

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

# BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI: QUẢN LÍ KÍ TÚC XÁ SINH VIÊN

Giảng viên hướng dẫn: TRẦN THỊ DUNG

Sinh viên thực hiện: TRẦN PHAN PHÚ GIANG

HỒ ANH TUẤN ĐỖ VĂN TÙNG

Lớp : **KỸ THUẬT ĐIỆN TỬ VIỄN THÔNG**

Khoá : 64



**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

# BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI: QUẢN LÍ KÍ TÚC XÁ SINH VIÊN

Giảng viên hướng dẫn: TRẦN THỊ DUNG

Sinh viên thực hiện: TRẦN PHAN PHÚ GIANG

HỒ ANH TUẤN ĐỖ VĂN TÙNG

Lớp : **KỸ THUẬT ĐIỆN TỬ VIỄN THÔNG**

Khoá : 64

# LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn đến quý thầy cô trong bộ môn công nghệ thông tin đã tận tình giảng dạy và hỗ trợ chúng em trong suốt quá trình học tập cũng như trong suốt quá trình hoàn thiện bài tập lớn. Những kiến thức mà thầy cô truyền đạt là nền tảng vững chắc để chúng em có thể triển khai những ý tưởng cho đề tài của mình và nó còn giúp chúng em hiểu biết thêm về lĩnh vực mình đang theo học cũng như kiến thức để sau này áp dụng vào thực tiễn. Những nhận xét và góp từ thầy cô đã giúp chúng em cải thiện bản thân và khắc phục những thiếu sót trong quá trình hoàn thành đề tài.

Chúng em cũng xin gửi lời cảm ơn chân thành đến các bạn trong lớp vì đã luôn sẵn sàng chia sẻ ý tưởng, cũng như hỗ trợ và giúp đỡ chúng em nhiệt tình trong suốt quá trình thực hiện. Sự đồng hành và đóng góp của các bạn đã giúp chúng em có thêm sức mạnh khi đối mặt với những khó khăn trong quá trình hoàn thành đề tài. Cùng nhau, chúng em đã tìm ra những giải pháp tối ưu, hiệu quả để hoàn thành một bài tập lớn có phiên bản hoàn hảo và đầy đủ nhất.

Bên cạnh đó, chúng em cũng xin gửi lời cảm ơn đặc biệt đến gia đình vì đã luôn ở bên cạnh, động viên và tạo mọi điều kiện để có thể giúp chúng em phát triển bản thân. Sự thấu hiểu và ủng hộ của gia đình là động lực để chúng em có thể đương đầu với những khó khăn, thử thách trong quá trình học tập và sự nghiệp tương lai.

Cuối cùng, chúng em xin bày tỏ lòng biết ơn tới tất cả mọi người đã hỗ trợ, đóng góp và chia sẻ kinh nghiệm với chúng em. Những sự giúp đỡ ấy là một phần sức mạnh giúp chúng em có thể hoàn thành được đề tài của mình!

Chúng em hi vọng đề tài này không chỉ là một thành quả trong học tập mà còn là một bước tiến trong quá trình phát triển bản thân, chuẩn bị cho những cơ hội và thử thách trong con đường bước đến thành công trong tương lai.

Trân trọng!

# MỤC LỤC

[LỜI CẢM ƠN 1](#_bookmark0)

[CHƯƠNG 1: CƠ SỞ LÍ THUYẾT 6](#_bookmark1)

* 1. [Giới thiệu hệ thống quản lí ký túc xá: 6](#_bookmark2)
  2. [Một số khái niệm trong quản lí dữ liệu 6](#_bookmark3)
     1. [Cấu trúc dữ liệu và giải thuật là gì? 6](#_bookmark4)
     2. [Cấu trúc dữ liệu (Data Structures) 6](#_bookmark5)
     3. [Giải thuật ( Algorithms) 7](#_bookmark6)
     4. [Cấu trúc dữ liệu và giải thuật có mối quan hệ gắn kết 7](#_bookmark7)
     5. [Một số khái niệm khác 7](#_bookmark8)
     6. [Phân tích cấu trúc dữ liệu 8](#_bookmark9)
     7. [Thuật toán tạo và quản lý danh sách liên kết đơn 9](#_bookmark10)
     8. [Các thuật toán tìm kiếm tuần tự 9](#_bookmark11)
  3. [Thuật toán kiểm tra và xử lý đầu vào: 10](#_bookmark12)
     1. [Thuật toán kiểm tra và xử lý đầu vào 10](#_bookmark13)

[CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT CHƯƠNG TRÌNH 11](#_bookmark14)

* 1. [Hàm nhập ngày, tháng, năm sinh cho sinh viên: 11](#_bookmark15)
     1. [Khai báo thư viện 11](#_bookmark16)
     2. [Khai báo hàm: Ngày/Tháng/Năm 11](#_bookmark17)
     3. [Khai báo biến:Ngày/Tháng/Năm 11](#_bookmark18)
     4. [Kiểm tra điều kiện hợp lệ 12](#_bookmark19)
     5. [Hiển thi thông báo lỗi 12](#_bookmark20)
     6. [Gọi hàm đệ quy 12](#_bookmark21)
     7. [Định dạng ngày tháng năm thành chuỗi: 12](#_bookmark22)
     8. [Trả về chuỗi kết quả 13](#_bookmark23)
     9. [Khai báo hàm 13](#_bookmark24)
     10. [Sử dụng vòng lặp for 13](#_bookmark25)
  2. [Hàm thêm sinh viên tại vị trí được chỉ định: 14](#_bookmark26)
     1. [Khai báo hàm có tên "ThemSinhVIenTaiViTri": 14](#_bookmark27)
     2. [Tạo Nodecho sinh viên: 14](#_bookmark28)
     3. [Kiểm tra vị trí để thêm sinh viên 15](#_bookmark29)
  3. [Hàm xóa sinh viên tại vị trí được chỉ định 16](#_bookmark30)
     1. [Khai báo hàm đặt tên là 'XoaSinhVienTaiViTri' 16](#_bookmark31)
     2. [Kiểm tra xem danh sách có đang trống hay không 17](#_bookmark32)
     3. [Sử dụng vòng lặp if di chuyển con trỏ đến vị trí cần xóa 17](#_bookmark33)
  4. [Hàm thêm phòng mới vào danh sách liên kết đơn của các phòng 18](#_bookmark34)
     1. [Khai báo hàm đặt tên là: 'ThemPhong' sử dụng kiểu trả về void: 18](#_bookmark35)
     2. [Tạo một Node mới là (Phong\* Nodemoi) là bản sao của phòng 'Phongmoi':](#_bookmark36)

[. 19](#_bookmark36)

* 1. [Hàm nhập thông tin sinh viên 19](#_bookmark37)
     1. [Khai báo hàm 19](#_bookmark38)
     2. [Sử dụng vòng lặp while để nhập mã sinh viên: 20](#_bookmark39)
     3. [Kiểm tra xem Mã số sinh viên có tồn tại hay không 20](#_bookmark40)
     4. [Kiểm tra độ dài và định dạng của MSSV 21](#_bookmark41)
     5. [Kiểm tra điều kiện hợp lệ: 21](#_bookmark42)
     6. [Khai báo nhập họ và tên 22](#_bookmark43)
     7. [Khai báo nhập lớp 22](#_bookmark44)
     8. [Khai báo hàm nhập quê quán 23](#_bookmark45)
     9. [Khai báo hàm nhập học lực 23](#_bookmark46)
     10. [Khai báo hàm nhập ngày bắt đầu 23](#_bookmark47)
     11. [Khai báo hàm nhập hoàn cảnh 24](#_bookmark48)
     12. [Khai báo hàm kiểm tra miễn giảm học phí 24](#_bookmark49)
  2. [Hàm nhập các thông tin về phòng ở: 24](#_bookmark50)
     1. [Khai báo hàm: 24](#_bookmark51)
     2. [In ra thông báo nhập phòng 25](#_bookmark52)
     3. [Kiểm tra ID phòng có trùng hay không 25](#_bookmark53)
  3. [Hàm kiểm tra ID phòng có tồn tại trong danh sách hay không ? 25](#_bookmark54)
     1. [Khởi tạo hàm 25](#_bookmark55)
     2. [Sử dụng vòng lặp while để duyệt qua từng phần tử trong danh sách 26](#_bookmark56)
  4. [Khai báo các hàm thông tin về cơ sơ vật chất trong phòng ở: 27](#_bookmark57)
     1. [Khai báo biến: 27](#_bookmark58)
  5. [Khai bao hàm nhập và tính toán số ký điện 28](#_bookmark59)
     1. [Khai báo biến kí điện và các hàm tính toán: 28](#_bookmark60)
     2. [Sử dụng vòng lặp để kiểm tra dữ liệu đầu vào: 28](#_bookmark61)
     3. [Kiểm tra dữ liệu 28](#_bookmark62)
     4. [Kiểm tra giá trị hợp lý chưa 28](#_bookmark63)
     5. [Gán giá trị điện vào biến lưu trữ 29](#_bookmark64)
  6. [Khai bao hàm nhập và tính toán số khối nước 29](#_bookmark65)
  7. [Tính toán tổng chi phí điện nước của phòng 29](#_bookmark66)
  8. [Hàm khai báo chức năng xử lý thông tin giải thưởng khi đạt đủ các tiêu chí](#_bookmark67) [sạch, đẹp, an toàn. 29](#_bookmark67)
     1. [Khai báo biến 29](#_bookmark68)
     2. [Khởi tạo vòng lặp while: 30](#_bookmark69)
     3. [Kiểm tra chỉ số độ dài của đầu vào khi người dùng nhập vào 30](#_bookmark70)
     4. [Hàm kiểm tra số tiêu chí đạt được 32](#_bookmark71)
  9. [Hàm in danh sách sinh viên ra màn hình 34](#_bookmark72)
     1. [Khai báo hàm chức năng: 34](#_bookmark73)
     2. [Xây dựng tiêu đề cột 34](#_bookmark74)
     3. [In thông tin sinh viên 35](#_bookmark75)
  10. [Hàm in danh sách phòng theo dạng bảng 35](#_bookmark76)
      1. [Khai báo hàm chức năng: 35](#_bookmark77)
      2. [In tiêu đề bảng và dòng phân cách 36](#_bookmark78)
      3. [Tính số lượng sinh viên trong phòng 36](#_bookmark79)
      4. [In thông tin từng phòng 37](#_bookmark80)
  11. [Hàm tìm sinh viên trong ký túc xá theo mã số sinh viên 37](#_bookmark81)
      1. [Khai báo hàm chức năng 37](#_bookmark82)
  12. [Hàm in danh sách sinh viên được miễn giảm tiền ký túc xá: 38](#_bookmark83)
      1. [Khai báo hàm chức năng: 38](#_bookmark84)
  13. [Hàm in ra danh sách sinh viên của tất cả phòng: 39](#_bookmark85)
  14. [Hàm in ra danh sách các sinh viên được miễn giảm của tất cả các phòng 40](#_bookmark86)
      1. [Khai báo hàm chức năng 40](#_bookmark87)
  15. [Hoạt động của chương trình 42](#_bookmark88)
      1. [Khởi tạo 42](#_bookmark89)
      2. [Vòng lặp menu 42](#_bookmark90)
      3. [Giải thích chi tiết từng lựa chọn 43](#_bookmark91)
  16. [Nhập sinh viên tại vị trí bất kì. 45](#_bookmark92)
      1. [Tìm kiếm phòng 45](#_bookmark93)
      2. [Nhập thông tin sinh viên mới 45](#_bookmark94)
      3. [Chèn sinh viên vào vị trí 46](#_bookmark95)
  17. [Xóa sinh viên tại vị trí bất kì 47](#_bookmark96)
      1. [Tìm kiếm phòng 47](#_bookmark97)
      2. [Nhập vị trí cần xóa 47](#_bookmark98)
      3. [Xóa sinh viên tại vị trí 47](#_bookmark99)
  18. [In danh sách sinh viên trong phòng 48](#_bookmark100)
      1. [Tìm kiếm phòng: 48](#_bookmark101)
      2. [In danh sách sinh viên 48](#_bookmark102)
  19. [In danh sách các phòng 48](#_bookmark103)
  20. [Tìm sinh viên cần tìm theo MSSV 49](#_bookmark104)
      1. [Nhập MSSV: 49](#_bookmark105)
      2. [Tìm kiếm sinh viên: 49](#_bookmark106)
  21. [In danh sách sinh viên được miễn giảm 50](#_bookmark107)
      1. [Tìm kiếm phòng 50](#_bookmark108)
  22. [In danh sách sinh viên của tất cả các phòng 50](#_bookmark109)
  23. [In danh sách sinh viên được miễn giảm ở tất cả các phòng 51](#_bookmark110)
  24. [Xóa phòng 51](#_bookmark111)

[Dòng 1: string IdPhong; 51](#_bookmark112)

[Dòng 2: cout << "Nhap ID phong can xoa: "; 52](#_bookmark113)

[Dòng 3: cin >> IdPhong; 52](#_bookmark114)

[Dòng 4: XoaPhong(DanhSachPhong, IdPhong); 52](#_bookmark115)

[Dòng 5: break; 52](#_bookmark116)

[CHƯƠNG 3: GIAO DIỆN CHƯƠNG TRÌNH VÀ CÁCH SỬ DỤNG 53](#_bookmark117)

* 1. [Kết quả đạt được. 53](#_bookmark118)
     1. [Chức năng nhập Phòng 53](#_bookmark119)
     2. [Chức năng thêm sinh viên vào phòng đã tạo 55](#_bookmark120)
     3. [Chức năng in danh sách Phòng 58](#_bookmark121)
     4. [Hàm in ra danh danh sách tất cả các sinh viên của KTX 58](#_bookmark122)
     5. [In danh sách sinh viên trong một phòng 58](#_bookmark123)
     6. [Chức năng thêm sinh viên vào vị trí bất kì. 59](#_bookmark124)
     7. [Chức năng xóa sinh viên tại vị trí bất kì. 60](#_bookmark125)
     8. [Chức năng tìm sinh viên theo mã số sinh viên. 60](#_bookmark126)
     9. [Chức năng in danh sách sinh viên được miễn giảm ở một phòng 61](#_bookmark127)
     10. [Chức năng in danh sách tất cả sinh viên được miễn giảm tiền phòng 61](#_bookmark128)
     11. [Chức năng xóa phòng 62](#_bookmark129)

[TÀI LIỆU THAM KHẢO 63](#_bookmark130)

# CHƯƠNG 1: CƠ SỞ LÍ THUYẾT

* 1. Giới thiệu hệ thống quản lí ký túc xá:

Thời đại ngày nay con người rất chú trọng đến viêc học tập và bồi bổ kiến thức, vì vậy số lượng sinh viên của các trường đại học ngày càng nhiều. Đa phần là sinh viên từ các tỉnh lẻ nên nhu cầu về chỗ ở là một vấn đề cấp thiết. Sinh viên tỉnh ở các trường thường hay xin vào ở ký túc xá do ở ký túc xá có an ninh tốt, tiền phòng không quá đắt và có mô hình học tập sinh hoạt tốt. Vì vậy việc cần một chương trình quản lý ký túc xá là điều cực kỳ quan trọng trong việc xử lý các thông tin, số liệu của từng cá nhân sinh viên khi sống và ở tại ký túc xá. Với các chức năng như : thêm phòng, thêm sinh viên, in ra danh sách thông tin của sinh viên ở các phòng, hay dựa vào các tiêu chí đánh giá để xếp hạng chất lượng phòng ở của các sinh viên,...v..v..

* 1. Một số khái niệm trong quản lí dữ liệu
     1. Cấu trúc dữ liệu và giải thuật là gì?

*Cấu trúc dữ liệu và giải thuật* là một trong những môn học cơ bản và quan trọng nhất trong khoa học máy tính và công nghệ thông tin. Môn học này tập trung vào việc nghiên cứu và phát triển các cấu trúc dữ liệu và thuật toán để giải quyết các vấn đề tính toán một cách hiệu quả và hiệu năng cao.

