**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ SINH VIÊN**

**Người hướng dẫn: GVC.ThS. Trần Hồ Thủy Tiên**

**Sinh viên thực hiện:**

**Đặng Đăng Khoa Lớp: 23T\_DT4 Nhóm: 23NH13B**

**Trần Nguyễn Văn Phát Lớp: 23T\_DT4 Nhóm: 23NH13B**

**Đà Nẵng, 5/2024**

**MỞ ĐẦU**

Trong xã hội hiện đại, sự phát triển nhanh chóng của công nghệ thông tin đã tạo ra nhu cầu cấp thiết về việc số hóa và tự động hóa các quy trình quản lý, đặc biệt trong lĩnh vực giáo dục. Việc quản lý sinh viên bằng phương pháp thủ công không chỉ tốn nhiều thời gian, công sức mà còn dễ dẫn đến sai sót và khó khăn trong việc lưu trữ, truy xuất thông tin. Phần mềm quản lý sinh viên sẽ giúp giải quyết các vấn đề này, đồng thời nâng cao hiệu quả quản lý và chất lượng giáo dục.

Việc xây dựng phần mềm quản lý sinh viên không chỉ giúp nhà trường và các cơ sở giáo dục theo dõi, quản lý thông tin của sinh viên một cách chính xác, nhanh chóng mà còn tạo ra một hệ thống làm việc hiệu quả và tiết kiệm thời gian. Phần mềm này cho phép thêm, xóa sinh viên, tìm kiếm thông tin và sắp xếp dữ liệu một cách dễ dàng, từ đó giúp cải thiện công tác quản lý và hỗ trợ quá trình học tập của sinh viên.

Áp dụng kiến kiến thức được tiếp thu trên trường chúng em đã xây dựng được một phần mềm với hệ thống dễ sử dụng, đáp ứng được các nhu cầu cơ bản. Phần mềm với những chức năng cơ bản đầy đủ như: thêm, xóa sinh viên, tìm kiếm và sắp xếp thông tin, cung cấp mã sinh viên và email đáp ứng các yêu cầu quản lý cơ bản. Dù hiện tại chỉ là phiên bản cơ bản, phần mềm có thể dễ dàng được nâng cấp và phát triển thêm các tính năng mới trong tương lai.

Trong quá trình thực hiện đồ án, do thời gian cũng như trình độ của em còn có những hạn chế nhất định nên không thể tránh khỏi những sai sót. Rất mong nhận được sự góp ý của các thầy, cô giáo và các bạn để đồ án hoàn thiện hơn. Em xin chân thành cảm ơn sự hướng dẫn, và giúp đỡ tận tình của cô Trần Hồ Thủy Tiên, các thầy cô trong khoa Công nghệ thông tin Trường Đại học Bách khoa - Đại học Đà Nẵng đã giúp đỡ em trong quá trình học tập cũng như trong quá trình làm đồ án.

**MỤC LỤC**

1. **TỔNG QUAN ĐỀ TÀI** 6
2. **CƠ SỞ LÝ THUYẾT** 7
   1. Ý TƯỞNG 7
   2. CƠ SỞ LÝ THUYẾT 7
      1. *Cấu trúc dữ liệu Array-based List* 7
      2. *Kiểu dữ liệu struct* 8
      3. *Các hàm xử lí chuỗi 9*
      4. *Các hàm thao tác với file* 10
3. **TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN** 11
   1. PHÁT BIỂU BÀI TOÁN 11
   2. CẤU TRÚC DỮ LIỆU 12
   3. THUẬT TOÁN 14
      1. *Thuật toán sắp xếp sinh viên* 14
      2. *Thuật toán thêm sinh viên* 15
      3. *Thuật toán cấp mã sinh viên* 16
      4. *Thuật toán cấp email sinh viên* 18
      5. *Thuật toán tìm kiếm sinh viên 20*
      6. *Thuật toán xóa sinh viên* 22
4. **CHƯƠNG TRÌNH VÀ KẾT QUẢ** 23
   1. TỔ CHỨC CHƯƠNG TRÌNH 23
   2. NGÔN NGỮ CÀI ĐẶT 24
   3. KẾT QUẢ 27
      1. *Giao diện chính của chương trình* 27
      2. *Kết quả thực thi của chương trình* 28
      3. *Nhận xét đánh giá* 39
5. **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 40
   1. KẾT LUẬN 40
   2. HƯỚNG PHÁT TRIỂN 40

**TÀI LIỆU THAM KHẢO** 41

**DANH MỤC HÌNH VẼ**

*Hình 2.1*: Minh họa cấu trúc dữ liệu Array-based List 7

*Hình 2.2*: Cách khai báo kiểu struct 8

*Hình 3.1*: Cấu trúc dữ liệu của chương trình. 12

*Hình 3.2*: Sơ đồ mô tả thuật toán QuickSort 14

*Hình 3.3*: Sơ đồ thuật toán thêm sinh viên 15

*Hình 3.4*: Sơ đồ thuật toán cấp mã sinh viên 16

*Hình 3.5*: Quy trình cấp mã sinh viên 17

*Hình 3.6*: Sơ đồ thuật toán cấp email sinh viên 18

*Hình 3.7*: Quy trình cấp email 19

*Hình 3.8*: Sơ đồ thuật toán tìm kiếm sinh viên 20

*Hình 3.9*: Quy trình tìm kiếm sinh viên 21

*Hình 3.10*: Sơ đồ thuật toán xóa sinh viên 22

*Hình 4.1*: Tổ chức chương trình 23

*Hình 4.2*: Giao diện chính 27

*Hình 4.3*: Giao diện menu lựa chọn đăng nhập/đăng kí 28

*Hình 4.4*: Giao diện đăng kí 29

*Hình 4.5*: Giao diện đăng nhập 29

*Hình 4.6*: Giao diện chức năng thêm sinh viên 32

*Hình 4.7*: Giao diện chức năng sắp xếp sinh viên 34

*Hình 4.8*: Giao diện chức năng cấp mã sinh viên 35

*Hình 4.9*: Giao diện chức năng email 36

*Hình 4.10*: Giao diện chức năng tìm kiếm bằng mã sinh viên 37

*Hình 4.11*: Giao diện chức năng tìm kiếm bằng tên 37

*Hình 4.12*: Giao diện chức năng xóa sinh viên 38

*Hình 4.13*: Giao diện chức năng in danh sách 39

# **TỔNG QUAN ĐỀ TÀI**

Đề tài “Xây dựng ứng dụng quản lý danh sách sinh viên” là một đề tài trong lĩnh vực công nghệ thông tin và phát triển phần mềm. Đề tài này tập trung vào xây dựng một ứng dụng quản lý danh sách sinh viên với các chức năng cơ bản như: Thêm sinh viên, sắp xếp danh sách của một lớp, xóa sinh viên, tìm kiếm sinh viên, cấp mã sinh viên, cấp email, in ra danh sách.

* 1. **Mục đích thực hiện đề tài**

Tạo ra một phần mềm có khả năng quản lý thông tin sinh viên, giúp cho việc quản lý và tra cứu thông tin về sinh viên trở nên dễ dàng hơn, tiết kiệm thời gian và công sức cho những người quản lý.

* 1. **Mục tiêu của đề tài**

Tạo ra một phần mềm quản lý thông tin sinh viên hiệu quả và tiện lợi hơn.

* *Xây dựng giao diện đơn giản.*
* *Cho phép thêm, xóa thông tin sinh viên một cách dễ dàng.*
* *Cung cấp tính năng tìm kiếm sinh viên theo nhiều tiêu chí khác nhau.*
* *Cung cấp tính năng sắp xếp danh sách sinh viên.*
* *Cấp mã sinh viên và email cho sinh viên mới và đảm bảo tính duy nhất của mã sinh viên và email.*
* *Cho phép in ra danh sách sinh viên để dễ dàng sử dụng các thông tin sinh viên*.
  1. **Phạm vi đề tài**

Tập trung vào việc phát triển một phần mềm quản lý thông tin sinh viên của trường Đại học Bách khoa Đà Nẵng.

* 1. **Đối tượng nghiên cứu**

Là những người quản lý thông tin sinh viên ở trong trường học.

* 1. **Phương pháp nghiên cứu**

Tổng hợp kiến thức đã học và sử dụng kiến thức ấy để tạo ra sản phẩm.

