МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Информационных технологий |
| Кафедра | Математическое и программное обеспечение ЭВМ |

КУРСОВАЯ РАБОТА

|  |
| --- |
| по дисциплине «Объектно-ориентированное программирование» |

|  |  |
| --- | --- |
| на тему | «Объектно-ориентированное программирование на языке C++» |
|  | |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-1оп-22 |
| направления подготовки (специальности) |
| 09.03.04, Программная инженерия |
| шифр, наименование |
|  |
| Харламов Денис Алексеевич |
| 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

Студент группы 1ПИб-02-1оп-22 Института информационных технологий, Харламов Денис Алексеевич, выполнил курсовую работу на тему "Объектно-ориентированное программирование на языке С++".

Целью проекта является создание иерархии типов с использованием абстрактных базовых классов, полиморфной обработки объектов, контейнерного класса и структур данных для хранения объектов. Программа позволяет проводить файловый ввод/вывод данных, ввод данных об объекте вручную, а также функции обработки данных, такие как сортировка и поиск по выбранным полям.

Оглавление

[Введение 4](#_Toc168033742)

[1. Объектно-ориентированный анализ предметной области 6](#_Toc168033743)

[2. Проектирование классов 8](#_Toc168033744)

[3. Логическая структура программы 13](#_Toc168033745)

[4. Модульная структура программы 14](#_Toc168033746)

[5. Тестирование программы 15](#_Toc168033747)

[Заключение 17](#_Toc168033748)

[Список литературы 18](#_Toc168033749)

[Приложение 1. Техническое задание 19](#_Toc168033750)

[Приложение 2. Текст программы 27](#_Toc168033751)

[Приложение 3. Руководство пользователя 35](#_Toc168033752)

[Приложение 4. Детальная диаграмма классов 38](#_Toc168033753)

# Введение

**Объектно-ориентированное программирование (ООП) - это методология разработки программного обеспечения, которая основана на концепции объектов и классов.**

**ООП рассматривает программу как набор взаимодействующих объектов. Каждый объект обладает своими уникальными характеристиками и поведением. Этот подход облегчает написание и понимание кода. Объект представляет собой экземпляр класса, существующий в памяти компьютера.**

**Объекты - это сущности с определенными свойствами и поведением, как правило, они являются экземплярами какого-то класса.**

**Класс - это пользовательский тип данных, содержащий свойства и методы, аналогичные типам данных, таким как String или Int. Основы ООП включают в себя принципы, определяющие этот подход:**

1. **Инкапсуляция;**
2. **Наследование;**
3. **Полиморфизм;**
4. **Абстракция.**

**Инкапсуляция представляет собой процесс отделения внутренней структуры объекта от его внешнего интерфейса. Для реализации инкапсуляции каждый класс должен иметь необходимые методы для работы с данными. Инкапсуляция связывает атрибуты класса с методами, обрабатывающими их. Согласно этому принципу, объект не должен иметь прямого доступа к атрибутам другого объекта.**

Для обеспечения принципа инкапсуляции каждый класс должен содержать все необходимые методы для работы с данными. Инкапсуляция подразумевает связь атрибутов класса с методами их обработки. Объекты не должны иметь прямого доступа к атрибутам других объектов в соответствии с этим принципом.

Наследование предполагает создание нового класса на основе уже существующего. Подкласс наследует свойства и методы суперкласса. Это помогает организовать объекты и классы, чтобы избежать рутинной работы по созданию множества элементов, и формировать классы с более общими характеристиками и функциями. Программист может последовательно наследовать от них функциональность.

Полиморфизм позволяет объектам принимать различные формы и обрабатываться как объекты одного класса. Это упрощает обработку разных типов информации с использованием одних и тех же методов. Полиморфизм также дает возможность переопределения методов в дочерних классах для обработки других типов информации или выполнения дополнительных действий.

Абстракция предполагает определение существенных характеристик объекта и игнорирование несущественных. Это позволяет создавать абстрактные классы, которые определяют общие свойства и поведение группы объектов, не утруждаясь деталями каждого объекта.

Класс-интерфейс (абстрактный базовый класс) – это класс, который не имеет данных, в нём прописаны только основные виртуальные функции, которые будут наследовать все потомки.

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

Цель курсовой работы заключается в создании программы для работы с объектами предметной области "Нагревательные приборы" с использованием принципов ООП. Проектирование включает создание иерархии классов предметной области, использование наследования и инкапсуляции, а также полиморфную обработку объектов данных классов. В работе описано функциональное назначение продукта, его логическая структура и основы работы программы.

# 1. Объектно-ориентированный анализ предметной области

В ходе выполнения курсовой работы необходимо разработать программный продукт для работы с объектами предметной области – «Нагревательные приборы». Необходимо разработать связанные родственные типы, относящиеся к данной области. Для достижения цели используется принципы наследования. Такое множество связанных между собой классов обычно называется иерархией классов. Обычно иерархию представляют в виде дерева, но так же бывают структуры с более общей структурой в виде графа.

В иерархии Нагревательные приборы (Heater/IHeater) является классом-интерфейсом или абстрактным базовым классом.

Нагревательные приборы – это специальные устройства, принцип действия которых основан на передаче теплоты от обогреваемой среды первичного теплоносителя.

Нагревательные приборы можно подразделить на:

* Герметичные;
* Негерметичные;
* Трубочно-электронные;

Для данной курсовой работы на основе этих групп были созданы соответствующие классы: Герметичные(SealedHeater), Негерметичные (UnsealedHeater) и Трубочно-электронные (Tub\_el\_heater).

Также создан обобщенный (void\*) контейнерный класс. Он хранит указатель void\*. Такой тип указателей может указывать на любую переменную, а значит все представленные классы, которые являются разными типами данных, можно хранить в одном контейнере, приведя их тип к void\*. Таким классом является void\_Lister. Однако для выполнения преобразования из void\* к типу класса-интерфейса и наоборот используется его наследник – класс Lister.

Диаграмма классов представлена на рис.1.

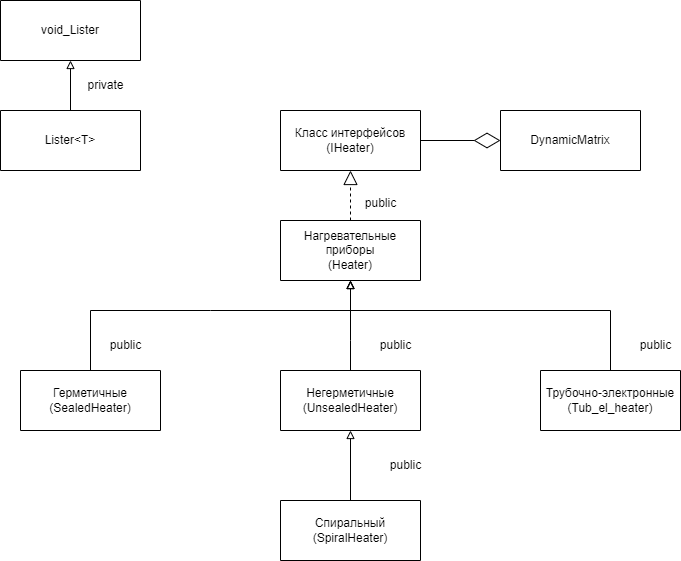


Рис. 1. Диаграмма классов

# 2. Проектирование классов

Классы нагревательных приборов поддерживают функции вывода информации в консоль и в файл, вместе с содержащейся в классе информацией в консоль и файл выводятся ещё и название классов, к которым относятся выводимые объекты, для более удобной подачи информации. Информация о всех функциях, которые поддерживаются классами приборов, находится в таблицах 1 – 9.

