

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**BÁO CÁO CUỐI KÌ**

**MÔN HỌC: GIẢI THUẬT VÀ CẤU TRÚC DỮ LIỆU**

**THIẾT KẾ CẤU TRÚC DỮ LIỆU, XÂY DỰNG THUẬT TOÁN VÀ CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN**

**LỚP: ALDS335764\_04**

**NHÓM: 16**

**Sinh viên:**

|  |  |
| --- | --- |
| Nguyễn Minh Tuấn | 20139095 |
| Phan Tấn Quốc | 20139086 |
| GVHD: PGS.TS. Hoàng Văn Dũng |  |

P. Hồ Chí Minh, tháng 5, năm 2022

Sinh viên thực hiện:

|  |  |  |  |
| --- | --- | --- | --- |
| **stt** | **Họ và tên** | **Mã số sinh viên** | **Tỉ lệ đóng góp** |
| 1 | Ngyễn Minh Tuấn | 20139095 | 100% |
| 2 | Phan Tấn Quốc | 20139086 | 100% |

**NHẬN XÉT CỦA GIÁO VIÊN:**

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

Tp. Hồ Chí Minh, ngày ….., tháng 5, năm 2022

Giáo viên chấm điểm

(Ký và ghi họ tên)

Hoàng Văn Dũng

**LỜI CẢM ƠN**

“Để có thể hoàn thành được hoàn chỉnh đề tài và bài báo cáo đã chọn , nhóm 16 chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến giảng viên Hoàng Văn Dũng, người đã trực tiếp hỗ trợ và theo dõi chúng em trong suốt thời gian làm đề tài. Chúng em xin cảm ơn thầy đã đưa ra những lời khuyên, lời nhận xét, những góp ý chân thành từ kinh nghiệm đi trước của mình để định hướng cho nhóm em đi đúng theo yêu cầu của đề tài đã chọn, cảm ơn thầy đã luôn giải đáp thắc mắc, khó khăn và đưa ra những góp ý, chỉnh sửa kịp thời giúp chúng em khắc phục những lỗi và sai sót và hoàn thành tốt cũng như đúng thời hạn đã đề ra.

Chúng em cũng xin gửi lời cảm ơn chân thành và sâu sắc đến các quý thầy cô trong khoa Điện – Điện tử đã tận tình truyền đạt những kiến thức cần thiết giúp chúng em có kiến thức nên đề tài này, cảm ơn vì đã tạo điều kiện để chúng em có thể tìm hiểu và thực hiện tốt đề tài. Nhóm em xin cảm ơn các bạn học cùng lớp đã cung cấp và hỗ trợ thêm nhiều thông tin và kiến thức hữu ích giúp chúng em có thể hoàn chỉnh tốt hơn về chương trình của nhóm.

Tuy đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian khá gấp rút do rơi vào thời điểm cuối kỳ , với những kiến thức còn hạn chế và còn nhiều thiếu sót khác về mặt lập trình và kinh nghiệm trong việc thực hiện một dự án cuối kỳ. Vì vậy, trong thời gian nhóm em thực hiện đề tài có những thiếu sót và lỗi là điều không thể tránh khỏi nên chúng em rất mong nhận được những góp ý quý báu của các quý thầy cô để kiến thức của chúng em được hoàn thiện và tối hơn.

Lời cuối , nhóm em kính chúc quý thầy, quý cô luôn dồi dào sức khỏe, tràn đầy năng lượng và thành công hơn nữa trong sự nghiệp giảng dạy của mình”.

Tp. Hồ Chí Minh, ngày 23 tháng 5 năm 2022

Nhóm sinh viên thực hiện

**Mục Lục**

[PHẦN 1: MỞ ĐẦU 1](#_Toc104884895)

[1. Lý do chọn đề tài 1](#_Toc104884896)

[2. Mục đích nghiên cứu 1](#_Toc104884898)

[3. Phương pháp nghiên cứu 1](#_Toc104884899)

[4. Đối tượng nghiên cứu 1](#_Toc104884900)

[PHẦN 2: CƠ SỞ LÝ THUYẾT 1](#_Toc104884901)

[2.1. Cấu trúc dữ liệu và giải thuật 2](#_Toc104884902)

[2.1.1 Khái niệm cấu trúc dữ liệu 2](#_Toc104884903)

[2.1.2 Khái niệm giải thuật 2](#_Toc104884904)

[2.1.3 Vì sao người học lập trình cần phải học cấu trúc dữ liệu và giải thuật? 2](#_Toc104884905)

[2.2. Ngôn ngữ lập trình C++ 3](#_Toc104884906)

[2.2.1 Giới thiệu ngôn ngữ C++ 3](#_Toc104884907)

[2.2.2 Các đặc điểm của C/C++ 3](#_Toc104884908)

[2.3. Cấu trúc STRUCT 4](#_Toc104884909)

[2.3.1 Khai báo struct 4](#_Toc104884910)

[2.3.3 Truy cập các trường dữ liệu của biến struct 6](#_Toc104884911)

[2.3.4 Nhập và xuất dữ liệu cho biến struct 7](#_Toc104884912)

[2.3.5 Structs và function 8](#_Toc104884920)

[PHẦN 3: PHÂN TÍCH VÀ GIẢI PHÁP 12](#_Toc104884921)

[3.1. Xác định hệ thống 12](#_Toc104884922)

[3.1.1 Xác định yêu cầu 12](#_Toc104884923)

[3.2. Phân tích hệ thống 12](#_Toc104884924)

[3.2.1 Xây dựng các thực thể 12](#_Toc104884925)

[3.2.2 Các mối quan hệ 13](#_Toc104884926)

[3.2.3 Chi tiết các thục thể 14](#_Toc104884927)

[3.2.4 Lưu đồ được sử dụng trong bài toán 16](#_Toc104884928)

[PHẦN 4: CHƯƠNG TRÌNH THỰC TẾ VÀ CHỨC NĂNG 18](#_Toc104884929)

[4.1 CÁC CHỨC NĂNG CƠ BẢN CỦA CHƯƠNG TRÌNH 18](#_Toc104884930)

[4.2. CHẠY CHƯƠNG TRÌNH VÀ PHÂN TÍCH CHỨC NĂNG 19](#_Toc104884931)

[4.2.1. Nhập thông tin sinh viên 19](#_Toc104884932)

[4.2.2. Nhập danh sách môn học 19](#_Toc104884933)

[4.2.3. Xuất danh sách sinh viên 20](#_Toc104884934)

[4.2.5 Xuất danh sách sinh viên theo ngành 21](#_Toc104884935)

[4.2.6 Xuất danh sách sinh viên dựa vào khóa học 23](#_Toc104884936)

[4.2.7. Tìm kiếm sinh viên theo mssv 24](#_Toc104884937)

[4.2.8 Tìm kiếm sinh viên theo tên 25](#_Toc104884938)

[4.2.9. Xóa sinh viên theo mssv 25](#_Toc104884939)

[4.2.10. Số tính chỉ và điểm tích lũy sinh viên cần tìm 26](#_Toc104884940)

[4.2.11. Cập nhật thông tin sinh viên 26](#_Toc104884941)

[4.2.12. Sắp xếp điểm theo môn tăng dần 26](#_Toc104884942)

[4.2.13. Ghi danh sách sinh viên được nhập vào file txt. 27](#_Toc104884943)

[4.2.14. Ghi bảng điểm vào file txt 28](#_Toc104884944)

[4.2.15. Đọc file txt 28](#_Toc104884945)

[KẾT LUẬN 31](#_Toc104884946)

[TÀI LIỆU THAM KHẢO 32](#_Toc104884947)

**PHẦN 1: MỞ ĐẦU**

**1. Lý do chọn đề tài**

“Công nghệ là một trong những thứ phát triển vượt bậc trong xã hội hiện nay. Ngày nay với sự phát triển vượt bậc và nhanh chóng của xã hội thì công nghệ được ứng dụng ở gần như toàn bộ tất cả các lĩnh vực và ngày càng đóng vai trò quan trọng và là một phần thiết yếu trong đời sống hằng ngày. Tạo ra công nghệ 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 em hiện đang là sinh viên công nghệ phần mềm không những cần có sự đầu tư mà còn không ngừng học hỏi và nâng cao kiến thức. Do đó để củng cố kiến thức đã học, đề tài mà em chọn là: Bài toán quản lý sinh viên bằng ngôn ngữ C++.

**2. Mục đích nghiên cứu**

* Thiết kế cấu trúc dữ liệu, thuật toán và xây dựng chương trình quản lý sinh viên bằng ngôn ngữ C++
* Chương trình có năng làm việc nhóm, lập trình cũng như củng cố các kiến thức về cấu trúc và giải thuật của 1 chương trình
* Nâng cao kỹ năng lập trình và kỹ năng sử dụng ngôn ngữ C++
* Quản lý thông tin sinh viên gồm có điểm và quá trình đạo tạo của sinh viên .

**3. Phương pháp nghiên cứu**

* Phương pháp nghiên cứu tài liệu
* Phương pháp thu thập dữ liệu
* Phương pháp lập trình
* Khảo sát thực tế các hệ thống đồng thời nghiên cứu lý thuyết để xây dựng cơ sở dữ liệu

**4. Đối tượng nghiên cứu**

* Cấu trúc dữ liệu, thuật toán và xây dựng trương trình quản lý sinh viên bằng cấu trúc STRUCT
* Các công cụ dùng để xây dựng chương trình: Visual studio, code BLOCKS”

**PHẦN 2: CƠ SỞ LÝ THUYẾT**

**2.1. Cấu trúc dữ liệu và giải thuật**

**2.1.1 Khái niệm cấu trúc dữ liệu**

Trong khoa học máy tính (hay lập trình) cấu trúc dữ liệu là “cách lưu trữ dữ liệu trong máy tính sao cho có thể giúp con người sử dụng dữ liệu đó một cách hiệu quả. Trong thực tế, có nhiều loại cấu trúc dữ liệu khác nhau, mỗi cấu trúc được áp dụng trong mỗi bài toán khác nhau vì vậy đặt ra một vấn đề trong nhiều hệ thống lớn. Để có hiệu năng và chất lượng tốt cần phải phụ thuộc rất nhiều vào việc tổ chức và lựa chọn loại cấu trúc dữ liệu.

**2.1.2 Khái niệm giải thuật**

Trong toán học và khoa học máy tính, một thuật toán hay còn gọi là giải thuật là một tập hợp hữu hạn các hướng dẫn được xác định rõ ràng, có thể thực hiện được bằng máy tính, thường để giải quyết một lớp vấn đề hoặc để thực hiện một phép tính.Các thuật toán luôn rõ ràng và luôn được sử dụng để chỉ rõ việc thực hiện các phép tính, suy luận tự động, xử lý dữ liệu và các tác vụ khác.

Trong lập trình các thuật toán được thiết kế để không phụ thuộc vào bất kì ngôn ngữ nào, nghĩa là chúng là những hướng dẫn có thể thực hiện với bất kỳ ngôn ngữ nào và đều nhận được kết quả như người học mong đợi.

