TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM

KHOA CÔNG NGHỆ THÔNG TIN

-----*-----**

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN "CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT"

_

Đề tài:

Cài đặt cấu trúc dữ liệu kiểu cây. Cài đặt các phương án duyệt cây nhị phân và cây tổng quát

GVHD: ***ThS. Nguyễn Hạnh Phúc***

Sinh viên thực hiện: ***Nguyễn Hữu Tiệp– Mã sv: 12345***

Trần trung Hiếu – Mã sv: 34567

Hải Phòng, tháng 04 năm 2017

Mã đề tài: 16

1. **Tên đề tài**

   Cài đặt cấu trúc dữ liệu kiểu cây. Cài đặt các phương án duyệt cây nhị phân và cây tổng quát.

2. **Mục đích**

Xây dựng chương trình quản lý các thí sinh dự thí vào một trường Đại học theo phương pháp lập trình hướng đối tượng.

3. **Công việc cần thực hiện**

- Xây dựng lớp đối tượng NGƯỜI với các thuộc tính sau:- Số chứng minh thư, - Tên, - Ngày sinh, - Quê quán, - Giới tính.
- Xây dựng lớp đối tượng THÍ SINH kế thừa từ lớp NGƯỜI với các thuộc tính sau: - Số báo danh,- Điểm toán, - Điểm lý, - Điểm hóa.
- Xây dựng các lớp đối tượng trên với các phương thức: - Hàm tạo, hàm hủy - Các phương thức nhập, xuất dữ liệu.
- Xây dựng lớp DANH SÁCH quản lý các thí sinh với các thuộc tính: - Số lượng các thí sinh, - Danh sách các thí sinh

  và phương thức sau: - Các hàm tạo, hàm hủy; - Các phương thức cho phép: Thêm 1 thí sinh vào danh sách; Loại bỏ 1 thí sinh ra khỏi danh sách; Tìm các thí sinh có tổng điểm lớn hơn giá trị đưa vào; In danh sách các thí sinh có tổng điểm lớn hơn hoặc bằng giá trị đưa vào.
- Làm báo cáo bài tập lớn
- Bảo vệ bài tập lớn

4. **Yêu cầu**

- Kết quả làm bài tập lớn: Báo cáo bài tập lớn
- Báo cáo bài tập lớn phải được trình bày theo mẫu quy định (kèm theo), báo cáo có thể kết xuất thành tệp định dạng PDF và nộp qua email (không bắt buộc phải in ấn)
- Hạn nộp báo cáo bài tập lớn:

5. **Tài liệu tham khảo**

- Nguyễn Thanh Thuỷ, **Lập trình HĐT với C++**, NXB ĐH QG HN, 2010.
- **Bài giảng Lập trình hướng đối tượng với C++**, Khoa CNTT, ĐH HH VN

Hải Phòng, tháng 04 năm 2017

NGƯỜI HƯỚNG DẪN

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ, BẢNG BIỂU

DANH MỤC CÁC TỪ VIẾT TẮT

Từ	**Ý nghĩa**
OOP	Object Oriented Progamming

GIỚI THIỆU

Bài toán:

Để quản lý các thí sinh dự thí vào một trường Đại học, người ta xây dựng chương trình với các lớp đối tượng, có các thuộc tính và phương thức tương ứng.

- ***Người:*** Số chứng minh thư, Tên, Ngày sinh, Quê quán, Giới tính.
- ***Thí Sinh*** (kế thừa từ lớp **Người**) : Số báo danh, Điểm môn thứ nhất, Điểm môn thứ hai, Điểm môn thứ ba, Điểm ưu tiên.

Yêu cầu:

- Xây dựng các lớp đối tượng trên với các phương thức:
  - Hàm tạo, hàm hủy
  - Các phương thức nhập xuất
- Xây dựng lớp DANH SÁCH quản lý các thí sinh với các thuộc tính và phương thức sau:
  - Số lượng các thí sinh
  - Danh sách các thí sinh
  - Các hàm tạo, hàm hủy
  - Các phương thức cho phép: Thêm 1 thí sinh vào danh sách; Loại bỏ 1 thí sinh ra khỏi danh sách; Tìm các thí sinh có tổng điểm lớn hơn giá trị đưa vào; In danh sách các thí sinh có tổng điểm lớn hơn hoặc bằng giá trị đưa vào.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Lớp và đối tượng

1.1.1. Định nghĩa Lớp trong C++

Khi bạn định nghĩa một lớp, bạn định nghĩa một blueprint cho một kiểu dữ liệu. Điều này không thực sự định nghĩa bất kỳ dữ liệu nào, nhưng nó định nghĩa ý nghĩa tên lớp là gì, đó là, những gì một đối tượng của lớp đó sẽ bao gồm và những hoạt động nào có thể được thực hiện trên một đối tượng đó.

