TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

----o0o----

BÁO CÁO BÀI TẬP LỚN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Đề tài: Bài toán Quản lý nhân viên, xây dựng cấu trúc danh sách liên kết đơn(Linked list) để quản lý dãy số

Giảng viên hướng dẫn:

TS. Hoàng Văn Thông

Sinh viên thực hiện:

Đặng Hoàng Huy – 231230797

Lớp:

Công nghệ thông tin 2 – K64

Hà Nội, tháng 11 năm 2024

MỤC LỤC

Lời mở đầu

Đầu tiên, em xin phép gửi lời cảm ơn và lòng biết ơn sâu sắc đến thầy Hoàng Văn Thông về sự hướng dẫn và cung cấp kiến thức quý báu trong môn học Cấu trúc dữ liệu và Giải thuật. Bài báo cáo này là kết quả của quá trình tìm hiểu, học hỏi những kiến thức mà thầy đã giảng dạy cũng như khai thác, đi đôi với quá trình thực hiện bài tập lớn. Em cũng mong nhận được sự đánh giá tổng quát từ thầy, để bù đắp cho sự thiếu sót của em.

Bài tập lớn này đã giúp em hiểu rõ hơn về quy trình thiết kế cấu trúc dữ liệu và giải thuật, cùng với khả năng áp dụng kiến thức vào quá trình thực hiện. Nó cũng giúp em phát triển kỹ năng, tư duy logic và khả năng giải quyết vấn đề.

Một lần nữa em xin gửi lời cảm ơn đặc biệt đến thầy, người đã hướng dẫn và để cho em những lời khuyên quý báu trong suốt thời gian qua. Không có được những sự chỉ dẫn của thầy, em không thể áp dụng chúng vào để hoàn thành bài tập này một cách hiệu quả.

Bài báo cáo này sẽ tập trung trình bày về nhiệm vụ của em, quá trình và triển khai cấu trúc dữ liệu và giải thuật, cũng như kết quả đạt được. Em rất mong nhận được sự đánh giá và phản hồi từ thầy Hoàng Văn Thông để cải thiện kiến thức và kỹ năng của bản thân mình.

I. Giới thiệu

1. Lợi ích

Để đáp ứng yêu cầu môn học cũng như mong muốn tìm hiểu, mở rộng kiến thức về ngôn ngữ lập trình nói chung và môn Cấu trúc dữ liệu và giải thuật (Data Structures and Algorithms) nói riêng, em đã hướng tới đề tài với các tiêu chí:

- Hữu ích.
- Áp dụng được kiến thức đã học được từ thầy và mọi người.
- Học hỏi được kiến thức mới.

2. Định nghĩa các bài được thầy giao

Bài toán 1 (Phần A - Bài toán quản lí):

- Đề tài của bài toán mà em lựa chọn là Quản lí nhân viên. Lí do em lựa chọn đề tài này vì:
    - Dễ hình dung cấu trúc dữ liệu : Nhân viên có những thuộc tính đặc trưng như tên, mã nhân viên, tuổi tác và chức vụ.
    - Tạo cơ hội rèn luyện kĩ năng lập trình: Đề tài này phù hợp để luyện tập các kĩ năng về lập trình như lập trình hướng đối tượng (OOP), sử dụng cấu trúc dữ liệu trong thư viện chuẩn (STL) như List, Vector, cũng như các thao tác sắp xếp, tìm kiếm và so sánh đối tượng.
- Input: Các thao tác:
    - Thêm nhân viên
    - Xóa nhân viên
    - In danh sách
    - Sắp xếp theo tên, tuổi, chức vụ
    - Tìm nhân viên theo mã nhân viên

  Yêu cầu của input :

  Mã nhân viên, tuổi(số nguyên) ,Tên, chức vụ(chuỗi kí tự).

  Lựa chọn chức năng thông qua người dùng
- Output: Hiển thị kết quả các thao tác thêm, xóa, tìm kiếm, sắp xếp và in danh sách.

Bài toán 2 (Phần B - Bài số 6 trong danh sách BTL):

- Input:
    - Xây dựng cấu trúc Linked List
    - Các thao tác : tạo danh sách, nhập số vào danh sách, thêm một phần tử kiểm tra, sắp xếp, xóa.
- Output:
    - Hiển thị kết quả của thao tác tạo, thêm, nhập, sắp xếp, kiểm tra, xóa.

3.Các nhiệm vụ mà mức độ hoàn thành

Nhiệm vụ	Mức độ hoàn thành
Tìm hiểu cấu trúc dữ liệu danh sách liên kết (List) trong C++. Lên ý tưởng cho bài toán quản lí nhân viên Triển khai bài toán với cấu trúc danh sách (List) để quản lí ô tô Kiểm thử và hoàn thiện chương trình.	*Đã hoàn thành*
Tìm hiểu cấu trúc dữ liệu danh sách liên kết đơn (Linked List). Tìm hiểu và cài đặt danh sách liên kết đơn. Xây dựng các hàm theo yêu cầu của bài toán. Kiểm thử và hoàn thiện chương trình.	*Đã hoàn thành*
Làm báo cáo Lưu trữ code bằng Github	*Đã hoàn thành*

