ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ "ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«Объектно-ориентированное программирование»

на тему: «Объектно-ориентированная система Confection-Filling»

Руководитель:

Выполнил:

асс. каф. ПИ

студент гр. ПИ-19Б

Грищенко Д.А. Носаченко Артём Александрович

Щедрин С.В.

. . 2020 г

. . 2020

Донецк 2020

# РЕФЕРАТ

Пояснительная записка к курсовому проекту содержит: 94 страниц, 53 рисунков, 7 таблицы, 7 источников, 8 приложений.

Объектно-ориентированное программирование, проектирование, моделирование, UML-диаграммы, C++, носители информации.

Объект исследования – носители информации.

Цель работы – закрепление практических навыков самостоятельной постановки и решения задачи обработки данных с помощью ЭВМ средствами объектно-ориентированного программирования (ООП).

Задачи, которые необходимо выполнить для достижения поставленной цели курсового проекта:

* выполнить объектно-ориентрованный анализ предметной области;
* осуществить техническое и рабочее проектирование объектно- ориентированной системы;
* выполнить программную реализацию и тестирование спроектированного программного комплекса.

Методы исследования – объектно-ориентированный анализ и проектирование, техническое и рабочее проектирование, имитационное моделирование (ситуационное моделирование).

Результаты работы – объектно-ориентированная система Confection-Filling для моделирования некоторых операций с носителями информации.

Область применения – подходит для демонстрации студентам возможностей объектно-ориентированного программирования.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, ПРОЕКТИРОВАНИЕ, ЯЗЫК UML, С++, ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_bookmark0)

1. [ПОСТАНОВКА ЗАДАЧИ 8](#_bookmark1)
2. [ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ 10](#_bookmark2)
   1. [Объектно-ориентированный анализ 10](#_bookmark3)
   2. [Диаграммы прецедентов 10](#_bookmark4)
   3. [Объектно-ориентированное проектирование 13](#_bookmark5)
      1. [Диаграммы классов 14](#_bookmark6)
      2. [Диаграммы объектов 15](#_bookmark7)
      3. [Диаграммы последовательностей 16](#_bookmark8)
      4. [Диаграммы состояний 19](#_bookmark9)
3. [РАБОЧЕЕ ПРОЕКТИРОВАНИЕ 21](#_bookmark10)
   1. [Модульная структура программы 21](#_bookmark11)

[3.1 Структура файлов и каталогов программы 22](#_bookmark12)

* 1. [Разработка классов 27](#_bookmark13)
     1. [Реализация инкапсуляции 28](#_bookmark14)
     2. [Реализация наследования 33](#_bookmark15)
     3. [Реализация полиморфизма 36](#_bookmark16)
  2. [Тестирование работоспособности системы классов 38](#_bookmark17)
  3. [Обоснование выбора структур данных 41](#_bookmark18)
  4. [Обоснование выбора алгоритмов 43](#_bookmark19)
     1. [Основной алгоритм программы 43](#_bookmark20)
     2. [Работа с файлами 46](#_bookmark21)
     3. [Подсчёт количества экземпляров класса 48](#_bookmark22)

[ЗАКЛЮЧЕНИЕ 50](#_bookmark23)

[ПЕРЕЧЕНЬ ССЫЛОК 51](#_bookmark24)

[ПРИЛОЖЕНИЕ А 52](#_bookmark25)

[ПРИЛОЖЕНИЕ Б 70](#_bookmark26)

[ПРИЛОЖЕНИЕ В 74](#_bookmark27)

[ПРИЛОЖЕНИЕ Г 77](#_bookmark28)

[ПРИЛОЖЕНИЕ Д 78](#_bookmark29)

[ПРИЛОЖЕНИЕ Е 101](#_bookmark30)

[ПРИЛОЖЕНИЕ Ж 103](#_bookmark31)

[ПРИЛОЖЕНИЕ З 104](#_bookmark32)

# ВВЕДЕНИЕ

На современном этапе развития программного обеспечения все большая роль отводится объектно-ориентированной разработке. Это позволяет создавать большие программные комплексы быстрее, используя повторно имеющиеся модули, обеспечивать инкапсуляцию данных, полиморфизм операций, наследование свойств и методов. Ведущие фирмы по разработке прикладного и системного ПО выполняют свои разработки средствами ООП. Современные операционные системы, например – платформа .NET, имеют внутренний объектно-ориентированный язык и аппаратную поддержку отдельных его компонент. Поэтому изучение и практическое освоение специалистами по программной инженерии этого стиля программирования является актуальной задачей.

Профессиональное обучение программированию по направлению подготовки “Программная инженерия” предусматривает освоение языка Си в качестве базового. Дальнейшее развитие навыков программирования на языке Си и изучение основ ООП выполняется в дисциплине “Объектно-ориентированное программирование”. Для закрепления навыков проектирования и разработки программ в объектно-ориентированном стиле, учебным планом направления подготовки “Программная инженерия” предусмотрено выполнение курсового проекта по этой дисциплине.

Целью курсового проекта является закрепление практических навыков самостоятельной постановки и решения задачи обработки данных с помощью ЭВМ средствами ООП. Во время выполнения курсового проекта студент должен овладеть методикой объектно-ориентированного проектирования программ по формализации и решению поставленной задачи, технологическими приемами разработки объектно- ориентированных программ на языке Си++.

Разработанная программа CourseWork.OOP.C++.exe является объектно-ориентированной системой для предметной области "Кондитерская". Она предназначена для использования в высших учебных заведениях с целью демонстрации знаний, приобретенных обучаемым в области технологии разработки объектно- ориентированных программ средствами языка Си++.

# 1 ПОСТАНОВКА ЗАДАЧИ

Программное изделие CourseWork.OOP.C++.exe должно являеться информационной системой типа «Объектно-ориентированная система».

Должно быть разработано программное обеспечение, которое может работать в следующих режимах:

а) демонстрация класса Confection и работоспособности функций-членов класса:

- поле «Название» char \* Name;

- поле «Дата изготовления» struct Date {int year; int month; int day} ProductDate;

- поле «Срок годности» struct Date {int year; int month; int day} BestBefore;

- поле класса Filling (начинка) Filling \* filling;

- метод получения даты изготовления int GetProductDate();

- метод получения названия int GetName();

- метод добавления начинки void AddFilling(Filling filling);

- конструктор с параметрами Confection (char \* Name, Date ProductDate, Date BestBefore, Filling \* filling);

б) демонстрация работоспособности класса Filling и работоспособности функций-членов класса:

- поле «количество ингредиентов» int Number;

- поле «Название» char \* Name;

- метод «Удалить ингредиент» void AddIngredient ();

- метод «Добавить ингредиент» void DelIngredient ();

- конструктор с параметрами Filling (int Number, char \* Name);

в) демонстрация перегруженного унарного оператора ~ класса Confection;

д) система помощи:

- о программе (программный документ 'Описание программы');

- руководство оператора (программный документ «Руководство оператора»);

- об авторе.

Программное изделие должно функционировать в среде MS Windows версии не ниже XP.

# 2 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ

## 2.1 Объектно-ориентированный анализ

Объектно-ориентированный анализ - это методология анализа предметной области, при которой требования к проектируемой системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области. В процессе объектно-ориентированного анализа производится анализ требований и выделяются основные процессы, происходящие в моделируемой системе и их формулировка в виде прецедентов. Задача этого шага в определении видов деятельности участников процесса и составлении концептуальной модели, которая отражает различные категории элементов предметной области.

## 2.2 Диаграммы прецедентов

Диаграммы прецедентов состоят из четырёх компонентов:

1. исполнитель;
2. прецедент;
3. остальные исполнители;
4. стрелки связи;

Стрелки связи указывают какие исполнители заняты в каких прецедентах. Прецедент – это набор действий, совершаемых исполнителем в системе для определённой цели.

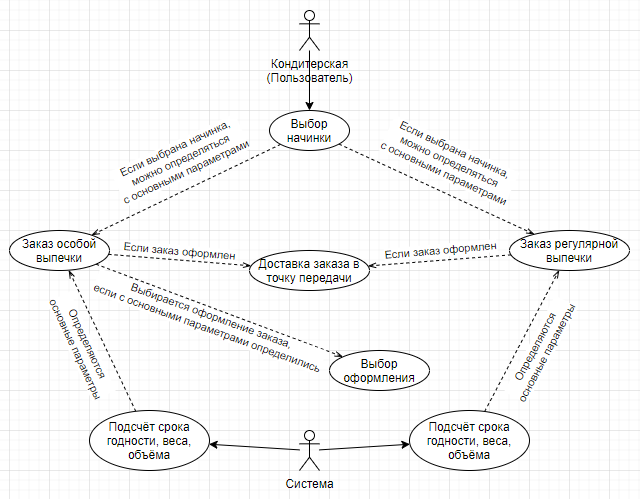


Рисунок 2.1 – Диаграмма прецедентов

На рисунке 2.1 показана диаграмма прецедентов для данной объектно- ориентированной системы. На ней изображены основные действия, которые может выполнять пользователь относительно системы, а также отношения между ними. Данная диаграмма четко иллюстрирует взаимодействие пользователя с системой.

Требования к разрабатываемой объектно-ориентированной системе в целом представлены на обобщенной диаграмме вариантов использования (см. рис. 2.2).



Рисунок 2.2 — Обобщенная диаграмма вариантов использования системы

Пользователь может проводить действия с контейнером и объектами в режиме демонстрации работы объектов после нажатия одноимённой кнопки (см. рис. 2.3).

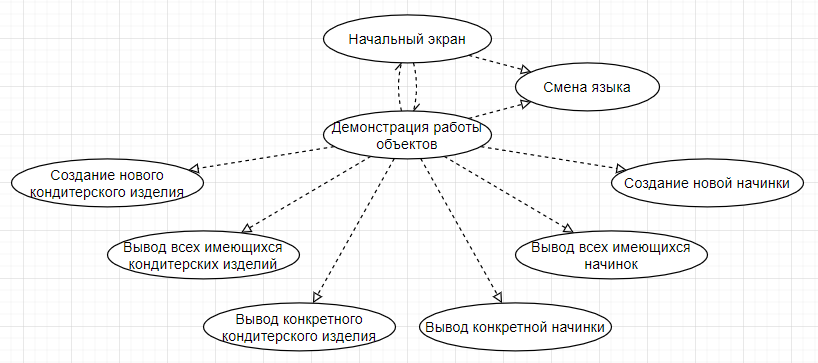


Рисунок 2.3 — Диаграмма вариантов использования для процессов демонстрации работы программы

В системе все объекты хранят Контейнеры через указатели на базовые классы. В Контейнеры можно добавлять объекты.

Изначальное состояние Контейнеров – «Контейнер пуст».

## 2.3 Объектно-ориентированное проектирование

Объектно-ориентированный подход использует объектную декомпозицию. При этом статическая структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Каждый объект системы обладает своим собственным поведением, моделирующим поведение объекта реального мира.

Основные понятия объектно-ориентированного подхода – объект и класс. Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Объект обладает состоянием, поведением и индивидуальностью; структура и поведение схожих объектов определяют общий для них класс. Поведение характеризует воздействие объекта на другие объекты и, наоборот, относительно изменения состояния этих объектов и передачи сообщений. Иначе говоря, поведение объекта полностью определяется его действиями. В процессе объектно-ориентированного проектирования определяются логические программные объекты, которые будут реализованы средствами объектно- ориентированного языка программирования. Эти программные объекты включают в себя атрибуты и методы.

### 2.3.1 Диаграммы классов

Построение статической диаграммы классов является первым этапом объектно-ориентированного проектирования. Такая диаграмма демонстрирует классы системы, их атрибуты, методы и взаимосвязи между ними.

На рисунке 2.4 изображена диаграмма классов для проектируемой объектно- ориентированной системы. На диаграмме показано взаимодействие классов (одиночное и множественное наследование, композиция, агрегация). Каждый класс имеет set- и get-методы для получения доступа к полям и аction-методы, реализующие поведение класса.

При запуске программы система спрашивает, хочет ли пользователь начать работу. Если да, то система предлагает ряд действий, которые возможно осуществить в системе. В зависимости от выбора пользователя объект создаётся и получает свои требуемые значения или выводится для ознакомления пользователя. Для работы системы создаются объекты классов-оболочек состояний приложения таких, как FrmTitle или FrmLangSet, отвечающих за «страницы» приложения начального экрана и экрана выбора языка соответственно. Объекты класса Frame не создаются.

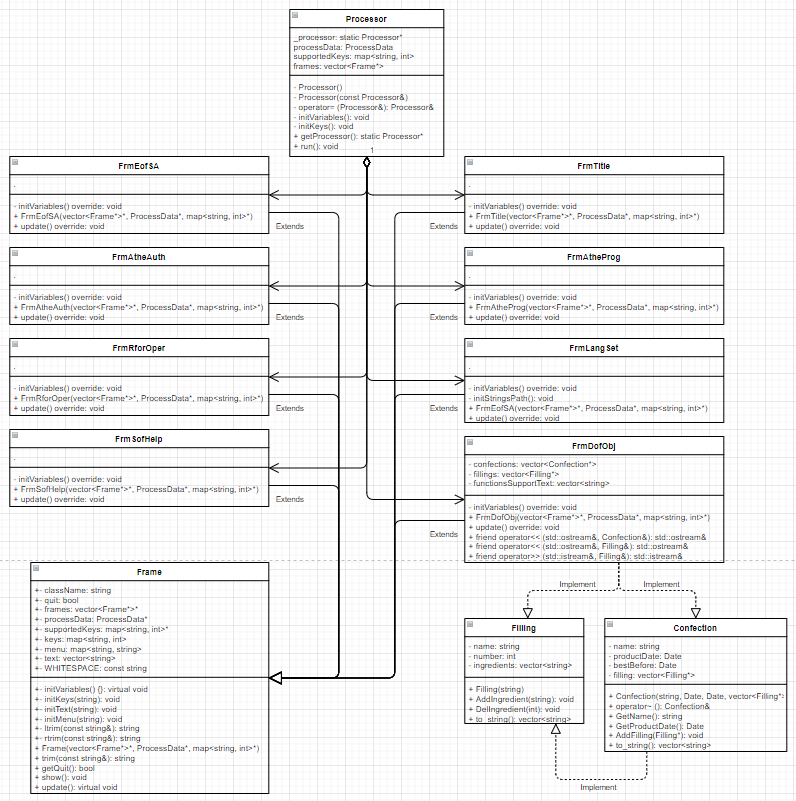


Рисунок 2.4 – Диаграмма классов

### 2.3.2 Диаграммы объектов

Диаграммы объектов – это моментальный снимок части структуры моделируемой системы. Диаграммы объектов аналогичны диаграммам классов, однако в диаграммах объектов указывают имя объекта и имя класса, а также значения у атрибутов.

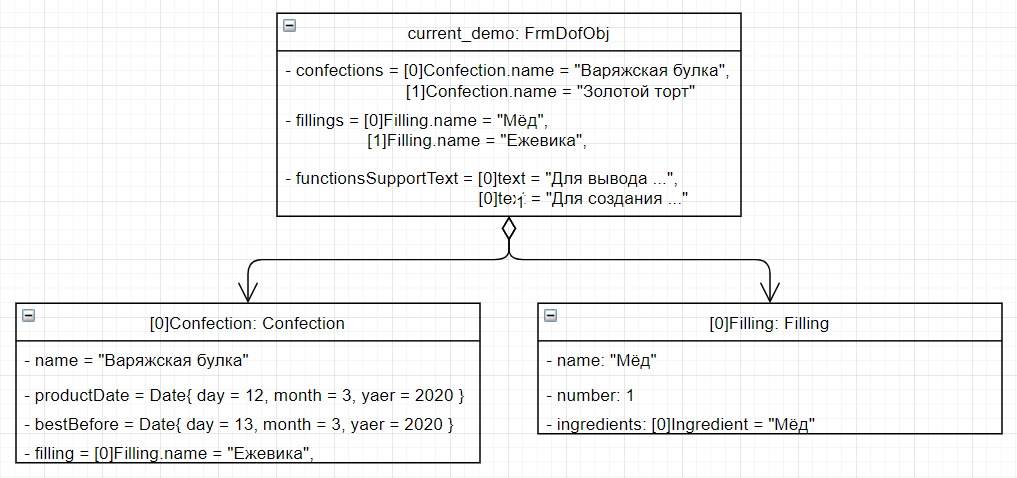


Рисунок 2.5 – Диаграмма объектов

На рисунке 2.5 приведена диаграмма объектов, которая показывает характерные значения атрибутов объектов классов данной объектно- ориентированной систем. В данном случае диаграмма объектов служит примером использования объектно-ориентированной системы и показывает какие значения могут принимать поля определенных объектов.

