**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**Đề tài 601: Xây dựng thư viện có các thao tác của một danh sách liên kết**

**SINH VIÊN THỰC HIỆN:**

**Phạm Quang Nhựt           LỚP: 21TCLT-Nhat1              NHÓM: 2**

**Huỳnh Hải Đăng                LỚP: 21TCLT-Nhat1              NHÓM: 2**

**GIẢNG VIÊN HƯỚNG DẪN:**

**Nguyễn Văn Hiệu                 GIẢNG VIÊN KHOA**

MỤC LỤC

**CHƯƠNG 1:** MỞ ĐẦU [**1**](#_heading=h.gjdgxs)

[1. Lý do chọn đề tài: 1](#_heading=h.30j0zll)

[2. Mục tiêu của đề tài. 1](#_heading=h.1fob9te)

[3. Phạm vi nghiên cứu. 1](#_heading=h.3znysh7)

[4. Phương pháp nghiên cứu.](#_heading=h.2et92p0) 1

I) Tổng quan đề tài [1](#_heading=h.2et92p0)

1.1. Định nghĩa 2

1.2. Phân loại 2

[**CHƯƠNG 2:** CƠ SỞ LÝ THUYẾT **2**](#_heading=h.tyjcwt) [2](#_heading=h.2s8eyo1)

2.[1. Khái niệm. 2](#_heading=h.17dp8vu)

2[.2.Thao tác trên 1 node. 3](#_heading=h.3rdcrjn)

2.[3. Các thao tác trên danh sách liên kết đơn 3](#_heading=h.26in1rg)

2.4[. Duyệt danh sách liên kết đơn 4](#_heading=h.lnxbz9)

[2.5. Chèn một phần tử vào danh sách 4](#_heading=h.35nkun2)

2.6[. Xóa một phần tử ra khỏi danh sách: có 2 trường hợp xảy ra 6](#_heading=h.1ksv4uv)

2.7[. Tìm kiếm phần tử trong danh sách 8](#_heading=h.44sinio)

2.8[. Đảo các phần tử trong danh sách 8](#_heading=h.2jxsxqh)

[**CHƯƠNG 3:**](#_heading=h.xvir7l) TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ BÀI TOÁN [**10**](#_heading=h.xvir7l)

3[.1. Đặc tả bài toán. 10](#_heading=h.z337ya)

3[.2. Yêu cầu hệ thống. 10](#_heading=h.3j2qqm3)

3[.3. Phân tích thiết kế chương trình. 10](#_heading=h.1y810tw)

3[.3.1. Cấu trúc dữ liệu chương trình. 10](#_heading=h.4i7ojhp)

3[.3.2. Các chức năng của chương trình. 12](#_heading=h.2xcytpi)

3[.3.2.1. Nhập danh sách sinh viên. 12](#_heading=h.1ci93xb)

3[.3.2.2. Hiển thị danh sách sinh viên. 12](#_heading=h.3whwml4)

3[.3.2.3. Tìm kiếm sinh viên theo mã sinh viên. 12](#_heading=h.2bn6wsx)

3[.3.2.4.Xóa sinh viên theo mã số sinh viên. 12](#_heading=h.qsh70q)

3[.3.2.5. Bổ sung sinh viên mới vào danh sách. 12](#_heading=h.3as4poj)

3[.3.2.6. Hiển thị sinh viên có điểm trung bình cao nhất. 12](#_heading=h.1pxezwc)

3[.3.2.7. Sắp xếp danh sách sinh viên tăng dần theo trường điểm trung bình 12](#_heading=h.49x2ik5)

**CHƯƠNG 4:** CÀI ĐẶT CHƯƠNG TRÌNH**12**

[4.1. Định nghĩa cấu trúc cho chương trình quản lý sinh viên 1](#_heading=h.147n2zr)2

[4.2. Ngôn ngữ cài đặt 13](#_heading=h.3o7alnk)

[4.3. Kết quả 1](#_heading=h.23ckvvd)3

[4.3.1. Khởi tạo danh sách sinh viên 1](#_heading=h.23ckvvd)4

[4.3.1.2. Hiển thị danh sách sinh viên](#_heading=h.ihv636) 15

[4.3.1.3. Tìm kiếm sinh viên theo mã sinh viên](#_heading=h.32hioqz) 15

[4.3.1.4. Xóa sinh viên theo mã sinh viên](#_heading=h.1hmsyys) 16

[4.3.1.5. Bổ sung sinh viên mới vào danh sách](#_heading=h.41mghml) 17

4.3.1.6. In sinh viên có điểm trung bình lớn nhất 17

4.3.1.7. Sắp xếp tăng dần theo điểm trung bình 18

4.3.1.8. Chỉnh sửa thông tin sinh viên 19

4.3.1.9. Sắp xếp sinh viên theo tên 20

4.3.1.10. Xuất danh sách ra file text 21

4.3.1.11. Đọc danh sách từ file text 21

4.3.1.12. Thêm sinh viên mới vào đầu danh sách 22

4.3.1.13. Xóa sinh viên ở cuối danh sách 23

4.3.1.14. Xóa sinh viên ở đâu danh sách 24

4.3.1.15. Xóa sinh viên theo tên 24

4.3.1.16. Tìm kiếm sinh viên theo tên 24

4.3.1.17. Tìm sinh viên có số điểm trên số điểm nhập vào 26

4.3.1.18. Tìm sinh viên có số điểm dưới số điểm nhập vào 27

4.3.2. Chương trình quản lý sinh viên 28

4.3.2.0. Màn hình kết quả 28

4.3.2.1. Menu chính của chương trình 29

4.3.2.2. Khởi tạo danh sách sinh viên 30

4.3.2.3. Hiển thị danh sách sinh viên 31

4.3.2.4. Tìm kiếm sinh viên theo mã sinh viên 32

4.3.2.5. Xóa sinh viên theo mã sinh viên 32

4.3.2.6. Bổ sung sinh viên mới vào danh sách 33

4.3.2.7. In sinh viên có điểm trung bình lớn nhất 33

4.3.2.8. Sắp xếp danh sách sinh viên tăng dần theo điểm trung bình 33

4.3.2.9. Chỉnh sửa thông tin sinh viên 34

4.3.2.10. Sắp xếp sinh viên theo tên 34

4.3.2.11. Xuất danh sách ra file text 35

4.3.2.12. Đọc danh sách từ file text 35

4.3.2.13. Thêm sinh viên mới vào đầu danh sách 36

4.3.2.14. Xóa sinh viên ở cuối danh sách 37

4.3.2.15. Xóa sinh viên ở đầu danh sách 38

4.3.2.16. Xóa sinh viên theo tên 38

4.3.2.17. Tìm kiếm sinh viên theo tên 39

4.3.2.18. Tìm sinh viên có số điểm trên số điểm nhập vào 39

4.3.2.19. Tìm sinh viên có số điểm dưới số điểm nhập vào 40 **CHƯƠNG 5**: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 40

5.1. Kết luận 40

5.2. Hướng phát triển 40

# TÀI LIỆU THAM KHẢO 41

# PHỤ LỤC 41

**Mở Đầu**

## 1. Lý do chọn đề tài:

**Công nghệ thông tin** là một trong những ngành phát triển vượt bậc trong những năm gần đây. Ngày nay với sự phát triển nhanh chóng của xã hội thì công nghệ thông tin được ứng dụng rộng rãi ở hầu hết tất cả các lĩnh vực và ngày càng đóng vai trò quan trọng, trở thành một phần thiết yếu trong đời sống hằng ngày. Công nghệ thông tin là một ngành đòi hỏi ở người học một nền tảng kiến thức vững chắc, sự tư duy logic cao, hiểu biết sâu rộng trên nhiều lĩnh vực. Với chúng em hiện đang là những sinh viên công nghệ thông tin cần phải có sự đầu tư, không ngừng học hỏi để nâng cao kiến thức. Do đó để củng cố lại kiến thức đã học, đề tài mà em chọn để thực hiện đồ án là: **Xây dựng thư viện có các thao tác của một danh sách liên kết.**

## 2. Mục tiêu của đề tài.

* Củng cố lại kiến thức đã học về danh sách liên kết nói chung và danh sách liên kết đơn nói riêng.
* Rèn luyện kỹ năng lập trình trên ngôn ngữ C++.
* Ứng dụng lý thuyết đã học giải quyết bài toán ứng dụng cụ thể

## 3. Phạm vi nghiên cứu.

* Lý thuyết về danh sách liên kết đơn
* Bài toán quản lý sinh viên

## 4. Phương pháp nghiên cứu.

* Nghiên cứu tài liệu cấu trúc dữ liệu và giải thuật trên sách.
* Tìm kiếm và nghiên cứu trên mạng Internet.

**1. TỔNG QUAN ĐỀ TÀI**

### 1.1. Định nghĩa

Danh sách liên kết là danh sách mà các phần tử ( Node) liên kết với nhau nhờ vào vùng liên kết của chúng. Mỗi node bao gồm 2 phần: Phần Data dùng để chứa dữ liệu cần xử lý và phần liên kết dùng để liên kết tới các node khác.

### 1.2 .Phân loại.

Tùy cách liên kết giữa các phần tử, danh sách liên kết gồm có nhiều loại khác nhau:

* Danh sách liên kết đơn: mỗi phần tử liên kết với phần tử đứng sau nó trong danh sách.
* Danh sách liên kết đôi/kép: mỗi phần tử liên kết với các phần tử đứng trước và sau nó trong danh sách.
* Danh sách liên kết vòng: phần tử cuối danh sách liên kết với phần tử đầu danh sách.

### 2. CƠ SỞ LÝ THUYẾT

#### 2.1. Khái niệm.

Danh sách liên kết đơn là một cấu trúc lưu trữ trong đó các phần tử của danh sách không cố định, các phần tử trong danh sách không thể truy cập trực tiếp. Muốn truy cập bất kỳ phần tử nào đó trong danh sách phải xuất phát từ phần tử đầu tiên. Mỗi Node trong danh sách có thể được biểu diễn như sau.



*Hình 1.0. Node*

Trong đó:

* Data: trường chứa nội dung thông tin của phần tử
* Next: chứa địa chỉ của Node tiếp theo hay còn gọi là vùng liên kết.

Cú pháp:

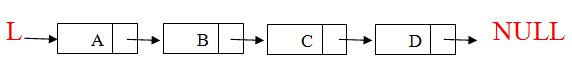
Struct Node {

Kiểu\_dữ\_liệu Data;

Node\*next;

}

Có thể hình dung danh sách liên kết đơn qua hình vẽ sau:



*Hình 1.1: Danh sách liên kết đơn*

#### 2.2.Thao tác trên 1 node.

* Khai báo biến con trỏ **p** là biến con trỏ kiểu Node: *Node \*p.*
* Cấp phát bộ nhớ cho Node mới – được trỏ bởi **p**:*p=new Node.*
* Truy cập vào trường **data**: *p->data.*
* Truy cập vào trường **next** :*p->next.*
* Hủy Node được trỏ bởi **p**: *delete (p).*

#### 2.3. Các thao tác trên danh sách liên kết đơn

Thao tác trên danh sách liên kết khác với thao tác trên mảng, khi cần thêm Node vào danh sách liên kết thì ta xin cấp phát bộ nhớ cho Node và nối Node đó vào danh sách và khi không sử dụng thì ta giải phóng nó khỏi danh sách. Truy cập trên danh sách cũng khác so với trên mảng. Nếu ở mảng ta truy cập trực tiếp thông qua chỉ số mảng (thông qua địa chỉ ô nhớ) thì ở danh sách liên kết ta phải truy cập một cách tuần tự. Khi làm việc trên danh sách liên kết đơn cũng tương tự như vậy nên ta cần chú ý những điểm sau:

* Danh sách luôn có con trỏ đầu danh sách: con trỏ L
* Danh sách luôn có giá trị báo kết thúc danh sách: NULL.
* Trường **next** của mỗi Node chỉ chứa địa chỉ của Node sau nó, trừ Node cuối.
* Trường **next** của Node cuối cùng chứa giá trị NULL.
* Không tách danh sách thành 2 danh sách con nếu danh sách phần sau chưa có con trỏ tới.
* Khởi tạo danh sách rỗng: L = NULL

##### **2.4. Duyệt danh sách liên kết đơn**

Duyệt lần lượt qua từng Node của danh sách.

*Mục\_đích(Node \*L)*

{

*Node \*p=L;* //cho con trỏ **p** trỏ vào đầu danh sách L.

*while(!điều kiện dừng)*

*{*

*[xử lý p->data];*//xuất thông tin ra màn hình, thực hiện các phép toán…

*p= p->Next;* //cho con trỏ **p** trỏ tới Node kế tiếp.

}

##### **2.5. Chèn một phần tử vào danh sách**

Để chèn một phần tử X vào danh sách ta cấp phát bộ nhớ cho 1 Node mới được trỏ bởi **q** và gán giá trị X cho trường **data**

Node \*q;

q=new Node;

q->data=X;

Chèn một Node mới vào danh sách có 4 trường hợp:

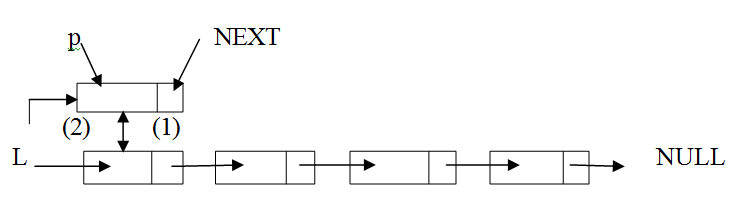
* **Trường hợp 1: Chèn Node được trỏ bởi q vào đầu sách**

Bước 1: cho vùng liên kết của **q** chứa địa chỉ Node đầu danh sách.

*q->next=L;(1)*

Bước 2: cho L trỏ tới **q** : cập nhập lại con trỏ đầu danh sách.

L = q; (2)

**

*Hình 1.2: Chèn node vào đầu danh sách*

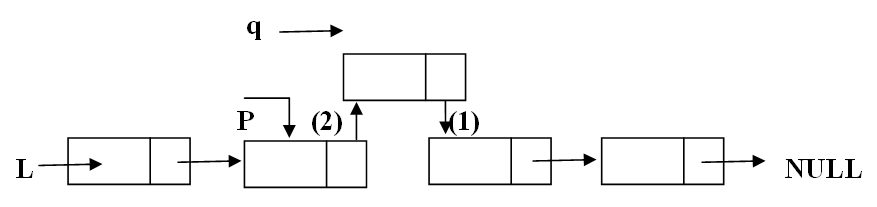
* **Trường hợp 2: Chèn Node được trỏ bởi q vào sau Node trỏ bởi p trong danh sách**

Bước 1: Cho vùng liên kết của Node được trỏ bởi q chứa địa chỉ Node ngay sau Node được trỏ bởi p.

q->next=p->next; (1)

Bước 2: Cho vùng liên kết của Node được trỏ bởi q chứa địa chỉ của Node được trỏ bởi p.

p->next=q; (2)

****

*Hình 1.3: Chèn Node vào Node được trỏ*

* **Trường hợp 3: Chèn Node được trỏ bởi q vào cuối danh sách**

Bước 1: Nếu danh sách rỗng thì danh sách hiện tại có 1 phần tử chính là **q** vừa mới chèn vào, return.

Bước 2: Ngược lại nếu danh sách không rỗng

* Cho con trỏ **p** chứa địa chỉ Node đầu danh sách:

Node \*p=L*;*

* Duyệt danh sách từ đầu cho tới Node cuối cùng

while(p->next!=NULL) p=p->next;

* Cho phần liên kết của Node được trỏ bởi **p** chứa địa chỉ của **q**

 p->next=q;

* Cập nhập lại Node cuối cùng bây giờ chính là Node **q** vừa mới tạo ra.

q->next=NULL;

* **Trường hợp 4: Chèn Node được trỏ bởi q vào trước Node được trỏ bởi p**

Bước 1: Nếu **p** trỏ đầu danh sách, thực hiện chèn đầu danh sách(trường hợp1), return.

Bước 2: Ngược lại cho con trỏ **t** chứa địa chỉ Node trước Node được trỏ bởi **p** trong danh sách: *Node \*t;*

*t->next=p;*

Chèn Node được trỏ bởi **q** vào sau Node được trỏ bởi **t (**trường hợp 2**).**

##### **2.6. Xóa một phần tử ra khỏi danh sách:** có 2 trường hợp xảy ra

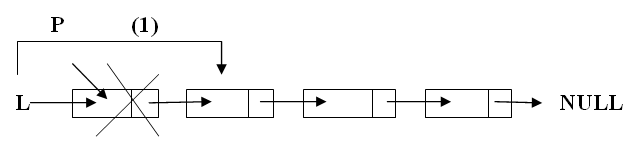
* **Trường hợp 1: Thực hiện xóa đầu danh sách nếu p chứa địa chỉ Node đầu danh sách (L)**

Bước 1: Cho con trỏ đầu danh sách L trỏ vào Node sau nó.

L=L->next; (1)

Bước 2: Giải phóng Node p ra khỏi danh sách.

