**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM  
KHOA CÔNG NGHỆ THÔNG TIN**

**-----\*\*\*-----**

****

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN “LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG”**

***Đề tài:***

***XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ DANH SÁCH SINH VIÊN***

***GVHD: ThS. Nguyễn Hạnh Phúc***

***Sinh viên thực hiện: Đồng Xuân Việt***

***Đàm Quang Hà***

***Hải Phòng, tháng 11 năm 2016***

**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN KHOA HỌC MÁY TÍNH**

**-----\*\*\*-----**

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**Mã đề tài: 11**

1. **Tên đề tài**

***Xây dựng chương trình quản lý danh sách sinh viên***

1. **Mục đích**

Sử dụng danh sách liên kết đơn để xây dựng chương trình quản lý danh sách sinh viên

1. **Công việc cần thực hiện**

* Xây dựng chương trình quản lý danh sách sinh viên thực hiện các công việc sau:
  + Add (student, p): Bổ sung thêm một sinh viên student vào một vị trí p bất kỳ trong danh sách (phần tử đầu tiên có vị trí là 1, và với p=0 tức là bổ sung vào cuối danh sách; Không cho phép 2 sinh viên có cùng MSSV trong danh sách).
  + Remove (mssv): Loại bỏ một sinh viên với MSSV mssv cho trước.
  + Remove (p): Loại bỏ sinh viên ở vị trí thứ p bất kỳ trong danh sách (phần tử đầu tiên có vị trí là 1).
  + Search(s): Tìm một sinh viên theo MSSV hoặc theo tên (với một thông tin s vào thì trước tiên tìm sinh viên có MSSV = s, nếu không thấy thì tìm sinh viên có Họ tên = s.
  + Print (): In ra nội dung danh sách sinh viên.
  + Save (filename): Lưu nội dung của danh sách sinh viên ra một tệp có tên filename.
  + Load (filename): load nội dung danh sách từ một tệp có tên filename.
  + Sort (type): sắp xếp danh sách theo MSSV (nếu type=ID) hoặc theo Họ tên (nếu type=NAME).
* Làm báo cáo bài tập lớn
* Bảo vệ bài tập lớn

1. **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: 18/11/2016

1. **Tài liệu tham khảo**

Bài giảng Lập trình hướng đối tượng – Th.S Nguyễn Hạnh Phúc – Trường Đại học Hàng hải Việt Nam.

***Hải Phòng, ngày 18 tháng 11 năm 2016***

NGƯỜI HƯỚNG DẪN

Th.S Nguyễn Hạnh Phúc

**MỤC LỤC**

[DANH MỤC CÁC HÌNH VẼ, BẢNG BIỂU iii](#_Toc467160651)

[DANH MỤC CÁC TỪ VIẾT TẮT iv](#_Toc467160652)

[GIỚI THIỆU 1](#_Toc467160653)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 2](#_Toc467160654)

[1.1 TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG.2](#_Toc467160655)

[1.2 LỚP VÀ ĐỐI TƯỢNG 3](#_Toc467160656)

[1.3 DANH SÁCH LIÊN KẾT 4](#_Toc467160657)

[CHƯƠNG 2: XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN 6](#_Toc467160658)

[2.1. CÁC HÀM THÀNH PHẦN QUAN TRỌNG 6](#_Toc467160659)

[2.2. KẾT QUẢ SAU KHI CHẠY CHƯƠNG TRÌNH 23](#_Toc467160660)

[KẾT LUẬN 31](#_Toc467160661)

[TÀI LIỆU THAM KHẢO 32](#_Toc467160662)

# DANH MỤC CÁC HÌNH VẼ, BẢNG BIỂU

|  |  |
| --- | --- |
| **Hình vẽ** | **Trang** |
| Hình 1. Mô hình tương tác giữa các đối tượng với nhau bằng cách gửi thông điệp | 2 |
| Hình 2. Mô hình danh sách liên kết đơn | 5 |

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ** | **Ý nghĩa** |
| OOP | Object Oriented Progamming |

# GIỚI THIỆU

1, Tính ứng dụng của chương trình quản lý danh sách sinh viên

Việc xây dựng 1 chương trình quản lý danh sách sinh viên có rất nhiều ứng dụng trong thực tế hiện nay. Nếu lúc trước việc lưu trữ chủ yếu được tiến hành bởi phương pháp thủ công thì giờ đây công việc đó hoàn toàn có thể được thay thế bởi 1 chiếc máy tính. Điều này sẽ thuận lợi hơn rất nhiều so với việc cách lưu trữ trước kia, những công việc tìm kiếm sẽ dễ dàng hơn rất nhiều, thuận tiện cho việc bổ sung thông tin sinh viên và quản lý danh sách sinh viên một cách có tổ chức.

