МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

КАФЕДРА ПРОГРАМНОЙ ИНЖЕНЕРИИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Объектно-ориентированное программирование»

на тему: «Объектно-ориентированная система

«Животный приют и собачий отдел»

|  |  |
| --- | --- |
| Руководитель:  кафедры ПИ | Выполнил:  Студент гр. ПИ-18а  Арефьев В. И. |

Донецк – 2019

Реферат

Пояснительная записка к курсовому проекту содержит: 84 страницы, 45 рисунков, 8 таблиц, 5 источников, 8 приложений.

Целью курсового проекта является закрепление теоретических знаний по дисциплине, а также практических навыков по разработке приложения с использованием объектно-ориентированного подхода.

Для достижения поставленной цели курсового проекта были решины следующие задачи:

* выполнен объектно-ориентрованный анализ предметной области;
* разработан комплекс статических и динамических моделей предметной области;
* осуществлено техническое и рабочее проектирование объектно-ориентированной системы;
* выполнен программную реализацию и тестирование спроектированного программного комплекса.

Методы исследования – объектно-ориентированный анализ и проектирование, техническое и рабочее проектирование, имитационное моделирование, объектно-ориентированное программирование.

Объект исследования – животный приют и магазин роботов.

Результаты работы – объектно-ориентированная система VAC.exe, являющаяся аналогом виртуального животного приюта.

С++, ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ, Приют, Магазин, Язык uml

СОДЕРЖАНИЕ

[Введение 5](#_Toc9027819)

[1 Постановка задачи 6](#_Toc9027820)

[2 Объектно-ориентированное программирование 8](#_Toc9027821)

[2.1 Объектно-ориентированный анализ 8](#_Toc9027822)

[2.2 Объектно-ориентированное проектирование 9](#_Toc9027823)

[2.2.1 Структура классов 9](#_Toc9027824)

[2.2.2 Отношения между классами 15](#_Toc9027825)

[2.2.3 Структура объектов 16](#_Toc9027826)

[2.2.4 Диаграмма состояний 17](#_Toc9027827)

[2.2.5 Диаграмма последовательности 18](#_Toc9027828)

[3 Рабочее проектирование 20](#_Toc9027829)

[3.1. Структура данных приложения и методы работы с ними 20](#_Toc9027830)

[3.2 Разработка классов 23](#_Toc9027831)

[3.2.1 Реализация инкапсуляции 23](#_Toc9027832)

[3.2.2 Реализация наследования 24](#_Toc9027833)

[3.2.3 Реализация полиморфизма 24](#_Toc9027834)

[3.3 Обоснование выбора алгоритмов 26](#_Toc9027835)

[3.3.1 Работа с файлами 28](#_Toc9027836)

[3.3.2 Подсчет количества экземпляров класса. 32](#_Toc9027837)

[3.3.3 Организация информационной поддержки 32](#_Toc9027838)

[3.4 Тестирование работоспособности системы классов 34](#_Toc9027839)

[Выводы 36](#_Toc9027840)

[Приложение А. Техническое задание 38](#_Toc9027841)

[Приложение Б. Описание программы 48](#_Toc9027842)

[Приложение В. Руководство программиста 49](#_Toc9027843)

[Приложение Г. Руководство оператора 50](#_Toc9027844)

[Приложение Д. Текст программы 51](#_Toc9027845)

[Приложение Е. Файлы исходных данных 74](#_Toc9027846)

[Приложение Ж. Файлы выходных данных 84](#_Toc9027847)

[Приложение К. Экранные формы 86](#_Toc9027848)

# Введение

На современном этапе развития программного обеспечения все большая роль отводится объектно-ориентированному стилю проектирования и программирования (ООП). Объектно-ориентированное программирование (ООП) позволяет разложить проблему на составные части, каждая из которых становится самостоятельным объектом. Каждый из объектов содержит свой собственный код и данные, которые относятся к этому объекту. Любая программа, написанная на языке ООП, отражает в своих данных состояние физических предметов либо абстрактных понятий – объектов программирования, для работы, с которыми она предназначена. Это позволяет разрабатывать большие программные комплексы быстрее, используя повторно имеющие модули, обеспечивать инкапсуляцию данных, полиморфизм операций, наследование свойств и методов. Поэтому изучение и практическое освоение специалистами по программной инженерии этого стиля программирования является актуальной задачей.

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.

В данной работе объектно-ориентированная подход применяется для моделирования процесса имитации работы виртуального животного приюта.

В первом разделе дана краткая характеристика основным объектам моделируемой предметной области.

Во втором разделе описаны дополнительные сущности, построены статические и динамические модели предметной области, приведены результаты объектно-ориентированного анализа и проектирования.

Третий раздел содержит описание программной реализации главных принципов ООП, применяемых алгоритмов моделирования, структур данных и файлов программы.

# 1 Постановка задачи

Создать класс Animal, включающий в себя следующие элементы:

- поле «вес» float Mass;

- поле «пол» string Sex;

- поле «цвет» string Color;

- поле «возраст» int Age;

- метод получения веса float GetMass();

- метод получения пола string GetSex();

- метод получения возраста int GetAge();

- метод получения цвета string GetColor();

- конструктор с параметрами Animal(float Mass, string Sex, string Color, int Age).

Унаследовать от класса Animal класс Dog, содержащий в себе элементы:

- поле «кличка» string Name;

- поле «порода» string Race;

- поле «вид» (охотничье, декоративная, бойцовая и т.д.) string Type;

- конструктор с параметрами Dog(float Mass, string Sex, string Color, int Age, string Name, string Race, string Type).

В классе Animal перегрузить оператор -- (унарный оператор класса), новая действие - уменьшить вес на 0.1 кг (вес должен быть не менее 0,1 кг).

В ходе разработки программы была расширена и изменена предметная область.

Класс Animal стал абстрактным.

Унаследовали от класса Animal класс Cat, содержащий в себе элементы:

- поле «кличка» string Name;

- поле «порода» string Race;

- конструктор с параметрами Cat(float Mass, string Sex, string Color, int Age, string Name, string Race).

Создан класс Robot, включающий в себя следующие элементы:

-поле «серийный номер» int m\_Series;

- поле «масса» float m\_Mass;

-поле «модель» string m\_Model;

-поле «дата» string m\_Date;

-поле «цена» int m\_Cost;

- конструктор с параметрами Robot (int series, float mass, string model, string date, int cost).

Унаследовали от класса Dog и Robot класс RoboDog , содержащий в себе элементы:

-поле «Процент роботизированности» int m\_PerRobo;

- конструктор с параметрами RoboDog(float mass, string sex, string color, int age, string race, string type, string name, int series, string model, string date, int cost).

Создан класс Shelter, включающий в себя следующие элементы:

-поле «список» struct list;

-поле «размер» int size;

-поле «деньи покупателя» int moneyB;

-конструктор без параметров Shelter();

В класах Robot, RoboDog, Сat и Dog добавлены Set-методы и Get-методы.

# 2 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ

## 2.1 Объектно-ориентированный анализ

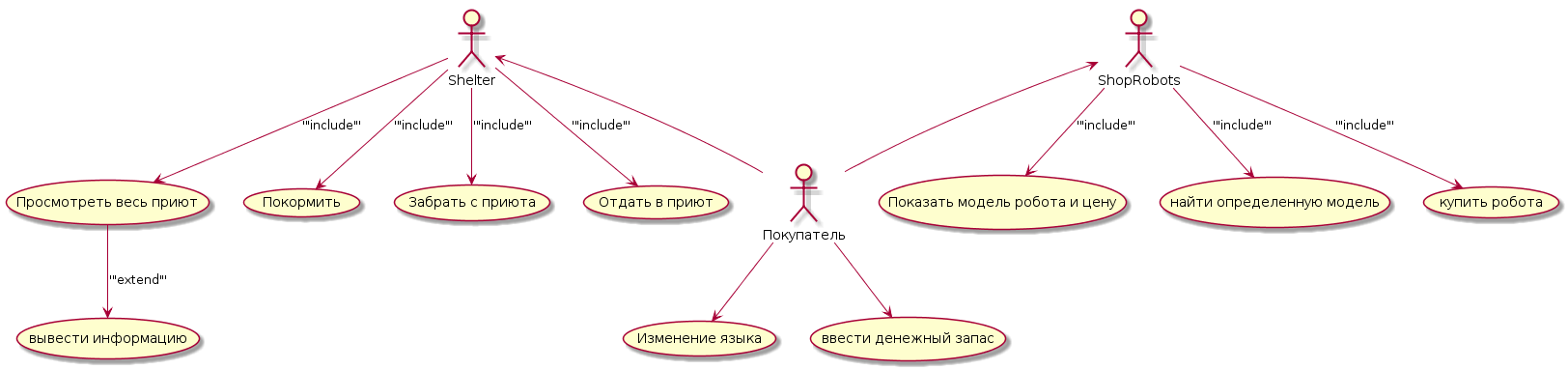
Основная идея объектно-ориентированного анализа и проектирования состоит в рассмотрении предметной области и логического решения задачи с точки зрения объектов (понятий и сущностей). В процессе объектно-ориентированного анализа основное внимание уделяется определению и описанию объектов (или понятий) в терминах предметной области. 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. И, наконец, в процессе конструирования или объектно-ориентированного программирования обеспечивается реализация разработанных компонентов и классов, что отображено на диаграмме прецедентов (см. рис. 2.1). 

Рисунок 2.1 – Обобщенная диаграмма прецедентов системы

Покупатель может забирать из приюта и отдавать животных в приют. Забирая животное, покупатель вносит сумму в фонд приюта. Отдавая животное, приют оплачивает стоимость животного. В магазине роботехники покупатель может приобрести робота. Покупатель может изменить вес собаки или кота, покормив их, или съесть еду самому. Покупатель может приобрести механизированного робопса, полученного путем скрещивания собаки и робота. Также покупатель может просмотреть каталог животных, роботов и робопсов.

Программа должна предоставлять пользователю интерфейс для работы с методами и свойствами объектов предметной области. Интерфейс программы должен быть организован в виде многоуровневого меню, предоставляющего пользователю доступ ко всем возможностям программы.

## 2.2 Объектно-ориентированное проектирование

В процессе объектно-ориентированного проектирования определяются логические программные объекты, которые будут реализованы средствами объектно-ориентированного языка программирования [1]. Эти программные объекты включают в себя атрибуты и методы.

## 2.2.1 Структура классов

Часть классов разработанной объектно-ориентированной системы являются производными от абстрактного класса Animal(Таблица 2.1). Также приведены спецификации всех классов программы (Таблица 2.2, Таблица 2.3, Таблица 2.4, Таблица 2.5, Таблица 2.6, Таблица 2.7, Таблица 2.8).

Таблица 2.1 – Спецификация класса Animal

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| float m\_Mass | Масса животного |
| string m\_Sex | Пол животного |
| string m\_Color | Цвет шерсти животного |
| int m\_Age | Возраст животного |
| Animal() | Конструктор без параметров |
| Animal(float mass, string sex, string color, int age) | Конструктор с параметрами |
| virtual ~Animal() = 0; | Чистый виртуальный деструктор |
| void setMass(); | Метод установление массы |
| void setSex(); | Метод установление пола |
| void setColor(); | Метод установление цвета |
| void setAge(); | Метода установление возраста |
| virtual void print() const = 0; | Чисто виртуальная функция вывода информации о животном |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| virtual const string getName() const = 0; | Чисто виртуальная функция получения клички |
| virtual const string getRace()const = 0; | Чисто виртуальная функция получения породы |
| virtual const int getCost() const = 0; | Чисто виртуальная функция получения цены |
| virtual void saveD() = 0; | Чисто виртуальная функция для сохранения информации в файл |
| const float getMass() | Получить массу |
| const string getColor() | Получить цвет |
| const string getSex() | Получить пол |
| const int getAge() | Получить возраст |
| Animal& operator--() | Уменьшение массы на 0,1 кг |
| friend Animal& operator++(Animal& am) | Увеличение массы на 0,1 кг |
| friend ostream& operator<<(ostream& out, const Animal& an) | Перегрузка оператора вывода |

Таблица 2.2 – Спецификация класса Dog

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| string m\_Name; | Кличка собаки |
| string m\_Race; | Порода собаки |
| string m\_Type; | Вид собаки |
| int m\_Cost; | Стоимость собаки |
| static int cntDog | Количество объектов класса |
| Dog() | Конструктор без параметров |
| Dog(float mass, string sex, string color, int age, string race, string type, string name, int cost) | Конструктор с парметрами |
| virtual~Dog() | Деструктор |
| void setName(); | Установление клички |
| void setRace(); | Установление порода |
| void setType(); | Установление вида |
| void setCost(); | Установление цены |
| virtual void print() const; | Метод вывода информации |
| virtual void saveD(); | Метод сохранения информации в файл |
| const int getCost()const | Получить стоимость |
| const string getName()const | Получить кличку |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| const string getRace()const | Получить породу |
| const string getType()const | Получить вид |
| friend istream& operator>>(istream& in, Dog& dog) | Перегрузка оператора ввода |
| friend ifstream& operator>>(ifstream& stin, Dog& dog) | Перегрузка оператора считывания из файла |
| friend int getCntDog(); | Получить количество объектов класса |

Таблица 2.3 – Спецификация класса Сat

|  |  |
| --- | --- |
| Поле или метод класса | Описание |
| string m\_Name; | Кличка кошки |
| string m\_Race; | Порода кошки |
| int m\_Cost; | Стоимость кошки |
| static int cntCat; | Количество объектов класса |
| Cat() | Конструктор без параметров |
| Cat(float mass, string sex, string color, int age, string race, string name, int cost) | Конструктор с парметрами |
| ~Cat() | Деструктор |
| void setName(); | Установление клички |
| void setRace(); | Установление порода |
| void setCost(); | Установление цены |
| virtual void print() const; | Метод вывода информации |
| virtual void saveD(); | Метод сохранения информации в файл |
| const int getCost()const | Получить стоимость |
| const string getName()const | Получить кличку |
| const string getRace()const | Получить породу |
| friend istream& operator>>(istream& in, Cat& cat) | Перегрузка оператора ввода |
| friend ifstream& operator>>(ifstream& stin, Cat& cat) | Перегрузка оператора считывания из файла |
| friend int getCntCat(); | Получить количество объектов класса |

Таблица 2.4 – Спецификация класса Robot

|  |  |
| --- | --- |
| Поля или методы класса | Описание |
| int m\_Series | Серийный номер робота |
| float m\_Mass | Масса робота |
| string m\_Model | Модель робота |

Продолжение таблицы 2.4

|  |  |
| --- | --- |
| string m\_Date | Дата изготовления робота |
| int m\_Cost | Стоимость робота |
| static int cntRobot | Количество объектов класса |
| Robot() | Конструктор без парметров |
| Robot(int series, float mass, string model, string date, int cost) | Конструктор с парметрами |
| virtual ~Robot() | Виртуальный деструктор |
| void setSeries() | Установление серийного номера |
| void setMass() | Установление массы |
| void setModel() | Установление модели |
| void setDate() | Установление даты изготовления |
| void setCost() | Установление стоимости |
| virtual void saveDR() | Сохранение информации в файл |
| virtual void downL() | Загрузка информации из файла |
| virtual void print() const | Вывод информации |
| const int getSeries() const | Получить серийный номер |
| const float getMass() const | Получить массу |
| const string getModel() const | Получить модель |
| const string getDate()const | Получить дату изготовления |
| const int getCost()const | Получить цену |
| friend ostream& operator<<(ostream& out, const Robot& rb) | Перегрузка оператора вывода |
| friend ifstream& operator>>(ifstream& stin, Robot\* rb) | Перегрузка оператора считывания из файла |

Таблица 2.5 – Спецификация класса RoboDog

|  |  |
| --- | --- |
| int m\_PerRobot | Процент роботизированности |
| static int cntRoboDog | Количество объектов класса |
| RoboDog() | Конструктор без параметров |
| RoboDog(float mass, string sex, string color, int age, string race, string type, string name, int series, string model, string date, int cost) | Конструктор с параметрами |
| ~RoboDog() | Деструктор |
| void setPerRobo() | Установление процента роботизированности |
| virtual void print() const | Метод вывода информации |
| virtual void saveDR() | Метод сохранения информации в файл |
| virtual void downL() | Метод загрузки из файла |
| const int getPerRobo() const | Метод установление процента роботизированности |

Продолжение таблицы 2.5

|  |  |
| --- | --- |
| friend istream& operator>>(istream& in, RoboDog& rd) | Перегрузка оператора ввода |
| friend int getCntRoboDog() | Получить количество объектов класса |

Таблица 2.6 – Спецификация класса Shelter

|  |  |
| --- | --- |
| Поля и методы классов | Описание |
| struct list | Список животных и робопсов |
| int size | Размер списка |
| list\* head, \*tail | Голова, хвост списка |
| int moneyB | Количество денег у покупателя |
| vector<Animal\*> anim | Животные, которых взяли |
| vector<Robot\*> rb | Роботы, которых купили |
| friend class Control | Дружественный класс |
| friend class Lang | Дружественный класс |
| Shelter() | Конструктор без параметров |
| ~Shelter() | Конструктор с параметрами |
| const int getMoney()const | Получить деньги покупателя |
| void setMoney() | Установить деньги |
| void setMoney(int) | Установить деньги с параметром |
| void addAnim(Animal\*) | Метод добавления животного |
| void addAnim(RoboDog\*) | Метод добавления робопса |
| void removeAllShelter() | Очищение приюта |
| void removeAnim(vector<string>) | Удаление определенного животноо |
| void showAllShelter() | Показать приют |
| void changeM(int) | Метод изменения массы животного |
| void saveSh() | Сохранение информации в файл |

Таблица 2.7 – Спецификация класса Interface

|  |  |
| --- | --- |
| Поля и методы класса | Описание |
| fstream ru\_interface | Объект класса |
| fstream eng\_interface; | Объект класса |
| fstream configFile; | Объект класса |
| ofstream fileToSave; | Объект класса |
| ifstream fileToDownload; | Объект класса |
| bool ln | Текущий язык |
| vector<string> waysFile | Пути файлов |
| friend class Dog | Дружественный класс |
| friend class Cat | Дружественный класс |
| friend class Robot | Дружественный класс |

Продолжение таблицы 2.7

|  |  |
| --- | --- |
| friend class RoboDog | Дружественный класс |
| friend class Shelter | Дружественный класс |
| Interface() | Конструктор без параметров |
| void beginningOfWork(); | Метод загрузки интерфейса |
| void titlePage() | Метод вывода главного меню |
| void switchLanguage() | Метод изменения языка |
| void aboutMe() | Метод вывода информации об авторе |
| void operatorsManual() | Методы вывода информации о программе |
| void exitFromTheProgram() | Метод выхода из программы |
| void operatorsManual() | Метод вывода помощи |
| void outputTextFromFile(int num, int k) | Метод вывода текста с файла |
| void fileSelectionToSave() | Метод выбора пути файла для сохранения |
| void fileSelectionForDownload() | Метод выбора пути файла для загрузки |
| void loadingFromFile() | Метод загрузки |
| void saveToFile() | Метод сохранения |
| friend ifstream& operator>>(ifstream& stin, Dog& dog) | Перегрузка оператора считывания из файла |

Таблица 2.8 – Спецификация класса Control

|  |  |
| --- | --- |
| Поля и методы класса | Описание |
| enum kC | Перечисления кодов клавиш |
| Shelter sh | Объект класса |
| vector<Robot\*> shRb | Вектор объектов класса |
| friend class Lang | Дружественный класс |
| void giveTheAnimalToTheShelter() | Метод передачи животного в приют |
| void pickUpAnimalsFromTheShelter() | Метод получения животного с приюта |
| void seeTheWholeShelter() | Показать весь приют |
| void changeMassAnimal() | Изменить массу животного |
| void displayInformationAboutTheShelter() | Вывести информацию о приюте |
| void viewModelAndPrice() | Показать модель и цену |
| void findByModel() | Найти модель |
| void buyRobot() | Купить робота |
| void mainMenuOfTheProgram() | Главное меню программы |
| void menuForShelter() | Меню для приюта |
| void menuForShopRobots() | Меню для магазина роботов |

Продолжение таблицы 2.8

|  |  |
| --- | --- |
| void menuForBuyer() | Меню для покупателя |
| void continueFuncSh() | Метод продолжения работы с программой |
| void continueFuncRSh() | Метод продолжения работы с программой |

## 2.2.2 Отношения между классами

Диаграмма классов демонстрирует классы системы, их атрибуты, методы и взаимосвязи между ними.

Диаграмма классов(см. рис. 2.2) для проектируемой объектно-ориентированной системы. На диаграмме показано взаимодействие классов (одиночное и множественное наследование, композиция). Каждый класс имеет get-методы для получения доступа к полям и аction-методы, реализующие поведение класса.

## 2.2.3 Структура объектов

Диаграмма объектов - диаграмма, на которой изображаются объекты и их отношения в данный момент времени. Диаграмму объектов можно считать особым случаем диаграммы классов, где могут отображаться как экземпляры, так и сами классы.

Диаграмма объектов (см. рис. 2.3), которая отображает характерные значения атрибутов объектов заданной объектно-ориентированной системы.