### 2.3.3 Диаграммы последовательностей

Диаграммы последовательностей отображают порядок сообщений, передаваемых между объектами. Они используются для распределения операций между классами.

На рисунке 2.6 изображена диаграмма последовательностей, отражающая работу пользователя с одним из классов. Для других классов диаграммы последовательностей подобны.

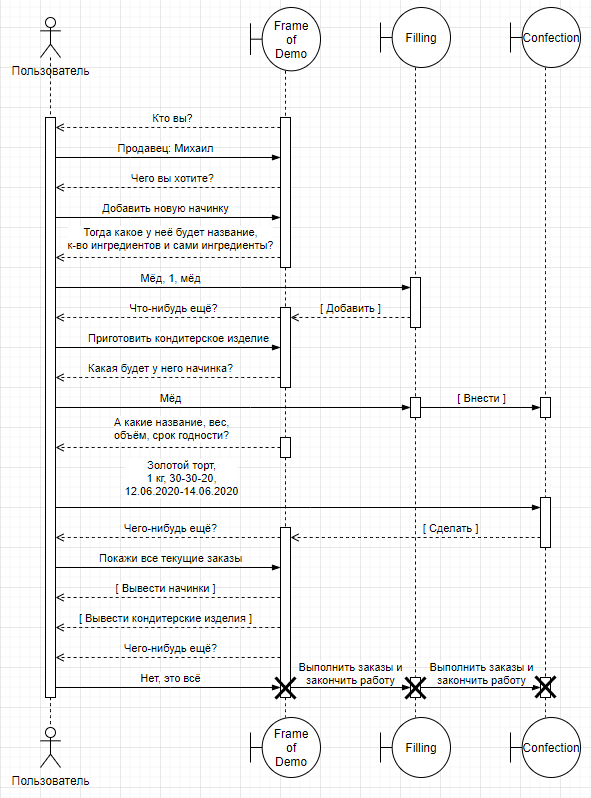


Рисунок 2.6 – Диаграмма последовательностей

### 2.3.4 Диаграммы состояний

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реакционного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

На рисунке 2.7 представлена диаграмма состояний специфицирующая ветвь сценария работы с объектами системы. Для остальных аналогично.

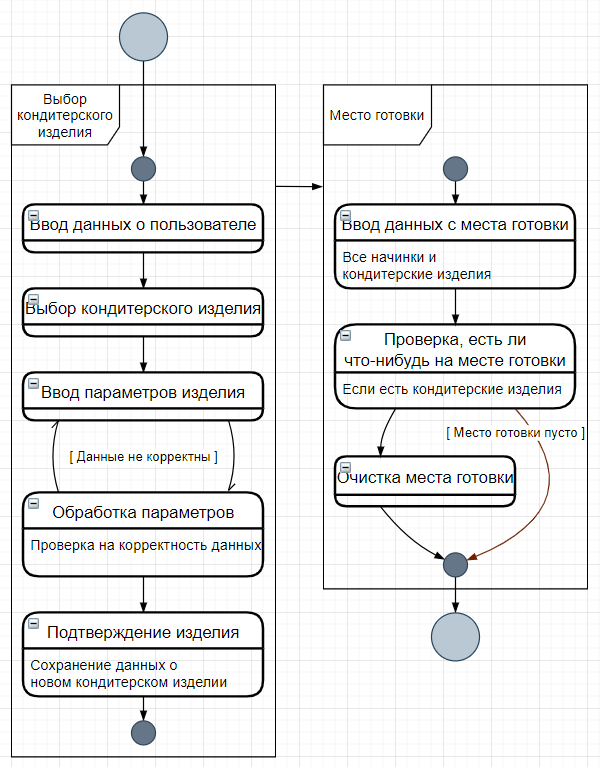


Рисунок 2.7– Диаграмма состояний

# 3 РАБОЧЕЕ ПРОЕКТИРОВАНИЕ

Рабочее и техническое проектирование осуществляется на заключительной стадии процесса проектирования. Эти этапы предусматривают уточнение и детализацию результатов предыдущих стадий, определение структуры программы, ее модулей и их взаимосвязей, разработку главного алгоритма функционирования программы, создание спецификации модулей и классов. Выбираются наиболее оптимальные для решения задачи структуры данных и разрабатываются алгоритмы работы модулей.

## 3.1 Модульная структура программы

Разработанная программная система реализует принцип раздельной компиляции, в соответствии с которым для каждого класса системы предусматривается файл заголовка, представляющий собой определение (интерфейс) класса и файл реализации, содержащий реализацию методов и дружественных функций класса.

Преимущество раздельной компиляции состоит в том, что изменение одного файла не влечет за собой необходимости повторной компиляции всего проекта. При работе со сложными проектами это экономит много времени. Раздельная компиляция позволяет также нескольким программистам работать над одним проектом, так как она служит средством организации исходного текста программы для большого проекта.

## 3.2 Структура файлов и каталогов программы

Программная система CourseWork.OOP.C++.exe реализует мультиязычный интерфейс, обеспечивает минимизацию количества вводимых пользователем данных, а также предоставляет ему подробную справочную информацию в режиме помощи. Перечисленные возможности были реализованы в программе с использованием внешних текстовых файлов, путь к которым задаётся в файле конфгурации.

Файл конфигурации расположен в каталоге «Strings» программы, и является якорем для определения пути ко всем файлам с языками.

Путь к файлу конфигурации может задаваться самим пользователем во время исполнения программы, при этом проверяется корректность нового пути для файла и, если возникает ошибка, оповещает пользователя (см. рис. 3.1).



Рисунок 3.1 — Проверка корректности нового пути для файла

Файлы локализации предназначены для реализации мультиязычности пользовательского интерфейса и содержат тексты сообщений, выдаваемых программой, на русском и английском языках. Данные из этих файлов считываются при старте программы.

Файлы справки необходимы для обеспечения режима помощи системы. Данный режим предусматривает демонстрацию пользователю текстов следующих документов:

* программный документ «О программе»;
* программный документ «Руководство оператора»;
* программный документ «Об авторе».

Текст документов считывается при обращении к соответсвующему пункту меню, что позволяет более рациаонально использовать оперативную память.

Примеры содержимого входных файлов и файла отладки системы приведены на рисунках 3.2 и 3.3 соответственно, а также в приложении Е и Ж.

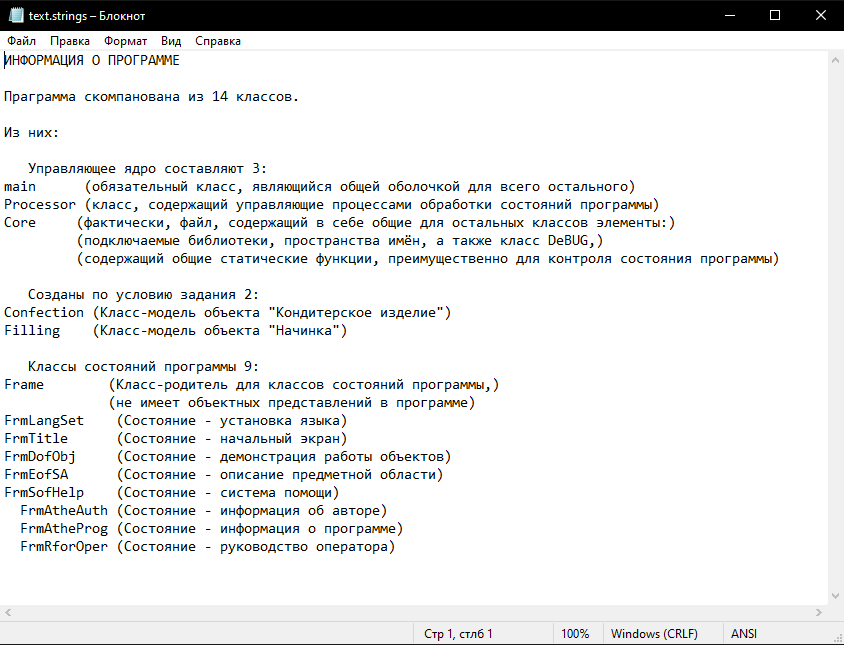


Рисунок 3.2 — Пример входного файла «Информацио о программе»

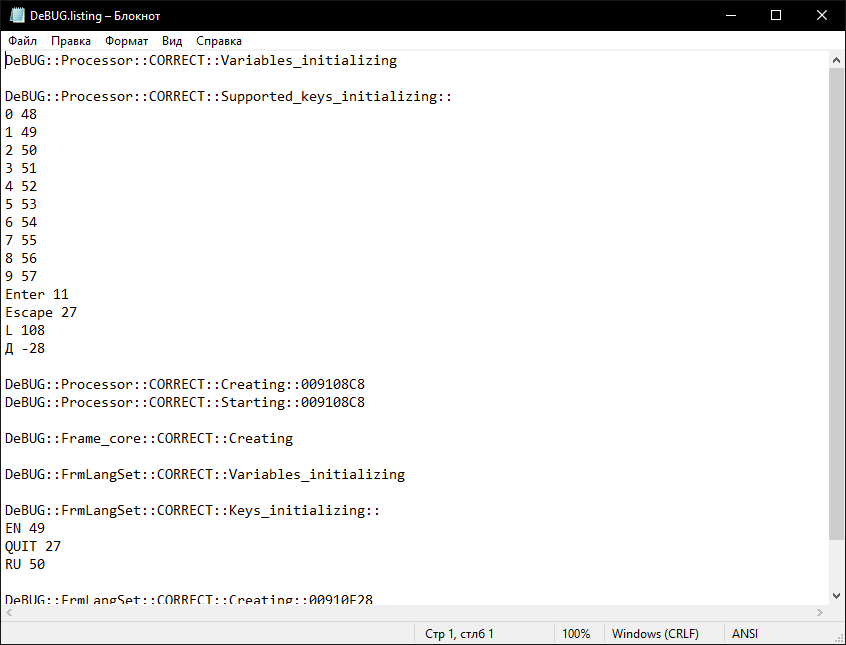


Рисунок 3.3 — Пример выходного файла отладки

## 3.3 Разработка классов

В данной системе созданы 12 основных классов, которые взаимодействуют между собой: Processor, Confection, Filling, Frame, FrmLangSet, FrmTitle, FrmDofObj, FrmEofSA, FrmSofHelp, FrmAtheAuth, FrmAtheProg, FrmRforOper.

### 3.3.1 Реализация инкапсуляции

Инкапсуляцией в терминах объектно-ориентированного программирования называется упаковка данных и/или функций в единый компонент или создание нового абстрактного типа данных (класса).

Целью инкапсуляции является обеспечение согласованности внутреннего состояния объекта. В языке C++ для инкапсуляции используются публичные методы класса, в частности, get-методы и set-методы. Поля, за редким исключением, не должны быть свободно доступными вне класса.

Главные цели инкапсуляции: предельная локализация изменений при необходимости таких изменений, прогнозируемость таких изменений (какие изменения нужно сделать для заданного изменения функциональности) и прогнозируемости изменений. Корректность присваиваемых значений контролируется изначально при вводе, а только потом передаётся в set-метод

Подробная спецификация полей и методов разработанных классов приведена в таблицах 3.1-3.6.

Таблица 3.1 — Спецификация класса Processor

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| processor: static Processor\* | Поле экземпляра обработчика |
| processData: ProcessData | Поле данных приложения |
| supportedKeys: map<string, int> | Поле поддерживаемых клавиш |
| frames: vector<Frame\*> | Поле состояний приложения |
| Processor() | Стандартный конструктор |
| Processor(const Processor&) | Конструктор с параметрами |
| operator= (Processor&): Processor& | Перегруженный оператор присваивания |
| initVariables(): void | Функция инициализации переменных |
| initKeys(): void | Функция инициализации клавиш |
| getProcessor(): static Processor\* | Функция выдачи экземпляра обработчика |
| run(): void | Функция-стартер |

Таблица 3.2 — Спецификация класса Confection

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| name: string | Поле «Название конд. изделия» |
| productDate: Date | Поле «Дата изготовления» |
| bestBefore: Date | Поле «Годен до» |
| filling: vector<Filling\*> | Поле «Начинки» |
| Confection(string, Date, Date, vector<Filling\*>) | Конструктор с параметрами |
| operator~ (): Confection& | Перегруженный оператор инверсии |
| GetName(): string | Get-метод поля name, |
| GetProductDate(): Date | Get-метод для поля productDate |
| AddFilling(Filling\*): void | Метод добавления начинки |
| to\_string(): vector<string> | Метод вывода информации в текстовом виде |

Таблица 3.3 — Спецификация класса Filling

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| name: string | Поле «Название начинки» |
| number: int | Поле «Количество ингредиентов» |
| ingredients: vector<string> | Поле «Ингредиенты» |
| Filling(string) | Конструктор с параметрами |
| AddIngredient(string): void | Метод добавления ингредиентов |
| DelIngredient(int): void | Метод удаления ингредиентов |
| to\_string(): vector<string> | Метод вывода информации в текстовом виде |

Таблица 3.4 – Спецификация класса Frame

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| className: string | Поле «Имя класса» |
| quit: bool | Поле «Выход» |
| frames: vector<Frame\*>\* | Поле «Состояния» |
| processData: ProcessData\* | Поле «Данные приложения» |
| supportedKeys: map<string, int>\* | Поле «Поддерживаемые клавиши» |
| keys: map<string, int> | Поле используемых клавиш |
| menu: map<string, string> | Поле «Меню» |
| text: vector<string> | Поле основного текста |
| WHITESPACE: const string | Поле с пустыми символами |
| initVariables() {}: virtual void | Функция инициализации переменных |
| initKeys(string): void | Функция инициализации клавиш |
| initText(string): void | Функция инициализации основного текста |
| initMenu(string): void | Функция инициализации меню |
| ltrim(const string&): string | Функция удаления пустоты слева в строках |
| rtrim(const string&): string | Функция удаления пустоты справа в строках |
| Frame(vector<Frame\*>\*, ProcessData\*, map<string, int>\*) | Конструктор с параметрами |
| trim(const string&): string | Функция удаления пустоты по краям строк |
| getQuit(): bool | Get-функция поля quit |
| show(): void | Функция отображения данного состояния |
| update(): virtual void | Функция обновления состояния приложения |

Таблица 3.5 – Спецификация класса FrmDofObj

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| confections: vector<Confection\*> | Поле кондитерских изделий |
| fillings: vector<Filling\*> | Поле начинок |
| functionsSupportText: vector<string> | Поле надписей в функциях |
| initVariables() override: void | Функция инициализации переменных |
| FrmDofObj(vector<Frame\*>\*, ProcessData\*, map<string, int>\*) | Конструктор с параметрами |
| update() override: void | Функция обновления состояния приложения |
| friend operator<< (std::ostream&, Confection&): std::ostream& | Перегруженный внешний оператор вывода для кондитерских изделий |
| friend operator<< (std::ostream&, Filling&): std::ostream& | Перегруженный внешний оператор вывода для начинок |
| friend operator>> (std::istream&, Filling&): std::istream& | Перегруженный внешний оператор ввода для начинок |

Таблица 3.6 – Спецификация класса FrmLangSet

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| initVariables() override: void | Функция инициализации переменных |
| initStringsPath(): void | Функция инициализации пути к текстам |
| FrmEofSA(vector<Frame\*>\*, ProcessData\*, map<string, int>\*) | Конструктор с параметрами |
| update() override: void | Функция обновления состояния приложения |

Классы FrmAtheProg, FrmEofSA, FrmTitle, FrmAtheAuth, FrmRforOper, FrmSofHelp имеют одинаковую труктуру, представленную на примере класса FrmTitle в табл. 3.7. Обязательное отличие есть лишь в названиях конструкторов.

Таблица 3.7 – Спецификация класса FrmTitle

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| initVariables() override: void | Функция инициализации переменных |
| FrmTitle (vector<Frame\*>\*, ProcessData\*, map<string, int>\*) | Конструктор с параметрами |
| update() override: void | Функция обновления состояния приложения |

### 3.3.2 Реализация наследования

Наследование – это принцип объектно-ориентированного программирования, позволяющий создавать новые классы на основе уже существующих, обеспечивая таким образом возможность повторного использования кода, и, как следствие, снижает трудоемкость и время разработки, а также число ошибок в программе. Класс-потомок при этом может добавить собственные поля и методы, а также пользоваться родительскими полями и методами.