Delete(p);



*Hình 1.4: Xóa một phần tử TH1*

* **Trường hợp 2: Con trỏ p chứa địa chỉ Node bất kỳ nhưng không phải Node đầu**

Bước 1: Cho con trỏ **q** trỏ đầu danh sách **L** , dùng vòng lặp cho nó trỏ tới địa chỉ Node trước Node được trỏ bởi **p** trong danh sách.

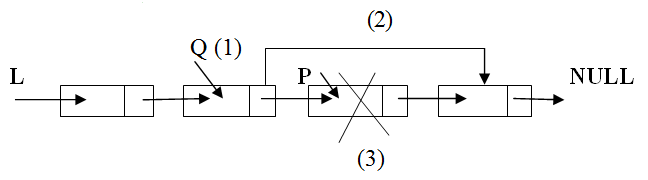
q=L; while(q->next!=p) q=q->next; (1)

Bước 2: Gán địa chỉ của Node ngay sau Node được trỏ bởi **p** vào ô nhớ  **next** của Node được trỏ bởi **q**

q->Next=p->next; (2)

Bước 3: Giải phóng Node được trỏ bởi **p**

delete(p);(3)

****

*Hình 1.5: Xóa 1 phần tử TH2*

##### **2.7. Tìm kiếm phần tử trong danh sách**

Bước 1: Cho **p** trỏ vào đầu danh sách L và khởi tạo biến found=0

Node \*p=L; int found=0;

Bước 2: Thực hiện vòng lặp để duyệt p chạy từ đầu danh sách tới cuối danh sách, tại mỗi Node nếu trường data của Node được trỏ bởi p =X thì cho found=1 để dừng vòng lặp. Ngược lại nếu p!=X thì cho p trỏ tới vị trí Node ngay sau Node được trỏ bởi p.

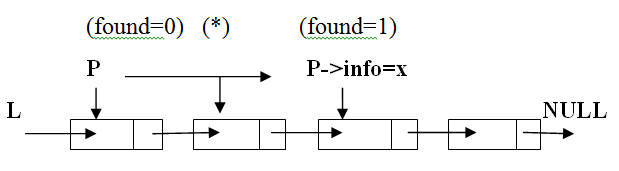
while(p != NULL && found == 0)

if(p->data = X) found=1;

else

  p=p->next; (\*)

Bước 3: Return **p**



*Hình 1.6: Tìm kiếm phần tử*

##### **2.8. Đảo các phần tử trong danh sách**

Bước 1: Khởi tạo 3 biến con trỏ **p**, **t**, **q.** Trong đó **p** trỏ vào đầu danh sách L, **t= NULL** (không trỏ vào địa chỉ nào cả), **q** trỏ vào địa chỉ Node được trỏ bởi **p.**

Node \*p=L,\*t=NULL,\*q=p ;

Bước 2: Dùng vòng lặp duyệt **p** chạy từ đầu tới cuối danh sách. Trong khi thực hiện vòng lặp

* Cho vùng liên kết của Node được trỏ bởi **q** chứa địa chỉ của **t**

q->next=t;

* Cho **t** trỏ đến vị trí Node được trỏ bởi **q** (cập nhập lại t).

t=q;

Cụ thể như sau:

While(p!=NULL)

{

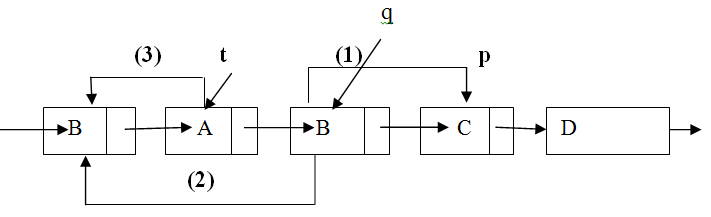
 q=p; p=p->next; (1)

q->next=t ; (2)

t=q; (3) }

Bước 3 :Cho L trỏ đến vị trí Node được trỏ bởi **t** (cập nhập lại L)

L=t

****

*Hình 1.7: Đảo các phần tử*

CHƯƠNG 3: TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ BÀI TOÁN

## 3.1. Đặc tả bài toán.

Quản lý sinh viên là công việc hằng ngày của phòng quản lý học sinh- sinh viên. Công việc quản lý sinh viên đòi hỏi tính tỉ mỉ, cẩn thận trong từng khâu ghi chép các thông tin cá nhân của sinh viên: mã số sinh viên, tên sinh viên, ngày tháng năm sinh, khóa, lớp….cũng như công việc thống kê kết quả học tập cần rõ ràng và chính xác. Trước đây công nghệ thông tin chưa phát triển mạnh mẽ, các công việc được xử lý thủ công,chủ yếu là ghi chép bằng bút, sổ sách chính vì vậy rất tốn công sức và khá nhiều thời gian. Ngày nay khi mà khoa học kỹ thuật phát triển, đặc biệt là sự bùng nổ công nghệ thông tin thì việc quản lý sinh viên sẽ dễ dàng hơn nhiều. Xuất phát từ nhu cầu đó mà bài toán **Quản lý sinh viên** ra đời. Yêu cầu của bài toán là tạo ra chương trình có thể thực hiện các thao tác quản lý sinh viên một cách dễ dàng, tiện lợi dựa trên sự trợ giúp của máy tính. Mọi công việc phải được thao tác trên một vùng dữ liệu chung để đảm bảo việc đồng bộ với nhau trong khâu quản lý.

## 3.2. Yêu cầu hệ thống.

Quản lý điểm sinh viên là chương trình quản lý hồ sơ, điểm học tập của sinh viên trong quá trình theo học tại trường. Chương trình có thể thực hiện các công việc thêm mới sinh viên, tìm kiếm sinh viên theo một điều kiện nào đó, xóa sinh viên, liệt kê danh sách sinh viên….

Chương trình được viết bằng ngôn ngữ C++ và dựa trên cấu trúc lưu trữ của danh sách liên kết đơn .

## 3.3. Phân tích thiết kế chương trình.

### 3.3.1. Cấu trúc dữ liệu chương trình.

Để giúp cho việc quản lý được chặt chẽ, dễ dàng, chương trình quản lý điểm sinh viên sẽ có các trường hợp sau:

* Masv (mã sinh viên) có kiểu dữ liệu char, với độ dài 20 ký tự, mỗi sinh viên có một mã số riêng, không trùng lặp.
* Hoten (họ tên) có kiểu dữ liệu nvarchar, với độ dài 40 ký tự, họ tên sinh viên có thể trùng nhau.
* Lop (lớp) có kiểu dữ liệu char, với độ dài 15 ký tự, một sinh viên chỉ được xếp vào một lớp với một mã lớp tương ứng.
* Toan (điểm môn toán) có kiểu dữ liệu int, đây là kết quả học tập môn toán của sinh viên được cập nhập từ các giảng viên dạy.
* Ly (điểm môn lý) có kiểu dữ liệu int, đây là kết quả học tập môn lý của sinh viên được cập nhập từ các giảng viên dạy.
* Hoa (điểm môn hoá) có kiểu dữ liệu int, kết quả học tập môn hóa của sinh viên được cập nhập từ các giảng viên dạy.
* Diemtb (điểm trung bình) có kiểu dữ liệu float, điểm trung bình được tính bằng cách lấy trung bình cộng điểm thi các môn học của sinh viên.

struct nut

{

string masv;

string hoten;

string lop;

float toan;

float ly;

float hoa;

float diemtb;

string xeploai;

nut \*next;

};

### 3.3.2. Các chức năng của chương trình.

#### 3.3.2.1. Nhập danh sách sinh viên.

* Nhập vào mã số sinh viên.
* Nhập thông tin chi tiết cho từng sinh viên.

#### 3.3.2.2. Hiển thị danh sách sinh viên.

* Duyệt qua danh sách sinh viên.
* In thông tin sinh viên ra màn hình.

#### 2.3.2.3. Tìm kiếm sinh viên theo mã sinh viên.

* Duyệt qua danh sách.
* Tìm kiếm theo mã sinh viên mà người dùng yêu cầu.
* Hiển thị sinh viên vừa tìm ra màn hình.

#### 3.3.2.4.Xóa sinh viên theo mã số sinh viên.

* Duyệt qua danh sách.
* Tìm kiếm sinh viên theo mã sinh viên mà người dùng yêu cầu
* Xóa sinh viên vừa tìm được.

#### 3.3.2.5. Bổ sung sinh viên mới vào danh sách.

* Nhập sinh viên cần bổ sung vào danh sách.
* Cập nhập sinh viên vào trong danh sách.
* Tăng số lượng sinh viên trong danh sách lên.

#### 3.3.2.6. Hiển thị sinh viên có điểm trung bình cao nhất.

* Nhập điểm từng môn của từng sinh viên vào danh sách.
* Tính điểm trung bình của từng sinh viên.
* Tìm kiếm sinh viên có điểm trung bình cao nhất.
* Hiển thị sinh viên có điểm trung bình cao nhất ra màn hình.

#### 3.3.2.7. Sắp xếp danh sách sinh viên tăng dần theo trường điểm trung bình

* Duyệt qua danh sách điểm trung bình của từng sinh viên.
* Sắp xếp sinh viên tăng dần theo điểm trung bình của từng sinh viên.
* Hiển thị danh sách sinh viên vừa sắp xếp ra màn hình.

## CHƯƠNG 4: CÀI ĐẶT CHƯƠNG TRÌNH

### 4.1. Định nghĩa cấu trúc cho chương trình quản lý sinh viên

* Void xuat\_file(node first, string path)
* void insert(string ma, string ten, string l, float toan, float ly, float hoa, float dtb, node &first)
* void doc\_file(node &first, string path)
* void themThongTin(string &ma, string &ten, string &lop, float &toan, float &ly, float &hoa, float &dtb, node &first)Void xuat(node first)
* void nhap(node &first)
* void xuat(node first)
* Void xoa(node&first)
* void tim\_ma(node first)
* void themVaoDauDanhSach(node &first)
* void themsv(node &first)
* Void dtb\_max(node first)
* Void hoanvi(float&a,float&b)
* void hoanvi1(string &a, string &b)
* void sapxep(node first)
* void sapxepTheoTen(node first)
* void sua(node first)
* void xoaDauDanhSach(node &first)
* void xoaCuoiDanhSach(node &first)
* void xoaTheoTen(node &first)
* void timTheoTen(node first)
* void inDiemTren(node first)
* void inDiemDuoi(node first)
* void menu()
* int main()

**4.2. Ngôn ngữ cài đặt:**

* C++

**4.3. Kết quả**

### 4.3.1. Khởi tạo danh sách sinh viên

| **void** **nhap**(node &first) // Ham nhap cua ds don  {  system("cls");  **char** ch;  first = NULL;  string ma, ten, lop;  **float** toan, ly, hoa, dtb;  **do**  {  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, first);  insert(ma, ten, lop, toan, ly, hoa, dtb, first);  cout << "**\n** PRESS ESC TO EXIT**\n** PRESS ENTER TO CONTINUE";  cin.sync();  ch = \_getch();  } **while** (ch != **27**);  } |
| --- |

### 4.3.1.2. Hiển thị danh sách sinh viên

| **void** **xuat**(node first) // xuat xuoi  {  **char** ch;  system("cls");  node p;  p = first;  **while** (p != NULL)  {  cout << "**\n**Student's ID : " << p->masv;  cout << "**\n**Full Name : " << p->hoten;  cout << "**\n**Class : " << p->lop;  cout << "**\n**Math Score : " << p->toan;  cout << "**\n**Physics Score : " << p->ly;  cout << "**\n**Chemistry Score : " << p->hoa;  cout << "**\n**Average Score : " << p->diemtb << "**\n**";  p = p->next;  }  cin.sync();  ch = \_getch();  } |
| --- |

### **4.3.**1.3**. Tìm kiếm sinh viên theo mã sinh viên**

| **void** **tim\_ma**(node first) // tim kiem sinh vien theo ma  {  system("cls");  node p;  p = first;  string ma;  **char** ch;  cout << "**\n**Input Student's ID you are looking for : ";  cin >> ma;  **while** (p != NULL)  {  **if** (p->masv == ma)  {  cout << "**\n**Student's ID : " << p->masv;  cout << "**\n**Full Name : " << p->hoten;  cout << "**\n**Class : " << p->lop;  cout << "**\n**Math Score : " << p->toan;  cout << "**\n**Physics Score : " << p->ly;  cout << "**\n**Chemistry Score : " << p->hoa;  cout << "**\n**Average Score : " << p->diemtb << "**\n**";  cin.sync();  ch = \_getch();  **return**;  }  p = p->next;  }  cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  } |
| --- |

### 4.3.1.4. Xóa sinh viên theo mã sinh viên.

| **void** **xoa**(node &first) // xoa sinh vien theo ma  {  system("cls");  node p, q;  p = first;  q = first;  string ma;  cout << "**\n**Input Student's ID to delete : ";  cin >> ma;  **while** (p != NULL && p->masv != ma)  p = p->next;  **if** (p == first)  first = first->next;  **else**  {  **if** (p == NULL)  {  cout << "**\n**Can's find student to delete!";  }  **else**  {  **while** (q != NULL && q->next != p)  q = q->next;  q->next = p->next;  }  }  system("cls");  xuat(first);  \_getch();  } |
| --- |

### 4.3.1.5. Bổ sung sinh viên mới vào danh sách.

| **void** **themsv**(node &first) // bo sung sv moi vao danh sach  {  system("cls");  **char** ch;  string ma, ten, lop;  **float** toan, ly, hoa, dtb;  **do**  {  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, first);  insert(ma, ten, lop, toan, ly, hoa, dtb, first);  cout << "**\n** PRESS ESC TO EXIT**\n** PRESS ENTER TO CONTINUE";  cin.sync();  cin.sync();  ch = \_getch();  } **while** (ch != **27**);  } |
| --- |

### 4.3.1.6. In sinh viên có điểm trung bình lớn nhất.

| **void** **dtb\_max**(node first) // in ra sinh vien co diem trung binh max  {  **char** ch;  system("cls");  **if** (first == NULL)  {  cout << "THERE IS NO STUDENT IN LIST";  cin.sync();  ch = \_getch();  **return**;  }  node p = first, q = first, l = first;  **float** max = p->diemtb;  p = p->next;  **while** (p != NULL)  {  **if** (p->diemtb > max)  {  max = p->diemtb;  q = p;  }  p = p->next;  }  **while** (l != NULL)  {  **if** (l == q)  {  cout << "**\n**Student's ID : " << l->masv;  cout << "**\n**Full Name : " << l->hoten;  cout << "**\n**Class : " << l->lop;  cout << "**\n**Math Score : " << l->toan;  cout << "**\n**Physics Score : " << l->ly;  cout << "**\n**Chemistry Score : " << l->hoa;  cout << "**\n**Average Score : " << l->diemtb << "**\n**";  **break**;  }  l = l->next;  }  cin.sync();  ch = \_getch();  } |
| --- |

### 4.3.1.7. Sắp xếp tăng dần theo điểm trung bình.

| **void** **sapxep**(node first) // sx sv tang dan theo dtb  {  **char** ch;  system("cls");  node p = first, q;  **while** (p != NULL)  {  q = p->next;  **while** (q != NULL)  {  **if** (p->diemtb > q->diemtb)  {  hoanvi1(p->masv, q->masv);  hoanvi1(p->hoten, q->hoten);  hoanvi1(p->lop, q->lop);  hoanvi(p->toan, q->toan);  hoanvi(p->ly, q->ly);  hoanvi(p->hoa, q->hoa);  hoanvi(p->diemtb, q->diemtb);  }  q = q->next;  }  p = p->next;  }  xuat(first);  cin.sync();  ch = \_getch();  } |
| --- |

## 4.3.1.8. Chỉnh sửa thông tin sinh viên

| **void** **sua**(node first) // sua sinh vien  {  **char** ch;  system("cls");  node p;  p = first;  string ma;  cout << "**\n**Input Student's ID you want to justify : ";  getline(cin, ma);  **while** (p != NULL)  {  **if** (p->masv == ma)  {  system("cls");  **char** ch;  string ma, ten, lop;  **float** toan, ly, hoa, dtb;  fflush(stdin);  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, p);  p->masv = ma;  p->hoten = ten;  p->lop = lop;  p->toan = toan;  p->ly = ly;  p->hoa = hoa;  p->diemtb = dtb;  cin.sync();  ch = \_getch();  **return**;  }  p = p->next;  }  printf("CANNOT FILE THIS ID");  cin.sync();  ch = \_getch();  **return**;  } |
| --- |

## 4.3.1.9. Sắp xếp sinh viên theo tên

| **void** **sapxepTheoTen**(node first) // sx sv tang dan theo ten  {  **char** ch;  system("cls");  node p = first, q;  **while** (p != NULL)  {  q = p->next;  **while** (q != NULL)  {  **if** (p->hoten > q->hoten)  {  hoanvi1(p->masv, q->masv);  hoanvi1(p->hoten, q->hoten);  hoanvi1(p->lop, q->lop);  hoanvi(p->toan, q->toan);  hoanvi(p->ly, q->ly);  hoanvi(p->hoa, q->hoa);  hoanvi(p->diemtb, q->diemtb);  }  q = q->next;  }  p = p->next;  }  xuat(first);  cin.sync();  ch = \_getch();  } |
| --- |

