Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

КАЛЬКУЛЯЦИЯ И СПИСАНИЕ ПРОДУКТОВ

БГУИР КП 1-40 02 01 117 ПЗ

Студент: гр. 150501 Пащенко М. Д.

Руководитель: старший преподаватель

каф. ЭВМ Ковальчук А. М.

МИНСК 2022

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc121981321)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc121981322)

[2 ОБЗОР ЛИТЕРАТУРЫ 5](#_Toc121981323)

[2.1 Обзор методов и алгоритмов решения поставленной задачи 5](#_Toc121981323)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 6](#_Toc121981324)

[3.1 Структура входных и выходных данных 6](#_Toc121981324)

[3.2 Разработка диаграммы классов 8](#_Toc121981324)

[3.3 Описание классов 8](#_Toc121981324)

[4 РАЗРАБОТКА ПРОГРАМНЫХ МОДУЛЕЙ 22](#_Toc121981325)

[4.1 Разработка схем алгоритмов 22](#_Toc121981325)

[4.2 Разработка алгоритмов 22](#_Toc121981325)

[5 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 25](#_Toc121981326)

[ЗАКЛЮЧЕНИЕ 28](#_Toc121981327)

[ЛИТЕРАТУРА 29](#_Toc121981328)

[ПРИЛОЖЕНИЕ А 30](#_Toc121981329)

[ПРИЛОЖЕНИЕ Б 31](#_Toc121981330)

[ПРИЛОЖЕНИЕ В 32](#_Toc121981331)

[ПРИЛОЖЕНИЕ Г 33](#_Toc121981332)

[ПРИЛОЖЕНИЕ Д 34](#_Toc121981333)

# ВВЕДЕНИЕ

Объектно-ориентированное программирование представляет собой технологию программирования, которая базируется на классификации и абстракции объектов. Одним из наиболее популярных средств объектно-ориентированного программирования, позволяющим разрабатывать программы, эффективные по объёму кода и скорости выполнения является С++.

Язык С++ реализует три основные концепции объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. В результате использования инкапсуляции, программа, написанные на данном языке, обладает повышенной защищённостью объектов от влияния на них кода других частей этой же программы. Наследование обеспечивает сокращение сроков построения программы, а полиморфизм — исключение избыточного кода.

Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. С++ сочетает в себе как возможности низкоуровневых языков программирования, так и возможности высокоуровневых.

Универсальность и гибкость языка позволяют использовать его в различных целях. Область применения данного языка включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также для создания игр с богатой насыщенной визуализацией. Исходя из этого, можно считать, что данный язык достаточно удобен для написания выбранного курсового проекта.

# 1 ПОСТАНОВКА ЗАДАЧИ

Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. Информация должна храниться в нескольких файлах, связанных определенным образом. В программе необходимо проводить учет товара, срок годности, а также производить списание продуктов. По запросу выдавать информацию о товаре, сроке годности товара.

При реализации операции редактирования, добавления, удаления информации необходимо предусмотреть операцию отмены последних действий.

Разработать иерархию классов с использованием наследования (не меньше 3-х уровней наследования). Разработать и использовать в программе классы контейнеров, итераторов и алгоритмов(свои и STL). Производить обработку исключительных ситуаций.

# 2 ОБЗОР ЛИТЕРАТУРЫ

2.1 Обзор методов и алгоритмов решения поставленной задачи

Одним из инструментов С++ являются шаблоны. В языке С++ существуют шаблоны функций и шаблоны классов.

Главная идея шаблонных функций – создание функций без указания точного типа некоторых или всех переменных.

Шаблонный (параметризированный) класс – некоторый шаблон, с помощью которых можно создавать классы, которые работают с типом данных, заданным как параметр.

Данный механизм позволяет решать пробему унификации алгоритма для различных типов: одну и ту же функцию или класс можно применять к разным типам данным, не используя отдельные варианты для каждого типа. Это наиболее полезно при определении контейнерных классов.

Контейнерный класс – шаблонный класс, который позволяет внутри себя хранить другие объекты заданного типа. Итерация элементов в контейнере и доступ к отдельным элементам осуществляются с помощью итераторов.

Итератор – интерфейс, предоставляющий доступ к элементам коллекции (массива или контейнера) и навигацию по ним.

Для реализации поставленной задачи в данном проекте используются шаблоны для создания классов: итератора, алгоритмов и собственного контейнера «Двунаправленная очередь», который применяется для хранения информации о продуктах питания и работы с ними. Для осуществления операции отмены последнего действия используется контейнер из стандартной библиотеки шаблонов vector (реализующий динамический массив).

