**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỀ TÀI: XÂY DỰNG BÀI TOÁN QUẢN LÝ SINH VIÊN DANH SÁCH LIÊN KẾT ĐÔI**

**Giáo viên hướng dẫn: ThS. Trần Thu Trang**

**Sinh viên thực hiện:**  **1. Nguyễn Tiến Lực**

**2. Nguyễn Trung Sơn**

**Hà Nội, năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**Logo, company name

Description automatically generated**

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỀ TÀI: XÂY DỰNG BÀI TOÁN QUẢN LÝ SINH VIÊN SỬ DỤNG DANH SÁCH LIÊN KẾT ĐÔI**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **1** | **1771020440** | **Nguyễn TIẾN LỰC** | **23/08/2005** |  |  |
| **2** | **1771020593** | **NGUYỄN TRUNG SƠN** | **27/04/2005** |  |  |

CÁN BỘ CHẤM THI

**Hà Nội, năm 2024**

**LỜI NÓI ĐẦU**

Cấu trúc dữ liệu và giải thuật là một lĩnh vực cực kỳ quan trọng trong khoa học máy tính và công nghệ thông tin, đóng vai trò nền tảng quan trọng trong việc xây dựng và tối ưu hóa các hệ thống phần mềm. Trong lĩnh vực này, việc áp dụng và hiểu sâu về các cấu trúc dữ liệu như danh sách liên kết đôi là điều không thể thiếu để giải quyết các bài toán thực tế một cách hiệu quả.

Đề tài "Xây dựng bài toán quản lý sinh viên sử dụng danh sách liên kết đôi" mà chúng tôi đưa ra là một nỗ lực nhằm áp dụng và thử nghiệm các khái niệm về cấu trúc dữ liệu này trong một ngữ cảnh cụ thể. Chúng tôi tập trung vào việc thiết kế và triển khai một hệ thống quản lý thông tin sinh viên linh hoạt và hiệu quả, sử dụng danh sách liên kết đôi để đáp ứng các yêu cầu khác nhau của bài toán này.

Bài toán quản lý sinh viên là một trong những bài toán phổ biến và thường xuyên gặp trong thực tế, yêu cầu phải xử lý các thao tác thêm, xoá, sắp xếp và tìm kiếm dữ liệu một cách nhanh chóng và chính xác. Sử dụng danh sách liên kết đôi giúp chúng tôi tối ưu hóa các hoạt động này, đồng thời cũng cung cấp cho chúng tôi cơ hội để khám phá và áp dụng các kỹ thuật tiên tiến trong lĩnh vực này.

Trong lời nói đầu này, chúng tôi sẽ trình bày chi tiết về cấu trúc của danh sách liên kết đôi, cùng với các thao tác quản lý cơ bản như thêm, xoá, sắp xếp và tìm kiếm. Chúng tôi cũng sẽ minh họa các ví dụ và đánh giá hiệu suất của các phương pháp được đề xuất.

Hi vọng rằng nghiên cứu này sẽ mang lại những kiến thức hữu ích và đóng góp tích cực vào lĩnh vực cấu trúc dữ liệu và giải thuật. Chúng tôi rất mong được chia sẻ và thảo luận về các kết quả và nhận xét của quý vị.

**MỤC LỤC**

Contents

[**CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI** 9](#_Toc171524742)

[**1.1.Đặt vấn đề** 9](#_Toc171524743)

[**1.3.Mục tiêu nghiên cứu** 10](#_Toc171524744)

[**1.4.Phạm vi nghiên cứu** 11](#_Toc171524745)

[**1.5.Phương pháp nghiên cứu** 12](#_Toc171524746)

[**1.6.Ý nghĩa của đề tài** 12](#_Toc171524747)

[**CHƯƠNG 2. PHÂN TÍCH ĐỀ TÀI** 14](#_Toc171524748)

[**2.1. Dữ liệu lưu trữ** 14](#_Toc171524749)

[**2.2. Cấu trúc dữ liệu** 14](#_Toc171524750)

[**2.3. Các chức năng** 14](#_Toc171524751)

[**2.4. Thiết kế menu** 20](#_Toc171524752)

[**CHƯƠNG 3. PHÂN TÍCH CHƯƠNG TRÌNH** 22](#_Toc171524753)

[**3.1. Cấu trúc dũ liệu** 22](#_Toc171524754)

[**3.2. Khai báo thư viện** 22](#_Toc171524755)

[**3.3. Khởi tạo nút** 22](#_Toc171524756)

[**3.4. Nhập dữ liệu** 23](#_Toc171524757)

[**3.5. Nhap danh sach** 24](#_Toc171524758)

[**3.6. Them sinh viên vao dau danh sach** 26](#_Toc171524759)

[**3.7. Them sinh viên cuối danh sach** 27](#_Toc171524760)

[**3.8. Chèn sinh viên vào vị trí bất kỳ** 28](#_Toc171524761)

[**3.9. Thay doi du lieu dau danh sach** 33](#_Toc171524762)

[**3.10. Thay doi du lieu** 34](#_Toc171524763)

[**3.11. Xoa sinh viên dau danh sach** 34](#_Toc171524764)

[**3.12. Xoa sinh viên cuoi danh sach** 36](#_Toc171524765)

[**3.13. Xoa sinh viên theo ID** 38](#_Toc171524766)

[**3.14. Xoa sinh viên theo ten** 39](#_Toc171524767)

[**3.15. Cập nhập thông tin liên lạc** 41](#_Toc171524768)

[**3.16. Sap xep sinh viên theo điểm trung binh** 42](#_Toc171524769)

[**3.17. Tính điểm trung bình** 45](#_Toc171524770)

[**3.18. Hiển thị sinh viên có học lực yếu** 46](#_Toc171524771)

[**3.19 Hiện thị giới tinh sinh viên** 48](#_Toc171524772)

[**3.20 Nhập danh sách từ tệp** 49](#_Toc171524773)

[**3.21 xuất danh sách sinh viên** 49](#_Toc171524774)

[**3.22 xuất file ( .txt)** 51](#_Toc171524775)

[**3.23. Xem danh sách sinh viên giỏi / khá** 52](#_Toc171524776)

[**3.24. Xem danh sách sinh viên suất xắc** 53](#_Toc171524777)

[**3.25. Gửi thông báo cho phụ huynh** 53](#_Toc171524778)

[**3.26. Xem điểm danh** 54](#_Toc171524779)

[**3.27. Cập nhập thông tin liên lạc** 54](#_Toc171524780)

[**3.28. Xóa tất cả các sinh viên** 55](#_Toc171524781)

[**3.29. Thoát** 56](#_Toc171524782)

[**CHƯƠNG 4. MÃ NGUỒN CHƯƠNG TRÌNH** 56](#_Toc171524783)

[**4.1.Khai báo cấu trúc dữ liệu** 56](#_Toc171524784)

[**4.2. Các hàm cơ bản** 57](#_Toc171524785)

[**4.3. Sắp xếp danh sách theo điểm trung bình và tính điểm trung bình** 63](#_Toc171524786)

[**4.4. Các chức năng tương tác với file** 64](#_Toc171524787)

[**4.5. Chương trình chính (main function)** 64](#_Toc171524788)

[**KẾT LUẬN** 68](#_Toc171524789)

[**KẾT LUẬN** 70](#_Toc171524790)

**MỤC LỤC HÌNH ẢNH**

[**Hình 1 thêm sinh viên vào đầu danh sách 28**](#_Toc171526493)

[**Hình 2 .thêm sinh viên cuối danh sách 29**](#_Toc171526494)

[**Hình 3 Thêm sinh viên vào vị trí bất kì 32**](#_Toc171526495)

[**Hình 4 Cập nhập thông tin sinh viên 35**](#_Toc171526496)

[**Hình 5 cập nhập ID 36**](#_Toc171526497)

[**Hình 6 xóa sinh viên 37**](#_Toc171526498)

[**Hình 7 xóa sinh viên cuối danh sách 38**](#_Toc171526499)

[**Hình 8 xóa sinh viên theo ID 40**](#_Toc171526500)

[**Hình 9 xóa sinh viên theo tên 42**](#_Toc171526501)

[**Hình 10 Cập nhập thông tin liên lạc 44**](#_Toc171526502)

[**Hình 11 Sắp xếp theo điểm trung bình 45**](#_Toc171526503)

[**Hình 12 tính điểm trung bình 47**](#_Toc171526504)

[**Hình 13 hiển thị học lực 48**](#_Toc171526505)

[**Hình 14 hiện thị giới tính 50**](#_Toc171526506)

[**Hình 15 nhập danh sách từ tệp 51**](#_Toc171526507)

[**Hình 16 xuất danh sách sinh viên 52**](#_Toc171526508)

[**Hình 17: xuất file ( vào txt) 53**](#_Toc171526509)

[**Hình 18 xem học sinh giỏi / khá 54**](#_Toc171526510)

[**Hình 19 Xem học sinh xuất sắc 55**](#_Toc171526511)

[**Hình 20 gửi thông báo phụ huynh 55**](#_Toc171526512)

[**Hình 21 xem điểm danh 56**](#_Toc171526513)

[**Hình 22 cập nhập thông tin liên lạc 56**](#_Toc171526514)

[**Hình 23 xóa tất cả sinh viên 57**](#_Toc171526515)

[**Hình 24 struct 58**](#_Toc171526516)

[**Hình 25 con trỏ 58**](#_Toc171526517)

[**Hình 26 hàm 59**](#_Toc171526518)

[**Hình 27 thêm , sửa 63**](#_Toc171526519)

[**Hình 28 xóa sinh viên 64**](#_Toc171526520)

[**Hình 29 sắp xếp và tính điểm 65**](#_Toc171526521)

[**Hình 30 xuất file vào txt 65**](#_Toc171526522)

[**Hình 31 Hàm main 1 66**](#_Toc171526523)

[**Hình 32 Hàm main 2 67**](#_Toc171526524)

[**Hình 33 Hàm main 3 67**](#_Toc171526525)

[**Hình 34 Hàm main 4 68**](#_Toc171526526)

[**Hình 35 Hàm main 5 68**](#_Toc171526527)

**BẢNG CÁC TỪ VIẾT TẮT**

**(Nếu có)**

|  |  |  |
| --- | --- | --- |
| **STT** | **TỪ VIẾT TẮT** | **VIẾT ĐẦY ĐỦ** |
| **1** | **CSDL** | **Cơ sở dữ liệu** |
| **2** |  |  |

# **CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI**

## **1.1.Đặt vấn đề**

Trước sự phát triển nhanh chóng của công nghệ thông tin và nhu cầu quản lý dữ liệu ngày càng tăng cao, các hệ thống quản lý thông tin đã trở thành một phần không thể thiếu trong hầu hết các tổ chức và doanh nghiệp hiện nay. Trong lĩnh vực giáo dục và đào tạo, việc quản lý thông tin sinh viên đóng vai trò quan trọng trong việc cải thiện chất lượng dịch vụ và tối ưu hóa quản lý.

Bài toán quản lý sinh viên không chỉ đơn giản là việc lưu trữ thông tin cá nhân, mà còn bao gồm các thao tác quản lý như thêm mới sinh viên, xoá thông tin, cập nhật dữ liệu, và tìm kiếm nhanh chóng theo nhiều tiêu chí khác nhau. Để giải quyết các yêu cầu này một cách hiệu quả, việc lựa chọn và triển khai cấu trúc dữ liệu phù hợp là vô cùng quan trọng.

Danh sách liên kết đôi (Doubly Linked List) là một trong những cấu trúc dữ liệu cơ bản nhưng mạnh mẽ, cho phép truy cập từ đầu đến cuối danh sách và ngược lại một cách dễ dàng. Sự linh hoạt của danh sách liên kết đôi là yếu tố quan trọng giúp giải quyết các thao tác quản lý dữ liệu sinh viên một cách hiệu quả và linh hoạt.

Trong phần này, chúng tôi sẽ giới thiệu cấu trúc dữ liệu danh sách liên kết đôi và nêu rõ lý do tại sao chúng tôi chọn cấu trúc này cho bài toán quản lý sinh viên. Chúng tôi cũng sẽ đi sâu vào các thao tác cơ bản như thêm mới sinh viên, xoá thông tin, cập nhật dữ liệu và tìm kiếm để minh họa sự linh hoạt và hiệu quả của danh sách liên kết đôi trong việc giải quyết bài toán này.

**1.2.Lý do chọn đề tài**

Việc lựa chọn đề tài "Xây dựng bài toán quản lý sinh viên sử dụng danh sách liên kết đôi" không chỉ đơn thuần dựa trên sự phổ biến và tính ứng dụng của cấu trúc dữ liệu này, mà còn dựa trên những lý do cụ thể sau đây:

Tính thực tiễn và sự phổ biến của bài toán: Quản lý thông tin sinh viên là một nhu cầu thường xuyên và cần thiết trong các tổ chức giáo dục, trường học và các cơ quan quản lý. Việc xây dựng một hệ thống quản lý hiệu quả không chỉ giúp cải thiện quản lý mà còn nâng cao chất lượng phục vụ.

Áp dụng cấu trúc dữ liệu danh sách liên kết đôi: Danh sách liên kết đôi là một trong những cấu trúc dữ liệu linh hoạt và mạnh mẽ, cho phép truy cập đồng thời từ đầu đến cuối và từ cuối về đầu danh sách. Đặc tính này rất phù hợp để giải quyết các thao tác quản lý như thêm, xoá, sắp xếp và tìm kiếm trong bài toán quản lý sinh viên.

Phát triển kỹ năng lập trình và hiểu biết về cấu trúc dữ liệu: Việc nghiên cứu và triển khai bài toán này không chỉ giúp củng cố kiến thức về cấu trúc dữ liệu và giải thuật mà còn phát triển kỹ năng lập trình và khả năng phân tích, thiết kế hệ thống.

Tiềm năng ứng dụng và mở rộng: Kết quả từ đề tài này có thể mở ra những tiềm năng ứng dụng rộng rãi trong các hệ thống quản lý thông tin khác nhau, không chỉ trong lĩnh vực giáo dục mà còn trong các ngành công nghiệp khác.

Việc lựa chọn đề tài này đồng nghĩa với việc khám phá và ứng dụng các kiến thức học thuật vào thực tiễn, đem lại lợi ích rõ rệt cho cả nghiên cứu và ứng dụng thực tế. Trong phần tiếp theo của đề tài, chúng tôi sẽ trình bày chi tiết về thiết kế và triển khai hệ thống quản lý sinh viên sử dụng danh sách liên kết đôi, cùng với các thí nghiệm và phân tích kết quả để minh họa sự hiệu quả của phương pháp này.

## **1.3.Mục tiêu nghiên cứu**

Mục tiêu của đề tài "Xây dựng bài toán quản lý sinh viên sử dụng danh sách liên kết đôi" được xác định nhằm đạt được những mục tiêu cụ thể sau đây:

Thiết kế và triển khai hệ thống quản lý sinh viên: Phát triển một hệ thống quản lý sinh viên hoàn chỉnh và có tính ứng dụng cao, sử dụng cấu trúc dữ liệu danh sách liên kết đôi để lưu trữ và quản lý thông tin sinh viên một cách hiệu quả.