**2.1.3 Vì sao người học lập trình cần phải học cấu trúc dữ liệu và giải thuật?**

Hiện nay các ứng dụng ngày càng cho thấy sự phức tạp ,trừu tượng vì lượng dữ liệu ngày càng lớn . Do đó có rất nhiều vấn đề cần giải quyết trong việc sử dụng dữ liệu cho hiệu quả”. Từ đó cấu trúc dữ liệu và giải thuật ra đời và áp dụng vào thực tế để”:

- Tăng tốc độ xử lý

- Xử lí vấn đề tìm kiếm dữ liệu

- Thực hiện được nhiều tác vụ mà con người yêu cầu

**2.2. Ngôn ngữ lập trình C++**

**2.2.1 Giới thiệu ngôn ngữ C++**

“Trước C++, “ngôn ngữ lập trình C được phát triển trong năm 1972 bởi **Dennis Ritchie** tại phòng thí nghiệm Bell Telephone, C chủ yếu là một ngôn ngữ lập trình hệ thống cũng là ngôn ngữ dùng để viết hệ điều hành. Hệ điều hành Windows một trong những hệ điều hành nổi tiếng cũng được viết từ C/C++

C++ (hay còn gọi C Plus Plus) một loại ngôn ngữ lập trình bậc trung. Ngôn ngữ C++ là ngôn ngữ lập trình đa năng được **Bjarne Stroustrup** tạo ra và được sử dụng rộng rãi cho đến nay.

Từ những năm 1990, C++ trở thành một trong những ngôn ngữ thương mại phổ biến và ưa thích của lập trình viên.

C++ là phiên bản mở rộng nâng cấp từ ngôn ngữ lập trình C. Những bản cập nhật gần đây nhất là C++ 14 và C++ 17, và sắp tới là C++ 20 đã và đang đưa đến cho lập trình viên C++ những tính năng hỗ trợ rất lớn.

**2.2.2 Các đặc điểm của C/C++**

**Ngôn ngữ lập trinh bậc trung**: Các bạn có thể sử dụng C/C++ để phát triển những chương trình bậc cao. Đối với những chương trình bậc thấp do C/C++ phát triển các chương trình này vẫn hoạt động tốt trên hầu hết phần cứng.

**Đơn giản và hiệu quả:** Cú pháp của C/C++ rất dễ hiểu và khá giống Tiếng Anh. Thậm chí ta có thể dùng C/C++ để thiết kế những ứng dụng dù trước đó đã được viết bằng *hợp ngữ* ( Assembly language).

**Hỗ trợ đa nền tảng:** C/C++ được hỗ trợ trên bất kỳ các nền tảng, hệ điều hành nào. Ta có thể viết một chương trình C/C++ trên Linux, biên dịch nó trên Windows và chạy trên Mac OS.

**Lập trình hướng đối tượng (** **Object-oriented programming):**Lí do cho sự phổ biến của C++ chính là nó hỗ trợ vô cùng hiệu quả việc lập trình hướng đối tượng, bao gồm việc trường tựu hóa dữ liệu (*data abstraction*), đóng gói dữ liệu (*data encapsulation )*, kế thừa ( *inheritance*) và đa hình ( *polymorphism*).”

**Con trỏ:**Ngôn ngữ C/C++ cung cấp cho chúng ta một công cụ vô cùng tiện lợi là *con trỏ (pointer)*, nhiệm vụ con trỏ đó là lưu địa chỉ bộ nhớ làm giá trị của nó và đồng thời lưu trữ , truy cập dữ liệu từ bộ nhớ . Con trỏ khá là quan trọng nó liên quan tới rất nhiều đến các môn học khác vậy nên chúng ta cần phải nắm vững kiến thức về con trỏ”.[[1]](#footnote-1)

**2.3. Cấu trúc STRUCT**

Một cấu trúc struct (viết tắt của structure) “cho phép chúng ta nhóm nhiều biến của nhiều kiểu dữ liệu khác nhau để lưu trữ một tập hợp các dữ liệu cần thiết cho việc mô tả hay diễn tả một hoặc nhiều đơn vị.

### **2.3.1 Khai báo struct**

Cách để khai báo một cấu trúc mới, chúng ta sử dụng từ khóa **struct**. Mặc dù một struct là kiểu dữ liệu do lập trình viên định nghĩa, nó cũng cần được khai báo theo một cú pháp nhất định để trình biên dịch có thể hiểu được. Đây là cú pháp để khai báo một struct mới:

**struct** <**name\_of\_new\_type**>

{

<variables>;

};

Trong đó:

* struct là từ khóa mà ngôn ngữ C++ cung cấp.
* Name\_of\_new\_type là tên của kiểu dữ liệu mới, sau khi khai báo xong một struct, chúng ta có thể dùng tên struct để khai báo biến như những kiểu dữ liệu thông thường.
* variables là danh sách của các biến được dùng để lưu trữ dữ liệu sao cho phù hợp với yêu cầu lưu trữ dữ liệu của một đơn vị nào đó.

Sau đây là ví dụ minh họa :

**#include <iostream>**

**#include <string>**

**struct** VietNamPeople

{

\_\_int32 ID;

std::string name;

\_\_int16 age;

**float** height;

**float** weight;

**bool** isStudent;

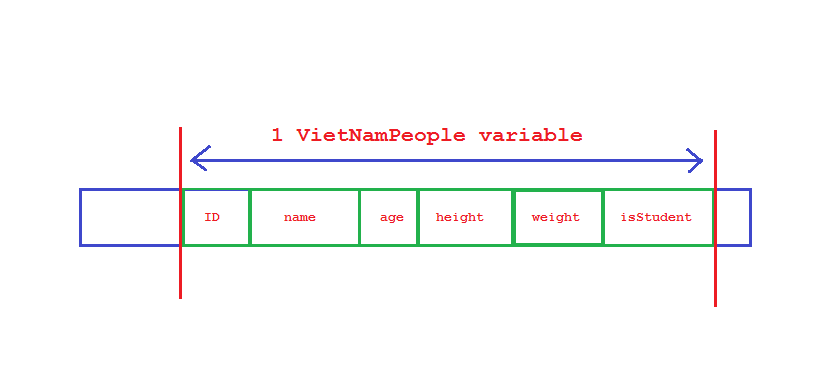
};

Như thế đã định nghĩa xong một kiểu dữ có tên là VietNamPeople, struct này bây giờ được coi là một kiểu dữ liệu mới là một tập hợp các biến ID, age, name, height, weight ,isStudent và height. Các biến này được sắp xếp chung vào một nhóm và mỗi biến sẽ lưu trữ một giá trị thông tin của một đơn vị là một con người Việt Nam.

Như thế các bạn cần chú ý về phạm vi sử dụng của các kiểu dữ liệu tự định nghĩa cũng tương tự phạm vi sử dụng của biến trong một chương trình. Nhưng việc định nghĩa kiểu dữ liệu không nhất thiết yêu cầu hệ điều hành cấp phát bộ nhớ nên hoàn toàn không làm ảnh hưởng đến tài nguyên của hệ thống. Do đó, chúng ta nên cần phải định nghĩa các kiểu dữ liệu mới cho phạm vi toàn cục để kiểu dữ liệu mới này có thể được sử dụng trong toàn bộ file, kể cả sử dụng trong những file mã nguồn khác trong cùng sự án.

Các biến này được đặt trong struct, chúng ta gọi chúng là các trường dữ liệu của struct (fields). Một trường dữ liệu là tập hợp các biến dùng để lưu trữ dữ liệu cần thiết cho một đơn vị. VD kiểu dữ liệu VietNamPeople có tổng cộng 6 trường dữ liệu là ID,age, name, weight ,height và isStudent. Trình biên dịch có thể hiểu rằng khi một biến được tạo ra từ kiểu dữ liệu VietNamPeople.

Ví dụ: VietNamPeople leTranDat thì bây giờ , leTranDat xem như là một cái tên của một đơn vị được tạo thành từ tập hợp các trường dữ liệu ID, age, name, weight, height và isStudent” đã được định nghĩa cho kiểu dữ liệu VietNamPeople.

**[[](https://github.com/nguyenchiemminhvu/CPP-Tutorial/blob/master/9-kieu-du-lieu-tu-dinh-nghia/9-1-structs/0.png?raw=true)](https://github.com/nguyenchiemminhvu/CPP-Tutorial/blob/master/9-kieu-du-lieu-tu-dinh-nghia/9-1-structs/0.png?raw=true" \o "0.png?raw=true)**

“Trên Đây chỉ là những hình ảnh minh họa cho việc tổ chức dữ liệu của các trường bên trong biến kiểu VietNamPeople sau khi được khai báo. Thực tế các trường này sẽ có kích thước khác so với phần dự đoán trước (phần nâng cao của struct sẽ trình bày về vấn đề này).

Chúng ta có thể sử dụng các kiểu dữ liệu đã định nghĩa ở trên VietNamPeople để tạo ra nhiều biến struct khác nhau:

VietNamPeople leTranDat*;*

VietNamPeople dayNhauHoc*;*

VietNamPeople ngoDoanTuan*;”*

Như các biến thông thường, các biến struct này sẽ được cấp phát bộ nhớ tùy vào cách chọn kỹ thuật cấp phát mà người lập trình đặt ra.

##### **2.3.2 Khởi tạo cho biến struct**

Chúng ta có thể khởi tạo “giá trị cho trường dữ liệu trong các biến struct rắc rối hơn khởi tạo giá trị cho biến thông thường một chút. Nhưng ngôn ngữ C++ đã hổ trợ cho chúng ta một cách dễ dành và nhanh chóng hơn là sử dụng một **initializer list**. Nó cho phép các bạn khởi tạo một hoặc một số trường dữ liệu trong khai báo biến struct”. Ví dụ:

**Struct** Sinhvien

{

\_\_int32 ID;

std::string ten;

\_\_int32 tuoi;

\_\_int32 nam;

};

*//................*

Sinhvien PhanTanQuoc = { 1, "Phan Tan Quoc", 20, 2 };

Từ các trường dữ liệu này được tạo ra lần lượt từ trên xuống dưới như trong phần định nghĩa struct có tên sinhvien. Bây giờ, biến leTranDat sẽ chứa được các thông tin được khởi tạo lần lượt là: ID = 1, ten = "Phan Tan Quoc", tuoi = 20 và nam = 2.

Nếu initializer list không cung cấp đầy đủ các dữ liệu cho các trường dữ liệu thì giá trị mặc định sẽ được dùng để khởi tạo có biến này. Ví dụ:

Sinhvien newsv = { 1, "new sv" }; //tuoi = 0, nam = 0 by default

Một tập hợp từ các giá trị của một biến struct được gọi là một Record (bản ghi)”.

### **2.3.3 Truy cập các trường dữ liệu của biến struct**

struct Employee

{

\_\_int32 ID*;*

std::string name*;*

\_\_int32 age*;*

\_\_int32 year\_of\_exp*;*

}*;*

“Các kiểu dữ liệu của Employee được mô tả rằng mỗi biến của kiểu Employee được tạo ra sẽ gồm 4 trường dữ liệu là ID, age, name và year\_of\_exp. Như thế, bất kỳ biến nào có kiểu Employee đều có đủ 4 trường dữ liệu trên.

Nếu muốn truy xuất được đến trường dữ liệu của các biến struct, chúng ta cần sử dụng **member selection operator** . Sau Đây là một ví dụ:

Sinhvien PhanTanQuoc*;*

PhanTanQuoc.ID = 1*;*

PhanTanQuoc.ten = "Phan Tan Quoc"*;*

PhanTanQuoc.tuoi = 20*;*

PhanTanQuoc.nam = 2.

Các trường dữ liệu của các biến struct như là những biến thông thường, nhưng nó được gói gọn bên trong một biến struct cụ thể, vì vậy chúng ta cần phải sử dụng đúng tên của biến struct và **member selection operator** để truy xuất đến nó.

Vì vậy, thông qua tên các biến struct, trong các trường dữ liệu được nhóm lại giúp ta biết được những trường dữ liệu đó được dùng cho đơn vị tổ chức nào, điều này giúp ta dễ dàng tổ chức chương trình và quản lý chương trình ở quy mô lớn.