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Структура входных и выходных данных

Входные и выходные данные будут представлены в виде файлов Alcohol, Non\_Alcohol, Finished\_prod, Milk, Semifinished\_prod, Raw\_prod и Alcoholspis, Non\_Alcoholspis, Finished\_prodspis, Milkspis, Semifinished\_prodspis, Raw\_prodspis.

В файле Alcohol будет храниться информация об алкогольной продукции, структура файла представлена в таблице 1.1

Таблица 1.1 – Alcohol

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Годен (с) | До | Цена (BYN) | Кол-во (шт.) | Объем (л) | % спирта |
| Вермут Martini | 2.2.2021 | 2.2.2031 | 27.26 | 25 | 100 | 16 |

В файле Non\_Alcohol будет храниться информация о безалкогольной продукции, структура данных представлена в таблице 1.2

Таблица 1.2 - Non\_Alcohol

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Годен (с) | До | Цена (BYN) | Кол-во (шт.) | Объем (л) | Степень газирования |
| Дарида  Снежок | 23.10.2022 | 23.01.2023 | 1.09 | 50 | 1 | газированная |

В файле Finished\_prod будет храниться информация о готовой продукции, структура данных представлена в таблице 1.3

Таблица 1.3 - Finished\_prod

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Годен (с) | До | Цена (BYN) | Кол-во (шт.) | Массса (г) | Температура |
| Пицца 4 сыра | 15.12.2022 | 21.12.2022 | 20.90 | 10 | 620 | только с печи |

В файле Milk будет храниться информация о молочной продукции, структура данных представлена в таблице 1.4

Таблица 1.4 - Milk

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Годен (с) | До | Цена (BYN) | Кол-во (шт.) | Объем (л) | % жирности |
| Ряженка  Агуша | 14.12.2022 | 14.01.2023 | 2,5 | 25 | 180 | 2.9 |

В файле Semifinished\_prod будет храниться информация о полуфабрикатах, структура данных представлена в таблице 1.5

Таблица 1.5 - Semifinished\_prod

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Годен (с) | До | Цена (BYN) | Кол-во (шт.) | Масса (г) | Способ готовки | Время (мин) |
| Шашлык из сезонныйх овощей | 14.12.2022 | 14.03.2023 | 12.26 | 10 | 500 | на мангале | 15 |

В файле Raw\_prod будет храниться информация о сырой продукции, структура данных представлена в таблице 1.6

Таблица 1.6. - Raw\_prod

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Годен (с) | До | Цена (BYN) | Кол-во (шт.) | Масса (г) | Производитель (страна поставщик) |
| Бананы фасованные | 14.12.2022 | 18.12.2022 | 2.97 | 45 | 1000 | Россия |

Файлы Alcoholspis, Non\_Alcoholspis, Finished\_prodspis, Milkspis, Semifinished\_prodspis, Raw\_prodspis будут иметь такую же структуру данных, как и вышеописанные файлы соответственно.

Промежуточные данные: Контейнер Queue (см. описание классов), написанный вручную, и STL контейнер vector.

При изменении данных в программе, происходит изменение в файлах после закрытия программы.

3.2 Разработка диаграммы классов

Разработанная диаграмма классов представлена в приложении А.

3.3 Описание классов

Базовый класс Продукты:

class Products{

protected:

char name[80]; //Название продукта

float price; //Цена

date data\_isgot; //Дата изготовления

date data\_end; //Дата, когда истекает срок годности

int need; //Кол-во поставленной продукции

public:

//Конструктор

Products(const char \*name = "", date date\_isgot = date(), date data\_end = date(), float price = 0,int need = 0);

//Методы для получения доступа к полям класса (геттеры и сеттеры)

const char\* getName() const;

void setName(const char \*features);

float getprice() const;

void setprice(float ch);

date getDate\_isgot() const;

void setDate\_isgot(date date\_isgot);

date getDate\_end() const;

void setDate\_end(date data\_end);

int getNeed() const; void setNeed(int sum);

//Метод для копирования инфомации из 1 объекта в другой

Products copy(Products&other);

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, Products &ticket);

friend ostream &operator<<(ostream &os, const Products &ticket);

//Перегрузка для чтения из файла

friend fstream &operator>>(fstream &in, Products &ticket);

//Перегрузка для записи в файл

friend fstream &operator<<(fstream &in, Products &ticket);

Products operator =(const Products& other); //Перегрузка оператора = bool operator ==(const Products& other); //Перегрузка оператора == MyBool operator<(Products& obj); //Перегрузка оператора <