Cấu trúc dữ liệu và giải thuật có mỗi liên hệ chặt chẽ với nhau, được thể hiện qua công thức sau:

Cấu trúc dữ liệu + Giải thuật = Chương trình

* + 1. Cấu trúc dữ liệu (Data Structures)

Cấu trúc dữ liệu là cách tổ chức, lưu trữ và quản lý dữ liệu để có thể truy cập và sử dụng dữ liệu một cách hiệu quả nhất. Các cấu trúc dữ liệu phổ biến bao gồm:

* + - * Mảng (Array): Tập hợp các phần tử có cùng kiểu dữ liệu được lưu trữ liên tiếp trong bộ nhớ.
      * Danh sách liên kết: Các nút dữ liệu liên kết với nhau một cách tuần tự.
      * Ngăn xếp (stack) và hàng đợi (queue) : Các cấu trúc dữ liệu cho phép truy xuất dữ liệu theo các cơ chế “LIFO”( Last In First Out ) và cơ chế “FIFO” (First In First Out)
    1. Giải thuật ( Algorithms)

Giải thuật là một tập hợp hữu hạn các chỉ thị để được thực thi theo một thứ tự nào đó nhằm thu được kết quả mong muốn. Một giải thuật được thiết kế tốt sẽ có độ phức tạp thời gian và không gian nó hiệu quả nhất để bài toán chạy được một cách tối ưu. Một số loại giải thuật phổ biến :

* + - * Sắp xếp (Sorting ): Sắp xếp dữ liệu theo thứ tự nhất định, ví dụ như sắp xếp nổi

bọt, sắp xếp chèn, sắp xếp chọn,…

* + - * Tìm kiếm (Searching ): Tìm kiếm phần tử trong một cấu trúc dữ liệu, ví dụ như

tìm kiếm tuần tự, tìm kiếm nhị phân,….

* + - * Đệ quy (Recursion): Các giải thuật sử dụng đệ quy để giải quyết vấn đề nhanh gọn.
    1. Cấu trúc dữ liệu và giải thuật có mối quan hệ gắn kết:
* Một cấu trúc dữ liệu phù hợp có thể làm cho một giải thuật trở nên hiệu quả hơn và ngược lại.
* Việc lựa chọn và triển khai cấu trúc dữ liệu và giải thuật phù hợp cũng ảnh hưởng đến trực tiếp đến hiệu năng của giải thuật.
* Ngoài ra, các giải thuật cũng đòi hỏi sự hiểu biết về các cấu trúc dữ liệu để có thể linh động chọn các giải thuật tốt, thực hiện chính xác và hiệu quả.
  + 1. Một số khái niệm khác

*Danh sách liên kết*: là tập hợp kiểu dữ liệu tuần tự mà mỗi phần tử chứa vị trí của

phần tử tiếp theo. Giúp cấp phát động lúc chạy chương trình. Kích thước bộ nhớ chỉ giới hạn do RAM.

*Danh sách liên kết đơn*: là một danh sách các nút, mỗi nút gồm 2 thành phần

* + - * Phần chứa dữ liệu data
      * Phần chỉ vị trí địa chỉ của phần tử tiếp theo trong danh sách

*Struct:* là một kiểu dữ liệu trong ngôn ngữ lập trình C++ được sử dụng để nhóm

nhiều dữ liệu khác nhau vào một thực thể duy nhất.

Struct giúp chương trình quản lí dữ liệu dễ dàng hơn vì có thể tạo ra các thực thể chứa đầy đủ các thông tin và truyền chúng qua các hàm.

*Thuật toán tìm kiếm tuần tự*: là một thuật toán tìm kiếm đơn giản, trong đó chương trình sẽ duyệt qua từng phần tử của danh sách để tìm kiếm phần tử có giá

trị mong muốn. Độ phức tạp của thuật toán là O( n), phù hợp với danh sách nhỏ hoặc không sắp xếp.

*Thuật toán sắp xếp chèn( Insertion Sort):* thường được sử dụng khi danh sách có kích thước nhỏ, nó hoạt động bằng cách duyệt qua từng phần tử và chèn chúng vào vị trí phù hợp trong danh sách đã được sắp xếp trước đó. Độ phức tạp của thuật toán là O (n2), thích hợp khi người dùng cần sắp xếp theo tên hoặc mã phim.

*Thuật toán nổi bọt( Bubble Sort):* Sắp xếp nổi bọt là thuật toán so sánh các phần tử liền kề và hoán đổi không đúng thứ tự. Độ phức tạp là O( n2), thích hợp cho danh sách nhỏ và đơn giản.

*Giao diện người dùng:* là phần trực tiếp tương tác với người dùng. Giao diện được thiết kế đơn giản dưới dạng menu các lựa chọn đề thực hiện các chức năng như thêm phim, xem danh sách phim, đặt vé và thanh toán.

* + 1. Phân tích cấu trúc dữ liệu

Cấu trúc dữ liệu được sử dụng trong chương trình bao gồm:

Struct Sinh viên: Lưu trữ thông tin chi tiết của một sinh viên như:

* + - * Số thứ tự của sinh viên
      * Mã số sinh viên
      * Họ và tên
      * Lớp học
      * Quê quán
      * Học lực
      * Ngày bắt đầu đăng ký ở ký túc xá
      * Hoàn cảnh gia đình

Struct Phòng: Lưu trữ thông tin của một phòng như:

* + - * Mã phòng ( Số phòng).
      * Cơ sở vật chất trong phòng.
      * Danh sách sinh viên có mặt trong phòng.
      * Chi phí điện.
      * Chi phí nước.
      * Tổng tiền điện, nước trong tháng.
      * Tiêu chí chấm phòng ở ( sạch sẽ, đẹp, văn minh).
    1. Thuật toán tạo và quản lý danh sách liên kết đơn:

**Cấu trúc dữ liệu:** Cả cấu trúc “**SinhVien**” và “**Phong**” đều sử dụng danh sách liên kết đơn. Mỗi phần tử trong danh sách (sinh viên hoặc phòng) trỏ đến phần tử tiếp theo thông qua con trỏ “**Next**”.

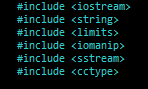
Các thao tác:

* + - * **Thêm phần tử:** Hàm “**ThemSinhVienTaiViTri**” và “**ThemPhong**” thưc hiện thêm một phần tử mới vào đầu hoặc một vị trí bất ký trong danh sách.
      * **Xoá phần tử:** Hàm “**XoaSinhVienTaiViTri**” và “**ThemPhong**” thực hiện việc xoá một phần tử tại vị trí cho trước trong danh sách.
      * **Duyệt danh sách:** Các hàm in danh sách (Ví dụ: “**InDanhSachSinhVien**” , “**InDanhSachPhong**”) duyệt qua từng phần tử trong danh sách để in thông tin ra màn hình.
    1. Các thuật toán tìm kiếm tuần tự:
* **Hàm “TimSinhVienTheoMSSV”** : Hàm này duyệt tuần tự qua danh sách sinh viên, so sánh MSSV của từng sinh viên với MSSV cần tìm. Nếu tìm thấy, hàm trả về con trỏ đến sinh viên đó, ngược lại trả về giá trị không xác định(NULL).
* **Ưu điểm:** Đơn giản, dễ hiểu
* **Nhược điểm:** Hiệu suất kém khi danh sách quá lớn, đặc biệt khi dữ liệu không được sắp xếp.
  1. Thuật toán kiểm tra và xử lý đầu vào:
* **Hàm “TimSinhVienTheoMSSV” :** Hàm này duyệt tuần tự qua danh sách sinh viên, so sánh MSSV của từng sinh viên với MSSV cần tìm. Nếu tìm thấy, hàm trả về con trỏ đến sinh viên đó, ngược lại trả về NULL.
* **Ưu điểm:** Đơn giản, dễ hiểu.
* **Nhược điểm:** Hiệu suất kém khi danh sách lớn, đặc biệt khi dữ liệu không được sắp xếp.
  + 1. Thuật toán kiểm tra và xử lý đầu vào:
       - **Hàm NhapNgayThangNam:** Kiểm tra tính hợp lệ của ngày tháng năm nhập vào.
       - **Hàm kiemTraChiChuCai:** Kiểm tra xem một chuỗi có chỉ chứa chữ cái và khoảng trắng hay không.
       - **Các hàm nhập thông tin khác:** Kiểm tra các ràng buộc về dữ liệu đầu vào như độ dài chuỗi, kiểu dữ liệu, giá trị hợp lệ.
       - **Tính tổng chi phí điện nước: e**Dựa trên số lượng kỳ điện và khối nước tiêu thụ để tính toán tổng chi phí.

**Xác định giải thưởng phòng:** Dựa trên các tiêu chí về phòng sạch, phòng đẹp và phòng an toàn để xác định giải thưởng.

# CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT CHƯƠNG TRÌNH

* 1. Hàm nhập ngày, tháng, năm sinh cho sinh viên:
     1. Khai báo thư viện:



* + - * Iostream: thư viện này để nhận đầu vào từ bàn phím và xuất dữ liệu ra.
      * String: Tạo ra đối tượng để lưu trữ chuỗi ký tự.
      * Limits: Xác định các thuộc tính đa dạng của các biến khác nhau.
      * Iomanip: Áp dụng các định dạng đầu ra tiên tiến, cải thiện rõ ràng thông tin được hiển thị.
      * Sstream: Có chức năng biến xâu string thành luồng(từ bàn phím là cin).
      * Cctype:
    1. Khai báo hàm: Ngày/Tháng/Năm



NhapNgayThangNam: Tên hàm, thể hiện chức năng nhập ngày tháng năm.

* + 1. Khai báo biến:Ngày/Tháng/Năm



Day, month, year: Lần lượt lưu trữ giá trị ngày, tháng, năm do người

dùng nhập Sep1, sep2: Ký tự phân cách giữa ngày, tháng, năm(ở đây sử dụng dấu '/')

* + 1. Nhập giá trị từ người dùng: Sử dụng "cin" để nhập giá trị từ người dùng theo thứ tự:
       1. Day: ngày
       2. Sep1: ký tự phân cách đầu
       3. Month: tháng
       4. Sep2: ký tự phân cách thứ hai
       5. Year: năm

Ví dụ: Nếu nhập 12/10/2024, thì xuất ra dữ liệu như sau:

* + - * 1. Day: 12
        2. Sep1:'/'
        3. Month: 10
        4. Sep2:'/'
        5. Year: 2024.
    1. Kiểm tra điều kiện hợp lệ:



Kiểm tra các điều kiện:

1. Sep1 != '/' || sep2 != '/': Đảm bảo các ký tự phân cách là dấu :'/'.
2. Day < 1 || day > 31: Ngày phải nằm trong khoảng hợp lệ( từ 1 đến 31).
3. Month < 1 || month > 12: Tháng phải nằm trong khoảng hợp lệ( từ 1 đến 12)
4. Year < 1900: Năm phải lớn hơn hoặc bằng 1900.

Nếu bất kỳ điều kiện nào không thỏa mãn, thì ngày tháng năm được coi là không hợp lệ.

* + 1. Hiển thi thông báo lỗi:



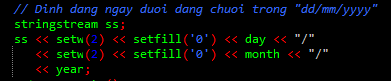
Thông báo lỗi khi phát hiện giá trị nhập không hợp lệ.

* + 1. Gọi hàm đệ quy:



Sử dụng đệ quy để yêu cầu người dùng nhập lại giá trị hợp lệ.

* + 1. Định dạng ngày tháng năm thành chuỗi:



* Stringstream ss: Tạo một luồng chuỗi để định dạng kết quả.
* Setw(2): Đảm bảo mỗi số(ngày/tháng)có độ rộng tối thiểu là 2 ký tự.
* Setfill('0'): Điền thêm ký tự '0' vào trước nếu số chỉ có 1 ký tự.
* Day và month: Được định dạng thành 2 chũ số, ví dụ:02, 09.
* Year: Giữ nguyên năm không thay đổi định dạng.

Kết quả: Chuỗi này tháng năm theo định dạng "dd/mm/yyyy".

* + 1. Trả về chuỗi kết quả:



Trả về chuỗi kết quả được định dạng từ luồng "ss".

* Chức năng toàn bộ: Hàm này cho phép nhập một ngày tháng năm. Kiểm tra tính hợp lệ(về phạm vi và định dạng), sau đó trả về dưới dạng chuỗi "dd/mm/yyyy". Nếu giá trị nhập sai, hàm sẽ yêu cầu nhập lại bằng cách gọi đệ quy chính nó.
  + 1. Khai báo hàm:



* Dòng này định nghĩa một hàm có tên " KiemTraChiChuCai", với kiểu trả về là "bool"(giá trị 'true' hoặc 'false').
* Hàm nhận một tham số 'str' là một 'string' hằng tham chiếu (được truyền vào mà không bị thay đổi).



* + 1. Sử dụng vòng lặp for:
* Dòng này sử dụng vòng lặp for kiểu range-based, để duyệt qua từng ký tự 'c' trong chuỗi 'str'.
* Mỗi ký tự của chuỗi 'str' sẽ lần lượt được gán vào biến 'c' trong vòng lặp này.



* Kiểm tra xem ký tự 'c' có phải là một chữ cái (isalpha( c )) hoặc khoảng trắng (c != ' ') hay không.
* isalpha ( c ) sẽ trả về 'true' nếu'c' là ký tự chữ cái (chữ cái viết hoa hoặc viết thường).
* !isalpha( c ) trả về 'true' nếu 'c' không phải là chữ cái.
* && c != ' ' kết hợp điều kiện này với điều kiện 'c' không phải là khoảng trắng.
* Nếu 'c' không phải chữ cái và cũng không phải là khoảng trắng, điều kiện này sẽ là 'true'.



* Nếu có bất kỳ ký tự nào khong phải là chữ cái hoặc khoảng trắng, hàm sẽ trả về 'false' ngay lập tức.
* 'return false' kết thúc hàm và cho biết chuỗi 'str' chúa ký tự không hợp lệ.



* Nếu vòng lặp 'for' kết thúc mà không tìm thấy ký tự nào không hợp lệ, hàm trả về 'true', có nghĩa là tất cả ký tự trong 'str' đều là chữ cái hoặc khoảng trắng.
  + Hàm 'KiemTraChiChuCai' sẽ duyệt qua từng ký tự trong chuỗi 'str'. Nếu tất cả ký tư đều là chữ cái hoặc khoảng trắng, hàm sẽ trả về 'true'. Nếu có bất kỳ ký tự nào không phải là chữ cái và không phải là khoảng trắng, hàm trả về ' false'.
  1. Hàm thêm sinh viên tại vị trí được chỉ định:
     1. Khai báo hàm có tên "ThemSinhVIenTaiViTri":

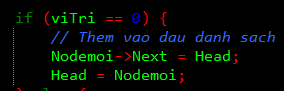


* Hàm sẽ gồm 3 tham số:
  1. SinhVien\*& Head: Tham chiếu đến con trỏ 'Head', là con trỏ đầu của danh sách. Tham chiếu này giúp hàm có thể thay đổi con trỏ 'Head' nếu cần(như khi thêm vào đầu danh sách).
  2. Const SinhVien& SinhVienmoi: Sinh viên mới cần được thêm vào danh sách. Tham số này là hằng tham chiếu, giúp tránh sao chép dữ liệu và bảo vệ dữ liệu gốc.
  3. Int viTri: Vị trí mà sinh viên mới cần được chèn vào trong danh sách (bắt đầu từ 0).
     1. Tạo Nodecho sinh viên:

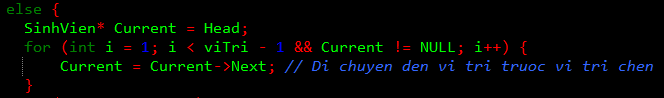


* Tạo một 'SinhVien' mới có tên là 'Nodemoi', đây là bản sao của sinh viên 'SinhVienmoi'.
* new SinhVien(SinhVienmoi): Khởi tạo một bản sao của 'SinhVienmoi' trên vùng nhớ động(heap).

-Nodemoi -> Next = NULL : Gán con trỏ 'Next' của 'Nodemoi' bằng giá trị NULL, vì dây là một node mới chưa được nối vào danh sách.



* + 1. Kiểm tra vị trí để thêm sinh viên.:
* Kiểm tra nếu 'viTri' bằng 0 (tức là thêm vào đầu danh sách).
* Nếu viTri == 0, gán Nodemoi -> Next bằng 'Head' để Nodemoi trỏ đến phần tử hiện tại ở đầu danh sách.
* Cập nhật 'Head' trỏ đến Nodemoi, nghĩa là Nodemoi bây giờ trở thành phần tử đầu của danh sách.



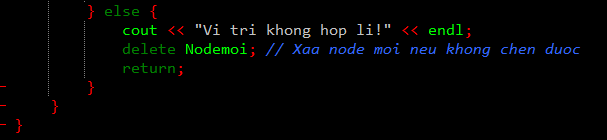
* Nếu không chèn vào đầu danh sách ( viTri != 0), tiến hành chèn vị trí bất kỳ.
* SinhVien\* Current = Head : Khởi tạo con trỏ Current để duyệt danh sách, bắt đầu từ đầu danh sách (Head).
* Vòng lặp for duyệt qua danh sách đến vị trí ngay trước 'viTri'.