II. Triển khai cài đặt

1. Ngôn ngữ lập trình và thư viện

- Ngôn ngữ lập trình : Sử dụng ngôn ngữ lập trình C++
- Thư viện : Sử dụng thư viện đã được tích hợp sẵn trong C++

2. Tổ chức chương trình và đóng gói

- Tổ chức chương trình : Viết chương trình C++ bằng DevC++ hoặc Visual Studio Code, mỗi đoạn code xử lý một chức năng.
- Đóng gói : Các phần code liên quan đến chương trình được đóng gói ở Github

- Link: https://github.com/ihzuiyo/BTL_CTDLGT

III. Phương pháp lựa chọn để giải quyết bài toán 1 (Bài toán quản lý)

1. Khái quát bài toán

a. Ý tưởng

Trong mô hình quản lý của một tổ chức, em có để ý đến cách quản lý theo danh sách được quản lý theo từng cột, mỗi cột chứa các loại thông tin cá nhân khác nhau cũng như cho biết người đó thuộc chức vụ nào để dễ dàng điều hành trong tổ chức. Vì vậy em chọn chủ đề này để xây dựng một chương trình minh họa lại quá trình quản lý danh sách nhân viên.

b. Hướng giải quyết bài toán

- **Xây dựng lớp NhanVien**
  - Chứa các thuộc tính cơ bản của nhân viên như mã nhân viên, tên, tuổi và chức vụ của người đó.
  - Xây dựng phương thức nhập, xuất thông tin của nhân viên.
- **Xây dựng lớp Danh sách nhân viên (nvList):**
  - Sử dụng cấu trúc List để lưu trữ danh sách nhân viên.
  - Triển khai các phương thức như thêm, xóa , in danh sách , sắp xếp , tìm kiếm nhân viên.
- **Xây dựng lớp UngDung:**
  - Hiển thị menu để người dùng thao tác với các chức năng như thêm, xóa, tìm kiếm, sắp xếp và hiển thị danh sách.
- **Kiểm thử:**
  - Chạy thử với nhiều dữ liệu khác nhau để đảm bảo các thao tác thực hiện đúng theo yêu cầu.

2. Phân tích chương trình

a. classNhanVien.cpp : Lớp nhân viên

i. **Mục đích :** Quản lý thông tin của từng nhân viên

ii. **Các thuộc tính chính:**

1. maNV : mã nhân viên
2. ten : tên nhân viên
3. tuoi : tuổi nhân viên
4. chucvu : chức vụ của nhân viên

iii. **Các phương thức:**

1. **Nhập/Xuất** : Định nghĩa toán tử >> để nhập thông tin nhân viên từ người dùng và << để xuất thông tin nhân viên ra màn hình.
2. **Getter:** Các phương thức lấy dữ liệu như lay_maNV, lay_ten, lay_tuoi, lay_chucvu giúp truy xuất thông tin nhân viên.

b. classnvList.cpp : Lớp nvList (Danh sách nhân viên)

i. **Mục đích :** Quản lý danh sách các nhân viên, sử dụng cấu trúc danh sách(List) trong STL

ii. Các phương thức chính:

1. **Thêm/xóa nhân viên :** themNhanVien để thêm nhân viên vào danh sách, xoaNhanVien để xóa nhân viên theo mã nhân viên (maNV) trong đó có kiểm tra và thông báo nếu như mã nhân viên không hợp lệ.
2. **Sắp xếp:** sapxeptheoTen để sắp xếp danh sách nhân viên thông qua tên của nhân viên đó theo thứ tự bảng chữ cái, sapxeptheoTuoi để sắp xếp danh sách độ tuổi từ thấp đến cao , sapxeptheoChucVu để sắp xếp danh sách nhân viên thông qua kí tự đầu tiên của chữ cái đầu tiên theo bảng chữ cái.
3. **Tìm kiếm:** timNhanVien để truy xuất thông tin của nhân viên đó thông qua nhập mã nhân viên (maNV).
4. **In danh sách:** inDanhSach hiển thị toàn bộ nhân viên hiện có trong danh sách.

c. UngDung.cpp : Lớp UngDung:

i. **Mục đích :** Là giao diện chính của chương trình, cho phép người dùng tương tác với hệ thống quản lý nhân viên thông qua các chức năng có trong lớp nvList.

ii. **Chức năng chính:** Chứa hàm main(), là điểm để chạy ứng dụng và gọi đến các lớp NhanVien,nvList và UngDung.

iii. Các chức năng trong menu:

1. Thêm nhân viên
2. Xóa nhân viên
3. In danh sách nhân viên
4. Sắp xếp nhân viên theo tên
5. Sắp xếp nhân viên theo tuổi
6. Sắp xếp nhân viên theo chức vụ
7. Tìm nhân viên theo mã nhân viên
8. Kết thúc chương trình

iv. Cấu trúc điều khiển:

1. Sử dụng vòng lặp do-while để giữ chương trình chạy cho đến khi người dùng chọn thoát.
2. Phản hồi lựa chọn của người dùng thông qua các câu lệnh switch, gọi các phương thức phù hợp từ lớp nvList như thêm, xóa, sắp xếp và tìm kiếm.