2, Phương pháp nghiên cứu

Nhằm xây dựng một chương trình quản lý danh sách sinh viên một cách vừa hiệu quả, vừa ít tốn kém thì cần phải tập trung nghiên cứu các các vấn đề sau:

+ Vấn đề sử dụng danh sách liên kết đơn để xây dựng chương trình quản lý danh sách sinh viên.

+ Xây dựng chương trình trên theo phương pháp hướng đối tượng.

+ Nghiên cứu, sử dụng ngôn ngữ lập trình C++ và công cụ lập trình DevCpp để tiến hành xây dựng chương trình theo đúng yêu cầu.

3, Kết quả đạt được

Bước đầu đã xây dựng được một chương trình quản lý danh sách sinh viên bằng cách sử dụng danh sách liên kết đơn.

Chỉ ra được những thuận lợi cũng như khó khăn khi sử dụng danh sách liên kết đơn thay vì sử dụng mảng tĩnh để xây dựng chương trình.

Chương trình được xây dựng bằng phương pháp hướng đối tượng và được lập trình trên công cụ Devcpp 5.11

Ngày này, do sự phát triển của khoa học kỹ thuật, các ngôn ngữ lập trình cũng như các công cụ lập trình mới đang ngày càng được ra đời nhiều hơn, chính vì thế việc xây dựng chương trình quản lý danh sách sinh viên trên các ngôn ngữ và công cụ mới này chắc chắn sẽ hiệu quả hơn rất nhiều. Tuy nhiên khó có thể phủ nhận việc sử dụng danh sách liên kết đơn vào việc quản lý danh sách sinh viên là khá hiệu quả và ít tốn kém chi phí so với các ngôn ngữ bậc cao khác.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

* Trong lập trình hướng đối tượng, mỗi chương trình sẽ có một số các đối tượng (object) có thể tương tác với nhau, những đối tượng này thuộc các lớp khác nhau và mỗi đối tượng tự quản lý các dữ liệu riêng của chúng.

|  |
| --- |
| **PROGRAM**  **Object 2 data**  **Object 4 data**  **Object 3 data**  **Object 1 data** |

*Hình 1.1: Các đối tượng tương tác với nhau bằng cách gửi thông điệp*

* Chương trình chính sẽ bao gồm một số đối tượng là thể hiện của các lớp, các đối tượng này tương tác với nhau để thực hiện các chức năng của chương trình
* Các lớp trong lập trình hướng đối tượng có thể được xem như là một kiểu dữ liệu người dùng tự định nghĩa trong các chương trình có cấu trúc cùng với việc tích hợp các toán tử và dữ liệu trên các kiểu dữ liệu tự định nghĩa đó.
* Các ưu điểm của lập trình hướng đối tượng:
  + Giải quyết được nhiều nhược điểm tồn tại trong lập trình có cấu trúc, xảy ra ít lỗi hơn đồng thời việc khắc phục lỗi cũng dễ dàng hơn rất nhiều. Đây được xem là ưu điểm lớn nhất của lập trình hướng đối tượng.
  + Việc thay đổi các cài đặt chi tiết bên dưới trong lập trình hướng đối tượng không làm ảnh hướng tới các phần khác của chương trình do đó việc mở rộng một chương trình sẽ dễ hàng được thực hiện đồng thời cũng làm giảm thời gian cần thiết để phát triển phân mềm.
  + Với khái niệm kế thừa trong lập trình hướng đối tượng, các lập trình viên có thể xây dựng các chương trình từ các phần mềm sẵn có.
  + Một được trình OOP được viết trên một hệ thống nền có thể chạy trên nhiều hệ thống nền khác nhau.
  + Các hệ thống xây dựng bằng OOP có hiệu năng cao.

## LỚP VÀ ĐỐI TƯỢNG

### LỚP

* Trong một chương trình hướng đối tượng sẽ bao gồm các đối tượng, các đối tượng này là đại diện cho các đối tượng trong thực tế. Có thể coi khái niệm đối tượng trong lập trình hướng đối tượng chính là các kiểu dữ liệu trong các ngôn ngữ lập trình có cấu trúc.
* Mỗi một đối tượng có các dữ liệu riêng của chúng, được gọi là các thành phần dữ liệu (data member). Các toán tử thao tác trên các dữ liệu này được gọi là các hàm thành phần (member function).

