# АННОТАЦИЯ

Пояснительная записка содержит описание программы для работы с табличными данными, а именно, записями о животных, разработанной в рамках курсового проектирования, целью которого закрепление и углубление знаний в области основ программирования.

# СОДЕРЖАНИЕ

[**ВВЕДЕНИЕ 4**](#_Toc91462029)

[**1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ 6**](#_Toc91462030)

[**2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ 7**](#_Toc91462031)

[**2.1** **Постановка задачи на разработку программы** 7](#_Toc91462038)

[**2.2**  **Применяемые математические методы** 8](#_Toc91462042)

[**2.3 Описание и обоснование выбора метода организации входных, выходных и промежуточных данных** 8](#_Toc91462045)

[**2.4** **Обоснование выбора языка и среды программирования** 9](#_Toc91462046)

[**2.5** **Разработка модульной структуры программы** 9](#_Toc91462047)

[**2.6** **Внутренние структуры данных** 10](#_Toc91462048)

[**2.7** **Дополнительные константы, массивы и перечисления** 10](#_Toc91462049)

[**2.8** **Функции Queue.cpp (ядра программы)** 11](#_Toc91462050)

[**2.9** **Функции Change.cpp** 12](#_Toc91462051)

[**2.10** **Функции Fileworks.cpp** 12](#_Toc91462052)

[**2.11** **Функции Input.cpp** 12](#_Toc91462053)

[**2.12** **Функции Menu.cpp** 13](#_Toc91462063)

[**2.13** **Функции Output.cpp** 14](#_Toc91462064)

[**2.14** **Функции Processing.cpp** 14](#_Toc91462065)

[**2.15** **Функции Search.cpp** 14](#_Toc91462066)

[**2.16** **Функции Sort.cpp** 15](#_Toc91462067)

[**2.17** **Описание алгоритмов программы** 16](#_Toc91462068)

[**2.18**  **Обоснование состава технических и программных средств** 32](#_Toc91462071)

[**3 ВЫПОЛНЕНИЕ ПРОГРАММЫ 34**](#_Toc91462072)

[**3.1** **Условия выполнения программы** 34](#_Toc91462089)

[**3.2** **Загрузка и запуск программы** 34](#_Toc91462090)

[**3.3** **Проверка работоспособности программы** 44](#_Toc91462091)

[**ВЫВОДЫ 50**](#_Toc91462092)

[**ПЕРЕЧЕНЬ ССЫЛОК 52**](#_Toc91462093)

[**ПРИЛОЖЕНИЕ А 54**](#_Toc91462094)

# ВВЕДЕНИЕ

В рамках настоящего курсового проектирования ведется разработка программы по теме «Программа обработки данных о потреблении продуктов животными зоопарка» на основании документа – техническое задание – и в рамках организации – Севастопольский государственный университет. Дата выдачи задания: 07.09.2021.

С появлением и широким распространением табличных процессоров, одним из самых известных представителей которых является Microsoft Excel, и иных программ, позволяющих обрабатывать большие объёмы табличных данных без необходимости знания пользователем языков программирования, написание узкоспециализиро-ванных программ, примером которых является разрабатываемая программа, утратило свою актуальность.

Целью курсового проектирования является систематизация, закрепление и углубление знаний в области основ программирования и совершенствование практических навыков разработки программ на языке С/C++ на примере разработки «программы обработки данных о потреблении продуктов животными зоопарка», представляющей собой упрощённое подобие базы данных и позволяющей выполнять различные операции над записями, в том числе осуществление выборки и подсчета задолжников. Для достижения цели на разных этапах курсового проектирования должны быть решены следующие задачи:

* выбор варианта задания и детализация поставки задачи;
* определение требований к функциям, выполняемых разрабатываемой программой;
* выбор типов и проектирование структур данных, определяющих способы представления, хранения и преобразования входных, выходных и промежуточных данных;
* разработка модульной структуры программы, определение функций модулей и способов их взаимодействия;
* написание текста программных модулей на алгоритмическом языке;
* разработка тестовых примеров;
* тестирование и отладка программы;
* разработка программных документов в соответствии с действующими стандартами.

# 1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ

Назначение программы – работа с записями, представляющих информацию о животных (Номер вольера, кличка, порода, ареал обитания и т.д.). Позволяет выполнять следующие действия со списком записей – редактирование, просмотр, сохранение в файлы, считывание из файлов, сортировка, поиск и т.п. Приложение применяется для организации работы зоопарка.