Vì các trường dữ liệu của một biến struct cũng là những biến thông thường, chúng ta cũng có thể sử dụng chúng để tính toán, so sánh, ...

Sinhvien PhanTanQuoc= { 1, "Phan Tan Quoc", 20, 2 };

Sinhvien juniorsv= { 2, "New sv", 20, 1 };

**if** (PhanTanQuoc.nam > juniorsv.nam)

{

std::cout << PhanTanQuoc.ten << " has more experience than " << juniorsv.ten<< std::endl;

}

Khi các trường dữ liệu của các struct sẽ cùng tồn tại với biến struct cho đến khi biến struct được xuất ra khỏi phạm vi sử dụng và hủy bỏ . Vì vậy, khi struct còn tồn tại, chúng ta vẫn có thể thể truy xuất được đến các trường dữ liệu của nó.

### **2.3.4 Nhập và xuất dữ liệu cho biến struct**

Giống như việc nhập, xuất dữ liệu cho các biến thông thường. Ở đây chỉ khác là chúng ta cần sử dụng thêm tên biến struct và **member selection operator”** để biên dịch để biết chúng ta nhập xuất cho trường dữ liệu của đơn vị nào. Ví dụ được lấy trong phần code của nhóm:

struct Sinhvien;

*//Input*

cout <<"Nhap ma so sinh vien: ";

cin >>sv.mssv;

fflush(stdin);

cout <<"Nhap ho ten: ";

fflush(stdin);

gets(sv.hoten);

cout <<"Nhap ngay sinh: ";

fflush(stdin);

gets(sv.ngaysinh);

cout <<"Nhap gioi tinh: ";

fflush(stdin);

gets(sv.gioitinh);

cout<<"Nhap email: ";

fflush(stdin);

gets(sv.email);

cout<<"Nhap so dien thoai: ";

fflush(stdin);

gets(sv.sdt);

cout<<"Nhap que quan: ";

fflush(stdin);

gets(sv.quequan);

*//Output*

### cout <<"\nMa so sinh vien: "<<sv.mssv;

### cout <<"\tHo ten: "<<sv.hoten;

### cout <<"\tNgay sinh: "<<sv.ngaysinh;

### cout <<"\tGioi tinh: "<<sv.gioitinh;

### cout <<"\tEmail: "<<sv.email;

### cout <<"\tSo dien thoai: "<<sv.sdt;

### cout <<"\tQue quan: "<<sv.quequan;

### **2.3.5 Structs và function**

Có một ưu điểm khi chúng ta sử dụng struct đó là chúng ta không cần truyền tất cả các trường dữ liệu của một đơn vị nào đó thay vào đó ta chỉ cần sử dụng một biến struct làm tham số cho hàm. Ví dụ:

**struct** Vector2D

{

**float** x;

**float** y;

};

**void** **printVector2D**(Vector2D vec)

{

std::cout << "(" << vec.x << "," << vec.y << ")" << std::endl;

}

**int** **main**()

{

Vector2D vec = { 1, 4 };

printVector2D(vec);

**return** 0;

}

“Từ ví dụ trên, sử dụng kiểu truyền dữ liệu giá trị nên tham số Vector2D vec của hàm printVector2D không làm thay đổi giá trị gốc của đối số. Chúng ta cũng có thể thử truyền đối số là biến struct theo kiểu tham chiếu hoặc con trỏ, về mặt cơ bản biến struct cũng tương tự như là một biến có địa chỉ cụ thể vì vậy chúng ta làm hoàn toàn tương tự như biến thông thường”.

void normalize(Vector2D &vec)

{

float length = sqrt((vec.x \* vec.x) + (vec.y \* vec.y));

vec.x = vec.x / length;

vec.y = vec.y / length;

}

int main()

{

Vector2D vec = { 1, 4 };

printVector2D(vec);

normalize(vec);

printVector2D(vec);

**return** 0;

}

Kiểu dữ liệu struct cũng có thể được dùng làm kiểu trả về của hàm. Ví dụ:

Vector2D addTwoVector(Vector2D vec1, Vector2D **vec2**)

{

Vector2D result = { vec1.x + **vec2**.x, vec1.y + **vec2**.y };

**return** result;

}

**int** main()

{

Vector2D vec1 = { 1, 2 };

Vector2D **vec2** = { 2, 2 };

Vector2D result = addTwoVector(vec1, **vec2**);

**return** 0;

}

Lập trình viên có thể làm như trên vì ngôn ngữ C++ cho phép chúng ta gán biến struct cho một biến cùng kiểu struct khác. Ví dụ:

Vector2D vec1 = { 1, 2 };

Vector2D vec2 = vec1;

Tuy nhiên, chúng ta không nên sử dụng phép gán trực tiếp như vậy. Bởi vì trong struct có thể còn có các yếu tố phức tạp khác như con trỏ, hoặc struct khác, ... Dễ gây ra nhiều vấn đề liên quan. Quay lại với chủ đề đang trình bày.

Chúng ta còn có thể định nghĩa cho các hàm bên trong phần định nghĩa của struct. Ví dụ:

struct Vector2D

{

**float** x;

**float** y;

**void** normalize()

{

**float** length = sqrt(x \* x + y \* y);

x = x / length;

y = y / length;

}

};

Ví dụ trên cho ta thấy, hàm normalize nó được định nghĩa trong cùng khối lệnh của struct Vector2D, vì vậy nó có thể trực tiếp truy cập đến biến x và y và thao tác với chúng. Nhưng x và y của struct Vector2D bản thân nó đang còn ở mức khái niệm, chỉ khi nào struct Vector2D được dùng để khai báo biến, biến x và y cũng như hàm normalize mới được tạo ra. Như mình , việc chúng ta định nghĩa một kiểu dữ liệu mới chỉ là định nghĩa những dữ liệu sẽ tồn tại trong biến struct nếu nó được tạo ra.

Hàm normalize được tạo ra, tuy nhiên chỉ được sử dụng khi một biến struct cụ thể gọi đến nó bằng **member selection operator**.

int main()

{

Vector2D vec = { 1, 4 };

vec.normalize();

printVector2D(vec);

**return** 0;

}

Hàm normalize được sử dụng trong struct Vector2D bản thân nó chỉ có thể được gọi thông qua một biến struct cụ thể. Vì vậy khi chúng ta muốn chuẩn hóa một Vector2D, hàm normalize(Vector2D) nữa mà chỉ cần gọi hàm normalize được định nghĩa trong chính nó. Đây cũng được xem như là một ưu điểm khi sử dụng struct.

Lấy ví dụ về hàm được định nghĩa bên trong struct:

struct Vector2D

{

**float** x;

**float** y;

**void** setPosition(**float** X, **float** Y)

{

x = X;

y = Y;

}

**void** normalize()

{

**float** length = sqrt(x \* x + y \* y);

x = x / length;

y = y / length;

}

};

Thiêm vào struct Vector2D hàm setPosition(float, float), lúc này chúng không cần sử dụng **initializer list** để khởi tạo nữa mà cho một biến kiểu Vector2D nữa, mà mình sẽ gọi hàm setPosition(float, float).

int main()

{

Vector2D vec;

vec.setPosition(1, 4);

printVector2D(vec);

**return** 0;}[[2]](#footnote-2)

# 

# PHẦN 3: PHÂN TÍCH VÀ GIẢI PHÁP

## 3.1. Xác định hệ thống

### **3.1.1 Xác định yêu cầu**

**- Đối tượng quản lý:** Sinh viên, điểm, xuất bảng điểm cá nhân cho sinh viên, bảng điểm cho lớp, danh sách sinh viên trong lớp, số tính chỉ của sinh viên, Danh sách điểm trung bình các sinh viên.

- Yêu cầu đối với hệ thống

* Quản lý

+ Quản lý sinh viên

+ Quản lý điểm.

+ Quản lý số tính chỉ

* Thống kê

+Thống kê danh sách sinh viên trong lớp

+ Thống kê bảng điểm sinh viên

+ Thống kê bảng điểm cho lớp

+ Thông kê danh sách sinh viên sau khi chỉnh sửa

* Tra cứu sinh viên

## 3.2. Phân tích hệ thống

### **3.2.1 Xây dựng các thực thể**

Thực thể **Sinhvien** (sinh viên): “Mã số sinh viên ( khóa chính), họ tên, ngày sinh, giới tính, email, số điện thoại, quê quán, điểm tích lũy, số tính chỉ tích lũy”.

Thực thể **Monhoc** (môn học): “Mã lớp, mã môn, điểm thưởng, điểm giữa kỳ, điểm cuối kỳ, điểm tổng kết”.

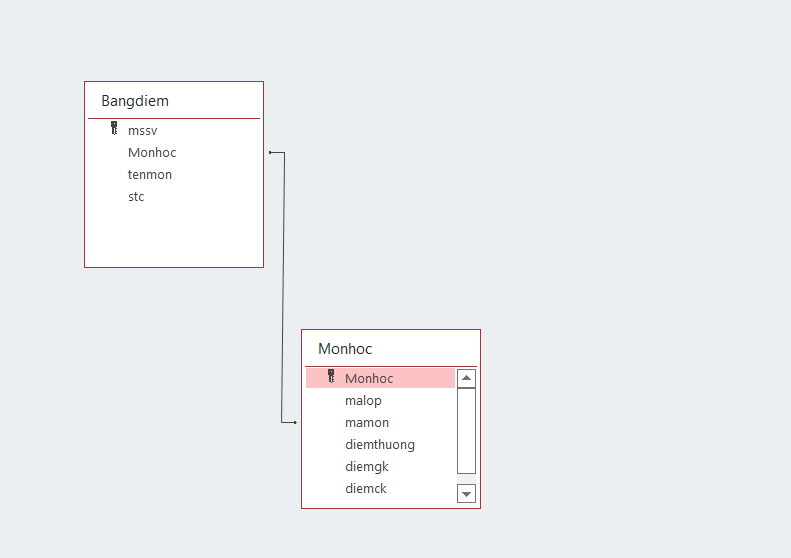
Thực thể **Bangdiem** (bảng điểm): “Mã số sinh viên (khóa chính), Môn học, tên môn, số tính chỉ”.