* 1. **Yêu cầu**
* *Ngôn ngữ lập trình*: Sử dụng ngôn ngữ lập trình C.
* *Cơ sở dữ liệu*: Để lưu trữ danh sách sinh viên ta có thể sử dụng một file text để lưu trữ thông tin danh sách sinh viên và khi cần sử dụng thì ta cần phải đọc file và xử lý nó bằng các thuật toán phù hợp.
* *Giao diện người dùng*: Ở dự án này ta sử dụng màn hình console để hiển thị giao diện mà ta thao tác sử dụng.
* *Thuật toán sắp xếp*: Để sắp xếp danh sách sinh viên, chúng tôi đã sử dụng thuật toán Quick Sort để sắp xếp.
* *Tạo mã sinh viên và email*: Tạo ra một mã sinh viên cho mỗi sinh viên dựa trên danh sách lớp đã được sắp xếp, cấp email ứng dụng sẽ tự động tạo và cấp phát một địa chỉ email cho sinh viên.

# **CƠ SỞ LÝ THUYẾT**

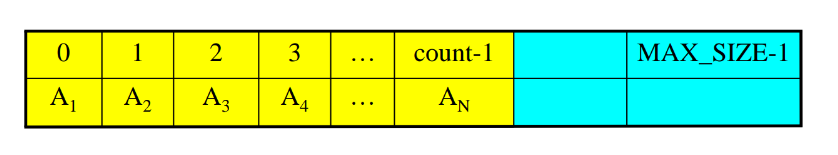
## **Ý tưởng**

* Tạo cấu trúc dữ liệu để lưu trữ thông tin sinh viên, bao gồm các trường như: Họ và tên lót, Ngày sinh, Giới tính, Địa chỉ, Mã sinh viên, Email.
* Tạo một danh sách để lưu trữ danh sách sinh viên. Mỗi phần tử trong danh sách sẽ chứa thông tin của một sinh viên.
* Xây dựng một vòng lặp vô hạn để hiển thị menu và xử lý lựa chọn của người dùng cho đến khi người dùng chọn thoát.
* Hiển thị menu cho người dùng và yêu cầu người dùng nhập lựa chọn của mình.

## **Cơ sở lý thuyết**

## **Cấu trúc dữ liệu Array-based List**

* Array-based List là một cấu trúc dữ liệu danh sách được triển khai dựa trên mảng (array). Đây là một kiểu dữ liệu trừu tượng (Abstract Data Type - ADT) cung cấp các thao tác cơ bản như chèn, xóa, truy cập và tìm kiếm phần tử. Array-based List lưu trữ các phần tử trong một mảng liên tục, cho phép truy cập trực tiếp và nhanh chóng đến các phần tử thông qua chỉ số của chúng.

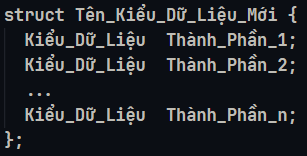


*Hình 2.1. Minh họa cấu trúc dữ liệu Array-based List*

* Ý tưởng cơ bản:
* Cấp phát trước một mảng lớn có kích thước **MAX\_SIZE**.
* Theo dõi kích thước hiện tại bằng cách sử dụng biến **count**.
* Di chuyển các phần tử khi chèn hoặc xóa*.*
* Ưu điểm:
* *Truy cập nhanh*: Truy cập phần tử theo chỉ số trong thời gian O(1).
* *Bộ nhớ liên tụ*c: Sử dụng bộ nhớ liên tục giúp tận dụng hiệu quả bộ nhớ cache của CPU.
* *Dễ triển khai*: Cấu trúc đơn giản và dễ hiểu.
* Nhược điểm:
* *Chèn/xóa phần tử tốn kém*: Các thao tác chèn hoặc xóa ở giữa danh sách có thể tốn O(n) thời gian do phải dịch chuyển các phần tử.
* *Mở rộng mảng tốn kém*: Khi mảng đầy, việc mở rộng mảng cần sao chép toàn bộ phần tử sang mảng mới, tốn O(n) thời gian và không gian tạm thời.
* *Không tối ưu về bộ nhớ*: Nếu mảng được khởi tạo với kích thước lớn hơn cần thiết hoặc chưa sử dụng hết, sẽ lãng phí bộ nhớ.

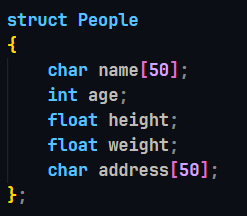
## **Kiểu dữ liệu struct**

* Định nghĩa: Struct là một kiểu dữ liệu tùy chỉnh, được sử dụng để lưu trữ các biến có kiểu dữ liệu khác nhau ở một đối tượng duy nhất. Một struct được định nghĩa bằng cách liệt kê tất cả các biến thành viên bên trong nó, và mỗi biến thành viên có thể có một kiểu dữ liệu khác nhau.
* Cú pháp khai báo:



*Hình 2.2. Cách khai báo kiểu struct*

* Ví dụ minh họa:



## **Các hàm xử lý chuỗi**

1. **Hàm strcmp:**

* Công dụng: So sánh 2 chuỗi s1 và s2.
* Cấu trúc:



Hàm sẽ trả 1 trong các giá trị sau:

* Giá trị âm nếu chuỗi s1 nhỏ hơn chuỗi s2.
* Giá trị 0 nếu hai chuỗi bằng nhau.
* Giá trị dương nếu chuỗi s1 lớn hơn chuỗi s2.

1. **Hàm strcpy**

* Công dụng: Sao chép chuỗi s2 vào chuỗi s1.
* Cấu trúc:



1. **Hàm strstr**

* Công dụng: Trả về địa chỉ vị trí xuất hiện đầu tiên của chuỗi s1 trong chuỗi s và sẽ trả về giá trị NULL trong trường hợp không tìm thấy.
* Cấu trúc:



1. **Hàm strchr**

* Công dụng: Trả về địa chỉ vị trí xuất hiện đầu tiên của kí tự ch trong chuỗi s và sẽ trả về giá trị NULL trong trường hợp không tìm thấy.
* Cấu trúc:



1. **Hàm sprintf**

* Công dụng: Định dạng chuỗi với các giá trị khác nhau, tương tự như cách “printf” định dạng chuỗi và in ra màn hình, nhưng “sprintf” lưu chuỗi đã định dạng vào một biến thay vì in ra.
* Cấu trúc:



‘str’: Đây là con trỏ đến mảng ký tự nơi chuỗi kết quả sẽ được lưu trữ.

‘format’: Đây là chuỗi định dạng, tương tự như trong “printf”, bao gồm các định dạng phần trăm (như %d, %s, %f, v.v.) để chèn các giá trị.

‘...’: Danh sách các đối số tương ứng với các định dạng trong format.

1. **Hàm strlen**

* Công dụng: cho biết độ dài của chuỗi s.
* Cấu trúc:



## **Các hàm xử lý file**

1. **Hàm rewind**

* Công dụng: Hàm này sẽ di chuyển con trỏ về đầu file.
* Cấu trúc:



1. **Hàm fgets**

* Công dụng: Hàm này dùng để đọc chuỗi n ký tự hoặc gặp ký tự kết thúc chuỗi. Hàm trả về con trỏ s nếu thành công, ngược lại trả con trỏ NULL.
* Cấu trúc:



1. **Hàm fprintf**

* Công dụng: Hàm này dùng để ghi nội dung vào file.
* Cấu trúc:



1. **Hàm fopen**

* Công dụng: Hàm này sử dụng để mở một tệp tin và trả về con trỏ đến cấu trúc FILE để thực hiện các thao tác đọc, ghi hoặc cả hai trên tệp tin đó.
* Cấu trúc:



‘mode’: Chuỗi ký tự chỉ định chế độ mở file. Các chế độ phổ biến bao gồm:

* “r”: Mở file chỉ để đọc. File phải tồn tại.
* “w”: Mở file chỉ để ghi. Nếu file đã tồn tại, nội dung sẽ bị xóa. Nếu file không tồn tại, nó sẽ được tạo mới.
* “a”: Mở file để ghi thêm. Dữ liệu sẽ được ghi vào cuối file nếu file đã tồn tại. Nếu file không tồn tại, nó sẽ được tạo mới.
* “r+”: Mở file để đọc và ghi. file phải tồn tại.
* “w+”: Mở file để đọc và ghi. Nếu file đã tồn tại, nội dung sẽ bị xóa. Nếu file không tồn tại, nó sẽ được tạo mới.
* “a+”: Mở file để đọc và ghi thêm. Dữ liệu sẽ được ghi vào cuối file nếu file đã tồn tại. Nếu file không tồn tại, nó sẽ được tạo mới.