};

Класс Еда:

class Food: public Products{

protected:

int weight; //Вес

public:

Food(const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0, int need = 0, int weight = 0); //Конструктор

//Методы для получения доступа к полям класса (геттеры и сеттеры)

int getWeight() const;

void setWeight(int sum);

//Метод для копирования информации из 1 объекта в другой

Food copy(Food&other);

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, Food &ticket);

friend ostream &operator<<(ostream &os, const Food&ticket);

//Перегрузка для чтения из файла

friend fstream &operator>>(fstream &in, Food &ticket);

//Перегрузка для записи в файл

friend fstream &operator<<(fstream &in, Food &ticket);

Food operator =(const Food& other); //Перегрузка оператора =

bool operator ==(const Food& other); //Перегрузка оператора ==

MyBool operator< (Food& other); //Перегрузка оператора <

};

Класс Напитки:

class Drinks: public Products{

protected:

float volume; //Объем

public:

Drinks(float volume = 0, const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0, int need = 0); //Конструктор

//Методы для получения доступа к полям класса (геттеры и сеттеры)

float getVolume() const;

void setVolume(float ch);

//Метод для копирования информации из одного объекта в другой

Drinks copy(Drinks&other);

//Перегрузки операторов ввода-вывода

friend istream &operator>>( istream &in, Drinks &ticket);

friend ostream &operator<<(ostream &os, const Drinks &ticket);

//Перегрузка операторов для записи и чтения из файла

friend fstream &operator>>(fstream &in, Drinks &ticket);

friend fstream &operator<<(fstream &in, Drinks &ticket);

MyBool operator <(Drinks & obj); //Перегрузка оператора <

Drinks operator =(const Drinks& other); //Перегрузка оператора ==

bool operator ==(const Drinks& other); //Перегрузка оператора =

};

Класс Алкоголь:

class Alcohol : public Drinks{

protected:

float alc\_proc; //% спирта

public:

//Конструктор

Alcohol(const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0, int need = 0,float volume = 0, float alc\_proc = 0);

//Методы для получения доступа к полям класса (геттеры и сеттеры)

float getAlc\_proc() const;

void setAlc\_proc(float ch);

//Метод для копирования информации с одного объекта в другой

Alcohol copy(Alcohol&other);

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, Alcohol &ticket);

friend ostream &operator<<(ostream &os, const Alcohol &ticket);

//Перегрузка операторов для записи и чтения из фйла

friend fstream &operator>>(fstream &in, Alcohol &ticket);

friend fstream &operator<<(fstream &in, Alcohol &ticket);

void output(); //Метод для вывода шапки таблицы

//Метод для выбора полей, по которым будет происходить поиск

void Search(Alcohol& Temp);

//Метод для выбора поля, по которому будет происходить сортировка

void choose\_need\_param(Alcohol& obj);

void menu(); //Метод для изменения полей класса

bool operator ==(const Alcohol& other); //Перегрузка оператора ==

Alcohol operator =(const Alcohol& other); //Перегрузка оператора =

bool operator<(Alcohol& obj); //Перегрузка оператора <

};

Класс Готовая продукция:

class Finished\_prod :public Food{

protected:

char t\_prod[80]; //Температура продукта

public:

Finished\_prod(const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0,int need = 0, int weight = 0, const char \*t\_prod = ""); //Конструктор

//Методы для получения доступа к полям класса (геттеры и сеттеры)

const char\* getT\_prod() const;

void setT\_prod(const char \*features);

//Метод для копирования информации с одного объекта в другой

Finished\_prod copy(Finished\_prod&other);

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, Finished\_prod &ticket);

friend ostream &operator<<(ostream &os, const Finished\_prod &ticket);

//Перегрузка операторов для записи и чтения из фйла

friend fstream &operator>>(fstream &in, Finished\_prod &ticket);

friend fstream &operator<<(fstream &in, Finished\_prod &ticket);

void output(); //Метод для вывода шапки таблицы

//Метод для выбора полей, по которым будет происходить поиск

void Search(Finished\_prod& Temp);

//Метод для выбора поля, по которому будет происходить сортировка void choose\_need\_param(Finished\_prod& obj);

void menu(); //Метод для изменения полей

bool operator==(const Finished\_prod& other); //Перегрузка оператора ==

//Перегрузка оператора =

Finished\_prod operator =(const Finished\_prod& other);

bool operator< (Finished\_prod& other); //Перегрузка оператора <

};