### **3.2.2 Các mối quan hệ**

- Bảng điểm với môn học

Trong bảng điểm sẽ có thực thể Monhoc trong thực thể Monhoc có nhiều thực thể điểm

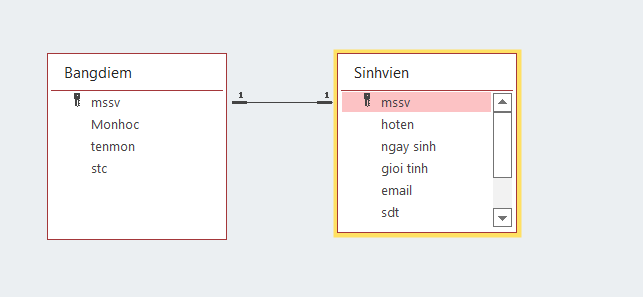
Mục đích: Quản lý điểm các môn học của sinh viên

****

**- Bảng điểm với sinh viên**

Thực thể bảng điểm liên kết với thực thể sinh viên thông qua mssv

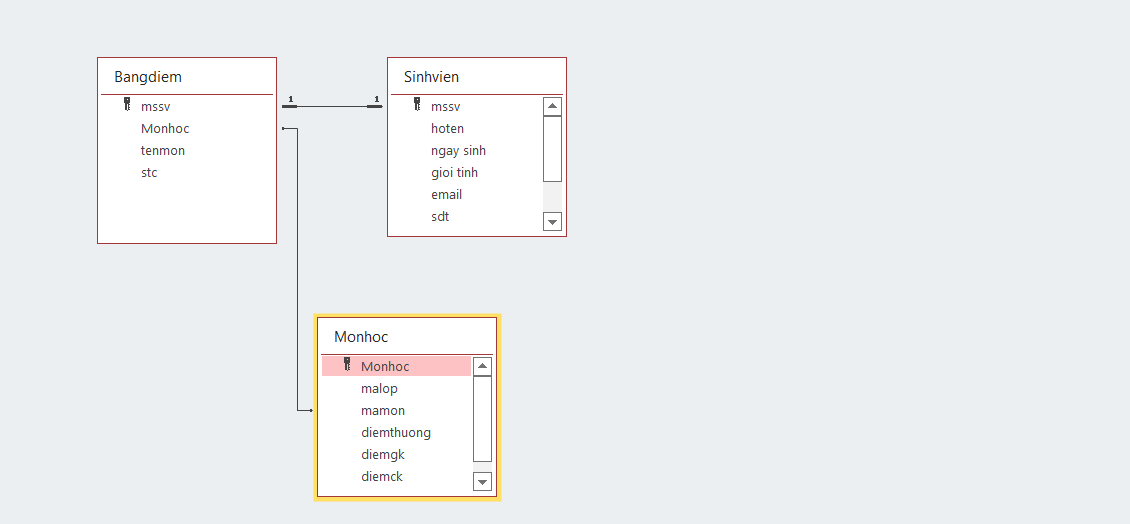
Mục đích: liên kết giữ liệu giữa 2 thực thể khác nhau



**- Bảng điểm với sinh viên với môn học**

Thực thể Monhoc được quản lý bởi thực thể bảng điểm, thực thể Bangdiem liên kết với thực thể sinh viên thông qua khóa chính là mssv

Mục đích: liên kết giữ liệu các thực thể với nhau dễ dàng tra cứu, cập nhật thông tin cho các dữ liệu trong các thực thể

****

### **3.2.3 Chi tiết các thục thể**

**- Sinhvien:** sinh viên

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên thực thể** | **Kiểu** | **Kích thước** | **Khóa chính** | **Khóa ngoại** | **Diễn Giải** |
| **Mssv** | **Int** |  | **x** |  | **Mã số sinh viên** |
| **Hoten** | **Char** | **30** |  |  | **Họ và tên** |
| **Ngaysinh** | **Char** | **30** |  |  | **Ngày sinh** |
| **Gioitinh** | **Char** | **60** |  |  | **Giới tính** |
| **Email** | **Char** | **51** |  |  | **Địa chỉ mail** |
| **Sdt** | **Char** | **51** |  |  | **Số điện thoại** |
| **Quequan** | **Char** | **30** |  |  | **Quê quán** |
| **Dtl** | **Float** |  |  |  | **Điểm tích lũy** |
| **Stctl** | **Float** |  |  |  | **Số tính chỉ tích lũy** |

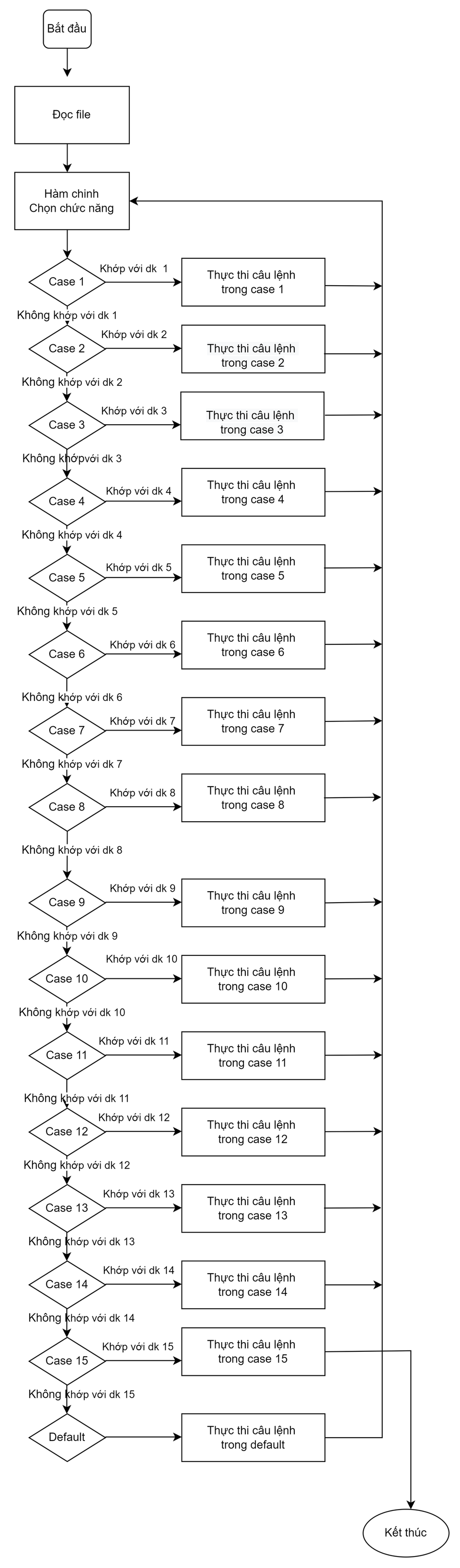
**- Monhoc: môn học**

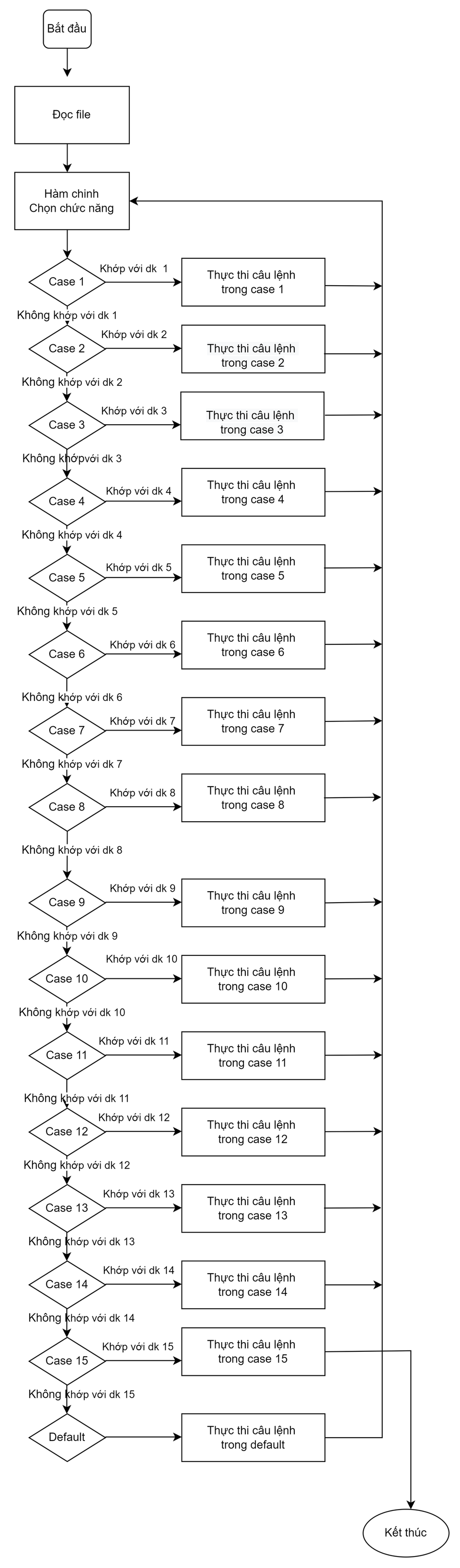
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên thực thể** | **Kiểu** | **Kích thước** | **Khóa chính** | **Khóa ngoại** | **Diễn Giải** |
| **Malop** | **Char** | **11** |  |  | **Mã lớp** |
| **Mamon** | **Int** |  |  |  | **Mã môn** |
| **Diemthuong** | **Float** |  |  |  | **Điểm thưởng** |
| **Diemgk** | **Float** |  |  |  | **Điểm giữa kỳ** |
| **Diemck** | **Float** |  |  |  | **Điểm cuối kỳ** |
| **Diemtk** | **Float** |  |  |  | **Điẻm tổng kết** |

**- Bangdiem (bảng điểm)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên thực thể** | **Kiểu** | **Kích thước** | **Khóa chính** | **Khóa ngoại** | **Diễn Giải** |
| **Mssv** | **int** |  | **x** |  | **Mã số sinh viên** |
| **Monhoc** | **Char** |  |  | **x** | **Môn học** |
| **Tenmon** | **Char** | **30** |  |  | **Tên môn** |
| **Stc** | **Int** |  |  |  | **Số tính chỉ** |