Приложение будет полезным в зоологических парках, зверинцах и заповедниках. Проект позволит работникам производить необходимые расчеты в разы быстрее, что облегчит им работу.

Также приложение подойдет для работы в ветеринарных клиниках и других местах, где есть работа с животными.

# 2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ



### 2.1 Постановка задачи на разработку программы

В ходе выполнения курсового проекта был выполнен 13 вариант задания.

Входной файл имеет следующую структуру:

* номер вольера;
* кличка животного;
* порода животного;
* ареал обитания;
* тип потребляемых продуктов;
* вес потребляемых продуктов;
* стоимость потребляемых продуктов;
* дата.

Программа должна предоставлять меню-ориентированный интерфейс, позволяющий выполнять следующий минимально необходимый набор действий:

* создание и добавление элементов в таблицу;
* просмотр существующих записей с функцией скроллинга;
* поиск элемента по выбранному полю;
* удаление записи по ключевому полю;
* редактирование записи;
* сортировка записей в таблице по выбранному полю;
* сохранение данных в бинарный/текстовый файл с заданным именем файла (по выбору пользователя);
* загрузка данных из бинарного/текстового файла с заданным именем файла (по выбору пользователя);
* корректное завершение работы программы.

В случае обработки записей по варианту программа должна вывести суммарный вес и стоимость потребляемых продуктов всеми видами пород животных за указанный период.



### 2.2 Применяемые математические методы

В ходе разработки применялись следующие функции:

1. Получение суммарного веса всех потребляемых продуктов.

float count\_weight(zoo\* beg\_p);

1. Получение суммарной стоимости всех продуктов.

float count\_money(zoo\* beg\_p);



### 2.3 Описание и обоснование выбора метода организации входных, выходных и промежуточных данных

Для хранения данных в оперативной памяти было принято решение использовать двунаправленный список, так как двунаправленный список позволяет ускорить поиск требуемых элементов и постраничный вывод данных на экран, а также упрощает алгоритмы взаимодействия со списком.

Для хранения символьных данных о пользователе было решено использовать char в связи с богатым спектром методов взаимодействия с ним. Потенциально это так же могло позволить сократить размер бинарного файла, если бы в него записывалась только минимально необходимая часть массива, отведённого под строку, но было решено не делать размер записи в файле варьируемым, чтобы не усложнять процесс чтения, и чтобы можно было легко сказать, сколько записей хранится в файле по его размеру и выделить необходимое количество памяти при запуске программы.

### 2.4 Обоснование выбора языка и среды программирования

Исходный код программы был написан на языке программирования C++. Причина выбора:

* скорость выполнения функций;
* эффективное потребление памяти;

для разработки программы была создана следующая среда разработки:

* компилятор: Microsoft Visual Studio 2022;

основные причины выбора: быстрая компиляция программы; легкие и эффективные методы отслеживания ошибок в коде;

* текстовый редактор: Microsoft Visual Studio 2022;

основные причины выбора:

богатый спектр возможностей редактирования кода; наличие макросов и удобный интерфейс

* система сборки проекта: Microsoft Visual Studio 2022.

Причина использования: простой интерфейс для сборки проекта проекта, состоящего из множества файлов.

### 2.5 Разработка модульной структуры программы

В основу организации программы был положен принцип событийного управления. До начала разработки было принято решение разделить программу на 20 частей:

- хеадер *zoo.h* – содержит основные структуры организации списка записей, а также необходимые константы;

- главный файл *main.cpp* – содержит основной сценарий выполнения программы, обработку главного меню*;*

- хеадер *Menu.h* – содержит объявление функций меню;

- хендлер *Menu.cpp* – содержит функции меню;

- хеадер *Input.h* – содержит объявление функций ввода;

- хендлер *Input.cpp* – содержит функции ввода;

- хеадер *Output.h* – содержит объявление функций вывода;

- хендлер *Output.cpp* – содержит функции вывода;

- хеадер *Fileworks.h* – содержит объявление функций работы с файлами;

- хендлер *Fileworks.cpp* – содержит функции ввода;

- хеадер *Queue.h* – содержит объявление функций ядра;

- ядро *Queue.cpp* – содержит функции работы со списком.

- хеадер *Change.h* – содержит объявление функций корректировки;

- хендлер *Change.cpp* – содержит функции корректировки;

- хеадер *Search.h* – содержит объявление функций поиска;

- хендлер *Search.cpp* – содержит функции поиска;

- хеадер *Sort.h* – содержит объявление функций сортировки;

- хендлер *Sort.cpp* – содержит функции сортировки;

- хеадер *Processing.h* – содержит объявление функций обработки списка;

- хендлер *Processing.cpp* – содержит функции обработки списка.