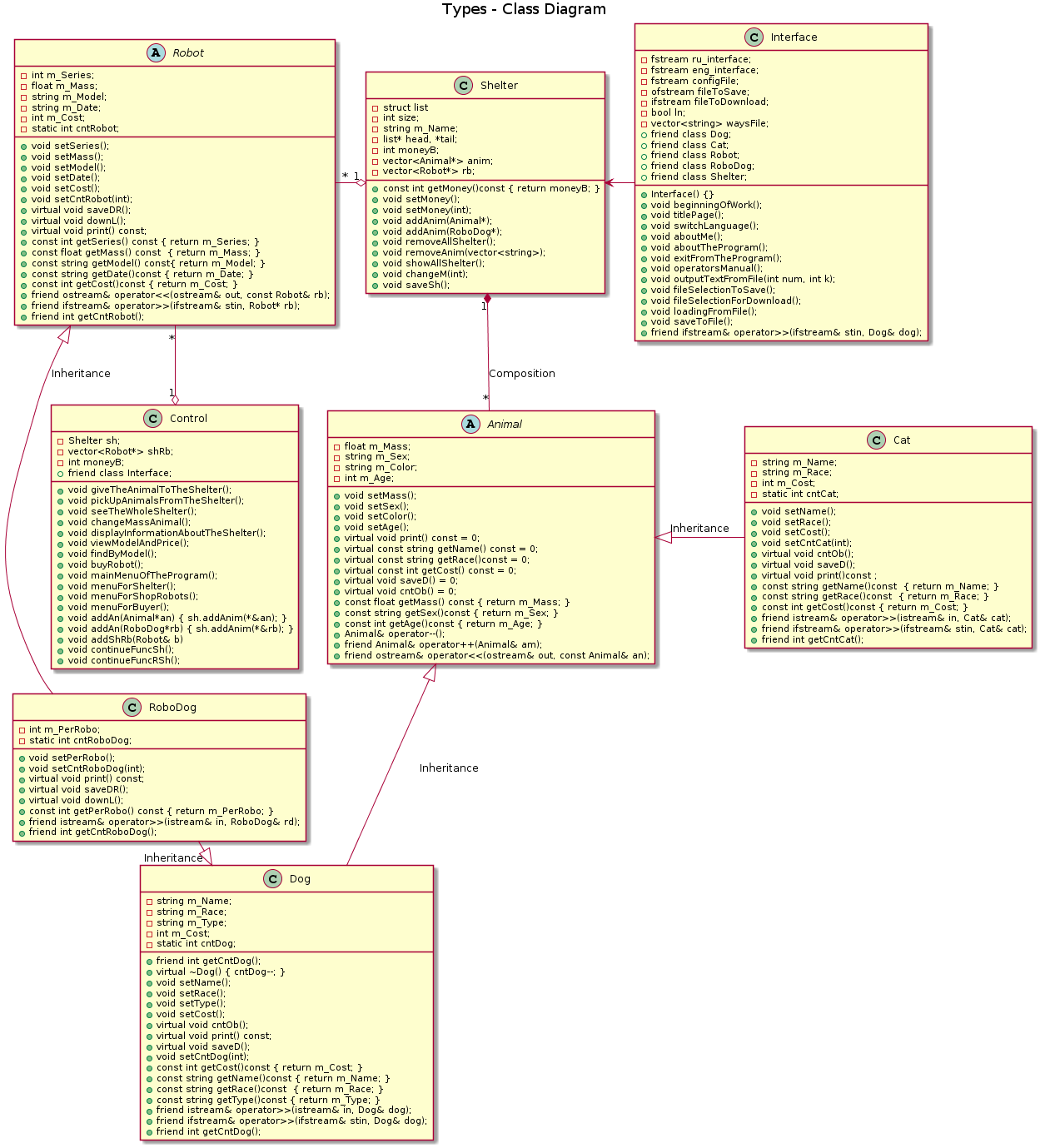


Рисунок 2.2 – Диаграмма классов проектируемой системы

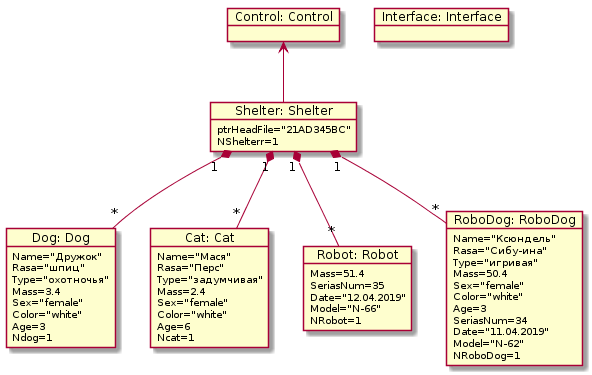


Рисунок 2.3 – Диаграмма объектов проектируемой системы

## 2.2.4 Диаграмма состояний

Диаграммы состояний (см. рис.2.4) являются хорошо известным средством описа­ния поведения систем. Они определяют все возможные состояния, в кото­рых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий.

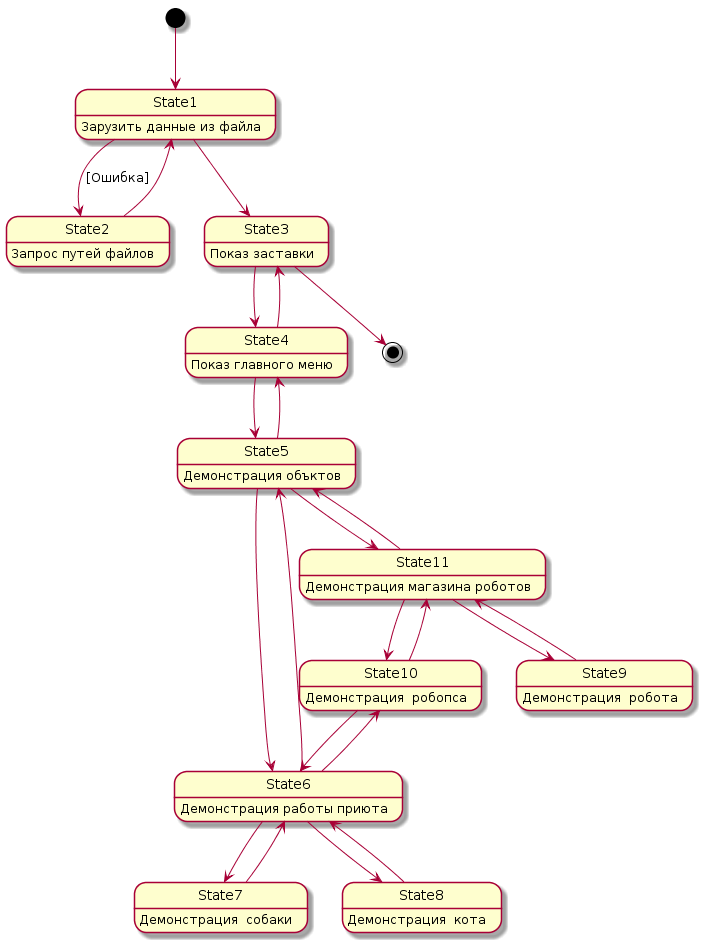


Рисунок 2.4 – Диаграмма состояний проектируемой системы

## 2.2.5 Диаграмма последовательности

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. На этих диаграммах изображаются только те объекты, которые непосредственно участвуют во взаимодействии т.к. ключевым моментом является именно динамика взаимодействия объектов во времени и не используются возможные статические ассоциации с другими объектами.

Диаграмма последовательности (см. рис 2.5), специфицирующая ветвь сценария работы с приютом.

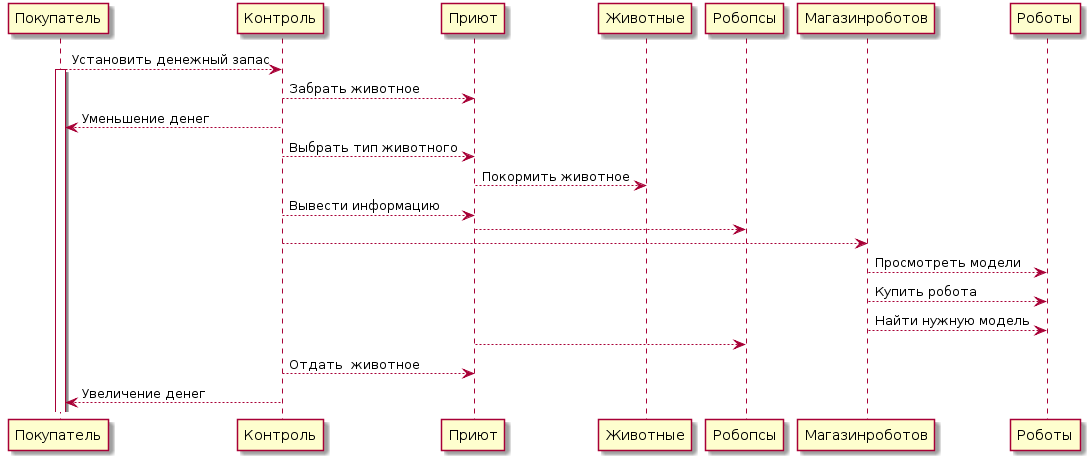


Рисунок 2.5 – Диаграмма последовательности проектируемой системы

# 3 Рабочее проектирование

Рабочее и техническое проектирование осуществляется на заключительной стадии процесса проектирования. Оно предусматривают детализацию результатов предыдущих стадий, определение структуры программы[1], 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. Структура данных приложения и методы работы с ними

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 (см. рис. 3.1).

Преимущество раздельной компиляции состоит в том, что при изменении 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[2].

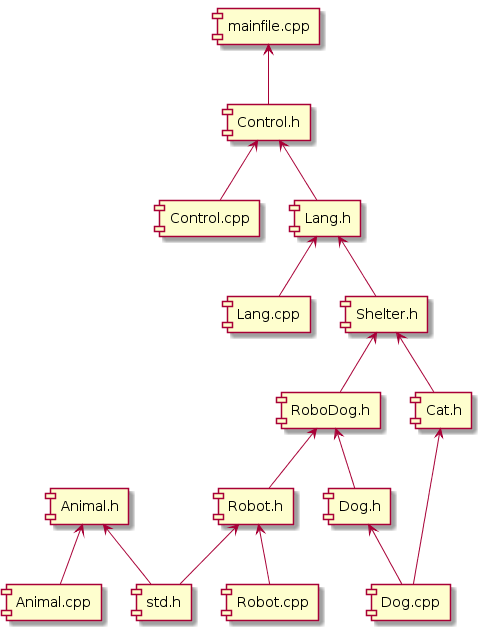


Рисунок 3.1 – Диаграмма компонентов системы

Программная система VAC.exe реализует мультиязычный интерфейс, обеспечивает минимизацию количества вводимых пользователем данных, а также предоставляет ему подробную справочную информацию в режиме помощи на первом слайде программы. Интерфейс был реализован с помощью четырех текстовых файлов.

Для реализации приюта выбрана такая структура данных: двусвязный список (см. рис. 3.2). Выбор обусловлен тем, что необходимо предоставления механизма для хранения и доступа к произвольному количеству данных.

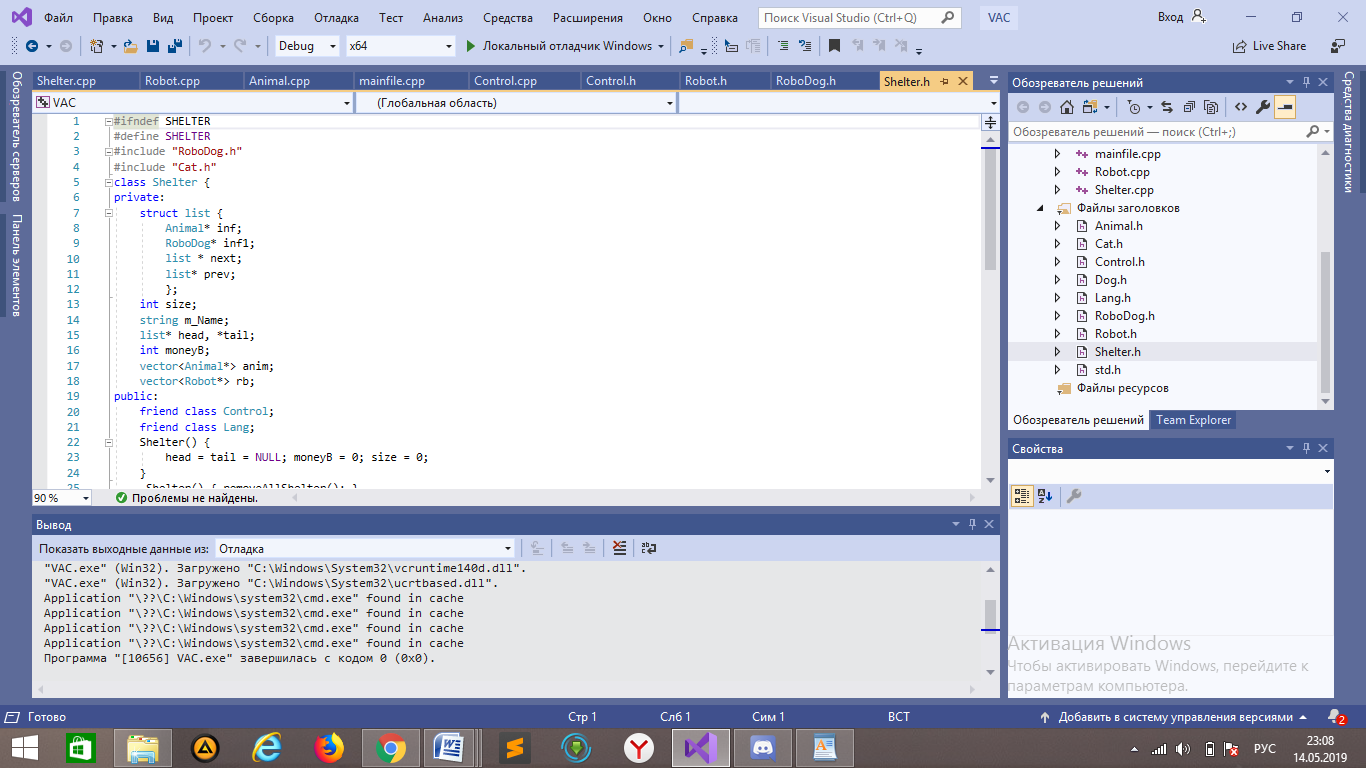


Рисунок 3.2 –Реализация двусвязного списка

Класс Control, который хранит в себе вектор на все объекты класса Robot. Данное решение позволяет программисту удобнее взаимодействовать с добавленными роботами в магазин. Например, с помощью данного подхода была реализована функция findByModel() (рис. 3.3)

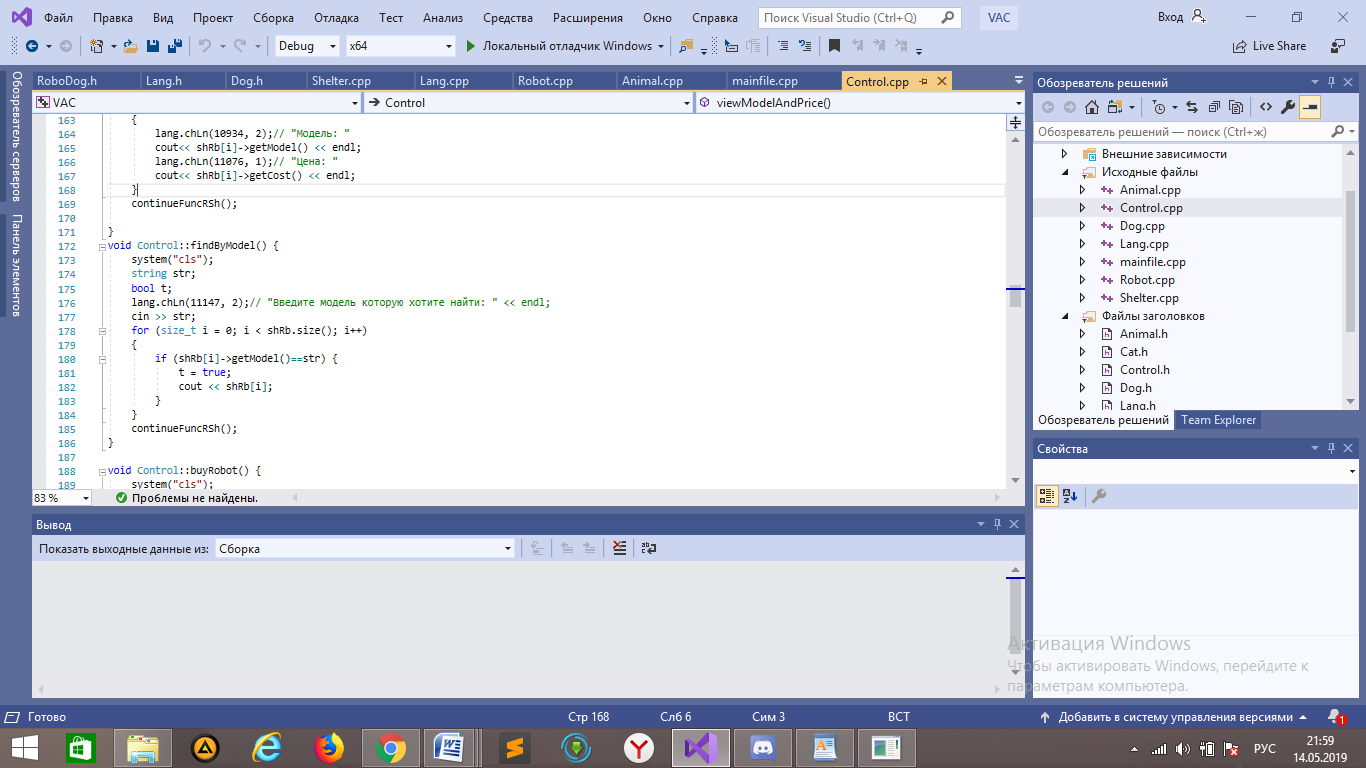


Рисунок 3.3–Функция поиска определенной модели робота

Функция позволяет найти определенную модель робота (рис. 3.4)

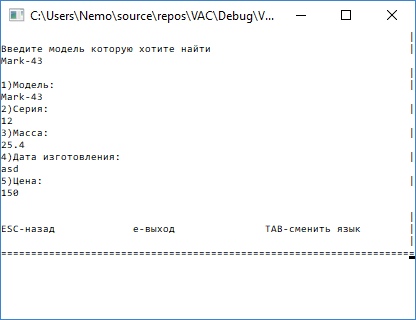


Рисунок 3.4 – Выбор определенной модели

## 3.2 Разработка классов

### 3.2.1 Реализация инкапсуляции

Инкапсуляция(см. рис. 3.5) – это механизм, который объединяет данные и код, манипулирующий этими данными, а также защищает и то, и другое от внешнего вмешательства или неправильного использования.

Для указания доступности свойства или метода класса в языке программирования С++ используются модификаторы доступа: public, private и protected. Поля и методы класса, объявлены с использованием модификатора public доступны как методам класса, так и любым функциям программы. Protected методы и поля доступны в классе и всем дочерним классам. Private методы и поля доступны только внутри класса, в котором они объявлены.

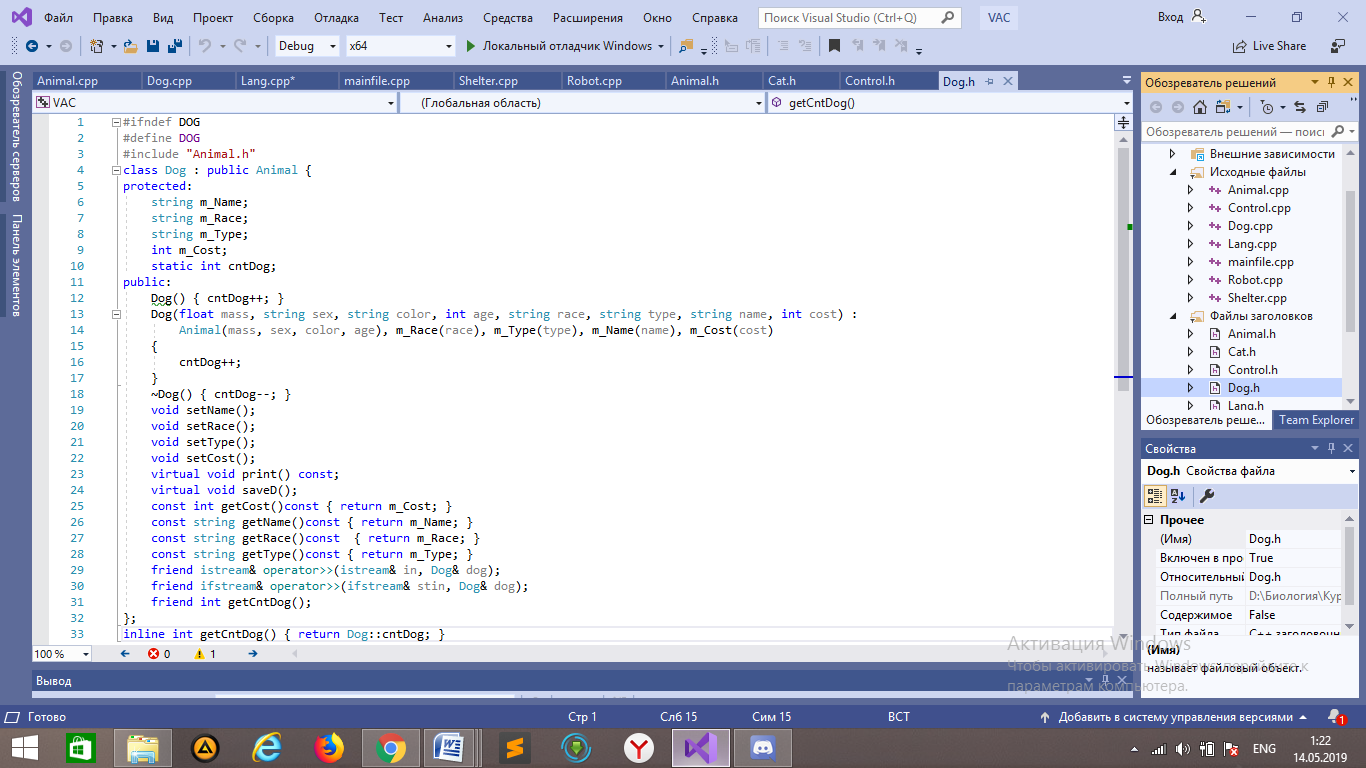
Все свойства объектов в курсовой работе объявлены, используя модификаторы доступа private или public. Таким образом, доступ к ним невозможен вне данного класса. 

Рисунок 3.5 –Реализация инкапсуляции для класса Dog и одиночного наследования

Здесь доступ к полям класса возможен только через публичные методы, контролирующие корректность присваиваемых полям значений. Это позволяет обеспечивать целостность хранимых в классе данных.

### 3.2.2 Реализация наследования

Наследование [2] – 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 рамках курсового проекта было реализовано одиночное и множественное наследование. Класс Dog наследует и расширяет данные и поведение класса Animal (см.рис. 3.5).

Множественное наследование позволяет классу RoboDog хранить данные, реализовывать и расширять поведение базовых классов Dog и Robot (см. рис. 3.6).