### **3.2.4 Lưu đồ được sử dụng trong bài toán**





# PHẦN 4: CHƯƠNG TRÌNH THỰC TẾ VÀ CHỨC NĂNG

## 4.1 CÁC CHỨC NĂNG CƠ BẢN CỦA CHƯƠNG TRÌNH

1. Nhập thông tin sinh viên

2. Nhập danh sách môn học

3. Xuất thông tin sinh viên

4. Xuất bảng điểm sinh viên

.Xuất danh sách sinh viên theo ngành

6.Xuất danh sách sinh viên theo khóa học

7. Tìm kiếm sinh viên theo mssv

8 Tìm kiếm sinh viên theo tên

9. Xóa sinh viên theo mssv

10. Số tín chỉ và điểm tích lũy sinh viên cần tìm

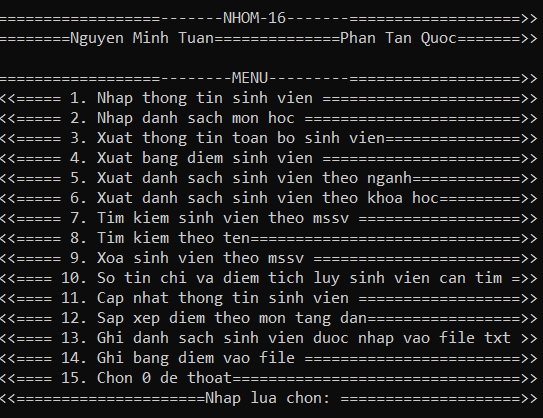
11. Cập nhật thông tin sinh viên

12 Sắp xếp điểm theo môn tăng dần.

13. Ghi danh sách sinh viên được nhập vào file txt

14. Ghi bảng điểm vào file

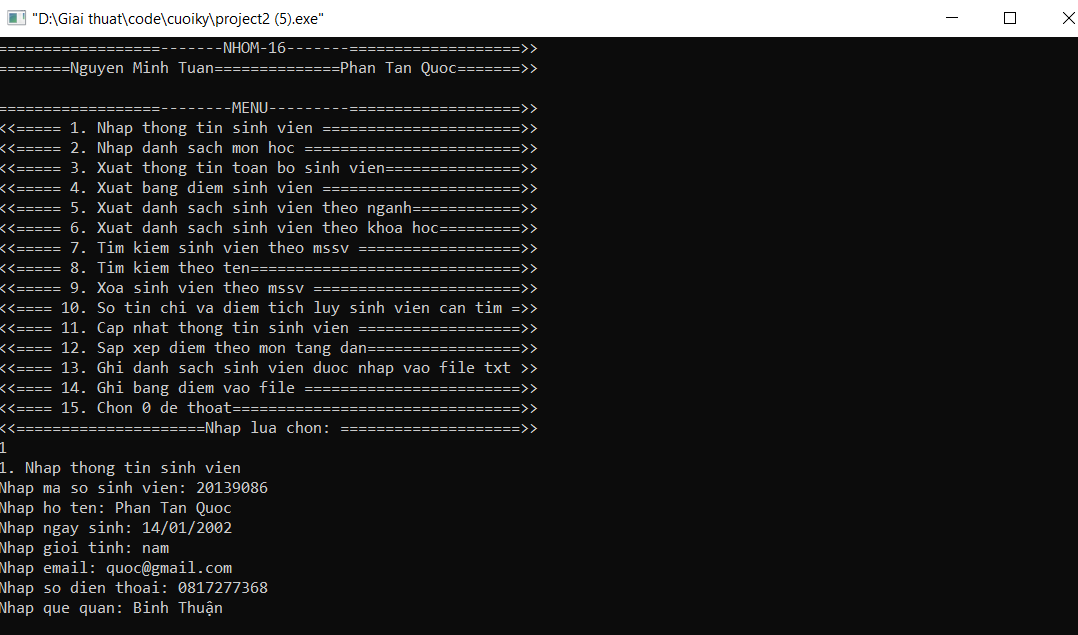
15. Thoát chương trình



## 4.2. CHẠY CHƯƠNG TRÌNH VÀ PHÂN TÍCH CHỨC NĂNG

### **4.2.1. Nhập thông tin sinh viên**

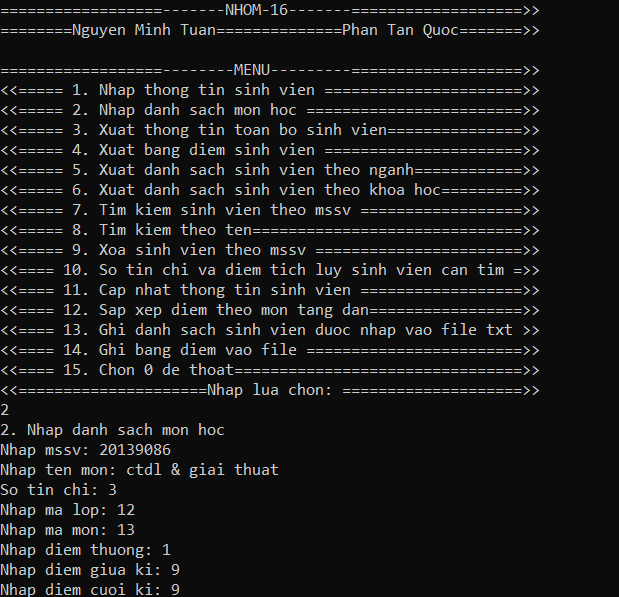
Chương trình cho phép người dùng nhập vào thông tin của từng sinh viên bao gồm: mã số sinh viên, họ và tên, ngày sinh, giới tính, địa chỉ mail, số điện thoại, quê quán.



### **4.2.2. Nhập danh sách môn học**

Chương trình cho phép người dùng nhập vào thông tin sách sách môn học của từng sinh viên theo mssv

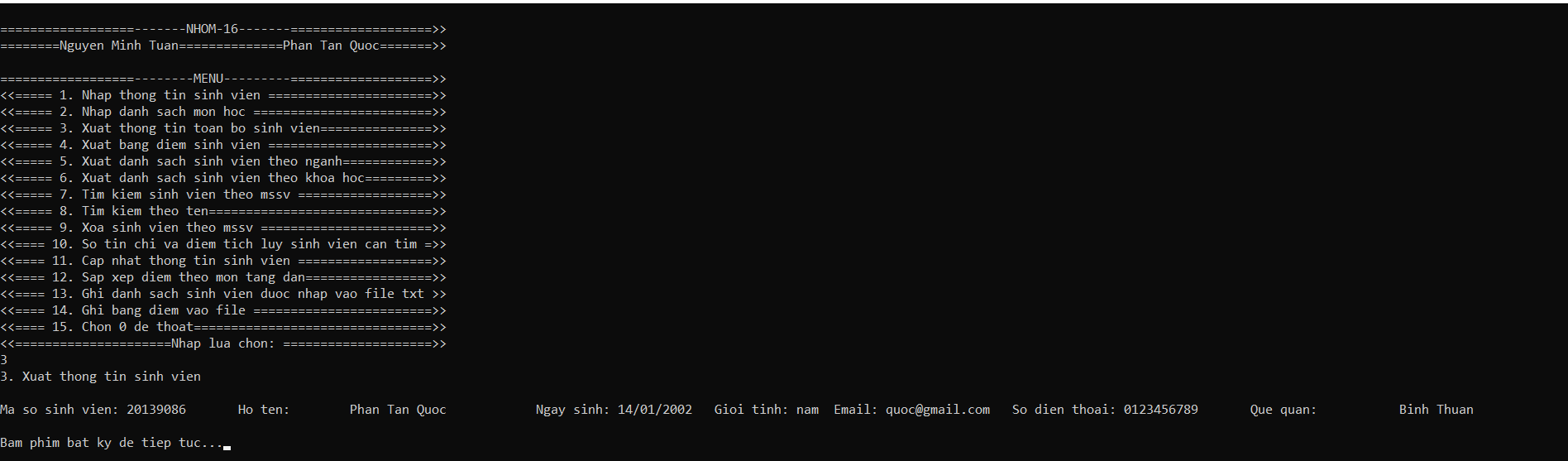
Bao gồm: số tính chỉ, mã lớp, mã môn, điểm thưởng, điểm giữa kỳ, điểm cuối kỳ.

****

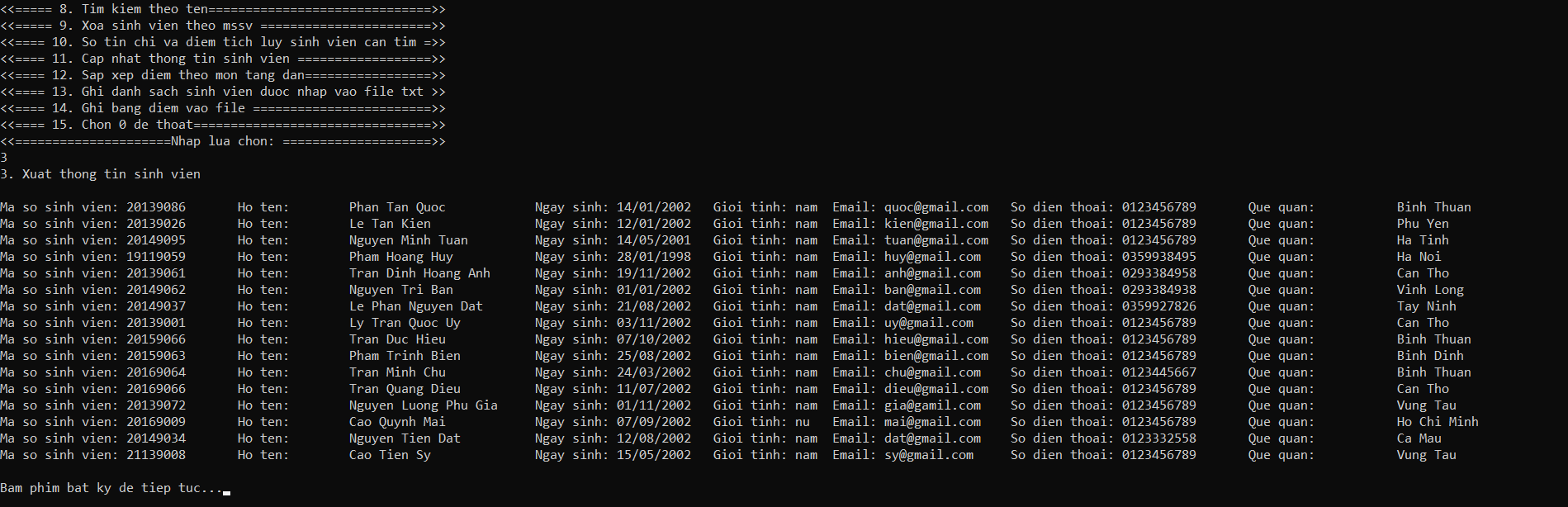
### **4.2.3. Xuất danh sách sinh viên**

Xuất thông tin tất cả sinh viên có đã nhập từ mục 1

Xuất thông tin của 1 sinh viên:

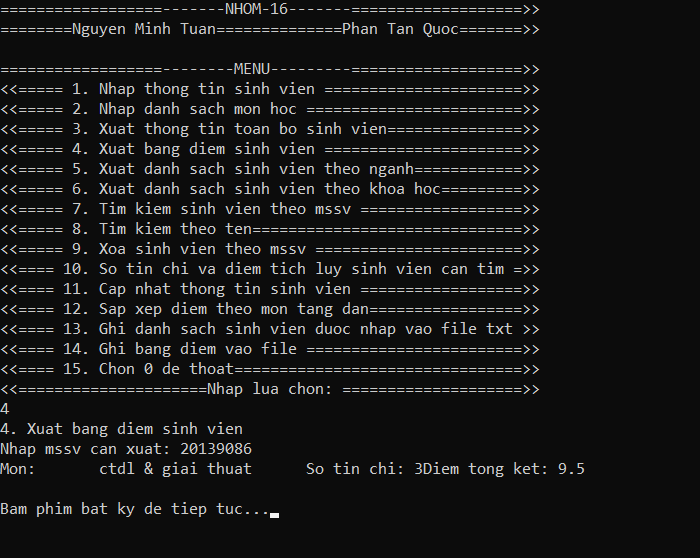


Xuất thông tin của nhiều sinh viên:



**4.2.4.Xuất bảng điểm sinh viên**

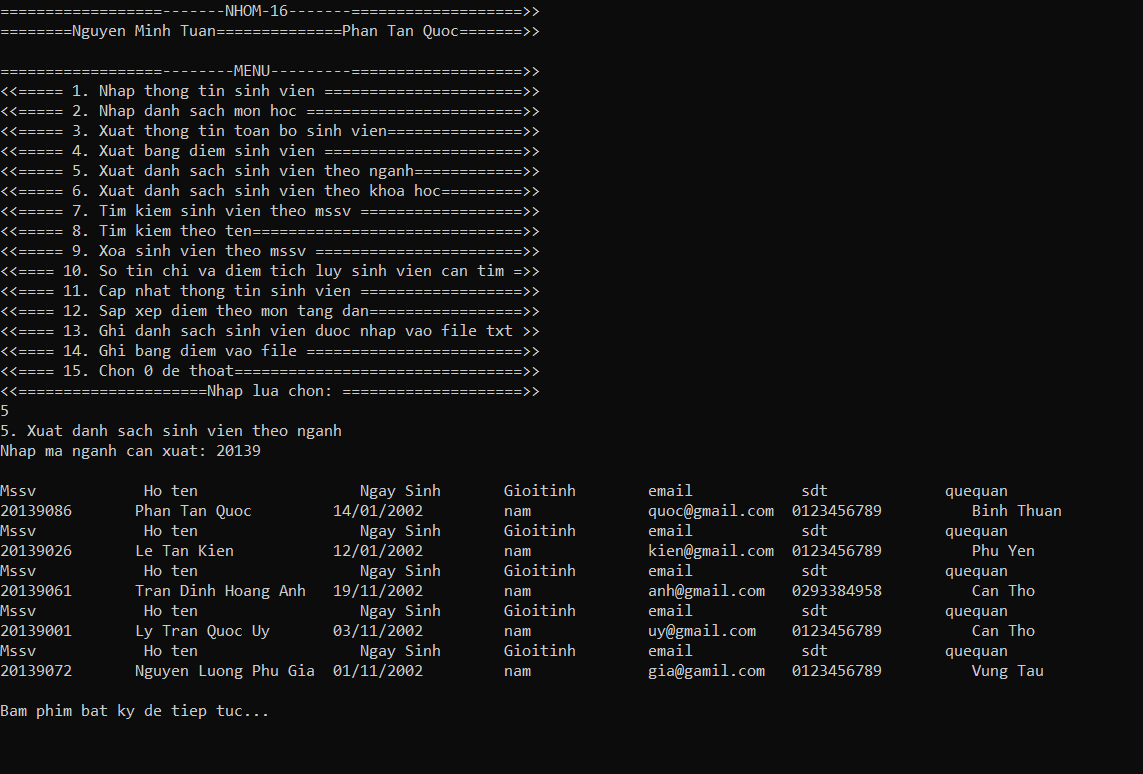
Xuất tất cả thông tin về điểm của sinh viên đã nhập ở phần 2 theo mssv cần xuất bao gồm: tên môn, số tính chỉ và tổng điểm.