В данной объектно-ориентированной системе реализовано два вида наследования: простое и множественное.

При простом наследовании у класса всего один предок. Примером одиночного наследования является public-наследование класса FrmTitle от класса Frame. На рисунке 3.4 изображен заголовочный файл производного класса FrmTitle, а на рисунке 3.5 изображен базовый класс Frame. Класс FrmTitle наследует от класса Furniture все protected-поля и методы. Это очень удобно для повторного использования кода. При вызове конструктора класса FrmTitle вызывается конструктор базового класса Frame, при вызове деструктора также вызывается деструктор базового класса. Также класс FrmTitle имеет свои собственные методы, кроме тех, которые наследуются от класса Furniture.

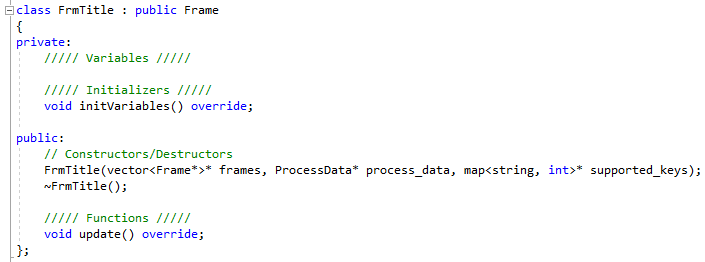


Рисунок 3.4– Заголовочный файл класса FrmTitle



Рисунок 3.5 – Заголовочный файл класса Frame

Множественное наследование предусматривает наличие у подкласса нескольких базовых классов.

Было решено создавать классы в иерархической структуре, потому что она является наиболее удобной в реализации данной объектно-ориентированной системы.

Верхней ступеней иерархии является класс Frame – от него уже наследуется классы FrmTitle, FrmLangSet, FrmAofProg и другие.

### 3.3.3 Реализация полиморфизма

Полиморфизм — это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач. Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных. Тип данных, который используется при вызове функции, определяет, какая конкретная версия функции действительно выполняется. В C++ можно использовать одно имя функции для множества различных действий. Это называется перегрузкой функций.

#### 3.3.3.1 Перегрузка операторов

Перегрузка операторов – один из способов реализации полиморфизма, заключающийся в возможности одновременного существования в одной области видимости нескольких различных вариантов применения оператора, имеющих одно и то же имя, но различающихся типами параметров, к которым они применяются.

Для класса Confection была выполнена перегрузка оператора инверсии “~”. В классе FrmDofObj была выполнена внешняя перегрузка дружественных операторов ввода и вывода потоков относительно классов Confection и Filling (см. рис. 3.6 – 3.7).

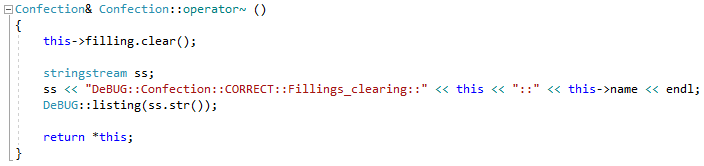


Рисунок 3.6 — Перегрузка оператора ~ для класса Confection

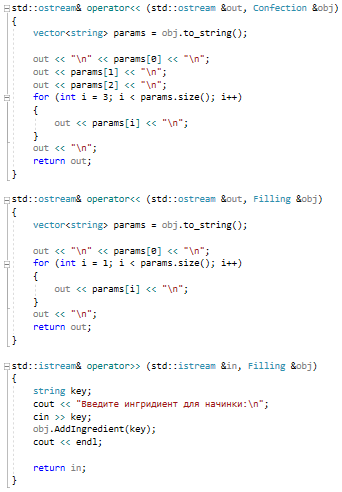


Рисунок 3.7 — Перегрузка операторов ввода\вывода

потоков в классе FrmDofObj.

#### 3.3.3.2 Абстрактный класс

Абстрактный класс – класс, который не имеет экземпляров.

В рамках курсового проекта был реализован абстрактный класс Frame – базовый абстрактный класс, который содержит общие обязательные поля и методы для всех состояний приложения, заголовочный файл которого представлен ранее на рис. 3.5.

## 3.4 Тестирование работоспособности системы классов

Программа обрабатывает следующие исключительные ситуации:

* отсутствие ресурсного файла с языковыми данными, ошибка его открытия или недостаток строк в файле или файл пуст;
* отсутствие ресурсного файла с главным меню, ошибка его открытия, недостаток строк в файле, файл пуст;
* отсутствие ресурсного файла с меню объектов, ошибка его открытия, недостаток строк в файле, файл пуст;
* отсутствие ресурсного файла с данными помощи, ошибка его открытия, недостаток строк в файле, файл пуст;
* отсутствие ресурсного файла с информацией о проекте, ошибка его открытия, недостаток строк в файле, файл пуст;
  + отсутствие ресурсного файла с руководством пользователя, ошибка его открытия, недостаток строк в файле, файл пуст;
  + обрабатываются ошибки выхода за диапазон правильных числовых значений;

Следует обратить внимание на то, что большинство исключительных ситуаций, обрабатываемых в системе, не приводят к выводу из программы, а незамедлительно обрабатываются и корректируются для свободного продолжения работы пользователя в системе.

## 3.5 Обоснование выбора структур данных

Для реализации меню используются операторы if()-else, так как они являются наиболее удобными для обработки исключительных ситуаций.

Для считывания информации из файла используется функция getline(). Она является наиболее удобной так как в её возможности входит простая организация считывания из файла с учётом пробелов и признаков конца строки.

Для хранения обьектов состояний системы был использован векторный список. Такой подход позовляет хранить практически неограниченое количество обьектов, в отличии от массивов объектов, и динамически менять их количество, при этом не теряя возможности обращаться к ним по их номеру в списке. Это позволяет эфективно управлять имеющимися элементами, не создавая лишних функций при разработке и объектов в реализации (см. рис 3.8).

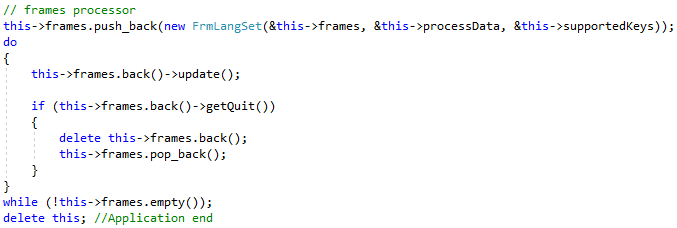


Рисунок 3.8 — Обработка векторного списка frames

## 3.6 Обоснование выбора алгоритмов

### 3.6.1 Основной алгоритм программы

На рисунке 3.9 изображен основной алгоритм работы программы.

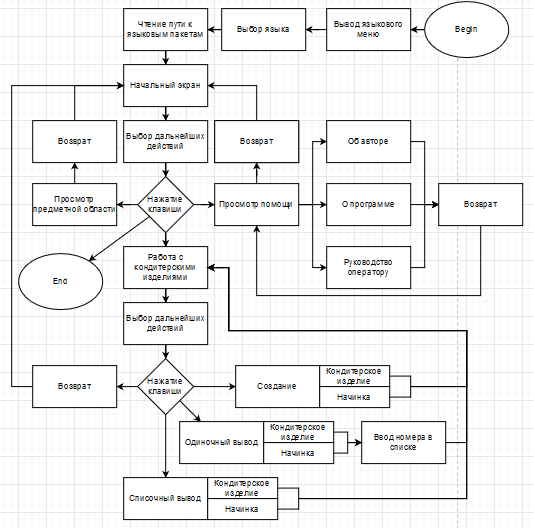


Рисунок 3.9 – Основной алгоритм

На рисунке 3.10 рассмотрен алгоритм чтения из файла и выбор языка. На первом этапе открывается файл на чтение, проверяется на открытие, на то, чтобы файл не был пуст и на достаточное количество строк в файле, и считывается с помощью функции getline: while (getline(fin, str, '\n')) и данные файла заносятся в вектор строк после чего идёт вывод нужной строки файла. На втором этапе пользователь выбирает язык работы. При нажатии «не правильной» клавиши пользователю выводится сообщение об ошибке.

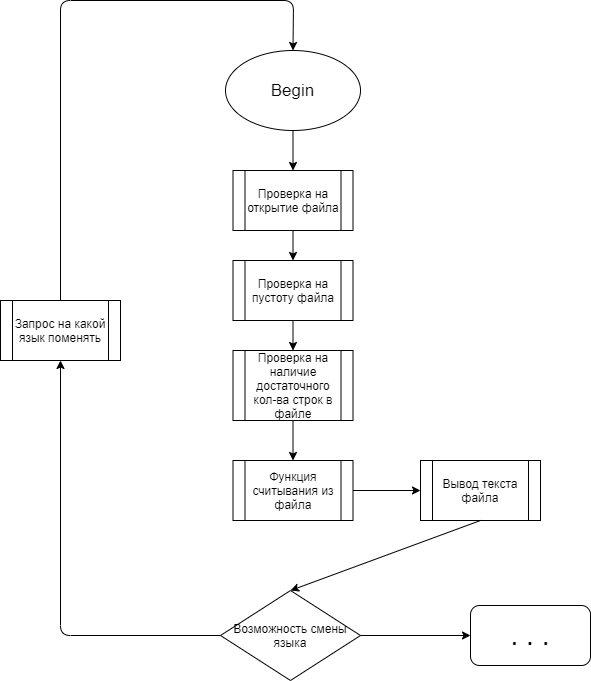


Рисунок 3.10 – Алгоритм смены языка

### 3.6.2 Работа с файлами

В рамках разработанной системы были реализованы вывода в файл и чтения из файла состояния объектов.

Ввод всего текста из файлов основывается на считывания строк файла в вектор. Это действие необходимо в каждом объекте состояния программы, по-этому логичным решением стало создание соответствующей функции (см. рис. 3.11). В качестве параметра для функции приходит строковое значение, содержащее путь к файлу ввода.

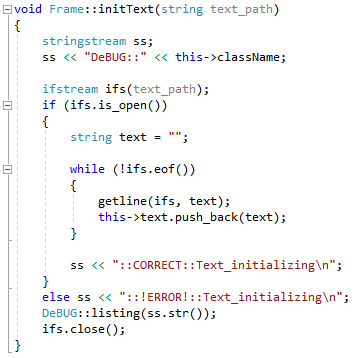


Рисунок 3.11 – Запись строк файла в вектор

### 3.6.3 Подсчёт количества экземпляров класса

Контроль количества экземпляров классов осуществляется блягодаря записи классов в нумеруемые списки – векторные списки, по-этому в любой момент времени возможно узнать точное количество экземпляров интересующих разработчика классов.

# ЗАКЛЮЧЕНИЕ

Разработанная программа является объектно-ориентированной системой для моделирования предметной области «Confection-Filling». Она предназначена для использования в высших учебных заведениях с целью демонстрации основных концепций объектно-ориентированного проектирования и программирования.

В ходе создания системы были пройдены следующие этапы объектно- ориентированной разработки программных продуктов:

* + объектно-ориентированный анализ предметной области;
  + моделирование статических и динамических аспектов предметной области средствами языка UML;
  + рабочее и техническое проектирование системы;
  + объектно-ориентированное программирование на языке С++;
  + системное тестирование и оценка качества ПО;
  + создание технической и пользовательской документации.

Наиболее перспективными направлениями дальнейшего развития программного продукта CourseWork.OOP.C++.exe является конкретизация и расширение реализованной имитационной модели, расширение набора объектов предметной области, доступных для добавления.

# ПЕРЕЧЕНЬ ССЫЛОК

1. Герб Саттер - Стандарты программирования на С++. 101 правило и рекомендация
2. Расширенные библиотеки STL для С++. Наборы и итераторы (+ CD-ROM). Мэтью Уилсон
3. UML 2.0. Объектно-ориентированное моделирование и разработка. Джеймс Рамбо, М. Блаха.
4. Керниган Б., Ритчи Д. Язык программирования Си – СПб. Невский диалект, 2001. – 352с.
5. Моделирование на UML. Учебно-методическое пособие. Иванов Д. Новиков Ф.
6. Эффективный и современный С++. .42 рекомендации по использованию С++. Наборы и итераторы (+ CD-ROM). Мэтью Уилсон.
7. Методические указания к выполнению курсового проекта по курсу «Объектно-ориентированное программирование» (для студентов специальности «Программная инженерия»)

# ПРИЛОЖЕНИЕ А

ГВУЗ "ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

Кафедра программной инженерии

Утверждаю

Федяев О.И.

 \_\_\_\_\_\_\_\_\_\_\_

\_\_.\_\_.\_\_\_\_\_ г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

к курсовому проекту по дисциплине

"Объектно-ориентированное программирование"

на тему: "Объектно-ориентированная система

"Confection-Filling"

Руководитель: Выполнил:

Асс. каф. ПИ студент гр. ПИ-19б

Грищенко Д. А. Носаченко А. А.

Щедрин С. В.

\_\_.\_\_.\_\_\_\_ г. \_\_.\_\_.\_\_\_\_ г.

Донецк 2020

## ВВЕДЕНИЕ

На современном этапе развития программного обеспечения все большая роль отводится объектно-ориентированному стилю проектирования и программирования (ООП). Это позволяет разрабатывать большие программные комплексы быстрее, используя повторно имеющие модули, обеспечивать инкапсуляцию данных, полиморфизм операций, наследование свойств и методов. Ведущие фирмы по разработке прикладного и системного ПО выполняют свои разработки средствами ООП. Современные операционные системы, например, платформа .NET имеет внутренний объектно-ориентированный язык и аппаратную поддержку отдельных его компонент. Поэтому изучение и практическое освоение специалистами по программной инженерии этого стиля программирования является актуальной задачей.

Профессиональное обучение программированию по направлению подготовки “Программная инженерия” предусматривает освоение языка Си в качестве базового. Дальнейшее развитие навыков программирования на языке Си и изучение основ ООП выполняется в дисциплине “Объектно-ориентированное программирование”. Для закрепления навыков проектирования и разработки больших программ в объектно-ориентированном стиле учебным планом направления подготовки “Программная инженерия” предусмотрено выполнение курсового проекта по этой дисциплине.

Целью курсового проекта является закрепление практических навыков самостоятельной постановки и решения задачи обработки данных с помощью ЭВМ средствами ООП. Во время выполнения курсового проекта студент должен овладеть методикой объектно-ориентированного проектирования программ по формализации и решению поставленной задачи, технологическими приемами разработки объектно-ориентированных программ на языке Си++.

Условием успешного выполнения курсового проекта являются практические навыки ООП, полученные при выполнении лабораторных работ по дисциплине “ООП”.

Разработанная программа market.exe является объектно-ориентированной системой для предметной области “Торговый центр и магазин самообслуживания”. Она предназначена для использования в высших учебных заведениях с целью демонстрации знаний, приобретенных обучаемым в области технологии разработки объектно-ориентированных программ средствами языка Си++.

## 1 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Курсовой проект выполняется на основании “Задания на курсовой проект” по дисциплине “Объектно-ориентированное программирование” для студентов направления подглотовки “Программная инженерия”, выданное кафедрой “Программная инженерия” ДонНТУ.

## 2 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Программное изделие CourseWork.OOP.C++.exe является информационной системой типа “Объектно-ориентированная система”.

Должно быть разработано программное, которое может работать в следующих режимах:

а) демонстрация описания классов Confection и Filling и работоспособности функций-членов классов:

1) конструкторов;

2) функций-членов Getxx (Get-методов);

3) функций-членов Setxx (Set-методов);

4) деструктора;

5) функций-членов Actionxx;

б) демонстрация работоспособности класса Confection при перегрузке оператора инверсии ~;

в) демонстрация работоспособности класса FrmDofObj при перегрузке операторов ввода\вывода в потоки << \ >>;

г) демонстрация работоспособности классов Confection и Filling при одиночном наследовании;

д) система помощи:

д.1) о программе (программный документ «Описание программы»);

д.2) руководство оператора (программный документ “Руководство оператора”);

д.3) об авторе.

Программное изделие должно функционировать в среде MS Windows версии не ниже XP.

