TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

|  |
| --- |
| A blue and white logo  Description automatically generated |

PBL1-ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN

XÂY DỰNG ỨNG DỤNG QUẢN LÍ SINH VIÊN

NGƯỜI HƯỚNG DẪN: GVC. ThS Trần Hồ Thủy Tiên

SINH VIÊN THỰC HIỆN:

Sinh viên 1: Ngô Văn Huy Lớp: 23T\_NHAT1 Nhóm: 23.99A

Sinh viên 2: Nguyễn Hữu Hoàng Luân Lớp: 23T\_NHAT1 Nhóm: 23.99A

*Đà Nẵng, 6/2024*

LỜI CẢM ƠN

Đầu tiên, nhóm chúng em xin gửi lời cảm ơn sâu sắc và chân thành nhất đến Th.S Trần Hồ Thủy Tiên, người đã không ngừng hướng dẫn, giúp đỡ và động viên chúng em trong suốt quá trình thực hiện đồ án này.

Chúng em cũng xin gửi lời cảm ơn đến toàn thể các thầy cô giáo trong khoa Công Nghệ Thông Tin, đã tận tình truyền đạt kiến thức, tạo điều kiện thuận lợi để chúng em có thể hoàn thành đồ án một cách tốt nhất.

Cuối cùng, chúng em xin gửi lời cảm ơn đến bạn bè, gia đình đã luôn ủng hộ, động viên và giúp đỡ chúng em trong suốt quá trình học tập và thực hiện đồ án.

Mặc dù đã rất cố gắng nhưng không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được sự góp ý chân thành từ phía ban giám khảo và mọi người để đồ án của chúng em được hoàn thiện hơn. Chúng em xin chân thành cảm ơn.

Đà Nẵng, ngày 26 tháng 06 năm 2024

Ngô Văn Huy

Nguyễn Hữu Hoàng Luân

LỜI MỞ ĐẦU

Trong cuộc cách mạng khoa học kỹ thuật và việc ứng dụng công nghệ thông tin trên nhiều lĩnh vực, việc thay đổi cách quản lí từ việc sử dụng giấy tờ tốn nhiều thời gian sang việc quản lí bằng máy tính đã trở nên phổ biến. Một trong những lĩnh vực mà máy tính được sử dụng nhiều nhất là các hệ thống thông tin quản lý.

Hiện nay, quy mô và tính phức tạp của công việc ngày càng cao, do đó việc xây dựng các ứng dụng quản lý danh sách trở nên cần thiết. Trong tương lai, chúng ta có thể xây dựng các ứng dụng hoàn chỉnh để quản lý thông tin Sinh viên.

Mục tiêu của đồ án nhằm xây dựng ứng dụng quản lý danh sách Sinh viên. Để đạt được mục tiêu này, chúng ta sẽ sử dụng các mảng song song để lưu trữ dữ liệu quan hệ, sử dụng các mảng một chiều, áp dụng các thuật toán thao tác với mảng, và viết các hàm xử lý chuỗi. Đồ án sẽ tập trung vào các quá trình quản lý, lưu trữ và thống kê, từ đó xây dựng chương trình quản lý Sinh viên.

Cấu trúc của đồ án sẽ bao gồm các phần sau:

1. Tổng quan đề tài: Giới thiệu về đề tài và mục tiêu nghiên cứu.
2. Cơ sở lí thuyết: Tổng hợp kiến thức liên quan đến quản lý thông tin và lập trình.
3. Tổ chức cấu trúc dữ liệu và thuật toán: Trình bày về cách tổ chức dữ liệu và áp dụng thuật toán trong ứng dụng.
4. Chương trình và kết quả: Hiển thị chương trình đã xây dựng và kết quả thực hiện.
5. Kết luận và hướng phát triển: Tổng kết kinh nghiệm và đề xuất hướng phát triển cho tương lai.

Chúng ta hy vọng rằng đồ án này sẽ đóng góp vào việc cải thiện quá trình quản lý thông tin và giúp tối ưu hóa công việc của các đơn vị quản lý Sinh viên.

MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc170056946)

[LỜI MỞ ĐẦU 2](#_Toc170056947)

[MỤC LỤC 3](#_Toc170056948)

[DANH MỤC HÌNH ẢNH 6](#_Toc170056949)

[CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI 8](#_Toc170056950)

[1.1. Mục tiêu 8](#_Toc170056951)

[1.2. Yêu cầu 8](#_Toc170056952)

[1.3. Mô tả chung về đề tài 8](#_Toc170056953)

[CHƯƠNG 2. CƠ SỞ LÍ THUYẾT 9](#_Toc170056954)

[2.1. Ý tưởng 9](#_Toc170056955)

[2.2. Cơ sở lí thuyết 9](#_Toc170056956)

[2.2.1. Xâu và các hàm có sẵn thao tác với xâu [#include <string.h>] 9](#_Toc170056957)

[2.2.2. File I/O [#include <stdio.h>] 10](#_Toc170056958)

[2.2.3. Thiết kế giao diện 11](#_Toc170056959)

[CHƯƠNG 3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 12](#_Toc170056960)

[3.1. Phát biểu bài toán 12](#_Toc170056961)

[3.2. Cấu trúc dữ liệu 13](#_Toc170056962)

[3.2.1. Cấu trúc Student 13](#_Toc170056963)

[3.2.2. Cấu trúc Date 13](#_Toc170056964)

[3.2.3. Cấu trúc Classe 14](#_Toc170056965)

[3.2.4. Cấu trúc University 14](#_Toc170056966)

[3.3. Khởi tạo danh sách các hằng số và các biến toàn cục đi kèm 14](#_Toc170056967)

[3.3.1. Danh sách các hằng số 14](#_Toc170056968)

[3.3.2. Danh sách các biến toàn cục 15](#_Toc170056969)

[3.4. Thuật toán 16](#_Toc170056970)

[3.4.1. Thuật toán Thêm Sinh viên 16](#_Toc170056971)

[3.4.2. Thuật toán Sắp xếp Sinh viên 17](#_Toc170056972)

[3.4.3. Thuật toán sắp xếp Họ và tên của Sinh viên 19](#_Toc170056973)

[3.4.4. Thuật toán Tìm kiếm bằng ID Sinh viên 19](#_Toc170056974)

[3.4.5. Thuật toán Tìm kiếm bằng Thông tin Sinh viên 20](#_Toc170056975)

[3.4.6. Thuật toán Tìm kiếm bằng Căn cước Công dân của Sinh viên 21](#_Toc170056976)

[3.4.7. Thuật toán Tìm kiếm bằng Số Điện thoại của Sinh viên 21](#_Toc170056977)

[3.4.8. Thuật toán Xóa bằng ID Sinh viên 22](#_Toc170056978)

[3.4.9. Thuật toán Xóa bằng Căn cước Công dân của Sinh viên 23](#_Toc170056979)

[3.4.10. Thuật toán Kiểm tra Cấp mã Sinh viên 23](#_Toc170056980)

[3.4.11. Thuật toán kiểm tra Cấp Email Sinh viên 24](#_Toc170056981)

[CHƯƠNG 4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 25](#_Toc170056982)

[4.1. Tổ chức chương trình 25](#_Toc170056983)

[4.1.1. Danh cách các nguyên mẫu hàm chính 25](#_Toc170056984)