### ĐỐI TƯỢNG

* Mỗi đối tượng là sự thể hiện của một lớp, có thể nói lớp là đại diện cho các đối tượng có các hàm thành phần (member function) giống nhau và các thành phần dữ liệu (data member) cùng kiểu.
* Lớp là sự trừu tượng hóa của các khái niệm đối tượng, tuy nhiên lớp không phải là một kiểu dữ liệu trừu tượng ADT, nó chỉ là một cài đặt của một đặc tả kiểu dữ liệu trừu tượng ADT.
* Các đối tượng thuộc cùng một lớp có thể chia sẻ dữ liệu dùng chung.

## DANH SÁCH LIÊN KẾT

### GIỚI THIỆU

* Danh sách liên kết (Linked list) là một cấu trúc dữ liệu bao gồm tập hợp các nút (Node) cùng kiểu dữ liệu. Trong đó mỗi nút bao gồm một phần dữ liệu và phần còn lại là liên kết trỏ đến nút kế tiếp trong danh sách.
* Danh sách liên kết có nhiều loại, bao gồm: Danh sách liên kết đơn, danh sách liên kết kép, danh sách liên kết vòng…
* Danh sách liên kết là một trong những cấu trúc dữ liệu đơn giản và được sử dụng phổ biến.
* So sánh ưu, nhược điểm của danh sách liên kết và danh sách đặc:

|  |  |  |
| --- | --- | --- |
| **Ưu, nhược điểm** | **Danh sách đặc** | **Danh sách liên kết** |
| Phải xác định số phần tử | X |  |
| Bộ nhớ có thể tăng hoặc giảm |  | X |
| Thời gian xem và xóa rất nhanh |  | X |
| Duyệt danh sách nhanh, truy cập nhanh | X |  |
| Tốn vùng nhớ cho con trỏ Next |  | X |
| Lưu trữ lượng dữ liệu lớn |  | X |
| Dễ dàng thêm, xóa dữ liệu |  | X |
| Không cho phép truy cập ngẫu nhiên |  | X |

### DANH SÁCH LIÊN KẾT ĐƠN

* Danh sách liên kết đơn là danh sách các nút (Node) liên kết 1 chiều với nhau. Mỗi nút là một bản ghi bao gồm 2 trường:
  + Trường thứ nhất: Chứa dữ liệu của nút.
  + Trường thứ 2: Chứa liên kết tới nút kế tiếp hay chính là địa chỉ của nút kế tiếp.
* Nút đầu tiên của danh sách liên kết đơn được trỏ bởi con trỏ head (được gọi là địa chỉ của danh sách liên kết).
* Nút cuối dùng trong danh sách được trỏ bởi con trỏ tail và trỏ tới NULL.

|  |
| --- |
| **Head Tail**  **NULL**  An  A2  A1   * **…** |

*Hình 2: Mô hình danh sách liên kết đơn*

# CHƯƠNG 2: XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN

## CÁC HÀM THÀNH PHẦN QUAN TRỌNG

### Hàm tạo Class SV

|  |
| --- |
| **class** Node;  **class** DSSV;  **class** SV  {  **int** msv, khoa;  string hoten, ngaysinh, gioitinh, tenlop;  **public**:  SV()  {  msv=0;  hoten="";  ngaysinh="";  gioitinh="";  tenlop="";  khoa=0;  }  **friend class** Node;  **friend class** DSSV;  }; |

### 2.1.2. Hàm tạo Class Node

|  |
| --- |
| **class** DSSV;  **class** Node  {  SV data;  Node\*next;  **public**:  Node(SV dt, Node\*next1)  {  data=dt;  next=next1;  }  **friend class** DSSV;  }; |

### 2.1.3. Hàm tạo Class DSSV

|  |
| --- |
| **class** DSSV  {  Node\*head;  **int** dem;  **public**:  DSSV()  {  dem=0;  head=NULL;  }  ~DSSV();  **void** Input\_SV(SV&d);  **int** Count\_SPT(**int** k);  **void** Add\_Head(SV d);  **void** Remove\_Head();  **void** Add\_Tail(SV d);  **void** Add\_VT(SV d, **int** vt);  **void** Remove\_VT(**int** vt);  **bool** IsExist(**int** d);  **void** Remove\_MSV(**int** k);  **void** Remove\_Same();  **int** Search\_MSV(**int** mssv);  **int** Search\_Ten(string ten);  **int** empty()**const**;  **void** Printall();  **void** Save\_File();  **void** Load\_File();  **void** Sort(string type);  }; |