Класс Безалкогольные напитки:

class Non\_alcohol: public Drinks {

protected:

char degree\_of\_soda[80]; //Степень газированния

public:

Non\_alcohol(const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0,int need = 0, float volume = 0, const char \*degree\_of\_soda = ""); //Конструктор

//Методы для получения доступа к полям класса (геттеры и сеттеры)

const char\* getDegree\_of\_soda() const;

void setDegree\_of\_soda(const char \*features);

Non\_alcohol copy(Non\_alcohol&other); //Метод для копирования

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, Non\_alcohol &ticket);

friend ostream &operator<<(ostream &os, const Non\_alcohol &ticket);

//Перегрузка оператора чтения из файла

friend fstream &operator>>(fstream &in, Non\_alcohol &ticket);

//Перегрузка оператора записи в файл

friend fstream &operator<<(fstream &in, Non\_alcohol &ticket);

void output(); //Вывод шапки таблицы

void Search(Non\_alcohol& Temp); //Для поиска

void menu(); //Для изменения информации в полях

//Для выбора поля, по которому будет производиться сортировка

void choose\_need\_param(Non\_alcohol& obj);

bool operator==(const Non\_alcohol& other);//Перегрузка оператора ==

bool operator< (Non\_alcohol& other); //Перегрузка оператора <

//Перегрузка оператора =

Non\_alcohol operator =(const Non\_alcohol& other);

};

Класс Молочная продукция

class Milk: public Drinks{

protected:

float fat\_proc; //% жирности

public:

Milk(const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0,int need = 0,float volume = 0, float fat\_proc = 0); //Конструктор

//Методы для получения доступа к полям класса (геттеры и сеттеры)

float getFat\_proc() const;

void setFat\_proc(float ch);

//Метод для копирования информ из 1 объекта в другой

Milk copy(Milk&other);

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, Milk &ticket);

friend ostream &operator<<(ostream &os, const Milk &ticket);

//Перегрузка операторов для записи и чтения из файл

friend fstream &operator>>(fstream &in, Milk &ticket);

friend fstream &operator<<(fstream &in, Milk &ticket);

void output(); //Для вывода шапки таблицы

void Search(Milk& Temp); //Для проведения поиска

void menu(); //Для измнения полей класса

//Для выбора поля, по которому будет производиться сортировка

void choose\_need\_param(Milk& obj);

Milk operator =(const Milk& other); //Перегрузка оператора =

bool operator==(const Milk& other); //Перегрузка оператора ==

bool operator< (Milk & other); //Перегрузка оператора <

};

Класс Сырая продукция:

class Raw\_prod: public Food{

protected:

char producer[80]; //Фирма производитель/страна

public:

//Конструктор

Raw\_prod(const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0,int need = 0, int weight = 0, const char \*producer = "");

//Методы для получения доступа к полям класса (геттеры и сеттеры)

const char\* getProducer() const;

void setProducer(const char \*features);

Raw\_prod copy(Raw\_prod&other); //Метод для копирования

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, Raw\_prod &ticket);

friend ostream &operator<<(ostream &os, const Raw\_prod &ticket);

//Перегрузка операторов для записи и чтения из файла

friend fstream &operator>>(fstream &in, Raw\_prod &ticket);

friend fstream &operator<<(fstream &in, Raw\_prod &ticket);

void output(); //Вывод шапки таблицы

void Search(Raw\_prod& Temp); //Для поиска

void menu(); //Для изменения информации в полях

//Для выбора поля, по которому будет производиться сортировка

void choose\_need\_param(Raw\_prod& obj);

bool operator==(const Raw\_prod& other); //Перегрузка оператора ==

Raw\_prod operator =(const Raw\_prod& other); //Перегрузка оператора =

bool operator< (Raw\_prod& other); //Перегрузка оператора <

};

Класс Полуфабрикаты:

class SemiFinished\_prod :public Food{

protected:

char cook\_way[80]; //Способ приготовления

int time\_cook; //Время готовки

public:

SemiFinished\_prod(const char \*name = "", date date\_isgot = date(), date date\_end = date(), float price = 0,int need = 0, int weight = 0, const char \*cook\_way = "", int time\_cook = 0 ); //Конструктор

//Методы для получения доступа к полям класса (геттеры и сеттеры)

const char\* getCook\_way() const;

void setCook\_way(const char \*features);

int getTime\_cook() const;

void setTime\_cook(int sum);