* Sau khi tìm đến vị trí trước 'viTri', kiểm tra nếu như Current != NULL thì:

Gán Nodemoi -> Next = Current -> Next để node mới trỏ đến node kế tiếp của Current.

Current -> Next = Nodemoi sau đó nối 'Current' với 'Nodemoi', chèn 'Nodemoi' vào 'viTri'.



Nếu Current == NULL có nghĩa là vị trí cần chèn không hợp lệ (vượ quá độ dài của danh sách).

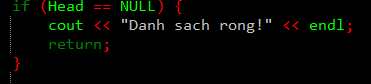
Thông báo ra dòng "Vị trí không hợp lí!" bằng hàm cout.

Delete Nodemoi: Giải phóng bộ nhớ của Nodemoi để tránh bị tràn bộ nhớ. Cuối cùng thoát khỏi hàm bằng dòng lệnh: return.

* + Việc gọi hàm 'ThemSinhVienTaiViTri' sẽ có chức năng chèn một sinh viên mới vào danh sách liên kết đơn tại vị trí được chỉ định:
* Nếu viTri == 0, sinh viên mới được thêm vào đầu danh sách.
* Nếu không, hàm sẽ duyệt đến vị trí trước 'viTri', và chèn sinh viên mới vào danh sách.
* Nếu 'viTri' không hợp lệ, hàm báo lỗi và không thực hiện chức năng chèn.
  1. Hàm xóa sinh viên tại vị trí được chỉ định:
     1. Khai báo hàm đặt tên là 'XoaSinhVienTaiViTri':



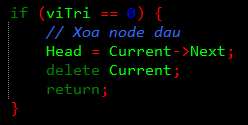
* Sử dụng kiểu mà hàm trả về là 'void', tức là không trả về bất kì giá trị nào.
* Hàm sẽ nhận 2 tham số mà người dùng nhập vào:
  1. SinhVien\*& Head: Tham chiếu đến con trỏ 'Head' của danh sách, cho phép thay đổi danh sách ban đầu (nếu cần).
  2. Int viTri: Vị trí cần xóa sinh viên trong danh sách (bắt đầu từ 0).
     1. Kiểm tra xem danh sách có đang trống hay không :



* Nếu Head = NULL: Danh sách rỗng, in ra thông báo "Danh sách rỗng!" và thoát hàm bằng returm.



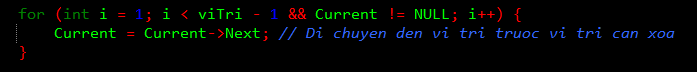
* Khởi tạo con trỏ 'Current' để duyệt qua danh sách, bắt đầu từ 'Head' (Phần tử đầu tiên).



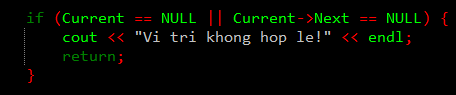
* Kiểm tra nếu như viTri == 0 (tức cần xóa phần tử đầu của danh sách).

-Nếu kiểm tra đúng, cập nhật lại Head = Current -> Next để trỏ 'Head' đến node thứ hai của danh sách (bỏ qua node đầu).

* Xóa node đầu (node hiện tại) bằng delete Current.
* Thoát hàm bằng return sau khi xóa xong.
  + 1. Sử dụng vòng lặp if di chuyển con trỏ đến vị trí cần xóa:

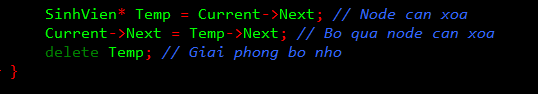


* Nếu không xóa phần tử đầu, thì ta dùng vòng lặp for để di chuyển 'Current' đến vị trí trước 'viTri'.
* Dòng lệnh 'int i = 1; i < viTri - 1 && Current != NULL; i++': Có chức năng duyệt danh sách từ vị trí đầu đến ngay trước viTri - 1.
* Current = Current -> Next: Giúp di chuyển 'Current' sang node kế tiếp trong mỗi lần lặp.



Sau khi vòng lặp kết thúc, kiểm tra tính hợp lệ của Current.

* Nếu Current == NULL hoặc Current -> Next == NULL, điều này nghĩa là vị trí cần xóa không hợp lệ (vượt quá độ dài danh sách).
* In ra thông báo "Vị trí không hợp lệ!" sau đó thoát chương trình qua dòng lệnh return;



* Nếu vị trí hợp lệ, tạo con trỏ 'Temp' trỏ đến node cần xóa (Current -> Next).

-Gán Current -> Next = Temp -> Next để 'Current' bỏ qua 'Temp' và trỏ trực tiếp đến node sau 'Temp'.

* + Hàm 'XoaSinhVienTaiViTri' thực hiện các bước sau để xóa một node tại vị trí 'viTri' trong DSLK đơn:

1. Kiểm tra nếu danh sách trống, in thông báo và thoát.
2. Nếu viTri == 0, xóa phần tử đầu danh sách, cập nhật con trỏ 'Head', và thoát hàm.
3. Nếu không phải xóa đầu danh sách, duyệt đến vị trí ngay trước viTri.
4. Nếu vị trí không hợp lệ, in thông báo lỗi và thoát hàm.
5. Nếu vị trí hợp lệ:
   * Tạo con trỏ 'Temp' trỏ đến node cần xóa.
   * Bỏ qua 'Temp' bằng cách gán Current -> Next trỏ đến node sau 'Temp'
   * Xóa 'Temp' để giải phóng bộ nhớ.
   1. Hàm thêm phòng mới vào danh sách liên kết đơn của các phòng:
      1. Khai báo hàm đặt tên là: 'ThemPhong' sử dụng kiểu trả về void:



* Hàm sẽ nhận 2 tham số mà người dùng nhập vào:
  1. Phong\*& Head: Tham chiếu đến con trỏ 'Head' của DSLK các phòng. Tham chiếu giúp hàm có thể thay đổi 'Head' nếu cần, như khi thêm vào đầu danh sách.
  2. Const Phong& Phongmoi: Tham chiếu hằng tới đối tượng

'Phongmoi' chứa thông tin của phòng mới cần thêm. Tham chiếu hằng giúp tránh sao chép dữ liệu và bảo vệ dữ liệu gốc khỏi bị thay đổi.

* + 1. Tạo một Node mới là (Phong\* Nodemoi) là bản sao của phòng 'Phongmoi':



* new Phong(Phongmoi): Khởi tạo bản sao của Phongmoi trê n vùng nhớ động (heap), giúp node mới tồn tại độc lập và không bị xóa khi ra khỏi hàm.



* Gán con trỏ 'Next' của 'Nodemoi' trỏ đến 'Head' (Phần tử đầu hiện tại của danh sách).
* Điều này có nghĩa là 'Nodemoi' sẽ trở thành phần tử đầu tiên của DSLK, còn 'Head' sẽ trỏ đến phần tử thứ hai.



* Đặt con trỏ 'Danhsachsinhvien' của 'Nodemoi' bằng NULL, biểu thị rằng danh sách sinh vien trong phòng mới ban đầu là trống (chưa có sinh viên nào).
* Cập nhật 'Head' trỏ đến 'Nodemoi' trở thành phần tử đầu tiên của danh sách.
* Sau khi hoàn thành, danh sách sẽ bắt đầu từ Nodemoi, với node tiếp theo là 'Head' ban đầu.
  + Hàm 'ThemPhong' thêm một phòng mới vào đầu DSLK các phòng:

1. Tạo một node mới từ 'Phongmoi' trên vùng nhớ động.
2. Gán 'Next' của node mới trỏ đến node đầu của danh sách hiện tại (Head).
3. Đặt danh sách sinh viên trong phòng mới (Danhsachsinhvien) thành NULL.
4. Cập nhật 'Head' trỏ đến node mới, để node này trở thành phần tử đầu tiên của danh sách.
   1. Hàm nhập thông tin sinh viên:
      1. Khai báo hàm:

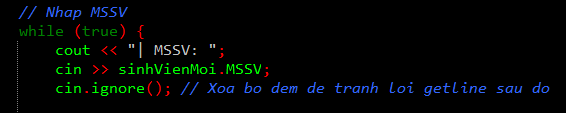
* Khai báo hàm 'nhapSinhVien', nhận tham số 'head' là tham chiếu đến con trỏ đầu của danh sách liên kết, cho phép thay đổi danh sách bên ngoài hàm.



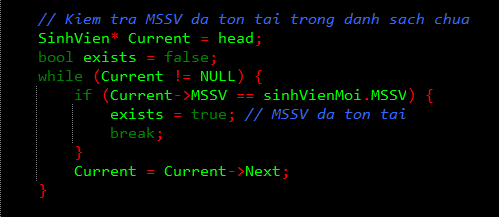
* Khởi tạo một đối tượng 'sinhVienMoi' kiểu 'SinhVien' để lưu thông tin sinh viên mới.



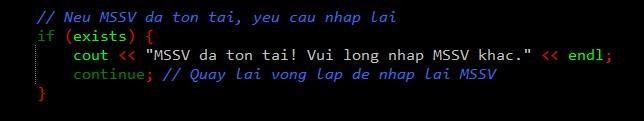
* In ra thông báo nhắc người dùng bắt đầu nhập thông tin sinh viên.
  + 1. Sử dụng vòng lặp while để nhập mã sinh viên:



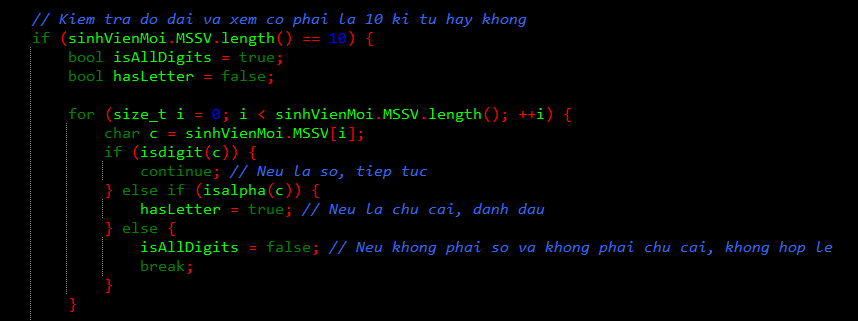
* + - * Vòng lặp while để yêu cầu nhập MSSV.
      * 'cin >> sinhVienMoi.MSSV' nhận mã số sinh viên từ người dùng.
      * 'cin.ignore(): xóa bộ đệm, tránh lỗi khi sử dụng getline sau đó.
    1. Kiểm tra xem Mã số sinh viên có tồn tại hay không :



* + - * Kiểm tra MSSV có tồn tại trong danh sách không bằng cách duyệt từng node từ 'head' đến cuối danh sách.
      * Nếu tìm thấy sinh viên có MSSV trùng, 'exists' được gán 'true' và thoát vòng lặp.



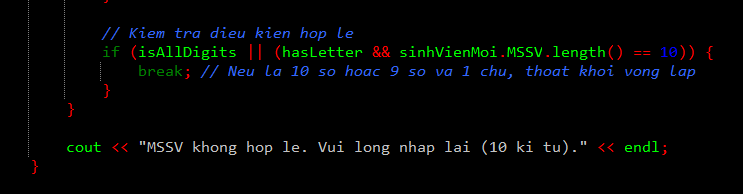
* + - * Nếu exists == true: Tức MSSV đã tồn tại, thông báo yêu cầu nhập lại MSSV và tiếp tục vòng lặp để yêu cầu nhập lại.
    1. Kiểm tra độ dài và định dạng của MSSV:



* + - * Kiểm tra độ dài MSSV có bằng 10 ký tự không.
      * Biến 'isAllDigits' để kiểm tra xem MSSV có phải là chuỗi số hoàn toàn, và 'hasLetter' kiểm tra có chứa ký tự chữ cái nào không.
      * Duyệt từng ký tự trong MSSV:
* Nếu là số (isdigit ( c )), tiếp tục kiểm tra ký tự tiếp theo.

-Nếu là chữ cái ( isalpha( c )), đánh dấu 'hasLetter = true'.

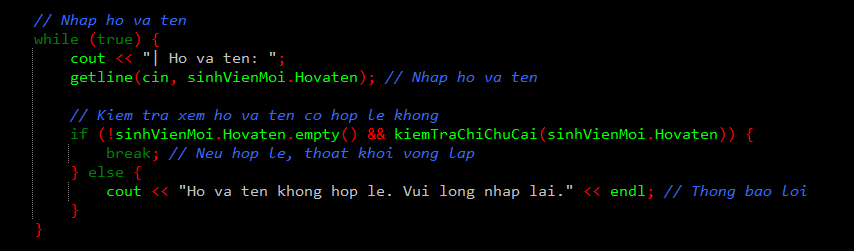
* Nếu không phải số cũng không phải chữ, gán 'isAllDIgits = false' và thoát khỏi vòng lặp.
  + 1. Kiểm tra điều kiện hợp lệ:



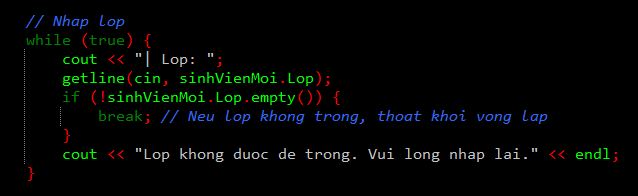
-Nếu MSSV có đúng định dạng 10 ký tự là số, hoặc có 9 số và 1 chữ cái, thoát khỏi vòng lặp.

-Nếu không, yêu cầu nhập lại.

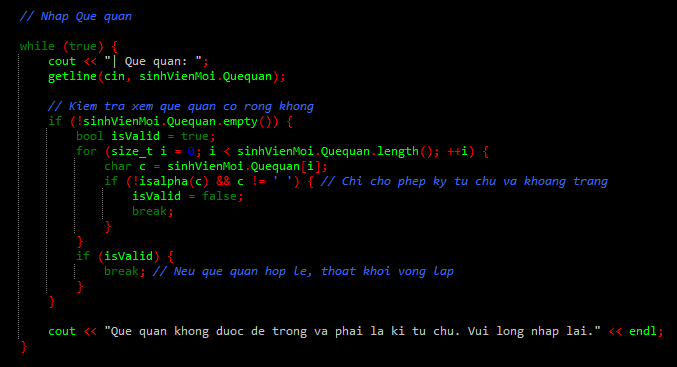
* + 1. Khai báo nhập họ và tên:



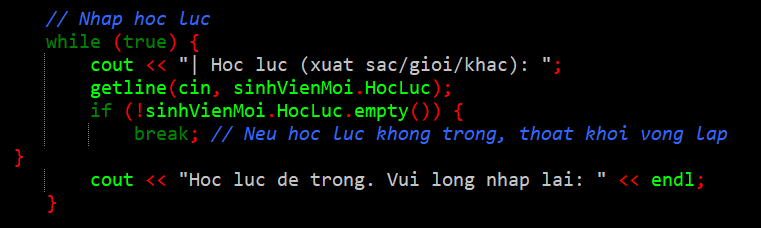
* + - * Sử dụng vòng lặp while để nhập 'Hovaten'.
      * Gọi hàm 'KiemTraChiChuCai' để kiểm tra 'Hovaten' chỉ chứa ký tự chữ và khoảng trắng.
      * Nếu hợp lệ, thoát vòng lặp
      * Nếu không hợp lệm yêu cầu nhập lại.
    1. Khai báo nhập lớp**:**



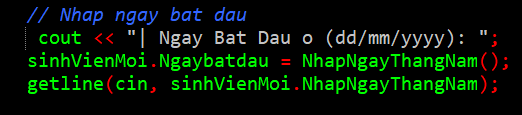
* + - * Yêu cầu nhập 'Lop' và kiểm tra không được để trống. Nếu hợp lệ, thoát vòng lặp; nếu không, yêu cầu nhập lại.
    1. Khai báo hàm nhập quê quán:



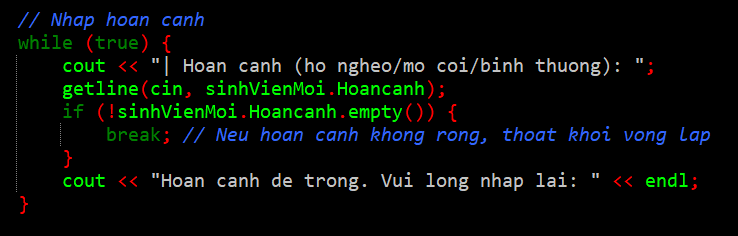
* + - * Kiểm tra 'Quequan' không để trống và chỉ chứa chữ cái hoặc khoảng trắng. Nếu hợp lệ, thoát vòng lặp; nếu không; yêu cầu nhập lại.
    1. Khai báo hàm nhập học lực:



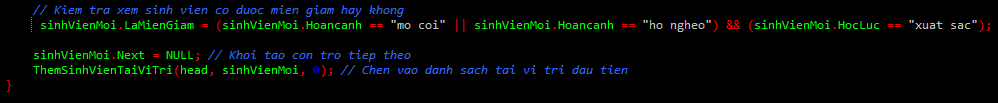
* + - * Kiểm tra 'HocLuc' không để trống. Nếu hợp lệ, thoát vòng lặp; nếu không, yêu cầu nhập lại.
    1. Khai báo hàm nhập ngày bắt đầu:



* + - * Yêu cầu nhập ngày bắt đầu, sử dụng hàm 'NhapNgayThangNam()' để nhập và kiểm tra định dạng ngày tháng năm.
    1. Khai báo hàm nhập hoàn cảnh:



* + - * Yêu cầu nhập 'Hoancanh', kiểm tra không để trống. Nếu hợp lệ, thoát khỏi vòng lặp; nếu không, yêu cầu nhập lại.
    1. Khai báo hàm kiểm tra miễn giảm học phí:



* + - * Xét điều kiện 'LaMienGiam': Nếu hoàn cảnh là "mồ côi" hoặc "hộ nghèo" và đạt học lực "xuất sắc" thì sinh viên đó được miễn giảm học phí.
      * 'sinhVienMoi.Next = NULL: khởi tạo con trỏ Next của sinhVienMoi là NULL.
      * Gọi 'ThemSinhVienTaiViTri' để thêm 'sinhVienMoi' vào đầu danh sách liên kết.
  1. Hàm nhập các thông tin về phòng ở:
     1. Khai báo hàm:

Khai báo hàm 'NhapPhong' với tham số 'Phong\*& Head', là tham chiếu con trỏ 'Head' của danh sách liên kết. Điều này cho phép hàm thay đổi danh sách bên ngoài hàm, chẳng hạn thêm một phòng mới vào danh sách.