### 2.1.4. Hàm thêm phần tử theo vị trí

|  |
| --- |
| **void** DSSV::Add\_VT(SV d, **int** vt)  {  Node\*p=head;  **int** i=1;  **if**(vt<0||vt>dem+1)  cout<<"VT chen k hop le !";  **else** **if**(vt==1)  Add\_Head(d);  **else** **if**(vt==0)  {  **if**(IsExist(d.msv)==**false**)  Add\_Tail(d);  }  **else**  {  Node\*q=**new** Node(d, head);  **while**(p!=NULL**&&**i!=vt-1)  {  i++;  p=p->next;  }  q->next=p->next;  p->next=q;  dem++;  }  } |

### 2.1.5. Hàm xóa phần tử theo mã SV

|  |
| --- |
| **void** DSSV::Remove\_MSV(**int** k)  {  Node \*p=head;  Node\*temp;  **if**(head==NULL||IsExist(k)==**false**)  {  cout<<"Khong tim thay !"<<endl;  }  **else**  {  if(p->data.msv==k)  {  head=p->next;  **delete** p;  }  **else**{  **while**(p!=NULL**&&**p->data.msv!=k)  {  temp=p;  p=p->next;  }  temp->next=p->next;  **delete** p;  }  dem--;  }  } |

### 2.1.6. Hàm xóa phần tử tại vị trí bất kỳ

|  |
| --- |
| **void** DSSV::Remove\_VT(**int** vt)  {  Node \*p=head;  **int** i=1;  **if** (vt<1 || vt>dem)  cout<<"Vi tri xoa khong hop le !"<<endl;  **else**  {  **if** (vt==1) Remove\_Head();  **else**  {  **while** (p != NULL **&&** i != vt-1)  {  p=p->next;  i++;  }  p->next = p->next->next;  dem--;  }  }  } |

### 2.1.7. Hàm xóa các phần tử trùng nhau (đảm bảo danh sách không có 2 sinh viên có cùng mã sinh viên)

|  |
| --- |
| **void** DSSV::Remove\_Same()  {  Node\*p=head;  Node\*q, \*t;  **while**(p!=NULL)  {  t=p;  q=p->next;  **while**(q!=NULL)  {  **if**(q->data.msv==p->data.msv)  {  Remove\_MSV(p->data.msv);  q=t->next;  }  **else**  {  t=q;  q=q->next;  }  }  p=p->next;  }  } |

### 2.1.8. Hàm tìm kiếm sinh viên theo mã sinh viên

|  |
| --- |
| **int** DSSV::Search\_MSV(**int** mssv)  {  Node\*p=head;  **int** i=1;  **while**(p!=NULL**&&**p->data.msv!=mssv)  {  p=p->next;  i++;  }  **if**(p!=NULL)  **return** i;  **else**  **return** 0;  } |

### 2.1.9. Hàm tìm kiếm sinh viên theo tên

|  |
| --- |
| **int** DSSV::Search\_Ten(string ten\_tk)  {    Node\*p=head;  **int** i=1;  **while**(p!=NULL**&&**p->data.hoten!=ten\_tk)  {  p=p->next;  i++;  }  **if**(p!=NULL)  **return** i;  **else**  **return** 0;  } |

### 2.1.10. Hàm in danh sách sinh viên

|  |
| --- |
| **void** DSSV::Printall()  {  cout<<"\n ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿"  <<"\n ³So SV trong DS: "<<dem<<setw(15-Count\_SPT(dem)+1)<<"³"  <<"\n ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ";  **int** i=1;  **for**(Node\*p=head;p!=NULL;p=p->next)  {  cout<<"\n ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿"  <<"\n ³Sinh Vien "<<i<<setw(21-Count\_SPT(i)+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´"  <<"\n ³MSV: "<<p->data.msv<<setw(26-Count\_SPT(p->data.msv)+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´"  <<"\n ³Ho Ten: "<<p->data.hoten<<setw(23-(p->data.hoten.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´"  <<"\n ³Ngay Sinh: "<<p->data.ngaysinh<<setw(20-(p->data.ngaysinh.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´"  <<"\n ³Gioi Tinh: "<<p->data.gioitinh<<setw(20-(p->data.gioitinh.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´"  <<"\n ³Lop: "<<p->data.tenlop<<setw(26-(p->data.tenlop.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´"  <<"\n ³Khoa: "<<p->data.khoa<<setw(25-Count\_SPT(p->data.khoa)+1)<<"³"  <<"\n ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ";  i++;  }  } |