[4.1.2. Danh sách các nguyên mẫu hàm phục vụ 25](#_Toc170056985)

[4.2. Kết quả 26](#_Toc170056986)

[4.2.1. Giao diện chính của chương trình 27](#_Toc170056987)

[4.2.2. Kết quả thực thi của chương trình 27](#_Toc170056988)

[4.2.3. Nhận xét đánh giá 40](#_Toc170056989)

[CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 42](#_Toc170056990)

[5.1. Kết Luận Chung 42](#_Toc170056991)

[5.1.1. Ưu điểm 42](#_Toc170056992)

[5.1.2. Nhược điểm: 42](#_Toc170056993)

[5.2. Hướng phát triển 42](#_Toc170056994)

[TÀI LIỆU THAM KHẢO 43](#_Toc170056995)

[PHỤ LỤC 44](#_Toc170056996)

DANH MỤC HÌNH ẢNH

[Hình 3.1. Thông tin trong một File 12](#_Toc170057040)

[Hình 3.2. Sơ đồ khối toàn chương trình 12](#_Toc170057041)

[Hình 4.1. Menu chính 27](#_Toc170057042)

[Hình 4.2. Chọn lớp học để thêm Sinh viên 27](#_Toc170057043)

[Hình 4.3. Thông báo lớp đã đầy 28](#_Toc170057044)

[Hình 4.4. Nhập thông tin Sinh viên trong lớp học 28](#_Toc170057045)

[Hình 4.5. Nhập lại thông tin do nhập bị sai xót 28](#_Toc170057046)

[Hình 4.6. Nhập tiếp Sinh viên trong lớp đã chọn 29](#_Toc170057047)

[Hình 4.7. Nhập tiếp Sinh viên với Khoa, Khóa khác 29](#_Toc170057048)

[Hình 4.8. Ghi vào File Thêm Sinh viên 29](#_Toc170057049)

[Hình 4.9. Chọn lớp cần sắp xếp 29](#_Toc170057050)

[Hình 4.10. Thông báo sắp xếp lớp khác 30](#_Toc170057051)

[Hình 4.11. Thông báo lỗi chưa có lớp để sắp xếp 30](#_Toc170057052)

[Hình 4.12. File trước khi Sắp xếp Sinh viên 30](#_Toc170057053)

[Hình 4.13. File sau khi Sắp xếp Sinh viên 30](#_Toc170057054)

[Hình 4.14. Giao diện chính của chức năng Tìm kiếm Sinh viên 31](#_Toc170057055)

[Hình 4.15. Thông báo lỗi chưa có lớp để tìm kiếm 31](#_Toc170057056)

[Hình 4.16. Tìm kiếm bằng ID Sinh viên 31](#_Toc170057057)

[Hình 4.17. Thông báo không tìm thấy Sinh viên bằng ID Sinh viên 32](#_Toc170057058)

[Hình 4.18. Tìm kiếm bằng Thông tin Sinh viên 32](#_Toc170057059)

[Hình 4.19. Tìm kiếm bằng Thông tin Sinh viên với những thông tin giống nhau 33](#_Toc170057060)

[Hình 4.20. Tìm kiếm bằng Thông tin Sinh viên với những thông tin giống nhau nhưng có lớp chưa được Sắp xếp. 33](#_Toc170057061)

[Hình 4.21. Thông báo không tìm thấy Sinh viên bằng Thông tin Sinh viên 33](#_Toc170057062)

[Hình 4.22. Tìm kiếm bằng số Căn cước Công dân 34](#_Toc170057063)

[Hình 4.23. Thông báo không tìm thấy Sinh viên bằng số Căn cước Công dân 34](#_Toc170057064)

[Hình 4.24. Tìm kiếm bằng Số điện thoại 34](#_Toc170057065)

[Hình 4.25. Thông báo không tìm thấy Sinh viên bằng Số điện thoại 35](#_Toc170057066)

[Hình 4.26. Tìm kiếm bằng chức năng đã chọn hoặc về Menu chính 35](#_Toc170057067)

[Hình 4.27. Giao diện chính của chức năng Xóa Sinh viên 35](#_Toc170057068)

[Hình 4.28. Thông báo lỗi chưa có lớp để Xóa Sinh viên 35](#_Toc170057069)

[Hình 4.29. Xóa bằng ID Sinh viên thành công 36](#_Toc170057070)

[Hình 4.30. Không tìm thấy thông tin để xóa bằng ID Sinh viên 36](#_Toc170057071)

[Hình 4.31. Xóa bằng số Căn cước Công dân 37](#_Toc170057072)

[Hình 4.32. Không tìm thấy thông tin để xóa băng Căn cước Công dân 37](#_Toc170057073)

[Hình 4.33. File trước khi Xóa Sinh viên 37](#_Toc170057074)

[Hình 4.34. File sau khi Xóa Sinh viên 37](#_Toc170057075)

[Hình 4.35. Xem thông tin Lớp học 38](#_Toc170057076)

[Hình 4.36. Thông báo lỗi chưa có lớp để in ra 38](#_Toc170057077)

[Hình 4.37. Thông báo lỗi chưa có lớp để cấp Mã ID 38](#_Toc170057078)

[Hình 4.38. Thông báo lỗi chưa sắp xếp Lớp trong cấp Mã ID 39](#_Toc170057079)

[Hình 4.39. Thông báo đã cấp Mã ID thành công 39](#_Toc170057080)

[Hình 4.40. Tiếp tục cấp Mã ID hoặc trở về Menu chính 39](#_Toc170057081)

[Hình 4.41. Thông báo lỗi chưa có lớp để cấp Email 39](#_Toc170057082)

[Hình 4.42. Thông báo lỗi chưa sắp xếp lớp trong cấp Email 40](#_Toc170057083)

[Hình 4.43. Thông báo đã cấp Email thành công 40](#_Toc170057084)

[Hình 4.44. Tiếp tục cấp Email hoặc về Menu chính 40](#_Toc170057085)

# TỔNG QUAN ĐỀ TÀI

## Mục tiêu

- Sử dụng các mảng song song để lưu dữ liệu quan hệ.

- Sử dụng các mảng một chiều.

- Sử dụng các thuật toán thao tác với mảng.

- Sử dụng các hàm xử lý chuỗi.

## Yêu cầu

- Viết chương trình bằng C/C++ để giải quyết bài toán.

- Viết các hàm để xây dựng các chức năng.

## Mô tả chung về đề tài

- Chương trình cho phép quản lý khoản 100 Sinh viên/mỗi lớp. Gồm menu sau:

1. Thêm Sinh viên.
2. Sắp xếp danh sách.
3. Xóa Sinh viên.
4. Tìm Sinh viên.
5. Cấp Mã Sinh viên.
6. Cấp email.
7. In ra danh sách.
8. Thoát.

- Mỗi một lớp sẽ được lưu trên 1 file.

- Chức năng Thêm Sinh viên: Nhập Họ lót, Tên, Ngày sinh, Giới tính, Địa chỉ, Số điện thoại, Căn cước Công dân;...

- Chức năng Cấp Mã Sinh viên chương trình tự động cấp sau khi sắp xếp danh sách theo thứ tự Alphabet của Tên và Họ lót. Nếu chưa sắp xếp chương trình sẽ thông báo lỗi.