Một định nghĩa lớp trong C++ bắt đầu với từ khóa class, được theo sau bởi tên lớp và phần thân lớp, được bao quanh trong một cặp dấu ngoặc móc. Một định nghĩa lớp phải được theo sau: hoặc bởi một dấu chấm phảy hoặc một danh sách các khai báo. Ví dụ, chúng ta định nghĩa kiểu dữ liệu Box bởi sử dụng từ khóa **class** trong C++ như sau:

```
class Box
{
 public:
 double chieudai; // chieu dai cua hop
 double chieurong; // chieu rong cua hop
 double chieucao; // chieu cao cua hop
};
```

Từ khóa **public** quyết định các thuộc tính truy cập của các thành viên lớp mà theo sau nó. Một thành viên public có thể được truy cập từ bên ngoài lớp bất cứ đâu bên trong phạm vi (scope) của đối tượng lớp đó. Bạn cũng có thể xác định các thành viên của lớp là **private**hoặc **protected** sẽ được bàn luận trong chương phụ.

1.1.2. Định nghĩa Đối tượng trong C++

Một lớp cung cấp các blueprint cho các đối tượng, vì thế về cơ bản, một đối tượng được tạo từ một lớp. Chúng ta khai báo các đối tượng của một lớp giống đúng như chúng ta khai báo các biến của kiểu cơ bản. Các lệnh sau khai báo hai đối tượng của lớp Box:

```
Box Box1; // Khai bao Box1 la cua kieu Box
Box Box2; // Khai bao Box2 la cua kieu Box
```

Cả hai đối tượng Box1 và Box2 sẽ có bản sao của các thành viên dữ liệu (Data Member) riêng.

1.1.3. Truy cập các thành viên dữ liệu trong C++

Các thành viên dữ liệu public của các đối tượng của một lớp có thể được truy cập bởi sử dụng toán tử truy cập thành viên trực tiếp là dấu chấm (.). Bạn sẽ thấy rõ ràng khi xem ví dụ sau:

```
#include <iostream>

using namespace std;

class Box
{
 public:
 double chieudai; // chieu dai cua hop
 double chieurong; // chieu rong cua hop
 double chieucao; // chieu cao cua hop
};

int main( )
{
Box Box1; // Khai bao Box1 la cua kieu Box
```

```
Box Box2; // Khai bao Box2 la cua kieu Box
 double thetich = 0.0; // Luu giu the tich cua Box vao bien thetich

 // box 1 specification
 Box1.chieucao = 4.5;
 Box1.chieudai = 2.2;
 Box1.chieurong = 1.5;

 // box 2 specification
 Box2.chieucao = 3.2;
 Box2.chieudai = 4.5;
 Box2.chieurong = 2.3;
 // thetich of box 1
 thetich = Box1.chieucao * Box1.chieudai * Box1.chieurong;
 cout << "The tich cua Box1 la: " << thetich <<endl;

 // thetich of box 2
 thetich = Box2.chieucao * Box2.chieudai * Box2.chieurong;
 cout << "The tich cua Box2 la: " << thetich <<endl;
 return 0;
}
```

Biên dịch và chạy chương trình C++ trên sẽ cho kết quả sau:

```
The tich cua Box1 la: 14.85
The tich cua Box2 la: 33.12

--------------------------------
```

Điểm quan trọng cần nhớ là: các thành viên private và protected không thể được truy cập một cách trực tiếp bởi sử dụng toán tử truy cập thành viên trự tiếp này. Chúng ta sẽ học cách các thành viên private và protected có thể được truy cập.

1.1.4. Chi tiết về Lớp & Đối tượng trong C++

Lúc này, bạn đã hiểu khá cơ bản về Lớp và Đối tượng trong C++. Dưới đây là các khái niệm khá thú vị liên quan tới Lớp và Đối tượng trong C++ mà bạn cần quan tâm. Bạn click vào link để xem chi tiết.

