**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 SÁCH THƯ VIỆN BẰNG**

**DANH SÁCH LIÊN KẾT**

**Giảng viên hướng dẫn**: TRẦN THỊ DUNG

**Sinh viên thực hiện:**  NGUYỄN THANH THẾ – MSV: 6451020056

TRƯƠNG HỒNG THỨC – MSV: 6451020061

NGUYỄN VĂN TIẾN – MSV: 6451020062

**Lớp** **:** KỸ THUẬT ĐIỆN TỬ VIỄN THÔNG

**Khoá** **:** 64

Tp. Hồ Chí Minh, năm 2024

# LỜI CẢM ƠN

Chúng em xin cảm ơn chân thành đến cô Trần Thị Dung, giảnh viên của học phần "Lập trình cấu trúc dữ liệu và giải thuật", người đã tận tâm giảng dạy và hỗ trợ chúng em trong suốt quá trình học tập. Những kiến thức quý báu mà cô truyền đạt đã giúp chúng em xây dựng được nền tảng vững chắc trong việc thực hiện bài tập lớn này. Những lời góp ý và chỉ dẫn tận tình của cô đã giúp chúng em khắc phục những thiếu sót và hoàn thiện đề tài một cách tốt nhất. Chúng em sẽ luôn trân trọng và áp dụng những kiến thức này vào thực tiễn trong quá trình học tập và làm việc sau này. Nhờ sự hướng dẫn chu đáo và sự tận tụy của cô, chúng em đã có thể tiếp thu những kiến thức quan trọng về cấu trúc dữ liệu, giải thuật và lập trình. Những bài giảng của cô không chỉ giúp chúng em hoàn thành bài tập mà còn mang lại những bài học quý giá cho việc áp dụng những kiến thức này trong tương lai.

Trong quá trình thực hiện bài tập, chúng em đã gặp phải nhiều thử thách trong việc hiểu và áp dụng các khái niệm phức tạp. Tuy nhiên, với sự giúp đỡ tận tình của cô, chúng em đã từng bước vượt qua khó khăn, tiếp thu và áp dụng được những kiến thức này vào các bài tập thực tế. Cô không chỉ giúp chúng em nâng cao kĩ năng lập trình mà còn phát triển khả năng tử duy có hệ thống và giải quyết vấn đề một cách khoa học.

Chúng em cũng xin cảm ơn các bạn trong lớp đã luôn đồng hành, hỗ trợ và chia sẻ những ý tưởng sáng tạo trong suốt quá trình thực hiện đề tài. Sự hợp tác và tinh thần đoàn kết của các bạn đã giúp chúng em vượt qua những thử thách và khó khăn, từ đó hoàn thành bài tập lớn với kết quả tốt nhất. Những đóng góp của các bạn là nguồn động lực lớn với chúng em trong suốt quá trình thực hiện

Một lần nưa, chúng em xin gửi lời cảm ơn sâu sắc đến cô Trần Thị Dung. Những kiến thức và kỹ năng mà cô truyền đạt sẽ là hành trang quý báu giúp chúng em vững bước trong con đường học tập và phát triển sự nghiệp sau này. Chúng em xin kính chúc cô luôn mạnh khỏe, thành công và hạnh phúc trong công việc cũng như trong cuộc sống.

MỤC LỤC

[**LỜI CẢM ƠN** 2](#_Toc182171424)

[**CHƯƠNG 1. CƠ SỞ LÝ THUYẾT** 6](#_Toc182171425)

[**1.1 Struct trong C++.** 6](#_Toc182171426)

[1.1.1 Định nghĩa Struct. 6](#_Toc182171427)

[1.1.2 Khai báo và khởi tạo Struct. 6](#_Toc182171435)

[1.1.3 Truy cập các thành viên của Struct. 7](#_Toc182171438)

[1.1.4 Constructor trong Struct. 7](#_Toc182171439)

[**1.2 Class và Đối tượng trong C++**. 8](#_Toc182171440)

[1.2.1 Định nghĩa Class. 8](#_Toc182171441)

[1.2.2 Đối tượng trong C++. 8](#_Toc182171442)

[1.2.3 Constructor và Destructor. 9](#_Toc182171443)

[1.2.4 Truy cập các thành viên và hàm của Class. 10](#_Toc182171444)

[1.2.5 Các phương thức trong Class 11](#_Toc182171445)

[**1.3 Danh sách liên kết**. 11](#_Toc182171446)

[1.3.1 Khái niệm danh sách liên kết. 11](#_Toc182171447)

[1.3.2 Cấu trúc của danh sách liên kết. 11](#_Toc182171448)

[1.3.3 Các loại danh sách liên kết. 12](#_Toc182171449)

[1.3.4 Ưu và nhược điểm của danh sách liên kết. 12](#_Toc182171450)

[1.3.5 Các thao tác cơ bản trên danh sách liên kết. 12](#_Toc182171451)

[**CHƯƠNG 2. XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÍ** 13](#_Toc182171452)

[**2.1 Giới thiệu về chương trình quản phòng gym bằng danh sách liên kết** 13](#_Toc182171454)

[2.1.1 Các tính năng chính của chương trình 13](#_Toc182171455)

[2.1.2 Cấu trúc chương trình 15](#_Toc182171456)

[**2.2 Mô tả chương trình** 15](#_Toc182171457)

[2.2.1 Khai báo thư viện 15](#_Toc182171458)

[2.2.2 Định nghĩa cấu trúc khách hàng (struct khachhang)**.** 16](#_Toc182171459)

[2.2.3 Phương thức nhap()**.** 16](#_Toc182171460)

[2.2.4 Phương thức xuat() 18](#_Toc182171461)

[2.2.5 Các phương thức truy cập và thay đổi thông tin 18](#_Toc182171462)

[2.2.6 Định nghĩa cấu trúc quản lý thiết bị (qlythietbi) 19](#_Toc182171463)

[2.2.7 Phương thức Input() 20](#_Toc182171464)

[2.2.8 Phương thức Output() 20](#_Toc182171465)

[2.2.9 Phương thức get\_masp(), themthietbi(), timkiemsp() 21](#_Toc182171466)

[2.2.10 Phương thức sắp xếp quicksort 22](#_Toc182171467)

[2.2.11 Phương thức isDuplicateProductCode() 23](#_Toc182171468)

[2.2.12 Hàm themthietbi() 24](#_Toc182171469)

[2.2.13 Hàm qlythietbi() 25](#_Toc182171470)

[2.2.14Hàm để thêm khách hàng mới. 28](#_Toc182171471)

[2.2.15 Hàm xóa khách hàng 29](#_Toc182171472)

[2.2.16 Hàm tìm khách hàng theo mã 30](#_Toc182171473)

[2.2.17Hàm tìm khách hàng bằng tên 31](#_Toc182171474)

[2.2.18 Hàm hiển thị danh sách khách hàng 32](#_Toc182171475)

[2.2.19 Hàm thêm một khách hàng mới vào đầu danh sách liên kết 33](#_Toc182171476)

[2.2.20 Hàm thêm khách hàng vào một vị trí bất kì 34](#_Toc182171477)

[2.2.21 Hàm sắp xếp theo mã khách hàng 35](#_Toc182171478)

[2.2.22 Hàm sắp xếp khách hàng theo tổng chi phí 36](#_Toc182171479)

[2.2.23 Hàm sửa thông tin khách hàng 38](#_Toc182171480)

[**2.3 Hàm menu()** 39](#_Toc182171481)

[**CHƯƠNG 3. KẾT QUẢ VÀ KIẾN NGHỊ** 40](#_Toc182171482)

[3.1 Kết quả đạt được 40](#_Toc182171483)

[3.2 Kiến nghị 40](#_Toc182171485)

[**PHỤ LỤC** 40](#_Toc182171486)

[**TÀI LIỆU THAM KHẢO** 41](#_Toc182171487)

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

* 1. Struct trong C++.

1.1.1 Định nghĩa Struct.

**Struct (cấu trúc)** trong C++ là một kiểu dữ liệu do người dùng tự định nghĩa, cho phép bạn nhóm nhiều biến khác nhau thuộc các kiểu dữ liệu khác nhau thành một đơn vị. Nó giống như một bản ghi, giúp bạn tổ chức thông tin một cách có hệ thống và dễ quản lý. Struck thường được sử dụng trong lập trình C++ để tạo ra các cấu trúc dữ liệu đơn giản và có liên kết chặt chẽ với nhau mà không cần sử dụng các phương pháp phức tạp của lập trình hướng đối tượng

Cú pháp khai báo cấu trúc như sau:

struct TenStruct {

// Các thành viên của struct

kiểu\_dữ\_liệu tên\_thành\_viên1;

kiểu\_dữ\_liệu tên\_thành\_viên2;

// ... };

1.1.2 Khai báo và khởi tạo Struct.

Khai báo struct: có nghĩa là định nghĩa một cấu trúc với tên gọi và các thành phần (biến) bên trong nó, có thể có kiểu dữ liệu khác nhau. Struct thường được dùng để nhóm các dữ liệu có liên quan với nhau thành một khối logic giúp mã nguồn dễ đọc, dễ hiểu, và thuận tiện cho việc quản lý dữ liệu. Các thành phần này có thể có kiểu dữ liệu khác nhau như int, float, string,...