- Chức năng địa chỉ email chương trình tự động cấp sau khi cấp Mã Sinh viên. Nếu chưa cấp mã chương trình sẽ thông báo lỗi.

# CƠ SỞ LÍ THUYẾT

## Ý tưởng

- Để quản lý Sinh viên ta sẽ xem Sinh viên như một kiểu struct và trong struct đó sẽ gồm thông tin cá nhân như: Lớp, Tên, Giới tính, Căn cước Công Dân, Số điện thoại;… .Sau đó để quản lí danh sách gồm nhiều Sinh viên ta sẽ dùng mảng có kiểu dữ liệu là kiểu struct nói trên.

## Cơ sở lí thuyết

### Xâu và các hàm có sẵn thao tác với xâu [#include <string.h>]

#### Hàm strcmp()

- Nguyên mẫu hàm:

|  |
| --- |
| int strcmp(const char \*str1, const char \*str2) |

- Hàm **strcmp()** trong ngôn ngữ C được sử dụng để so sánh hai chuỗi. Trong đó, str1 là chuỗi đầu tiên để được so sánh, str2 là chuỗi thứ hai để được so sánh. Hàm **strcmp()** trả về các giá trị như sau:

* Nếu giá trị trả về nhỏ hơn 0, thì hàm chỉ rằng str1 là ngắn hơn str2.
* Nếu giá trị trả về lớn hơn 0, thì hàm chỉ rằng str2 là ngắn hơn str1.
* Nếu giá trị trả về bằng 0, thì hàm chỉ rằng str1 là bằng str2.

#### Hàm strcpy()

- Nguyên mẫu hàm:

|  |
| --- |
| char \*strcpy(char \*dest, const char \*src); |

- Hàm **strcpy()** trong ngôn ngữ C được sử dụng để sao chép nội dung từ một chuỗi nguồn (src) sang một chuỗi đích (dest). Trong đó: *dest* là con trỏ trỏ tới mảng chứa chuỗi đích, *src* là chuỗi nguồn để sao chép. Hàm **strcpy()** trả về một con trỏ trỏ tới chuỗi đích sau khi đã sao chép xong.

#### Hàm strlen()

- Nguyên mẫu hàm:

|  |
| --- |
| size\_t strlen(const char \*str) |

- Hàm **strlen()** trong ngôn ngữ C tính toán độ dài của chuỗi str (không bao gồm ký tự NULL kết thúc). Đây là một hàm hữu ích để xác định số lượng ký tự trong một chuỗi.

#### Hàm strcat()

- Nguyên mẫu hàm:

|  |
| --- |
| char \*strcat(char \*dest, const char \*src); |

- Hàm **strcat()** trong ngôn ngữ C được sử dụng để phụ thêm (append) chuỗi được trỏ tới bởi src vào cuối chuỗi được trỏ tới bởi dest. Trong đó, *dest* là con trỏ trỏ tới mảng chứa chuỗi đích, *src* là chuỗi nguồn để được phụ thêm (append) vào cuối. Hàm **strcat()** trả về một con trỏ trỏ tới chuỗi kết quả sau khi đã nối chuỗi nguồn.

### File I/O [#include <stdio.h>]

- Trong lập trình C, bạn có thể thực hiện các thao tác trên **File** như đọc, ghi, di chuyển và tạo **File** mới một cách dễ dàng bằng cách sử dụng một số hàm và thành phần trong hệ thống **File I/O** của C.

- Khai báo:

|  |
| --- |
| FILE \*fopen |

- Làm việc với File:

1. Mở File:

* Nguyễn mẫu hàm:

|  |
| --- |
| FILE \*fopen(const char \*name, const char \*mode) |

* Muốn thao tác với Tệp trước hết phải mở Tệp.
* Mode kiểu mở: “ w” mở để ghi, “r”: mở để đọc, “a”: mở để ghi vào cuối Tệp.

1. Đóng File:

* Phải đóng Tệp khi không làm việc với nó nữa.
* Nguyễn mẫu hàm:

|  |
| --- |
| int fclose(FILE \*fp) |

1. Đọc dữ liệu theo định dạng fscanf():

* Nguyên mẫu hàm:

|  |
| --- |
| fscanf int fscanf(FILE \*fp, const char \*chuỗi\_điều\_khiển, danh\_sách\_đối) |

1. Ghi dữ liệu theo định dạng fprintf():

* Nguyễn mẫu hàm:

|  |
| --- |
| int fprintf(FILE \*fp, const char \*chuỗi\_điều\_khiển, danh\_sách\_đối). |

### Thiết kế giao diện

- Thiết kế giao diện đơn giản và chạy trên giao diện Console (giao diện Dòng lệnh).

- Thông tin thông báo tới người sử dụng ngắn gọn, rõ ràng.

# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

- Input: Nhập dữ liệu từ bàn phím.

- Output: Có 2 dạng:

* Hiển thị trên màn hình Console.
* Ghi dữ liệu vào Tệp tin văn bản (File Text):
* Tên File và Cấu trúc: Được tạo ra từ 2 kí tự Khóa + Mã Khoa + “\_” + Tên Khoa + Số thứ tự lớp học.
* Thông tin trong File: gồm các thông tin của Sinh viên, mỗi thông tin của Sinh viên được ngăn cách bởi dấu “\n” và các thông tin của Sinh viên ngăn cách nhau bởi dấu “;;”.

A screen shot of a computer

Description automatically generated

Thông tin trong một File

- Sơ đồ toàn chương trình:

A diagram of a computer

Description automatically generated

Sơ đồ khối toàn chương trình

## Cấu trúc dữ liệu

### Cấu trúc Student

|  |
| --- |
| typedef struct Students{  char Last\_Name[MAX\_OF\_NAME];  char First\_Name[MAX\_OF\_NAME];  char Sex[MAX\_OF\_SEX];  Date Birth;  char Address[MAX\_OF\_ADDRESS];  char Student\_ID[MAX\_OF\_ID];  char Email[MAX\_OF\_EMAIL];  char Falcuty[MAX\_OF\_FALCUTY];  char Phone\_Number[MAX\_OF\_PHONE\_NUMBER];  char Citizen\_ID[MAX\_OF\_CID];  } Student; |

- Struct định nghĩa thông tin Sinh viên, bao gồm các thuộc tính:

* Last\_Name : Họ của Sinh viên.
* First\_Name : Tên của Sinh viên.
* Sex : Giới tính của Sinh viên.
* Birth : Ngày, tháng, năm sinh của Sinh viên.
* Address : Địa chỉ của Sinh viên.
* Student\_ID : Mã số Sinh viên của Sinh viên.
* Email : Email của Sinh viên.
* Falcuty : Khoa của Sinh viên.
* Phone\_Number : Số điện thoại của Sinh viên.
* Citizen\_ID : Số căn cước công dân của Sinh viên.

### Cấu trúc Date

|  |
| --- |
| typedef struct Date {  int Day;  int Month;  int Year;  } Date; |

- Struct định nghĩa ngày tháng năm, bao gồm các thuộc tính “Day”, “Month”, “Year”.

