TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

----------------o0o---------------



**Bài tập lớn môn học**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**Đề tài: Quản lý thông tin nhân viên.  
Xây dựng lớp vector và quản lý hàng hóa**

*Giảng viên hướng dẫn:*

*TS. Hoàng Văn Thông*

*Sinh viên thực hiện:*

*Nguyễn Ngọc Anh Linh*

*Lớp :*

*Công nghệ thông tin 2 – K64*

MỤC LỤC

1. Phần A
2. GIỚI THIỆU
3. Định nghĩa
4. Tiến trình hoàn thành công việc
5. PHƯƠNG PHÁP TIẾN HÀNH
6. **Phần A**
7. **Đề bài**

PHẦN A

Quản lý nhân viên bằng danh sách liên kết, mỗi nhân viên là một cấu trúc gồm:

* Mã nhân viên
* Tên nhân viên
* Bộ phận
* Số ngày đi làm
* Hệ số lương

1. Với các phương thức sau:

a. Thêm nhân viên vào danh sách.

b. Tính lương cho nhân viên và in ra, biết rằng :

Lương = Hệ số lương \* Số ngày đi làm

(đơn vị: triệu đồng).

c. In lên màn hình tất cả nhân viên có hệ số lương >= 2.5.

d. Xóa nhân viên theo mã nhân viên.

e. Sắp xếp danh sách nhân viên theo thứ tự của tên.

2. Xây dựng hàm main để kiểm tra các chức năng của lớp.

1. **Phân tích bài toán**
2. **Yêu cầu bài toán**
   1. Quản lý danh sách nhân viên bằng danh sách liên kết
   2. Mỗi nhân viên gồm:

* Mã nhân viên
* Tên nhân viên
* Tên bộ phận
* Số ngày đi làm
* Hệ số lương
  1. Các phương thức yêu cầu
* Thêm nhân viên vào danh sách
* Tính lương cho nhân viên và in ra
* In ra màn hình các nhân viên có hệ số lương >= 2.5
* Xóa nhân viên theo mã nhân viên
* Sắp xếp danh sách nhân viên theo thứ tự tên
  1. Xây dựng hàm main để kiểm tra các chức năng trên

1. **Các lớp và thuộc tính**
   1. **Lớp Employee**

**a. Thuộc tính**

* **empcode**: string
* **name**: string
* **dept**: string
* **day\_working**: int
* **salary\_perf**: float

1. **Phương thức**

* **Employee()**: Hàm tạo không đối
* **friend istream& operator>>(istream& is, Employee& emp)**: Nhập dữ liệu cho nhân viên từ đầu vào.
* **friend ostream& operator<<(ostream& os, const Employee& emp):** Xuất dữ liệu của nhân viên ra đầu ra.
* **bool operator<(const Employee& other):** So sánh tên của nhân viên để sắp xếp.
* **float tinhLuong() const**: Tính lương của nhân viên.
* **string get\_empcode() const**: Trả về mã nhân viên.
  1. **Lớp Node:**

1. **Thuộc tính**

* **ds**: Danh sách liên kết của Employee

1. **Phương thức**

* **themNV()**: Thêm nhân viên vào danh sách.
* **inDs()**: In danh sách nhân viên.
* **tinhluong()**: Tính lương cho nhân viên và in ra.
* **inHsluong()**: In các nhân viên có hệ số lương >= 2.5.
* xoaNv(): Xóa nhân viên theo mã nhân viên.
* **sapxepTen()**: Sắp xếp danh sách nhân viên theo thứ tự tên.