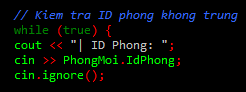


* + 1. In ra thông báo nhập phòng:



- In ra dòng thông báo để báo cho người dùng rằng họ sẽ nhập thông tin của phòng.

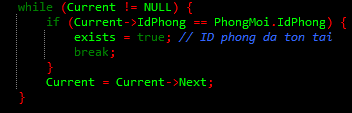
* + 1. Kiểm tra ID phòng có trùng hay không:



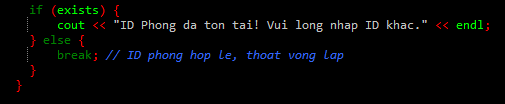
1. while (true):
   * Mở đầu một vòng lặp vô hạn while (true), cho phép tiếp tục yêu cầu người dùng nhập ID Phong cho đến khi nhập thành công và thỏa mãn điều kiện không trùng lặp.
2. cout << "| ID Phong : ";:
   * In ra dòng ID phòng :
3. cin >> PhongMoi.IdPhong; :
   * Dùng 'cin' để nhận ID Phong từ người dùng và lưu vào thuộc tính 'IdPhong' của đối tượng 'PhongMoi'.
   * 'IdPhong': sẽ là một giá trị duy nhất đại diện cho phòng, vì vậy, kiểm tra trùng lặp là cần thiết để đảm bảo mỗi phòng có một ID duy nhất.
4. cin.ignore(); :
   * Dòng 'cin.ignore();' giúp xóa bỏ ký tự xuống dòng (newline) còn lại trong bộ đệm sau khi người dùng nhập 'ID Phong'.
   * Điều này ngăn chặn lỗi khi sử dụng 'getline' để nhập chuỗi sau đó, vì 'getline' có thể đọc ký tự xuống dòng còn sót trong bộ đệm thay vì chờ người dùng nhập thông tin mới.
   1. Hàm kiểm tra ID phòng có tồn tại trong danh sách hay không ?
      1. Khởi tạo hàm:
         * Khởi tạo con trỏ 'Current' kiểu 'Phong' và trỏ đến 'Head', tức là phần đầu của danh sách liên kết.
         * Mục đích để duyệt qua toàn bộ danh sách liên kết, bắt đầu từ 'Head'
         * Khởi tạo thêm một biến boolean 'exists' và gán giá trị là 'false'.
         * Biến này dùng để đánh dấu xem ID phòng mới(Phongmoi.IdPhong) có tồn tại trong danh sách liên kết hay không.



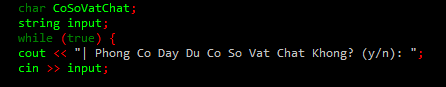
* + 1. Sử dụng vòng lặp while để duyệt qua từng phần tử trong danh sách:



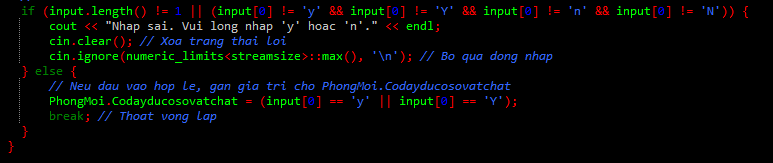
* + - * Vòng lặp sẽ tiếp tục chạy khi 'Current' đến khi giá trị bằng NULL, tức là sẽ chạy vòng lặp đến khi hết phần tử trong danh sách.
      * Kiểm tra xem 'IdPhong' của phần tử hiện tại (Current) có trùng với 'IdPhong' của phòng mới(Phongmoi.IdPhong) hay không.
      * Nếu như điều kiện trên đúng, nghĩa là ID phòng đã tồn tại trong danh sách.
      * Khi đã biết ID phòng đã tồn tại, gán 'exists' là 'true'.
      * 'Break' để thoát vòng lặp while sau khi tìm thấy ID phòng đã tồn tại.
      * Di chuyển con trỏ'Current' đến phần tử tiếp theo trong danh sách liên kết.



* + - * Sau khi kết thúc vòng lặp while, kiểm tra giá trị của 'exists'.
      * Nếu'exists' là true, tức ID phòng đã tồn tại trong danh sách.
      * Dùng 'cout' để in ra màn hình dòng: "ID phòng đã tồn tại! Vui lòng nhập ID khác"
      * Nếu ID phòng không tồn tại( exists lúc này trả ề giá trị false), thực hiện chức năng 'break' để thoát vòng lặp.
  1. Khai báo các hàm thông tin về cơ sơ vật chất trong phòng ở:
     1. Khai báo biến:



* + - * Tạo một biến tên "input" kiểu dữ liệu là 'string' để lưu tạm giá trị đầu vào của người dùng.
      * Khởi tạo vòng lặp while yêu cầu người dùng nhập thông tin về cơ sở vật chất đến khi nhận được đầu vào hợp lệ.
      * Trước khi nhập dữ liệu đầu vào, in ra màn hình thông báo:"Phòng có đầy đủ cơ sở vật chất hay không?"
      * Nhập đầu vào từ người dùng và lưu vào biến 'input'.



* Dòng lệnh kiểm tra đầu vào người dùng có hợp lệ hay không.
  + Input.length() !=1: kiểm tra độ dài input. Độ dài chỉ hợp lệ nếu độ dài là 1, tức là chỉ có 1 ký tự.
  + (input[0] !='y' && input[0] != 'Y' && input[0] != 'n' && input[0] != 'N': Kiểm tra xem ký tự đầu tiên (ivnàpduut y nhất) của

có phải là 'y', 'Y', 'n', hoặc 'N' hay không. Nếu không phải, đầu vào không hợp lệ.

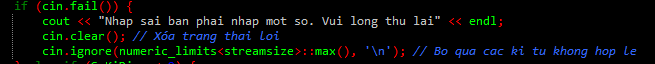
* + Nếu một trong hai điều kiện trên đúng, tức là đầu vào không hợp lệ, chương trình sẽ đi vào nhánh if.
  + In ra thông báo lỗi, yêu cầu người dùng nhập lại với các lựa chọn không hợp lệ('y' hoặc 'n').
  + Xóa trạng thái lỗi của cin để chuẩn bị cho lần nhập tiếp theo(Điều này rất hữu ích khi có lỗi trong quá trình nhập liệu, tránh việc cin bị kẹt ở trạng thái lỗi).
  + Cin.ignore(numeric\_limits<streamsize>::max(), '\n');: Bỏ qua tất cả ký tự còn lại trong dòng nhập
  + Else chỉ được thực thi khi đầu vào hợp lệ, nghĩa là input chứa một ký tự duy nhất 'y','n','Y' hoặc 'N'.
* PhongMoi.Codayducosovatchat = (input[0] == 'y' || input [0] == 'Y');
  + Nếu đầu vào là 'y' hoặc 'Y', gán

PhongMoi.Codayducosovatchat là false(không có đầy đủ cơ sở vật chất).

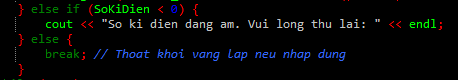
* + Thoát vòng lặp while(true) cho toàn đoạn mã.
  + Sau khi nhận được đầu vào hợp lệ và gán giá trị cho 'PhongMoi.Codayducosovatchat'.
  1. Khai bao hàm nhập và tính toán số ký điện:
     1. Khai báo biến kí điện và các hàm tính toán:
        + Khởi tạo biến SoKiDien kiểu Double để lưu số kWh điện người dùng nhập vào.
        + Kiểu double cho phép lưu giá trị thập phân, phù hợp với số kWh có thể là số lẻ.
     2. Sử dụng vòng lặp để kiểm tra dữ liệu đầu vào:
        + Bắt đầu vòng lặp do-while ể thực hiện việc yêu cầu nhập số kWh điện cho đến khi nhận được đầu vào hợp lệ.
        + In ra câu nhắc yêu cầu người dùng nhập số lượng kWh điện.
        + Lấy dữ liệu từ người dùng nhập vào và lưu vào biến SoKiDien.
        + Nếu người dùng nhập một giá trị không phải là số, cin sẽ chuyển sang trạng thái lỗi, và SoKiDien sẽ không được gán giá trị.



* + 1. Kiểm tra dữ liệu :
       - Kiểm tra xem cin có gặp lỗi trong quá trình nhập liệu hay không.
       - Cin.fail() trả về true nếu đầu vào không phải là số hợp lệ, ví dụ như khi người dùng nhập ký tự thay vì số.
       - Nếu đầu vào không hợp lệ, in ra thông báo lỗi, yêu cầu người dùng nhập lại số.
       - Xóa trạng thái lỗi của cin để chuẩn bị cho lần nhập tiếp theo.
       - Bỏ qua các ký tự còn lại trong dòng nhập.



* + 1. Kiểm tra giá trị hợp lý chưa :
       - Kiểm tra xem giá trị SoKiDien có âm không.
       - Số kWh điện không thể là số âm, vì vậy đây là điều kiện không hợp lệ.
       - Nếu người dùng nhập một số âm, in ra thông báo lỗi yêu cầu nhập lại số kWh dương.
       - Nếu đầu vào hợp lệ và là số dương, thoát khỏi vòng lặp do-while.
       - Lệnh break kết thúc vòng lặp ngay khi nhập đúng.



* + 1. Gán giá trị điện vào biến lưu trữ:
       - Kết thúc vòng lặp do-while.
       - Vòng lặp này sẽ tiếp tục chạy cho đến khi nhận được đầu vào hợp lệ.
       - Tính toán chi phí điện và gán kết quả cho thuộc tính SoKiDien với 2000 (giả định là đơn giá điện là 2000 đồng/kWh).



* 1. Khai bao hàm nhập và tính toán số khối nước :

- Tương tự như phần tính toán số ký điện

* 1. Tính toán tổng chi phí điện nước của phòng.
* Với dữ liệu được nhập và gán kết quả vào 'Chiphidien' và 'Chiphinuoc' của 'PhongMoi', ta sẽ tính được tổng chi phí điện nước dựa trên dữ liệu đã nhập, cuối cùng gán kết quả tính được vào 'TongTienDienNuoc' của 'PhongMoi'.



* 1. Hàm khai báo chức năng xử lý thông tin giải thưởng khi đạt đủ các tiêu chí sạch, đẹp, an toàn.
     1. Khai báo biến:
        + Khởi tạo ba biến kiểu 'char' : 'PhongSach', 'PhongDep', 'PhongAnToan'.
        + Mỗi biến đại diện cho một tiêu chí của phòng:
          1. PhongSach(sạch sẽ)
          2. PhongDep(đẹp)
          3. PhongAnToan(an toàn)



* + 1. Khởi tạo vòng lặp while:
       - Bắt đầu một vòng lặp whie(true).
       - Vòng lặp này sẽ tiếp tục chạy cho đến khi nhận được đầu vào hợp lệ từ người dùng hoặc có một lệnh break để thoát khỏi vòng lặp.



* + - * In ra yêu cầu người dùng nhập thông tin cho tiêu chí "phòng sạch sẽ".
      * Người dùng được yêu cầu nhập 'y' nếu phòng sạch sẽ, hoặc 'n' nếu không.
      * Khai báo biến input kiểu string để nhận đầu vào của người dùng,
      * Sử dụng kiểu string để dễ dàng kiểm tra độ dài và tính hợp lệ của đầu vào.
      * Nhận đầu vào từ người dùng và lưu vào biến input.
      * Sau bước này, các đoạn mã tiếp theo sẽ kiểm tra độ dài và tính hợp lệ của input để đảm bảo người dùng chỉ nhập 'y' hoặc 'n'.



=> Với các thông tin về phòng đẹp và phòng an toàn ta cũng khai báo tương tự như phòng sạch ở trên.

* + 1. Kiểm tra chỉ số độ dài của đầu vào khi người dùng nhập vào:



* + - * Kiểm tra độ dài của input để đảm bảo nó chỉ chứa đúng một ký tự.
      * Điều này giúp xác nhận rằng người dùng không nhập thừa hoặc thiếu ký tự.



* + - * Gán ký tự đầu tiên (và duy nhất) của input cho biến PhongSach.
      * Sau dòng này, PhongSach sẽ chứa ký tự mà người dùng đã nhập, giúp dễ dàng kiểm tra tính hợp lệ.



* + - * Kiểm tra nếu PhongSach là 'y' hoặc 'Y' (tức là phòng sạch sẽ).
      * Nếu điều kiện đúng, mã sẽ vào nhánh if này.



* + - * Gán true cho thuộc tính PhongSach của PhongMoi, biểu thị rằng phòng đạt tiêu chí sạch sẽ.
      * Break: thoát khỏi vòng lặp while(true) bên ngoài sau khi đầu vào hợp lệ và giá trị đã được gán.
      * Điều này cho thấy quá trình kiểm tra đầu vào cho tiêu chí sạch sẽ đã hoàn tất.



* + - * Kiểm tra nếu PhongSach là 'n' hoặc 'N' (tức là phòng không sạch sẽ).
      * Nếu điều kiện đúng, mã sẽ vào nhánh else if này.



* + - * Gán false cho thuộc tính PhongSach của PhongMoi, biểu thị rằng phòng không đạt tiêu chí sạch sẽ.
      * Tương tự như trên, thoát khỏi vòng lặp sau khi gán giá trị hợp lệ



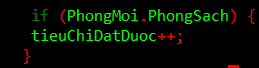
* + - * Kết thúc vòng lặp if (input.length() == 1).
      * Nếu đầu vào không hợp lệ (không phải một ký tự hoặc không phải 'y'/'Y' hay 'n'/'N'), in ra thông báo lỗi.
      * Câu lệnh này nằm ngoài if , nên sẽ thực thi mỗi khi input không hợp lệ.

=> Tương tự cho phần khai báo các kiểu dữ liệu cho phòng sạch, phòng đẹp và phòng an toàn ở trên. Với hàm kiểm tra chỉ số độ dài ta áp dụng cho 2 phần phòng an toàn và phòng đẹp tương tự như phòng sạch.

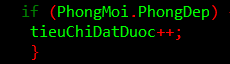
* + 1. Hàm kiểm tra số tiêu chí đạt được:



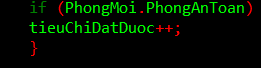
* + - * Khởi tạo biến tieuChiDatDuoc kiểu trả về int với giá trị ban đầu là 0.
      * Biến này sẽ dùng để đếm số tiêu chí mà phòng đáp ứng được.



* + - * Kiểm tra nếu PhongMoi.PhongSach là true, nghĩa là phòng đạt tiêu chí sạch sẽ.
      * Nếu điều kiện đúng, tăng tieuChiDatDuoc lên 1 lần.



* + - * Kiểm tra nếu PhongMoi.PhongDep là true, nghĩa là phòng đạt tiêu chí đẹp
      * Nếu điều kiện đúng, tăng tieuChiDatDuoc lên 1 lần.