## 4.3.1.10. Xuất danh sách ra file text

| **void** **xuat\_file**(node first, string path)  {  ofstream o;  o.open(path.c\_str());  node p;  p = first;  o << soLuongSinhVien(first) << "**\n**";  **while** (p != NULL)  {  o << p->masv << "**\n**";  o << p->hoten << "**\n**";  o << p->lop << "**\n**";  o << p->toan << "**\n**";  o << p->ly << "**\n**";  o << p->hoa << "**\n**";  o << p->diemtb << "**\n**";  p = p->next;  }  } |
| --- |

## 4.3.1.11. Đọc danh sách từ file text

| **void** **doc\_file**(node &first, string path)  {  ifstream i;  i.open(path.c\_str());  string ma;  string ten;  string l;  **float** toan;  **float** ly;  **float** hoa;  **float** dtb;  **int** n;  i >> n;  string bf;  getline(i, bf);  **while** (n--)  {  getline(i, ma);  getline(i, ten);  getline(i, l);  i >> toan;  i >> ly;  i >> hoa;  i >> dtb;  getline(i, bf);  insert(ma, ten, l, toan, ly, hoa, dtb, first);  }  } |
| --- |

## 4.3.1.12. Thêm sinh viên mới vào đầu danh sách

| **void** **themVaoDauDanhSach**(node &first)  {  system("cls");  **char** ch;  string ma, ten, lop;  **float** toan, ly, hoa, dtb;  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, first);  node p;  p = **new** (nut);  p->masv = ma;  p->hoten = ten;  p->lop = lop;  p->toan = toan;  p->ly = ly;  p->hoa = hoa;  p->diemtb = dtb;  p->next = first;  first = p;  cin.sync();  cin.sync();  ch = \_getch();  ;  } |
| --- |

## 4.3.1.13. Xóa sinh viên ở cuối danh sách

| **void** **xoaCuoiDanhSach**(node &first)  {  **char** ch;  node p;  p = first;  **if** (first == NULL)  {  system("cls");  cout << "**\n**THERE IS NO STUDENT IN LIST**\n**";  cin.sync();  ch = \_getch();  **return**;  }  **if** (p->next == NULL)  {  first = NULL;  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  **return**;  }  **while** (p->next->next != NULL)  {  p = p->next;  }  p->next = NULL;  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  } |
| --- |

## 4.3.1.14. Xóa sinh viên ở đầu danh sách

| **void** **xoaDauDanhSach**(node &first)  {  **char** ch;  **if** (first == NULL)  {  system("cls");  cout << "**\n**THERE IS NO STUDENT IN LIST**\n**";  cin.sync();  ch = \_getch();  **return**;  }  first = first->next;  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  } |
| --- |

## 4.3.1.15. Xóa sinh viên theo tên

| **void** **xoaTheoTen**(node &first) // xoa sinh vien theo ma  {  **char** ch;  system("cls");  node p, q;  p = first;  q = first;  string name;  cout << "**\n**Input Student's name to delete : ";  getline(cin, name);  **while** (p != NULL && p->hoten != name)  p = p->next;  **if** (p == first)  first = first->next;  **else**  {  **if** (p == NULL)  {  cout << "**\n**Can't find student to delete!";  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  }  **else**  {  **while** (q != NULL && q->next != p)  q = q->next;  q->next = p->next;  }  }  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  } |
| --- |

## 4.3.1.16. Tìm kiếm sinh viên theo tên

| **void** **timTheoTen**(node first)  {  **char** ch;  system("cls");  node p;  p = first;  string name;  cout << "**\n**Input Student's name you are looking for : ";  getline(cin, name);  **while** (p != NULL)  {  **if** (p->hoten == name)  {  cout << "**\n**Student's ID : " << p->masv;  cout << "**\n**Full Name : " << p->hoten;  cout << "**\n**Class : " << p->lop;  cout << "**\n**Math Score : " << p->toan;  cout << "**\n**Physics Score : " << p->ly;  cout << "**\n**Chemistry Score : " << p->hoa;  cout << "**\n**Average Score : " << p->diemtb << "**\n**";  // cin.sync();  ch = \_getch();  // return;  }  p = p->next;  }  // cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  } |
| --- |

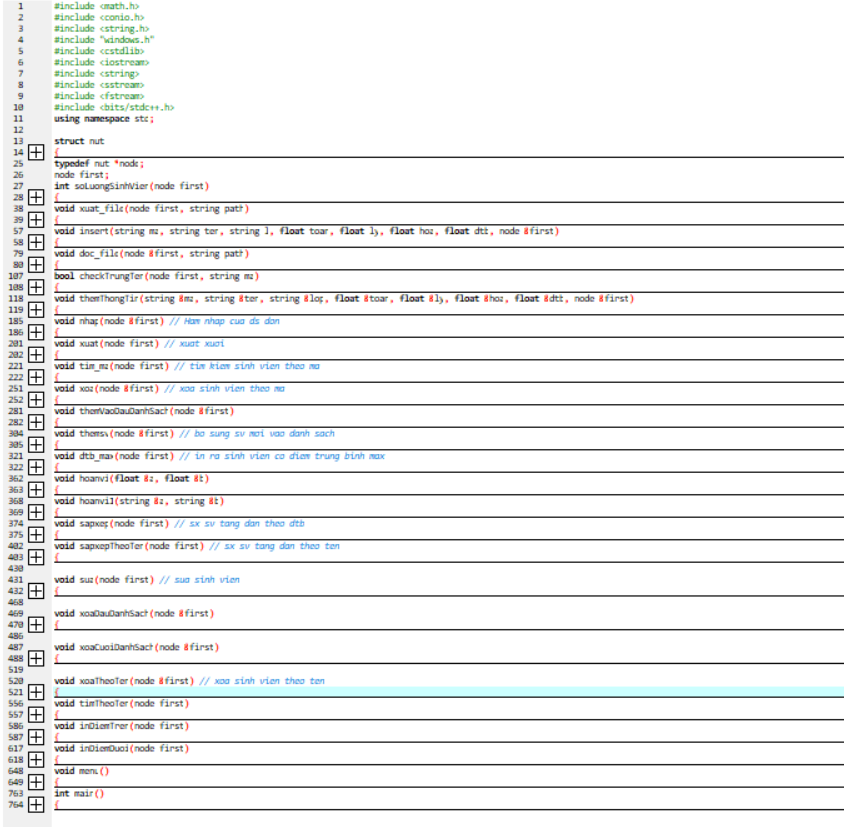
## 4.3.1.17. Tìm sinh viên có số điểm trên số điểm nhập vào

| **void** **inDiemTren**(node first)  {  **char** ch;  system("cls");  node p;  p = first;  **int** score;  cout << "**\n**Input Student's average score : ";  cin >> score;  cout << "**\n**Nhung hoc sinh co diem lon hon hoac bang " << score << " la: ";  **while** (p != NULL)  {  **if** (p->diemtb >= score)  {  cout << "**\n**Student's ID : " << p->masv;  cout << "**\n**Full Name : " << p->hoten;  cout << "**\n**Class : " << p->lop;  cout << "**\n**Math Score : " << p->toan;  cout << "**\n**Physics Score : " << p->ly;  cout << "**\n**Chemistry Score : " << p->hoa;  cout << "**\n**Average Score : " << p->diemtb << "**\n**";  // cin.sync();  // ch = \_getch();  // return;  }  p = p->next;  }  // cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  } |
| --- |

## 4.3.1.18. Tìm sinh viên có số điểm dưới số điểm nhập vào

| **void** **inDiemDuoi**(node first)  {  **char** ch;  system("cls");  node p;  p = first;  **int** score;  cout << "**\n**Input Student's average score : ";  cin >> score;  cout << "**\n**Nhung hoc sinh co diem be hon bang " << score << " la: ";  **while** (p != NULL)  {  **if** (p->diemtb <= score)  {  cout << "**\n**Student's ID : " << p->masv;  cout << "**\n**Full Name : " << p->hoten;  cout << "**\n**Class : " << p->lop;  cout << "**\n**Math Score : " << p->toan;  cout << "**\n**Physics Score : " << p->ly;  cout << "**\n**Chemistry Score : " << p->hoa;  cout << "**\n**Average Score : " << p->diemtb << "**\n**";  // cin.sync();  // ch = \_getch();  // return;  }  p = p->next;  }  // cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  } |
| --- |

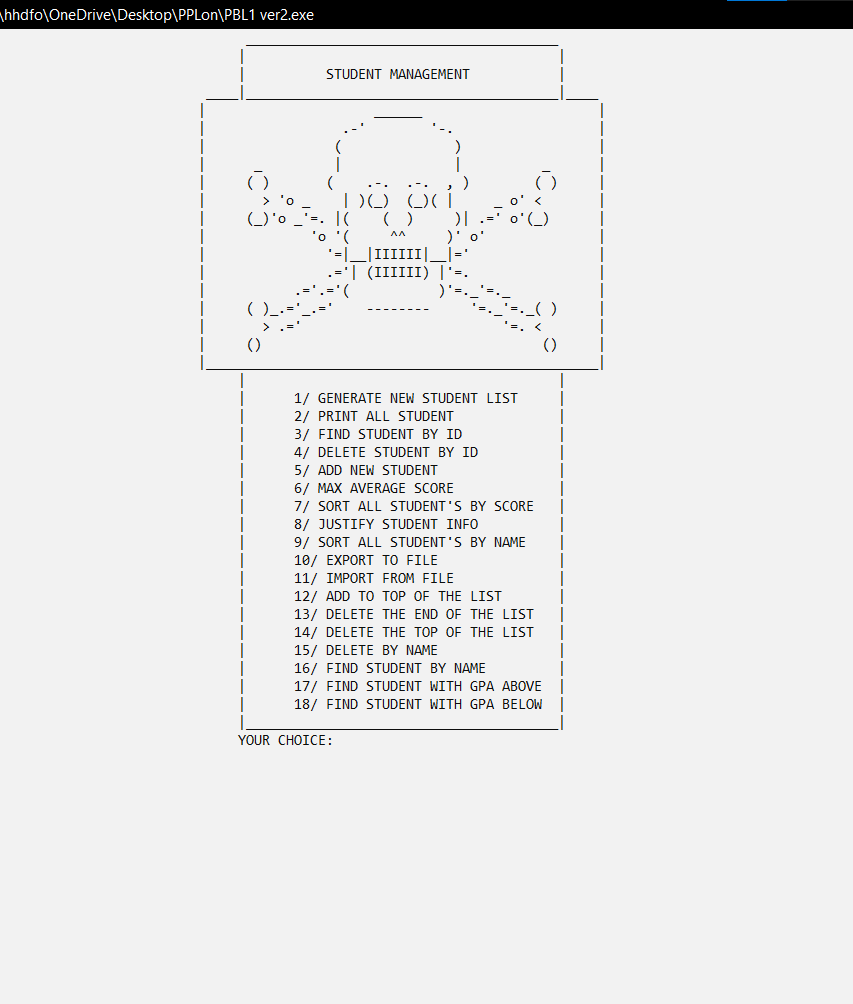
## 4.3.2. Chương trình quản lý sinh viên