Khởi tạo struct: Đối tượng struct có thể được khởi tạo với các giá trị mặc định cho các thành viên. Nếu có một constructor trong struct, ta có thể khởi tạo các thành viên trong lúc tạo đối tượng. Ngoài ra, có thể gán trực tiếp giá trị cho từng thành viên thông qua dấu =.

Ví dụ:

khachHang1.id = 1;

KhachHang1.tenKhachHang = "Nguyễn Văn A";

GoiTap1.giatien = 350;

* + 1. 1.1.3 Truy cập các thành viên của Struct.

**Truy cập các thành viên của struct** là cách lấy giá trị của các thành phần (biến thành viên) bên trong một cấu trúc dữ liệu struct trong C++. Để truy cập các thành viên, có hai cách phổ biến tùy thuộc vào cách sử dụng struct:

* *Truy cập với toán tử dấu chấm (.):*Toán tử dấu chấm (.) được sử dụng khi bạn làm việc với một biến struct không phải là con trỏ. Để truy cập một thành viên cụ thể của struct, bạn chỉ cần viết tên biến struct, tiếp theo là dấu chấm, rồi tên thành viên. Đây là cách phổ biến và trực tiếp khi làm việc với một biến struct.

Ví dụ: KhachHang1.id = 1; // Gán giá trị cho id của KhachHang1

cout << KhachHang1.tenKhachHang; // Xuất tên kh của KhachHang1

* *Truy cập với toán tử mũi tên (->):*Toán tử mũi tên (->) được sử dụng khi bạn làm việc với một con trỏ trỏ đến struct. Đối với con trỏ, bạn không thể sử dụng dấu chấm trực tiếp, vì vậy bạn phải sử dụng toán tử mũi tên để truy cập các thành viên của struct qua con trỏ.

Ví dụ: Customer\* ptr = &KhachHang1;

cout << ptr->tenKhachHang;

### 1.1.4 Constructor trong Struct.

* Định nghĩa: Constructor là một hàm đặc biệt trong struct (hoặc class), được gọi tự động khi một đối tượng của struct được tạo ra. Nó dùng để khởi tạo các thành viên dữ liệu của đối tượng đó.
* Mục đích:
* Khởi tạo giá trị ban đầu: Gán giá trị ban đầu cho các thành viên dữ liệu của struct khi đối tượng được tạo.
* Kiểm tra dữ liệu đầu vào: Đảm bảo rằng dữ liệu truyền vào cho constructor là hợp lệ.
* Thực hiện các tác vụ khác: Ví dụ, mở file, kết nối đến cơ sở dữ liệu,...

1.2 Class và Đối tượng trong C++.

* + 1. 1.2.1 Định nghĩa Class.

**Class trong C++** hay còn gọi là lớp trong C++ là một bản thiết kế cho đối tượng, là một kiểu dữ liệu do người dùng định nghĩa, được sử dụng để tạo ra các đối tượng có cùng bản chất. Class bao gồm các thuộc tính (dữ liệu) và phương thức (hàm thành viên) để thao tác với dữ liệu đó Trong lập trình hướng đối tượng (OOP), class là khái niệm trung tâm để tổ chức và quản lý dữ liệu và chức năng của chương trình.

Cú pháp định nghĩa một class:

class TênClass {

public:

// Các thành viên dữ liệu

// Các phương thức

};

Class trong C++ có ba mức độ truy cập chính: public, private, và protected:

* public: Các thành viên public có thể truy cập từ bất kì đâu.
* private Các thành viên private chỉ có thể truy cập từ bên trong Class.
* protected: Các thành viên protected chỉ có thể truy cập từ class và các lớp dẫn xuất.

1.2.2 Đối tượng trong C++.

Đối tượng là một thể hiện (instance) cụ thể của một class. Khi một class được định nghĩa, nó chỉ là một bản thiết kế (blueprint), và không chiếm bộ nhớ. Khi tạo một đối tượng từ class, bộ nhớ được cấp phát cho các thành viên của class và các phương thức có thể được gọi để thao tác với dữ liệu của đối tượng đó.

Cú pháp tạo đối tượng: TênClass tênĐốiTượng;

Ví dụ:

List danhSach; // danhSach là một đối tượng của class List

Khi một đối tượng được tạo, nó có thể truy cập các thành viên và phương thức của class thông qua dấu chấm (.), hoặc dấu -> nếu là con trỏ.

### 1.2.3 Constructor và Destructor.

*Constructor (Hàm Khởi Tạo):*

* **Định nghĩa**: Constructor là một phương thức đặc biệt được gọi tự động khi một đối tượng của class được tạo. Nhiệm vụ chính của constructor là khởi tạo các thành viên của class với giá trị mặc định hoặc giá trị do người dùng cung cấp. Được gọi khi một đối tượng được tạo, có thể là tự động hoặc thông qua từ khóa new.
* Đặc điểm:
* Tên của constructor trùng với tên của class và không có kiểu trả về (ngay cả void).
* Có thể có nhiều constructor với các tham số khác nhau (tính chất **đa hình**), gọi là **constructor quá tải.**
* Nếu không khai báo constructor nào, C++ sẽ cung cấp một **constructor mặc định** không tham số.
* Constructor có thể là công khai (public) hoặc bảo mật (protected), tuỳ vào thiết kế của class.

*Destructor (Hàm hủy):*

* **Định nghĩa**: Destructor là một phương thức đặc biệt được gọi tự động khi một đối tượng của class được hủy (ví dụ, khi đối tượng ra khỏi phạm vi hoặc được xóa thủ công). Nhiệm vụ chính của destructor là giải phóng tài nguyên (bộ nhớ hoặc tài nguyên hệ thống) mà đối tượng có thể đã sử dụng.
* **Đặc điểm**:
  + Tên của destructor trùng với tên của class, nhưng có thêm dấu ~ (tilde) đứng trước.
  + Destructor không có tham số và cũng không có kiểu trả về.
  + Một class chỉ có một destructor duy nhất, không thể quá tải destructor.
  + Thường được sử dụng khi class cần giải phóng bộ nhớ cấp phát động hoặc đóng kết nối.

1.2.4 Truy cập các thành viên và hàm của Class.

Để truy cập thành viên dữ liệu và hàm của class, chúng ta sử dụng dấu chấm (.) khi thao tác với đối tượng. Khi sử dụng con trỏ, ta dùng dấu ->. Trong ví dụ sau, một đối tượng danhSach của class List được dùng để gọi các phương thức addFirst và duyetsach:

List danhSach;

danhSach.addFirst(...); // Thêm một sách vào danh sách

danhSach.duyetsach(); // Duyệt qua và hiển thị thông tin sách

Khi khai báo các hàm thành viên trong class, chúng ta chỉ cần liệt kê tên hàm bên trong phần khai báo của class. Để triển khai các hàm này, C++ cung cấp cú pháp TênLớp::TênHàm để chỉ rõ hàm đó thuộc về class nào. Dấu :: trong C++ gọi là toán tử phạm vi (scope resolution operator), dùng để xác định thành viên nào thuộc class nào, đặc biệt khi có thể có nhiều class với các hàm có tên trùng nhau.

Ví dụ, khai báo phương thức addFirst trong class List sẽ như sau:

class List {

public:

void addFirst(int id, string tensach, string theloai, string tacgia, string namxb, int giatien);

};

Trong phần định nghĩa (triển khai) của phương thức này, List::addFirst được sử dụng để cho C++ biết rằng đây là hàm addFirst của class List.

1.2.5 Các phương thức trong Class

Phương thức (hàm thành viên) trong một class là các hàm được định nghĩa để thao tác với dữ liệu của class và thực hiện các hành vi liên quan đến class đó. Các phương thức này giúp đóng gói các thao tác xử lý dữ liệu bên trong class, tăng tính bảo mật và cho phép sử dụng lại mã dễ dàng hơn.

* 1. 1.3 Danh sách liên kết.
     1. 1.3.1 Khái niệm danh sách liên kết.
     2. Danh sách liên kết (Linked List) là một loại cấu trúc dữ liệu động, trong đó các phần tử được liên kết với nhau thông qua các tham chiếu (con trỏ). Mỗi phần tử, hay "node," chứa dữ liệu và một con trỏ đến phần tử tiếp theo. Cách thức này cho phép các phần tử không cần được lưu liên tục trong bộ nhớ, giúp tận dụng bộ nhớ hiệu quả.
     3. 1.3.2 Cấu trúc của danh sách liên kết.

Một danh sách liên kết được tạo thành từ nhiều node, và mỗi node thường bao gồm hai phần chính:

* Dữ liệu: Đại diện cho giá trị hoặc thông tin mà node lưu trữ.
* Con trỏ: Địa chỉ trỏ đến node tiếp theo trong danh sách, tạo sự liên kết giữa các node.

Ví dụ, một node trong danh sách liên kết đơn có thể được định nghĩa như sau:

struct Node {

int data;

// Dữ liệu của node Node\* next;

// Con trỏ đến node kế tiếp

};

### 1.3.3 Các loại danh sách liên kết.

Danh sách liên kết được phân loại thành ba loại chính:

1. Danh sách liên kết đơn (Singly Linked List): Chỉ có liên kết từ mỗi node đến node kế tiếp.
2. Danh sách liên kết đôi (Doubly Linked List): Mỗi node có hai liên kết, một đến node phía trước và một đến node phía sau.
3. Danh sách liên kết vòng (Circular Linked List): Node cuối cùng trỏ quay lại node đầu tiên, tạo thành một vòng lặp kín.