Таблица 1

Функции класса IHeater

|  |  |
| --- | --- |
| Функция | Назначение функции |
| virtual void setname(const string name) | Функция изменения параметра имени объекта |
| virtual void setprice(int price) | Функция изменения параметра цены |
| virtual void setpower(double power) | Функция изменения параметра мощности |
| virtual string getname() const | Функция получения параметра имени объекта |
| virtual int getprice() const | Функция получения параметра цены |
| virtual string getcolor() const | Функция получения параметра цвета |
| virtual double getpower() const | Функция получения параметра мощности |
| virtual void print() const | Функция вывода объекта в консоль |
| virtual void sfile(ofstream& f) | Функция сохранения объекта в файл |

Таблица 2

Функции класса Heater

|  |  |
| --- | --- |
| Функция | Назначение функции |
| Heater() | Конструктор по умолчанию |
| Heater(const string name, Color cl, int price, double power) | Конструктор с параметрами |
| Heater (const Heater& name) | Копирующий конструктор |

|  |  |
| --- | --- |
|  | Продолжение таблицы 2 |
| ~Heater () | Деструктор |
| string getname() | Функция получения параметра имени объекта |
| string getcolor() | Функция получения параметра цвета |
| int getprice() | Функция получения параметра цены |
| double getpower() | Функция получения параметра мощности |
| virtual void print\_header () const | Функции вывода имени класса в консоль |
| void setname(const string name) | Функция изменения параметра имени объекта |
| void setcolor(Color color) | Функция изменения параметра цвета |
| void setprice(int rice) | Функция изменения параметра цены |
| void setpower(double power) | Функция изменения параметра мощности |
| void print() const | Функция вывода данных объекта класса |
| virtual void s\_f(ofstream& f) | Функция сохранения объекта в файл |

Таблица 3

Функция класса SealedHeater

|  |  |
| --- | --- |
| Функция | Назначение функции |
| SealedHeater(const string name, Color cl, int price, double power, double temperature) | Конструктор с параметрами |
| virtual void print\_header () const | Функция вывода имени класса в консоль |
| double gettp() const | Функция получения параметра температуры |
| void settp(double tp) | Функция изменения параметра температуры |
| void print\_info() const | Функция вывода индивидуальных данных класса |
| virtual void sfile(ofstream& f) | Функция сохранения объекта в файл |

|  |  |
| --- | --- |
|  |  |

Таблица 4

Функции класса UnsealedHeater

|  |  |
| --- | --- |
| Функция | Назначение функции |
| UnsealedHeater(const string name, Color cl, int price, double power, float lenght) | Конструктор с параметрами |
| float getlenght() const | Функция получения параметра длины |
| void setlenght(float len) | Функция изменения параметра длины |
| virtual void print\_header () const | Функция вывода имени класса в консоль |
| void print\_info() const | Функция вывода индивидуальных данных класса |
| virtual void sfile(ofstream& f) | Функция сохранения объекта в файл |

Таблица 5

Функция класса Tub\_El\_Heater

|  |  |
| --- | --- |
| Функция | Назначение функции |
| Tub\_el\_Heater(const string name, Color cl, int price, double power, float thickness) | Конструктор с параметрами |
| float getthickness() const | Функция получения параметра толщины |
| void setthickness(float thick) | Функция изменения параметра толщины |
| void print\_info() const | Функция вывода индивидуальных данных класса |
| virtual void print\_header () const | Функция вывода имени класса в консоль |
| virtual void sfile(ofstream& f) | Функция сохранения объекта в файл |

Таблица 6

Функция класса SpiralHeater

|  |  |
| --- | --- |
| Функция | Назначение функции |
| SpiralHeater(const string name, Color cl, int price, double power, float lenght, bool el) | Конструктор с параметрами |
| void setel(bool el) | Функция получения параметра замкнутость контура |
| virtual void print\_header () const | Функция вывода имени класса в консоль |
| void print\_info() const | Функция вывода индивидуальных данных класса |
| virtual void s\_f(ofstream& f) | Функция сохранения объекта в файл |

Таблица 7

Функции класса TStatic\_Matrix

|  |  |
| --- | --- |
| Функция | Назначение функции |
| TemplateMtrx(int i, int j) | Конструктор с параметрами |
| ~TemplateMtrx() | Деструктор |
| void Matrix\_Add(T\* ConstrData) | Функция добавления элемента в матрицу по координатам (Автоматическое вычисление координат) |
| void del() | Удаление элемента из матрицы |
| virtual void print() const | Функция вывода матрицы в консоль |
| void find(int price) const | Поиск объектов матрицы по цене |
| void sortprice() | Сортировка объектов матрицы по цене |

Основной функциональный класс однонаправленного списка Lister. Данный класс наследует множество функций от класса VoidLister. Lister поддерживает функции добавления элемента в матрицу, сортировка элементов. Также пользователь может осуществлять поиск элементов по значениям параметров имени, цены и мощности. Все функции этих классов находится дальше в таблицах 8 – 9.

Таблица 8

Функции класса VoidLister

|  |  |
| --- | --- |
| Функция | Назначение функции |
| void add(void\* obj) | Функция добавления указателя на объект в конец списка |
| void remove() | Функция удаления объектов с конца списка |
| bool empty() | Функция проверки списка на пустоту |
| ~VoidLister() | Деструктор |

Таблица 9

Функции класса List

|  |  |
| --- | --- |
| Функция | Назначение функции |
| void add(T\* obj) | Функция добавления указателя на объект в конец списка |
| void remove() | Функция удаления объектов с конца списка |
| bool empty() | Функция проверки списка на пустоту |
| void Print() | Функция вывода списка в консоль |
| void save(ofstream& f) | Функция сохранения списка в файл |
| void sort\_price() | Функция сортировки по цене |
| void findpow(int power) | Функция поиска элементов по мощности |
| void findpr(int price) | Функция поиска элементов по цене |
| void findnm(string name) | Функция поиска элемента по названию объекта |

После разработки контекстной диаграммы родственных классов (рис.1) была создана Детальная диаграмма классов, полученная в ходе проектирования с учётом контейнеров, представлена в приложении 4.

# 3. Логическая структура программы

При запуске программы основной управляющий блок main, получающий данные, зависящие от запроса пользователя, вызывает соответствующий блок. Пользователь предоставляется возможность загрузить информацию из файла, далее вся информация, считанная из файла, будет распределена по контейнерам. Существует 4 матрицы, которые хранят объекты классов нагревательных приборов, также существует линейный список, хранящий указатель на объекты класса интерфейса IHeater. Эти контейнеры заполняются объектами/указателями на объекты. При создании новых нагревателей с помощью ручного ввода будет тот же результат.

