МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет) \_\_\_\_\_\_\_\_\_Институт информационных технологий\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_МПО ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КУРСОВАЯ РАБОТА

по дисциплине Объектно-ориентированное программирование

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

на тему Объектно-ориентированное программирование на языке С ++

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил студент группы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1ПИб-02-1оп-22

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

направления подготовки (специальности)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Программная инженерия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*шифр, наименование*

\_\_\_\_Микуцких Григорий Андреевич\_\_\_\_

*фамилия, имя, отчество*

Руководитель

Шаханов Никита Иванович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*должность*

Дата представления работы

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г.

Заключение о допуске к защите

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

количество баллов

Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Череповец, \_2024\_

*год*

Аннотация

Курсовая работа посвящена освоению и закреплению материала, полученного в ходе изучения дисциплины «Объектно-ориентированного программирования».

В ходе работы разработана программа, содержащая структуру хранения и обработки объектов бытовой техники.

Оглавление

[Введение 4](#_Toc167290580)

[1. Объектно-ориентированный анализ предметной области 5](#_Toc167290581)

[2. Проектирование классов 7](#_Toc167290582)

[3. Логическая структура программы 9](#_Toc167290583)

[4. Модульная структура программы 10](#_Toc167290584)

[5. Тестирование программы 14](#_Toc167290585)

[Заключение 20](#_Toc167290586)

[Список литературы 21](#_Toc167290587)

[Приложение 1 22](#_Toc167290588)

[Приложение 2 30](#_Toc167290589)

[Приложение 3 31](#_Toc167290590)

# Введение

Для того чтобы эффективно использовать и реализовывать языки программирования, необходимо хорошо знать их фундаментальные понятия. Для изучения инструментов объектно-ориентированного программирования требуется хорошее знание основ языка программирования С++, понимание концепции данных и механизма функций С++, а также умение использовать их при решении практических задач.

Цель: освоение концепций объектно-ориентированного программирования и методов составления программ.

Задача: разработать структуру хранения и обработки объектов бытовой техники.

План решения: разработать иерархию родственных типов, корневой класс которой абстрактный базовый класс, создать обобщенный контейнерный класс и от него производный класс – шаблон для хранения указателей на абстрактный базовый класс-интерфейс (линейным двусвязным списком), для хранения объектов каждого производного класса использовать статический вектор, реализовать функции обработки данных, файловый ввод/вывод, ввод данных с клавиатуры, вывод данных на дисплей и обработку различных исключительных ситуаций, используя язык программирования C++.

# Объектно-ориентированный анализ предметной области

В центре объектно-ориентированной модели стоит понятие объекта. В реальном мире существуют простые понятия «собака», «дом» или обобщенные «животное», «жилище». Все они могут быть заменены термином «объект», абстрагирующим понятие, или порождающим класс одинаковых (однотипных) понятий. В основе идеи объектно-ориентированного программирования лежит природная способность человеческого мышления к абстрагированию и классификации. Так, любой из объектов:

* находится в определенном состоянии;
* имеет определенное поведение;
* существует во взаимодействии с окружающим миром.

Объект – это сущность, способная сохранять свое состояние (информацию) и обеспечивающая набор операций для проверки и изменения состояния объекта.

В определении объекта отчетливо видны две составляющие части:

* состояние (синонимы «свойства», «данные», «атрибуты» объекта);
* методы (синонимы «набор операций» или «функции обработки данных» объекта).

Можно утверждать, что объект, это абстрактная сущность, наделенная реальными характеристиками объектов окружающего мира. В программировании это методология написания программ, представляющая объектную модель предметной области [1].

В основе ООП находятся три базовых понятия, три кита объектного подхода:

* инкапсуляция (Encapsulation);
* наследование (Inheritance);
* полиморфизм (Polymorphism).

Под инкапсуляцией понимается слияние данных и действий, которые можно выполнить над этими данными. Действия реализуются с помощью функций и называются методами. Такое слияние порождает абстрактные типы данных, называемые классами.

Под наследованием понимается один из главных механизмов проектирования, который позволяет в высокой степени абстрагировать проект, разрабатывая базовые (общие) классы, порождающие новые (частные), наследующие свойства и методы классов предков и приобретающие новые качества.

Идея наследования объектов заимствована у живой природы, решает проблему модификации объектов и придает ООП гибкость. Классы наследуют свойства других классов и сами могут порождать новые классы. Так наследование позволяет легко расширять функциональность классов. Но самое главное, что дает наследование, это возможность выстраивать иерархию объектов, тем самым моделируя задачи предметной области. Объектная модель более приближена к реальной задаче, чем функциональная [1].

Под полиморфизмом понимается способность объекта реагировать на запрос сообразно своему типу. В узком смысле, это свойство, которое позволяет использовать одно и то же имя для решения одинаковых внешне, но технически разных задач. Например, мы знаем, как найти площадь геометрической фигуры. Обобщенный метод «площадь» реализуется различным образом для дочерних объектов: круг, прямоугольник, трапеция и другие, тем самым, алгоритм становится конкретным тогда, когда применяется к конкретному объекту.

Объект полиморфичен, когда один и тот же метод применяется к объектам различных классов, а реакция объектов различная. Запрос посылается одинаково объектам разных классов, но в ответ на запрос сама операция выполняется различным образом [1].

Класс-интерфейс представляет собой именованный набор отложенных методов (объявлений функций). Абстрактные методы являются чисто виртуальными функциями, поскольку не имеют никакой реализации.

Виртуальная функция – это функция-член, которую предполагается переопределить в производных классах.

Диаграмма классов на унифицированном языке моделирования (UML) – это диаграмма статической структуры, которая описывает структуру системы, показывая её классы, их атрибуты, операции (методы) и отношения между объектами. Их можно использовать для моделирования объектов, составляющих систему, демонстрации отношений между объектами, описания ролей этих объектов и предоставляемых ими услуг.

Бытовая техника — это оборудование и приборы, облегчающие ведение домашнего хозяйства благодаря его механизации. К ней относят телевизоры, пылесосы, холодильники, плиты (газовые, электрические), чайники, утюги и т.д.

Бытовая техника, как и все товары, имеет свои характеристики: цена, потребляемая мощность, компания выпуска, габариты, ёмкость и прочие параметры, по которым можно найти подходящий каждому.

Программа содержит следующие классы:

1. IAppliances – абстрактный класс-интерфейс, содержит объявления методов (чистые виртуальные функции);
2. Appliances – АТД, содержит реализацию функций класса-интерфейса и базовый набор параметров, которыми обладают все объекты бытовой техники;
3. Refrigerator – наследует интерфейс класса Appliances, для холодильников;
4. Microwave – наследует интерфейс класса Appliances, для микроволновок;
5. Compact\_electric\_stove – наследует интерфейс класса Microwave, для компактных электроплит (электроплит без духовки);
6. Bread\_maker – наследует интерфейс класса Compact\_electric\_stove, для хлебопечек;
7. Electric\_stove – наследует интерфейс класса Bread\_maker, для электрических плит (с духовкой);
8. Container – обобщённый (void\*) контейнерный класс для хранения указателей на абстрактный базовый класс-интерфейс;
9. Container\_IAppliances – производный от класса Container шаблонный класс для обработки объектов производных классов от Appliances;
10. Vector\_IAppliances – шаблонный класс для хранения объектов производных классов от Appliances.

Отношения между классами представлены на рис. 1.

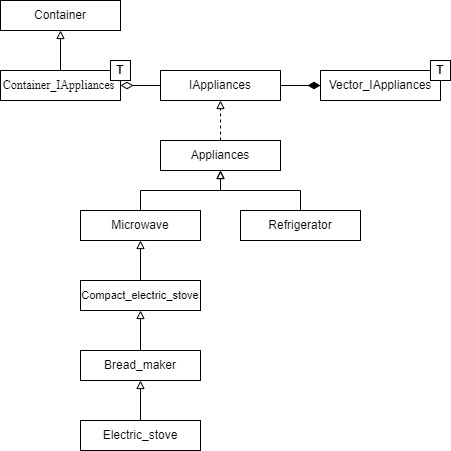


Рис. 1. Контекстная диаграмма классов

Типы отношений между классами:

* IAppliances и Appliances – реализация;
* Refrigerator, Microwave и Appliances; Compact\_electric\_stove и Microwave; Bread\_maker и Compact\_electric\_stove; Electric\_stove и Bread\_maker – наследование;
* IAppliances и Container\_IAppliances – агрегация;
* IAppliances и Vector\_IAppliances – композиция.

# Проектирование классов

Методы и поля классов представлены в табл. 1 – 17.