1.3.4 Ưu và nhược điểm của danh sách liên kết.

*Ưu điểm:*

* Có thể thay đổi kích thước một cách linh hoạt do sử dụng bộ nhớ động.
* Phù hợp cho các thao tác chèn hoặc xóa phần tử nhanh chóng mà không cần di chuyển các phần tử khác như trong mảng.

*Nhược điểm:*

* Không hỗ trợ truy cập ngẫu nhiên, khiến việc tìm kiếm chậm hơn so với mảng.
* Tốn bộ nhớ hơn mảng do mỗi node cần một con trỏ phụ.

1.3.5 Các thao tác cơ bản trên danh sách liên kết.

Các thao tác phổ biến trên danh sách liên kết bao gồm:

* Thêm node: Chèn một phần tử mới vào đầu, cuối, hoặc một vị trí cụ thể trong danh sách.
* Xóa node: Xóa một node khỏi danh sách ở đầu, cuối, hoặc một vị trí bất kỳ.
* Duyệt qua danh sách: Lần lượt truy cập các node để xử lý dữ liệu hoặc thực hiện các thao tác khác trên chúng.

# CHƯƠNG 2. XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÍ

# PHÒNG GYM

* 1. 2.1 Giới thiệu về chương trình quản phòng gym bằng danh sách liên kết
     1. Chương trình quản lý phòng gym là một hệ thống nhỏ, được phát triển nhằm giúp quản lý thông tin hội viên và các gói tập luyện tại phòng gym một cách hiệu quả. Sử dụng ngôn ngữ C++ và danh sách liên kết đơn, chương trình cho phép lưu trữ và xử lý dữ liệu về các gói tập, giúp nhân viên quản lý dễ dàng thêm, xóa, và cập nhật thông tin về các gói tập và tình trạng hội viên. Mục tiêu chính của hệ thống là hỗ trợ quản lý và truy xuất dữ liệu nhanh chóng, đáp ứng nhu cầu cơ bản của một phòng gym với quy mô vừa và nhỏ.
     2. 2.1.1 Các tính năng chính của chương trình

Chương trình quản lý phòng gym cung cấp các tính năng cơ bản sau:

* **Quản lí khách hàng phòng gym**  
  Cung cấp công cụ để quản lý thông tin hội viên, giúp phòng gym lưu trữ thông tin khách hàng bao gồm mã khách hàng, tên khách hàng, ngày đăng ký, và các gói tập đã đăng ký. Tính năng này cũng bao gồm chỉnh sửa và sắp xếp khách hàng theo nhiều tiêu chí khác nhau.
* **Xuất danh sách khách hàng**  
  Hiển thị danh sách tất cả khách hàng hiện có trong hệ thống cùng với các thông tin chi tiết về mỗi khách hàng. Chức năng này giúp nhân viên phòng gym nắm rõ và theo dõi thông tin toàn bộ hội viên.
* **Tìm kiếm theo mã khách hàng**  
  Cho phép tìm kiếm một khách hàng cụ thể thông qua mã khách hàng. Tính năng này rất hữu ích để nhanh chóng xác định thông tin của khách hàng mà không cần duyệt qua toàn bộ danh sách.
* **Tìm kiếm theo tên khách hàng**  
  Tìm kiếm khách hàng theo tên hoặc một phần của tên, hỗ trợ trường hợp phòng gym cần truy xuất nhanh thông tin của một hoặc nhiều khách hàng có tên tương tự.
* **Xóa thông tin khách hàng**  
  Xóa thông tin của khách hàng từ danh sách. Chức năng này hỗ trợ cập nhật thông tin khách hàng khi một khách hàng ngừng dịch vụ hoặc yêu cầu hủy đăng ký.
* **Thêm thông tin khách hàng**  
  Thêm thông tin khách hàng mới vào danh sách, lưu trữ đầy đủ các chi tiết về mã khách hàng, tên, và gói tập đăng ký.
* **Thêm thông tin khách hàng vào đầu danh sách**  
  Cho phép thêm thông tin của khách hàng vào đầu danh sách, hữu ích khi cần ưu tiên hoặc theo dõi các khách hàng đặc biệt.
* **Thêm thông tin khách hàng vào vị trí tùy ý**  
  Thêm khách hàng vào một vị trí cụ thể trong danh sách, giúp tùy chỉnh cách lưu trữ và quản lý dữ liệu khách hàng theo các tiêu chí riêng.
* **Sắp xếp theo mã khách hàng**  
  Sắp xếp danh sách khách hàng theo mã khách hàng. Tính năng này hỗ trợ việc quản lý danh sách dễ dàng hơn, đảm bảo dữ liệu được tổ chức rõ ràng.
* **Sắp xếp theo tổng chi phí**  
  Sắp xếp danh sách khách hàng theo tổng chi phí đã chi trả cho các gói tập, giúp dễ dàng theo dõi khách hàng theo tiêu chí tài chính và tối ưu hóa hoạt động kinh doanh.
* **Quản lý thiết bị phòng gym**  
  Theo dõi tình trạng và thông tin thiết bị phòng gym, bao gồm việc cập nhật thông tin thiết bị, kiểm tra số lượng, tình trạng bảo dưỡng. Điều này giúp phòng gym luôn đảm bảo các thiết bị hoạt động tốt và có sẵn cho khách hàng.
* **Sửa thông tin khách hàng**  
  Chỉnh sửa thông tin khách hàng khi cần, như cập nhật họ tên, mã khách hàng, hoặc thông tin gói tập đã đăng ký, giúp đảm bảo dữ liệu luôn chính xác và cập nhật.
* **Thoát chương trình**  
  Kết thúc chương trình một cách an toàn, giải phóng bộ nhớ và lưu trữ mọi dữ liệu cần thiết.
  + 1. 2.1.2 Cấu trúc chương trình

***Struct*** *Node*: Đại diện cho từng node trong danh sách liên kết, mỗi node chứa một khách hàng/thiết bị và con trỏ trỏ đến node tiếp theo.

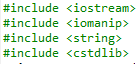
*Class List*: Quản lý danh sách gói tập và khách hàng, gồm các phương thức thao tác với danh sách liên kết như thêm, xóa, tìm kiếm, sắp xếp, và cập nhật tình trạng của từng gói tập.

* 1. 2.2 Mô tả chương trình
     1. 2.2.1 Khai báo thư viện

*iostream*: Hỗ trợ nhập và xuất dữ liệu qua bàn phím.

*string*: Xử lý chuỗi ký tự, hỗ trợ lưu trữ và thao tác trên tên gói tập và tên khách hàng. *Cstdlib:* Cho phép lưu trữ và đọc dữ liệu từ tệp, tiện cho việc lưu thông tin khách hàng và thiết bị phòng tập.

*Iomanip*: Cung cấp các công cụ để định dạng đầu ra, như đặt độ rộng cột (setw), căn lề (left, right), và điều chỉnh hiển thị số thập phân (setprecision).



**2.2.2Định nghĩa cấu trúc khách hàng (struct khachhang).**

A black text on a white background

Description automatically generated

*struct khachhang*: Định nghĩa một cấu trúc dữ liệu để lưu trữ thông tin khách hàng. Cấu trúc này bao gồm các thuộc tính như mã khách hàng, tên khách hàng, ngày sinh, số điện thoại, ngày đăng ký, phí gói tập và tổng tiền phải trả.

*int makh*: Lưu trữ mã số duy nhất của khách hàng.

*string tenkh*: Lưu trữ tên của khách hàng.

*string ngaysinh*: Lưu trữ ngày sinh của khách hàng.

*string sdt*: Lưu trữ số điện thoại của khách hàng.

*string ngaydk*: Lưu trữ ngày khách hàng đăng ký gói tập.

*float phigoitap*: Lưu trữ phí gói tập mà khách hàng phải trả.

*float tongtien*: Lưu trữ tổng số tiền mà khách hàng phải trả.

**2.2.3Phương thức nhap().**

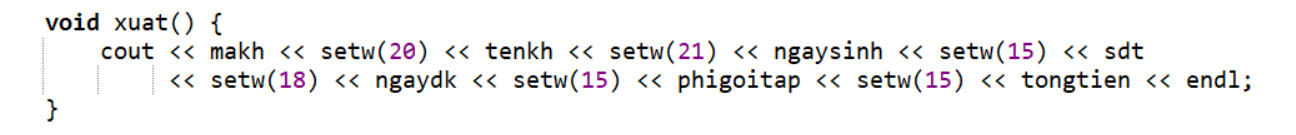
A white screen with text and numbers

Description automatically generated

nhap(): Phương thức này cho phép người dùng nhập thông tin của khách hàng từ bàn phím.

