МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.03 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | САПРиУ | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Информационные технологии и программирование | | |
| Курс | I | | | Группа | 424 |

**Отчёт по контрольной работе № 1**

**Вариант № 21**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 425 |  |  |  | Харисов Ильяс Ренатович |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверил(и): |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |
|  |  |  |  |

СОДЕРЖАНИЕ

[1. Постановка задачи 3](#_Toc129861267)

[2. Исходные данные 3](#_Toc129861268)

[3. Особые ситуации 3](#_Toc129861269)

[4. Математические методы и алгоритмы решения задач 3](#_Toc129861270)

[5. Блок-схемы алгоритмов программы 4](#_Toc129861271)

[6. Форматы представления данных 5](#_Toc129861272)

[7. Структура программы 6](#_Toc129861273)

[8. Описание хода выполнения контрольной работы 9](#_Toc129861274)

[9. Результат работы программы 9](#_Toc129861275)

[10. Исходный текст программы 12](#_Toc129861276)

# 1. Постановка задачи

Необходимо разработать класс Товар: наименование, производитель, цена, срок хранения, количество.

Создать массив объектов. Реализовать возможность получения:

– списка товаров для заданного наименования,

– списка товаров для заданного наименования, цена которых не

превышает указанной,

– списка товаров, срок хранения которых больше заданного.

# 2. Исходные данные

В качестве исходных данных программа использует введенные пользователем и/или считаные из файла количество и данные клиентов.

# 3. Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

* Ввод пользователем некорректного имени файла
* Загрузка некорректных данных с файла (отрицательные поля: цена, количество, срок годности)
* Некорректный ввод значений переменных
* Непредвиденное поведение программы при нехватке памяти

# 4. Математические методы и алгоритмы решения задач

При отборе товаров по заданному наименованию используется метод перебора. Программа определённое количество итераций сравнивает наименование товара, введенное пользователем с наименованиями, хранящимися в массиве объектов, и если находит нужный товар, то записывает объект в отсортированный массив товаров.

При отборе по наименованию и цене, не выше чем указанная пользователем, программа аналогично сравнивает каждое наименование товара, и при этом еще выбирает только те, цена которых не

превышает указанной.

При отборе по сроку хранения, программа действует аналогично предыдущим случаям.

# 5. Блок-схемы алгоритмов программы



Рисунок 1 – Блок-схема алгоритма отбора по заданному наименованию.



Рисунок 2 – Блок-схема алгоритма отбора по заданному наименованию и цене не выше указанной.



Рисунок 3 – Блок-схема алгоритма отбора товаров, срок хранения которых больше заданного.

# 6. Форматы представления данных

Программа использует следующие переменные:

Таблица 1 — Список переменных, используемых в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| variant | int | Пользовательский выбор во всех меню |
| boarMenu | int | Граница допустимых значений пользовательского ввода в меню |
| products | Product\* | Массив, содержащий список товаров |
| sortList | Product\* | Массив, содержащий список товаров, удовлетворяющих условию поиска |
| size | int | Размер массива product |
| sizeSort | int | Размер массива sortList |
| mUI | ui | Экземпляр класса ui, для вызовов методов класса |
| mReader | fileReader | Экземпляр класса fileReader, для вызовов методов класса |
| mTest | test | Экземпляр класса test, для вызовов методов класса |
| name | string | Имя файла, которое задает пользователь |
| myFile | ofstream | Вывод данных в файл |
| input | ifstream | Ввод данных из файла |
| isDataSaved | bool | Проверка на то, сохранились ли данные в файл |
| data | string | Считываемая строка из файла |
| token | string | Считываемые подстроки из файла, нужные для заполнения полей класса |
| way | string | В зависимости от передаваемой строки выводится информация |
| mInput | string | Вывод информации для ввода |
| mOutput | string | Вывод информации для вывода |
| out | ofstream | Вывод данных для тестовых файлов |
| out2 |
| out3 |
| out4 |
| out5 |
| iter | int | Итератор, инициализирует размер отфильтрованного массива |

Для хранения данных о товараз используются объекты класса Product, в котором задаются наименование, производитель, цена, срок хранения и количество товара. Модульные тесты описаны как методы класса test, представлено 5 различных наборов данных, которые проверяют 6 основных алгоритмов программы — ввод данных из файла, заполнение полей класса подстроками, обнуление объектов с отрицательными или нулевыми полями (цена, срок хранения, количество), отбор по заданному наименованию, отбор по заданному наименованию и цене, не выше указанной и отбор товаров срок хранения, которых больше заданного.

# 7. Структура программы

Программа разделена на 6 исходных и 5 заголовочных файлов.

Таблица 2 — Список заголовочных файлов

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Product.h | Содержит объявления класса Client и его методов. |
| fileReaser.h | Содержит объявления класса fileReader и его методов для работы с файлами. |
| sorting.h | Содержит объявления класса sorting и его методов отбора нужных по условию товаров. |
| ui.h | Содержит объявления класса ui и его методов для работы с консольным вводом и выводом. |
| test.h | Содержит объявления класса test и его методов для выполнения модульных тестов. |

Таблица 3 — Список исходных файлов

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| main.cpp | Содержит определения функций запуска программы и механизма работы зацикленных меню |
| fileReader.cpp | Содержит определения функций для работы с файлами |
| ui.cpp | Содержит определения функций для ввода и вывода в консоли |
| sorting.cpp | Содержит определения функций отбора нужных по заданию товаров |
| Product.cpp | Содержит определение функции для инициализации полей класса Product и для получения полей. |
| test.cpp | Содержит определение функций реализации модульных тестов программы |

Таблица 4 – Функции, составляющие программу

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| launchMenu | Реализация внешнего зацикленного меню и 2-х вложенных зацикленных меню. |
| selectInput | Выбор способа ввода данных |
| selectImplTask | Выбор способа отбора товаров в новый список |
| selectOutput | Выбор вывода отобранных товаров |
| printInputWays | Вывод информации по навигации в меню способов ввода данных |
| printAction | Вывод информации по навигации в меню способов отбора данных в список |
| printOutputWays | Вывод информации по навигации в меню способов вывода отобранных товаров |
| outConsole | Реализует вывод отобранных товаров в консоль |
| printProduct | Реализует вывод товаров |
| readConsole | Реализует ввод товаров с консоли |
| filling | Реализует заполнение полей класса переданными данными |
| getDouble | Вызывает функцию getInput требуя вернуть ввод типа double. |
| getInt | Вызывает функцию getInput требуя вернуть ввод типа int. |
| getSize | Возвращает ввод размера массива типа int и зацикливается если ввод неположительный |
| getMenuVar | Возвращает ввод пункта меню типа int и зацикливается если ввод не соответствует пунктам меню |
| getInput | Возвращает ввод числа переданного типа и зацикливается если ввод некорректен |
| fillFile | Реализует заполнение файла данными об отобранных товарах |
| readFile | Реализует чтение и проверку данных из файла в массив. |
| removeNegative | Обнуляет объекты с неположительными полями (цена, срок хранения, количество) |
| implFile | Реализует чтение данных из файла. |
| filling | Реализует заполнения полей класса переданными данными. |
| printInform | Информирует пользователя о формате файла и данных в нем |
| startTest | Запускает функции тестов. |
| testSortBySH | Тестирует функцию sortByShelfLife |
| testSortByNamePrice | Тестирует функцию sortByNamePrice |
| testSortByName | Тестирует функцию sortByName |
| testRemoveNegative | Тестирует функцию removeNegative |
| deleteProductData | Отчищает память после каждого тестирования |
| fillingForTest | Заполняет тестовыми данными поля класса |
| testParsingTXT | Тестирования функции filling |
| testImplFile | Тестирование функции implFile |
| creatingTestFiles | Создание тестовых файлов |
| checkingOutput | Проверяет был ли выведен предупредительный текст о некорректности введенных данных в файле |
| checkingData | Проверяет соответствует ли полученные данные ожидаемым |
| printCorrect | Проверяет отличались ли полученные данные от ожидаемых, если нет, информирует о пройденных тестов. |
| printInvalid | Информирует пользователя об ошибках в модульном тестировании |
| sortByShelfLife | Отбор товаров срок хранения которых, больше заданного |
| sortByNamePrice | Отбор товаров указанного наименования, цена которых меньше заданной |
| sortByName | Отбор товаров по указанному наименованию |
| printInputInfo | Информирования об необходимости ввода |

