

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A blue and yellow logo

Description automatically generated---------------o0o---------------**

**Bài tập lớn môn học**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**Giảng viên hướng dẫn: Nguyễn Việt Hưng**

**Nhóm thực hiện: Nhóm 22 - Lớp CNTT 4 - K64**

Lại Ngọc Mai - 231230833

Phạm Thị Thu Hằng - 23123

Nguyễn Thị Phương Thảo - 23123

**Hà Nội tháng 11 năm 2024**

**Mục lục**

[Lời nói đầu 4](#_Toc183118932)

[PHẦN A 5](#_Toc183118933)

[I. ĐỊNH NGHĨA BÀI TOÁN 5](#_Toc183118934)

[1. Đề bài 5](#_Toc183118935)

[2. Phân tích bài toán 5](#_Toc183118936)

[II. CẤU TRÚC DỮ LIỆU VECTOR 5](#_Toc183118937)

[III. PHÂN TÍCH CHƯƠNG TRÌNH 7](#_Toc183118938)

[1. Cài đặt cấu trúc dữ liệu trừu tượng vector 7](#_Toc183118939)

[IV. KẾT QUẢ THỰC NGHIỆM 12](#_Toc183118940)

[1. Dữ liệu 12](#_Toc183118941)

[2. Kết quả 12](#_Toc183118942)

[PHẦN B 14](#_Toc183118943)

[I. ĐỀ TÀI 14](#_Toc183118944)

[II. PHÂN TÍCH CHƯƠNG TRÌNH 14](#_Toc183118945)

[1. Lớp PhongTro 14](#_Toc183118946)

[2. Lớp DanhSachPhongTro 17](#_Toc183118947)

[3. Lớp App 19](#_Toc183118948)

[III. KẾT QUẢ THỰC NGHIỆM 20](#_Toc183118949)

[1. Thêm một phòng trọ 20](#_Toc183118950)

[2. Xóa một phòng trọ 21](#_Toc183118951)

[3. In ra danh sách phòng trọ 22](#_Toc183118952)

[4. Tìm kiếm phòng trọ theo tên người thuê 23](#_Toc183118953)

[5. Tìm phòng trọ có giá thuê cao nhất 23](#_Toc183118954)

[6. Tìm phòng trọ có giá thuê thấp nhất 24](#_Toc183118955)

[7. Tìm và tính tiền điện cho một phòng trọ theo tên người thuê 24](#_Toc183118956)

[8. Sắp xếp danh sách phòng trọ theo giá tăng dần 25](#_Toc183118957)

[V. BÀI LÀM 26](#_Toc183118958)

[1. vector.cpp 26](#_Toc183118959)

[2. b22.cpp 42](#_Toc183118960)

[3. input22.txt 47](#_Toc183118961)

[4. quanlinhatro.cpp 48](#_Toc183118962)

[PHẦN C 59](#_Toc183118963)

# Lời nói đầu

Để đáp ứng yêu cầu của môn học cũng như mong muốn mở rộng kiến thức chuyên sâu về môn Cấu trúc Dữ liệu và Giải thuật, nhóm em đã lựa chọn đề tài với tiêu chí vừa mang tính thực tiễn, vừa tận dụng tối đa kiến thức đã học. Qua đó, chúng em hy vọng rằng dự án không chỉ là một bài tập học thuật mà còn là cơ hội để áp dụng những cấu trúc dữ liệu và giải thuật cơ bản vào các bài toán thực tiễn, giúp tối ưu hóa quy trình xử lý dữ liệu và cải thiện hiệu suất chương trình.

Dự án này nhằm giải quyết một vấn đề cụ thể bằng cách ứng dụng các giải thuật và cấu trúc dữ liệu đã học, đồng thời tìm hiểu và học hỏi thêm những kỹ thuật mới, đặc biệt là trong việc phân tích và tối ưu các giải thuật. Việc thực hiện dự án giúp chúng em có cái nhìn sâu sắc hơn về cách lựa chọn và triển khai các cấu trúc dữ liệu thích hợp, cũng như hiểu rõ hơn về tầm quan trọng của giải thuật trong việc nâng cao hiệu quả tính toán và tiết kiệm tài nguyên hệ thống.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Việt Hưng – Giảng viên bộ môn Cấu trúc Dữ liệu và Giải thuật, khoa Công nghệ thông tin, người đã nhiệt tình hướng dẫn và tạo điều kiện cho chúng em trong suốt quá trình thực hiện bài tập này. Những kiến thức và sự hỗ trợ từ thầy là nguồn động lực to lớn giúp nhóm em hoàn thành bài báo cáo này.

Hy vọng rằng bản báo cáo sẽ không chỉ là kết quả của nỗ lực cá nhân mà còn là nền tảng cho những nghiên cứu và dự án chuyên sâu hơn về Cấu trúc Dữ liệu và Giải thuật trong tương lai.

Chúng em xin chân thành cảm ơn!

# PHẦN A

1. **ĐỊNH NGHĨA BÀI TOÁN**
2. **Đề bài**

*Bài toán 1(Bài số 22 trong danh sách BTL)*

* Cài đặt cấu trúc dữ liệu trừu tượng vector có bộ lặp xuôi và ngược
* Cài đặt class câu hỏi trắc nghiệm mỗi câu hỏi gồm câu hỏi dạng văn bản và 4 phương án trong đó có 1 phương án đúng
* Cài đặt class thi dùng cấu trúc vector đă xây dựng để đọc đữ liệu từng câu hỏi từng file văn bản lưu vào vector. Cho phép tạo ra đề thi gồm k câu hỏi ngẫu nhiên trong vector các câu hỏi, khi hiến thị xáo trộn ví trí các phương án trả lời cho phép người thi và tính điểm.