*Hình 4.3.1 Tổng quan chương trình*

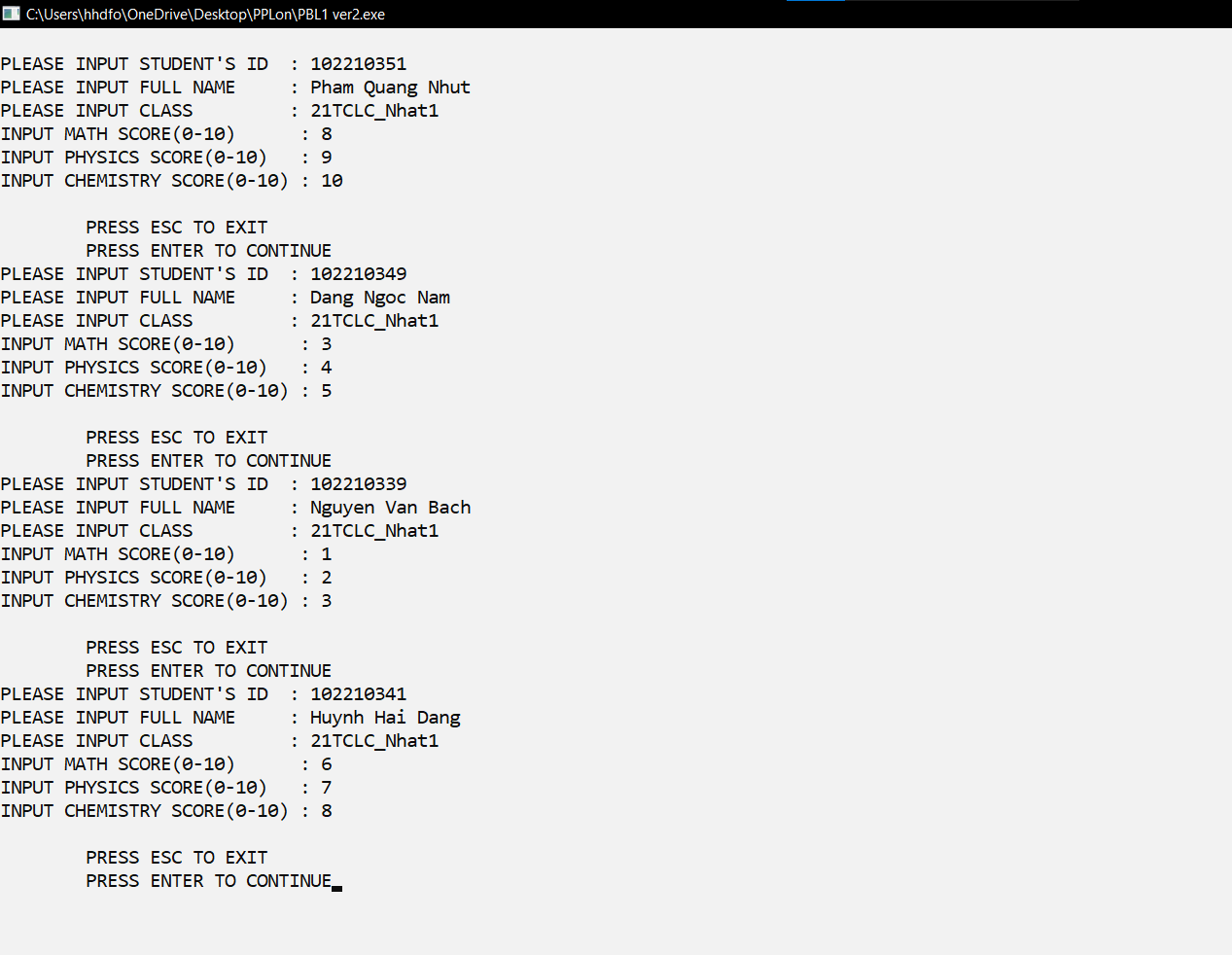
## 4.3.2.0. Màn hình kết quả

### 4.3.2.1. Menu chính của chương trình.

****

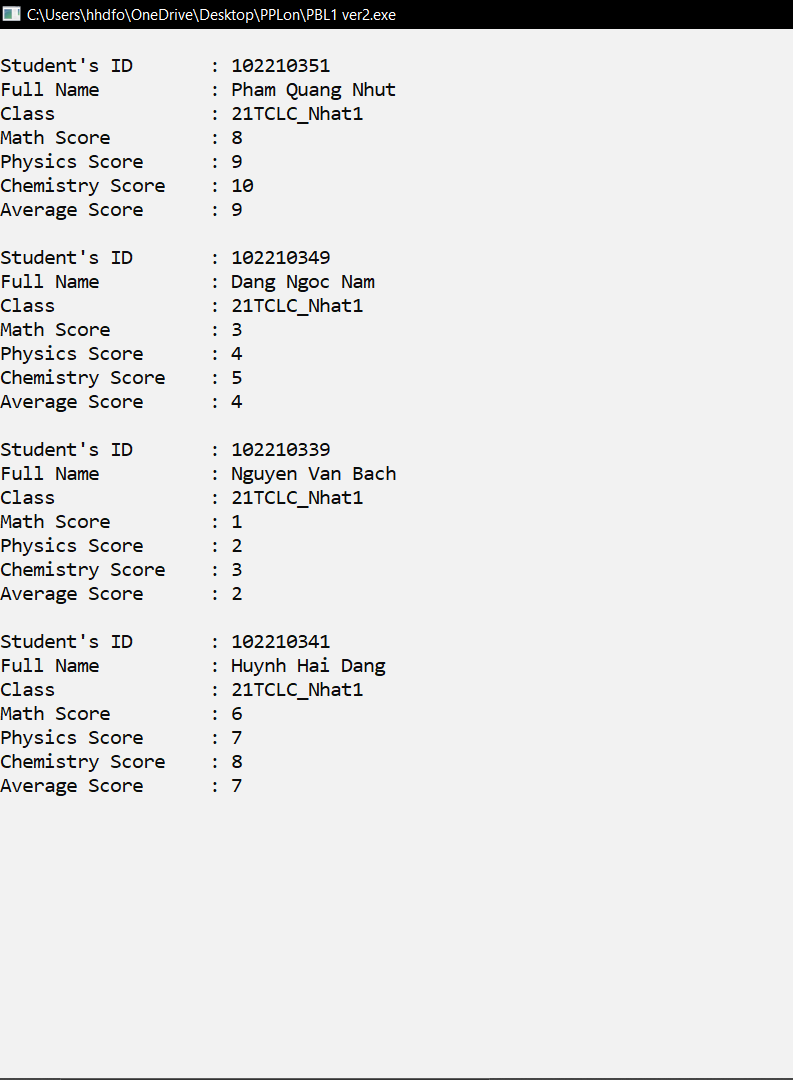
*Hình 4.3.2.1. Menu chính*

### 4.3.2.2. Khởi tạo danh sách sinh viên



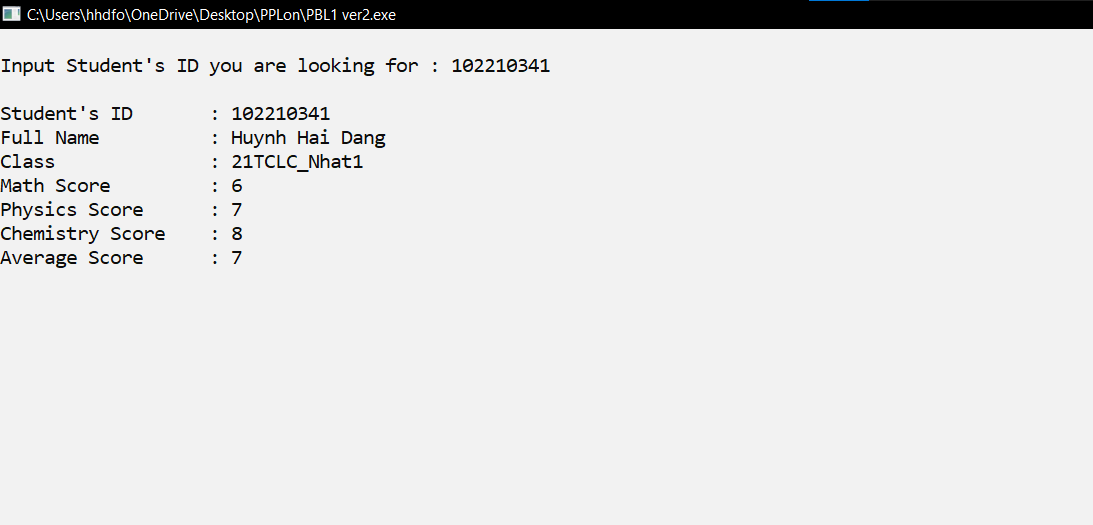
*Hình 4.3.2.2. Khởi tạo danh sách sinh viên*

### 4.3.2.3. Hiển thị danh sách sinh viên



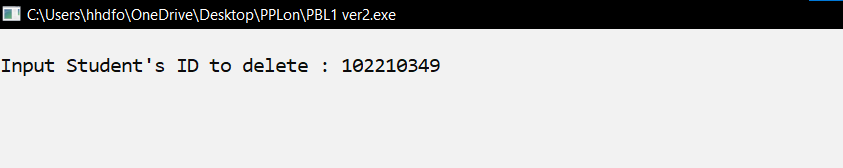
*Hình 4.3.2.3. In ra màn hình danh sách sinh viên*

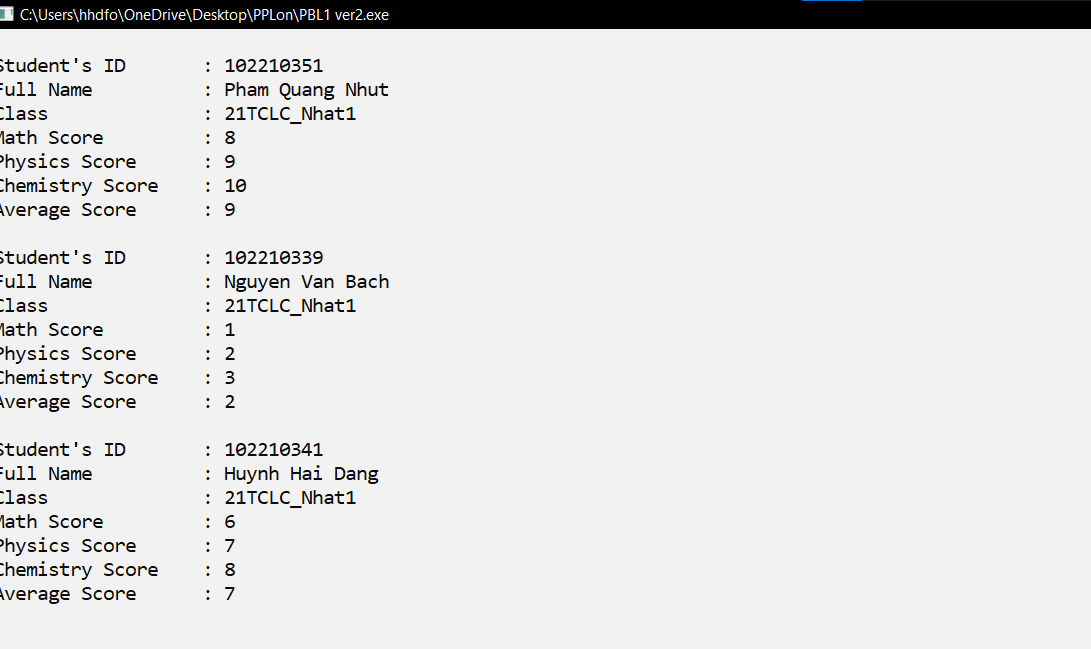
### 4.3.2.4. Tìm kiếm sinh viên theo mã sinh viên

****

*Hình 4.3.2.4. Tìm kiếm sinh viên*

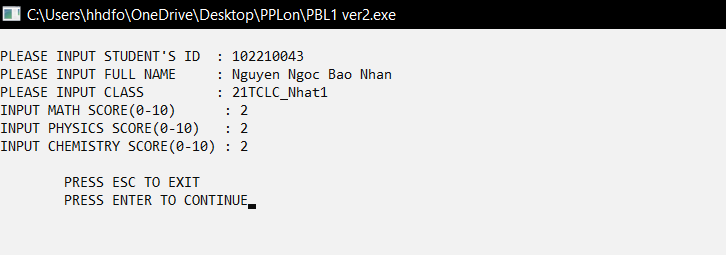
### 4.3.2.5. Xóa sinh viên theo mã sinh viên.

****



*Hình 4.3.2.5. Xóa sinh viên*

### 4.3.2.6. Bổ sung sinh viên mới vào danh sách.



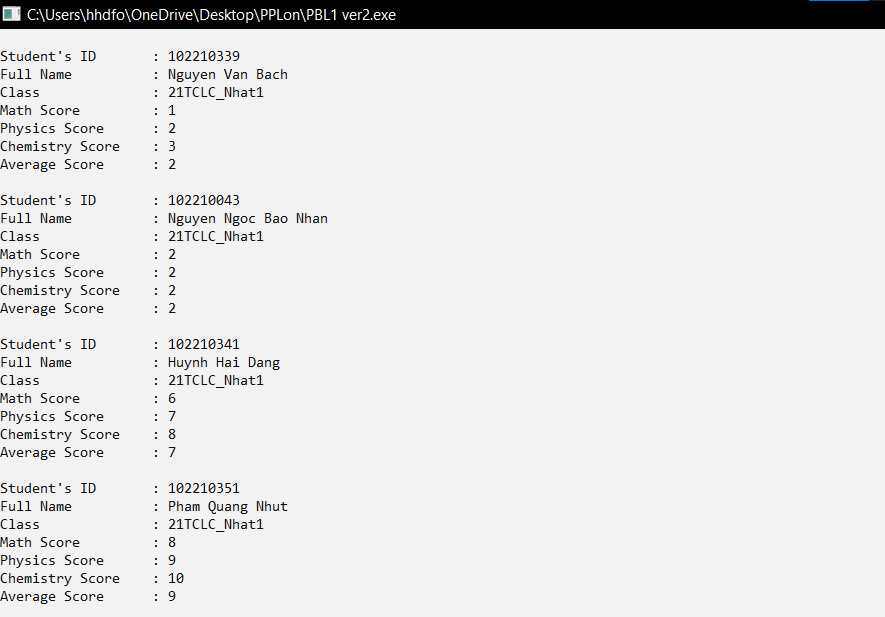
*Hình 4.3.2.6. Thêm sinh viên*

### 4.3.2.7. In sinh viên có điểm trung bình lớn nhất.

### 

*Hình 4.3.2.7. In sinh viên có điểm trung bình lớn nhất*

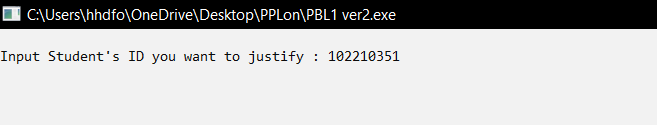
### 4.3.2.8. Sắp xếp danh sách sinh viên tăng dần theo điểm trung bình.

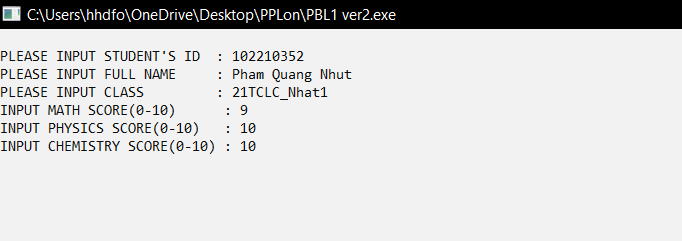


*Hình 4.3.2.8. Sắp xếp sinh viên tăng dần theo điểm trung bình*

### 

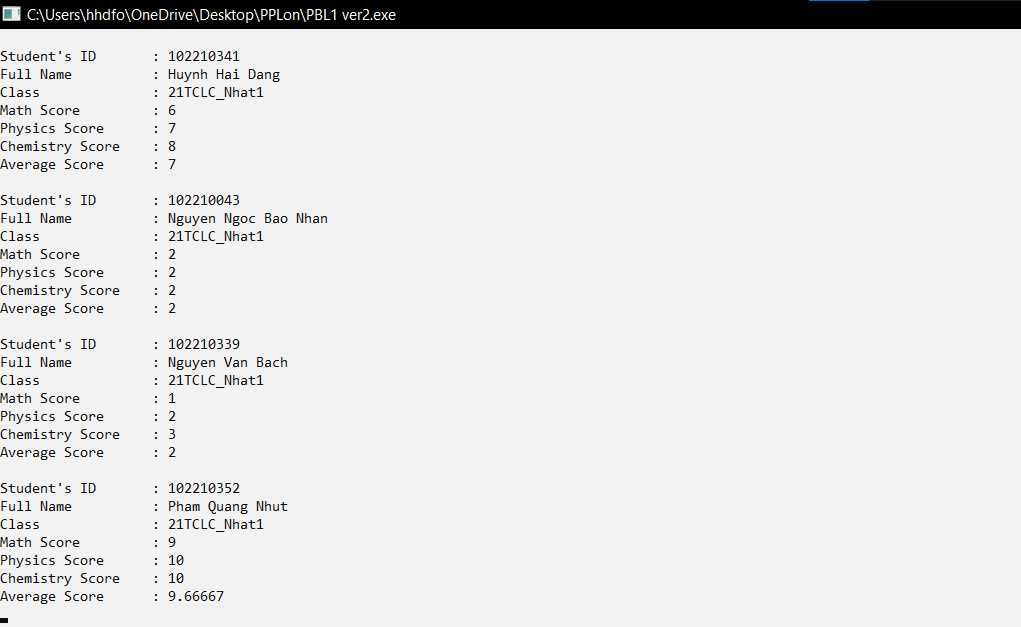
### 4.3.2.9. Chỉnh sửa thông tin sinh viên





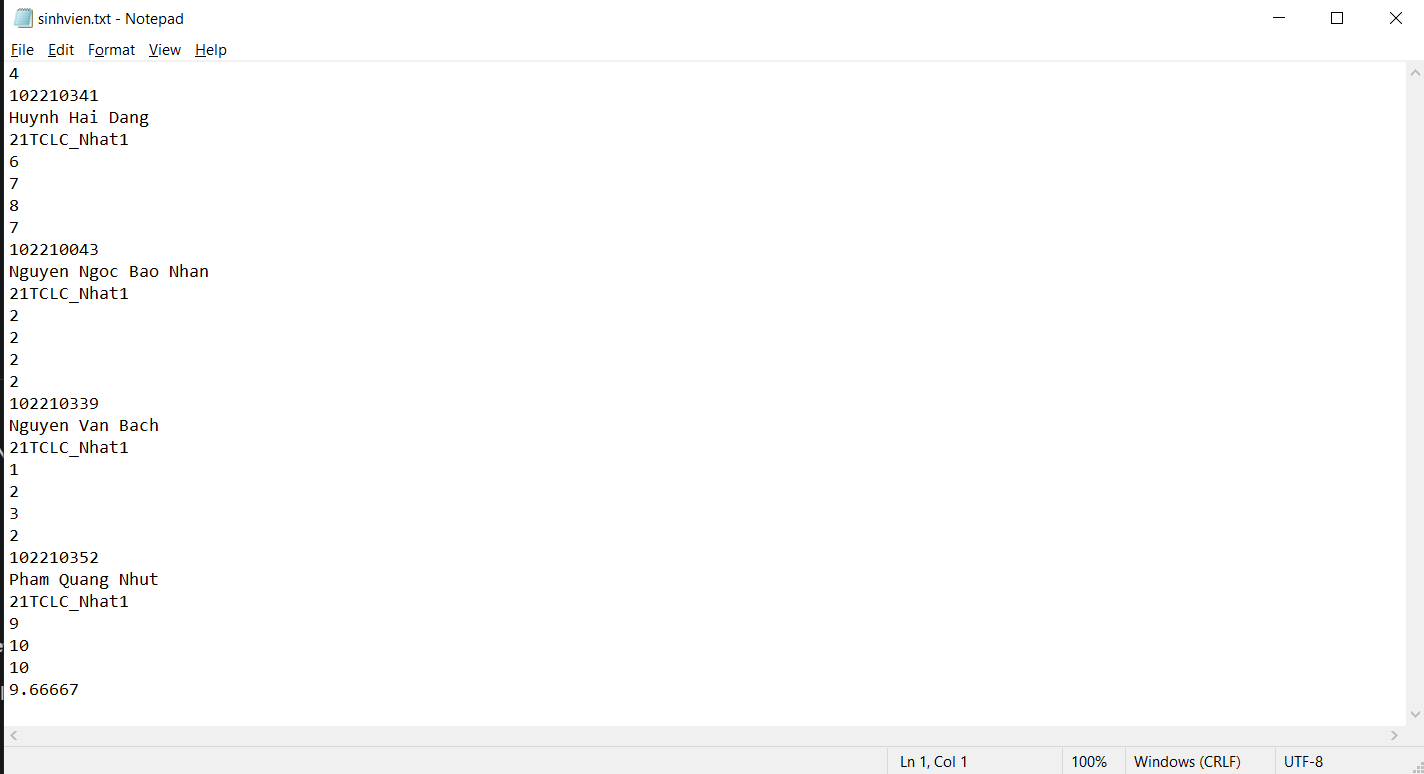
*Hình 4.3.2.9. Chỉnh sửa thông tin*

### 4.3.2.10. Sắp xếp sinh viên theo tên.



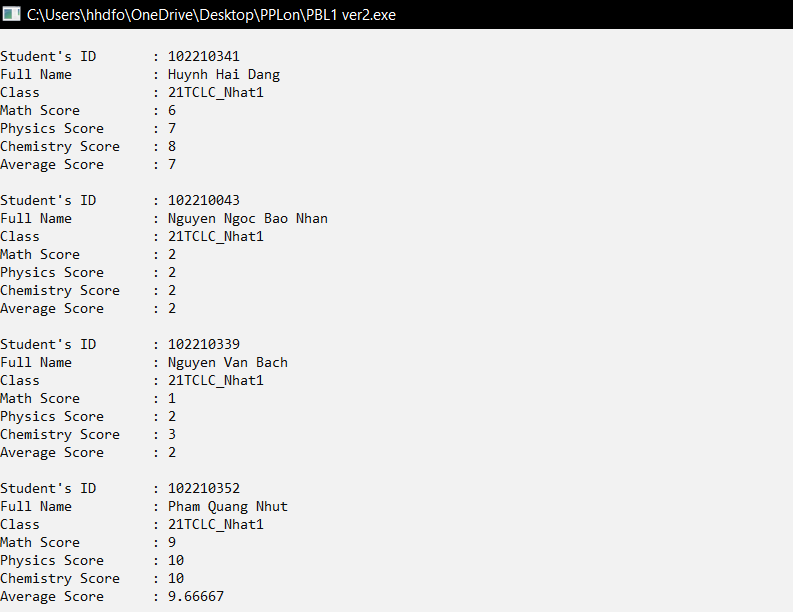
*Hình 4.3.2.10. Sắp xếp sinh viên theo tên*