### 2.1.11. Hàm lưu dữ liệu ra file

|  |
| --- |
| **void** DSSV::Save\_File()  {  ofstream fout1("**sinhvien.txt**");  fout1<<dem<<endl;  **int** i=1;  **for**(Node\*p=head;p!=NULL;p=p->next)  {  fout1<<i<<endl  <<p->data.msv<<endl  <<p->data.hoten<<endl  <<p->data.ngaysinh<<endl  <<p->data.gioitinh<<endl  <<p->data.tenlop<<endl  <<p->data.khoa<<endl;  i++;  }  fout1.close();  } |

### 2.1.12. Hàm đọc dữ liệu từ file

|  |
| --- |
| **void** DSSV::Load\_File()  {  ifstream fin1("**sinhvien.txt**");  fin1>>dem;  cout<<"\n ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿"  <<"\n ³So SV trong DS: "<<dem<<setw(15-Count\_SPT(dem)+1)<<"³"  <<"\n ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ";  **int** i;  **for**(Node\*p=head;p!=NULL;p=p->next)  {  fin1>>i;  cout<<"\n ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿"  <<"\n ³Sinh Vien "<<i<<setw(21-Count\_SPT(i)+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´";  fin1>>p->data.msv;  cout<<"\n ³MSV: "<<p->data.msv<<setw(26-Count\_SPT(p->data.msv)+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´";  fin1.ignore();  getline(fin1, p->data.hoten);  cout<<"\n ³Ho Ten: "<<p->data.hoten<<setw(23-(p->data.hoten.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´";  getline(fin1,p->data.ngaysinh);  cout<<"\n ³Ngay Sinh: "<<p->data.ngaysinh<<setw(20-(p->data.ngaysinh.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´";  getline(fin1,p->data.gioitinh);  cout<<"\n ³Gioi Tinh: "<<p->data.gioitinh<<setw(20-(p->data.gioitinh.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´";  getline(fin1,p->data.tenlop);  cout<<"\n ³Lop: "<<p->data.tenlop<<setw(26-(p->data.tenlop.size())+1)<<"³"  <<"\n ÃÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ´";  fin1>>p->data.khoa;  cout<<"\n ³Khoa: "<<p->data.khoa<<setw(25-Count\_SPT(p->data.khoa)+1)<<"³"  <<"\n ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ";  }  fin1.close();  } |

### 2.1.13. Hàm sắp xếp danh sách theo kiểu sắp xếp

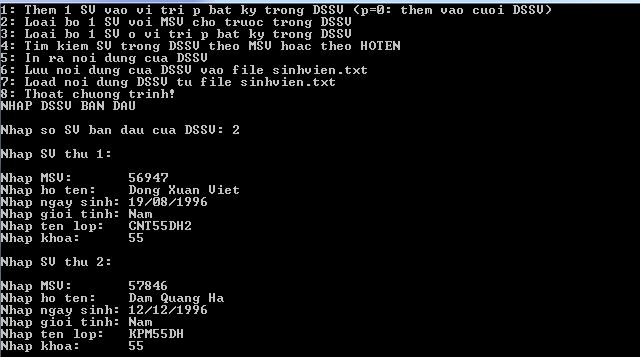
|  |
| --- |
| **void** DSSV::Sort(string type)  {  Node\*p=head, \*q;  **for**(p;p!=NULL;p=p->next)  **for**(q=p->next;q!=NULL;q=q->next)  {  **if**((type=="ID"||type=="id")&&p->data.msv>q->data.msv)  {  SV temp=p->data;  p->data=q->data;  q->data=temp;  }  **else** **if**((type=="NAME"||type=="name")&&  p->data.hoten>q->data.hoten)  {  SV temp=p->data;  p->data=q->data;  q->data=temp;  }  }  } |