1. **Phân tích bài toán**

* Tìm hiểu lớp vector
* Triển khai cài đặt cấu trúc dữ liệu trừu tượng có bộ lặp xuôi và ngược
* Áp dụng vào bài toán tạo đề thi trắc nghiệm

*Input:*

Các câu hỏi trắc nghiệm (bao gồm câu hỏi dạng văn bản và bốn phương án trong đó có một phương án đúng).

Đọc dữ liệu câu hỏi từ file văn bản và lưu vào vector.

*Ouput:*

Cho phép tạo ra đề thi gồm k câu hỏi ngẫu nhiên trong vector các câu hỏi, khi hiển thị xáo trộn vị trí các phương án trả lời cho phép người thi và tính điểm.

1. **CẤU TRÚC DỮ LIỆU VECTOR**

Vector C++ là một lớp, một cấu trúc dữ liệu có trong thư viện của C++ Standard Template Library dùng để lưu các đối tượng khác giống với mảng (Array). C++ rất mạnh về tính linh hoạt và cấu trúc chính vì thế vector được sinh ra nhằm khắc phục các mặt hạn chế của mảng. Nếu như mảng chỉ có một kích thước cố định khi khai báo thì vector hoàn toàn có thể thay đổi trong khi chương trình đang hoạt động. Tức là một đối tượng vector có thể linh động về kích thước tương tự như con trỏ.

Kích thước không cố định chính là điểm nổi bật nhất của vector. Tuy nhiên không dừng lại ở đó, đối tượng này còn được trang bị thêm khá nhiều phương thức xử lý thông dụng như sort, insert, delete, legth, … Chính nhờ tính linh hoạt của nó, đôi khi sẽ hữu ích hơn trong việc giải quyết các bài toán lập trình.

*Các thao tác trên Vector:*

**Modifiers**

1. push\_back(): Hàm đẩy một phần tử vào vị trí sau cùng của vector.

2. assign(): Nó gán một giá trị mới cho các phần tử vector bằng cách thay thế các giá trị cũ.

3. pop\_back(): Hàm pop\_back () được sử dụng để xóa đi phần tử cuối cùng một vector.

4. insert(): Hàm này chèn các phần tử mới vào trước phần tử trước vị trí được trỏ bởi vòng lặp.

5. erase(): Hàm được sử dụng để xóa các phần tử tùy theo vị trí vùng chứa.

6. swap(): Hàm được sử dụng để hoán đổi nội dung của một vector này với một vector khác cùng kiểu. Kích thước có thể khác nhau.

7. clear(): Hàm được sử dụng để loại bỏ tất cả các phần tử của vùng chứa vector.

**Iterators**

1. begin(): đặt iterator đến phần tử đầu tiên trong vector.
2. end(): đặt iterator đến sau phần tử cuối cùng trong vector.
3. rbegin(): đặt reverse iterator (trình lặp đảo) đến phần tử cuối cùng trong vector (reverse begin). Nó di chuyển từ phần tử cuối cùng đến phần tử đầu tiên.
4. rend(): đặt reverse iterator (trình lặp đảo) đến phần tử đầu tiên trong vector (reverse end).
5. cbegin(): đặt constant iterator (trình vòng lặp) đến phần tử đầu tiên trong vector.
6. cend(): đặt constant iterator (trình vòng lặp) đến phần tử cuối cùng trong vector.
7. crbegin(): đặt constant reverse iterator (trình lặp đảo liên tục) đến phần tử cuối cùng trong vector (reverse begin). Nó di chuyển từ phần tử cuối cùng đến phần tử đầu tiên.
8. crend(): đặt constant reverse iterator (trình lặp đảo liên tục) đến phần tử đầu tiên trong vector.

**Capacity**

1. size(): hàm sẽ trả về số lượng phần tử đang được sử dụng trong vector.
2. max\_size(): hàm trả về số phần tử tối đa mà vector có thể chứa.
3. capacity(): hàm trả về số phần tử được cấp phát cho vector nằm trong bộ nhớ.
4. resize(n): Hàm này thay đổi kích thước vùng chứa để nó chứa đủ n phần tử. Nếu kích thước hiện tại của vector lớn hơn n thì các phần tử phía sau sẽ bị xóa khỏi vector và ngược lại nếu kích thước hiện tại nhỏ hơn n thì các phần tử bổ sung sẽ được chèn vào phía sau vector.
5. empty(): Trả về liệu vùng chứa có trống hay không, nếu trống thì trả về True, nếu có phần tử thì trả về False.
6. shrink\_to\_fit(): Giảm dung lượng của vùng chứa để phù hợp với kích thước của nó và hủy tất cả các phần tử vượt quá dung lượng.
7. reserve(n): hàm cấp cho vector số dung lượng vừa đủ để chứa n phần tử.

**Element access**

1. at(g): Trả về một tham chiếu đến phần tử ở vị trí ‘g’ trong vector.
2. data(): Trả về một con trỏ trực tiếp đến (memory array) bộ nhớ mảng được vector sử dụng bên trong để lưu trữ các phần tử thuộc sở hữu của nó.
3. front(): hàm dùng để lấy ra phần tử đầu tiên của vector.
4. back(): hàm dùng để lấy ra phần tử cuối cùng của vector.
5. **PHÂN TÍCH CHƯƠNG TRÌNH**
6. **Cài đặt cấu trúc dữ liệu trừu tượng vector**
   1. **Phương thức khởi tạo**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* 1. **Bộ lặp xuôi, bộ lặp ngược**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* 1. **Capacity**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* 1. **Modifiers**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, đại số

Mô tả được tạo tự động

* 1. **Element Access**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động** **Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

1. **KẾT QUẢ THỰC NGHIỆM**
2. **Dữ liệu**

Bộ câu hỏi và câu trả lời được lưu trữ ở file input22.txt. Chương trình sẽ đọc dữ liệu từ file input22.txt và ghi dữ liệu vào file output22.txt. Người dùng tương tác chọn k câu hỏi, chương trình tạo ngẫu nhiên và xáo trộn đáp án của k câu hỏi và tính điểm.