1. **Mô tả chức năng của từng lớp và phương thức**
   1. **Lớp Employee**

* **Hàm tạo**: Khởi tạo đối tượng nhân viên với các thuộc tính cần thiết.
* **istream& operator>>(istream& is, Employee& emp)**: Nhập dữ liệu từ đầu vào cho các thuộc tính của nhân viên.
* **ostream& operator<<(ostream& os, const Employee& emp)**: Xuất dữ liệu của nhân viên ra đầu ra.
* **bool operator<(const Employee& other)**: So sánh tên của nhân viên để sắp xếp danh sách.
* **tinhLuong()**: Tính lương của nhân viên dựa trên công thức Lương = Hệ số lương \* Số ngày đi làm.
* **get\_empcode()**: Trả về mã nhân viên.
  1. **Lớp Node**
* **themNV()**:
* Chức năng: Thêm một hoặc nhiều nhân viên vào danh sách liên kết.
* Hoạt động: Nhập số lượng nhân viên cần thêm, sau đó nhập thông tin của từng nhân viên và thêm vào danh sách.
* **inDs()**:
* Chức năng: In danh sách nhân viên hiện có trong danh sách liên kết.
* Hoạt động: Kiểm tra nếu danh sách rỗng, in thông báo "Danh sách rỗng". Nếu không, in thông tin của từng nhân viên trong danh sách.
* **tinhluong()**:
  + Chức năng: Tính và in lương của tất cả nhân viên trong danh sách.
  + Hoạt động: Tính lương cho từng nhân viên dựa trên số ngày đi làm và hệ số lương, sau đó in lương ra màn hình.
* **inHsluong()**:
  + Chức năng: In các nhân viên có hệ số lương >= 2.5.
  + Hoạt động: Duyệt danh sách nhân viên, kiểm tra hệ số lương và in thông tin các nhân viên có hệ số lương >= 2.5.
* **xoaNv()**:
  + Chức năng: Xóa một nhân viên khỏi danh sách dựa trên mã nhân viên.
  + Hoạt động: Nhập mã nhân viên cần xóa, tìm nhân viên có mã này trong danh sách và xóa khỏi danh sách.
* **sapxepTen()**:
  + Chức năng: Sắp xếp danh sách nhân viên theo thứ tự tên.
  + Hoạt động: Sử dụng hàm sort của STL để sắp xếp danh sách dựa trên tiêu chí tên nhân viên.
  1. **Hàm main để kiểm tra chức năng của lớp**

Hàm main thực hiện các chức năng như thêm, xóa, in danh sách, tính lương, in nhân viên có hệ số lương cao và sắp xếp danh sách nhân viên

1. **Cài đặt các lớp và hàm main bằng C++**
2. **Phân tích thời gian chạy**

**1. Lớp Employee**

**a. Hàm tạo Employee()**

* Thời gian chạy: O(1)
* Giải thích: Hàm này khởi tạo các thuộc tính với giá trị mặc định. Thời gian chạy không phụ thuộc vào bất kỳ yếu tố nào khác, chỉ cần thực hiện một số phép gán đơn giản.

1. **istream& operator>>(istream& is, Employee& emp)**

* Thời gian chạy: O(1)
* Giải thích: Hàm này đọc dữ liệu từ đầu vào (tối đa 5 lần đọc: mã nhân viên, tên, bộ phận, số ngày làm việc, hệ số lương). Mỗi thao tác gán giá trị chỉ tốn thời gian cố định.

1. **ostream& operator<<(ostream& os, const Employee& emp)**

* Thời gian chạy: O(1)
* Giải thích: Hàm này xuất thông tin của một nhân viên ra đầu ra (tối đa 5 lần xuất: mã nhân viên, tên, bộ phận, số ngày làm việc, hệ số lương). Mỗi thao tác in giá trị chỉ tốn thời gian cố định.

1. **bool operator<(const Employee& other)**

* Thời gian chạy: O(1)
* Giải thích: Hàm này so sánh tên của hai nhân viên. Thao tác so sánh hai chuỗi thường có thời gian chạy O(1) trong trường hợp so sánh một số lượng ký tự nhỏ và không thay đổi, do tên nhân viên thường ngắn.

1. **float tinhLuong() const**

* Thời gian chạy: O(1)
* Giải thích: Hàm này tính lương của nhân viên bằng cách nhân hệ số lương với số ngày đi làm. Đây là một phép nhân đơn giản, thời gian cố định.

1. **string get\_empcode() const**

* Thời gian chạy: O(1)
* Giải thích: Hàm này trả về mã nhân viên, thời gian cố định.

**2. Lớp Node**

**a. Hàm tạo (Node())**

* Thời gian chạy: O(1)
* Giải thích: Hàm này khởi tạo danh sách liên kết rỗng, thời gian chạy cố định.

**b. void themNV()**

* **Thời gian chạy**: O(n)
* **Giải thích**: Hàm này thêm n nhân viên vào danh sách liên kết. Giả sử n là số nhân viên cần thêm. Mỗi thao tác thêm nhân viên là O(1), nhưng cần lặp qua n nhân viên để thêm tất cả.

1. **void inDs()**

* **Thời gian chạy**: O(n)
* **Giải thích**: Hàm này in thông tin của tất cả n nhân viên trong danh sách liên kết. Mỗi thao tác in là O(1), nhưng cần lặp qua tất cả n nhân viên.