SemiFinished\_prod copy(SemiFinished\_prod&other); //Для копирования

//Перегрузки операторов ввода-вывода

friend istream &operator>>(istream &in, SemiFinished\_prod &ticket);

friend ostream &operator<<(ostream &os, const SemiFinished\_prod &ticket);

//Перегрузка операторов для записи и чтения из файла

friend fstream &operator>>(fstream &in, SemiFinished\_prod &ticket);

friend fstream &operator<<(fstream &in, SemiFinished\_prod &ticket);

void output(); //Вывод шапки таблицы

void Search(SemiFinished\_prod& Temp); //Для поиска

void menu(); //Для изменения информации в полях

//Для выбора поля, по которому будет производиться сортировка

void choose\_need\_param(SemiFinished\_prod& obj);

//Перегрузка оператора =

SemiFinished\_prod operator =(const SemiFinished\_prod& other);

//Перегрузка оператора ==

bool operator==(const SemiFinished\_prod& other);

//Перегрузка оператора <

bool operator< (SemiFinished\_prod& other);

};

Шаблонный класс Очередь:

template <class T>

class Queue {

private:

Node<T> \*head = nullptr; //Указатель на начало очереди

Node<T> \*tail = nullptr; //Указатель на конец очереди

int amount = 0; //Кол-во узлов очереди

public:

Queue() {}; //Конструктор

~Queue() { while (this->head)this->popHead(); }; //Деструктор

int size() { return this->amount; }//Функция, возвращающая кол-во узлов

void pushHead(T obj); //Добавление элемента в начало очереди

//Установление элемента в определенную позицию

void push(T obj, int num);

void pushTail(T obj); //Добавление элемента в конец очереди

T popHead(); //Удаление 1 элемента

T pop(int num); //Удаление определенного элемента

T popTail(); //Удаление последнего элемента

T& operator [](int num); //Функция для нахождение n-ого элемента

void output1(); //Функция для вывода информ полей

void deleteAll(); //Полная очистка очереди

Node<T>\* begin();

Node<T>\* end();

};

Шаблонный класс Итератор:

template<class T>

class Iterator {

public:

Node<T>\* current; //Указатель на текущий элемент

Iterator(Node<T>\* temp = nullptr); //Конструктор

~Iterator(); //Деструктор

T& operator\*(); //Получение данных с узла

void operator=(Node<T>\* temp); //Перегрузка оператора =

bool operator++(); //Переход к след узлу очереди

bool operator--(); //Переход к предыдущему узлу очереди

bool operator==(Iterator<T>& ob);

bool operator==(Node<T>\* ob);

bool operator!=(Iterator<T>& ob);

bool operator!=(Node<T>\* ob);

};

Класс ошибка для обработки исключительных ситуаций:

class MyException{ //Базовый класс обработки исключений

protected:

int code; //Код ошибки

char type\_error[80]; //Тип ошибки

public:

MyException(int \_code = 0, const char\* \_type = ""); //Конструктор

~ MyException(){}; //Деструктор

void Print(); //Ф-я для вывода сообщения на экран

};

Класс ошибка ввода для обработки исключительных ситуаций при вводе:

class inputException : public MyException{

protected:

char message[80]; //Сообщение об ошибке

public:

inputException(int code = 0, const char\* type\_error = "", const char\* \_message = ""); //Конструктор

~inputException() {}; //Деструтор

void Print(); //Вывод на экран

};

Класс для обработки исключений при работе с очередью:

class QueueException : public MyException{

protected:

char ms1[80]; //Сообщение об ошибке

public:

QueueException(int code = 0, const char\* type\_error = "", const char\* \_message = ""); //Конструктор

~QueueException() {}; //Деструктор

void Print(); //Вывод на экран

};

Класс для обработки исключений при работе с файлами:

class FileException : public MyException{

protected:

char ms1[80]; //Сообщение об ошибке

public:

FileException(int code = 0, const char\* type\_error = "", const char\* \_message = ""); //Конструктор

~FileException() {}; //Деструктор

void Print(); //Метод для вывода сообщения на экран

};

Базовый класс файл:

class File {

protected:

string file\_nam; //Название файла

fstream ft; //Имя потока

public:

File(string \_nameFile) : file\_name(\_nameFile) {};//Конструктор

File(){}; //Деструктор

};

Шаблонный класс бинарный файл, для записи массива объектов в бинарный файл:

template<class T>

class bFile :public File {

public:

bFile(string \_name = ""); //Конструктор

~bFile(); //Деструктор

bool feof(); //Проверка на конец файла

bool openReadFile(); //Проверка на открытие файла для чтения

bool openWriteFile(); //Проверка на открытие файла для записи

void readFile(T& temp); //Чтение из файла

void writeFile(T& temp); //Запись в файл

};