* + - * Kiểm tra nếu PhongMoi.PhongAnToan là true, nghĩa là phòng đạt tiêu chí an toàn.
      * Nếu điều kiện đúng, tăng tieuChiDatDuoc lên 1 lần.



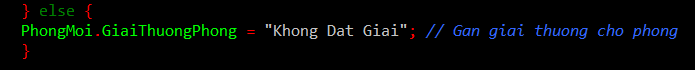
* + - * Kiểm tra nếu tieuChiDatDuoc bằng 3 (nghĩa là phòng đạt đủ 3 tiêu chí).



* + - * Nếu điều kiện đúng, gán giá trị "Giai Nhat" cho thuộc tính GiaiThuongPhong của PhongMoi.



* + - * Nếu tieuChiDatDuoc không bằng 3, chương trình kiểm tra tiếp nếu nó bằng 2 (nghĩa là phòng đạt 2 tiêu chí).
      * Nếu điều kiện này đúng, gán giá trị "Giai Khuyen Khich" cho thuộc tính GiaiThuongPhong của PhongMoi.



* + - * Nếu tieuChiDatDuoc nhỏ hơn 2 (nghĩa là phòng chỉ đạt 1 hoặc không đạt tiêu chí nào), chương trình gán "Khong Dat Giai" cho GiaiThuongPhong của PhongMoi.



* + - * Khởi tạo thuộc tính Danhsachsinhvien của PhongMoi là NULL.
      * Danhsachsinhvien có thể là một con trỏ hoặc một danh sách lưu trữ danh sách sinh viên trong phòng. Gán NULL để biểu thị rằng danh sách sinh viên đang trống khi mới khởi tạo.



* + - * Khởi tạo thuộc tính Next của PhongMoi là NULL.
      * Next có thể là một con trỏ chỉ đến phòng kế tiếp trong danh sách phòng (giống như trong danh sách liên kết). Gán NULL cho Next biểu thị rằng PhongMoi hiện chưa liên kết với phòng nào khác.



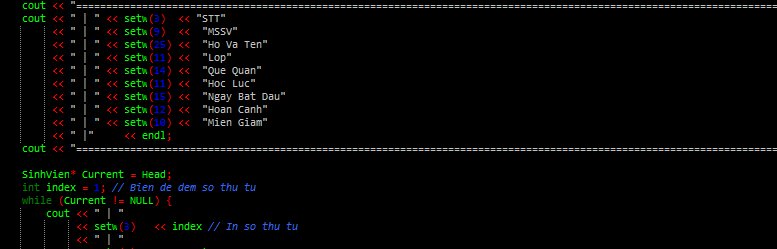
* + - * Gọi hàm ThemPhong, truyền vào Head(con trỏ đầu danh sách các phòng) và PhongMoi.
      * Hàm ThemPhong có khả năng thêm PhongMoi vào danh sách các phòng, cập nhật danh sách để lưu trữ phòng mới này.

=> Đoạn mã này xác định giải thưởng cho phòng dựa trên số tiêu chí đạt được, khởi tạo các thuộc tính liên quan đến danh sách sinh viên và liên kết phòng, và cuối cùng thêmPhongMoi vào danh sách các phòng.

* 1. Hàm in danh sách sinh viên ra màn hình:
     1. Khai báo hàm chức năng:



* + - * Khai báo hàm InDanhSachSinhVien nhận vào một con trỏ Head kiểu SinhVien, trỏ đến sinh viên đầu tiên trong danh sách liên kết.
    1. Xây dựng tiêu đề cột:



* + - * In ra dòng tiêu đề của bảng, biểu thị các thông tin của sinh viên bao gồm số thứ tự (STT), mã số sinh viên (MSSV), họ và tên, lớp, quê quán, học lực, ngày bắt đầu, hoàn cảnh, và miễn giảm học phí.
      * Setw() dùng để canh lề và định kích thước cột.



* + - * Khai báo một con trỏ Current trỏ đến sinh viên đầu tiên trong danh sách, bắt đầu việc duyệt danh sách liên kết.



* + - * Khai báo biến index để giữ số thứ tự (STT) cho từng sinh viên.



* + - * Lặp qua từng sinh viên trong danh sách liên kết cho đến khi current trỏ đến NULL(kết thúc danh sách).
    1. In thông tin sinh viên:



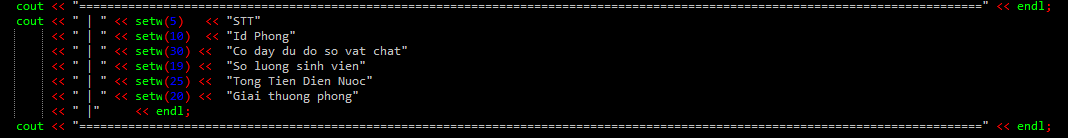
* + - * Mỗi lần lặp, hàm in thông tin chi tiết của sinh viên mà Current đang trỏ tới.
      * Các thuộc tính của sinh viên gồm:
* MSSV: Mã số sinh viên.
* Hovaten: Họ và tên.
* Lop: Lớp.
* Quequan: Quê quán.
* HocLuc: Học lực.
* Ngaybatdau: Ngày bắt đầu ở ký túc xá.
* Hoancanh: Hoàn cảnh.
* LaMienGiam: Cho biết sinh viên có được miễn giảm học phí hay không (in "Co" nếu đúng, "Khong" nếu sai).
* Di chuyển con trỏ Current đến sinh viên tiếp theo trong danh sách liên kết.



* Tăng index lên 1 để cập nhật số thứ tự cho sinh viên tiếp theo.
* Hàm InDanhSachSinhVien duyệt qua từng sinh viên trong danh sách liên kết, in thông tin của từng sinh viên theo định dạng bảng với các cột, và thêm các dòng phân cách để dễ nhìn.
  1. Hàm in danh sách phòng theo dạng bảng:
     1. Khai báo hàm chức năng:



* + - * Khai báo hàm InDanhSachPhong nhận vào con trỏ Head trỏ đến đầu của danh sách liên kết Phong.
    1. In tiêu đề bảng và dòng phân cách:



* + - * Hai dòng đầu và cuối tạo các đường phân cách trên và dưới tiêu đề bảng.
      * Setw() dùng để canh lề và định kích thước cột
      * Các tiêu đề cột là:

1. STT: Số thứ tự của phòng.
2. IdPhong: Mã ID của phòng.
3. Co day du co so vat chat : Xác định xem phòng có đầy đủ cơ sở vật chất hay không.
4. So luong sinh vien: Số lượng sinh viên trong phòng.
5. Tong tien dien nuoc: Tổng tiền điện nước của phòng.
6. Giai thuong phong: Giải thưởng của phòng(nếu có).



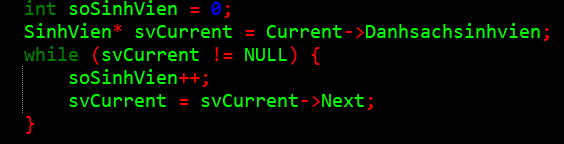
* + Khai báo con trỏ Current trỏ đến đầu danh sách liên kết Phong.



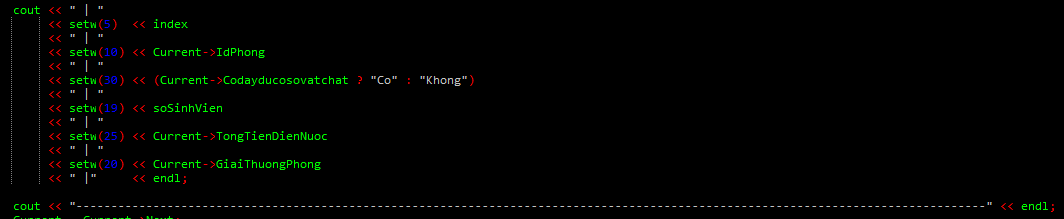
* + Khai báo biến index để giữ số thứ tự (STT) cho từng phòng.



* + Lặp qua từng phòng trong danh sách liên kết, dừng khi Current trỏ đến NULL(Kết thúc danh sách).
    1. Tính số lượng sinh viên trong phòng:



* + - * Khởi tạo soSinhVien với giá trị 0 để đếm số sinh viên trong phòng.
      * svCurrent trỏ đến danh sách sinh viên trong phòng (Danhsachsinhvien).
      * Vòng lặp while duyệt qua danh sách sinh viên, tăng soSinhVien lên 1 với mỗi sinh viên cho đến khi svCurrent là NULL.
    1. In thông tin từng phòng:



* + - * In ra dòng thông tin của phòng hiện tại với các cột:

1. Index: Số thứ tự.
2. Current -> IdPhong: Mã ID của phòng.
3. Current -> Codayducosovatchat : Trạng thái cơ sở vật chất (hiển thị "Co" nếu đúng và "Khong" nếu sai).
4. soSinhVien: Số lượng sinh viên trong phòng.
5. Current -> TongTienDienNuoc : Tổng tiền điện nước của phòng.
6. Current -> GiaiThuongPhong: Giải thưởng của phòng.



* + Di chuyển con trỏ Current đến phòng tiếp theo trong danh sách.



* + Tăng index lên 1 để cập nhật số thứ tự cho phòng tiếp theo.
* Hàm Indanhsachphong duyệt qua danh sách phòng, đếm số sinh viên của từng phòng, và in thông tin phòng ra màn hình theo dạng bảng, giúp hiển thị dữ liệu theo cách trực quan và dễ đọc.
  1. Hàm tìm sinh viên trong ký túc xá theo mã số sinh viên:
     1. Khai báo hàm chức năng:



* + - * Khai báo hàm TimSinhVienTheoMSSV nhận hai tham số:
      * SinhVien\* Head: Con trỏ Head trỏ ddesnd đầu danh sách liên kết SinhVien.
      * Const string& MSSV: Một chuỗi MSSV là mã số sinh viên cần tìm. Const string& được dùng để tránh tạo bản sao và ngăn sửa đổi biến này.



* + - * Khởi tạo con trỏ Current trỏ đến Head (đầu danh sách), dùng để duyệt qua từng phần tử (sinh viên) trong danh sách liên kết.



* + - * Vòng lặp While sẽ tiếp tục lặp qua danh sách cho đến khi Current trỏ đến NULL, tức là đã duyệt hết danh sách.



* + - * Kiểm tra xem mã số sinh viên của sinh viên hiện tại (Current -

>MSSV) có trùng với mã số cần tìm (MSSV) hay không.

* + - * Nếu có, điều này nghĩa là sinh viên với mã số đó đã được tìm thấy.
      * Nếu điều kiện if đúng, hàm trả về con trỏ Current, tức là con trỏ trỏ đến sinh viên có mã số trùng khớp.
      * Việc return ngay trong vòng lặp cũng dừng hàm ngay lập tức, không tiếp tục duyệt qua danh sách nữa vì đã tìm thấy sinh viên.



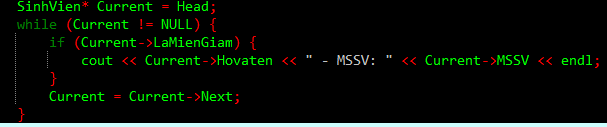
* + - * Nếu vòng lặp kết thúc mà không tìm thấy sinh viên nào có mã số trùng khớp với MSSV, hàm sẽ trả về NULL để báo hiệu sinh viên đó không tồn tại trong danh sách.
* Hàm TimSinhVienTheoMSSV duyệt qua danh sách sinh viên để tìm sinh viên có mã số phù hợp. Khi tìm thấy sinh viên, hàm trả về con trỏ trỏ đến sinh viên đó; nếu không, hàm trả về NULL để biểu thị sinh viên không tồn tại.
  1. Hàm in danh sách sinh viên được miễn giảm tiền ký túc xá:
     1. Khai báo hàm chức năng:

;

* + - * Khai báo hàm InDanhSachSinhVienMienGiam nhận tham số SinhVien\* Head, là con trỏ trỏ đến đầu danh sách liên kết SinhVien.
      * Hàm có kiểu void.



* + - * In ra tiêu đề "Danh sách sinh viên được miễn giảm" để hiển thị rõ danh sách những sinh viên được miễn giảm.



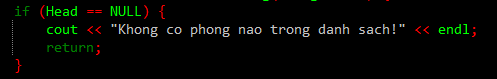
* + - * Khởi tạo con trỏ Current trỏ đến đầu danh sách sinh viên (Head), dùng để duyệt qua từng sinh viên trong danh sách liên kết.
      * Vòng lặp While sẽ tiếp tục chạy cho đến khi Current là NULL, tức là đã duyệt qua hết danh sách.
      * Kiểm tra thuộc tính LaMienGiam của sinh viên hiện tại. Nếu LaMienGiam trả về true, nghĩa là sinh viên này được miễn giảm, thì khối lệnh if sẽ thực hiện.
      * In ra tên (Hovaten) và mã số sinh viên (MSSV) của sinh viên hiện tại được miễn giảm, theo định dạng: "<Tên> - MSSV:

<MSSV>".

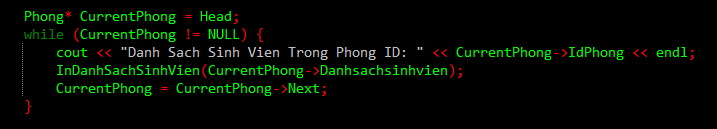
* + - * Di chuyển con trỏ Current đến sinh viên tiếp theo trong danh sách để tiếp tục duyệt qua danh sách liên kết.
* Hàm InDanhSachSinhVienMienGiam duyệt qua danh sách sinh viên và in ra tên và mã số sinh viên của những sinh viên có thuộc tính LaMienGiam là true.
  1. Hàm in ra danh sách sinh viên của tất cả phòng:



* Khai báo hàm InDanhSachSinhVienCuaTatCaPhong nhận tham số Phong\* Head, là con trỏ trỏ đến đầu danh sách liên kết các phòng (Phong).
* Hàm có kiểu void.



* Kiểm tra xem danh sách các phòng có rỗng không. Nếu Head là NULL, nghĩa là không có phòng nào trong danh sách.
* Nếu danh sách phòng trống (Head == NULL), hàm sẽ in thông báo "Khong co phong nao trong danh sach!" để người dùng biết rằng không có phòng nào để in danh sách sinh viên.
* Sau khi in thông báo, return sẽ thoát khỏi hàm ngay lập tức nếu không có phòng nào trong danh sách.



* Khởi tạo con trỏ CurrentPhong trỏ đến Head (đầu danh sách phòng), dùng để duyệt qua từng phòng trong danh sách liên kết.
* Vòng lặp While tiếp tục lặp qua danh sách phòng cho đến khi CurrentPhong là NULL, tức là đã duyệt qua tất cả các phòng.
* In ra thông báo "Danh sách sinh viên trong phòng ID:

<IdPhong>" để hiển thị ID của phòng hiện tại (Current -> IdPhong) và cho biết rằng danh sách sinh viên của phòng này sẽ được in ra tiếp theo.

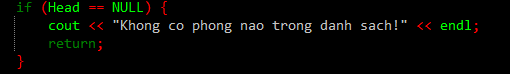
* Gọi hàm InDanhSachSinhVien với tham số là CurrentPhong-

>Danhsachsinhvien, danh sách sinh viên của phòng hiện tại. Hàm InDanhSachSinhVien sẽ in danh sách sinh viên của phòng đó.

* Di chuyển CurrentPhong đến phòng tiếp theo trong danh sách liên kết (CurrentPhong->Next) để tiếp tục duyệt qua các phòng.
* Hàm InDanhSachSinhVienCuaTatCaPhong duyệt qua từng phòng trong danh sách liên kết phòng và in danh sách sinh viên của mỗi phòng. Nếu danh sách phòng trống, hàm sẽ in thông báo và thoát.
  1. Hàm in ra danh sách các sinh viên được miễn giảm của tất cả các phòng:
     1. Khai báo hàm chức năng:



* + - * .Khai báo hàm void mang tên InDanhSachSinhVienMienGiamCuaTatCaPhong.



* + - * hàm kiểm tra xem Head (điểm bắt đầu của danh sách các phòng) có phải là NULL không. Nếu Head là NULL, điều này có nghĩa là danh sách phòng trống và hàm sẽ in ra thông báo "Khong co phong nao trong danh sach!" rồi thoát.



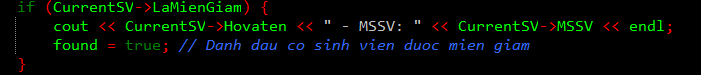
* + - * Khởi tạo biến với giá trị false, dùng để kiểm tra xem có sinh viên nào được miễn giảm trong danh sách hay không.



* CurrentPhong là con trỏ giúp duyêt qua từng phòng trong danh sách. Vòng lặp while tiếp tục chạy đến khi CurrentPhong bằng NULL thì dừng.