1. **Kết quả**

Các chức năng trong chương trình:

* Đọc bộ test trong file input, lưu vào vector, đồng thời ghi vào file output.
* Tạo bộ k câu hỏi ngẫu nhiên.
* Xáo trộn câu trả lời trong mỗi câu hỏi.
* Tính điểm của người dùng làm bài test.

*Tính năng 1: Đọc file input, ghi kết quả vào file output*

|  |  |
| --- | --- |
|  |  |

*Tính năng 2: Tạo bộ k câu hỏi ngẫu nhiên với các đáp án được xáo trộn*

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Sử dụng hàm rand() để sinh số ngẫu nhiên cho bộ k câu hỏi. Với mỗi câu hỏi, sinh một số ngẫu nhiên trong đoạn [0, 4) tượng trưng cho đáp án sẽ xáo trộn. Thay đổi hai đáp án num và 3 – num cho nhau, và chỉnh sửa lại đáp án đúng sau khi xáo trộn.

*Tính năng 3: Cho phép người dùng làm bài test và tính điểm*

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động**

# PHẦN B

* 1. **ĐỀ TÀI**

Bài toán quản lí nhà trọ

Xây dựng chương trình có các chức năng:

* Thêm phòng trọ
* Xóa phòng trọ
* In ra danh sách phòng trọ
* Tìm phòng trọ có giá thuê cao nhất
* Tìm phòng trọ có giá thuê thấp nhất
* Sắp xếp danh sách phòng trọ theo thứ tự tăng dần giá thuê
* Tìm kiếm phòng trọ theo tên người thuê trọ
* Tính tiền điện của một phòng trọ
  1. **PHÂN TÍCH CHƯƠNG TRÌNH**

1. **Lớp PhongTro**

**Chức năng** của lớp PhongTro là quản lý và lưu trữ thông tin về một phòng trọ trong danh sách các phòng trọ. Lớp này cung cấp các thuộc tính và phương thức cần thiết để:

* Lưu trữ thông tin phòng trọ(số phòng, tên người thuê, số lượng người, thời hạn, ngày thuê, tiền cọc, giá thuê, số điện, dịch vụ)
* Nhập và xuất thông tin phòng trọ: Cung cấp các toán tử nhập và xuất để dễ dàng đọc và in thông tin về một phòng trọ.
* So sánh giá thuê trọ: Cho phép so sánh gia thuê của các phòng, giúp sắp xếp các phòng theo thứ tự.
* Truy xuất giá thuê phòng, tên người thuê phòng: Hỗ trợ lấy giá thuê phòng, tên người thuê phòng để thực hiện các tác vụ khác như tìm kiếm hoặc lọc dữ liệu.
  1. **Thuộc tính**
* soPhong: Lưu trữ số phòng(tên phòng)
* tenNguoiThue: Lưu trữ họ tên người thuê phòng
* soNguoi: Lưu trữ số lượng người ở
* thoiGianThue: Lưu trữ thời hạn thuê phòng trên hợp đồng
* ngayThue, thangThue, namThue: Lưu trữ ngày tháng bắt đầu thuê phòng
* tienCoc: Lưu trữ số tiền đã cọc
* giaThue: Lưu trữ giá thuê phòng
* soDienDauKy: Lưu trữ số điện đầu kỳ
* dichVu: Lưu trữ giá dịch vụ trên một tháng
  1. **Phương thức**

Khởi tạo một đối tượng PhongTro với các thông tin của phòng trọ.

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Phương thức nhập xuất thông tin của phòng trọ.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Toán tử nhập, xuất cho phép nhập và xuất thông tin của một đối tượng.

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Phương thức tính tiền điện của một phòng trọ

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Toán tử so sánh giá thuê trọ

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, màu trắng

Mô tả được tạo tự động

Phương thức trả về giá thuê phòng, trả về tên người thuê phòng

**Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, màu trắng

Mô tả được tạo tự động**

1. **Lớp DanhSachPhongTro**

**Chức năng** của lớp DanSachPhongTro là quản lý danh sách các phòng trọ(PhongTro. Nó cung cấp các phương thức để thêm, xóa, sắp xếp, tìm kiếm và hiển thị thông tin các phòng trọ trong danh sách.

* 1. **Thuộc tính**

vector<PhongTro> danhSach: Một vector lưu trữ các đối tượng PhongTro đại diện cho các PhongTro có trong thời khóa biểu.

* 1. **Phương thức**

Thêm một phòng trọ mới vào cuối danh sách

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, hàng

Mô tả được tạo tự động

Tìm và xóa một phòng trọ theo tên người thuê. Nếu tìm thấy và xóa thành công thì hiển thị thông báo “Xoa phong tro thanh cong!” nếu không thì thông báo “Phong tro khong tim thay!”

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Hiển thị toàn bộ danh sách phòng trọ. Nếu danh sách rỗng, thông báo "Danh sach phong tro trong".

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Tìm và hiển thị phòng trọ theo tên người thuê phòng. Nếu tìm thấy, hiển thị thông tin phòng trọ. Nếu không tìm thấy, thông báo "Khong tim thay phong tro".

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Hiển thị thông tin phòng trọ có giá thuê cao nhất và thấp nhất

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Sắp xếp danh sách phòng trọ theo thứ tự tăng dần của giá thuê phòng

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Trả về một tham chiếu tới đối tượng vector<PhongTro> (một danh sách các đối tượng kiểu PhongTro).

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, màu trắng

Mô tả được tạo tự động

1. **Lớp App**

Lớp App có **chức năng** quản lý các thao tác trên danh sách phòng trọ bao gồm:

* Thêm phòng trọ mới.
* Xóa phòng trọ.
* Hiển thị danh sách phòng trọ.
* Tìm phòng có giá thuê cao nhất, thấp nhất.
* Sắp xếp danh sách phòng trọ.
* Tìm kiếm phòng theo tên người thuê.
* Tính tiền điện cho phòng trọ.
  1. **Thuộc tính:**

dsPhongTro: Đây là đối tượng của lớp DanhSachPhongTro, lưu trữ danh sách tất cả các phòng trọ.

