Individual Report

------COMP 1011 CHEN Ziyang 21095751d

Content

- **Problem statement**
- **Objectives of the program**
- **Program design & The structure of the program**
- **User manual**

Problem statement

Problem 1. Store the data of each block, and the index is continuous.

Solution: Use nested structs with a struct containing an array of structs.

Problem 2. Add Blocks to Blockchain and calculate the hash value.

Solution: Check if the size blockchain is full, if not, add it to the corresponding Block structure. Regarding the hash value, I added the given file "SHA1.hpp" and "SHA1.cpp" to modify the char string that can affect the hash value, and directly use "hash" to get the answer.

Problem 3. When querying blocks, when the user enters the wrong data, the loop cannot be entered, and the query menu cannot be displayed again.

Solution: Add a Boolean value, and return true when wrong data is entered. Make it possible to continue running in “while” instruction.

Problem 4. To better understand the “Check the integrity of the block” function for the user.

Solution: To visualize this function, after checking if Blocks are complete, the program will print out all hash values and previous hash values.

Objectives of the program

This program is about Blockchain which is a cryptographic tool. The goal is to be used in Fintech to protect the chronological order of transactions. The user can use this program to store the information that he wants to save, the information can neither be deleted nor modified. And provide the function of displaying chain data to facilitate user queries. If there are malicious changes to transaction data, the program detects and reflects them.

Program design & The structure of the program

First of all, here is the library of the program. It includes 3 libraries and given files hpp. and a cpp.

```
#include <iostream>
#include <cstring>
#include "SHA1.hpp"
#include "SHA1.cpp"
#define MAX 1000
using namespace std;
```

The program designs two structures, namely Block and Blockchain. Among them, the Block structure contains the data, number, hash value, and previous hash value of the Block. In order to facilitate the calculation, when we designed the Blockchain structure, we maintained an array of Blocks with a capacity of 1000 and recorded the number of Blocks in the current Blockchain.

```
struct Block { // here is the structure of block
    string Data;  // The data that the user wants to enter(store)
    int Number; // the index of the blocks
    string p_hash; // previous hash value
    string hash; // hash value
};

struct Blockchain{ // here is the structure of blockchain which is including blocks
    struct Block BlockArray[MAX]; // By default, 1000 blocks can be added
    int m_Size;
};
```

The project is built on the blockchain. This program allows the user to perform four operations: 1. Add a new block 2. Display the chain data 3. Check the integrity of the data 0. Exit the program

If the user writes numbers other than these four, such as 4, 5, 6. The system redisplays the menu and asks the user to retype. When the user enters 0 to exit the program.

```
void showMenu(){ // display the menu
    cout << "**************************************************************************************************" << endl;
    cout << "************************************ Blockchain ************************************" << endl;
    cout << "********* This is the blockchain menu, you can run related functions by number **********" << endl;
    cout << "*                                1. Add a new block                                    *" << endl;
    cout << "*                                2. Search block by given information                  *" << endl;
    cout << "*                                3. Check the integrity of the data                    *" << endl;
    cout << "*                                0. Exit the program                                   *" << endl;
    cout << "**************************************************************************************************" << endl;
    cout << endl;
}
```

In the code, the program creates the Blockchain structure variable "blo" and initializes the current number of blocks in the Blockchain. Calling the menu guarantees that the menu is available to the user every time. select is the user's input to perform the corresponding operation.

```
int main(){
    Blockchain blo; // naming structure
    blo.m_Size = 0; // Initialize the size of blocks
    int select = 0;
    while (true){
        showMenu();
        cout << "Enter the number of the Menu number: "; cin >> select;
        switch (select) {
            case 1:
                addBlocks(&blo); // This is the function to add blocks
                break;
            case 2:
                findBlock(&blo); // display the chain data in a number of ways
                break;
            case 3:
                Check_integrity(&blo); // check the integrity of the blocks
                break;
            case 0: // enter 0 to end the program
                cout << "Thank you for using." << endl;
                return 0;
                break;

            default:
                cout << "Please enter correct a number!" << endl;
        }
    }

    return 0;
}
```

Regarding the first operation, the user can add blocks by himself, store the information (data) that the user wants to store into the blocks, and generate a hash value to ensure security.

First, the program checks whether the blockchain's memory is full and if not, the following operations are performed.