IV. Phương pháp lựa chọn để giải quyết bài toán 2 (Bài số 6 trong danh sách)

- Khái quát nội dung:

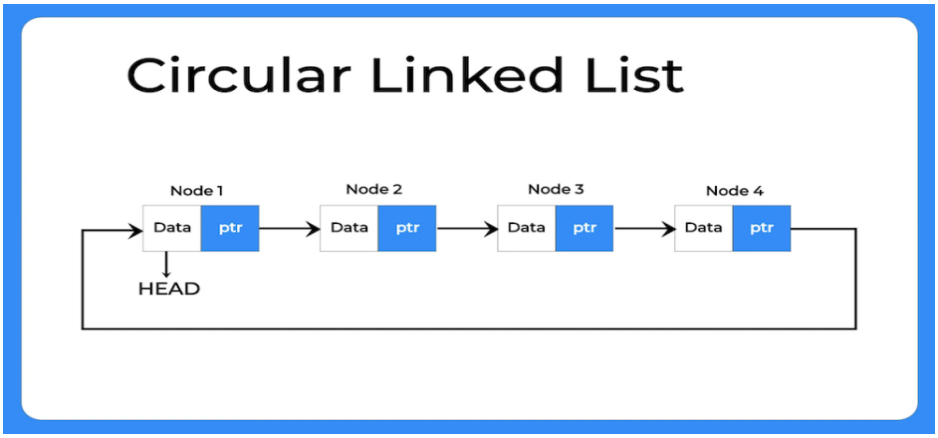
- Để giải quyết được bài toán số 6 trong danh sách BTL :
  - Cài đặt cấu trúc **Linked List** theo yêu cầu của bài toán.

1. Cấu trúc dữ liệu

- Danh sách liên kết đơn(Linked list):

Danh sách liên kết đơn là một tập hợp tuyến tính gồm các phần tử dữ liệu, với thứ tự không được đưa ra bởi vị trí vật lý của chúng trong bộ nhớ. Thay vào đó, mỗi phần tử chỉ đến phần tử tiếp theo. Là một cấu trúc dữ liệu bao gồm một tập hợp các nút, mỗi nút chứa dữ liệu và một tham chiếu tới nút tiếp theo trong dãy. Gồm các phương thức như:

- Phương thức truy cập:
  - Front() : Truy cập phần tử đầu
  - Back() : Truy cập phần tử cuối
- Chèn và xóa:
  - Insert() : Chèn phần tử tại vị trí cố định
  - Erase() : Xóa phần tử tại vị trí cố định
- Iterator (bộ lặp):
  - Begin() : trỏ đến phần tử đầu tiên.
  - End() : trỏ đến phần tử sau phần tử cuối cùng (thường có giá trị là null hoặc con trỏ nullptr).
- Phương thức cập nhật:
  - Push_front(T e) : Chèn một phần tử vào đầu danh sách
  - Push_back(T e) : Chèn một phần tử vào cuối danh sách
  - Pop_front() : Xóa phần tử đầu
  - Pop_back() : Xóa phần tử cuối
  - Swap() : Hoán đổi nội dung của một node này với một node khác cùng kiểu dữ liệu.
- Phương thức chung:
  - Size() : Kiểm tra kích thước của danh sách
  - Empty() : kiểm tra xem danh sách có rỗng hay không.



Hình 1: Minh họa danh sách liên kết đơn.

V. Phân tích chương trình của bài toán 2

1. Cài đặt cấu trúc trừu tượng List

a. Phần khai báo và định nghĩa chung

```
#include<bits/stdc++.h>

#ifndef __List__
#define __List__
using namespace std;
...
#endif
```

- bits/stdc++.h : gọi tất cả các thư viện chuẩn của C++.
- ifndef và define : giúp tránh việc khai báo lại lớp List nếu mã nguồn này được đưa vào nhiều lần trong một chương trình.
- using namespace std: sử dụng tất cả các tính năng của thư viện chuẩn(std) mà không cần phải thêm tiền tố std:: trước mỗi lần gọi hàm hoặc đối tượng chuẩn.
- endif: kết thúc một khối điều kiện được bắt đầu với ifndef.

b. Khai báo các thuộc tính và hàm của lớp Node

```
class Node{
    public:
        int data; // chua value
        Node* next; // chua vi tri cua node tiep theo
       Node (int value) : data(value),next(nullptr){} //Ham tao Node

};
```

- data: biến để lưu trữ giá trị của một node
- next : con trỏ trỏ tới địa chỉ của node tiếp theo trong danh sách liên kết.
- Node() : là hàm khởi tạo giá trị ban đầu cho các biến data (giá trị do người dùng nhập vào) và next(ban đầu là null).
- Sử dụng kiểu dữ liệu int để thực hiện được những yêu cầu tiếp theo của bài toán.

c. Khai báo các thuộc tính và hàm của lớp Dayso

```
class Dayso{
    private:
        Node* head;
    public:
        Dayso() : head(nullptr) {}
        ...
        ~Dayso(){
            while(head){
                Node* temp = head;
                head = head->next;
                delete temp;
            }
        }
};
```

- head : là một con trỏ trỏ đến một đối tượng thuộc kiểu Node, đóng vai trò như một điểm bắt đầu của một danh sách liên kết.