## 3 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ

### 3.1 Требования к функциональным характеристикам

Программное изделие должно удовлетворять следующим общим требованиям:

а) использование технологии раздельной компиляции файлов;

б) режим работы монитора для программы – текстовый (консольное приложение);

в) использование файлов для хранения всех данных;

г) обеспечение двуязычия в общении с пользователем.

Критерии оценивания программного изделия по степени выполнения требований технологии раздельной компиляции:

а) для оценки «удовлетворительно»: отсутствие файла проекта, объединение исходных файлов с помощью директив препроцессора;

б) для оценок «хорошо» и «отлично»: использование файла проекта и технологии раздельной компиляции файлов;

Критерии оценивания программного изделия по степени выполнения требований для инкапсуляции:

а) для оценки «удовлетворительно»: реализованы только два класса, описание которых представлено в словесном описании предметной области;

б) для оценки «хорошо»:

1) реализованы классы, описание которых представлено в словесном описании предметной области;

2) реализованы дополнительные классы, расширяющие словесное описание предметной области.

в) для оценки «отлично»:

1) реализованы два класса, описание которых представлено в словесном описании предметной области;

2) реализованы дополнительные классы, расширяющие словесное описание предметной области и возможности интерфейса программы;

3) для каждого из классов предусмотрена возможность подсчета количества объектов, существующих в текущий момент в программе.

Критерии оценивания программного изделия по степени выполнения требований для наследования:

а) для оценки «удовлетворительно»: реализованы только два класса, описание которых представлено в словесном описании предметной области, и указанные отношения между ними;

б) для оценки «хорошо»:

1) реализованы классы, описание которых представлено в словесном описании предметной области, и указанные отношения между ними;

2) реализованы дополнительные классы - иерархия классов и/или контейнерные классы (только одиночное наследование);

в) для оценки «отлично»:

1) реализованы классы, описание которых представлено в словесном описании предметной области, и указанные отношения между ними;

2) реализованы дополнительные классы - иерархия классов и/или контейнерные классы (одиночное наследование);

3) реализовано множественное наследование.

Критерии оценивания программного изделия по степени выполнения требований для полиморфизма:

а) для оценки «удовлетворительно»: реализована перегрузка только тех операторов, описание которых представлено в словесном описании предметной области;

б) для оценки «хорошо»:

1) реализована перегрузка операторов, описание которых представлено в словесном описании предметной области;

2) реализована перегрузка операторов для некоторых дополнительных классов;

3) перегрузка операторов выполнена как внутри, так и вне класса.

в) для оценки «отлично»:

1) реализована перегрузка только тех операторов, описание которых представлено в словесном описании предметной области;

2) реализована перегрузка операторов для всех дополнительных классов;

3) перегрузка операторов выполнена как внутри, так и вне класса.

4) в реализации полиморфизма использованы абстрактные классы и виртуальные функции.

Критерии оценивания программного изделия по степени выполнения требований к демонстрации работы системы:

а) для оценки «удовлетворительно»: все результаты работы выводятся в системный поток вывода;

б) для оценок «хорошо» и «отлично»:

1) результаты работы выводятся в системный поток вывода;

2) состояние объектов системы выводится в файловый поток вывода.

Критерии оценивания программного изделия по степени выполнения требований в режиме помощи:

а) для оценки «удовлетворительно»: использование файла с условием задачи и файла помощи;

б) для оценок «хорошо» и «отлично» - использование файлов:

1) о программе (программный документ “Описание программы”);

2) руководство оператора (программный документ);

3) об авторе.

Критерии оценивания программного изделия по степени выполнения требований по организации работы с файлами:

а) для оценки «удовлетворительно»:

1) путь доступа файлам данных установлен программным путем (является константой в программе);

б) для оценки «хорошо»:

1) путь доступа к файлам определяется пользователем на этапе выполнения программы (наличие в главном меню проекта опции настройки, в которой пользователь указывает путь доступа к файлам данных и/или их имена);

в) для оценки «отлично»:

1) путь доступа к файлам определяется пользователем на этапе выполнения программы (наличие файла конфигурации с данными о пути доступа к файлам данных и/или их именах);

2) предусмотрена обработка исключительных ситуаций при работе с файлами.

Критерии оценивания программного изделия по степени выполнения требований по обеспечению двуязычия интерфейса пользователя:

а) для оценки «удовлетворительно»:

1) приложение имеет фиксированную пару языков общения (тексты надписей являются константами в программе);

2) переключение языков реализовано только в одном кадре интерфейса;

3) двуязычие реализовано не для всех режимов работы;

б) для оценок «хорошо» и «отлично»:

1) приложение имеет произвольную пару языков общения (тексты надписей читаются из файлов);

2) переключение языков реализовано на каждом кадре интерфейса;

3) двуязычие реализовано для всех режимов работы;

### 3.2 Требование к надежности

Программное изделие для обеспечения надежности функционирования должно:

а) проверять наличие всех файлов данных;

б) обеспечить минимизацию количества информации, вводимой пользователем;

в) контролировать корректность ввода данных пользователем;

г) обрабатывать исключительные ситуации, вызванные не корректным вводом данных пользователем с целью предотвращения прерывания выполнения программы.

### 3.3 Условия эксплуатации

Пользователь должен иметь квалификацию не ниже “оператор ПЭВМ”.

Для эксплуатации программного изделия необходимо наличие программиста, в функции которого входит внесение изменений в информационную часть системы.

### 3.4 Требования к составу и параметрам технических средств

Для функционирования программного изделия необходим персональный компьютер со стандартным набором периферийных устройств (монитор, клавиатура, мышь).

### 3.5 Требования к информационной и программной совместимости

Курсовой проект должен быть выполнен на языке Си++ в операционной системе MS Windows версии не ниже XP.

## 4 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Пояснительная записка по курсовому проекту должна сопровождаться следующими программными документами:

а) описание программы;

б) руководство оператора;

в) текст программы.

## 5 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Перечень этапов и сроки приведен в табл.5.1

Таблица 5.1 - График выполнения курсового проекта

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование этапа | Срок выполнения | |
| Неделя | Дата |
| 1 | Выдача задания на курсовой проект.Уточнение задачи. | 1 | 03.02.2020 |
| 2 | Постановка задачи. Определение требований к программе.  а) запись постановки задачи;  б) составление технического задания и его утверждение. | 1-2:  2  2 | 03.02.2020-16.02.2020 |
| 3 | Объектно-ориентированный анализ и проектирование:  а) объектно-ориентированный анализ;  б) проектирование структуры классов;  в) проектирование отношений между классами;  г) проектирование структуры объектов и их поведения. | 3-8:  3-4  4-5  5-6  6-8 | 17.02.2020-  29.03.2020 |
| 4 | Техническое проектирование:  а) модульный анализ: определение структуры программы, ее модулей и их взаимосвязей;  б) разработка основного алгоритма функционирования программы;  в) создание спецификации модулей. | 5-9 | 02.03.2020-05.04.2020 |
|  | Рабочее проектирование:  а) определение структур данных и разработка алгоритмов работы модулей;  б) разработка классов:  1) реализация инкапсуляции;  2) реализация наследования;  3) реализация полиморфизма. | 5-10:  5-6  6-7  7-9  9-10 | 02.03.2020-12.04.2020 |
| 6 | Написание программы | 9-12 | 30.03.2020-19.04.2020 |
| 7 | Настройка программы | 11-12 | 06.04.2020-19.04.2020 |
| 8 | Комплексная отладка и тестирование | 12 | 13.04.2020-19.04.2020 |
| 9 | Написание пояснительной записки | 13 | 20.04.2020-26.04.2020 |
| 10 | Защита курсового проекта | 14 | 27.04.2020-03.05.2020 |

## 6 ПОРЯДОК КОНТРОЛЯ И ПРИЕМА

Курсовой проект выполняется 14 недель. Пояснительная записка по курсовому проекту предоставляется на проверку преподавателю не менее чем за 3 рабочих дня до даты защиты.

Защита происходит в присутствии комиссии в составе 2-3 человек и включает:

а) доклад (до 3 минут), отражающий все этапы выполнения курсового проекта;

б) презентацию программы (с демонстрацией всех ее функций);

в) ответы на вопросы комиссии.

## 7 СЛОВЕСНОЕ ОПИСАНИЕ ПPЕДМЕТНОЙ ОБЛАСТИ

***Вариант 10***

Создать класс Confection, включающий в себя следующие элементы:

- поле «Название» char \* Name;

- поле «Дата изготовления» struct Date {int year; int month; int day} ProductDate;

- поле «Срок годности» struct Date {int year; int month; int day} BestBefore;

- включить в класс поле класса Filling (начинка) Filling \* filling;

- метод получения даты изготовления int GetProductDate();

- метод получения названия int GetName();

- метод добавления начинки void AddFilling(Filling filling);

- конструктор с параметрами Confection (char \* Name, Date ProductDate, Date BestBefore, Filling \* filling).

Создать класс Filling, содержащий в себе элементы:

- поле «количество ингредиентов» int Number;

- поле «Название» char \* Name;

- конструктор с параметрами Filling (int Number, char \* Name);

- метод «Удалить ингредиент» void AddIngredient ();

- метод «Добавить ингредиент» void DelIngredient ().

В классе Confection перегрузить оператор ~ (унарный оператор класса), новое действие - удалить начинку.

## 8 ПРИМЕР СОДЕРЖАНИ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Титульный лист

Реферат

Содержание

Введение

1 Постановка задачи

2 Объектно-ориентированный анализ и проектирование

2.1 Объектно-ориентированный анализ

2.2 Объектно-ориентированное проектирование

2.2.1 Структура классов

2.2.2 Отношения между классами

2.2.3 Структура объектов

3 Рабочее проектирование

3.1 Структуры данных приложения и методы работы с ними

3.2 Разработка классов

3.2.1 Реализация инкапсуляции

3.2.2 Реализация наследования

3.2.3 Реализация полиморфизма, перегрузка операторов

3.3 Обоснование выбора алгоритмов

3.3.1 работа с файлами

3.3.2 подсчет количества экземпляров класса

3.3.3 организация информационной поддержки

3.4 Тестирование работоспособности системы классов

Выводы

Перечень ссылок

Приложение А. Техническое задание

Приложение Б. Описание программы

Приложение В. Руководство программиста

Приложение Г. Руководство оператора

Приложение Д. Текст программы

Приложение Е. Файлы исходных данных

Приложение Ж. Файлы выходных данных

Приложение К. Экранные формы

# ПРИЛОЖЕНИЕ Б ОПИСАНИЕ ПРОГРАМЫ

## Б.1 Общие сведения

Программное изделие CourseWork.OOP.C++.exe является информационной системой типа ''Объектно-ориентированная система’’.

Таблица Б.1 – Описание программы

|  |  |
| --- | --- |
| Заголовочные файлы | Файлы реализации |
| Confection.h | Confection.cpp |
| Filling.h | Filling.cpp |
| Frame.h | Frame.cpp |
| FrmAtheAuth.h | FrmAtheAuth.cpp |
| FrmAtheProg.h | FrmAtheProg.cpp |
| FrmDofObj.h | FrmDofObj.cpp |
| FrmEofSA.h | FrmEofSA.cpp |
| FrmLangSet.h | FrmLangSet.cpp |
| FrmRforOper.h | FrmRforOper.cpp |
| FrmSofHelp.h | FrmSofHelp.cpp |
| FrmTitle.h | FrmTitle.cpp |
| Processor.h | Processor.cpp |
| Core.h | main.cpp |

Проект CourseWork.OOP.C++.exe состоит из файлов, находящихся во второй колонке таблицы Б.1. В первой колонке находятся заголовочные файлы, в которых описаны классы, прототипы функций и константы.

Системные программные средства, используемые программой CourseWork.OOP.C++.exe, должны быть представлены локализованной версией операционной системы Windows XP/Vista/7/8/10.

Программное изделие выполнено в объектно-ориентированном стиле как консольное приложение, с применением языка С++.

## Б.2 Функциональное назначение

Программа CourseWork.OOP.C++.exe предназначена для создания и работы с объектами двух классов: Confection и Filling. Также реализован ряд классов, которые отвечают за текущее состояние приложения.

Класс Confection (Кондитерсокое изделие) имеет поля name (Название), productDate (Дата изготовления), bestBefore (Годен до..), filling (Начинка). Для класса Confection предусмотрены действия: создание объекта, перегрузка оператора инверсии ~, которых при новой реализации очищает список начинок для рассматриваемого кондитерсого изделия, добавление начинки, возвраты имени и даты изготовления, возврат полного текстового описания

Класс Filling имеет поля name (Название), number (Количество ингредиентов), ingredients (ингредиенты). Также в нём предусмотрены действия: создание объекта, добавление ингредиента, удаление ингредиента, возврат полного текстового описания.

Классы контроля состояния сходны между собой и являются реализациями абстрактного класса Frame, содержащего поля: className (Название класса), quit (Необходимость выхода), frames (Состояния), processData (Данные приложения), supportedKeys (Поддерживаемые клавиши), keys (Клавиши), menu (Меню), text (Текст), WHITESPACE («Пустые» символы); а также методы: инициализация переменных, инициализация клавиш, инициализация текста, инициализация меню, очистка пустых символов слева в строке, очистка путых символов справа в строке, очистка пустых символов по краям строки, добавление объекта, вывод значения поля quit, отображение состояния программы, обновление состояния программы.

## Б.3 Описание логической структуры

В программе реализовано меню с несколькими вариантами работы (взаимодействие с объектами различных классов), доступны методы сохранения и чтения данных в/из файл/а, реализован мультиязычный интерфейс (по умолчанию: русский, английский). Выходные файлы сохраненных результатов можно просмотреть в любом текстовом редакторе.

## Б.4 Используемые технические средства

Для корректной работы программы необходим стандартный набор периферийных устройств (монитор, клавиатура, мышь), предустановленная ОС MS Windown версии не ниже XP, процессор с тактовой частотой не менее 1.2 ГГц и оперативная память объемом не менее 256 Мб.

Б.5 Вызов и загрузка

При запуске программа требует ввести путь к файлу-якорю, являющимся привязкой пути ко всей необходимой для работы с файлами информацией. При наличии файла-якоря и правильного указания его расположения программа переходит в рабочий режим: появляется начальная страница, подсказки, главное меню, позже – взаимодействие с объектами системы. Также путь расположения файла-якоря можно изменить вручную.

Б.6 Входные данные

Необходимые для демонстрации возможностей программы данные пользователь вводит через клавиатуру. Для удобства и уменьшения объема ввода разработано меню с выбором направления работы по нажатию одной клавиши. При чтении данных об объектах из файла информация читается из соответствующих объектам текстовых файлов.

Б.7 Выходные данные

Режим работы программы – консольный, все данные программа выводит в текстовом виде на выбранном языке в консоль. При сохранении данных об объектах в файл информация сохраняется в соответствующие объектам текстовые файлы.

ПРИЛОЖЕНИЕ В РУКОВОДСТВО ПРОГРАММИСТА

В.1 Назначение и условия применения программы Программа выполнена на языке программирования С++ в среде разработки MS Visual Studio 2015 в 2020 году как курсовой проект по дисциплине «Объектно-ориентированное программирование». Для корректной работы программы необходим стандартный набор периферийных устройств (монитор, клавиатура, мышь), предустановленная ОС MS Windown версии не ниже XP, процессор с тактовой частотой не менее 1.2 ГГц и оперативная память объемом не менее 256 Мб.

В.2 Характеристики программы

В программе разработано меню с выбором режимов работы системы. Режим работы – консольный. Для минимизации ввода каких-либо команд пользователем все действия в меню происходят по нажатию на одну из командных клавиш, значение которых объяснено в подсказке. При некорректном ввод каких-либо данных программа сообщает о некорректности ввода и запрашивает повторный ввод. В случае ошибки чтения файлов алгоритм рассчитан на получение максимального объема информации.

В.3 Обращение к программе

Программа запускается через файл Furniture.exe из любого места на жестком диске и требует ввода пути к конфиг-файлу, расположенному в папке с файлами программы.

В.4 Входные и выходные данные

Входные данные:

* + выбор режима работы программы в главном меню – нажатие клавиши управления;
  + ввод информации о каком-либо объекте системы – ввод строки или целого

числа;

* + чтение информации из текстовых файлов .txt. Выходные данные:
  + вывод сообщений в консоль;
  + сохранение информации в текстовые файлы .txt.