1. **void tinhluong()**

* **Thời gian chạy**: O(n)
* **Giải thích**: Hàm này tính lương cho tất cả n nhân viên trong danh sách. Mỗi thao tác tính lương là O(1), nhưng cần lặp qua tất cả n nhân viên.

1. void inHsluong()

* **Thời gian chạy**: O(n)
* **Giải thích**: Hàm này in thông tin của tất cả nhân viên có hệ số lương >= 2.5. Cần lặp qua n nhân viên để kiểm tra điều kiện và in thông tin. Mỗi thao tác kiểm tra và in là O(1).

1. void xoaNv()

* **Thời gian chạy**: O(n)
* **Giải thích**: Hàm này xóa một nhân viên khỏi danh sách dựa trên mã nhân viên. Cần lặp qua tối đa n nhân viên để tìm và xóa. Thao tác tìm kiếm và xóa có thể là O(1), nhưng cần lặp tối đa qua n nhân viên.

1. void sapxepTen()

* **Thời gian chạy**: O(n log n)
* **Giải thích**: Hàm này sắp xếp danh sách nhân viên theo tên sử dụng thuật toán sort của STL. Thuật toán sort của STL thường có độ phức tạp O(n log n).

1. Phần B
2. Đề bài

Bài 15

1. Xây dựng lớp Vector

2. Xây dựng quản lý một cửa hàng bán tạp hóa, thông tin quản lý của cửa hàng gồm:

+ Danh mục mặt hàng gồm các thông tin: Mã hàng, Tên hàng, nhà sản xuất

+ Hóa đơn mua bán hàng: Mã hóa đơn, Mã hàng, Loại hóa đơn (Bán hoặc Mua), Số lượng, Ngày mua bán, Giá mua bán.

Với các phương thức:

a. Đọc danh thông tin quản lý của cửa hàng từ file vào chương trình.

b. Ghi thông tin quản lý của cửa hàng vào file

c. Nhập xuất các mặt hàng

d. Hiển thị toàn bộ danh sách mặt hàng hiện có e. Thống kê số lượng mặt hàng tồn của mỗi mặt hàng

3. Viết hàm main thực hiện các chức năng trên

1. Phân tích bài toán
2. Yêu cầu bài toán
   1. Xây dựng một lớp Vector để lưu trữ động danh sách các đối tượng.
   2. Xây dựng các lớp để quản lý thông tin của cửa hàng:

* **Danh mục mặt hàng**: chứa thông tin về mã hàng, tên hàng, nhà sản xuất.
* **Hóa đơn mua bán hàng**: chứa thông tin về mã hóa đơn, mã hàng, loại hóa đơn (mua/bán), số lượng, ngày và giá mua bán.
  1. Các phương thức yêu cầu
* Đọc thông tin từ file vào chương trình.
* Ghi thông tin từ chương trình vào file.
* Nhập, xuất danh sách các mặt hàng và hóa đơn.
* Hiển thị danh sách mặt hàng.
* Thống kê số lượng tồn kho của từng mặt hàng.
  1. Viết hàm main để thực hiện các chức năng trên.

1. Các lớp và thuộc tính
   1. Lớp Vector<T>
   2. Thuộc tính

* data: Mảng động chứa dữ liệu.
* size: Số lượng phần tử hiện có trong vector.
* capacity: Dung lượng hiện tại của vector.
  1. Phương thức
* Vector(): Hàm tạo khởi tạo vector với kích thước và dung lượng mặc định.
* ~Vector(): Hàm hủy giải phóng bộ nhớ.
* get\_size(): Trả về số lượng phần tử hiện có trong vector.
* resize(int new\_size): Thay đổi dung lượng của vector.
* push\_back(const T& v): Thêm phần tử vào cuối vector.
* operator[](int index): Truy xuất phần tử tại vị trí index.
* display(): Hiển thị tất cả phần tử trong vector.
* erase(const T& value): Xóa phần tử có giá trị value.
* insert(int index, const T& value): Chèn phần tử vào vị trí index.
* assign(int count, const T& value): Gán một giá trị hoặc một danh sách các giá trị cho các phần tử của vector.
* clear(): Xóa tất cả các phần tử trong vector nhưng không thay đổi dung lượng của vector.
* swap(Vector& other): Hoán đổi nội dung của vector hiện tại với một vector khác.
* max\_size(): Trả về số lượng phần tử tối đa mà vector có thể chứa.
* empty(): Kiểm tra xem vector có rỗng hay không.
* reserve(int new\_capacity): Đặt dung lượng của vector ít nhất bằng giá trị được chỉ định.
  1. Lớp DanhMuc

a. Thuộc tính

* mahang: Mã hàng.
* tenhang: Tên hàng.
* nsx: Nhà sản xuất.
* sl: Số lượng.