Таблица 1

Методы класса IAppliances

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | set\_name | Отложенный метод сеттера имени |
| public | void | set\_firm | Отложенный метод сеттера фирмы |
| public | void | set\_series | Отложенный метод сеттера серии |
| public | void | set\_price | Отложенный метод сеттера цены |
| public | char\* | get\_name | Отложенный метод геттера имени |
| public | Firm | get\_firm | Отложенный метод геттера фирмы |
| public | std::string | get\_series | Отложенный метод геттера серии |
| public | int | get\_price | Отложенный метод геттера цены |
| public | void | print | Отложенный метод вывода всех данных о товаре |
| public | void | print\_to\_file | Отложенный метод записи товаров в файл |
| public | void | get\_class\_name | Отложенный метод возвращения типа товара |

Таблица 2

Поля класса Appliances

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| private | char\* | name | Имя товара |
| private | std::string | series | Серия товара |
| private | int | price | Цена товара |
| private | Firm | firm | Фирма товара |

Таблица 3

Методы класса Appliances

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | set\_name | Переопределение сеттера имени |
| public | void | set\_firm | Переопределение сеттера фирмы |
| public | void | set\_series | Переопределение сеттера серии |
| public | void | set\_price | Переопределение сеттера цены |
| public | char\* | get\_name | Переопределение геттера имени |
| public | Firm | get\_firm | Переопределение геттера фирмы |
| public | std::string | get\_series | Переопределение геттера серии |
| public | int | get\_price | Переопределение геттера цены |
| public | void | print | Переопределение вывода данных |
| public | void | print\_to\_file | Переопределение записи в файл |

Таблица 4

Поля класса Refrigerator

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| protected | int | min\_temp | Минимальная температура |
| protected | bool | freezer | Наличие морозильной камеры |

Таблица 5

Методы класса Refrigerator

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | set\_min\_temp | Сеттер минимальной температуры |
| public | void | set\_freezer | Сеттер наличия морозильной камеры |
| public | std::string | get\_freezer | Геттер минимальной температуры |
| public | int | get\_min\_temp | Геттер наличия морозильной камеры |
| public | void | print | Переопределение вывода данных |
| public | void | print\_to\_file | Переопределение записи в файл |
| protected | void | get\_class\_name | Переопределение геттера типа товара |

Таблица 6

Поля класса Microwave

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| protected | int | min\_power | Потребляемая мощность |

Таблица 7

Методы класса Microwave

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | set\_power | Сеттер потребляемой мощности |
| public | int | get\_power | Геттер потребляемой мощности |
| public | void | print | Переопределение вывода данных |
| public | void | print\_to\_file | Переопределение записи в файл |
| protected | void | get\_class\_name | Переопределение геттера типа |

Таблица 8

Поля класса Compact\_electric\_stove

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| protected | int | mumber\_of\_modes | Число режимов |

Таблица 9

Методы класса Compact\_electric\_stove

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | set\_mumber\_of\_modes | Сеттер количества режимов |
| public | int | get\_mumber\_of\_modes | Геттер количества режимов |
| public | void | print | Переопределение вывода данных о товаре |
| public | void | print\_to\_file | Переопределение записи данных в файл |
| protected | void | get\_class\_name | Переопределение геттера типа товара |

Таблица 10

Поля класса Bread\_maker

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| protected | int | volume | Объём |

Таблица 11

Методы класса Bread\_maker

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | set\_volume | Сеттер объёма |
| public | int | get\_volume | Геттер объёма |
| public | void | print | Переопределение вывода данных |
| public | void | print\_to\_file | Переопределение записи в файл |
| protected | void | get\_class\_name | Переопределение геттера типа |

Таблица 12

Поля класса Electric\_stove

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| protected | int | number\_of\_burners | Количество конфорок |
| protected | int | max\_temperature | Максимальная температура |

Таблица 13

Методы класса Electric\_stove

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | set\_max\_temperature | Сеттер максимальной температуры |
| public | void | set\_number\_of\_burners | Сеттер количества конфорок |
| public | int | get\_max\_temperature | Геттер максимальной температуры |
| public | int | get\_number\_of\_burners | Геттер количества конфорок |
| public | void | print | Переопределение вывода данных о товаре |
| public | void | print\_to\_file | Переопределение записи данных в файл |
| protected | void | get\_class\_name | Переопределение геттера типа товара |

Таблица 14

Поля класса Container

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| private | std::list<void\*> | list | Список для хранения указателей |

Таблица 15

Методы класса Container

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | push\_back | Добавление элемента в конец списка |
| public | void | push\_front | Добавление элемента в начало списка |
| public | void | erase | Удаление элемента списка |
| public | int | size | Возвращение размера списка |
| public | bool | empty | Возвращение пустой ли список |
| public | void | clear | Очистка всего списка |
| public | list<void\*>::const\_iterator | cbegin | Возвращает итератор начала списка |
| public | list<void\*>::const\_iterator | cend | Возвращает итератор конца списка |
| public | list<void\*>::iterator | begin | Возвращает итератор начала списка |
| public | list<void\*>::iterator | end | Возвращает итератор конца списка |

Таблица 16

Методы класса Container\_IAppliances

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | push\_back | Вызов добавления в конец списка |
| public | void | erase | Вызов удаление элемента из списка |
| public | bool | empty | Вызов проверки на пустоту списка |
| public | void | clear | Вызов очистки списка |
| public | bool | search\_firm | Вывод элементов по указанной фирме |
| public | void | search\_price\_is\_less\_than | Вывод элементов ниже указанной цены |
| public | void | sort\_firm | Сортировка списка по указанной фирме |
| public | T\* | return\_object | Возвращение адреса элемента в списке |
| public | void | print | Вывод элементов списка |
| public | void | output\_in\_file | Запись элементов списка в файл |

Таблица 17

Поля класса Vector\_IAppliances

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| private | std::vector<T\*> | vector\_IAppliances | Вектор для хранения объектов |

Таблица 18

Методы класса Vector\_IAppliances

|  |  |  |  |
| --- | --- | --- | --- |
| Область видимости | Тип данных | Имя | Краткое описание |
| public | void | push\_back | Добавление элемента в конец вектора |
| public | void | add\_one\_to\_list | Добавление элемента в вектор и список |
| public | void | add\_from\_vector\_to\_list | Добавление элемента вектора в список |
| public | T\* | return\_object | Возвращение адреса элемента вектора |
| public | void | del\_object | Удаление элемента из вектора |

Детальная диаграмма классов представлена на рис. 2.

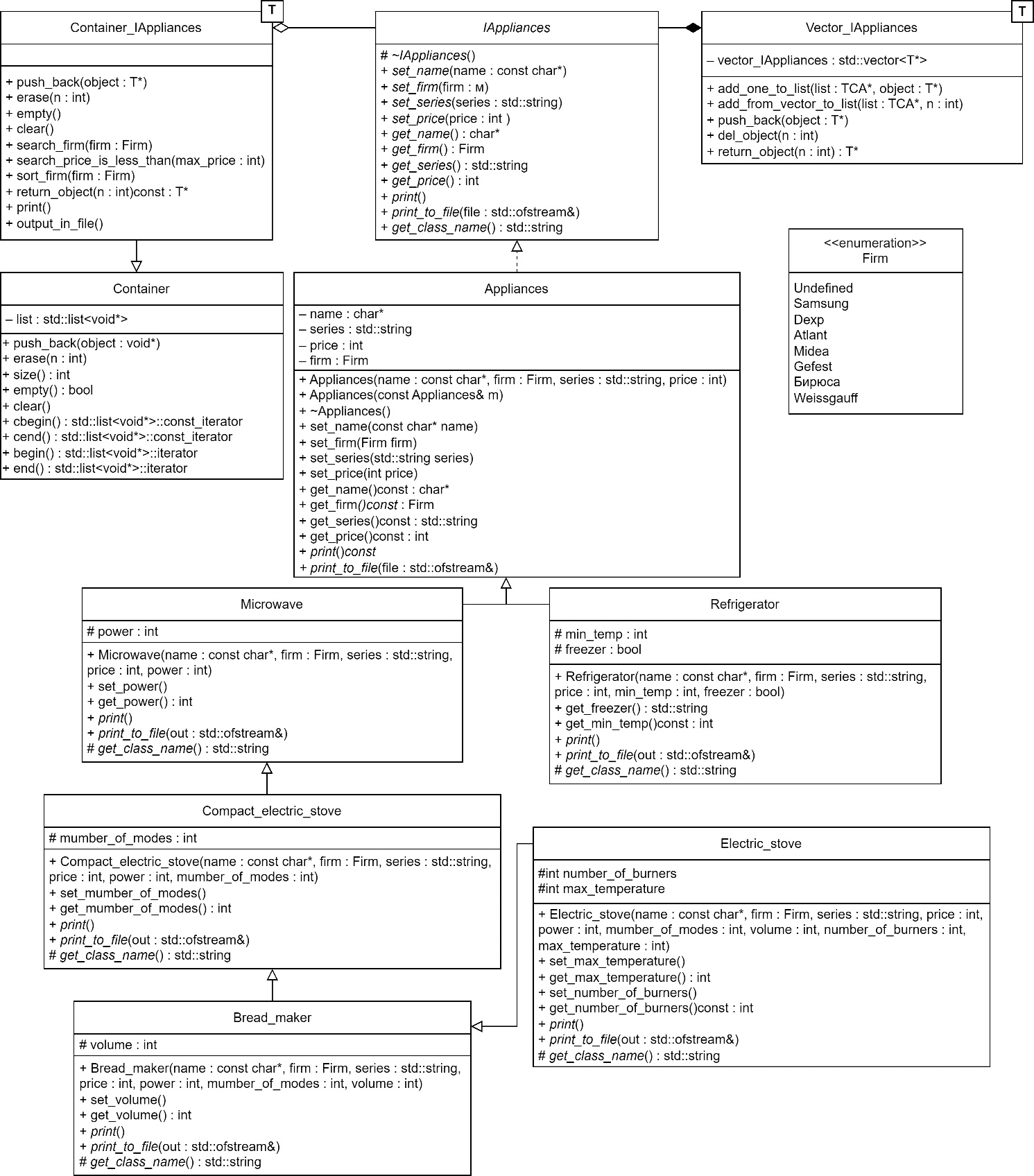


Рис. 2. Детальная диаграмма классов

# Логическая структура программы

Класс Firm содержит перечисление фирм, доступных для выбора в программе. Вместо имени выводится константное значение (код) фирмы.