* 1. **Phương thức:**

hienThiMenu():

Hiển thị menu với các lựa chọn thao tác mà người dùng có thể thực hiện.

chay():

Đây là phương thức điều khiển chính của chương trình. Vòng lặp do-while giúp người dùng chọn thao tác và thực hiện các chức năng tương ứng cho đến khi chọn "Thoát".

Các thao tác cụ thể trong phương thức chay():

* Case 1: Nhập thông tin phòng trọ mới và thêm vào danh sách.
* Case2:Nhập tên người thuê để xóa phòng trọ tương ứng.
* Case 3: Hiển thị tất cả phòng trọ trong danh sách.
* Case 4:Tìm và in ra phòng trọ có giá thuê cao nhất.
* Case 5:Tìm và in ra phòng trọ có giá thuê thấp nhất.
* Case 6:Sắp xếp danh sách các phòng trọ theo giá thuê và hiển thị
* Case 7:Nhập tên người thuê để tìm phòng trọ tương ứng.
* Case 8:Nhập tên người thuê và số điện hiện tại, sau đó tính tiền điện của phòng trọ.
  1. **KẾT QUẢ THỰC NGHIỆM**

1. **Thêm một phòng trọ**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

1. **Xóa một phòng trọ**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thực đơn

Mô tả được tạo tự động**

1. **In ra danh sách phòng trọ**

**Ảnh có chứa văn bản, ảnh chụp màn hình, thực đơn, Phông chữ

Mô tả được tạo tự động**

1. **Tìm kiếm phòng trọ theo tên người thuê**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

1. **Tìm phòng trọ có giá thuê cao nhất**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động**

1. **Tìm phòng trọ có giá thuê thấp nhất**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

1. **Tìm và tính tiền điện cho một phòng trọ theo tên người thuê**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

1. **Sắp xếp danh sách phòng trọ theo giá tăng dần**

**Ảnh có chứa văn bản, ảnh chụp màn hình, thực đơn, Phông chữ

Mô tả được tạo tự động**

1. **BÀI LÀM**
2. **vector.cpp**

#include<iostream>

using namespace std;

#ifndef \_\_Vector\_\_

#define \_\_Vector\_\_

template <typename T>

class Vector {

public:

class iterator;

class rev\_iterator;

/\*-----CONSTRUCTOR-----\*/

Vector(size\_t s, T value);

Vector(size\_t s);

Vector() : Size(0), Space(0) {Elements = NULL;}

Vector(T\* arr\_0, T\* arr\_n); //constructor sao chep mang

Vector(typename Vector<T>::iterator first, typename Vector<T>::iterator last); // constructor sao chep tu day iterator

Vector(const Vector<T>& V); //constructor sao chep

Vector<T>& operator=(const Vector<T>& V); //toan tu gan sao chep

/\*-----DESTRUCTOR-----\*/

~Vector();

/\*-----ITERATORS-----\*/

iterator begin() const;

iterator end() const;

const iterator cbegin() const;

const iterator cend() const;

/\*-----REVERSE ITERATORS-----\*/

rev\_iterator rbegin() const;

rev\_iterator rend() const;

const rev\_iterator crbegin() const;

const rev\_iterator crend() const;

/\*-----MODIFIERS-----\*/

void clear(); //xoa tat ca phan tu trong vector(khong thay doi dung luong)

void push\_back(const T& val);

void pop\_back();

void assign(size\_t s, const T& value); // gan s phan tu voi gia tri mac dinh value

void assign(T\* first, T\* last);

void assign(typename Vector<T>::iterator first, typename Vector<T>::iterator last);

void assign(initializer\_list<T> ilist);

iterator insert(typename Vector<T>::iterator position, const T& value);

void insert(typename Vector<T>::iterator position, size\_t n, const T& value); //chen n phan tu voi gia tri value

//iterator insert(typename Vector<T>::iterator position, typename Vector<T>::iterator first, typename Vector<T>::iterator last); //range

iterator erase(typename Vector<T>::iterator position);

iterator erase(typename Vector<T>::iterator start\_position, typename Vector<T>::iterator end\_position); //xoa phan tu tu start -> end

void swap(Vector<T>& V);

/\*-----ELEMENT ACCESS-----\*/

T& at(int position);

const T& at(int position) const;

T\* data();

const T\* data() const;

T& front();

const T& front() const;

T& back();

const T& back() const;

T& operator[](int i);

const T& operator[](int i) const;

/\*-----CAPACITY-----\*/

bool empty() const;

size\_t size() const;

size\_t capacity() const;

void resize(int new\_size, T val);

void resize(int new\_size);

void reserve(int new\_malloc);

void shrink\_to\_fit();

private:

size\_t Size; //so luong phan tu

size\_t Space; //dung luong(gom ca phan tu va khong gian trong)

T\* Elements; //con tro toi phan tu dau tien cua vectorvector

};

//rev\_iterator

template<typename T>

class Vector<T>::rev\_iterator {

private:

T\* curr;

public:

rev\_iterator (T\* p = NULL) : curr(p) {}

rev\_iterator& operator++() {

curr--;

return \*this;

}

rev\_iterator& operator++(int) {

curr--;

return \*this;

}

rev\_iterator& operator--() {

curr++;

return \*this;

}

rev\_iterator& operator--(int) {

curr--;

return \*this;

}

rev\_iterator operator+(int x) {

return curr - x;

}

rev\_iterator operator-(int x) {

return curr + x;

}

int operator-(rev\_iterator B) {

return curr - B.curr;

}

T& operator\*() {

return \*curr;

}

bool operator==(rev\_iterator B) {

return this->curr == B.curr;

}

bool operator!=(rev\_iterator B) {

return !((\*this) == B);

}

};

//iterator

template <typename T>

class Vector<T>::iterator {

public:

iterator(T\* p = NULL) : curr(p) {}

iterator& operator++() {

curr++;

return \*this;