* cin >> makh; : Nhập mã khách hàng.
* while (makh < 0): Kiểm tra xem mã khách hàng có hợp lệ không (phải lớn hơn 0). Nếu không, yêu cầu người dùng nhập lại.
* cin.ignore(); : Xóa bỏ ký tự \n còn lại trong bộ đệm để tránh lỗi khi nhập chuỗi.
* getline(cin, tenkh); : Nhập tên khách hàng bằng cách sử dụng getline() để đảm bảo nhận được toàn bộ chuỗi.
* getline(cin, ngaysinh); : Nhập ngày sinh của khách hàng.
* getline(cin, sdt); : Nhập số điện thoại của khách hàng.
* getline(cin, ngaydk); : Nhập ngày đăng ký của khách hàng.
* cin >> phigoitap; : Nhập phí gói tập mà khách hàng phải trả.
* while ((phigoitap != 350) && (phigoitap != 2100) && (phigoitap != 3000)): Kiểm tra phí gói tập có hợp lệ không (phải là 350, 2100 hoặc 3000). Nếu không, yêu cầu người dùng nhập lại.
* tongtien = phigoitap; : Gán giá trị của phigoitap cho tongtien.

**2.2.4Phương thức xuat()**



xuat(): Phương thức này xuất thông tin của khách hàng ra màn hình dưới dạng bảng với các cột được căn chỉnh độ rộng.

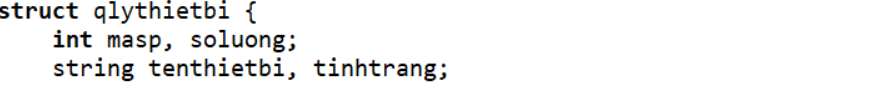
* cout << makh: Xuất mã khách hàng.
* setw(20) << tenkh: Xuất tên khách hàng và căn chỉnh độ rộng của cột là 20 ký tự.
* setw(21) << ngaysinh: Xuất ngày sinh và căn chỉnh độ rộng của cột là 21 ký tự.
* setw(15) << sdt: Xuất số điện thoại và căn chỉnh độ rộng của cột là 15 ký tự.
* setw(18) << ngaydk: Xuất ngày đăng ký và căn chỉnh độ rộng của cột là 18 ký tự.
* setw(15) << phigoitap: Xuất phí gói tập và căn chỉnh độ rộng của cột là 15 ký tự.
* setw(15) << tongtien: Xuất tổng tiền và căn chỉnh độ rộng của cột là 15 ký tự.

**2.2.5 Các phương thức truy cập và thay đổi thông tin**

A screenshot of a computer code

Description automatically generated

**2.2.6 Định nghĩa cấu trúc quản lý thiết bị (qlythietbi)**



int masp, soluong: masp lưu trữ mã sản phẩm, soluong lưu trữ số lượng tồn kho của thiết bị.

string tenthietbi, tinhtrang: tenthietbi lưu trữ tên thiết bị, tinhtrang lưu trữ tình trạng thiết bị (mới hoặc cũ).

**2.2.7 Phương thức Input()**

A screenshot of a computer code

Description automatically generated

Input(): Phương thức này cho phép người dùng nhập thông tin thiết bị từ bàn phím.

* cin >> masp; : Nhập mã sản phẩm.
* cin.ignore(); : Xóa bỏ ký tự \n còn lại trong bộ đệm để tránh lỗi khi nhập chuỗi.
* cin >> soluong; : Nhập số lượng tồn kho.
* getline(cin, tenthietbi); : Nhập tên thiết bị.
* getline(cin, tinhtrang); : Nhập tình trạng thiết bị (mới hoặc cũ).

**2.2.8 Phương thức Output()**

A close up of text

Description automatically generated

Output(bool o): Phương thức này xuất thông tin của thiết bị ra màn hình.

* Nếu o là true, in ra tiêu đề cột.
* Sau đó, in ra mã sản phẩm, số lượng, tên thiết bị và tình trạng thiết bị với định dạng căn chỉnh theo cột.

### 2.2.9 Phương thức get\_masp(), themthietbi(), timkiemsp()

A computer code with red and blue text

Description automatically generated

get\_masp(): Trả về mã sản phẩm.

themthietbi(): Gọi phương thức Input() để nhập thông tin thiết bị mới.

timkiemsp(qlythietbi tb[], int n): Tìm kiếm thiết bị theo mã sản phẩm trong danh sách.

* Nhập mã sản phẩm cần tìm.
* In tiêu đề cột.
* Duyệt qua danh sách thiết bị, nếu tìm thấy mã sản phẩm khớp, xuất thông tin thiết bị đó.
* Nếu không tìm thấy, in thông báo không tìm thấy mã sản phẩm.

### 2.2.10 Phương thức sắp xếp quicksort

A screenshot of a computer code

Description automatically generated

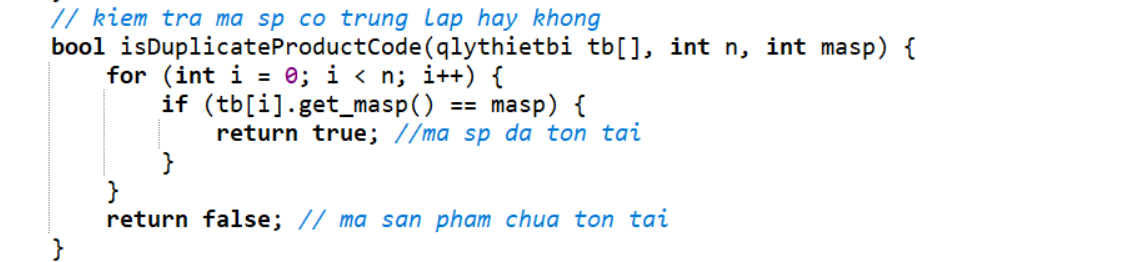
partition(qlythietbi tb[], int l, int r): Phân hoạch mảng dựa trên mã sản phẩm để sử dụng trong thuật toán sắp xếp nhanh (quicksort).

* Chọn phần tử trục (pivot) là mã sản phẩm của phần tử ở vị trí r.
* Duyệt từ hai đầu mảng, sắp xếp các phần tử nhỏ hơn trục sang bên trái và lớn hơn trục sang bên phải.
* Trả về chỉ số cuối cùng của phần tử ở phía bên trái sau khi phân hoạch.

quicksort(qlythietbi tb[], int l, int r): Sử dụng thuật toán sắp xếp nhanh (quicksort) để sắp xếp thiết bị theo mã sản phẩm.

* Nếu l lớn hơn hoặc bằng r, kết thúc hàm.
* Gọi hàm partition() để phân hoạch mảng và trả về vị trí mới của trục.
* Gọi đệ quy quicksort() cho các phần tử bên trái và bên phải của trục.

### 2.2.11 Phương thức isDuplicateProductCode()



isDuplicateProductCode(qlythietbi tb[], int n, int masp): Kiểm tra xem mã sản phẩm có bị trùng lặp trong danh sách hay không.

* Duyệt qua danh sách thiết bị.
* Nếu tìm thấy mã sản phẩm trùng, trả về true.
* Nếu không tìm thấy, trả về false.

### 2.2.12 Hàm themthietbi()

A screenshot of a computer code

Description automatically generated

* Khai báo biếnnewDevice: Tạo một đối tượng qlythietbi mới để lưu thông tin của thiết bị cần thêm.
* Lặp lại cho đến khi nhập mã sản phẩm hợp lệ: Sử dụng vòng lặp do-while để yêu cầu người dùng nhập mã sản phẩm. Kiểm tra mã sản phẩm có bị trùng lặp không bằng cách gọi phương thức isDuplicateProductCode(). Nếu mã sản phẩm bị trùng lặp, yêu cầu người dùng nhập lại. Nếu mã sản phẩm hợp lệ, thoát khỏi vòng lặp.
* Nhập các thông tin còn lại: Sau khi mã sản phẩm hợp lệ, nhập các thông tin còn lại của thiết bị như số lượng tồn kho, tên thiết bị và tình trạng.
* Thêm thiết bị vào danh sách: Gán đối tượng newDevice vào mảng tb tại vị trí n và tăng n lên một đơn vị để cập nhật số lượng thiết bị trong danh sách.

### 2.2.13 Hàm qlythietbi()

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

Khai báo biến:

* int luachon: Lựa chọn của người dùng.
* bool exit = true: Biến kiểm soát vòng lặp để thoát chương trình.
* qlythietbi tb[100]: Mảng lưu trữ thông tin của 100 thiết bị.
* int n = 0: Số lượng thiết bị hiện tại trong danh sách.
* bool o = true: Biến kiểm soát việc in tiêu đề của bảng khi xuất thông tin thiết bị.

Vòng lặpdo-while: Chạy liên tục cho đến khi người dùng chọn thoát (exit).

* system("cls"): Xóa màn hình console (trên Windows).
* Hiển thị menu: Hiển thị các tùy chọn cho người dùng với các chức năng: nhập thiết bị, xuất thiết bị, tìm kiếm theo mã sản phẩm, thêm thiết bị mới, sắp xếp thiết bị theo mã sản phẩm và thoát khỏi chương trình.
* Nhận lựa chọn của người dùng: Người dùng nhập lựa chọn thông qua cin >> luachon;.