b. Phương thức

* DanhMuc(): Hàm tạo khởi tạo với giá trị mặc định.
* DanhMuc(string ma, string ten, string sx): Hàm tạo khởi tạo với các giá trị truyền vào.
* display(): Hiển thị thông tin sản phẩm.
* get\_mahang(): Trả về mã hàng.
* get\_tenhang(): Trả về tên hàng.
* get\_sl(): Trả về số lượng.
* set\_sl(int sl): Thiết lập số lượng.
* operator==(const DanhMuc& other): So sánh hai đối tượng DanhMuc.
* friend ostream& operator<<(ostream& os, const DanhMuc& mathang): Xuất thông tin sản phẩm ra luồng đầu ra.
* friend istream& operator>>(istream& is, DanhMuc& mathang): Nhập thông tin sản phẩm từ luồng đầu vào.
* friend ofstream& operator<<(ofstream& os, const DanhMuc& mathang): Xuất thông tin sản phẩm ra file.
* friend ifstream& operator>>(ifstream& is, DanhMuc& mathang): Nhập thông tin sản phẩm từ file.
  1. Lớp HoaDon

a. Thuộc tính

* mahd: Mã hóa đơn.
* mahang: Mã hàng.
* loaihd: Loại hóa đơn.
* slg: Số lượng.
* ngay: ngày
* gia: giá

b. Phương thức

* HoaDon(): Hàm tạo khởi tạo với giá trị mặc định.
* HoaDon(string hd, string ma, string loai, int solg, int day, int price): Hàm tạo khởi tạo với các giá trị truyền vào.
* display(): Hiển thị thông tin hóa đơn.
* get\_loaihd(): Trả về loại hóa đơn.
* get\_mahang(): Trả về mã hàng.
* get\_slg(): Trả về số lượng hàng.
* friend ostream& operator<<(ostream& os, const HoaDon& hd): Xuất thông tin hóa đơn ra luồng đầu ra.
* friend istream& operator>>(istream& is, HoaDon& hd): Nhập thông tin hóa đơn từ luồng đầu vào.
* friend ofstream& operator<<(ofstream& os, const HoaDon& hd): Xuất thông tin hóa đơn ra file.
* friend ifstream& operator>>(ifstream& is, HoaDon& hd): Nhập thông tin hóa đơn từ file.
  1. Lớp CuaHang

a. Thuộc tính

* mathang: Vector chứa các đối tượng DanhMuc.
* hd: Vector chứa các đối tượng HoaDon.
* newmh: Vector chứa các đối tượng DanhMuc mới nhập.
* newhd: Vector chứa các đối tượng HoaDon mới nhập.

b. Phương thức

* laydl(const string& filename, bool isMathang): Đọc dữ liệu từ file vào vector.
* ghidl(const string& filename, bool isMathang): Ghi dữ liệu từ vector vào file.
* nhapmathang(): Nhập thông tin mặt hàng từ người dùng và lưu vào vector.
* xuatmathang(): Hiển thị thông tin các mặt hàng mới nhập.
* nhaphoadon(): Nhập thông tin hóa đơn từ người dùng và lưu vào vector.
* xuathoadon(): Hiển thị thông tin các hóa đơn mới nhập.
* hienthimathang(const string& filename): Hiển thị thông tin mặt hàng từ file và từ người dùng.
* thongke(): Thống kê số lượng hàng tồn kho.