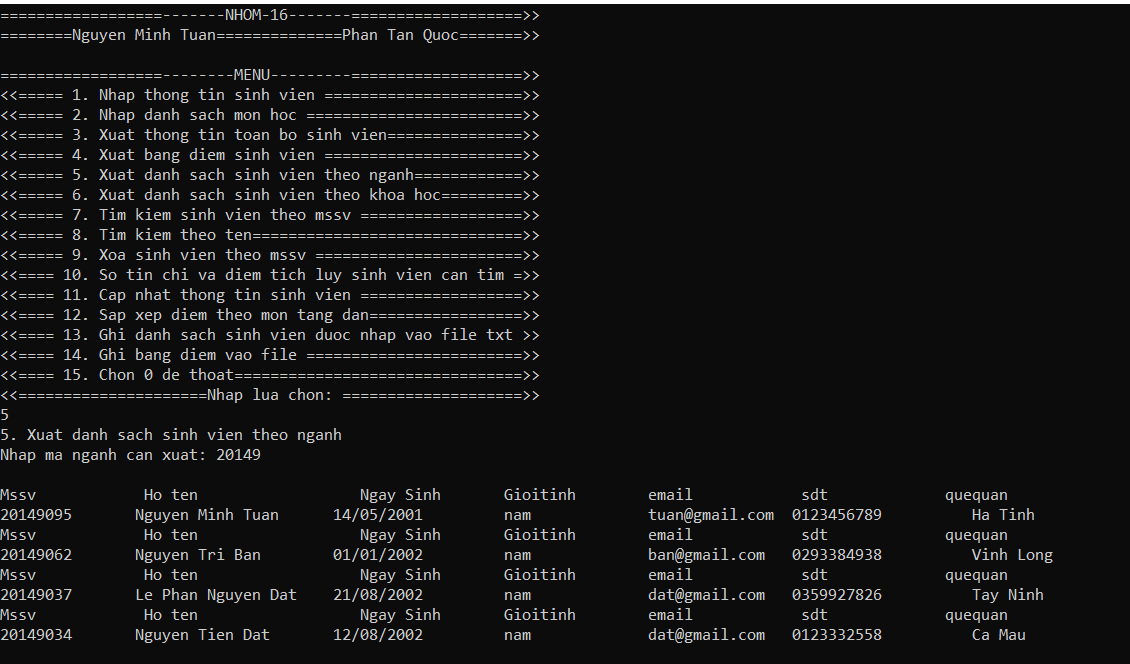
### **4.2.5 Xuất danh sách sinh viên theo ngành**

Chương trình sẽ xuất các sinh viên có các ngành giống nhau dựa vào mã ngành của mssv

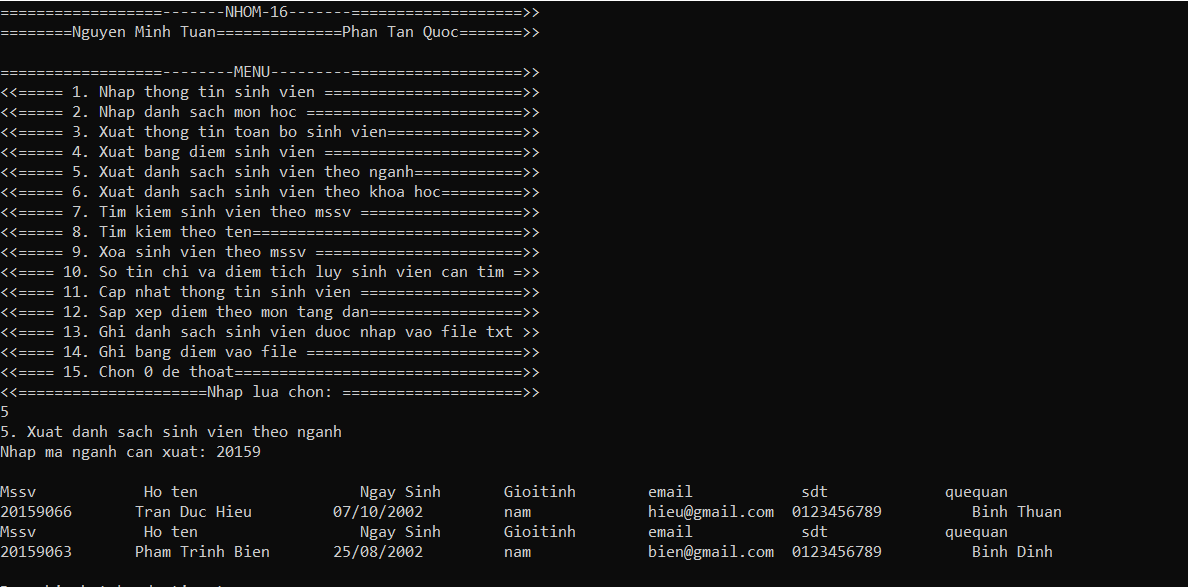
- Xuất mã ngành 20139:



- Xuất mã ngành 20149:



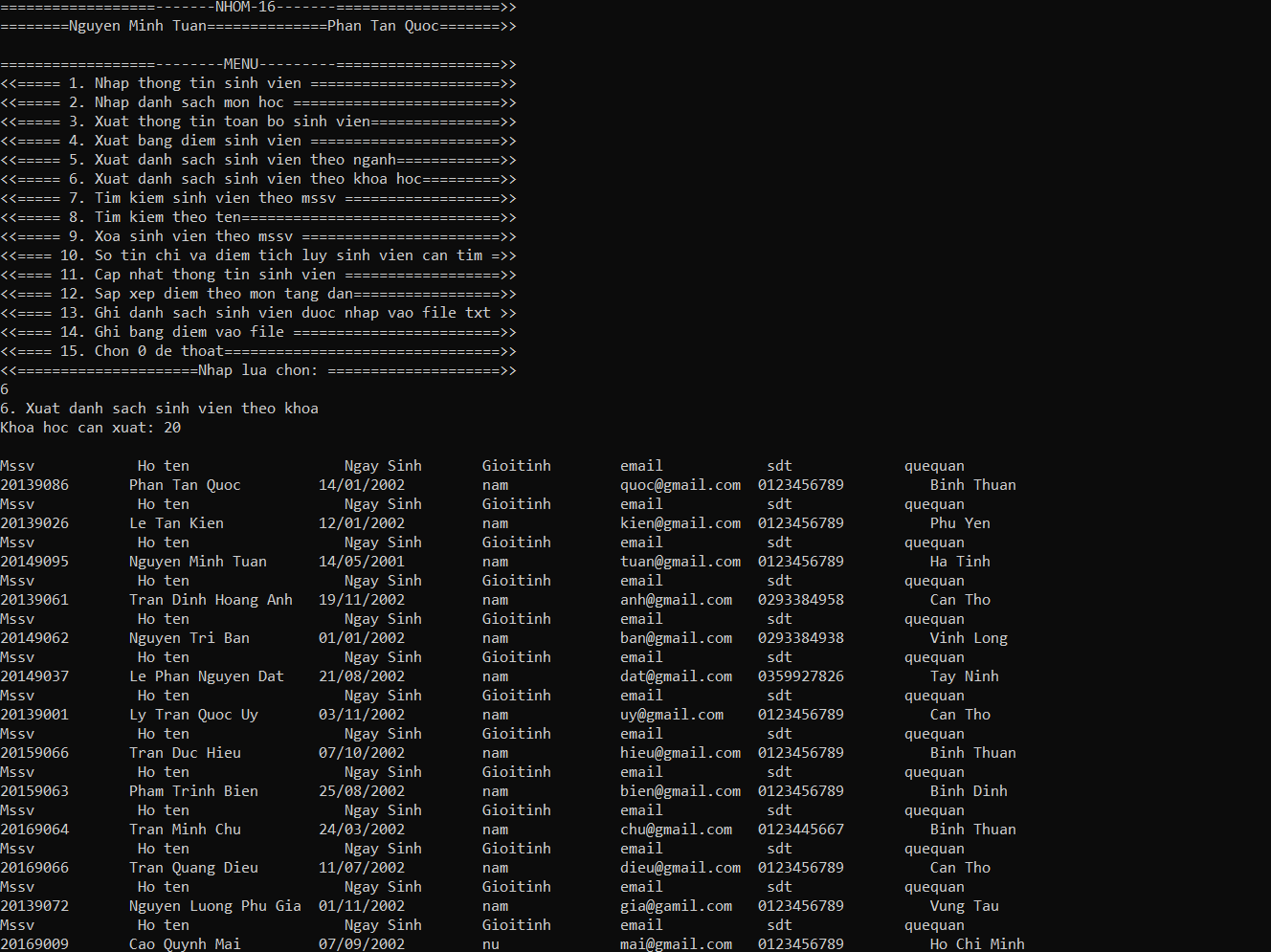
- Xuất mã ngành 20159:



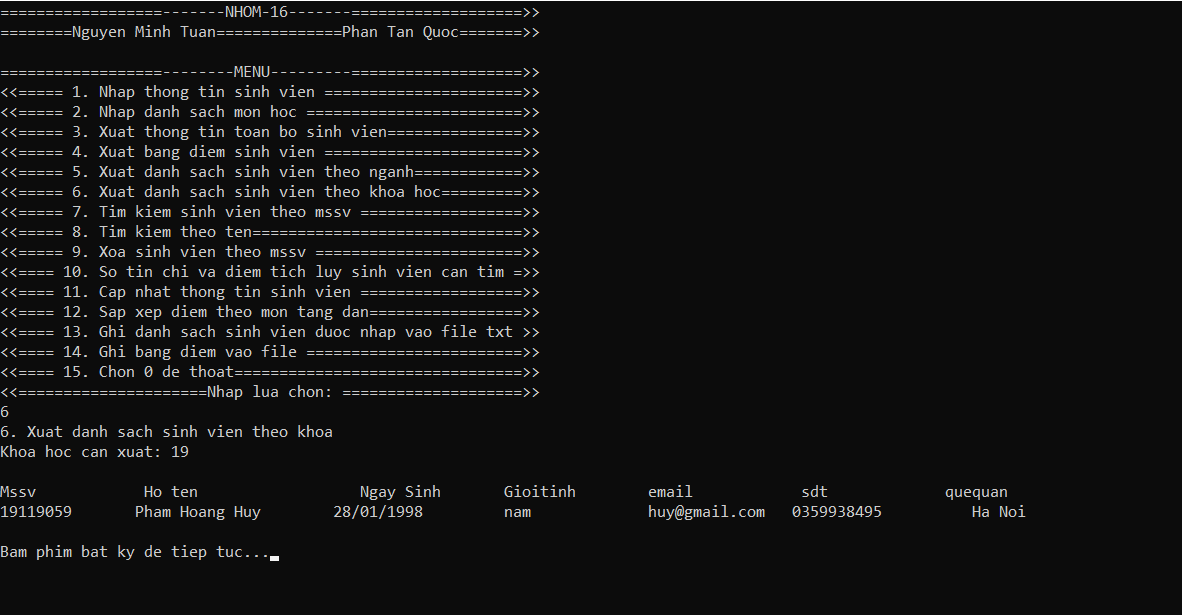
### **4.2.6 Xuất danh sách sinh viên dựa vào khóa học**