Таблица 5 – Методы классов, составляющих программу

|  |  |  |
| --- | --- | --- |
| **Имя** | **Класс** | **Описание** |
| getName | Product | Получить наименование товара |
| setName | Product | Устанавливает наименование товара |
| getManufacturer | Product | Получить название производителя товара |
| setManufacturer | Product | Устанавливает название производителя товара |
| getPrice | Product | Получить цену товара |
| setPrice | Product | Устанавливает цену товара |
| getShelfLife | Product | Получить срок хранения товара |
| setShelfLife | Product | Устанавливает срок хранения товара |
| getQuantity | Product | Получить количество товара |
| setQuantity | Product | Устанавливает количество товара |
| isNegavive() | Product | Проверяет является ли поле (цена, срок хранения, количество) неположительным |

# 8. Описание хода выполнения контрольной работы

1. В ходе контрольной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio C++ 2022. В нём был создан проект. Так же проект был размещен на github в публичном репозитории, в котором отслеживались все commit и исправления в проекте.
2. В начальном варианте программы не использовались конструкции try-catch из-за чего функции считывания данных из файла использовали большое количество bool переменных (флагов). С изучением данной конструкции надобность флагов пропала.
3. Для сохранения данных о товаре и работе с ними, был создан класс Product, содержащий методы get, set для каждого private поля класса.
4. В начальном варианте программы были проблемы с чтением размера массива, но все проблемы были исправлены с помощью считывания количества строк в файле.
5. Функции сортировок (отбора) были реализованы без записывания результатов в новый массив товаров. Решение проблемы стало передача указателя на новый массив в функции сортировок.

# 9. Результат работы программы

При выборе соответствующих пунктов меню, в результате, программа выводит список товаров заданного наименования или заданного наименования с ценой, которая ниже чем указанная или товаров срок годности, которых выше чем заданного.