Thực hiện các thao tác quản lý cơ bản: Triển khai các chức năng quản lý cơ bản như thêm mới sinh viên, xoá thông tin, cập nhật dữ liệu và tìm kiếm theo nhiều tiêu chí khác nhau để đảm bảo tính toàn vẹn và đáp ứng nhu cầu thực tế của người dùng.

Đánh giá hiệu suất và tính năng của danh sách liên kết đôi: Đo lường và phân tích hiệu suất của danh sách liên kết đôi trong các thao tác quản lý, bao gồm thời gian thực thi và tốn tài nguyên, để đánh giá tính khả thi và hiệu quả của phương pháp này.

Nghiên cứu và phát triển các cải tiến: Nghiên cứu và đề xuất các cải tiến hoặc mở rộng sử dụng danh sách liên kết đôi trong việc quản lý sinh viên, nhằm cải thiện hiệu suất và linh hoạt của hệ thống.

Tạo ra tài liệu và hướng dẫn sử dụng: Biên soạn tài liệu hướng dẫn chi tiết về cách sử dụng hệ thống và cấu trúc dữ liệu danh sách liên kết đôi cho các nhà phát triển và người quản lý hệ thống.

Mục tiêu nghiên cứu được đặt ra nhằm giúp đảm bảo rằng đề tài không chỉ mang tính lý thuyết mà còn có ứng dụng thực tế cao, đem lại giá trị và lợi ích cho cả lĩnh vực giáo dục và công nghệ thông tin. Phần tiếp theo của đề tài sẽ trình bày chi tiết về quá trình thiết kế, triển khai, và đánh giá các mục tiêu này.

## **1.4.Phạm vi nghiên cứu**

Phạm vi chức năng:

Triển khai các chức năng quản lý cơ bản: bao gồm thêm mới sinh viên, xoá thông tin, cập nhật dữ liệu và tìm kiếm.

Đảm bảo tính toàn vẹn và hiệu quả của các thao tác quản lý.

Đánh giá và so sánh hiệu suất của cấu trúc dữ liệu danh sách liên kết đôi so với các cấu trúc dữ liệu khác trong bài toán quản lý sinh viên.

Phạm vi kỹ thuật:

Thiết kế và triển khai cấu trúc dữ liệu danh sách liên kết đôi.

Xây dựng các thuật toán cho các thao tác quản lý.

Kiểm thử, đánh giá và tối ưu hóa hiệu suất của hệ thống.

Phạm vi ứng dụng:

Áp dụng vào việc quản lý thông tin sinh viên trong một môi trường giả định, đáp ứng các yêu cầu cơ bản của hệ thống quản lý thông tin.

Không bao gồm các yếu tố mở rộng như tích hợp với các hệ thống khác hoặc mở rộng chức năng quản lý nâng cao.

Phạm vi thời gian:

Nghiên cứu và triển khai trong khoảng thời gian nhất định, tập trung vào việc đạt được các mục tiêu nghiên cứu cơ bản đã đề ra.

Phạm vi nghiên cứu này được định hướng để tập trung vào việc áp dụng cấu trúc dữ liệu danh sách liên kết đôi vào bài toán quản lý sinh viên một cách hiệu quả và có tính ứng dụng cao. Việc hạn chế phạm vi giúp đảm bảo rằng đề tài có thể được thực hiện một cách có hệ thống và đem lại những kết quả rõ ràng và có giá trị.

## **1.5.Phương pháp nghiên cứu**

Nghiên cứu lý thuyết: Trước khi triển khai thực nghiệm, chúng tôi sẽ tiến hành nghiên cứu sâu về cấu trúc dữ liệu danh sách liên kết đôi và các thuật toán liên quan. Nghiên cứu này sẽ bao gồm việc đọc và phân tích các tài liệu, sách vở, bài báo khoa học và các tài liệu trực tuyến có liên quan.

Thiết kế và triển khai hệ thống: Sau khi có nền tảng lý thuyết vững chắc, chúng tôi sẽ tiến hành thiết kế chi tiết và triển khai hệ thống quản lý sinh viên. Việc này sẽ bao gồm lựa chọn và cài đặt môi trường phát triển, thiết kế giao diện và các module chức năng.

Thử nghiệm và đánh giá: Sau khi triển khai, chúng tôi sẽ tiến hành các thử nghiệm để đảm bảo tính ổn định và đúng đắn của hệ thống. Chúng tôi sẽ thực hiện các ca kiểm thử đa dạng để kiểm tra tính toàn vẹn dữ liệu, tính năng của các thao tác quản lý và hiệu suất của hệ thống.

Phân tích và so sánh: Dựa trên kết quả thực nghiệm, chúng tôi sẽ phân tích và so sánh hiệu suất của cấu trúc dữ liệu danh sách liên kết đôi với các cấu trúc dữ liệu khác được áp dụng trong cùng bài toán quản lý sinh viên. Đây là bước quan trọng để đánh giá tính hiệu quả và sự lựa chọn đúng đắn của phương pháp nghiên cứu.

Tổng kết và báo cáo: Cuối cùng, chúng tôi sẽ tổng kết kết quả nghiên cứu và viết báo cáo khoa học, trình bày chi tiết về quá trình nghiên cứu, thiết kế hệ thống, kết quả thực nghiệm và những kết luận rút ra từ nghiên cứu.

Phương pháp nghiên cứu được lựa chọn sẽ giúp đảm bảo tính khoa học và ứng dụng thực tiễn của đề tài, từ đó đem lại những kết quả đáng tin cậy và có giá trị cho lĩnh vực cấu trúc dữ liệu và quản lý thông tin sinh viên.

## **1.6.Ý nghĩa của đề tài**

Đề tài "Xây dựng bài toán quản lý sinh viên sử dụng danh sách liên kết đôi" mang lại nhiều ý nghĩa quan trọng như sau:

Ứng dụng thực tế: Hệ thống quản lý sinh viên là một ứng dụng thực tế quan trọng trong các tổ chức giáo dục và đào tạo. Việc xây dựng một hệ thống sử dụng cấu trúc dữ liệu danh sách liên kết đôi giúp cải thiện quản lý thông tin sinh viên một cách hiệu quả và linh hoạt.

Nghiên cứu và phát triển công nghệ: Đề tài cung cấp cơ hội để nghiên cứu và phát triển các phương pháp, thuật toán và công nghệ trong lĩnh vực cấu trúc dữ liệu và giải thuật, đặc biệt là ứng dụng của danh sách liên kết đôi trong quản lý dữ liệu.

Giáo dục và đào tạo: Đề tài cung cấp tài liệu tham khảo và mô hình minh họa cho giảng dạy và học tập trong các khoa học máy tính, khoa học thông tin và các lĩnh vực liên quan đến quản lý thông tin và cơ sở dữ liệu.

Cải tiến và tối ưu hóa: Kết quả từ đề tài có thể đóng góp vào việc cải tiến và tối ưu hóa các hệ thống quản lý thông tin sinh viên hiện tại, từ đó nâng cao hiệu quả và chất lượng dịch vụ.

Mở rộng và áp dụng: Các kết quả nghiên cứu có thể được mở rộng và áp dụng vào các lĩnh vực khác trong công nghệ thông tin và quản lý dữ liệu, không chỉ giới hạn trong lĩnh vực giáo dục mà còn trong các ngành công nghiệp khác.

Đề tài mang lại ý nghĩa lớn đối với nghiên cứu khoa học, phát triển công nghệ và cải thiện chất lượng dịch vụ trong quản lý thông tin sinh viên. Nó cũng đóng góp vào việc nâng cao kiến thức và kỹ năng của các nhà nghiên cứu và sinh viên trong lĩnh vực khoa học máy tính và công nghệ thông tin.

# **CHƯƠNG 2. PHÂN TÍCH ĐỀ TÀI**

## **2.1. Dữ liệu lưu trữ**

− Dữ liệu cần quản lý và lưu trữ về thuê bao gồm:

ID sinh viên – kiểu int  
 Tên- kiểu string   
Giới tinh – kiểu string  
Điểm- kiểu float

## **2.2. Cấu trúc dữ liệu**

struct SinhVien {

int id;

std::string ten;

std::string gioiTinh;

float diemTB;

SinhVien\* next;

SinhVien\* prev;

};

## **2.3. Các chức năng**

Chương trình quản lý thuê bao cần cung cấp các chức năng sau:

**1. Nhập danh sách.**

Cho người dùng nhập vào:

* Nhập số ID
* Giới tinh
* Tên
* Số điểm

**2.** **Thêm Sinh viên vào đầu danh sách.**

Cho người dùng nhập vào:

* Tên học sinh
* Điểm
* Giới tinh

**3. Thêm Sinh viên vào cuối danh sách.**

Cho người dùng nhập vào:

* Tên học sinh
* Điểm
* Giới tinh

**4.Chèn Sinh viên vào vị trí bất kỳ**

Cho người dùng nhập vào:

* Tên học sinh
* Điểm
* Giới tinh
* Nhập vị trí cần chèn

Chức năng này để giúp người dùng nhập thêm 1 Sinh viên ở bất kỳ vị trí nào trong danh sách,rồi nhập thông tin như mới đó.

**5. Thay đổi dữ liệu đầu danh sách**

Cho người dùng nhập vào:

* Tên học sinh
* Điểm
* Giới tinh

Chức năng này để giúp người dùng thay đổi thông tin đầu tiên trong danh sách.

**6. Thay đổi dữ liệu cuối danh sách**

Cho người dùng nhập vào:

* Tên học sinh
* Điểm
* Giới tinh

Chức năng này để giúp người dùng thay đổi thông tin cuối cùng trong danh sách.

**7. Xóa Sinh viên danh sách**

* Tên học sinh
* Điểm
* Giới tinh

Chức năng này để giúp người dùng xóa sinh viên đầu trong danh sách.

**8. Xóa sinh viên cuối danh sách**

* Tên học sinh
* Điểm
* Giới tinh

Chức năng này để giúp người dùng xóa sinh viên cuối cùng trong danh sách.

**9. Xóa sinh viên theo tên**

Cho người dùng nhập vào:

* Tên học sinh
* Điểm
* Giới tinh

Chức năng này giúp người dùng xóa theo tên mà người dùng nhập.

**10. Xóa sinh viên theo số ID**

Cho người dùng nhập vào:

* Tên học sinh Hoặc ID
* Điểm
* Giới tinh

Chức năng này giúp người dùng xóa theo số ID mà người dùng nhập.

**11. Tìm sinh viên theo số ID**

Cho người dùng nhập vào:

* Tên học sinh Hoặc ID
* Điểm
* Giới tinh

Chức năng này giúp ngươi dùng tìm bằng cách nhập số id của sinh viên đó rồi in ra màn hình.

**12. Tìm sinh viên theo tên**

Cho người dùng nhập vào:

* Tên học sinh Hoặc ID
* Điểm
* Giới tinh

Chức năng này giúp ngươi dùng tìm sinh viên bằng cách nhập tên sinh viên sinh viên đó

Rồi in ra màn hình.

**13.Sắp xếp sinh viên theo điểm trung binh theo thứ tự tăng dần**

* Tên học sinh Hoặc ID
* Điểm
* Giới tinh

Chức năng này giúp cho người dùng sắp xếp sinh viên theo điểm trung bình theo thứ tự tăng dần rồi in ra màn hình

**14. Tính trung bình**

* Tên học sinh Hoặc ID
* Điểm ( tổng các môn / )
* Giới tinh

Chức năng sẽ tính trung bình rồi in ra màn hình.

**15. Xóa các sinh viên có điểm hơn 5**

* Tên học sinh Hoặc ID
* Điểm ( tổng các môn / )
* Giới tinh

Chức năng này sẽ giúp người dùng xóa các sinh viên có điểm nhỏ hơn 5

**16. In danh sác ra tệp**

In ra tệp .txt

**17. Nhập danh sách từ tệp**

Nhập tất cả những gì minh liệt kê vào trong tệp txt điểm tên giới tinh

**18. In ra danh sách sinh viên**

Chức năng này sẽ in toàn bộ sinh viên trong danh sách ra màn hình.

**19. Xóa toàn bộ danh sách**

Chức năng này sẽ xóa toàn sinh viên trong danh sách.

**20. In danh sách theo thứ tự ngược lại**

Chức năng này sẽ in toàn bộ sinh viên trong danh sách ra màn hình theo thứ tự ngược lại.

**21. Đếm số lượng sinh viên**

Chức năng này sẽ đém số sinh viên cho người dùng rồi thông báo số lượng sinh viên ra màn hình.

**22. Thêm sinh viên và sắp xếp thueo thứ tự giảm dần**

Cho người dùng nhập vào:

* Tên học sinh Hoặc ID
* Điểm ( tổng các môn / )
* Giới tinh

Chức năng này sẽ thêm sinh viên rồi sắp xếp thuê bao theo thứ tự giảm dàn rồi in ra màn hình.

**23. In danh sach theo thu tu nguoc lai**

Chức năng này sẽ in danh sách theo thứ tự ngược lại.

**24. Tìm và in sinh viên theo số ID**

Chức năng này sẽ giúp người dùng tìm sinh viên theo số ID và in ra màn hình.

**25.Tìm sinh viên theo tên rồi in ra.**

Chức năng này sẽ giúp người dùng tìm và in ra màn hình toàn bộ sinh viên có ký tự mà người dùng nhập.

**26. Thoát khỏi chương trình**

Chức năng này thoát khỏi chương trình.