- Dayso() : là hàm khởi tạo(constructor) mặc định cho lớp dãy số, khởi tạo head bằng nullptr biểu thị một danh sách liên kết trống.
- ~Dayso(): là hàm hủy(destructor) dùng để giải phóng bộ nhớ đã cấp phát động cho danh sách liên kết khi một đối tượng của lớp Dayso bị hủy.

```
void add(int value){
        Node* newNode = new Node(value);
        if(!head){
            head = newNode;
        }
        else{
            Node* temp = head;
            while(temp->next){
                temp = temp->next;
            }
            temp->next = newNode;
        }
}
```

- Khi thêm phần tử, hàm sẽ khởi tạo một nút mới newNode với giá trị là value
- Kiểm tra nếu danh sách trống thì sẽ đặt head trỏ đến newNode, newNode sẽ là phần tử đầu tiên của danh sách.
- Nếu danh sách không trống thì sẽ tạo một con trỏ temp duyệt qua danh sách bắt đầu từ head duyệt đến cuối danh sách và liên kết newNode vào cuối danh sách.

```
void print() const{
        cout << "DAY SO DA NHAP: ";
        Node* temp = head;
        while(temp){
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
}
```

- Con trỏ temp bắt đầu từ head duyệt qua từng nút trong danh sách cho tới khi không còn nút nào để xử lý.
- In ra giá trị trong data của mỗi nút.
- “const” ở cuối phương thức chỉ ra hàm trên không thay đổi trạng thái của đối tượng.

```
void createList(){
        cout << "Nhap cac so vao danh sach (nhap # de dung qua trinh nhap cac so): ";
        while(true){
           string input; //Chuoi kiem tra dau #
           cin >> input;
           if(input == "#"){
               break;
           }
           int value = static_cast<int>(stoi(input));
           add(value);
        }

     }
```

- Dùng vòng lặp vô hạn while(true) để đọc từng giá trị nhập vào, quá trình sẽ dừng lại khi chúng ta nhập #.
- Tạo biến value nhằm mục đích lưu các giá trị được nhập vào, chuyển kiểu dữ liệu của "input" từ string sang kiểu int bằng stoi(input), static_cast<int> đảm bảo chính xác kiểu dữ liệu là kiểu int.
- Khi xác nhận giá trị đó thuộc kiểu dữ liệu int, sử dụng hàm add trước đó để thêm giá trị vào danh sách liên kết.

```
void insertAt(int pos,int value){
        Node* newNode = new Node(value);
        if(pos == 1){ // neu chen vao vi tri dau tien danh sach
           newNode->next = head;
           head = newNode;
           return;
        }
        Node* temp = head;
        for(int i=1;i < pos-1 && temp; i++){
           temp = temp->next;
        }
        if(!temp){
           cout << "Vi tri chen khong hop le" << endl;
           delete newNode;
        }
        else{
           newNode->next = temp->next;
           temp->next = newNode;
        }
     }
```

- Khi vị trí chèn vào vị trí thứ nhất, thì nút chứa giá trị được chèn sẽ trở thành nút đầu tiên của danh sách liên kết.

- Dùng vòng lặp duyệt tới vị trí cần chèn vào trong danh sách, điều kiện temp đảm bảo rằng vòng lặp không duyệt quá số lượng nút trong danh sách.
- Nếu nhập vào vị trí lớn hơn số nút trong danh sách, sẽ giải phóng bộ nhớ vì thao tác chèn không thể thực hiện.
- Nếu vị trí hợp lệ, tiến hành duyệt qua các nút trong danh sách sau đó chèn giá trị vào vị trí đã nhập.

```
int countEqualK(int k){
        int count = 0;
        Node* temp = head;
        while(temp){
            if(temp->data == k){
                count++;
            }
            temp = temp->next;
        }
        return count;
    }
```

- Khai báo một biến để đếm số giá trị trùng với giá trị k nhập vào từ bàn phím.
- Sử dụng vòng lặp duyệt từ đầu danh sách, nếu phát hiện giá trị bằng giá trị k thì biến đếm sẽ tăng cho tới cuối danh sách.
- Hàm sẽ trả về biến đếm.

```
void checkThreeValueEqual(){
        Node* temp = head;
        int index = 1;
        while(temp && temp->next && temp->next->next){
            if((temp->data > 0 && temp->data % 2 == 0) && (temp->next->data > 0 &&
temp->next->data % 2 == 0) && (temp->next->next->data > 0 && temp->next->next->data
% 2 == 0)){
                cout << "THONG BAO - Co 3 so chan dung canh nhau" << endl;
                cout << "THONG BAO - Vi tri cua 3 so chan la: " << index << ", " << index + 1
<< ", " << index + 2 << endl;
                return;
            }
            temp = temp->next;
            index++;
        }
        cout << "KHONG CO 3 SO CHAN DUONG DUNG CANH NHAU!" << endl;
    }
```

- Khai báo biến index đánh dấu vị trí được duyệt tới trong danh sách.
- Dùng vòng lặp duyệt từ đầu danh sách với điều kiện “temp && temp->next && temp->next->next” để đảm bảo rằng nút đầu tiên và 2 nút tiếp theo đều tồn tại, chỉ duyệt khi trong danh sách có ít nhất 3 nút.