1. Mô tả chức năng của từng phương thức
   1. Lớp Vector<T>

* Vector(): Khởi tạo vector với kích thước và dung lượng mặc định.
* ~Vector(): Giải phóng bộ nhớ đã cấp phát cho mảng data.
* get\_size(): Trả về số lượng phần tử hiện có trong vector.
* resize(int new\_size): Thay đổi dung lượng của vector, cấp phát lại bộ nhớ nếu cần.
* push\_back(const T& v): Thêm phần tử v vào cuối vector, tăng kích thước vector.
* operator[](int index): Truy xuất phần tử tại vị trí index.
* display(): Hiển thị tất cả phần tử trong vector.
* erase(const T& value): Tìm và xóa phần tử có giá trị bằng value.
* insert(int index, const T& value): Chèn phần tử value vào vị trí index.
* assign(int count, const T& value): Gán một giá trị hoặc một danh sách các giá trị cho các phần tử của vector.
* clear(): Xóa tất cả các phần tử trong vector nhưng không thay đổi dung lượng của vector.
* swap(Vector& other): Hoán đổi nội dung của vector hiện tại với một vector khác.
* max\_size(): Trả về số lượng phần tử tối đa mà vector có thể chứa.
* empty(): Kiểm tra xem vector có rỗng hay không.
* reserve(int new\_capacity): Đặt dung lượng của vector ít nhất bằng giá trị được chỉ định.
  1. Lớp DanhMuc
* **DanhMuc()**: Khởi tạo các thuộc tính với giá trị mặc định.
* **DanhMuc(string ma, string ten, string sx)**: Khởi tạo các thuộc tính với các giá trị truyền vào.
* **display()**: Hiển thị thông tin sản phẩm.
* **get\_mahang()**: Trả về mã hàng.
* **get\_tenhang()**: Trả về tên hàng.
* **get\_sl()**: Trả về số lượng.
* **set\_sl(int sl)**: Thiết lập số lượng.
* **operator==(const DanhMuc& other)**: So sánh hai đối tượng DanhMuc.
* **friend ostream& operator<<(ostream& os, const DanhMuc& mathang)**: Xuất thông tin sản phẩm ra luồng đầu ra.
* **friend istream& operator>>(istream& is, DanhMuc& mathang)**: Nhập thông tin sản phẩm từ luồng đầu vào.
* **friend ofstream& operator<<(ofstream& os, const DanhMuc& mathang)**: Xuất thông tin sản phẩm ra file.
* **friend ifstream& operator>>(ifstream& is, DanhMuc& mathang)**: Nhập thông tin sản phẩm từ file.
  1. Lớp HoaDon
* **HoaDon()**: Khởi tạo các thuộc tính với giá trị mặc định.
* **HoaDon(string hd, string ma, string loai, int solg, int day, int price)**: Khởi tạo các thuộc tính với các giá trị truyền vào.
* **display()**: Hiển thị thông tin hóa đơn.
* **get\_loaihd()**: Trả về loại hóa đơn.
* **get\_mahang()**: Trả về mã hàng.
* **get\_slg()**: Trả về số lượng hàng.
* **friend ostream& operator<<(ostream& os, const HoaDon& hd)**: Xuất thông tin hóa đơn ra luồng đầu ra.
* **friend istream& operator>>(istream& is, HoaDon& hd)**: Nhập thông tin hóa đơn từ luồng đầu vào.
* **friend ofstream& operator<<(ofstream& os, const HoaDon& hd)**: Xuất thông tin hóa đơn ra file.
* **friend ifstream& operator>>(ifstream& is, HoaDon& hd)**: Nhập thông tin hóa đơn từ file.
  1. Lớp CuaHang
* **laydl(const string& filename, bool isMathang)**:
* **Chức năng**: Đọc dữ liệu từ file và lưu vào vector.
* **Mô tả chi tiết**: Hàm này kiểm tra file có mở được không. Nếu không, thông báo lỗi và trả về. Nếu mở được file, hàm đọc từng dòng dữ liệu và lưu vào vector tương ứng (mặt hàng hoặc hóa đơn).
* **ghidl(const string& filename, bool isMathang)**:
* **Chức năng**: Ghi dữ liệu từ vector vào file.
* **Mô tả chi tiết**: Hàm này kiểm tra file có mở được không. Nếu không, thông báo lỗi và trả về. Nếu mở được file, hàm ghi từng phần tử từ vector tương ứng vào file.
* **nhapmathang()**:
  + - **Chức năng**: Nhập thông tin mặt hàng từ người dùng và lưu vào vector.
* **Mô tả chi tiết**: Hàm này yêu cầu người dùng nhập số lượng mặt hàng cần nhập, sau đó nhập thông tin cho từng mặt hàng và lưu vào vector mathang và newmh.
* **xuatmathang()**:
  + - * **Chức năng**: Hiển thị thông tin các mặt hàng mới nhập.
      * **Mô tả chi tiết**: Hàm này kiểm tra vector newmh có phần tử nào không. Nếu không có, thông báo danh sách trống. Nếu có, hàm hiển thị thông tin từng mặt hàng trong vector.
* **nhaphoadon()**:
  + - * **Chức năng**: Nhập thông tin hóa đơn từ người dùng và lưu vào vector.
      * **Mô tả chi tiết**: Hàm này yêu cầu người dùng nhập số lượng hóa đơn cần nhập, sau đó nhập thông tin cho từng hóa đơn và lưu vào vector hd và newhd.
* **xuathoadon()**:
  + - * **Chức năng**: Hiển thị thông tin các hóa đơn mới nhập.
      * **Mô tả chi tiết**: Hàm này kiểm tra vector newhd có phần tử nào không. Nếu không có, thông báo danh sách trống. Nếu có, hàm hiển thị thông tin từng hóa đơn trong vector.
* **hienthimathang(const string& filename)**:
  + - * **Chức năng**: Hiển thị thông tin mặt hàng từ file và từ người dùng.
      * **Mô tả chi tiết**: Hàm này mở file và đọc từng mặt hàng từ file, sau đó hiển thị thông tin của chúng. Sau khi đóng file, hàm kiểm tra vector newmh có phần tử nào không. Nếu có, hàm hiển thị thông tin từng mặt hàng từ vector newmh.
* **thongke()**:
  + - * **Chức năng**: Thống kê số lượng hàng tồn kho.
      * **Mô tả chi tiết**: Hàm này lặp qua từng mặt hàng trong vector mathang và newmh, tính toán số lượng tồn kho bằng cách cộng dồn số lượng từ vector mathang và newmh, sau đó trừ đi số lượng đã bán (dựa trên hóa đơn). Hàm hiển thị thông tin số lượng tồn kho của từng mặt hàng.
  1. **Hàm main để kiểm tra chức năng của lớp**