## **2.4. Thiết kế menu**

---------CAC CHUC NANG CUA CHUONG TRINH---------

1.Nhập danh sách sinh viên (nhapDanhSachSinhVien())

2.Thêm sinh viên vào đầu danh sách (themSinhVienDauDanhSach())

3.Thêm sinh viên vào cuối danh sách (themSinhVienCuoiDanhSach())

4.Thêm sinh viên vào vị trí bất kỳ (themSinhVienViTriBatKy())

5.Cập nhật thông tin sinh viên (capNhapThongTinSinhVien(SinhVien\* sv))

6.Xóa sinh viên đầu danh sách (xoaSinhVienDauDanhSach())

7.Xóa sinh viên cuối danh sách (xoaSinhVienCuoiDanhSach())

8.Xóa sinh viên theo ID (xoaSinhVienTheoID(int id))

9.Cập nhật ID sinh viên (capNhapID(int oldID, int newID))

10.Xóa sinh viên theo tên (xoaSinhVienTheoTen(std::string ten))

11.Sắp xếp sinh viên theo điểm trung bình giảm dần (sapXepTheoDiemTB())

12.Tính điểm trung bình của toàn bộ sinh viên (tinhDiemTrungBinh())

13.Hiển thị danh sách sinh viên học lực yếu (hienThiHocLucYeu())

14.Hiển thị danh sách sinh viên nam (hienThiSinhVienNam())

15.Hiển thị danh sách sinh viên nữ (hienThiSinhVienNu())

16.Xuất danh sách sinh viên ra màn hình (xuatDanhSachSinhVien())

17.Xuất danh sách sinh viên vào file (xuatFile(std::string filename))

18.Xem danh sách sinh viên học sinh giỏi (xemDanhSachSinhVienGioi())

19.Xem danh sách sinh viên học sinh khá (xemDanhSachSinhVienKha())

20.Xem danh sách sinh viên xuất sắc (xemHocSinhXuatSac(int n))

21.Danh sách học sinh nam đủ điều kiện lên lớp (hocSinhDuYeuCauLenLop())

22.Danh sách học sinh không đủ điều kiện lên lớp (danhSachHocSinhOLaiLop())

23.Gửi thông báo cho phụ huynh học sinh yếu (guiThongBaoChoPhuHuynh())

24.Xem điểm danh của sinh viên (xemDiemDanh())

25.Cập nhật thông tin liên lạc của sinh viên (capNhatThongTinLienLac())

26.Xóa toàn bộ sinh viên khỏi danh sách (xoaTatCaSinhVien())

27.Thoát chương trình (menu() sẽ tiếp tục lặp đến khi người dùng chọn thoát)

# **CHƯƠNG 3. PHÂN TÍCH CHƯƠNG TRÌNH**

## **3.1. Cấu trúc dũ liệu**

struct SinhVien {

int id;

std::string ten;

std::string gioiTinh;

float diemTB;

SinhVien\* next;

SinhVien\* prev;

};

struct QuanLySinhVien {

private:

SinhVien\* head; // dau

SinhVien\* tail; // cuoi

## **3.2. Khai báo thư viện**

− Khai báo các thư viện của C++

#include <iostream>

#include <fstream>

#include <string>

## **3.3. Khởi tạo nút**

struct QuanLySinhVien {

private:

SinhVien\* head; // dau

SinhVien\* tail; // cuoi

public:

QuanLySinhVien();

~QuanLySinhVien();

## **3.4. Nhập dữ liệu**

− Mô tả chức năng

Chương trình cung cấp các chức năng để nhập dữ liệu cho các sinh viên.

− Modul mã nguồn

QuanLySinhVien::~QuanLySinhVien() {

SinhVien\* current = head;

while (current != NULL) {

SinhVien\* next = current->next;

delete current;

current = next;

}

head = NULL;

tail = NULL; // huy

}

Khởi tạo con trỏ current là head:

Bắt đầu từ đầu danh sách (sinh viên đầu tiên), gán current là head.

Vòng lặp while:

Lặp qua danh sách liên kết đôi, dừng khi current là NULL (tức là đã duyệt hết danh sách).

Trong mỗi vòng lặp:

Gán next là current->next để lưu giữ con trỏ đến sinh viên tiếp theo trong danh sách.

Xóa current để giải phóng bộ nhớ của sinh viên hiện tại.

Cập nhật current thành next, tức là con trỏ đến sinh viên kế tiếp trong danh sách.

Gán head và tail về NULL:

Sau khi thoát khỏi vòng lặp, cả head và tail được gán lại thành NULL, đảm bảo rằng danh sách không còn tồn tại nữa sau khi đã giải phóng hết bộ nhớ

**3.5. Nhap danh sach**

void QuanLySinhVien::nhapDanhSachSinhVien() {

int n;

std::cout << "so luong sinh ";

std::cin >> n;

for (int i = 0; i < n; ++i) {

SinhVien\* sv = new SinhVien;

std::cout << "nhap thong tin sinh vien " << i + 1 << ":" << std::endl;

std::cout << "ID: ";

std::cin >> sv->id;

std::cout << "ten: ";

std::cin.ignore(); // don dep bo nho

std::getline(std::cin, sv->ten);

std::cout << "Gioi tinh: ";

std::getline(std::cin, sv->gioiTinh);

std::cout << "diem trung binh: ";

std::cin >> sv->diemTB;

sv->next = NULL;

sv->prev = NULL;

if (head == NULL) {

head = sv;

tail = sv;

} else {

tail->next = sv;

sv->prev = tail;

tail = sv;

}

}

Nhập số lượng sinh viên (n):

Người dùng nhập vào số lượng sinh viên mà họ muốn nhập thông tin.

Vòng lặp nhập thông tin từng sinh viên:

Đối với mỗi sinh viên:

Tạo một đối tượng SinhVien mới.

Nhập các thông tin như ID, tên, giới tính, điểm trung bình.

Dùng std::cin.ignore() để xóa bộ đệm để tránh vấn đề khi sử dụng std::getline() sau khi sử dụng std::cin.

Cấu trúc dữ liệu của sinh viên:

Mỗi sinh viên được lưu trữ trong một cấu trúc (struct hoặc class) có các thành phần như id, ten, gioiTinh, diemTB, và các con trỏ next, prev để tạo danh sách liên kết đôi.

Thêm sinh viên vào danh sách liên kết đôi:

Kiểm tra nếu danh sách chưa có sinh viên (head == NULL), thì sinh viên đó trở thành head và tail.

Nếu danh sách đã có sinh viên, thêm sinh viên mới vào cuối danh sách:

Gán tail->next là sinh viên mới.

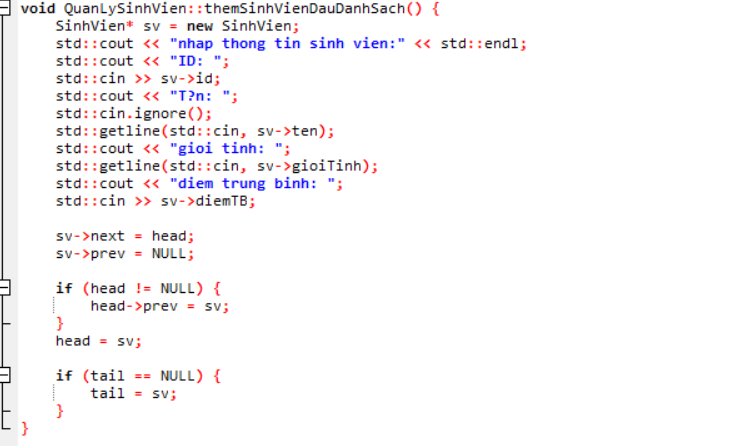
Gán sv->prev là tail.

Cập nhật tail thành sinh viên mới.

Sau khi nhập xong danh sách sinh viên, head sẽ trỏ vào sinh viên đầu tiên trong danh sách.

tail sẽ trỏ vào sinh viên cuối cùng trong danh sách, giúp quản lý dễ dàng thao tác thêm, xóa, sửa trong danh sách.

## **3.6. Them sinh viên vao dau danh sach**



Hình 1 thêm sinh viên vào đầu danh sách

Tạo một nút mới: Hàm createNode(d) được gọi để tạo một nút mới với dữ liệu d. Nút mới này được lưu trong con trỏ pNode.

Kiểm tra danh sách rỗng: Nếu danh sách hiện tại đang rỗng (tức là list.head == NULL), thì:

list.head được gán bằng pNode.

list.tail cũng được gán bằng pNode.

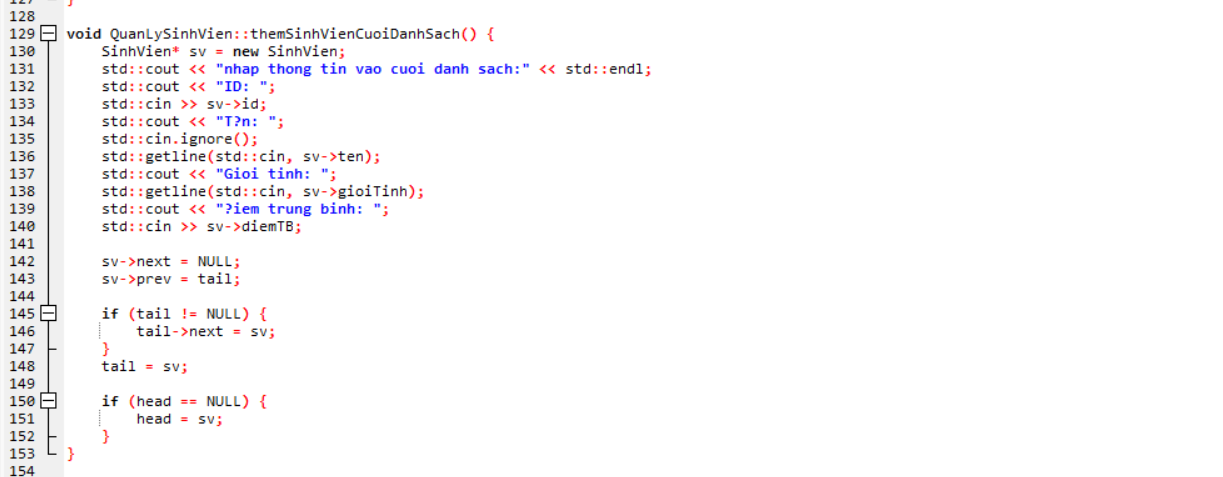
Danh sách không rỗng: Nếu danh sách không rỗng, tức là có ít nhất một phần tử:

pNode->next được gán bằng list.head (nút mới sẽ trỏ đến nút đầu hiện tại).

list.head->prev được gán bằng pNode (nút đầu hiện tại sẽ trỏ ngược lại nút mới).

list.head được cập nhật để trở thành pNode (nút mới trở thành nút đầu của danh sách).

**3.7. Them sinh viên cuối danh sach**



Hình 2 .thêm sinh viên cuối danh sách

Đầu tiên, hàm cấp phát một đối tượng SinhVien mới thông qua new SinhVien.

Sau đó, nó yêu cầu người dùng nhập các thông tin của sinh viên bao gồm ID, tên, giới tính và điểm trung bình. Lưu ý sử dụng std::cin.ignore() để loại bỏ kí tự '\n' thừa trong bộ đệm nhập, đảm bảo việc std::getline() hoạt động chính xác.

sv->next = NULL; // Vì đây là phần tử cuối cùng nên next của sv sẽ là NULL

sv->prev = tail; // Phần tử trước đó của sv sẽ là phần tử tail hiện tại

if (tail != NULL) {

tail->next = sv; // Nếu danh sách không rỗng, cập nhật phần tử tail hiện tại để trỏ tới sv

}

Sau khi có thông tin của sinh viên (sv), hàm thiết lập sv->next thành NULL vì sv sẽ là phần tử cuối cùng của danh sách.

sv->prev được thiết lập là tail, tức là phần tử trước đó của sv là tail (phần tử cuối cùng hiện tại của danh sách).

Nếu tail không rỗng (tail != NULL), nghĩa là danh sách đã có phần tử, hàm cập nhật tail->next để trỏ về sv, thiết lập mối liên kết tiếp theo từ phần tử cuối danh sách hiện tại.

// Cập nhật head và tail của danh sách

tail = sv; // sv trở thành phần tử cuối cùng của danh sách

if (head == NULL) {

head = sv; // Nếu danh sách trống, sv cũng là phần tử đầu tiên

}

}

Cuối cùng, hàm cập nhật tail bằng sv, để sv trở thành phần tử cuối cùng của danh sách.

Nếu head đang là NULL (tức là danh sách ban đầu rỗng), thì hàm gán head bằng sv, bởi vì lúc này sv cũng là phần tử duy nhất trong danh sách.

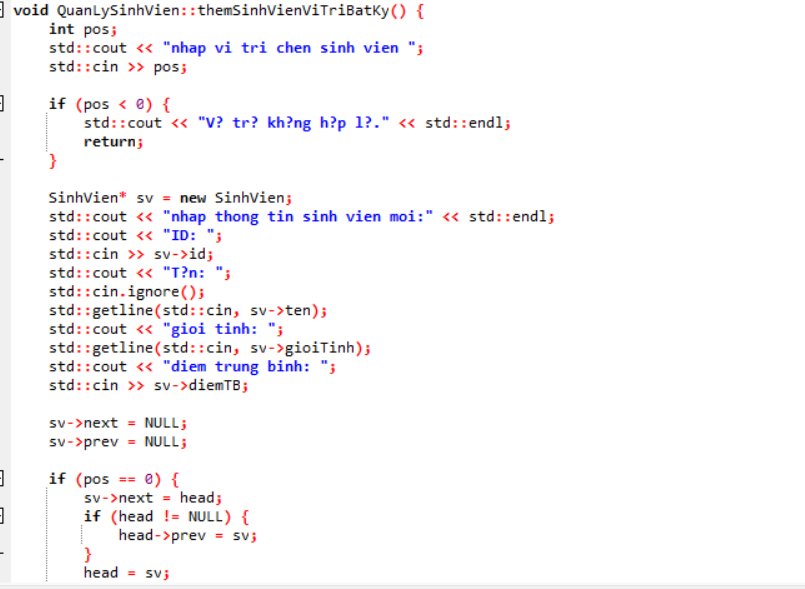
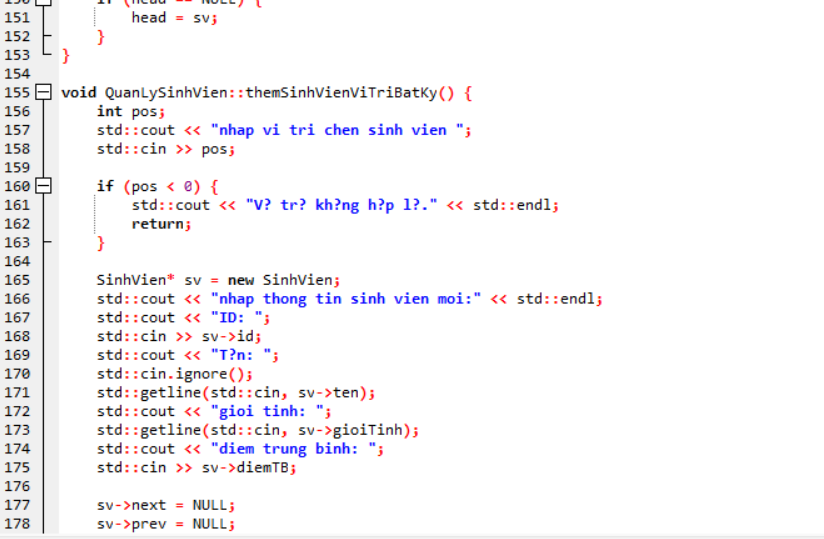
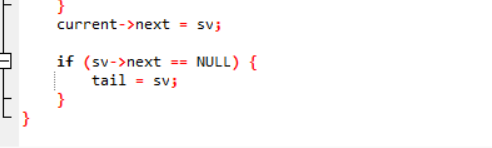
Hàm themSinhVienCuoiDanhSach() thực hiện các bước sau:

Cấp phát bộ nhớ cho một sinh viên mới và nhập thông tin của sinh viên từ người dùng.