Khái niệm	Miêu tả
Hàm thành viên lớp trong C++	Một hàm thành viên của một lớp là một hàm có định nghĩa của nó hoặc prototype của nó bên trong định nghĩa lớp giống như bất kỳ biến nào khác
Access Modifier cho lớp trong C++	Một thành viên lớp có thể được định nghĩa là public, private hoặc protected. Theo mặc định, các thành viên sẽ là private
Constructor & Destructor trong C++	Một class constructor là một hàm đặc biệt trong một lớp mà được gọi khi một đối tượng mới của lớp đó được tạo. Một class destructor cũng là một hàm đặc biệt mà được gọi khi đối tượng đã tạo bị hủy
Copy Constructor trong C++	copy constructor là một constructor mà tạo một đối tượng bằng việc khởi tạo nó với một đối tượng của cùng lớp đó, mà đã được tạo trước đó
Hàm Friend trong C++	Hàm *friend* được cho phép truy cập tới các thành viên là private và protected của một lớp

Hàm Inline trong C++	Với một hàm inline, trình biên dịch cố gắng mở rộng code trong thân hàm thế chỗ cho một lời gọi tới hàm đó
Con trỏ this trong C++	Mỗi đối tượng có một con trỏ **this** đặc biệt, mà trỏ tới chính đối tượng đó
Con trỏ tới lớp trong C++	Một con trỏ tới một lớp được thực hiện giống hệt như cách một con trỏ tới một cấu trúc. Thực ra, một lớp là một cấu trúc với các hàm trong nó
Thành viên Static trong C++	Cả các thành viên dữ liệu (data member) và các thành viên hàm (function member) có thể được khai báo là static

Một trong những khái niệm quan trọng nhất trong lập trình hướng đối tượng là **Tính kế thừa (Inheritance)**. Tính kế thừa cho phép chúng ta định nghĩa một lớp trong điều kiện một lớp khác, mà làm cho nó dễ dàng hơn để tạo và duy trì một ứng dụng. Điều này cũng cung cấp một cơ hội để tái sử dụng tính năng code và thời gian thực thi nhanh hơn.

Khi tạo một lớp, thay vì viết toàn bộ các thành viên dữ liệu và các hàm thành viên mới, lập trình viên có thể nên kế thừa các thành viên của một lớp đang tồn tại. Lớp đang tồn tại này được gọi là **Base Class - lớp cơ sở**, và lớp mới được xem như là **Derived Class – lớp thừa kế**.

Ý tưởng của tính kế thừa triển khai mối quan hệ **IS A** (Là Một). Ví dụ: mammal IS-A animal, dog IS-A mammal, vì thế dog IS-A animal và ....

1.1.5. Lớp cơ sở (Base Class) và Lớp thừa kế (Derived Class) trong C++

Một lớp có thể được kế thừa từ hơn một lớp khác, nghĩa là, nó có thể kế thừa dữ liệu và hàm từ nhiều lớp cơ sở. Để định nghĩa một lớp kế thừa

(Derived Class), chúng ta sử dụng một danh sách để xác định các lớp cơ sở. Danh sách này liệt kê một hoặc nhiều lớp cơ sở và có form sau:

```
class lop_ke_thua: access_modifier lop_co_so
```

Ở đây, *access_modifier* là **public, protected** hoặc **private**, và *lop_co_so* là tên của lớp đã được định nghĩa trước đó. Nếu access_modifier không được sử dụng, thì mặc định là private.

Xét ví dụ sau với **Hinh** là lớp cơ sở và **HinhChuNhat** là lớp kế thừa:

```
#include <iostream>

using namespace std;

// lop co so: Hinh
class Hinh
{
 public:
 void setChieuRong(int rong)
 {
 chieurong = rong;
 }
 void setChieuCao(int cao)
 {
 chieucao = cao;
 }
 protected:
 int chieurong;
 int chieucao;
};
```

```
// day la lop ke thua: HinhChuNhat
class HinhChuNhat: public Hinh
{
 public:
 int tinhDienTich()
 {
 return chieurong * chieucao;
 }
};

int main(void)
{
 HinhChuNhat Hcn;

 Hcn.setChieuRong(14);
 Hcn.setChieuCao(30);

 // in dien tich cua doi tuong.
 cout << "Tong dien tich la: " << Hcn.tinhDienTich() << endl;

 return 0;
}
```

Biên dịch và chạy chương trình C++ trên sẽ cho kết quả sau:

```
Tong dien tich la: 420

--------------------------------
```

1.1.6. Điều khiển truy cập và Tính kế thừa trong C++

Một lớp kế thừa có thể truy cập tất cả thành viên không phải là private của lớp cơ sở của nó. Vì thế, các thành viên lớp cơ sở, mà là hạn chế truy cập

tới các hàm thành viên của lớp kế thừa, nên được khai báo là private trong lớp cơ sở.

