the deposit amount. Although the simulator doesn't simulate the insertion of banknotes, the written amount (in this case, since banknote insertion is not simulated) is added to the existing balance, effectively updating the account balance.

Thirs option simply shows person phone number to contact the bank.

TESTING:

Well, the testing process was to make sure that ATM followes algorithm as intended, Tested to make sure that if PIN does not match then user will be asked to input again, tested if withdrawal system works, withdrawal chosen amount checks and balance check. And file content encryption itself ofcourse, In some cases I couldn't understand why wouldn't a value be assigned to an variable, needing to use JetBrains CLion's breakpoints to see the steps process to understand where was the error,

LESSONS LEARNED

Throughout this project, I honed my skills in writing C++ syntax, enhancing the readability and modularity of the code. Additionally, the experience contributed to my proficiency in formatting and crafting well-structured report documents.

FUTURE ENCHANTMENTS:

The code exhibits a high degree of modularity and readability, ensuring that future additions will seamlessly integrate without causing implementation issues. Personally, I would have included additional services such as funds transfer and balance checks—standard functionalities of an ATM. However, as per the guidelines, these features were not obligatory for inclusion (though not difficult to implement if required).

KNOWN ISSUES:

Balance between Withdraw and Deposit services are not synchnorised

Full Source code can be found on the Github Page.