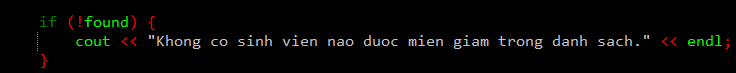
* CurrentSV là con trỏ trỏ đến sinh viên đầu tiên trong danh sách sinh viên của phòng hiện tại. Vòng lặp while tiếp tục chạy đến khi CurrentSV bằng NULL thì dừng.
* Khi duyệt đến một phòng, chương trình sẽ sin ra tiêu đề phòng bằng cách hiển thị ID phòng(mã ID của phòng) để chỉ định danh sách sinh viên được miễn giảm trong phòng đó.



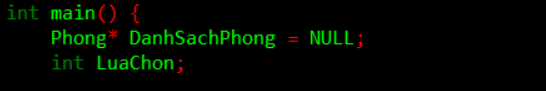
* Trong vòng lặp, hàm kiểm tra nếu LaMienGiam của sinh viên là true, sinh viên đó sẽ được in ra với tên và mã số sinh viên(MSSV). Đồng thời, found được đặt thành true để báo rằng có ít nhất một sinh viên được miễn giảm.



* Sau khi xử lý một sinh viên, con trỏ CurrentSV sẽ chuyển sang sinh viên kế tiếp trong danh sách.
* Kết thúc vòng lặp duyệt sinh viên, con trỏ CurrentPhong sẽ chuyển sang phòng kế tiếp.



* Sau khi đã duyệt qua tất cả các phòng, nếu không có sinh viên nào được miễn giảm(found trả về false), hàm sẽ in ra thông báo"Khong co sinh vien nao duoc mien giam trong danh sach".
* Hàm này đi qua từng phòng trong danh sách, kiểm tra và in ra các sinh viên được miễn giảm tiền phòng cho mỗi phòng.
  1. Hoạt động của chương trình
     1. Khởi tạo
        + Danh sách phòng ban đầu là rỗng



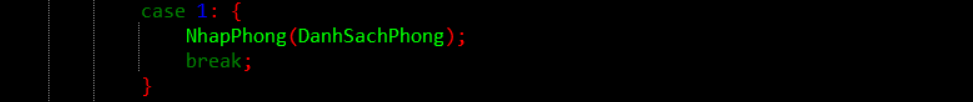
Giao diện khi phòng rỗng.

* + 1. Vòng lặp menu
       - Hiển thị menu cho người dùng
       - Nhập lựa chọn cho người dùng
       - Thực hiện chức năng tương ứng với lựa chọn



Giao diện của tất cả lựa chọn.

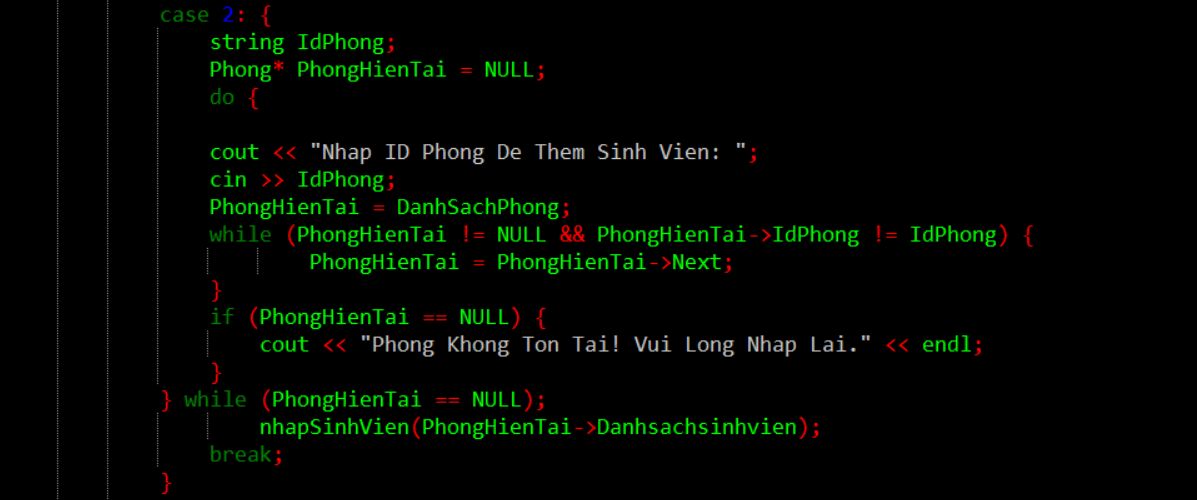
* Với tất cả lựa chọn trong menu giúp người dung dễ dàng nhập mong muốn quản lí theo ý mình.
* Và đây là những lựa chọn đã được nghiên cứu thực tế với nhu cầu sử dụng cho việc quản lí của người dùng chương trình này.
  + 1. Giải thích chi tiết từng lựa chọn
       - Nhập phòng



Giao diện của chương trình ở lựa chọn 1.

1. Khi người dùng chọn tùy chọn 1( có giá trị là 1), chương trình sẽ gọi hàm NhapPhong để thực hiện việc nhập thông tin của một phòng mới vào danh sách các phòng.
   * Với yêu cầu người dùng nhập các thông tin cần thiếtcủa một phòng mới (ví dụ: mã phòng, diện tích, số lượng, người tối đa,…)
   * Thêm cấu trúc dữ liệu vừa tạo vào danh sách các phòng (được biểu diễn bởi con trỏ DanhSachPhong).

* Nhập sinh viên vào phòng
  + Các bước hoạt động:
* **Khai báo biến:** Khai báo các biến cần thiết để lưu trữ ID phòng và con trỏ đến phòng hiện tại.
* **Vòng lặp do-while:** Vòng lặp này sẽ lặp đi lặp lại cho đến khi tìm thấy phòng có ID trùng khớp.
* **Nhập ID phòng:** Yêu cầu người dùng nhập ID phòng.
* **Tìm kiếm phòng:** Duyệt qua danh sách các phòng để tìm phòng có ID trùng khớp.
* **Kiểm tra kết quả:** Nếu không tìm thấy phòng, in ra thông báo lỗi và yêu cầu nhập lại.
* **Thêm sinh viên:** Khi tìm thấy phòng, gọi hàm nhapSinhVien để thêm sinh viên vào danh sách sinh viên của phòng đó.

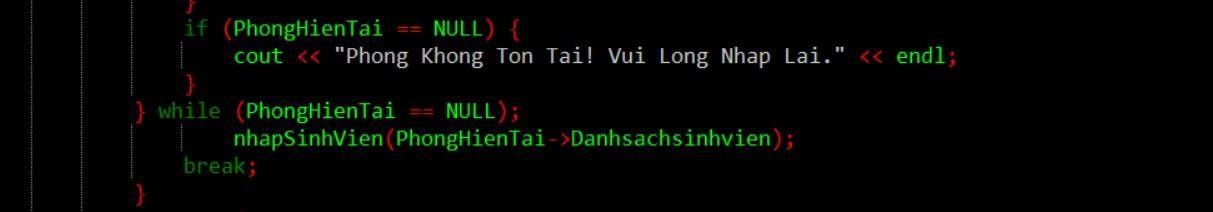


Giao diện của code khi nhập sinh viên vào phòng.

* Giải thích chi tiết từng dòng code

Giao diện của đoạn code

* Trước tiên khai báo một biến chuỗi để lưu trữ ID phòng mà người dùng nhập vào.
* Khai báo một con trỏ PhongHienTai và khởi tạo nó bằng NULL. Con trỏ này sẽ được dùng để duyệt qua danh sách các phòng.
* In ra màn hình yêu cầu người dùng nhập ID phòng.
* Đọc ID phòng từ bàn phím và gán vào biến IdPhong.
* Khởi tạo lại PhongHienTai bằng con trỏ đầu danh sách phòng. Điều này cần thiết cho mỗi vòng lặp để bắt đầu tìm kiếm từ đầu danh sách.
* Vòng lặp này sẽ duyệt qua danh sách các phòng cho đến khi: Tìm thấy phòng có ID trùng khớp. Đến cuối danh sách (PhongHienTai trở thành NULL).
* Di chuyển con trỏ đến phòng tiếp theo.



* Nếu vòng lặp kết thúc mà không tìm thấy phòng, in ra thông báo lỗi.
* Nếu lỗi thì phòng đó không tồn tại và yêu cầu người dung nhập lại.
* Tiếp tục vòng lặp cho đến khi tìm thấy phòng.
* Gọi hàm nhapSinhVien để thêm sinh viên vào danh sách sinh viên của phòng vừa tìm được. PhongHienTai-

>Danhsachsinhvien trỏ đến danh sách sinh viên của phòng hiện tại. Và vòng lặp kết thúc.

* Đoạn code này thực hiện việc tìm kiếm một phòng trong một danh sách các phòng dựa trên ID phòng mà người dùng nhập vào. Sau khi tìm thấy phòng, nó sẽ gọi hàm nhapSinhVien để thêm một sinh viên mới vào danh sách sinh viên của phòng đó.
  1. Nhập sinh viên tại vị trí bất kì.
* Đoạn code này được chia làm 3 phần chính
  + 1. Tìm kiếm phòng
       - **string IdPhong;:** Khai báo biến chuỗi để lưu trữ ID phòng người dùng nhập.
       - **Phong\* PhongHienTai = NULL;:** Khai báo con trỏ PhongHienTai để lưu trữ địa chỉ của phòng hiện tại đang xét. Khởi tạo bằng NULL để bắt đầu tìm kiếm từ đầu danh sách.
       - **do-while:** Vòng lặp này sẽ thực hiện cho đến khi tìm thấy phòng có ID trùng khớp.
       - **Nhập ID phòng:** Yêu cầu người dùng nhập ID phòng.
       - **Tìm kiếm phòng:** Duyệt qua danh sách các phòng để tìm phòng có ID trùng khớp.
       - **Kiểm tra kết quả:** Nếu không tìm thấy phòng, in ra thông báo lỗi và yêu cầu nhập lại.



Giao diện của phần code tìm kiếm phòng.

* + 1. Nhập thông tin sinh viên mới
       - **SinhVien SinhVienMoi;:** Khai báo một đối tượng SinhVien để lưu trữ thông tin sinh viên mới.
       - **Nhập thông tin:** Dùng cin và getline để nhập các thông tin

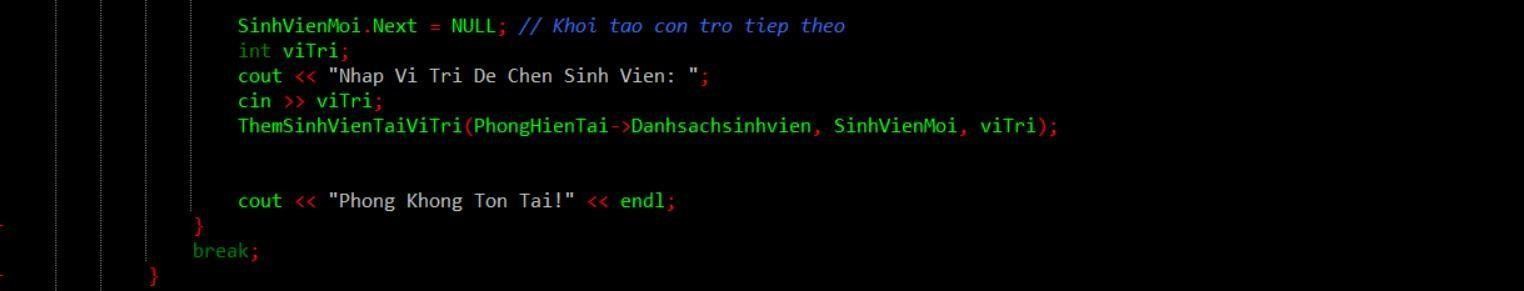
của sinh viên như MSSV, họ tên, lớp, quê quán, học lực, ngày bắt đầu, hoàn cảnh, và tính toán cờ hiệu miễn giảm.

* + - * **Khởi tạo Next:** Gán NULL cho Next của sinh viên mới để đánh dấu cuối danh sách.

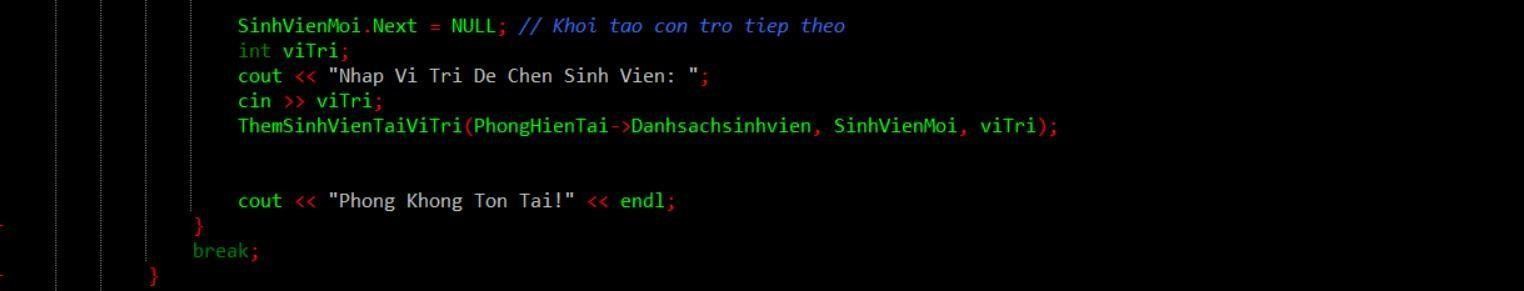


Giao diện đoạn code của nhập thông tin sinh viên mới.

* + 1. Chèn sinh viên vào vị trí
       - **Nhập vị trí:** Yêu cầu người dùng nhập vị trí muốn chèn sinh viên.
       - **Gọi hàm ThemSinhVienTaiViTri:** Gọi hàm này để thực hiện việc chèn sinh viên vào vị trí đã chọn trong danh sách sinh viên của phòng. Hàm này sẽ có logic để chèn sinh viên vào đúng vị trí trong danh sách liên kết.

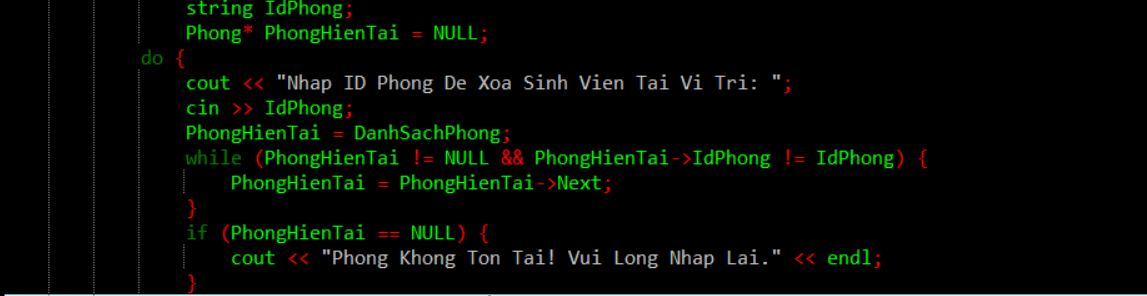


Giao diện của đoạn code tìm kiếm phòng. Giao diện của đoạn code chèn sinh viên vào ví trí mong muốn.

* Đoạn code này thực hiện một chuỗi các thao tác: tìm kiếm phòng, nhập thông tin sinh viên và chèn sinh viên vào một vị trí cụ thể trong danh sách sinh viên của phòng đó. Nó sử dụng danh sách liên kết để quản lý danh sách sinh viên.
  1. Xóa sinh viên tại vị trí bất kì

Đoạn code này được chia làm 3 phần chính

* + 1. Tìm kiếm phòng
       - **string IdPhong;:** Khai báo biến chuỗi để lưu trữ ID phòng người dùng nhập.
       - **Phong\* PhongHienTai = NULL;:** Khai báo con trỏ PhongHienTai để lưu trữ địa chỉ của phòng hiện tại đang xét. Khởi tạo bằng NULL để bắt đầu tìm kiếm từ đầu danh sách.
       - **do-while:** Vòng lặp này sẽ thực hiện cho đến khi tìm thấy phòng có ID trùng khớp.
* **Nhập ID phòng:** Yêu cầu người dùng nhập ID phòng.
* **Tìm kiếm phòng:** Duyệt qua danh sách các phòng để tìm phòng có ID trùng khớp.
* **Kiểm tra kết quả:** Nếu không tìm thấy phòng, in ra thông báo lỗi và yêu cầu nhập lại



Giao diện của đoạn code tìm kiếm phòng.

* + 1. Nhập vị trí cần xóa
       - **int viTri;:** Khai báo biến nguyên để lưu trữ vị trí mà người dùng muốn xóa.
       - **Nhập vị trí:** Yêu cầu người dùng nhập vị trí cần xóa.