Chúng ta tổng kết các kiểu truy cập khác nhau, tương ứng với ai đó có thể truy cập chúng như sau:

Truy cập	**public**	**protected**	**private**
Trong cùng lớp	Có	Có	Có
Lớp kế thừa	Có	Có	Không
Bên ngoài lớp	Có	Không	Không

Một lớp kế thừa (Derived Class) sẽ kế thừa tất cả các phương thức của lớp cơ sở, ngoại trừ:

- Constructor, destructor và copy constructor của lớp cơ sở.
- Overloaded operator (toán tử nạp chồng) của lớp cơ sở.
- Hàm friend của lớp cơ sở.

1.1.7. Kiểu kế thừa trong C++

Khi kế thừa từ một lớp cơ sở, lớp cơ sở đó có thể được kế thừa thông qua kiểu kế thừa là**public, protected** hoặc **private**. Kiểu kế thừa trong C++ được xác định bởi Access-specifier đã được giải thíc ở trên.

Chúng ta hiếm khi sử dụng kiểu kế thừa **protected** hoặc **private**, nhưng kiểu kế thừa**public** thì được sử dụng phổ biến hơn. Trong khi sử dụng các kiểu kế thừa khác sau, bạn nên ghi nhớ các quy tắc sau:

- **Kiểu kế thừa Public:** Khi kế thừa từ một lớp cơ sở là **public**, thì các thành viên **public** của lớp cơ sở trở thành các thành viên **public** của lớp

kế thừa; và các thành viên **protected** của lớp có sở trở thành các thành viên **protected** của lớp kế thừa. Một thành viên là **private** của lớp cơ sở là không bao giờ có thể được truy cập trực tiếp từ một lớp kế thừa, nhưng có thể truy cập thông qua các lời gọi tới các thành viên **public** và **protected** của lớp cơ sở đó.

- **Kiểu kế thừa protected:** Khi kế thừa từ một lớp cơ sở là **protected**, thì các thành viên **public** và **protected** của lớp cơ sở trở thành các thành viên **protected** của lớp kế thừa

- **Kiểu kế thừa private:** Khi kế thừa từ một lớp cơ sở là **private**, thì các thành viên**public** và **protected** của lớp cơ sở trở thành các thành viên **private** của lớp kế thừa

1.1.8. Đa kế thừa trong C++

Một lớp trong C++ có thể kế thừa các thành viên từ nhiều lớp, và đây là cú pháp:

```
class lop_ke_thua: access_modifier lop_co_so_1, access_modifier lop_co_so_2 ...
```

Tại đây, access_modifier là **public, protected** hoặc **private** và sẽ được cung cấp cho mỗi lớp cơ sở, và chúng sẽ được phân biệt với nhau bởi dấu phảy như trên. Bạn thử ví dụ sau:

```
#include <iostream>

using namespace std;

// lop co so: Hinh
class Hinh
{
 public:
```

```
 void setChieuRong(int rong)
 {
 chieurong = rong;
 }
 void setChieuCao(int cao)
 {
 chieucao = cao;
 }
 protected:
 int chieurong;
 int chieucao;
};

// lop co so: ChiPhiSonMau
class ChiPhiSonMau
{
 public:
 int tinhChiPhi(int dientich)
 {
 return dientich * 300000;
 }
};

// day la lop ke thua: HinhChuNhat
class HinhChuNhat: public Hinh, public ChiPhiSonMau
{
 public:
 int tinhDienTich()
 {
```

```
 return chieurong * chieucao;
 }
};

int main(void)
{
 HinhChuNhat Hcn;
 int dientich;

 Hcn.setChieuRong(14);
 Hcn.setChieuCao(30);

 dientich = Hcn.tinhDienTich();

 // in dien tich cua doi tuong.
 cout << "Tong dien tich la: " << Hcn.tinhDienTich() << " m2." << endl;

 // in tong chi phi de son mau
 cout << "Tong chi phi de son mau la: " << Hcn.tinhChiPhi(dientich) << "
VND." << endl;

 return 0;
}
```

Biên dịch và chạy chương trình C++ trên sẽ cho kết quả sau:

```
Tong dien tich la: 420 m2.
Tong chi phi de son mau la: 126000000 VND.

--------------------------------
```

CHƯƠNG 2. THIẾT KẾ CÁC LỚP ĐỐI TƯỢNG

Chương trình bao gồm 3 lớp đối tượng: Người, Thí Sinh, Danh Sách. Trong đó, lớp Thí Sinh thừa kế từ lớp Người.