В.5 Сообщения

В ходе работы программы могут возникнуть определенные ошибки. Ниже приведены фотографии с возможными ошибками.

При работе могут появиться следующие ошибки:

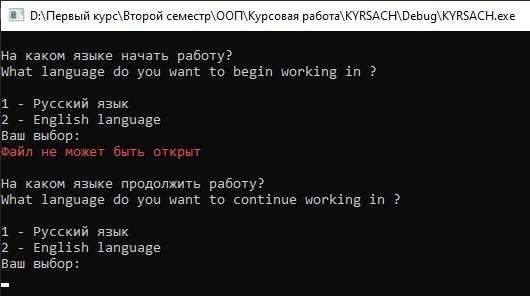


Рисунок В.1 – Ошибка файла

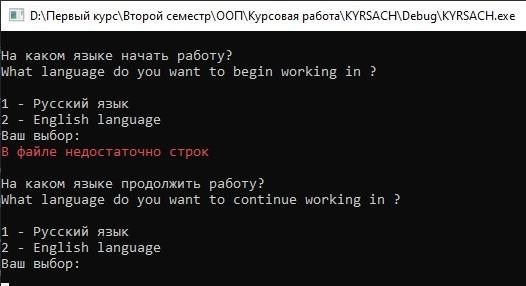


Рисунок В.2 – Недостаток строк в файле

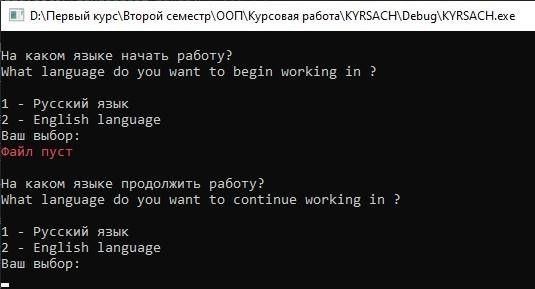


Рисунок В.3 – Файл пуст

ПРИЛОЖЕНИЕ Г РУКОВОДСТВО ОПЕРАТОРА

Г.1 Назначение программы

Программа выполнена на языке программирования С++ в среде разработки MS Visual Studio 2010 в 2020 году как курсовой проект по дисциплине «Объектно- ориентированное программирование».

Г.2 Условия выполнения программы

Для корректной работы программы необходим стандартный набор периферийных устройств (монитор, клавиатура, мышь), предустановленная ОС MS Windown версии не ниже XP, процессор с тактовой частотой не менее 1.2 ГГц и оперативная память объемом не менее 256 Мб.

Г.3 Сообщения оператору

В ходе работы будут выводиться определенные сообщения. Ниже приведены некоторые из них с объяснениями.

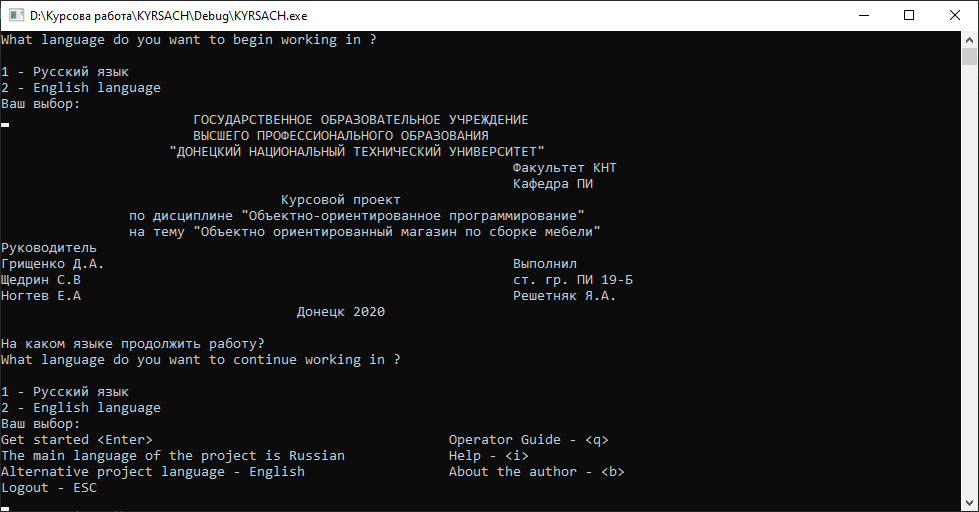


Рисунок Г.1 – выбор пользователя

ПРИЛОЖЕНИЕ Д ТЕКСТ ПРОГРАММЫ

|  |  |
| --- | --- |
| **Furniture.h** #pragma once #include <string> using namespace std; class Furniture  {  protected:  string Vendor; string Material;  public:  Furniture();  Furniture(string value\_Vendor, string value\_Matrial); string GetVendor();  string GetMaterial();  void SetVendor(string value\_Vendor); void SetMaterial(string value\_Material); Furniture &operator /(const Furniture&);  ~Furniture();  };  **Furniture.cpp**  #include "stdafx.h" #include "Furniture.h" #include <iostream> #include <string> using namespace std;  Furniture::Furniture()  {  Vendor = ""; Material = "";  }  Furniture::Furniture(string value\_Vendor, string value\_Material)  {  Vendor = value\_Vendor; Material = value\_Material;  }  string Furniture::GetVendor()  {  return Vendor;  }  string Furniture::GetMaterial()  {  return Material;  }  void Furniture::SetVendor(string value\_Vendor)  {  Vendor = value\_Vendor;  }  void Furniture::SetMaterial(string value\_Material)  {  Material = value\_Material;  }  Furniture & Furniture::operator /(const Furniture &furniture1)  {  Vendor = this->Vendor + furniture1.Vendor; | #include "stdafx.h" #include <stdio.h> #include <iostream> #include <conio.h> #include <locale.h> #include <stdlib.h> #include "Furniture.h" #include "Table.h" #include "Color.h" #include "ColorTable.h" #include "Methods.h" #include "Window.h" #include "Warehouse.h" #include <Windows.h> #include <vector> #include <fstream> #undef max  using namespace std;  void func(ifstream & file, string name)  {  file.open(name); if (!file.is\_open())  {  throw "Файл не может быть открыт";  }  if (file.peek() == EOF)  {  throw "Файл пуст";  }  if (file.is\_open() && file.peek() != EOF)  {  int i = 0, k = 0; string str;  file >> i; if (i == 0)  {  throw "Невозможно проверить количества нужных даннных в файле";  }  else  {  getline(file, str); while (!file.eof())  {  str = ""; getline(file, str); k++;  }  if (k < i)  {  throw "В файле недостаточно  строк";  }  }  }  file.close();  }  void Peregryz1()  {  Table table("nice", "good", 4, 10, 15, 20); table++;  cout << table.GetVendor() << endl; cout << table.GetMaterial() << endl; cout << table.GetNum() << endl; |

|  |  |
| --- | --- |
| Material = this->Material + furniture1.Material; return \*this;  }  Furniture::~Furniture()  {  Vendor.clear(); Material.clear();  }  **Table.h**  #pragma once #include "Methods.h" #include "Furniture.h" #include <string> using namespace std;  class Table : public Furniture  {  protected:  int Num; float Height; float Width; float Length;  public:  Table();  Table(string value\_Vendor, string value\_Material, int value\_Num, float value\_Height, float value\_Width, float value\_Length);  int GetNum(); float GetHeight(); float GetWidth(); float GetLength(); void SetNum(int);  void SetHeight(float); void SetWidth(float); void SetLength(float);  Table &operator ++(int value);  ~Table();  };  **Table.cpp** #include "stdafx.h" #include <stdio.h> #include <conio.h> #include "Table.h"  #include "Furniture.h" #include <iostream> #include <Windows.h> using namespace std; Table::Table()  {  Num = 0;  Height = 0;  Width = 0;  Length = 0;  }  Table::Table(string value\_Vendor, string value\_Material, int value\_Num, float value\_Height, float value\_Width, float value\_Length)  {  Vendor = value\_Vendor; Material = value\_Material; Num = value\_Num; Height = value\_Height; Width = value\_Width; Length = value\_Length;  }  int Table::GetNum()  { | cout << table.GetHeight() << endl; cout << table.GetWidth() << endl; cout << table.GetLength() << endl;  }  void Peregryz2()  {  Window window("good", "look", 10, 15, 34);  Window okno("book", "table", 567, 324, 5); window += okno;  cout << window.GetVendor() << endl; cout << window.GetMaterial() << endl; cout << window.GetHeight() << endl; cout << window.GetWidth() << endl; cout << window.GetLength() << endl;  }  void Peregryz3()  {  Color color; cin >> color; Color radyga; cin >> radyga;  Color c = color\*radyga; cout << c;  }  void Peregryz4()  {  ColorTable colortable("nice", "good", 4, 10, 15, 20, "lalala");  ColorTable change("nnytyujrt", "tyr", 4, 10, 15, 20, "jyujrher");  ColorTable b;  b = colortable % change;  cout << b.GetVendor() << endl; cout << b.GetMaterial() << endl; cout << b.GetNum() << endl; cout << b.GetHeight() << endl; cout << b.GetWidth() << endl; cout << b.GetLength() << endl; cout << b.GetColor() << endl;  }  void Peregryz5()  {  Furniture furniture1("nice", "good"); Furniture furniture2("nnytyujrt", "tyr"); Furniture b;  b = furniture1 / furniture2;  cout << b.GetVendor() << endl; cout << b.GetMaterial() << endl;  }  void language()  {  int key1; bool lang; do  {  cout << endl;  cout << "\n1 - Русский язык" << endl; cout << "2 - English language" << endl; cout << "Ваш выбор: " << endl;  key1 = \_getch(); switch (key1)  {  case '1':  lang = TRUE;  if (lang == TRUE)  {  lang = FALSE; BOOL bFile; bFile = CopyFile(  L"D:\\Курсова |

|  |  |
| --- | --- |
| return Num;  }  float Table::GetHeight()  {  return Height;  }  float Table::GetWidth()  {  return Width;  }  float Table::GetLength()  {  return Length;  }  void Table::SetNum(int value\_Num)  {  Num = value\_Num;  }  void Table::SetHeight(float value\_Height)  {  Height = value\_Height;  }  void Table::SetWidth(float value\_Width)  {  Width = value\_Width;  }  void Table::SetLength(float value\_Length)  {  Length = value\_Length;  }  Table & Table::operator++(int value)  {  this->Vendor = "Booood"; this->Material = "Dance"; this->Num = 5-10\*Length; this->Height = 234 / Width; this->Width = 32454 % 45;  this->Length = 100; return \*this;  }  Table::~Table()  {  Num = 0;  Height = 0;  Width = 0;  Length = 0;  }  **Color.h**  #pragma once #include "stdafx.h" #include <stdio.h> #include <conio.h> #include <iostream> #include <locale.h> #include <string> #include <stdlib.h> #include <fstream> using namespace std; class Color  {  protected: | работа\\KYRSACH\\KYRSACH\\main\_russian.txt",  L"D:\\Курсова  работа\\KYRSACH\\KYRSACH\\Example.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  }  break;  case'2':  lang = FALSE;  if (lang == FALSE)  {  lang = TRUE; BOOL bFile; bFile = CopyFile(  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\main\_english.txt",  L"D:\\Курсова  работа\\KYRSACH\\KYRSACH\\Example.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  break;  }  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50);  }  typedef vector<string> konfig; typedef vector<string> languages;  languages& BuildVector(languages& langu)  {  string path = "Konfig\_file.txt"; konfig files;  string str;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  files.push\_back(str);  }  fout.close(); int i = 0;  ifstream fin(files.at(10)); getline(fin, str, '\n');  while (getline(fin, str, '\n'))  {  langu[i] = str; i++;  }  fin.close(); return langu;  }  void menu\_file(konfig &files) |

|  |  |
| --- | --- |
| string color\_name; int count;  public:  Color();  void SetColor(string value\_color\_name); string GetColor();  Color &operator \*(const Color&);  friend std::istream & operator >> (istream &input, Color  &other);  friend std::ostream& operator << (std::ostream& output, Color &other);  ~Color();  };  **Color.cpp** #include "stdafx.h" #include "stdafx.h" #include <stdio.h>  #include <iostream> #include "Color.h" #include <string> using namespace std;  Color::Color()  {  count = 0; color\_name = "";  }  void Color::SetColor(string value\_color\_name)  {  color\_name = value\_color\_name;  }  string Color::GetColor()  {  return color\_name;  }  Color & Color::operator\*(const Color &other)  {  if (color\_name.length() < 5)  {  color\_name = color\_name + other.color\_name; count = color\_name.length();  }  else  {  color\_name = other.color\_name + color\_name; count = color\_name.length();  }  return \*this;  }  Color::~Color()  {  color\_name.clear();  }  std::istream & operator >> (istream &input, Color &other)  {  string key;  cout << "Введите слово: "; cin >> key; other.color\_name = key;; cout << endl;  return input; | {  try  {  ifstream put\_table; ifstream put\_window; string path = files.at(11); string path1 = files.at(12);  func(put\_table, path); func(put\_window, path1); put\_table.open(files.at(11)); put\_window.open(files.at(12)); ColorTable colortable; Window window;  Warehouse first;  cout << "Выберите язык на котором хотите продолжать работать" << endl;  language(); ifstream fin;  string path2 = files.at(10); func(fin, path2); languages langu;  string str; fin.open(files.at(10));  getline(fin, str, '\n');  while (getline(fin, str, '\n'))  {  langu.push\_back(str);  }  fin.close();  string s, s1, s2, s3; BuildVector(langu);  int key, key1, key2, key3, num, flag = 0; float width, height, lenght;  cout << langu.at(0) << endl; cout << langu.at(1) << endl; do  {  language(); BuildVector(langu);  cout << langu.at(2) << endl; cout << langu.at(3) << endl; cout << langu.at(4) << endl; key1 = \_getch();  switch (key1)  {  case'1':  language(); BuildVector(langu);  cout << langu.at(5) << endl; put\_table >> s;  put\_table >> s; cout << s << endl;  colortable.SetVendor(s);  window.SetVendor(s); do  {  cout << langu.at(6)  << endl;  cout << langu.at(7)  << endl;  cout << langu.at(8)  << endl;  cout << langu.at(9)  << endl;  cout << langu.at(10)  << endl;  key2 = \_getch(); switch (key2)  {  case'1': |

|  |  |
| --- | --- |
| }  std::ostream& operator << (std::ostream& output, Color &other)  {  int i;  i = other.count; while (i >= 0)  {  \_putch(other.color\_name[i--]);  }  printf("\n"); return output;  }  **ColorTable.h** #pragma once #include "Table.h" #include "Color.h" #include "Methods.h" #include <string> using namespace std;  class ColorTable : public Table, public Color, public Methods  {  protected:  float Perimetr; float Ploshad; float Volume;  static int count\_table;  public:  static int getNextID(); ColorTable();  ColorTable(string value\_Vendor, string value\_Material, int value\_Num, float value\_Height, float value\_Width, float value\_Length, string value\_color\_name);  float getPloshad(); float getPerimeter(); float getVolume(); void show();  void language();  void func(ifstream &, string);  ColorTable &operator %(const ColorTable&);  ~ColorTable();  };  **ColorTable.cpp** #include "stdafx.h" #include <string> #include "ColorTable.h" #include "Table.h" #include "Color.h" #include "Methods.h" #include <Windows.h> #include <vector> #include <fstream> #include <string> #include <iostream> using namespace std;  int ColorTable::count\_table = 0;  int ColorTable::getNextID()  {  return count\_table;  }  ColorTable::ColorTable()  {  Ploshad = 0;  Perimetr = 0;  Volume = 0;  }  ColorTable::ColorTable(string value\_Vendor, string value\_Material, | language(); BuildVector(langu);  cout <<  langu.at(11) << endl;  put\_table  >> s1;  cout << s1  << endl;  colortable.SetMaterial(s1);  cout <<  langu.at(12) << endl;  put\_table  >> s2;  cout << s2  << endl;  colortable.SetColor(s2);  cout <<  langu.at(13) << endl;  put\_table  >> num;  if  (!put\_table || num <= 0)  {  put\_table.clear(); put\_table.ignore(std::numeric\_limits<std::streamsize>::max(),  '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Измените содержимое файла на число" << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  else  {  cout << num << endl; colortable.SetNum(num);  }  cout <<  langu.at(14) << endl;  put\_table  >> width;  if  (!put\_table || width <= 0)  {  put\_table.clear(); put\_table.ignore(std::numeric\_limits<std::streamsize>::max(),  '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE); |

