

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
HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
ĐỀ TÀI: TOURIST MANAGEMENT SYSTEM

Giảng viên hướng dẫn:

ThS. TRẦN THỊ DUNG

MSSV:

Sinh viên thực hiện:

6551071031

TRẦN TIẾN HỢP

6551071067

VÕ HOÀNG PHÚC

6551071077

ĐỖ NHẬT THANH

6551071066

TRẦN HUỖNH HÒA PHÚC

Lớp : CQ.65.CNTT

Khoá : K65

Tp. Hồ Chí Minh, năm 2025

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
HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
ĐỀ TÀI: TOURIST MANAGEMENT SYSTEM

Giảng viên hướng dẫn:

ThS. TRẦN THỊ DUNG

MSSV:

Sinh viên thực hiện:

6551071031

TRẦN TIẾN HỢP

6551071067

VÕ HOÀNG PHÚC

6551071077

ĐỖ NHẬT THANH

6551071066

TRẦN HUỖNH HÒA PHÚC

Lớp : CQ.65.CNTT

Khoá : K65

Tp. Hồ Chí Minh, năm 2025

LỜI CẢM ƠN

Lời nói đầu tiên, em xin gửi tới Quý Thầy Cô Bộ môn Công nghệ Thông tin Trường Đại học Giao thông vận tải phân hiệu tại thành phố Hồ Chí Minh lời chúc sức khỏe và lòng biết ơn sâu sắc.

Trong suốt quá trình học tập và thực hiện bài tập lớn môn Lập trình hướng đối tượng và môn Cấu trúc dữ liệu và giải thuật, chúng em đã nhận được rất nhiều sự hỗ trợ và động viên quý báu từ cô giáo Trần Thị Dung – người đã trực tiếp giảng dạy và hướng dẫn chúng em trong suốt học kỳ vừa qua. Tuy đã cố gắng trong quá trình nghiên cứu tìm hiểu tuy nhiên do kiến thức còn hạn chế nên vẫn còn tồn tại nhiều thiếu sót. Vì vậy em rất mong nhận được sự đóng góp ý kiến của Quý thầy cô bộ môn để đề tài của em có thể hoàn thiện hơn.

Chúng em xin bày tỏ lòng biết ơn sâu sắc đến cô Trần Thị Dung đã tận tình giảng dạy, truyền đạt những kiến thức chuyên môn vững chắc cùng với những bài học kinh nghiệm thực tế quý báu. Những bài giảng tâm huyết, những chỉ dẫn tỉ mỉ, sự tận tâm và trách nhiệm của cô không chỉ giúp chúng em hiểu rõ hơn về môn Lập trình hướng đối tượng, Cấu trúc dữ liệu và giải thuật mà còn trang bị cho chúng em những tư duy logic, kỹ năng giải quyết vấn đề và phương pháp làm việc khoa học, đây hành trang vô cùng quý giá cho con đường học tập và sự nghiệp sau này.

Chúng em xin chân thành cảm ơn cô vì không chỉ dừng lại ở việc truyền đạt kiến thức, mà còn luôn động viên, tạo điều kiện kết nối với các anh chị cựu sinh viên để khích lệ tinh thần học hỏi, sáng tạo của chúng em. Sự tận tâm và nhiệt huyết của cô là nguồn động lực lớn giúp chúng em không ngừng cố gắng và hoàn thiện mình.

Một lần nữa, chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến cô Trần Thị Dung. Kính chúc cô luôn mạnh khỏe, hạnh phúc và đạt nhiều thành công trong sự nghiệp trồng người cao quý.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp. Hồ Chí Minh, ngày tháng năm

Giảng viên hướng dẫn

Th.S Trần Thị Dung

MỤC LỤC

LỜI CẢM ƠN	i
NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN.....	ii
MỤC LỤC	iii
BẢNG BIỂU, SƠ ĐỒ, HÌNH VẼ	vi
BẢNG PHÂN CHIA CÔNG VIỆC.....	vii
CHƯƠNG 1. MỞ ĐẦU	1
1.1 Tổng quan về đề tài	1
1.2 Mục tiêu nghiên cứu.....	1
1.3 Phạm vi và đối tượng nghiên cứu.....	2
1.4 Cấu trúc báo cáo	2
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	3
2.1 Tổng quan về ngôn ngữ lập trình C++	3
2.1.1 Ưu điểm và nhược điểm của C++	3
2.2 Cơ sở lý thuyết liên quan đến hệ thống	4
2.3 Tổng quan về lập trình hướng đối tượng (OOP)	10
2.3.1 Khái niệm về Lập trình hướng đối tượng.....	10
2.3.1.1 Đối tượng.....	10
2.3.1.2 Lớp.....	10
2.3.2 Các nguyên lý cơ bản của OOP.....	11
2.3.3 Hàm khởi tạo	12
2.3.4 Hàm hủy (Destructor).....	14
2.3.5 Hàm bạn (Friend function)	15
2.4 Cấu trúc dữ liệu và giải thuật (DSA).....	16

2.4.1 Danh sách liên kết đơn (Singly Linked List).....	16
CHƯƠNG 3 HỆ THỐNG QUẢN LÝ DU LỊCH.....	19
3.1 Tổng quan về Hệ thống	19
3.1.1 Mục tiêu.....	19
3.1.2 Cấu trúc dữ liệu	19
3.1.2.1 Các lớp cơ sở	20
3.1.2.2 Các lớp hỗ trợ	21
3.1.2.3 Cấu trúc Node và Danh sách	25
3.1.3 Các thao tác chính	27
3.1.3.1 Quản lý Khách hàng	27
3.1.3.3 Quản lý Phương tiện.....	28
3.1.3.4 Quản lý Địa danh.....	29
3.1.3.5 Quản lý Lưu trú	30
3.1.3.6 Quản lý Thanh toán	30
3.2.5 Đọc dữ liệu từ file.....	32
3.2.5.1 Danh sách các file.....	32
3.2.6 Các thuật toán sắp xếp được sử dụng	32
3.2.7 Đặc điểm nổi bật của hệ thống	33
3.2.7.1 Tính module hóa cao	33
3.2.7.2 Tính linh hoạt	34
3.2.7.3 Giao diện thân thiện.....	34
3.2.7.4 Quản lý bộ nhớ an toàn.....	34
3.2.7.5 Tích hợp dữ liệu.....	34

3.3 Các chức năng chính trong hệ thống	34
3.3.1 Giao diện người dùng	34
3.3.2 CHẾ ĐỘ ADMIN	35
3.3.3 QUẢN LÝ KHÁCH HÀNG	36
3.3.4 QUẢN LÝ NHÂN VIÊN	37
3.3.5 QUẢN LÝ PHƯƠNG TIỆN	38
3.3.6 QUẢN LÝ ĐỊA DANH.....	39
3.3.7 QUẢN LÝ LƯU TRỮ.....	39
3.3.8 QUẢN LÝ THANH TOÁN	40
3.3.9 QUẢN LÝ DỊCH VỤ.....	40
CHƯƠNG 4. KẾT QUẢ VÀ KIẾN NGHỊ.....	42
4.1 Kết quả đạt được.....	42
4.2 Kiến nghị	43
4.2.1. Về xử lý lỗi và kiểm tra hợp lệ.....	43
4.2.2. Về lưu trữ và quản lý dữ liệu.....	43
4.2.3. Về hiệu năng và tối ưu.....	44
4.2.4. Về giao diện người dùng	44
4.2.5. Về chức năng bổ sung	44
TÀI LIỆU THAM KHẢO.....	45

BẢNG BIỂU, SƠ ĐỒ, HÌNH VẼ

Bảng 2.1 Sự khác nhau giữa 2 kiểu dữ liệu người dùng định nghĩa được liệt kê	7
Bảng 2.2 Các toán tử không thể nạp chồng.....	8
Bảng 2.3: Các chế độ mở file trong C++.....	9
Hình 2.1 Tính đóng gói	11
Hình 2.2 Danh sách liên kết đơn	17
Hình 3.1 Sơ đồ các Class của hệ thống trong UML.....	20
Hình 3.2 Lớp Người và Lớp Nhân Viên.....	21
Hình 3.3 Class Phương Tiện và class Lưu Trữ	21
Hình 3.4 Class Date	22
Hình 3.5 Class Name	22
Hình 3.6 Class Địa Chỉ	22
Hình 3.7 Class Thời Gian	23
Hình 3.8 Class Địa Danh	23
Hình 3.8 Class Thanh Toán	24
Hình 3.9 Class Danh Sách Khách Hàng	25
Hình 3.10 Class Danh Sách Nhân Viên.....	26
Hình 3.11 Class Danh Sách Phương Tiện	26
Hình 3.12 Class Danh Sách Nhân Viên.....	26
Hình 3.13 Class Danh Sách Thanh Toán.....	27
Hình 3.14 Class Danh Sách Dịch Vụ	27
Hình 3.15 Danh sách file TXT	32
Hình 3.16 Quy trình đọc file khởi động	32

BẢNG PHÂN CHIA CÔNG VIỆC

STT	Họ và tên	Công việc	Đánh giá
1	Trần Tiến Hợp (Nhóm trưởng)	<ul style="list-style-type: none"> - Code class KháchHang - Code class CongTy - Code class DichVu - Code các class con liên quan - Check, fix lỗi trong mã nguồn 	10/10
2	Võ Hoàng Phúc	<ul style="list-style-type: none"> - Code class NhanVien - Code class DiaChi - Code clclasThanhToan - Code các class con liên quan - Viết báo cáo 	10/10
3	Đỗ Nhật Thanh	<ul style="list-style-type: none"> - Code class Name - Code class Date - Code class LuuTru - Code các class con liên quan - Check, fix lỗi trong mã nguồn 	10/10
4	Trần Huỳnh Hòa Phúc	<ul style="list-style-type: none"> - Code class PhuongTien - Code class DiaDanh - Code class Time - Code các class con liên quan - Thiết kế slide báo cáo 	10/10

CHƯƠNG 1. MỞ ĐẦU

1.1 Tổng quan về đề tài

Trong lĩnh vực phát triển phần mềm hiện đại, lập trình hướng đối tượng (OOP) và cấu trúc dữ liệu cùng giải thuật (CTDL>) giữ vai trò nền tảng trong việc xây dựng các hệ thống có khả năng mở rộng, dễ bảo trì và đạt hiệu suất cao. OOP cung cấp mô hình tổ chức chương trình dựa trên các đối tượng, giúp mô phỏng các thực thể trong thế giới thực thông qua việc đóng gói dữ liệu và hành vi, từ đó hỗ trợ lập trình viên thiết kế phần mềm một cách trực quan, linh hoạt và tái sử dụng cao. Song song với đó, CTDL> cung cấp các phương pháp lưu trữ, tổ chức và xử lý dữ liệu tối ưu, là cơ sở để giải quyết những bài toán phức tạp về hiệu năng và khả năng xử lý.

Việc nắm vững các nguyên lý của OOP như kế thừa, đa hình, trừu tượng hóa và đóng gói, kết hợp với hiểu biết sâu sắc về các cấu trúc dữ liệu như danh sách liên kết, cây, hàng đợi, ngăn xếp, đồ thị, cùng các giải thuật tìm kiếm – sắp xếp, không chỉ giúp lập trình viên xây dựng chương trình một cách khoa học, mà còn hình thành tư duy logic, khả năng tối ưu mã nguồn và giải quyết vấn đề hiệu quả. Đây cũng là nền tảng quan trọng để tiếp cận các lĩnh vực nâng cao như phát triển phần mềm quy mô lớn, thiết kế hệ thống, trí tuệ nhân tạo và kỹ thuật tối ưu hóa.