Hàm main thực hiện các chức năng như thêm, xóa, in danh sách, tính lương, in nhân viên có hệ số lương cao và sắp xếp danh sách nhân viên

1. Cài đặt các lớp và hàm main bằng C++

}

1. **Phân tích thời gian chạy**
   1. Lớp Vector<T>
   2. **Hàm tạo (**Vector()**)**

* **Thời gian chạy**: O(1)
* **Giải thích**: Khởi tạo vector với kích thước và dung lượng mặc định. Thời gian chạy cố định do chỉ có các thao tác khởi tạo đơn giản.
  1. **Hàm hủy (**~Vector()**)**
* **Thời gian chạy**: O(1)
* **Giải thích**: Giải phóng bộ nhớ đã cấp phát cho mảng data. Thời gian chạy cố định do chỉ có một thao tác giải phóng bộ nhớ.
  1. int get\_size()
* **Thời gian chạy**: O(1)
* **Giải thích**: Trả về kích thước hiện tại của vector. Thao tác truy xuất đơn giản.
  1. void resize(int new\_size)
* **Thời gian chạy**: O(n)
* **Giải thích**: Nếu new\_size lớn hơn capacity, cần cấp phát bộ nhớ mới và sao chép dữ liệu từ mảng cũ sang mảng mới. Thời gian chạy phụ thuộc vào kích thước hiện tại của vector (n).
  1. void push\_back(const T& v)
* **Thời gian chạy**: O(1) trung bình, O(n) trong trường hợp xấu nhất
* **Giải thích**: Khi vector chưa đầy, chỉ cần thêm phần tử vào cuối mảng (O(1)). Nếu vector đầy, cần cấp phát lại bộ nhớ và sao chép dữ liệu cũ sang mảng mới (O(n)).
  1. T& operator[](int index)
* **Thời gian chạy**: O(1)
* **Giải thích**: Truy xuất phần tử tại vị trí index. Thao tác truy xuất đơn giản.
  1. void display() const
* **Thời gian chạy**: O(n)
* **Giải thích**: In tất cả các phần tử trong vector. Cần lặp qua tất cả n phần tử để in.
  1. void erase(const T& value)
* **Thời gian chạy**: O(n)
* **Giải thích**: Tìm và xóa phần tử có giá trị bằng value. ần lặp qua tất cả n phần tử để tìm và xóa.
  1. void insert(int index, const T& value)
* **Thời gian chạy**: O(n)
* **Giải thích**: Chèn phần tử vào vị trí index. Cần dịch chuyển các phần tử từ index đến cuối mảng sang phải. Thao tác này yêu cầu thời gian O(n).
  1. **assign(int count, const T& value)**
* **Thời gian chạy**: O(n)
* **Giải thích**: Trong trường hợp bình thường, hàm này cần gán count giá trị cho các phần tử của vector, điều này yêu cầu thời gian O(n). Nếu dung lượng hiện tại của vector không đủ để chứa count phần tử, vector cần cấp phát lại bộ nhớ và sao chép dữ liệu cũ, dẫn đến thời gian chạy có thể tăng lên O(n).
  1. **clear()**
* **Thời gian chạy**: O(1)
* **Giải thích**: Hàm này chỉ cần thay đổi kích thước (size) của vector về 0 mà không cần thực hiện bất kỳ thao tác phức tạp nào. Do đó, thời gian chạy là O(1).
  1. **swap(Vector& other)**
* **Thời gian chạy**: O(1)
* **Giải thích**: Hàm này hoán đổi con trỏ dữ liệu và các thuộc tính kích thước, dung lượng giữa hai vector. Do các thao tác này thực hiện trong thời gian cố định, thời gian chạy là O(1).
  1. **max\_size()**
* **Thời gian chạy**: O(1)
* **Giải thích**: Hàm này chỉ trả về một hằng số cố định đại diện cho số lượng phần tử tối đa mà vector có thể chứa. Do đó, thời gian chạy là O(1).
  1. **empty()**
* **Thời gian chạy**: O(1)
* **Giải thích**: Hàm này chỉ kiểm tra kích thước hiện tại của vector (size) có bằng 0 hay không và trả về kết quả. Do đó, thời gian chạy là O(1).
  1. **reserve(int new\_capacity)**
* **Thời gian chạy**: O(n)
* **Giải thích**: Nếu new\_capacity lớn hơn dung lượng hiện tại, vector cần cấp phát lại bộ nhớ mới và sao chép dữ liệu cũ sang vùng nhớ mới. Thời gian chạy phụ thuộc vào số lượng phần tử hiện có trong vector, dẫn đến thời gian chạy O(n).
  1. Lớp DanhMuc