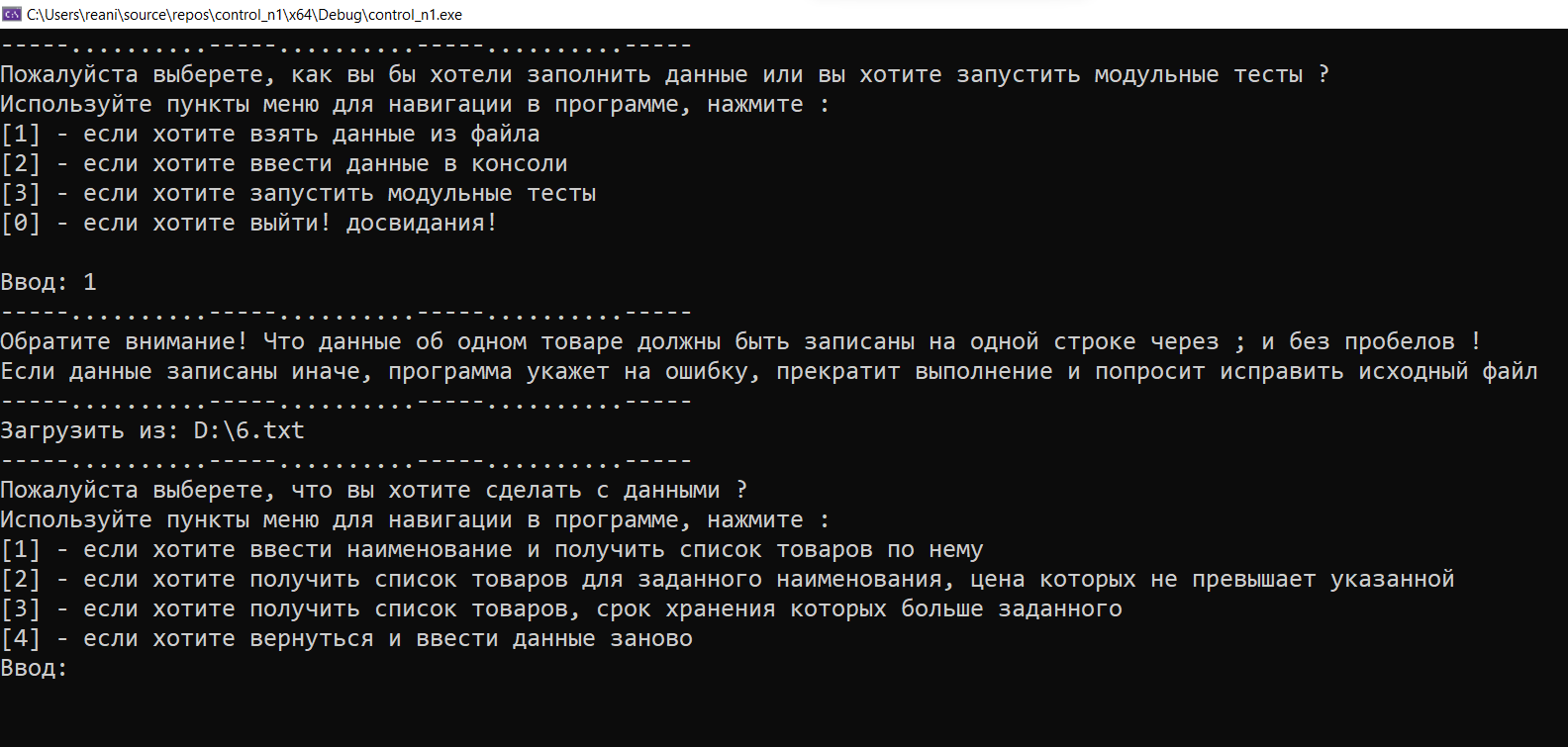


Рисунок 4 – Выбор способа вывода списка

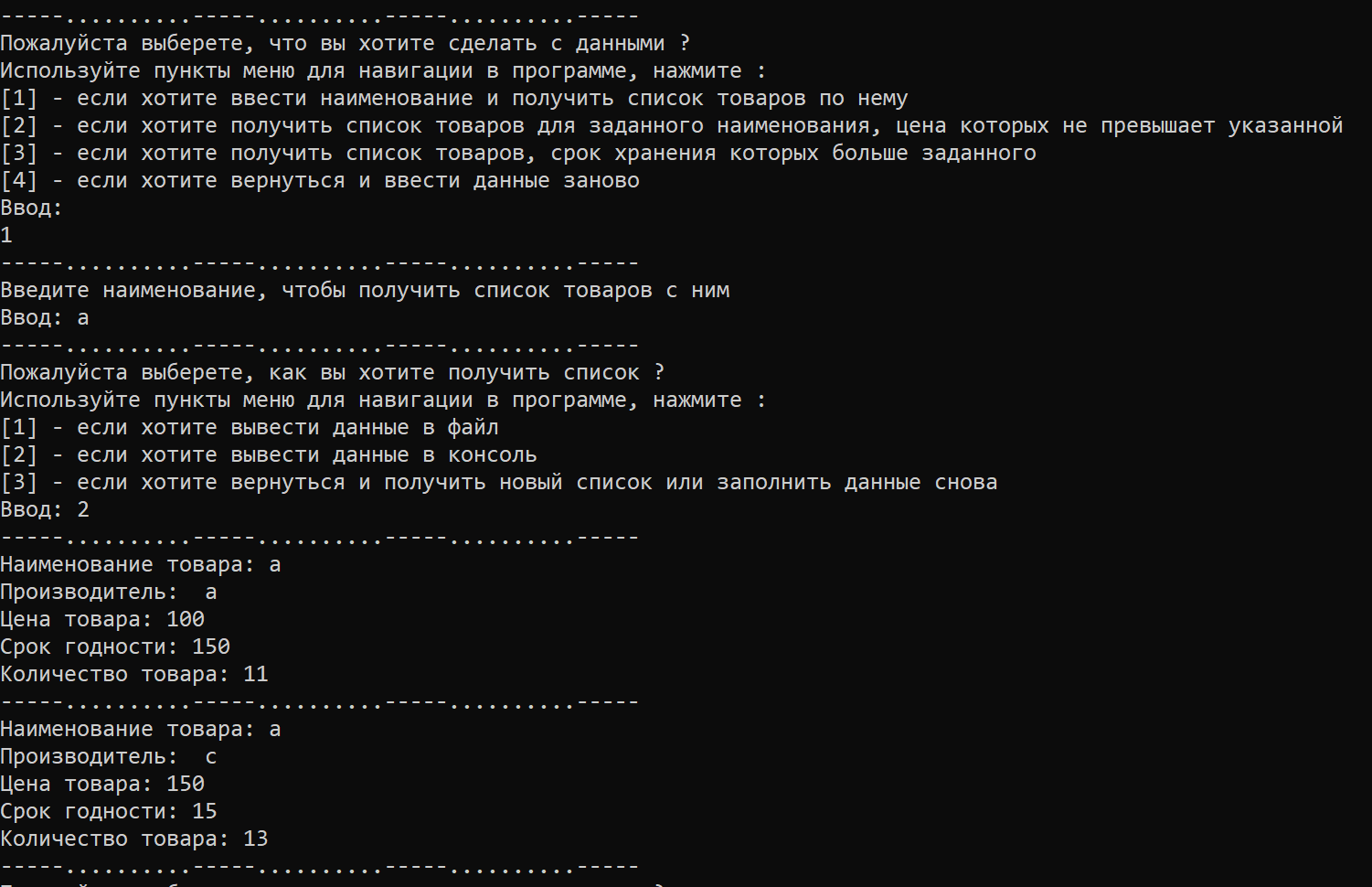


Рисунок 5 – Пример вывода списка c товарами с наименованием ‘a’