### 2.6 Внутренние структуры данных

Структура данных о животном:

struct zoo //структура с информацией

{

unsigned int id,

cage;

char name[n\_n],

specie[n\_s],

area[n\_a],

p\_type[n\_p],

date[n\_d],

float p\_weight,

p\_money;

struct zoo\* next;

struct zoo\* prev;

};

### 2.7 Дополнительные константы, массивы и перечисления

Константы для выделения памяти.

const int n\_n = 16,

n\_s = 21,

n\_a = 21,

n\_p = 16,

n\_d = 11,

print = 13;

Коды специальных клавиш.

enum Keys {

NoPressed,

Enter = 13,//Клавиша "Enter"

Esc = 27,//Клавиша "ESC"

Up = 72,//Клавиша "Up"

Left = 75,//Клавиша "Left"

Right = 77,//Клавиша "Right"

Down = 80,//Клавиша "Down"

};

Коды цветов.

enum ConsoleColor {

Black = 0,

Blue = 1,

Green = 2,

Cyan = 3,

Red = 4,

Magenta = 5,

Brown = 6,

LightGray = 7,

DarkGray = 8,

LightBlue = 9,

LightGreen = 10,

LightCyan = 11,

LightRed = 12,

LightMagenta = 13,

Yellow = 14,

White = 15

};

### 2.8 Функции Queue.cpp (ядра программы)

Функции создания элемента списка. Функция add\_first отличается от стандартного add тем, что используется для добавления первого элемента. Обе функции используются при извлечении данных из файла

zoo\* add\_first(const zoo& z);

zoo\* add(zoo\* end, const zoo& z);

Функция удаления первого элемента текущего списка. При отсутствии элементов выводит сообщение об ошибке. При успешном удалении – возвращает новое начало списка.

zoo\* dellete\_id\_beg(zoo\* beg);

Функция удаления последнего элемента текущего списка. При отсутствии элементов выводит сообщение об ошибке. При успешном удалении – возвращает новый конец списка.

zoo\* dellete\_id\_end(zoo\* beg, zoo\* end);

Функция безопасного удаления элемента из списка.

zoo\* dellete\_id(zoo\* beg, zoo\* end, int num);

Функция корректного удаления всего списка.

void dellete\_all(zoo\* beg);

### 2.9 Функции Change.cpp

Функция отрисовки меню корректировки и корректировки элемента списка. Редактирует элемент по введённому id.

void change(zoo\* beg, zoo\* end, int id\_num);

### 2.10 Функции Fileworks.cpp

Функция считывания информации из файла с именем filename.

int read\_file(string filename, zoo\*\* beg, zoo\*\* end);

Функция записи информации в файл с именем filename.

int write\_in\_file(string filename, zoo\* temp);

### 2.11 Функции Input.cpp

Функция ввода и проверки данных типа int.

int inp\_int(int n, const char\* inf);

Функция ввода даты.

char\* inp\_date(char\* date);

Функция проверки даты. Если данные не проходят проверку – функция возвращает false.

bool check\_date(char\* date);

Функция ввода и проверки данных типа float.

float inp\_float(float temp, const char\* inf);

Функция ввода данных типа char.

char\* inp\_char(char\* temp, const char\* inf, int size);

Функция ввода и проверки имени файла.

string inp\_filename(string temp, const char\* inf);

Функция включения и отключения мигающей каретки в консоли.

void cursor\_off\_on(bool inf);

Функция проверки данных типа char. Если данные не проходят проверку – функция возвращает false.

bool check\_char(char\* temp);

Функция удаления лишних пробелов из массива типа char.

char\* delete\_space(char\* temp, int size);

Функция ввода полей элемента.

zoo input();



### 2.12 Функции Menu.cpp

Функции отрисовки меню и взаимодействия с ним. Функция menu\_category отличается от стандартной menu тем, что может вернуть пользователя в главное меню программы при нажатии ESC.

int menu(string menuElems[], int size, string inf);

int menu\_category(string menuElems[], int size, string inf, jmp\_buf env);

Функция отрисовки стартового меню и взаимодействия с ним.

int start\_menu();

Функция перемещения позиции курсора в консоли. Нужна для отрисовки некоторых элементов.

void goto\_x\_y(short x, short y);

### 2.13 Функции Output.cpp

Функции отрисовки заголовка таблицы. Функция header\_processing отличается от стандартного header тем, что используется только при выводе результатов обработки.

