main.cpp

/\*

Определите базовый класс для электронных компонентов,

заданных номенклатурным номером,

названием (например: сопротивление, конденсатор и т.д.),

значением (например: 50 100 и т.д.)

и единицей измерения (Ом, Н и т.д.)

и производный класс для конкретного электронного компонента,

заданный страной-производителем и ценой.

Классы включают конструкторы, деструкторы и функции вывода членов.

Предметы должны добавляться и удаляться.

Найдите конденсаторы, значение которых находятся в заданных пределах,

и отобразите информацию об этих объектах.

Переопределите операцию <<, которая будет использоваться для отображения данных.

Данные должны сохраняться в файл.

\*/

#include "specificElectronicComponent.h"

#include <vector>

#include <fstream>

#include <Windows.h>

using namespace std;

void menu()

{

cout << "1) Добавить компоненту\n"

<< "2) Удалить компоненту\n"

<< "3) Показать все компоненты\n"

<< "4) Вывести компоненты в файл\n"

<< "5) Конденсаторы, значение которых находятся в заданных пределах\n"

<< "6) Выход\n";

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

int command = 0;

vector<electronicСomponent\*> vectorElectronicСomponent;

ifstream ifs("input.txt");

specificElectronicComponent tmp;

while (!ifs.eof())

{

ifs >> tmp;

vectorElectronicСomponent.push\_back(new specificElectronicComponent(tmp));

}

do

{

menu();

cout << "> ";

cin >> command;

switch (command)

{

case 1:

{

long long stockNumber;

string name;

long long value;

string unit;

string manufacturerCountry;

double price;

cout << "Введите номенклатурный номер: ";

cin >> stockNumber;

cout << "Введите название: ";

cin >> name;

cout << "Введите значение: ";

cin >> value;

cout << "Введите единицу измерения: ";

cin >> unit;

cout << "Введите страну производитель: ";

cin >> manufacturerCountry;

cout << "Введите цену: ";

cin >> price;

vectorElectronicСomponent.push\_back(new specificElectronicComponent(stockNumber, name, value, unit,

manufacturerCountry, price));

break;

}

case 2:

{

int number;

cout << "Введите номер элемента который надо удалить: ";

cin >> number;

if (number < vectorElectronicСomponent.size())

{

vectorElectronicСomponent.erase(vectorElectronicСomponent.begin() + number);

}

else

{

cout << "Номер больше, чем количество элементов!\n";

}

break;

}

case 3:

{

int size = vectorElectronicСomponent.size();

for (int i = 0; i < size; i++)

{

std::cout << \*dynamic\_cast<specificElectronicComponent\*>(vectorElectronicСomponent[i]) << '\n';

}

break;

}

case 4:

{

ofstream ofs("out.txt");

int size = vectorElectronicСomponent.size();

for (int i = 0; i < size; i++)

{

ofs << vectorElectronicСomponent[i] << '\n';

}

break;

}

case 5:

{

int size = vectorElectronicСomponent.size();

long long leftBorder, rightBorder;

cout << "Введите левую границу: ";

cin >> leftBorder;

cout << "Введите правую границу: ";

cin >> rightBorder;

if (leftBorder > rightBorder)

{

cout << "Левая граница больше, чем правая!\n";

}

for (int i = 0; i < size; i++)

{

if (vectorElectronicСomponent[i]->get\_name() == "конденсатор" &&

vectorElectronicСomponent[i]->get\_value() > leftBorder &&

vectorElectronicСomponent[i]->get\_value() < rightBorder)

{

cout << \*dynamic\_cast<specificElectronicComponent\*>(vectorElectronicСomponent[i]) << '\n';

}

}

break;

}

default:

{

cout << "Команда введена неправильно! Попробуйте ещё раз!\n";

break;

}

}

} while (command != 6);

ofstream ofs("out.txt");

int size = vectorElectronicСomponent.size();

for (int i = 0; i < size; i++)

{

ofs << vectorElectronicСomponent[i] << '\n';

}

}

electronicСomponent.h

#pragma once

#include <string>

class electronicСomponent

{

public:

electronicСomponent();

electronicСomponent(const long long& a\_stockNumber,

const std::string& name,

const long long& value,

const std::string& unit);

electronicСomponent(const electronicСomponent& e);

virtual ~electronicСomponent() = 0;

std::string get\_name();

long long get\_value();

protected:

// номенклатурным номером

long long stockNumber;

// название

std::string name;

// значение

long long value;

// единица измерения

std::string unit;

};

electronicСomponent.cpp

#include "electronicСomponent.h"

electronicСomponent::electronicСomponent()

{

stockNumber = 0;

value = 0;

}

electronicСomponent::electronicСomponent(const long long& a\_stockNumber,

const std::string& a\_name,

const long long& a\_value,

const std::string& a\_unit)

:stockNumber(a\_stockNumber), name(a\_name), value(a\_value), unit(a\_unit) {}

electronicСomponent::electronicСomponent(const electronicСomponent& e)

:stockNumber(e.stockNumber), name(e.name), value(e.value), unit(e.unit) {}

electronicСomponent::~electronicСomponent() {}

std::string electronicСomponent::get\_name()

{

return name;

}

long long electronicСomponent::get\_value()

{

return value;

}

specificElectronicComponent.h

#pragma once

#include <iostream>

#include "electronicСomponent.h"

class specificElectronicComponent :

public electronicСomponent

{

public:

specificElectronicComponent();

specificElectronicComponent(const long long& a\_stockNumber,

const std::string& a\_name,

const long long& a\_value,

const std::string& a\_unit,

const std::string& a\_manufacturerCountry,

const double& a\_price);

specificElectronicComponent(const specificElectronicComponent& s);

~specificElectronicComponent() override {}

private:

// страна производитель

std::string manufacturerCountry;

// цена

double price;

friend std::ostream& operator<<(std::ostream& out, const specificElectronicComponent& s);

friend std::istream& operator>>(std::istream& is, specificElectronicComponent& s);

};

specificElectronicComponent.cpp  
  
#include "specificElectronicComponent.h"

std::ostream& operator<<(std::ostream& out, const specificElectronicComponent& s)

{

out << s.stockNumber << ' '

<< s.name << ' '

<< s.value << ' '

<< s.unit << ' '

<< s.manufacturerCountry << ' '

<< s.price;

return out;

}

std::istream& operator>>(std::istream& is, specificElectronicComponent& s)

{

is >> s.stockNumber

>> s.name

>> s.value

>> s.unit

>> s.manufacturerCountry

>> s.price;

return is;

}

specificElectronicComponent::specificElectronicComponent()

{

price = 0;

}

specificElectronicComponent::specificElectronicComponent(const long long& a\_stockNumber,

const std::string& a\_name,

const long long& a\_value,

const std::string& a\_unit,

const std::string& a\_manufacturerCountry,

const double& a\_price)

:electronicСomponent(a\_stockNumber, a\_name, a\_value, a\_unit), manufacturerCountry(a\_manufacturerCountry), price(a\_price) {}

specificElectronicComponent::specificElectronicComponent(const specificElectronicComponent& s)

:electronicСomponent(s), manufacturerCountry(s.manufacturerCountry), price(s.price) {}