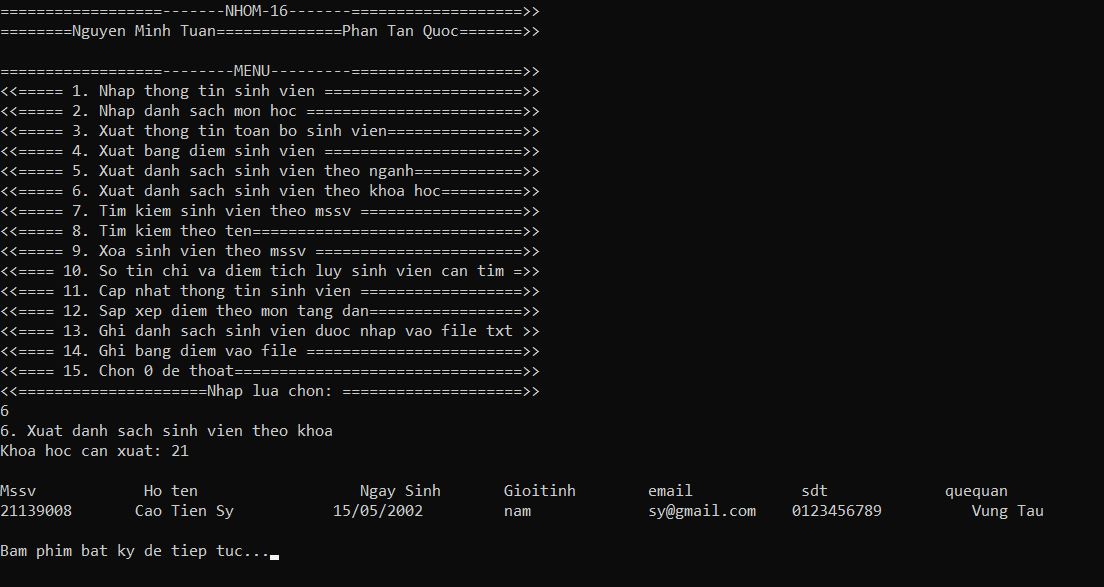
Chương trình sẽ xuất tất cả danh sách sinh viên theo khóa học dựa vào mssv của sinh viên.

- Xuất danh sách sinh viên khóa 20:



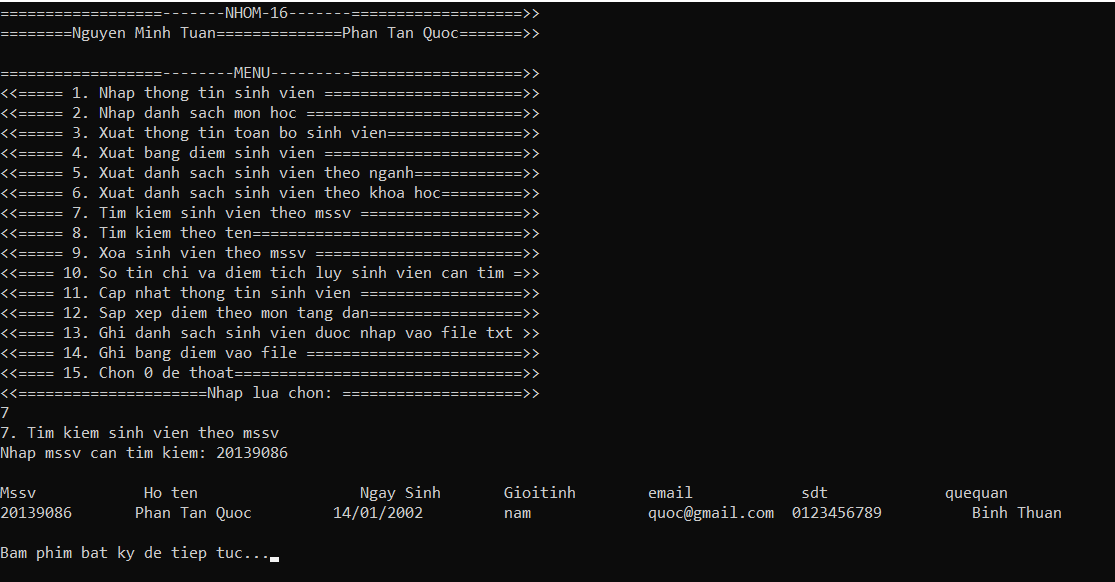
-Xuất Danh sách sinh viên khóa k19:

- Xuất danh sách sinh viên khóa k21:



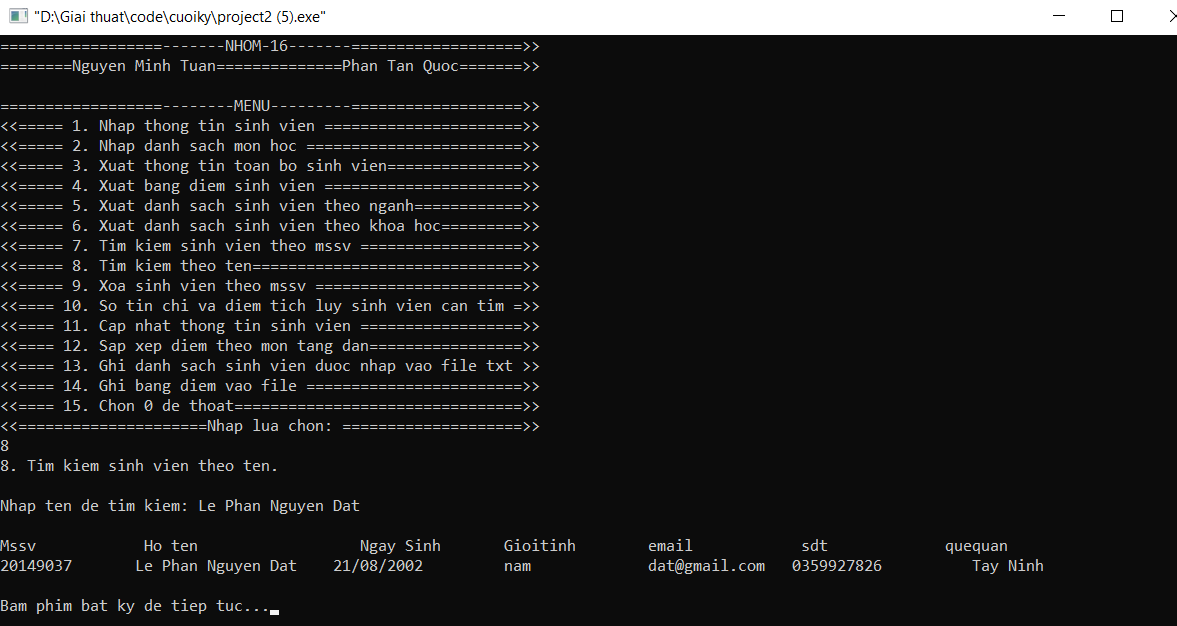
### **4.2.7. Tìm kiếm sinh viên theo mssv**

Chương trình sẽ tìm kiếm thông tin sinh viên bạn cần tìm thông qua mssv của sinh viên và hiển thị tất cả thông tin của sinh viên đó.



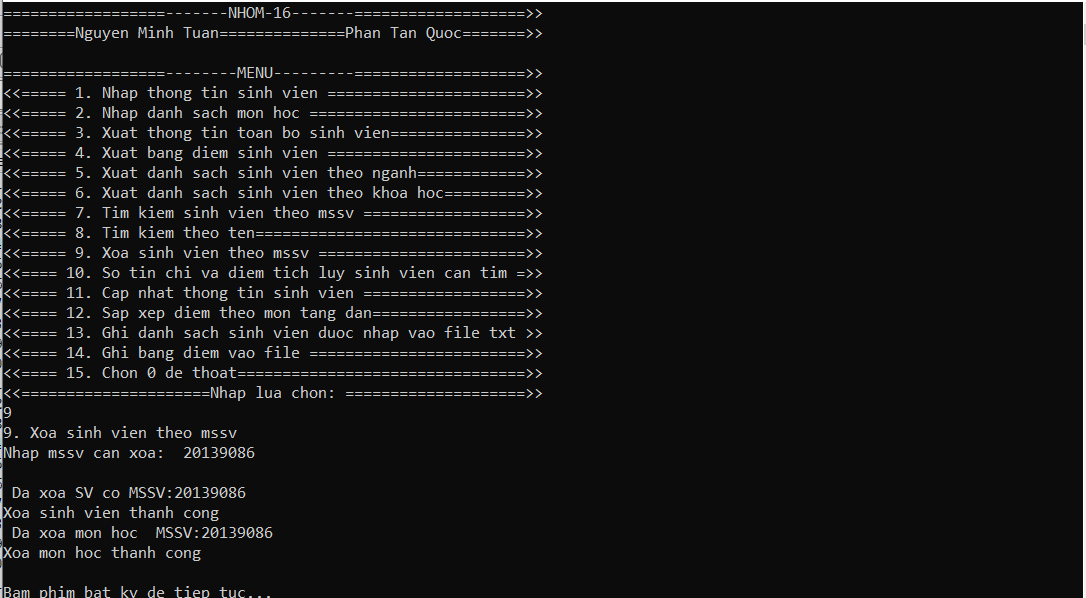
### **4.2.8 Tìm kiếm sinh viên theo tên**

Chương trình sẽ tìm kiếm thông tin của sinh viên thông qua tên của sinh viên cần tìm kiếm.



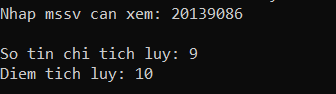
### **4.2.9. Xóa sinh viên theo mssv**

Chương trình sẽ xóa tất cả thông tin của sinh viên bạn cần xóa ra khỏi danh sách thông qua mssv của sinh viên đó.



### **4.2.10. Số tính chỉ và điểm tích lũy sinh viên cần tìm**

Chương trình sẽ hiển thị tổng số tính chỉ đã đạt được của sinh viên và điểm tích lũy của sinh viên thông qua số tính chỉ và số điểm đã nhập ở mục 2



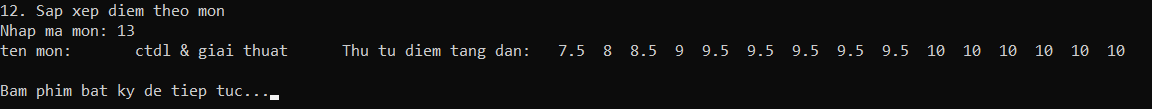
### **4.2.11. Cập nhật thông tin sinh viên**

Chương trình sẽ yêu cầu nhập mssv của sinh viên cần thay đổi bằng 1 mssv mới và điền tất cả các thông tin của sinh viên tương ứng sau đó sẽ cập nhật lại danh sách tất cả sinh viên.