a. **Hàm tạo không đối (**DanhMuc()**)**

* **Thời gian chạy**: O(1)
* **Giải thích**: Khởi tạo các thuộc tính với giá trị mặc định. Thao tác đơn giản.

b. **Hàm tạo có đối (**DanhMuc(string ma, string ten, string sx)**)**

* **Thời gian chạy**: O(1)
* **Giải thích**: Khởi tạo các thuộc tính với giá trị được truyền vào. Thao tác đơn giản.

c. void display()

* **Thời gian chạy**: O(1)
* **Giải thích**: In thông tin của danh mục. Thao tác in đơn giản.

d. string get\_mahang()

* **Thời gian chạy**: O(1)
* **Giải thích**: Trả về mã hàng. Thao tác truy xuất đơn giản.

e. string get\_tenhang()

* **Thời gian chạy**: O(1)
* **Giải thích**: Trả về tên hàng. Thao tác truy xuất đơn giản.

f. int get\_sl()

* **Thời gian chạy**: O(1)
* **Giải thích**: Trả về số lượng. Thao tác truy xuất đơn giản.

g. void set\_sl(int sl)

* **Thời gian chạy**: O(1)
* **Giải thích**: Thiết lập số lượng. Thao tác gán đơn giản.

h. bool operator==(const DanhMuc& other) const

* **Thời gian chạy**: O(1)
* **Giải thích**: So sánh các thuộc tính của hai đối tượng. Thao tác so sánh đơn giản.

i. friend ostream& operator<<(ostream& os, const DanhMuc& mathang)

* **Thời gian chạy**: O(1)
* **Giải thích**: Xuất thông tin của danh mục ra đầu ra. Thao tác in đơn giản.
  1. friend istream& operator>>(istream& is, DanhMuc& mathang)
* **Thời gian chạy**: O(1)
* **Giải thích**: Nhập thông tin của danh mục từ đầu vào. Thao tác nhập đơn giản.
  1. friend ofstream& operator<<(ofstream& os, const DanhMuc& mathang)
* **Thời gian chạy**: O(1)
* **Giải thích**: Xuất thông tin của danh mục ra file. Thao tác ghi đơn giản.
  1. friend ifstream& operator>>(ifstream& is, DanhMuc& mathang)
* **Thời gian chạy**: O(1)
* **Giải thích**: Nhập thông tin của danh mục từ file. Thao tác đọc đơn giản.
  1. Lớp HoaDon