Thiết lập các liên kết để đưa sinh viên mới vào cuối danh sách liên kết đôi.

Cập nhật head và tail của danh sách nếu cần thiết.

**3.8. Chèn sinh viên vào vị trí bất kỳ**

Hình 3 Thêm sinh viên vào vị trí bất kì

void QuanLySinhVien::themSinhVienViTriBatKy() {

int pos;

std::cout << "Nhap vi tri chen sinh vien: ";

std::cin >> pos;

if (pos < 0) {

std::cout << "Vi tri khong hop le." << std::endl;

return;

}

Giải thích:

Người dùng nhập vào vị trí (pos) mà họ muốn chèn sinh viên mới vào.

Nếu pos nhỏ hơn 0, thông báo vị trí không hợp lệ và kết thúc hàm.

2. Tạo sinh viên mới và nhập thông tin

SinhVien\* sv = new SinhVien;

std::cout << "Nhap thong tin sinh vien moi:" << std::endl;

std::cout << "ID: ";

std::cin >> sv->id;

std::cout << "Ten: ";

std::cin.ignore();

std::getline(std::cin, sv->ten);

std::cout << "Gioi tinh: ";

std::getline(std::cin, sv->gioiTinh);

std::cout << "Diem trung binh: ";

std::cin >> sv->diemTB;

sv->next = NULL;

sv->prev = NULL;

Tạo một sinh viên mới (sv) và yêu cầu người dùng nhập các thông tin liên quan như ID, tên, giới tính và điểm trung bình.

Khởi tạo sv->next và sv->prev bằng NULL để đảm bảo rằng sv là một nút độc lập và chưa được liên kết với bất kỳ phần tử nào trong danh sách.

3. Xử lý trường hợp chèn vào đầu danh sách

if (pos == 0) {

sv->next = head;

if (head != NULL) {

head->prev = sv;

}

head = sv;

if (tail == NULL) {

tail = sv;

}

return;

}

Nếu pos == 0, tức là người dùng muốn chèn sinh viên vào đầu danh sách:

Thiết lập sv->next để trỏ tới head hiện tại.

Nếu head không rỗng, thiết lập head->prev để trỏ tới sv.

Cập nhật head thành sv.

Nếu tail đang là NULL, gán tail bằng sv.

Kết thúc hàm.

4. Xử lý trường hợp chèn vào vị trí bất kỳ trong danh sách

SinhVien\* current = head;

int count = 0;

while (current != NULL && count < pos - 1) {

current = current->next;

count++;

}

if (current == NULL) {

std::cout << "Vi tri chen khong hop le." << std::endl;

delete sv;

return;

}

sv->next = current->next;

sv->prev = current;

if (current->next != NULL) {

current->next->prev = sv;

}

current->next = sv;

if (sv->next == NULL) {

tail = sv;

}

Nếu không phải chèn vào đầu danh sách (pos != 0):

Duyệt danh sách để tìm nút ở vị trí pos - 1.

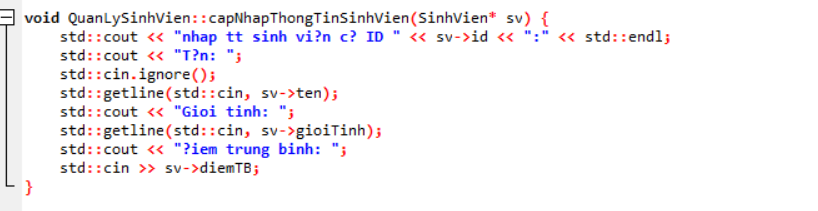
Nếu current == NULL, tức là không tìm thấy vị trí hợp lệ để chèn (pos lớn hơn số phần tử trong danh sách).

Trong trường hợp hợp lệ, thiết lập sv->next để trỏ tới current->next và sv->prev để trỏ tới current.

Nếu current->next không rỗng, thiết lập current->next->prev để trỏ tới sv.

Cập nhật current->next thành sv.

**3.9. Thay doi du lieu dau danh sach**

****

Hình 4Cập nhập thông tin sinh viên

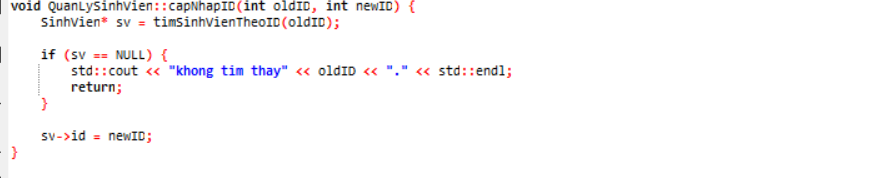
Hàm yêu cầu người dùng cập nhật thông tin cho sinh viên có sẵn với ID là sv->id.

Dùng std::cin.ignore() để xóa bộ nhớ đệm của cin, đảm bảo rằng không có dữ liệu cũ nào còn lại trong bộ nhớ đệm trước khi nhập chuỗi (sv->ten).

Sau đó, sử dụng std::getline(std::cin, sv->ten) để nhập tên sinh viên, bao gồm cả trường hợp tên chứa khoảng trắng.

Tiếp theo, nhập giới tính (sv->gioiTinh) và điểm trung bình (sv->diemTB).

**3.10. Thay doi du lieu**

****

Hình 5 cập nhập ID

SinhVien\* sv = timSinhVienTheoID(oldID);

Hàm timSinhVienTheoID(oldID) được gọi để tìm sinh viên có id bằng oldID. Nếu không tìm thấy, sv sẽ nhận giá trị NULL.

Kiểm tra sinh viên có tồn tại hay không:

if (sv == NULL) {

std::cout << "khong tim thay" << oldID << "." << std::endl;

return;

}

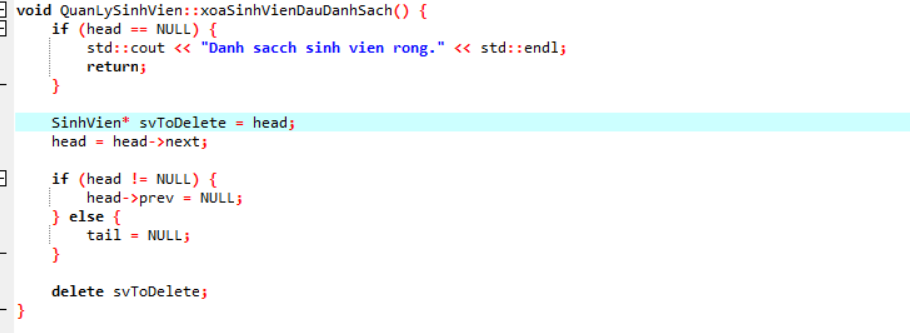
Nếu sv là NULL, tức là không tìm thấy sinh viên có id là oldID. Hàm sẽ in ra thông báo lỗi và kết thúc hàm.

Cập nhật ID mới cho sinh viên:

sv->id = newID;

Nếu sinh viên được tìm thấy (sv không phải NULL), id của sinh viên đó được cập nhật thành newID.

**3.11. Xoa sinh viên dau danh sach**



Hình 6 xóa sinh viên

Kiểm tra danh sách rỗng

void QuanLySinhVien::xoaSinhVienDauDanhSach() {

if (head == NULL) {

std::cout << "Danh sach sinh vien rong." << std::endl;

return;

}

Đầu tiên, hàm kiểm tra nếu head (con trỏ đầu danh sách) là NULL, tức là danh sách rỗng. Trong trường hợp này, hàm in ra thông báo và kết thúc hàm bằng lệnh return, không có sinh viên nào để xóa.

Cập nhật con trỏ head và tail

SinhVien\* svToDelete = head;

head = head->next;

if (head != NULL) {

head->prev = NULL;

} else {

tail = NULL;

}

Nếu danh sách không rỗng (head != NULL), con trỏ svToDelete được gán bằng head, tức là sinh viên đầu tiên trong danh sách.

Con trỏ head được cập nhật để trỏ vào sinh viên kế tiếp của sinh viên hiện tại (head = head->next).

Nếu head không phải là NULL sau khi cập nhật, tức là danh sách vẫn chứa các sinh viên khác, ta cần cập nhật con trỏ prev của head thành NULL để ngắt kết nối với svToDelete.

Nếu head đã là NULL sau khi cập nhật, tức là danh sách chỉ có một sinh viên và sau khi xóa thì danh sách trở thành rỗng, ta cũng cần cập nhật tail thành NULL.

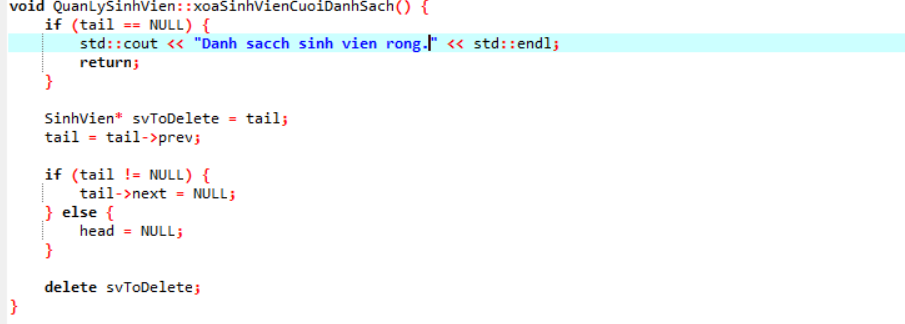
Xóa sinh viên và giải phóng bộ nhớ

delete svToDelete;

}

Cuối cùng, con trỏ svToDelete trỏ đến sinh viên đầu tiên trong danh sách được giải phóng bằng lệnh delete. Điều này giải phóng bộ nhớ đã cấp phát cho đối tượng SinhVien mà svToDelete đang trỏ đến.

**3.12. Xoa sinh viên cuoi danh sach**



Hình 7 xóa sinh viên cuối danh sách

void QuanLySinhVien::xoaSinhVienCuoiDanhSach() {

if (tail == NULL) {

std::cout << "Danh sach sinh vien rong." << std::endl;

return;

}

Đầu tiên, hàm kiểm tra nếu tail (con trỏ cuối danh sách) là NULL, tức là danh sách rỗng. Trong trường hợp này, hàm in ra thông báo và kết thúc hàm bằng lệnh return, không có sinh viên nào để xóa.

2. Cập nhật con trỏ tail và head

SinhVien\* svToDelete = tail;

tail = tail->prev;

if (tail != NULL) {

tail->next = NULL;

} else {

head = NULL;

}

Nếu danh sách không rỗng (tail != NULL), con trỏ svToDelete được gán bằng tail, tức là sinh viên cuối cùng trong danh sách.

Con trỏ tail được cập nhật để trỏ vào sinh viên trước sinh viên hiện tại (tail = tail->prev).

Nếu tail không phải là NULL sau khi cập nhật, tức là danh sách vẫn chứa các sinh viên khác, ta cần cập nhật con trỏ next của tail thành NULL để ngắt kết nối với svToDelete.

Nếu tail đã là NULL sau khi cập nhật, tức là danh sách chỉ có một sinh viên và sau khi xóa thì danh sách trở thành rỗng, ta cũng cần cập nhật head thành NULL.

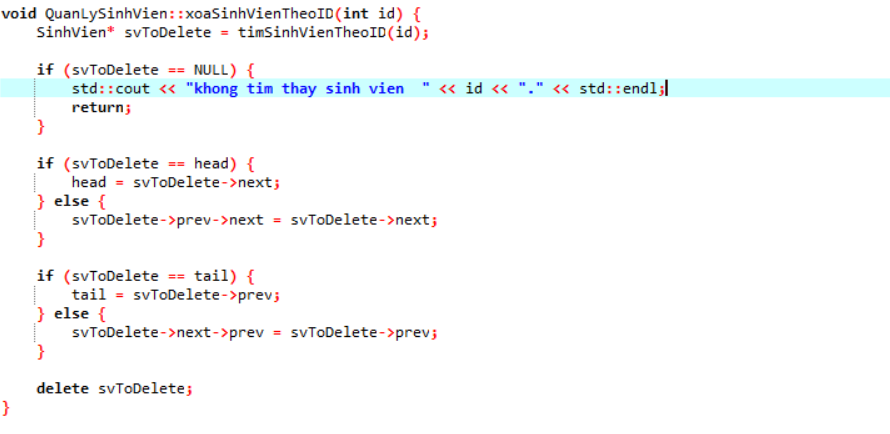
3. Xóa sinh viên và giải phóng bộ nhớ

delete svToDelete;

}

Cuối cùng, con trỏ svToDelete trỏ đến sinh viên cuối cùng trong danh sách được giải phóng bằng lệnh delete. Điều này giải phóng bộ nhớ đã cấp phát cho đối tượng SinhVien mà svToDelete đang trỏ đến.

**3.13. Xoa sinh viên theo ID**

****

Hình 8 xóa sinh viên theo ID

Đầu tiên, hàm gọi hàm timSinhVienTheoID(id) để tìm sinh viên trong danh sách có id như được chỉ định. Nếu không tìm thấy sinh viên (svToDelete == NULL), hàm sẽ in ra thông báo lỗi và kết thúc hàm.

Xử lý các trường hợp xóa

if (svToDelete == head) {

head = svToDelete->next;

} else {

svToDelete->prev->next = svToDelete->next;

}

if (svToDelete == tail) {

tail = svToDelete->prev;

} else {

svToDelete->next->prev = svToDelete->prev;

}

Sau khi tìm thấy sinh viên cần xóa (svToDelete), hàm tiến hành kiểm tra xem sinh viên này có phải là head (đầu danh sách) hay tail (cuối danh sách) hay không.

Nếu svToDelete là head, con trỏ head được cập nhật để trỏ tới sinh viên kế tiếp (head = svToDelete->next).

Nếu svToDelete không phải là head, con trỏ next của nút trước svToDelete được cập nhật để trỏ tới nút sau svToDelete (svToDelete->prev->next = svToDelete->next).

Tương tự, nếu svToDelete là tail, con trỏ tail được cập nhật để trỏ tới nút trước svToDelete (tail = svToDelete->prev).

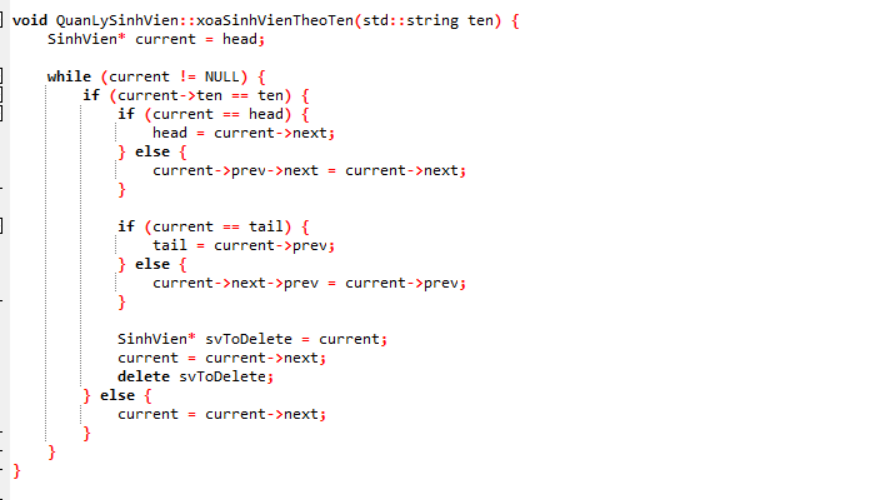
Nếu svToDelete không phải là tail, con trỏ prev của nút sau svToDelete được cập nhật để trỏ tới nút trước svToDelete (svToDelete->next->prev = svToDelete->prev).