Класс IAppliances является классом-интерфейсом и содержит отложенные методы изменения и возвращения базовых полей объектов (имя, фирма, серия, цена), вывода в консоль, записи в файл и возвращения типа товара, которые обязательно должны быть переопределены в дочерних классах. Чисто виртуальные функции не позволяют создавать объекты данного класса.

Класс Appliances содержит конструктор для базовых параметров бытовой техники и переопределяет методы класса-интерфейса. Для того, чтобы не получалось создать объект этого класса, не был переопределён метод возвращения типа товара.

Объекты можно создавать от классов Refrigerator, Microwave, Compact\_electric\_stove, Bread\_maker, Electric\_stove, которые переопределяют методы вывода в консоль, записи в файл и возвращения типа товара, а также содержат свои поля и методы для работы с ними.

Благодаря использованию механизмов замещения (определения абстракции объекта с последующим переопределением в производных классах) и наследования (переноса структуры и поведения от одного классу другому) достигается полиморфная обработка родственных объектов, что увеличивает степень повторного использования кода и упрощает логику программы.

В программе также содержится обработка исключений: ввод и вывод неверных параметров выдаст исключения – сообщения о неверных данных в объекте или невозможность изменить данные на некорректные.

У пользователя есть возможность: ввести данные из файла, с клавиатуры, вывести все товары в консоль, удалить товар, отсортировать товары по компании, вывести товары по стоимости, не превосходящие определённого значения, заданного пользователем, вывести товары определённой компании, изменить имя, фирму, серию или цену выбранного товара.

Логическая структура программы представлена на рис. 3.

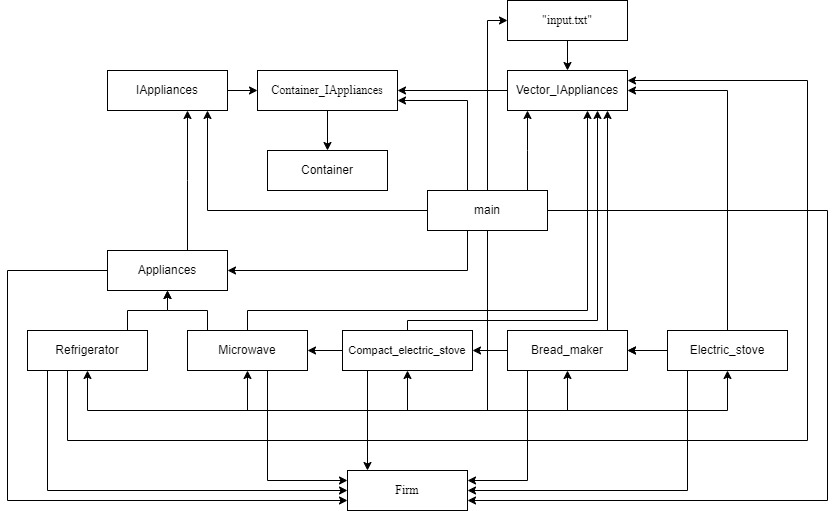


Рис. 3. Логическая структура

# Модульная структура программы

Модульная структура программы представлена на рис. 4.

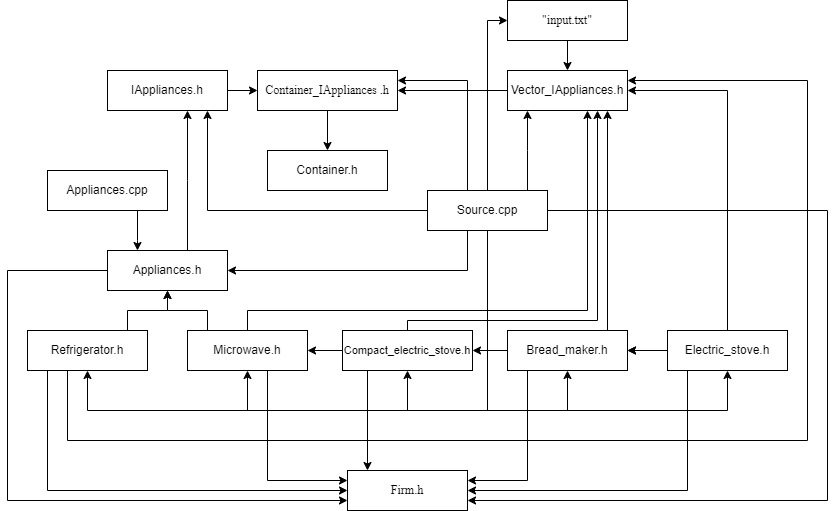


Рис. 4. Модульная структура

# Тестирование программы

Для тестирования классов были подобраны тестовые данные в табл. 19.

Таблица 19

Тестовые данные

|  |  |  |
| --- | --- | --- |
| Исходные данные | Тестируемый класс | Ожидаемый результат |
| 1 | 2 | 3 |
| IAppliances a; | IAppliances | Попытка создания объекта АТД выдаст ошибку |
| Appliances b("b", Firm::Undefined, "B", 10000); | Appliances | Попытка создания объекта АТД выдаст ошибку |
| Refrigerator c("c", Firm::Бирюса, "C", 20000, -18, true) | Refrigerator | Объект «Холодильника» создан успешно |
| Microwave d("d", Firm::Samsung, "D", 30000, -700) | Microwave | Объект «Микроволновки» создан успешно |
| Compact\_electric\_stove e("e", Firm::Midea, "E", 4000, 2000, 8) | Compact\_electric\_stove | Объект «Компактной электроплиты» создан успешно |
| Bread\_maker f("f", Firm::Dexp, "F", 5000, 710, 12, 5) | Bread\_maker | Объект «Хлебопечки» создан успешно |
| Electric\_stove g("g", Firm::Gefest, "G", -60000, 7400, 65, 8, 4, -250) | Electric\_stove | Объект «Электроплиты» создан успешно |
| Container\_IAppliances<IAppliances>h | Container\_IAppliances | Контейнер указателей АТД создан успешно |
| Vector\_IAppliances<Refrigerator, Container\_IAppliances  <IAppliances>>v\_refrigerator | Vector\_IAppliances | Контейнер объектов произ-водных классов создан |

Для тестирования функций были подобраны тестовые данные в табл. 20.

Таблица 20

Тестовые данные

|  |  |  |
| --- | --- | --- |
| Исходные данные | Тестируемый модуль | Ожидаемый результат |
| std::cout << list.  return\_object(0)->get\_name();  std::cout << v\_refrigerator.  return\_object(0)->get\_name(); | Vector\_IAppliances::  return\_object,  Container\_IAppliances::  return\_object | Результаты работы обоих функций должны быть одинаковы |
| list.output\_in\_file();  list.print(); | Container\_IAppliances::  output\_in\_file | Должна производиться загрузка данных из контейнера в файл |
| v\_refrigerator.add\_one\_to\_list  (&list, new Refrigerator (name1.c\_str(), Firm1, series1, price, min\_temp, Freezer1)); | Container\_IAppliances::  add\_one\_to\_list | Должен создаваться объект в векторе, а ссылка на него добавляться в лист |
| list.search\_price\_is\_less\_than  (10000); | Container\_IAppliances::  search\_price\_is\_less\_than | Должен производится вывод товаров дешевле 10000 |
| if (!list.empty()) list.print();  else cout << "Список пуст\n"; | Container\_IAppliances::  empty | Отсутствие данных в листе должно вернуть сообщение, а наличие элементов - вывести их |
| list.print();  list.sort\_firm(Firm1);  list.print(); | Container\_IAppliances::  sort\_firm | Производится вывод данных в разном порядке |
| list. search\_firm (Firm1);  list.print(); | Container\_IAppliances::  search\_firm | Производится вывод товаров выбранной компании, остальные товары игнорируются |

Таблица 21

Протокол тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Название теста | Класс | Проводил | Результаты |
| 1 | 2 | 3 | 4 | 5 |
| 30.04.24 | Создание объекта АТД | IAppliances | Микуцких Г. А. | Успех |
| 30.04.24 | Создание объекта АТД | Appliances | Микуцких Г. А. | Успех |