- Kiểm tra lần lượt 3 nút liên tiếp điều kiện thỏa mãn yêu cầu bài toán.

```
void sortList(){
        if(!head || !head->next){
            cout << "THONG BAO - Trong danh sach can it nhat 2 so de tien hanh sap xep."
<< endl;

            return;
        }
        for(Node* i = head;i->next != NULL;i = i->next){
            for(Node* j = i->next;j != NULL;j = j->next){
                if(i->data > j->data){
                    swap(i->data,j->data);
                }
            }
        }
    }
```

- Đầu tiên kiểm tra xem trong danh sách có từ 2 nút trở lên không.
- Nếu thỏa mãn điều kiện ban đầu, sử dụng 2 vòng lặp để kiểm tra lần lượt 1 phần tử với các phần tử còn lại trong danh sách.
- Kiểm tra giá trị nhỏ hơn sau đó tiến hành đổi vị trí của 2 giá trị tương ứng.

```
        bool isPrime(int n){
                if(n <= 1){
                    return false;
                }
                for(int i = 2;i*i <= n;i++){
                    if(n % i == 0){
                        return false;
                    }
                }
                return true;
            }
```

- Kiểm tra xem giá trị được đưa vào hàm có phải số nguyên tố hay không. Nếu phải trả về giá trị true, nếu không thì trả giá trị false.

```
void removePrime(){
        while(head && isPrime(head->data)){
           Node* temp = head;
           head = head->next;
           delete temp;
        }

        Node* temp = head;
        while(temp && temp->next){
           if(isPrime(temp->next->data)){
              Node* goDelete = temp->next;
              temp->next = temp->next->next;
              delete goDelete;
           }
           else{
              temp = temp->next;
           }
        }
     }
```

- Vòng lặp đầu tiên để xử lý các nút có giá trị nguyên tố ở đầu danh sách. Điều kiện kiểm tra là "head" không rỗng và giá trị của "head" là số nguyên tố.
    - Nếu điều kiện đúng thì tạo một con trỏ temp để giữ lại nút đầu hiện tại.
    - Di chuyển nút head tới nút tiếp theo để cập nhật nút đầu của danh sách
    - Xóa nút temp để giải phóng bộ nhớ.
    - Vòng lặp sẽ tiếp tục cho đến khi head không còn trỏ tới nút có giá trị là số nguyên tố hoặc danh sách trống.
- Vòng lặp tiếp theo để duyệt qua các nút còn lại trong danh sách và xóa các nút có giá trị nguyên tố.
    - Điều kiện của vòng lặp này đảm bảo rằng nút hiện tại và nút tiếp theo tồn tại.
    - Nếu giá trị của nút tiếp theo là số nguyên tố thì sẽ tạo một con trỏ trỏ tới nút cần xóa, sau đó cập nhật lại nút tiếp theo của nút đang được cho là số nguyên tố hiện tại, sau đó xóa nút "goDelete" để loại bỏ số nguyên tố cũng như giải phóng bộ nhớ.
    - Nếu giá trị của nút tiếp theo không phải là số nguyên tố, thì sẽ tiếp tục duyệt qua các nút tiếp theo trong danh sách cho tới nút cuối cùng.

```
void remoteDuplicates(){
        Node* temp = head;
        while(temp){
            Node* runner = temp;
            while(runner->next){
                if(runner->next->data == temp->data){
                    Node* goDelete = runner->next;
                    runner->next = runner->next->next;
                    delete goDelete;
                }
                else{
                    runner = runner->next;
                }

            }
            temp = temp->next;
        }
    }
```

- Tạo một con trỏ runner bắt đầu từ temp(tức là từ đầu danh sách)
- Điều kiện của vòng lặp đảm bảo rằng vòng lặp sẽ tiếp tục duyệt cho tới khi không còn nút nào để xử lý.
- Vòng lặp bên trong duyệt qua các nút còn lại sau nút temp, đảm bảo trỏ đến nút cuối cùng trong danh sách
    - Điều kiện của if kiểm tra dữ liệu của nút đang trỏ tới bằng với dữ liệu của temp hay không? Nếu có, nghĩa là đã tìm thấy phần tử trùng lặp.
    - Loại bỏ phần tử trùng lặp bằng cách tạo một con trỏ goDelete trỏ tới nút cần xóa, sau đó cập nhật lại con trỏ runner->next bỏ qua nút đang được coi là trùng lặp.
    - Cuối cùng, xóa nút goDelete để loại bỏ phần tử trùng lặp cũng như là giải phóng bộ nhớ.
    - Nếu điều kiện của if không đúng, tức là không có phần tử trùng lặp thì sẽ di chuyển tới nút tiếp theo để tiếp tục so sánh.
- Di chuyển con trỏ temp trỏ tới nút tiếp theo trong danh sách và tiếp tục vòng lặp.

d. Khai báo các thuộc tính ở hàm main()

```
main(){
        Dayso Ds;
        while(1){
          ...
        }
}
```

- Khai báo đối tượng thuộc lớp Dayso.
- Sử dụng vòng lặp vô hạn để xây dựng ứng dụng phù hợp với yêu cầu của bài toán.