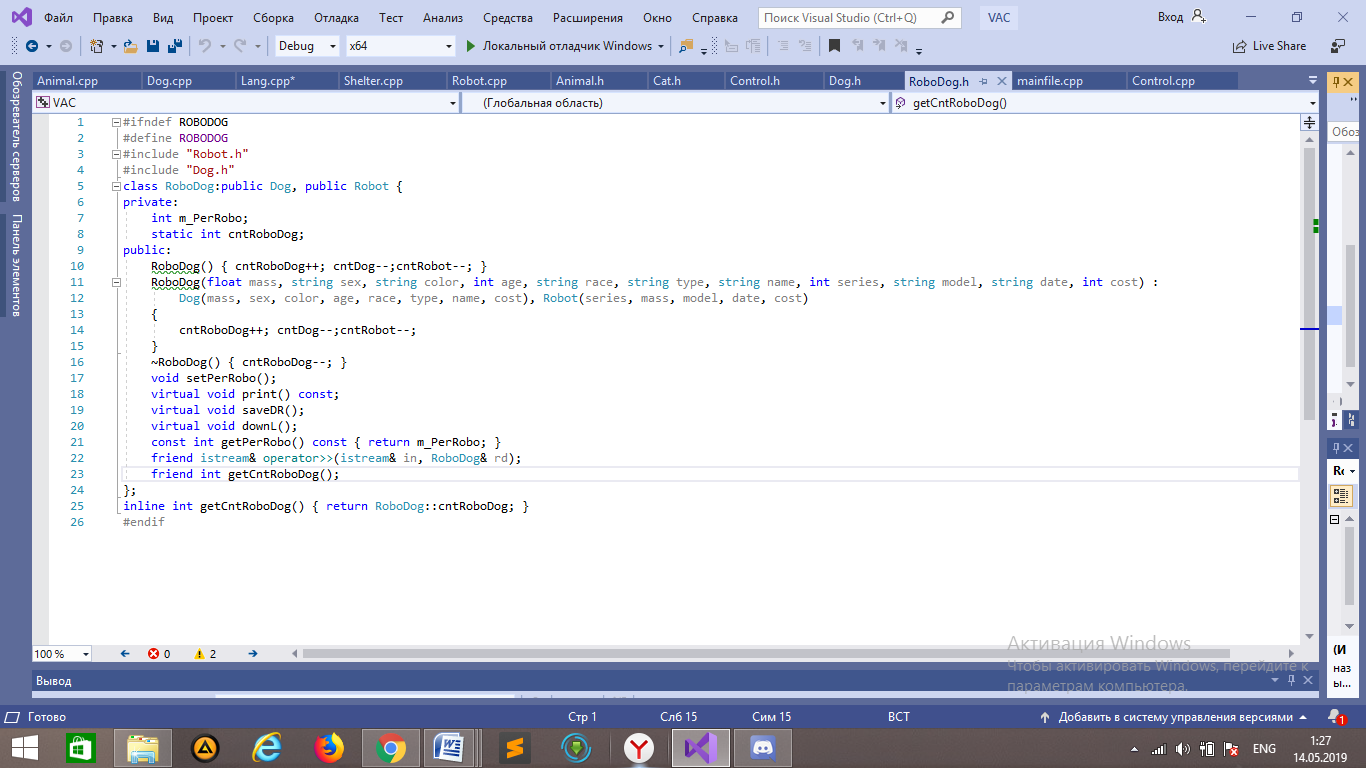


Рисунок 3.6 – Реализация множественного наследования

### 3.2.3 Реализация полиморфизма

Полиморфизм — возможность объектов с одинаковой спецификацией иметь различную реализацию. Перегрузка операторов [2] – 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.

В рамках курсового проекта была выполнена перегрузка унарного оператора префиксного декремента и инкремента для класса Animal, , а также перегрузка оператора вывода << для классов Animal, Robot. Также оператор ввода >> для классов Dog, Cat, Robot, RoboDog.

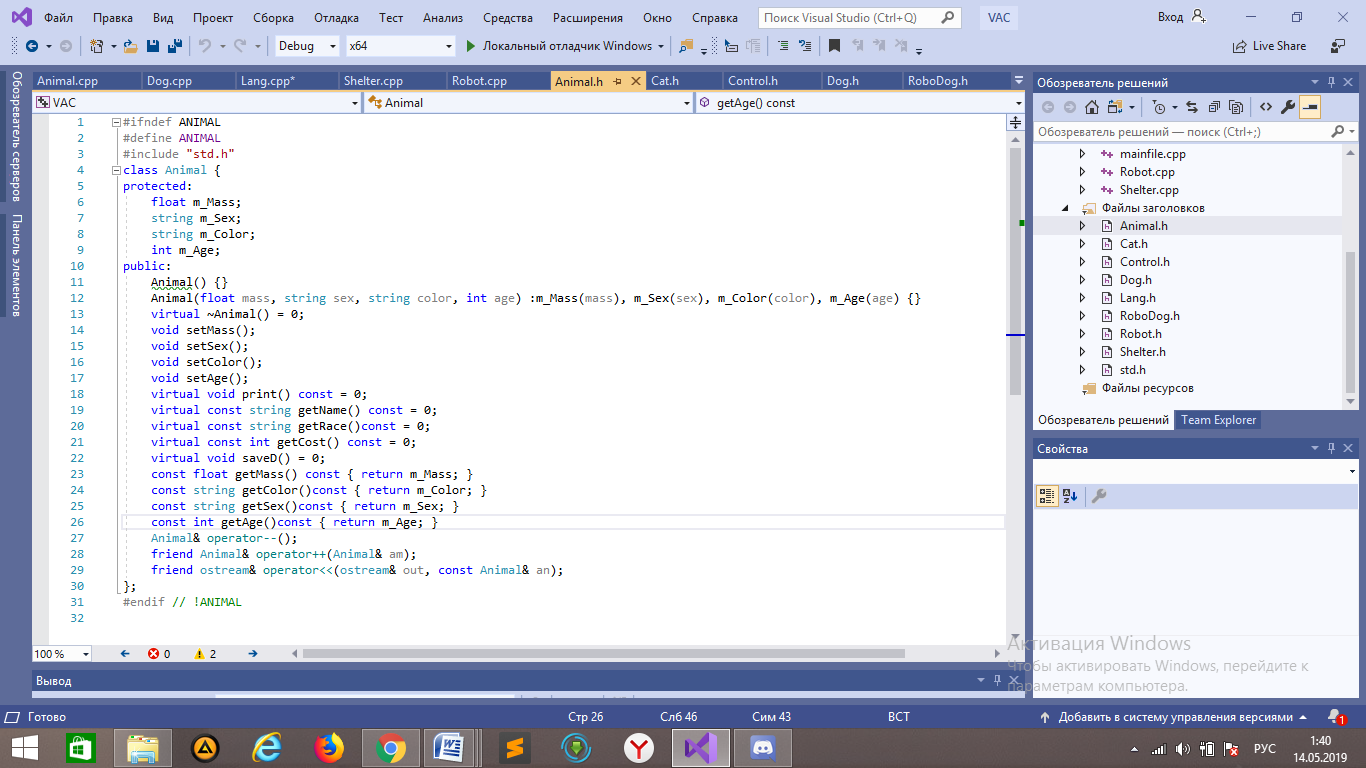
Префиксный оператор -- и ++ перегружен внутри класса Animal и определяется в виде метода c параметром и в виде дружественной функции соответственно. (см. рис. 3.7). Новое действие, выполняемое оператором --–уменьшение массы на 0,1 кг. Новое действие, выполняемое оператором ++ –увеличение массы на 0,1 кг.

Рисунок 3.7 – Перегрузка операторов ++ и -- абстрактного класса Animal

Данные операторы перегружены как внутри та и вне класса (см.рис. 3.8)

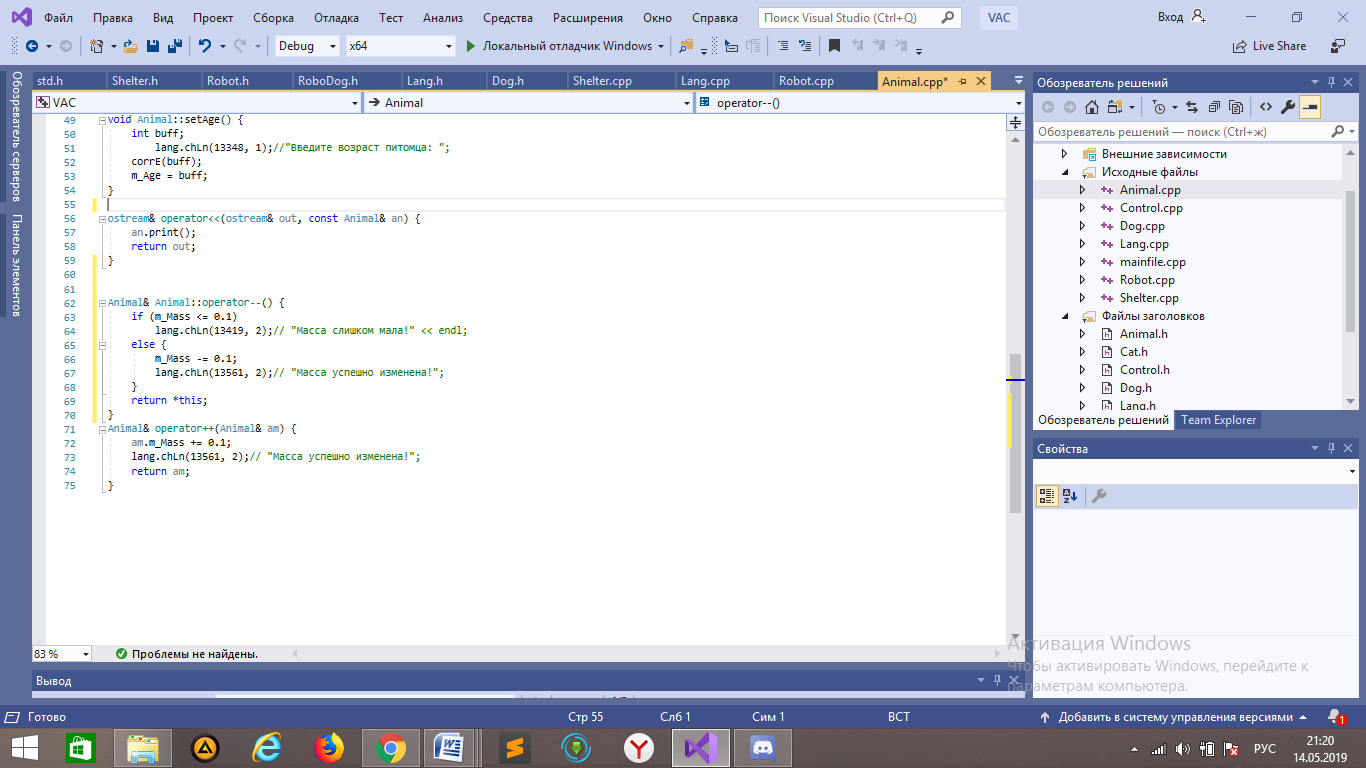


Рисунок 3.8 – Перегрузка операторов ++ и -- внутри и вне абстрактного класса Animal

Класс Animal абстрактный поскольку содержит в себе чисто виртуальный деструктор и чисто виртуальные методы, которые впоследствии переопределяются в дочерних классах.

Операторы вывода << и ввода >> перегружены вне класса в виде дружественной функции (см. рис. 3.9).

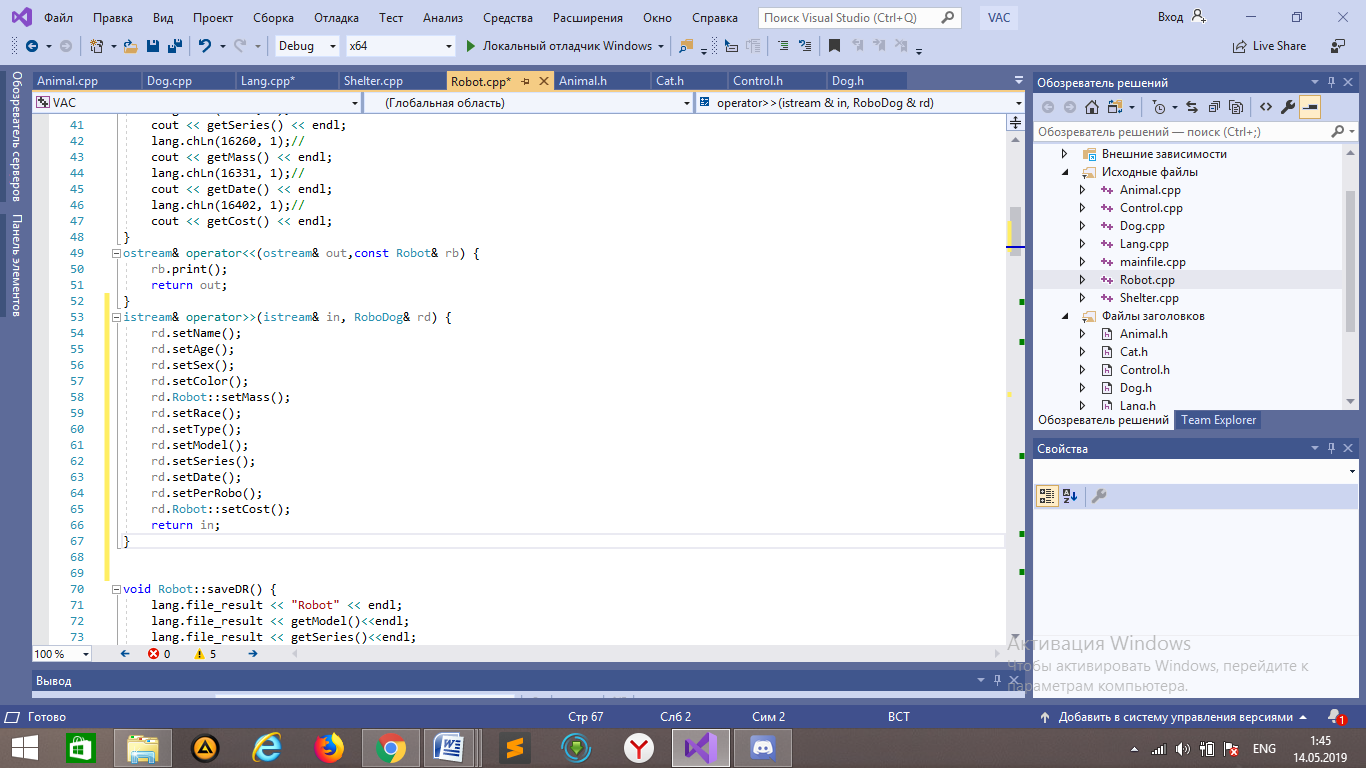


Рисунок 3.9 – Перегрузка оператора << и >> для класса Robot

## 3.3 Обоснование выбора алгоритмов

Основной алгоритм программы, реализующий диалог с пользователем посредством интерфейса-меню, приведен в виде схемы навигации между экранами (см. рис. 3.10), переходы между которыми осуществляются по нажатию определенной клавиши.

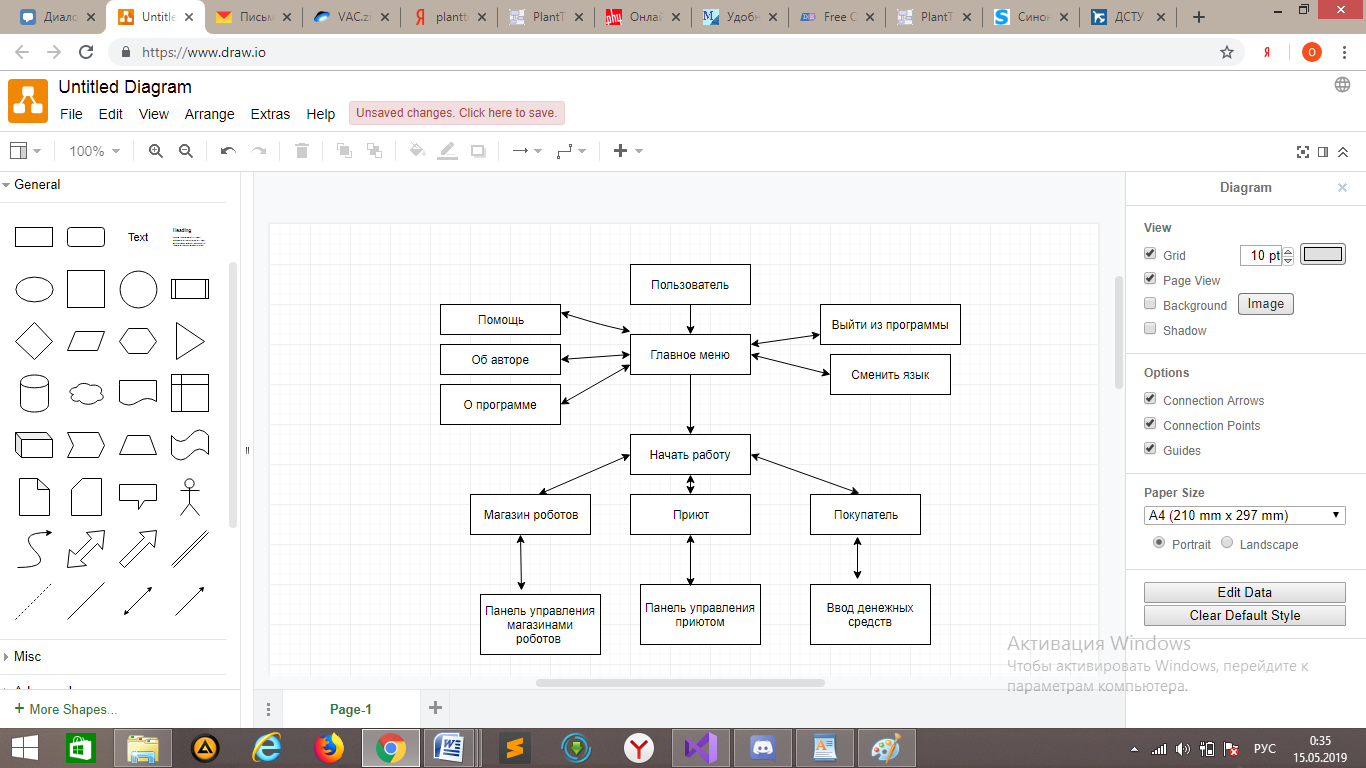


Рисунок 3.10 – Навигации между экранами системы

На заставке программы пользователь может выбрать пункт начала работы и попасть в главное меню. Также у пользователя есть подсказки в виде помощи. Пользователь может произвести смену языка. Этот алгоритм удобный, так как позволяет структурировать все необходимые функции для пользователя на одном слайде программы.  
 В меню присутствует возможность сохранить текущую сессию или загрузить другую. Выбрав один из пунктов меню, пользователю будут предложены различные действия, которые он может совершить над объектами.

На рисунке (см. рис. 3.11) изображена блок-схема функции вывода сообщений из файла на экран, в функцию передаются два параметра: первый-число байт файла, откуда производить считывание, а второй - количество строк для считывания, при вызове функции первым делом производится проверка на язык, если она равна истине, то считываются надписи из русского файла, иначе из английского. Считывание происходит с помощью цикла, оно работает пока не прочитает заданное количество строк, строки записываются в переменную типа стринг и после она выводится.

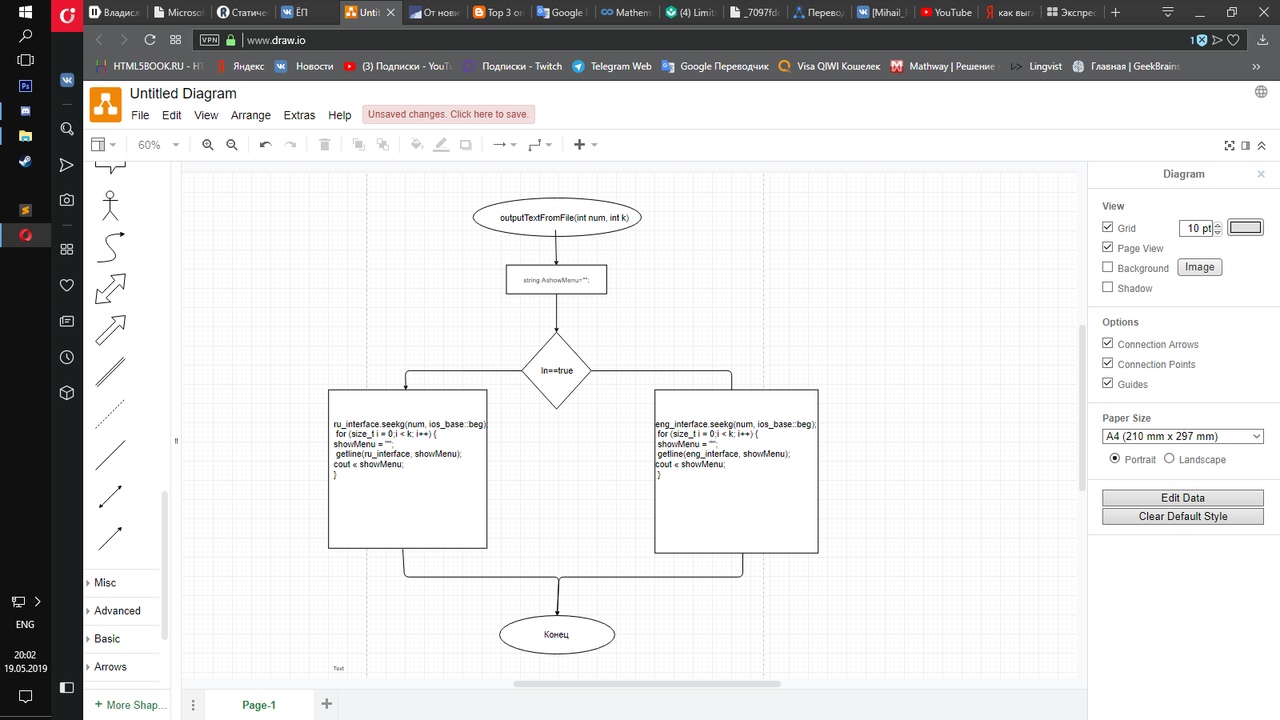


Рисунок 3.11 –Алгоритм вывода надписей на двух языках.

### 3.3.1 Работа с файлами

Файл configuration.txt находится в корневой папки программы и содержит в себе пути к russiun.txt(текстовый файл с русским интерфейсом), к english.txt (текстовый файл с английским интерфейсом) и save.txt(текстовый файл, в который сохраняется состояние объектов).

При запуске система просит пользователя ввести путь к файлу конфигурации. При указании неверного пути программа сообщает пользователю об ошибке (см.рис. 3.12). Также выводит сообщение об ошибке, если файл пуст.