1.2 Mục tiêu nghiên cứu

Đề tài này nhằm mục tiêu:

Trình bày một cách có hệ thống các kiến thức lý thuyết liên quan đến việc xây dựng và vận hành một hệ thống quản lý du lịch. Điều này bao gồm tổng hợp các hiểu biết về tổ chức dữ liệu, các mô hình quản lý tour, quản lý khách hàng, quy trình đặt tour, xử lý thanh toán, cùng những yêu cầu cơ bản trong phát triển các phần mềm quản trị dịch vụ du lịch.

Phân tích mối quan hệ giữa các thành phần dữ liệu chính trong hệ thống như danh sách tour, thông tin khách hàng, thông tin hướng dẫn viên, lịch trình di chuyển, danh sách đặt tour và các thuộc tính liên quan. Qua đó, xác định cách thức lưu trữ, kết nối và

trao đổi dữ liệu giữa các cấu trúc nhằm đảm bảo tính khoa học, dễ mở rộng và thuận tiện cho việc truy xuất cũng như xử lý thông tin.

Áp dụng các kiến thức về lập trình, cấu trúc dữ liệu và xử lý thông tin để xây dựng bộ chức năng quản lý du lịch bao gồm: thêm, sửa, xóa tour; quản lý danh sách khách hàng; xử lý đặt tour; thống kê số lượng khách và doanh thu; và các thao tác hỗ trợ khác. Việc triển khai các chức năng này không chỉ giúp hệ thống đáp ứng yêu cầu bài toán mà còn giúp người học rèn luyện kỹ năng viết mã, tư duy hệ thống và khả năng tổ chức dữ liệu trong các chương trình thực tiễn.

1.3 Phạm vi và đối tượng nghiên cứu

Đối tượng nghiên cứu chính của đề tài là hệ thống quản lý tour (Tour Management System) được xây dựng bằng ngôn ngữ lập trình C/C++, cụ thể tập trung vào các kỹ thuật xử lý dữ liệu bằng cấu trúc (struct), lớp (class – trong C++), thao tác với file và tổ chức chương trình để quản lý thông tin tour, khách hàng và đặt tour.

Phạm vi của báo cáo không bao gồm các thư viện ngoài chuẩn, các công nghệ lập trình khác ngoài C và C++, hoặc các mô hình hệ thống phức tạp như cơ sở dữ liệu SQL, lập trình web hay ứng dụng giao diện đồ họa.

1.4 Cấu trúc báo cáo

1.1.1 Chương 1: Mở đầu.

1.1.2 Chương 2: Cơ sở lý thuyết.

1.1.3 Chương 3: Chương trình.

1.1.4 Chương 4: Kết quả và kiến nghị.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về ngôn ngữ lập trình C++

C++ cho phép các nhà phát triển phần mềm định nghĩa các kiểu dữ liệu của riêng họ và thao tác chúng bằng các hàm và phương thức. Nó cũng cho phép lập trình cấp thấp và cung cấp quyền truy cập vào bộ nhớ, giúp thực thi mã nhanh chóng và hiệu quả. Nó cũng hỗ trợ lập trình tổng quát bằng cách sử dụng các mẫu, cho phép mã được viết dưới dạng tổng quát và tái sử dụng cho các kiểu dữ liệu khác nhau.

2.1.1 Ưu điểm và nhược điểm của C++

Ưu điểm:

- Hiệu suất cao: C++ cung cấp khả năng thao tác bộ nhớ cấp thấp và kiểm soát tối ưu tài nguyên hệ thống, phù hợp cho các ứng dụng yêu cầu hiệu suất cao như game, hệ thống nhúng và phần mềm hệ thống.
- Tính linh hoạt: Hỗ trợ lập trình hướng đối tượng, tổng quát và hàm, giúp giải quyết nhiều loại vấn đề khác nhau.
- Nền tảng vững chắc: Là nền tảng cho nhiều thư viện và framework, hỗ trợ phát triển phần mềm phức tạp.
- Ứng dụng rộng rãi: Sử dụng trong nhiều lĩnh vực như hệ điều hành, trình biên dịch, phần mềm mô phỏng khoa học, tài chính, và nhiều lĩnh vực khác.
- Tiêu chuẩn hóa và cộng đồng: C++ được tiêu chuẩn hóa bởi ISO và có một cộng đồng lập trình viên lớn hỗ trợ.

Nhược điểm:

- Độ phức tạp và khó học: C++ có nhiều tính năng và cú pháp phức tạp, làm cho việc học và sử dụng trở nên khó khăn, đặc biệt đối với người mới bắt đầu.
- Quản lý bộ nhớ thủ công: C++ yêu cầu lập trình viên phải tự quản lý bộ nhớ, dễ dẫn đến rò rỉ bộ nhớ và các vấn đề khác nếu không được xử lý đúng cách.
- Rủi ro bảo mật: Truy cập trực tiếp vào bộ nhớ có thể gây ra các lỗi bảo mật nếu không được kiểm soát cẩn thận.

- Không có thu gom rác: Thiếu cơ chế tự động giải phóng bộ nhớ, lập trình viên phải tự tay giải phóng bộ nhớ không cần thiết.
- Cú pháp phức tạp: Cú pháp của C++ phức tạp hơn nhiều so với các ngôn ngữ lập trình khác, làm cho việc đọc và viết mã tốn nhiều thời gian hơn.
- Thời gian biên dịch lâu: C++ thường có thời gian biên dịch lâu hơn so với các ngôn ngữ có kiểu động hoặc biên dịch ngay lập tức.

2.1.4 Lý do chọn ngôn ngữ C++ cho đề tài

C++ được lựa chọn vì đây là ngôn ngữ mạnh mẽ hỗ trợ lập trình hướng đối tượng, giúp mô hình hóa các thành phần của hệ thống quản lý tour như Tour, Khách hàng hay Booking một cách rõ ràng, dễ mở rộng và dễ bảo trì, từ đó phù hợp cho việc xây dựng một hệ thống quản lý có cấu trúc.

2.2 Cơ sở lý thuyết liên quan đến hệ thống

2.2.1 Kiểu dữ liệu (Data Types)

Kiểu dữ liệu (Data Types) là một hệ thống phân loại, giúp máy tính hiểu rõ loại thông tin mà một biến (variable) đang lưu trữ và những phép toán nào có thể thực hiện trên dữ liệu đó. Nó là khái niệm nền tảng trong lập trình (programming), đảm bảo dữ liệu được xử lý chính xác.

2.2.1.1 Enum – Kiểu liệt kê

Kiểu liệt kê (enum) là một kiểu dữ liệu do người dùng định nghĩa, chứa một tập hợp các hằng số nguyên được đặt tên. Nó được dùng để gán tên có ý nghĩa cho các giá trị nguyên, giúp chương trình dễ đọc và bảo trì. Enum phải được định nghĩa trước khi chúng ta có thể sử dụng nó trong chương trình.

Không bắt buộc phải gán tất cả các hằng số. Tên tiếp theo sẽ được tự động gán giá trị bằng cách tăng giá trị của tên trước đó thêm 1. Ví dụ: n3 ở đây sẽ là (val2 + 1). Ngoài ra, chúng ta có thể gán giá trị theo bất kỳ thứ tự nào, nhưng chúng phải là số nguyên.

Kích thước của Enum

Định nghĩa enum không được cấp phát bộ nhớ, nó chỉ được cấp phát cho các biến enum. Thông thường, enum được triển khai dưới dạng số nguyên, do đó kích thước của chúng giống như int. Tuy nhiên, một số trình biên dịch có thể sử dụng short int (hoặc thậm chí là kiểu nhỏ hơn) để biểu diễn enum. Chúng ta có thể sử dụng toán tử sizeof để kiểm tra kích thước của biến enum.

Ứng dụng của Enum

Enum được sử dụng rộng rãi trong nhiều ứng dụng thực tế trong lập trình, chẳng hạn như:

- Biểu diễn trạng thái: Enum thường được dùng để biểu diễn các trạng thái hoặc chế độ khác nhau trong máy trạng thái, như trong phát triển trò chơi hoặc hệ thống điều khiển quy trình.
- Mã lỗi: Enum hữu ích trong việc xác định một tập hợp các mã lỗi liên quan với tên có ý nghĩa, giúp việc gỡ lỗi và bảo trì mã dễ dàng hơn.
- Lựa chọn menu hoặc đầu vào của người dùng: Enum hữu ích để xử lý các tùy chọn menu hoặc đầu vào của người dùng và có thể biểu diễn nhiều loại đầu vào của người dùng như các ngày trong tuần.
- Quyền tệp: Enum hữu ích trong việc xử lý quyền tệp trong hệ điều hành hoặc hệ thống quản lý tệp, với các mức quyền khác nhau như ĐỌC, GHI và THỰC THI.

Một số kiểu dữ liệu enum đã được dùng trong hệ thống:

- TrangThai (CON_TRONG, DA_DAT, DANG_BAO_TRI)
- LoaiThanhToan (TIEN_MAT, THE, VI_DIEN_TU)
- LoaiXeKhach, LoaiXeHoi, SoGhe

2.2.1.3 Union – Hợp kiểu

Union cũng là kiểu dữ liệu mà người dùng tự định nghĩa tương tự như struct, union cũng có các thành phần dữ liệu để mô tả thông tin của đối tượng mà bạn đang muốn lưu trữ

Tuy nhiên điểm khác biệt với struct đó là struct thì các thành phần của nó có thể lưu trữ giá trị đồng thời tại cùng một thời điểm còn union thì ngược lại, tại mỗi thời điểm bạn chỉ có một thuộc tính của union lưu trữ 1 giá trị nào đó.

Tất cả các thành phần của union sẽ chia sẻ chung 1 vùng nhớ

Cú pháp:

```
union union_name{  
  
    data_type1 member1;  
  
    data_type2 member2;  
  
    data_type3 member3;  
  
    ....  
  
    data_typeN memberN;  
  
};
```

Để truy cập các thành phần của union bạn cũng dùng toán tử '.' với biến union bình thường hoặc toán tử '->' với kiểu con trỏ union.

Sự giống nhau và khác nhau giữa Union và Struct

Union và struct giống nhau bao gồm :

- Luôn là kiểu dữ liệu mà người dùng tự định nghĩa ra nhằm phục vụ các bài toán thực tế
- Có các thuộc tính thành phần
- Truy cập thuộc tính thông qua toán tử '.' hoặc toán tử '->' đối với biến kiểu con trỏ

Struct	Union
Kích thước của struct thường lớn hơn hoặc bằng tổng tất cả các kích thước của các thuộc tính thành phần	Kích thước của union bằng với kích thước của thuộc tính có kích thước lớn nhất của nó
struct có thể lưu trữ đồng thời nhiều giá trị cho các thuộc tính thành phần ở cùng một thời điểm	Chỉ có 1 thuộc tính thành phần của union có thể lưu giá trị tại 1 thời điểm
Khai báo với từ khóa struct	Khai báo với từ khóa union

Bảng 2.1 Sự khác nhau giữa 2 kiểu dữ liệu người dùng định nghĩa được liệt kê

2.2.1.4 Operator Overloading - Nạp Chồng Toán Tử

Nạp chồng toán tử (overload operator) là bạn định nghĩa lại toán tử đã có trên kiểu dữ liệu người dùng tự định nghĩa để dễ dàng thể hiện các câu lệnh trong chương trình.