Xử lý các lựa chọn của người dùng:

* Nhập thiết bị **(**case 1**)**: Gọi phương thức Input() để nhập thông tin thiết bị và tăng n lên một đơn vị. Tạm dừng chương trình (system("pause")) để người dùng có thể thấy kết quả trước khi tiếp tục.
* Xuất thiết bị **(**case 2**)**: Xuất thông tin của tất cả các thiết bị trong danh sách bằng cách gọi phương thức Output(). Biến o kiểm soát việc in tiêu đề của bảng. Tạm dừng chương trình để người dùng có thể thấy kết quả trước khi tiếp tục.
* Tìm kiếm theo mã sản phẩm **(**case 3**)**: Gọi phương thức timkiemsp() để tìm kiếm thiết bị theo mã sản phẩm. Tạm dừng chương trình để người dùng có thể thấy kết quả trước khi tiếp tục.
* Thêm thiết bị mới **(**case 4**)**: Gọi phương thức themthietbi() để thêm thiết bị mới vào danh sách. Tạm dừng chương trình để người dùng có thể thấy kết quả trước khi tiếp tục.
* Sắp xếp thiết bị theo mã sản phẩm **(**case 5**)**: Gọi phương thức quicksort() để sắp xếp thiết bị theo mã sản phẩm. Thông báo sắp xếp thành công và tạm dừng chương trình để người dùng có thể thấy kết quả trước khi tiếp tục.
* Thoát chương trình **(**case 6**)**: Đặt exit = false để thoát vòng lặp và kết thúc chương trình.
* Lựa chọn không hợp lệ **(**default**)**: Thông báo lỗi và tiếp tục vòng lặp.

### 2.2.14Hàm để thêm khách hàng mới.

A screenshot of a computer program

Description automatically generated

Hàm addCustomer có nhiệm vụ sẽ thêm khách hàng mới vào danh sách liên kết nhưng trước khi thêm thì hàm sẽ kiểm tra xem mã khách hàng đã tồn tại trong danh sách chưa.

Ban đầu con trỏ của nút temp sẽ được gán cho head, tức là phần tử đầu tiên trong danh sách liên kết. Sau đó dùng vòng lặp while để duyệt qua tất cả các nút trong danh sách cho đến khi temp bằng nullptr (nghĩa là đã duyệt hết danh sách).

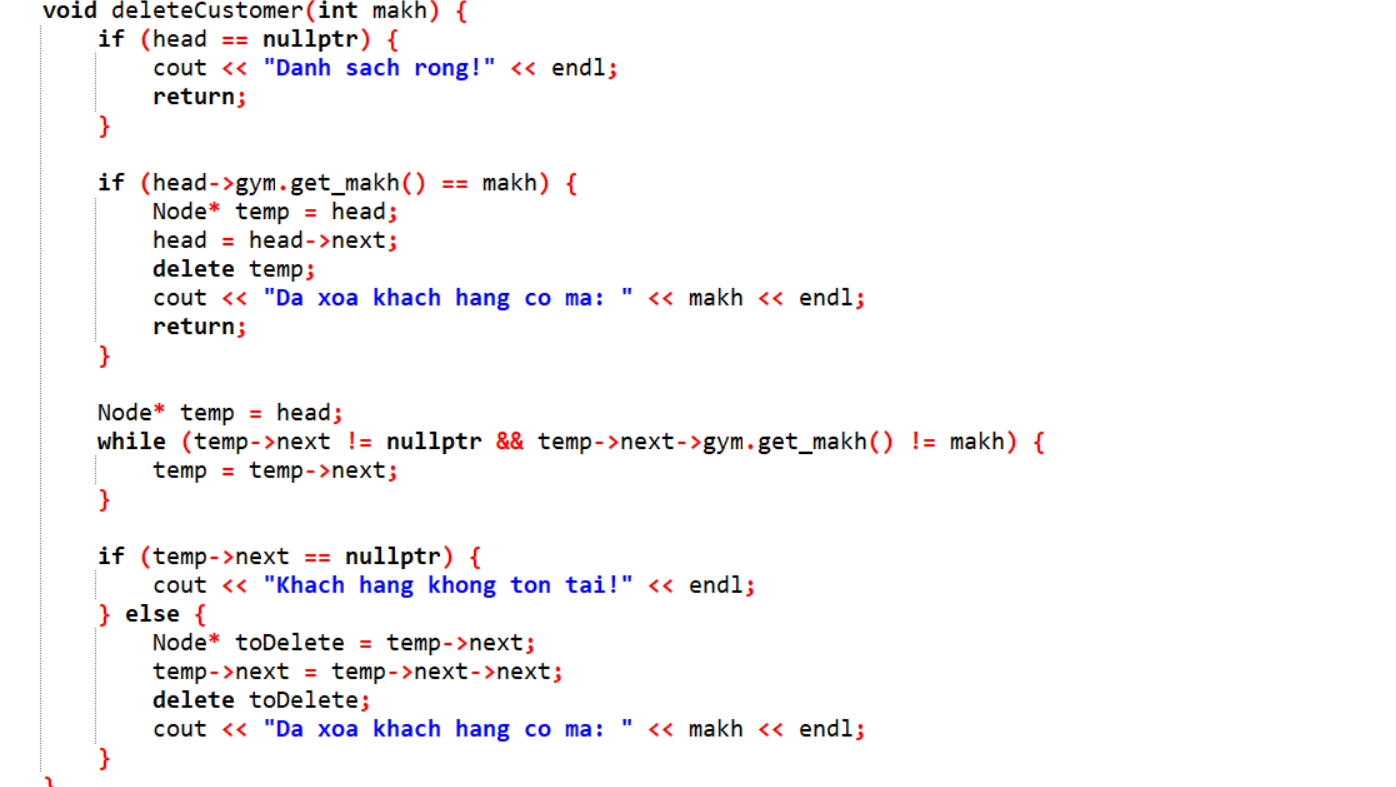
Trong mỗi vòng lặp, hàm so sánh mã khách hàng của nút hiện tại (temp->gym.get\_makh()) với mã khách hàng của khách hàng mới (newGym.get\_makh()), nếu chúng trùng nhau nghĩa là khách hàng đã tồn tại.

Nếu mã khách hàng không trùng nhau thì hàm sẽ nút mới newNode và được cấp phát động trong bộ nhớ bằng từ khóa new, thuộc tính gym của newNode được gán bằng khách hàng mới (newGym) và thuộc tính next được gán cho nullptr (chỉ ra rằng nút này là nút cuối trong danh sách).

\*Nếu danh sách rỗng thì newNode sẽ được gán bằng nullptr, khách hàng mới sẽ làm phần tử đầu tiên trong danh sách.

\*Nếu danh sách không rỗng con trỏ temp sẽ duyệt bắt đầu từ head cho tới khi tìm được nút cuối. Sau đó nút mới sẽ được thêm vào cuối danh sách.

### 2.2.15 Hàm xóa khách hàng



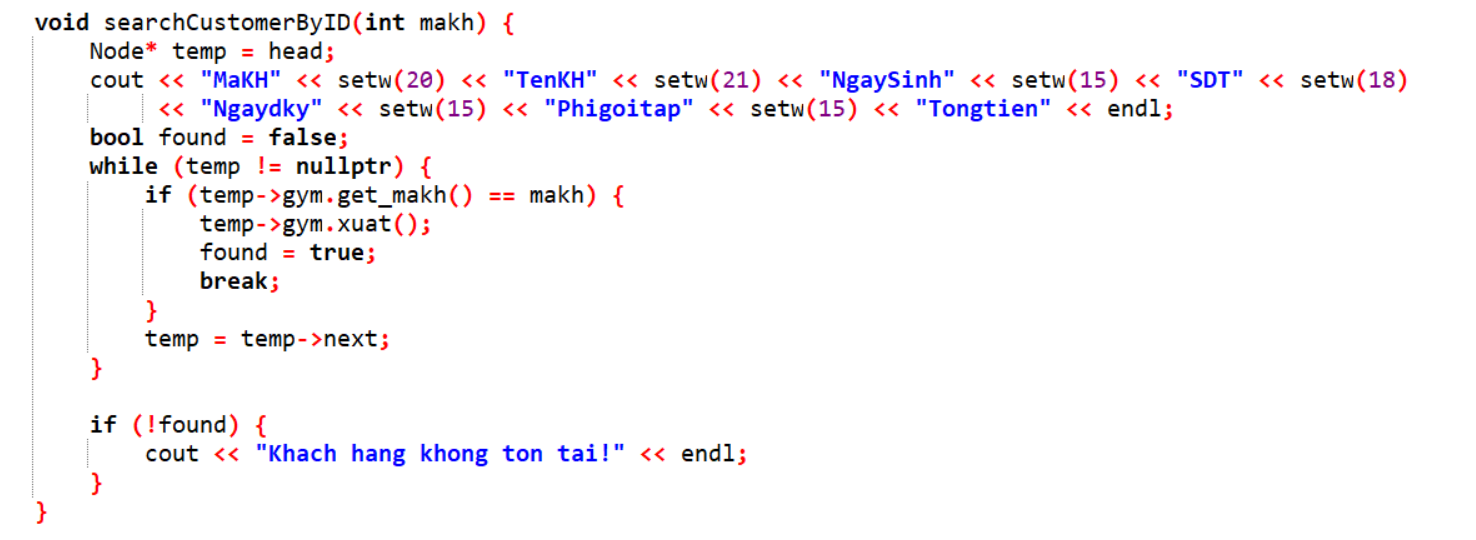
Hàm deleteCustomer sẽ thực hiện các bước kiểm tra sự tồn tại của khách hàng trong danh sách và xóa nếu khách hàng tồn tại.