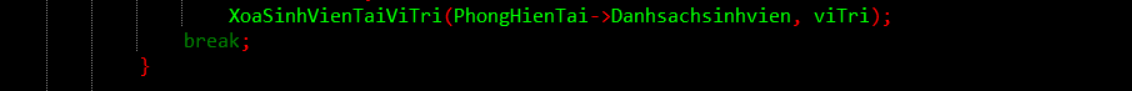


Giao diện của đoạn code mà người dùng cần xóa.

* + 1. Xóa sinh viên tại vị trí

## XoaSinhVienTaiViTri(PhongHienTai-

**>Danhsachsinhvien, viTri);:** Gọi hàm XoaSinhVienTaiViTri để thực hiện việc xóa sinh viên tại vị trí đã chọn trong danh sách sinh viên của phòng.



* 1. In danh sách sinh viên trong phòng
* Đoạn code này sẽ tìm kiếm một phòng theo ID và in ra danh sách tất cả sinh viên trong phòng đó.
  + 1. Tìm kiếm phòng:
       - Phần đầu của đoạn code (từ string IdPhong; đến } while
       - PhongHienTai == NULL);) hoàn toàn giống như các case trước. Nó dùng để tìm kiếm phòng theo ID và đảm bảo rằng phòng tồn tại trước khi thực hiện các thao tác tiếp theo.

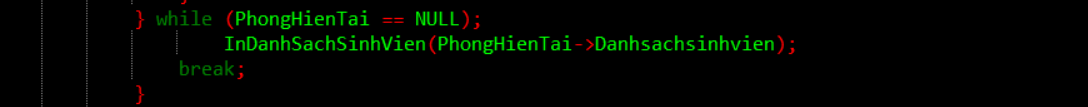


Giao diện tìm kiếm phòng.

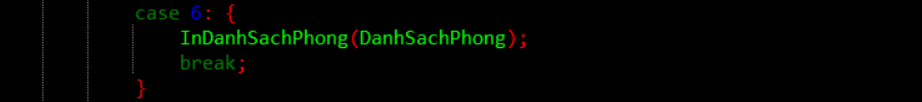
* + 1. In danh sách sinh viên

## InDanhSachSinhVien(PhongHienTai-

**>Danhsachsinhvien);:** Gọi hàm InDanhSachSinhVien để in danh sách sinh viên của phòng hiện tại. Hàm này sẽ nhận vào một con trỏ đến đầu danh sách sinh viên của phòng và thực hiện việc duyệt qua danh sách và in thông tin của từng sinh viên ra màn hình.

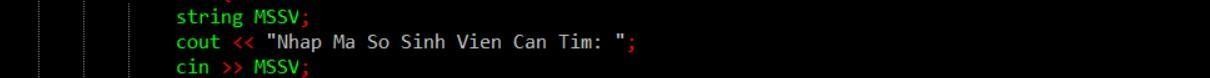


* 1. In danh sách các phòng
* Hàm này giúp người dùng có thể tìm kiếm thông tin của các phòng một cách đầy đủ và chi tiết của các phòng.
* Cung cấp chp người dung những thông tin các phòng trong kí túc xá.
* Dưới đây là đoạn code giúp người dung in ra các thông tin của các phòng.



Giao diện của đoạn code in thông tin của các phòng.

* 1. Tìm sinh viên cần tìm theo MSSV
     1. Nhập MSSV:
        + **string MSSV;:** Khai báo một biến chuỗi để lưu trữ mã số sinh viên mà người dùng nhập vào.
* **cout << "Nhap Ma So Sinh Vien Can Tim: ";:** In ra màn hình yêu cầu người dùng nhập mã số sinh viên.
* **cin >> MSSV;:** Đọc mã số sinh viên từ bàn phím và gán vào biến MSSV.



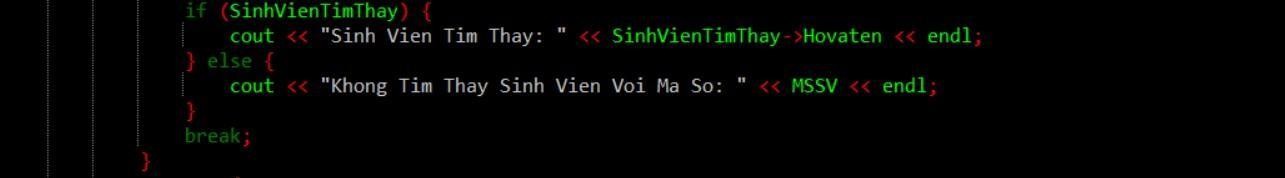
Giao diện đoạn code khi nhập mssv.

* + 1. Tìm kiếm sinh viên:

SinhVien\* SinhVienTimThay = TimSinhVienTheoMSSV(DanhSachPhong-

**>Danhsachsinhvien, MSSV);:** Gọi hàm TimSinhVienTheoMSSV để tìm kiếm sinh viên có mã số trùng khớp với MSSV trong danh sách sinh viên của phòng đầu tiên. Hàm này sẽ trả về con trỏ đến sinh viên được tìm thấy hoặc NULL nếu không tìm thấy.



* + - * Hiển thị kết quả:
* **if (SinhVienTimThay):** Kiểm tra xem có tìm thấy sinh viên hay không.
* **Nếu tìm thấy:** In ra họ và tên của sinh viên đó.
* **Nếu không tìm thấy:** In ra thông báo không tìm thấy sinh viên.

Giao diện kết quả tìm kiếm sinh viên theo mssv.

* + Đoạn code này thực hiện việc tìm kiếm một sinh viên cụ thể trong danh sách sinh viên của một phòng dựa trên mã số sinh viên. Nó sử dụng danh sách liên kết để quản lý danh sách sinh viên và hàm TimSinhVienTheoMSSV để thực hiện việc tìm kiếm.
  1. In danh sách sinh viên được miễn giảm
     1. Tìm kiếm phòng

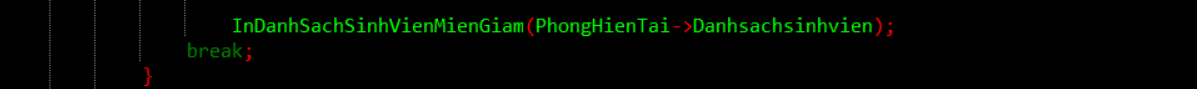
 Phần đầu của đoạn code (từ string IdPhong; đến } while (PhongHienTai

== NULL);) hoàn toàn giống như các case trước. Nó dùng để tìm kiếm phòng theo ID và đảm bảo rằng phòng tồn tại trước khi thực hiện các thao tác tiếp theo.



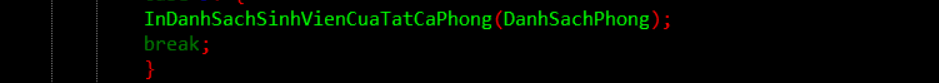
* + - * In danh sách sinh viên được miễn giảm: InDanhSachSinhVienMienGiam(PhongHienTai-

>Danhsachsinhvien);: Gọi hàm InDanhSachSinhVienMienGiam để in danh sách sinh viên được miễn giảm của phòng hiện tại. Hàm này sẽ nhận vào một con trỏ đến đầu danh sách sinh viên của phòng và thực hiện việc duyệt qua danh sách, kiểm tra điều kiện miễn giảm và in thông tin của các sinh viên đủ điều kiện.

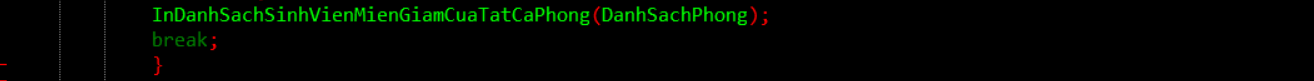


Giao diện đoạn code in sinh viên được miễn giảm

* Đoạn code này thực hiện việc tìm kiếm một phòng và in ra danh sách các sinh viên trong phòng đó được hưởng ưu đãi miễn giảm tiền phòng. Nó cho thấy cách sử dụng cờ hiệu để đánh dấu các đối tượng đặc biệt (trong trường hợp này là sinh viên được miễn giảm) và cách lọc dữ liệu dựa trên các điều kiện này.
  1. In danh sách sinh viên của tất cả các phòng
* **Gọi hàm:** Dòng lệnh này gọi hàm InDanhSachSinhVienCuaTatCaPhong với tham số là DanhSachPhong.
* **DanhSachPhong:** Đây là con trỏ đến phòng đầu tiên trong danh sách các phòng. Nó là điểm bắt đầu để duyệt qua toàn bộ danh sách phòng.
* **Chức năng của hàm:** Hàm InDanhSachSinhVienCuaTatCaPhong sẽ thực hiện việc duyệt qua toàn bộ danh sách phòng, với mỗi phòng sẽ gọi hàm InDanhSachSinhVien để in ra danh sách sinh viên của phòng đó.



* 1. In danh sách sinh viên được miễn giảm ở tất cả các phòng
* In danh sách sinh viên ở tất cả các phòng là một mở rộng , cho phép chúng ta có một cái nhìn bao quát hơn về tình hình miễn giảm tiền phòng trong toàn bộ hệ thống. Bằng cách duyệt qua tất cả các phòng và lọc ra những sinh viên đủ điều kiện miễn giảm, chúng ta có thể dễ dàng quản lý và theo dõi các trường hợp này.
* Giúp người dùng tìm kiếm thông tin của các sinh viên được miễn giảm được chi tiết và nhanh chóng hơn.



Giao diện đoạn code của in danh sách sinh viên được miễn giảm ở các phòng.

* 1. Xóa phòng



Dòng 1: string IdPhong;

* + - **Khai báo biến:** Tạo một biến chuỗi tên là IdPhong để lưu trữ mã số của phòng cần xóa. Biến này sẽ được sử dụng để tìm kiếm và xóa phòng trong danh sách.

Dòng 2: cout << "Nhap ID phong can xoa: ";

* + - **In ra màn hình:** Hiển thị một thông báo yêu cầu người dùng nhập vào mã số của phòng cần xóa. Thông báo này sẽ xuất hiện trên màn hình console để người dùng biết cần nhập gì.

Dòng 3: cin >> IdPhong;

* + - **Nhập dữ liệu từ bàn phím:** Đọc giá trị mà người dùng nhập vào từ bàn phím và gán vào biến IdPhong. Dữ liệu này sẽ là mã số phòng mà người dùng muốn xóa.

Dòng 4: XoaPhong(DanhSachPhong, IdPhong);

* + - **Gọi hàm:** Gọi hàm XoaPhong để thực hiện việc xóa phòng. Hàm này sẽ nhận vào hai đối số:
      * DanhSachPhong: Con trỏ đến đầu danh sách các phòng. Đây là nơi lưu trữ thông tin của tất cả các phòng trong hệ thống.
      * IdPhong: Mã số phòng cần xóa, được nhập vào bởi người dùng ở dòng 3.
      * Chức năng của hàm XoaPhong:
      * **Tìm kiếm phòng:** Hàm này sẽ duyệt qua danh sách các phòng, so sánh mã số của từng phòng với giá trị IdPhong đã nhập.
      * **Xóa phòng:** Khi tìm thấy phòng cần xóa, hàm sẽ thực hiện các thao tác để loại bỏ phòng đó khỏi danh sách và giải phóng bộ nhớ được cấp phát cho phòng đó.
      * **Cập nhật danh sách:** Sau khi xóa phòng, danh sách các phòng sẽ được cập nhật lại để đảm bảo tính nhất quán.

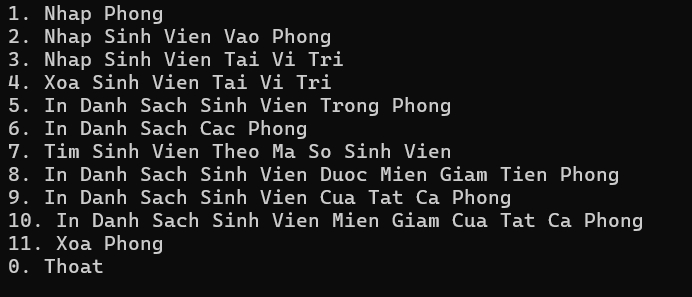
Dòng 5: break;

* + - **Kết thúc case:** Dòng lệnh này kết thúc việc thực hiện case 11. Sau khi xóa phòng xong, chương trình sẽ chuyển sang thực hiện các case khác nếu có.

# CHƯƠNG 3: GIAO DIỆN CHƯƠNG TRÌNH VÀ CÁCH SỬ DỤNG.

* 1. Kết quả đạt được.

Sau khi hoàn thành đoạn code ta in ra kết quả đạt được sẽ hiện ra tổng 12 chức năng mà người quản lí KTX cần phải có để quản lí các phòng và các sinh viên trong phòng:



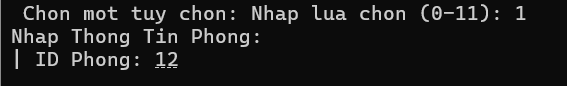
*Giao diện tất cả các chức năng của quản lí KTX*

* + 1. Chức năng nhập Phòng
       - Để quản lí tốt một KTX ta cần chia ra nhiều phòng tiện trong việc quản lí những yêu cầu mà phòng đó cần có hoặc để kiểm tra đánh giá chất lượng hay tính tiền điện nước trong phòng:



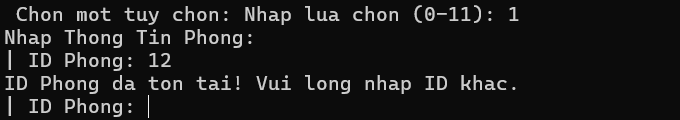
*Giao diện chức năng Nhập Phòng*

* Khi click vào 1.Nhap Phong kết qủa sẽ yêu cầu ta tạo một phòng mới để lưu lại và thêm các yêu tố cần thiết vào đó:



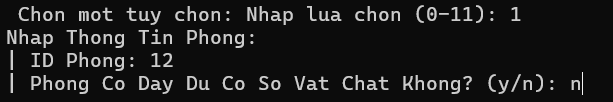
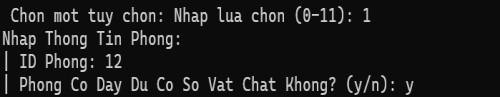
*Giao diện yêu cầu tạo một phòng mới*

* Sau khi tạo Phòng đầu tiên và nhập các thông tin yêu cầu của chức năng Nhap Phong, nếu muốn tạo phòng tiếp theo thì phải nhập khác kí tự của phòng tạo trước đó nếu nhập sai code sẽ yêu cầu bạn nhập lại một ID khác:



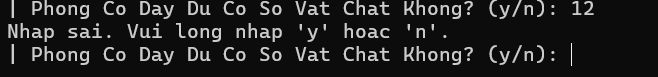
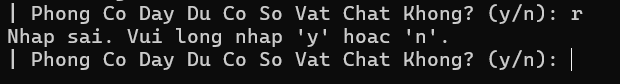
*Code thông báo nhập lỗi và yêu cầu nhập lại*

* Kế tiếp bước tạo ID phòng mới thì đến thông báo kiểm tra phòng đó đã đầy đủ những cơ sở vật chất sinh hoạt chung của sinh viên hay chưa (BQL KTX phải đi kiểm tra khảo sát) nếu đã đầy đủ thì nhập ‘y’ và ngược lại nếu không đủ thì ta nhập ‘n’ để BQL xem xét thêm các đồ dùng sinh hoạt chung nếu thiếu:



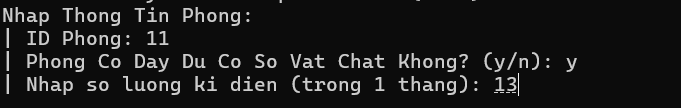
Nhập ‘y’ Nhập ‘n’

* Ở đây nếu nhập một kí tự chữ không phải là ‘y’ hoặc ‘n’ hay nhập một số thì code sẽ yêu cầu nhập lại cho đúng định dạng yêu cầu để BQL nắm rõ tình hình của phòng:

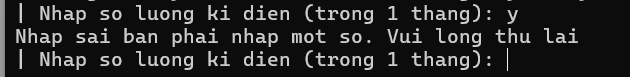


Thông báo nhập sai khi nhập kí tự chữ khác Thông báo nhập sai khi nhập kí tự số

* Bước tiếp ta cần nhập thông tin số kí điện sử dụng trong 1 tháng để mỗi phòng phải chi trả lại số tiền tương ứng với giá 2.000VNĐ/1KW cho BQL:



Giao diện yêu cầu nhập số kí điện

* Tại đây phải nhập một kí tự số để biểu hiện rõ số kí điện sử dụng, để BQL tính toán số tiền điện, nếu nhập kí tự chữ code sẽ thông báo sai và yêu cầu nhập lại:

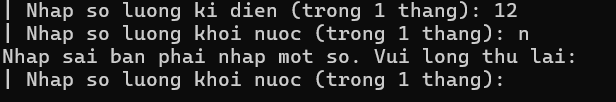
Thông báo nhập sai yêu cầu nhập lại kí tự chữ

* Sau khi nhập số kí điện thì ta phải nhập số khối nước mà phòng đã sử dụng trong 1 tháng với giá 8.000VNĐ/1m3 :

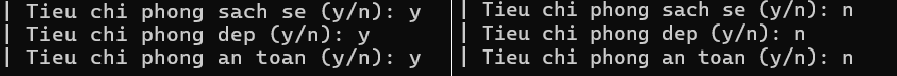


Giao diện yêu cầu nhập số khối nước

* Cũng như ở số kí điện, nhập số khối nước ta cũng phải nhập kí tự số nếu nhập sai code sẽ yêu cầu nhập lại:

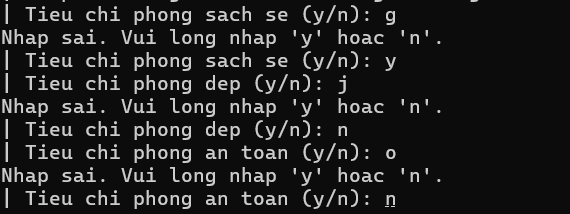


* Ngay sau đó là những thông tin mà BLQ đi kiểm tra về vệ sinh, cách bố trí và độ an toàn của mỗi mỗi phòng mà KTX đang có để đó làm cơ sở kiểm tra hoặc điều kiện một số cuộc thi mà KTX tổ chức. Ta phải nhập ‘y’ nếu đạt tiêu chí của BQL và ngược lại nhập ‘n’:



Giao diện các tiêu chí của phòng

* Các tiêu chí này cũng phải nhập đúng kí tự ‘y’ hoặc ‘n’ nếu nhập sai code thông báo lỗi và yêu cầu phải nhập lại:



Giao diện code thông báo nhập lỗi và yêu cầu nhập lại

Và đó cũng đã kết thúc chức năng đầu tiên của bài code là tạo 1 phòng với những yêu cầu nhập từ code để quá trình quản lí của BQL tốt hơn và hoàn thiện hơn đối với mỗi phòng.