Ngôi	Group	Operators
1 Ngôi (Unary)	Tăng giảm	++, --
	Dấu số học	+, -
	Logic	!, ~
	Con trỏ	*, &
	Ép kiểu	int, float, double, ...
2 Ngôi (Binary)	Số học	+, -, *, /, %, +=, -=, *=, /=, %=
	So sánh	>, <, ==, >=, <=, !=
	Logic	&&, , &,

	Nhập xuất	<<, >>
	Gán	=
	Lấy chỉ số mảng	[]

Bảng 2.1 Các toán tử có thể nạp chồng

Operators	Meaning
.	Truy xuất phần tử
.*	Truy xuất con trỏ phần tử
::	Toán tử ::
? :	Toán tử điều kiện
#	Chỉ thị tiền xử lý
##	Chỉ thị tiền xử lý

Bảng 2.2 Các toán tử không thể nạp chồng

2.2.1.5 Xử Lý File (File I/O)

Trong lập trình C++, File I/O (File Input/Output) là quá trình trao đổi dữ liệu giữa chương trình và các tệp tin trên bộ nhớ ngoài. Thông qua cơ chế này, chương trình có thể:

- Đọc dữ liệu từ file (Input)
- Ghi dữ liệu ra file (Output)

File I/O giúp lưu trữ dữ liệu một cách lâu dài, hỗ trợ quản lý thông tin và liên kết giữa các lần chạy chương trình.

Luồng vào/ra trong C++

C++ cung cấp các công cụ xử lý tệp tin thông qua thư viện <fstream>. Thư viện này định nghĩa ba lớp quan trọng:

ifstream – dùng để đọc dữ liệu từ file.

ofstream – dùng để ghi dữ liệu ra file.

fstream – cho phép vừa đọc vừa ghi trên cùng một file.

Các lớp trên hoạt động dựa trên mô hình luồng (stream), tương tự như cin và cout.

Mở và đóng file

Mở File

File có thể được mở bằng hai cách:

Mở trực tiếp khi khai báo đối tượng

```
ifstream fin("input.txt");
```

```
ofstream fout("output.txt");
```

Mở thông qua hàm **open()**

```
fstream f;
```

```
f.open("data.txt", ios::in | ios::out);
```

Chế độ	Tác dụng
ios::in	Mở file để đọc
ios::out	Mở file để ghi
ios::app	Ghi nối vào cuối file
ios::binary	Mở file ở chế độ nhị phân
ios::trunc	Xóa nội dung cũ của file khi ghi
ios::ate	Con trỏ file được đặt ở cuối ngay sau khi mở

Bảng 2.3: Các chế độ mở file trong C++

Đóng File

Sau khi sử dụng, cần đóng file bằng hàm close():

```
fin.close();
```

```
fout.close();
```

Việc đóng file giúp giải phóng tài nguyên và đảm bảo dữ liệu được ghi đúng vào bộ nhớ.

Ưu điểm của File I/O trong C++

- Hỗ trợ đầy đủ các thao tác nhập/xuất thông qua mô hình luồng.
- Tích hợp tốt với lập trình hướng đối tượng.
- Dễ sử dụng nhờ toán tử >> và << .
- Hỗ trợ nhiều chế độ thao tác file linh hoạt.
- Cho phép xử lý cả file văn bản và file nhị phân.

2.3 Tổng quan về lập trình hướng đối tượng (OOP)

Lập trình hướng đối tượng (Object Oriented Programming – OOP) là một trong những kỹ thuật lập trình rất quan trọng và sử dụng nhiều hiện nay. Hầu hết các ngôn ngữ lập trình hiện nay như Java, PHP, .NET, Ruby, Python... đều hỗ trợ OOP.

2.3.1 Khái niệm về Lập trình hướng đối tượng

Lập trình hướng đối tượng (OOP) là một kỹ thuật lập trình cho phép lập trình viên tạo ra các đối tượng trong code trừu tượng hóa các đối tượng.

2.3.1.1 Đối tượng

Một đối tượng bao gồm 2 thông tin: thuộc tính và phương thức.

- Thuộc tính chính là những thông tin, đặc điểm của đối tượng. Ví dụ: con người có các đặc tính như mắt, mũi, tay, chân...
- Phương thức là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: một người sẽ có thể thực hiện hành động nói, đi, ăn, uống, . . .

2.3.1.2 Lớp

Một lớp là một kiểu dữ liệu bao gồm các thuộc tính và các phương thức được định nghĩa từ trước. Đây là sự trừu tượng hóa của đối tượng. Khác với kiểu dữ liệu thông thường, một lớp là một đơn vị (trừu tượng) bao gồm sự kết hợp giữa các phương thức và

các thuộc tính. Hiểu nôm na hơn là các đối tượng có các đặc tính tương tự nhau được gom lại thành một lớp đối tượng.

Lớp bạn có thể hiểu nó như là khuôn mẫu, đối tượng là một thực thể thể hiện dựa trên khuôn mẫu đó. Ví dụ: Ta nói về loài chó, bạn có thể hiểu nó là class (lớp) chó có:

- Các thông tin, đặc điểm: 4 chân, 2 mắt, có đuôi, có chiều cao, có cân nặng, màu lông...
- Các hành động như: sửa, đi, ăn, ngủ...

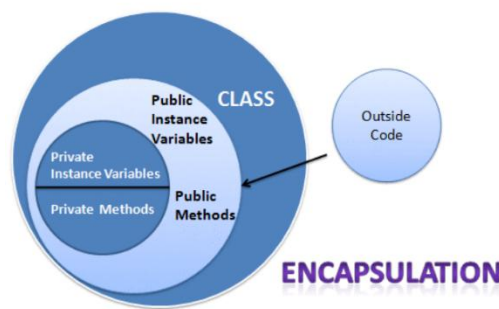
Đối tượng thì chính là con chó Phú Quốc ta đang nuôi trong nhà cũng mang đặc tính của lớp chó.

2.3.2 Các nguyên lý cơ bản của OOP

2.3.2.1 Tính đóng gói (Encapsulation)

Các dữ liệu và phương thức có liên quan với nhau được đóng gói thành các lớp để tiện cho việc quản lý và sử dụng. Tức là mỗi lớp được xây dựng để thực hiện một nhóm chức năng đặc trưng của riêng lớp đó.

Ngoài ra, đóng gói còn để che giấu một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không thể nhìn thấy.



Hình 2.1 Tính đóng gói

Nói chung trạng thái đối tượng không hợp lệ thường do: chưa được kiểm tra tính hợp lệ, các bước thực hiện không đúng trình tự hoặc bị bỏ qua nên trong OOP có một quy tắc quan trọng cần nhớ đó là phải luôn khai báo các trạng thái bên trong của đối tượng là private và chỉ cho truy cập qua các publi/protected method/property.

2.3.2.3 Tính kế thừa (Inheritance)

Nó cho phép xây dựng một lớp mới dựa trên các định nghĩa của lớp đã có. Có nghĩa là lớp cha có thể chia sẻ dữ liệu và phương thức cho các lớp con. Các lớp con khỏi phải định nghĩa lại, ngoài ra có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới. Tái sử dụng mã nguồn 1 cách tối ưu, tận dụng được mã nguồn. Một số loại kế thừa thường gặp: đơn kế thừa, đa kế thừa, kế thừa đa cấp, kế thừa thứ bậc.

Khi bắt đầu xây dựng ứng dụng chúng ta sẽ bắt đầu thiết kế định nghĩa các lớp trước. Thông thường một số lớp có quan hệ với những lớp khác, chúng có những đặc tính giống nhau.

2.3.2.4 Tính đa hình (Polymorphism)

Tính đa hình là một hành động có thể được thực hiện bằng nhiều cách khác nhau. Đây lại là một tính chất có thể nói là chứa đựng hầu hết sức mạnh của lập trình hướng đối tượng.

2.3.2.5 Tính trừu tượng (Abstraction)

Trừu tượng có nghĩa là tổng quát hóa một cái gì đó lên, không cần chú ý chi tiết bên trong. Nó không màng đến chi tiết bên trong là gì và người ta vẫn hiểu nó mỗi khi nghe về nó. Ở đây trong lập trình OOP, tính trừu tượng nghĩa là chọn ra các thuộc tính, phương thức của đối tượng cần cho việc giải quyết bài toán đang lập trình. Vì một đối tượng có rất nhiều thuộc tính phương thức, nhưng với bài toán cụ thể không nhất thiết phải chọn tất cả.

2.3.3 Hàm khởi tạo

Hàm khởi tạo trong c++ hay còn gọi là constructor trong C++ là các hàm thành viên được tự động được thực thi khi chương trình tạo ra một đối tượng (instance) từ class, và được sử dụng với mục đích chính là khởi tạo các biến thành viên trong class đó.

Hàm khởi tạo trong c++ có 2 tính chất như sau:

- Tên hàm khởi tạo phải trùng với tên của class

- Hàm không mang kiểu dữ liệu trong nó, cũng như không sử dụng **void** khi khai báo nó.

3 loại hàm khởi tạo trong C++ phụ thuộc vào việc có tham số trong hàm hay không. Và nếu một hàm khởi tạo không chứa tham số trong nó thì chúng ta gọi nó là Hàm tạo mặc định (default constructor) trong C++

Hàm khởi tạo mặc định

Là hàm khởi tạo không có tham số.

Nếu người lập trình không tự viết, C++ sẽ tự tạo một constructor mặc định *rỗng*.

vd:

```
class SinhVien {  
private:  
    int tuoi;  
public:  
    SinhVien() {    // Constructor mặc định  
        tuoi = 18;  
    }  
};
```

Hàm khởi tạo có tham số

Là constructor **có truyền tham số**, dùng để khởi tạo dữ liệu cho đối tượng ngay lúc tạo.

Vd:

```
class SinhVien {  
private:  
    int tuoi;  
public:  
    SinhVien(int t) { // Constructor có tham số  
        tuoi = t;  
    }  
};
```

```
}  
};
```

Hàm khởi tạo sao chép

Là constructor dùng để **sao chép** dữ liệu từ một đối tượng này sang đối tượng khác.

Dạng chuẩn: *ClassName(const ClassName &obj);*

Vd:

```
class SinhVien {  
private:  
    int tuoi;  
public:  
    SinhVien(int t) {  
        tuoi = t;  
    }  
    SinhVien(const SinhVien &sv) { // Constructor sao chép  
        tuoi = sv.tuoi;  
    }  
};
```

2.3.4 Hàm hủy (Destructor)

Hàm hủy (Destructor) trong C++ ngược lại với hàm xây dựng, trong khi hàm xây dựng dùng để khởi tạo giá trị cho đối tượng thì hàm hủy dùng để hủy đối tượng.

Chỉ có duy nhất một hàm hủy trong 1 lớp. Hàm hủy tự động được gọi. Nếu như chúng ta không định nghĩa hàm hủy thì mặc định trình biên dịch sẽ tự tạo ra 1 hàm hủy mặc nhiên

Cũng giống như hàm xây dựng, hàm hủy được định nghĩa có cùng tên với tên lớp, không có bất cứ kiểu gì trả về kể cả kiểu void, tuy nhiên phải có dấu ~ trước tên của hàm hủy.

Cú pháp

Cú pháp của hàm hủy (Destructor) trong C++ như sau:

```
~TenLop() { };
```

Ví dụ cụ thể là lớp nhân viên, thì chúng ta sẽ tạo hàm hủy cho lớp nhân viên như sau:

```
class NhanVien {  
  
    public:  
  
        ~NhanVien(){};  
  
};
```

Khi Nào Hàm Destructor Được Gọi

Hàm hủy (Destructor) trong C++ được gọi tự động lúc đối tượng đi ra khỏi phạm vi:

- Kết thúc hàm
- Kết thúc chương trình
- Kết thúc 1 block
- Toán tử delete được gọi
- Có hai hạn chế lúc dùng hàm hủy đó là:
 - Chúng ta không thể lấy địa chỉ của nó
 - Lớp con không có thừa kế hàm hủy từ lớp cha của nó

2.3.5 Hàm bạn (Friend function)

Hàm bạn của một lớp không phải là hàm thành viên nên nó không phụ thuộc vào lớp và có thể định nghĩa ở trong hoặc ngoài lớp.