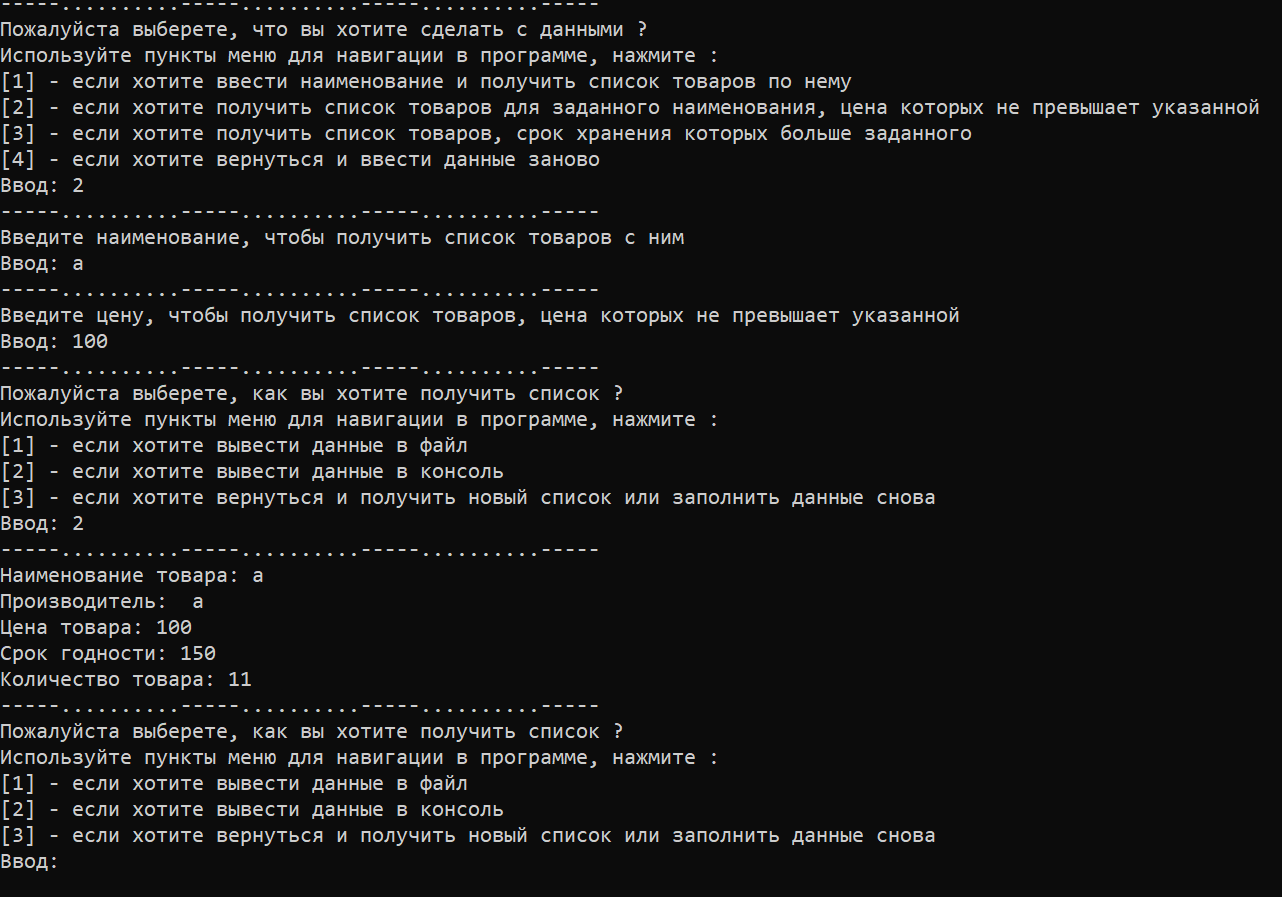


Рисунок 5 – Пример работы функции поиска клиентов в заданном интервале

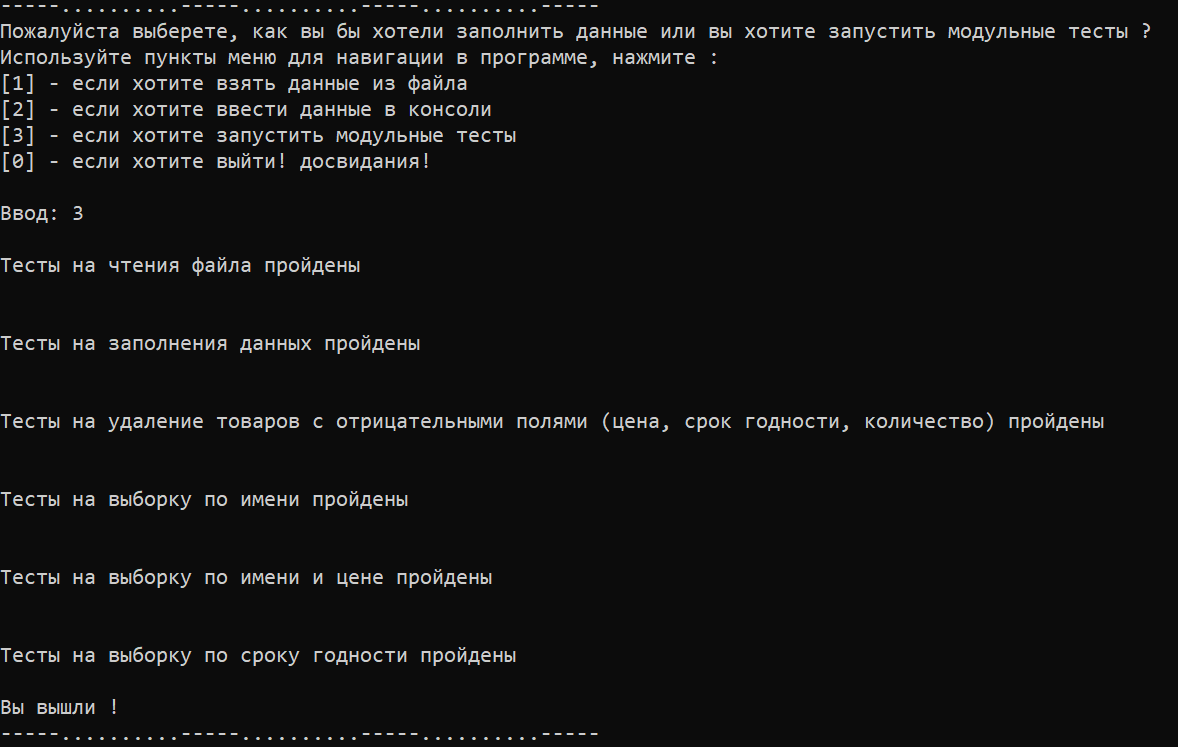


Рисунок 6 – Пример модульных тестов

# 10. Исходный текст программы

**[ Начало программы ]**

**[ Начало файла main.cpp ]**

#include <iostream>

#include "fileReader.h"

#include "Product.h"

#include "ui.h"

#include "sorting.h"

#include "test.h"

using namespace UI;

using namespace reader;

using namespace sorted;

using namespace testUnit;

enum { task1 = 1, task2 = 2, task3 = 3, returnp = 4 };

enum { exitp = 0, rfile = 1, rconsole = 2, unitTest = 3 };

enum { fileOutput = 1, coutput = 2, outreturn = 3};

void printInputWays() {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Пожалуйста выберете, как вы бы хотели заполнить данные или вы хотите запустить модульные тесты ?" << endl;

cout << "Используйте пункты меню для навигации в программе, нажмите :" << endl;

cout << "[1] - если хотите взять данные из файла" << endl;

cout << "[2] - если хотите ввести данные в консоли" << endl;

cout << "[3] - если хотите запустить модульные тесты" << endl;

cout << "[0] - если хотите выйти! досвидания!" << endl << endl;

}

void printAction() {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Пожалуйста выберете, что вы хотите сделать с данными ?" << endl;

cout << "Используйте пункты меню для навигации в программе, нажмите :" << endl;

cout << "[1] - если хотите ввести наименование и получить список товаров по нему" << endl;

cout << "[2] - если хотите получить список товаров для заданного наименования, цена которых не превышает указанной" << endl;

cout << "[3] - если хотите получить список товаров, срок хранения которых больше заданного" << endl;

cout << "[4] - если хотите вернуться и ввести данные заново" << endl;

}

void printOutputWays() {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Пожалуйста выберете, как вы хотите получить список ?" << endl;

cout << "Используйте пункты меню для навигации в программе, нажмите :" << endl;

cout << "[1] - если хотите вывести данные в файл" << endl;

cout << "[2] - если хотите вывести данные в консоль" << endl;

cout << "[3] - если хотите вернуться и получить новый список или заполнить данные снова" << endl;

}

void selectImplTask(int variant, Product\*\* products, Product\*\* sortList, int\* size, int\* sizeSort) {

sorting sort;

switch (variant)

{

case task1:

\*sizeSort = sort.sortByName(products, sortList, \*size, {}, false);

break;

case task2:

\*sizeSort = sort.sortByNamePrice(products, sortList, \*size, {}, 0, false);

break;

case task3:

\*sizeSort = sort.sortByShelfLife(products, sortList, \*size, 0, false);

break;

case returnp:

cout << "Вы вышли !" << endl;

break;

}

}

void selectInput(int variant, Product\*\* products, int\* size) {

ui mUI;

fileReader mReader;

test mTest;

switch (variant)

{

case rfile:

\*size = mReader.readFile(products, {});

break;

case rconsole:

\*size = mUI.readConsole(products);

break;

case unitTest:

mTest.startTest();

case exitp:

cout<<"Вы вышли !"<<endl;

break;

}

}

void selectOutput(int variant, Product\*\* sortList, int size) {

ui mUI;

fileReader mReader;

test mTest;

switch (variant)

{

case fileOutput:

mReader.fillFile(sortList, size);

break;

case coutput:

mUI.outConsole(sortList, size);

break;

case outreturn:

cout<<"Вы вышли !"<<endl;

break;

}

}

void launchMenu() {

ui mUI;

int variant = 0;

int boardMenu = unitTest;

int sizeSort = 0;

int size = 0;