void header();

void header\_proccessing();

Функции вывода элемента на экран. Функция print\_processing отличается от стандартного print\_on\_the\_screen тем, что используется только при выводе элементов обработки.

void print\_on\_the\_screen(const zoo& z);

void print\_processing(const zoo& z);

Функции вывода списка на экран. Функция show\_processing отличается от стандартного see\_the\_information тем, что используется только при выводе списка обработки.

void see\_the\_information(zoo\* beg, zoo\* end);

void show\_processing(zoo\* beg);

### 2.14 Функции Processing.cpp

Функции обработки списка. Объединяет остальные функции, необходимые для обработки.

void processing(zoo\* beg);

Функции подсчёта суммарного веса и суммы денег.

float count\_money(zoo\* beg\_p);

float count\_weight(zoo\* beg\_p);

Функция записи в файл с именем filename результатов обработки.

int processing\_write\_in\_file(string filename, zoo\* temp);

### 2.15 Функции Search.cpp

Функция отрисовки и взаимодействия с меню поиска.

void search(zoo\* beg,jmp\_buf env);

Функция поиска по номеру вольера.

void search\_cage(zoo\* beg, int num\_cage);

Функция поиска по кличке животного.

void search\_name(zoo\* beg, char\* pet\_name)

Функция поиска по породе животного.

void search\_specie(zoo\* beg, char\* pet\_specie);

Функция поиска по ареалу обитания.

void search\_area(zoo\* beg, char\* pet\_are);

Функция поиска по типу продуктов.

void search\_p\_type(zoo\* beg, char\* p\_type);

Функция поиска весу продуктов.

void search\_p\_weight(zoo\* beg, float p\_weight);

Функция поиска по сумме денег потраченных на продукты.

void search\_p\_money(zoo\* beg, float sum\_money);

Функция поиска по дате.

void search\_date(zoo\* beg, char\* data);

Функция поиска по id. Необходима для функций удаления и корректировки.

void search\_id(zoo\* beg, int id);

### 2.16 Функции Sort.cpp

Функция отрисовки и взаимодействия с меню сортировки. Также передаёт остальным функциям сортировать функцию по убыванию или возрастанию.

void sort\_by\_field(zoo\* beg,jmp\_buf env);

Функция сортировки по номеру вольера.

void sort\_cage(zoo\* beg, int how\_sort);

Функция сортировки по кличке животного.

void sort\_name(zoo\* beg, int how\_sort);

Функция сортировки по породе животного.

void sort\_specie(zoo\* beg, int how\_sort);

Функция сортировки по ареалу обитания животного.

void sort\_area(zoo\* beg, int how\_sort);

Функция сортировки по типу продуктов.

void sort\_p\_type(zoo\* beg, int how\_sort);

Функция сортировки по весу продуктов.

void sort\_p\_weight(zoo\* beg, int how\_sort);

Функция сортировки по сумме денег потраченных на продукты.

void sort\_p\_money(zoo\* beg, int how\_sort);

Функция сортировки по номеру дате.

void sort\_date(zoo\* beg, int how\_sort);

Функция перестановки местами элементов при сортировке.

void sort\_replace(zoo\* left, zoo\* right, zoo\* temp);

### 2.17 Описание алгоритмов программы

Структурная схема main приведена на рисунке 2.1.

Описание блоков main:

* блок 1 – инициализация переменных. получение дескриптора консоли;
* блок 2 – вывод и работа со стартовым меню программы с помощью функции start\_menu;
* блок 3 – создание нового списка или использование готового списка из файла;
* блок 4 – вечный цикл – while(1);
* блок 5 – вывод и работа с главным меню программы с помощью функции menu в swith-case конструкции;
* блок 6 – создание нового списка;
* блок 7 – вызов функции add или add\_first для добавления нового элемента в список;
* блок 8 – вызов функций удаления для удаления элементов из списка;
* блок 9 – вызов функции Change для корректировки элемента;
* блок 10 – вызов функции Sort\_by\_field для сортировки списка;
* блок 11 – вызов функции Search для поиска элемента в списке;



Рисунок 2.1 – Структурная схема main

* блок 12 – вызов функции see\_the\_information для отображения списка;
* блок 13 – вызов функции write\_in\_file для сохранения списка в файл;
* блок 14 – вызов функции read\_file для чтения списка из файла;
* блок 15 – вызов функции Processing для обработки списка;
* блок 16 – выход из программы.