### 4.3.2.11. Xuất danh sách ra file text



*Hình 4.3.2.11. Xuất ra file text*

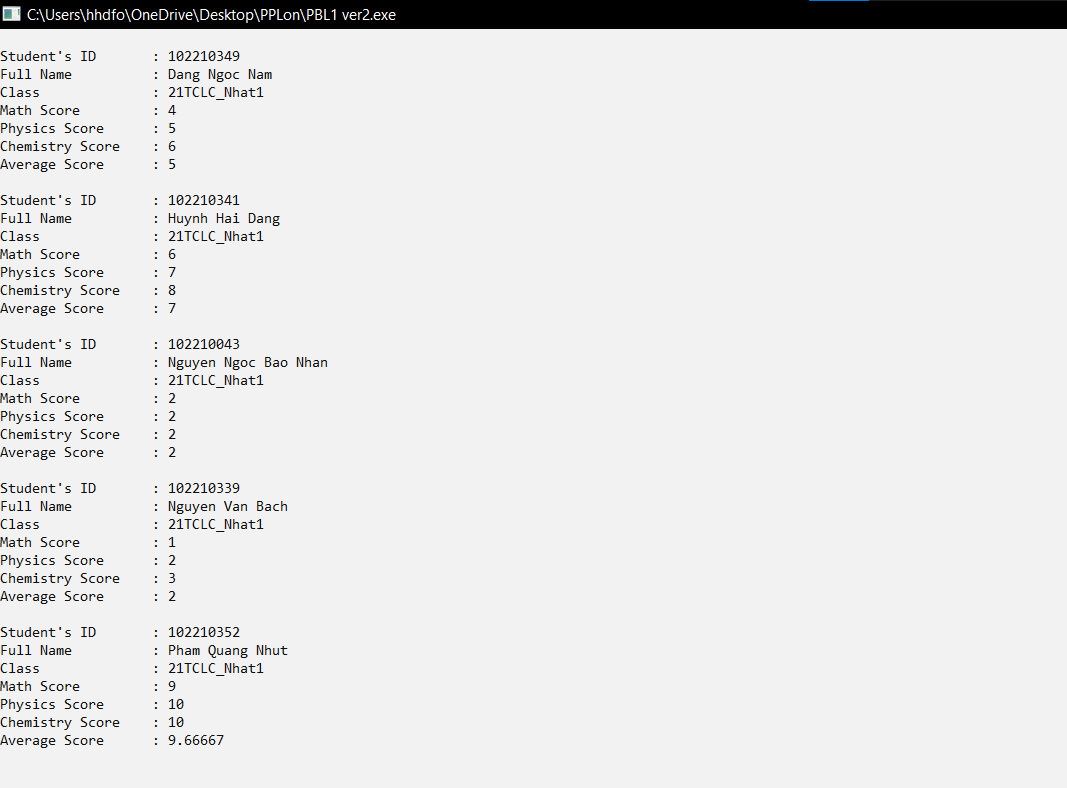
### 4.3.2.12. Đọc danh sách từ file text



*Hình 4.3.2.12. Đọc từ file text*

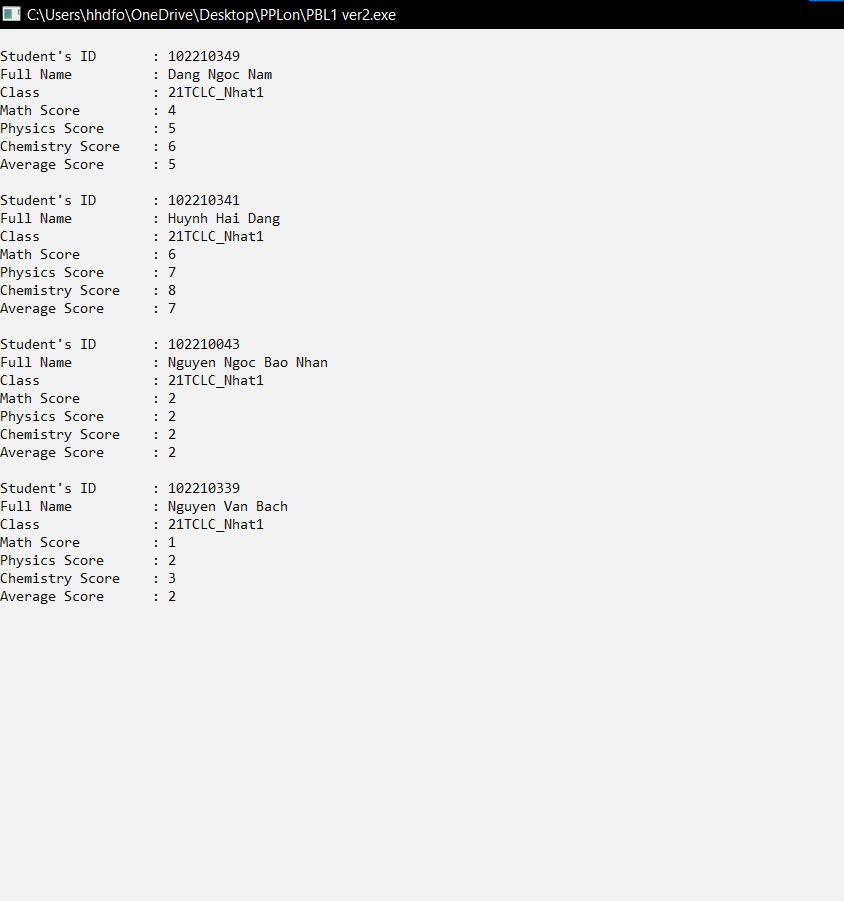
### 4.3.2.13. Thêm sinh viên mới vào đầu danh sách





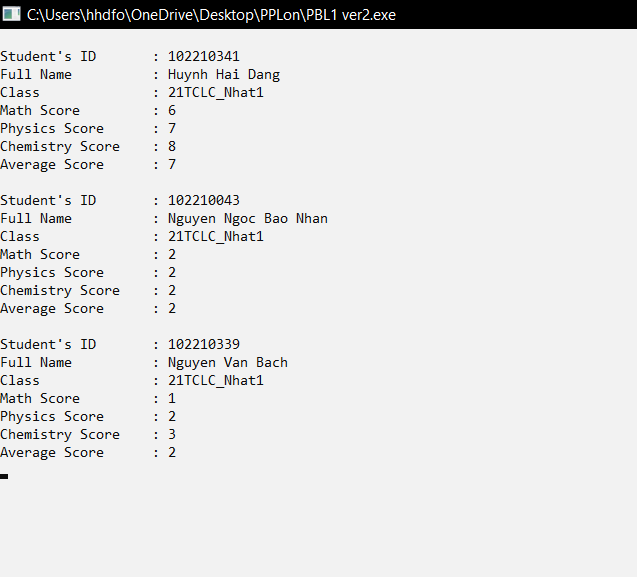
*Hình 4.3.2.13. Thêm sinh viên vào đầu danh sách*

### 4.3.2.14. Xóa sinh viên ở cuối danh sách



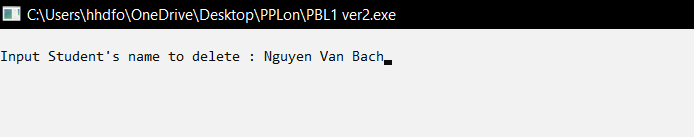
*Hình 4.3.2.14. Xóa sinh viên ở cuối danh sách*

### 4.3.2.15. Xóa sinh viên ở đầu danh sách



*Hình 4.3.2.15. Xóa sinh viên ở đầu danh sách*

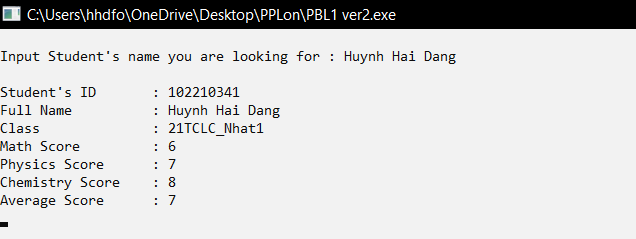
### 4.3.2.16. Xóa sinh viên theo tên





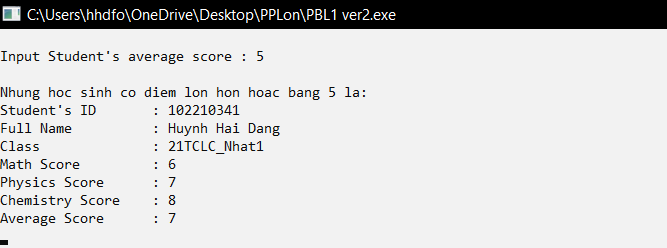
*Hình 4.3.2.16. Xóa sinh viên theo tên*

### 4.3.2.17. Tìm kiếm sinh viên theo tên



*Hình 4.3.2.17. Tìm kiếm sinh viên theo tên*

### 4.3.2.18. Tìm sinh viên có số điểm trên số điểm nhập vào



*Hình 4.3.2.18. Tìm sinh viên có số điểm lớn hơn số điểm nhập vào*

### 4.3.2.19. Tìm sinh viên có số điểm dưới số điểm nhập vào

### 

*Hình 4.3.2.19. Tìm sinh viên có số điểm dưới số điểm nhập vào*

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

***5.1. KẾT LUẬN***

**1. Các kết quả đạt được.**

- Trình bày khái quát lý thuyết về danh sách liên kết đơn

- Xây dựng được chương trình Quản lý sinh viên

**2. Ưu và nhược điểm của đề tài**

**- Ưu điểm**

* Chương trình dễ sử dụng, giao diện thân thiện
* Chương trình quản lý sinh viên ứng dụng danh sách liên kết đơn do vậy có ưu điểm là ….

**- Nhược điểm:**

* Vẫn chưa chú trọng về đồ họa, mới chỉ đi về phần lập trình.
* Code chương trình còn dài, chưa tối ưu

***5.2. HƯỚNG PHÁT TRIỂN***

Xây dựng thêm các tính năng như giao diện, thêm các thuộc tính như năm sinh, địa chỉ, số điện thoại…

# TÀI LIỆU THAM KHẢO

[1]https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=linked-lists

[2]<http://tailieu.vn/tag/tailieu/danh%20s%C3%A1ch%20li%C3%AAn%20k%E1%BA%BFt.html>

[3]. Đỗ Xuân Lôi. Cấu trúc dữ liệu và giải thuật. Nhà xuất bản Khoa học và Kỹ thuật, 1998

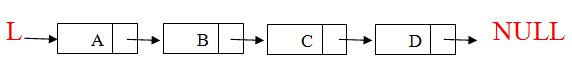
[4] https://www.geeksforgeeks.org/data-structures/linked-list/