The system will ask the number of blocks the user wants to add. And then the user could enter the information(data) into it. If x blocks are added successfully, the system will display "You have added x block(s) successfully!!! ". After the user enters the information, the system will automatically calculate its hash value. And store the information of the block in the corresponding structure.

After the first function completes, the system displays the menu again to the user, allowing the user to choose the next step.

```
void addBlocks(Blockchain * blo){ // This is the function to add blocks
    if (blo -> m_Size == MAX) { // Blocks cannot be added if they reach 1000
        cout << "Adding failure!" << endl;
        return;
    }
    else{
        cout << "Enter the number of blocks you want to add: ";
        int addNum = 0; // Initialize the number of blocks that user want to add
        cin >> addNum;
        if (addNum > 0){
            // Calculate the new space size, the adding number should not be negative
            int newSize; // 0
            newSize = blo->m_Size + addNum;
            string data; // Initialize the data of block
            string p_hash = ""; // Initialize the previous hash of block

            if (blo->m_Size != 0){
                p_hash = blo->BlockArray[blo->m_Size - 1].hash; // in case previous hash become the initialize value
            }

            for (int i = blo->m_Size; i < newSize; i++) { // i = 0, i < 2
                cout << endl;
                cout<<"***** Please enter the information of block No."<<i+1<<" *****"<<endl;
                cout<<"\tPlease enter the data: "; cin >> data;
                cout << endl;

                string input(data);
                SHA1 checksum;
                checksum.update(input);
                char hash[41];
                strcpy(hash, checksum.final().c_str());

                blo -> BlockArray[i].Data = data;
                blo -> BlockArray[i].Number = i+1;
                blo -> BlockArray[i].p_hash = p_hash;
                blo -> BlockArray[i].hash = hash;

                p_hash = hash;
                blo -> m_Size++; // the size of block could plus 1 after adding 1 block.
            }
            cout << "You have added " << addNum << " block(s) successfully!!!" << endl;

        }
        else{
        cout << "Incorrect input!" << endl;
        }
    }

}
```

First, the second operation checks if there are blocks in it. If the user directly selects this action without entering a new block, the system will display Empty, the system will redisplay the menu and let the user re-enter.

If there is the block(s), the second operation can display the data of the chain in three ways: 1. the user can display the data of the whole chain by entering the block number. 2. the data of the whole chain. 3. the hash value of a specific block.

If the user enters another number, the program will display: "Input wrong or empty information!". The And then program redisplays the three query methods.

```
void findBlock(Blockchain * blo){   // display the chain data in a number of ways
                                    // by block number
                                    // the data of the whole chain
                                    // the hash values of a particular block

    if (blo->m_Size == 0){ // if the size of block is zero, output Empty!
        cout << "Empty!!" << endl;
    }
    else{
        bool loop = true;
        while (loop) {
            cout << "1.Search by block number" << endl;
            cout << "2.Search by the data of the whole chain" << endl;
            cout << "3.Search by the hash values of a particular block" << endl;
            cout << endl;
            cout << "Please enter the search method: ";
            int select;
            int S_number; // initialize the block number that user will input
            string data;
            string hash;
            cin >> select;
```

Take case 1 as an example. When the "select" input by the user is 1, execute case 1. Check if the block is not empty, then compare the "Number" stored in the structure with the number that the user wants to find for traversal and comparison. If found, output the information of the entire chain. The program also designs a Boolean variable check, which is used if the data entered by the user matches, if not found, the program outputs "No corresponding block was found!". And asked to re-enter the numbers. If the data entered when querying the method is incorrect, the function will exit, and the menu will be redisplayed.

```
if (blo->m_Size == 0){
    cout << "Empty!!" << endl;
}
else{
    // Display the informations of the block that the user wants to see according to the block number e
    bool check = false;

    for (int i = 0; i < blo->m_Size; i++){
        if (blo->BlockArray[i].Number == S_number) {
            cout << "*************************************************************************" << endl;
            cout << "***************** Here is the information of block " << i+1 << " ****************
            cout << "*                                                                       *" << endl;
            cout << "\tNumber: " << blo->BlockArray[i].Number << endl;
            cout << "\tData: " << blo->BlockArray[i].Data << endl;
            cout << "\tHash value: " << blo->BlockArray[i].hash << endl;
            cout << "\tPrevious hash value: " << blo->BlockArray[i].p_hash << endl;
            cout << "*                                                                       *" << endl;
            cout << "*************************************************************************" << endl;
            check = true;
        }
    }
    if (!check) {
        cout << "No corresponding block was found!" << endl;
```

The third operation checks whether there are blocks in it. If not, the program will display "Empty" and the system will redisplay the menu and ask the user to retype.