Продолжение табл. 21

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 30.04.24 | Создание объекта «Холодильника» | Refrigerator | Микуцких Г. А. | Успех |
| 30.04.24 | Создание объекта «Микроволновки» | Microwave | Микуцких Г. А. | Успех |
| 30.04.24 | Создание объекта «Компактной электроплиты» | Compact\_electric\_stove | Микуцких Г. А. | Успех |
| 30.04.24 | Создание объекта «Хлебопечки» | Bread\_maker | Микуцких Г. А. | Успех |
| 30.04.24 | Создание объекта «Электроплиты» | Electric\_stove | Микуцких Г. А. | Успех |
| 10.05.24 | Создание списка указателей | Container\_IAppliances | Микуцких Г. А. | Успех |
| 10.05.24 | Создание вектора объектов | Vector\_IAppliances | Микуцких Г. А. | Успех |

Таблица 22

Протокол тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Название теста | Модуль | Проводил | Результаты |
| 10.05.24 | Возвращение адреса первых элементов вектора и списка | Vector\_IAppliances::  return\_object, Container\_  IAppliances::return\_object | Микуцких Г. А. | Успех |
| 10.05.24 | Запись в файл элементов списка | Container\_IAppliances::  output\_in\_file | Микуцких Г. А. | Успех |
| 10.05.24 | Создание объекта в векторе и запись адреса в список | Container\_IAppliances::  add\_one\_to\_list | Микуцких Г. А. | Успех |
| 10.05.24 | Поиск товаров по цене не больше выбранной | Container\_IAppliances::  search\_price\_is\_less\_than | Микуцких Г. А. | Успех |
| 10.05.24 | Проверка списка на пустоту | Container\_IAppliances::  empty | Микуцких Г. А. | Успех |
| 10.05.24 | Сортировка списка по фирме | Container\_IAppliances::  sort\_firm | Микуцких Г. А. | Успех |
| 10.05.24 | Поиск всех товаров выбранной фирмы | Container\_IAppliances::  search\_firm | Микуцких Г. А. | Успех |

# Заключение

В ходе работы была разработана программа, содержащая структуру хранения и обработки объектов бытовой техники с возможностью работы пользователем; материал, полученный в ходе изучения дисциплины «Объектно-ориентированного программирования», освоен и закреплён.

# Список литературы

[1] Конова, Е.А. Объектно-ориентированное программирование с примерами на C++: учебное пособие / Е.А. Конова; под ред. Б.М. Суховилова. – Челябинск: Издательский центр ЮУрГУ, 2019. – 161 с.

# Приложение 1

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

­­ Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Объектно-ориентированное программирование

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ,

д.т.н., профессор \_\_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С++

Техническое задание на курсовую работу

Листов 8

|  |  |
| --- | --- |
| Руководители | Шаханов Н.И. |
|  | Ф.И.О преподавателя |
| Исполнитель |  |
| студент | 1ПИб-02-1оп-22 |
|  | группа |
|  | Микуцких Г. А. |
|  | Фамилия, имя, отчество |

2024 год

Введение

Правильное понимание принципов объектно-ориентированного программирования (ООП) – это ключ к грамотному моделированию систем. ООП помогает создавать более надежные и масштабируемые программы, а хорошо спроектированная система – проще переносить модель в код и поддерживать его в будущем. Целью курсовой работы является освоение и закрепление базовых концепций ООП, разработав структуру хранения и обработки объектов бытовой техники.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Объектно-ориентированное программирование», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 5 февраля 2024 года.

Наименование темы разработки: «Объектно-ориентированное программирование на языке С++».

1. Назначение разработки

Освоение на практике и закрепление материала, полученного в ходе изучения дисциплины «Объектно-ориентированного программирования».

1. Требования к программе
2. Требования к функциональным характеристикам

В программе должны быть реализованы:

1. иерархия родственных типов, корневой класс которой абстрактный базовый класс (класс-интерфейс), для моделирования и обработки данных предметной области набором отложенных методов – полиморфная обработка родственных объектов;
2. обобщенный (void\*) контейнерный класс (базовый) и от него, используя закрытое наследование, производный класс – шаблон для хранения указателей на абстрактный базовый класс-интерфейс (линейным двусвязным списком);
3. функции обработки данных (сортировка и поиск по выбранным полям и задаваемым диапазонам значений, другие функции, в том числе перегруженные);
4. хранение объектов каждого производного класса реализовать очередью;
5. файловый ввод/вывод и ввод данных с клавиатуры, вывод на дисплей;
6. обработка различных исключительных ситуаций.

Дополнительные требования:

1. работа всех функций должна быть проверена и результаты проверки оформлены протоколом тестирования.
2. Требования к структуре программы

В структуре программы должны использоваться статический вектор для хранения объектов каждого производного класса и линейный двусвязный список для хранения указателей на абстрактный базовый класс-интерфейс.

1. Требования к надежности

Должна быть обеспечена защита от неправильных загружаемых данных пользователем и реализована обработка соответствующих исключений.

1. Условия эксплуатации

Предъявляются с условиями эксплуатации ПК пользователя.

1. Требования к составу и параметрам технических средств

ЭВМ должна соответствовать следующим параметрам:

* исходные данные и результаты представляются в виде текста и текстовых файлов;
* минимальные системные требования:

1. 64-разрядный (x64) процессор с тактовой частотой 1 ГГц или выше;
2. 2 ГБ ОЗУ, 128 ГБ свободного места на накопителе (HDD, SSD);
3. графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии;

* поддержка клавиатуры, компьютерной мыши, монитора разрешением 1920×1080 пикселей.

1. Требования к информационной и программной совместимости

Программа должна корректно работать на ОС Windows 10 и выше с установленной программой Microsoft Visual Studio 2022 или новее.

1. Требования к маркировке и упаковке

Требования не предъявляются.

1. Требования к транспортированию и хранению

Всем пользователям запрещается осуществлять действия под видом оригинального ПО, нарушающие статью 273 «Создание, использование и распространение вредоносных компьютерных программ».

1. Специальные требования

Для эффективной работы с программой рекомендуется наличие опыта нахождения в компьютерной и информационной средах.

1. Требование к программной документации
2. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку, которая содержит:

Титульный лист

Аннотацию

Оглавление

Введение

1. Объектно-ориентированный анализ предметной области

2. Проектирование класса

3. Логическая структура программы

4. Модульная структура программы

5. Тестирование программы

Заключение

Список литературы

Приложения:

1. Техническое задание

2. Программный код

3. Руководство пользователя

1. Технико-экономические показатели

Требования не предъявляются.

1. Требования к оформлению

Элементы курсовой работы оформлены в соответствии с табл. П1.1.

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Элемент | Требования |
| 1 | 2 |
| Документ | 1. Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б.  2. Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код.  3. Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |

Продолжение табл. П1.1

|  |  |
| --- | --- |
| 1 | 2 |
| Страницы | 1. Ориентация – книжная; отдельные страницы, при необходимости, альбомная.  2. Поля: верхнее, нижнее – по 2 см, левое – 3 см, правое – 1 см. |
| Абзацы | Межстрочный интервал – 1.5, перед и после абзаца – 0. |
| Шрифты | Кегль – 14. В таблицах шрифт 12. Шрифт программного кода – 8 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: Рис. Х. Название  В приложениях: Рис. ПX.X. Название |
| Таблицы | 1. Подписывается: над таблицей, выравнивание по правому: «Таблица Х».  2. В следующей строке по центру Название  3. Надписи в «шапке» (имена столбцов, полей) – по центру.  4. В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по правому. |

1. Стадии и этапы разработки

Курсовая работа и программа будут разрабатываться в течение следующих этапов (табл. П1.2):

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| 1 | 2 | 3 | 4 |
| Оформление технического задания | 09.03.2024 | Составлено техническое задание |  |
| Разработка иерархии классов в соответствии с вариантом | 10.03.2024 – 16.03.2024 | Разработана иерархия классов и диаграмма |  |
| Создание контейнерного класса, от него создать класс с шаблоном хранения указателей на класс-интерфейс | 16.03.2024 | Создан контейнерный класс и класс шаблонов хранения указателей на класс-интерфейс |  |

Продолжение табл. П1.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Создание структуры данных для хранения объектов производных классов | 17.03.2024 – 22.03.2024 | Создана структура данных для хранения объектов производных классов |  |
| Реализации функций обработки данных | 23.03.2024 – 19.04.2024 | Созданы функции обработки данных |  |
| Реализация файлового ввода/вывода и ввода данных с клавиатуры, вывод данных на дисплей | 20.04.2024 – 27.04.2024 | Для программы реализовано получение исходных данных с использованием файлового ввода/вывода и ввода с клавиатуры, вывода на дисплей |  |
| Тестирование | 30.04.2024 – 10.05.2024 | Отладка программы, доработка функционала |  |
| Оформление расчётно-поясни-тельной записки | 10.05.2024 – 30.05.2024 | Оформлена расчётно-пояснительная записка |  |

1. Порядок контроля и приемки

Курсовая работа будет оформляться в течение этапов в табл. П1.3.