1. **Hàm fclose**

* Công dụng: Hàm này sử dụng để đóng một tệp tin đã được mở bằng hàm “fopen”.
* Cấu trúc:



Giá trị trả về:

* Trả về 0 nếu đóng thành công.
* Trả về EOF (End Of File, thường là -1) nếu xảy ra lỗi.

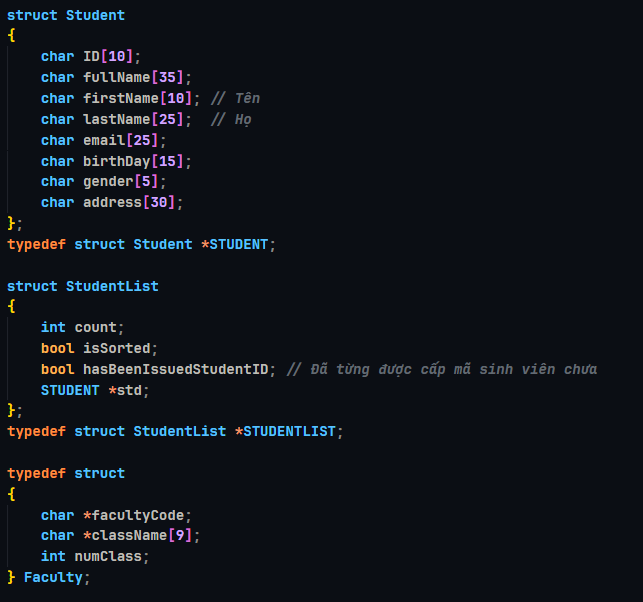
# **TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN**

## **Phát biểu bài toán**

* Đầu vào (Input) của chương trình bao gồm:
* Thông tin về sinh viên mới bao gồm Họ lót, Tên, Ngày tháng năm sinh, Giới tính và Địa chỉ.
* Yêu cầu sắp xếp danh sách.
* Yêu cầu xóa sinh viên dựa trên Mã sinh viên.
* Yêu cầu tìm kiếm sinh viên dựa trên Mã sinh viên hoặc Tên.
* Đầu ra (Output) của chương trình bao gồm:
* File danh sách sinh viên với thông tin về Mã sinh viên, Họ lót, Tên, Ngày sinh, Giới tính, Địa chỉ và Email.

## **Cấu trúc dữ liệu**

Sử dụng cấu trúc dữ liệu Array-based List



*Hình 3.1. Cấu trúc dữ liệu chương trình*

1. struct Student: là một cấu trúc dữ liệu dùng để lưu trữ thông tin của một sinh viên. Mỗi sinh viên có các thuộc tính sau:

* ‘ID’: Mã sinh viên.
* ‘fullName’: Họ và tên đầy đủ của sinh viên.
* ‘firstName’: Tên riêng của sinh viên.
* ‘lastName’: Họ của sinh viên.
* ‘email’: Địa chỉ email của sinh viên.
* ‘birthDay’: Ngày sinh của sinh viên, có định dạng “dd/mm/yy”.
* ‘gender’: Giới tính của sinh viên.
* ‘address’: Địa chỉ của sinh viên.

1. struct StudentList: là một cấu trúc dữ liệu dùng để quản lý danh sách các sinh viên. Nó bao gồm các thuộc tính sau:

* ‘count’: Số lượng sinh viên hiện có trong danh sách.
* ‘isSorted’: Biến boolean cho biết danh sách đã được sắp xếp hay chưa.
* ‘hasBeenIssuedStudentID’: Biến boolean cho biết các lớp đó đã từng cấp mã sinh viên hay chưa.
* ‘std’: Con trỏ tới mảng các con trỏ tới struct Student. Mảng này chứa các sinh viên trong danh sách.

1. struct Faculty là một cấu trúc dữ liệu dùng để lưu trữ thông tin về khoa và các lớp học liên quan. Nó bao gồm các thuộc tính sau:

* ‘facultyCode’: Mã khoa, là một con trỏ đến chuỗi ký tự đại diện cho mã của khoa.
* ‘className’: Mảng các con trỏ đến chuỗi ký tự, mỗi chuỗi đại diện cho tên của một lớp. Mảng này có thể chứa tối đa 9 lớp.
* ‘numClass’: Số lượng lớp hiện có trong khoa.

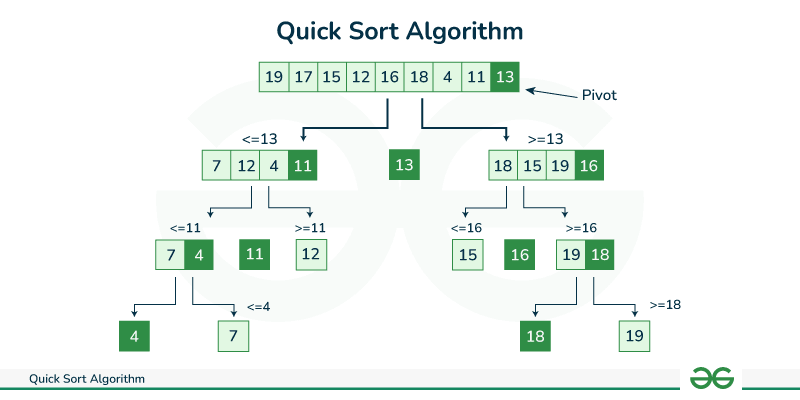
## **Thuật toán**

## **Thuật toán sắp xếp sinh viên**

* Sử dụng thuật toán sắp xếp nhanh (Quick Sort) để sắp xếp danh sách sinh viên theo tên và họ lót. Thuật toán sắp xếp nhanh là thuật toán phổ biến và hiệu quả với số lượng sinh viên lớn (100 sinh viên).
* Ý tưởng:  Thuật toán sắp xếp Quick Sort dựa trên kỹ thuật chia để trị. Thuật toán chia mảng thành các mảng con dựa vào phần tử chốt đã chọn. Dãy con trái gồm những phần tử nhỏ hơn hoặc bằng phần tử chốt và dãy con phải gồm những phần tử lớn hơn phần tử chốt. Sau đó, thuật toán lặp lại một cách đệ quy với hai dãy con trái và phải.
* Cách thực hiện:

**Bước 1**: Chọn phần tử chốt (ở đây chúng ta chọn phần tử cuối cùng).

**Bước 2**: Đưa những phần tử nhỏ hơn hoặc bằng phần tử chốt vào mảng bên trái và những phần tử lớn hơn phần tử chốt vào mảng bên phải.

**Bước 3**: Thực hiện lại hai bước trên với mảng bên trái và mảng bên phải.

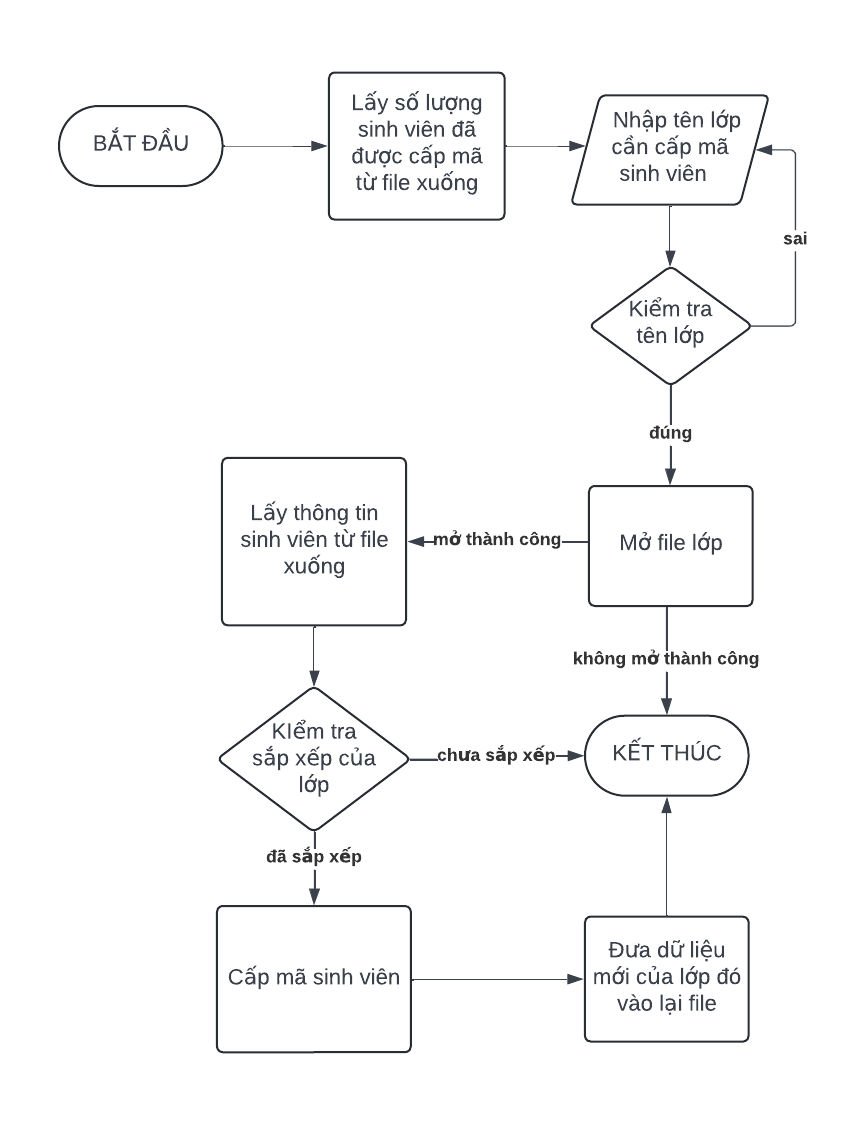
* Độ phức tạp của thuật toán:
* Trường hợp xấu nhất: O(n2).
* Trường hợp trung bình và tốt nhất: O(nlogn).