### Cấu trúc Classe

|  |
| --- |
| typedef struct Classe{  int Check\_Sort;  int Number\_of\_Student;  char Name\_of\_Class[MAX\_OF\_CLASS];  } Classe; |

- Struct định nghĩa thông tin về lớp học, bao gồm các thuộc tính:

* Check\_Sort: Kiểm tra xem lớp đã được sắp xếp hay chưa.
* Number\_of\_Student: Quản lí số lượng Sinh viên trong một lớp học.
* Name\_of\_Class[MAX\_OF\_CLASS]: Quản lí tên lớp học.

### Cấu trúc University

|  |
| --- |
| struct University{  char Check\_ID\_falcuty[MAX\_OF\_FALCUTY][MAX\_OF\_ID];  Classe Classroom[MAX\_OF\_CLASS];  }; |

- Struct định nghĩa số hồ sơ trong lớp học, bao gồm các thuộc tính:

* Check\_ID\_falcuty: Kiểm tra Khoa đã được cấp ID chưa. Dùng để cấp mã ID kết hợp với *“Check\_ID\_class”* trong struct “Classe”.
* Classroom: Mảng dùng để lưu thông tin hồ sơ lớp học.

## Khởi tạo danh sách các hằng số và các biến toàn cục đi kèm

### Danh sách các hằng số

|  |
| --- |
| #define MAX\_SIZE 300  #define MAX\_OF\_ADDRESS 30  #define MAX\_OF\_NAME 30  #define MAX\_OF\_SEX 7  #define MAX\_OF\_EMAIL 30  #define MAX\_OF\_YEAR 5  #define MAX\_OF\_FALCUTY 6  #define MAX\_OF\_CLASS\_IN\_FALCUTY 5  #define MAX\_OF\_STUDENTS\_LIST 101  #define MAX\_OF\_CLASS 30  #define MAX\_OF\_ID 10  #define MAX\_OF\_PHONE\_NUMBER 11  #define MAX\_OF\_CID 13  #define MAX\_OF\_RE\_CHOOSE 10 |

### 

### Danh sách các biến toàn cục

|  |
| --- |
| int Number\_of\_Class\_in\_School = 0;  int Class\_Index = 0;  int Number\_of\_Student\_Index = 0;  int re\_choose[MAX\_OF\_RE\_CHOOSE]; |

- Chức năng của các biến:

* Number\_of\_Class\_in\_School: Dùng để quản lí tổng số lớp học của toàn trường trong một niên khóa nào đó.
* Class\_Index: Dùng để quản lí số lượng lớp học đang sử dụng.
* Number\_of\_Student\_Index: Dùng để quản lí số lượng Sinh viên của lớp đang được sử dụng.
* re\_choose[MAX\_OF\_RE\_CHOOSE]: Dùng để lưu lại thông tin nội dung để in ra danh sách Sinh viên.

## Thuật toán

### Thuật toán Thêm Sinh viên

|  |
| --- |
| void addStudent(); |

- Input: Không có tham số đầu vào.

- Output: File lớp sau khi đã thêm Sinh viên.

- Thuật toán:

* Bước 1: Nhập choose\_year, choose\_falcuty → Kiểm tra đúng định dạng.

→ Tạo tên chuỗi create = choose\_year + choose\_falcuty.

* Bước 2: Chuyển đổi kí hiệu cho choose\_falcuty.
  + ‘102’: IT1;
  + ‘107’: H1;
  + ‘105’: E1;
  + ‘123’: PFIEV1;
  + ‘121’: KT1;
* Bước 3: Lặp i = 2 → MAX\_OF\_CLASS\_IN\_FALCUTY.
* create[strlen(create) – 1]++; và in ra màn hình.
* Bước 4: Nhập choose\_class để chọn lớp muốn thêm Sinh viên → Kiểm tra

định dạng.

* Bước 5: create[strlen(create) – 1] = ‘\0’;

sprintf(name\_class, “%s%d”, create, choose\_class);

* Bước 6: Lặp i = 0 → Number\_of\_Class\_in\_School.
* Nếu strcmp (University.Classroom[i].Name\_of\_Class, name\_class) = 0

→ found = 1; Class\_Index = i;

* Bước 7: Nếu found = 0.

→ Number\_of\_Class\_in\_School++;

Class\_Index = Number\_of\_Class\_in\_School;

strcpy(University.Classroom[Class\_Index].Name\_of\_Class

, name\_class);

* Bước 8: Number\_of\_Student\_Index=University.Classroom[Class\_Index]

.Number\_of\_Student;

* Bước 9: Nếu Number\_of\_Student\_Index=MAX\_OF\_STUDENTS\_LIST

– 1

→ Thông báo lớp đã đầy.

* Bước 10: Number\_of\_Student\_Index++;
* Bước 11: Nhập thông tin cho Sinh viên.
* Bước 12: Chọn choose:

1. Nhập lại thông tin Sinh viên.
2. Xác nhận thông tin Sinh viên.

* Bước 13: Kiểm tra đúng định dạng. Nếu sai quay lại Bước 12.

Nếu choose = 2:

* Nhập các thông tin cần sửa. switch(re\_choose):
  + - ‘1’: Lastname of Student.
    - ‘2’: Firstname of Student.
    - ‘3’: Date of Birth of Student.
    - ‘4’: Gender of Student.
    - ‘5’: Address of Student.
    - ‘6’: Phone Number of Student.
    - ‘7’: Citizen ID of Student.
* Nhập lại các thông tin cần nhập lại cho Sinh viên.
* Bước 14: Gán Student\_ID và Email của Sinh viên là ‘x’.
* Bước 15: Mở file và ghi vào file với tên file là tên Lớp được tạo và đã chọn

ở bước 4.

* Bước 16: University.Classroom[Class\_Index].Check\_Sort = 0;
* Bước 17: Nhập tiếp thông tin cho Sinh viên trong Lớp. Nếu có quay lại

bước 9.

* Bước 18: Nhập tiếp thông tin cho Sinh viên khác. Nếu có quay lại bước 1.
* Bước 19: Kết thúc.

### Thuật toán Sắp xếp Sinh viên

|  |
| --- |
| void sortStudent(); |

- Input: Không có tham số đầu vào.

- Output: File lớp sau khi được sắp xếp.

- Thuật toán:

* Bước 1: Nếu Number\_of\_Class\_in\_School = 0

→ Thông báo chưa có Lớp để sắp xếp và thoát hàm.

* Bước 2: Lặp i = 1 → Number\_of\_Class\_in\_School.
* Nếu University.Classroom[i].Check\_Sort = 0

→ In tên lớp ra màn hình.

* Ngược lại thì thông báo “Toàn bộ lớp đã được sắp xếp” và thoát.
* Bước 3: Nhập choose → Kiểm tra đúng định dạng.
* Bước 4: Mở tệp với tên Lớp đã được chọn.
* Bước 5: QuickSort LastName và FirstName cho toàn bộ Sinh viên trong

lớp.

* Bước 6: University.Classroom[a[choose]].Check\_Sort = 1;
* Bước 7: Tìm falcuty và year bằng tên Lớp đã chọn.
* switch( atoi( falcuty )).
* ‘102’: k = 1.
* ‘107’: k = 2.
* ‘105’: k = 3.
* ‘123’: k = 4.
* ‘121’: k = 5.
* Bước 8: Nếu strcmp(University.Check\_ID\_falcuty[k],"") = 0.