2.1. Lớp Người

- Tên lớp: Nguoi
- Thuộc tính:
  - Số chứng minh thư : cmt (int)
  - Họ tên : hoten (string)
  - Ngày sinh : ngaysinh (int)
  - Tháng sinh : thangsinh (int)
  - Năm sinh : namsinh (int)
  - Quê quán : que (string)
  - Giới tính : gioitinh (bool)
- Phương thức:
  - Khởi tạo : Nguoi()
  - Hủy bỏ : ~Nguoi()
  - Nhập dữ liệu : void nhap()
  - Xuất dữ liệu : void xuat()
  - Lấy giá trị chứng minh thư : int getcmt()
  - Lấy giá trị họ tên : string gethoten()

2.2. Lớp Thí Sinh

- Tên lớp: ThiSinh
- Thừa kế: từ lớp Nguoi
- Thuộc tính:
  - Số báo danh : sbd (int)
  - Điểm toán : diemT (float)
  - Điểm lý : diemL (float)
  - Điểm hóa : diemH (float)
- Phương thức:
  - Khởi tạo : ThiSinh()
  - Hủy bỏ : ~ThiSinh()
  - Nhập dữ liệu : void nhapts()
  - Xuất dữ liệu : void xuatts()

- Lấy giá trị tổng điểm : float gettd()

2.3. Lớp Danh Sách

- Tên lớp: DS
- Thuộc tính:
  - Số lượng các thí sinh : spt (int)
  - Danh sách các thí sinh : a (mảng)
- Phương thức:
  - Khỏi tạo : DS()
  - Hủy bỏ : ~DS()
  - Nhập dữ liệu : void nhapDS()
  - Xuất dữ liệu : void inDS()

CHƯƠNG 3. CÀI ĐẶT CÁC LỚP

3.1. Lớp Người

```
class Nguoi
{
    protected:
        int cmt;
        string hoten;
        int ngaysinh;
        int thangsinh;
        int namsinh;
        string que;
        bool gioitinh;
    public:
        Nguoi();
        ~Nguoi();
        void nhap();
        void xuat();
        void setcmt(int a_cmt);
```

```
        int getcmt();
        void sethoten(string a_hoten);
        string gethoten();
};
Nguoi::Nguoi()
{
    ngaysinh = 0;
    thangsinh = 0;
    namsinh = 0;
    cmt = 0;
    hoten = "";
    que = "";
    gioitinh = 1;
}
Nguoi::~Nguoi()
{
}
void Nguoi::nhap()
{
    cout << "Nhap so chung minh thu: ";
    cin >> cmt;
    cout << "Nhap ho ten: ";
    fflush(stdin);
    getline(cin,hoten,'\n');
    cout << "Nhap ngay sinh: ";
    cin >> ngaysinh >> thangsinh >> namsinh;
    cout << "Nhap que quan: ";
    fflush(stdin);
    getline(cin,que,'\n');
```

```
        cout << "Nhap gioi tinh (0-nu, 1-nam): ";
        cin >> gioitinh;
}
void Nguoi::xuat()
{
        cout<< "So chung minh thu: " << cmt << endl
                << "Ho ten : " << hoten << endl
                << "Ngay sinh : " << ngaysinh << "/" << thangsinh << "/"
<< namsinh << endl
                << "Que quan : " << que << endl
                << "Gioi tinh : " << (gioitinh == true ? "nam" : "nu") <<
endl;
}
int Nguoi::getcmt()
{
        return cmt;
}
string Nguoi::gethoten()
{
        return hoten;
}
```

