DOCUMENTATION DS #2

Muhammad Ismail Ramzan

20I-0941 Cysec T

Contents

Documentation

This is the documentation of the assignment No 2 of Data structure assignment. Code is fully commented. It is not necessary to read this documentation. Code can be easily understood 😊.

Basic Features

- ➔ CBID is doubly linked list
- ➔ New data is stored in ascending order on cnic
- ➔ Data can be easily updated using cnic
- ➔ CCID is circular doubly linked list
- ➔ CCID each Node has singly linked list of crime
- ➔ CCID also store new data on ascending order depending on cnic
- ➔ Data or crime can be updated,added or deleted on CCID using cnic
- ➔ CCID has pointer to CBID node while CBID node has pointer to CCID node which has same Cnic.
- ➔ Data is easily accessable between CBID and CCID

Working of the code

This documentation will explain the working of the code.

When the database object is made then the first thing that will run is the constructor of the database.

Constructor of the Database:

This piece of code opens the file CBID.txt and read it line by line. For each line we need to extract information. So, I used the string functions & operations that simply fetch out each word from the line. Then using some programming magic I assigned them to the information object (which is Data

object). Now, we open the CCID.txt and read the information and assign them to the information object (if cnic has crime then it is only sent in the queue otherwise nothing is sent to the queue belonging to that particular cnic).

```
if (line2.substr(0, line2.find(" ")) == info.cnic)
```

Now, It sends the information to the Queue and we keep track of count that how many cnic we have sent into the queue.

After this, Data is extracted from the queue using the dequeue method and a temporary node is made that takes the data from the information object.

```
                // Now let's move the data to the CBID
                CBID_NODE<string> obj_node;
                // take data from the queue
                info = dataPipeline.dequeue();
                // Put the data into the object
                // cout << "\n testing \n";
                obj_node.cnic = info.cnic;
                obj_node.father_name = info.father_name;
                obj_node.name = info.name;
                obj_node.gender = info.gender;
                obj_node.address = info.address;
                obj_node.nationality = info.nationality;
                // let's scan the ccid list!
```

and call the functions which are

```
 // This is the function that is called for arranging nodes in the ascending
order uon entry
    T *insert_ascending_according_to_cnic(T &node_Data)
    {
        // some helper varaibles
```

The basic job of the above function is to insert the data into list by finding the correct position of the node.

Working of insert_ascending_according_to_cnic:

- ➔ Check if node is empty if not then insert and return node pointer
- ➔ Check if node has one element if yes insert and return node pointer
- ➔ Now iterate through the list and find the correct position according to cnic and then insert the node at that index and return the node pointer
- ➔ If these conditions are false then insert the node at the end.

While doing this the above function uses another function which is

```
                // check if previous node cnic matches
                if (stoi(new_node->cnic) < stoi(temp->cnic))
                {
                    flag = true;
                    // if we found the correct position then call the function to
do it's job
                    insert_node_at_index(new_node, index);
                    return new_node;
                    break;
                } // then this is the position to insert
```

Working of insert_node_at_index

- ➔ It traverse to that particular index in the list
- ➔ Now set the pointers and put the node into it’s correct position

That is how Nodes are inserted into the ascending order into the CCID AND CBID LIST. CCID has a linked list of crimes as well and this function of ccid also takes care of it and append the crimes accordingly too ☺.

Working of CBID_Search_by_CNIC

```
    // impleting the functions
    string CBID_Search_by_CNIC(int cnic)
    {
        // convert cnic to string
        string cnic_find = to_string(cnic);
```

```
        // now call the function that will return the string after finding the
cnic!

        // I have pointer in cbid that points to same element in the ccid that
has same cnic. So, i don't need to use ccid.

        // Alternative this code can be directly written here that is in the
function and can also by written in CCID list

        string n = CBID.return_string_by_cnic(cnic_find);
        return n;
    }
```

The above code calls another function which is return return_string_by_cnic let's see it's working

Working of return_string_by_cnic:

First of all this function check's if the cnic is present at the tail or not. If yes then it execute the following if conditio

```
            // First of all check the tail if it is equal to the cnic
            if ((tail->cnic == cnic_find))
            {
                // if it is then Simply Get the Information of the User
                string final = "";
                cout << "\n Found you \n";
                // Get the name,father_name,crimes and everything & store them
into a string variable named as final string the return that final string
                final = temp->ptr_to_cbid_list->name + " " + temp-
>ptr_to_cbid_list->father_name + " " + temp->ptr_to_cbid_list->gender + " " +
                        temp->ptr_to_cbid_list->address +
                        temp->ptr_to_cbid_list->nationality + " " + temp-
>crime_ccid.description + " " + temp->crime_ccid.punishment + temp-
>crime_ccid.fine;
                return final;
            }
```

The above code foams a final string and return to the function. It takes out this information from the linked list file.