[5] Nguyễn Xuân Huy. Thuật toán. Nhà xuất bản Thống kê, 1988

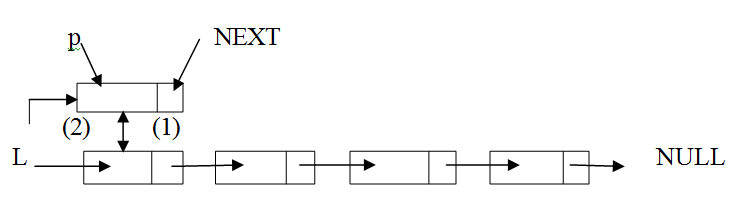
# PHỤ LỤC

**Mục lục hình ảnh:**

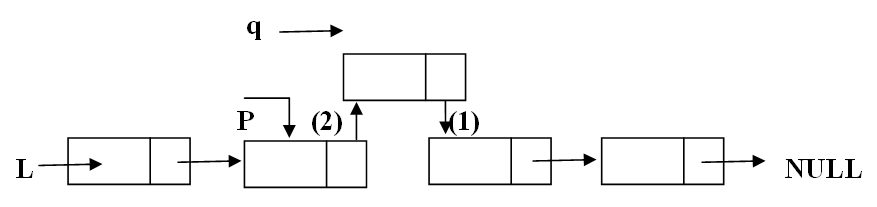
*Hình 1.0. Node*



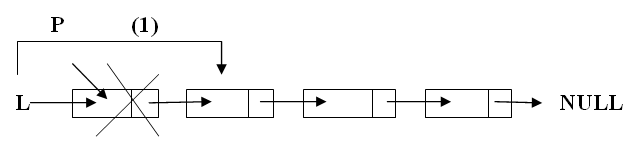
*Hình 1.1: Danh sách liên kết đơn*

**

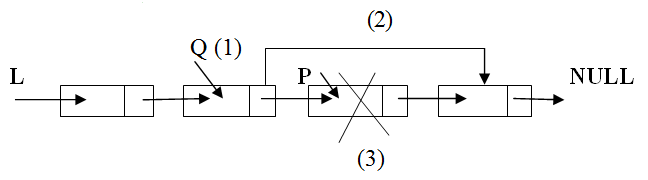
*Hình 1.2: Chèn node vào đầu danh sách*

****

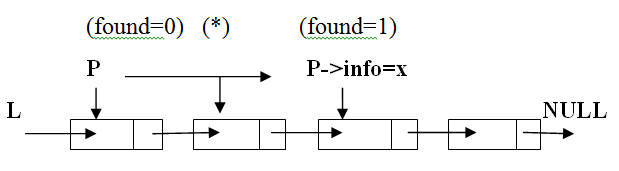
*Hình 1.3: Chèn Node vào Node được trỏ*



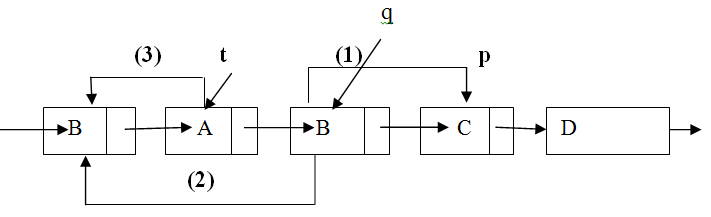
*Hình 1.4: Xóa một phần tử TH1*

****

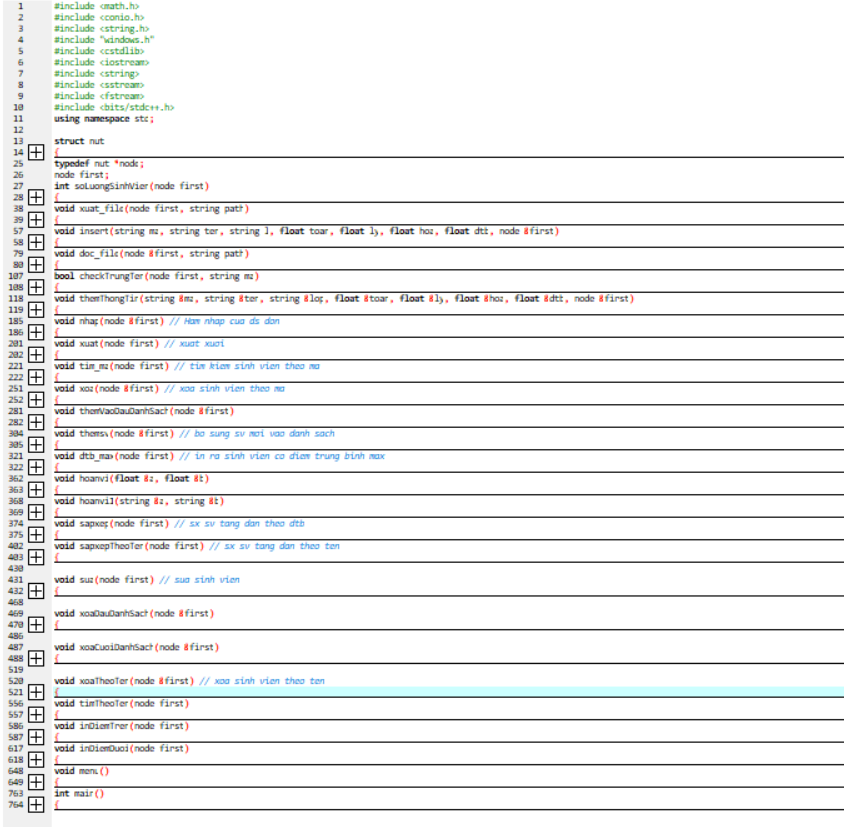
*Hình 1.5: Xóa 1 phần tử TH2*



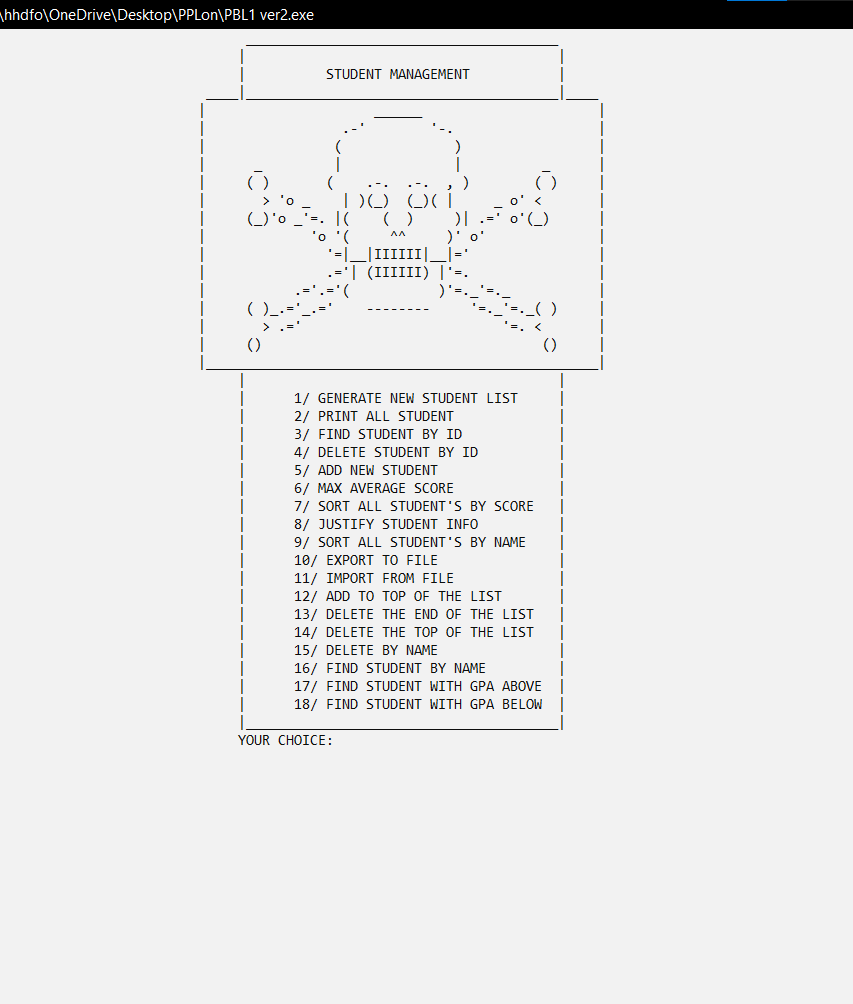
*Hình 1.6: Tìm kiếm phần tử*

****

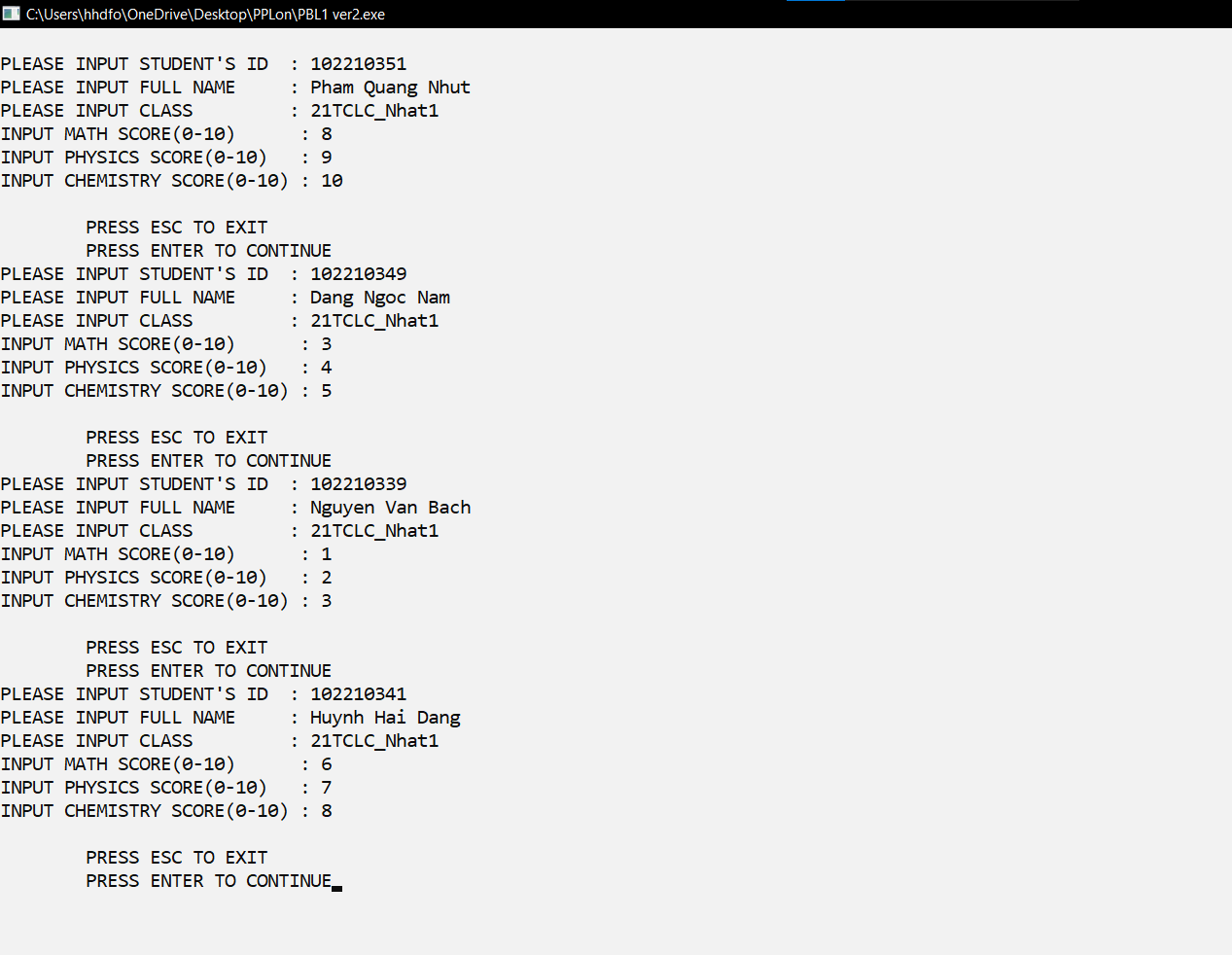
*Hình 1.7: Đảo các phần tử*



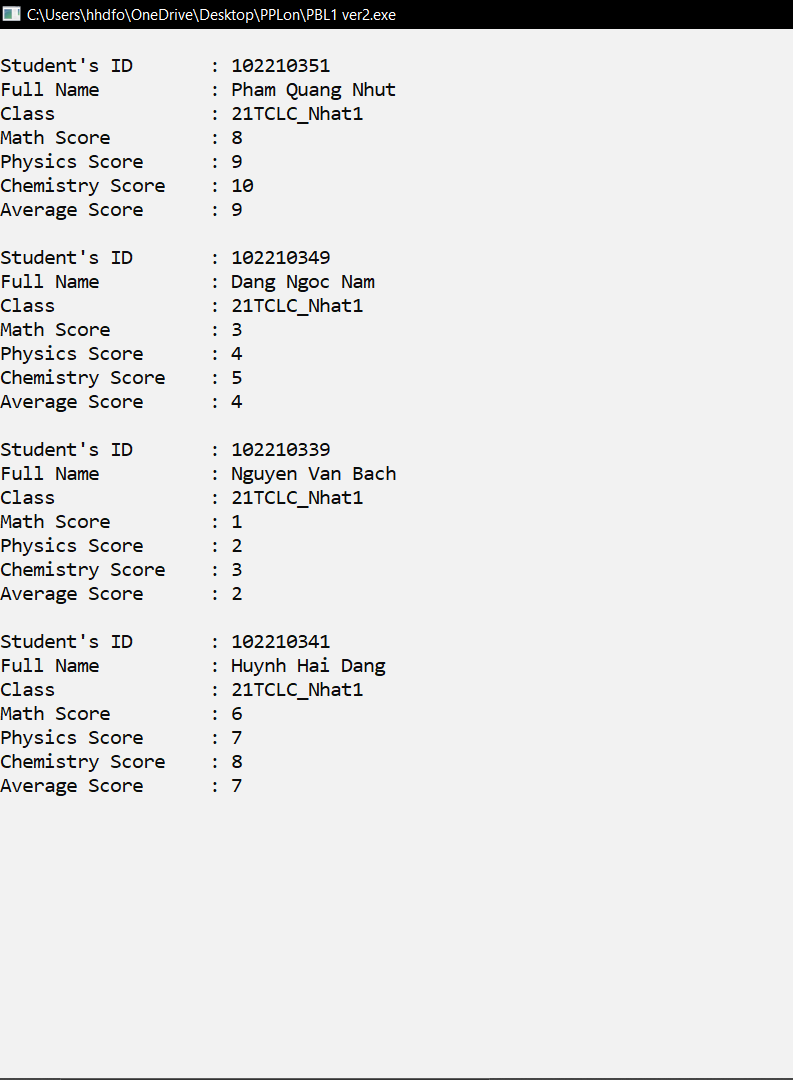
*Hình 4.3.1 Tổng quan chương trình*

****

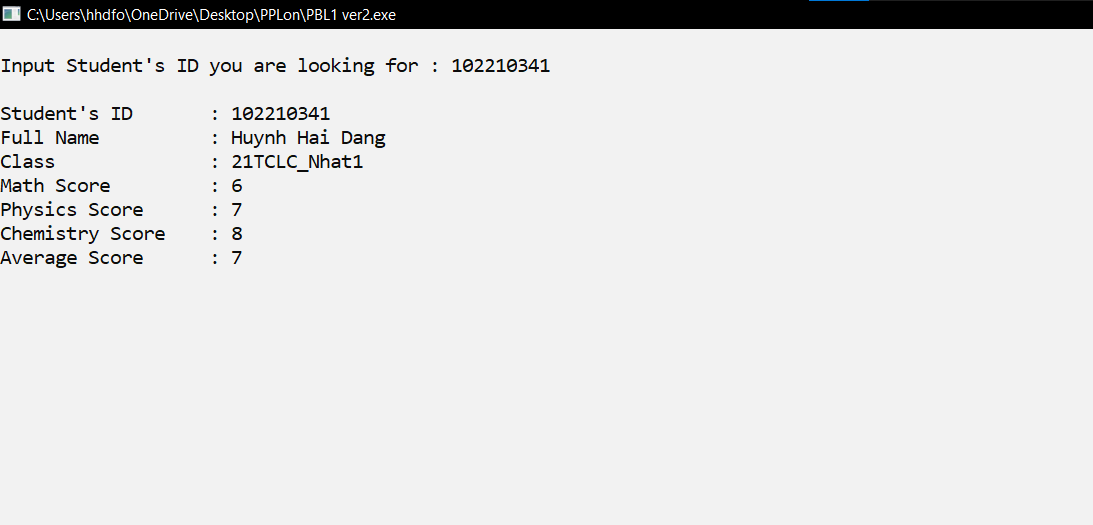
*Hình 4.3.2.1. Menu chính*



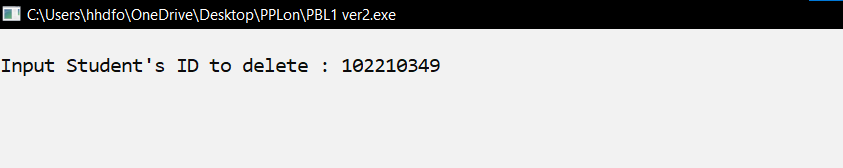
*Hình 4.3.2.2. Khởi tạo danh sách sinh viên*