This function allows the user to check the integrity of the data stored in the chain. The integrity of the data in the chain is judged by comparing the hash value with the previous hash value.

When the blockchain already has block information entered by the user. When the user enters Menu number 3, the program will compare the hash value of each block with the previous hash value. Traverse each block, if the previous hash value of the next one is equal to the previous hash value, output "The blocks are integral".

```
void Check_integrity(Blockchain * blo){ // Check the integrity of the data

    if (blo->m_Size == 0){ // if the size of block is zero, output Empty!
        cout << "Empty!!" << endl;
    }
    else {
        bool check = true; // Initialize a Boolean value
        for (int i = 0; i < blo->m_Size-1; i++) { // Check all hash values, compare with their previous hash values
            if (blo->BlockArray[i+1].p_hash != blo->BlockArray[i].hash) {
                check = false;
            }
        }
        cout << endl;
        if (check) { // output the integrity according to the Boolean value
            cout << "\tThe blocks are integral" << endl;
        }
        else{
            cout << "\tThe blocks are NOT integral!!!" << endl;
        }
        cout << endl;
        cout << "Below is all hash value of the blockchain: " << endl; // display all the hash values and previous hash
            blockchain
        for (int i = 0; i < blo->m_Size; i++){
            cout << "\tHash value: " << blo->BlockArray[i].hash << endl;
            cout << "\tPrevious hash value: " << blo->BlockArray[i].p_hash << endl;
            cout << endl;
        }
    }
}
```

User manual

When running the program, the user sees a menu. The menu provides four options for the user to choose from:

```
*************************************************************************************************
************************************** Blockchain ***********************************************
********** This is the blockchain menu, you can run related functions by number ***********
*                                  1. Add a new block                                           *
*                                  2. Search block by given information                         *
*                                  3. Check the integrity of the data                           *
*                                  0. Exit the program                                          *
*************************************************************************************************

Enter the number of the Menu number:
```

If the user inputs numbers other than these four, such as 4, 5, 6. The system redisplays the menu and asks the user to retype.

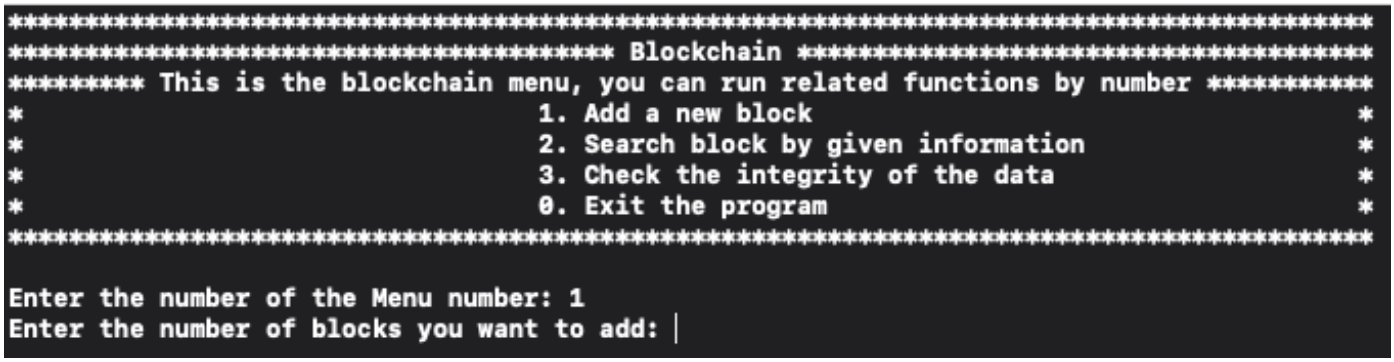
```
**********************************************************************************************
**************************************** Blockchain ****************************************
********** This is the blockchain menu, you can run related functions by number ************
*                                  1. Add a new block                                        *
*                                  2. Search block by given information                      *
*                                  3. Check the integrity of the data                        *
*                                  0. Exit the program                                       *
**********************************************************************************************

Enter the number of the Menu number: 4
Please enter correct a number!
**********************************************************************************************
**************************************** Blockchain ****************************************
********** This is the blockchain menu, you can run related functions by number ************
*                                  1. Add a new block                                        *
*                                  2. Search block by given information                      *
*                                  3. Check the integrity of the data                        *
*                                  0. Exit the program                                       *
**********************************************************************************************

Enter the number of the Menu number:
```