Code đầy đủ có thể xem ở Github : https://github.com/ihzuiyo/BTL_CTDLGT

V. Kết quả thực nghiệm

1. Dữ liệu

- Với bài toán 1 (bài toán quản lý) thì người dùng tương tác với chương trình để có được kết quả mong muốn.

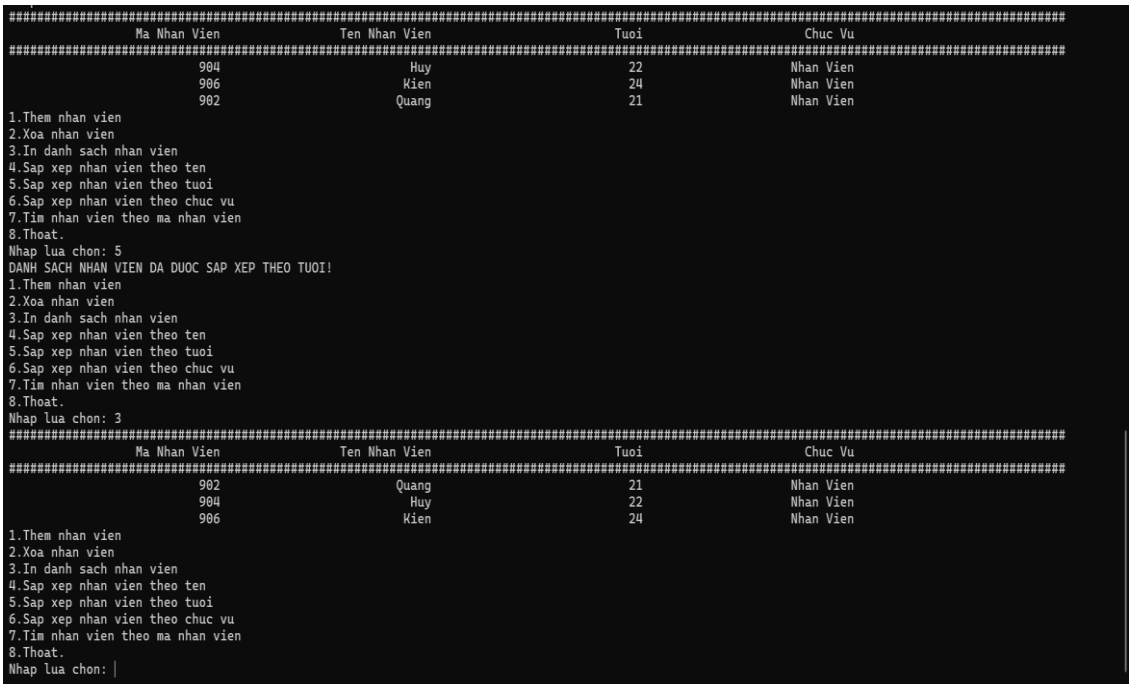
- Với bài toán 2 (Bài 6 trong danh sách) thì người dùng tương tác với chương trình để có được kết quả mong muốn.

2.Các kết quả

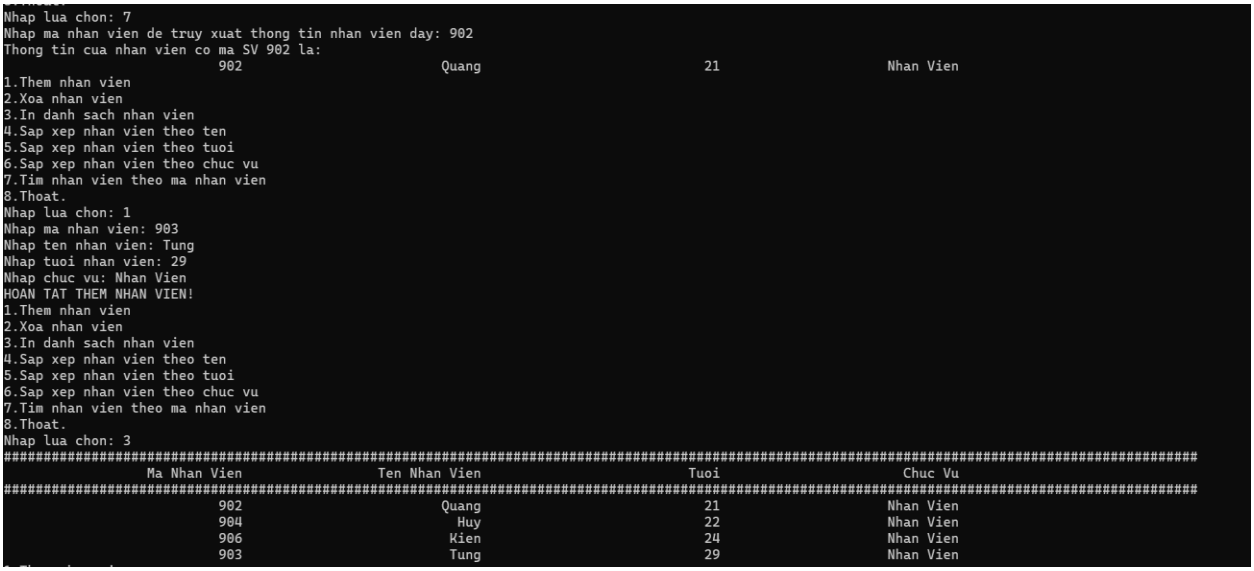
- Đối với bài toán 1:

- Ví dụ về nhập, xuất các thông tin.