Hàm bạn có thể truy cập trực tiếp các thành viên private và một lớp có thể có nhiều hàm bạn

Cú pháp khai báo hàm bạn đơn giản là thêm từ khóa friend trước hàm bình thường ta vẫn sử dụng.

2.3.2.6 Các ưu điểm của lập trình hướng đối tượng

- Dựa trên nguyên lý kế thừa, trong quá trình mô tả các lớp có thể loại bỏ những chương trình bị lặp, dư. Và có thể mở rộng khả năng sử dụng các lớp mà không cần thực hiện lại. Tối ưu và tái sử dụng code hiệu quả.
- Đảm bảo rút ngắn thời gian xây dựng hệ thống và tăng năng suất thực hiện.
- Sự xuất hiện của 2 khái niệm mới là lớp và đối tượng chính là đặc trưng của phương pháp lập trình hướng đối tượng. Nó đã giải quyết được các khuyết điểm của phương pháp lập trình hướng cấu trúc để lại. Ngoài ra 2 khái niệm này đã giúp biểu diễn tốt hơn thế giới thực trên máy tính.

2.3.2.7 Lợi ích của OOP trong việc xây dựng hệ thống quản lý tour

Lập trình hướng đối tượng (OOP) mang lại nhiều lợi ích quan trọng cho việc xây dựng hệ thống quản lý tour. Nhờ khả năng mô hình hóa các thực thể như Tour, Khách hàng và Booking thành các lớp độc lập, OOP giúp chương trình trở nên rõ ràng, dễ tổ chức và dễ bảo trì. Tính đóng gói giúp quản lý dữ liệu an toàn và tránh lỗi truy cập trực tiếp, trong khi tính kế thừa và đa hình cho phép mở rộng chức năng mà không cần chỉnh sửa quá nhiều mã nguồn cũ. Nhờ đó, hệ thống có thể phát triển linh hoạt, dễ thêm mới chức năng và phù hợp với các bài toán quản lý quy mô lớn.

2.4 Cấu trúc dữ liệu và giải thuật (DSA)

2.4.1 Danh sách liên kết đơn (Singly Linked List)

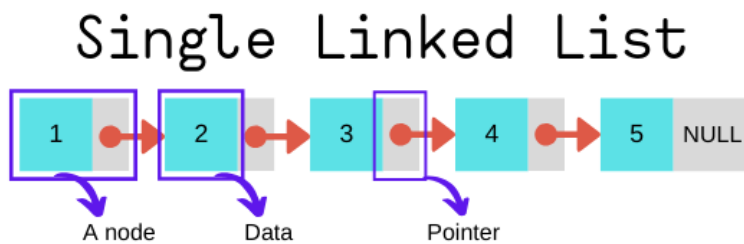
Danh sách liên kết đơn (Single Linked List) là một cấu trúc dữ liệu động, nó là một danh sách mà mỗi phần tử đều liên kết với phần tử đứng sau nó trong danh sách. Mỗi phần tử (được gọi là một node hay nút) trong danh sách liên kết đơn là một cấu trúc có hai thành phần:

- Thành phần dữ liệu: lưu thông tin về bản thân phần tử đó.

- Thành phần liên kết: lưu địa chỉ phần tử đứng sau trong danh sách, nếu phần tử đó là phần tử cuối cùng thì thành phần này bằng NULL.

Vd:

```
struct NodeNhanVien {
    NhanVien* data;    // Con trỏ đến đối tượng dữ liệu
    NodeNhanVien* next; // Con trỏ đến node tiếp theo
};
```



Hình 2.2 Danh sách liên kết đơn

Đặc điểm của danh sách liên kết đơn

Do danh sách liên kết đơn là một cấu trúc dữ liệu động, được tạo nên nhờ việc cấp phát động nên nó có một số đặc điểm sau đây:

- Được cấp phát bộ nhớ khi chạy chương trình.
- Có thể thay đổi kích thước qua việc thêm, xóa phần tử.
- Kích thước tối đa phụ thuộc vào bộ nhớ khả dụng của RAM.
- Các phần tử được lưu trữ ngẫu nhiên (không liên tiếp) trong RAM.

Và do tính liên kết của phần tử đầu và phần tử đứng sau nó trong danh sách liên kết đơn, nó có các đặc điểm sau:

- Chỉ cần nắm được phần tử đầu và cuối là có thể quản lý được danh sách
- Truy cập tới phần tử ngẫu nhiên phải duyệt từ đầu đến vị trí đó
- Chỉ có thể tìm kiếm tuyến tính một phần tử

Ưu điểm và Nhược điểm

Ưu điểm:

- Kích thước động, có thể mở rộng hoặc thu hẹp tùy ý
- Thao tác thêm/xóa phần tử hiệu quả khi đã biết vị trí ($O(1)$)
- Không lãng phí bộ nhớ như mảng tĩnh
- Dễ dàng triển khai các cấu trúc dữ liệu khác (Stack, Queue)

Nhược điểm:

- Truy cập ngẫu nhiên chậm ($O(n)$)
- Tốn thêm bộ nhớ để lưu con trỏ next
- Không cache-friendly (các node có thể nằm rời rạc trong bộ nhớ)
- Khó debug và dễ bị lỗi con trỏ

Các thao tác cơ bản:

- Thêm phần tử (Insertion)
- Xóa phần tử
- Tìm kiếm (Search)
- Duyệt danh sách

CHƯƠNG 3 HỆ THỐNG QUẢN LÝ DU LỊCH

3.1 Tổng quan về Hệ thống

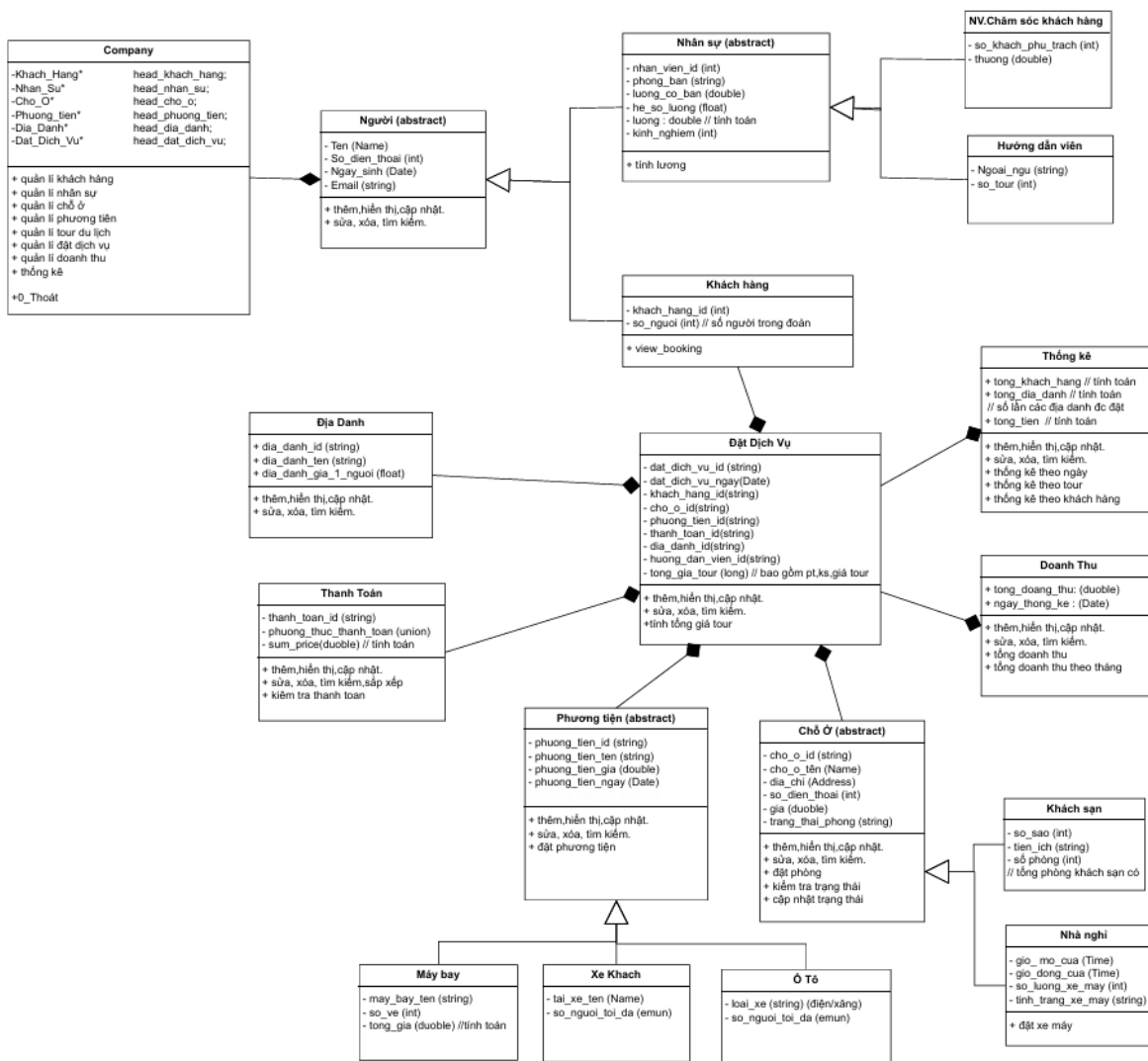
3.1.1 Mục tiêu

Mục tiêu của đề tài là xây dựng một hệ thống quản lý du lịch toàn diện bằng ngôn ngữ lập trình C++, áp dụng lập trình hướng đối tượng (OOP) và sử dụng danh sách liên kết đơn làm cấu trúc dữ liệu chính. Chương trình hỗ trợ quản lý đa dạng các đối tượng trong lĩnh vực du lịch bao gồm: khách hàng, nhân viên, phương tiện, địa danh, lưu trú, thanh toán và dịch vụ du lịch. Hệ thống cung cấp các thao tác quản lý cơ bản như thêm, sửa, xóa, tìm kiếm, sắp xếp và hiển thị thông tin, đồng thời tích hợp khả năng đọc/ghi dữ liệu từ file văn bản để phục vụ cho việc lưu trữ và khôi phục dữ liệu.

3.1.2 Cấu trúc dữ liệu

Chương trình sử dụng cấu trúc dữ liệu danh sách liên kết đơn (singly linked list) để quản lý các đối tượng. Mỗi danh sách bao gồm các node, trong đó mỗi node chứa con trỏ tới đối tượng dữ liệu tương ứng và con trỏ tới node tiếp theo.