Product\* products = nullptr;

Product\* sortList = nullptr;

do

{

printInputWays();

variant = mUI.getMenuVar(exitp, unitTest);

try {

selectInput(variant, &products, &size);

}

catch (const exception& e) {

continue;

}

if (variant == exitp) break;

if (variant == unitTest) continue;

do {

printAction();

boardMenu = returnp;

variant = mUI.getMenuVar(task1, boardMenu);

try {

selectImplTask(variant, &products, &sortList, &size, &sizeSort);

}

catch (const exception& e) {

continue;

}

if (variant == returnp) break;

do {

printOutputWays();

boardMenu = returnp;

variant = mUI.getMenuVar(fileOutput, boardMenu);

if (variant == outreturn) break;

try {

selectOutput(variant, &sortList, sizeSort);

}

catch (const exception& e) {

continue;

}

} while (variant != returnp);

} while (variant != returnp);

delete[]products;

products = nullptr;

} while (variant != exitp);

}

int main() {

setlocale(LC\_CTYPE, "Russian");

launchMenu();

system("pause");

}

**[ Конец файла main.cpp ]**

**[ Начало файла fileReader.h ]**

#pragma once

#include "Product.h"

using namespace cproduct;

namespace reader {

class fileReader {

public:

int readFile(Product\*\*, string);

void fillFile(Product\*\*, int);

void implFile(Product\*\*, int\*, string);

void filling(Product\*\*, const char\*, int );

void removeNegative(Product\*\*, int\*);

};

**[ Конец файла fileReader.h ]**

**[ Начало файла fileReader.cpp ]**

#include <iostream>

#include <filesystem>

#include "fileReader.h"

#include "ui.h"

#include "product.h"

#include <string>

#include <fstream>

#include <cstring>

using namespace std;

using namespace reader;

using namespace UI;

using namespace cproduct;

static const string mInput = "input";

static const string mOutput = "output";

void printInform(string way) {

if (way == mInput) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Обратите внимание! Что данные об одном товаре должны быть записаны на одной строке через ; и без пробелов !" << endl;

cout << "Если данные записаны иначе, программа укажет на ошибку, прекратит выполнение и попросит исправить исходный файл" << endl;

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Загрузить из: ";

}

if (way == mOutput) {

cout << "Введите название файла в который вы хотите сохранить результат" << endl;

}

}

// Разбиваем строку на подстроки, используя символ ';' в качестве разделителя

void fileReader::filling(Product\*\* products, const char\* str\_ptr, int i) {

int iter = 0;

char\* str\_copy = new char[strlen(str\_ptr) + 1];

strcpy\_s(str\_copy, strlen(str\_ptr) + 1, str\_ptr);

char\* context = nullptr;

char\* token = strtok\_s(str\_copy, ";", &context);

while (token != nullptr) {

switch (iter) {

case 0:

(\*products)[i].setName(token);

break;

case 1:

(\*products)[i].setManufacturer(token);

break;

case 2:

if (strtod(token, nullptr) <= 0) {

cout << "Отрицательная или нулевая цена товара! Исправьте данные в файле! В " << i + 1 << " строке" << endl;

}

(\*products)[i].setPrice(strtod(token, nullptr));

break;

case 3:

if (atoi(token) <= 0) {

cout << "Отрицательный или нулевой срок годности товара! Исправьте данные в файле! В " << i + 1 << " строке"<<endl;

}

(\*products)[i].setShelfLife(atoi(token));

break;

case 4:

if (atoi(token) <= 0) {

cout << "Отрицательное или нулевое количество товара! Исправьте данные в файле! В " << i + 1 << " строке"<<endl;

}

(\*products)[i].setQuantity(atoi(token));

break;

default:

cout << "Слишком много данных в одной строке! Данные будут проигнорированы" << endl;

}

token = strtok\_s(nullptr, ";", &context);

iter++;

}

delete[] str\_copy;

}

void fileReader::implFile(Product\*\* products, int\* size, string name) {

int i = 0;

string data{};

ifstream input;

input.open(name);

if (input.is\_open()) {

while (getline(input, data)) {

\*size += 1;

}

input.close();

}else{

cout << endl << "Файла с таким именем не существует!" << endl << endl;

throw runtime\_error("error");

}

input.open(name);

if (input.is\_open()) {

\*products = new Product[\*size];

while (getline(input, data)) {

if (data.empty()) {

cout << "Ввод товара на " << i+1 << " строке не был произведен! Исправьте файл и введите данные заново" << endl;

\*size -= 1;

}

else {

filling(products, data.c\_str(), i);

}

i++;

}

input.close();

}

else {

cout << endl << "Ошибка чтения файла или выделения памяти!" << endl << endl;

throw runtime\_error("An error occurred");

}

}

void fileReader::removeNegative(Product\*\* products, int\* size) {

int newSize = 0;

for (int i = 0; i < \*size; i++) {

if (!(\*products)[i].isNegavive()) {

(\*products)[newSize++] = (\*products)[i];

}

}

// обнуляем плохие элементы

for (int i = newSize; i < \*size; i++) {

(\*products)[i] = Product();

}

\*size = newSize;

}

int fileReader::readFile(Product\*\* products, string name) {

int size = 0;

printInform(mInput);

cin >> name;

implFile(products, &size, name);

removeNegative(products, &size);

return size;

}

void fileReader::fillFile(Product\*\* products, int size) {

printInform(mOutput);

string filePath = "";

bool isDataSaved = false;

ui mUI;

do {

cout << "Сохранить в:";

getline(cin, filePath);

if (ifstream(filePath)) {

cout << "Файл уже существует." << endl;

cout << "[0] - Перезаписать существующий файл." << endl;

cout << "[1] - Повторить ввод." << endl;

int tryAnotherFile = mUI.getMenuVar(0, 1);

if (tryAnotherFile) {

continue;

}

}

ofstream myFile(filePath, ofstream::app);

error\_code ec{};

if (!myFile) {

cout << "Запись запрещена. Повторите ввод." << endl;

myFile.close();

continue;

}

myFile.close();

myFile.open(filePath, ofstream::trunc);

myFile << "Наименование\tПроизводитель\tЦена\t\tСрок годности (в днях)\tКоличество\n";

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

myFile << (\*products)[i].getName() << "\t\t|\t" << (\*products)[i].getManufacturer() << "\t\t|\t" << (\*products)[i].getPrice()

<< "\t\t|\t" << (\*products)[i].getShelfLife() << "\t\t|\t" << (\*products)[i].getQuantity() << endl;

}

myFile.close();

cout << "Запись завершена." << endl;

isDataSaved = true;

} while (!isDataSaved);

}

**[ Конец файла fileReader.cpp ]**

**[ Начало файла Product.h ]**

#pragma once

#include <string>

using namespace std;

namespace cproduct {

class Product

{

public:

string getName();

void setName(string);

string getManufacturer();

void setManufacturer(string);

double getPrice();

void setPrice(double);

int getShelfLife();

void setShelfLife(int);

int getQuantity();

void setQuantity(int);

bool isNegavive() const {

return (price <= 0.0 || quantity <= 0 || shelfLife <= 0);