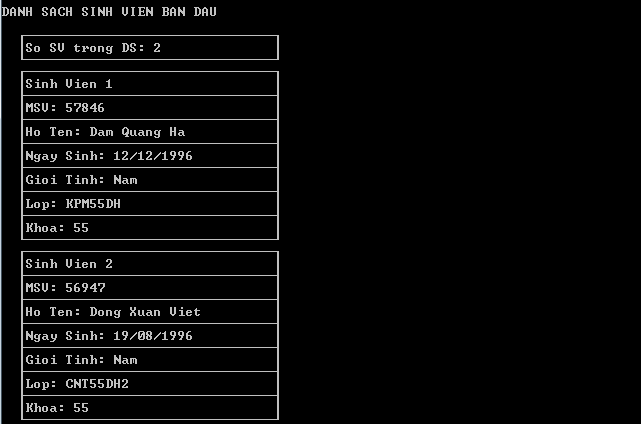
### 2.1.14. Hàm main

|  |
| --- |
| main()  {  SV a;  DSSV L;  **int** lua\_chon;  cout<<"1: Them 1 SV vao vi tri p bat ky trong DSSV (p=0: them vao cuoi DSSV)"<<endl  <<"2: Loai bo 1 SV voi MSV cho truoc trong DSSV"<<endl  <<"3: Loai bo 1 SV o vi tri p bat ky trong DSSV"<<endl  <<"4: Tim kiem SV trong DSSV theo MSV hoac theo HOTEN"<<endl  <<"5: In ra noi dung cua DSSV"<<endl  <<"6: Luu noi dung cua DSSV vao file sinhvien.txt"<<endl  <<"7: Load noi dung DSSV tu file sinhvien.txt"<<endl  <<"8: Sap xep danh sach "<<endl  <<"9: Thoat chuong trinh! "<<endl;  **int** n;  cout<<"NHAP DSSV BAN DAU\n"<<endl;;  cout<<"Nhap so SV ban dau cua DSSV: ";  cin>>n;  **for**(**int** i=1;i<=n;i++)  {  cout<<"\nNhap SV thu "<<i<<": \n\n";  L.Input\_SV(a);  L.Add\_Head(a);  }  cout<<"\nDANH SACH SINH VIEN BAN DAU\n";  L.Remove\_Same();  L.Printall();  **do**{  cout<<"\nBan chon: ";  cin>>lua\_chon;  **switch** (lua\_chon)  {  **case** 1:  **int** p;  cout<<"Nhap vi tri can chen vao DSSV: ";  cin>>p;  L.Input\_SV(a);  L.Add\_VT(a, p);  **break;**  **case** 2:  {  **int** k;  cout<<"Nhap MSV can xoa khoi DSSV: ";  cin>>k;  L.Remove\_MSV(k);  **break;**  }  **case** 3:  {  **int** vtri;  cout<<"Nhap vi tri SV can xoa khoi DSSV: ";  cin>>vtri;  L.Remove\_VT(vtri);  **break;**  }  **case** 4:  {  **int** mssv;  cout<<"Nhap MSV can tim: ";  cin>>mssv;  if(L.Search\_MSV(mssv)!=0)  cout<<"Tim thay SV co ma "<<mssv<<" tai vi tri "<<L.Search\_MSV(mssv) << " trong DS"<<endl;  **else**  {  string ten;  cout<<"Nhap ten can tim kiem: ";  cin.ignore();  getline(cin, ten);  **if**(L.Search\_Ten(ten)!=0)  cout<<"Tim thay SV co ten "<<ten<<" tai vi tri "<< L.Search\_Ten(ten)<<" trong DS"<<endl;  **else**  cout<<"Khong tim thay SV co ten "<<ten<<" trong DS"<<endl;  }  **break;**  }  **case** 5:  {  **if**(L.empty()==0)  {  cout<<"\nDANH SACH SINH VIEN HIEN TAI\n";  L.Remove\_Same();  L.Printall();  }  **else**  cout<<"DSSV rong!";  **break;**  }    **case** 6:  {  L.Save\_File();  **break;**  }    **case** 7:  {  cout<<"\nDSSV LOAD RA TU FILE sinhvien.txt la: \n";  L.Load\_File();  **break;**  }  **case** 8:  {  string type;  cout<<"Nhap kieu muon sap xep: ";  cin.ignore();  getline(cin, type);  L.Sort(type);  L.Printall();  **break;**  }  **case** 9:**break;**  }  }**while**(lua\_chon>0&&lua\_chon<10);  } |