Структурная схема схема add приведена на рисунке 2.2.



Рисунок 2.2 – Структурная схема add

Описание блоков add:

* блок 1 – инициализация элемента;
* блок 2 – присваивание элементу последней позиции в списке;
* блок 3 – присваивание элемменту id – на 1 больше , чем у прошлого элемента.

На рисунке 2.3 представлена структурная схема алгоритма dellete\_all.

Описание блоков dellete\_all:

* блок 1 – инициализация переменной temp ;
* блок 2 – проверка на то , кончился список или нет. Если список кончился – прекратить выполнение функции;
* блок 3 – выполняется , если список не кончился. Инициализация переменной temp\_1;
* блок 4 – удаление переменной temp;
* блок 5 – temp\_1 – начало списка. Возвращение к блоку 2.

Рисунок 2.3 – Структурная схема dellete\_all

На рисунке 2.4 представлена структурная схема алгоритма Menu. Описание блоков Menu:

* блок 1 – инициализация переменнных;
* блок 2 – вечный цикл while(1);
* блок 3 – вывод меню на экран;
* блок 4 – цикл с переменной i (int i = 0; i < size / sizeof(menuElems[0]); i++);
* блок 5 – проверка равен ли pointer i. Если да выполняется 7 блок, если нет – выполняется 6 блок;
* блок 6 – вывод пункта меню;



Рисунок 2.4 – Структурная схема Menu

* блок 7 – вывод символа «🡪»;
* блок 8 – считывание нажатой клавиши;
* блок 9 – switch-case конструкция;
* блок 10 – увеличение pointer , если нажата стрелочка вверх;
* блок 11 – проверка достиг ли pointer конца меню. Если достиг – выполняется блок 12;
* блок 12 – pointer ставим в начало меню;
* блок 13 – уменьшение pointer , если нажата стрелочка вниз;
* блок 14 - проверка достиг ли pointer начала меню. Если достиг – выполняется блок 16;
* блок 15 – return pointer , если нажата клавиша Enter;
* блок 16 – pointer ставим в конец меню;
* блок 17 – в любом другом случае – просто игнорируем нажатие.

На рисунке 2.5 представлена структурная схема алгоритма Input.

Описание блоков Input:

* блок 1 – инициализация переменных;
* блок 2 – ввод номера вольера с помощью функции inp\_int;
* блок 3 – ввод клички животного с помощью функции inp\_char;
* блок 4 – ввод породы животного с помощью функции inp\_char;
* блок 5 – ввод ареала обитания животного с помощью функции inp\_char;
* блок 6 – ввод типа потребляемых продуктов с помощью функции inp\_char;
* блок 7 – ввод веса потребляемых продуктов с помощью функции inp\_float;
* блок 8 –ввод стоимости потребляемых продуктов с помощью функции inp\_float;
* блок 9 – ввод даты с помощью функции inp\_date.



Рисунок 2.5 – Структурная схема Input

На рисунке 2.6 представлена структурная схема алгоритма Search.

Описание блоков Search:

* блок 1 – проверка на существования элементов в списке. Если список существует – выполняется блок 2. Если нет – работа функции завершается;
* блок 2 – инициализация переменных;
* блок 3 – выбор пункта меню с помощью конструкции switch-case;
* блок 4 – поиск по номеру вольера с помощью функции search\_cage;
* блок 5 – поиск по кличке животного с помощью функции search\_name;
* блок 6 – поиск по породе животного с помощью функции search\_specie;
* блок 7 – поиск по ареалу обитания животного с помощью функции search\_area;



Рисунок 2.6 – Структурная схема Search

* блок 8 – поиск по типу продуктов с помощью функции search\_p\_type;
* блок 9 – поиск по весу потребляемых продуктов с помощью функции search\_p\_weight;
* блок 10 – поиск по стоимости потребляемых продуктов с помощью функции search\_p\_money;
* блок 11 – поиск по дате с помощью функции search\_date.

На рисунке 2.7 представлена структурная схема Sort\_by\_field.

Описание блоков Sort\_by\_field:

* блок 1 – проверка на существования элементов в списке. Если список существует – выполняется блок 2. Если нет – работа функции завершается;
* блок 2 – инициализация переменных;
* блок 3 – выбор пункта меню с помощью конструкции switch-case;
* блок 4 – сортировка по номеру вольера с помощью функции sort\_cage;
* блок 5 – сортировка по кличке животного с помощью функции sort\_name;
* блок 6 – сортировка по породе животного с помощью функции sort\_specie;
* блок 7 – сортировка по ареалу обитания животного с помощью функции sort\_area;
* блок 8 – сортировка по типу продуктов с помощью функции sort\_p\_type;
* блок 9 – сортировка по весу потребляемых продуктов с помощью функции sort\_p\_weight;
* блок 10 – сортировка по стоимости потребляемых продуктов с помощью функции sort\_p\_money;
* блок 11 – сортировка по дате с помощью функции sort\_date.