3.1.2.1 Các lớp cơ sở



Hình 3.1 Sơ đồ các Class của hệ thống trong UML

Hệ thống được thiết kế theo mô hình phân cấp với các lớp cơ sở trừu tượng:

```
class Nguoi {
protected:
    Name ten;
    string so_dien_thoai;
    Date ngay_sinh;
    string email;

public:
    Nguoi();
    virtual ~Nguoi();

    void set_ten(const Name& ten);
    void set_so_dien_thoai(string so_dien_thoai);
    void set_ngay_sinh(const Date& ngay_sinh);
    void set_email(const string& email);

    Name get_ten();
    string get_so_dien_thoai();
    Date get_ngay_sinh();
    string get_email();

    virtual void nhap() = 0;
    virtual void hien_thi() = 0;

    virtual void cap_nhat() = 0;
    virtual void chinh_sua() = 0;
};

class NhanVien : public Nguoi {
protected:
    int ma_nhan_vien;
    string phong_ban;
    double luong_co_ban;
    double luong;
    float he_so_luong;
    int kinh_nghiem;

public:
    NhanVien();
    virtual ~NhanVien();

    void set_ma_nhan_vien(int ma_nhan_vien);
    void set_phong_ban(const string& phong_ban);
    void set_luong_co_ban(double luong_co_ban);
    void set_he_so_luong(float he_so_luong);
    void set_kinh_nghiem(int kinh_nghiem);

    int get_ma_nhan_vien();
    string get_phong_ban();
    double get_luong_co_ban();
    float get_he_so_luong();
    int get_kinh_nghiem();

    // Lớp nhân viên là lớp abstract kế thừa từ lớp abstract Nguoi
    virtual void nhap() = 0;
    virtual void hien_thi() = 0;
    virtual void cap_nhat() = 0;
    virtual void chinh_sua() = 0;
    //virtual void luu() = 0;
    virtual double tinh_luong() = 0;
    virtual void doc_file(string) = 0;
};
```

Hình3.2 Lớp Nguoi và Lớp NhanVien

```
class PhuongTien {
public:
    enum TrangThai { CON TRONG, DA DAT, DANG BAO TRI };

protected:
    string ma_phuong_tien;
    string ten_phuong_tien;
    double gia_thue;
    string diem_den;
    int so_ngay_thue; // Lớp máy bay không kế thừa thuộc tính này
    TrangThai trang_thai;

public:
    PhuongTien();
    virtual ~PhuongTien();

    void set_ma_phuong_tien(string ma_phuong_tien);
    void set_ten_phuong_tien(string ten_phuong_tien);
    void set_gia_thue(double gia_thue);
    void set_diem_den(string diem_den);
    void set_so_ngay_thue(int so_ngay_thue);
    void set_trang_thai(TrangThai trang_thai);

    string get_ma_phuong_tien();
    string get_ten_phuong_tien();
    double get_gia_thue();
    string get_diem_den();
    int get_so_ngay_thue();
    TrangThai get_trang_thai();
    string trang_thai_to_string();

    virtual void nhap() = 0;
    virtual void hien_thi() = 0;
    virtual void cap_nhat() = 0;
    virtual void chinh_sua() = 0;
    virtual void dat_phuong_tien() = 0;
    virtual voidkiem_tra_trang_thai() = 0;
    virtual void cap_nhat_trang_thai() = 0;
    virtual void doc_file(string) = 0;
};

class LuuTru {
protected:
    string ma_luu_tru;
    string ten_luu_tru;
    DiaChi dia_chi;
    string so_dien_thoai;
    double gia;
    TrangThai trang_thai;

public:
    LuuTru();
    virtual ~LuuTru();

    void set_ma_luu_tru(string);
    void set_ten_luu_tru(string);
    void set_dia_chi(DiaChi);
    void set_so_dien_thoai(string);
    void set_gia(double);
    void set_trang_thai(TrangThai);

    string get_ma_luu_tru();
    string get_ten_luu_tru();
    DiaChi get_dia_chi();
    string get_so_dien_thoai();
    double get_gia();
    TrangThai get_trang_thai();

    virtual void nhap() = 0;
    virtual void hien_thi() = 0;
    virtual void cap_nhat() = 0;
    virtual void sua() = 0;

    virtual void dat_phong() = 0;
    virtual void kt_trang_thai() = 0;
    virtual void cap_nhat_trang_thai() = 0;

    virtual void doc_file(string line) = 0;
};
```

Hình 3.3 Class PhuongTien và class LuuTru

3.1.2.2 Các lớp hỗ trợ

Lớp Date: Quản lý thông tin ngày tháng năm với các phép so sánh (>, <, ==), nhập xuất bằng toán tử overloading, và chuyển đổi chuỗi.

```

class Date {
private:
    int ngay;
    int thang;
    int nam;
public:
    Date();
    Date(int, int, int);
    Date(const Date&);
    ~Date();

    friend istream& operator>>(istream&, Date&);
    friend ostream& operator<<(ostream&, const Date&);

    int get_ngay();
    int get_thang();
    int get_nam();

    void set_ngay(int);
    void set_thang(int);
    void set_nam(int);
    void set_ngay_thang_nam(Date);

    void phan_tich_chuoi_ngay(const string&);
    string chuoi_ngay_thang_nam();

    bool operator==(const Date&);
    bool operator<(const Date&);
    bool operator>=(const Date&);
};

```

Hình 3.4 Class Date

Lớp Name: Quản lý họ tên với khả năng tách và ghép họ, tên đệm, tên. Hỗ trợ nhập xuất bằng toán tử overloading.

```

class Name {
private:
    string ho;
    string ten;
    string ten_dem;
    string ho_ten; // lưu full tên
public:
    Name();
    Name(string, string, string);
    Name(const Name&);
    ~Name();

    friend istream& operator>>(istream&, Name&);
    friend ostream& operator<<(ostream&, const Name&);

    string get_ho();
    string get_ten_dem();
    string get_ten();
    string get_ho_ten();

    void set_ho(string);
    void set_ten_dem(string);
    void set_ten(string);
    void set_hoten(string full_name); // chỉ gán full name, không tách
};

```

Hình 3.5 Class Name

Lớp DiaChi: Lưu trữ thông tin địa chỉ bao gồm số nhà, tên đường, tên tỉnh. Hỗ trợ nhập xuất bằng toán tử overloading.

```

class DiaChi {
private:
    string so_nha;
    string ten_duong;
    string ten_tinh;
public:
    DiaChi();
    DiaChi(string, string, string);
    DiaChi(const DiaChi&);
    ~DiaChi();

    friend istream& operator>>(istream&, DiaChi&);
    friend ostream& operator<<(ostream&, const DiaChi&);

    string get_so_nha();
    string get_ten_duong();
    string get_ten_tinh();
    string get_dia_chi_day_du();

    void set_so_nha(string);
    void set_ten_duong(string);
    void set_ten_tinh(string);
    void set_dia_chi_day_du(string&, string&, string&);
};

```

Hình 3.6 Class DiaChi

Lớp Time: Quản lý thông tin giờ, phút, giây với validation và chuyển đổi chuỗi. Hỗ trợ nhập xuất bằng toán tử overloading.

```

};
class Time {
private:
    int gio;
    int phut;
    int giay;
public:
    Time();
    Time(int, int, int);
    Time(const Time&);
    ~Time();

    friend ostream& operator>>(ostream&, Time&);
    friend ostream& operator<<(ostream&, const Time&);

    int get_gio();
    int get_phut();
    int get_giay();

    void set_gio(int);
    void set_phut(int);
    void set_giay(int);

    void phan_tich_chuoi_thoi_gian(const string&);
    string chuoi_thoi_gian();
};

```

Hình 3.7 Class Time

Lớp DiaDanh: Quản lý thông tin điểm du lịch với mã, tên, ngày khởi hành và giá.

```

class DiaDanh {
private:
    string ma_dia_danh;
    string ten_dia_danh;
    Date ngay_khoi_hanh;
    float gia_du_lich;
public:
    DiaDanh();
    ~DiaDanh();

    void set_ma_dia_danh(string);
    void set_ten_dia_danh(string);
    void set_ngay_khoi_hanh(Date);
    void set_gia(float);

    string get_ma_dia_danh();
    string get_ten_dia_danh();
    Date get_ngay_khoi_hanh();
    float get_gia();

    void nhap();
    void chinh_sua();
    void hien_thi();
    void cap_nhat();
};

```

Hình 3.8 Class DiaDanh

Lớp ThanhToan: Quản lý thanh toán với các thuộc tính.

Union DuLieuThanhToan: Sử dụng union để tiết kiệm bộ nhớ khi lưu thông tin thanh toán:

```

union DuLieuThanhToan {
    double so_tien_mat;
    char so_the[20];
    char tai_khoan_vi[20];
};

```

Lớp DatDichVu: Kết hợp các dịch vụ thành một đơn đặt.

```

class ThanhToan {
private:
    string ma_thanh_toan;
    string ma_dat_dich_vu;
    float so_tien;
    LoaiThanhToan loai;
    DuLieuThanhToan du_lieu;
    Date ngay_thanh_toan;
    bool da_thanh_toan;

public:
    ThanhToan();
    ~ThanhToan();

    void gan_ma_thanh_toan(string);
    void gan_ma_dat_dich_vu(string);
    void gan_so_tien(float);
    void gan_loai(LoaiThanhToan);
    void gan_du_lieu_tien_mat(double);
    void gan_du_lieu_the(const char*);
    void gan_du_lieu_vi(const char*);
    void gan_ngay_thanh_toan(Date);
    void gan_trang_thai(bool);

    string lay_ma_thanh_toan();
    string lay_ma_dat_dich_vu();
    float lay_so_tien();
    LoaiThanhToan lay_loai();
    DuLieuThanhToan lay_du_lieu();
    Date lay_ngay_thanh_toan();
    bool lay_trang_thai();

    void nhap();
    void hien_thi();
    void chinh_sua();
    void cap_nhat();
};

```

Hình 3.8 Class ThanhToan

3.1.2.2 Các lớp dẫn xuất (Derived Classes)

Từ lớp NhanVien:

- NhanVienChamSoc: Thêm thuộc tính so_khach_phu_trach và thuong. Công thức tính lương: $Lương = lương_co_ban \times he_so_luong + thuong$
- HuongDanVien: Thêm thuộc tính ngon_ngu và so_tour_da_dan. Công thức tính lương: $Lương = lương_co_ban \times he_so_luong + so_tour_da_dan \times 200000$
- Từ lớp Nguoi:
- KháchHang: Thêm ma_khach_hang, so_nguoi, tong_chi_tieu
- Từ lớp PhuongTien:
- MayBay: Thêm hang_hang_khong và ngay_khoi_hanh (không sử dụng so_ngay_thue)
- XeKhach: Thêm ten_tai_xe và loai_xe (enum: XE_16_CHO, XE_45_CHO)
- XeHoi: Thêm loai_xe (enum: XE_DIEN, XE_XANG) và so_ghe (enum: XE_5_CHO, XE_7_CHO)

Từ lớp LuuTru:

- KháchSan: Thêm sao và tong_so_phong
- NhaNghỉ: Thêm gio_hoat_dong, so_xe và tinh_trang_xe

3.1.2.3 Cấu trúc Node và Danh sách

cấu trúc Node (ví dụ với KháchHang):

```
struct NodeKhachHang {  
    KháchHang* data;  
    NodeKhachHang* next;  
    NodeKhachHang(KháchHang* kh) : data(kh), next(nullptr) {}  
};
```

Các lớp quản lý danh sách:

- DanhSachKhachHang
- DanhSachNhanVien
- DanhSachPhuongTien
- DanhSachDiaDanh
- DanhSachLuuTru
- DanhSachThanhToan
- DanhSachDichVu
- Danhsachkhachhang

```
class DanhSachKhachHang {  
private:  
    NodeKhachHang* head;  
public:  
    DanhSachKhachHang();  
    ~DanhSachKhachHang();  
  
    void them(KhachHang*);  
    void hien_thi();  
    void xoa(const string&);  
    KháchHang* tim_kiem(const string&);  
    void chinh_sua(const string&);  
    void cap_nhat(const string&);  
    void merge_sort_chi_tieu();  
    NodeKhachHang* chia_doi(NodeKhachHang* head);  
    NodeKhachHang* tron(NodeKhachHang* a, NodeKhachHang* b);  
    NodeKhachHang* merge_sort(NodeKhachHang* head);  
    void bubble_sort_ten();  
    void doc_file();  
    void giai_phong();  
    void hien_thi_menu_khach_hang();  
};
```