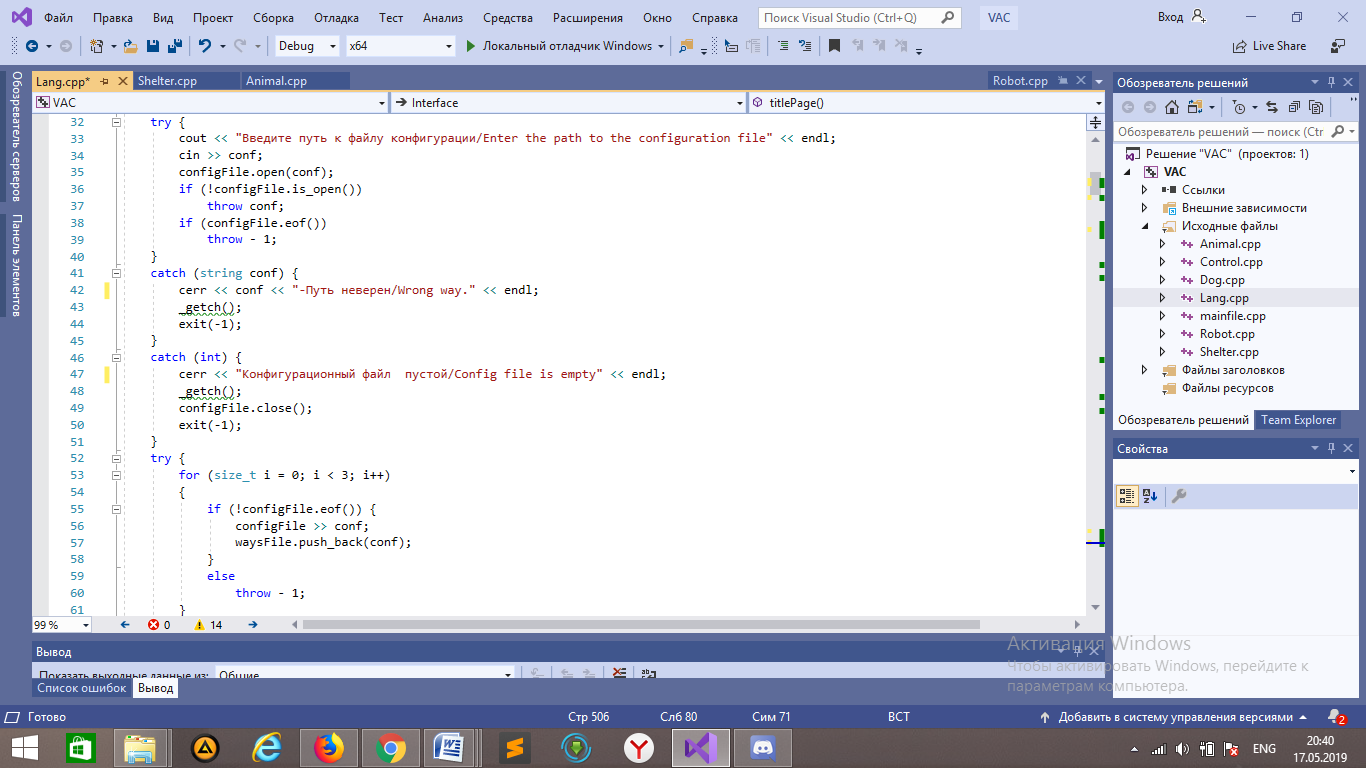


Рисунок 3.12 –Обработка исключительных ситуаций при работе с файлом конфигурации

Также присутствует обработка таких ошибок, как отсутствие файлов интерфейсов и файла для записей данных.(см.рис 3.13)

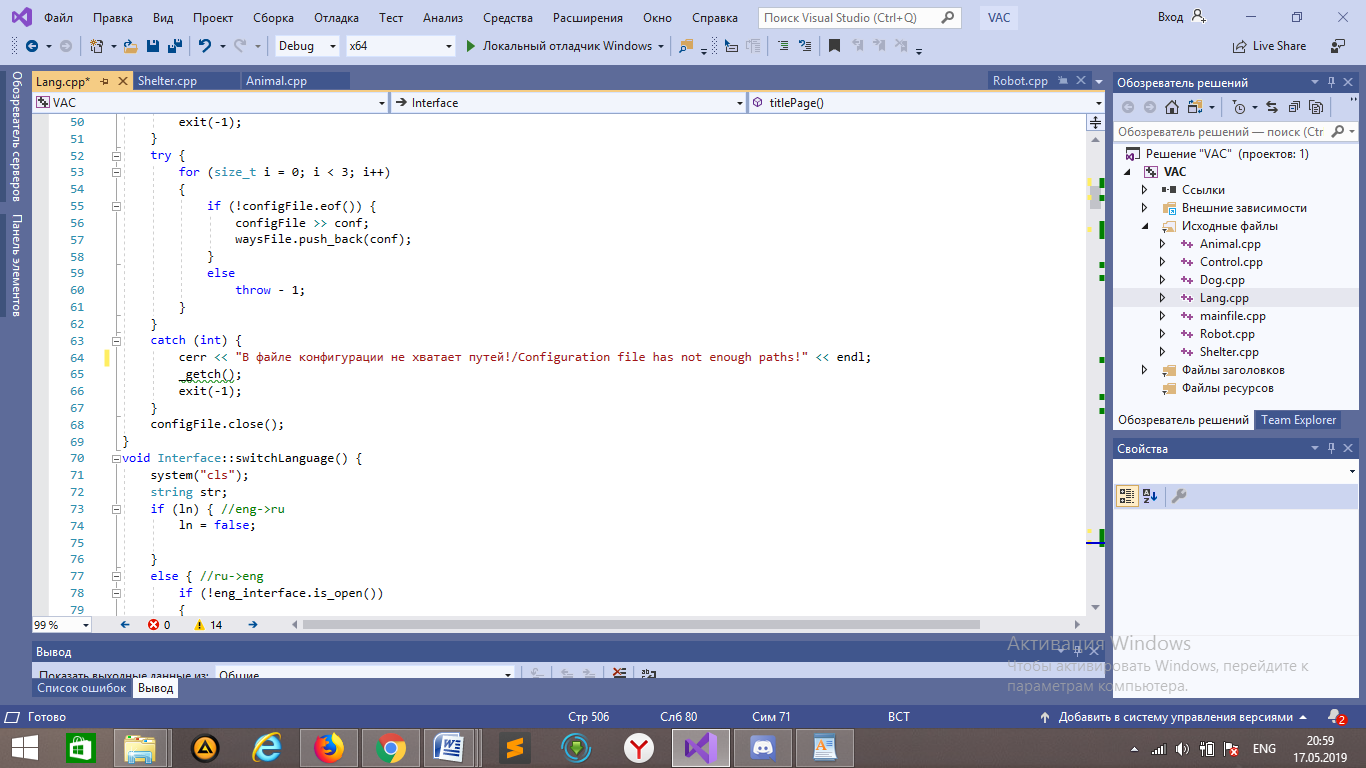


Рисунок 3.13–Обработка исключительных ситуаций при работе с файлами

Вдобавок в коде программы присутствует обработка исключительных ситуаций при работе с файлами интерфейса.(см.рис. 3.14)

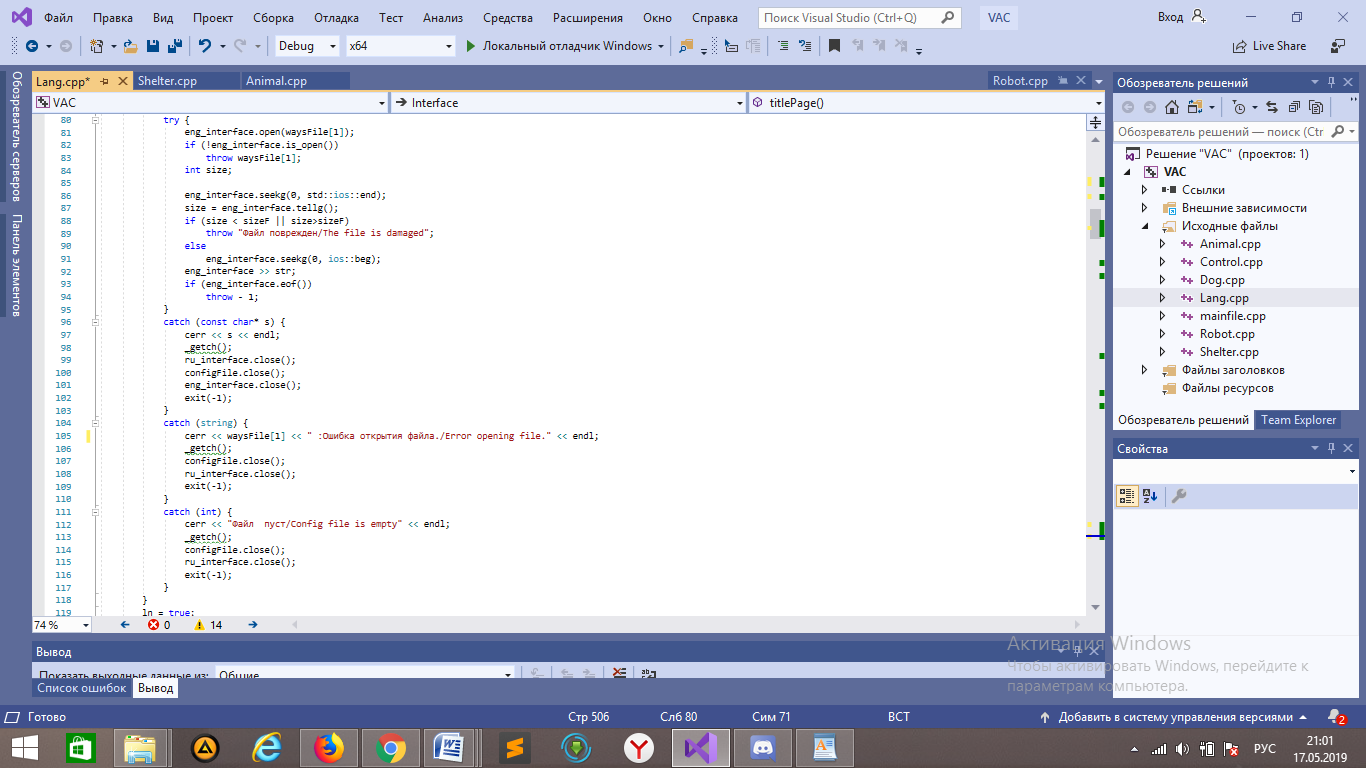


Рисунок 3.14 –Обработка исключительных ситуаций при работе с файлами интерфейса

Если файл поврежден, т.е. количество байт не соответствует заданному в константе sizeF, то будет выведено сообщение об ошибке.

Надписи из файлов читаются напрямую.(см.рис.3.15)



Рисунок 3.15–Чтения надписей из файла.

Пользователь может записать текущее состояние объектов программы в файл, предоставленный в файле конфигурации, либо в другой файл на компьютере, предварительно указав к нему путь. Опять же как и с другими файлами предусмотрена обработка непредвиденных ситуаций. (см.рис. 3.16)

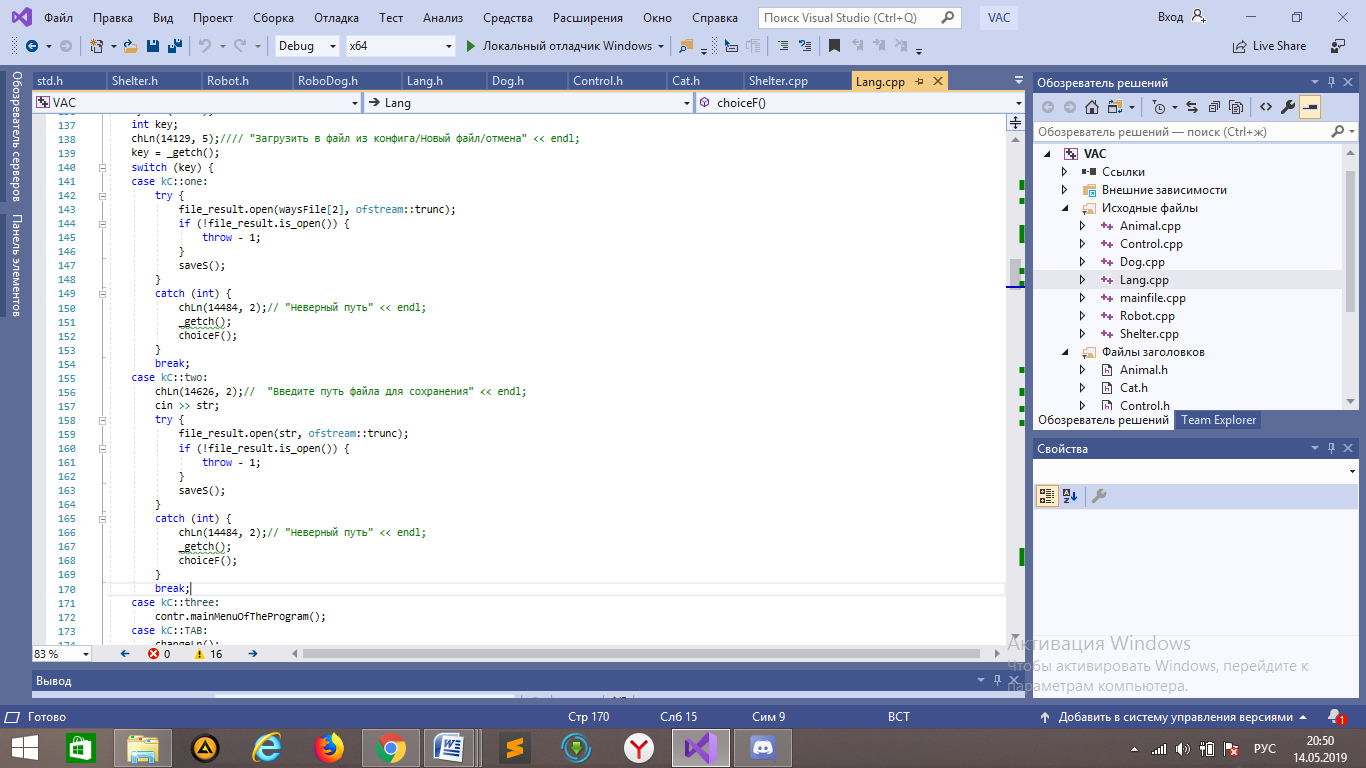


Рисунок 3.16 –Работа с файлами загрузки.

Ниже приведены функции записи состояния объектов в файл.(см.рис. 3.17, 3.18, 3.19)

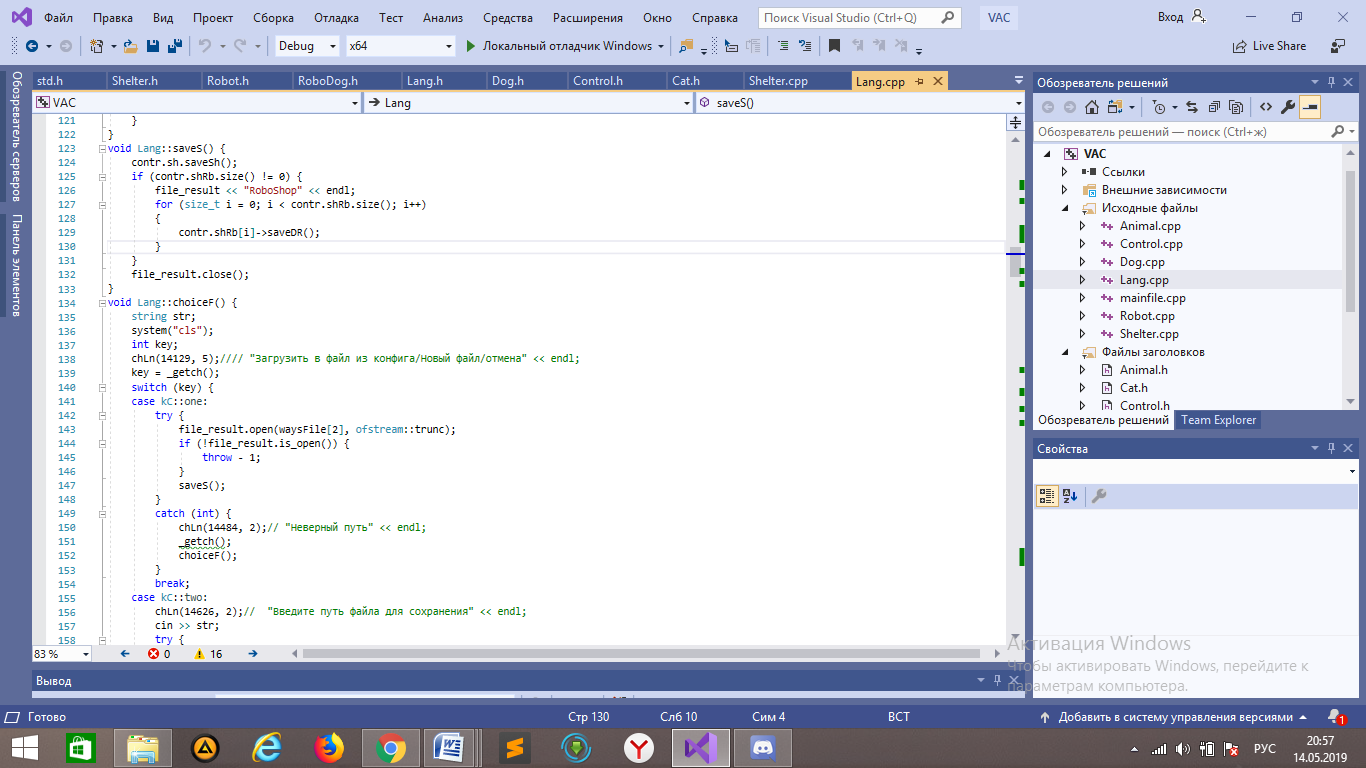


Рисунок 3.17–Основная функция записи в файл.

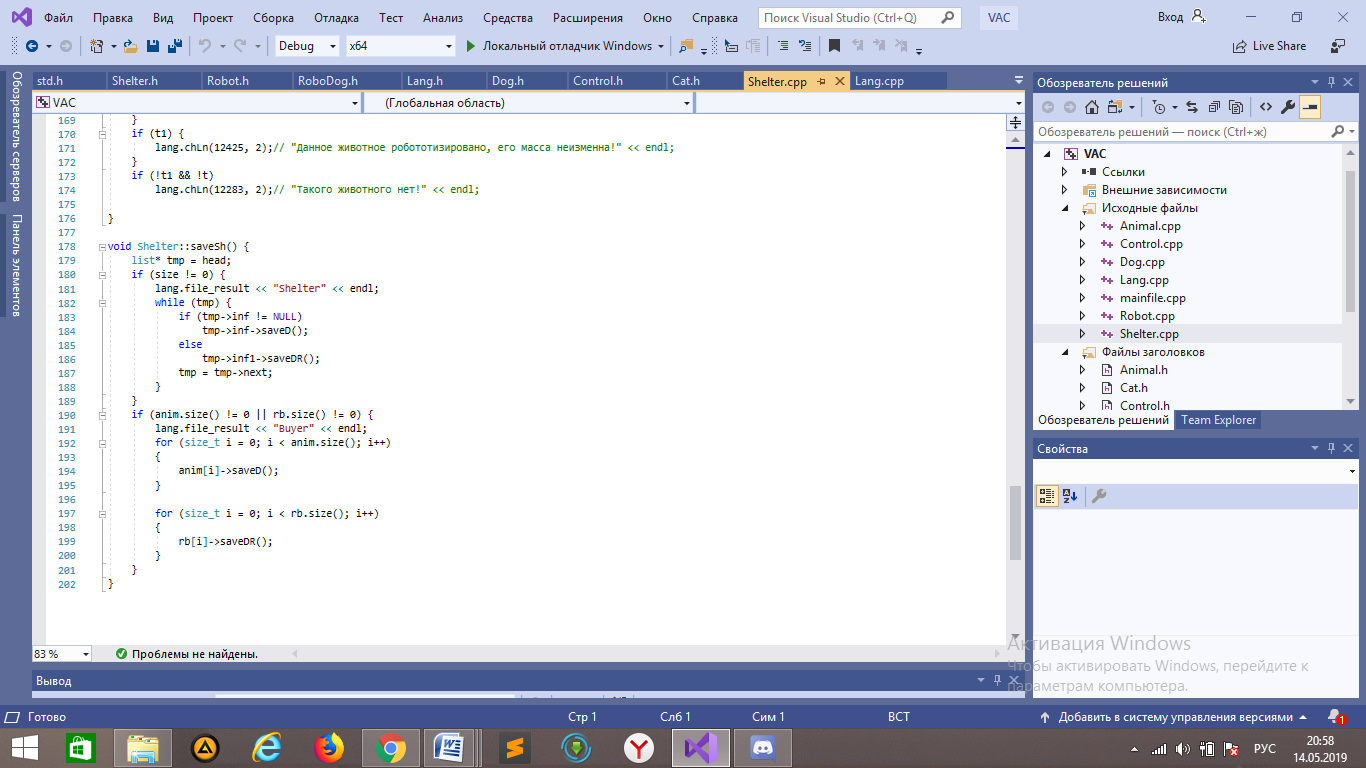


Рисунок 3.18–Функция записи объектов приюта и сделанных покупок.

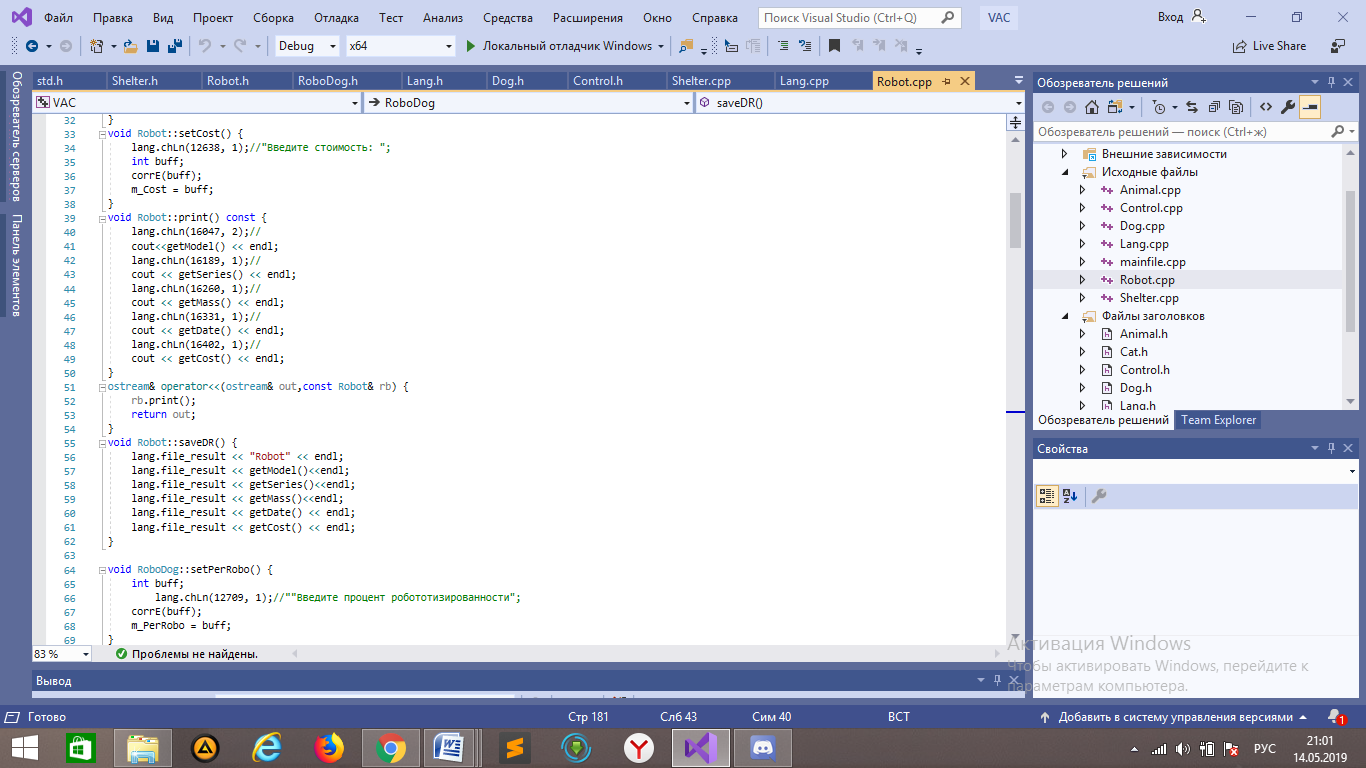


Рисунок 3.19–Функция записи объектов класса Robot и сделанных покупок.