*Hình 3.2. Sơ đồ mô tả thuật toán Quick Sort*

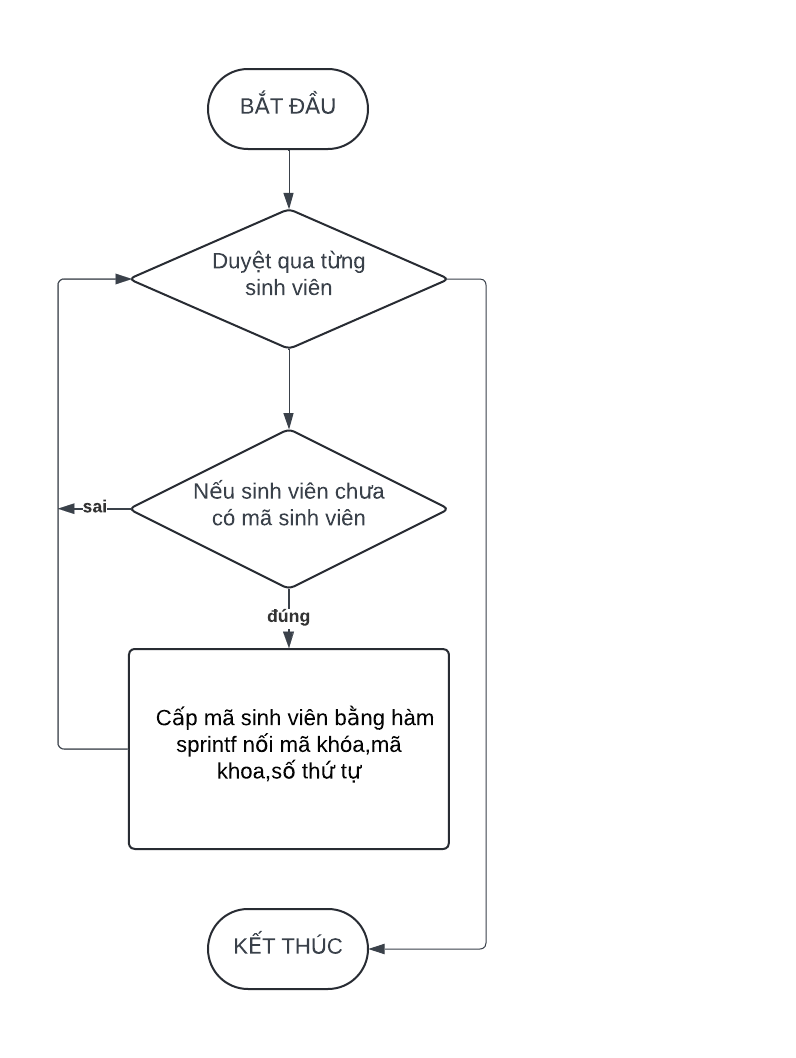
## **Thuật toán thêm sinh viên**

*Hình 3.3. Sơ đồ thuật toán thêm sinh viên*

## **Thuật toán cấp mã sinh viên**



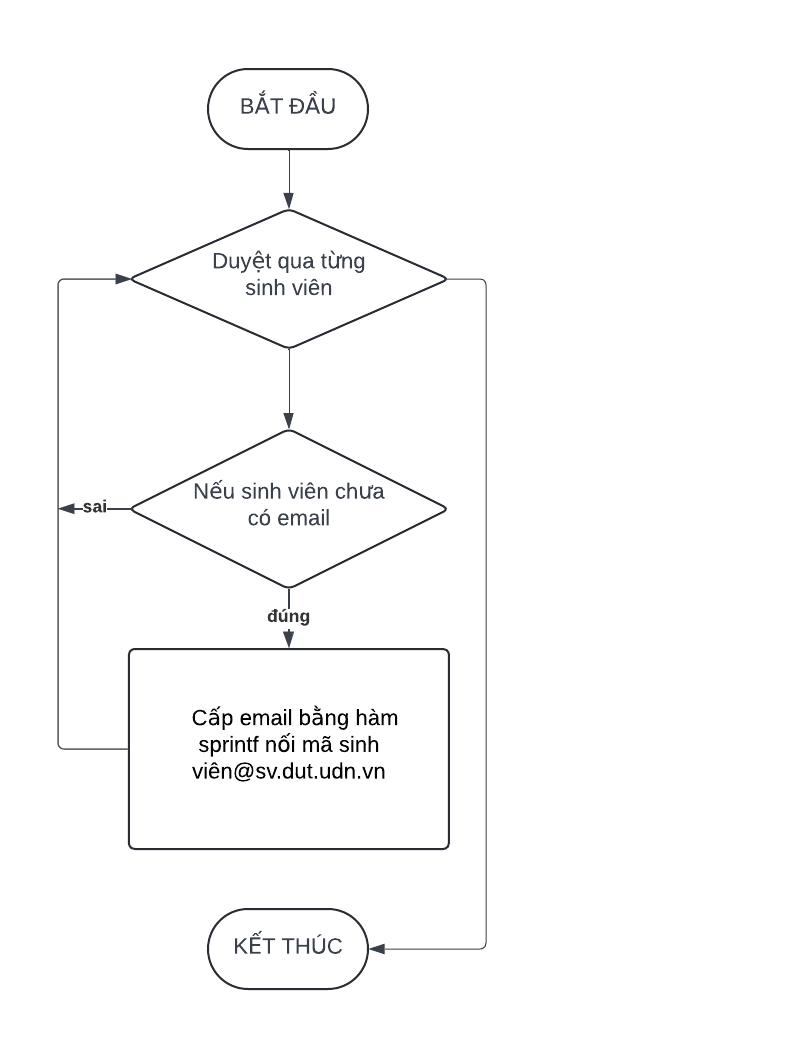
*Hình 3.4. Sơ đồ thuật toán cấp mã sinh viên*