|  |  |
| --- | --- |
| int value\_Num, float value\_Height, float value\_Width, float value\_Length, string value\_color\_name)  {  Vendor = value\_Vendor; Material = value\_Material; Num = value\_Num; Height = value\_Height; Width = value\_Width; Length = value\_Length;  color\_name = value\_color\_name; count\_table++;  }  float ColorTable::getPloshad()  {  Ploshad = Width\*Length; return Ploshad;  }  float ColorTable::getPerimeter()  {  count++;  Perimetr = 2 \* Width + 2 \* Length; return Perimetr;  }  float ColorTable::getVolume()  {  Volume = Width\*Height\*Length; return Volume;  }  void ColorTable::show()  {  language(); ifstream fin1; ifstream fin2; string path; int key1;  cout << "Хотите ли вы вручную прописать путь к файлу конфигурации?" << endl;  cout << "Do you want to manually specify the path to the configuration file?";  do  {  cout << "\n1 - Да" << endl; cout << "2 - Нет" << endl;  cout << "Ваш выбор: " << endl; key1 = \_getch();  switch (key1)  {  case '1':  cout << "Введите полный путь к  файлу или имя файла: ";  cin >> path; cout << endl; break;  case'2':  path = "Konfig\_file2.txt"; break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE); | SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Измените содержимое файла на число" << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  else  {  cout << width << endl; colortable.SetWidth(width);  }  cout <<  langu.at(15) << endl;  put\_table  >> lenght;  if  (!put\_table || lenght <= 0)  {  put\_table.clear(); put\_table.ignore(std::numeric\_limits<std::streamsize>::max(),  '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Измените содержимое файла на число" << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  else  {  cout << lenght << endl; colortable.SetLength(lenght);  }  cout <<  langu.at(16) << endl;  put\_table  >> height;  if  (!put\_table || height <= 0)  {  put\_table.clear(); put\_table.ignore(std::numeric\_limits<std::streamsize>::max(),  '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Измените содержимое файла на число" << endl; |

|  |  |
| --- | --- |
| break;  }  } while (key1 != 49 && key1 != 50); try  {  func(fin2, path); string str; vector<string> konfig;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  konfig.push\_back(str);  }  fout.close();  func(fin1, konfig.at(0));  vector<string> change;  fin1.open(konfig.at(0)); getline(fin1, str, '\n');  while (getline(fin1, str, '\n'))  {  change.push\_back(str);  }  fin1.close();  cout << change.at(0) << endl;  cout << change.at(1) << GetVendor() << endl; cout << change.at(2) << GetMaterial() << endl; cout << change.at(3) << GetNum() << endl; cout << change.at(4) << GetWidth() << endl; cout << change.at(5) << GetLength() << endl; cout << change.at(6) << GetHeight() << endl; cout << change.at(7) << GetColor() << endl; cout << change.at(8) << getPerimeter() << endl; cout << change.at(9) << getPloshad() << endl; cout << change.at(10) << getVolume() << endl;  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void ColorTable::language()  {  int key1; bool lang; do  {  cout << "\n1 - Русский язык" << endl; cout << "2 - Английский язык" << endl; cout << "Ваш выбор: " << endl;  key1 = \_getch(); switch (key1)  {  case '1':  lang = TRUE;  if (lang == TRUE)  {  lang = FALSE; BOOL bFile; | SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  else  {  cout << height << endl; colortable.SetLength(height);  }  first.AddTable(new ColorTable(s, s1, num, height, width, lenght, s2));  flag = 1;  colortable.show();  break;  case'2':  language(); BuildVector(langu);  cout <<  langu.at(11) << endl;  put\_window >> s1; put\_window >> s1;  cout << s1  << endl;  window.SetMaterial(s1);  cout <<  langu.at(17) << endl;  put\_window >> width;  if  (!put\_window || width <= 0)  {  put\_table.clear(); put\_table.ignore(std::numeric\_limits<std::streamsize>::max(),  '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Измените содержимое файла на число" << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  else  {  cout << width << endl; window.SetWidth(width);  }  cout <<  langu.at(18) << endl; |

|  |  |
| --- | --- |
| bFile = CopyFile(  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\table\_rush.txt",  L"D:\\Курсова  работа\\KYRSACH\\KYRSACH\\Temp.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  }  break;  case'2':  lang = FALSE;  if (lang == FALSE)  {  lang = TRUE; BOOL bFile; bFile = CopyFile(  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\table\_eng.txt",  L"D:\\Курсова  работа\\KYRSACH\\KYRSACH\\Temp.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  break;  }  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50);  }  void ColorTable::func(ifstream & file, string name)  {  file.open(name); if (!file.is\_open())  {  throw "Файл не может быть открыт";  }  if (file.peek() == EOF)  {  throw "Файл пуст";  }  if (file.is\_open() && file.peek() != EOF)  {  int i = 0, k = 0; string str;  file >> i; if (i == 0)  {  throw "Невозможно проверить количества нужных даннных в файле";  }  else  { | put\_window >> lenght;  if  (!put\_window || lenght <= 0)  {  put\_table.clear(); put\_table.ignore(std::numeric\_limits<std::streamsize>::max(),  '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Измените содержимое файла на число" << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  else  {  cout << lenght << endl; window.SetLength(lenght);  }  cout <<  langu.at(19) << endl;  put\_window >> height;  if  (!put\_window || height <= 0)  {  put\_table.clear(); put\_table.ignore(std::numeric\_limits<std::streamsize>::max(),  '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Измените содержимое файла на число" << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  else  {  cout << height << endl; window.SetHeight(height);  }  first.AddWindow(new Window(s, s1, height, width, lenght));  flag = 1;  window.show();  break;  }  } while (key2 != 27); |

|  |  |
| --- | --- |
| getline(file, str); while (!file.eof())  {  str = ""; getline(file, str); k++;  }  if (k < i)  {  throw "В файле  недостаточно строк";  }  }  }  file.close();  }  ColorTable & ColorTable::operator%(const ColorTable &now)  {  Vendor = this->Vendor + now.Vendor; Material = this->Material + now.Material; Num = this->Num + now.Num \* 2; Height = this->Height + now.Height + 54; Width = this->Width + now.Width / 435; Length = this->Length + now.Length - 5;  color\_name = this->color\_name + now.color\_name; return \*this;  }  ColorTable::~ColorTable()  {  Perimetr = 0;  Ploshad = 0;  Volume = 0;  }  **Window.h**  #pragma once #include "Methods.h" #include "Furniture.h" #include <string> using namespace std;  class Window : public Furniture, public Methods  {  protected:  float Height; float Width; float Length;  static int count\_window;  public:  static int getNextID(); Window();  Window(string value\_Vendor, string value\_Material, float value\_Height, float value\_Width, float value\_Length);  float getPerimeter(); float getPloshad(); float getVolume(); float GetHeight(); float GetWidth(); float GetLength(); void SetHeight(float); void SetWidth(float); void SetLength(float); void show();  void func(ifstream &, string); void language();  Window &operator +=(const Window&);  ~Window();  }; | break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  if (flag == 1)  {  do  {  language(); BuildVector(langu); cout << langu.at(20)  << endl;  cout << langu.at(21)  << endl;  cout << langu.at(22)  << endl;  key3 = \_getch(); switch (key3)  {  case'1':  first.DisplayTable(); first.DisplayWindow(); language(); BuildVector(langu);  do  {  cout << langu.at(23) << endl; cout << langu.at(24) << endl; cout << langu.at(25) << endl; key = \_getch();  switch (key)  {  case'1':  first.ReleaseTable(); first.ReleaseWindow(); break;  case'2':  break; default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" |

|  |  |
| --- | --- |
| **Window.cpp** #include "stdafx.h" #include "Window.h" #include <conio.h> #include <string> #include "Methods.h" #include "Furniture.h"  #include <Windows.h> #include <vector> #include <fstream> #include <iostream> using namespace std;  int Window::count\_window = 0;  int Window::getNextID()  {  return count\_window;  }  Window::Window() : Furniture()  {  Height = 0;  Width = 0;  Length = 0;  }  Window::Window(string value\_Vendor, string value\_Material, float value\_Height, float value\_Width, float value\_Length)  {  Vendor = value\_Vendor; Material = value\_Material; Height = value\_Height; Width = value\_Width; Length = value\_Length; count\_window++;  }  void Window::SetHeight(float value\_Height)  {  Height = value\_Height;  }  void Window::SetWidth(float value\_Width)  {  Width = value\_Width;  }  void Window::SetLength(float value\_Length)  {  Length = value\_Length;  }  float Window::getPerimeter()  {  return 2\*Width + 2\*Length;  }  float Window::getPloshad()  {  return Width\*Length;  }  float Window::getVolume()  {  return Width\*Height\*Length;  }  float Window::GetHeight()  { | << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while  (key != 49 && key != 50);  }  } while (key3 != 27);  }  }while (key1 != 27 && key1 != 49);  cout << langu.at(26) << colortable.getNextID() <<  endl;  cout << langu.at(27) << window.getNextID() <<  endl;  put\_table.close(); put\_window.close();  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void menu\_hands(konfig &files)  {  ofstream out; out.open("hello.txt");  if (!out.is\_open())  {  cout << "Ошибка открытия файла для записи  данных!" << endl;  }  try  {  cout << "Выберите язык на котором хотите продолжать работать" << endl;  language(); ifstream fin;  string path = files.at(10); func(fin, path); ColorTable colortable; Window window; Warehouse first; languages langu;  string str;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  langu.push\_back(str);  }  fout.close(); string s, s1, s2, s3; int flag;  int key, key1, key2, key3, num; float width, height, lenght;  cout << langu.at(0) << endl; //Здравствуйте, рады приветствовать вас в нашем магазине!  cout << langu.at(1) << endl; //выберите материал для своей детали и имя человека, который будет вас обслуживать  do |

|  |  |
| --- | --- |
| return Height;  }  float Window::GetWidth()  {  return Width;  }  float Window::GetLength()  {  return Length;  }  void Window::func(ifstream & file, string name)  {  file.open(name); if (!file.is\_open())  {  throw "Файл не может быть открыт";  }  if (file.peek() == EOF)  {  throw "Файл пуст";  }  if (file.is\_open() && file.peek() != EOF)  {  int i = 0, k = 0; string str;  file >> i; if (i == 0)  {  throw "Невозможно проверить количества нужных даннных в файле";  }  else  {  getline(file, str); while (!file.eof())  {  str = ""; getline(file, str); k++;  }  if (k < i)  {  throw "В файле  недостаточно строк";  }  }  }  file.close();  }  void Window::show()  {  language(); ifstream fin2; ifstream fin; string path; int key1;  cout << "Хотите ли вы вручную прописать путь к файлу конфигурации?" << endl;  cout << "Do you want to manually specify the path to the configuration file?";  do  {  cout << "\n1 - Да" << endl; cout << "2 - Нет" << endl;  cout << "Ваш выбор: " << endl; key1 = \_getch(); | {  language(); BuildVector(langu); flag = 0;  cout << langu.at(2) << endl; //1 -  Назвать материал и имя продавца  cout << langu.at(3) << endl; //ESC -  выход  cout << langu.at(4) << endl; //Ваш  выбор:  key1 = \_getch();  switch (key1)  {  case'1':  language(); BuildVector(langu);  cout << langu.at(5) << endl;  //Введите имя продавца который будет вас обслуживать:  cin >> s;  out << s << endl; colortable.SetVendor(s); window.SetVendor(s); do  {  cout << langu.at(6)  << endl; //Посмотрите наш каталог и выберите деталь для изготовки  cout << langu.at(7)  << endl; //1 - Заказать стол  cout << langu.at(8)  << endl; //2 - Заказать окно  cout << langu.at(9)  << endl; //ESC - выход  cout << langu.at(10)  << endl; //Ваш выбор:  key2 = \_getch(); switch (key2)  {  case'1':  language(); BuildVector(langu);  cout <<  langu.at(11) << endl; //Введите материал:  cin >> s1; out << s1  << endl;  colortable.SetMaterial(s1);  cout <<  langu.at(12) << endl; //Введите цвет покраски стола:  cin >> s2; out << s2  << endl;  colortable.SetColor(s2);  cout <<  langu.at(13) << endl; //Введите количество ножек стола =  cin >>  num;  while  (!cin || num <= 0)  {  cin.clear(); |

|  |  |
| --- | --- |
| switch (key1)  {  case '1':  cout << "Введите полный путь к  файлу или имя файла: ";  cin >> path; cout << endl; break;  case'2':  path = "Konfig\_file1.txt"; break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50);  try  {  func(fin2, path); string str; vector<string> konfig;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  konfig.push\_back(str);  }  fout.close();  func(fin, konfig.at(0)); vector<string> round; fin.open(konfig.at(0));  getline(fin, str, '\n');  while (getline(fin, str, '\n'))  {  round.push\_back(str);  }  fin.close();  cout << round.at(0) << endl;  cout << round.at(1) << GetVendor() << endl; cout << round.at(2) << GetMaterial() << endl; cout << round.at(3) << GetWidth() << endl; cout << round.at(4) << GetLength() << endl; cout << round.at(5) << GetHeight() << endl; cout << round.at(6) << getPerimeter() << endl; cout << round.at(7) << getPloshad() << endl; cout << round.at(8) << getVolume() << endl;  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  } | cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << langu.at(34) << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  cin >> num; cout << endl;  }  out <<  num << endl;  colortable.SetNum(num);  cout <<  langu.at(14) << endl; //Введите ширину стола =  cin >>  width;  while  (!cin || width <= 0)  {  cin.clear(); cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << langu.at(34) << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  cin >> width; cout << endl;  }  out <<  width << endl;  colortable.SetWidth(width);  cout <<  langu.at(15) << endl; //Введите длину стола =  cin >>  lenght;  while  (!cin || lenght <= 0)  {  cin.clear(); cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << langu.at(34) << endl; |

|  |  |
| --- | --- |
| }  Window & Window::operator+=(const Window &other)  {  Vendor = this->Vendor.insert(3,other.Vendor); Material = this->Material.insert(2,other.Material); Height = (this->Height / other.Height) +54; Width = this->Width / 456 + other.Width;  Length = this->Length \* other.Length; return \*this;  }  void Window::language()  {  int key1; bool lang; do  {  cout << "\n1 - Русский язык" << endl; cout << "2 - Английский язык" << endl; cout << "Ваш выбор: " << endl;  key1 = \_getch(); switch (key1)  {  case '1':  lang = TRUE;  if (lang == TRUE)  {  lang = FALSE; BOOL bFile; bFile = CopyFile(  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\window\_rush.txt",  L"D:\\Курсова  работа\\KYRSACH\\KYRSACH\\Test.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  }  break;  case'2':  lang = FALSE;  if (lang == FALSE)  {  lang = TRUE; BOOL bFile; bFile = CopyFile(  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\window\_eng.txt",  L"D:\\Курсова  работа\\KYRSACH\\KYRSACH\\Test.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  break;  }  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl; | SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  cin >> lenght; cout << endl;  }  out <<  lenght << endl;  colortable.SetLength(lenght);  cout <<  langu.at(16) << endl; //Введите высоту стола =  cin >>  height;  while  (!cin || height <= 0)  {  cin.clear(); cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << langu.at(34) << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  cin >> height; cout << endl;  }  out <<  height << endl;  colortable.SetHeight(height);  first.AddTable(new ColorTable(s, s1, num, height, width, lenght, s2));  colortable.show();  flag = 1; break;  case'2':  language(); BuildVector(langu);  cout <<  langu.at(11) << endl; //Введите материал:  cin >> s1; out << s1  << endl;  window.SetMaterial(s1);  cout <<  langu.at(17) << endl; //Введите ширину окна =  cin >>  width;  while  (!cin || width <= 0)  {  cin.clear(); |