}

private:

string name{};

string manufacturer{};

double price{};

int shelfLife{};

int quantity{};

};

}

**[ Конец файла Product.h ]**

**[ Начало файла Product.cpp ]**

#include "Product.h"

using namespace cproduct;

string Product::getName() {

return Product::name;

}

void Product::setName(string mName) {

Product::name = mName;

}

string Product::getManufacturer() {

return Product::manufacturer;

}

void Product::setManufacturer(string mManufacturer) {

Product::manufacturer = mManufacturer;

}

double Product::getPrice() {

return Product::price;

}

void Product::setPrice(double mPrice) {

Product::price = mPrice;

}

int Product::getShelfLife() {

return Product::shelfLife;

}

void Product::setShelfLife(int mShelfLife) {

Product::shelfLife = mShelfLife;

}

int Product::getQuantity() {

return Product::quantity;

}

void Product::setQuantity(int mQuantity) {

Product::quantity = mQuantity;

}

**[ Конец файла Product.cpp ]**

**[ Начало файла sorting.h ]**

#pragma once

namespace sorted {

class sorting {

public:

int sortByName(Product\*\*, Product\*\*, int, string, bool);

int sortByNamePrice(Product\*\*, Product\*\*, int, string, double, bool);

int sortByShelfLife(Product\*\*, Product\*\*, int, int, bool);

};

}

**[ Конец файла sorting.h ]**

**[ Начало файла sorting.cpp ]**

#include <iostream>

#include "ui.h"

#include "product.h"

#include "sorting.h"

#include <string>

#include <cstring>

using namespace std;

using namespace UI;

using namespace cproduct;

using namespace sorted;

static string byName = "inputName";

static string byPrice = "inputPrice";

static string byShelfLife = "inputSL";

void printInputInfo(string way) {

if (way == byName) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите наименование, чтобы получить список товаров с ним" << endl;

cout << "Ввод: ";

}

if (way == byPrice) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите цену, чтобы получить список товаров, цена которых не превышает указанной" << endl;

}

if (way == byShelfLife) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите срок хранения товара, чтобы получить спискок товаров, срок хранения которых больше заданного" << endl;

}

}

int sorting::sortByName(Product\*\* products, Product\*\* sortList, int size, string nameSearch, bool isTest) {

int i = 0;

int iter = 0;

\*sortList = new Product[size];

if(!isTest){

printInputInfo(byName);

cin >> nameSearch;

}

for (; i < size; i++) {

if ((\*products)[i].getName() == nameSearch) {

(\*sortList)[iter] = (\*products)[i];

iter++;

}

}

return iter;

}

int sorting::sortByNamePrice(Product\*\* products, Product\*\* sortList, int size, string nameSearch, double priceSearch, bool isTest) {

ui mUI;

int i = 0;

int iter = 0;

\*sortList = new Product[size];

if(!isTest){

printInputInfo(byName);

cin >> nameSearch;

printInputInfo(byPrice);

priceSearch = mUI.getDouble();

}

for (; i < size; i++) {

if ((\*products)[i].getName() == nameSearch && (\*products)[i].getPrice() <= priceSearch) {

(\*sortList)[iter] = (\*products)[i];

iter++;

}

}

return iter;

}

int sorting::sortByShelfLife(Product\*\* products, Product\*\* sortList, int size, int sh, bool isTest) {

ui mUI;

\*sortList = new Product[size];

int i = 0;

int iter = 0;

if(!isTest){

printInputInfo(byShelfLife);

sh = mUI.getInt();

}

for ( ; i < size; i++) {

if ((\*products)[i].getShelfLife() > sh) {

(\*sortList)[iter] = (\*products)[i];

iter++;

}

}

return iter;

}

**[ Конец файла sorting.cpp ]**

**[ Начало файла test.h ]**

#pragma once

#include "Product.h"

using namespace cproduct;

namespace testUnit {

class test {

public:

void startTest();

};

}

**[ Конец файла test.h ]**

**[ Начало файла test.cpp ]**

#include <iostream>

#include <cassert>

#include <fstream>

#include <sstream>

#include "test.h"

#include "fileReader.h"

#include "sorting.h"

using namespace testUnit;

using namespace sorted;

using namespace reader;

using namespace std;

const string testParsing = "parsing";

const string testFileRead = "read";

template<typename T, typename U>

void printInvalid(T received, U expected, string message) {

cout << message;

cout << "Ожидалось: " << expected << endl << "Было получено: " << received << endl;

}

void printCorrect(void (\*function)(), string message) {

stringstream buffer{};

streambuf\* old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

(\*function)();

cout.rdbuf(old\_cout\_buf);

string output = buffer.str();

if (output.empty()) {

cout << endl << message << endl << endl;

}

else {

cout << output;

}

}

void checkingData(Product\* products, int iter,

string name, string manufacturer, double price, int sh, int q) {

string message = {};

if (products[iter].getName() != name)

printInvalid(products[0].getName(), name, message);

if (products[iter].getManufacturer() != manufacturer)

printInvalid(products[0].getManufacturer(), manufacturer, message);

if (products[iter].getPrice() != price)

printInvalid(products[0].getPrice(), price, message);

if (products[iter].getShelfLife() != sh)

printInvalid(products[0].getShelfLife(), sh, message);

if (products[iter].getQuantity() != q)

printInvalid(products[0].getQuantity(), q, message);

}

string checkingOutput(Product\*\* products, const char\* str, int i) {

fileReader mReader;

stringstream buffer{};

streambuf\* old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

mReader.filling(products, str, i);

cout.rdbuf(old\_cout\_buf);

return buffer.str();

}

void creatingTestFiles() {

ofstream out("test1.txt");

out << "milk;Nestle;3.5;10;100" << endl;

out << "bread;Dobryanka;3;7;50" << endl;

out.close();

ofstream out2("test2.txt");

out2 << "" << endl;

out2 << "" << endl;

out2.close();

ofstream out3("test3.txt");

out3 << "milk;Nestle;3.5;10;100;Extra" << endl;

out3.close();

ofstream out4("test4.txt");

out4 << "peach;Google;9.2;11;111;Extra;Extra2" << endl;

out4 << "bread;Dobryanka;-2;7;50" << endl;

out4 << "" << endl;

out4.close();

ofstream out5("test5.txt");

out5 << "peach;Google;009.2;11;111;" << endl;

out5 << "bread;Dobryanka;02;7;50" << endl;

out5 << "breadik;Dobr;002;7;50" << endl;

out5 << "milk;Nestle;3.5;10;-100;Extra" << endl;

out5 << "" << endl;

out5.close();

}

void testImplFile() {

int size = 0;

string myOutput = {};

string message = {};

fileReader reader;