Логическая структура программы представлена на рис.2.

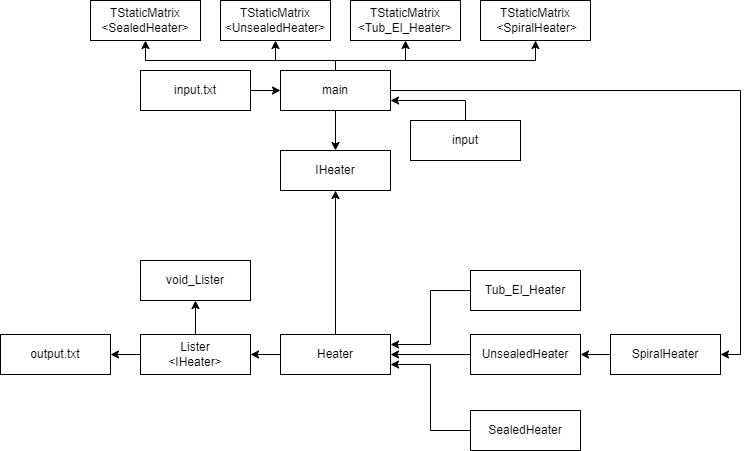


Рис. 2. Логическая диаграмма

# 4. Модульная структура программы

На рисунке 4 видно, что программа кроме основного модуля (main.cpp) обладает также вспомогательными модулями, которые вызываются в процессе работы. Многие модули, кроме main, IHeater, Lister, TStaticMatrix, и VoidLister состоят из двух файлов: заголовочного (.h – header) и основного (.cpp). При вызове такого блока сначала вызывается заголовочный файл, а после вызывается уже основной файл.

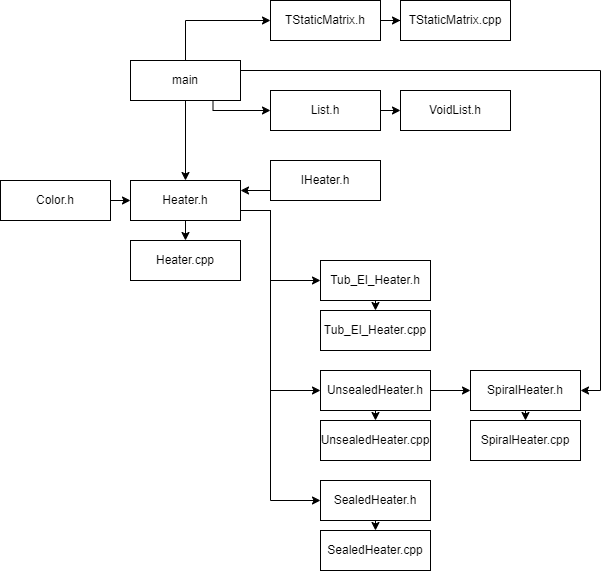


Рис. 3. Модульная диаграмма

# 5. Тестирование программы

В таблицах 10-12 представлены результаты тестирования программы.

Таблица 10

Протокол тестирования классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Класс | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 12.05.24 | SealedHeater.h | Харламов Д.А. | Создание объектов и действия с ними | Успех |
| 12.05.24 | UnsealedHeater.h | Харламов Д.А. | Создание объектов и действия с ними | Успех |
| 12.05.24 | Tub\_el\_heater.h | Харламов Д.А. | Создание объектов и действия с ними | Успех |
| 12.05.24 | SpiralHeater.h | Харламов Д.А. | Создание объектов и действие с ними | Успех |
| 12.05.24 | Lister.h | Харламов Д.А. | Cоздание списка, заполнение его объектами | Успех |
| 12.05.24 | TStaticMatrix.h | Харламов Д.А. | Создание матрицы, заполнение ее объектами | Успех |
| 12.05.24 | Lister.h | Харламов Д.А. | Сортировка списка и поиск элементов по нему | Успех |

Таблица 11

Протокол тестирования внешних функций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата  тестирования | Функция | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 12.05.24 | main | Харламов Д.А. | Ввод данных с помощью файла | Успех |
| Ручной ввод данных | Успех |
| Вывод данных | Успех |
| Сохранение данных в файле | Успех |
| Вывод данных | Успех |

Таблица 12

Протокол тестирования по техническому заданию

|  |  |  |  |
| --- | --- | --- | --- |
| Дата тестирования | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 16.05.24 | Белоус Г.П. | Проверка считывания данных из входного файла | Успех |
| 16.05.24 | Новиков И.С. | Проверка заполнения данных в контейнеры | Успех |
| 16.05.24 | Белоус Г.П. | Проверка сохранения данных объектов | Успех |
| 16.05.24 | Харламов Д.А. | Проверка заполнения данных в матрицу | Успех |
| 16.05.24 | Новиков И.С. | Проверка сортировки объектов | Успех |
| 16.05.24 | Новиков И.С. | Проверка поиска объектов | Успех |
| 16.05.24 | Белоус Г.П. | Проверка вывода данных в выходной файл | Успех |
| 16.05.24 | Белоус Г.П. | Проверка возможных исключений во время ручного ввода | Успех |

# Заключение

Курсовая работа была посвящена созданию программного обеспечения для работы с нагревательными приборами. В результате курсовой работы было разработано программное обеспечение для работы с объектами предметной области – «Нагревательные приборы». Разработанная программа умеет хранить информацию о приборах в двух форматах: в контейнерном классе и статической матрице. Данные могут быть загружены из файла или введены пользователем. Программа также позволяет сортировать приборы по цене и производить поиск по этому параметру. В ходе работы были реализованы механизмы обработки исключительных ситуаций.

# Список литературы

1. Ершов, Е.В. Методика и организация самостоятельной работы: учебное пособие [Текст] / Ершов Е.В., Виноградова Л.Н., Селивановских В.В. – Череповец: ЧГУ, 2015.

2. https://gb.ru/blog/obektno-orientirovannoe-programmirovanie/ [Электронный ресурс] – Основы объектно-ориентированного программирования.

3. https://ros-pipe.ru/tekh\_info/tekhnicheskie-stati/montazh-i-remont-vodosnabzheniya-zhilykh-domov/nagrevatelnye-pribory-naznachenie-vidy-trebovaniya/ [Электронный ресурс] - информация, посвященная нагревательным приборам.

# Приложение 1. Техническое задание

МИНОБРАНАУКИ 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. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2024 г.

«Объектно-ориентированное программирование на языке С ++»

Техническое задание на курсовую работу

Листов \_8\_

Руководитель: Шаханов Н.И.

*Ф.И.О. преподавателя*

Исполнитель: студент гр. 1ПИб–02–1оп–22

*Группа*

Харламов Денис Алексеевич

*Фамилия, имя, отчество*

2024 г.

Введение

Объектно-ориентированное программирование (ООП) — это подход к разработке программного обеспечения, который фокусируется на создании программ из взаимосвязанных "блоков" — объектов. Каждый объект представляет собой самостоятельную сущность, объединяющую данные и действия, которые можно с ней выполнять.