По аналогии с сохранением объектов класса в файл существует возможность загрузки объектов из файла в программу. (см.рис.3.20)

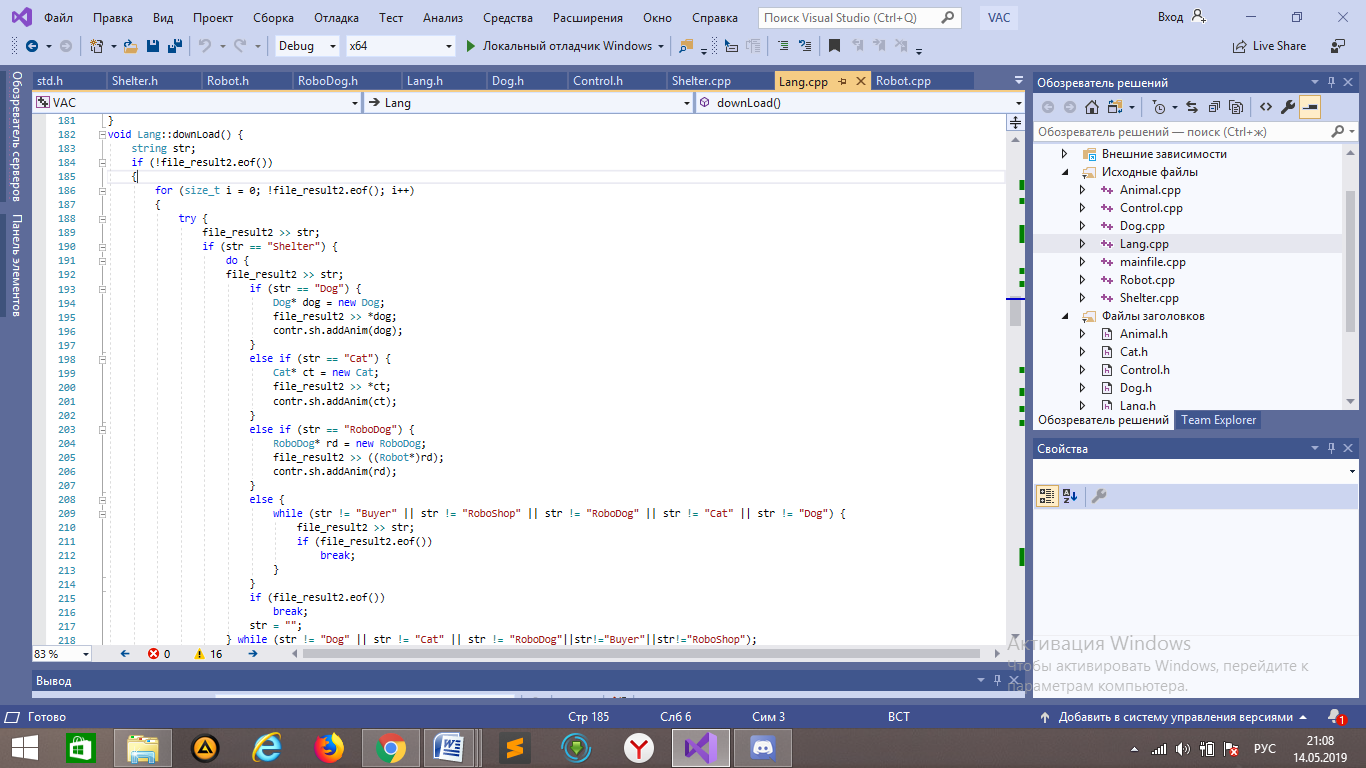


Рисунок 3.20–Фрагмент кода функции загрузки из файла.

## 3.3.2 Подсчет количества экземпляров класса.

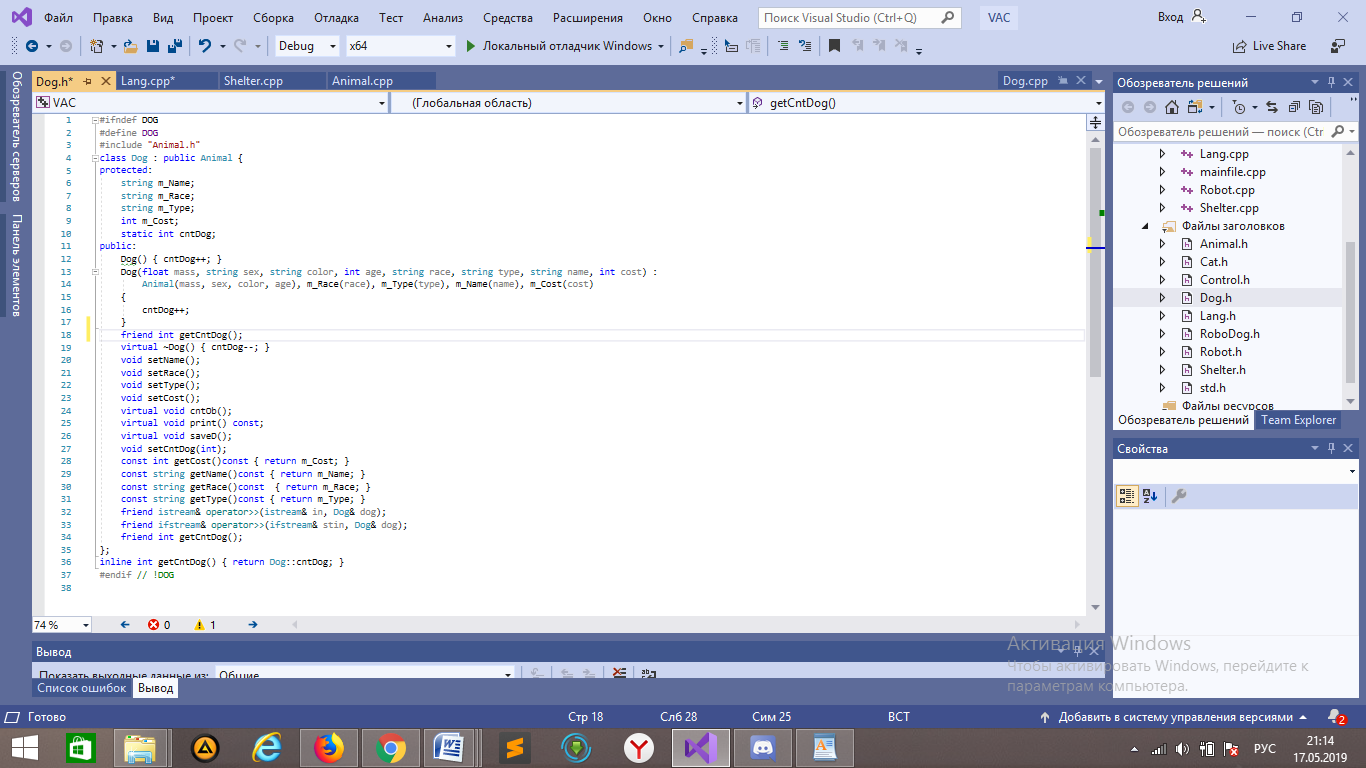
Объекты классов подсчитываются за счет статической переменной, объявленной в классах, в которых создаются объекты. При удалении объекта в деструкторе эта переменная уменьшается на 1, а при создании - увеличивается на 1(см. рис. 3.21). Чтобы получить эту переменную пользуемся get-методом. Подсчет количества объектов ведется в классах Dog, Cat, Robodog и Robot.

Рисунок 3.21–Пример реализации подсчета количества объектов.

### 3.3.3 Организация информационной поддержки

В программе предусмотрена система подсказок для пользователя в виде «О программе» (см. рис. 3.22), «Помощь»(см.рис. 3.23), «Об авторе»(см. рис. 3.24) Файлы справки необходимы для обеспечения режима помощи системы. Текст документов считывается при обращении к соответствующему пункту меню, что позволяет более рационально использовать оперативную память.

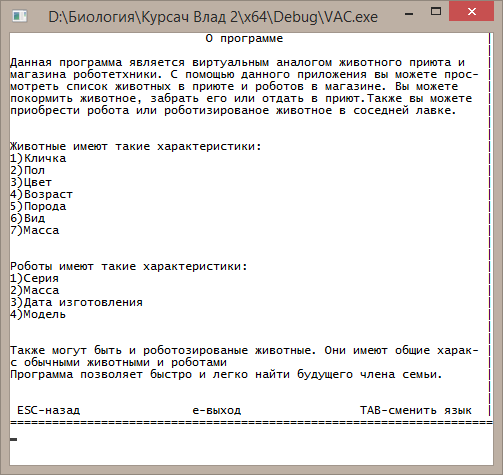


Рисунок 3.22 – О программе

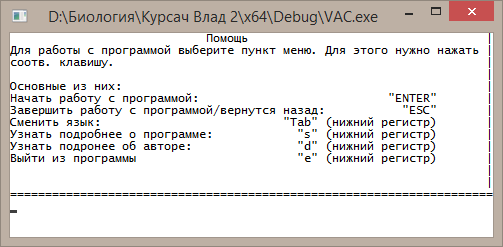


Рисунок 3.23 ­– Помощь на 1м слайде

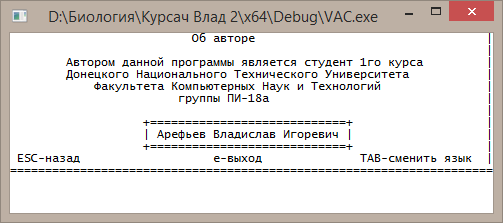


Рисунок 3.24­– Об авторе

## 3.4 Тестирование работоспособности системы классов

Система VAC.exe обрабатывает ошибочные ситуации связанные как с ошибками пользователя, так и с ошибками, вызванными отсутствия необходимых для функционирования программы файлов.

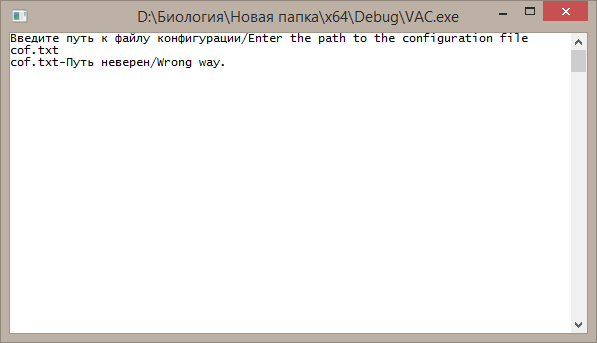
При отсутствии файла конфигурации, пользователь получает соответствующее сообщение(см.рис. 3.25).

Рисунок 3.25– Сообщение об отсутствии файла конфигурации

При отсутствии файла интерфейса пользователь получает следующее сообщение(см.рис. 3.26)

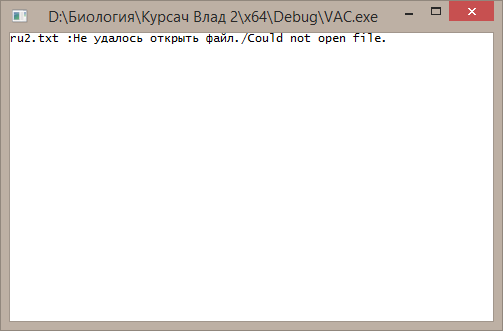


Рисунок 3.26– Сообщение об отсутствии файла интерфейса.

При изменении файла интерфейса выводится такое сообщение(см.рис. 3.27).

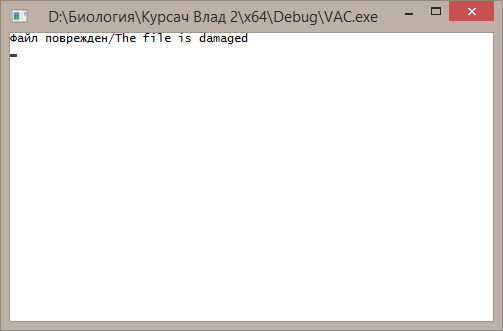


Рисунок 3.27– Сообщение о повреждении файла.

Для каждой исключительной ситуации при работе с файлами в программе существует соответствующее сообщение об ошибки.

Все данные, вводимые пользователем, проверяются на корректность(см.рис. 3.28).

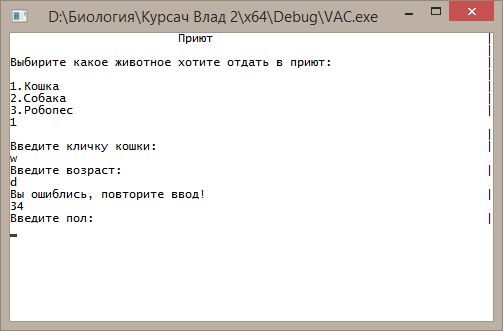


Рисунок 3.28– Сообщение о некорректности ввода данных.

Например, при попытке пользователя ввести в поле возраста отрицательное число, буквы программа сообщит об ошибке.

На каждом кадре интерфейса происходит смена языка(см.рис. 3.29,3.30).

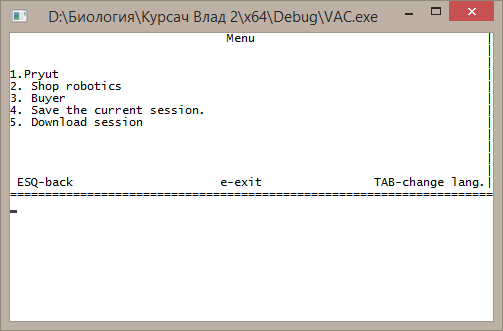


Рисунок 3.29 – Меню на английском языке.

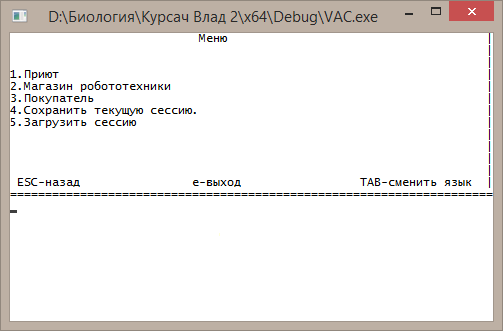


Рисунок 3.30 –Меню на русском языке.

При закрытии программы у пользователя спрашивается, хочет ли он сохранить текущую сессию. (см.рис. 3.31)

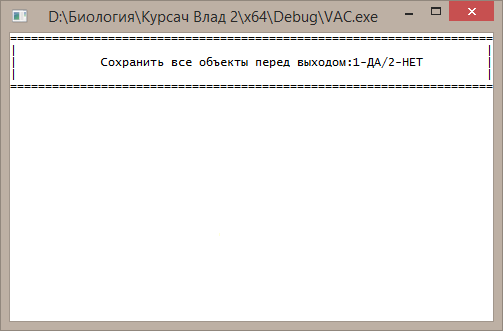


Рисунок 3.31 –Запрос на сохранение.

# Выводы

Разработанная программа VAC.exe является объектно-ориентированной системой для моделирования предметной области «Приют». 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 проектной, технической и пользовательской документации.

Наиболее перспективными направлениями дальнейшего развития программного продукта VAC.exe – конкретизация и расширение функциональности реализованной имитационной модели, добавление новых животных в приют, оптимизация программы.

Перечень ссылок

1. Д. Р. Мюссер, Ж. Дж. Дердж, А. Сейни - C++ и STL. Справочное руководство, 2-е издание // «Вильямс», 2010. – 432c.
2. Разработка и стандартизация программных средств и информационных технологий. Конспект лекций [Электронный ресурс]. – Режим доступа: http://ooad.asf.ru/students/lectures\_risp/index.aspx
3. Коплиен Дж. - Мультипарадигменное проектирование для с++// «Питер», 2014. – 235с.
4. Романов Е.Л. Си++. От дилетанта до профессионала++ // «Питер», 2014. – 600с.
5. Либерти, Д. Освой самостоятельно С++ за 21 день, 5-е издание/ Джесс Либерти, Бредли Джонс//: «Вильямс», 2007.–784

# Приложение А

Техническое задание

# Приложение Б

Описание программы

Б.1 Общие сведения

Программа VAC.exe предназначена для моделирования процесса работы с виртуальным приютом.

Б.2 Функциональное назначение

Программа предназначена для использования в высших учебных заведениях с целью демонстрации главных концепций объектно-ориентированного подхода при разработке ПО средствами языка С++.

Б.3 Используемые технические средства

          Для запуска программы требуется персональный компьютер с частотой процессора не менее 1.2 ГГц и объемом оперативной памяти не менее 128 Мб.

Б.4 Вызов и загрузка

Вызов и загрузка программы осуществляется путем запуска исполняемого файла VAC.exe.

Б.5 Входные данные

В качестве исходных данных программа использует данные вводимые пользователем с клавиатуры и считанные с файла.

Б.6 Выходные данные

Выходные данные выводятся на экран и файлы которые указываются пользователем или прописаны в коде программы.

# Приложение В

Руководство программиста

Файл конфигурации программы configuration.txt состоит из 3 строк, в которых указаны пути к языковым файлам и к файлу для сохранения.

Содержимое файлов локализации разбито на несколько разделов:

* текст заставки программы;
* основные сообщения системы;
* руководство оператора;
* предметная область;
* сообщения оператору.

Наиболее перспективными направлениями дальнейшего развития разработанного программного продукта является добавление животных в приют, расширение их функциональности, улучшенное взаимодействие между объектами классов, добавление новых возможностей взаимодействия пользователя с программой.

# Приложение Г

Руководство оператора

Г.1 Назначение программы

Программа предназначена для моделирования процесса работы покупателя с магазином роботов и приютом.

Г.2 Условия выполнения программы

Для выполнения программы необходима ПЭВМ типа IBM PC или программно совместимая с ней, функционирующая под управлением операционной системы семейства Windows (не ниже Windows XP).

Г.3 Выполнение программы

Для переключения языка пользователь должен нажать клавишу желаемого языка в кадре интерфейса. При запуске приложения появляется заставка, для выбора пользователя. На каждом экране можно поменять язык с русского на английский и наоборот.

Зайдя в главное меню, требуется выбрать необходимый пункт. Если необходимо приобрести робота,то для начала нужно перечислить деньги в магазин роботов и вырать нужную модель. Отдать животное в приют или приобрести, просмотреть информацию о приюте или покормить животных можно в пункте “Приют”.

Для выхода программы следует нажать клавишу ‘e’.Выбор необходимого пункта меню осуществляется путем нажатия клавиш с числами (0-9). Для возврата в предыдущее меню используется клавиша ‘ESC’. При нажатии ‘e’ в главном меню программа завершит свою работу.