Hàm sẽ kiểm tra xem danh sách có rỗng hay không; nếu danh sách chưa có phần tử nào (head==nullptr) thì sẽ in ra thông báo “Danh sach rong!” và kết thúc.

Nếu khách hàng đầu tiên trong danh sách có mã khách hàng (makh) trùng với mã cần xóa, nghĩa là khách hàng cần xóa là phần tử đầu tiên của danh sách. (Node\* temp = head; )tạo một con trỏ temp trỏ đến phần tử đầu tiên, sau đó (head = head->next; ) thay đổi head để trỏ đến phần tử tiếp theo trong danh sách, tức là phần tử sau khi xóa cuối cùng giải phóng bộ nhớ của phần tử đầu tiên, xóa khách hàng khỏi danh sách. Nếu khách hàng không phải là phần tử đầu tiên, hàm sẽ duyệt qua danh sách để tìm khách hàng có mã makh cần xóa (temp->next != nullptr) đảm bảo rằng temp không vượt quá cuối danh sách, (temp->next->gym.get\_makh() !=makh) kiểm tra mã của phần tử tiếp theo, nếu mã không trùng thì chuyển đến phần tử tiếp theo (temp = temp->next).

Kiểm tra xem khách hàng có tồn tại hay không: nếu sau khi duyệt xong mà không tìm thấy khách hàng có mã makh (tức là temp->next == nullptr), hàm sẽ in ra thông báo "Khach hang khong ton tai!". Nếu tìm thấy khách hàng cần xóa (tức là temp->next != nullptr) , (Node\* toDelete = temp->next; ) lưu trữ con trỏ đến nút cần xóa (nút sau temp). (temp->next = temp->next->next; ) thay đổi liên kết của temp sao cho phần tử cần xóa bị loại khỏi danh sách. Con trỏ next của temp sẽ trỏ đến phần tử sau phần tử cần xóa. Sau đó giải phóng bộ nhớ của phần tử đó. Sau khi xóa thành công, hàm in ra thông báo "Da xoa khach hang co ma: " và mã khách hàng.

### 2.2.16 Hàm tìm khách hàng theo mã



Hàm searchCustomerByID sẽ thực hiện tìm kiếm khách hàng trong danh sách liên kết theo mã khách hàng (makh).

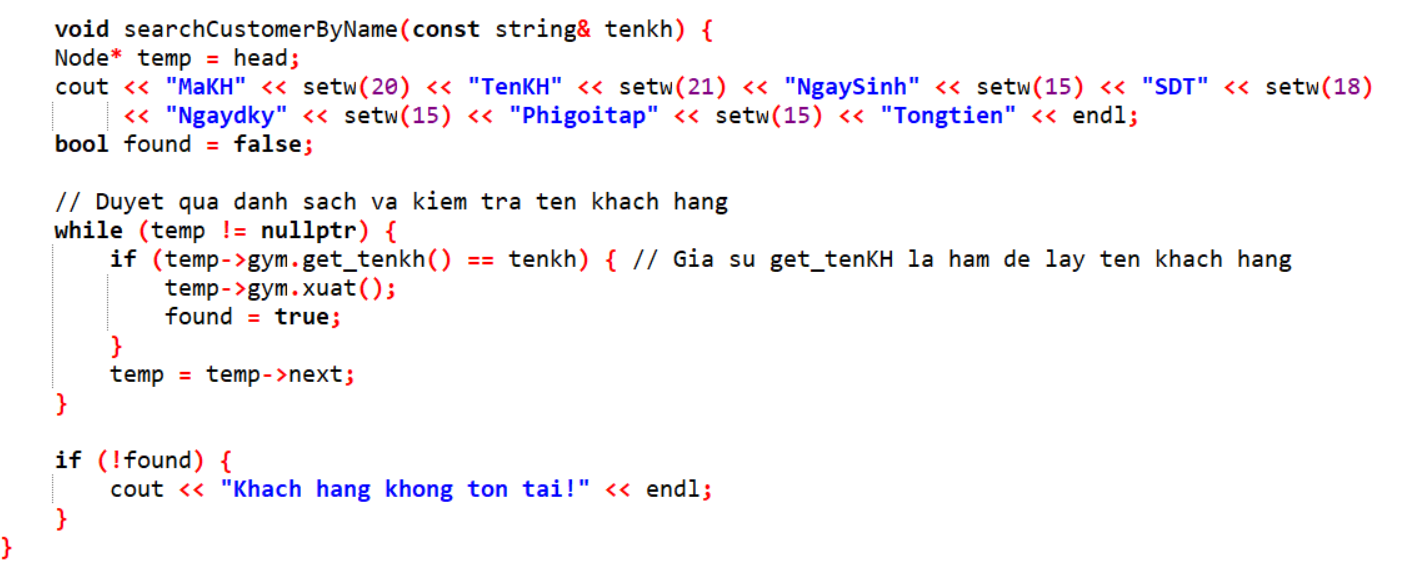
Sử dụng setw để căn chỉnh các cột giữa thông tin khách hàng cho hợp lí.

Khởi tạo biến tìm kiếm: found được khởi tạo bằng false, dùng để theo dõi xem khách hàng có được tìm thấy trong danh sách không.

Hàm sẽ duyệt qua từng phần tử trong danh sách liên kết (bắt đầu từ head). Nếu mã khách hàng (temp->gym.get\_makh()) bằng mã cần tìm (makh), hàm sẽ gọi phương thức xuat() của đối tượng gym để in thông tin của khách hàng và đánh dấu found là true. Sau khi tìm thấy khách hàng, vòng lặp dừng lại nhờ lệnh break.

Nếu sau khi duyệt hết danh sách mà không tìm thấy khách hàng (biến found vẫn là false), hàm sẽ in thông báo "Khach hang khong ton tai!".

### 2.2.17 Hàm tìm khách hàng bằng tên



Hàm searchCustomerByName dùng để tìm kiếm khách hàng theo tên trong danh sách liên kết các khách hàng.

void searchCustomerByName(const string& tenkh) hàm này nhận vào một tham số tenkh, là tên của khách hàng mà bạn muốn tìm kiếm. Tham số này được truyền theo kiểu tham chiếu (reference) và là hằng số (const), nhằm tránh sao chép và bảo vệ giá trị của nó không bị thay đổi trong hàm.

Khởi tạo biến found sử dụng để đánh dấu xem khách hàng có được tìm thấy hay không. Ban đầu, nó được gán là false (chưa tìm thấy).

Vòng lặp while này duyệt qua các phần tử trong danh sách liên kết, bắt đầu từ phần tử đầu tiên (giả sử head là con trỏ trỏ đến khách hàng đầu tiên trong danh sách). temp là con trỏ duyệt qua các phần tử trong danh sách.

Với mỗi phần tử trong danh sách, hàm kiểm tra nếu tên khách hàng trong phần tử đó (temp->gym.get\_tenkh()) trùng với tên tenkh được cung cấp trong tham số.

Nếu tìm thấy khách hàng có tên trùng với tenkh, hàm gọi phương thức xuat() của đối tượng gym để in ra thông tin của khách hàng đó. Cờ found được gán là true, tức là đã tìm thấy khách hàng.

Sau khi kiểm tra xong phần tử hiện tại, con trỏ temp sẽ di chuyển đến phần tử tiếp theo trong danh sách liên kết (dùng con trỏ next).

Sau khi vòng lặp kết thúc, nếu không tìm thấy khách hàng nào có tên trùng với tenkh (tức là found vẫn là false), hàm sẽ in thông báo "Khách hàng không tồn tại!"

### 2.2.18 Hàm hiển thị danh sách khách hàng

Hàm displayAllCustomers() có chức năng hiển thị danh sách tất cả khách hàng trong một danh sách liên kết.

Đầu tiên sẽ kiểm tra xem danh sách liên kết có rỗng hay không bằng cách kiểm tra xem head có phải là nullptr hay không. Nếu head là nullptr, tức là danh sách không có khách hàng nào, và hàm sẽ in ra thông báo "Danh sach khach hang rong!" (Danh sách khách hàng rỗng) rồi kết thúc hàm với câu lệnh return.

Vòng lặp while sẽ lặp lại cho đến khi temp trở thành nullptr (tức là khi đã duyệt hết tất cả các node trong danh sách).Trong mỗi vòng lặp, hàm sẽ gọi temp->gym.xuat() để hiển thị thông tin của khách hàng. Ở đây, gym có thể là một đối tượng chứa thông tin của khách hàng và xuat() là phương thức dùng để in hoặc hiển thị thông tin của khách hàng.Sau khi hiển thị thông tin của một khách hàng, con trỏ temp được cập nhật để trỏ đến node kế tiếp trong danh sách (temp = temp->next), và tiếp tục vòng lặp cho đến khi không còn khách hàng nào.