Преимущества ООП:

\* Структурированный код: ООП делает программы более понятными, организованными и легкими для понимания.

\* Модульность: Программы, созданные с использованием ООП, легко разбить на независимые модули, что упрощает их модификацию и поддержку.

\* Гибкость: ООП позволяет легко добавлять новые функции и изменять существующие, без нарушения работы всей программы.

\* Повторное использование кода: Разработанные объекты можно использовать в других проектах, что сокращает время и затраты на разработку.

В целом, ООП делает разработку программного обеспечения более эффективной, гибкой и позволяет создавать качественные продукты. Важность ООП в мире программирования заключается в том, что она позволяет разработчикам создавать сложные программы, состоящие из множества взаимосвязанных объектов, которые могут взаимодействовать друг с другом. Это способствует повышению производительности и эффективности разработки, а также упрощает поддержку и модификацию кода. Кроме того, ООП способствует повторному использованию кода, что позволяет сократить время и затраты на разработку новых программных продуктов. В целом, использование ООП помогает создавать более надежные, масштабируемые и удобные в обслуживании программы.

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: 9 марта 2024 года.

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

1. Назначение разработки

Главной задачей курсовой работы является закрепление полученных на лабораторных работах и лекциях навыков по дисциплине «Объектно-ориентированное программирование» и работа на высокоуровневом языке программирования C++.

1. Требования к программе

3.1. Требования к функциональным характеристикам

1. Разработать иерархию родственных типов, корневой класс которой абстрактный базовый класс (**класс-интерфейс)**, для моделирования и обработки данных предметной области набором отложенных методов - полиморфная обработка родственных объектов в соответствии с вариантом А18 «Нагревательные приборы».
2. Создать обобщенный (**void\***) контейнерный класс (базовый) и от него, используя закрытое наследование, производный класс – шаблон для хранения указателей на абстрактный базовый класс-интерфейс в соответствии с вариантом B3 «Линейный односвязный список».
3. Для хранения объектов каждого производного класса использовать структуру данных в соответствии с вариантом С3 «Статическая матрица».
4. Реализовать функции обработки данных (сортировка и поиск по выбранным полям и задаваемым диапазонам значений, другие функции, в том числе перегруженные).
5. Реализовать файловый ввод/вывод и ввод данных с клавиатуры, вывод данных на дисплей.
6. Предусмотреть обработку различных исключительных ситуаций.
7. Работа всех функций должна быть проверена, и результаты проверки оформлены протоколом тестирования.

3.2. Требования к надежности

В ходе проектирования и разработки обеспечить защиту при попытках пользователя совершить не предусмотренное программой действие, которое может привести к нарушению работы приложения или его закрытию.

3.3. Условия эксплуатации

Не требуются.

3.4. Требования к составу и параметрам технических средств

1. Процессор с тактовой частотой не менее 2.4 ГГц.
2. Оперативная память не менее 8 Гб.
3. Свободное место на жестком диске не менее 5 Гб.
4. Видеокарта с поддержкой DirectX12.
5. Устройства ввода: клавиатура и мышь.
6. Наличие устройств ввода/вывода для считывания и записи файлов.

3.5. Требования к информационной и программной совместимости.

Для запуска итоговой программы должны выполняться следующие условия:

1. На компьютере должна быть установлена операционная система Windows 10/11;
2. На компьютере должно быть установлено программное обеспечение Microsoft Visual Studio 2022.

3.6. Требования к маркировке и упаковке

Распространение данной программы будут реализовано с помощью съёмных носителей, а также интернет ресурсов в открытом доступе.

3.7. Требования к транспортированию и хранению

Данный проект будет размещён в папку на персональном компьютере для хранения и в съёмный носитель для возможности транспортирования продукта.

4. Требование к программной документации

4.1. Содержание расчётно–пояснительной записки

Программная документация должна содержать расчётно–пояснительную записку с содержанием:

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.

4.2. Требования к оформлению

Требования к оформлению документации перечислены в таблице (табл. П1.1). Они должны выполняться на протяжении всей работы с данным продуктом.

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. Файлы предъявляются на компакт–диске: РПЗ с ТЗ; программный код. Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| Страницы | Ориентация — книжная; отдельные страницы, при необходимости, альбомная. 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 листинга — 10 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: «Рис.Х. Название В» приложениях: «Рис.П1.3. Название» |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х». В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) — по центру. В теле таблицы (записи) текстовые значения — выравнены по левому краю, числа, даты — по правому. |

5. Стадии и этапы разработки

Стадии и этапы разработки представлены в таблице П1.2

Таблица П1.2.

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Выбор темы курсовой работы | 14.03.2024 | Утверждена тема для разработки | Выполнено |
| Поиск информации, необходимой для написания программы | 01.05.2024 | Найдены все необходимые материалы | Выполнено |
| Разработка иерархии классов в соответствии с вариантом | 03.05.2024 | Разработана иерархия классов и диаграмма | Выполнено |
| Создание контейнерного класса, от него создать класс с шаблоном хранения указателей на класс-интерфейс | 03.05.2024 | Создан контейнерный класс и класс шаблонов хранения указателей на класс-интерфейс | Выполнено |
| Создание структуры данных для хранения объектов производных классов | 06.05.2024 | Создана структура данных для хранения объектов производных классов | Выполнено |
| Реализация файлового и дисплейного ввода/вывода . | 12.05.2024 | В программе реализовано файловый и дисплейный ввод/вывод информации. | Выполнено |

Продолжение таблицы П.1.2

|  |  |  |  |
| --- | --- | --- | --- |
| Обработка исключительных ситуаций | 12.05.2024 | Реализована проверка  и обработка исключительных ситуаций | Выполнено |
| Тестирование программы | 10.05.2024 – 16.05.2024 | Программа протестирована | Выполнено |

6. Порядок контроля и приемки

Порядок контроля и приемки приведён 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 |
| Разработка технического задания | 19.04.2024 | Оформленное техническое задание | Выполнено |
| Создание программы | 03.05.2024 – 12.05.2024 | Конечная версия программы | Выполнено |
| Оформление руководства пользователя | 18.05.2024 | Оформлено руководство | Выполнено |
| Оформление РПЗ | 25.05.2024 | Оформленная РПЗ | Выполнено |
| Сдача РПЗ | 31.05.2024 | Курсовая работа защищена |  |

# Приложение 2. Текст программы

Файл «Heater.h»

#pragma once

#include <fstream>

#include "string"

#include "iostream"

#include "Color.h"

using namespace std;

class Heater : public IHeater {

private:

protected:

string name;

Color cl;

int price;

double power;

void print\_header() const;

void print\_info() const = 0;

void sfile(ofstream& f) = 0;

public:

Heater(const string name, Color cl, int price, double power);

Heater(const Heater& name);

~Heater();

void setname(const string name);

string getname() const;

void setcolor(Color color);

string getcolor() const;

void setprice(int rice);

int getprice() const;

void setpower(double power);

double getpower() const;

void print() const;

//перегрузка операторов

bool operator == (int N);

bool operator < (const IHeater& obj);

bool operator > (const IHeater& obj);

friend std::ostream& operator<< (std::ostream& cout, const Heater& obj);