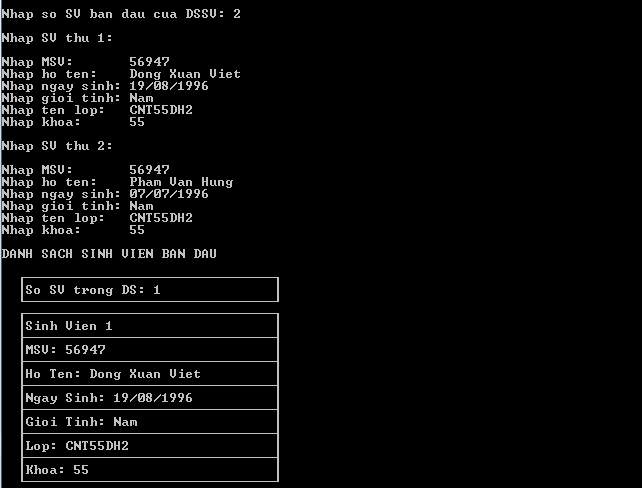
## KẾT QUẢ SAU KHI CHẠY CHƯƠNG TRÌNH

### DANH SÁCH BAN ĐẦU

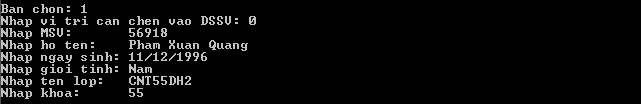


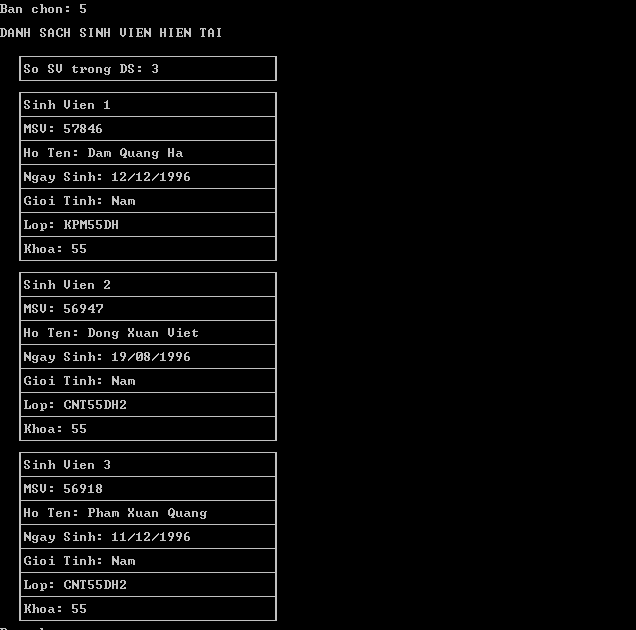


### LOẠI BỎ SINH VIÊN CÓ TRÙNG MÃ SỐ SINH VIÊN TRONG DANH SÁCH

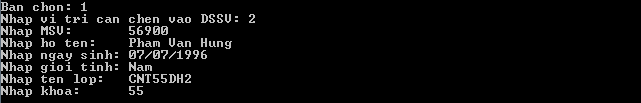


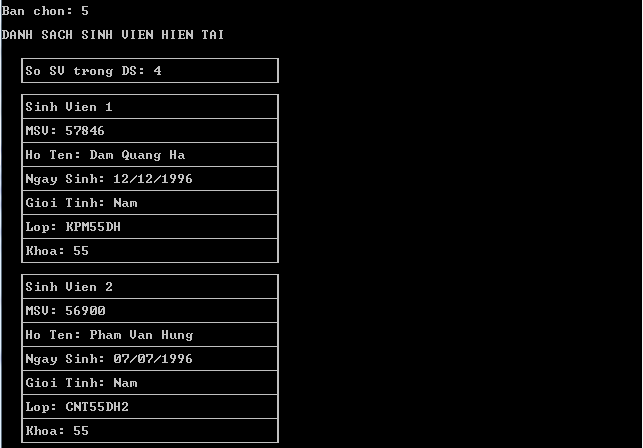
### THÊM 1 PHẦN TỬ VÀO VỊ TRÍ p=0

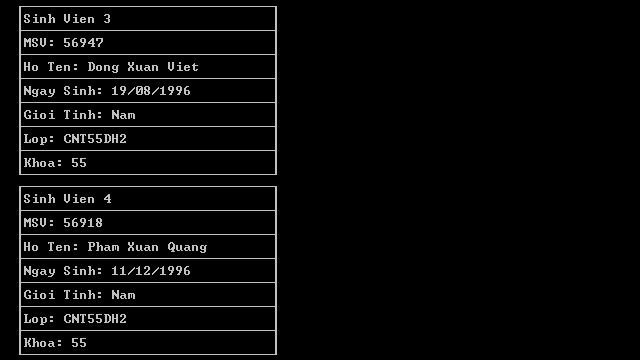




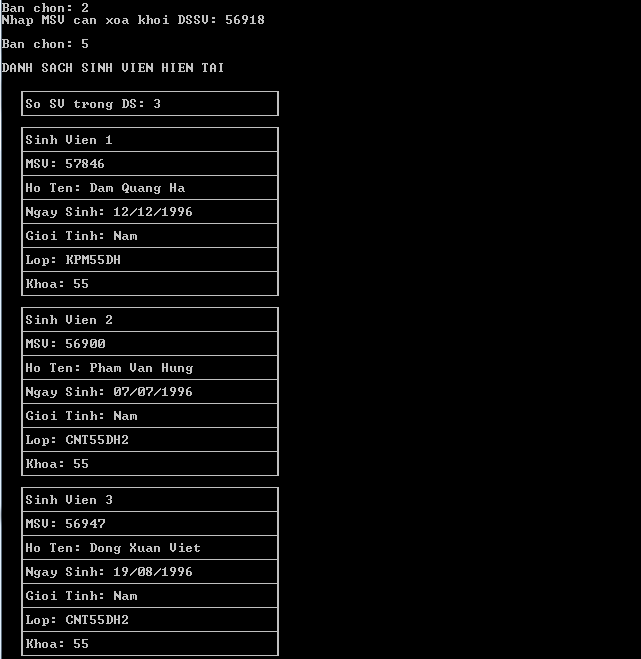
### THÊM 1 PHẦN TỬ VÀO VỊ TRÍ BẤT KỲ KHÁC 0



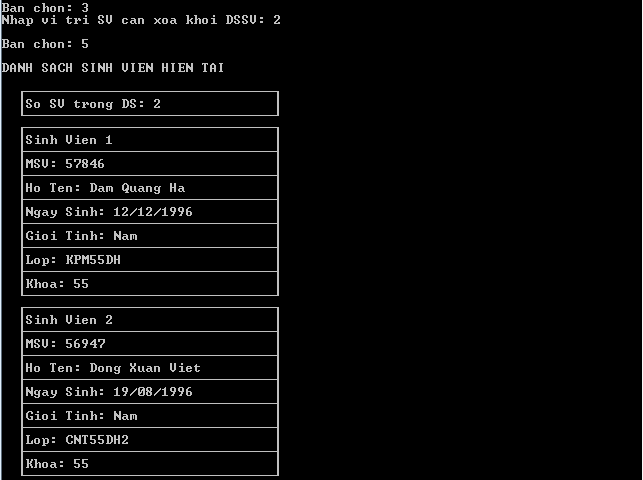




### LOẠI BỎ SINH VIÊN VỚI MÃ SINH VIÊN CHO TRƯỚC



### LOẠI BỎ SINH VIÊN Ở VỊ TRÍ BẤT KỲ TRONG DANH SÁCH



### TÌM KIẾM SINH VIÊN





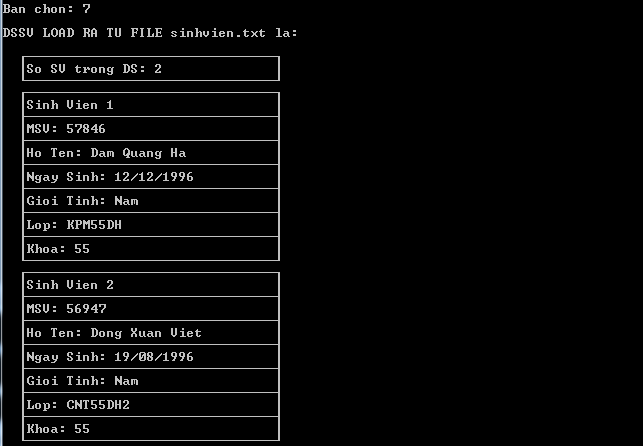


### LƯU NỘI DUNG CỦA DANH SÁCH SINH VIÊN RA FILE





### ĐỌC NỘI DUNG FILE DANH SÁCH SINH VIÊN VỪA GHI



### SẮP XẾP DANH SÁCH THEO KIỂU SẮP XẾP

# KẾT LUẬN

* Việc xây dựng một chương trình quản lý danh sách sinh viên bằng cách sử dụng danh sách liên kết đơn là một phương pháp khá hiệu quả và ít tốt kém.
* Sử dụng danh sách liên kết đơn cho phép tận dụng tối đa bộ nhớ để lưu trữ dữ liệu, đó là điều mà một danh sách đặc không thể làm được.
* Ngoài danh sách liên kết thì cũng có thể sử dụng các cấu trúc dữ liệu khác để tiến hành xây dựng chương trình quản lý sinh viên như trên.

# TÀI LIỆU THAM KHẢO

Bài giảng Lập trình hướng đối tượng – Th.S Nguyễn Hạnh Phúc – Trường Đại học Hàng hải Việt Nam.