### **4.2.12. Sắp xếp điểm theo môn tăng dần**

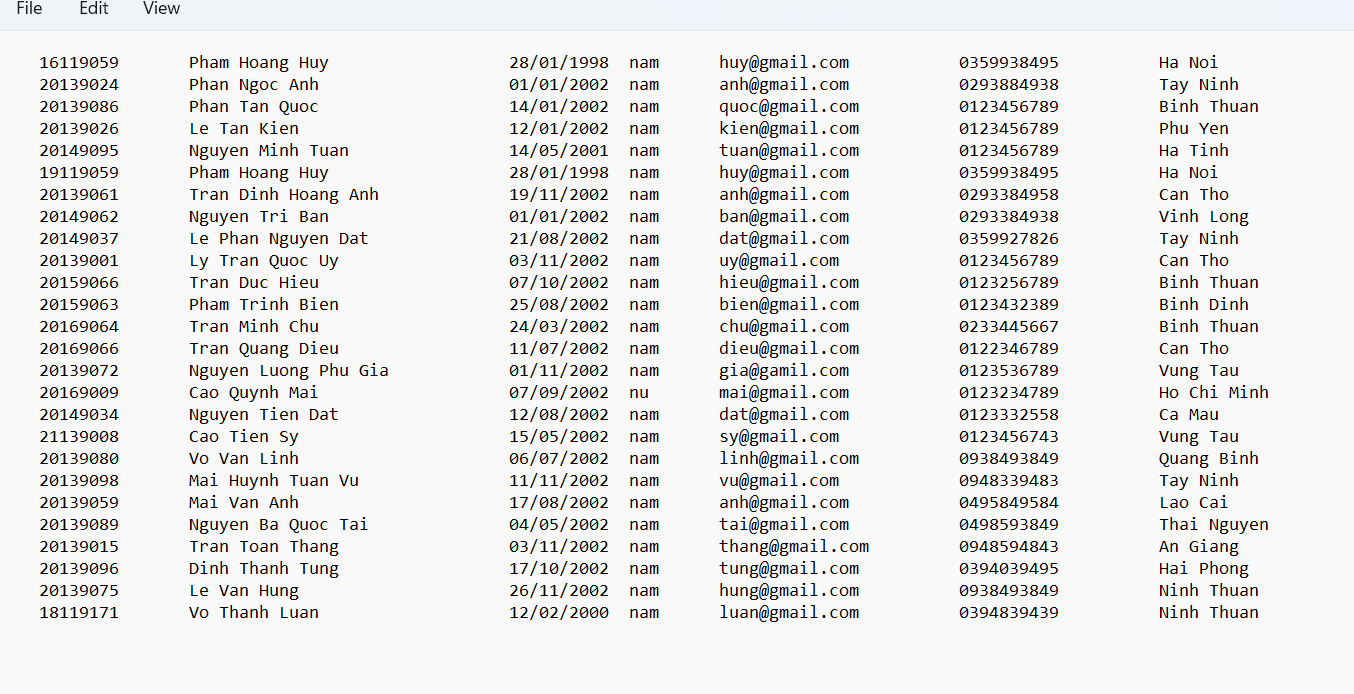
Chương trình sẽ hiển thị thứ tự điểm tăng dần của sinh viên theo mã môn đã nhập

****

### 

### **4.2.13. Ghi danh sách sinh viên được nhập vào file txt.**

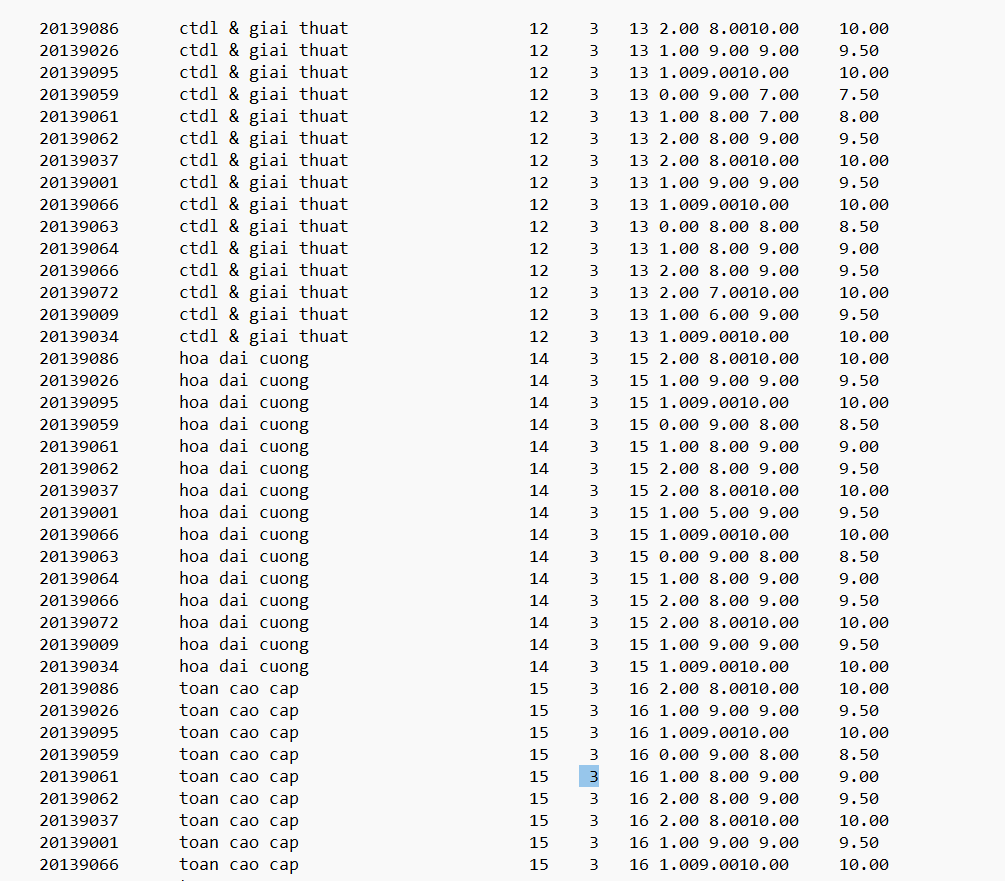
Tất cả thông tin của sinh viên đã nhập ở mục 1 sẽ được ghi vào file sinhvien.txt để lưu trữ.



### 

### **4.2.14. Ghi bảng điểm vào file txt**

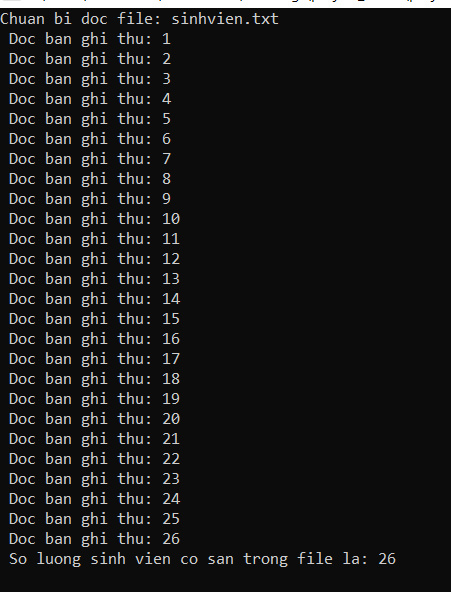
Tất cả thông tin về điểm của sinh viên được nhập ở mục 2 sẽ được ghi vào file bangdiem.txt để lưu trữ.

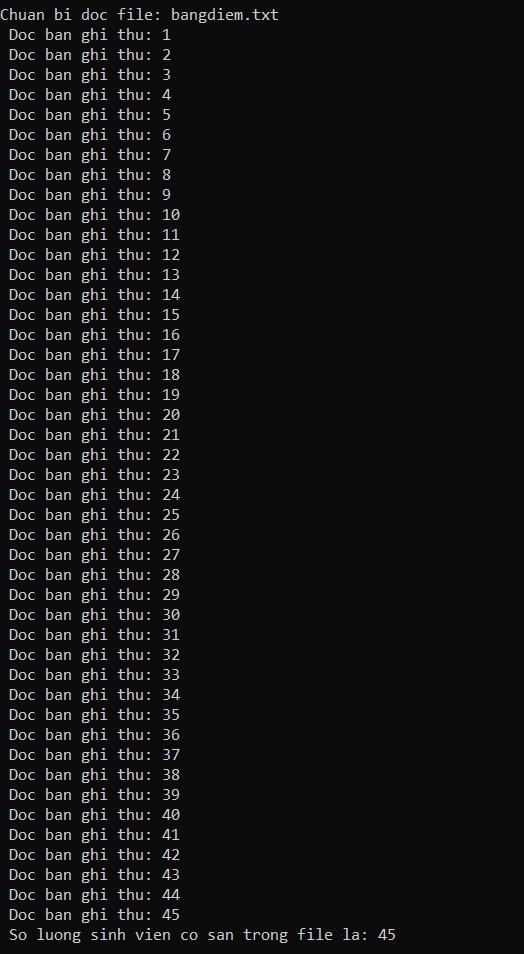
****

### 

### **4.2.15. Đọc file txt**

Chương trình sẽ đọc file sinhvien.txt và file bangdiem.txt đã ghi vào từ mục 13 và 14.





# 

# KẾT LUẬN

Nhìn chung đề tài đã thực hiện được đầy đủ các chức năng của một chương trình quản lý thông tin sinh viên. Quản lý được các thông tin cở bản của sinh viên như thông tin sinh viên, điểm của sinh viên và có thể tra cứu các thông tin cần thiết của sinh viên và

*Ưu điểm:*

• Giao diện gọn gàng, trực quan dễ tiếp cận, dễ làm quen.

• Dung lượng khá nhẹ.

• Chương trình tốn rất ít tài nguyên của hệ thống.

• Chương trình chạy ổn định, được kiểm thử nhiều lần và cho ra kết quả chính

xác, không bị crash, debug, except trong quá trình thực thi đảm bảo đúng yêu

cầu người dùng

*Nhược điểm:*

• Thuật toán chưa tối ưu khi xử lí nhiều luồng thông tin dữ liệu lớn (Đôi lúc

xử lí còn chậm).

• Chưa thể tìm laị dữ liệu khi chẳng may bị mất.

• Tìm kiếm thông tin chưa có nhiều tuỳ chọn

- HƯỚNG PHÁT TRIỂN

• Tối ưu hóa khi xử lí dữ liệu đầu vào lớn.

• Nâng cấp giao diện, cải thiện trải nghiệm người dùng

• Public cơ sở dữ liệu trên mạng internet để có thể truy cập từ nhiều máy khác

Nhau.

- Thông qua đề tài nhóm em đã học hỏi được nhiều kiến thức bổ ích về ngôn ngữ C++, về cấu trúc dữ liệu và giải thuật của 1 chương trình và học được kiến thức về hệ cơ sở dữ liệu

# TÀI LIỆU THAM KHẢO

[1] Giới thiệu về ngôn ngữ lập trình C++ <https://cpp.daynhauhoc.com/0/2-gioi-thieu-ve-ngon-ngu-lap-trinh-c/> (truy cập ngày 27/05/2022).

[2] STRUCT <https://cpp.daynhauhoc.com/9/1-structs/> (truy cập 27/05/2002).

1. Giới thiệu về ngôn ngữ lập trình C++ <https://cpp.daynhauhoc.com/0/2-gioi-thieu-ve-ngon-ngu-lap-trinh-c/> (truy cập ngày 27/05/2022) [↑](#footnote-ref-1)
2. STRUCT <https://cpp.daynhauhoc.com/9/1-structs/> (truy cập 27/05/2002). [↑](#footnote-ref-2)