If above conditions fail then it iterate through the loop and find cnic.

```
                // Ierate through the end of the list
                while ((temp->next != head && temp->next != nullptr))
                {
                    // if you find the cnic then Go to this condition
                    if ((cnic_find) == (temp->cnic))
                    {
                        // Get the name,father_name,crimes and everything & store
them into a string variable named as final string the return that final string
                        string final = "";
                        cout << "\n Found you \n";
                        final = temp->ptr_to_cbid_list->name + " " + temp-
>ptr_to_cbid_list->father_name + " " + temp->ptr_to_cbid_list->gender + " " +
                                temp->ptr_to_cbid_list->address +
                                temp->ptr_to_cbid_list->nationality + " " + temp-
>crime_ccid.description + " " + temp->crime_ccid.punishment + temp-
>crime_ccid.fine;
                        // return a single string in the end!
                        return final;
                    }
                    // keep increasing the temp until it reaches the head because
it is a circular linked list
                    temp = temp->next;
                }
```

If somehow it is unable to find the cnic then it will simply return “NOT FOUND”

Working of CCID_Search_by_CNIC

It works same as above code but the only difference is it iterate through the ccid list instead of cbid.

Working of updateCBIDName

It checks the tail if it is unable to find the information then it iterate through the whole list and find the specific cnic. Once, it find the cnic then it easily changes it name to whatever function is called with. 😊

```
if (head != NULL)
        {
            // check the tail for the cnic
            if ((tail->cnic == cnic_find))
            {
                // change the name
                temp = tail;
                temp->name = fnae;
                return true;
            }
            else
            {
                // traverse the whole list for the element
                while (temp->next != head && temp->next != nullptr)
                {
                    // if cnic is found then
                    if ((cnic_find) == (temp->cnic))
                    {
                        // change the name and return true
                        temp->name = fnae;
                        return true;
                    }
                    // go to the next node
                    temp = temp->next;
                }
            }
        }
        // if not then return false
        return false;
```

The functions

```
updateCBIDNationality
updateCBIDAddress
updateCBIDFName
```

works the same way as above. The only difference will be the parameters.

Working of addCrime

This function iterate through the loop to find the specific cnic. Then it creates a new node and add the information of the crime to the node. After that It cakks the function

`insert_ascending_according_to_cnic`

which does it's job by creating a node of the crime and append it to the linked list.

```
  CCID_NODE<string> new_node;
                // add data to the node
                new_node.crime_ccid.description = des;
                new_node.crime_ccid.fine = fin;
                new_node.crime_ccid.punishment = pun;
                new_node.cnic = cnic_find;
                // now i need to set the correct pointers
                new_node.ptr_to_cbid_list = temp;
                // inserrt the crime in the crime linked list and get the pointer
to that node
                CCID_NODE<string> *ptr =
obje.insert_ascending_according_to_cnic(new_node);
                // once i get the pointer now i can set the ccid list, Simple :)
                temp->ptr_to_ccid_list = ptr;
```

Now, take a look at this code so it appends the crime in the crime linked list too.

```
T *insert_ascending_according_to_cnic(T &node_Data)
    {
        // some helper varaibles
        bool flag = false;
        int index = 0;
        // creating the Node
        T *new_node = new T;
        new_node->cnic = node_Data.cnic;
        // Now using the linked list inside the ccid make a node of the crime
        new_node->append(node_Data.crime_ccid);
        // This is also senting the data in the crime linked lis
```

It is calling an append function. Let's see that.

This function mantains the linked list of the crime and appends the crimes node according to the cnic.

```
        // 1. allocate node
        crime<string> *new_node = new crime<string>;
        // put the data inside the node
        new_node->description = information.description;
        new_node->punishment = information.punishment;
        new_node->fine = information.fine;
        // // this is going to be last node
        new_node->next = NULL;
        if (head == NULL)
        {
            head = new_node;
            return;
        }
        crime<string> *last = head;
        // 5. Else traverse till the last node
        while (last->next != NULL)
        {
            last = last->next;
        }
        last->next = new_node;
        return;
```

working of delete crime

It checks for the cnic. If cnic matches then it checks if that cnic has some crime or not. If it finds out it has some crimes then it correct the pointers and then delete that node from the linked list of the crime.

```
 // check the tail for the crime record
          if ((tail->cnic == cnic_find))
          {
              // if criminal record is found
              temp = tail;
              if (temp->ptr_to_ccid_list != NULL)
              {
                  // then make a temp pointer that points to that record
                  CCID_NODE<string> *ptr = temp->ptr_to_ccid_list;
                  // make the pointers correct
```