Таблица П1.3

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки  контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| 1 | 2 | 3 | 4 |
| Сдача технического задания | 04.05.2024 | Техническое задание принято |  |

Продолжение табл. П1.3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Исправление ошибок | 04.05.2024 – 24.05.2024 | Функционал программы доработан, тестирование прошло успешно |  |
| Демонстрация программы | 25.05.2024 | Программа, реализующая поставленную задачу, принята |  |
| Сдача расчётно-пояснительной записки | 31.05.2024 | Расчётно-пояснительная записка принята |  |
| Защита курсовой работы | 31.05.2024 | Курсовая работа защищена |  |

# Приложение 2

Программный код

Файл IAppliances.h

#pragma once

#include "Firm.h"

#include <iostream>

#include <string>

#include <fstream>

//класс-интерфейс для бытовой техники

class IAppliances {

public:

virtual void set\_name(const char\*) = 0;

virtual void set\_firm(Firm) = 0;

virtual void set\_series(std::string) = 0;

virtual void set\_price(int) = 0;

virtual char\* get\_name() const = 0;

virtual Firm get\_firm() const = 0;

virtual std::string get\_series() const = 0;

virtual int get\_price() const = 0;

virtual void print()const = 0;

virtual void print\_to\_file(std::ofstream&)const = 0;

virtual std::string get\_class\_name() const = 0;

protected:

virtual ~IAppliances() {};

};

Файл Appliances.h

#pragma once

#include "IAppliances.h"

#include "Firm.h"

#include <iostream>

#include <string>

//Создание класса для бытовой техники, производный класса-интерфейса

class Appliances :virtual public IAppliances {

public:

Appliances(const char\*, Firm, std::string, int);

Appliances(const Appliances&);

~Appliances() { std::cout << "Вызов деструктора" << this << std::endl; delete[]name; }

void set\_name(const char\*);

void set\_firm(Firm);

void set\_series(std::string);

void set\_price(int);

char\* get\_name() const;

Firm get\_firm() const;

std::string get\_series() const;

int get\_price() const;

virtual void print()const;

virtual void print\_to\_file(std::ofstream&)const;

private:

char\* name;

std::string series;

int price;

Firm firm;

};

Файл Appliances.cpp

#include "Appliances.h"

Appliances::Appliances(const char\* name, Firm firm, std::string series, int price)

: series(series), firm(firm), price(price)

{

set\_name(name);

//std::cout << "Объект " << this << std::endl;

}

Appliances::Appliances(const Appliances& pt)

: Appliances(pt.name, pt.firm, pt.series, pt.price) {}

void Appliances::set\_name(const char\* name)

{

int x = strlen(name) + 1;

delete[] this->name;

this->name = new char[x];

strcpy\_s(this->name, x, name);

}

void Appliances::set\_firm(Firm firm) { this->firm = firm; }

void Appliances::set\_series(std::string series) { this->series = series; }

void Appliances::set\_price(int price)

{

if (price >= 0)

this->price = price;

else throw std::exception("\_negative\_price");

}

char\* Appliances::get\_name()const { return name; }

Firm Appliances::get\_firm()const { return firm; }

std::string Appliances::get\_series()const { return series; }

int Appliances::get\_price()const { return price; }

void Appliances::print()const {

std::cout << get\_class\_name() << " | " << get\_name() << " | " << get\_series() << " | "

<< get\_price() << " рублей" << " | Код фирмы: " << static\_cast<int>(get\_firm());

}

void Appliances::print\_to\_file(std::ofstream& out)const

{

out << get\_class\_name() << " | " << get\_name() << " | " << get\_series() << " | "

<< get\_price() << " рублей" << " | Код фирмы: " << static\_cast<int>(get\_firm());

}

Файл Container.h

#pragma once

#include <list>

class Container//обобщенный базовый класс-контейнер

{

public:

//добавление элемента

void push\_back(void\* object) { list.push\_back(object); }

//удаление

void erase(int n) {

if (n < list.size()) {

std::list<void\*>::const\_iterator it = Container::cbegin();

advance(it, n);

list.erase(it);

std::cout << "Файл " << n +1 << " удалён" << std::endl;

}

else std::cout << "Элемента не существует" << std::endl;

}

int size()const { return list.size(); }

bool empty()const { return list.empty(); }

void clear() { list.clear(); }

std::list<void\*>::const\_iterator cbegin()const { return list.cbegin(); }

std::list<void\*>::const\_iterator cend()const { return list.cend(); }

std::list<void\*>::iterator begin() { return list.begin(); }

std::list<void\*>::iterator end() { return list.end(); }

private:

std::list<void\*> list;

};

Файл Container\_IAppliances.h

#pragma once

#include "Container.h"

#include "Firm.h"

//шаблонный класс-контейнер для хранения указателей на абстрактный базовый класс-интерфейс

template<class T>

class Container\_IAppliances : private Container

{

public:

//добавление элемента

void push\_back(T\* object) { Container::push\_back(object); }

//удаление элемента

void erase(int n) { Container::erase(n); }

//проверка пустоты

bool empty()const { return Container::empty(); }

//Очистка списка

void clear() { Container::clear(); }

//поиск по фирме

void search\_firm(Firm firm)

{

for (std::list<void\*>::const\_iterator it = Container::cbegin(); it != Container::cend(); ++it)

if (((T\*)(\*it))->get\_firm() == firm)

((T\*)(\*it))->print(), std::cout << std::endl;

}

//сортировка по цене

void search\_price\_is\_less\_than(int max\_price)

{

for (std::list<void\*>::const\_iterator it = Container::cbegin(); it != Container::cend(); ++it)

if (((T\*)(\*it))->get\_price() <= max\_price)

((T\*)(\*it))->print(), std::cout << std::endl;

}

//сортировка по фирме

void sort\_firm(Firm firm)

{

std::list<void\*>::iterator it\_up = Container::begin();

for (std::list<void\*>::iterator it = ++Container::begin(); it != Container::end(); ++it)

if (((T\*)(\*it))->get\_firm() == firm) {

while (((T\*)(\*it\_up))->get\_firm() == firm)

it\_up++;

std::swap(\*it\_up, \*it);

}

}

//адрес объекта

T\* return\_object(int n)const

{

std::list<void\*>::const\_iterator it = Container::cbegin();

advance(it, n);

return (T\*)\*it;

}

//вывод всех элементов из контейнера

void print()const

{

int i = 0;

for (std::list<void\*>::const\_iterator it = Container::cbegin(); it != Container::cend(); ++it) {

std::cout << ++i << ". ";

((T\*)(\*it))->print();

std::cout << std::endl;

}

}

void output\_in\_file()const

{

std::ofstream fout("output.txt");

for (std::list<void\*>::const\_iterator it = Container::cbegin(); it != Container::cend(); ++it)

((T\*)(\*it))->print\_to\_file(fout), fout << std::endl;

fout.close();

}

}; }};

Файл Vector\_IAppliances.h

#pragma once

#include <vector>

//шаблонный класс-контейнер для хранения объектов

template<class T, class TCA>

class Vector\_IAppliances

{

public:

void add\_one\_to\_list(TCA\* list, T\* object)

{

vector\_IAppliances.push\_back(object);

list->push\_back(vector\_IAppliances[vector\_IAppliances.size() - 1]);

}

void add\_from\_vector\_to\_list(TCA\* list, int n)

{

list->push\_back(vector\_IAppliances[n]);

}

void push\_back(T\* object) { vector\_IAppliances.push\_back(object); }

void del\_object(int n)

{

typename std::vector<T\*>::iterator it = vector\_IAppliances.begin();

advance(it, n);

vector\_IAppliances.erase(it);

}

T\* return\_object(int n)const

{

try {

if (n < vector\_IAppliances.size())

return vector\_IAppliances[n];

else throw "\_index\_outside\_size\_of\_vector";

}

catch (const char\* error\_message) {

if (error\_message == "\_index\_outside\_size\_of\_vector") {

std::cerr << error\_message << ": Выход за пределы индекса\n";

return vector\_IAppliances[vector\_IAppliances.size() - 1];

}}}

private:

std::vector<T\*> vector\_IAppliances;

};

Файл Refrigerator.h

#pragma once

#include "Appliances.h"

class Refrigerator : public Appliances {

public:

Refrigerator(const char\* name, Firm firm, std::string series, int price, int min\_temp, bool freezer)

: Appliances(name, firm, series, price), min\_temp(min\_temp), freezer(freezer) {};

void set\_min\_temp(int min\_temp) { this->min\_temp = min\_temp; }

void set\_freezer(int freezer) { this->freezer = freezer; }

std::string get\_freezer()const { return freezer ? "да" : "нет"; }

int get\_min\_temp()const { return min\_temp; }

virtual void print()const {

Appliances::print();

std::cout << " | " << "Минимальная температура: " << get\_min\_temp() << "C"

<< " | " << "Наличие морозильной камеры: " << get\_freezer();

}

virtual void print\_to\_file(std::ofstream& out)const {

Appliances::print\_to\_file(out);

out << " | " << "Минимальная температура: " << get\_min\_temp() << "C"