};

Файл «IHeater.h»

#pragma once

#include <string>

using namespace std;

class IHeater{

protected:

virtual void print\_header() const = 0;

virtual void print\_info() const = 0;

public:

virtual string getname() const = 0;

virtual int getprice() const = 0;

virtual string getcolor() const = 0;

virtual double getpower() const = 0;

virtual void print() const = 0;

virtual void sfile(ofstream& f) = 0;

virtual bool operator > (const IHeater& obj) = 0;

virtual bool operator < (const IHeater& obj) = 0;

};

Файл «Heater.cpp»

#include "Heater.h"

Heater::Heater(const string name, Color cl, int price, double power)

{

setname(name);

setcolor(cl);

setprice(price);

setpower(power);

}

Heater::Heater(const Heater& name)

:Heater(name.name, name.cl, name.price, name.power)

{}

Heater::~Heater()

{

}

void Heater::setname(const string newname)

{

if (newname.length() == 0) throw exception("\nError name\n");

name = newname;

}

string Heater::getname() const

{

return name;

}

void Heater::setcolor(Color color)

{

cl = color;

}

string Heater::getcolor() const

{

switch (cl) {

case Color::red: return "red"; break;

case Color::green: return "green"; break;

case Color::blue: return "blue"; break;

case Color::white: return "white"; break;

}

}

void Heater::setprice(int newprice)

{

if (newprice < 0) throw out\_of\_range("Error price\n");

price = newprice;

}

int Heater::getprice() const

{

if (price)

return price;

else return 0;

}

void Heater::setpower(double p)

{

if (p < 0) throw out\_of\_range("Error power\n");

power = p;

}

double Heater::getpower() const

{

if (power)

return power;

else return 0;

}

void Heater::print() const

{

print\_header();

cout << "Name: " << getname() << endl;

cout << "Color: " << getcolor() << endl;

cout << "Price: " << getprice() << endl;

cout << "Power: " << getpower() << endl;

print\_info();

}

void Heater::print\_header() const

{

cout << endl << "Heater " << endl;

}

bool Heater::operator==(int N)

{

return price == N;

}

bool Heater::operator<(const IHeater& obj)

{

return price < obj.getprice();

}

bool Heater::operator>(const IHeater& obj)

{

return price > obj.getprice();

}

std::ostream& operator<<(std::ostream& cout, const Heater& obj)

{

obj.getprice();

return cout;

}

Файл «Source.cpp»

#pragma once

#include "Lister.h"

#include "TStatic\_matrix.h"

#include "SpiralHeater.h"

#include "SealedHeater.h"

#include "Tub\_el\_Heater.h"

void input(string &s, int &price, Color &cl, double &power) {

cout << "Введите название нагревательного прибора: ";

cin >> s;

cout << "Введите цену нагревателя: ";

if (!(cin >> price)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

cout << "Введите мощность нагревателя: ";

if (!(cin >> power)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

cout << "Введите цвет нагревательного прибора, если red - 1, green - 2, blue - 3, white - 4: ";

int c; cin >> c;

switch (c) {

case 1: cl = Color::red; break;

case 2: cl = Color::green; break;

case 3: cl = Color::blue; break;

case 4: cl = Color::white; break;

default: throw exception("Ошибка выбора цвета\n"); break;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int a = 1;

const int N = 5;

TemplateMtrx<SealedHeater> SH(N,N);

TemplateMtrx<UnsealedHeater> UH(N, N);

TemplateMtrx<SpiralHeater> SPH(N, N);

TemplateMtrx<Tub\_el\_Heater> TH(N, N);

Lister<IHeater> lst1;

while (a)

{

try

{

cout << "Выберите действие" << endl;

cout << "1: Чтение из файла" << endl;

cout << "2: Добавить \"SealedHeater\" " << endl;

cout << "3: Добавить \"UnsealedHeater\" " << endl;

cout << "4: Добавить \"SpiralHeater\" " << endl;

cout << "5: Добавить \"Tub\_el\_heater\" " << endl;

cout << "6. Вывод элементов в консоль" << endl;

cout << "7. Вывод элементов в файл" << endl;

cout << "8: Поиск элемента " << endl;

cout << "9: Сортировать по убыванию цены" << endl;

cout << "0: Закончить работу" << endl;

cin >> a;

switch (a)

{

case 1: // ввод из файла

{

ifstream f;

f.open("input.txt");

if (!f.is\_open()) {

cout << "Ошибка открытия файла" << endl;

break;

}

while (!f.eof())

{

string c\_name, s, color;

Color cl;

int price;

double power;

if (!(f >> c\_name >> s >> price >> power >> color)) {

cerr << "Ошибка чтения параметров" << endl;

break;

}

if (color == "red" || color == "Red") cl = Color::red;

else if (color == "green" || color == "Green") cl = Color::green;

else if (color == "blue" || color == "Blue") cl = Color::blue;

else if (color == "white" || color == "White") cl = Color::white;

else {

cout << "Ошибка выбора цвета" << endl;

break;

}

if (c\_name == "SealedHeater" || c\_name == "sealedHeater") {

double tp;

if (!(f >> tp)) {

cerr << "Ошибка чтения параметров" << endl;

break;

}

SealedHeater\* obj = new SealedHeater(s, cl, price, power, tp);

SH.Matrix\_Add(obj);

lst1.add(obj);

}

else if (c\_name == "UnsealedHeater" || c\_name == "unsealedHeater") {

float ln;

if (!(f >> ln)) {

cerr << "Ошибка чтения параметров" << endl;

break;

}

UnsealedHeater\* obj = new UnsealedHeater(s, cl, price, power, ln);

UH.Matrix\_Add(obj);

lst1.add(obj);

}

else if (c\_name == "SpiralHeater" || c\_name == "spiralHeater") {

float ln;

bool en;

if (!(f >> ln >> en)) {

cerr << "Ошибка чтения параметров" << endl;

break;

}

SpiralHeater\* obj = new SpiralHeater(s, cl, price, power, ln, en);

SPH.Matrix\_Add(obj);

lst1.add(obj);

}

else if (c\_name == "Tub\_el\_heater" || c\_name == "tub\_el\_heater") {

float th;

if (!(f >> th)) {

cerr << "Ошибка чтения параметров" << endl;

break;

}

Tub\_el\_Heater\* obj = new Tub\_el\_Heater(s, cl, price, power, th);

TH.Matrix\_Add(obj);

lst1.add(obj);

}

}

break;

}

case 2: // SealedHeater

{

int price;

double power, tp;

Color cl;

string s;

input(s, price, cl, power);

cout << "Введите максимальную температуру нагревателя: ";

if (!(cin >> tp)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

SealedHeater\* obj = new SealedHeater(s, cl, price, power, tp);

SH.Matrix\_Add(obj);

lst1.add(obj);

delete obj;

break;

}

case 3: // UnsealedHeater

{

int price;

double power;

float ln;

Color cl;

string s;

input(s, price, cl, power);

cout << "Введите длинну нагревателя: ";

if (!(cin >> ln)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

UnsealedHeater\* obj = new UnsealedHeater(s, cl, price, power, ln);