Hình 3.9 Class DanhSachKhachHang

```

class DanhSachNhanVien {
private:
    NodeNhanVien* head;
public:
    DanhSachNhanVien();
    ~DanhSachNhanVien();

    void them(NhanVien*);
    void hien_thi();
    void xoa(int ma_nv);
    NhanVien* tim_kiem(int);
    void chinh_sua(int);
    void insertion_sort_luong();
    void bubble_sort_ten();
    void doc_file();
    void giai_phong();
    void hien_thi_menu_nhan_vien();
};

```

Hình 3.10 Class DanhSachNhanVien

```

class DanhSachPhuongTien {
private:
    NodePhuongTien* head;
public:
    DanhSachPhuongTien();
    ~DanhSachPhuongTien();

    void them(PhuongTien* phuong_tien);
    void xoa(string ma_phuong_tien);
    void hien_thi();
    void chinh_sua(string ma_phuong_tien);
    PhuongTien* tim_kiem(string ma_phuong_tien);
    void insertion_sort_theo_gia();
    void bubble_sort_theo_trang_thai();
    void doc_file();
    void hien_thi_menu_phuong_tien();
};

```

Hình 3.11 Class DanhSachPhuongTien

```

class DanhSachDiaDanh {
private:
    NodeDiaDanh* head;
public:
    DanhSachDiaDanh();
    ~DanhSachDiaDanh();
    void them();
    void xoa(string);
    void cap_nhat(string);
    void chinh_sua(string);
    void hien_thi();
    DiaDanh* tim_kiem(string);
    void insertion_sort_gia();
    void selection_sort_dia_danh();
    void bubble_sort_ngay();
    void doc_file();
    void hien_thi_menu_dia_danh();
};

```

Hình 3.12 Class DanhSachNhanVien

```

class DanhSachThanhToan {
private:
    NodeThanhToan* head;
public:
    DanhSachThanhToan();
    ~DanhSachThanhToan();

    void them();
    void xoa(string);
    void cap_nhat(string);
    void chinh_sua(string);
    void hien_thi();
    ThanhToan* tim_kiem(string);
    void insertion_sort_ngay();
    void selection_sort_so_tien();
    void bubble_sort_loai();
    void doc_file();
    void hien_thi_menu_thanh_toan();
};

```

Hình 3.13 Class DanhSachThanhToan

```

class DanhSachDichVu {
private:
    NodeDichVu* head;
public:
    DanhSachDichVu();
    ~DanhSachDichVu();

    void them();
    void hien_thi();
    void xoa(string);
    void chinh_sua(string);
    void cap_nhat(string);
    DatDichVu* tim_kiem(string);
    void hien_thi_menu_dich_vu();

    void doc_file(DanhSachKhachHang& dsKH,
        DanhSachNhanVien& dsNV,
        DanhSachPhuongTien& dsPT,
        DanhSachDiaDanh& dsDD,
        DanhSachLuuTru& dsLT,
        DanhSachThanhToan& dsTT);
};

```

Hình 3.14 Class DanhSachDichVu

3.1.3 Các thao tác chính

3.1.3.1 Quản lý Khách hàng

Chức năng:

- Thêm khách hàng mới với thông tin: mã, họ tên, số điện thoại, email, ngày sinh, số người, tổng chi tiêu
- Xóa khách hàng theo mã
- Tìm kiếm khách hàng theo mã
- Cập nhật số điện thoại khách hàng
- Chỉnh sửa toàn bộ thông tin khách hàng
- Sắp xếp theo tên (Bubble Sort) hoặc tổng chi tiêu (Merge Sort)
- Đọc dữ liệu từ file ds_khach_hang.txt

Thuật toán sắp xếp:

- Bubble Sort theo tên: So sánh tên (thuộc tính ten trong lớp Name) và hoán đổi nếu không đúng thứ tự
- Merge Sort theo chi tiêu: Sử dụng kỹ thuật chia đôi danh sách liên kết với hai hàm phụ chia_doi() và tron()

3.1.3.2 Quản lý Nhân viên

Hai loại nhân viên:

Nhân viên chăm sóc: Có số khách phụ trách và thưởng

Hướng dẫn viên: Có ngôn ngữ hướng dẫn và số tour đã dẫn

Chức năng:

Thêm nhân viên (phân biệt loại khi nhập)

- Xóa nhân viên theo mã
- Tìm kiếm nhân viên theo mã
- Chỉnh sửa thông tin nhân viên
- Cập nhật thưởng (cho nhân viên chăm sóc) hoặc số tour (cho hướng dẫn viên)
- Tính lương tự động theo công thức riêng cho từng loại
- Sắp xếp theo tên (Bubble Sort) hoặc lương (Insertion Sort)
- Đọc dữ liệu từ file ds_nhan_vien.txt (có dòng phân loại "ChamSoc" hoặc "HuongDan")

Công thức tính lương:

- Nhân viên chăm sóc: $Lương = lương_co_ban \times he_so_luong + thưởng$
- Hướng dẫn viên: $Lương = lương_co_ban \times he_so_luong + so_tour_da_dan \times 200000$

3.1.3.3 Quản lý Phương tiện

Ba loại phương tiện:

- Máy bay: Mã bắt đầu bằng "MB", có hãng hàng không và ngày khởi hành
- Xe khách: Mã bắt đầu bằng "XK", có tên tài xế, loại xe (16/45 chỗ)
- Xe hơi: Mã bắt đầu bằng "XH", có loại xe (điện/xăng), số ghế (5/7 chỗ)

Chức năng:

- Thêm phương tiện (chọn loại trước khi nhập)
- Xóa phương tiện theo mã
- Tìm kiếm phương tiện theo mã
- Chỉnh sửa thông tin phương tiện
- Cập nhật điểm đến và thông tin riêng của từng loại
- Đặt phương tiện (chuyển trạng thái sang DA_DAT)
- Kiểm tra và cập nhật trạng thái (CON_TRONG, DA_DAT, DANG_BAO_TRI)
- Sắp xếp theo giá thuê (Insertion Sort) hoặc trạng thái (Bubble Sort)
- Đọc dữ liệu từ file ds_phuong_tien.txt (phân biệt loại theo mã đầu dòng)

Enum TrangThai: Được định nghĩa chung cho nhiều lớp

cpp

```
enum TrangThai { CON_TRONG, DA_DAT, DANG_BAO_TRI };
```

3.1.3.4 Quản lý Địa danh

Chức năng:

- Thêm địa danh du lịch mới
- Xóa địa danh theo mã
- Tìm kiếm địa danh theo mã
- Chỉnh sửa/cập nhật thông tin địa danh
- Quản lý thông tin: mã, tên, ngày khởi hành, giá mỗi người
- Sắp xếp theo giá (Insertion Sort), tên (Selection Sort), hoặc ngày khởi hành (Bubble Sort)
- Đọc dữ liệu từ file ds_dia_danh.txt
- Ba thuật toán sắp xếp:
- Insertion Sort theo giá: Tạo danh sách sorted mới và chèn từng node vào đúng vị trí
- Selection Sort theo tên: Tìm node có tên nhỏ nhất và hoán đổi
- Bubble Sort theo ngày: So sánh ngày khởi hành sử dụng toán tử > đã overload trong lớp Date

3.1.3.5 Quản lý Lưu trú

Hai loại lưu trú:

- Khách sạn: Mã bắt đầu bằng "KS", có số sao và tổng số phòng
- Nhà nghỉ: Mã bắt đầu bằng "NN", có giờ hoạt động, số xe và tình trạng xe
- Chức năng:
 - Thêm lưu trú (chọn loại khách sạn hoặc nhà nghỉ)
 - Xóa lưu trú theo mã
 - Tìm kiếm lưu trú theo mã
 - Chính sửa thông tin lưu trú
 - Cập nhật thông tin cụ thể (tên, số điện thoại, giá, số sao/giờ hoạt động)
 - Đặt phòng (chuyển trạng thái)
 - Đặt xe (chỉ cho nhà nghỉ)
 - Kiểm tra trạng thái theo mã
 - Cập nhật trạng thái
 - Sắp xếp theo giá (Insertion Sort) hoặc trạng thái (Bubble Sort)
 - Đọc dữ liệu từ file ds_luu_tru.txt (phân biệt loại theo mã KS/NN)

Hàm hỗ trợ:

TrangThai chuyen_str_thanh_trang_thai(const string& str);

3.1.3.6 Quản lý Thanh toán

Đặc điểm:

- Sử dụng Union để lưu thông tin thanh toán tiết kiệm bộ nhớ
- Ba loại thanh toán: TIEN_MAT, THE, VI_DIEN_TU (enum)

Chức năng:

- Thêm giao dịch thanh toán mới
- Xóa giao dịch theo mã
- Tìm kiếm giao dịch theo mã
- Chính sửa số tiền và trạng thái thanh toán
- Cập nhật trạng thái thanh toán (đã thanh toán/chưa thanh toán)

- Sắp xếp theo ngày (Insertion Sort), số tiền (Selection Sort), hoặc loại thanh toán (Bubble Sort)
- Đọc dữ liệu từ file ds_thanh_toan.txt

Union DuLieuThanhToan:

- Chỉ một trong ba trường được sử dụng tại một thời điểm
- Tiết kiệm bộ nhớ so với việc sử dụng ba biến riêng biệt

3.1.3.7. Quản lý Dịch vụ

Chức năng:

- Tạo đơn đặt dịch vụ liên kết nhiều thành phần: Khách hàng, Nhân viên hướng dẫn, Phương tiện, Địa danh, Lưu trú
- Nhập mã để liên kết với các đối tượng có sẵn trong các danh sách khác
- Tự động tính tổng tiền dựa trên giá của các dịch vụ đã chọn:
- Xóa dịch vụ theo mã
- Tìm kiếm dịch vụ theo mã
- Chỉnh sửa dịch vụ (nhập lại toàn bộ)
- Cập nhật ngày đặt dịch vụ
- Hiển thị chi tiết đầy đủ các thành phần của dịch vụ
- Đọc dữ liệu từ file ds_dich_vu.txt (liên kết với các danh sách khác thông qua mã)

Đặc điểm quan trọng:

- Sử dụng con trỏ để tham chiếu đến các đối tượng trong danh sách khác
- Không sao chép đối tượng, chỉ lưu địa chỉ
- Khi đọc file, cần truyền tham chiếu tất cả các danh sách liên quan

3.2.4 Quản lý bộ nhớ

Nguyên tắc cấp phát và giải phóng:

- Cấp phát động: Tất cả đối tượng được tạo bằng toán tử new để cấp phát bộ nhớ động
- Giải phóng bộ nhớ: Mỗi danh sách có phương thức giai_phong() hoặc destructor để giải phóng toàn bộ node và đối tượng

- Tránh memory leak: Khi xóa một đối tượng, chương trình giải phóng cả dữ liệu đối tượng (delete data) và node (delete node)
- Con trỏ: Sử dụng con trỏ để liên kết các đối tượng trong danh sách và giữa các module

