1. Các loại thư viện

* #include <iomanip>: khai báo để sử dụng hàm setw
* #include <fstream>: khai báo để làm việc với file
* #include <vector>: cung cấp các hàm xử lí vector, vector là một mảng động có thể thay đổi kích thước
* #include <string>: khai báo để sử dụng chuỗi
* #include <sstream>: khai báo để sử dụng stringstream
* #include <ctime>: sử dụng hàm sleep
* #include <windows.h>: khai báo để sử dụng các hàm đồ họa như SetColor, GoTo, SetRandomColor.
* Còn cái #include “tên file”: để bao gồm file đó vào file hiện tại giúp tránh việc lặp lại file. Việc chia file giúp quản lí dữ liệu và chỉnh sửa các hàm dễ dàng.

1. Class SanPham

* **Private:**

+ STT, Hang, TenSanPham: Các biến kiểu string lưu thông tin về số thứ tự, hãng và tên sản phẩm của một sản phẩm.

+ Gia, SoLuong: Các biến kiểu int lưu giá và số lượng sản phẩm.

* **Public:**
* **Constructor:**

+ SanPham (): Constructor mặc định không có đối số.

+SanPham (string STT, string Hang, string TenSanPham, int Gia, int SoLuong): Constructor có tham số để khởi tạo các thuộc tính của đối tượng.

* **Các phương thức set:**

+ setSTT, setHang, setTenSanPham, setGia, setSoLuong: Các phương thức này sử dụng để thiết lập cho các thuộc tính của đối tượng SanPham.

* **Các phương thức get:**

+ getSTT, getHang, getTenSanPham, getGia, getSoLuong: Các phương thức này được sử dụng để trả về giá trị của các thuộc tính của đối tượng SanPham. Các phương thức này được định nghĩa là const, chúng không thay đổi trạng thái của đối tượng.

1. Class User

* **Private:**

+ un, pw: Các biến kiểu string lưu thông tin về tên đăng nhập và mật khẩu của một tài khoản User.

* **Public:**
* **Constructor:**
* **Set:**
* **Get:**
* **Mua(vector<SanPham> &DS\_SanPham, const string &stt, int soluong)**

+ Tham số:

* DS\_SanPham: tham chiếu đến 1 vector có đối tượng là SanPham 🡪 Giúp tạo ra một danh sách vector chứa các sản phẩm có sẵn.
* stt: xác định stt sản phẩm cần mua
* soluong: xác định số lượng cần mua

+ Khởi tạo 1 biến kt = false để kiểm tra sản phẩm đã được mua hay có tồn tại trong danh sách sản phẩm hay chưa.

+ Chạy vòng lặp for each duyệt qua các đối tượng trong vector DS\_SanPham

+ Kiểm tra xem stt mà mình nhập có khớp với stt có trong danh sách sản phẩm hay không

+ Nếu có là chỉ ra danh sách sản phẩm có tồn tại. Kiểm tra xem số lượng mình nhập có lớn hơn số lượng sản phẩm hiện có hay không. Tạo biến số lượng mới = số lượng sản phẩm hiện có – số lượng sản phẩm mình mua và set lại số lượng mới

+ Nếu không thì số lượng ko đủ để mua

+ Kiểm tra lại biến kt nếu true thì có sp mua, nếu false thì thôi

* **Hàm tăng dần sản phẩm theo giá tiền**
* **Hàm giảm dần sản phẩm theo giá tiền**
* **Hàm** **us1(const vector<SanPham> &DS\_SanPham) // Chức năng User**

+ Duyệt vòng lặp for each và in ra thông tin từng sản phẩm

* **Hàm convert (string s)**

+ Chuyển đổi các chữ cái về dạng thường

* **Hàm us2(const vector<SanPham> &DS\_SanPham, const string &tensp)**

+ Tham số:

* DS\_SanPham: tham chiếu đến 1 vector có đối tượng là SanPham 🡪 Giúp tạo ra một danh sách vector chứa các sản phẩm có sẵn.
* tensp: biến để người dùng tìm sản phẩm cần tìm kiếm

+ Khởi tạo biến kt = false

+ Khởi tạo biến tentim = convert(tensp): chuyển tên cần tìm sang chữ thường

+ Chạy vòng lặp for each duyệt qua các đối tượng trong vector DS\_SanPham

+ Khởi tạo biến TenspHienCo = convert(sp.getTenSanPham()): chuyển tên sản phẩm trong danh sách sản phẩm sang chữ thường

+ Kiểm tra xem tentim có tồn tại trong TenspHienCo hay không

+ Nếu có thì in ra thông tin sản phẩm

+ Kiểm tra lại biến kt nếu true thì tìm thấy sản phẩm, nếu false là không tìm thấy sản phẩm

* **Hàm us3(const vector<SanPham> &DS\_SanPham, const string &hang)**

+ Tương tự us2

* **Hàm us4(const vector<SanPham> &DS\_SanPham)**

+ Hàm ni dễ chỉ có sắp xếp thôi

* **Hàm MenuUser(User &us, vector<SanPham> DS\_SanPham)**