UH.Matrix\_Add(obj);

lst1.add(obj);

delete obj;

break;

}

case 4: // SpiralHeater

{

int price;

double power;

float ln;

bool en;

Color cl;

string s;

input(s, price, cl, power);

cout << "Введите длинну нагревателя: ";

if (!(cin >> ln)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

cout << "Введите замкнут ли нагревательный элемент(Yes - 1, No - 0): ";

if (!(cin >> en)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

SpiralHeater\* obj = new SpiralHeater(s, cl, price, power, ln, en);

SPH.Matrix\_Add(obj);

lst1.add(obj);

delete obj;

break;

}

case 5: // Tub\_el\_heater

{

int tmp, price;

double power;

float th;

Color cl;

string s;

input(s, price, cl, power);

cout << "Введите толщину стенок нагревателя: ";

if (!(cin >> th)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

Tub\_el\_Heater\* obj = new Tub\_el\_Heater(s, cl, price, power, th);

TH.Matrix\_Add(obj);

lst1.add(obj);

delete obj;

break;

}

case 6: // вывод в консоль

{

lst1.Print();

break;

}

case 7: // вывод в файл

{

ofstream F;

string f\_name = "output.txt";

F.open(f\_name);

lst1.save(F);

F.close();

break;

}

case 8: // поиск элемента

{

cout << "Выберите критерий поиска: " << endl;

cout << "1: цена" << endl;

cout << "2: мощность" << endl;

cout << "3: имя" << endl;

int tmp;

if (!(cin >> tmp)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

switch (tmp)

{

case 1:

{

int price;

cout << "Введите цену ";

if (!(cin >> price)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

lst1.findpr(price);

break;

}

case 2:

{

int power;

cout << "Введите мощность ";

if (!(cin >> power)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

lst1.findpow(power);

break;

}

case 3:

{

string name;

cout << "Введите название ";

if (!(cin >> name)) {

throw runtime\_error("\nНекорректный ввод! Пожалуйста, введите число.\n");

}

lst1.findnm(name);

break;

}

}

break;

}

case 9: // сортировка

{

cout << "Признак сортировки цена" << endl;

SH.sortprice();

UH.sortprice();

SPH.sortprice();

TH.sortprice();

lst1.sortprice();

break;

}

}

}

catch (runtime\_error& e) {

cerr << e.what() << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

catch (exception& ex)

{

cout << ex.what() << endl;

}

catch (...)

{

}

}

system("pause");

return 0;

}

Файл «Color.h»

#pragma once

#include "IHeater.h"

enum class Color { red, green, blue, white };

Файл «TStatic\_Matrix.h»

template <class T>

class TemplateMtrx

{

public:

TemplateMtrx(int i, int j);

void Matrix\_Add(T\* ConstrData);

void del();

virtual void print() const;

void find(int price) const;

void sortprice();

~TemplateMtrx();

private:

int m;

int n;

int Elem1 = 0;

int Elem2 = 0;

T\*\*\* matrx;

};

template <class T>

TemplateMtrx<T>::TemplateMtrx(int m, int n)

{

this->m = m;

this->n = n;

matrx = new T \* \*[m];

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

matrx[i] = new T \* [n];

for (int j = 0; j < n; j++) {

matrx[i][j] = nullptr;

}

}

}

template <class T>

void TemplateMtrx<T>::Matrix\_Add(T\* ConstrData)

{

if (Elem1 == m - 1 && Elem2 == n - 1 && matrx[Elem1][Elem2] != nullptr) {

throw exception("\nMatrix owerflow\n");

};

if (Elem1 == m - 1 && Elem2 == n - 1 && matrx[Elem1][Elem2] == nullptr) {

matrx[Elem1][Elem2] = ConstrData;

}

else if ((Elem1 != m - 1) && (Elem2 == n - 1)) {

matrx[Elem1][Elem2] = ConstrData;

Elem1++;

Elem2 = 0;

}

else {

matrx[Elem1][Elem2] = ConstrData;

Elem2++;

}

}

template <class T>

void TemplateMtrx<T>::del()

{

if (Elem1 == m - 1 && Elem2 == n - 1 && matrx[Elem1][Elem2] != nullptr) {

matrx[Elem1][Elem2] = nullptr;

}

else if (Elem2 == 0) {

Elem1--;

Elem2 = n - 1;

matrx[Elem1][Elem2] = nullptr;

}

else {

Elem2--;

matrx[Elem1][Elem2] = nullptr;

}

}

template <class T>

void TemplateMtrx<T>::print() const

{

{

if (Elem1 == 0 && Elem2 == 0) throw exception("\nMatrix empty\n");

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

for (int j = 0; j < n; j++) {

if (matrx[i][j] != nullptr) {

matrx[i][j]->print();

cout << endl;

}

}

}

}

}

template <class T>

void TemplateMtrx<T>::find(int price) const

{

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

for (int j = 0; j < n; j++) {

if (matrx[i][j] != NULL) {

if (price == matrx[i][j]->getprice()) {

matrx[i][j]->print();

}

}

}

}

}

template <class T>

void TemplateMtrx<T>::sortprice()

{

for (int k = 0; k < m \* n; k++) {

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

for (int j = 0; j < n - 1; j++) {

{

if (matrx[i][j] != nullptr && matrx[i][j + 1] != nullptr) {

if (matrx[i][j]->getprice() < matrx[i][j + 1]->getprice()) {

swap(matrx[i][j], matrx[i][j + 1]);

cout << matrx[i][j]->getprice() << "~";

cout << matrx[i][j + 1]->getprice() << "[" << i << j << "]";

}

}

}

}

}

}

}

template <class T>

TemplateMtrx<T>::~TemplateMtrx()

{

for (int a = 0; a < m; a++) {

if (matrx[a] != NULL)

{

delete matrx[a];

}

}

delete[] matrx;

}

Файл «Lister.h»

#pragma once

#include "Heater.h"

#include "Void\_Lister.h"

#include "SpiralHeater.h"

#include "SealedHeater.h"

#include "Tub\_el\_Heater.h"

template <typename T>

class Lister : private VoidLister {

public:

void add(T\* obj) {

VoidLister::add(static\_cast<void\*>(obj));

}

void remove() {

VoidLister::remove();

}

bool empty() {

return VoidLister::empty();

}

void Print() const {

Node\* current = head;

while (current != nullptr) {

T\* obj = static\_cast<T\*>(current->data);

obj->print();

current = current->next;

}

}

void save(ofstream& f) {

Node\* current = head;

while (current != nullptr) {

T\* obj = static\_cast<T\*>(current->data);

obj->sfile(f);

current = current->next;

}

}

void sortprice() {

Node\* current = head;

while (current != nullptr) {

T\* obj1 = static\_cast<T\*>(current->data);

Node\* prev = head;

while (prev != nullptr) {

T\* obj2 = static\_cast<T\*>(prev->data);

if (\*obj1 > \*obj2) swap(current->data, prev->data);

prev = prev->next;

}

current = current->next;