Giải phóng bộ nhớ

delete svToDelete;

Cuối cùng, con trỏ svToDelete trỏ tới sinh viên cần xóa được giải phóng bằng lệnh delete, từ đó giải phóng bộ nhớ đã cấp phát cho đối tượng SinhVien này.

**3.14. Xoa sinh viên theo ten**



Hình 9 xóa sinh viên theo tên

SinhVien\* current = head;

while (current != NULL) {

if (current->ten == ten) {

// Xử lý xóa sinh viên khi tên trùng khớp

} else {

current = current->next;

}

}

Biến current được khởi tạo để duyệt từ đầu danh sách head đến cuối danh sách tail.

Trong vòng lặp while, kiểm tra điều kiện current != NULL để đảm bảo vẫn còn nút trong danh sách để duyệt.

2. Xử lý xóa sinh viên khi tên trùng khớp

if (current == head) {

head = current->next;

} else {

current->prev->next = current->next;

}

if (current == tail) {

tail = current->prev;

} else {

current->next->prev = current->prev;

}

SinhVien\* svToDelete = current;

current = current->next;

delete svToDelete;

Nếu tên của sinh viên hiện tại (current->ten) trùng khớp với ten được chỉ định:

Kiểm tra nếu current là head, thì cập nhật head để trỏ tới current->next.

Nếu current không phải là head, thì cập nhật next của nút trước current để trỏ tới current->next.

Kiểm tra nếu current là tail, thì cập nhật tail để trỏ tới current->prev.

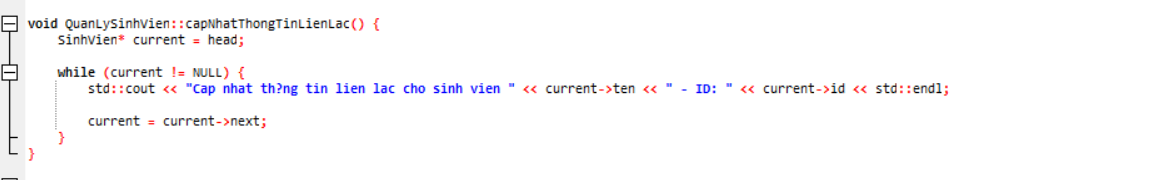
Nếu current không phải là tail, thì cập nhật prev của nút sau current để trỏ tới current->prev.

Lưu con trỏ current vào svToDelete để chuẩn bị giải phóng bộ nhớ.

Cập nhật current để tiếp tục duyệt danh sách.

Cuối cùng, sử dụng delete để giải phóng bộ nhớ của svToDelete.

**3.15. Cập nhập thông tin liên lạc**



Hình 10 Cập nhập thông tin liên lạc

Duyệt qua Danh sách Liên kết: Hàm bắt đầu bằng cách khởi tạo con trỏ current để duyệt qua danh sách bắt đầu từ head.

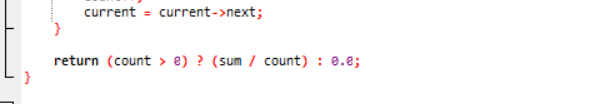
Hiển thị Thông tin: Trong vòng lặp, nó in ra một thông báo cho mỗi sinh viên cho biết đang cập nhật thông tin liên lạc. Nó sử dụng current->ten và current->id để lấy và hiển thị tên và mã số sinh viên.

Logic Bổ sung: Nếu bạn cần thực sự cập nhật thông tin liên lạc (như địa chỉ, số điện thoại, vv.), bạn sẽ thêm logic đó trong vòng lặp ở đoạn được chú thích. Điều này có thể bao gồm yêu cầu nhập từ người dùng, chỉnh sửa các thành viên dữ liệu của SinhVien, hoặc gọi các hàm/methods khác để thực hiện các cập nhật cụ thể.

Duyệt: Vòng lặp tiếp tục cho đến khi current trở thành NULL, điều này đồng nghĩa với việc đã duyệt hết danh sách liên kết.

**3.16. Sap xep sinh viên theo điểm trung binh**





Hình 11 Sắp xếp theo điểm trung bình

void QuanLySinhVien::sapXepTheoDiemTB() {

if (head == NULL || head == tail) {

return;

}

bool swapped;

SinhVien\* ptr1;

SinhVien\* lptr = NULL;

do {

swapped = false;

ptr1 = head;

while (ptr1->next != lptr) {

if (ptr1->diemTB < ptr1->next->diemTB) {

std::swap(ptr1->id, ptr1->next->id);

std::swap(ptr1->ten, ptr1->next->ten);

std::swap(ptr1->gioiTinh, ptr1->next->gioiTinh);

std::swap(ptr1->diemTB, ptr1->next->diemTB);

swapped = true;

}

ptr1 = ptr1->next;

}

lptr = ptr1;

} while (swapped);

}

Hàm này sắp xếp các sinh viên trong danh sách liên kết đôi theo điểm trung bình (diemTB) giảm dần.

Đầu tiên, nếu danh sách chỉ có một phần tử (head == tail hoặc danh sách rỗng head == NULL), hàm sẽ kết thúc ngay.

Sử dụng thuật toán sắp xếp nổi bọt (bubble sort) để thực hiện việc sắp xếp:

ptr1 là con trỏ duyệt từ đầu danh sách đến lptr.

lptr là con trỏ để giới hạn phần tử đã được sắp xếp.

Trong mỗi vòng lặp, so sánh diemTB của ptr1 và ptr1->next, nếu diemTB của ptr1 nhỏ hơn ptr1->next, hoán đổi thông tin của hai sinh viên này và đặt swapped = true.

Cứ tiếp tục lặp cho đến khi không còn phần tử nào để hoán đổi, tức là swapped vẫn là false.

Hàm tinhDiemTrungBinh()

float QuanLySinhVien::tinhDiemTrungBinh() {

float sum = 0.0;

int count = 0;

SinhVien\* current = head;

while (current != NULL) {

sum += current->diemTB;

count++;

current = current->next;

}

return (count > 0) ? (sum / count) : 0.0;

}

Hàm này tính điểm trung bình của tất cả sinh viên trong danh sách liên kết đôi.

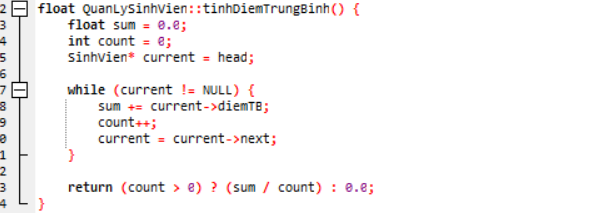
Khởi tạo sum là tổng điểm của các sinh viên và count là số lượng sinh viên.

Duyệt qua danh sách từ head đến khi current trỏ đến NULL.

Tại mỗi bước lặp, cộng thêm diemTB của current vào sum và tăng count lên 1.

Trả về giá trị (sum / count) nếu count > 0, ngược lại trả về 0.0.

**3.17. Tính điểm trung bình**

****

Hình 12 tính điểm trung bình

Khởi tạo biến:

sum: Biến để tính tổng điểm trung bình của tất cả sinh viên.

count: Biến để đếm số lượng sinh viên trong danh sách.

current: Con trỏ dùng để duyệt từ đầu danh sách đến khi NULL (hết danh sách).

Duyệt qua danh sách:

Sử dụng vòng lặp while để duyệt qua từng nút (SinhVien) trong danh sách liên kết đôi.

Mỗi lần lặp, cộng giá trị diemTB của sinh viên hiện tại vào sum.

Tăng biến count lên 1 để đếm số lượng sinh viên đã duyệt qua.

Dịch chuyển con trỏ current sang nút kế tiếp trong danh sách (current = current->next).

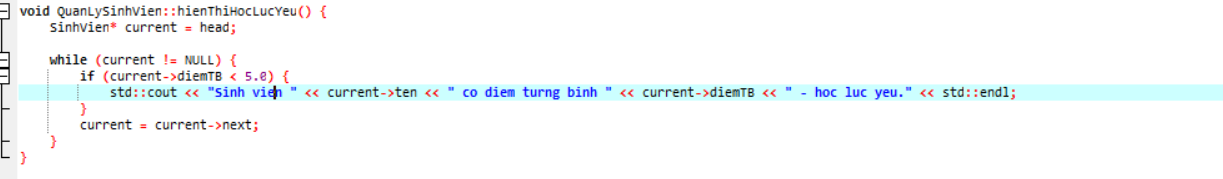
Trả về điểm trung bình:

Sau khi đã duyệt hết danh sách, kiểm tra count để xác định xem có sinh viên nào trong danh sách không.

Nếu count > 0, tức là danh sách có ít nhất một sinh viên, trả về giá trị sum / count, tức là điểm trung bình của danh sách sinh viên.

Nếu count == 0, tức là danh sách rỗng, trả về 0.0.

**3.18. Hiển thị sinh viên có học lực yếu**

****

Hình 13 hiển thị học lực

Khởi tạo biến:

current: Con trỏ dùng để duyệt từ đầu danh sách đến khi NULL (hết danh sách).

Duyệt qua danh sách:

Sử dụng vòng lặp while để duyệt qua từng nút (SinhVien) trong danh sách liên kết đôi.

Trong mỗi lần lặp, kiểm tra điều kiện: nếu current->diemTB < 5.0, tức là sinh viên có điểm trung bình dưới 5.0, thì hiển thị thông tin sinh viên này.

Sử dụng std::cout để in ra thông tin về sinh viên đó bao gồm tên và điểm trung bình.

Nếu không thỏa điều kiện, dịch chuyển con trỏ current sang sinh viên kế tiếp trong danh sách (current = current->next).

Khởi tạo con trỏ current: Bắt đầu từ đầu danh sách (list.head), current được sử dụng để duyệt qua từng nút trong danh sách.

Duyệt qua danh sách:

Sử dụng vòng lặp while để duyệt qua danh sách liên kết từ đầu đến cuối (current != NULL).

Kiểm tra giá trị tiền cước:

Trong mỗi vòng lặp, kiểm tra xem giá trị tienCuoc của nút hiện tại (current->data.tienCuoc) có nhỏ hơn value hay không.

Xử lý nút cần xóa:

Nếu nút hiện tại cần xóa (tức là current->data.tienCuoc < value):

Lưu con trỏ toDelete trỏ đến nút cần xóa.

Xác định trường hợp xóa:

Nếu current là nút đầu danh sách (current == list.head):

Cập nhật list.head thành current->next.

Nếu list.head không rỗng, gán list.head->prev bằng NULL.

Nếu list.head rỗng sau khi xóa, cập nhật list.tail thành NULL.

Nếu current là nút cuối danh sách (current == list.tail):

Cập nhật list.tail thành current->prev.

Gán list.tail->next bằng NULL.

Nếu current ở giữa danh sách:

Cập nhật liên kết của nút trước và nút sau current để bỏ qua current.

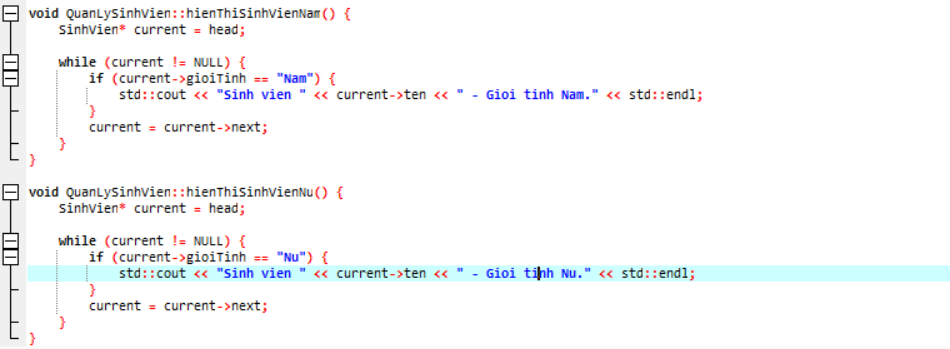
Xóa nút toDelete.

Di chuyển current đến nút kế tiếp của toDelete (nếu toDelete không phải là nút cuối cùng).

Di chuyển current: Nếu không xóa nút hiện tại, di chuyển current đến nút tiếp theo trong danh sách.

Thông báo kết quả: Sau khi hoàn thành vòng lặp, in ra thông báo cho biết đã xóa các nút có tienCuoc nhỏ hơn value.

**3.19 Hiện thị giới tinh sinh viên**



Hình 14 hiện thị giới tính

Khởi tạo biến:

current: Con trỏ dùng để duyệt từ đầu danh sách đến khi NULL (hết danh sách).

Duyệt qua danh sách:

Mỗi hàm sử dụng vòng lặp while để duyệt qua từng nút (SinhVien) trong danh sách liên kết đôi.

Trong mỗi lần lặp, kiểm tra điều kiện: current->gioiTinh == "Nam" (hoặc current->gioiTinh == "Nu").

Nếu điều kiện thỏa mãn, sử dụng std::cout để in ra thông tin của sinh viên đó kèm theo giới tính tương ứng.

Di chuyển con trỏ:

Sau khi kiểm tra, dịch chuyển con trỏ current sang sinh viên kế tiếp trong danh sách (current = current->next).

**3.20 Nhập danh sách từ tệp**

A screenshot of a computer code

Description automatically generated

Hình 15 nhập danh sách từ tệp

Mở tệp để đọc: Sử dụng ifstream để mở tệp văn bản với đường dẫn được cung cấp bởi filename.

Kiểm tra tệp có mở thành công không: Sử dụng if (inFile.is\_open()) để đảm bảo rằng tệp đã được mở thành công trước khi tiếp tục.

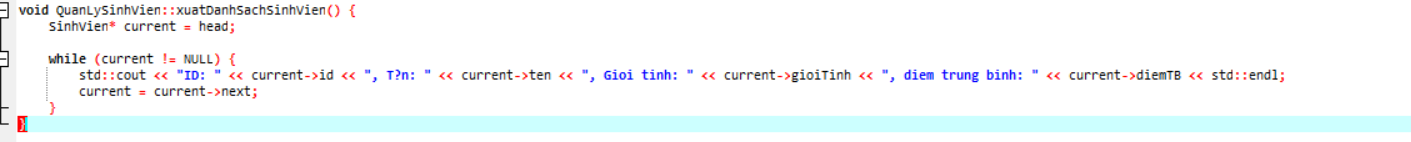
Đọc từng dòng từ tệp và chèn vào danh sách: Sử dụng một vòng lặp while để đọc từng dòng từ tệp văn bản. Trong mỗi lần lặp, dữ liệu được đọc và lưu vào biến b thuộc kiểu thueBao, sau đó sử dụng hàm insertLast(list, b) để chèn dữ liệu vào cuối danh sách.