Г.4 Сообщения оператору

Все сообщения оператору заданы в файле (см. Приложение E)

# Приложение д

ТЕКСТ ПРОГРАММЫ

//Animal.h

#ifndef ANIMAL

#define ANIMAL

#include "std.h"

class Animal {

protected:

float m\_Mass;

string m\_Sex;

string m\_Color;

int m\_Age;

public:

Animal() {}

Animal(float mass, string sex, string color, int age) :m\_Mass(mass), m\_Sex(sex), m\_Color(color), m\_Age(age) {}

virtual ~Animal() = 0;

void setMass();

void setSex();

void setColor();

void setAge();

virtual void print() const = 0;

virtual const string getName() const = 0;

virtual const string getRace()const = 0;

virtual const int getCost() const = 0;

virtual void saveD() = 0;

virtual void cntOb() = 0;

const float getMass() const { return m\_Mass; }

const string getColor()const { return m\_Color; }

const string getSex()const { return m\_Sex; }

const int getAge()const { return m\_Age; }

Animal& operator--();

friend Animal& operator++(Animal& am);

friend ostream& operator<<(ostream& out, const Animal& an);

};

#endif // !ANIMAL

//Animal.cpp

#include "Dog.h"

#include "Cat.h"

#include "Lang.h"

extern Interface lang;

void corrE(int& buff) {

do {

cin >> buff;

if (cin.fail() || buff < 1) {

lang.outputTextFromFile(13703, 1);// "Вы ошиблсиь.\n";

cin.clear();

cin.ignore(32767, '\n');

continue;

}

} while (buff < 1 || cin.fail());

}

void corrE(float& buff) {

do {

cin >> buff;

if (cin.fail() || buff < 0.1) {

lang.outputTextFromFile(13774, 1);// "Вы ошиблсиь.\n";

cin.clear();

cin.ignore(32767, '\n');

continue;

}

} while (buff < 0.1 || cin.fail());

}

Animal::~Animal(){}

void Animal::setMass() {

float buff;

lang.outputTextFromFile(12567, 1);//"Введите массу: ";

corrE(buff);

m\_Mass = buff;

}

void Animal::setSex() {

string str;

lang.outputTextFromFile(13206, 1);// "Введите пол питомца: " << endl;

cin >> str;

m\_Sex = str;

}

void Animal::setColor() {

string str;

lang.outputTextFromFile(13277, 1);// "Введите цвет питомца: " << endl;

cin >> str;

m\_Color = str;

}

void Animal::setAge() {

int buff;

lang.outputTextFromFile(13348, 1);//"Введите возраст питомца: ";

corrE(buff);

m\_Age = buff;

}

Animal& Animal::operator--() {

if (m\_Mass <= 0.1)

lang.outputTextFromFile(13419, 2);// "Масса слишком мала!" << endl;

else {

m\_Mass -= 0.1;

lang.outputTextFromFile(13561, 2);// "Масса успешно изменена!";

}

return \*this;

}

ostream& operator<<(ostream& out, const Animal& an) {

an.print();

return out;

}

Animal& operator++(Animal& am) {

am.m\_Mass += 0.1;

lang.outputTextFromFile(13561, 2);// "Масса успешно изменена!";

return am;

}

//Cat.h

#ifndef CAT

#define CAT

#include "Animal.h"

class Cat :public Animal {

private:

string m\_Name;

string m\_Race;

int m\_Cost;

static int cntCat;

public:

Cat() { cntCat++; }

Cat(float mass, string sex, string color, int age, string race, string name, int cost) :

Animal(mass, sex, color, age), m\_Race(race), m\_Name(name), m\_Cost(cost) {

cntCat++;

}

~Cat() { cntCat--; }

void setName();

void setRace();

void setCost();

void setCntCat(int);

virtual void cntOb();

virtual void saveD();

virtual void print()const ;

const string getName()const { return m\_Name; }

const string getRace()const { return m\_Race; }

const int getCost()const { return m\_Cost; }

friend istream& operator>>(istream& in, Cat& cat);

friend ifstream& operator>>(ifstream& stin, Cat& cat);

friend int getCntCat();

};

inline int getCntCat() { return Cat::cntCat; }

#endif // !CAT

//Dog.h

#ifndef DOG

#define DOG

#include "Animal.h"

class Dog : public Animal {

protected:

string m\_Name;

string m\_Race;

string m\_Type;

int m\_Cost;

static int cntDog;

public:

Dog() { cntDog++; }

Dog(float mass, string sex, string color, int age, string race, string type, string name, int cost) :

Animal(mass, sex, color, age), m\_Race(race), m\_Type(type), m\_Name(name), m\_Cost(cost)

{

cntDog++;

}

friend int getCntDog();

virtual ~Dog() { cntDog--; }

void setName();

void setRace();

void setType();

void setCost();

virtual void cntOb();

virtual void print() const;

virtual void saveD();

void setCntDog(int);

const int getCost()const { return m\_Cost; }

const string getName()const { return m\_Name; }

const string getRace()const { return m\_Race; }

const string getType()const { return m\_Type; }

friend istream& operator>>(istream& in, Dog& dog);

friend ifstream& operator>>(ifstream& stin, Dog& dog);

friend int getCntDog();

};

inline int getCntDog() { return Dog::cntDog; }

#endif // !DOG

//Dog.cpp

#include "Dog.h"

#include "Cat.h"

#include "Lang.h"

extern Interface lang;

extern void corrE(int& buff);

void Dog::setCntDog(int n) {

cntDog = n;

}

void Dog::setName() {

string str;

lang.outputTextFromFile(12709, 2);// "Введите кличку собаки: " << endl;

cin >> str;

m\_Name = str;

}

void Dog::setRace() {

string str;

lang.outputTextFromFile(12851, 1);// "Введите породу собаки: " << endl;

cin >> str;

m\_Race = str;

}

void Dog::setType() {

string str;

lang.outputTextFromFile(12922, 1);// "Введите вид собаки" << endl;

cin >> str;

m\_Type = str;

}

void Dog::setCost() {

int buff;

lang.outputTextFromFile(12638, 1);//"Введите стоимость: ";

corrE(buff);

m\_Cost = buff;

}

void Dog::print() const {

lang.outputTextFromFile(15408, 2);//

cout << getName()<<endl;

lang.outputTextFromFile(15550, 1);//

cout << getAge() << endl;

lang.outputTextFromFile(15621, 1);//

cout << getSex() << endl;

lang.outputTextFromFile(15692, 1);//

cout << getColor() << endl;

lang.outputTextFromFile(15763, 1);//

cout << getMass() << endl;

lang.outputTextFromFile(15834, 1);//

cout << getRace() << endl;

lang.outputTextFromFile(15905, 1);//

cout << getCost() << endl;

lang.outputTextFromFile(15976, 1);//

cout << getType() << endl;

}

istream& operator>>(istream& in, Dog& dog) {

dog.setName();

dog.setAge();

dog.setSex();

dog.setColor();

dog.setMass();

dog.setRace();

dog.setType();

dog.setCost();

return in;

}

void Dog::saveD(){

lang.fileToSave << "Dog" << endl;

lang.fileToSave << getName()<<endl;

lang.fileToSave << getAge()<<endl;

lang.fileToSave << getSex()<<endl;

lang.fileToSave << getColor()<<endl;

lang.fileToSave << getMass()<<endl;

lang.fileToSave << getRace() << endl;

lang.fileToSave << getType()<<endl;

lang.fileToSave << getCost() << endl;

}

void Dog::cntOb() {

cntDog--;

}

ifstream& operator>>(ifstream& stin, Dog& dog) {

string str;

long int buff;

if (!stin.eof())

{

stin >> str;

dog.m\_Name = str;

}

else

throw - 1;

try {

if (!stin.eof()) {

stin >> str;

buff = atoi(str.c\_str());

if (!isdigit(buff))

dog.m\_Age = buff;

else

throw "k";

}

else throw - 1;

}

catch (string) {

cerr << "Некорректные данные" << endl;

\_getch();

lang.fileSelectionForDownload();

}

if (!stin.eof())

{

stin >> str;

dog.m\_Sex = str;

}else

throw - 1;

if (!stin.eof())

{

stin >> str;

dog.m\_Color = str;

}else

throw - 1;

if (!stin.eof())

{

stin >> str;

buff = atof(str.c\_str());

dog.m\_Mass = buff;

}

else throw - 1;

if (!stin.eof()){

stin >> str;

dog.m\_Race = str;

}else

throw - 1;

try {

if (!stin.eof()) {

stin >> str;

buff = atoi(str.c\_str());

if (!isdigit(buff))

dog.m\_Cost = buff;

else

throw "k";

}

else throw - 1;

}

catch (string) {

cerr << "Некорректные данные" << endl;

\_getch();

lang.fileSelectionForDownload();

}

if (!stin.eof()) {

stin >> str;

dog.m\_Type = str;

}

else

throw - 1;

return stin;

}

void Cat::cntOb() {

cntCat--;

}

void Cat::setCntCat(int n) {

cntCat = n;

}

void Cat::setCost() {

int buff;

lang.outputTextFromFile(12638, 1);//"Введите стоимость: ";

corrE(buff);

m\_Cost = buff;

}

void Cat::setName() {

string str;

lang.outputTextFromFile(12993, 2);// "Введите кличку кошки: " << endl;

cin >> str;

m\_Name = str;

}

void Cat::setRace() {

string str;

lang.outputTextFromFile(13135, 1);// "Введите породу кошки: " << endl;

cin >> str;

m\_Race = str;

}

void Cat::print()const {

lang.outputTextFromFile(15408, 2);//

cout << getName() << endl;

lang.outputTextFromFile(15550, 1);//

cout << getAge() << endl;

lang.outputTextFromFile(15621, 1);//

cout << getSex() << endl;

lang.outputTextFromFile(15692, 1);//

cout << getColor() << endl;

lang.outputTextFromFile(15763, 1);//

cout << getMass() << endl;

lang.outputTextFromFile(15834, 1);//

cout << getRace() << endl;

lang.outputTextFromFile(15905, 1);//

cout << getCost() << endl;

}

istream& operator>>(istream& in, Cat& cat) {

cat.setName();

cat.setAge();

cat.setSex();

cat.setColor();

cat.setMass();

cat.setRace();

cat.setCost();

return in;

}

void Cat::saveD() {

lang.fileToSave << "Cat" << endl;

lang.fileToSave << getName() << endl;

lang.fileToSave << getAge() << endl;

lang.fileToSave << getSex() << endl;

lang.fileToSave << getColor() << endl;

lang.fileToSave << getMass() << endl;

lang.fileToSave << getRace() << endl;

lang.fileToSave << getCost() << endl;

}

ifstream& operator>>(ifstream& stin, Cat& cat) {

string str;

long int buff;

if (!stin.eof())

{

stin >> str;

cat.m\_Name = str;

}

else

throw - 1;

try {

if (!stin.eof()) {

stin >> str;

buff = atoi(str.c\_str());

if (!isdigit(buff))

cat.m\_Age = buff;

else

throw "k";

}

else throw - 1;

}

catch (string) {

cerr << "Некорректные данные" << endl;

\_getch();

lang.fileSelectionForDownload();

}

if (!stin.eof())

{

stin >> str;

cat.m\_Sex = str;

}

else

throw - 1;

if (!stin.eof())

{

stin >> str;

cat.m\_Color = str;

}

else

throw - 1;

if (!stin.eof())

{

stin >> str;

buff = atof(str.c\_str());

cat.m\_Mass = buff;

}

else throw - 1;

if (!stin.eof()) {

stin >> str;

cat.m\_Race = str;

}

else

throw - 1;

try {

if (!stin.eof()) {

stin >> str;

buff = atoi(str.c\_str());

if (!isdigit(buff))

cat.m\_Cost = buff;

else

throw "k";

}

else throw - 1;

}

catch (string) {

cerr << "Некорректные данные" << endl;

\_getch();

lang.fileSelectionForDownload();

}

return stin;

}

//Shelter.h

#ifndef SHELTER

#define SHELTER

#include "RoboDog.h"

#include "Cat.h"

class Shelter {

private:

struct list {

Animal\* inf;

RoboDog\* inf1;

list \* next;

list\* prev;

};

int size;

string m\_Name;

list\* head, \*tail;

int moneyB;

vector<Animal\*> anim;

vector<Robot\*> rb;

public:

friend class Control;

friend class Interface;

Shelter() {

head = tail = NULL; moneyB = 0; size = 0;

}

~Shelter() { removeAllShelter(); }

const int getMoney()const { return moneyB; }

void setMoney();

void setMoney(int);

void addAnim(Animal\*);

void addAnim(RoboDog\*);

void removeAllShelter();

void removeAnim(vector<string>);

void showAllShelter();

void changeM(int);

void saveSh();

};

#endif // !SHELTER

//Shelter.cpp

#include "Shelter.h"

#include "Lang.h"

extern void corrE(int& buff);

extern Interface lang;

void Shelter::setMoney() {

int buff;

lang.outputTextFromFile(11999, 2);//"Введите ваш денежный запас: ";

corrE(buff);

moneyB = buff;

}

void Shelter::setMoney(int buff) {

moneyB = buff;

}

void Shelter::addAnim(Animal\* an) {

list\* tmp=new list;

tmp->inf = an;

tmp->inf1 = NULL;

tmp->next = NULL;

tmp->prev = NULL;

size++;

if (head == NULL) tail = head = tmp;

else

{

tmp->prev = tail;

tail->next = tmp;

tail = tmp;

}

moneyB += an->getCost();

}

void Shelter::addAnim(RoboDog\* rb) {

list\* tmp = new list;

tmp->inf = NULL;

tmp->inf1 = rb;

tmp->next = NULL;

tmp->prev = NULL;

size++;

if (head == NULL) tail = head = tmp;

else

{

tmp->prev = tail;

tail->next = tmp;

tail = tmp;

}

moneyB += rb->Robot::getCost();

}

void Shelter::showAllShelter() {

list\* tmp = head;

if (head != NULL) {

while (tmp) {

if (tmp->inf != NULL)

cout << \*tmp->inf << endl;

else

cout << (Robot)\*tmp->inf1 << endl;

tmp = tmp->next;

}

}

else

lang.outputTextFromFile(12141, 2);//" "Приют пуст!" << endl;

}

void Shelter::removeAllShelter() {

list\* p = head;

if (head != NULL) {

while (head) {

p = head;

head = head->next;

if (p->inf != NULL)

delete p->inf;

else

delete p->inf1;

delete p;

}

size = 0;

}

}

void Shelter::removeAnim(vector<string> str) {

bool t = false;

list\* tmp = head;

while (tmp) {

if (str.size() == 4) {

if (tmp->inf1 != NULL && tmp->inf1->getName() == str[0] && tmp->inf1->getSex() == str[1] && tmp->inf1->getRace() == str[2]

&& tmp->inf1->getModel() == str[3]) {

t = true;

break;

}

}else

if (tmp->inf != NULL && tmp->inf->getName() == str[0] && tmp->inf->getSex() == str[1]

&& tmp->inf->getRace() == str[2]) {

t = true;

break;

}

tmp = tmp->next;

}

if (t) {

if ((tmp->inf != NULL && tmp->inf->getCost() <= getMoney()) || (tmp->inf1 != NULL && tmp->inf1->Robot::getCost() <= getMoney()))

{if (tmp == head) {

head = head->next;

if (tmp->inf != NULL) {

anim.push\_back(tmp->inf);

tmp->inf->cntOb();

}

if (tmp->inf1 != NULL) {

tmp->inf1->setCntRoboDog(getCntRoboDog()-1);

rb.push\_back(tmp->inf1);

}

delete tmp;

tmp = head;

size--;

}

else

{

tmp->prev->next = tmp->next;

if (tmp->inf != NULL) {

anim.push\_back(tmp->inf);

tmp->inf->cntOb();

}

if (tmp->inf1 != NULL) {

tmp->inf1->setCntRoboDog(getCntRoboDog() - 1);

rb.push\_back(tmp->inf1);

}

delete tmp;

size--;

}

lang.outputTextFromFile(17254, 2);//Успешно куплено!

}

else

lang.outputTextFromFile(11715, 2);// "У вас недостаточно денег!" << endl;

}

else

lang.outputTextFromFile(12283, 2);//" "Такого животного нет в приюте!" << endl;

////

}

void Shelter::changeM(int buff) {

string str[3];

lang.outputTextFromFile(10082, 2);// "Введите кличку, пол, породу животного: " << endl;

lang.outputTextFromFile(10224, 1);// "Введите кличку: " << endl;

cin >> str[0];

lang.outputTextFromFile(10295, 1);// "Введите пол: " << endl;

cin >> str[1];

lang.outputTextFromFile(10366, 1);// "Введите породу: " << endl;

cin >> str[2];

list\* tmp=head;

bool t = false, t1 = false;

while (tmp) {

if (tmp->inf != NULL && tmp->inf->getName() == str[0] && tmp->inf->getSex() == str[1]

&& tmp->inf->getRace() == str[2]) {

t = true;

break;

}

if (tmp->inf1 != NULL && tmp->inf1->getName() == str[0] && tmp->inf1->getSex() == str[1]

&& tmp->inf1->getRace() == str[2]) {

t1 = true;

break;

}

tmp = tmp->next;

}

if (t) {

if (buff == 1)

++\*tmp->inf;

else if (buff == 2)

--\*tmp->inf;

else if (buff == 3)

tmp->inf->setMass();

}

if (t1) {

lang.outputTextFromFile(12425, 2);// "Данное животное робототизировано, его масса неизменна!" << endl;

}

if (!t1 && !t)

lang.outputTextFromFile(12283, 2);// "Такого животного нет!" << endl;

}

void Shelter::saveSh() {

list\* tmp = head;

if (size != 0) {

lang.fileToSave << "Shelter" << endl;

while (tmp) {

if (tmp->inf != NULL)

tmp->inf->saveD();

else

tmp->inf1->saveDR();

tmp = tmp->next;

}

}

if (anim.size() != 0 || rb.size() != 0) {

lang.fileToSave << "Buyer" << endl;

for (size\_t i = 0; i < anim.size(); i++)

{

anim[i]->saveD();

}

for (size\_t i = 0; i < rb.size(); i++)

{

rb[i]->saveDR();

}

}

}

//Robot.h

#ifndef ROBOT

#define ROBOT

#include "std.h"

class Robot {

protected:

int m\_Series;

float m\_Mass;

string m\_Model;

string m\_Date;

int m\_Cost;

static int cntRobot;

public:

Robot() { cntRobot++; }

Robot(int series, float mass, string model, string date, int cost):

m\_Series(series), m\_Mass(mass), m\_Model(model), m\_Date(date), m\_Cost(cost) {

cntRobot++;

}

virtual ~Robot() { cntRobot--; }

void setSeries();

void setMass();

void setModel();

void setDate();

void setCost();

void setCntRobot(int);

virtual void saveDR();

virtual void downL();

virtual void print() const;

const int getSeries() const { return m\_Series; }

const float getMass() const { return m\_Mass; }

const string getModel() const{ return m\_Model; }

const string getDate()const { return m\_Date; }

const int getCost()const { return m\_Cost; }

friend ostream& operator<<(ostream& out, const Robot& rb);

friend ifstream& operator>>(ifstream& stin, Robot\* rb);

friend int getCntRobot();

};

inline int getCntRobot() { return Robot::cntRobot; }

#endif

//Interface.h

#ifndef INTER

#define INTER

#include "Control.h"

class Interface {

private:

fstream ru\_interface;

fstream eng\_interface;

fstream configFile;

ofstream fileToSave;

ifstream fileToDownload;

bool ln;

vector<string> waysFile;

public:

friend class Dog;

friend class Cat;

friend class Robot;

friend class RoboDog;

friend class Shelter;

Interface() {}

void beginningOfWork();

void titlePage();

void switchLanguage();

void aboutMe();

void aboutTheProgram();

void exitFromTheProgram();

void operatorsManual();

void outputTextFromFile(int num, int k);

void fileSelectionToSave();

void fileSelectionForDownload();

void loadingFromFile();

void saveToFile();

friend ifstream& operator>>(ifstream& stin, Dog& dog);

};

#endif

// Interface.cpp

#include "Lang.h"

const int sizeF = 17680;

extern Control contr;

void Interface::beginningOfWork() {

ln = false;

string conf;

try {

cout << "Введите путь к файлу конфигурации/Enter the path to the configuration file" << endl;

cin >> conf;

configFile.open(conf);

if (!configFile.is\_open())

throw conf;

if (configFile.eof())

throw - 1;

}

catch (string conf) {

cerr << conf << "-Путь неверен/Wrong way." << endl;

\_getch();

exit(-1);

}

catch (int) {

cerr << "Конфигурационный файл пустой/Config file is empty" << endl;

\_getch();

configFile.close();

exit(-1);

}

try {

for (size\_t i = 0; i < 3; i++)

{

if (!configFile.eof()) {

configFile >> conf;

waysFile.push\_back(conf);

}

else

throw - 1;

}

}

catch (int) {

cerr << "В файле конфигурации не хватает путей!/Configuration file has not enough paths!" << endl;

\_getch();

exit(-1);

}

configFile.close();

}

void Interface::outputTextFromFile(int num, int k) {

string showMenu = " ";

if (!ln) {

ru\_interface.seekg(num, ios\_base::beg);

for (size\_t i = 0;i < k; i++)

{

showMenu = "";

getline(ru\_interface, showMenu);

cout << showMenu;

}

int l = ru\_interface.tellg();

}

else

{

eng\_interface.seekg(num, ios\_base::beg);

for (size\_t i = 0;i < k; i++)

{

showMenu = "";

getline(eng\_interface, showMenu);

cout << showMenu;

}

}

}

void Interface::switchLanguage() {

system("cls");

string str;

if (ln) { //eng->ru

ln = false;

}

else { //ru->eng

if (!eng\_interface.is\_open())

{

try {

eng\_interface.open(waysFile[1]);

if (!eng\_interface.is\_open())

throw waysFile[1];

int size;

eng\_interface.seekg(0, std::ios::end);

size = eng\_interface.tellg();

if (size < sizeF || size>sizeF)

throw "Файл поврежден/The file is damaged";

else

eng\_interface.seekg(0, ios::beg);

eng\_interface >> str;

if (eng\_interface.eof())

throw - 1;

}

catch (const char\* s) {

cerr << s << endl;

\_getch();

ru\_interface.close();

configFile.close();

eng\_interface.close();

exit(-1);

}

catch (string) {

cerr << waysFile[1] << " :Ошибка открытия файла./Error opening file." << endl;

\_getch();

configFile.close();

ru\_interface.close();

exit(-1);

}

catch (int) {

cerr << "Файл пуст/Config file is empty" << endl;

\_getch();

configFile.close();

ru\_interface.close();

exit(-1);

}

}

ln = true;

}

}

void Interface::saveToFile() {

contr.sh.saveSh();

if (contr.shRb.size() != 0) {

fileToSave << "RoboShop" << endl;

for (size\_t i = 0; i < contr.shRb.size(); i++)

{

contr.shRb[i]->saveDR();

}

}

fileToSave.close();

}

void Interface::fileSelectionToSave() {

string str;

system("cls");

int key;

outputTextFromFile(14129, 5);//// "Загрузить в файл из конфига/Новый файл/отмена" << endl;

key = \_getch();

switch (key) {

case kC::one:

try {

fileToSave.open(waysFile[2], ofstream::trunc);

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

throw - 1;

}

saveToFile();

}

catch (int) {

outputTextFromFile(14484, 2);// "Неверный путь" << endl;

\_getch();

fileSelectionToSave();

}

break;

case kC::two:

outputTextFromFile(14626, 2);// "Введите путь файла для сохранения" << endl;

cin >> str;

try {

fileToSave.open(str, ofstream::trunc);

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

throw - 1;

}

saveToFile();

}

catch (int) {

outputTextFromFile(14484, 2);// "Неверный путь" << endl;

\_getch();

fileSelectionToSave();

}

break;

case kC::three:

contr.mainMenuOfTheProgram();

case kC::TAB:

switchLanguage();

fileSelectionToSave();

break;

default:

fileSelectionToSave();

break;

}

}

void Interface::loadingFromFile() {

string str;

if (!fileToDownload.eof())

{

for (size\_t i = 0; !fileToDownload.eof(); i++)

{

try {

fileToDownload >> str;

if (str == "Shelter") {

do {

fileToDownload >> str;

if (str == "Dog") {

Dog\* dog = new Dog;

fileToDownload >> \*dog;

contr.sh.addAnim(dog);

}

else if (str == "Cat") {

Cat\* ct = new Cat;

fileToDownload >> \*ct;

contr.sh.addAnim(ct);

}

else if (str == "RoboDog") {

RoboDog\* rd = new RoboDog;

fileToDownload >> ((Robot\*)rd);

contr.sh.addAnim(rd);

}

else {

while (str != "Buyer" || str != "RoboShop" || str != "RoboDog" || str != "Cat" || str != "Dog") {

fileToDownload >> str;

if (fileToDownload.eof())

break;

}

}

if (fileToDownload.eof())

break;

str = "";

} while (str != "Dog" || str != "Cat" || str != "RoboDog"||str!="Buyer"||str!="RoboShop");

}

else if (str == "Buyer") {

fileToDownload >> str;

if (str == "Dog") {

Dog\* dog = new Dog;

fileToDownload >> \*dog;

contr.sh.anim.push\_back(dog);

}

else if (str == "Cat") {

Cat\* ct = new Cat;

fileToDownload >> \*ct;

contr.sh.anim.push\_back(ct);

}

else if (str == "RoboDog") {

RoboDog\* rd = new RoboDog;

fileToDownload >> ((Robot\*)rd);

contr.sh.rb.push\_back(rd);

}

else if (str == "Robot") {

Robot\* rd = new Robot;

fileToDownload >> rd;

contr.sh.rb.push\_back(rd);

}

else{

while (str != "RoboShop" || str != "RoboDog" || str != "Cat" || str != "Dog"||str!="Robot") {

fileToDownload >> str;

if (fileToDownload.eof())

break;

}

}

}

else if (str == "RoboShop") {

fileToDownload >> str;

if (str == "Robot") {

Robot\* rd = new Robot;

fileToDownload >> rd;

contr.shRb.push\_back(rd);

}

else

{

while ( str != "Robot" ) {

fileToDownload >> str;

if (fileToDownload.eof())

break;

}

}

}

outputTextFromFile(17396, 2);//Успешно загружено!