If the user directly selects the second and third actions without entering a new block, the system will display Empty. And it will redisplay the menu, and let the user retype.

```
**********************************************************************************************
**************************************** Blockchain ****************************************
********** This is the blockchain menu, you can run related functions by number ************
*                                  1. Add a new block                                        *
*                                  2. Search block by given information                      *
*                                  3. Check the integrity of the data                        *
*                                  0. Exit the program                                       *
**********************************************************************************************

Enter the number of the Menu number: 2
Empty!!
**********************************************************************************************
**************************************** Blockchain ****************************************
********** This is the blockchain menu, you can run related functions by number ************
*                                  1. Add a new block                                        *
*                                  2. Search block by given information                      *
*                                  3. Check the integrity of the data                        *
*                                  0. Exit the program                                       *
**********************************************************************************************

Enter the number of the Menu number: 3
Empty!!
**********************************************************************************************
**************************************** Blockchain ****************************************
********** This is the blockchain menu, you can run related functions by number ************
*                                  1. Add a new block                                        *
*                                  2. Search block by given information                      *
*                                  3. Check the integrity of the data                        *
*                                  0. Exit the program                                       *
**********************************************************************************************

Enter the number of the Menu number:
```

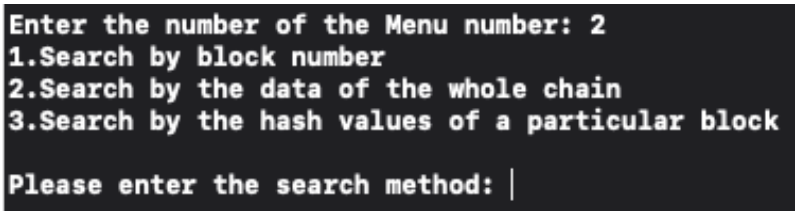
When the user input “1”, the program will run the first operation: add a new block. Entering this operation, the users can select the number of blocks they want to add. After that, they can store the information they want to input into the blocks they want to add.

The system will ask the number of blocks the user wants to add. And then the user could enter the information(data) into it. It could be number or string. And program will store them after they entered.



If x blocks are added successfully, the system will display "You have added x block(s) successfully!!!". After the first action is complete, the system displays the menu to the user again. The user can choose to proceed to the next step.

The second operation can display the data of the chain in three ways:



If the user enters other numbers, the program will display: "Input wrong or empty information!". And then program redisplays those three query methods.

When the correct number is entered to start the query, the system will output the entire block of information provided by the user. Below is an example showing the entire block:

If the user enters another number, the program will display: No corresponding block was found! and ask to re-enter the number. If the data entered when querying the method is incorrect, the function will exit and the menu will be redisplayed.

The third operation allows users to check the integrity of the data stored in the chain. After ensuring that the blockchain already has the block information entered by the user. When the user enters Menu number "3".

Output "The blocks are integral" if the block is not destroyed.

Instead, it outputs "The blocks are NOT integral!!!".

In order to facilitate the user to understand more intuitively, the program will output the hash value at the same time.

```
Enter the number of the Menu number: 3

    The blocks are integral

Below is all hash value of the blockchain:
    Hash value: 40bd001563085fc35165329ea1ff5c5ecbdbbeef
    Previous hash value:

    Hash value: 5f6955d227a320c7f1f6c7da2a6d96a851a8118f
    Previous hash value: 40bd001563085fc35165329ea1ff5c5ecbdbbeef

    Hash value: 92cfceb39d57d914ed8b14d0e37643de0797ae56
    Previous hash value: 5f6955d227a320c7f1f6c7da2a6d96a851a8118f
```

When the user enters 0 to exit the program:

```
**************************************************************************************************
****************************************** Blockchain ********************************************
********** This is the blockchain menu, you can run related functions by number ***********
*                                  1. Add a new block                                            *
*                                  2. Search block by given information                          *
*                                  3. Check the integrity of the data                            *
*                                  0. Exit the program                                           *
**************************************************************************************************

Enter the number of the Menu number: 0
Thank you for using.
```