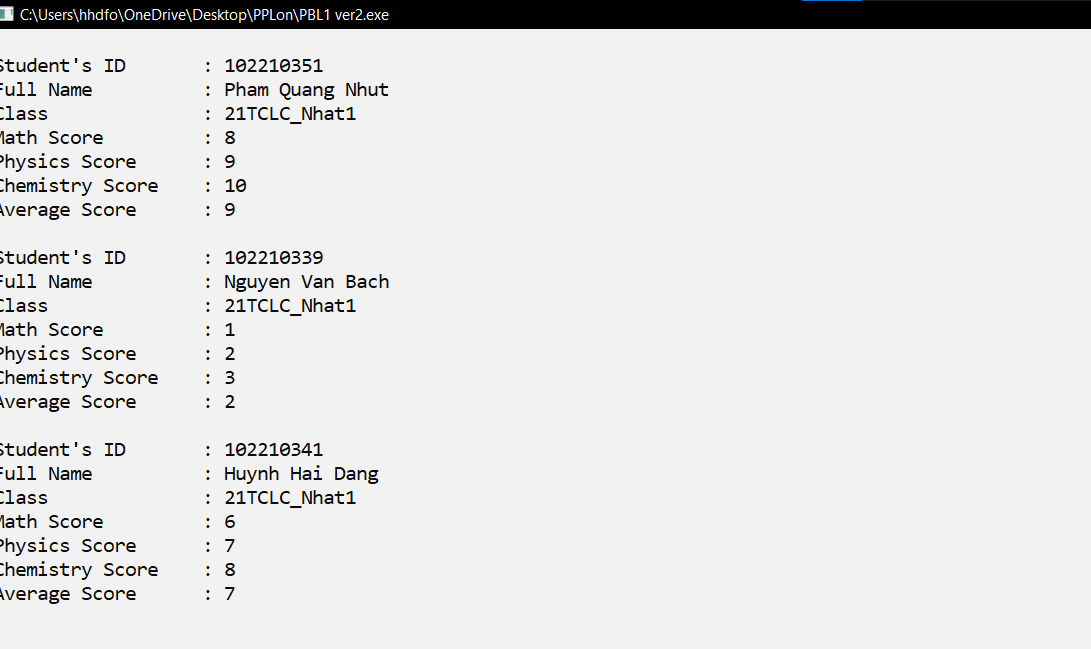


*Hình 4.3.2.3. In ra màn hình danh sách sinh viên*

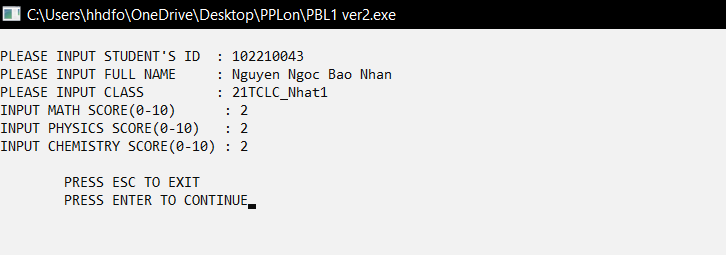
****

*Hình 4.3.2.4. Tìm kiếm sinh viên*

****



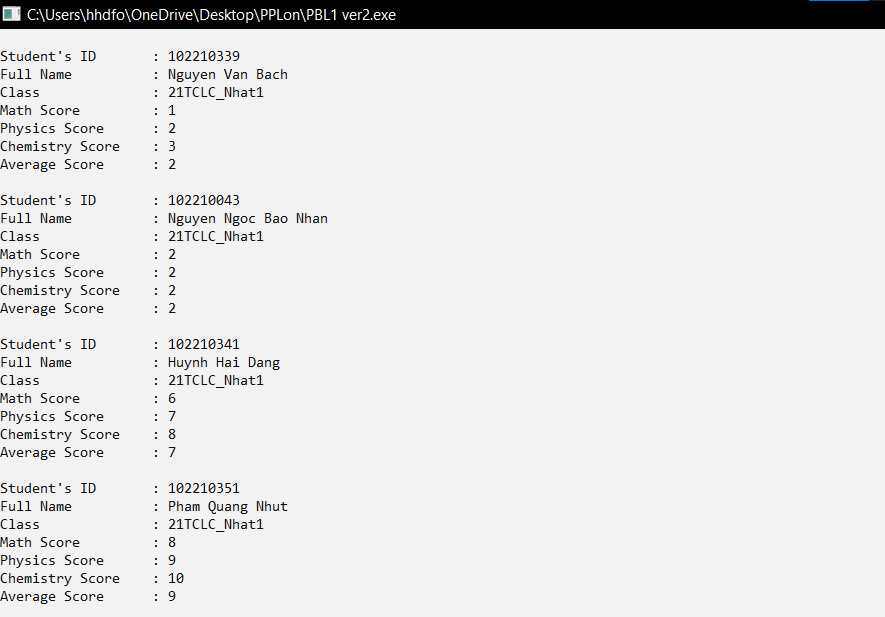
*Hình 4.3.2.5. Xóa sinh viên*



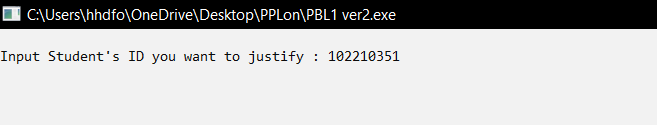
*Hình 4.3.2.6. Thêm sinh viên*

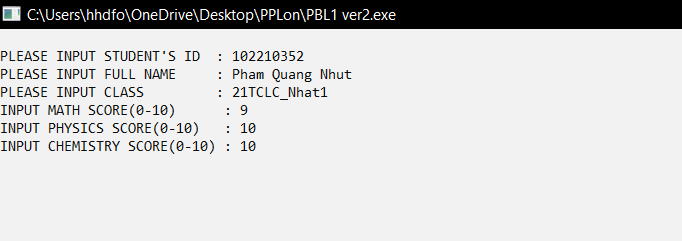
### 

*Hình 4.3.2.7. In sinh viên có điểm trung bình lớn nhất*

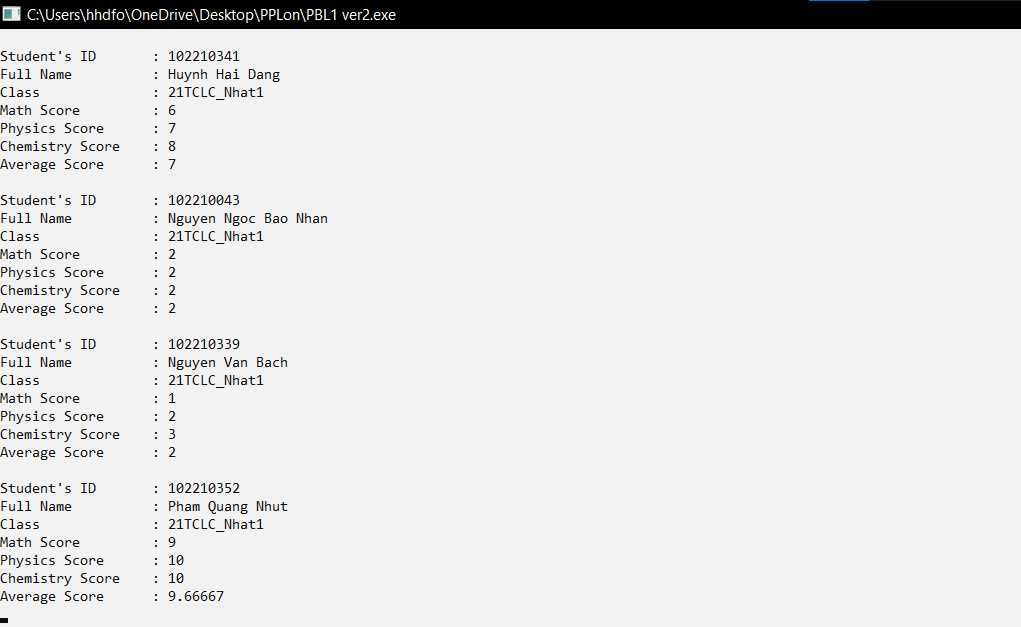


*Hình 4.3.2.8. Sắp xếp sinh viên tăng dần theo điểm trung bình*

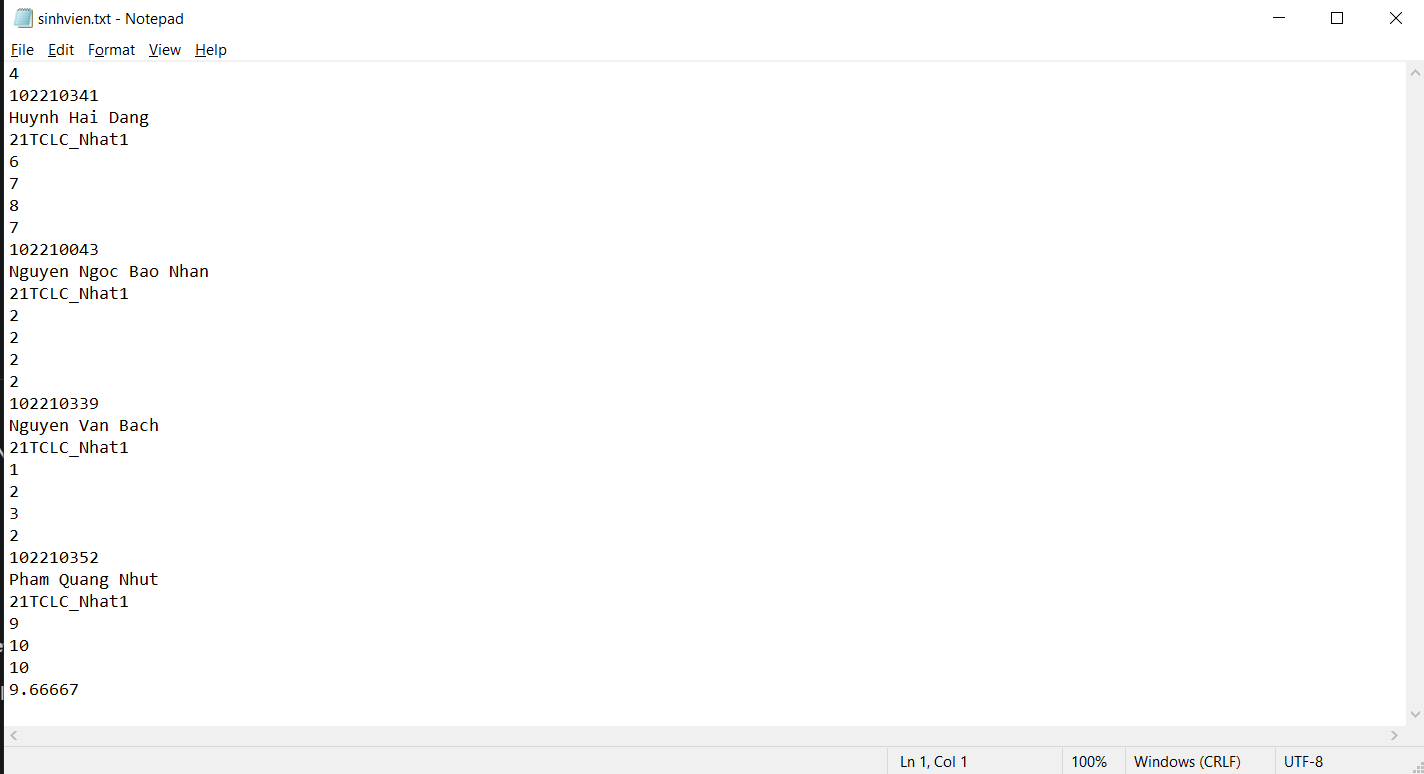




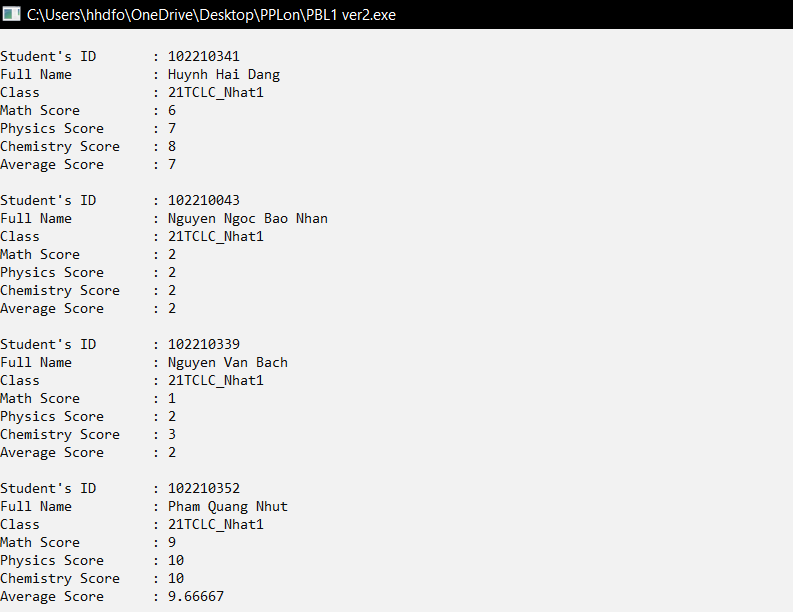
*Hình 4.3.2.9. Chỉnh sửa thông tin*



*Hình 4.3.2.10. Sắp xếp sinh viên theo tên*

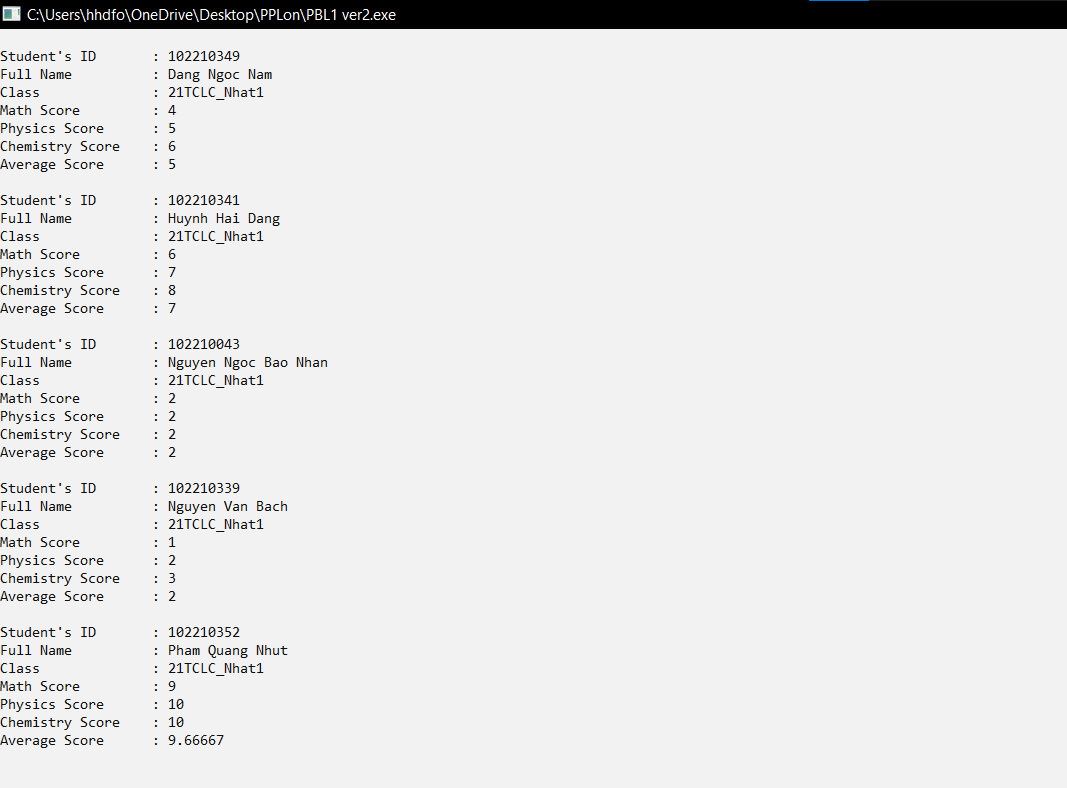


*Hình 4.3.2.11. Xuất ra file text*

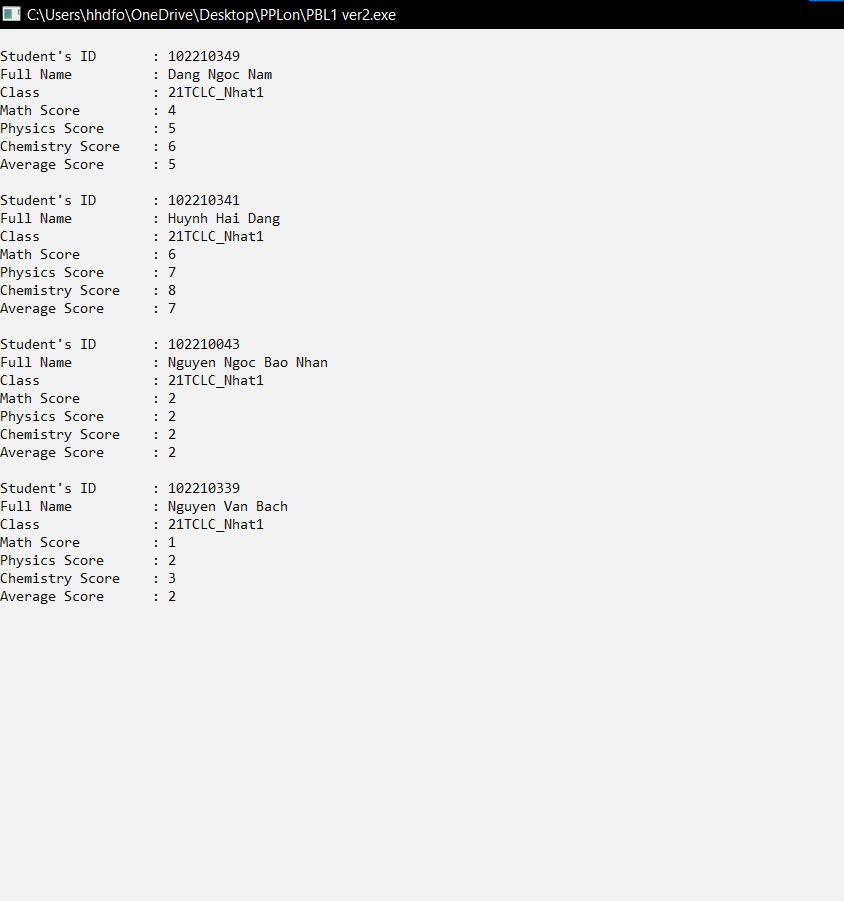


*Hình 4.3.2.12. Đọc từ file text*

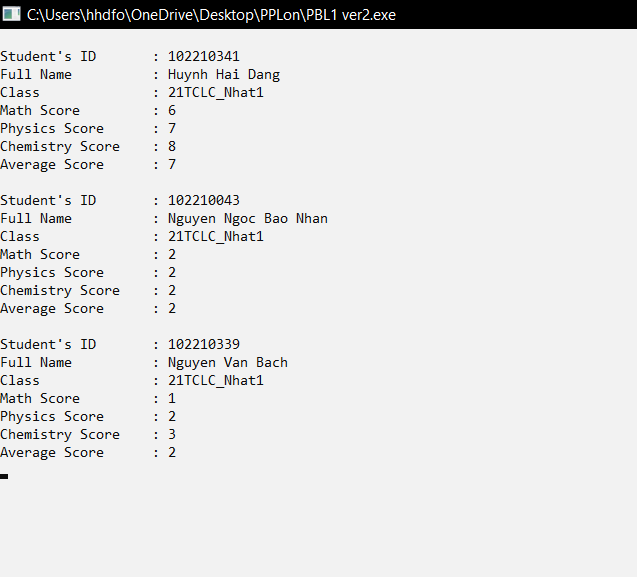




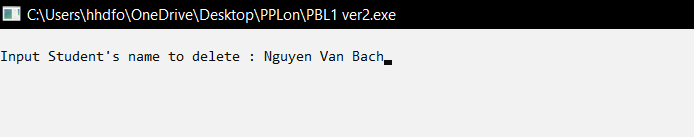
*Hình 4.3.2.13. Thêm sinh viên vào đầu danh sách*