}

iterator& operator++(int) {

curr++;

return \*this;

}

iterator& operator--() {

curr--;

return \*this;

}

iterator& operator--(int) {

curr--;

return \*this;

}

iterator operator+(int x) {

return curr + x;

}

iterator operator-(int x) {

return curr - x;

}

int operator-(iterator B) {

return this->curr - B.curr;

}

T& operator\*() {

return \*curr;

}

bool operator==(const iterator& B) const {

return curr == B.curr;

}

bool operator!=(const iterator& B) const {

return ! ((\*this ) == B);

}

private:

T\* curr;

};

//constructor

template <typename T>

inline Vector<T>::Vector(size\_t s, T value) : Size(s), Space(s), Elements(new T[s]) {

for (int i = 0; i < s; i++) Elements[i] = value;

}

template <typename T>

inline Vector<T>::Vector(size\_t s) {

Size = s;

Space = s;

Elements = new T[s];

for (int i = 0; i < s; i++) Elements[i] = T();

}

template <typename T>

inline Vector<T>::Vector(T\* arr\_0, T\* arr\_n) : Size(arr\_n - arr\_0), Space(arr\_n - arr\_0), Elements(new T[arr\_n - arr\_0]) {

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

Elements[i] = \*arr\_0;

arr\_0++;

}

}

template <typename T>

inline Vector<T>::Vector(typename Vector<T>::iterator first, typename Vector<T>::iterator last) {

Size = last - first;

Space = Size;

Elements = new T[Size];

for (int i = 0; i < Size; i++) Elements[i] = \*(first + i);

}

template <typename T>

inline Vector<T>::Vector(const Vector<T>& V) : Size(V.Size), Space(V.Space), Elements(new T[V.Space]) {

for (int i = 0; i < V.Size; i++) Elements[i] = V.Elements[i];

}

template <typename T>

inline Vector<T>& Vector<T>::operator=(const Vector<T>& V) {

if (this == &V) return \*this;

if (V.Size <= Space) {

for (int i = 0; i < V.Size; i++) {

Elements[i] = V.Elements[i];

}

Size = V.Size;

}

T\* temp = new T[V.Size];

for (int i = 0; i < V.Size; i++) temp[i] = V.Elements[i];

if (Elements != NULL) delete[] Elements;

Elements = temp;

Size = V.Size;

Space = V.Size;

return \*this;

}

template <typename T>

Vector<T>::~Vector() {

delete[] Elements;

}

//iterator

template <typename T>

inline typename Vector<T>::iterator Vector<T>::begin() const { //typename:Vector<t>::iterator

return Vector<T>::iterator(&Elements[0]);

}

template <typename T>

inline typename Vector<T>::iterator Vector<T>::end() const {

return Vector<T>::iterator(&Elements[Size]);

}

template <typename T>

inline const typename Vector<T>::iterator Vector<T>::cend() const {

return Vector<T>::iterator(&Elements[Size]);

}

template <typename T>

inline const typename Vector<T>::iterator Vector<T>::cbegin() const {

return Vector<T>::iterator(&Elements[0]);

}

//reverse iterator

template<typename T>

inline typename Vector<T>::rev\_iterator Vector<T>::rbegin() const {

return Vector<T>::rev\_iterator(&Elements[Size - 1]);

}

template<typename T>

inline typename Vector<T>::rev\_iterator Vector<T>::rend() const {

return Vector<T>::rev\_iterator(Elements - 1);

}

template<typename T>

inline const typename Vector<T>::rev\_iterator Vector<T>::crbegin() const {

return Vector<T>::rev\_iterator(&Elements[size - 1]);

}

template<typename T>

inline const typename Vector<T>::rev\_iterator Vector<T>::crend() const {

return Vector<T>::rev\_iterator(Elements - 1);

}

//capacity

template <typename T>

inline bool Vector<T>::empty() const {

return Size == 0;

}

template <typename T>

inline size\_t Vector<T>::size() const {

return Size;

}

template <typename T>

inline size\_t Vector<T>::capacity() const {

return Space;

}

template <typename T>

inline void Vector<T>::reserve(int new\_malloc) {

if (new\_malloc <= Space) return;

T\* temp = new T[new\_malloc];

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

temp[i] = Elements[i];

}

if (Elements) delete[] Elements;

Elements = temp;

Space = new\_malloc;

}

template <typename T>

inline void Vector<T>::resize(int new\_size, T val) {

reserve(new\_size);

for (int i = Size; i < new\_size; i++) Elements[i] = val;

Size = new\_size;

}

template <typename T>

inline void Vector<T>::resize(int new\_size) {

reserve(new\_size);

for (int i = Size; i < new\_size; i++) Elements[i] = T();

Size = new\_size;

}

template <typename T>

inline void Vector<T>::shrink\_to\_fit() {

T\* temp = new T[Size];

for (int i = 0; i < Size; i++) temp[i] = Elements[i];

if (Elements != NULL) delete[] Elements;

Elements = temp;

Space = Size;

}

//modifiers

template <typename T>

inline void Vector<T>::clear() {

Size = 0;

}

template <typename T>

inline void Vector<T>::push\_back(const T& val) {

if (Space == 0) reserve(8);

if (Size == Space) {

reserve(2 \* Space);

}

Elements[Size] = val;

Size++;

}

template <typename T>

inline void Vector<T>::pop\_back() {

if (Size > 0) --Size;

}

template <typename T>

inline void Vector<T>::assign(size\_t s, const T& value) {

if (Space < s) reserve(s);

Size = s;

for (int i = 0; i < Size; i++) Elements[i] = value;

}

template <typename T>

inline void Vector<T>::assign(T\* first, T\* last) {

int temp = last - first;

if (Space < temp) reserve(temp);

Size = temp;

for (int i = 0; i < Size; i++) Elements[i] = \*(first + i);