Product\* products = nullptr;

string outputInputError = "Ввод товара на 1 строке не был произведен! Исправьте файл и введите данные заново\nВвод товара на 2 строке не был произведен! Исправьте файл и введите данные заново\n";

string outputDataError = "Слишком много данных в одной строке! Данные будут проигнорированы\n";

string outputNegativePrice = "Отрицательная или нулевая цена товара! Исправьте данные в файле! В 2 строке\n";

string outputNegativeQuantity = "Отрицательное или нулевое количество товара! Исправьте данные в файле! В 4 строке\n";

string outputInputError3 = "Ввод товара на 3 строке не был произведен! Исправьте файл и введите данные заново\n";

string outputInputError5 = "Ввод товара на 5 строке не был произведен! Исправьте файл и введите данные заново\n";

// Тест с корректными данными

stringstream buffer{};

streambuf\* old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(&products, &size, "test1.txt");

cout.rdbuf(old\_cout\_buf);

string output = buffer.str();

if(size != 2) printInvalid(size, 2, message);

checkingData(products, 0, "milk", "Nestle", 3.5, 10, 100);

checkingData(products, 1, "bread", "Dobryanka", 3, 7, 50);

// Тест с 0 количеством товаров

products = nullptr;

size = 0;

buffer = {};

old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(&products, &size, "test2.txt");

cout.rdbuf(old\_cout\_buf);

output = buffer.str();

if (size != 0) printInvalid(size, 0, message);

if (output != outputInputError) printInvalid(output, outputInputError, message);

// Тест с некорректным форматом данных

products = nullptr;

size = 0;

buffer = {};

old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(&products, &size, "test3.txt");

cout.rdbuf(old\_cout\_buf);

output = buffer.str();

if(output != outputDataError) printInvalid(output, outputDataError, message);

if(size != 1) printInvalid(size, 1, message);

// Тест с отрицательной ценой и с некорректным форматом данных

products = nullptr;

size = 0;

buffer = {};

old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(&products, &size, "test4.txt");

cout.rdbuf(old\_cout\_buf);

output = buffer.str();

if(output != outputDataError + outputDataError + outputNegativePrice + outputInputError3)

printInvalid(output, outputDataError + outputDataError + outputNegativePrice + outputInputError3, message);

if(size != 2) printInvalid(size, 2, message);

// Тест с отрицательным количеством и пустыми строчками

products = nullptr;

size = 0;

buffer = {};

old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(&products, &size, "test5.txt");

cout.rdbuf(old\_cout\_buf);

output = buffer.str();

if(output != outputNegativeQuantity + outputDataError + outputInputError5)

printInvalid(output, outputNegativeQuantity + outputDataError + outputInputError5, message);

if(size != 4) printInvalid(size, 4, message);

}

void testParsingTXT() {

fileReader fr;

string output = {};

string myOutput = {};

string message = {};

const char\* str1 = "Milk;Manufacturer;1.99;7;10";

const char\* str2 = "Bread;Manufacturer;0.99;7;10";

const char\* str3 = "Sugar;Manufacturer;2.49;30;5";

const char\* str4 = "Juice;Manufacturer;-1.5;1;5";

const char\* str5 = "Salt;Manufacturer;-0.49;0;-2";

string outputNegativePrice4 = "Отрицательная или нулевая цена товара! Исправьте данные в файле! В 4 строке\n";

string outputNegativePrice5 = "Отрицательная или нулевая цена товара! Исправьте данные в файле! В 5 строке\n";

string outputNegativeSH5 = "Отрицательный или нулевой срок годности товара! Исправьте данные в файле! В 5 строке\n";

string outputNegativeQuantity5 = "Отрицательное или нулевое количество товара! Исправьте данные в файле! В 5 строке\n";

Product\* products = new Product[5];

fr.filling(&products, str1, 0);

checkingData(products, 0, "Milk", "Manufacturer", 1.99, 7, 10);

fr.filling(&products, str2, 1);

checkingData(products, 1, "Bread", "Manufacturer", 0.99, 7, 10);

fr.filling(&products, str3, 2);

checkingData(products, 2, "Sugar", "Manufacturer", 2.49, 30, 5);

// Тесты на обработку некорректных данных

//TODO: надо сделать проверку на вывод

myOutput = outputNegativePrice4;

output = checkingOutput(&products, str4, 3);

if (output != myOutput) printInvalid(output, myOutput, message);

checkingData(products, 3, "Juice", "Manufacturer", - 1.5, 1, 5);

myOutput = outputNegativePrice5 + outputNegativeSH5 + outputNegativeQuantity5;

output = checkingOutput(&products, str5, 4);

if (output != myOutput) printInvalid(output, myOutput, message);

checkingData(products, 4, "Salt", "Manufacturer", -0.49, 0, -2);

delete[] products;

products = nullptr;

}

void fillingForTest(Product\*\* products, int iter, double price, int sh, int q){

(\*products)[iter].setName("test name");

(\*products)[iter].setManufacturer("test manufacturer");

(\*products)[iter].setPrice(price);

(\*products)[iter].setShelfLife(sh);

(\*products)[iter].setQuantity(q);

}

void deleteProductData(Product\*\* products) {

delete[] \*products;

\*products = nullptr;

}

void testRemoveNegative() {

fileReader fr;

Product\* products = new Product[3];

string message{};

int size = 3;

fillingForTest(&products, 0, 9.2, 11, 111);

fillingForTest(&products, 1, 1, 11, 1411);

fillingForTest(&products, 2, 4, 141, 111);

//тест на 1 продукт с <= 0 полем

fr.removeNegative(&products, &size);

if (size != 3) printInvalid(size, 3, message);

deleteProductData(&products);

size = 3;

products = new Product[3];

fillingForTest(&products, 0, -9.2, 011, 0111);

fillingForTest(&products, 1, 0, 0, 0);

fillingForTest(&products, 2, 4, 141, 111);

//тест на 1 продукт с > 0 полями

fr.removeNegative(&products, &size);

if (size != 1) printInvalid(size, 1, message);

deleteProductData(&products);

size = 3;

products = new Product[3];

fillingForTest(&products, 0, -9.2, -11, -111);

fillingForTest(&products, 1, -91, -11, -071411);

fillingForTest(&products, 2, -8.4, -1041, -111);

//тест на все продукты с <= 0 полями

fr.removeNegative(&products, &size);

if (size != 0) printInvalid(size, 0, message);

deleteProductData(&products);

}

void testSortByName() {

int result = 0;

int size = 5;

sorting sort;

Product\* products = new Product[5];

Product\* sortList = new Product[5];

string message = {};

products[0].setName("Milk");

products[0].setManufacturer("Looped Menu Milk Fabrik");

products[0].setPrice(1.35);

products[0].setShelfLife(1000);

products[0].setQuantity(1024);

products[1].setName("milk");

products[1].setManufacturer("Google Milk Fabrik");

products[1].setPrice(8.996);

products[1].setShelfLife(10);

products[1].setQuantity(1);

products[2].setName("LIPSIHA");

products[2].setManufacturer("instasamka");

products[2].setPrice(1.345);

products[2].setShelfLife(10);

products[2].setQuantity(14);

products[3].setName("Milk");

products[3].setManufacturer("Fabrik Milkis");

products[3].setPrice(1.35);

products[3].setShelfLife(1000);

products[3].setQuantity(1024);

products[4].setName("Milk");

products[4].setManufacturer("Dora goblin fabtik");

products[4].setPrice(1.35);

products[4].setShelfLife(1000);

products[4].setQuantity(1024);

result = sort.sortByName(&products, &sortList, size, "Milk", true);

if (result != 3) printInvalid(result, 3, message);

deleteProductData(&products);