<< " | " << "Наличие морозильной камеры: " << get\_freezer();

}

protected:

virtual std::string get\_class\_name() const {

return "Холодильник";

}

int min\_temp;

bool freezer;

};

Файл Microwave.h

#pragma once

#include "Appliances.h"

class Microwave : public Appliances {

public:

Microwave(const char\* name, Firm firm, std::string series, int price, int power)

: Appliances(name, firm, series, price), power(power) {};

void set\_power(int power) {

if (power >= 0)

this->power = power;

else throw std::exception("\_negative\_wattage");

}

int get\_power()const { return power; }

virtual void print()const {

Appliances::print();

try {

if (get\_power() >= 0)

std::cout << " | " << get\_power() << "W";

else throw "\_negative\_wattage";

}

catch (const char\* error\_message) {

if (error\_message == "\_negative\_wattage")

std::cerr << " | " << error\_message << ": " << get\_power() << "W";

}

}

virtual void print\_to\_file(std::ofstream& out)const {

Appliances::print\_to\_file(out);

out << " | " << get\_power() << "W";

}

protected:

virtual std::string get\_class\_name() const {

return "Микроволновка";

}

int power;

};

Файл Compact\_electric\_stove.h

#pragma once

#include "Microwave.h"

class Compact\_electric\_stove : public Microwave {

public:

Compact\_electric\_stove(const char\* name, Firm firm, std::string series, int price, int power, int mumber\_of\_modes)

: Microwave(name, firm, series, price, power), mumber\_of\_modes(mumber\_of\_modes) {};

void set\_mumber\_of\_modes(int mumber\_of\_modes) {

if (mumber\_of\_modes >= 0)

this->mumber\_of\_modes = mumber\_of\_modes;

else throw std::exception("\_negative\_mumber\_of\_modes");

}

int get\_mumber\_of\_modes()const { return mumber\_of\_modes; }

virtual void print()const {

Microwave::print();

try {

if (get\_mumber\_of\_modes() >= 0)

std::cout << " | " << get\_mumber\_of\_modes();

else throw "\_negative\_mumber\_of\_modes";

}

catch (const char\* error\_message) {

if (error\_message == "\_negative\_mumber\_of\_modes")

std::cerr << " | " << error\_message << ": " << get\_mumber\_of\_modes();

}

}

virtual void print\_to\_file(std::ofstream& out)const {

Appliances::print\_to\_file(out);

out << " | " << get\_mumber\_of\_modes();

}

protected:

virtual std::string get\_class\_name() const {

return "Компактная электрическая плита";

}

int mumber\_of\_modes;

};

Файл Bread\_maker.h

#pragma once

#include "Compact\_electric\_stove.h"

class Bread\_maker : public Compact\_electric\_stove {

public:

Bread\_maker(const char\* name, Firm firm, std::string series, int price, int power, int mumber\_of\_modes, int volume)

: Compact\_electric\_stove(name, firm, series, price, power, mumber\_of\_modes), volume(volume) {};

void set\_volume(int volume) {

if (volume >= 0)

this->volume = volume;

else throw std::exception("\_negative\_volume");

}

int get\_volume()const { return volume; }

virtual void print()const {

Compact\_electric\_stove::print();

try {

if (get\_volume() >= 0)

std::cout << " | " << get\_volume() << "L";

else throw "\_negative\_volume";

}

catch (const char\* error\_message) {

if (error\_message == "\_negative\_volume")

std::cerr << " | " << error\_message << ": " << get\_volume() << "L";

}

}

virtual void print\_to\_file(std::ofstream& out)const {

Appliances::print\_to\_file(out);

out << " | " << get\_volume() << "L";

}

protected:

virtual std::string get\_class\_name() const {

return "Хлебопечь";}

int volume;

};

Файл Electric\_stove.h

#pragma once

#include "Bread\_maker.h"

class Electric\_stove : public Bread\_maker {

public:

Electric\_stove(const char\* name, Firm firm, std::string series, int price, int power, int mumber\_of\_modes, int volume, int number\_of\_burners, int max\_temperature)

: Bread\_maker(name, firm, series, price, power, mumber\_of\_modes, volume), number\_of\_burners(number\_of\_burners), max\_temperature(max\_temperature) {};

void set\_max\_temperature(int max\_temperature) { this->max\_temperature = max\_temperature; }

void set\_number\_of\_burners(int number\_of\_burners) {

if (number\_of\_burners >= 0)

this->number\_of\_burners = number\_of\_burners;

else throw std::exception("\_negative\_number\_of\_burners");

}

int get\_max\_temperature()const { return max\_temperature; }

int get\_number\_of\_burners()const { return number\_of\_burners; }

virtual void print()const {

Bread\_maker::print();

try {

if (get\_number\_of\_burners() >= 0)

std::cout << " | " << get\_number\_of\_burners();

else throw "\_negative\_temperature";

}

catch (const char\* error\_message) {

if (error\_message == "\_negative\_temperature")

std::cerr << " | " << error\_message << ": " << get\_max\_temperature() << "C";

}

try {

if (get\_max\_temperature() >= 0)

std::cout << " | " << get\_max\_temperature() << "C";

else throw "\_negative\_number\_of\_burners";

}

catch (const char\* error\_message) {

if (error\_message == "\_negative\_number\_of\_burners")

std::cerr << " | " << error\_message << ": " << get\_number\_of\_burners();

}

}

virtual void print\_to\_file(std::ofstream& out)const {

Appliances::print\_to\_file(out);

out << " | " << get\_number\_of\_burners() << " | " << get\_max\_temperature() << "C";

}

protected:

virtual std::string get\_class\_name() const {

return "Электрическая плита";

}

int number\_of\_burners;

int max\_temperature;

};

Файл Source.cpp

#pragma once

#include "Appliances.h"

#include "Container\_IAppliances.h"

#include "Vector\_IAppliances.h"

#include "Refrigerator.h"

#include "Microwave.h"

#include "Compact\_electric\_stove.h"

#include "Bread\_maker.h"

#include "Electric\_stove.h"

#include <iostream>

#include <string>

#include <fstream>

#include <windows.h>

int main() {

setlocale(LC\_ALL, "ru");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int S = 0, s = 0;

int num, price, power, number\_of\_modes, volume, number\_of\_burners, max\_temperature, min\_temp;

std::string name1, series1, series2, price1, firm1, min\_temp1, freezer1, power1, number\_of\_modes1, volume1, number\_of\_burners1, max\_temperature1;

Firm Firm1 = Firm::Undefined;

bool Freezer1;

Container\_IAppliances<IAppliances>list;

Vector\_IAppliances<Refrigerator, Container\_IAppliances<IAppliances>> v\_refrigerator;

Vector\_IAppliances<Compact\_electric\_stove, Container\_IAppliances<IAppliances>> v\_compact\_electric\_stove;

Vector\_IAppliances<Bread\_maker, Container\_IAppliances<IAppliances>> v\_bread\_maker;

Vector\_IAppliances<Microwave, Container\_IAppliances<IAppliances>> v\_microwave;

Vector\_IAppliances<Electric\_stove, Container\_IAppliances<IAppliances>> v\_electric\_stove;

std::ifstream fin("input.txt");

while (fin.is\_open())

{

switch (S)

{

case 0:

std::cout << "\nВыберите режим работы:\n1. Чтение из файла\n2. Вывод данных\n3. Сохранение данных в файл.\n4. Сортировка по товарам от ...\n5. Поиск товаров от компании...";

std::cout << "\n6. Поиск товаров дешевле...\n7. Удалить товар по номером...\n8. Изменить данные о товаре\n9. Очистить лист\n10. Создать товар\n11. Конец работы\n";

std::cin >> s;

std::cout << std::endl;

if (s == 1 && !fin.eof()) S = 1;//чтение из файла

else if (s == 1 && fin.eof()) fin.seekg(0), S = 1;//чтение из файла

else if (s == 2) S = 10;//вывод

else if (s == 3) {//запись в файл

list.output\_in\_file();

std::cout << "\nЗапись завершена в файл 'output.txt'\n";

}

else if (s == 4) S = 7;//сортировка по фирме

else if (s == 5) S = 8;//поиск товаров компании

else if (s == 6) S = 9;//поиск товаров дешевле

else if (s == 7) S = 11;//удалить товар под номером

else if (s == 8) S = 12;//изменить товар

else if (s == 9) S = 20;//очистка списка

else if (s == 10) S = 13;

else if (s == 11) fin.close();

else std::cout << "Ошибка: неверный режим работы\n";

break;

case 1:

if (fin.eof()) {

S = 0;

break;

}

std::getline(fin, name1);

if (name1 == "Холодильник") S = 2;

else if (name1 == "Микроволновка") S = 3;

else if (name1 == "Компактная электроплита") S = 4;

else if (name1 == "Хлебопечь") S = 5;

else if (name1 == "Электроплита") S = 6;

else {

std::cout << "Ошибка: некорректный файл\n";

fin.close();

list.print();

break;