*Hình 3.5. Quy trình cấp mã sinh viên*

## **Thuật toán cấp email sinh viên**

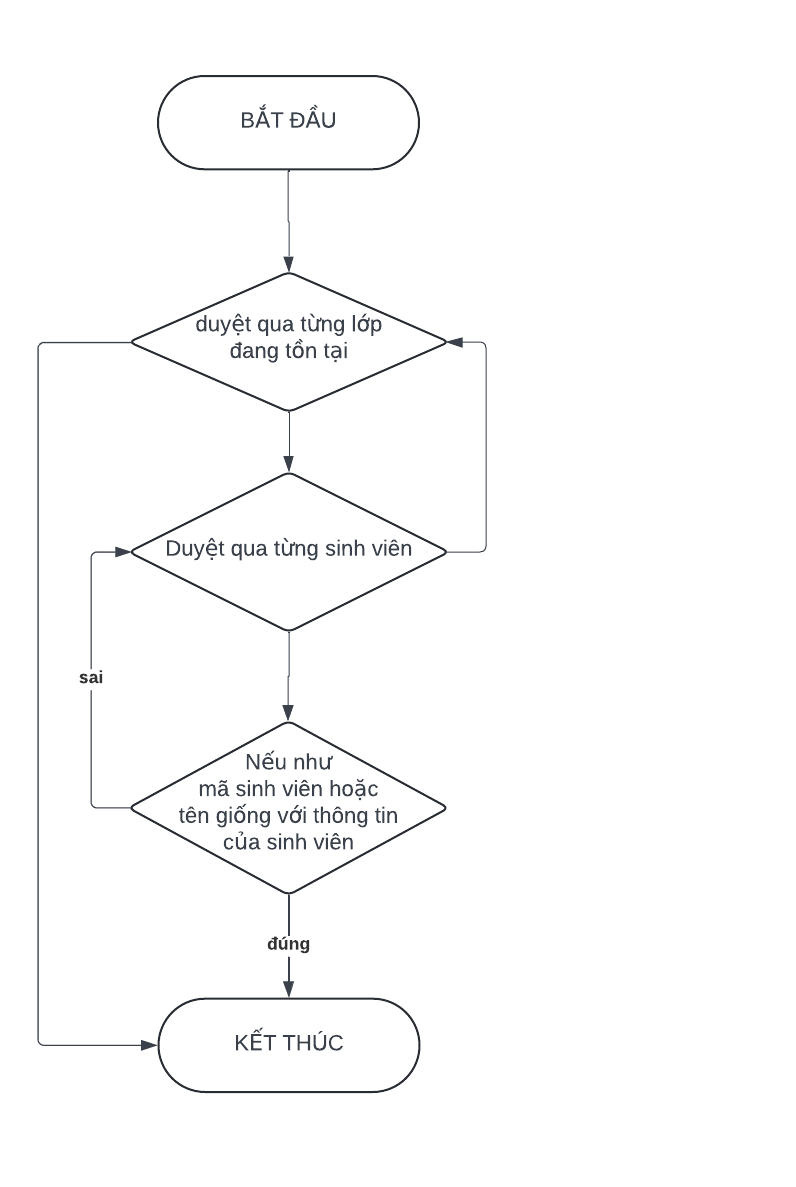
*Hình 3.6. Sơ đồ thuật toán cấp email sinh viên*



*Hình 3.7. Quy trình cấp email*

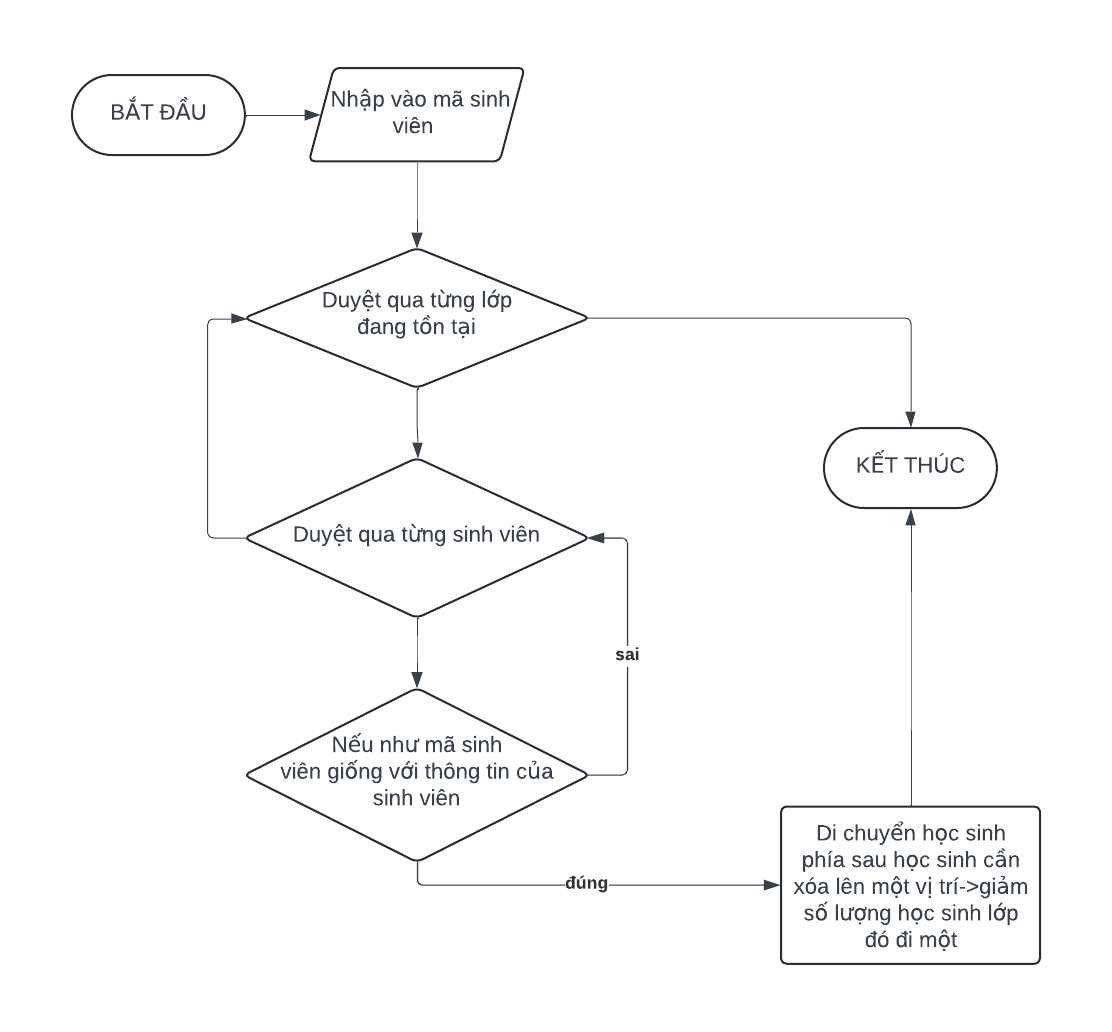
## A diagram of a flowchart Description automatically generated**Thuật toán tìm kiếm sinh viên**

*Hình 3.8. Sơ đồ thuật toán tìm kiếm sinh viên*



*Hình 3.9. Quy trình tìm kiếm sinh viên*

## **Thuật toán xóa sinh viên**



*Hình 3.10. Sơ đồ thuật toán xóa sinh viên*

# **CHƯƠNG TRÌNH VÀ KẾT QUẢ**

## **Tổ chức chương trình**

*Hình 4.1. Tổ chức chương trình*

## 

## **Xây dựng chương trình**

Từ ngững thiết kế bên trên, chương trình định nghĩa các hằng, cấu trúc dữ iệu, nguyên mẫu các hàm như sau:

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <windows.h>  #include <conio.h>  #include <stdbool.h>  #include <ctype.h>  #define MAX\_USERNAME\_LENGTH 30  #define MIN\_USERNAME\_LENGTH 5  #define MAX\_PASSWORD\_LENGTH 30  #define MIN\_PASSWORD\_LENGTH 8  #define MAX\_SIZE 100  struct Student  {  char ID[10];  char fullName[35];  char firstName[10]; // Tên  char lastName[25]; // Họ  char email[25];  char birthDay[15];  char gender[5];  char address[30];  };  typedef struct Student \*STUDENT;  struct StudentList  {  int count;  bool isSorted;  bool hasBeenIssuedStudentID; // Đã từng được cấp mã sinh viên chưa  STUDENT \*std;  };  typedef struct StudentList \*STUDENTLIST;  typedef struct  {  char \*facultyCode;  char \*className[9];  int numClass;  } Faculty;  void removeEnter(char \*string): dùng để xóa dấu “\n” khi nhập vào chuỗi bằng hàm fgets().  char \*toLower(char \*string): chuyển thành chuỗi in thường.  char \*toUpper(char \*string): chuyển thành chuỗi in hoa.  char \*ltrim(char \*string): xóa dấu cách dư thừa bên trái của chuỗi.  char \*rtrim(char \*string): xóa dấu cách dư thừa bên phải của chuỗi.  char \*trim(char \*string): xóa khoảng trống dư thừa ở 2 đầu và ở giữa.  char \*removeSpecialChar(char \*string): xóa kí tự đặc biệt trong chuỗi.  char \*toName(char \*string): chuyển chuỗi thành tên.  bool isNumer(char \*string): kiểm tra trong chuỗi có phải tất cả đều là số.  bool isString(char \*string): kiểm tra trong chuỗi có phải tất cả đều là chữ.  STUDENTLIST createClass(): tạo lớp.  bool checkFacultyCode(): kiểm tra mã khoa.  bool checkAcademicYear(): kiểm tra mã khóa.  void printListClassName(): in danh sách các lớp.  void printFacultyName(): in tên khoa.  void inputCode(): nhập mã khóa và mã khoa.  bool checkClassName(char \*className): kiểm tra tên lớp.  void insertStudent(STUDENTLIST Class, STUDENT std): thêm sinh viên vào lớp.  void printClassToFile(STUDENTLIST Class, FILE \*f): in danh sách sinh viên được định dạng ra file.  void printNewStudentToFile(FILE \*f, STUDENT newStudent): thêm sinh viên mới vào file.  void getStudentInfo(STUDENT student): nhập thông tin sinh viên.  bool isSorted(STUDENTLIST Class): kiểm tra lớp đã được sắp xếp chưa.  void sortClass(): sắp xếp các lớp từ file.  void readStudentInfoFromFile(FILE \*f, STUDENTLIST Class): lấy thông tin sinh viên từ file.  bool confirmExit(): xác nhận thoát khỏi hàm.  void addStudent(): chức năng thêm sinh viên.  void swapStudent(STUDENT \*student1, STUDENT \*student2): đổi chỗ 2 sinh viên.  void quickSort(STUDENT Class, int start, int end): sắp xếp lớp.  void sortStudent(): chức năng sắp xếp sinh viên.  void printListStudent(): chức năng in danh sách sinh viên.  void generateID(): chức năng cấp mã sinh viên.  void generateEmail(): chức năng cấp email.  void removeStudent(): chức năng xóa sinh viên.  void find(char \*search, char \*status): tìm kiếm sinh viên.  void searchStudent(): chức năng tìm kiếm sinh viên.  void menu(): in ra menu lựa chọn ở chương trình chính.  void loading(): tạo hiệu ứng loading.  void encode(char \*string): mã hóa thành base64.  void decode(char \*string): giải mã.  void inputPassword(char \*password): nhập password.  bool checkPassword(char \*password, char \*userInputPass): kiểm tra password nhập vào còn đúng hay không.  bool checkUsername(char \*userame, char \*usernameInput): kiểm tra username có trùng/đúng hay không.  bool readUsernameAndPasswordFromFile(FILE \*f, char \*username, char \*password): lấy username và password từ file.  bool login(): đăng nhập.  void printBanner(): in banner.  void runProgram(): chạy chương trình chính.  void menuLoginAndRegister(): in menu lựa chọn đăng nhập và đăng kí.  void loginAndRegister(): lựa chọn đăng nhập hoặc đăng kí |