|  |  |
| --- | --- |
| SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50);  }  Window::~Window()  {  Height = 0;  Width = 0;  Length = 0;  }  **Warehouse.h**  #pragma once  #include "ColorTable.h" #include "Window.h" #include <string>  using namespace std;  class Warehouse  {  struct comp  {  ColorTable \*colortable; struct comp \*next;  };  struct comp \*head; struct comp \*last;  struct list  {  Window \*window; struct list \*current;  };  struct list \*head1; struct list \*last1; int kol\_vo;  public:  Warehouse();  ~Warehouse();  void AddTable(ColorTable\* table); void AddWindow(Window\* window); void DisplayTable();  void DisplayWindow();  void func(ifstream &, string); void ReleaseTable();  void ReleaseWindow();  void ReleaseTable\_command(); void ReleaseWindow\_command(); void language();  };  **Warehouse.cpp** #include "stdafx.h" #include <stdio.h> #include <string.h> #include "Warehouse.h" #include <iostream> #include <conio.h> #include <Windows.h> #include <vector> #include <fstream> using namespace std;  Warehouse::Warehouse()  {  head = NULL; last = NULL; | cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << langu.at(34) << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  cin >> width; cout << endl;  }  out <<  width << endl;  window.SetWidth(width);  cout <<  langu.at(18) << endl; //Введите длину окна =  cin >>  lenght;  while  (!cin || lenght <= 0)  {  cin.clear(); cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << langu.at(34) << endl; SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED |  FOREGROUND\_GREEN | FOREGROUND\_BLUE);  cin >> lenght; cout << endl;  }  out <<  lenght << endl;  window.SetLength(lenght);  cout <<  langu.at(19) << endl; //Введите высоту окна =  cin >>  height;  while  (!cin || height <= 0)  {  cin.clear(); cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << langu.at(34) << endl; |

|  |  |
| --- | --- |
| head1 = NULL; last1 = NULL; kol\_vo = 0;  }  Warehouse::~Warehouse()  {  ReleaseTable\_command(); ReleaseWindow\_command();  }  void Warehouse::AddTable(ColorTable\* colortable)  {  kol\_vo += 1;  comp \*q = new comp; q->next = NULL;  q->colortable = colortable;  if (last != NULL)  {  (last)->next = q;  }  last = q;  if (head == NULL)  {  head = last;  }  }  void Warehouse::AddWindow(Window\* window)  {  kol\_vo += 1;  list \*q = new list;  q->current = NULL;  q->window = window;  if (last1 != NULL)  {  (last1)->current = q;  }  last1 = q;  if (head1 == NULL)  {  head1 = last1 ;  }  }  void Warehouse::DisplayTable()  {  language(); ifstream fin1; ifstream fin2; string path; int key1;  cout << "Хотите ли вы вручную прописать путь к файлу конфигурации?" << endl;  cout << "Do you want to manually specify the path to the configuration file?";  do  {  cout << "\n1 - Да" << endl; cout << "2 - Нет" << endl;  cout << "Ваш выбор: " << endl; key1 = \_getch();  switch (key1)  {  case '1':  cout << "Введите полный путь к  файлу или имя файла: ";  cin >> path; | SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  cin >> height; cout << endl;  }  out <<  height << endl;  window.SetHeight(height);  first.AddWindow(new Window(s, s1, height, width, lenght)); window.show();  flag = 1; break;  }  } while (key2 != 27); break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  if (flag == 1)  {  do  {  language(); BuildVector(langu); cout << langu.at(20)  << endl; //Все ваши заказы прибыли на склад, хотите посмотреть общий список?  cout << langu.at(21)  << endl; //1 - Да  cout << langu.at(22)  << endl; //ESC - Нет  key1 = 0;  key3 = \_getch(); switch (key3)  {  case'1':  first.DisplayTable(); first.DisplayWindow(); language();  BuildVector(langu);  do  {  cout << langu.at(23) << endl; //Хотите очистить склад? cout << langu.at(24) << endl; //1 - Да  cout << langu.at(25) << endl; //2 - Нет key = \_getch();  switch (key) |

|  |  |
| --- | --- |
| cout << endl; break;  case'2':  path = "Konfig\_file3.txt"; break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50);  try  {  func(fin2, path); string str; vector<string> konfig;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  konfig.push\_back(str);  }  fout.close();  func(fin1, konfig.at(0));  vector<string> change;  fin1.open(konfig.at(0)); getline(fin1, str, '\n');  while (getline(fin1, str, '\n'))  {  change.push\_back(str);  }  fin1.close(); comp \*x = head; int i = 0;  if (head != NULL)  {  while (head != NULL)  {  i = i + 1;  cout << change.at(0) << i <<  endl << endl;  cout << change.at(1) << head->colortable->GetMaterial() << endl;  cout << change.at(2) <<  head->colortable->GetColor() << endl;  cout << change.at(3) <<  head->colortable->GetNum() << endl;  cout << change.at(4) << head->colortable->GetHeight() << endl;  cout << change.at(5) << head->colortable->GetWidth() << endl;  cout << change.at(6) << head->colortable->GetLength() << endl;  head = head->next;  }  head = x;  }  if (kol\_vo == 0)  { | {  case'1':  first.ReleaseTable(); first.ReleaseWindow(); break;  case'2':  break; default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод"  << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while  (key != 49 && key != 50);  }  } while (key3 != 27);  }  } while (key1 != 27 && key1 != 49);  cout << langu.at(26) << colortable.getNextID() << endl; //Количество созданных объектов(столов) =  cout << langu.at(27) << window.getNextID() << endl; //Количество созданных объектов(окон) =  out.close();  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void vce\_peregryzki(konfig &files)  {  int key;  cout << "Выберите язык на котором хотите продолжать работать" << endl;  language(); ifstream fout;  string path = files.at(10); try  {  func(fout, path); fout.open(path); languages langu; string str; getline(fout, str);  while (getline(fout, str, '\n'))  {  langu.push\_back(str); |

|  |  |
| --- | --- |
| cout << change.at(7) << endl;  }  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void Warehouse::DisplayWindow()  {  ifstream fin1; ifstream fin2; string path; int key1;  cout << "Хотите ли вы вручную прописать путь к файлу конфигурации?" << endl;  cout << "Do you want to manually specify the path to the configuration file?";  do  {  cout << "\n1 - Да" << endl; cout << "2 - Нет" << endl;  cout << "Ваш выбор: " << endl; key1 = \_getch();  switch (key1)  {  case '1':  cout << "Введите полный путь к  файлу или имя файла: ";  cin >> path; cout << endl; break;  case'2':  path = "Konfig\_file3.txt"; break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50); try  {  func(fin2, path); string str; vector<string> konfig;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  konfig.push\_back(str);  }  fout.close(); | }  fout.close(); do  {  cout << langu.at(28) << endl; cout << langu.at(29) << endl; cout << langu.at(30) << endl; cout << langu.at(31) << endl; cout << langu.at(32) << endl; cout << langu.at(33) << endl; key = \_getch();  switch (key)  {  case'1':  Peregryz1(); break;  case'2':  Peregryz2(); break;  case'3':  Peregryz3(); break;  case'4':  Peregryz4(); break;  case'5':  Peregryz5(); break;  }  } while (key != 27);  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void begin(konfig &files)  {  ifstream fout; string path; int key;  do  {  cout << endl;  cout << "На каком языке начать работу?" <<  endl;  cout << "What language do you want to begin working in ?" << endl;  cout << "\n1 - Русский язык" << endl; cout << "2 - English language" << endl; cout << "Ваш выбор: " << endl;  key = \_getch(); switch (key)  {  case '1':  path = files.at(0); break;  case '2':  path = files.at(1); break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное |

|  |  |
| --- | --- |
| func(fin1, konfig.at(0));  vector<string> change;  fin1.open(konfig.at(0)); getline(fin1, str, '\n');  while (getline(fin1, str, '\n'))  {  change.push\_back(str);  }  fin1.close(); list \*x = head1; int i = 0;  if (head1 != NULL)  {  while (head1 != NULL)  {  i = i + 1;  cout << endl << change.at(8)  << i << endl << endl;  cout << change.at(9) << head1->window->GetMaterial() << endl;  cout << change.at(10) << head1->window->GetHeight() << endl;  cout << change.at(11) <<  head1->window->GetWidth() << endl;  cout << change.at(12) << head1->window->GetLength() << endl;  head1 = head1->current;  }  head1 = x;  }  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void Warehouse::func(ifstream &file, string name)  {  file.open(name); if (!file.is\_open())  {  throw "Файл не может быть открыт";  }  if (file.peek() == EOF)  {  throw "Файл пуст";  }  if (file.is\_open() && file.peek() != EOF)  {  int i = 0, k = 0; string str;  file >> i; if (i == 0)  {  throw "Невозможно проверить количества нужных даннных в файле";  }  else  {  getline(file, str); | значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key != 49 && key != 50); try  {  func(fout, path); fout.open(path); string str; getline(fout, str); while (!fout.eof())  {  str = ""; getline(fout, str); cout << str << endl;  }  fout.close();  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void work\_menu(konfig &files)  {  ifstream fout; string path; int key;  do  {  cout << endl;  cout << "На каком языке продолжить работу?"  << endl;  cout << "What language do you want to continue working in ?" << endl;  cout << "\n1 - Русский язык" << endl; cout << "2 - English language" << endl; cout << "Ваш выбор: " << endl;  key = \_getch(); switch (key)  {  case '1':  path = files.at(2);; break;  case '2':  path = files.at(3);; break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key != 49 && key != 50); try  {  func(fout,path); |

|  |  |
| --- | --- |
| while (!file.eof())  {  str = ""; getline(file, str); k++;  }  if (k < i)  {  throw "В файле  недостаточно строк";  }  }  }  file.close();  }  void Warehouse::ReleaseTable()  {  kol\_vo = 0;  if (head != NULL)  {  struct comp \*p;  while (head != NULL)  {  p = head;  head = head->next; delete p;  }  head = NULL;  }  try  {  language(); ifstream fin1; ifstream fin2; string path; int key1;  cout << "Хотите ли вы вручную прописать путь к файлу конфигурации?" << endl;  cout << "Do you want to manually specify the path to the configuration file?";  do  {  cout << "\n1 - Да" << endl; cout << "2 - Нет" << endl;  cout << "Ваш выбор: " << endl; key1 = \_getch();  switch (key1)  {  case '1':  cout << "Введите полный  путь к файлу или имя файла: ";  cin >> path; cout << endl; break;  case'2':  path = "Konfig\_file3.txt"; break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE); | fout.open(path); string str; getline(fout, str); while (!fout.eof())  {  str = ""; getline(fout, str); cout << str << endl;  }  fout.close();  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void rykovodstvo\_user(konfig &files)  {  ifstream fout; string path; int key;  do  {  cout << endl;  cout << "На каком языке хотите продолжать  работу?" << endl;  cout << "What language do you want to continue working in ?" << endl;  cout << "\n1 - Русский язык" << endl; cout << "2 - English language" << endl; cout << "Ваш выбор: " << endl;  key = \_getch(); switch (key)  {  case '1':  path = files.at(4); break;  case '2':  path = files.at(5); break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key != 49 && key != 50); try  {  func(fout, path); fout.open(path); string str; getline(fout, str); while (!fout.eof())  {  str = ""; getline(fout, str); cout << str << endl;  }  fout.close(); |

|  |  |
| --- | --- |
| break;  }  } while (key1 != 49 && key1 != 50); func(fin2, path);  string str; vector<string> konfig;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  konfig.push\_back(str);  }  fout.close();  func(fin1, konfig.at(0)); vector<string> change;  fin1.open(konfig.at(0)); getline(fin1, str, '\n');  while (getline(fin1, str, '\n'))  {  change.push\_back(str);  }  fin1.close();  cout << change.at(13) << endl;  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void Warehouse::ReleaseWindow()  {  kol\_vo = 0;  if (head1 != NULL)  {  struct list \*p;  while (head1 != NULL)  {  p = head1;  head1 = head1->current; delete p;  }  head1 = NULL;  }  }  void Warehouse::ReleaseTable\_command()  {  if (head != NULL)  {  struct comp \*p;  while (head != NULL)  {  p = head;  head = head->next; delete p;  }  head = NULL;  } | }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  void help(konfig &files)  {  ifstream fout; string path; int key;  do  {  cout << endl;  cout << "На каком языке продолжить работу?"  << endl;  cout << "What language do you want to continue working in ?" << endl;  cout << "\n1 - Русский язык" << endl; cout << "2 - English language" << endl; cout << "Ваш выбор: " << endl;  key = \_getch(); switch (key)  {  case '1':  path = files.at(6); break;  case '2':  path = files.at(7); break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key != 49 && key != 50); try  {  func(fout, path); fout.open(path); string str; getline(fout, str); while (!fout.eof())  {  str = ""; getline(fout, str); cout << str << endl;  }  fout.close();  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | |

|  |  |
| --- | --- |
| }  void Warehouse::ReleaseWindow\_command()  {  if (head1 != NULL)  {  struct list \*p;  while (head1 != NULL)  {  p = head1;  head1 = head1->current; delete p;  }  head1 = NULL;  }  }  void Warehouse::language()  {  int key1; bool lang;  cout << "\n1 - Русский язык" << endl; cout << "2 - Английский язык" << endl; cout << "Ваш выбор: " << endl;  key1 = \_getch(); do  {  switch (key1)  {  case '1':  lang = TRUE;  if (lang == TRUE)  {  lang = FALSE; BOOL bFile; bFile = CopyFile(  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\sklad\_rush.txt",  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\Experiment.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  }  break;  case'2':  lang = FALSE;  if (lang == FALSE)  {  lang = TRUE; BOOL bFile; bFile = CopyFile(  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\sklad\_eng.txt",  L"D:\\Курсова работа\\KYRSACH\\KYRSACH\\Experiment.txt",  FALSE);  if (bFile == FALSE)  {  cout << "CopyFile  Failed" << GetLastError() << endl;  }  break;  }  default:  HANDLE hOUTPUT = | FOREGROUND\_BLUE);  }  }  void about\_athour(konfig &files)  {  ifstream fout; string path; int key;  do  {  cout << endl;  cout << "На каком языке хотите продолжать  работу?" << endl;  cout << "What language do you want to continue working in ?" << endl;  cout << "\n1 - Русский язык" << endl; cout << "2 - English language" << endl; cout << "Ваш выбор: " << endl;  key = \_getch(); switch (key)  {  case '1':  path = files.at(8); break;  case '2':  path = files.at(9); break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key != 49 && key != 50); try  {  func(fout, path); fout.open(path); string str; getline(fout, str); while (!fout.eof())  {  str = ""; getline(fout, str); cout << str << endl;  }  fout.close();  }  catch (const char \*ex)  {  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,  FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  }  }  int main()  {  setlocale(LC\_ALL, "rus"); int key1;  ifstream fin; string path; |

|  |  |
| --- | --- |
| GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50);  }  **Mathods.h** #pragma once class Methods  {  public:  virtual float getPloshad() = 0; virtual float getPerimeter() = 0; virtual float getVolume() = 0;  }; | cout << "Хотите ли вы вручную прописать путь к файлу конфигурации?" << endl;  cout << "Do you want to manually specify the path to the configuration file?";  do  {  cout << "\n1 - Да" << endl; cout << "2 - Нет" << endl;  cout << "Ваш выбор: " << endl; key1 = \_getch();  switch (key1)  {  case '1':  cout << "Введите полный путь к файлу  или имя файла: ";  cin >> path; cout << endl; break;  case'2':  path = "Konfig\_file.txt"; break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);  cout << "Введено неправильное значение, повторите ввод" << endl;  SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);  break;  }  } while (key1 != 49 && key1 != 50);  try  {  func(fin, path); konfig files; string str;  ifstream fout(path); getline(fout, str, '\n');  while (getline(fout, str, '\n'))  {  files.push\_back(str);  }  fout.close();  begin(files); do  {  work\_menu(files); key1 = \_getch(); switch (key1)  {  case'q':  rykovodstvo\_user(files); break;  case'i':  help(files); break;  case'b':  about\_athour(files); break;  case 13:  break;  case 27:  break;  default:  HANDLE hOUTPUT =  GetStdHandle(STD\_OUTPUT\_HANDLE); |

SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

cout << "Введено неправильное значение, повторите ввод" << endl;

SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

break;

}

} while (key1 != 27 && key1 != 13); if (key1 != 27)

{

do

{

cout << "Каким образом вы

хотите ввести данные в программу?" << endl;

cout << "\n1 - Ввести с

клавиатуры" << endl;

<< endl; endl;

cout << "2 - Ввести с файла"

cout << "ESC - Exit" << endl; cout << "Ваш выбор: " <<

key1 = \_getch(); switch (key1)

{

case '1':

case'2':

}

menu\_hands(files); break;

menu\_file(files); break;

} while (key1 != 27);

vce\_peregryzki(files);

}

}

catch (const char \*ex)

{

HANDLE hOUTPUT =

GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hOUTPUT, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

cout << ex << endl; SetConsoleTextAttribute(hOUTPUT,

FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

}

system("pause"); return 0;

}

ПРИЛОЖЕНИЕ Е ФАЙЛЫ ИСХОДНЫХ ДАННЫХ

Для работы создаётся ресурсные файлы и исходные.