* + 1. Chức năng thêm sinh viên vào phòng đã tạo.
       - Để quản lí các sinh viên trong phòng phải nhập các thông tin cơ bản của sinh viên vào phòng để thuận tiện hơn



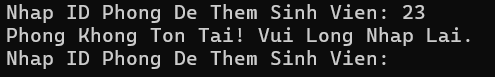
Giao diện chức năng thêm sinh viên vào phòng

* Code sẽ yêu cầu nhập ID phòng để thêm sinh viên vào phòng, ở đây các bạn phải nhập đúng một trong các ID đã tạo trước đó:

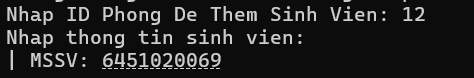


Giao diện yêu cầu nhập ID phòng

* Nếu nhập một Phòng chưa tạo trước đó, thì code sẽ thông báo phòng không tồn tại và phải nhập lại:

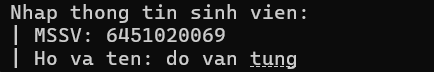


* Tiếp đến code sẽ yêu cầu nhập mã số sinh viên vào phòng và phải nhập 10 số:



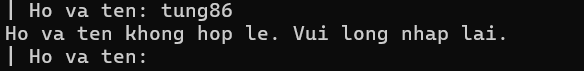
Giao diện khi nhập Mã Số Sinh Viên

* Mã số sinh viên phải nhập 10 số và nếu thêm một
* Sau MSSV thì ta cần phải thêm tên của sinh viên đó:



Giao diện nhập tên sinh viên

* Tại họ và tên sinh viên yêu cầu phải chỉ nhập kí tự chữ và nhập sai yêu cầu nhập lại:

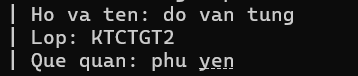


Thông báo nhập sai và yêu cầu nhập lại

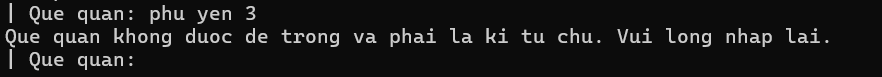
* Tiếp đến thì cần phải nhập lớp của sinh viên:



Giao diện nhập lớp sinh viên

* Sau đó ta nhập quê quán của sinh viên để BQL nắm rõ hơn về thông tin của sinh viên:

Giao diện yêu cầu nhập code

* Tại đây quê quán không được nhập kí tự số, nếu sai đoạn code yêu cầu nhập lại:

Thông báo lỗi và yêu cầu nhập lại

* Ngay sau khi nhập quê quán sẽ đến yêu cầu nhâp học lực của sinh viên để xem xét điều kiện miễn giảm tiền phòng ở KTX và phải nhập đúng yêu cầu trong dấu ngoặc đơn ( xuất sắc/ giỏi / khác):



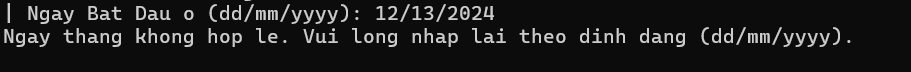
Giao diện nhập học gồm xuất sắc, giỏi và khác

* Tiếp đến code yêu cầu nhập ngày tháng năm mà sinh viên đó bắt đầu dọn vào ở trong phòng nhằm mục đích để BQL KTX dễ dàng kiểm tra và xem xét hợp đồng:



Giao diện ngày bắt đầu ở theo định dạng ngày tháng năm

* Nếu ở đây người dùng nhập sai ngày lớn hơn 31 và nhỏ hơn 30 kèm thêm tháng thì trong khoảng từ 1 đến 12:



Nhập ngày tháng năm sai định dạng và code yêu cầu nhập lại

* Để hoàn thành các tiêu chí để xem xét miễn giảm cho sinh viên thì sau đó code yêu cầu ta nhập một trong các mục trong ngoặc:



Giao diện nhập hoàn cảnh sinh viên

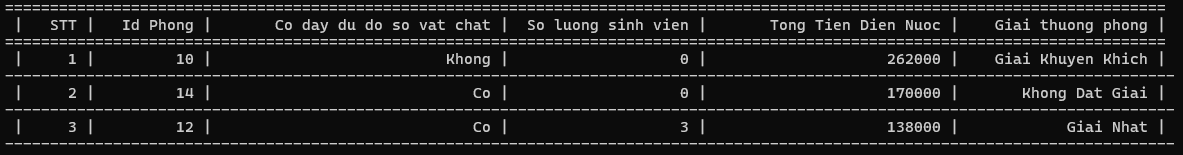
Như vậy đó cũng là kết thúc chức năng kế tiếp là nhập thông tin sinh viên vào phòng đã đầy đủ các thông tin cần thiết để BQL KXT nắm bắt được thông tin sinh viên mà

trong đó cũng để tìm hiểu được sinh viên làm cơ sở hỗ trợ sinh viên trong việc miễn giảm tiền KTX.

* + 1. Chức năng in danh sách Phòng.



Giao diện chức năng in danh sách các phòng

* + - * Để in ra các phòng đã tạo trước đó và kết quả cho ra gồm STT, Cơ sở vật chất của phòng, số lượng sinh viên trong phòng, tổng điện nước và giải thưởng của phòng dựa trên các tiêu chí mà BQL đã đi kiểm tra và nhập trước đó:

Giao diện in ra màn hình danh sách các phòng đã tạo

* + 1. Hàm in ra danh danh sách tất cả các sinh viên của KTX.



Giao diện chức năng in danh sách sinh tất cả các phòng

* + - * Nhằm để quản lí chặt chẽ hơn nữa tất cả các sinh viên hiện tại đang còn ở KTX hay để điểm danh hay tham gia các cuộc thi do KTX tổ chức:



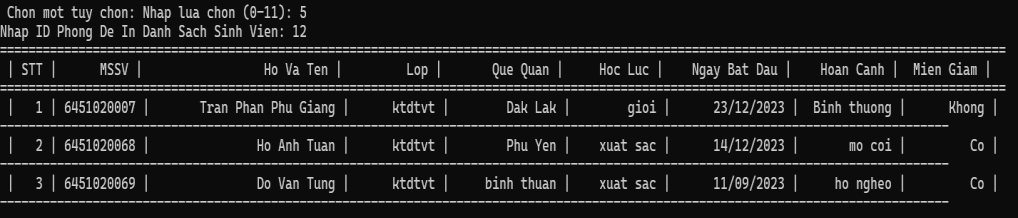
Giao diện khi in ra danh sách sinh viên tất cả các phòng

* + 1. In danh sách sinh viên trong một phòng.



Chức năng in danh sách sinh viên trong một phòng

* + - * Tại đây BQL có thể thấy rõ hơn vè những thông tin của từng sinh viên trong từng để dễ dàng kiểm soát các thành viên trong phòng hơn:



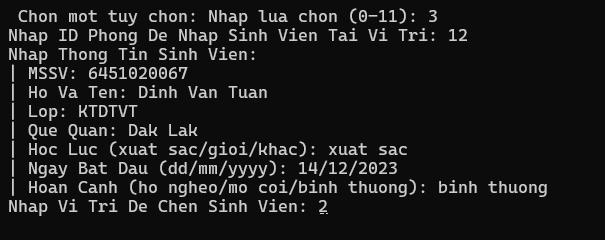
Giao diện in danh sách sinh viên phòng ID 12

* + 1. Chức năng thêm sinh viên vào vị trí bất kì.
       - Có thể để sắp xếp theo ý muốn của người sử dụng để chèn vào bất kì vị trí nào trong danh sách sinh viên của phòng:



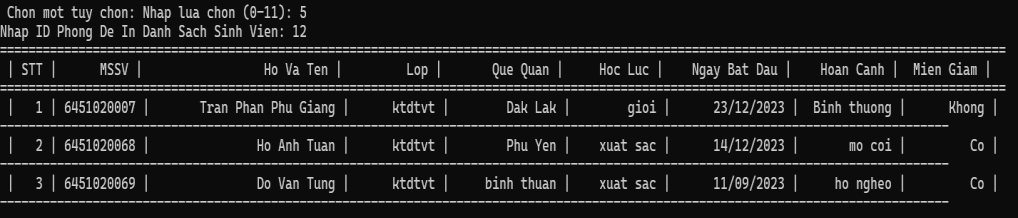
Giao diện chức năng nhập sinh viên tại vị trí

* Khi chọn thêm sinh viên vào vị trí bất kì nó cũng giống như thêm một sinh viên vào phòng đã tạo (3.1.2) và cuối chức năng code sẽ yêu cầu nhập vị trí mà người sử dụng muốn chèn vào trong danh sách:



Các thông tin cần nhập để thêm sinh viên vào phòng

* Trước và sau khi chèn vào vị trí:



Bảng danh sách sinh viên của P12 ban đầu



Bảng danh sách sinh viên của P12 sau khi chèn vào vị trí

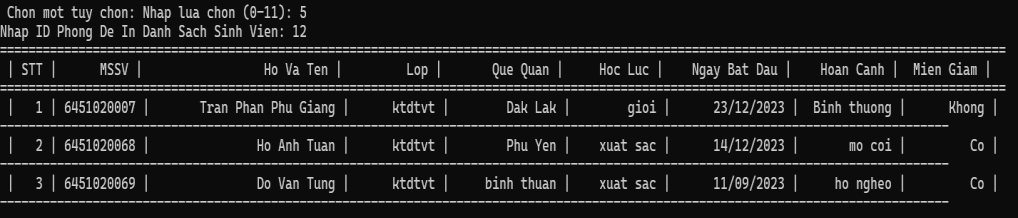
* + 1. Chức năng xóa sinh viên tại vị trí bất kì.
       - Cũng giống như thêm sinh viên vào vị trí bất kì (3.1.6) trong quá trình sử dụng ta có thể thêm nhầm sinh viên, nhập sai thông tin sinh viên hay sinh viên đó không còn ở KTX nữa. Khi có nhu cầu để để xóa 1 sinh viên:



* Trước và sau khi xóa sinh viên bất kì trong 1 phòng:



Trước khi xóa sinh viên



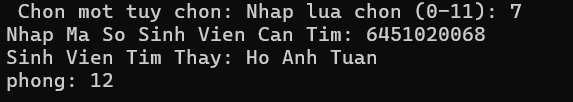
Sau khi xóa sinh viên

* + 1. Chức năng tìm sinh viên theo mã số sinh viên.
       - Khi có quá nhiều sinh viên ở KTX việc tra từng sinh viên để tìm ra sinh viên cần tìm rất tốn thời gian người tìm kiếm. Việc sử dụng chức năng tìm kiếm giúp cho người sử dụng dễ dàng hơn trong việc tìm ra 1 sinh viên:



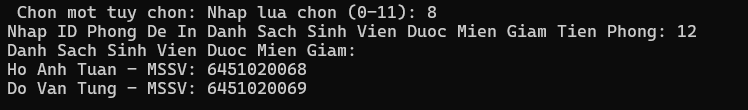
Giao diện Chức năng tìm sinh viên theo mã số sinh viên

* Khi sử dụng, Code yêu cầu người sử dụng phải nhập MSSV cần tìm kiếm thông tin:



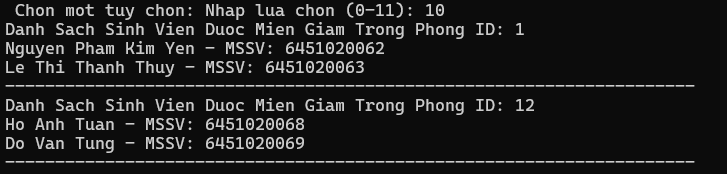
Kết quả in ra màn hình sau khi tìm kiếm

* + 1. Chức năng in danh sách sinh viên được miễn giảm ở một phòng.
       - Để kiểm tra xem phòng có bao nhiêu sinh viên được miễn giảm tiền phòng và gồm những ai:



* + 1. Chức năng in danh sách tất cả sinh viên được miễn giảm tiền phòng.

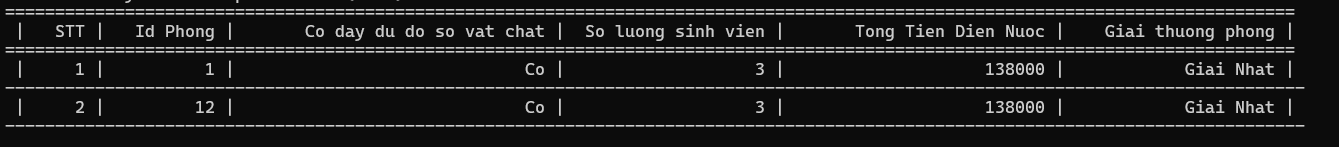
\* Cũng như (3.1.9 ) ở đây in ra tất cả các sinh viên đủ điều kiện miễn giảm trong KTX gồm họ tên, MSSV và phòng của sinh viên đó:



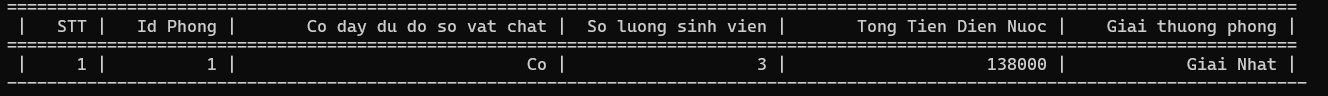
Giao diện khi in ra kết quả tìm kiếm ở toàn KTX

* + 1. Chức năng xóa phòng.

\* Một chức năng cần thiết trong quá trình quản lí yêu cầu có thể nhập sai những yêu cầu trong (3.1.1) hoặc phòng đó ban đầu có sinh viên ở nhưng giờ đã trống:



Danh sách sinh viên các phòng trước khi xóa



Danh sách sinh viên các phòng sau khi xóa

# TÀI LIỆU THAM KHẢO

[1]. Giáo trình cấu trúc dữ liệu và giải thuật, Đại học Bách Khoa Hà Nội <https://hust.nxbbachkhoa.vn/cau-truc-du-lieu-va-giai-thuat-b10776.html>

[2]. Tham khảo các video bài giảng trên các nền tảng internet, <https://blog.28tech.com.vn/ctdl-giai-thuat>

[3]. Tham khảo mảng học trên mạng https://youtu.be/V85332IZVs0?si=DIYD\_6XjcFUHP3yn