Рисунок 2.7 – Структурная схема Sort\_by\_field

На рисунке 2.8 представлена стркутурная схема Change

Описание блоков Change:

* блок 1 – проверка на существования элементов в списке. Если список существует – выполняется блок 2. Если нет – работа функции завершается;
* блок 2 – инициализация переменных;
* блок 3 – выбор пункта меню с помощью конструкции switch-case;
* блок 4 – корректировка номера вольера;
* блок 5 – корректировка клички животного;
* блок 6 – корректировка породы животного;
* блок 7 – корректировка ареала обитания животного;
* блок 8 – корректировка типа продуктов;
* блок 9 – корректировка веса потребляемых продуктов;
* блок 10 – корректировка стоимости потребляемых продуктов;
* блок 11 – корректировка даты.



Рисунок 2.8 – Структурная схема Change

На рисунке 2.9 представлена структурная схема Processing

Рисунок 2.9 – Структурная схема Processing

Описание блоков Processing:

* блок 1 – инициализация перменных;
* блок 2 – ввод даты начала и конца с помощью функции inp\_date;
* блок 3 – если дата конца раньше даты начала – перейти к блоку 2, иначе – перейти к блоку 4;
* блок 4 – Если список кончился – перейти к блоку 5. В противном случае – перейти к блоку 6;
* блок 5 – Если элементы были найдены – перейти к блоку 7. В другом случае – перейти к блоку 8;
* блок 6 – в случае если дата элемента находится в указанном диапазоне перейти к блоку 9. В противном случае – перейти к блоку 4;
* блок 7 – вывести полученные элементы , сохранить данные обработки в файл с помощью функции processing\_write\_in\_file и завершить работу функции;;
* блок 8 – вывести сообщение об отсутствии найденных элементов и завершить работу функции;
* блок 9 – если можно найти элемент с данной породой – перейти к блоку 10. Иначе – перейти к блоку 11;
* блок 10 –добавить к элементу вес и стоимость продуктов и пройти к блоку 4;
* блок 11 – добавление элемента в список обработки и перейти к блоку 4.

На рисунке 2.10 представлена структурная схема алгоритма read\_file.

Описание блоков read\_file:

* блок 1 – инициализация переменнных;
* блок 2 – проверка на содержание в имени файла «.data». Если содержится – перейти к блоку 4. Если нет – перейти к блоку 3;
* блок 3 – открыть файл как текстовый;
* блок 4 – открыть файл как бинарный;
* блок 5 – файл можно открыть ? Если да – перейти к блоку 6. В противном случае – перейти к блоку 7;
* блок 6 – считывание информации из файла;
* блок 7 – вывод сообщения об ошибке;
* блок 8 – добавление элемента в очередь;
* блок 9 – файл закончился ? Если да – завершить выполнение функции. Если нет – перейти к блоку 6.



Рисунок 2.10 – Структурная схема read\_file

На рисунке 2.11 представлена структруная схема write\_in\_file

Описание блоков write\_in\_file:

* блок 1 – инициализация переменнных;
* блок 2 – проверка на содержание в имени файла «.data». Если содержится – перейти к блоку 4. Если нет – перейти к блоку 3;
* блок 3 – сохранение в текстовый файл;
* блок 4 – сохранение в бинарный файл;
* блок 5 – файл можно открыть для записи ? Если да – перейти к блоку 6. В противном случае – перейти к блоку 7;
* блок 6 – считывание элементов в очереди;
* блок 7 – вывод сообщения об ошибке;
* блок 8 – запись информации в файл;
* блок 9 – список закончился ? Если да – завершить выполнение функции. Если нет – перейти к блоку 6.



Рисунок 2.11 – Структурная схема write\_in\_file

На рисунке 2.12 представлена структурная схема inp\_int.

Описание блоков inp\_int:

* блок 1 – bool int = true;



Рисунок 2.12 – Структурная схема inp\_int

* блок 2 – ввод переменной типа int;
* блок 3 – проверка на наличие ошибок в потоке. Если они есть – выполняется блок 4. Если нет – функция завершает своё выполнение;
* блок 4 – обнуление флагов потока и возвращение к блоку 1.

