Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: Об’єктно-орієнтоване програмування

**Лабораторна робота №7**

**Тема:** **«**Контейнерні класи. Стандартна бібліотека шаблонів (STL) в С++.**»**

|  |
| --- |
| Виконав: ст. гр. КI-23 |
| Чепіль В.О. |
| Перевірив викладач:  Козірова Н. Л. |
|  |

Кропивницький 2024

**Тема:** Контейнерні класи. Стандартна бібліотека шаблонів (STL) в С++.

**Мета:** ознайомитись Контейнерні класи та навчитись їх програмно реалізовувати мовою С++.

**Варіант №4**

Розробіть систему управління складом магазину автозапчастин. У цій

системі потрібно зберігати та керувати інформацією про наявні

автозапчастини. Використовуйте контейнерні класи зі стандартної бібліотеки

шаблонів (STL) для збереження та операцій над списком запчастин.

• Додайте декілька запчастин до складу, включаючи їх назву, виробника,

ціну та кількість на складі.

• Виведіть список запчастин на складі.

• Видаліть одну або декілька запчастин зі складу за назвою.

• Оновіть ціну та кількість для певної запчастини.

• Виведіть оновлений список запчастин на складі.

Ви маєте розширити програму, додати додаткові методи та функціональні

можливості, які вам здаються відповідними.

**Лістинг Task:**

**SparePart.h:**

#ifndef SPAREPART\_H

#define SPAREPART\_H

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

class SparePart {

private:

string name;

string manufacturer;

double price;

int quantity;

public:

SparePart();

SparePart(string name, string manufacturer, double price, int quantity);

string getName() const;

string getManufacturer() const;

double getPrice() const;

int getQuantity() const;

void setName(string name);

void setManufacturer(string manufacturer);

void setPrice(double price);

void setQuantity(int quantity);

void display() const;

};

#endif // SPAREPART\_H

**SparePart.cpp:**

#include "sparePart.h"

#include <iostream>

#include <iomanip>

SparePart::SparePart() : name(""), manufacturer(""), price(0.0), quantity(0) {}

SparePart::SparePart(string name, string manufacturer, double price, int quantity)

: name(name), manufacturer(manufacturer), price(price), quantity(quantity) {}

string SparePart::getName() const

{

return name;

}

string SparePart::getManufacturer() const

{

return manufacturer;

}

double SparePart::getPrice() const

{

return price;

}

int SparePart::getQuantity() const

{

return quantity;

}

void SparePart::setName(string name)

{

this->name = name;

}

void SparePart::setManufacturer(string manufacturer)

{

this->manufacturer = manufacturer;

}

void SparePart::setPrice(double price)

{

if (price >= 0)

{

this->price = price;

}

else

{

cout << "The price cannot be negative" << endl;

}

}

void SparePart::setQuantity(int quantity)

{

if (quantity >= 0)

{

this->quantity = quantity;

}

else

{

cout << "The number cannot be negative!" << endl;

}

}

void SparePart::display() const

{

cout << left << "|" << setw(20) << name

<< "|" << setw(20) << manufacturer

<< "|" << setw(10) << price

<< "|" << setw(10) << quantity << "|" << endl;

}

**Warehouse.h:**

#ifndef WAREHOUSE\_H

#define WAREHOUSE\_H

#include <vector>

#include "sparePart.h"

using namespace std;

class Warehouse {

private:

vector<SparePart> parts;

vector<SparePart>::iterator findPartByName(const string& name);

public:

void addPart(const SparePart& part);

void displayParts() const;

void removePart(const string& name);

void updatePart(const string& name, double newPrice, int newQuantity);

};

#endif // WAREHOUSE\_H

**Warehouse.cpp:**

#include "warehouse.h"

#include <iostream>

#include <algorithm>

#include <iomanip>

using namespace std;

vector<SparePart>::iterator Warehouse::findPartByName(const string& name)

{

return find\_if(parts.begin(), parts.end(), [&name](const SparePart& part)

{

return part.getName() == name;

});

}

void Warehouse::addPart(const SparePart& part)

{

parts.push\_back(part);

}

void Warehouse::displayParts() const

{

if (parts.empty())

{

cout << "The warehouse is empty." << endl;

return;

}

cout << string(65, '~') << endl;

cout << left << "|" << setw(20) << "Name"

<< "|" << setw(20) << "Manufacturer"

<< "|" << setw(10) << "Price"

<< "|" << setw(10) << "Quantity" << "|" << endl;

for (const SparePart& part : parts)

{

part.display();

}

cout << string(65, '~') << endl;

}

void Warehouse::removePart(const string& name)

{

vector<SparePart>::iterator it = findPartByName(name);

string message = "The part \"";

if (it != parts.end())

{

parts.erase(it);

message += name + "\" was removed from the warehouse.";

}

else

{

message += name + "\" was not found.";

}

cout << message << endl;

}

void Warehouse::updatePart(const string& name, double newPrice, int newQuantity)

{

vector<SparePart>::iterator it = findPartByName(name);

string message = "The part \"";

if (it != parts.end())

{

it->setPrice(newPrice);

it->setQuantity(newQuantity);

message += name + "\" was updated.";

}

else

{

message += name + "\" was not found.";

}

cout << message << endl;

}

**main.cpp:**

#include <iostream>

#include "warehouse.h"

using namespace std;

int main()

{

Warehouse warehouse;

warehouse.addPart({"Oil Filter", "Bosch", 250.0, 15});

warehouse.addPart({"Brake Pads", "ATE", 1200.0, 8});

warehouse.addPart({"Spark Plugs", "NGK", 500.0, 25});

cout << "List of spare parts in the warehouse:" << endl;

warehouse.displayParts();

cout << endl << "Removing the part \"Brake Pads\":" << endl;

warehouse.removePart("Brake Pads");

cout << endl << "Updating the part \"Spark Plugs\":" << endl;

warehouse.updatePart("Spark Plugs", 550.0, 30);

cout << endl << "Updated list of spare parts in the warehouse:" << endl;

warehouse.displayParts();

return 0;

}

**Висновок:**

У цьому завданні я ознайомився з контейнерними класами STL мови C++ та використав їх для створення системи управління складом автозапчастин. Реалізовано функції додавання, видалення, оновлення запчастин, виведення списків і пошуку. Розширив програму, додавши сортування та підрахунок загальної вартості запасів. Це допомогло засвоїти ефективну роботу з контейнерами, такими як vector і map, у реальних задачах.