Đóng tệp sau khi đọc xong: Sau khi đọc và chèn dữ liệu vào danh sách, sử dụng inFile.close() để đóng tệp.

Thông báo kết quả: Nếu đọc từ tệp thành công, in ra thông báo cho người dùng biết danh sách đã được đọc từ tệp với tên là filename.

Xử lý khi không thể mở tệp: Nếu không thể mở tệp để đọc (else trong if (inFile.is\_open())), in ra thông báo lỗi.

**3.21 xuất danh sách sinh viên**

****

Hình 16 xuất danh sách sinh viên

void QuanLySinhVien::xuatDanhSachSinhVien() {

SinhVien\* current = head;

// Duyệt qua danh sách liên kết đôi để xuất thông tin từng sinh viên

while (current != NULL) {

std::cout << "ID: " << current->id << ", Tên: " << current->ten << ", Giới tính: " << current->gioiTinh << ", Điểm trung bình: " << current->diemTB << std::endl;

current = current->next; // Chuyển sang sinh viên kế tiếp trong danh sách

}

}

Giải thích chi tiết:

Khởi tạo biến:

current: Con trỏ được sử dụng để duyệt từng nút (SinhVien) trong danh sách liên kết đôi, bắt đầu từ head.

Duyệt qua danh sách:

Hàm sử dụng vòng lặp while để duyệt qua từng nút trong danh sách liên kết đôi cho đến khi current trỏ đến NULL (hết danh sách).

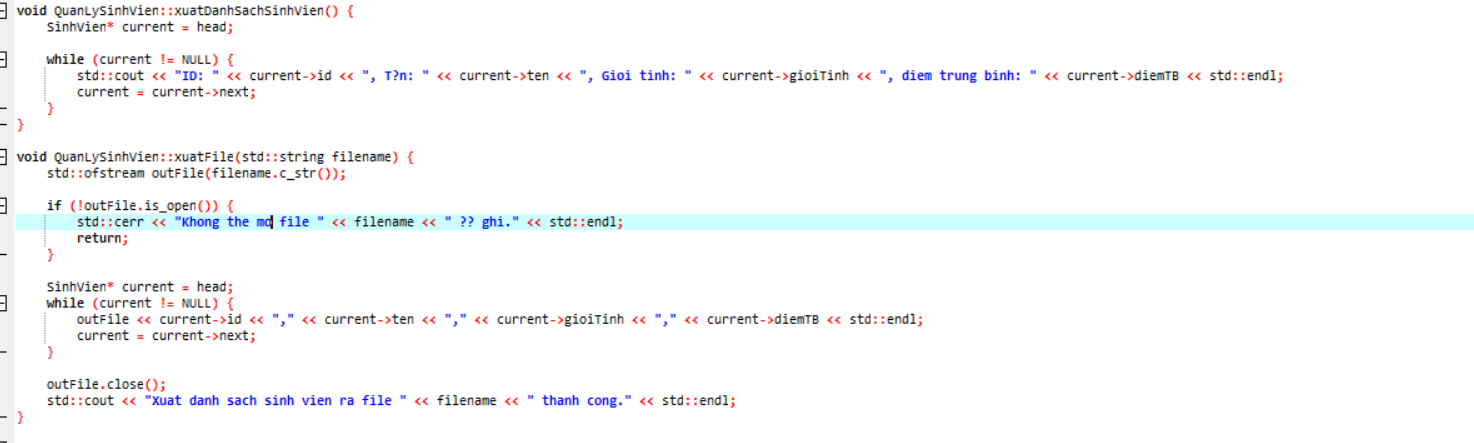
Trong mỗi lần lặp, sử dụng std::cout để in ra thông tin của sinh viên tại vị trí hiện tại của current.

Dữ liệu được in gồm có: ID, Tên, Giới tính, và Điểm trung bình.

Di chuyển con trỏ:

Sau khi in thông tin của sinh viên hiện tại, dịch chuyển con trỏ current sang sinh viên kế tiếp trong danh sách (current = current->next).

## **3.22 xuất file ( .txt)**



Hình 17: xuất file ( vào txt)

Mở tệp để ghi:

Sử dụng std::ofstream để mở tệp với tên filename để ghi dữ liệu (outFile(filename.c\_str())).

Nếu không thể mở được tệp, hàm sẽ in ra thông báo lỗi trên std::cerr và kết thúc hàm.

Ghi dữ liệu vào tệp:

Sau khi tệp được mở thành công, hàm sử dụng vòng lặp while để duyệt qua danh sách liên kết đôi.

Với mỗi sinh viên (current), hàm ghi thông tin id, ten, gioiTinh, và diemTB của sinh viên đó vào tệp outFile.

Mỗi sinh viên được ghi trên một dòng, với các thông tin ngăn cách bằng dấu phẩy.

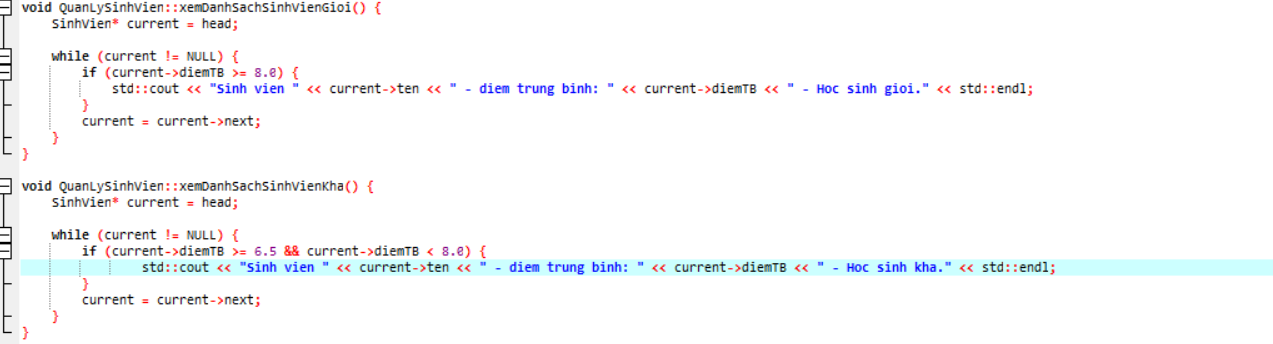
Đóng tệp sau khi ghi xong:

Sau khi hoàn thành việc ghi dữ liệu, hàm sử dụng outFile.close() để đóng tệp.

Thông báo thành công:

Cuối cùng, hàm in ra màn hình thông báo cho người dùng biết rằng quá trình xuất danh sách sinh viên ra tệp đã hoàn thành thành công.

## **3.23. Xem danh sách sinh viên giỏi / khá**



Hình 18 xem học sinh giỏi / khá

Hàm xemDanhSachSinhVienGioi()

Mô tả: Hàm này được thiết kế để hiển thị danh sách các sinh viên có học lực giỏi, tức là sinh viên có diemTB từ 8.0 trở lên.

Cách thức hoạt động:

Bắt đầu từ head, hàm duyệt từng nút của danh sách liên kết đôi.

Đối với mỗi sinh viên:

Kiểm tra điều kiện diemTB >= 8.0.

Nếu điều kiện thành công, in ra thông tin của sinh viên với dòng chữ "Hoc sinh gioi.".

Sau đó, chuyển sang sinh viên tiếp theo bằng cách di chuyển con trỏ current đến next.

Lưu ý: Hàm này chỉ duyệt qua danh sách một lần duy nhất và in ra thông tin trực tiếp trên màn hình.

Hàm xemDanhSachSinhVienKha()

Mô tả: Hàm này tương tự như hàm xemDanhSachSinhVienGioi(), nhưng được thiết kế để hiển thị danh sách sinh viên có học lực khá, tức là sinh viên có diemTB từ 6.5 đến dưới 8.0.

Cách thức hoạt động:

Bắt đầu từ head, hàm duyệt từng nút của danh sách liên kết đôi.

Đối với mỗi sinh viên:

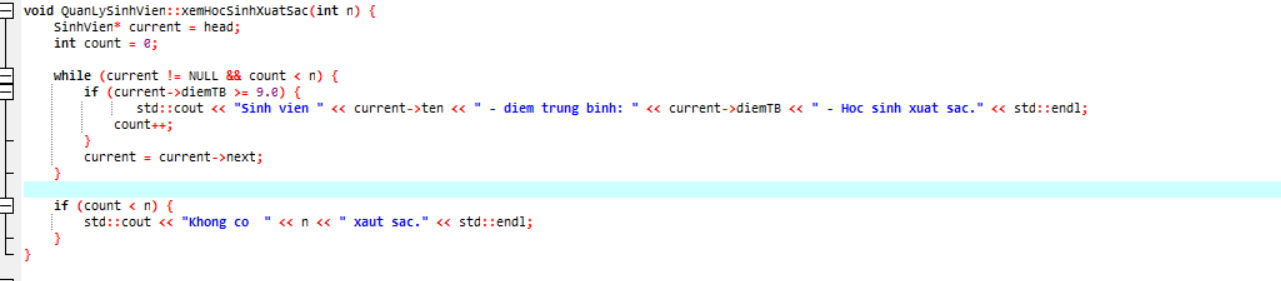
Kiểm tra điều kiện diemTB >= 6.5 && diemTB < 8.0.

Nếu điều kiện thành công, in ra thông tin của sinh viên với dòng chữ "Hoc sinh kha.".

Tiếp tục chuyển sang sinh viên tiếp theo bằng cách di chuyển con trỏ current đến next.

Lưu ý: Như hàm trước, hàm này cũng chỉ duyệt qua danh sách một lần duy nhất và in ra thông tin trực tiếp.

## **3.24. Xem danh sách sinh viên suất xắc**



Hình 19 Xem học sinh xuất sắc

Khởi tạo biến và con trỏ:

current là con trỏ dùng để duyệt qua danh sách các sinh viên từ head đến khi gặp NULL (hết danh sách) hoặc đã đủ n sinh viên xuất sắc.

count là biến đếm số lượng sinh viên xuất sắc đã được hiển thị.

Vòng lặp while:

Vòng lặp này sẽ tiếp tục cho đến khi duyệt hết danh sách sinh viên (current != NULL) hoặc đã hiển thị đủ n sinh viên xuất sắc (count < n).

Trong mỗi vòng lặp, kiểm tra điều kiện current->diemTB >= 9.0.

Nếu điều kiện đúng, in ra thông tin của sinh viên đó và tăng biến count lên 1.

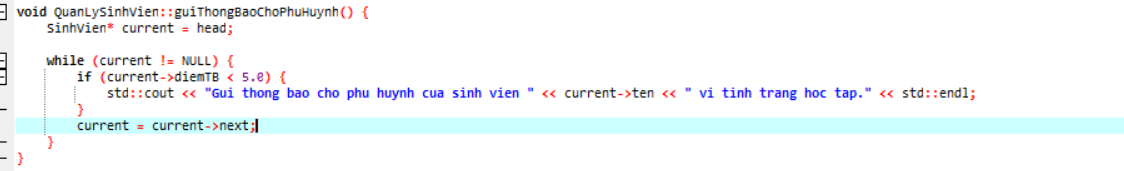
Tiếp tục di chuyển current sang sinh viên kế tiếp.

Kiểm tra số lượng sinh viên:

Sau khi kết thúc vòng lặp, kiểm tra xem count có đủ n sinh viên xuất sắc hay không.

Nếu count < n, in ra thông báo là "Không có n xuất sắc."

## **3.25. Gửi thông báo cho phụ huynh**



Hình 20 gửi thông báo phụ huynh

Khởi tạo biến và con trỏ:

current là con trỏ dùng để duyệt qua danh sách các sinh viên từ head đến khi gặp NULL (hết danh sách).

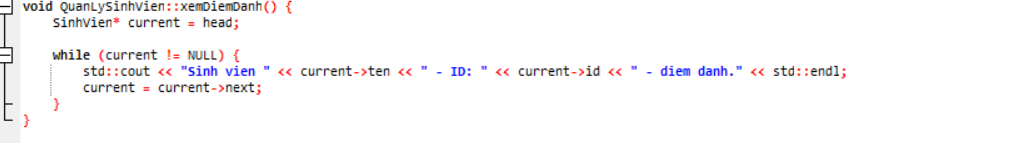
Vòng lặp while:

Vòng lặp này sẽ tiếp tục cho đến khi duyệt hết danh sách sinh viên (current != NULL).

Trong mỗi vòng lặp, kiểm tra điều kiện current->diemTB < 5.0.

Nếu điều kiện đúng, in ra thông báo gửi thư cho phụ huynh của sinh viên current->ten về tình trạng học tập của họ.

## **3.26. Xem điểm danh**



Hình 21 xem điểm danh

Khởi tạo biến và con trỏ:

current là con trỏ dùng để duyệt qua từng sinh viên trong danh sách từ head đến khi gặp NULL (hết danh sách).

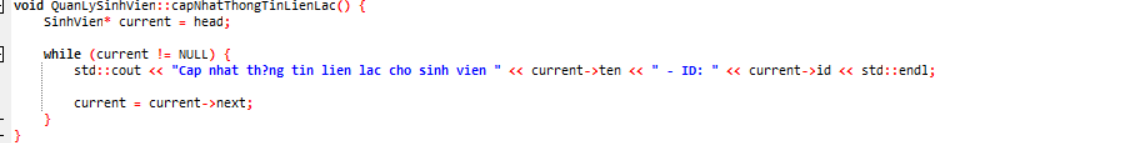
Vòng lặp while:

Vòng lặp này sẽ duyệt qua từng sinh viên trong danh sách (current != NULL).

Trong mỗi vòng lặp, in ra thông tin điểm danh của sinh viên current->ten và current->id.

Nội dung in ra sẽ có dạng "Sinh vien [tên sinh viên] - ID: [ID của sinh viên] - diem danh."

## **3.27. Cập nhập thông tin liên lạc**



Hình 22 cập nhập thông tin liên lạc

Khởi tạo biến và con trỏ:

current là con trỏ dùng để duyệt từng sinh viên trong danh sách, bắt đầu từ head.

Vòng lặp while:

Vòng lặp này duyệt qua danh sách các sinh viên (chạy cho đến khi current không còn trỏ đến sinh viên nào).