}

void testSortByNamePrice() {

int result = 0;

int size = 5;

sorting sort;

Product\* products = new Product[5];

Product\* sortList = new Product[5];

string message = {};

products[0].setName("Bread");

products[0].setManufacturer("Looped Menu Milk Fabrik");

products[0].setPrice(1.99);

products[0].setShelfLife(1000);

products[0].setQuantity(1024);

products[1].setName("milk");

products[1].setManufacturer("Google Milk Fabrik");

products[1].setPrice(8.996);

products[1].setShelfLife(10);

products[1].setQuantity(1);

products[2].setName("Bread");

products[2].setManufacturer("instasamka");

products[2].setPrice(1.345);

products[2].setShelfLife(10);

products[2].setQuantity(14);

products[3].setName("Milk");

products[3].setManufacturer("Fabrik Milkis");

products[3].setPrice(1.25);

products[3].setShelfLife(1000);

products[3].setQuantity(1024);

products[4].setName("Bread");

products[4].setManufacturer("Dora goblin fabtik");

products[4].setPrice(1.36);

products[4].setShelfLife(1000);

products[4].setQuantity(1024);

result = sort.sortByNamePrice(&products, &sortList, size, "Bread", 1.35, true);

if (result != 1) printInvalid(result, 1, message);

deleteProductData(&products);

}

void testSortBySH() {

int result = 0;

int size = 5;

sorting sort;

Product\* products = new Product[5];

Product\* sortList = new Product[5];

string message = {};

products[0].setName("Milk");

products[0].setManufacturer("Looped Menu Milk Fabrik");

products[0].setPrice(1.35);

products[0].setShelfLife(1004);

products[0].setQuantity(1024);

products[1].setName("milk");

products[1].setManufacturer("Google Milk Fabrik");

products[1].setPrice(8.996);

products[1].setShelfLife(10);

products[1].setQuantity(1);

products[2].setName("LIPSI HA");

products[2].setManufacturer("instasamka");

products[2].setPrice(1.345);

products[2].setShelfLife(10);

products[2].setQuantity(14);

products[3].setName("Milk");

products[3].setManufacturer("Fabrik Milkis");

products[3].setPrice(1.35);

products[3].setShelfLife(999);

products[3].setQuantity(1024);

products[4].setName("Milk");

products[4].setManufacturer("Dora goblin fabtik");

products[4].setPrice(1.35);

products[4].setShelfLife(1009);

products[4].setQuantity(1024);

result = sort.sortByShelfLife(&products, &sortList, size, 1000, true);

if (result != 2) printInvalid(result, 2, message);

deleteProductData(&products);

}

void test::startTest() {

creatingTestFiles();

printCorrect(&testImplFile, "Тесты на чтения файла пройдены");

printCorrect(&testParsingTXT, "Тесты на заполнения данных пройдены");

printCorrect(&testRemoveNegative, "Тесты на удаление товаров с отрицательными полями (цена, срок годности, количество) пройдены");

printCorrect(&testSortByName, "Тесты на выборку по имени пройдены");

printCorrect(&testSortByNamePrice, "Тесты на выборку по имени и цене пройдены");

printCorrect(&testSortBySH, "Тесты на выборку по сроку годности пройдены");

}

**[ Конец файла test.cpp ]**

**[ Начало файла ui.h ]**

#pragma once

#include "Product.h"

using namespace cproduct;

namespace UI {

class ui

{

public:

int getInt();

int getMenuVar(int, int);

int getSize();

double getDouble();

int readConsole(Product\*\*);

void outConsole(Product\*\*, int);

};

}

**[ Конец файла ui.h ]**

**[ Начало файла ui.cpp ]**

#include "ui.h"

#include "Product.h"

#include <iostream>

using namespace UI;

using namespace std;

using namespace cproduct;

template<typename T>

T getInput() {

T userInput = 0;

cout << "Ввод: ";

cin >> userInput;

if (cin.fail()) {

cout << "Некоректный ввод. Попробуйте еще раз" << endl;

while (cin.fail()) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

cout << "Ввод: ";

cin >> userInput;

}

}

cin.ignore(INT\_MAX, '\n');

return userInput;

}

int ui::getMenuVar(int lower, int upper) {

bool inputCorrect = false;

int in = 0;

while (!inputCorrect) {

in = getInt();

if (in >= lower && in <= upper) {

inputCorrect = true;

}

else {

cout << "Нет такого пункта меню!" << endl;

}

}

return in;

}

int ui::getSize() {

bool inputCorrect = false;

int in = 0;

while (!inputCorrect) {

in = getInt();

if (in >= 1) {

inputCorrect = true;

}

else {

cout << "Введен отрицательный размер! Попробуйте еще раз" << endl;

}

}

return in;

}

int ui::getInt() {

return getInput<int>();

}

double ui::getDouble() {

return getInput<double>();

}

void filling(Product\*\* products, int i) {

ui mUI;

string name;

string manufacturer;

double price = 0;

int shelfLife = 0;

int quantity = 0;

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите название продукта" << endl;

cout << "Ввод: ";

cin >> name;

(\*products)[i].setName(name);

cout << "Введите производителя продукта" << endl;

cout << "Ввод: ";

cin >> manufacturer;

(\*products)[i].setManufacturer(manufacturer);

cout << "Введите цену продукта" << endl;

price = mUI.getDouble();

(\*products)[i].setPrice(price);

cout << "Введите срок годности продукта" << endl;

shelfLife = mUI.getInt();

(\*products)[i].setShelfLife(shelfLife);

cout << "Введите количество продуктов" << endl;

quantity = mUI.getInt();

(\*products)[i].setQuantity(quantity);

}

int ui::readConsole(Product\*\* products) {

ui mUI;

int size = 0;

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите количество продуктов" << endl;

size = mUI.getSize();

cout << "-----..........-----..........-----..........-----" << endl;

\*products = new Product[size];

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

cout << "Вы вводите товар №" << i+1 << endl;

filling(products, i);

cout << "-----..........-----..........-----..........-----" << endl;

}

return size;

}

void printProduct(Product\*\* products, int i) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Наименование товара: " << (\*products)[i].getName() << endl;

cout << "Производитель: " << (\*products)[i].getManufacturer() << endl;

cout << "Цена товара: " << (\*products)[i].getPrice() << endl;

cout << "Срок годности: " << (\*products)[i].getShelfLife() << endl;

cout << "Количество товара: " << (\*products)[i].getQuantity() << endl;

}

void ui::outConsole(Product\*\* products, int size) {

if (size != 0 ) {

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

printProduct(products, i);

}

}

else {

cout << "Попробуйте ввести данные снова" << endl <<endl;

throw runtime\_error("error");

}

}

**[ Конец файла ui.cpp ]**