→sprintf(University.Check\_ID\_falcuty[k],"%s%s0000",falcuty,year);

* Bước 9: Lặp i = 1 → Number\_of\_Student\_Index.
* Nếu strcmp(Students[i]->Student\_ID,"x") = 0.

→ increase\_String(University.Check\_ID\_falcuty[k]);

strcpy(Students[i]->Student\_ID,University. Check\_ID\_

falcuty[k]);

strcpy(Students[i]->Email, Students[i]->Student\_ID);

strcat(Students[i]->Email,"@sv1.dut.udn.vn");

* Bước 10: Mở tệp và ghi toàn bộ thông tin Sinh viên vào.
* Bước 11: Sắp xếp tiếp cho Lớp khác. Nếu có quay lại Bước 1.
* Bước 12: Kết thúc.

### Thuật toán sắp xếp Họ và tên của Sinh viên

|  |
| --- |
| void quickSortByLastName(Student\* arr[], int low, int high);  void quickSortByFirstName(Student\* arr[], int low, int high); |

- Input:

* arr[]: Một mảng con trỏ đến các đối tượng Student.
* low và high: Hai chỉ số đánh dấu phạm vi của mảng cần sắp xếp.

- Output: Không trả về giá trị.

- Thuật toán:

* Bước 1: Nếu low < high.
* Lặp j = low → high -1.
* Nếu strcmp(arr[j]->First\_Name, pivot->First\_Name) < 0.
* i = (low – 1) ++;

Hoán đổi arr[i], arr[j];

* Hoán đổi arr[i + 1], arr[high].
* quickSortByFirstName(arr, low, i);
* quickSortByFirstName(arr, i + 2, high);
* Bước 2: Kết thúc.

### Thuật toán Tìm kiếm bằng ID Sinh viên

|  |
| --- |
| void searchStudent\_ID(); |

- Input: Không có tham số đầu vào.

- Output: Không trả về giá trị.

- Thuật toán:

* Bước 1: Nhập ID Sinh viên.
* Bước 2: Kiểm tra đúng định dạng. Nếu sai quay lại Bước 1.
* Bước 3: Tìm falcuty và name với 5 kí tự đầu của ID Sinh viên.
* Bước 4: switch (atoi (falcuty)).
* ‘102’: k = 1.
* ‘107’: k = 2.
* ‘105’: k = 3.
* ‘123’: k = 4.
* ‘121’: k = 5.
* Bước 5: Nếu strcmp(University.Check\_ID\_falcuty[k],ID)>=0.
* Lặp i = 1 → Number\_of\_Class\_in\_School.
* Tìm tên lớp trong Khoa hiện có trong trường.
* Lặp j = 1 → Number\_of\_Student\_Index.
* Nếu tìm thấy Sinh viên → found = j.
* Bước 6: Nếu found != -1 → In ra thông tin Sinh viên. Ngược lại thì

thông báo “Không tìm thấy Sinh viên với ID đó”.

* Bước 7: Tìm tiếp với ID Sinh viên khác. Quay lại bước 1.
* Bước 8: Kết thúc.

### Thuật toán Tìm kiếm bằng Thông tin Sinh viên

|  |
| --- |
| void searchStudent\_info(); |

- Input: Không có tham số đầu vào.

- Output: Không trả về giá trị.

- Thuật toán:

* Bước 1: Nhập year.
* Bước 2: Kiểm tra đúng định dạng. Nếu sai quay lại Bước 1.
* Bước 3: Nhập falcuty.
* Bước 4: Kiểm tra đúng định dạng. Nếu sai quay lại Bước 3.
* Bước 5: Nhập Họ và tên Sinh viên cần tìm.
* Bước 6: Lặp i = 1 → Number\_of\_Class\_in\_School.
* Tạo biến name\_class từ thông tin year và falcuty nhập vào.
* Kiểm tra xem những Lớp trong Khoa đó đã được sắp xếp hay chưa.
* Mở tệp.
* found = binarySearch(Students,1, Number\_of\_Student\_Index, x);.
* Bước 7: Nếu found != -1 thì in ra thông tin Sinh viên. Đồng thời tìm kiếm

2 Sinh viên kề nhau trong Lớp với thông tin tìm kiếm. Ngược lại thì thông báo “Không tìm thấy Sinh viên”.

* Bước 8: Tìm kiếm với thông tin Sinh viên khác. Quay lại bước 1.
* Bước 9: Kết thúc.

### Thuật toán Tìm kiếm bằng Căn cước Công dân của Sinh viên

|  |
| --- |
| void searchStudent\_ID(); |

- Input: Không có tham số đầu vào.

- Output: Không trả về giá trị.

- Thuật toán:

* Bước 1: Nhập Citizen ID của Sinh viên.
* Bước 2: Lặp i = 1 → Number\_of\_Class\_in\_School.
* Mở tệp Lớp học.
* Lặp j = 1 → Number\_of\_Student\_Index.
* Nếu strcmp(Students[j]->Citizen\_ID,Citizen\_ID) = 0.
* Nếu tìm thấy Sinh viên → found = j.
* Bước 3: Nếu found != -1 → In ra thông tin Sinh viên. Ngược lại thì

thông báo “Không tìm thấy Sinh viên với Citizen ID đó”.

* Bước 4: Tìm tiếp với Citizen ID Sinh viên khác. Quay lại bước 1.
* Bước 5: Kết thúc.

### Thuật toán Tìm kiếm bằng Số Điện thoại của Sinh viên

|  |
| --- |
| void searchStudent\_Phone\_Number(); |

- Input: Không có tham số đầu vào.

- Output: Không trả về giá trị.

- Thuật toán:

* Bước 1: Nhập Phone Number của Sinh viên.
* Bước 2: Kiểm tra đúng định dạng. Nếu sai quay lại Bước 1.
* Bước 3: Lặp i = 1 → Number\_of\_Class\_in\_School.
* Mở tệp Lớp học.
* Lặp j = 1 → Number\_of\_Student\_Index.
* Nếu strcmp(Students[j]->Phone\_Number,Phone\_Number) = 0.
* Nếu tìm thấy Sinh viên → found = j.
* Bước 4: Nếu found != -1 → In ra thông tin Sinh viên. Ngược lại thì thông

báo “Không tìm thấy Sinh viên với Phone Number đó”.

* Bước 5: Tìm tiếp với Phone Number Sinh viên khác. Quay lại bước 1.
* Bước 6: Kết thúc.

### Thuật toán Xóa bằng ID Sinh viên

|  |
| --- |
| void RemoveStudent\_ID(); |

- Input: Không có tham số đầu vào.

- Output: File lớp sau khi xóa Sinh viên.

- Thuật toán:

* Bước 1: Nhập ID Sinh viên.
* Bước 2: Kiểm tra đúng định dạng. Nếu sai quay lại Bước 1.
* Bước 3: Tìm falcuty và name với 5 kí tự đầu của ID Sinh viên.
* Bước 4: switch (atoi (falcuty)).
* ‘102’: k = 1.
* ‘107’: k = 2.
* ‘105’: k = 3.
* ‘123’: k = 4.
* ‘121’: k = 5.
* Bước 5: Nếu strcmp(University.Check\_ID\_falcuty[k],ID)>=0.
* Lặp i = 1 → Number\_of\_Class\_in\_School.
* Tìm tên lớp trong Khoa hiện có trong Trường.
* Lặp j = 1 → Number\_of\_Student\_Index.
* Nếu tìm thấy Sinh viên → found = j.
* Bước 6: Nếu found != -1 → In ra thông tin Sinh viên. Ngược lại thì thông

báo “Không tìm thấy Sinh viên với ID đó” và thoát.