3.2.5 Đọc dữ liệu từ file

3.2.5.1 Danh sách các file

1. ds_khach_hang.txt - Dữ liệu khách hàng
2. ds_nhan_vien.txt - Dữ liệu nhân viên (có dòng phân loại)
3. ds_phuong_tien.txt - Dữ liệu phương tiện (phân biệt theo mã)
4. ds_dia_danh.txt - Dữ liệu địa danh
5. ds_luu_tru.txt - Dữ liệu lưu trú (phân biệt theo mã KS/NN)
6. ds_thanh_toan.txt - Dữ liệu thanh toán
7. ds_dich_vu.txt - Dữ liệu dịch vụ (liên kết với các danh sách khác)

ds_dia_danh	Text Document	1 KB	Không	1 KB	42%	08/11/2025 4:44 CH
ds_dich_vu	Text Document	1 KB	Không	1 KB	66%	09/11/2025 10:22 CH
ds_khach_hang	Text Document	1 KB	Không	1 KB	47%	06/11/2025 11:30 CH
ds_luu_tru	Text Document	1 KB	Không	1 KB	47%	09/11/2025 10:05 SA
ds_nhan_vien	Text Document	1 KB	Không	1 KB	49%	06/11/2025 10:45 CH
ds_phuong_tien	Text Document	1 KB	Không	1 KB	61%	08/11/2025 10:16 SA
ds_thanh_toan	Text Document	1 KB	Không	1 KB	47%	08/11/2025 5:28 CH

Hình 3.15 Danh sách file TXT

Quy trình đọc file tại khởi động

Phương thức CongTy::doc_file() được gọi trong hàm main() để tự động nạp dữ liệu:

```
void CongTy::doc_file() {
    cout << "\n=== DANG DOC DU LIEU TU CAC FILE ===\n";

    ds_khach_hang.doc_file();
    ds_nhan_vien.doc_file();
    ds_phuong_tien.doc_file();
    ds_dia_danh.doc_file();
    ds_luu_tru.doc_file();
    ds_thanh_toan.doc_file();
    ds_dich_vu.doc_file(ds_khach_hang, ds_nhan_vien, ds_phuong_tien, ds_dia_danh, ds_luu_tru, ds_thanh_toan);

    cout << "\n=== DA HOAN TAT QUA TRINH DOC FILE ===\n";
}
```

Hình 3.16 Quy trình đọc file khởi động

3.2.6 Các thuật toán sắp xếp được sử dụng

Hệ thống triển khai ba thuật toán sắp xếp cơ bản trên danh sách liên kết:

3.2.6.1 Bubble Sort

Sử dụng cho:

- Sắp xếp khách hàng theo tên
- Sắp xếp nhân viên theo tên
- Sắp xếp phương tiện theo trạng thái
- Sắp xếp địa danh theo ngày khởi hành
- Sắp xếp lưu trú theo trạng thái
- Sắp xếp thanh toán theo loại

3.2.6.2 Insertion Sort

Sử dụng cho:

- Sắp xếp nhân viên theo lương
- Sắp xếp phương tiện theo giá thuê
- Sắp xếp địa danh theo giá
- Sắp xếp lưu trú theo giá
- Sắp xếp thanh toán theo ngày

3.2.6.3 Selection Sort

Sử dụng cho:

- Sắp xếp địa danh theo tên
- Sắp xếp thanh toán theo số tiền

3.2.6.4 Merge Sort

Sử dụng cho:

- Sắp xếp khách hàng theo tổng chi tiêu

3.2.7 Đặc điểm nổi bật của hệ thống

3.2.7.1 Tính module hóa cao

- Mỗi lớp có trách nhiệm riêng biệt
- Các danh sách độc lập, dễ bảo trì và mở rộng
- Lớp CongTy tổng hợp và điều phối các module

3.2.7.2 Tính linh hoạt

- Hỗ trợ nhiều loại đối tượng con (2-3 loại) cho mỗi danh sách chính
- Cho phép thêm loại mới dễ dàng thông qua kế thừa

3.2.7.3 Giao diện thân thiện

- Menu phân cấp rõ ràng
- Hiển thị dữ liệu dạng bảng chuyên nghiệp
- Thông báo lỗi và xác nhận thao tác

3.2.7.4 Quản lý bộ nhớ an toàn

- Giải phóng bộ nhớ tự động qua destructor
- Phương thức `giai_phong()` rõ ràng
- Không để rò rỉ bộ nhớ (memory leak)

3.2.7.5 Tích hợp dữ liệu

- Dịch vụ liên kết nhiều đối tượng từ các danh sách khác
- Tính toán tự động tổng tiền
- Đồng bộ dữ liệu giữa các module

3.3 Các chức năng chính trong hệ thống

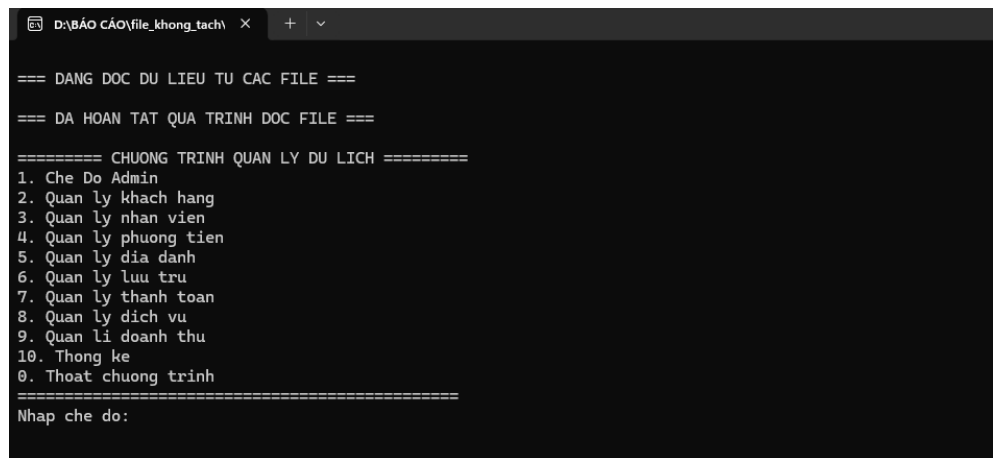
3.3.1 Giao diện người dùng

Hệ thống quản lý du lịch sử dụng giao diện dòng lệnh với thiết kế menu phân cấp trực quan và dễ sử dụng. Giao diện được tổ chức theo mô hình đa tầng, bao gồm menu chính với 8 module quản lý độc lập và 1 menu Admin tổng hợp. Mỗi module có menu riêng với 8-10 chức năng chuẩn hóa, giúp người dùng dễ dàng thực hiện các thao tác như thêm, xóa, sửa, tìm kiếm và sắp xếp dữ liệu.

Điểm nổi bật của giao diện là việc hiển thị dữ liệu dưới dạng bảng có định dạng chuyên nghiệp với đường viền, tiêu đề cột rõ ràng và căn chỉnh chuẩn (text căn trái, số căn phải). Hệ thống cung cấp các thông báo rõ ràng về trạng thái thao tác, xác nhận hành động quan trọng và hiển thị lỗi khi có vấn đề xảy ra. Menu điều hướng cho phép người

dùng dễ dàng di chuyển giữa các module và quay lại menu chính bất kỳ lúc nào bằng cách chọn "0. Thoát". Cơ chế nhập liệu được thiết kế với validation cơ bản, đảm bảo dữ liệu nhập vào hợp lệ trước khi lưu vào hệ thống.

Giao diện console tuy đơn giản nhưng đầy đủ chức năng, phù hợp với mục đích quản lý và dễ dàng triển khai trên mọi nền tảng hỗ trợ C++.



```
D:\BẢO CÁO\file_khong_tach\ >
=== DANG DOC DU LIEU TU CAC FILE ===
=== DA HOAN TAT QUA TRINH DOC FILE ===

===== CHUONG TRINH QUAN LY DU LICH =====
1. Che Do Admin
2. Quan ly khách hàng
3. Quan ly nhân viên
4. Quan ly phương tiện
5. Quan ly địa danh
6. Quan ly lưu trú
7. Quan ly thanh toán
8. Quan ly dịch vụ
9. Quan li doanh thu
10. Thông ke
0. Thoát chương trình
=====
Nhap che do:
```

Hình 3.17 Menu hệ thống

3.3.2 CHẾ ĐỘ ADMIN

1.1 Nhập Dữ Liệu

- Nhập khách hàng mới
- Nhập nhân viên mới (chọn loại: Chăm sóc hoặc Hướng dẫn)
- Nhập phương tiện (chọn: Máy bay, Xe khách, Xe hơi)
- Nhập địa danh du lịch
- Nhập lưu trú (chọn: Khách sạn hoặc Nhà nghỉ)
- Nhập thanh toán
- Nhập dịch vụ

1.2 Hiện Thị Dữ Liệu

- Hiện thị danh sách khách hàng
- Hiện thị danh sách nhân viên
- Hiện thị danh sách phương tiện
- Hiện thị danh sách địa danh

- Hiển thị danh sách lưu trú
- Hiển thị danh sách thanh toán
- Hiển thị danh sách đặt dịch vụ

1.3 Tìm Kiếm Dữ Liệu

- Tìm khách hàng theo mã
- Tìm nhân viên theo mã
- Tìm phương tiện theo mã
- Tìm địa danh theo mã
- Tìm lưu trú theo mã
- Tìm thanh toán theo mã

1.4 Sắp Xếp Dữ Liệu

- Sắp xếp khách hàng theo tên
- Sắp xếp nhân viên theo tên
- Sắp xếp phương tiện theo trạng thái
- Sắp xếp địa danh theo giá
- Sắp xếp lưu trú theo trạng thái
- Sắp xếp thanh toán theo loại

1.5 Chỉnh Sửa Dữ Liệu

- Chỉnh sửa thông tin khách hàng
- Chỉnh sửa thông tin nhân viên
- Chỉnh sửa thông tin phương tiện
- Chỉnh sửa thông tin địa danh
- Chỉnh sửa thông tin lưu trú
- Chỉnh sửa thông tin thanh toán

3.3.3 QUẢN LÝ KHÁCH HÀNG

1. Thêm khách hàng mới
2. Hiện thị danh sách khách hàng
3. Tìm khách hàng theo mã

4. Xóa khách hàng theo mã
5. Chỉnh sửa thông tin khách hàng
6. Sắp xếp theo tổng chi tiêu (Merge Sort)
7. Sắp xếp theo tên (Bubble Sort)
8. Ghi danh sách ra file (đang bảo trì)
0. Thoát

Thông tin khách hàng bao gồm:

- Mã khách hàng
- Họ tên (tách thành họ, tên đệm, tên)
- Số điện thoại
- Email
- Ngày sinh
- Số người trong đoàn
- Tổng chi tiêu

3.3.4 QUẢN LÝ NHÂN VIÊN

1. Thêm nhân viên mới
 - Chọn loại: Chăm sóc / Hướng dẫn
2. Hiện thị danh sách
3. Xóa nhân viên theo mã
4. Tìm nhân viên theo mã
5. Chỉnh sửa nhân viên
6. Sắp xếp theo tên (Bubble Sort)
7. Sắp xếp theo lương (Insertion Sort)
8. Ghi danh sách ra file (đang bảo trì)
0. Thoát

Thông tin nhân viên:

- Mã nhân viên

- Họ tên
- Phòng ban
- Lương cơ bản
- Hệ số lương
- Kinh nghiệm

Riêng Nhân viên chăm sóc:

- Số khách phụ trách
- Tiền thưởng

Riêng Hướng dẫn viên:

- Ngôn ngữ hướng dẫn
- Số tour đã dẫn

3.3.5 QUẢN LÝ PHƯƠNG TIỆN

1. Thêm phương tiện mới

- Máy bay (hãng hàng không, ngày khởi hành)
- Xe khách (16 chỗ / 45 chỗ, tài xế)
- Xe hơi (xăng/điện, 5 chỗ/7 chỗ)

2. Hiện thị danh sách

3. Xóa phương tiện theo mã

4. Tìm phương tiện theo mã

5. Chỉnh sửa phương tiện

6. Sắp xếp theo giá thuê (Insertion Sort)

7. Sắp xếp theo trạng thái (Bubble Sort)

8. Ghi danh sách ra file (đang bảo trì)

0. Thoát

Trạng thái phương tiện:

- Còn trống
- Đã đặt
- Đang bảo trì

3.3.6 QUẢN LÝ ĐỊA DANH

1. Thêm địa danh mới
2. Hiện thị danh sách
3. Xóa địa danh theo mã
4. Tìm địa danh theo mã
5. Chỉnh sửa địa danh
6. Sắp xếp theo giá (Insertion Sort)
7. Sắp xếp theo tên địa danh (Selection Sort)
8. Sắp xếp theo ngày khởi hành (Bubble Sort)
9. Ghi danh sách ra file (đang bảo trì)
0. Thoát

Thông tin địa danh:

- Mã địa danh
- Tên địa danh
- Ngày khởi hành
- Giá du lịch/người

3.3.7 QUẢN LÝ LƯU TRÚ

1. Thêm lưu trú mới
 - Khách sạn (số sao, tổng số phòng)
 - Nhà nghỉ (giờ hoạt động, số xe)
2. Hiện thị danh sách
3. Xóa lưu trú theo mã
4. Tìm lưu trú theo mã
5. Chỉnh sửa lưu trú
6. Sắp xếp theo giá (Insertion Sort)
7. Sắp xếp theo trạng thái (Bubble Sort)
8. Kiểm tra trạng thái theo mã
9. Ghi danh sách ra file (đang bảo trì)

0. Thoat

Thông tin lưu trữ:

- Mã lưu trữ
- Tên lưu trữ
- Địa chỉ (số nhà, tên đường, tên tỉnh)
- Số điện thoại
- Giá
- Trạng thái (Còn trống/Đã đặt/Đang bảo trì)

3.3.8 QUẢN LÝ THANH TOÁN

1. Thêm thanh toán mới
2. Hiện thị danh sách thanh toán
3. Xóa thanh toán theo mã
4. Tìm thanh toán theo mã
5. Chỉnh sửa thanh toán
6. Cập nhật trạng thái
7. Sắp xếp theo ngày thanh toán (Insertion Sort)
8. Sắp xếp theo số tiền (Selection Sort)
9. Sắp xếp theo loại thanh toán (Bubble Sort)
10. Ghi danh sách ra file (đang bảo trì)

0. Thoat

Loại thanh toán:

- Tiền mặt
- Thẻ (lưu số thẻ)
- Ví điện tử (lưu tài khoản)

Sử dụng Union để tiết kiệm bộ nhớ

3.3.9 QUẢN LÝ DỊCH VỤ

1. Thêm dịch vụ mới
2. Hiện thị danh sách dịch vụ

3. Xóa dịch vụ theo mã
4. Tìm dịch vụ theo mã
5. Chỉnh sửa dịch vụ
6. Cập nhật ngay dịch vụ
7. Tính tổng tiền dịch vụ
8. Ghi danh sách ra file (đang bảo trì)
0. Thoát

Thông tin dịch vụ:

- Mã dịch vụ
- Mã khách hàng (liên kết)
- Mã nhân viên (liên kết)
- Mã phương tiện (liên kết)
- Mã địa danh (liên kết)
- Mã lưu trú (liên kết)
- Tổng tiền tự động = Giá phương tiện + Giá địa danh + Giá lưu trú

CHƯƠNG 4. KẾT QUẢ VÀ KIẾN NGHỊ

4.1 Kết quả đạt được

Sau quá trình nghiên cứu và triển khai, hệ thống quản lý du lịch đã được hoàn thiện với đầy đủ các chức năng cơ bản và nâng cao, bao gồm quản lý khách hàng, nhân viên, phương tiện, địa danh, lưu trú, thanh toán và dịch vụ. Hệ thống hỗ trợ đầy đủ các thao tác CRUD (Create, Read, Update, Delete) cho 7 module chính, tìm kiếm theo mã định danh, sắp xếp dữ liệu theo nhiều tiêu chí khác nhau, và lưu trữ dữ liệu bằng file văn bản.

Chương trình hoạt động ổn định với gần 5,000 dòng code tách ra thành nhiều file header và cpp, xử lý đúng dữ liệu phức tạp, hiển thị thông tin dạng bảng rõ ràng và hỗ trợ đọc dữ liệu tự động từ file khi khởi động. Giao diện console được thiết kế với menu phân cấp trực quan, giúp người dùng dễ dàng điều hướng giữa các chức năng. Hệ thống áp dụng 4 thuật toán sắp xếp khác nhau (Insertion Sort, Bubble Sort, Selection Sort, Merge Sort) phù hợp với từng loại dữ liệu và yêu cầu về hiệu năng.

Việc áp dụng danh sách liên kết đơn cho phép xử lý linh hoạt dữ liệu không giới hạn số lượng, tránh lãng phí bộ nhớ và giúp người thực hiện củng cố kiến thức về cấp phát động, thao tác con trỏ, cấu trúc dữ liệu và quản lý bộ nhớ. Đặc biệt, hệ thống đã triển khai thành công các nguyên lý lập trình hướng đối tượng (OOP) với 4 tính chất đầy đủ: đóng gói (encapsulation), kế thừa (inheritance), đa hình (polymorphism) và trừu tượng (abstraction). Các lớp abstract như *Nguoi*, *NhanVien*, *PhuongTien*, *LuuTru* được kế thừa và triển khai bởi các lớp con, tạo nên cấu trúc phân cấp rõ ràng và dễ mở rộng. Hệ thống cũng sử dụng hiệu quả các kiểu dữ liệu nâng cao như union để tiết kiệm bộ nhớ trong lớp *ThanhToan* (tiết kiệm từ 48 bytes xuống 20 bytes), và enum để tăng tính type-safe cho các trạng thái và phân loại. Việc overload toán tử cho các lớp *Date*, *Time*, *Name*, *DiaChi* giúp code ngắn gọn và dễ đọc hơn. Ngoài ra, chương trình có cơ chế validation dữ liệu đầu vào cho các trường quan trọng như ngày tháng, giờ phút, đảm bảo tính toàn vẹn dữ liệu.

Thông qua đồ án này, người thực hiện đã nâng cao đáng kể khả năng lập trình bằng ngôn ngữ C++, rèn luyện kỹ năng thiết kế hệ thống phức tạp với nhiều module liên kết, tổ chức code theo hướng đối tượng, và tư duy giải quyết vấn đề thực tế trong lĩnh vực quản lý du lịch. Đặc biệt, việc triển khai và so sánh 4 thuật toán sắp xếp khác nhau giúp củng cố kiến thức về độ phức tạp thuật toán và lựa chọn giải pháp tối ưu.

4.2 Kiến nghị

Mặc dù hệ thống đã hoàn thành các chức năng cơ bản và hoạt động ổn định, vẫn còn một số mặt có thể cải tiến trong tương lai nhằm nâng cao trải nghiệm người dùng và khả năng mở rộng ứng dụng.

4.2.1. Về xử lý lỗi và kiểm tra hợp lệ

Hệ thống cần bổ sung xử lý ngoại lệ (exception handling) để tránh crash khi gặp lỗi. Hiện tại, nếu file dữ liệu có format sai, chương trình có thể bị lỗi mà không có thông báo rõ ràng. Nên sử dụng try-catch để bắt các lỗi như file không đọc được, format sai, hoặc parse dữ liệu thất bại. Ngoài ra, cần cải thiện validation đầu vào toàn diện hơn: kiểm tra email hợp lệ (phải có @), số điện thoại đúng format (10-11 số), mã không được trùng lặp, giá trị số không âm, và ngày tháng hợp lệ (không cho phép 31/2/2024). Đặc biệt, khi người dùng nhập sai kiểu dữ liệu (ví dụ nhập chữ vào trường số), chương trình nên báo lỗi và yêu cầu nhập lại thay vì bị treo hoặc hành vi không xác định.

4.2.2. Về lưu trữ và quản lý dữ liệu

Về lưu trữ dữ liệu, hệ thống cần triển khai chức năng ghi file để dữ liệu không bị mất khi thoát chương trình. Hiện tại menu có option "Ghi danh sách ra file (đang bao trì)" nhưng chưa được implement. Nên thêm tính năng tự động lưu (auto-save) sau mỗi thao tác thêm/sửa/xóa, hoặc hỏi người dùng trước khi thoát có muốn lưu hay không. Ngoài ra, có thể nâng cấp lên database như SQLite (nhẹ, không cần server) hoặc MySQL/PostgreSQL (cho hệ thống lớn) để đảm bảo an toàn, hiệu quả truy xuất cao hơn, hỗ trợ transaction (rollback khi lỗi), và dễ backup/restore. Đồng thời, nên thêm chức

năng backup tự động theo định kỳ (hàng ngày/tuần) và cho phép restore từ backup khi cần.

4.2.3. Về hiệu năng và tối ưu

Chương trình cần tối ưu thao tác thêm vào cuối danh sách từ $O(n)$ xuống $O(1)$ bằng cách sử dụng tail pointer. Hiện tại, mỗi lần thêm phải duyệt từ đầu đến cuối, rất chậm khi danh sách lớn. Nên thêm biến `NodeType* tail` vào mỗi class `DanhSach*` và cập nhật khi thêm/xóa.

4.2.4. Về giao diện người dùng

Cần cải thiện giao diện console bằng cách thêm màu sắc (sử dụng thư viện `windows.h` cho Windows hoặc ANSI escape codes cho Linux), làm rõ hướng dẫn nhập liệu, hiển thị progress bar khi xử lý dữ liệu lớn, và thêm xác nhận trước khi xóa (Y/N). Ngoài ra, có thể phát triển giao diện đồ họa (GUI) bằng Qt, wxWidgets, hoặc GTK+ để tăng tính trực quan và chuyên nghiệp. GUI sẽ dễ sử dụng hơn cho người không quen command line, có thể hiển thị dữ liệu dạng table với scroll, search realtime, và xuất báo cáo đẹp hơn.

4.2.5. Về chức năng bổ sung

Cuối cùng, hệ thống có thể được phát triển thêm các chức năng nâng cao như: Module Doanh thu (báo cáo doanh thu theo ngày/tháng/năm, so sánh với kỳ trước, biểu đồ doanh thu); Module Thống kê (top 10 khách hàng VIP, nhân viên xuất sắc, tour hot nhất, tỷ lệ lấp đầy phương tiện/lưu trú, phân tích xu hướng); Lọc và tìm kiếm nâng cao (tìm khách hàng theo khoảng chi tiêu, lọc tour theo ngày/giá/địa điểm, tìm phương tiện available trong khoảng thời gian); Quản lý quyền truy cập (Admin full quyền, Manager xem/sửa, Staff chỉ xem); Tích hợp email/SMS (gửi xác nhận booking, nhắc lịch tour, khuyến mãi); Xuất báo cáo PDF/Excel (invoice, booking confirmation, báo cáo tài chính); Tích hợp thanh toán online (VNPay, Momo, ZaloPay).

TÀI LIỆU THAM KHẢO

- [1]. <https://topdev.vn/blog/oop-la-gi/#lap-trinh-huong-doi-tuong-oop-la-gi>, “Khái niệm về lập trình hướng đối tượng OOP, truy cập ngày 15 tháng 11 năm 2025”
- [2]. <https://byvn.net/s4V5>, “Lập trình C++ cơ bản, truy cập ngày 15 tháng 11 năm 2025”
- [3]. <https://blog.28tech.com.vn/>, “Một số lý thuyết cơ bản của lập trình C/ C++, truy cập ngày 15 tháng 11 năm 2025”
- [4]. <https://techacademy.edu.vn/>, “Tham khảo một số khái niệm và lý thuyết trong Lập trình hướng đối tượng, truy cập ngày 15 tháng 11 năm 2025”



QR GITHUB