### 2.18 Обоснование состава технических и программных средств

Для работы программы рекомендуется использовать 64-разрядный компьютер архитектуры IBM PC с подключённым экраном, клавиатурой, под управлением ОС Windows 10 или старше или любой другой ОС с установленным Wine версии не менее 5.19. Компьютер должен соответствовать минимальным требованиям, необходимым для запуска упомянутых операционных систем, иметь 256 КБ дискового или иного пространства для хранения исполняемого файла и 7 МБ свободной оперативной памяти для запуска.

# ВЫПОЛНЕНИЕ ПРОГРАММЫ



### 3.1 Условия выполнения программы

Минимально необходимые системные требования представлены в таблице ниже (Таблица 3.1).

Таблица . – Системные требования

|  |  |
| --- | --- |
| Операционная система | Windows 10 x64 |
| Процессор | Одноядерный процессор, x86\_64, не менее 1ГГц |
| ОЗУ | Не менее 7 МБ |
| Место на внешнем носителе | Не менее 256 КБ |

### 3.2 Загрузка и запуск программы

Запустить программу можно двумя способами:

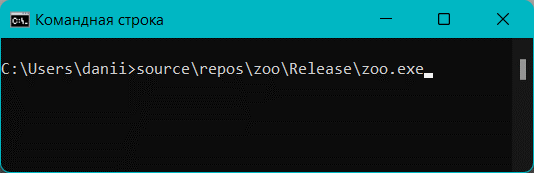
1. c помощью проводника;
2. c помощью командной строки, введя название программы, что представлено на рисунке 3.1.

Рисунок 3.1 – Запуск программы с помощью консоли

После запуска программы пользователю будет предложено выбрать один из пунктов стартового меню (рисунок 3.2). Навигация по меню осуществляется с помощью стрелочек. Подтверждение выбора производиться с помощью клавиши ENTER.

При выборе определенного пункта меню, приложение перейдет в один из режимов работы, представленных на рисунках 3.2 – 3.4.