}

}

void findpr(int price) {

Node\* current = head;

bool f = 1;

while (current != nullptr) {

T\* obj1 = static\_cast<T\*>(current->data);

if (obj1->getprice() == price) {

f = 0;

obj1->print();

}

current = current->next;

}

if (f) {

cout << "Такого элемента нет" << endl;

}

}

void findpow(int power) {

Node\* current = head;

bool f = 1;

while (current != nullptr) {

T\* obj1 = static\_cast<T\*>(current->data);

if (obj1->getpower() == power) {

f = 0;

obj1->print();

}

current = current->next;

}

if (f) {

cout << "Такого элемента нет" << endl;

}

}

void findnm(string name) {

Node\* current = head;

bool f = 1;

while (current != nullptr) {

T\* obj1 = static\_cast<T\*>(current->data);

if (obj1->getname() == name) {

f = 0;

obj1->print();

}

current = current->next;

}

if (f) {

cout << "Такого элемента нет" << endl;

}

}

};

Файл «Void\_Lister.h»

#pragma once

#include <list>

#include <iostream>

class VoidLister {

protected:

struct Node {

void\* data;

Node\* next;

};

Node\* head;

public:

void add(void\* obj) { // добавление в конец списка

if (head == nullptr) {

Node\* newNode = new Node;

newNode->data = obj;

newNode->next = head;

head = newNode;

}

else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

Node\* newNode = new Node;

newNode->data = obj;

newNode->next = nullptr;

current->next = newNode;

}

}

void remove() { // удаление из конца списка

Node\* current = head;

Node\* prev = nullptr;

while (current->next != nullptr) {

prev = current;

current = current->next;

}

if (prev == nullptr) head = nullptr;

else prev->next = nullptr;

delete current;

}

bool empty() // пустота

{

return head == nullptr;

}

~VoidLister() { // деструктор

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

};

Файл «SealedHeater.h»

#pragma once

#include "Heater.h"

using namespace std;

class SealedHeater : public Heater

{

private:

double temperature;

public:

SealedHeater(const string name, Color cl, int price, double power, double temperature);

void settp(double tp);

double gettp() const;

void print\_header() const;

void print\_info() const;

void sfile(ofstream& f);

};

Файл «SealedHeater.cpp»

#include "SealedHeater.h"

SealedHeater::SealedHeater(const string name, Color cl, int price, double power, double temperature) :Heater(name, cl, price, power)

{

setname(name);

setprice(price);

setpower(power);

setcolor(cl);

settp(temperature);

}

void SealedHeater::settp(double tp)

{

if (tp < 0) throw out\_of\_range("Error price\n");

temperature = tp;

}

double SealedHeater::gettp() const

{

return temperature;

}

void SealedHeater::print\_header() const

{

cout << endl << "Sealed Heater " << endl;

}

void SealedHeater::print\_info() const

{

cout << "Heater max temperature: " << gettp() << endl;

}

void SealedHeater::sfile(ofstream& f)

{

f << "Sealed Heater" << endl;

f << getname() << endl;

f << getcolor() << endl;

f << to\_string(getprice()) << endl;

f << to\_string(getpower()) << endl;

f << to\_string(gettp()) << endl;

}

Файл «UnsealedHeater.h»

#pragma once

#include "Heater.h"

class UnsealedHeater :

public Heater

{

private:

float lenght;

public:

UnsealedHeater(const string name, Color cl, int price, double power, float lenght);

void setlenght(float len);

float getlenght() const;

void print\_header() const;

void print\_info() const;

void sfile(ofstream& f);

};

Файл «UnsealedHeater.cpp»

#include "UnsealedHeater.h"

UnsealedHeater::UnsealedHeater(const string name, Color cl, int price, double power, float lenght) :Heater(name, cl, price, power)

{

setname(name);

setprice(price);

setpower(power);

setcolor(cl);

setlenght(lenght);

getlenght();

}

void UnsealedHeater::setlenght(float len)

{

if (len < 0) throw out\_of\_range("Error price\n");

lenght = len;

}

float UnsealedHeater::getlenght() const

{

return lenght;

}

void UnsealedHeater::print\_header() const

{

cout << endl << "Unsealed Heater" << endl;

}

void UnsealedHeater::print\_info() const

{

cout << "Heater lenght: " << getlenght() << endl;

}

void UnsealedHeater::sfile(ofstream& f)

{

f << "Unsealed Heater" << endl;

f << getname() << endl;

f << getcolor() << endl;

f << to\_string(getprice()) << endl;

f << to\_string(getpower()) << endl;

f << to\_string(getlenght()) << endl;

}

Файл «Tub\_el\_heater.h»

#pragma once

#include "Heater.h"

class Tub\_el\_Heater :

public Heater

{

private:

float thickness;

public:

Tub\_el\_Heater(const string name, Color cl, int price, double power, float thickness);

void setthickness(float thick);

float getthickness() const;

void print\_header() const;

void print\_info() const;

void sfile(ofstream& f);

};

Файл «Tub\_el\_heater.cpp»

#include "Tub\_el\_heater.h"

Tub\_el\_Heater::Tub\_el\_Heater(const string name, Color cl, int price, double power, float thickness) :Heater(name, cl, price, power)

{

setname(name);

setprice(price);

setpower(power);

setcolor(cl);

setthickness(thickness);

}

void Tub\_el\_Heater::setthickness(float thick)

{

if (thick < 0) throw out\_of\_range("Error price\n");

thickness = thick;

}

float Tub\_el\_Heater::getthickness() const

{

return thickness;

}

void Tub\_el\_Heater::print\_header() const

{

cout << endl << "Tubular electric heater " << endl;

}

void Tub\_el\_Heater::print\_info() const

{

cout << "Heater tub wall thickness: " << getthickness() << endl;

}

void Tub\_el\_Heater::sfile(ofstream& f)

{

f << "Unsealed Heater" << endl;

f << getname() << endl;

f << getcolor() << endl;

f << to\_string(getprice()) << endl;

f << to\_string(getpower()) << endl;

f << to\_string(getthickness()) << endl;

}

Файл «SpiralHeater.h»

#pragma once

#include "UnsealedHeater.h"

using namespace std;

class SpiralHeater : public UnsealedHeater

{

private:

bool elongated;

public:

SpiralHeater(const string name, Color cl, int price, double power, float lenght, bool el);

void setel(bool el);

void print\_header() const;

void print\_info() const;

void sfile(ofstream& f);

};

Файл «SpiralHeater.cpp»

#include "SpiralHeater.h"

SpiralHeater::SpiralHeater(const string name, Color cl, int price, double power, float lenght, bool el) :UnsealedHeater(name, cl, price, power, lenght)

{

setname(name);

setprice(price);

setpower(power);

setcolor(cl);

UnsealedHeater::setlenght(lenght);

setel(elongated);

}

void SpiralHeater::setel(bool el)

{

elongated = el;

}

void SpiralHeater::print\_header() const

{

cout << endl << "Spiral Heater " << endl;

}

void SpiralHeater::print\_info() const

{

UnsealedHeater::print\_info();

cout << "Heater elongated: " << (elongated ? "True" : "False") << endl;

}

void SpiralHeater::sfile(ofstream& f)

{

UnsealedHeater::sfile(f);

f << (elongated ? "True" : "False") << endl;

}

# Приложение 3. Руководство пользователя

1. Общие сведения о программе

Исполнительный файл: Курс.exe.