Trong mỗi lần lặp:

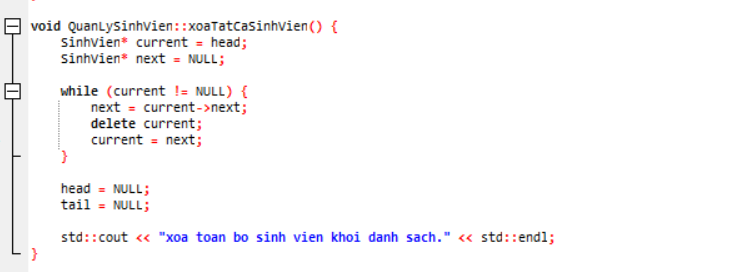
In ra thông tin cập nhật liên lạc cho sinh viên hiện tại (current->ten và current->id).

Sau khi in thông tin, cập nhật current là con trỏ tới sinh viên tiếp theo (current = current->next).

In ra thông báo hoàn thành:

Hàm không có thao tác thay đổi dữ liệu của sinh viên, chỉ đơn giản là hiển thị thông tin.

## **3.28. Xóa tất cả các sinh viên**



Hình 23 xóa tất cả sinh viên

Khởi tạo biến và con trỏ:

current là con trỏ dùng để duyệt từng sinh viên trong danh sách, bắt đầu từ head.

next là con trỏ để lưu giữ con trỏ tới sinh viên tiếp theo của current trước khi current bị xóa.

Vòng lặp while:

Vòng lặp này duyệt qua danh sách các sinh viên (chạy cho đến khi current không còn trỏ đến sinh viên nào).

Trong mỗi lần lặp:

Gán next là con trỏ tới sinh viên tiếp theo của current (next = current->next).

Xóa current khỏi danh sách bằng lệnh delete current.

Cập nhật current là con trỏ tới sinh viên tiếp theo (current = next).

Gán head và tail về NULL:

Sau khi vòng lặp kết thúc, cả head và tail đều được gán về NULL, đánh dấu rằng danh sách sinh viên đã rỗng hoàn toàn.

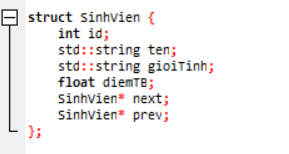
Thông báo hoàn thành:

Cuối cùng, in ra thông báo trên màn hình cho biết đã xóa toàn bộ sinh viên khỏi danh sách.

## **3.29. Thoát**

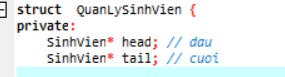
# **CHƯƠNG 4. MÃ NGUỒN CHƯƠNG TRÌNH**

## **4.1.Khai báo cấu trúc dữ liệu**



Hình 24 struct

thueBao: Lưu thông tin của mỗi sinh viên(số ID , tên, điểm).



Hình 25 con trỏ

Định nghĩa nút trong danh sách liên kết hai chiều.

## **4.2. Các hàm cơ bản**



Hình 26 hàm

void nhapDanhSachSinhVien()

Chức năng: Nhập thông tin danh sách sinh viên từ người dùng.

Thực hiện: Yêu cầu người dùng nhập thông tin của từng sinh viên và thêm vào danh sách.

void themSinhVienDauDanhSach()

Chức năng: Thêm một sinh viên vào đầu danh sách.

Thực hiện: Tạo một sinh viên mới và thêm vào phía đầu của danh sách hiện tại.

void themSinhVienCuoiDanhSach()

Chức năng: Thêm một sinh viên vào cuối danh sách.

Thực hiện: Tạo một sinh viên mới và thêm vào phía cuối của danh sách hiện tại.

void themSinhVienViTriBatKy()

Chức năng: Thêm một sinh viên vào vị trí bất kỳ trong danh sách.

Thực hiện: Yêu cầu người dùng nhập vị trí và thông tin sinh viên cần thêm, sau đó thêm vào danh sách tại vị trí đó.

void capNhapThongTinSinhVien(SinhVien sv)\*

Chức năng: Cập nhật thông tin của một sinh viên.

Thực hiện: Nhận đối tượng SinhVien cần cập nhật thông tin và cho phép người dùng chỉnh sửa thông tin của sinh viên đó.

void xoaSinhVienDauDanhSach()

Chức năng: Xóa sinh viên ở đầu danh sách.

Thực hiện: Xóa sinh viên đầu tiên trong danh sách hiện tại.

void xoaSinhVienCuoiDanhSach()

Chức năng: Xóa sinh viên ở cuối danh sách.

Thực hiện: Xóa sinh viên cuối cùng trong danh sách hiện tại.

void xoaSinhVienTheoID(int id)

Chức năng: Xóa sinh viên dựa trên ID cho trước.

Thực hiện: Tìm và xóa sinh viên có ID nhất định khỏi danh sách.

void capNhapID(int oldID, int newID)

Chức năng: Cập nhật ID của sinh viên từ oldID sang newID.

Thực hiện: Tìm sinh viên có oldID, sau đó cập nhật ID thành newID.

void xoaSinhVienTheoTen(std::string ten)

Chức năng: Xóa sinh viên dựa trên tên cho trước.

Thực hiện: Tìm và xóa tất cả các sinh viên có tên nhất định khỏi danh sách.

void sapXepTheoDiemTB()

Chức năng: Sắp xếp danh sách sinh viên theo điểm trung bình tăng dần.

Thực hiện: Sử dụng thuật toán sắp xếp để sắp xếp các sinh viên trong danh sách dựa trên điểm trung bình của họ.

float tinhDiemTrungBinh()

Chức năng: Tính điểm trung bình của tất cả sinh viên trong danh sách.

Thực hiện: Lặp qua từng sinh viên trong danh sách và tính toán điểm trung bình của từng sinh viên, sau đó trả về điểm trung bình tổng thể.

void hienThiHocLucYeu()

Chức năng: Hiển thị thông tin các sinh viên có học lực yếu.

Thực hiện: Duyệt danh sách sinh viên và hiển thị thông tin của các sinh viên có điểm trung bình dưới mức đủ điều kiện để qua môn.

void hienThiSinhVienNam()

Chức năng: Hiển thị thông tin các sinh viên là nam.

Thực hiện: Lọc và hiển thị thông tin của các sinh viên có giới tính nam từ danh sách.

void hienThiSinhVienNu()

Chức năng: Hiển thị thông tin các sinh viên là nữ.

Thực hiện: Lọc và hiển thị thông tin của các sinh viên có giới tính nữ từ danh sách.

void xuatDanhSachSinhVien()

Chức năng: In ra màn hình thông tin của tất cả sinh viên trong danh sách.

Thực hiện: Lặp qua danh sách sinh viên và in ra thông tin của từng sinh viên.

void xuatFile(std::string filename)

Chức năng: Xuất thông tin của tất cả sinh viên trong danh sách ra một file với tên filename.

Thực hiện: Mở file filename và ghi thông tin của từng sinh viên vào file đó.

void xemDanhSachSinhVienGioi()

Chức năng: Hiển thị danh sách các sinh viên xuất sắc.

Thực hiện: Lọc và hiển thị thông tin của các sinh viên có học lực xuất sắc từ danh sách.

void xemDanhSachSinhVienKha()

Chức năng: Hiển thị danh sách các sinh viên khá.

Thực hiện: Lọc và hiển thị thông tin của các sinh viên có học lực khá từ danh sách.

void xemHocSinhXuatSac(int n)

Chức năng: Hiển thị thông tin n sinh viên có điểm cao nhất.

Thực hiện: Sắp xếp danh sách sinh viên theo điểm trung bình giảm dần và hiển thị thông tin của n sinh viên có điểm cao nhất.

void hocSinhDuYeuCauLenLop()

Chức năng: Hiển thị danh sách sinh viên đủ điều kiện lên lớp.

Thực hiện: Lọc và hiển thị thông tin của các sinh viên có điểm trung bình đủ điều kiện để qua môn.

void danhSachHocSinhOLaiLop()

Chức năng: Hiển thị danh sách sinh viên cần học lại lớp.

Thực hiện: Lọc và hiển thị thông tin của các sinh viên không đạt điều kiện để qua môn.

void guiThongBaoChoPhuHuynh()

Chức năng: Gửi thông báo cho phụ huynh của các sinh viên có học lực yếu.

Thực hiện: Duyệt danh sách sinh viên, lọc ra các sinh viên có học lực yếu và gửi thông báo đến phụ huynh của họ.

void xemDiemDanh()

Chức năng: Xem bản điểm danh của tất cả sinh viên.

Thực hiện: Lặp qua danh sách sinh viên và hiển thị bản điểm danh của từng sinh viên.

void capNhatThongTinLienLac()

Chức năng: Cập nhật thông tin liên lạc cho từng sinh viên trong danh sách.

Thực hiện: Duyệt qua danh sách sinh viên và cập nhật thông tin liên lạc của từng sinh viên.

void xoaTatCaSinhVien()

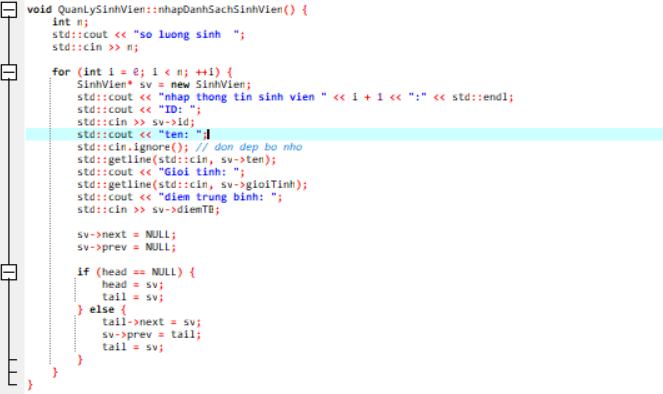
Chức năng: Xóa tất cả sinh viên trong danh sách.

Thực hiện: Giải phóng bộ nhớ của từng nút trong danh sách liên kết, để danh sách trở thành rỗng.

void menu()

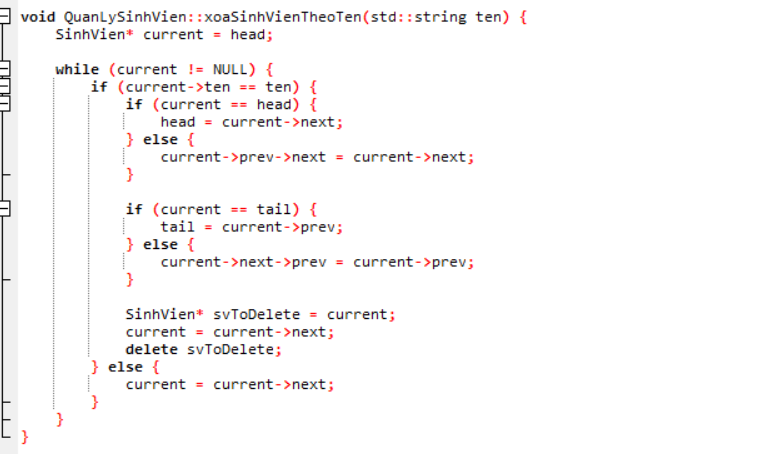
Chức năng: Hiển thị menu các lựa chọn cho người dùng.

**4.3. Thêm, sửa, xóa thông tin sinh viên.**



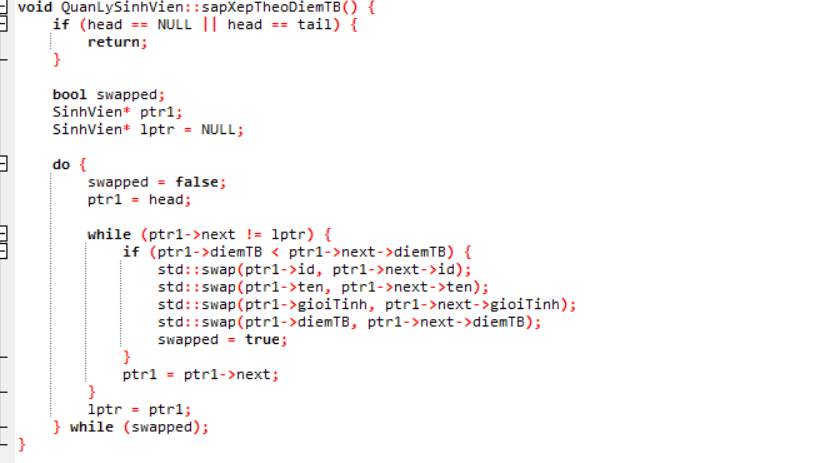
Hình 27 thêm , sửa

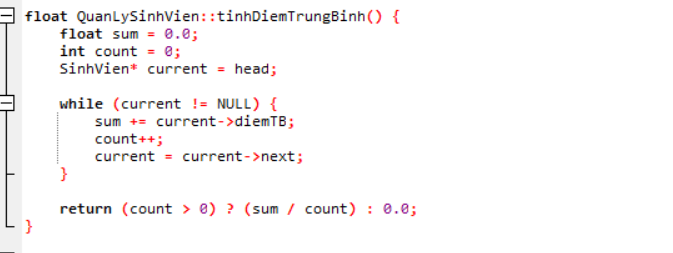
**4.3.2. Xóa sinh viên theo tên**



Hình 28 xóa sinh viên

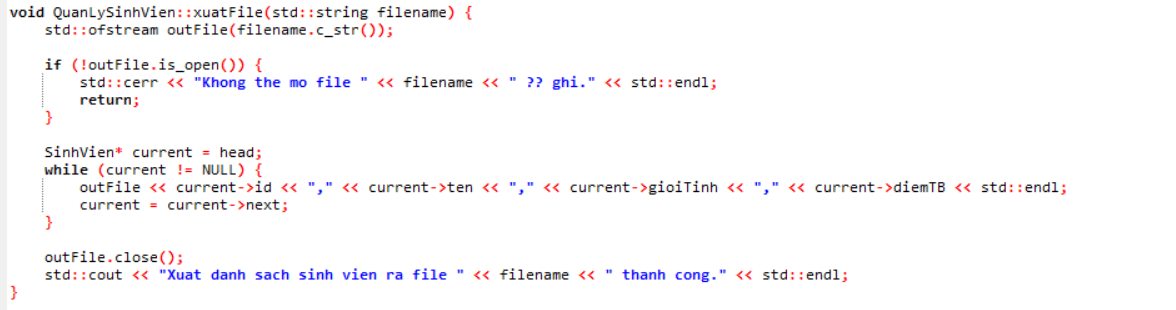
## **4.3. Sắp xếp danh sách theo điểm trung bình và tính điểm trung bình**





Hình 29 sắp xếp và tính điểm

**4.4. Các chức năng tương tác với file**



Hình 30 xuất file vào txt

std::ofstream là lớp để ghi vào file.

filename.c\_str() chuyển đổi chuỗi filename thành một con trỏ const char\* (loại dữ liệu mà hàm mở file yêu cầu).

outFile.is\_open() kiểm tra xem file đã mở thành công hay chưa.

Nếu không mở được (!outFile.is\_open()), thông báo lỗi trên std::cerr và kết thúc hàm.

Duyệt qua danh sách các sinh viên từ head đến khi current là NULL.

outFile << current->id << "," << current->ten << "," << current->gioiTinh << "," << current->diemTB << std::endl; ghi thông tin của từng sinh viên vào file, mỗi thông tin cách nhau bằng dấu phẩy và kết thúc dòng bằng std::endl (xuống dòng).