}

std::getline(fin, name1);

std::getline(fin, firm1);

std::getline(fin, series1);

std::getline(fin, price1);

price = std::stoi(price1);

if (firm1 == "Бирюса") Firm1 = Firm::Бирюса;

else if (firm1 == "Atlant") Firm1 = Firm::Atlant;

else if (firm1 == "Dexp") Firm1 = Firm::Dexp;

else if (firm1 == "Gefest") Firm1 = Firm::Gefest;

else if (firm1 == "Midea") Firm1 = Firm::Midea;

else if (firm1 == "Samsung") Firm1 = Firm::Samsung;

else if (firm1 == "Weissgauff") Firm1 = Firm::Weissgauff;

else Firm1 = Firm::Undefined;

break;

case 2:

std::getline(fin, min\_temp1);

min\_temp = std::stoi(min\_temp1);

std::getline(fin, freezer1);

if (freezer1 == "true") Freezer1 = true;

else Freezer1 = false;

v\_refrigerator.add\_one\_to\_list(&list, new Refrigerator(name1.c\_str(), Firm1, series1, price, min\_temp, Freezer1));

S = 1;

break;

case 3:

std::getline(fin, power1);

power = std::stoi(power1);

v\_microwave.add\_one\_to\_list(&list, new Microwave(name1.c\_str(), Firm1, series1, price, power));

S = 1;

break;

case 4:

std::getline(fin, power1);

power = std::stoi(power1);

std::getline(fin, number\_of\_modes1);

number\_of\_modes = std::stoi(number\_of\_modes1);

v\_compact\_electric\_stove.add\_one\_to\_list(&list, new Compact\_electric\_stove(name1.c\_str(), Firm1, series1, price, power, number\_of\_modes));

S = 1;

break;

case 5:

std::getline(fin, power1);

power = std::stoi(power1);

std::getline(fin, number\_of\_modes1);

number\_of\_modes = std::stoi(number\_of\_modes1);

std::getline(fin, volume1);

volume = std::stoi(volume1);

v\_bread\_maker.add\_one\_to\_list(&list, new Bread\_maker(name1.c\_str(), Firm1, series1, price, power, number\_of\_modes, volume));

S = 1;

break;

case 6:

std::getline(fin, power1);

power = std::stoi(power1);

std::getline(fin, number\_of\_modes1);

number\_of\_modes = std::stoi(number\_of\_modes1);

std::getline(fin, volume1);

volume = std::stoi(volume1);

std::getline(fin, number\_of\_burners1);

number\_of\_burners = std::stoi(number\_of\_burners1);

std::getline(fin, max\_temperature1);

max\_temperature = std::stoi(max\_temperature1);

v\_electric\_stove.add\_one\_to\_list(&list, new Electric\_stove(name1.c\_str(), Firm1, series1, price, power, number\_of\_modes, volume, number\_of\_burners, max\_temperature));

S = 1;

break;

case 7:

std::cout << "\n1. Undefined\n2. Samsung\n3. Dexp\n4. Atlant\n5. Midea";

std::cout << "\n6. Gefest\n7. Бирюса\n8. Weissgauff\nВыберите компанию: ";

std::cin >> s;

if (s == 1) Firm1 = Firm::Undefined;

else if (s == 2) Firm1 = Firm::Samsung;

else if (s == 3) Firm1 = Firm::Dexp;

else if (s == 4) Firm1 = Firm::Atlant;

else if (s == 5) Firm1 = Firm::Midea;

else if (s == 6) Firm1 = Firm::Gefest;

else if (s == 7) Firm1 = Firm::Бирюса;

else if (s == 8) Firm1 = Firm::Weissgauff;

else {

std::cout << "\nОшибка, неверная компания";

break;

}

list.sort\_firm(Firm1);

list.print();

S = 0;

break;

case 8:

std::cout << "\n1. Undefined\n2. Samsung\n3. Dexp\n4. Atlant\n5. Midea";

std::cout << "\n6. Gefest\n7. Бирюса\n8. Weissgauff\nВыберите компанию: ";

std::cin >> s;

if (s == 1) Firm1 = Firm::Undefined;

else if (s == 2) Firm1 = Firm::Samsung;

else if (s == 3) Firm1 = Firm::Dexp;

else if (s == 4) Firm1 = Firm::Atlant;

else if (s == 5) Firm1 = Firm::Midea;

else if (s == 6) Firm1 = Firm::Gefest;

else if (s == 7) Firm1 = Firm::Бирюса;

else if (s == 8) Firm1 = Firm::Weissgauff;

else {

std::cout << "\nОшибка, неверная компания";

break;

}

std::cout << std::endl;

list.search\_firm(Firm1);

S = 0;

break;

case 9:

std::cout << "\nТовары дешевле: ";

std::cin >> s;

list.search\_price\_is\_less\_than(s);

std::cout << std::endl;

S = 0;

break;

case 10:

if (!list.empty())

list.print();

else std::cout << "Список пуст\n";

std::cout << std::endl;

S = 0;

break;

case 11:

if (!list.empty()) {

list.print();

std::cout << "\nУдалить элемент под номером: ";

std::cin >> s;

list.erase(s - 1);

}

else std::cout << "Список пуст\n";

S = 0;

break;

case 12:

if (!list.empty()) {

std::cout << std::endl;

list.print();

std::cout << "\nИзменить товар под номером: ";

std::cin >> num;

std::cout << "\nИзменить:\n1. Имя\n2. Фирму\n3. Серию\n4. Цену\n";

std::cin >> s;

try {

if (s == 1) {

std::cout << "\nИзменить имя на: ";

std::cin >> name1;

std::getline(std::cin, series1);

list.return\_object(num - 1)->set\_name((name1 + series1).c\_str());

}

else if (s == 2) {

std::cout << "\n1. Undefined\n2. Samsung\n3. Dexp\n4. Atlant\n5. Midea";

std::cout << "\n6. Gefest\n7. Бирюса\n8. Weissgauff\nВыберите компанию: ";

std::cout << "\nИзменить фирму на: ";

std::cin >> s;

if (s == 1) Firm1 = Firm::Undefined;

else if (s == 2) Firm1 = Firm::Samsung;

else if (s == 3) Firm1 = Firm::Dexp;

else if (s == 4) Firm1 = Firm::Atlant;

else if (s == 5) Firm1 = Firm::Midea;

else if (s == 6) Firm1 = Firm::Gefest;

else if (s == 7) Firm1 = Firm::Бирюса;

else if (s == 8) Firm1 = Firm::Weissgauff;

list.return\_object(num - 1)->set\_firm(Firm1);

}

else if (s == 3) {

std::cout << "\nИзменить серию на: ";

std::cin >> name1;

std::getline(std::cin, series1);

list.return\_object(num - 1)->set\_series((name1 + series1).c\_str());

}

else if (s == 4) {

std::cout << "\nИзменить цену на: ";

std::cin >> price;

list.return\_object(num - 1)->set\_price(price);

}

else {

std::cout << "\nНеверная операция";

break;

}

list.print();

S = 0;

break;

}

catch (const std::exception& error\_message) {

std::cerr << error\_message.what() << ", действие отменено" << std::endl;

}

catch (...) {

std::cerr << "\_unexpected exception" << std::endl;

}

}

else std::cout << "Список пуст\n";

break;

case 13:

std::cout << "\n1. Холодильник\n2. Микроволновка\n3. Компактная электроплита\n4. Хлебопечь\n5. Электроплита\nСоздать товар категории: ";

std::cin >> s;

if (s == 1) name1 = "Холодильник", S = 14;

else if (s == 2) name1 = "Микроволновка", S = 15;

else if (s == 3) name1 = "Компактная электроплита", S = 16;

else if (s == 4) name1 = "Хлебопечь", S = 17;

else if (s == 5) name1 = "Электроплита", S = 18;

else {

std::cout << "\nОшибка: некорректный номер\n";

break;

}

std::cout << "\n1. Undefined\n2. Samsung\n3. Dexp\n4. Atlant\n5. Midea";

std::cout << "\n6. Gefest\n7. Бирюса\n8. Weissgauff\nВыберите компанию: ";

std::cin >> s;

if (s == 1) Firm1 = Firm::Undefined;

else if (s == 2) Firm1 = Firm::Samsung;

else if (s == 3) Firm1 = Firm::Dexp;

else if (s == 4) Firm1 = Firm::Atlant;

else if (s == 5) Firm1 = Firm::Midea;

else if (s == 6) Firm1 = Firm::Gefest;

else if (s == 7) Firm1 = Firm::Бирюса;

else if (s == 8) Firm1 = Firm::Weissgauff;

else {

std::cout << "\nОшибка: некорректный номер\n";

break;

}

std::cout << "\nВведите серию: ";

std::cin >> series1;

std::getline(std::cin, series2);

series1 += series2;

std::cout << "\nВведите цену: ";

std::cin >> price;

break;

case 14:

std::cout << "\nУ холодильника есть морозилка?\n1. Да\n2. Нет\n";

std::cin >> s;

if (s == 1) Freezer1 = true;

else if (s == 2)Freezer1 = false;

else {

std::cout << "\nОшибка: некорректный номер\n";

break;