Шаблонный класс Алгоритм, в котором содержатся следующие методы:

template <class T>

class Algoritm {

public:

//Шаблонная функция для поиска нужных продуктов

Queue<T> poisk(Iterator<T> begin, Iterator<T> end, T buff);

//Шаблоннная функция для поиска продуктов с истекшим сроком годности и занесения их в отдельную очередь

void poisk\_prod(Queue<T>& st, Iterator<T> begin, Iterator<T> end, Queue<T>& spis, T buff);

//Шаблонная функция сортировки

void sort(Iterator<T> begin, Iterator<T> end, T buff);

//Шаблонная функция для добавления элемента в очередь

void add(Queue<T>& st, vector<T>& returnPred, vector<int>& action, vector<int>& place);

//Шаблонная функция для удаления элементов из очереди

void delete\_el(Queue<T>& st, Queue<T>& spis, vector<T>& returnPred, vector<int>& action, vector<int>& place, string spis\_prod);

//Шаблонная функция для отмены последнего действия

void cancel(Queue<T>& queue, Queue <T>& spis, vector<T>& returnPred, vector<int>& action, vector<int>& place);

//Шаблонная функция записи в файл

void FillToFile(Queue<T>& st, bFile<T> &name, Iterator<T> begin, Iterator<T> end);

//Шаблонная функция чтения из файла

void FillFromFile(Queue<T>& st, bFile<T> &name);

};

Шаблонный класс Интерфейс, в котором содержатся следующие методы:

template<class T>

class Interface {

public:

//Меню для выбора класса

void menu\_of\_class();

//Меню для работы с продуктами

void menu(Queue<T>& queue, string bName, string spisprod);

//Меню для работы со списанными продуктами

void work\_with\_spis\_prod(Queue<T>& queue, string spisprod);

};

# 

# 4 РАЗРАБОТКА ПРОГРАМНЫХ МОДУЛЕЙ

**4.1 Разработка схем алгоритмов**

Схема алгоритма FillFromFile () описана в приложении Б.

Схема алгоритма poisk\_prod () описана в приложении В.

**4.2 Разработка алгоритмов**

Алгоритм по шагам шаблонной функции poisk() для поиска продукта

по выбранным параметрам.

1. Начало.

2. Входные данные:

Queue<T>:: Iterator begin – итератор, указывающий на первый элемент очереди;

Queue<T>::Iterator end – итератор, указывающий на элемент, следующий за последним в очереди;

T buff – объект с параметрами, по которым ищем.

Выходные данные:

Queue<T> q – контейнер Queue найденных элементов.

3. Создаем объект q класса Queue для хранения найденных элементов.

4. Цикл по контейнеру Queue<T> от begin до end.

5. Если информация в узле \*begin, на который указывает итератор begin, равна информации в объекте buff переход на шаг 6, иначе 7.

6. Добавляем в контейнер Queue<T> q узел \*begin, на который указывает итератор begin.

7. begin = begin + 1 перемещение итератора на следующий элемент.

8. Конец цикла по контейнеру Queue<T>.

9. Возвращаем Queue<T> q.

10. Конец.

Алгоритм по шагам шаблонной функции sort() для сортировки очереди.

1. Начало.

2. Входные данные:

Queue<T>:: Iterator begin – итератор, указывающий на первый элемент очереди;

Queue<T>::Iterator end – итератор, указывающий на элемент, следующий за последним в очереди;

T buff – объект с одним заполненным полем, по которому и будем проводить сортировку.

Выходные данные:

Отсортированный контейнер Queue<T> от begin до end.

Промежуточные данные:

Iterator<T> min – итератор, указывающий на минимальный элемент очереди;

Iterator<T> j – итератор для цикла по контейнеру;

T tmp – объект для временного хранения минимального значения.

3. Цикл по контейнеру Queue<T> от begin до end.

4. Устанавливаем Iterator<T> min и Iterator<T> j на начало очереди.

5. Цикл по контейнеру Queue<T> от j до end.

6. С помощью ф-ии copy() копируем поле, по которому будет проводиться сортировка, из объекта, на который указывает итератор min в buff.

7. Если информация в узле \*j, на который указывает итератор j меньше информации в объекте buff переход на шаг 8, иначе 9.

8. Присваиваем итератору min значение итератора j.

9. j = j+1 перемещение итератора на следующий элемент.

10. Конец цикла по j.