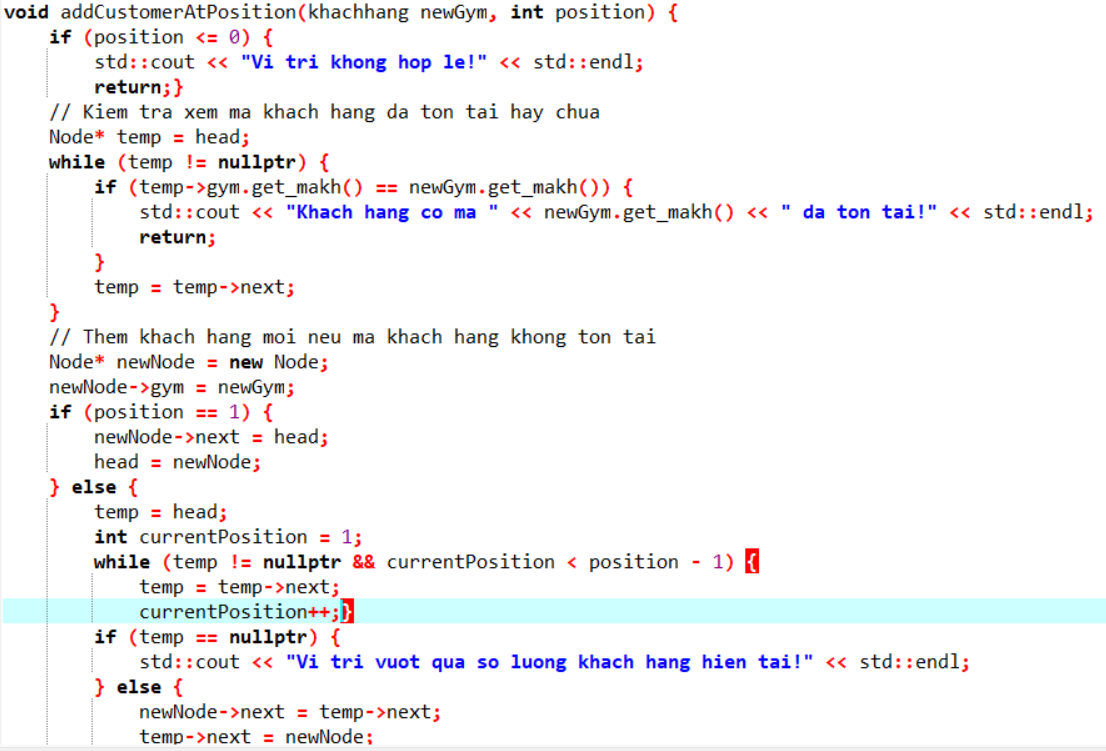
### 2.2.19 Hàm thêm một khách hàng mới vào đầu danh sách liên kết

Hàm addCustomerAtHead(khachhang newGym) có chức năng thêm một khách hàng mới vào đầu danh sách liên kết.

Trước tiên kiểm tra xem mã khách hàng đã tồn tại trong danh sách hay chưa. Con trỏ temp được gán với head (node đầu tiên của danh sách). Vòng lặp while sẽ duyệt qua tất cả các node trong danh sách liên kết (khi temp != nullptr nghĩa là còn node trong danh sách). Trong mỗi vòng lặp, hàm kiểm tra xem mã khách hàng (newGym.get\_makh()) của khách hàng mới có trùng với mã khách hàng của node hiện tại (temp->gym.get\_makh()) không. Nếu có sự trùng lặp (nghĩa là khách hàng đã tồn tại trong danh sách), hàm sẽ in ra thông báo "Khach hang co ma ... da ton tai!" và kết thúc hàm với câu lệnh return, không thêm khách hàng mới vào danh sách.

Nếu mã khách hàng không tồn tại trong danh sách (không có sự trùng lặp), hàm sẽ tạo một node mới. Dòng Node\* newNode = new Node; cấp phát bộ nhớ cho node mới. Tiếp theo, thông tin khách hàng (newGym) được gán vào trường gym của node mới (newNode->gym = newGym). Dòng newNode->next = head; thiết lập con trỏ next của node mới trỏ đến node hiện tại đầu danh sách (tức là head), đảm bảo liên kết giữa các node trong danh sách. Cuối cùng, head = newNode; thay đổi con trỏ head để trỏ đến node mới, khiến node mới trở thành node đầu tiên trong danh sách liên kết.

### 2.2.20 Hàm thêm khách hàng vào một vị trí bất kì



Hàm addCustomerAtPosition có chức năng thêm một khách hàng mới vào danh sách liên kết tại một vị trí cụ thể (được chỉ định bởi tham số position).

Hàm đầu tiên kiểm tra xem vị trí được yêu cầu có hợp lệ hay không. Nếu position <= 0, tức là vị trí không hợp lệ (vị trí phải là số nguyên dương), hàm sẽ in ra thông báo "Vi tri khong hop le!" và kết thúc hàm bằng return.

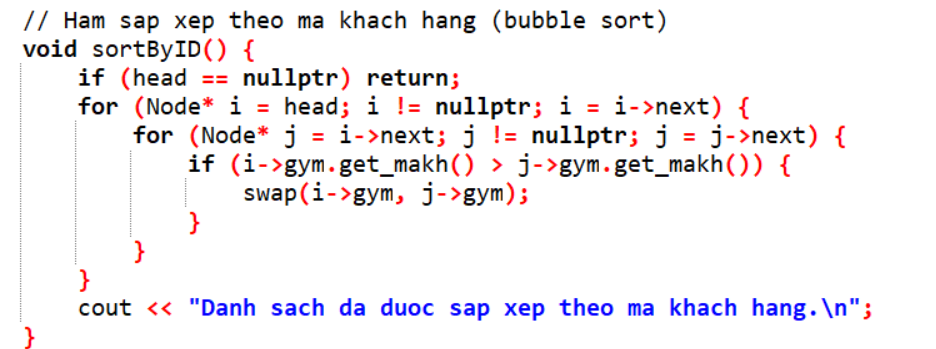
Con trỏ temp được gán với head, tức là node đầu tiên của danh sách. Vòng lặp while sẽ duyệt qua tất cả các node trong danh sách liên kết (khi temp != nullptr).Trong mỗi vòng lặp, hàm sẽ kiểm tra mã khách hàng của node hiện tại (temp->gym.get\_makh()) với mã khách hàng của khách hàng mới (newGym.get\_makh()). Nếu mã khách hàng đã tồn tại, hàm in ra thông báo "Khach hang co ma ... da ton tai!" và kết thúc hàm mà không thêm khách hàng mới vào danh sách.

Nếu position == 1, hàm sẽ thêm khách hàng mới vào đầu danh sách. newNode là một node mới được cấp phát bộ nhớ. Thông tin khách hàng mới (newGym) được gán vào node này (newNode->gym = newGym). Dòng newNode->next = head thiết lập con trỏ next của node mới trỏ tới node đầu tiên hiện tại (tức là head). Cuối cùng, head = newNode; thay đổi con trỏ head để trỏ đến node mới, khiến khách hàng mới trở thành node đầu tiên trong danh sách.

Nếu position > 1, hàm sẽ tìm đến node ở vị trí ngay trước vị trí cần thêm khách hàng mới (vị trí position - 1). Vòng lặp while duyệt qua danh sách để đến node có vị trí bằng position - 1. Biến currentPosition được sử dụng để đếm số lượng node đã duyệt. Nếu temp trở thành nullptr, điều đó có nghĩa là danh sách không đủ dài để chứa vị trí yêu cầu, và hàm sẽ in ra thông báo lỗi "Vi tri vuot qua so luong khach hang hien tai!". Nếu temp != nullptr (tức là vị trí hợp lệ), hàm sẽ thêm khách hàng vào đúng vị trí:

* Dòng newNode->next = temp->next; thiết lập con trỏ next của node mới trỏ đến node sau temp.
* Dòng temp->next = newNode; cập nhật con trỏ next của node temp để trỏ đến node mới, hoàn thành việc chèn khách hàng vào danh sách.

### 2.2.21 Hàm sắp xếp theo mã khách hàng



Hàm sortByID() có chức năng sắp xếp danh sách khách hàng liên kết theo mã khách hàng (mã KH). Sử dụng thuật toán sắp xếp nổi bọt (Bubble Sort) để thực hiện việc này.

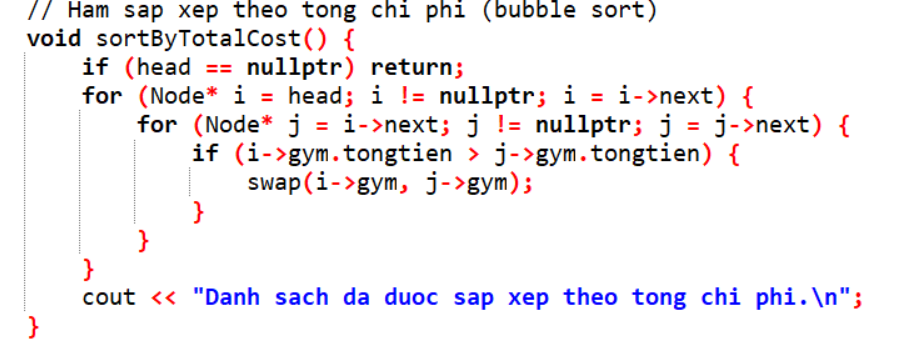
Hàm kiểm tra xem danh sách liên kết có rỗng hay không (khi head == nullptr). Nếu danh sách rỗng, không cần thực hiện sắp xếp nữa, hàm sẽ kết thúc ngay lập tức với câu lệnh return.