* Bước 7: Xác nhận có muốn xóa Sinh viên không.
* Bước 8: Xóa tiếp với ID Sinh viên khác. Quay lại bước 1.
* Bước 9: Kết thúc.

### Thuật toán Xóa bằng Căn cước Công dân của Sinh viên

|  |
| --- |
| void RemoveStudent\_CID(); |

- Input: Không có tham số đầu vào.

- Output: File lớp sau khi xóa Sinh viên.

- Thuật toán:

* Bước 1: Nhập Citizen ID của Sinh viên.
* Bước 2: Lặp i = 1 → Number\_of\_Class\_in\_School.
* Mở tệp Lớp học.
* Lặp j = 1 → Number\_of\_Student\_Index.
* Nếu strcmp(Students[j]->Citizen\_ID,Citizen\_ID) = 0.
* Nếu tìm thấy Sinh viên → found = j.
* Bước 3: Nếu found != -1 → In ra thông tin Sinh viên. Ngược lại thì thông

báo “Không tìm thấy Sinh viên với Citizen ID đó” và thoát.

* Bước 4: Xác nhận có muốn xóa Sinh viên không.
* Bước 5: Tìm tiếp với Citizen ID Sinh viên khác. Quay lại bước 1.
* Bước 6: Kết thúc.

### Thuật toán Kiểm tra Cấp mã Sinh viên

|  |
| --- |
| void checkID(); |

- Input: Không có tham số đầu vào.

- Output: Không trả về giá trị.

- Thuật toán:

* Bước 1: Nếu Number\_of\_Class\_in\_School = 0.

→ Thông báo chưa có Lớp để kiểm tra và thoát hàm.

* Bước 2: In ra danh sách toàn bộ Lớp học hiện có trong trường và chọn lớp cần cấp mã Sinh viên.
* Bước 3: Nếu University.Classroom[choose].Check\_Sort = 0 thì in ra

thông báo “Lớp chưa được sắp xếp”. Ngược lại thì in ra “Lớp đã được cấp ID thành công”.

* Bước 4: Kiểm tra việc cấp ID cho Lớp khác. Quay lại bước 1.
* Bước 5: Kết thúc.

### Thuật toán kiểm tra Cấp Email Sinh viên

|  |
| --- |
| void checkEmail(); |

- Input: Không có tham số đầu vào.

- Output: Không trả về giá trị.

- Thuật toán:

* Bước 1: Nếu Number\_of\_Class\_in\_School = 0.

→ Thông báo chưa có Lớp để kiểm tra và thoát hàm.

* Bước 2: In ra danh sách toàn bộ Lớp học hiện có trong trường và chọn lớp

cần cấp Email.

* Bước 3: Nếu University.Classroom[choose].Check\_Sort = 0 thì in ra

thông báo “Lớp chưa được sắp xếp”. Ngược lại thì in ra “Lớp đã được cấp Email thành công”.

* Bước 4: Kiểm tra việc cấp Email cho Lớp khác. Quay lại bước 1.
* Bước 5: Kết thúc.

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

### Danh cách các nguyên mẫu hàm chính

|  |  |
| --- | --- |
| Nguyên mẫu hàm | Chức năng |
| void addStudent(); | Nhập thông tin Sinh viên |
| void sortStudent(); | Sắp xếp danh sách Sinh viên lớp học |
| void removeStudent(); | Xóa Sinh viên |
| void searchStudent(); | Tìm kiếm Sinh viên |
| void printClass(); | In ra danh sách lớp học |
| void checkID(); | Kiểm tra cấp Mã số Sinh viên cho lớp |
| void checkEmail(); | Kiểm tra cấp Email cho lớp |

### Danh sách các nguyên mẫu hàm phục vụ

|  |  |
| --- | --- |
| Nguyên mẫu hàm | Chức năng |
| void reset(); | Cài đặt lại toàn bộ dữ liệu |
| void free\_Data(); | Cấp bộ nhớ cho con trỏ Student |
| void add\_Data(); | Giải phóng bộ nhớ cho con trỏ Student |
| void logo(); | In ra logo (tilte) chương trình |
| int readFile(char\* name\_class); | Đọc file lớp học |
| void rewiteStudents(int i , FILE \*file); | Viết thông tin Sinh viên vào file |
| char\* nameStr(char \*s); | Định dạng tên đúng |
| void reset(); | Cài đặt lại toàn bộ dữ liệu |
| void rewiteStudents(int i , FILE \*file); | Viết thông tin Sinh viên vào file |
| void quickSortByLastName(Student\* arr[], int low, int high); | Sắp xếp Sinh viên theo Họ lót |
| char\* increase\_String(char\* s); | Tăng chuỗi |
| void quickSortByFirstName(Student\* arr[], int low, int high); | Sắp xếp Sinh viên theo Tên |
| void QuickSortClass(Classe s[], int left, int right); | Sắp xếp tên các lớp học hiện có |
| int binarySearch(Student\* arr[], int l, int r, Student\* x); | Tìm kiếm nhị phân |
| void conditionSearch(); | In ra bảng những thông tin được yêu cầu |
| void printStudent(int found); | In thông tin Sinh viên ra màn hình với lựa chọn tương ứng |
| void RemoveStudent\_Citizen\_ID(); | Xóa Sinh viên khỏi danh sách lớp học bằng CCCD |
| void RemoveStudent\_ID(); | Xóa Sinh viên khỏi danh sách lớp học bằng MSSV |
| void searchStudent\_ID(); | Tìm kiếm Sinh viên bằng MSSV |
| void searchStudent\_Info(); | Tìm kiếm Sinh viên bằng thông tin khác |
| void searchStudent\_Citizen\_ID(); | Tìm kiếm Sinh viên bằng CCCD |
| void searchStudent\_Phone\_Number(); | Tìm kiếm Sinh viên bằng số điện thoại |

## 

## Kết quả

- Chương trình chạy trên giao diện Console với đầy đủ, hiệu quả các chức năng quản lí danh sách Sinh viên của một trường hay một tổ chức như các chức năng: Thêm Sinh viên, Sắp xếp danh sách, Xóa Sinh viên, Tìm kiếm Sinh viên, Cấp mã Sinh viên, Cấp Email và In ra danh sách lớp học.

### Giao diện chính của chương trình

#### Menu chính

A screenshot of a computer

Description automatically generated

Menu chính

### Kết quả thực thi của chương trình

#### Chức năng Thêm Sinh viên

- Để thêm một Sinh viên, đầu tiên ta chọn niên khóa, mã Khoa, lớp của Sinh viên cần được thêm vào.

A screenshot of a computer

Description automatically generated

Chọn lớp học để thêm Sinh viên

- Nếu lớp học đã đủ Sinh viên (100 Sinh viên/lớp), màn hình sẽ in ra thông báo lớp đã đầy.

A screenshot of a computer

Description automatically generated

Thông báo lớp đã đầy

- Tiếp theo, ta nhập các thông tin của Sinh viên lớp đã được chọn.