}

std::cout << "\nВведите минимальную температуру: ";

std::cin >> min\_temp1;

v\_refrigerator.add\_one\_to\_list(&list, new Refrigerator(name1.c\_str(), Firm1, series1, price, min\_temp, Freezer1));

S = 0;

break;

case 15:

std::cout << "\nВведите потребляемую мощность: ";

std::cin >> power;

v\_microwave.add\_one\_to\_list(&list, new Microwave(name1.c\_str(), Firm1, series1, price, power));

S = 0;

break;

case 16:

std::cout << "\nВведите потребляемую мощность: ";

std::cin >> power;

std::cout << "\nВведите количество режимов: ";

std::cin >> number\_of\_modes;

v\_compact\_electric\_stove.add\_one\_to\_list(&list, new Compact\_electric\_stove(name1.c\_str(), Firm1, series1, price, power, number\_of\_modes));

S = 0;

break;

case 17:

std::cout << "\nВведите потребляемую мощность: ";

std::cin >> power;

std::cout << "\nВведите количество режимов: ";

std::cin >> number\_of\_modes;

std::cout << "\nВведите объём: ";

std::cin >> volume;

v\_bread\_maker.add\_one\_to\_list(&list, new Bread\_maker(name1.c\_str(), Firm1, series1, price, power, number\_of\_modes, volume));

S = 0;

break;

case 18:

std::cout << "\nВведите потребляемую мощность: ";

std::cin >> power;

std::cout << "\nВведите количество режимов: ";

std::cin >> number\_of\_modes;

std::cout << "\nВведите объём: ";

std::cin >> volume;

std::cout << "\nВведите количество конфорок: ";

std::cin >> number\_of\_burners;

std::cout << "\nВведите максимальную температуру: ";

std::cin >> max\_temperature;

v\_electric\_stove.add\_one\_to\_list(&list, new Electric\_stove(name1.c\_str(), Firm1, series1, price, power, number\_of\_modes, volume, number\_of\_burners, max\_temperature));

S = 0;

break;

case 20:

list.clear();

std::cout << "Список очищен\n";

S = 0;

break;

}

}

std::cout << std::endl;

system("pause");

return 0;

}

# Приложение 3

Руководство пользователя

Общие сведения о программе

В программе реализована структура хранения и обработки данных для товаров бытовой техники.

Описание установки

Установки не требует. Программа работает «из коробки».

Описание запуска

Для запуска необходимо нажать на ярлык с именем «KursOOP». После запуска программа встречает окном с перечислением доступных возможностей.

Инструкции по работе

Работа с программой производится в консоли. Доступные операции:

1. Чтение из файла;
2. Вывод данных;
3. Сохранение данных в файл;
4. Сортировка по товарам от компании;
5. Поиск товаров от компании;
6. Поиск товаров дешевле…;
7. Удалить товар по номером;
8. Изменить данные о товаре;
9. Очистить лист;
10. Создать товар;
11. Конец работы.

Ввод номера операций или данных, указанных в последнем сообщении в консоли, осуществляет работу с данными товаров (рис. П3.1).

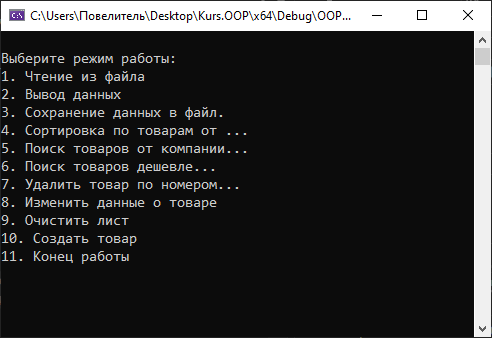


Рис. П3.1. Стартовое окно программы

Для того, чтобы воспользоваться файловым вводом, необходимо создать текстовый файл «input.txt» в корне программы, построчно ввести в него свойства товара и выбрать операцию «1. Чтение из файла». Для удобства заранее заготовлен шаблон, в котором указано какие типы товаров доступны для создания и их свойства. Пример вывода шаблона представлен на рис. П3.2. Пример вывода некоторых данных представлен на рис. П3.3.

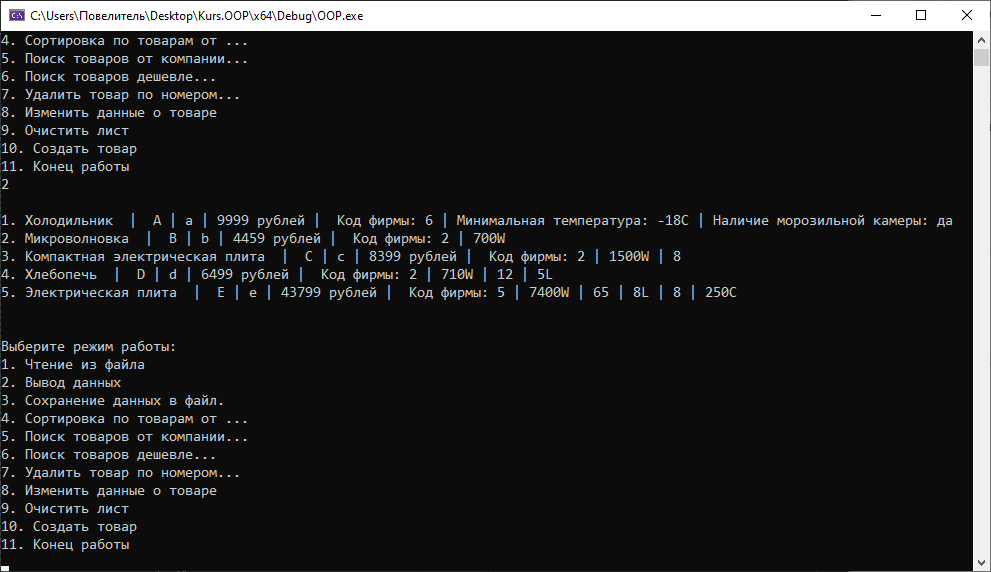


Рис. П3.2. Вывод шаблона

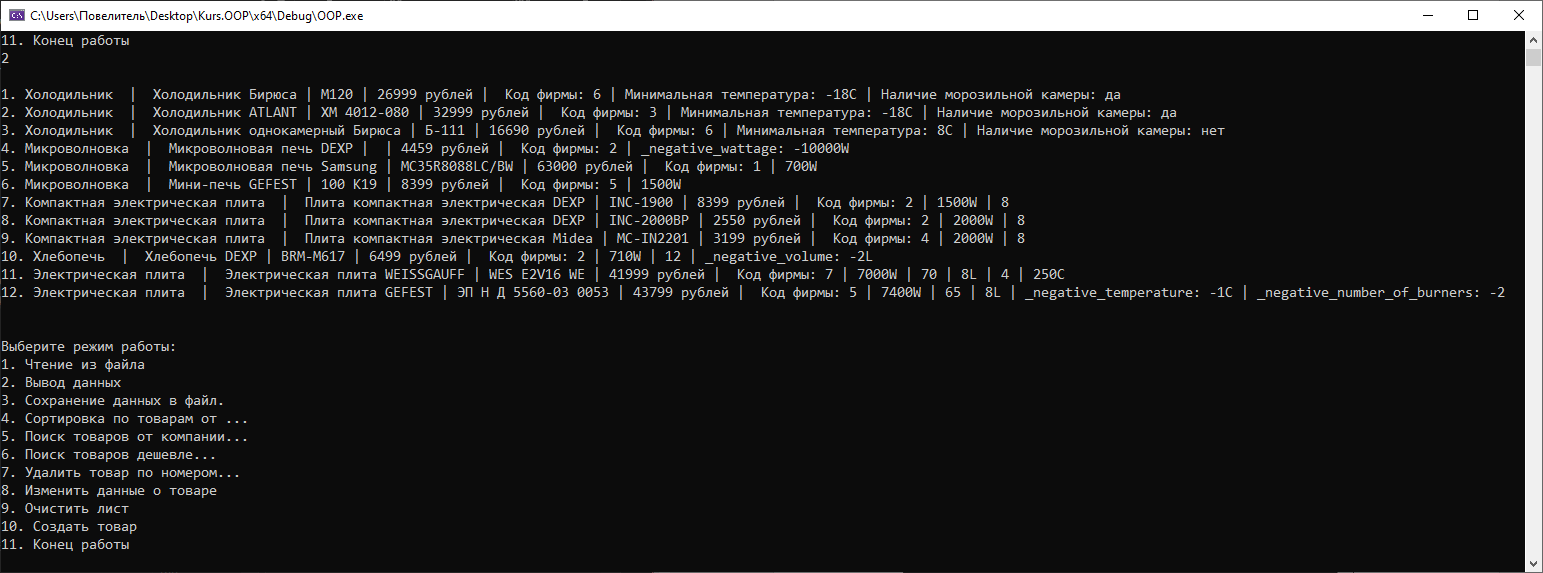


Рис. П3.3. Пример вывода всех товаров