*Hình 4.3.2.14. Xóa sinh viên ở cuối danh sách*

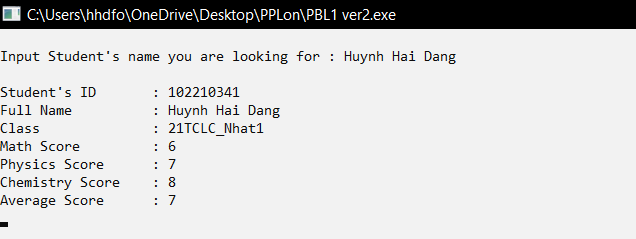


*Hình 4.3.2.15. Xóa sinh viên ở đầu danh sách*

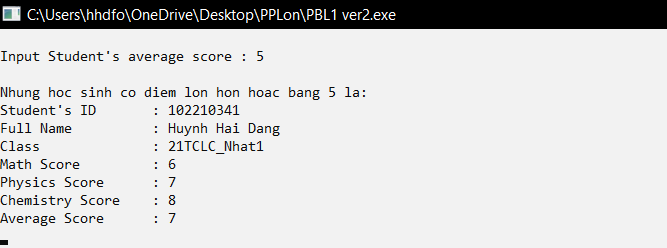




*Hình 4.3.2.16. Xóa sinh viên theo tên*



*Hình 4.3.2.17. Tìm kiếm sinh viên theo tên*



*Hình 4.3.2.18. Tìm sinh viên có số điểm lớn hơn số điểm nhập vào*

### 

*Hình 4.3.2.19. Tìm sinh viên có số điểm dưới số điểm nhập vào*

***Mục lục chương trình***

| #include <math.h>  #include <conio.h>  #include <string.h>  #include "windows.h"  #include <cstdlib>  #include <iostream>  #include <string>  #include <sstream>  #include <fstream>  #include <bits/stdc++.h>  using namespace std;  struct nut  {  string masv;  string hoten;  string lop;  float toan;  float ly;  float hoa;  float diemtb;  string xeploai;  nut \*next;  };  typedef nut \*node;  node first;  int soLuongSinhVien(node first)  {  int n = 0;  node p = first;  while (p != NULL)  {  n++;  p = p->next;  }  return n;  }  void xuat\_file(node first, string path)  {  ofstream o;  o.open(path.c\_str());  node p;  p = first;  o << soLuongSinhVien(first) << "\n";  while (p != NULL)  {  o << p->masv << "\n";  o << p->hoten << "\n";  o << p->lop << "\n";  o << p->toan << "\n";  o << p->ly << "\n";  o << p->hoa << "\n";  o << p->diemtb << "\n";  p = p->next;  }  }  void insert(string ma, string ten, string l, float toan, float ly, float hoa, float dtb, node &first)  {  node p, q;  p = new (nut);  p->masv = ma;  p->hoten = ten;  p->lop = l;  p->toan = toan;  p->ly = ly;  p->hoa = hoa;  p->diemtb = dtb;  p->next = NULL;  if (first == NULL)  first = p;  else  {  q = first;  while (q->next != NULL)  q = q->next;  q->next = p;  }  }  void doc\_file(node &first, string path)  {  ifstream i;  i.open(path.c\_str());  string ma;  string ten;  string l;  float toan;  float ly;  float hoa;  float dtb;  int n;  i >> n;  string bf;  getline(i, bf);  while (n--)  {  getline(i, ma);  getline(i, ten);  getline(i, l);  i >> toan;  i >> ly;  i >> hoa;  i >> dtb;  getline(i, bf);  insert(ma, ten, l, toan, ly, hoa, dtb, first);  }  }  bool checkTrungTen(node first, string ma)  {  node q = first;  while (q != NULL)  {  if (q->masv == ma)  return 0;  q = q->next;  }  return 1;  }  void themThongTin(string &ma, string &ten, string &lop, float &toan, float &ly, float &hoa, float &dtb, node &first)  {  while (true)  {  cout << "\nPLEASE INPUT STUDENT'S ID : ";  getline(cin, ma);  if (ma.length() > 0 && checkTrungTen(first, ma))  break;  else  cout << "ERROR, ";  }  while (true)  {  cout << "PLEASE INPUT FULL NAME : ";  getline(cin, ten);  if (ten.length() > 0)  break;  else  cout << "ERROR, ";  }  while (true)  {  cout << "PLEASE INPUT CLASS : ";  getline(cin, lop);  if (lop.length() > 0)  break;  else  cout << "ERROR, ";  }  while (true)  {  string input;  cout << "INPUT MATH SCORE(0-10) : ";  getline(cin, input);  // This code converts from string to number safely.  stringstream myStream(input);  if (myStream >> toan && toan >= 0 && toan <= 10)  break;  cout << "ERROR, PLEASE ";  }  while (true)  {  string input;  cout << "INPUT PHYSICS SCORE(0-10) : ";  getline(cin, input);  // This code converts from string to number safely.  stringstream myStream(input);  if (myStream >> ly && ly >= 0 && ly <= 10)  break;  cout << "ERROR, PLEASE ";  }  while (true)  {  string input;  cout << "INPUT CHEMISTRY SCORE(0-10) : ";  getline(cin, input);  // This code converts from string to number safely.  stringstream myStream(input);  if (myStream >> hoa && hoa >= 0 && hoa <= 10)  break;  cout << "ERROR, PLEASE ";  }  dtb = (toan + ly + hoa) / 3;  }  void nhap(node &first) // Ham nhap cua ds don  {  system("cls");  char ch;  first = NULL;  string ma, ten, lop;  float toan, ly, hoa, dtb;  do  {  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, first);  insert(ma, ten, lop, toan, ly, hoa, dtb, first);  cout << "\n PRESS ESC TO EXIT\n PRESS ENTER TO CONTINUE";  cin.sync();  ch = \_getch();  } while (ch != 27);  }  void xuat(node first) // xuat xuoi  {  char ch;  system("cls");  node p;  p = first;  while (p != NULL)  {  cout << "\nStudent's ID : " << p->masv;  cout << "\nFull Name : " << p->hoten;  cout << "\nClass : " << p->lop;  cout << "\nMath Score : " << p->toan;  cout << "\nPhysics Score : " << p->ly;  cout << "\nChemistry Score : " << p->hoa;  cout << "\nAverage Score : " << p->diemtb << "\n";  p = p->next;  }  cin.sync();  ch = \_getch();  }  void tim\_ma(node first) // tim kiem sinh vien theo ma  {  system("cls");  node p;  p = first;  string ma;  char ch;  cout << "\nInput Student's ID you are looking for : ";  cin >> ma;  while (p != NULL)  {  if (p->masv == ma)  {  cout << "\nStudent's ID : " << p->masv;  cout << "\nFull Name : " << p->hoten;  cout << "\nClass : " << p->lop;  cout << "\nMath Score : " << p->toan;  cout << "\nPhysics Score : " << p->ly;  cout << "\nChemistry Score : " << p->hoa;  cout << "\nAverage Score : " << p->diemtb << "\n";  cin.sync();  ch = \_getch();  return;  }  p = p->next;  }  cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  }  void xoa(node &first) // xoa sinh vien theo ma  {  system("cls");  node p, q;  p = first;  q = first;  string ma;  cout << "\nInput Student's ID to delete : ";  cin >> ma;  while (p != NULL && p->masv != ma)  p = p->next;  if (p == first)  first = first->next;  else  {  if (p == NULL)  {  cout << "\nCan's find student to delete!";  }  else  {  while (q != NULL && q->next != p)  q = q->next;  q->next = p->next;  }  }  system("cls");  xuat(first);  \_getch();  }  void themVaoDauDanhSach(node &first)  {  system("cls");  char ch;  string ma, ten, lop;  float toan, ly, hoa, dtb;  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, first);  node p;  p = new (nut);  p->masv = ma;  p->hoten = ten;  p->lop = lop;  p->toan = toan;  p->ly = ly;  p->hoa = hoa;  p->diemtb = dtb;  p->next = first;  first = p;  cin.sync();  cin.sync();  ch = \_getch();  ;  }  void themsv(node &first) // bo sung sv moi vao danh sach  {  system("cls");  char ch;  string ma, ten, lop;  float toan, ly, hoa, dtb;  do  {  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, first);  insert(ma, ten, lop, toan, ly, hoa, dtb, first);  cout << "\n PRESS ESC TO EXIT\n PRESS ENTER TO CONTINUE";  cin.sync();  cin.sync();  ch = \_getch();  } while (ch != 27);  }  void dtb\_max(node first) // in ra sinh vien co diem trung binh max  {  char ch;  system("cls");  if (first == NULL)  {  cout << "THERE IS NO STUDENT IN LIST";  cin.sync();  ch = \_getch();  return;  }  node p = first, q = first, l = first;  float max = p->diemtb;  p = p->next;  while (p != NULL)  {  if (p->diemtb > max)  {  max = p->diemtb;  q = p;  }  p = p->next;  }  while (l != NULL)  {  if (l == q)  {  cout << "\nStudent's ID : " << l->masv;  cout << "\nFull Name : " << l->hoten;  cout << "\nClass : " << l->lop;  cout << "\nMath Score : " << l->toan;  cout << "\nPhysics Score : " << l->ly;  cout << "\nChemistry Score : " << l->hoa;  cout << "\nAverage Score : " << l->diemtb << "\n";  break;  }  l = l->next;  }  cin.sync();  ch = \_getch();  }  void hoanvi(float &a, float &b)  {  float c = a;  a = b;  b = c;  }  void hoanvi1(string &a, string &b)  {  string c = a;  a = b;  b = c;  }  void sapxep(node first) // sx sv tang dan theo dtb  {  char ch;  system("cls");  node p = first, q;  while (p != NULL)  {  q = p->next;  while (q != NULL)  {  if (p->diemtb > q->diemtb)  {  hoanvi1(p->masv, q->masv);  hoanvi1(p->hoten, q->hoten);  hoanvi1(p->lop, q->lop);  hoanvi(p->toan, q->toan);  hoanvi(p->ly, q->ly);  hoanvi(p->hoa, q->hoa);  hoanvi(p->diemtb, q->diemtb);  }  q = q->next;  }  p = p->next;  }  xuat(first);  cin.sync();  ch = \_getch();  }  void sapxepTheoTen(node first) // sx sv tang dan theo ten  {  char ch;  system("cls");  node p = first, q;  while (p != NULL)  {  q = p->next;  while (q != NULL)  {  if (p->hoten > q->hoten)  {  hoanvi1(p->masv, q->masv);  hoanvi1(p->hoten, q->hoten);  hoanvi1(p->lop, q->lop);  hoanvi(p->toan, q->toan);  hoanvi(p->ly, q->ly);  hoanvi(p->hoa, q->hoa);  hoanvi(p->diemtb, q->diemtb);  }  q = q->next;  }  p = p->next;  }  xuat(first);  cin.sync();  ch = \_getch();  }  void sua(node first) // sua sinh vien  {  char ch;  system("cls");  node p;  p = first;  string ma;  cout << "\nInput Student's ID you want to justify : ";  getline(cin, ma);  while (p != NULL)  {  if (p->masv == ma)  {  system("cls");  char ch;  string ma, ten, lop;  float toan, ly, hoa, dtb;  fflush(stdin);  themThongTin(ma, ten, lop, toan, ly, hoa, dtb, p);  p->masv = ma;  p->hoten = ten;  p->lop = lop;  p->toan = toan;  p->ly = ly;  p->hoa = hoa;  p->diemtb = dtb;  cin.sync();  ch = \_getch();  return;  }  p = p->next;  }  printf("CANNOT FILE THIS ID");  cin.sync();  ch = \_getch();  return;  }  void xoaDauDanhSach(node &first)  {  char ch;  if (first == NULL)  {  system("cls");  cout << "\nTHERE IS NO STUDENT IN LIST\n";  cin.sync();  ch = \_getch();  return;  }  first = first->next;  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  }  void xoaCuoiDanhSach(node &first)  {  char ch;  node p;  p = first;  if (first == NULL)  {  system("cls");  cout << "\nTHERE IS NO STUDENT IN LIST\n";  cin.sync();  ch = \_getch();  return;  }  if (p->next == NULL)  {  first = NULL;  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  return;  }  while (p->next->next != NULL)  {  p = p->next;  }  p->next = NULL;  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  }  void xoaTheoTen(node &first) // xoa sinh vien theo ma  {  char ch;  system("cls");  node p, q;  p = first;  q = first;  string name;  cout << "\nInput Student's name to delete : ";  getline(cin, name);  while (p != NULL && p->hoten != name)  p = p->next;  if (p == first)  first = first->next;  else  {  if (p == NULL)  {  cout << "\nCan't find student to delete!";  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  }  else  {  while (q != NULL && q->next != p)  q = q->next;  q->next = p->next;  }  }  system("cls");  xuat(first);  cin.sync();  ch = \_getch();  }  void timTheoTen(node first)  {  char ch;  system("cls");  node p;  p = first;  string name;  cout << "\nInput Student's name you are looking for : ";  getline(cin, name);  while (p != NULL)  {  if (p->hoten == name)  {  cout << "\nStudent's ID : " << p->masv;  cout << "\nFull Name : " << p->hoten;  cout << "\nClass : " << p->lop;  cout << "\nMath Score : " << p->toan;  cout << "\nPhysics Score : " << p->ly;  cout << "\nChemistry Score : " << p->hoa;  cout << "\nAverage Score : " << p->diemtb << "\n";  // cin.sync();  ch = \_getch();  // return;  }  p = p->next;  }  // cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  }  void inDiemTren(node first)  {  char ch;  system("cls");  node p;  p = first;  int score;  cout << "\nInput Student's average score : ";  cin >> score;  cout << "\nNhung hoc sinh co diem lon hon hoac bang " << score << " la: ";  while (p != NULL)  {  if (p->diemtb >= score)  {  cout << "\nStudent's ID : " << p->masv;  cout << "\nFull Name : " << p->hoten;  cout << "\nClass : " << p->lop;  cout << "\nMath Score : " << p->toan;  cout << "\nPhysics Score : " << p->ly;  cout << "\nChemistry Score : " << p->hoa;  cout << "\nAverage Score : " << p->diemtb << "\n";  // cin.sync();  // ch = \_getch();  // return;  }  p = p->next;  }  // cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  }  void inDiemDuoi(node first)  {  char ch;  system("cls");  node p;  p = first;  int score;  cout << "\nInput Student's average score : ";  cin >> score;  cout << "\nNhung hoc sinh co diem be hon bang " << score << " la: ";  while (p != NULL)  {  if (p->diemtb <= score)  {  cout << "\nStudent's ID : " << p->masv;  cout << "\nFull Name : " << p->hoten;  cout << "\nClass : " << p->lop;  cout << "\nMath Score : " << p->toan;  cout << "\nPhysics Score : " << p->ly;  cout << "\nChemistry Score : " << p->hoa;  cout << "\nAverage Score : " << p->diemtb << "\n";  // cin.sync();  // ch = \_getch();  // return;  }  p = p->next;  }  // cout << "CANNOT FIND THIS STUDENT BY NAME";  cin.sync();  ch = \_getch();  }  void menu()  {  int opt = 1;  while (opt)  {  cout << "\n\t\t\t\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";  cout << "\t\t\t\t\t| |\n";  cout << "\t\t\t\t\t| STUDENT MANAGEMENT |";  // cout << "\t\t\t\t\t| |";  cout << "\n\t\t\t\t \_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\n";  cout << "\t\t\t\t | \_\_\_\_\_\_ |\n";  cout << "\t\t\t\t | .-' '-. |\n";  cout << "\t\t\t\t | ( ) |\n";  cout << "\t\t\t\t | \_ | | \_ |\n";  cout << "\t\t\t\t | ( ) ( .-. .-. , ) ( ) |\n";  cout << "\t\t\t\t | > 'o \_ | )(\_) (\_)( | \_ o' < |\n";  cout << "\t\t\t\t | (\_)'o \_'=. |( ( ) )| .=' o'(\_) |\n";  cout << "\t\t\t\t | 'o '( ^^ )' o' |\n";  cout << "\t\t\t\t | '=|\_\_|IIIIII|\_\_|=' |\n";  cout << "\t\t\t\t | .='| (IIIIII) |'=. |\n";  cout << "\t\t\t\t | .='.='( )'=.\_'=.\_ |\n";  cout << "\t\t\t\t | ( )\_.='\_.=' -------- '=.\_'=.\_( ) |\n";  cout << "\t\t\t\t | > .=' '=. < |\n";  cout << "\t\t\t\t | () () |";  cout << "\n\t\t\t\t |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";  cout << "\t\t\t\t\t| |\n";  cout << "\t\t\t\t\t| 1/ GENERATE NEW STUDENT LIST |\n";  cout << "\t\t\t\t\t| 2/ PRINT ALL STUDENT |\n";  cout << "\t\t\t\t\t| 3/ FIND STUDENT BY ID |\n";  cout << "\t\t\t\t\t| 4/ DELETE STUDENT BY ID |\n";  cout << "\t\t\t\t\t| 5/ ADD NEW STUDENT |\n";  cout << "\t\t\t\t\t| 6/ MAX AVERAGE SCORE |\n";  cout << "\t\t\t\t\t| 7/ SORT ALL STUDENT'S BY SCORE |\n";  cout << "\t\t\t\t\t| 8/ JUSTIFY STUDENT INFO |\n";  cout << "\t\t\t\t\t| 9/ SORT ALL STUDENT'S BY NAME |\n";  cout << "\t\t\t\t\t| 10/ EXPORT TO FILE |\n";  cout << "\t\t\t\t\t| 11/ IMPORT FROM FILE |\n";  cout << "\t\t\t\t\t| 12/ ADD TO TOP OF THE LIST |\n";  cout << "\t\t\t\t\t| 13/ DELETE THE END OF THE LIST |\n";  cout << "\t\t\t\t\t| 14/ DELETE THE TOP OF THE LIST |\n";  cout << "\t\t\t\t\t| 15/ DELETE BY NAME |\n";  cout << "\t\t\t\t\t| 16/ FIND STUDENT BY NAME |\n";  cout << "\t\t\t\t\t| 17/ FIND STUDENT WITH GPA ABOVE |\n";  cout << "\t\t\t\t\t| 18/ FIND STUDENT WITH GPA BELOW |\n";  cout << "\t\t\t\t\t|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";  cout << "\t\t\t\t\tYOUR CHOICE: ";  cin >> opt;  string bf;  getline(cin, bf);  switch (opt)  {  case 1:  nhap(first);  break;  case 2:  xuat(first);  break;  case 3:  tim\_ma(first);  break;  case 4:  xoa(first);  break;  case 5:  themsv(first);  break;  case 6:  dtb\_max(first);  break;  case 7:  sapxep(first);  break;  case 8:  sua(first);  break;  case 9:  sapxepTheoTen(first);  break;  case 10:  xuat\_file(first, "sinhvien.txt");  break;  case 11:  doc\_file(first, "sinhvien.txt");  break;  case 12:  themVaoDauDanhSach(first);  break;  case 13:  xoaCuoiDanhSach(first);  break;  case 14:  xoaDauDanhSach(first);  break;  case 15:  xoaTheoTen(first);  break;  case 16:  timTheoTen(first);  break;  case 17:  inDiemTren(first);  break;  case 18:  inDiemDuoi(first);  break;  default:  cout << "\n\t\t\t\t\t\t \_\_\_\_\_\_\_ \n";  cout << "\t\t\t\t\t\t | |\n";  cout << "\t\t\t\t\t\t | ERROR |\n";  cout << "\t\t\t\t\t\t |\_\_\_\_\_\_\_|\n";  opt = 0;  break;  }  }  }  int main()  {  // nhap(first);  // xuat(first);111  // // tim\_ma(first);  // // xoa(first);  // // xuat(first); // xuat sau khi xoa  // // themsv(first);  // // xuat(first); // xuat sau khi them  // // dtb\_max(first);  // // sapxep(first);  // // xuat(first); // xuat sau khi sap xep  menu();  return 0;} |
| --- |