}

catch (int) {

outputTextFromFile(15124, 2);// "Непредвиденный конец файла" << endl;

\_getch();

contr.mainMenuOfTheProgram();

}

}

}

else {

outputTextFromFile(15266, 2);// "File empty!" << endl;

}

outputTextFromFile(8875, 4);//ЕСК

int key;

key = \_getch();

do {

switch (key)

{

case kC::TAB:

switchLanguage();

fileSelectionForDownload();

break;

case kC::ESC:

contr.mainMenuOfTheProgram();

break;

case kC::e:

exitFromTheProgram();

default:

do {

key = \_getch();

} while (key != kC::ESC && key != kC::e);

}

} while (key != kC::ESC && key != kC::e);

}

void Interface::fileSelectionForDownload() {

string str;

system("cls");

int key;

outputTextFromFile(14769, 5);// "Выгрузить из файла из конфига/Новый файл/отмена" << endl;

key = \_getch();

switch (key) {

case kC::one:

try {

fileToDownload.open(waysFile[2]);

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

throw - 1;

}

loadingFromFile();

}

catch (int) {

outputTextFromFile(14484, 2);// "Неверный путь" << endl;

\_getch();

fileSelectionForDownload();

}

break;

case kC::two:

outputTextFromFile(14626, 2);// "Введите путь файла для сохранения" << endl;

cin >> str;

try {

fileToDownload.open(str);

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

throw - 1;

}

loadingFromFile();

}

catch (int) {

outputTextFromFile(14484, 2);// "Неверный путь" << endl;

\_getch();

fileSelectionForDownload();

}

break;

case kC::three:

contr.mainMenuOfTheProgram();

case kC::TAB:

switchLanguage();

fileSelectionForDownload();

break;

default:

fileSelectionForDownload();

break;

}

}

void Interface::exitFromTheProgram() {

system("cls");

int key;

outputTextFromFile(13774, 5);////EXIT

key = \_getch();

switch (key) {

case kC::one:

fileSelectionToSave();

ru\_interface.close();

if (fileToDownload.is\_open())

fileToDownload.close();

if (eng\_interface.is\_open())

eng\_interface.close();

if (fileToSave.is\_open())

fileToSave.close();

exit(-1);

break;

case kC::two:

ru\_interface.close();

if (fileToDownload.is\_open())

fileToDownload.close();

if (fileToSave.is\_open())

fileToSave.close();

if (eng\_interface.is\_open())

eng\_interface.close();

exit(-1);

default:

exitFromTheProgram();

break;

}

}

void Interface::operatorsManual() {

system("cls");

system("mode con cols=69 lines=14");

string showMenu;

int key;

outputTextFromFile(2556, 14);//Помощь

key = \_getch();

switch (key) {

case kC::TAB: {

switchLanguage();

operatorsManual();

break;

}

case e:

{

exitFromTheProgram();

}

case kC::ESC: {

titlePage();

break;

}

default: {

operatorsManual();

break;

}

}

}

void Interface::aboutTheProgram() {

system("cls");

system("mode con cols=69 lines=33");

string showMenu;

int key;

outputTextFromFile(3550, 33);//О Программе

key = \_getch();

switch (key) {

case kC::TAB: {

switchLanguage();

aboutTheProgram();

break;

}

case e:

{

exitFromTheProgram();

}

case kC::ESC: {

titlePage();

break;

}

default: {

aboutTheProgram();

break;

}

}

}

void Interface::aboutMe() {

system("cls");

system("mode con cols=69 lines=12");

string showMenu;

int key;

outputTextFromFile(1704, 12);//Об авторе

key = \_getch();

switch (key) {

case kC::TAB: {

switchLanguage();

aboutMe();

break;

}

case e:

{

exitFromTheProgram();

}

case kC::ESC: {

titlePage();

break;

}

default: {

aboutMe();

key = 0;

break;

}

}

}

void Interface::titlePage() {

system("cls");

system("mode con cols=69 lines=24");

string str;

if (!ln && !ru\_interface.is\_open()) {

try {

ru\_interface.open(waysFile[0]);

if (!ru\_interface.is\_open())

throw waysFile[0];

ru\_interface >> str;

if (ru\_interface.eof())

throw - 1;

int size = 0;

ru\_interface.seekg(0, std::ios::end);

size = ru\_interface.tellg();

if (size < sizeF || size>sizeF) {

throw "Файл поврежден/The file is damaged";

}

else

ru\_interface.seekg(0, ios::beg);

}

catch (int) {

cerr << "Файл русского интерфейса пуст/Russian file is empty" << endl;

\_getch();

configFile.close();

ru\_interface.close();

exit(-1);

}

catch (const char\* s) {

cerr << s << endl;

if (ru\_interface.is\_open())

ru\_interface.close();

\_getch();

configFile.close();

exit(-1);

}

catch (string) {

cerr << waysFile[0] << " :Ошибка открытия файла./Error opening file." << endl;

\_getch();

configFile.close();

exit(-1);

}

}

outputTextFromFile(0, 24);//Основное меню

int key;

key = \_getch();

switch (key) {

case kC::ENTER: {

contr.mainMenuOfTheProgram();

break;

}

case kC::a: {

operatorsManual();

break;

}

case kC::s: {

aboutTheProgram();

break;

}

case kC::d: {

aboutMe();

break;

}

case kC::TAB: {

switchLanguage();

titlePage();

break;

}

case kC::ESC: {

exitFromTheProgram();

}

default: {

titlePage();

key = 0;

break;

}

}

}

//Robot.cpp

#include "RoboDog.h"

#include "Lang.h"

extern Interface lang;

extern void corrE(int& buff);

extern void corrE(float& buff);

void Robot::setCntRobot(int n) {

cntRobot = n;

}

void Robot::setSeries() {

int buff;

lang.outputTextFromFile(11502, 1);//"Введите серию робота: ";

corrE(buff);

m\_Series = buff;

}

void Robot::setMass() {

float buff;

lang.outputTextFromFile(12567, 1);//"Введите массу: ";

corrE(buff);

m\_Mass = buff;

}

void Robot::setModel() {

string str;

lang.outputTextFromFile(11289, 2);// "Введите модель: " << endl;

cin >> str;

m\_Model = str;

}

void Robot::setDate() {

string str;

lang.outputTextFromFile(11431, 1);// "ВВедите дату: " << endl;

cin >> str;

m\_Date = str;

}

void Robot::setCost() {

lang.outputTextFromFile(12638, 1);//"Введите стоимость: ";

int buff;

corrE(buff);

m\_Cost = buff;

}

void Robot::print() const {

lang.outputTextFromFile(16047, 2);//

cout<<getModel() << endl;

lang.outputTextFromFile(16189, 1);//

cout << getSeries() << endl;

lang.outputTextFromFile(16260, 1);//

cout << getMass() << endl;

lang.outputTextFromFile(16331, 1);//

cout << getDate() << endl;

lang.outputTextFromFile(16402, 1);//

cout << getCost() << endl;

}

ostream& operator<<(ostream& out,const Robot& rb) {

rb.print();

return out;

}

void Robot::saveDR() {

lang.fileToSave << "Robot" << endl;

lang.fileToSave << getModel()<<endl;

lang.fileToSave << getSeries()<<endl;

lang.fileToSave << getMass()<<endl;

lang.fileToSave << getDate() << endl;

lang.fileToSave << getCost() << endl;

}

void RoboDog::setPerRobo() {

int buff;

lang.outputTextFromFile(12709, 1);//""Введите процент робототизированности";

corrE(buff);

m\_PerRobo = buff;

}

void Robot::downL() {

string str;

int buff;

if (!lang.fileToDownload.eof())

lang.fileToDownload >> m\_Model;

else throw - 1;

try {

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

buff = atoi(str.c\_str());

if (!isdigit(buff))

m\_Series = buff;

else

throw "k";

}

else throw - 1;

}

catch (string) {

cerr << "Некорректные данные" << endl;

\_getch();

lang.fileSelectionForDownload();

}

if (!lang.fileToDownload.eof())

{

lang.fileToDownload >> str;

buff = atof(str.c\_str());

m\_Mass = buff;

}

else throw - 1;

if (!lang.fileToDownload.eof())

lang.fileToDownload >> m\_Date;

else throw - 1;

try {

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

buff = atoi(str.c\_str());

if (!isdigit(buff))

m\_Cost = buff;

else

throw "k";

}

else throw - 1;

}

catch (string) {

cerr << "Некорректные данные" << endl;

\_getch();

lang.fileSelectionForDownload();

}

}

ifstream& operator>>(ifstream& stin, Robot\* rb) {

rb->downL();

return stin;

}

void RoboDog::setCntRoboDog(int n) {

cntRoboDog = n;

}

void RoboDog::print() const {

lang.outputTextFromFile(15408, 2);//

cout << getName() << endl;

lang.outputTextFromFile(15550, 1);//

cout << getAge() << endl;

lang.outputTextFromFile(15621, 1);//

cout << getSex() << endl;

lang.outputTextFromFile(15692, 1);//

cout << getColor() << endl;

lang.outputTextFromFile(15763, 1);//

cout << Robot::getMass() << endl;

lang.outputTextFromFile(15834, 1);//

cout << getRace() << endl;

lang.outputTextFromFile(16473, 1);//

cout << getType() << endl;

lang.outputTextFromFile(16544, 1);//

cout << getModel()<< endl;

lang.outputTextFromFile(16615, 1);//

cout << getSeries() << endl;

lang.outputTextFromFile(16686, 1);//

cout << getDate() << endl;

lang.outputTextFromFile(16757, 1);//

cout << getPerRobo() << endl;

lang.outputTextFromFile(16828, 1);//

cout << Robot::getCost() << endl;

}

istream& operator>>(istream& in, RoboDog& rd) {

rd.setName();

rd.setAge();

rd.setSex();

rd.setColor();

rd.Robot::setMass();

rd.setRace();

rd.setType();

rd.setModel();

rd.setSeries();

rd.setDate();

rd.setPerRobo();

rd.Robot::setCost();

return in;

}

void RoboDog::saveDR() {

lang.fileToSave << "RoboDog" << endl;

lang.fileToSave << getName() << endl;

lang.fileToSave << getAge() << endl;

lang.fileToSave << getSex() << endl;

lang.fileToSave << getColor() << endl;

lang.fileToSave << Robot::getMass() << endl;

lang.fileToSave << getRace() << endl;

lang.fileToSave << getType() << endl;

lang.fileToSave << getModel() << endl;

lang.fileToSave << getSeries() << endl;

lang.fileToSave << getDate() << endl;

lang.fileToSave << getPerRobo() << endl;

lang.fileToSave << Robot::getCost() << endl;

}

void RoboDog::downL() {

string str;

long int buff;

if (!lang.fileToDownload.eof())

{

lang.fileToDownload >> str;

m\_Name = str;

}

else

throw - 1;

try {

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

buff = atoi(str.c\_str());

if (!isdigit(buff))

m\_Age = buff;

else

throw "k";

}

else throw - 1;

}

catch (string) {

cerr << "Некорректные данные" << endl;

\_getch();

lang.fileSelectionToSave();

}

if (!lang.fileToDownload.eof())

{

lang.fileToDownload >> str;

m\_Sex = str;

}

else

throw - 1;

if (!lang.fileToDownload.eof())

{

lang.fileToDownload >> str;

m\_Color = str;

}

else

throw - 1;

if (!lang.fileToDownload.eof())

{

lang.fileToDownload >> str;

buff = atof(str.c\_str());

Robot::m\_Mass = buff;

}

else throw - 1;

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

m\_Race = str;

}

else

throw - 1;

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

m\_Type = str;

}

else

throw - 1;

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

m\_Model = str;

}

else throw - 1;

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

buff = atoi(str.c\_str());

m\_Series = buff;

}

else throw - 1;

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

m\_Date = str;

}

else throw - 1;

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

buff = atoi(str.c\_str());

m\_PerRobo = buff;

}

else throw - 1;

if (!lang.fileToDownload.eof()) {

lang.fileToDownload >> str;

buff = atoi(str.c\_str());

Robot::m\_Cost = buff;

}

else throw - 1;

}

//RoboDog.h

#ifndef ROBODOG

#define ROBODOG

#include "Robot.h"

#include "Dog.h"

class RoboDog:public Dog, public Robot {

private:

int m\_PerRobo;

static int cntRoboDog;

public:

RoboDog() { cntRoboDog++; cntDog--;cntRobot--; }

RoboDog(float mass, string sex, string color, int age, string race, string type, string name, int series, string model, string date, int cost, int per) :

Dog(mass, sex, color, age, race, type, name, cost), Robot(series, mass, model, date, cost), m\_PerRobo(per)

{

cntRoboDog++; cntDog--;cntRobot--;

}

~RoboDog() { cntRoboDog--; }

void setPerRobo();

void setCntRoboDog(int);

virtual void print() const;

virtual void saveDR();

virtual void downL();

const int getPerRobo() const { return m\_PerRobo; }

friend istream& operator>>(istream& in, RoboDog& rd);

friend int getCntRoboDog();

};

inline int getCntRoboDog() { return RoboDog::cntRoboDog; }

#endif

//std.h

#ifndef STD

#define STD

#include <iostream>

#include <stdlib.h>

#include <conio.h>

#include <fstream>

#include <string>

#include <vector>

#include <locale.h>

#include <Windows.h>

#include <functional>

#include <cstdio>

#include <cstdlib>

using namespace std;

#endif

//Control.h

#ifndef CONTROL

#define CONTROL

#include "Shelter.h"

enum kC

{

TAB = 9,

ENTER = 13,

ESC = 27,

a = 97,

d = 100,

e = 101,

s = 115,

one = 49,

two = 50,

three = 51,

four = 52,

five = 53,

six = 54,

seven = 55,

eigth = 56,

nine = 57,

};

class Control {

private:

Shelter sh;

vector<Robot\*> shRb;

int moneyB;

public:

friend class Interface;

void giveTheAnimalToTheShelter();

void pickUpAnimalsFromTheShelter();

void seeTheWholeShelter();

void changeMassAnimal();

void displayInformationAboutTheShelter();

void viewModelAndPrice();

void findByModel();

void buyRobot();

void mainMenuOfTheProgram();

void menuForShelter();

void menuForShopRobots();

void menuForBuyer();

void addAn(Animal\*an) { sh.addAnim(\*&an); }

void addAn(RoboDog\*rb) { sh.addAnim(\*&rb); }

void addShRb(Robot& b) {

shRb.push\_back(&b);

}

void continueFuncSh();

void continueFuncRSh();

};

#endif

//Control.cpp

#include "Control.h"

#include "Lang.h"

extern void corrE(int& buff);

extern Interface lang;

void Control::continueFuncSh() {

lang.outputTextFromFile(8875, 4);//ЕСК

int key;

key = \_getch();

do {

switch (key)

{

case kC::TAB:

lang.switchLanguage();

menuForShelter();

break;

case kC::ESC:

menuForShelter();

break;

case kC::e:

lang.exitFromTheProgram();

default:

do {

key = \_getch();

} while (key != kC::ESC && key != kC::e);

}

} while (key != kC::ESC || key != kC::e);

}

void Control::giveTheAnimalToTheShelter() {

system("cls");

int key;

lang.outputTextFromFile(9159, 7);//"Выбирите какое животное хотите отдать в приют: 1.Кошку.2.Собаку.3.Робопса" << endl;

corrE(key);

if (key == 1) {

Cat\* c = new Cat;

cin >> \*c;

sh.addAnim(c);

} else if (key == 2) {

Dog\* d = new Dog;

cin >> \*d;

sh.addAnim(d);

}

else if (key == 3) {

RoboDog\* rd = new RoboDog;

cin >> \*rd;

sh.addAnim(rd);

}

continueFuncSh();

}

void Control::pickUpAnimalsFromTheShelter() {

system("cls");

vector<string> par;

string str;

int key;

lang.outputTextFromFile(9656, 6);// "Забрать 1.Обычное животное.2.Робототизированное: ";

corrE(key);

if (key == 1) {

lang.outputTextFromFile(10082, 2);// "Введите кличку, пол, породу животного: " << endl;

lang.outputTextFromFile(10224, 1);// "Введите кличку: " << endl;

cin >> str;

par.push\_back(str);

lang.outputTextFromFile(10295, 1);// "Введите пол: " << endl;

cin >> str;

par.push\_back(str);

lang.outputTextFromFile(10366, 1);// "Введите породу: " << endl;

cin >> str;

par.push\_back(str);

sh.removeAnim(par);

}

else {

lang.outputTextFromFile(10437, 2);// "Введите кличку, пол, породу и модель животного: " << endl;

lang.outputTextFromFile(10224, 1);// "Введите кличку: " << endl;

cin >> str;

par.push\_back(str);

lang.outputTextFromFile(10295, 1);// "Введите пол: " << endl;

cin >> str;

par.push\_back(str);

lang.outputTextFromFile(10366, 1);// "Введите породу: " << endl;

cin >> str;

par.push\_back(str);

lang.outputTextFromFile(10579, 1);// "Введите модель: " << endl;

cin >> str;

par.push\_back(str);

sh.removeAnim(par);

}

continueFuncSh();

}

void Control::seeTheWholeShelter() {

system("cls");

system("mode con cols=69 lines=35");

sh.showAllShelter();

continueFuncSh();

}

void Control::changeMassAnimal() {

system("cls");

int buff;

lang.outputTextFromFile(10650, 4);// "1.Покормить животное, 2.Уменьшить массу, 1.Кардинально изменить: ";

corrE(buff);

sh.changeM(buff);

continueFuncSh();

}

void Control::displayInformationAboutTheShelter() {

system("cls");

string str;

lang.outputTextFromFile(16899, 2);//

cout << sh.size << endl;

lang.outputTextFromFile(17041, 1);//

cout << getCntDog() << endl;

lang.outputTextFromFile(17112, 1);//

cout << getCntCat() << endl;

lang.outputTextFromFile(17183, 1);//

cout << getCntRoboDog() << endl;

lang.outputTextFromFile(8875, 4);//ЕСК

int key;

key = \_getch();

do {

switch (key)

{

case kC::TAB:

lang.switchLanguage();

displayInformationAboutTheShelter();

break;

case kC::ESC:

menuForShelter();

break;

case kC::e:

lang.exitFromTheProgram();

default:

do {

key = \_getch();

} while (key != kC::ESC && key != kC::e);

}

} while (key != kC::ESC || key != kC::e);

}

void Control::continueFuncRSh() {

lang.outputTextFromFile(8875, 4);//ЕСК

int key;

key = \_getch();

do {

switch (key)

{

case kC::TAB:

lang.switchLanguage();

menuForShopRobots();

break;

case kC::ESC:

menuForShopRobots();

break;

case kC::e:

lang.exitFromTheProgram();

default:

do {

key = \_getch();

} while (key != kC::ESC || key != kC::e);

}

} while (key != kC::ESC && key != kC::e);

}

void Control::viewModelAndPrice() {

system("cls");

system("mode con cols=100 lines=50");

lang.outputTextFromFile(17538, 2);//Количество роботов в магазе

cout << getCntRobot() << endl;

for (size\_t i = 0; i < shRb.size(); i++)

{

lang.outputTextFromFile(10934, 2);// "Модель: "

cout<< shRb[i]->getModel() << endl;

lang.outputTextFromFile(11076, 1);// "Цена: "

cout<< shRb[i]->getCost() << endl;

}

continueFuncRSh();

}

void Control::findByModel() {

system("cls");

system("mode con cols=69 lines=100");

string str;

bool t;

lang.outputTextFromFile(11147, 2);// "Введите модель которую хотите найти: " << endl;

cin >> str;

for (size\_t i = 0; i < shRb.size(); i++)

{

if (shRb[i]->getModel()==str) {

t = true;

cout << \*shRb[i] << endl;

}

}

continueFuncRSh();

}

void Control::buyRobot() {

system("cls");

bool t=false, t1=true;

string str[2], str1;

int buff;

lang.outputTextFromFile(11289, 2);// "Введите модель: " << endl;

cin >> str[0];

lang.outputTextFromFile(11431, 1);// "ВВедите дату: " << endl;

cin >> str[1];

lang.outputTextFromFile(11502, 1);//"Введите серию робота: ";

corrE(buff);

for (size\_t i = 0; i < shRb.size(); i++)

{

if (shRb[i]->getModel() == str[0]&&shRb[i]->getDate()==str[1]&&shRb[i]->getSeries()==buff) {

t = true;

if (sh.getMoney() >= shRb[i]->getCost()) {

sh.rb.push\_back(&\*shRb[i]);

for (size\_t j = i; j < shRb.size() - 1; j++)

{

shRb[j] = shRb[j + 1];

}

shRb.pop\_back();

break;

}

else {

t1 = false;

}

}

}

if (t&&t1) {

lang.outputTextFromFile(11573, 2);// "Вы успешно приобрели робота" << endl;

}

else

if (t && !t1)

lang.outputTextFromFile(11715, 2);// "У вас недостаточно денег" << endl;

else

lang.outputTextFromFile(11857, 2);// "Такого робота нет в магазине" << endl;

continueFuncRSh();

}

void Control::menuForShelter() {

system("cls");

lang.outputTextFromFile(6887, 14);//меню приюта

int key;

key = \_getch();

switch (key) {

case kC::one:

pickUpAnimalsFromTheShelter();

break;

case kC::two:

giveTheAnimalToTheShelter();

break;

case kC::three:

seeTheWholeShelter();

break;

case kC::four:

changeMassAnimal();

break;

case kC::five:

displayInformationAboutTheShelter();

break;

case kC::TAB:

lang.switchLanguage();

menuForShelter();

break;

case kC::ESC:

mainMenuOfTheProgram();

break;

case kC::e:

lang.exitFromTheProgram();

break;

default:

menuForShelter();

break;