Thuật toán sắp xếp nổi bọt dùng: vòng lặp ngoài (i) duyệt qua tất cả các node trong danh sách, từ đầu (node đầu tiên, head) đến node cuối cùng, for (Node\* i = head; i != nullptr; i = i->next): i sẽ lần lượt trỏ đến từng node trong danh sách; vòng lặp trong (j) duyệt qua tất cả các node phía sau node hiện tại của vòng lặp ngoài, for (Node\* j = i->next; j != nullptr; j = j->next): j bắt đầu từ node ngay sau node i và di chuyển đến hết danh sách.

Trong mỗi cặp node i và j, nếu mã khách hàng của node i (i->gym.get\_makh()) lớn hơn mã khách hàng của node j (j->gym.get\_makh()), thì hoán đổi thông tin của hai node này để đảm bảo rằng mã khách hàng của i luôn nhỏ hơn hoặc bằng mã khách hàng của j. swap(i->gym, j->gym); dùng để hoán đổi đối tượng gym của hai node i và j.

Sau khi hoàn thành việc sắp xếp, hàm in ra thông báo "Danh sach da duoc sap xep theo ma khach hang." để người dùng biết rằng danh sách đã được sắp xếp theo mã khách hàng.

### 2.2.22 Hàm sắp xếp khách hàng theo tổng chi phí



Hàm sortByTotalCost() có chức năng sắp xếp danh sách khách hàng liên kết theo tổng chi phí (tổng tiền) của mỗi khách hang.

Hàm kiểm tra nếu danh sách rỗng (head == nullptr). Nếu danh sách không có phần tử nào, hàm sẽ kết thúc sớm mà không thực hiện sắp xếp.

Vòng lặp ngoài(i): duyệt qua tất cả các node trong danh sách, từ đầu (node head) đến node cuối cùng. i là con trỏ đại diện cho node hiện tại trong vòng lặp.

* for (Node\* i = head; i != nullptr; i = i->next): i bắt đầu từ node đầu tiên và di chuyển lần lượt qua các node tiếp theo trong danh sách.

Vòng lặp trong (j: Duyệt qua tất cả các node phía sau node hiện tại của vòng lặp ngoài. j bắt đầu từ node ngay sau node i.

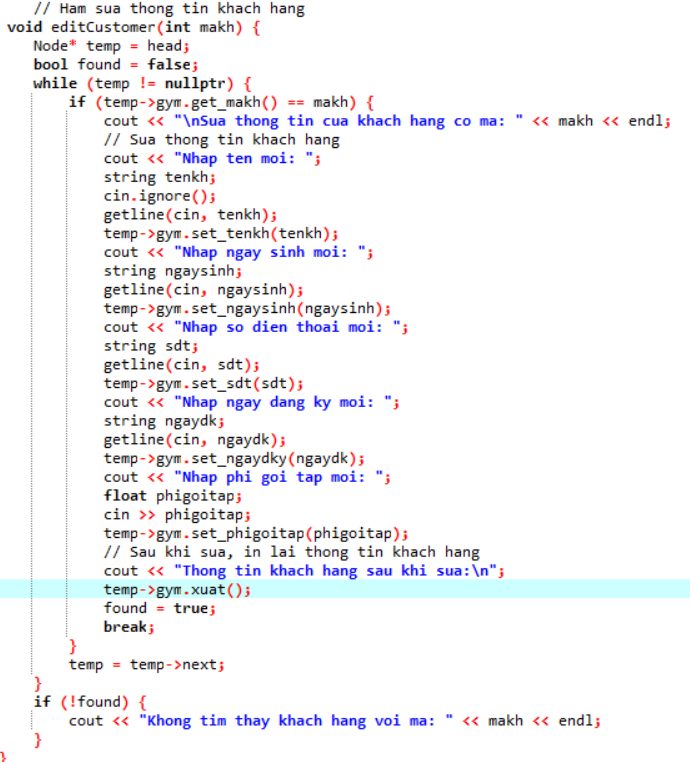
* for (Node\* j = i->next; j != nullptr; j = j->next): j bắt đầu từ node sau i và di chuyển qua các node tiếp theo trong danh sách.

Trong mỗi cặp node i và j, hàm so sánh tổng chi phí (tongtien) của hai khách hàng:

* Nếu tổng chi phí của khách hàng tại node i (i->gym.tongtien) lớn hơn tổng chi phí của khách hàng tại node j (j->gym.tongtien), hàm sẽ hoán đổi thông tin giữa hai khách hàng này. Điều này đảm bảo rằng sau mỗi vòng lặp, các khách hàng có tổng chi phí nhỏ hơn sẽ được đẩy lên phía trước trong danh sách.
* swap(i->gym, j->gym);: Hoán đổi các đối tượng gym của hai node i và j, từ đó đổi chỗ thông tin của hai khách hàng.

Sau khi hoàn thành việc sắp xếp, hàm in ra thông báo "Danh sach da duoc sap xep theo tong chi phi." để người dùng biết rằng danh sách đã được sắp xếp theo tổng chi phí của các khách hàng.

### 2.2.23Hàm sửa thông tin khách hàng



Hàm editCustomer(int makh) dùng để chỉnh sửa thông tin của khách hàng dựa trên mã khách hàng (makh) trong danh sách liên kết. Hàm bắt đầu bằng cách khởi tạo con trỏ temp trỏ đến node đầu tiên (head) và sử dụng biến cờ found (giá trị ban đầu là false) để xác định xem có tìm thấy khách hàng hay không.

Vòng lặp while duyệt qua từng node trong danh sách. Khi tìm thấy khách hàng có mã trùng với makh, hàm cho phép người dùng nhập thông tin mới, sau đó in thông tin đã cập nhật. Lúc này, found được đặt thành true và thoát vòng lặp bằng break.

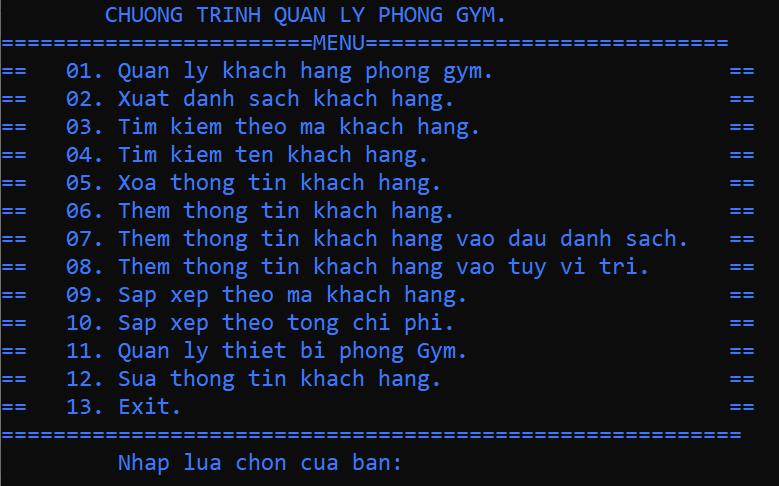
Nếu không tìm thấy khách hàng (khi temp duyệt hết danh sách), found vẫn là false, hàm sẽ thông báo rằng không tìm thấy khách hàng với mã đã cho.

## 2.3 Hàm menu()

Hàm menu và hàm main sẽ được trình bày rõ qua link github sau:

<https://github.com/thanhthe636/baitaplon.git>

Đây là giao diện của chương trình sau khi chạy trên màn hình console



# CHƯƠNG 3. KẾT QUẢ VÀ KIẾN NGHỊ

* 1. 3.1 Kết quả đạt được

**Quản lý khách hàng**: Thêm, sửa, xóa và tìm kiếm thông tin khách hàng nhanh chóng.

**Quản lý gói tập**: Thêm mới và sắp xếp gói tập theo tên hoặc chi phí.

**Sắp xếp và tìm kiếm**: Hỗ trợ sắp xếp khách hàng theo mã, chi phí và tìm kiếm theo mã hoặc tên.

**Quản lý thiết bị**: Theo dõi tình trạng hoạt động của thiết bị.

* 1. Hệ thống đảm bảo quản lý phòng gym hiệu quả, dễ sử dụng, giúp duy trì dịch vụ tốt cho khách hàng và tăng cường hiệu quả hoạt động của phòng gym.
  2. 3.2 Kiến nghị

Cải tiến hệ thống.

Hướng nghiên cứu và phát triển mới cho đề tài .

# PHỤ LỤC

Phụ lục 1: hướng dẫn cài đặt và sử dụng.

Truy cập vào đường link github này để có source code: <https://github.com/thanhthe636/baitaplon.git>

Source code gồm có các đoạn code ở trên và một vài hàm khác như hàm main chưa được nêu ở trên.

# TÀI LIỆU THAM KHẢO

1. Giáo trình ngôn ngữ lập trình c++, TS. Vũ Việt Vũ(Chủ biên), ThS. Phùng Thị Thu Hiền, NXB Khoa học và Kĩ thuật.
2. Giáo trình cấu trúc dữ liệu và giải thuật, PGS.Đỗ Xuân Lôi, Cấu trúc dữ liệu và giải thuật, Nhà xuất bản ĐH QG, 2016.
3. https://www.codeproject.com/Articles/6819/SendKeys-in-**C.**