- Các phương thức thêm, xóa, sắp xếp, tìm kiếm đều hoạt động tốt.

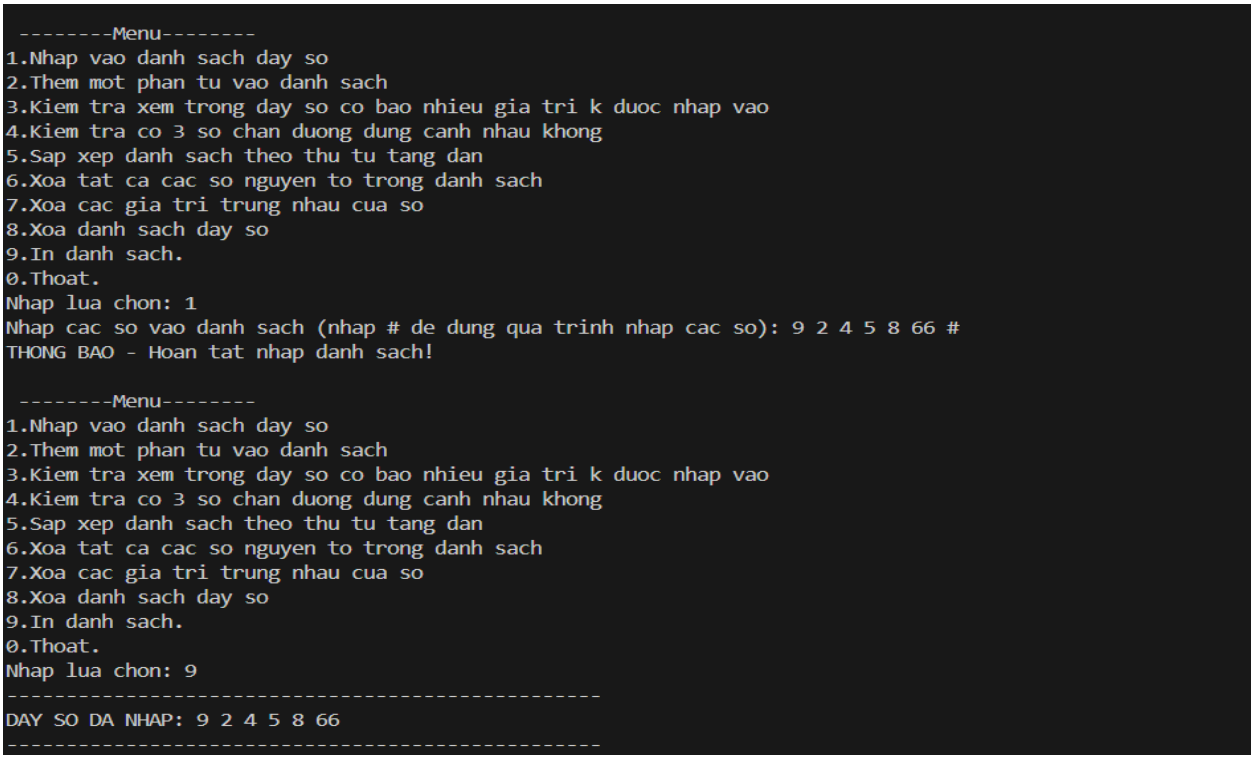


Hình 2: Mô tả thao tác sắp xếp nhân viên theo độ tuổi.



Hình 3: Mô tả quá trình tìm kiếm để truy xuất thông tin theo mã nhân viên và thao tác thêm nhân viên vào danh sách.

- **Đối với bài toán 2:**



Hình 4: Mô tả quá trình nhập và in dãy số

```
-------------------------------------------------
DAY SO DA NHAP: 9 2 4 5 8 66
-------------------------------------------------
 --------Menu--------
1.Nhap vao danh sach day so
2.Them mot phan tu vao danh sach
3.Kiem tra xem trong day so co bao nhieu gia tri k duoc nhap vao
4.Kiem tra co 3 so chan duong dung canh nhau khong
5.Sap xep danh sach theo thu tu tang dan
6.Xoa tat ca cac so nguyen to trong danh sach
7.Xoa cac gia tri trung nhau cua so
8.Xoa danh sach day so
9.In danh sach.
0.Thoat.
Nhap lua chon: 2
THONG BAO - Nhap gia tri can chen: 8
THONG BAO - Nhap vi tri can chen: 4

 --------Menu--------
1.Nhap vao danh sach day so
2.Them mot phan tu vao danh sach
3.Kiem tra xem trong day so co bao nhieu gia tri k duoc nhap vao
4.Kiem tra co 3 so chan duong dung canh nhau khong
5.Sap xep danh sach theo thu tu tang dan
6.Xoa tat ca cac so nguyen to trong danh sach
7.Xoa cac gia tri trung nhau cua so
8.Xoa danh sach day so
9.In danh sach.
0.Thoat.
Nhap lua chon: 9
-------------------------------------------------
DAY SO DA NHAP: 9 2 4 8 5 8 66
-------------------------------------------------
```