}

}

void Control::menuForShopRobots() {

system("cls");

lang.outputTextFromFile(7881, 14);//Меню робмагазина

int key;

key = \_getch();

switch (key) {

case kC::one:

viewModelAndPrice();

break;

case kC::two:

findByModel();

break;

case kC::three:

buyRobot();

break;

case kC::TAB:

lang.switchLanguage();

menuForShopRobots();

break;

case kC::ESC:

mainMenuOfTheProgram();

break;

case kC::e:

lang.exitFromTheProgram();

break;

default:

menuForShopRobots();

break;

}

}

void Control::menuForBuyer() {

system("cls");

sh.setMoney();

lang.outputTextFromFile(8875, 4);//ЕСК

int key;

key = \_getch();

do{

switch (key)

{

case kC::TAB:

lang.switchLanguage();

menuForBuyer();

break;

case kC::ESC:

mainMenuOfTheProgram();

break;

case kC::e:

lang.exitFromTheProgram();

default:

{

do {

key = \_getch();

} while (key != kC::ESC&&key!=kC::e);

}

}

} while (key != kC::ESC || key != kC::e);

}

void Control::mainMenuOfTheProgram() {

system("cls");

lang.outputTextFromFile(5893, 14);

int key;

key = \_getch();

switch (key) {

case kC::one:

menuForShelter();

break;

case kC::two:

menuForShopRobots();

break;

case kC::three:

menuForBuyer();

break;

case kC::four:

lang.fileSelectionToSave();

break;

case kC::five:

lang.fileSelectionForDownload();

break;

case kC::TAB:

lang.switchLanguage();

mainMenuOfTheProgram();

break;

case kC::ESC:

lang.titlePage();

break;

case kC::e:

lang.exitFromTheProgram();

break;

default:

mainMenuOfTheProgram();

break;

}

}

# Приложение Е

Файлы исходных данных

E.1 Файл configuration.txt

russian.txt

english.txt

save.txt

E.2 Файл russian.txt

Донецкий Национальный Технический Университе |

Факультет КНТ |

Кафедра ПИ |

|

|

Курсовой проект |

по дицсциплине: Объектно ориентированное программирование |

на тему: Животный приют и магазин роботов |

|

|

|

Выполнил: Проверили: |

ст. гр. ПИ-18а Щедрин С. В.|

Арефьев В. И. Ногтев Е. А.|

|

|

|

Донецк 2019г. |

|

|

Выход из программы: "ESC" Сменить язык: "TAB"|

Начать работу: "ENTER" О программе: "s"|

Помощь: "a" Об авторе: "d"|

=====================================================================

Об авторе |

|

Автором данной программы является студент 1го курса |

Донецкого Национального Технического Университета |

Факультета Компьютерных Наук и Технологий |

группы ПИ-18а |

|

+============================+ |

| Арефьев Владислав Игоревич | |

+============================+ |

ESC-назад e-выход TAB-сменить язык |

=====================================================================

Помощь |

Для работы с программой выберите пункт меню. Для этого нужно нажать |

соотв. клавишу. |

|

Основные из них: |

Начать работу с программой: "ENTER" |

Завершить работу с программой/вернутся назад: "ESC" |

Сменить язык: "Tab" (нижний регистр) |

Узнать подробнее о программе: "s" (нижний регистр) |

Узнать подронее об авторе: "d" (нижний регистр) |

Выйти из программы "e" (нижний регистр) |

|

|

=====================================================================

О программе |

|

Данная программа является виртуальным аналогом животного приюта и |

магазина роботетхники. С помощью данного приложения вы можете прос- |

мотреть список животных в приюте и роботов в магазине. Вы можете |

покормить животное, забрать его или отдать в приют.Также вы можете |

приобрести робота или роботизированое животное в соседней лавке. |

|

|

Животные имеют такие характеристики: |

1)Кличка |

2)Пол |

3)Цвет |

4)Возраст |

5)Порода |

6)Вид |

7)Масса |

|

|

Роботы имеют такие характеристики: |

1)Серия |

2)Масса |

3)Дата изготовления |

4)Модель |

|

|

Также могут быть и роботозированые животные. Они имеют общие харак- |

с обычными животными и роботами |

Программа позволяет быстро и легко найти будущего члена семьи. |

|

|

ESC-назад e-выход TAB-сменить язык |

=====================================================================

Меню |

|

|

1.Приют |

2.Магазин робототехники |

3.Покупатель |

4.Сохранить текущую сессию. |

5.Загрузить сессию |

|

|

|

|

ESC-назад e-выход TAB-сменить язык |

=====================================================================

Приют |

|

|

1.Забрать животное с приюта |

2.Отдать животное в приют |

3.Посмототреть весь приют |

4.Изменить массу животного |

5.Вывести инфорамцию о приюте |

|

|

|

|

ESC-назад e-выход TAB-сменить язык |

=====================================================================

Магазин |

|

|

1.Показать модель робота и ее цену |

2.Найти определенную модель |

3.Купить робота |

|

|

|

|

|

|

ESC-назад e-выход TAB-сменить язык |

=====================================================================

|

ESC-назад e-выход TAB-сменить язык |

|

=====================================================================

Приют |

|

Выбирите какое животное хотите отдать в приют: |

|

1.Кошка |

2.Собака |

3.Робопес |

Приют |

|

Выбирите какое животное хотите забрать из приюта: |

|

1.Обычное |

2.Робототизированое |

|

Введите кличку, пол и породу животного |

Введите кличку: |

Введите пол: |

Введите породу: |

|

Введите кличку, пол и породу животного и модель животного |

Введите модель: |

|

1.Покормить животное |

2.Уменьшить массу на 0,1 |

3.Кардинально изменить массу |

|

Модель: |

Цена: |

|

Введите модель которую хотите найти |

|

Введите модель: |

Введите дату изготовления: |

Введите серию: |

|

Вы успешно приобрели робота! |

|

У вас недостаточно денег! |

|

Такого робота нет в магазине! |

|

Введите ваш денежный запас: |

|

Приют пуст! |

|

Такого животного нет в приюте! |

|

Данное животное роботизированно, его масса неизменна! |

Введите массу: |

Введите стоимость: |

Введите процент роботизированности: |

Введите кличку собаки: |

Введите породу собаки: |

Введите вид собаки: |

|

Введите кличку кошки: |

Введите породу кошки: |

Введите пол: |

Введите цвет: |

Введите возраст: |

|

Масса слишком мала! |

|

Масса успешно изменена! |

Вы ошиблись, повторите ввод! |

=====================================================================

| |

| Сохранить все объекты перед выходом:1-ДА/2-НЕТ |

| |

=====================================================================

|

1.Загрузить в файл из конфига |

2.Новый путь к файлу |

3.Отмена |

|

|

Неверный путь! |

|

Введите путь нового файла: |

|

1.Загрузить из файл в конфиге |

2.Новый путь к файлу |

3.Отмена |

|

|

Непредвиденный конец файла! |

|

Файл пуст! |

|

1)Кличка: |

2)Возраст: |

3)Пол: |

4)Цвет: |

5)Масса: |

6)Порода: |

7)Цена: |

8)Вид: |

|

1)Модель: |

2)Серия: |

3)Масса: |

4)Дата изготовления: |

5)Цена: |

7)Вид: |

8)Модель: |

9)Серия: |

10)Дата изготовления: |

11)Процент робототизированности |

12)Цена: |

|

Всего животных в приюте: |

Собак в приюте: |

Кошек в приюте: |

Робопсов в приюте: |

|

Животное успешно приобретено! |

|

Успешно загружено! |

|

Количество роботов в магазине: |

E.3 Файл english.txt

Donetsk National Technical University |

Faculty of CST |

Department of PI |

|

|

Course project |

in the discipline: Object-oriented programmingе |

on the theme: Animal shelter |

|

|

|

Finished: Checked: |

st. gr. PI-18а Shchedrin S. V.|

Arefiev V. I. Nogtev E. A.|

|

|

|

Donetsk 2019y. |

|

|

Exit program: "ESC" Change language: "TAB"|

To start: "ENTER" About the program: "s"|

Help: "a" About the author: "d"|

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

About the Author |

|

The author of this program is a student of the 1st year |

Donetsk National Technical University |

Faculty of Computer Science and Technology |

group Pi-18а |

|

+~~~~~~~~~~~~~~~~~~~~~~~~~~~~+ |

| Arefiev Vladislav Igorevich| |

+~~~~~~~~~~~~~~~~~~~~~~~~~~~~+ |

ESQ-back e-exit TAB-change lang. |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Help |

To work with the program, select the menu item. To do this, click |

acc. key |

                                                                    |

|

Start working with the program: "ENTER" |

End the program / go back: "ESC" |

Change language: "Tab" (lower case) |

Learn more about the program: "s" (lower case) |

Learn more about the author: "d" (lower case) |

Quit program "e" (lower case) |

|

|

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

About the program |

                                                                    |

This program is a virtual analogue of animal shelter and |

store robotics. With this application you can request |

Check out the list of animals in the shelter and the robots in the |

store. You can feed an animal, pick it up or give it to a shelter. |

store. You can purchase a robot or a robotic animal in a nearby shop|

                                                                    |

                                                                    |

Animals have the following characteristics: |

1) Nickname |

2) Gender |

3) Color |

4) Age |

5) Breed |

6) View |

7) Mass |

                                                                    |

                                                                    |

Robots have the following characteristics: |

1) Series |

2) Mass |

3) Date of manufacture |

4) Model |

                                                                    |

                                                                    |

There may also be robotic animals. They have common characteristics |

with ordinary animals and robots |

The program allows you to quickly and easily find a future family |

member.                                                             |

                                                                    |

 ESQ-back e-exit TAB-change lang.|

=====================================================================

Menu |

                                                                    |

                                                                    |

1.Pryut |

2. Shop robotics |

3. Buyer |

4. Save the current session. |

5. Download session |

                                                                    |

                                                                    |

                                                                    |

                                                                    |

 ESQ-back e-exit TAB-change lang.|

=====================================================================

                            Shelter |

                                                                    |

                                                                    |

1. Pick up animal from shelter |

2. To give the animal to the shelter |

3. View the entire shelter |

4. Change the mass of the animal |

5. Get information about the shelter |

                                                                    |

                                                                    |

                                                                    |

                                                                    |

 ESQ-back e-exit TAB-change lang.|

=====================================================================

                            Shop |

                                                                    |

                                                                    |

1. Show the model of the robot and its price |

2. Find a specific model |

3. Buy a robot |

                                                                    |

                                                                    |

                                                                    |

                                                                    |

                                                                    |

                                                                    |

 ESQ-back e-exit TAB-change lang.|

=====================================================================

|

 ESQ-back e-exit TAB-change lang.|

|

=====================================================================

Shelter |

                                                                    |

Choose which animal you want to give to the shelter: |

                                                                    |

1.Cat |

2. Dog |

3. Robot |

                        Shelter |

                                                                    |

Choose which animal you want to pick up from the shelter: |

                                                                    |

1. Usual |

2.Robotized |

                                                                    |

Enter the nickname, gender and breed of the animal |

Enter the nickname: |

Enter the floor: |

Enter the breed: |

                                                                    |

Enter the nickname, gender and breed of the animal and animal model |

Enter the model: |

                                                                    |

1. Feed the animal |

2. Reduce weight by 0.1 |

3.Cardinally change the mass |

                                                                    |

Model: |

Price: |

                                                                    |

Enter the model you want to find |

                                                                    |

Enter the model: |

Enter the production date: |

Enter the series: |

                                                                    |

You have successfully acquired a robot! |

                                                                    |

You do not have enough money! |

                                                                    |

Such a robot is not in the store! |

                                                                    |

Enter your cash reserve: |

                                                                    |

The shelter is empty! |

                                                                    |

This animal is not in the shelter! |

                                                                    |

This animal is robotic, its mass is unchanged! |

Enter weight: |

Enter the cost: |

                                                                    |

Enter the dog's name: |

Enter the breed of dog: |

Enter the type of dog: |

                                                                    |

Enter the name of the cat: |

Enter cat breed: |

Enter the floor: |

Enter color: |

Enter the age: |

                                                                    |

Mass is too small! |

                                                                    |

Mass successfully changed! |

You made a mistake, repeat your input! |

=====================================================================

| |

| Save all objects before quitting: 1-YES / 2-NO |

| |

=====================================================================

|

1. Download to file from config |

2.New file path |

3.Cancel |

                                                                    |

                                                                    |

  Wrong way! |

                                                                    |

  Enter the path of the new file: |

                                                                    |

1. Download from file in config |

2.New file path |

3.Cancel |

                                                                    |

                                                                    |

Unexpected end of file! |

                                                                    |

File is empty! |

                                                                    |

1) Nickname: |

2) Age: |

3) Gender: |

4) Color: |

5) Mass: |

6) Breed: |

7) Price: |

8) View: |

                                                                    |

1) Model: |

2) Series: |

3) Mass: |

4) Date of manufacture: |

5) Price: |

7) View: |

8) Model: |

9) Series: |

10) Date of manufacture: |

11) Percentage of robots |

12) Price: |

                                                                    |

Total animals in the shelter: |

Dogs in the shelter: |

Cats in a shelter: |

Robodog in the shelter: |

|

Animal successfully purchased! |

                                                                    |

Successfully uploaded! |

|

Number of robots in the store: |

# Приложение Ж

Файлы выходных данных

Shelter

Cat

Мурка

10

female

white

3.2

Сибирская

102

Cat

Муся

8

male

черный

4.1

Сиамская

120

Cat

Мурка

5

female

белый

2.8

Персидская

122

Dog

Пышка

7

female

серый

5.8

Овчарка

бойцовская

132

Dog

Вальмир

2

male

рыжий

6.8

Сибу-ину

охотничья

188

RoboDog

Петро

3

male

черный

21.8

Сибу-ину

бойцовская

Mark-101

1888

20.01

20

1455

RoboDog

Сева

2

female

рыжий

25.4

Сибу-ину

охотничья

Mark-101

2888

20.01

10

1855

RoboShop

Robot

Mark-43

12

25.4

12.04

150

Robot

Mark-45

3656

45.8

15.07

457

Robot

Mark-45

3684

46.8

20.07

480

Robot

Mark-52

3656

80.8

14.09

1200

Robot

Mark-53

7945

45.8

01.03

821

# Приложение К

Экранные формы

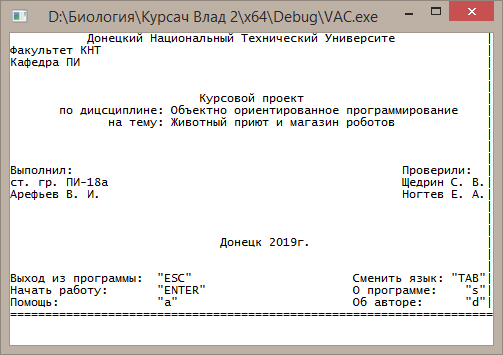


Рисунок К.1 ­– Заставка программы

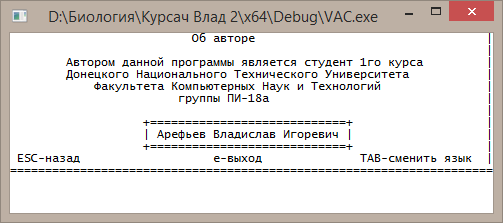


Рисунок К.2 ­– Об авторе

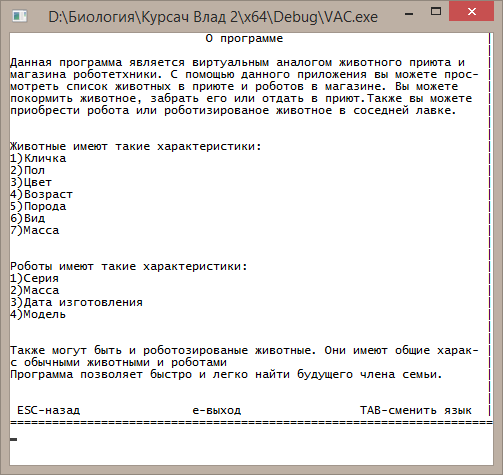


Рисунок К.3 – О программе

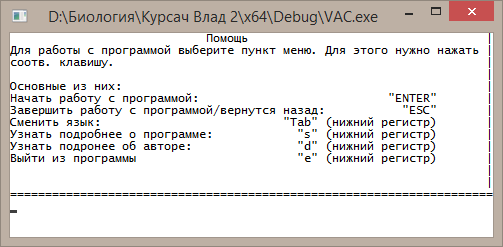


Рисунок К.4 ­– Помощь на 1м слайде

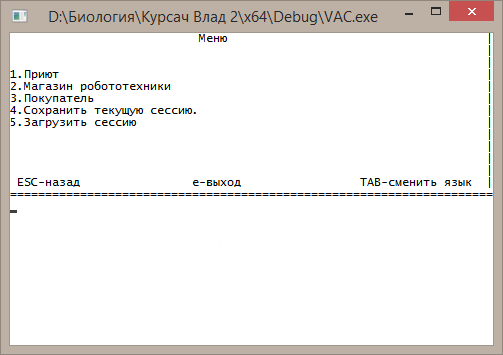


Рисунок К.5 – главное меню

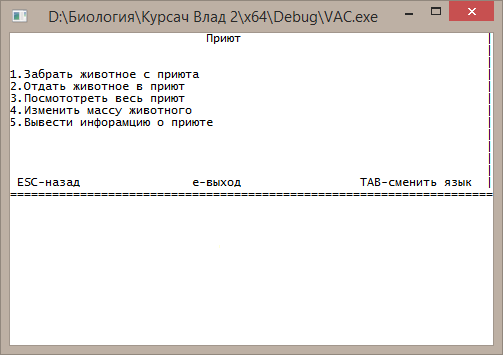


Рисунок К.6 –меню приюта

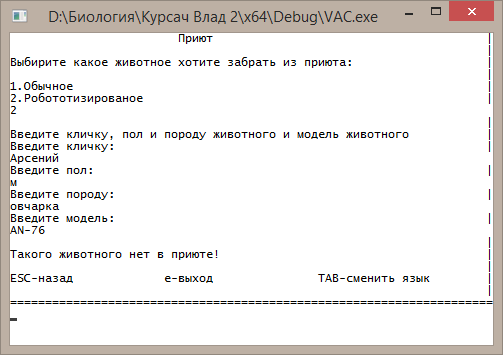


Рисунок К.7 –меню выбора вида животного, которого пользователь хочет забрать из приюта

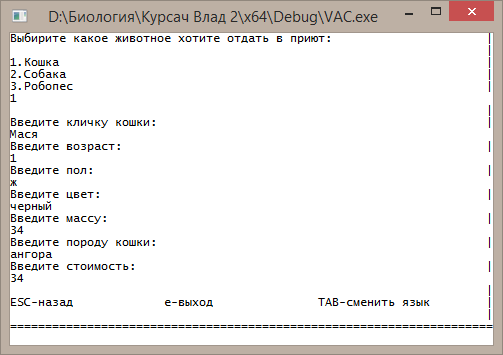


Рисунок К.8 –меню выбора вида животного, которого пользователь отдает в приют

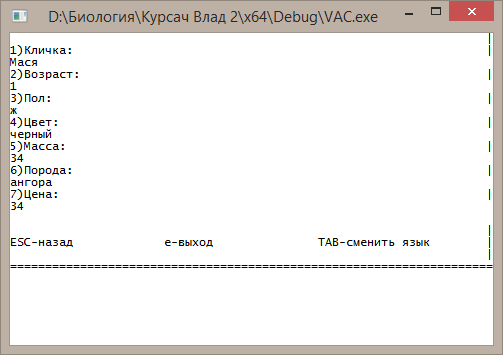


Рисунок К.9 –Просмотр всего приюта

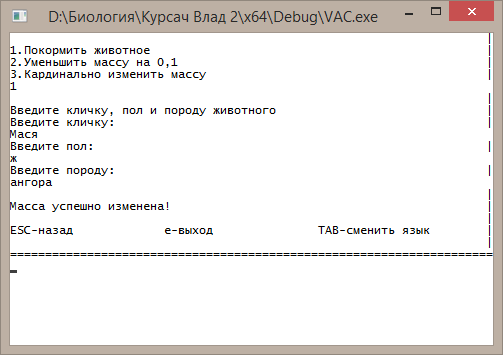


Рисунок К.10 –Изменение массы животного

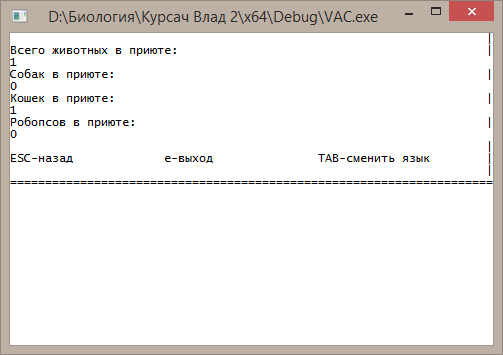


Рисунок К.11 –Подсчет объектов

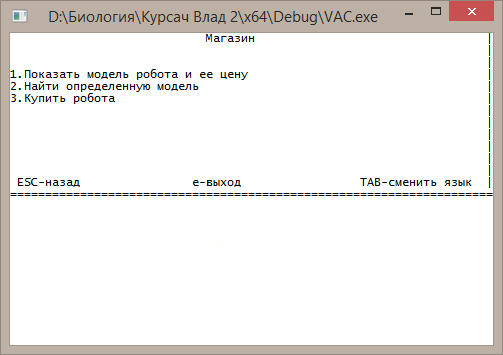


Рисунок К.12 –Меню магазина робототехники

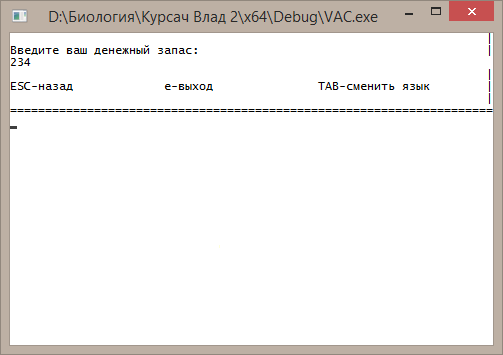


Рисунок К.13 –Меню покупателя

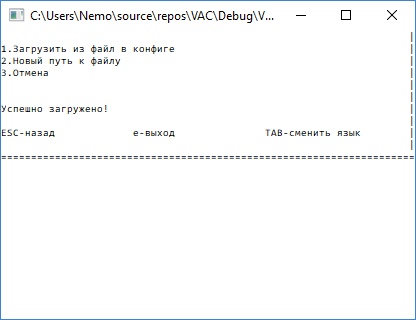


Рисунок К.14 – Загрузка объектов из файла