A computer screen shot of a black screen

Description automatically generated

Nhập thông tin Sinh viên trong lớp học

- Chương trình in ra bảng yêu cầu xác nhận thông tin của Sinh viên đó, nếu thông tin bị sai thì nhấn phím 1, bảng sẽ đưa ra các thông tin và người nhập sẽ chọn các thông tin cần sửa và sửa lại.

A screenshot of a computer

Description automatically generated

Nhập lại thông tin do nhập bị sai xót

- Sau khi nhấn phím 2-xác nhận thông tin Sinh viên đã nhập, màn hình sẽ đưa ra thông báo yêu cầu người dùng có muốn nhập tiếp thông tin cho lớp đó, hay niên khóa và khoa khác hay không. Nếu có thì màn hình sẽ quay lại phần nhập thông tin Sinh viên, nếu không thì sẽ trở về Menu chính.

A screenshot of a computer

Description automatically generated

Nhập tiếp Sinh viên trong lớp đã chọn

A screenshot of a computer

Description automatically generated

Nhập tiếp Sinh viên với Khoa, Khóa khác

A screen shot of a computer

Description automatically generated

Ghi vào File Thêm Sinh viên

#### Chức năng Sắp xếp Sinh viên:

- Đầu tiên, màn hình sẽ đưa ra các lớp cần sắp xếp, người nhập cần chọn lớp mình muốn sắp xếp, sau đó sẽ in ra thông báo đã sắp xếp thành công. Chương trình sẽ tự động sắp xếp và cấp mã Sinh viên và Email cho từng Sinh viên.

A screenshot of a computer

Description automatically generated

Chọn lớp cần sắp xếp

- Sau khi đã sắp xếp lớp đã chọn, màn hình sẽ yêu cầu muốn sắp xếp cho lớp khác hay không, nếu có chương trình sẽ quay lại yêu cầu chọn lớp để sắp xếp, nếu không sẽ về màn hình chính.

A black screen with white lines

Description automatically generated with medium confidence

Thông báo sắp xếp lớp khác

- Nếu chưa có lớp nào được tạo, màn hình sẽ in ra thông báo như sau:

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa có lớp để sắp xếp

- Dữ liệu từ File trước và sau khi thực hiện chức năng Sắp xếp Sinh viên.

A screen shot of a computer

Description automatically generated

File trước khi Sắp xếp Sinh viên

A screen shot of a computer

Description automatically generated

File sau khi Sắp xếp Sinh viên

#### Chức năng Tìm kiếm Sinh viên

- Giao diện:

A screenshot of a computer

Description automatically generated

Giao diện chính của chức năng Tìm kiếm Sinh viên

- Nếu chưa có lớp nào được tạo, màn hình sẽ in ra thông báo như sau:

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa có lớp để tìm kiếm

- Khi này chương trình sẽ yêu cầu tìm bằng ID của Sinh viên, Thông tin Sinh viên, Số Điện thoại hoặc Số CCCD của Sinh viên.

* Tìm kiếm bằng ID Sinh viên:

- Màn hình sẽ in ra yêu cầu người nhập nhập ID Sinh viên cần tìm, nếu có thì chương trình sẽ yêu cầu nhập các thông tin cần in ra, nếu không sẽ báo không tìm thấy Sinh viên cần tìm.

A screenshot of a computer

Description automatically generated

Tìm kiếm bằng ID Sinh viên

A screenshot of a computer

Description automatically generated

Thông báo không tìm thấy Sinh viên bằng ID Sinh viên

* Tìm kiếm bằng Thông tin Sinh viên:

- Màn hình sẽ in ra yêu cầu người nhập nhập thông tin Sinh viên cần tìm, nếu có thì chương trình sẽ yêu cầu nhập các thông tin cần in ra, nếu không sẽ báo không tìm thấy Sinh viên cần tìm.

A screenshot of a computer

Description automatically generated

Tìm kiếm bằng Thông tin Sinh viên

- Nếu trong Khoa có các Sinh viên có tên trùng nhau, màn hình sẽ in ra những thông tin của các Sinh viên có cùng Họ và tên với điều kiện là lớp đó đã được sắp xếp, nếu không màn hình sẽ in ra thông báo “Lớp chưa được sắp xếp nên không thể tìm kiếm”.

A screenshot of a computer

Description automatically generated

Tìm kiếm bằng Thông tin Sinh viên với những thông tin giống nhau

A screenshot of a computer

Description automatically generated

Tìm kiếm bằng Thông tin Sinh viên với những thông tin giống nhau nhưng có lớp chưa được Sắp xếp.

A screenshot of a computer

Description automatically generated

Thông báo không tìm thấy Sinh viên bằng Thông tin Sinh viên

* Tìm kiếm bằng số Căn cước Công dân:

- Màn hình sẽ in ra yêu cầu người nhập nhập số Căn cước Công dân của Sinh viên cần tìm, nếu có thì chương trình sẽ yêu cầu nhập các thông tin cần in ra, nếu không sẽ báo không tìm thấy Sinh viên cần tìm.

A screenshot of a computer

Description automatically generated

Tìm kiếm bằng số Căn cước Công dân

A computer screen shot of a black screen

Description automatically generated

Thông báo không tìm thấy Sinh viên bằng số Căn cước Công dân

* Tìm kiếm bằng Số điện thoại:

- Màn hình sẽ in ra yêu cầu người nhập nhập Số điện thoại của Sinh viên cần tìm, nếu có thì chương trình sẽ yêu cầu nhập các thông tin cần in ra, nếu không sẽ báo không tìm thấy Sinh viên cần tìm.

A screenshot of a computer

Description automatically generated

Tìm kiếm bằng Số điện thoại

A screenshot of a computer

Description automatically generated

Thông báo không tìm thấy Sinh viên bằng Số điện thoại

- Khi kết thúc mỗi lần tìm kiếm bằng các cách tìm kiếm khác nhau, chương trình sẽ hỏi người dùng có muốn tiếp tục tìm kiếm bằng chức năng đó nữa hay không, nếu không thì sẽ quay lại giao diện tìm kiếm và tìm kiếm với chức năng khác hoặc về giao diện chính.

A screenshot of a computer

Description automatically generated

Tìm kiếm bằng chức năng đã chọn hoặc về Menu chính

#### Chức năng Xóa Sinh viên

- Giao diện:

A screenshot of a computer

Description automatically generated

Giao diện chính của chức năng Xóa Sinh viên

- Nếu chưa có lớp được tạo, màn hình sẽ in ra thông báo như sau:

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa có lớp để Xóa Sinh viên

- Khi này chương trình sẽ yêu cầu xóa bằng ID của Sinh viên, Số CCCD của Sinh viên.