Hình 5: Mô tả thao tác thêm giá trị vào dãy số, vị trí do người dùng chọn.

```
Nhap lua chon: 9
-------------------------------------------------
DAY SO DA NHAP: 9 2 4 8 5 8 66
-------------------------------------------------
 --------Menu--------
1.Nhap vao danh sach day so
2.Them mot phan tu vao danh sach
3.Kiem tra xem trong day so co bao nhieu gia tri k duoc nhap vao
4.Kiem tra co 3 so chan duong dung canh nhau khong
5.Sap xep danh sach theo thu tu tang dan
6.Xoa tat ca cac so nguyen to trong danh sach
7.Xoa cac gia tri trung nhau cua so
8.Xoa danh sach day so
9.In danh sach.
0.Thoat.
Nhap lua chon: 3
THONG BAO - Nhap gia tri k: 8
THONG BAO - Co 2 gia tri trung voi k

 --------Menu--------
1.Nhap vao danh sach day so
2.Them mot phan tu vao danh sach
3.Kiem tra xem trong day so co bao nhieu gia tri k duoc nhap vao
4.Kiem tra co 3 so chan duong dung canh nhau khong
5.Sap xep danh sach theo thu tu tang dan
6.Xoa tat ca cac so nguyen to trong danh sach
7.Xoa cac gia tri trung nhau cua so
8.Xoa danh sach day so
9.In danh sach.
0.Thoat.
Nhap lua chon: 4
THONG BAO - Co 3 so chan dung canh nhau
THONG BAO - Vi tri cua 3 so chan la: 2, 3, 4
```

Hình 6: Mô tả thao tác kiểm tra có bao nhiêu giá trị k và kiểm tra 3 số chẵn dương đứng cạnh nhau.

```
Nhap lua chon: 9
-------------------------------------------------
DAY SO DA NHAP: 9 2 4 8 5 8 66
-------------------------------------------------
 --------Menu--------
1.Nhap vao danh sach day so
2.Them mot phan tu vao danh sach
3.Kiem tra xem trong day so co bao nhieu gia tri k duoc nhap vao
4.Kiem tra co 3 so chan duong dung canh nhau khong
5.Sap xep danh sach theo thu tu tang dan
6.Xoa tat ca cac so nguyen to trong danh sach
7.Xoa cac gia tri trung nhau cua so
8.Xoa danh sach day so
9.In danh sach.
0.Thoat.
Nhap lua chon: 7
THONG BAO - Da xoa cac gia tri trung nhau

 --------Menu--------
1.Nhap vao danh sach day so
2.Them mot phan tu vao danh sach
3.Kiem tra xem trong day so co bao nhieu gia tri k duoc nhap vao
4.Kiem tra co 3 so chan duong dung canh nhau khong
5.Sap xep danh sach theo thu tu tang dan
6.Xoa tat ca cac so nguyen to trong danh sach
7.Xoa cac gia tri trung nhau cua so
8.Xoa danh sach day so
9.In danh sach.
0.Thoat.
Nhap lua chon: 9
-------------------------------------------------
DAY SO DA NHAP: 9 2 4 8 5 66
-------------------------------------------------
```

Hình 7: Mô tả thao tác xóa các giá trị trùng nhau và chỉ để lại một số.

VI. Kết luận

1. Mức độ hoàn thành

- Đã hoàn thành các yêu cầu, đáp ứng đủ các tính năng lẫn chức năng từ danh sách đối tượng và các thao tác tìm kiếm, sắp xếp, thêm/bớt.

- Áp dụng được các kiến thức của môn học, kết hợp với một số kiến thức tự trao dồi để làm sản phẩm.

2. Khó khăn

- Thuật toán còn có độ phức tạp thời gian lớn, cần có thêm cũng như nắm vững các kiến thức đã học để vận dụng tốt vào bài tập.

- Cần hiểu bản chất của cấu trúc và thuật toán

- Lượng kiến thức rất rộng.

3. Bài học rút ra

- Môn học Cấu trúc dữ liệu và giải thuật là vô cùng quan trọng, là phần kiến thức không thể thiếu đối với sinh viên khoa Công Nghệ Thông Tin.

- Cần sử dụng kiến thức đã học để áp dụng vào một việc nào đó để giúp nắm vững kiến thức và tìm ra những điều mới mẻ.

- Sau khi hoàn thành sản phẩm, cần nghĩ thêm để phát triển sản phẩm khắc phục những hạn chế của sản phẩm.

VII. Lời cảm ơn

Cấu trúc dữ liệu và giải thuật là môn học rất quan trọng đối với sinh viên khoa Công nghệ thông tin. Em xin chân thành cảm ơn sự chỉ dạy nhiệt tình và những ví dụ minh họa của thầy giúp em học tập. Cảm ơn thầy vì đã giao một bài tập lớn để em có được cơ hội thử thách bản thân vào một project không chỉ đơn giản là code mà còn là tinh thần làm việc, tìm hiểu những thứ mới mẻ.

Cuối cùng, sản phẩm em vẫn còn nhiều thiếu sót do thiếu nhiều kinh nghiệm cũng như kỹ năng, mong thầy nhận xét và góp ý để em hoàn thiện hơn.