* 1. **Kết quả**
     1. **Giao diện chính của chương trình**

*Hình 4.2. Giao diện chính*

* + 1. **Kết quả thực thi chương trình**

1. **Chương trình đăng nhập, đăng kí**

Chương trình sẽ hiển thi ra ba lựa chọn:

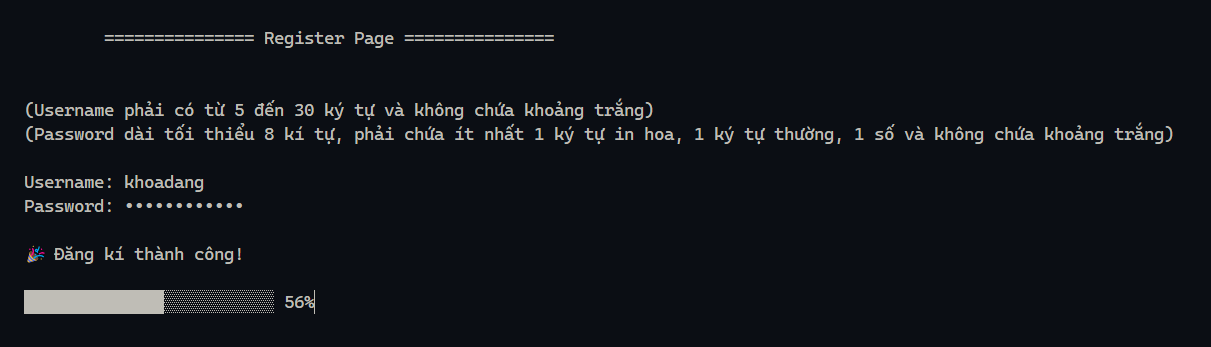
1. Đăng kí
2. Đăng nhập
3. Thoát



*Hình 4.3. Giao diện menu lựa chọn đăng nhập/đăng kí*

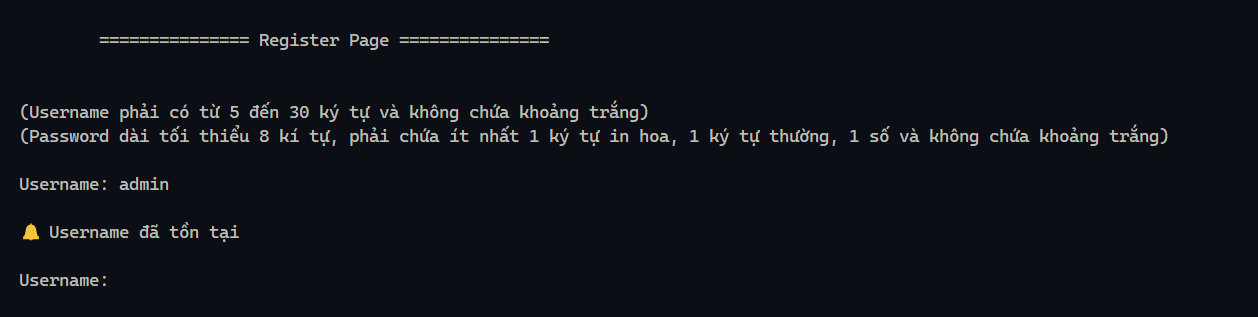
* Với lựa chọn 1 thì chương trình sẽ đưa đến trang đăng kí.

Sau khi đăng kí thành công thì chương trình sẽ quay trở lại trang trước để người dùng chọn đăng nhập hoặc thoát chương trình.



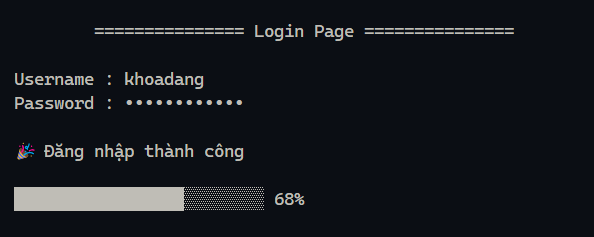
*Hình 4.4. Giao diện đăng kí*

Nếu như nhập vào tên đăng nhập mà đã có tài khoản có tên đăng nhập này thì sẽ thông báo là tài khoản đã tồn tại và yêu cầu người dùng nhập lại tên đăng nhập.



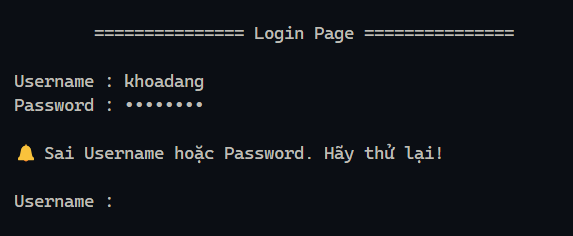
* Với lựa chọn 2 thì chương trình sẽ đưa đến trang đăng nhập.

Nếu như nhập đúng tài khoản và mật khẩu thì chương trình sẽ thông báo đăng nhập thành công và sử dụng chương trình.

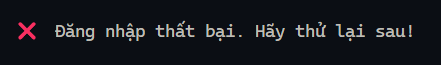


*Hình 4.5. Giao diện đăng nhập*

Nếu như nhập sai tài khoản và mật khẩu thì chương trình sẽ thông báo sai và tiến hành đăng nhập lại.



Nếu như nhập sai quá 5 lần thì sẽ thoát khỏi chương trình.

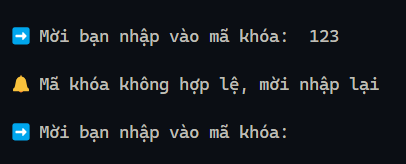


1. **Chức năng nhập mã khoa, nhập mã khóa**

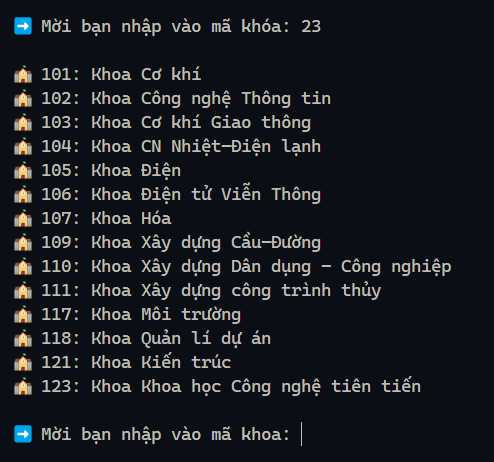
Sau khi đăng nhập thành công thì chương trình sẽ sử dụng chương trình

Chương trình sẽ yêu cầu người dùng nhập vào mã khóa.