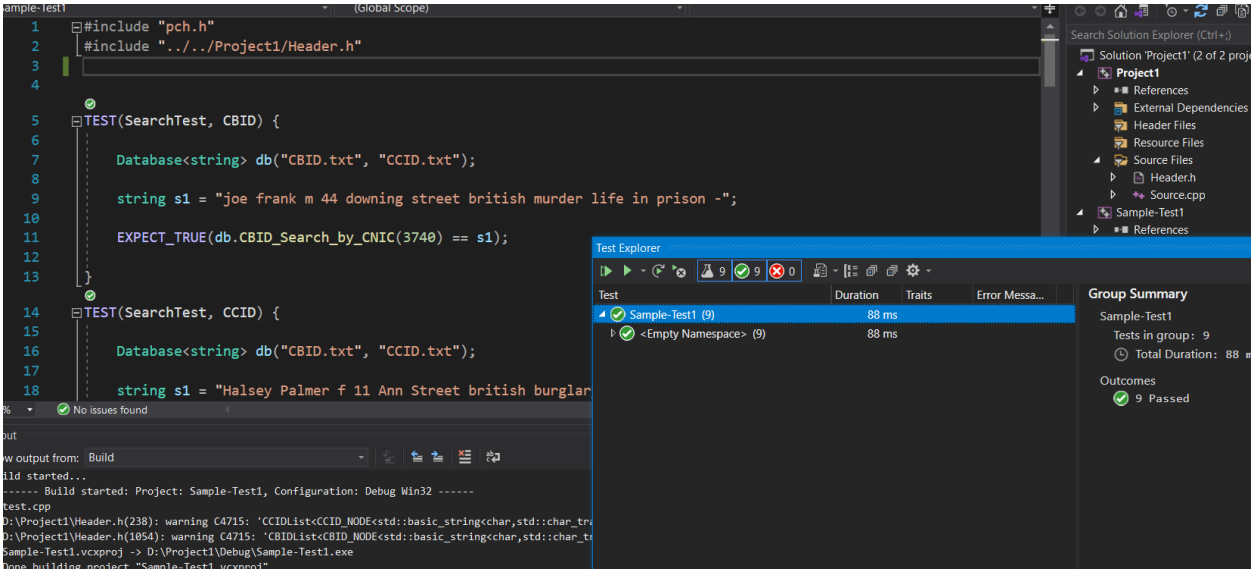
```
                ptr->previous->next = ptr->next;
                ptr->next->previous = ptr->previous;
                // then delete that pointer
                delete ptr;
                temp->ptr_to_ccid_list = NULL;
                // if deleted successfuly return true
                return true;
            }
            // if no cnic found then return false
            return false;
        }
```

Working of updateCrime

It iterate through the list and finds that particular cnic. If that criminal has any crimes then it updates it by the argument passed.

```
// check the tail for the cnci
        if ((tail->cnic == cnic_find))
        {
            // if found then check if the criminal has some crime
            temp = tail;
            if (temp->ptr_to_ccid_list != NULL)
            {
                // if criminal has crimes then change them
                temp->ptr_to_ccid_list->crime_ccid.fine = fin;
                temp->ptr_to_ccid_list->crime_ccid.punishment = pun;
                temp->ptr_to_ccid_list->crime_ccid.description = des;
                // after changing return true
                return true;
            }
            // if there is no data of the criminal then return falses
            return false;
        }
```

Result of test cases




```
TEST(UpdateTest, CBID_Nationality) {

    Database<string> db("CBID.txt", "CCID.txt");

    EXPECT_TRUE(db.updateCBIDNationality("Alien", 8000) == 0);
    EXPECT_TRUE(db.updateCBIDNationality("Alien", 9832) == 0);


}

TEST(CrimeAddTest, CCID) {

    Database<string> db("CBID.txt", "CCID.txt");

    EXPECT_TRUE(db.addCrime(7564, "Robbery", "2 years in prison", "1000") == 1);

    EXPECT_TRUE(db.addCrime(8099, "Kidnapping", "20 years in prison", "2000") == 0);
}

TEST(CrimeDeleteTest, CCID) {

    Database<string> db("CBID.txt", "CCID.txt");

    EXPECT_TRUE(db.deleteCrime(5960, "burglary", "6 years in prison", "7000") == 1);

    EXPECT_TRUE(db.deleteCrime(4064, "murder", "2 years in prison", "10000") == 0);
}

TEST(CrimeUpdateTest, CCID) {

    Database<string> db("CBID.txt", "CCID.txt");

    EXPECT_TRUE(db.updateCrime(5960, "murder", "20 years in prison", "10000") == 1);

    EXPECT_TRUE(db.updateCrime(9630, "murder", "20 years in prison", "10000") == 0);
}
```

```
    Database<string> db("CBID.txt", "CCID.txt");

    string s1 = "Halsey Palmer f 11 Ann Street british burglary 6 years in prison 7000";

    EXPECT_TRUE(db.CCID_Search_by_CNIC(5960) == s1);

}

TEST(UpdateTest, CBID_Name) {

    Database<string> db("CBID.txt", "CCID.txt");

    EXPECT_TRUE(db.updateCBIDName("Alice", 9831) == 1);
    EXPECT_TRUE(db.updateCBIDName("Doc", 9832) == 0);

}

TEST(UpdateTest, CBID_FName) {

    Database<string> db("CBID.txt", "CCID.txt");

    EXPECT_TRUE(db.updateCBIDFName("Kevin", 9177) == 1);
    EXPECT_TRUE(db.updateCBIDFName("Steve", 1234) == 0);

}

TEST(UpdateTest, CBID_Address) {

    Database<string> db("CBID.txt", "CCID.txt");

    EXPECT_TRUE(db.updateCBIDAddress("12 Ann Street", 8372) == 1);
    EXPECT_TRUE(db.updateCBIDAddress("21 Downing Street", 9639) == 0);

}
```