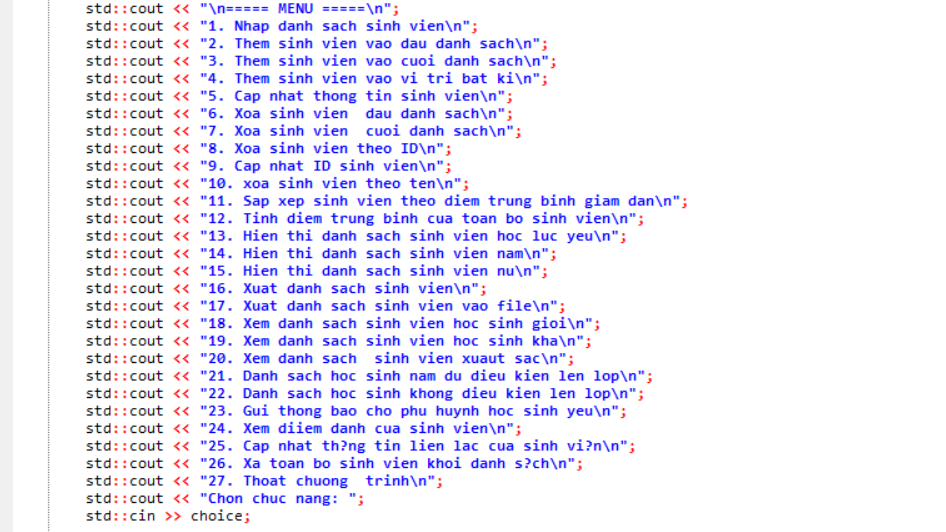
outFile.close() đóng file sau khi ghi xong.

std::cout << "Xuat danh sach sinh vien ra file " << filename << " thanh cong." << std::endl; thông báo ra màn hình rằng việc xuất danh sách sinh viên ra file đã thành công.

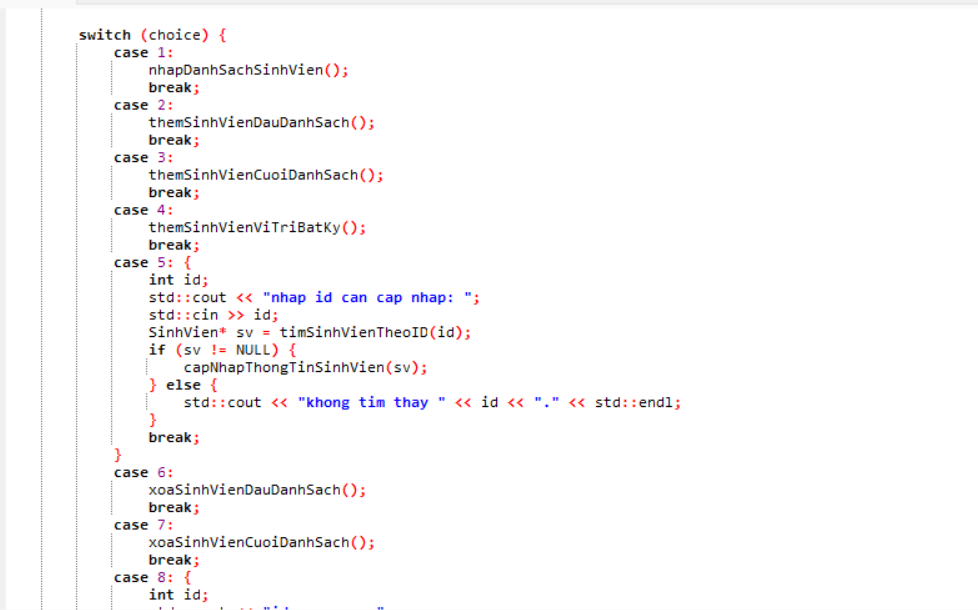
## **4.5. Chương trình chính (main function)**

Hiển thị menu lựa chọn các chức năng.

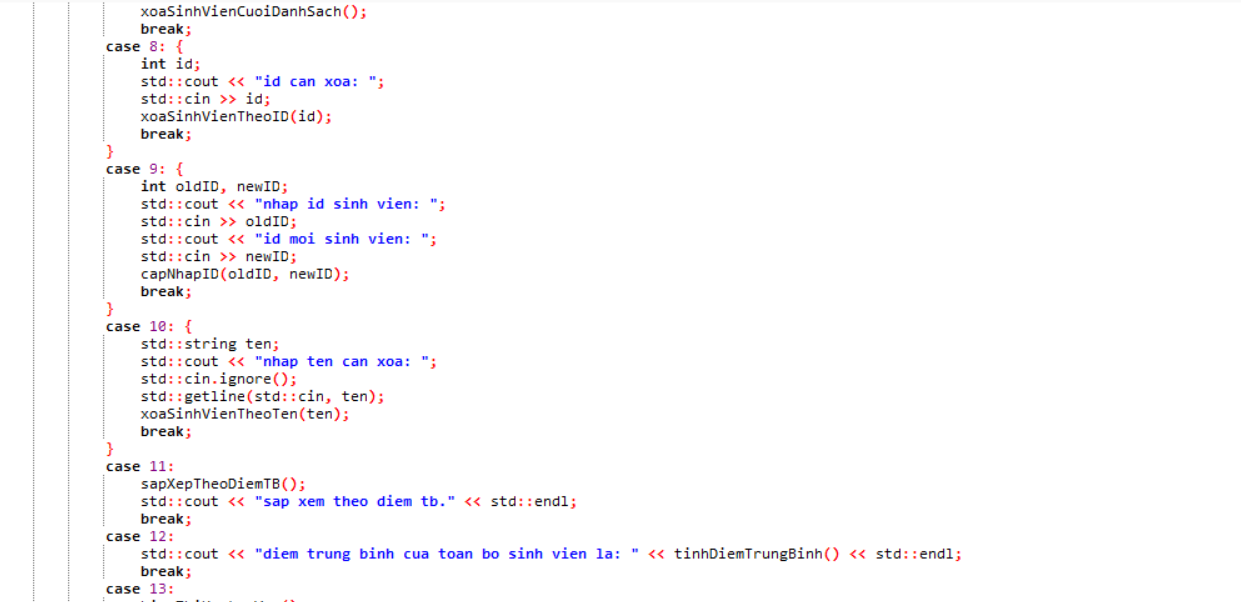
Gọi các hàm xử lý dữ liệu tương ứng với lựa chọn của người dùng.



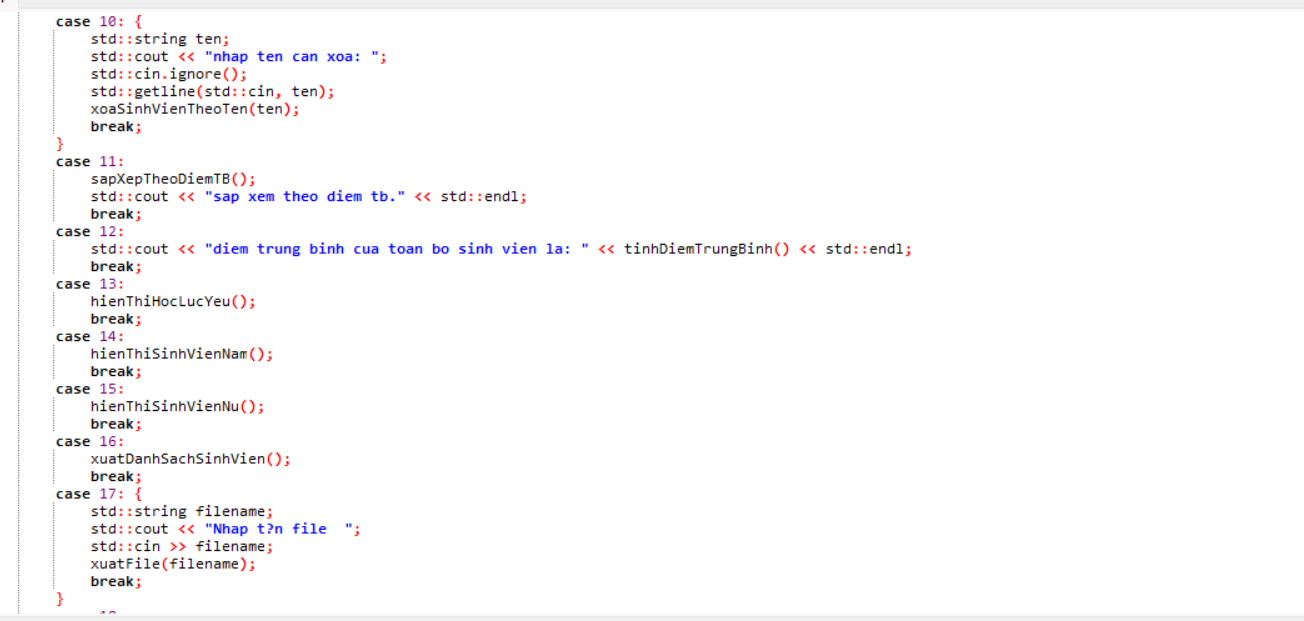
Hình 31 Hàm main 1



Hình 32 Hàm main 2



Hình 33 Hàm main 3



Hình 34 Hàm main 4



Hình 35 Hàm main 5

## **KẾT LUẬN**

Mô tả bài toán và mục tiêu:

Bài toán của chúng ta là thiết kế và triển khai một hệ thống quản lý sinh viên.

Mục tiêu là tạo ra một ứng dụng cho phép quản lý danh sách sinh viên, bao gồm thêm, sửa, xóa thông tin sinh viên, cũng như thực hiện các thao tác phân loại và thống kê.

Các chức năng chính của ứng dụng:

Thêm, sửa, xóa sinh viên: Cho phép người dùng thêm sinh viên mới, cập nhật thông tin của sinh viên và xoá sinh viên khỏi danh sách.

Tìm kiếm và sắp xếp: Các chức năng cho phép tìm kiếm sinh viên theo tên, ID, giới tính và sắp xếp theo điểm trung bình.

Phân loại và hiển thị: Hiển thị danh sách sinh viên theo từng loại học lực, giới tính và các yêu cầu khác của người dùng.

Xuất và lưu trữ: Có khả năng lưu thông tin sinh viên vào tập tin để sau này có thể xuất ra hoặc nhập lại từ tập tin.

Công nghệ và kiến trúc sử dụng:

Sử dụng ngôn ngữ lập trình C++ để xây dựng ứng dụng.

Áp dụng cấu trúc dữ liệu danh sách liên kết đôi để quản lý danh sách sinh viên.

Sử dụng các phương pháp và kỹ thuật lập trình như đa hình, kế thừa, và xử lý ngoại lệ để tối ưu hóa và bảo vệ dữ liệu.

Kết quả và hạn chế:

Kết quả đạt được là một hệ thống quản lý sinh viên hoàn chỉnh, hỗ trợ nhiều chức năng và dễ sử dụng.

Những hạn chế có thể gặp phải bao gồm hiệu suất thấp khi làm việc với số lượng lớn sinh viên hoặc hạn chế về giao diện người dùng nếu không được thiết kế một cách tối ưu.

Kế hoạch phát triển và cải tiến:

Đề xuất phát triển thêm tính năng như hệ thống xếp lớp tự động, tương tác với học sinh và phụ huynh qua giao diện đồ họa.

Đưa ra kế hoạch nâng cấp hệ thống để hỗ trợ các nhu cầu mới của người dùng và cải thiện hiệu suất.

Tổng kết:

Bài toán quản lý sinh viên không chỉ giới hạn trong việc lưu trữ thông tin mà còn đặt ra nhiều thách thức về quản lý, tối ưu hóa và mở rộng ứng dụng.

Qua bài tập này, tôi đã có cơ hội áp dụng và nâng cao kỹ năng lập trình và quản lý dữ liệu, từ đó củng cố kiến thức về cấu trúc dữ liệu và giải thuật.

**KẾT LUẬN**

Sau khi đã cùng nhau tìm hiểu về cấu trúc dữ liệu và giải thuật, chúng ta có thể thấy rằng đây là những yếu tố nền tảng trong lĩnh vực khoa học máy tính và lập trình. Việc hiểu rõ và áp dụng đúng cấu trúc dữ liệu và giải thuật sẽ giúp chúng ta giải quyết các vấn đề một cách hiệu quả, tối ưu hóa tài nguyên và nâng cao hiệu suất của các chương trình.

Tóm tắt những điểm chính:

1. **Cấu trúc dữ liệu**: Là cách tổ chức và quản lý dữ liệu trong bộ nhớ. Các cấu trúc dữ liệu khác nhau có thể phù hợp với các tình huống khác nhau, và việc lựa chọn cấu trúc dữ liệu phù hợp là quan trọng để đạt được hiệu suất tốt.
2. **Giải thuật**: Là các quy trình hoặc tập hợp các bước để giải quyết vấn đề. Chúng ta đã thảo luận về các loại giải thuật phổ biến và cách chúng giúp tối ưu hóa quá trình xử lý dữ liệu.
3. **Tối ưu hóa**: Sự kết hợp giữa cấu trúc dữ liệu và giải thuật phù hợp sẽ giúp tối ưu hóa việc xử lý dữ liệu, giảm thiểu thời gian và tài nguyên sử dụng, từ đó cải thiện hiệu suất và khả năng mở rộng của ứng dụng.

Lợi ích:

* **Hiểu biết sâu rộng**: Nắm vững cấu trúc dữ liệu và giải thuật giúp bạn hiểu rõ cách các hệ thống máy tính và phần mềm hoạt động ở mức độ cơ bản.
* **Nâng cao kỹ năng lập trình**: Khả năng lựa chọn và triển khai cấu trúc dữ liệu và giải thuật phù hợp giúp bạn viết mã hiệu quả hơn, dễ bảo trì và mở rộng.
* **Giải quyết vấn đề tốt hơn**: Kỹ năng phân tích và áp dụng các giải thuật khác nhau giúp bạn giải quyết các vấn đề phức tạp một cách sáng tạo và hiệu quả.

Hy vọng rằng sau khóa học này, các bạn đã có được những kiến thức và kỹ năng cần thiết để áp dụng cấu trúc dữ liệu và giải thuật vào các dự án thực tế. Chúng ta đã đi qua một chặng đường dài, từ những khái niệm cơ bản đến những kỹ thuật nâng cao, và giờ đây các bạn đã sẵn sàng để tiếp tục khám phá và phát triển trong lĩnh vực này.

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Nguyễn Hồng Sơn (2007), *Giáo trình hệ thống Mạng máy tính CCNA* (Semester 1), NXB Lao động xã hội.
2. Phạm Quốc Hùng (2017), *Đề cương bài giảng Mạng máy tính*, Đại học SPKT Hưng Yên.
3. James F. Kurose and Keith W. Ross (2013), *Computer Networking: A top-down approach sixth Edition*, Pearson Education.