11. Если итератор min не равен итератору begin переход на шаг 12, иначе 15.

12. Запоминаем новое значение объекта tmp, равное узлу \*min, на который указывает итератор min.

13. Адрес узла, на который указывает min, приравниваем к адресу узла, на который указывает begin.

14. Узел \*begin, на который указывает итератор begin, приравниваем к объекту tmp.

15. begin = begin + 1 перемещение итератора на следующий элемент.

16. Конец цикла по begin.

17. Конец.

# 5 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

После запуска программы откроется окно, продемонстрированное на рисунке 1.

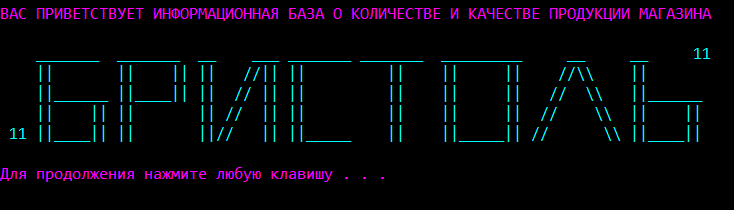


Рисунок 1 – Приветственное окно

В окне на рисунке 2 предоставлен выбор класса, с которым далее будет произодиться работа.

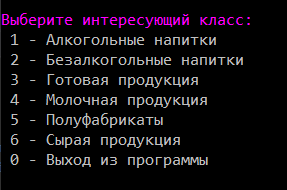
****

Рисунок 2 - Основное меню

После выбора определенного класса пользователь увидит меню действий, представленное на рисунке 3.

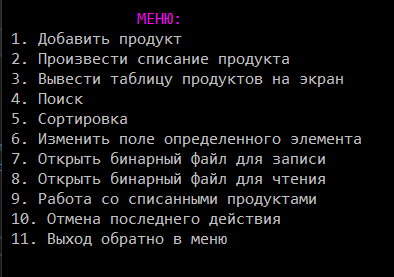


Рисунок 3 - Меню действий

Далее на рисунках 4, 5, 6 представлена реализация основных методов класса Algoritm.

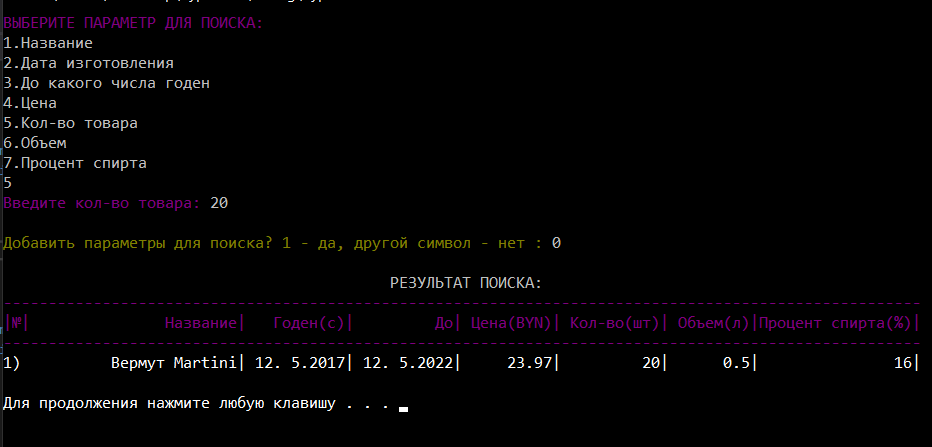


Рисунок 4 - Реализация метода поиска

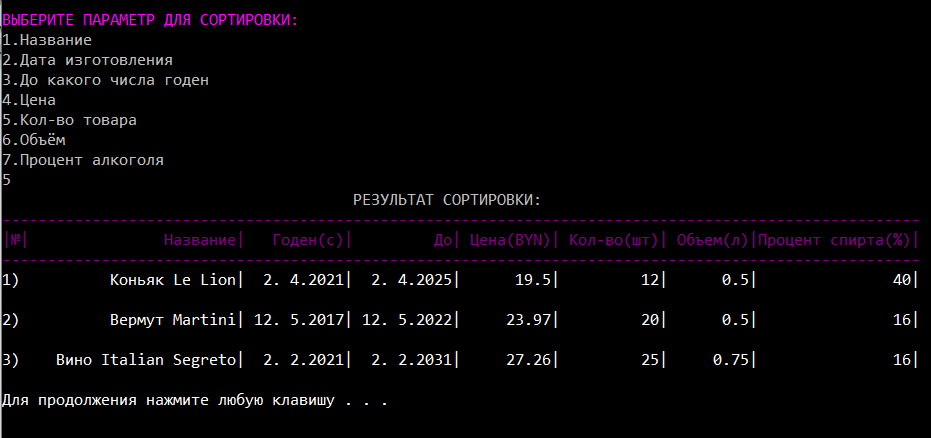


Рисунок 5 - Реализация метода сортировки (по кол-ву)