3.2. Lớp Thí Sinh

```
class ThiSinh: public Nguoi
{
        //friend class Nguoi;
        private:
                int sbd;
                float diemT;
```

```
		float diemL;
		float diemH;
	public:
		ThiSinh();
		ThiSinh(int a_sbd, float a_diemT, float a_diemL, float
a_diemH);
		~ThiSinh();
		void nhapts();
		void xuatts();
		float gettd();
};

ThiSinh::ThiSinh()
{
	sbd = 0;
	diemT = 0.0;
	diemL = 0.0;
	diemH = 0.0;
}

ThiSinh::ThiSinh(int a_sbd, float a_diemT, float a_diemL, float
a_diemH)
{
	sbd = a_sbd;
	diemT = a_diemT;
	diemL = a_diemL;
	diemH = a_diemH;
}

ThiSinh::~ThiSinh()
{
}

void ThiSinh::nhapts()
{
	Nguoi::nhap();
	cout << "Nhap so bao danh: ";
```

```
        cin >> sbd;
        cout << "Nhap diem mon toan: ";
        cin >> diemT;
        cout << "Nhap diem mon ly: ";
        cin >> diemL;
        cout << "Nhap diem mon hoa: ";
        cin >> diemH;
}

void ThiSinh::xuatts()
{
        Nguoi::xuat();
        cout<< "So bao danh: " << sbd << endl
                << "Diem mon toan: " << diemT << endl
                << "Diem mon ly: " << diemL << endl
                << "Diem mon hoa: " << diemH << endl;
}

float ThiSinh::gettd()
{
        return (diemT + diemL + diemH);
}
```

3.3. Lớp Danh Sách

```
class DS
{
        private:
                int spt;
                ThiSinh a[100];

        public:
                DS();
                ~DS();
                void nhapDS();
                void inDS();
                void them();
                void xoa();
```

```
		void timkiem();
};

DS::DS()
{
	spt = 0;
}

DS::~DS()
{
}

void DS::nhapDS()
{
	do
	{
		cout << "nhap so thi sinh: ";
		cin >> spt;
	} while (spt <= 0);
	cout << "+----------NHAP THONG TIN THI
SINH----------+" << endl;
	for (int i = 0; i<spt; i++)
		{
			cout << "\nNhap thong tin cua thi sinh thu " << i+1
<< ": " << endl;
			a[i].nhapts();
		}
}

void DS::inDS()
{
	cout << "+----------DS THONG TIN THI SINH----------+"
<< endl;
	for(int i=0; i<spt; i++)
	{
		cout << "\nThong tin cua thi sinh thu " << i+1 << ": " <<
endl;
```

```
                a[i].xuatts();
        }
}

void DS::them()
{
        ThiSinh *ts = new ThiSinh();
        ts->nhapts();
        a[spt] = *ts;
        spt++;
}

void DS::xoa()
{
        int vt;
        cout << "Nhap vi tri can xoa: ";
        cin >> vt;
        vt--;
        if (vt<0 && vt>spt) return;
        for (int i = vt; i<spt; i++)
                {
                        a[i] = a[i + 1];
                }
        spt--;
}

void DS::timkiem()
{
        float td;
        cout << "Nhap tong diem gioi han: ";
        cin >> td;
        for (int i = 0; i < spt; i++)
                {
                        if (a[i].gettd() >= td)
                                {
                                        cout << a[i].gethoten() << ", Tong
diem: " << a[i].gettd() << endl;
```

```
                    }
                }
        }
```

3.4. Hàm chính (main())

```
main()
{
        DS p;
        int menu;
        while (true)
        {
                //system("cls");
                cout << "\n CHUONG TRINH QUAN LY THI SINH";
                cout << "\n 1.Nhap danh sach thi sinh";
                cout << "\n 2.Hien thi ds thi sinh";
                cout << "\n 3.Them mot thi sinh";
                cout << "\n 4.Xoa mot thi sinh";
                cout << "\n 5.Tim thi sinh co tong diem lon hon so cho truoc";
                cout << "\n Phim 0 thoat khoi chuong trinh ";
                cout << "\n Chon: ";
                cin >> menu;

                switch (menu)
                {
                        case 1:system("cls"); p.nhapDS(); break;
                        case 2:system("cls"); p.inDS(); break;
                        case 3:system("cls"); p.them(); break;
                        case 4:system("cls"); p.xoa(); break;
                        case 5:system("cls"); p.timkiem(); break;
                        case 0:return 0; break;
                }
        }
        return 0;
}
```

KẾT LUẬN

- Nhóm đã giải quyết được những gì theo yêu cầu của bài tập lớn
- Những điểm gì nhóm chưa làm được
- Hướng phát triển

TÀI LIỆU THAM KHẢO

1) Phạm Văn Ất, **Lập trình Hướng đối tượng với C++**, Nhà xuất bản KHKT, 2015

2) Đỗ Xuân Lôi, **Cấu trúc dữ liệu và giải thuật**, Nhà xuất bản ĐH QG, 2016

3) https://www.codeproject.com/Articles/6819/SendKeys-in-C