2. Описание установки

Установка программы не требуется.

3. Описание запуска

Двойным нажатием левой кнопки мыши запустить файл ЛБ.exe.

4. Инструкция по работе

4.1. Описание возможностей программы

В открывшемся консольном окне выбрать режим работы путем ввода цифры от 0 до 9 (рис. П3.1). При вводе цифры 0 программа завершит работу. Действия 1-5 нужны для ввода, 6,7 для вывода, а 8-9 для работы с контейнером. При выборе ручного ввода данные вводятся строго согласно пояснениям в самой консоли. Во время работы программы можно выбирать любые действия, пока вы не закончите работу.

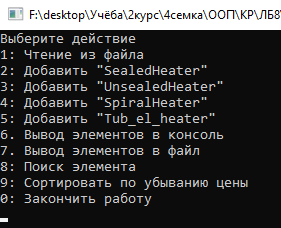


рис. П3.1 Консольное окно с выбором действия

4.2 Описание файлового ввода

При выборе действия чтения из файла, данные заполняются автоматически, так как файл задан в программе (Рис. П3.2).. Файл должен находится непосредственно в директории рядом с Курс.exe

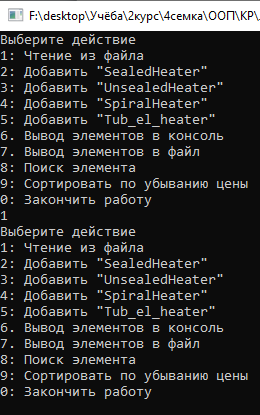


рис. П3.2 Консольное окно после чтения файла

Ниже приведен пример заполнения данных в файле. В самой первой строке прописывается класс к которому принадлежит данный объект. Каждый параметр устройства прописывается в отдельной строке в порядке, который задан при ручном вводе в консольном окне (Рис. П3.3).

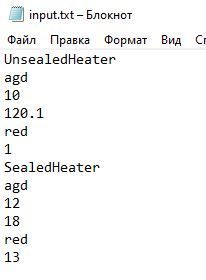


Рис. П3.3. Пример ввода данных в файле

4.3 Описание файлового вывода

В открывшемся консольном окне ввести 7 номер действия (файловый вывод) (Рис. П3.5). После производится вывод в файл output.txt расположенный в директории рядом с Курс.exe.

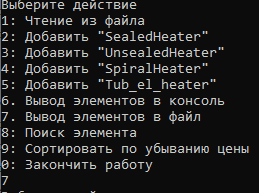


Рис. П3.4. Консольное окно после вывода в файл

Пример вывода информации в файл (рис. П.3.5).

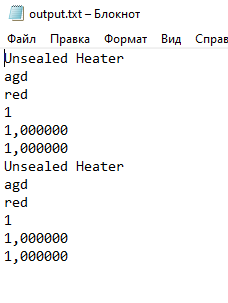


Рис. П3.5. Пример вывода данных в файле

# Приложение 4. Детальная диаграмма классов

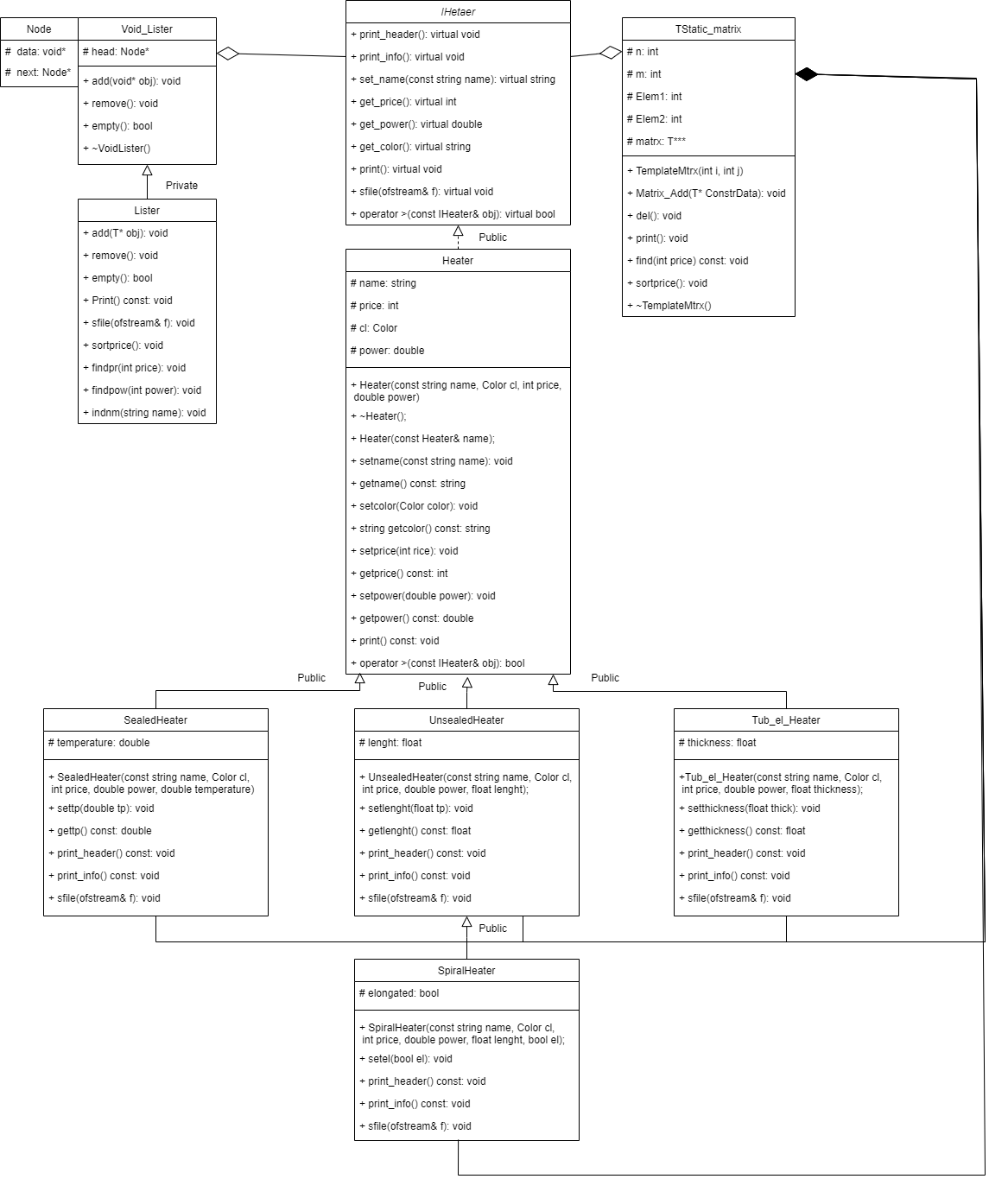


Рис. П.4.1. Детальная диаграмма классов