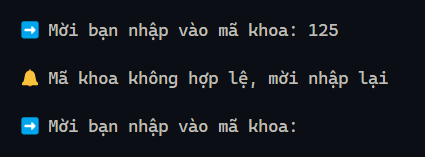
Nếu như nhập sai mã khóa thì chương trình sẽ thông báo và yêu cầu nhập lại.



Sau khi nhập xong mã khóa thành công thì sẽ đến mã khoa.

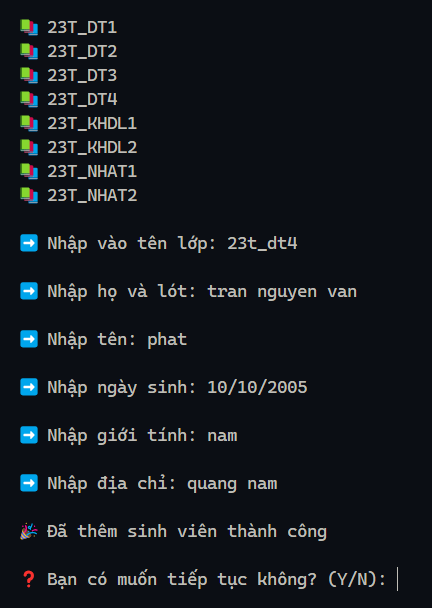


Nếu như nhập sai mã khoa thì chương trình sẽ thông báo và yêu cầu nhập lại.



1. **Chức năng thêm sinh viên**

Với chức năng thêm sinh viên, ban đầu sẽ tiến hành nhập vào tên lớp cần thêm sinh viên (với những lớp của khoa mà người dùng đã nhập ở mã khoa trước đó), sau đó tiến hành nhập các thông tin Họ và tên lót, Tên, Ngày sinh, Giới tính, Địa chỉ.



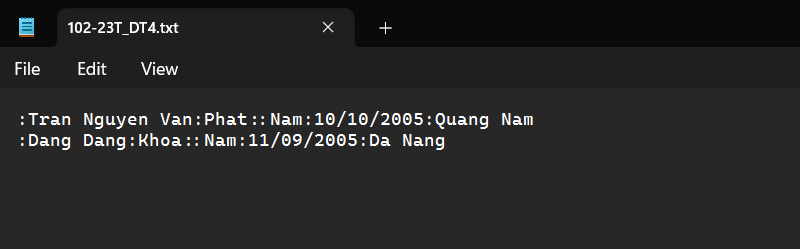
*Hình 4.6. Giao diện chức năng thêm sinh viên*

Sau khi nhập xong thông tin của sinh viên đầu tiên, chương trình sẽ hỏi rằng “Bạn có muốn tiếp tục không? (Y/N):”.

Nếu như nhập Y (hoặc y) thì sẽ tiếp tục nhập thông tin sinh viên tiếp theo.



Nếu như nhập N (hoặc n) thì sẽ trở lại menu.

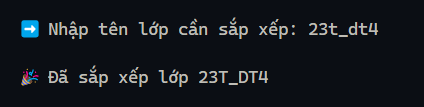


Sau khi thêm sinh viên thành công thì các sinh viên đó sẽ được lưu trên file có dạng

<mã khóa>-<tên lớp>.txt. Mỗi sinh viên sẽ được in trên từng dòng và mỗi thông tin của sinh viên sẽ cách nhau bởi dấu ‘:’. Các thông tin lần lượt là Mã sinh viên, Họ và tên lót, Tên, Email, Giới tính, Ngày sinh, Địa chỉ. Mã sinh viên và Email nếu chưa có thì bỏ trống.

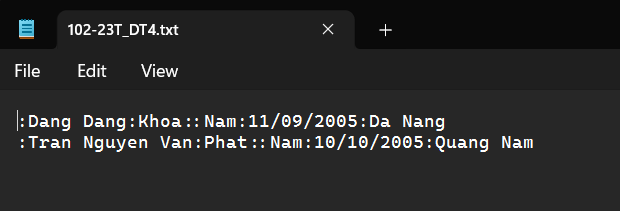
1. **Chức năng sắp xếp danh sách**

Với chức năng sắp xếp danh sách, chương trình sẽ yêu cầu nhập vào tên lớp cần sắp xếp danh sách sinh viên. Lúc này sẽ tiến hành mở file lớp đọc thông tin. Nếu như lớp chưa được sắp xếp sẽ tiến hành sắp xếp.

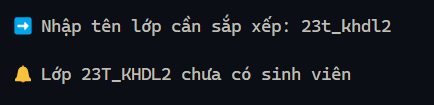


*Hình 4.7. Giao diện chức năng sắp xếp sinh viên*

Danh sách lớp sau khi sắp xếp:



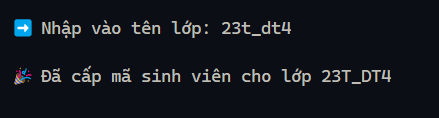
Nếu như lớp không có sinh viên nào sẽ báo là lớp chưa có sinh viên.



1. **Chức năng cấp mã sinh viên**

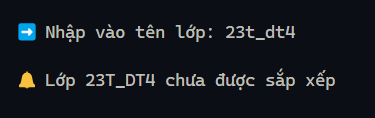
Với chức năng này chương trình sẽ yêu cầu nhập vào tên lớp cần cấp mã sinh viên.

Nếu lớp đã được sắp xếp thì sẽ tiến hành cấp mã sinh viên cho từng sinh viên trong lớp và in ra thông báo cấp mã thành công.

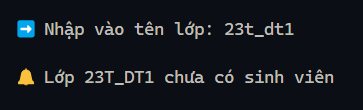


*Hình 4.8. Giao diện chức năng cấp mã sinh viên*

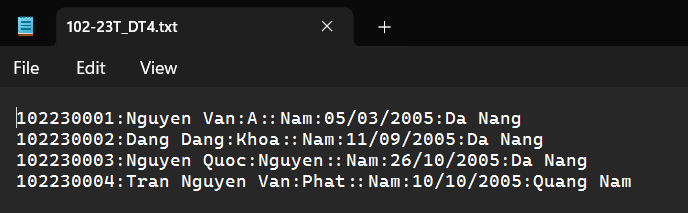
Nếu lớp chưa được sắp xếp thì sẽ in ra thông báo là lớp chưa được sắp xếp.



Nếu lớp chưa có sinh viên nào thì sẽ in ra thông báo là lớp chưa có sinh viên.



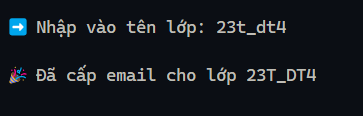
Danh sách lớp sau khi cấp mã:



1. **Chức năng cấp email**

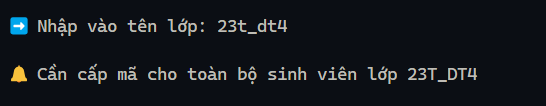
Chương trình sẽ yêu cầu nhập vào lớp cần cấp email.

Nếu như tất cả sinh viên trong lớp đều có mã sinh viên thì tiến hành cấp email cho từng sinh viên.

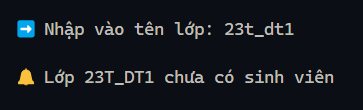


*Hình 4.9. Giao diện chức năng cấp email*

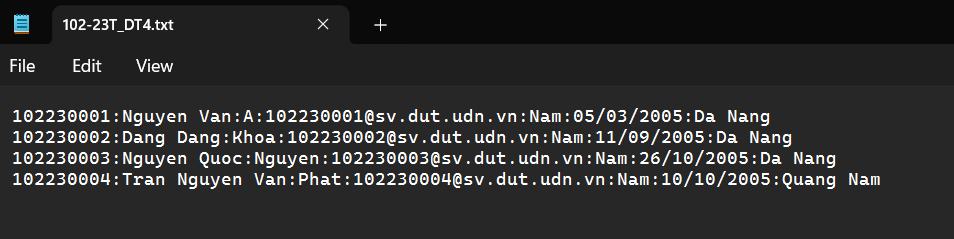
Nếu lớp có sinh viên chưa có mã sinh viên thì sẽ thông báo:



Nếu lớp chưa có sinh viên thì sẽ thông báo lớp chưa có sinh viên.