}

template <typename T>

inline void Vector<T>::assign(typename Vector<T>::iterator first, typename Vector<T>::iterator last) {

int temp = last - first;

if (Space < temp) reserve(temp);

Size = temp;

for (int i = 0; i < Size; i++) Elements[i] = \*(first + i);

}

template <typename T>

inline void Vector<T>::assign(initializer\_list<T> ilist) {

int temp = ilist.size();

if (Space < temp) reserve(temp);

Size = temp;

int id = 0;

for (auto x : ilist) {Elements[id++] = x;}

}

template <typename T>

inline typename Vector<T>::iterator Vector<T>::insert(typename Vector<T>::iterator position, const T& value) {

int temp = position - begin();

if (Size == Space) {reserve(Space \* 2);}

Size++;

for (int i = Size; i > temp; i--) {

Elements[i] = Elements[i - 1];

}

\*(begin() + temp) = value;

return begin() + temp;

} // tra ve con tro tro toi phan tu moi chen

template <typename T>

inline void Vector<T>::insert(typename Vector<T>::iterator position, size\_t n, const T& value) {

int temp = position - begin();

//cap nhat kich thuoc

if (Size + n > Space) {reserve(Size + n);}

Size = Size + n;

for (int i = Size + n - 1; i > (temp) + n - 1; i--) {

Elements[i] = Elements[i - n];

}

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

Elements[i] = value;

}

}

template <typename T>

inline typename Vector<T>::iterator Vector<T>::erase(typename Vector<T>::iterator position) {

int temp = position - begin();

for (int i = temp; i < Size - 1; i++) {

Elements[i] = Elements[i + 1];

}

Size--;

return begin() + temp;

}

template <typename T>

inline typename Vector<T>::iterator Vector<T>::erase(typename Vector<T>::iterator start\_position, typename Vector<T>::iterator end\_position) {

int temp = start\_position - begin();

int l = end\_position - start\_position;

for (int i = temp; i < Size - l; i++) {

Elements[i] = Elements[i + l];

}

Size -= l;

return begin() + temp;

}

template <typename T>

void Vector<T>::swap(Vector<T>& V) {

Vector<T> temp = V;

V = \*this;

\*this = temp;

}

//element access

template <typename T>

T& Vector<T>::at(int position) {

return Elements[position];

}

template <typename T>

const T& Vector<T>::at(int position) const {

return Elements[position];

}

template <typename T>

T& Vector<T>::front() {

return Elements[0];

}

template <typename T>

const T& Vector<T>::front() const {

return Elements[0];

}

template <typename T>

T& Vector<T>::back() {

return Elements[Size - 1];

}

template <typename T>

const T& Vector<T>::back() const {

return Elements[Size - 1];

}

template <typename T>

T& Vector<T>::operator[](int i) {

return Elements[i];

}

template <typename T>

const T& Vector<T>::operator[](int i) const {

return Elements[i];

}

template <typename T>

T\* Vector<T>::data() {

return Elements;

}

template <typename T>

const T\* Vector<T>::data() const {

return Elements;

}

#endif

1. **b22.cpp**

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include <algorithm>

#include <set>

#include "D:\Document\BTLDSA\vector.cpp"

#define endl "\n"

#define Nhom22 main

#define int long long

using namespace std;

class Answer {

private:

Vector<string> ans;

public:

Answer() {

ans.assign({"", "", "", ""});

}

string getAns(int i) {return ans[i];}

void setAns(int i, string s) {ans[i] = s;}

friend istream& operator>> (istream& is, Answer& a) {

for (int i = 0; i < 4; i++) getline(is, a.ans[i]);

return is;

}

friend ostream& operator<< (ostream& os, Answer& a) {

for (int i = 0; i < 4; i++) os << a.ans[i] << endl;

return os;

}

};

class Test {

private:

string Question;

Answer Ans;

int correctAns;

public:

Test() {}

string getQs() {return Question;}

string getAnswer(int i) {return Ans.getAns(i);}

int getCorrectAns() {return correctAns;}

void setCorrectAns(int x) {correctAns = x;}

void setAnswer(int i, string s) {Ans.setAns(i, s);}

friend istream& operator>> (istream& is, Test& t) {

getline(is, t.Question);

is >> t.Ans;

is >> t.correctAns;

is.ignore();

return is;

}

friend ostream& operator<< (ostream& os, Test& t) {

os << t.Question << endl;

os << t.Ans;

os << t.correctAns;

return os;

}

};

Nhom22() {

ifstream ifs;

ofstream ofs;

ifs.open("D:\\Document\\BTLDSA\\input22.txt");

ofs.open("D:\\Document\\BTLDSA\\output22.txt");

if (!ifs) {

cout << "Failed to open input22.txt!" << endl;

return 1;

}

if (!ofs) {

cout << "Failed to open output22.txt!" << endl;

return 1;

}

int n;

ifs >> n;

ifs.ignore();

Vector<Test> LTest;

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

Test temp;

ifs >> temp;

LTest.push\_back(temp);

}

random\_shuffle(LTest.begin(), LTest.end());

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

ofs << LTest[i] << endl;

}

srand(time(NULL));

int k;

do {

cout << "Nhap so cau hoi cho bai lam moi: ";

cin >> k;

} while (k > n);

Vector<Test> newTest;

set<int> se;

while (se.size() < k) {

int num = rand();

num %= n;

se.insert(num);

}

for (auto x : se) {

newTest.push\_back(LTest[x]);

}

for (int i = 0; i < newTest.size(); i++) {

int num = rand() % 4;

if (num == newTest[i].getCorrectAns()) {

newTest[i].setCorrectAns(3 - num);

} else if (3 - num == newTest[i].getCorrectAns()) {

newTest[i].setCorrectAns(num);

}

string tg1 = newTest[i].getAnswer(num);

string tg2 = newTest[i].getAnswer(3 - num);

newTest[i].setAnswer(num, tg2);

newTest[i].setAnswer(3 - num, tg1);