* Xóa bằng ID Sinh viên:

- Chương trình sẽ yêu cầu người nhập nhập ID của Sinh viên cần xóa. Sau đó màn hình sẽ in ra thông tin Sinh viên với mã ID được tìm thấy. Nếu xóa thành công sẽ xuất hiện thông báo *“Đã xóa thành công”* nếu không sẽ in ra thông báo *“Không tìm thấy Sinh viên đó”.*

A screenshot of a computer

Description automatically generated

Xóa bằng ID Sinh viên thành công

A screenshot of a computer

Description automatically generated

Không tìm thấy thông tin để xóa bằng ID Sinh viên

* Xóa bằng số Căn cước Công dân của Sinh viên:

- Chương trình sẽ yêu cầu người nhập nhập số Căn cước Công dân của Sinh viên cần xóa. Sau đó màn hình sẽ in ra thông tin Sinh viên với mã ID được tìm thấy. Nếu xóa thành công sẽ xuất hiện thông báo *“Đã xóa thành công”* nếu không sẽ in ra thông báo *“Không tìm thấy Sinh viên đó”.*

A screenshot of a computer

Description automatically generated

Xóa bằng số Căn cước Công dân

A screenshot of a computer

Description automatically generated

Không tìm thấy thông tin để xóa băng Căn cước Công dân

- Dữ liệu từ File trước và sau khi thực hiện chức năng Sắp xếp Sinh viên.

A screen shot of a computer

Description automatically generated

File trước khi Xóa Sinh viên

A screen shot of a computer

Description automatically generated

File sau khi Xóa Sinh viên

#### Chức năng Xem thông tin Lớp học

- Màn hình sẽ hiển thị các Lớp hiện có trong Trường. Người dùng nhập số hiển thị cho Lớp đó, các thông tin cần in ra của Lớp đó. Sau đó màn hình sẽ hiển thị ra thông tin Lớp học theo yêu cầu.

A screenshot of a computer

Description automatically generated

Xem thông tin Lớp học

- Các Sinh viên được thêm vào sau và qua bước sắp xếp cấp Mã, Email thì sẽ được in ra theo thứ tự Alphabet và có Mã được cấp theo thuật toán.

- Nếu chưa có lớp được tạo, màn hình sẽ in ra thông báo như sau:

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa có lớp để in ra

#### Chức năng Cấp mã Sinh viên

- Nếu chưa có lớp được tạo, màn hình sẽ in ra thông báo như sau:

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa có lớp để cấp Mã ID

- Nếu trường hợp đã có Lớp học tồn tại nhưng chưa được sắp xếp, màn hình sẽ thông báo lỗi.

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa sắp xếp Lớp trong cấp Mã ID

- Nếu trường hợp cấp thành công, màn hình sẽ in ra thông báo sau:

A screenshot of a computer

Description automatically generated

Thông báo đã cấp Mã ID thành công

- Sau khi cấp Mã Sinh viên, màn hình sẽ yêu cầu người dùng có muốn cấp Mã Sinh viên cho lớp khác hay không hoặc trở về Menu chính.

A screenshot of a computer

Description automatically generated

Tiếp tục cấp Mã ID hoặc trở về Menu chính

#### Chức năng cấp Email Sinh viên

- Nếu chưa có lớp được tạo, màn hình sẽ in ra thông báo như sau:

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa có lớp để cấp Email

- Nếu trường hợp đã có Lớp học tồn tại nhưng chưa được sắp xếp, màn hình sẽ thông báo lỗi.

A screenshot of a computer

Description automatically generated

Thông báo lỗi chưa sắp xếp lớp trong cấp Email

- Nếu trường hợp cấp Email thành công, màn hình sẽ in ra thông báo sau:

A screenshot of a computer

Description automatically generated

Thông báo đã cấp Email thành công

- Sau khi cấp Email Sinh viên, màn hình sẽ yêu cầu người dùng có muốn cấp Email cho lớp khác hay không hoặc trở về Menu chính.

A screenshot of a computer

Description automatically generated

Tiếp tục cấp Email hoặc về Menu chính

### Nhận xét đánh giá

- Chương trình đã thể hiện khả năng thực hiện các chức năng cơ bản như: Thêm Sinh viên, Sắp xếp danh sách, Xóa Sinh viên, Tìm kiếm, Cấp Mã Sinh viên và Email và In danh sách theo yêu cầu.

- Mặc dù mỗi chương trình có những đặc điểm nổi bật (Tìm và Xóa bằng Căn cước Công dân,...), nhưng cần phải xem xét và khắc phục các thiếu sót để hoàn thiện hơn.

- Có thể phương pháp, thuật toán chưa được tối ưu. Trong tương lai, chúng em sẽ tiếp thu được nhiều kiến thức mới để quản lí, nâng cao hiệu suất và độ chính xác của chương trình.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết Luận Chung

### Ưu điểm

- **Mục tiêu đề ra và Tính Linh hoạt:** Chương trình đã thực hiện đầy đủ các mục tiêu do Đồ án PBL1 đề ra cũng như các chức năng cơ bản để Quản lí Sinh viên. Dựa vào sự linh hoạt của chương trình mà chúng ta cũng dễ dàng có thể mở rộng Quản lí các Học viên trung tâm, Nhân viên của công ty và nhiều hơn nữa bằng cách thay đổi cấu trúc Struct.

- **Tính dễ sử dụng:** Chương trình được thực hiện giao diện Console với các thao tác dễ sử dụng, không rườm rà. Từ đó giúp giảm thiểu thời gian đào tạo và tăng hiệu suất công việc, Người dùng có thể dễ dàng nhập, cập nhật và truy xuất thông tin Sinh viên một cách nhanh chóng và dễ dàng.

### Nhược điểm:

- **Không hỗ trợ trực tuyến:** Mô hình này được xây dựng dựa trên một hệ thống cục bộ, do đó nó không hỗ trợ truy cập và quản lý dữ liệu trực tuyến. Điều này có thể gây khó khăn cho việc truy cập và cập nhật dữ liệu từ xa.

- **Không đồng bộ dữ liệu:** Nếu có nhiều người dùng cùng sử dụng hệ thống, việc không đồng bộ dữ liệu có thể dẫn đến những thông tin không chính xác hoặc mất mát dữ liệu.

## Hướng phát triển

**- Hỗ trợ trực tuyến:** Phát triển một phiên bản trực tuyến của phần mềm để người dùng có thể truy cập và quản lý dữ liệu từ bất kỳ đâu thông qua Internet.

**- Đồng bộ dữ liệu:** Thêm chức năng đồng bộ dữ liệu giữa nhiều người dùng hoặc thiết bị, giúp cập nhật thông tin nhanh chóng và chính xác.

**- Tích hợp với các hệ thống khác:** Tích hợp với các hệ thống quản lý khác như hệ thống quản lý học phần, hệ thống quản lý thư viện, quản lý điểm số;...

**- Cải tiến bảo mật:** Tăng cường các biện pháp bảo mật để bảo vệ thông tin Sinh viên khỏi các mối đe dọa và rủi ro.

**- Tích hợp AI và Machine Learning:** Sử dụng AI và Machine Learning để phân tích dữ liệu và đưa ra các gợi ý hữu ích cho người dùng.

TÀI LIỆU THAM KHẢO

1. Brian W.Kernighan và Dennis M.Ritchie, *The C Programming Language* xuất bản lần thứ II.
2. Nguyễn Thị Lệ Quyên, *Bài giảng Kỹ thuật Lập trình bằng ngôn ngữ C*, Khoa Công nghệ Thông tin, Trường Đại học Bách Khoa, ĐHĐN.
3. Trần Hồ Thủy Tiên, *Bài giảng Strings*, Khoa Công nghệ Thông tin, Trường Đại học Bách Khoa, ĐHĐN.

PHỤ LỤC

(Chương trình *Quản lí Sinh viên* bằng Ngôn ngữ C)