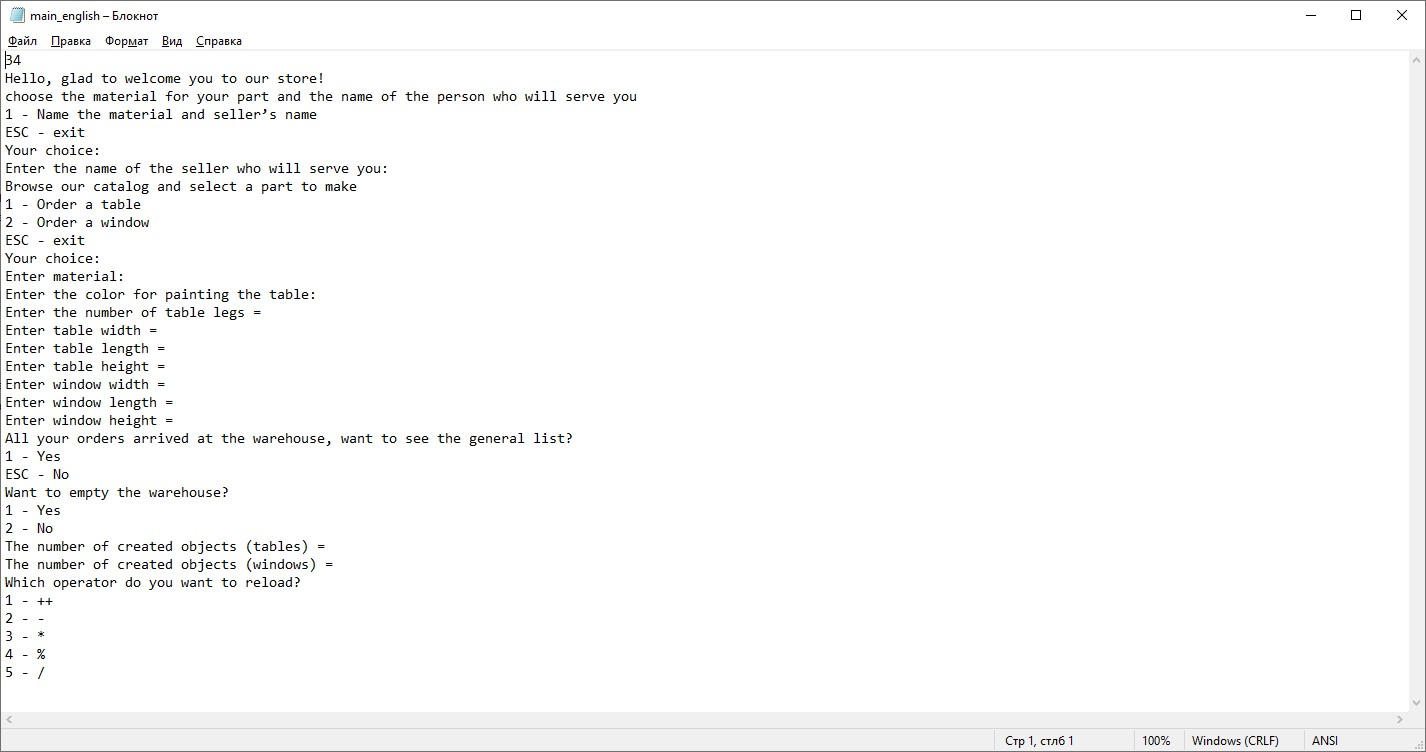


Рисунок Е.1 – файл с английским меню

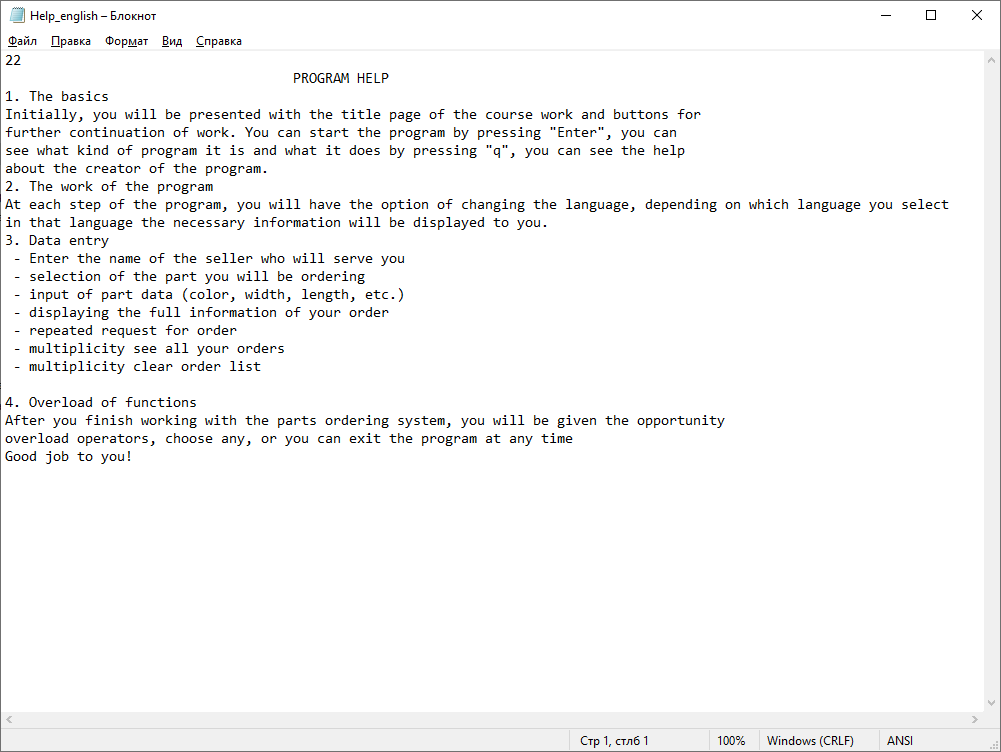


Рисунок Е.2 – файл – помощь пользователю

На рисунке Е.1, и рисунке Е.2 показан пример организации файла-меню и файла-помощи для всего проекта. Для классов и всего остального файлы подобны.

ПРИЛОЖЕНИЕ Ж ФАЙЛЫ ВХОДНЫХ ДАННЫХ

На рисунках, находящихся в приложении «З», представлены тексты файлов, образованных после выполнения программы В течении всей программы, вся информация, введенная пользователем, записывается в текстовый файл.

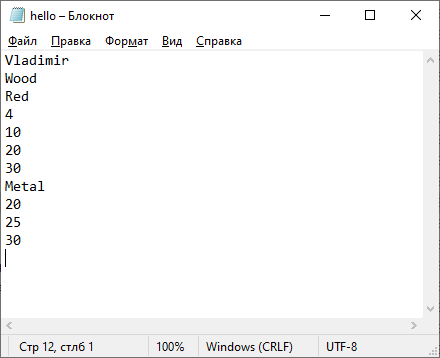


Рисунок Ж.1 – файл “hello.txt” после окончания работы программы

ПРИЛОЖЕНИЕ З ЭКРАННЫЕ ФОРМЫ

Рассмотрим работу с языковым меню, главным меню и одним из объектов.

Для других объектов экранные формы будут подобны.

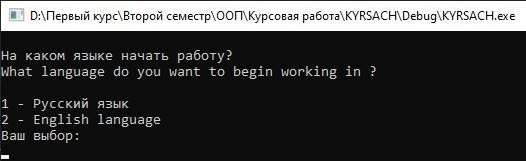


Рисунок З.1- начало работы

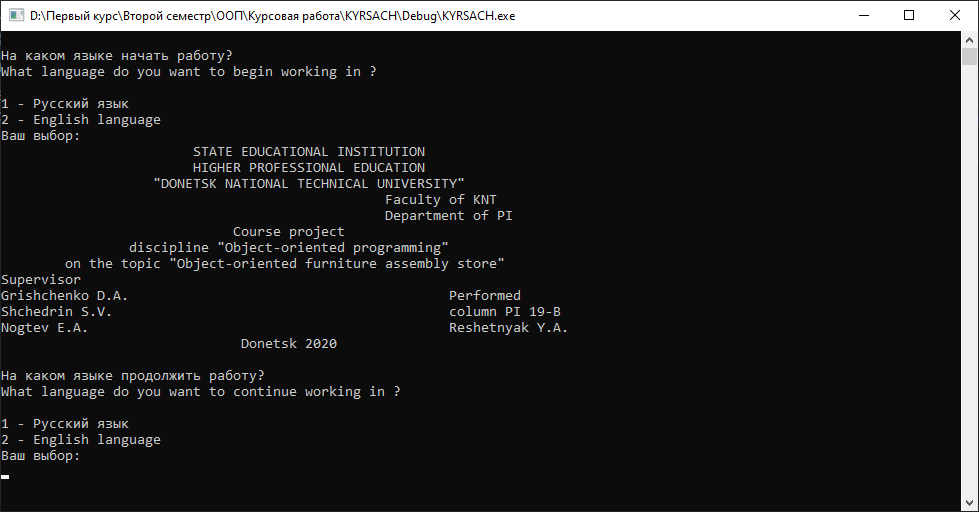


Рисунок З.2- титульный лист

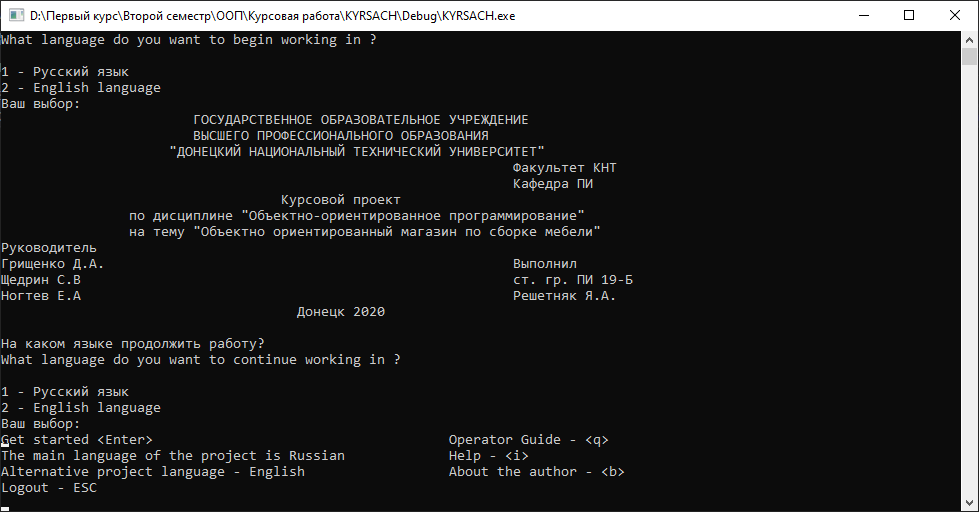


Рисунок З.3- главное меню программы

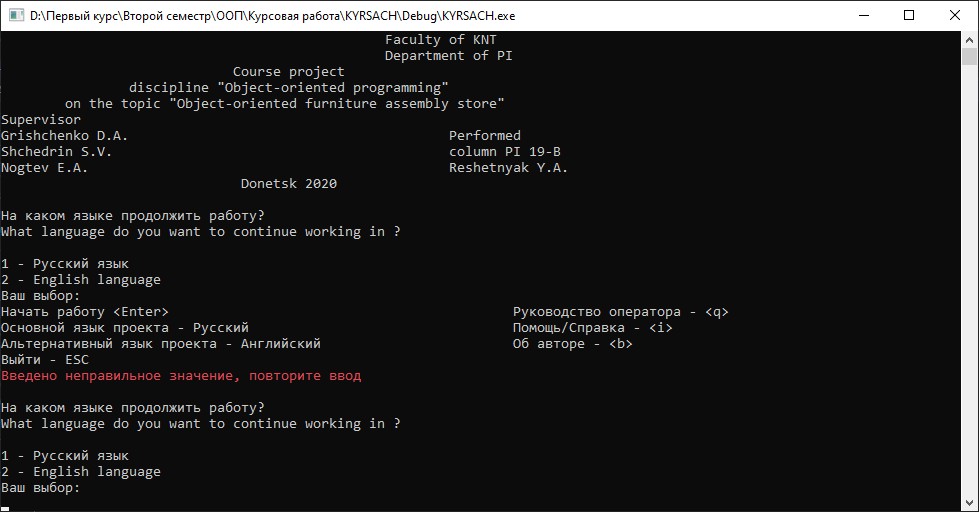


Рисунок З.4- неверный выбор в главном меню

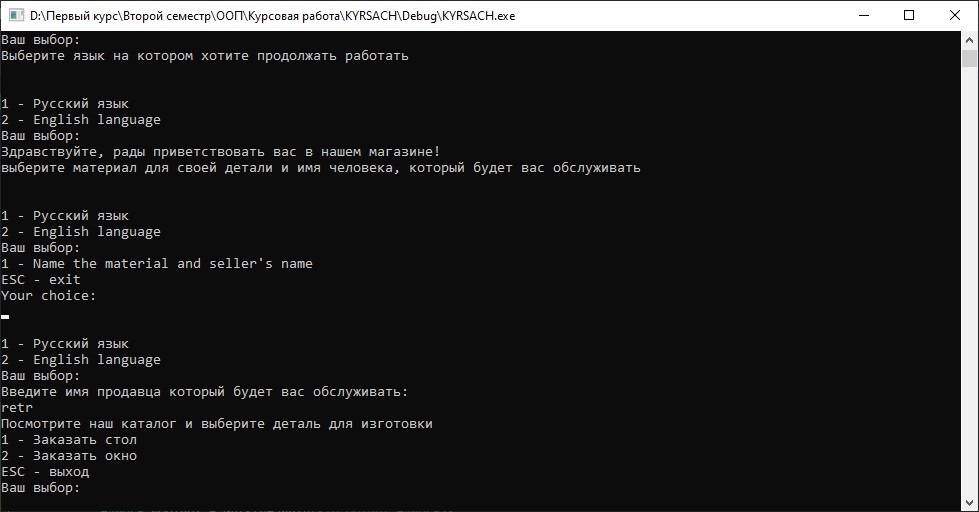


Рисунок З.5 – меню для выбора создания объекта

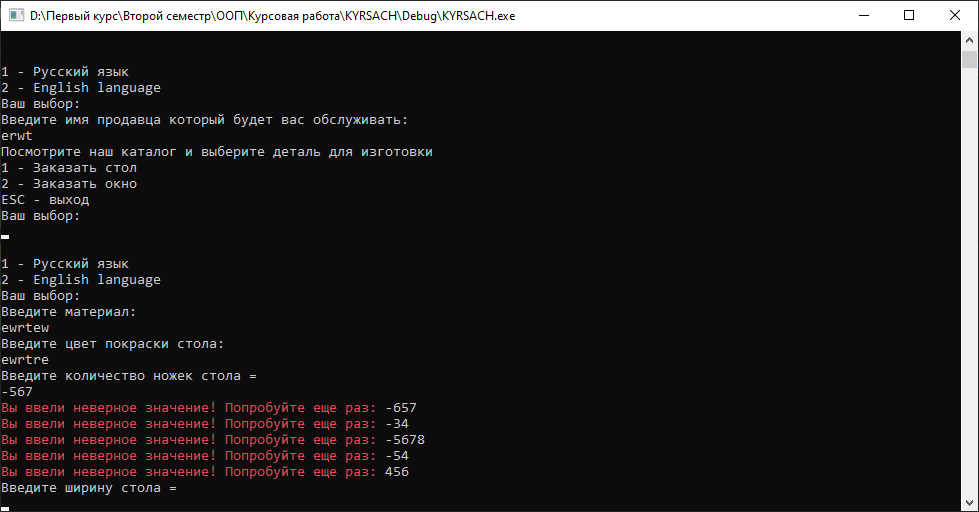


Рисунок З.6 – устанавливаем количество ножек стола неверным значением