}

int user\_point = 0;

for (int i = 0; i < newTest.size(); i++) {

cout << newTest[i].getQs() << endl;

for (int j = 0; j < 4; j++) {

cout << j << ". " << newTest[i].getAnswer(j) << endl;

}

int user\_ans; cin >> user\_ans;

if (user\_ans == newTest[i].getCorrectAns()) ++user\_point;

}

cout << "You got " << user\_point << " point! Congratulations!!!";

ofs.close();

}

1. **input22.txt**

7

1. Code nay la cua ai?

Ngoc Mai

Thu Hang

Phuong Thao

Nhom 22

3

2. Ngay sinh cua Phuong Thao?

16/1

28/11

27/7

16/12

3

3. Ngay sinh cua Thu Hang?

27/11

28/11

27/7

29/7

0

4. Ngay sinh cua Ngoc Mai?

25/12

28/11

27/7

29/7

0

5. 1 + 1 = ?

2

3

4

5

0

6. Ai la giang vien mon CTDL&GT

Thay Hoang Van Thong

Thay Pham Xuan Tich

Thay Nguyen Viet Hung

Thay Le Dinh Duong

2

7. Lop CNTT4 co bao nhieu ban nu?

2

4

1

5

3

1. **quanlinhatro.cpp**

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

using namespace std;

class PhongTro {

private:

int soPhong;

string tenNguoiThue;

int soNguoi;

int thoiGianThue;

int ngayThue;

int thangThue;

int namThue;

double tienCoc;

double giaThue;

int soDienDauKy;

int dichVu;

public:

PhongTro()

: tenNguoiThue(""), soNguoi(0), thoiGianThue(0), ngayThue(0), thangThue(0),

namThue(0), tienCoc(0.0), giaThue(0.0), soDienDauKy(0), dichVu(0) {}

PhongTro(string ten, int soLuong, int thoiGian, int ngay, int thang,

int nam, double coc, double gia, int dienDau, int dv)

: tenNguoiThue(ten), soNguoi(soLuong), thoiGianThue(thoiGian), ngayThue(ngay),

thangThue(thang), namThue(nam), tienCoc(coc), giaThue(gia), soDienDauKy(dienDau), dichVu(dv) {}

// Nhap thong tin

void nhap() {

cout << "Nhap so phong:";

cin >> soPhong;

cout << "Nhap ten nguoi thue:";

cin.ignore();

getline(cin, tenNguoiThue);

cout << "Nhap so luong nguoi o:";

cin >> soNguoi;

cout << "Nhap thoi gian thue (thang):";

cin >> thoiGianThue;

cout << "Nhap ngay thue (dd/mm/yy):";

cin >> ngayThue >> thangThue >> namThue;

cout << "Nhap so tien da coc (trieu vnd):";

cin >> tienCoc;

cout << "Nhap gia thue (trieu vnd):";

cin >> giaThue;

cout << "Nhap so dien dau ky:";

cin >> soDienDauKy;

cout << "Nhap gia dich vu (internet, cap, nuoc, may giat, dien chung, rac)(nghin VND/thang):";

cin >> dichVu;

}

// Xuat thong tin

void xuat() const {

cout << "Phong " << soPhong << endl;

cout << "Ten nguoi thue: " << tenNguoiThue << endl;

cout << "Soluong nguoi o: " << soNguoi << endl;

cout << "Thoi gian thue: " << thoiGianThue << " thang" << endl;

cout << "Ngay thue: " << ngayThue << "/" << thangThue << "/" << namThue << endl;

cout << "Tien coc: " << tienCoc << " trieu VND" << endl;

cout << "Gia thue: " << giaThue << " trieu VND" << endl;

cout << "So dien dau ky: " << soDienDauKy << endl;

cout << "Dich vu: " << dichVu << "000 VND" << endl;

}

// Tinh tien dien

double tinhTienDien(int soDienHienTai) const {

int soDienSuDung = soDienHienTai - soDienDauKy;

if (soDienSuDung < 0) {

cout << "So dien hien tai khong hop le!" << endl;

return 0;

}

return soDienSuDung \* 3000.0;

}

// Toan tu <<

friend ostream& operator<<(ostream& out, const PhongTro& pt) {

pt.xuat();

return out;

}

// Toan tu >>

friend istream& operator>>(istream& in, PhongTro& pt) {

pt.nhap();

return in;

}

// Toan tu so sanh < gia thue tro

bool operator<(const PhongTro& other) const {

return this->giaThue < other.giaThue;

}

double getGiaThue() const {

return giaThue;

}

string getTenNguoiThue() const {

return tenNguoiThue;

}

};

class DanhSachPhongTro {

private:

vector<PhongTro> danhSach;

public:

// Them 1 phong tro

void themPhongTro(const PhongTro& pt) {

danhSach.push\_back(pt);

}

// Xoa phong tro theo ten nguoi thue

void xoaPhongTro(const string& tenNguoiThue) {

auto it = find\_if(danhSach.begin(), danhSach.end(), [&tenNguoiThue](const PhongTro& pt) {

return pt.getTenNguoiThue() == tenNguoiThue;

});

if (it != danhSach.end()) {

danhSach.erase(it);

cout << "Xoa phong tro thanh cong!" << endl;

} else {

cout << "Phong tro khong tim thay!" << endl;

}

}

// Tim kiem phong tro theo ten

void timPhongTroByTen(const string& tenNguoiThue) const {

auto it = find\_if(danhSach.begin(), danhSach.end(), [&tenNguoiThue](const PhongTro& pt) {

return pt.getTenNguoiThue() == tenNguoiThue;

});

if (it != danhSach.end()) {

cout << "Phong tro cua " << tenNguoiThue << " la:" << endl;

cout << \*it << endl;

} else {

cout << "Khong tim thay phong tro cua " << tenNguoiThue << endl;