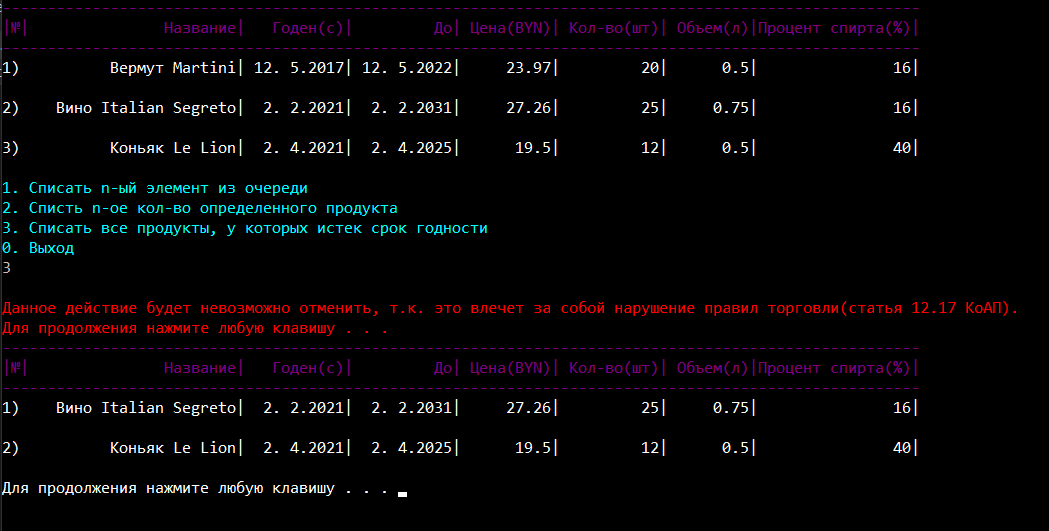


Рисунок 6 - Списание продуктов, у которых истек срок годности

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта мною была разработана программа, которая позволяет работать с информационной базой продуктового магазина “Бристоль”. Программа рассчитана на упрощение работы бухгалтера этого магазина. Он может вводить различную информацию, выводить ее на экран, изменять, добавлять, производить калькуляцию, списание продуктов, а в случае допущения ошибки, отменять последние действия.

Использовались среда разработки Visual Studio 2017 и операционная система Windows 10. К достоинствам программы можно отнести простой и понятный интерфейс, что в свою очередь обеспечивает удобство пользования. Программа написана в Visual Studio, так как, на мой взгляд, Visual Studio является действительно мощной средой разработки программного обеспечения, в связи с тем, что она включает в себя компиляторы и многие другие функции для упрощения процесса разработки. В программе реализована система обработки исключительных ситуаций при вводе необходимой информации, что упрощает взаимодействие с программой.

Системные требования:

– Операционная система Windows 10, XP;

– Процессор Intel Pentium III 1 GHz;

– Оперативная память 500 МВ;

– Свободное место на жестком диске: 5 MB.

# ЛИТЕРАТУРА

1. Объектно-ориентированное программирование на языке С++: учеб. пособие / Ю. А. Луцик, В. Н. Комличенко. – Минск : БГУИР, 2008.

2. Конструирование программ и языки программирования: метод. указания по К65 курсовому проектированию для студ. спец. I-40 02 01 “Вычислительные машины, системы и сети” для всех форм обуч. / сост. А. В. Бушкевич,   
А. М. Ковальчук, И. В. Лукьянова. – Минск : БГУИР, 2009.

3. Шилд, Г. Программирование на C++/Г. Шилд. – Минск : ООО «Попури», 1998.

# ПРИЛОЖЕНИЕ А

*(обязательное)*

Диаграмма классов

# ПРИЛОЖЕНИЕ Б

*(обязательное)*

Схема алгоритма FillFromFile()

# ПРИЛОЖЕНИЕ В

*(обязательное)*

Схема алгоритма poisk\_prod()

# ПРИЛОЖЕНИЕ Г

*(обязательное)*

Код программы

# ПРИЛОЖЕНИЕ Д

*(обязательное)*

Ведомость документов