**a. Hàm tạo không đối (**HoaDon()**)**

* **Thời gian chạy**: O(1)
* **Giải thích**: Hàm này khởi tạo các thuộc tính với giá trị mặc định. Thao tác đơn giản không phụ thuộc vào bất kỳ yếu tố nào khác.
  1. **Hàm tạo có đối (**HoaDon(string hd, string ma, string loai, int solg, int day, int price)**)**
* **Thời gian chạy**: O(1)
* **Giải thích**: Hàm này khởi tạo các thuộc tính với các giá trị truyền vào. Thao tác đơn giản và cố định.
  1. void display()
* **Thời gian chạy**: O(1)
* **Giải thích**: Hàm này in thông tin của hóa đơn. Thao tác in đơn giản không phụ thuộc vào bất kỳ yếu tố nào khác.
  1. string get\_loaihd()
* **Thời gian chạy**: O(1)
* **Giải thích**: Trả về loại hóa đơn. Thao tác truy xuất thuộc tính đơn giản.
  1. string get\_mahang()
* **Thời gian chạy**: O(1)
* **Giải thích**: Trả về mã hàng. Thao tác truy xuất thuộc tính đơn giản.
  1. int get\_slg()
* **Thời gian chạy**: O(1)
* **Giải thích**: Trả về số lượng hàng. Thao tác truy xuất thuộc tính đơn giản.
  1. friend ostream& operator<<(ostream& os, const HoaDon& hd)
* **Thời gian chạy**: O(1)
* **Giải thích**: Xuất thông tin của hóa đơn ra luồng đầu ra. Thao tác in đơn giản không phụ thuộc vào bất kỳ yếu tố nào khác.
  1. friend istream& operator>>(istream& is, HoaDon& hd)
* **Thời gian chạy**: O(1)
* **Giải thích**: Nhập thông tin của hóa đơn từ luồng đầu vào. Thao tác nhập đơn giản không phụ thuộc vào bất kỳ yếu tố nào khác.
  1. friend ofstream& operator<<(ofstream& os, const HoaDon& hd)
* **Thời gian chạy**: O(1)
* **Giải thích**: Xuất thông tin của hóa đơn ra file. Thao tác ghi đơn giản không phụ thuộc vào bất kỳ yếu tố nào khác.
  1. friend ifstream& operator>>(ifstream& is, HoaDon& hd)
* **Thời gian chạy**: O(1)
* **Giải thích**: Nhập thông tin của hóa đơn từ file. Thao tác đọc đơn giản không phụ thuộc vào bất kỳ yếu tố nào khác.
  1. Lớp CuaHang

a. void laydl(const string& filename, bool isMathang)

* **Thời gian chạy**: O(n)
* **Giải thích**: Đọc dữ liệu từ file và lưu vào vector. Thời gian chạy phụ thuộc vào số lượng phần tử n trong file.

b. void ghidl(const string& filename, bool isMathang)

* **Thời gian chạy**: O(n)
* **Giải thích**: Ghi dữ liệu từ vector vào file. Thời gian chạy phụ thuộc vào số lượng phần tử n trong vector.
  1. void nhapmathang()
* **Thời gian chạy**: O(n)
* **Giải thích**: Nhập n mặt hàng từ đầu vào và lưu vào vector. Thao tác nhập dữ liệu và lưu vào vector cần lặp qua tất cả n phần tử.
  1. void xuatmathang()
* **Thời gian chạy**: O(n)
* **Giải thích**: Hiển thị thông tin các mặt hàng mới nhập. Thao tác in dữ liệu của từng phần tử trong vector newmh cần lặp qua n phần tử.
  1. void nhaphoadon()
* **Thời gian chạy**: O(n)
* **Giải thích**: Nhập n hóa đơn từ đầu vào và lưu vào vector. Thao tác nhập dữ liệu và lưu vào vector cần lặp qua tất cả n phần tử.
  1. void xuathoadon()
* **Thời gian chạy**: O(n)
* **Giải thích**: Hiển thị thông tin các hóa đơn mới nhập. Thao tác in dữ liệu của từng phần tử trong vector newhd cần lặp qua n phần tử.
  1. void hienthimathang(const string& filename)
* **Thời gian chạy**: O(n)
* **Giải thích**: Đọc và hiển thị thông tin mặt hàng từ file và từ vector newmh. Thao tác đọc và in dữ liệu cần lặp qua n phần tử trong file và newmh.
  1. void thongke()
* **Thời gian chạy**: O(n^2)
* **Giải thích**: Thống kê số lượng hàng tồn kho. Cần lặp qua tất cả n phần tử trong mathang và newmh, và kiểm tra n phần tử trong hd để tính toán số lượng tồn kho. Thao tác lặp lồng nhau dẫn đến thời gian chạy O(n^2).

1. Danh sách tài liệu tham khảo
   1. <https://lms.utc.edu.vn/>
   2. <https://www.w3schools.com/dsa/>
   3. <https://topdev.vn/blog/vector-trong-c/>
   4. <https://chatgpt.com/>