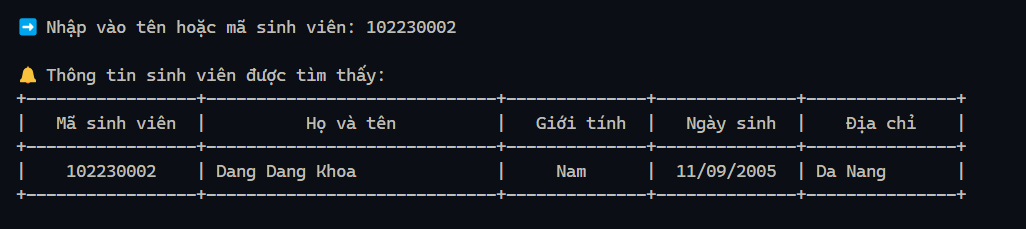
Danh sách sinh viên sau khi được cấp email:



1. **Chức năng tìm kiếm sinh viên**

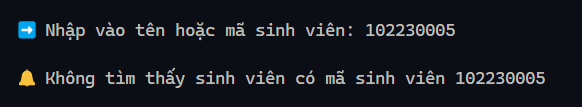
Chương trình yêu cầu người dùng nhập vào mã sinh viên hoặc tên sinh viên.

* Nếu nhập mã sinh viên:

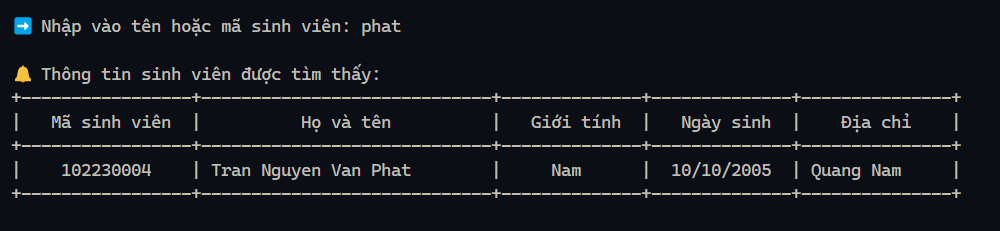


*Hình 4.10. Giao diện chức năng tìm kiếm bằng mã sinh viên*

Nếu không tìm sinh viên có mã sinh viên đã nhập thì thông báo.

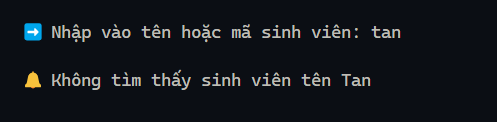


* Nếu nhập tên sinh viên:



*Hình 4.11. Giao diện chức năng tìm kiếm bằng tên*

Nếu không tìm thấy sinh viên có tên theo yêu cầu thì thông báo.



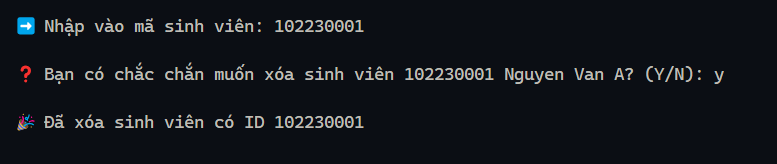
1. **Chức năng xóa sinh viên**

Chức năng yêu cầu người dùng nhập vào mã sinh viên cần xóa.

Sau khi nhập vào mã sinh viên chương trình sẽ thông báo “Bạn có chắc chắn muốn xóa sinh viên…? (Y/N):”.

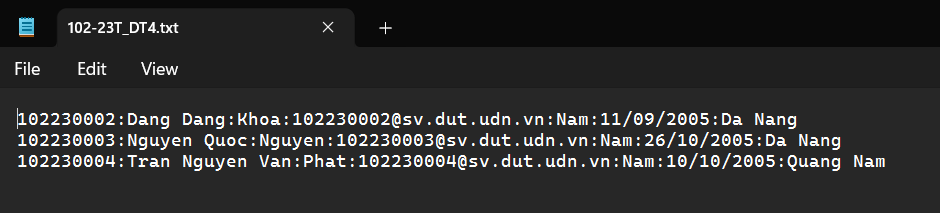
Nếu chọn N (hoặc n) thì sinh viên sẽ không được xóa và quay trở lại menu.

Nếu chọn Y (hoặc y) thì sinh viên sẽ bị xóa khỏi danh sách và in ra thông báo.



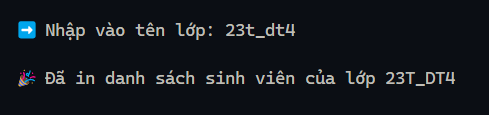
*Hình 4.12. Giao diện chức năng xóa sinh viên*

Danh sách sinh viên sau khi xóa:



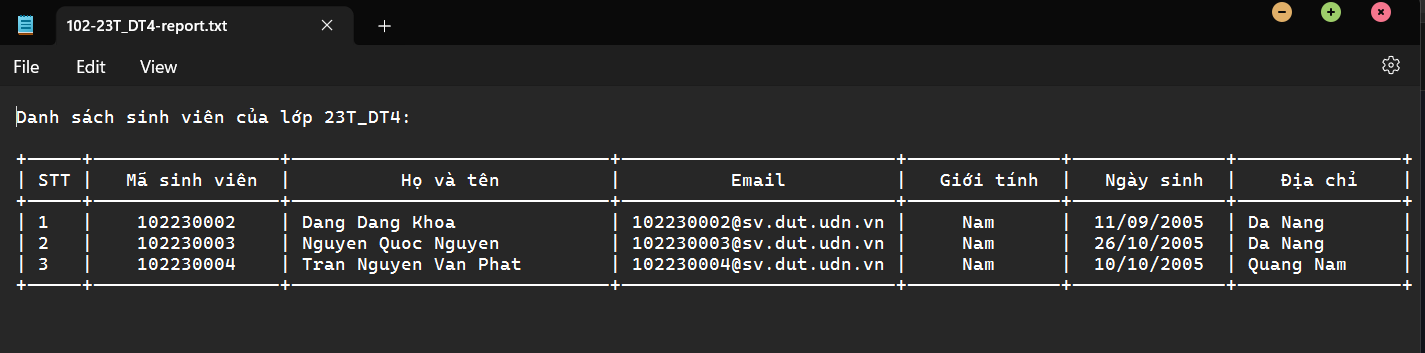
1. **Chức năng in danh sách ra file**

Chương trình yêu cầu nhập vào tên lớp cần in danh sách.



*Hình 4.13. Giao diện chức năng in danh sách*

Danh sách sinh viên được in ra file dưới dạng <mã khoa>-<tên lớp>-report.txt.



1. **Chức năng thoát**

Dùng để thoát chương trình

* + 1. **Nhận xét đánh giá**
* Nhìn chung thì chương trình đã đáp ứng đầy đủ các chức năng như là thêm sinh viên, sắp xếp sinh viên, xóa sinh viên, tìm kiếm sinh viên, cấp mã sinh viên, cấp email và in ra danh sách theo yêu cầu nhưng vẫn còn một số hạn chế như thuật toán chưa tối ưu.

# **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **Kết luận**

* Tổng quan lại, bài toán quản lý danh sách sinh viên là rất cơ bản nhưng vô cùng hữu ích và quan trọng trong việc quản lý thông tin sinh viên với các chức năng này người dùng có thể dễ dàng quản lý thông tin của các sinh viên đảm bảo tính chính xác và tăng tính hiệu quả trong việc quản lý các chức năng này có thể được áp dụng trong các trường học hoặc các tổ chức

## **Hướng phát triển**

* Tích hợp với hệ thống quản lý điểm của sinh viên,
* Thêm chức năng gửi thông báo: Bài toán có thể được phát triển thêm chức năng gửi thông báo đến sinh viên và giảng viên để thông báo về các sự kiện, chương trình đào tạo, v.v.

**TÀI LIỆU THAM KHẢO**

1. Dạy Nhau Học, Các hàm xử lý chuỗi trong C/C++

<https://daynhauhoc.com/t/wiki-cac-ham-xu-ly-chuoi-thong-dung-trong-c-c/2052>

1. 28Tech, Đọc ghi file

<https://blog.28tech.com.vn/c-doc-ghi-file-text>

1. Lập Trình Nhanh, Các hàm xử lý file trong C/C++

<https://laptrinhnhanh.com/cac-ham-xu-ly-file-trong-c-c/>

1. Trần Hồ Thủy Tiên, *Bài giảng Kỹ thuật lập trình*, Khoa Công nghệ thông tin trường Đại học Bách khoa Đà Nẵng