}

}

// Phong tro co gia thue cao nhat

PhongTro timMaxGiaThue() const {

return \*max\_element(danhSach.begin(), danhSach.end(), [](const PhongTro& a, const PhongTro& b) {

return a.getGiaThue() < b.getGiaThue();

});

}

// Phong tro co gia thue thap nhat

PhongTro timMinGiaThue() const {

return \*min\_element(danhSach.begin(), danhSach.end(), [](const PhongTro& a, const PhongTro& b) {

return a.getGiaThue() < b.getGiaThue();

});

}

// Sap xep danh sach phong tro theo gia thue

void sapXep() {

sort(danhSach.begin(), danhSach.end());

if (danhSach.empty()) {

cout << "Danh sach phong tro trong!" << endl;

return;

}

for (const auto& pt : danhSach) {

cout << pt << endl;

}

}

// Xuat danh sach phong tro

void xuatDanhSach() const {

if (danhSach.empty()) {

cout << "Danh sach phong tro trong!" << endl;

return;

}

for (const auto& pt : danhSach) {

cout << pt << endl;

}

}

vector<PhongTro>& getDanhSach() {

return danhSach;

}

};

class App {

private:

DanhSachPhongTro dsPhongTro;

public:

void hienThiMenu() {

cout << "====== Quan ly phong tro ======" << endl;

cout << "1. Them phong tro" << endl;

cout << "2. Xoa phong tro" << endl;

cout << "3. Xuat danh sach phong tro" << endl;

cout << "4. Tim phong tro co gia thue cao nhat" << endl;

cout << "5. Tim phong tro co gia thue thap nhat" << endl;

cout << "6. Sap xep danh sach phong tro" << endl;

cout << "7. Tim kiem phong tro" << endl;

cout << "8. Tinh tien dien cua phong tro" << endl;

cout << "9. Thoat" << endl;

cout << "Chon thao tac: ";

}

void chay() {

int luaChon;

do {

hienThiMenu();

cin >> luaChon;

switch (luaChon) {

case 1: {

PhongTro pt;

cin >> pt;

dsPhongTro.themPhongTro(pt);

break;

}

case 2: {

string ten;

cout << "Nhap ten nguoi thue phong can xoa: ";

cin.ignore();

getline(cin, ten);

dsPhongTro.xoaPhongTro(ten);

break;

}

case 3:

dsPhongTro.xuatDanhSach();

break;

case 4: {

PhongTro pt = dsPhongTro.timMaxGiaThue();

cout << "Phong tro co gia thue cao nhat: " << pt << endl;

break;

}

case 5: {

PhongTro pt = dsPhongTro.timMinGiaThue();

cout << "Phong tro co gia thue thap nhat: " << pt << endl;

break;

}

case 6:

dsPhongTro.sapXep();

cout << "Danh sach phong tro da duoc sap xep!" << endl;

break;

case 7: {

string ten;

cout << "Nhap ten nguoi thue phong can tim: ";

cin.ignore();

getline(cin, ten);

dsPhongTro.timPhongTroByTen(ten);

break;

}

case 8: {

string ten;

cout << "Nhap ten nguoi thue phong can tinh tien dien: ";

cin.ignore();

getline(cin, ten);

auto it = find\_if(dsPhongTro.getDanhSach().begin(), dsPhongTro.getDanhSach().end(), [&ten](const PhongTro& pt) {

return pt.getTenNguoiThue() == ten;

});

if (it != dsPhongTro.getDanhSach().end()) {

int soDienHienTai;

cout << "Nhap so dien hien tai: ";

cin >> soDienHienTai;

double tienDien = it->tinhTienDien(soDienHienTai);

cout << "Tien dien cua phong tro: " << tienDien << " VND" << endl;

} else {

cout << "Khong tim thay phong tro cua " << ten << endl;

}

break;

}

case 9:

cout << "Thoat khoi chuong trinh." << endl;

break;

default:

cout << "Lua chon khong hop le!" << endl;

}

} while (luaChon != 7);

}

};

int main() {

App app;

app.chay();

return 0;

}

# PHẦN C

1. **KẾT LUẬN**
2. **Đánh giá mức độ hoàn thành**

* Hoàn thành đúng với mục tiêu ban đầu đề ra.
* Đã ứng dụng được kiến thức của môn học, kết hợp với một số kiến thức tự tìm hiểu để làm sản phẩm.
* Cần nghiên cứu thêm kiến thức để hình thức sản phẩm trở thành một web hay app đặc thù.

1. **Bài học rút ra**

* Môn học Cấu trúc dữ liệu và giải thuật là vô cùng quan trọng, là phần kiến thức không thể thiếu đối với sinh viên khoa Công nghệ thông tin.
* Cần sử dụng kiến thức đã học để áp dụng vào một việc nào đó để giúp nắm vững kiến thức và tìm ra những điều mới mẻ.
* Sau khi hoàn thành sản phẩm, cần nghĩ thêm để phát triển sản phẩm, khắc phục những hạn chế của sản phẩm.

1. **Khó khăn đối với môn học**

* Để học tốt môn Cấu trúc dữ liệu và giải thuật cần nắm vững kiến thức môn Kỹ thuật lập trình và Hướng đối tượng.
* Cần hiểu bản chất.
* Lượng kiến thức lớn.

1. **LỜI CẢM ƠN**

Cấu trúc dữ liệu và giải thuật là môn học rất quan trọng đối với sinh viên khoa Công nghệ thông tin. Chúng em xin chân thành cảm ơn sự chỉ dạy nhiệt tình và những ví dụ minh họa của thầy giúp em học tập. Cảm ơn thầy vì đã giao một bài tập lớn để chúng em có được cơ hội thử thách bản thân vào một project không chỉ đơn giản là code mà còn là tinh thần làm việc, tìm hiểu những thứ mới mẻ.

Cuối cùng, sản phẩm của chúng em vẫn còn nhiều thiếu sót do thiếu nhiều kinh nghiệm cũng như kỹ năng, mong thầy nhận xét và góp ý để chúng em hoàn thiện hơn.