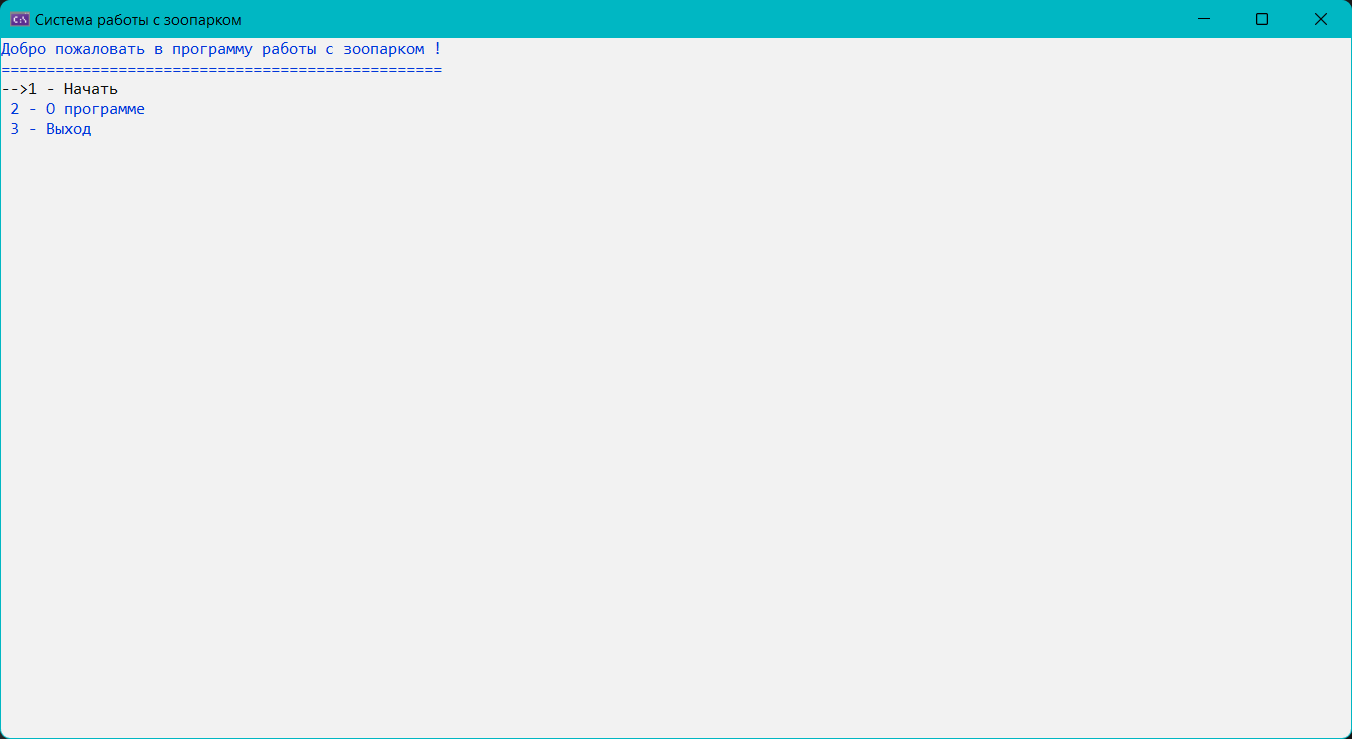


Рисунок 3.2 – Стартовое меню программы

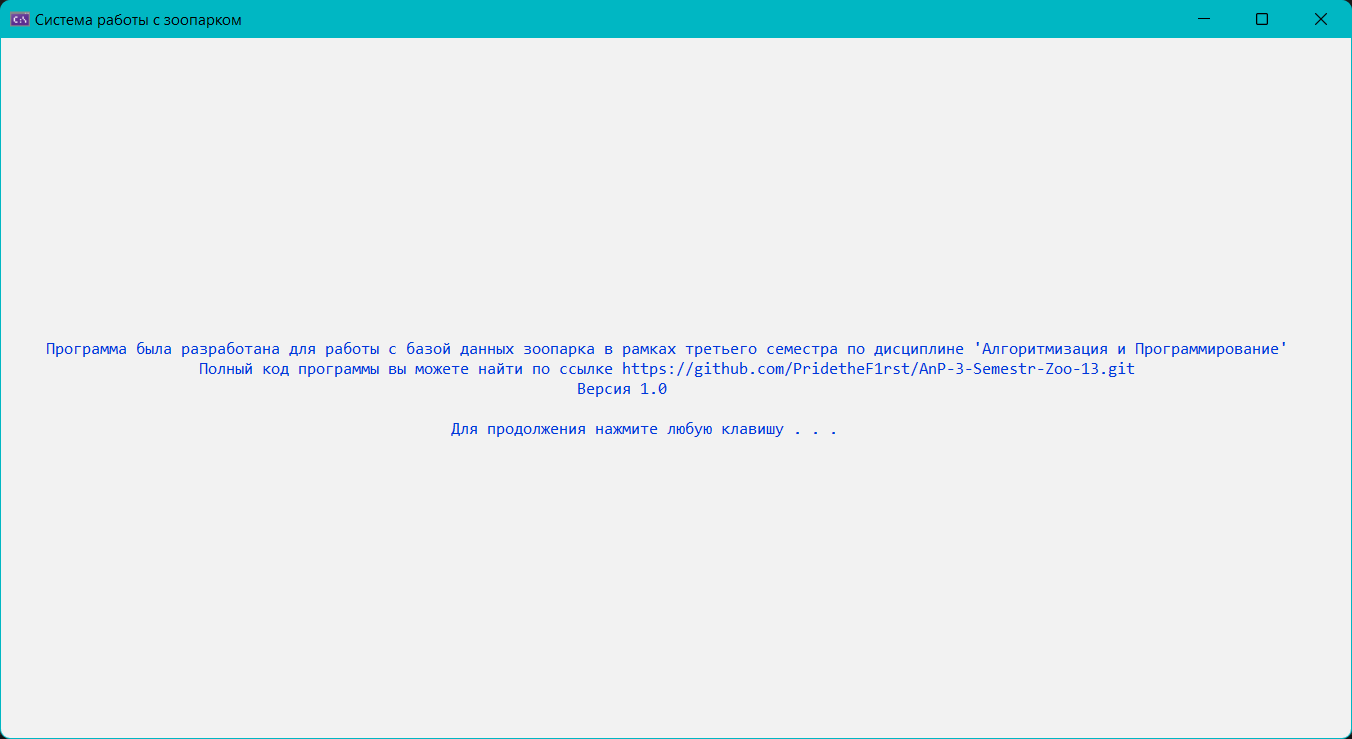


Рисунок 3.3 – Выбран пункт меню «О программе»

Для того , чтобы начать использовать программу – необходимо использовать существующий файл для импорта в программу или создать новый список. Это продемонстриванно на рисунке 3.4 .

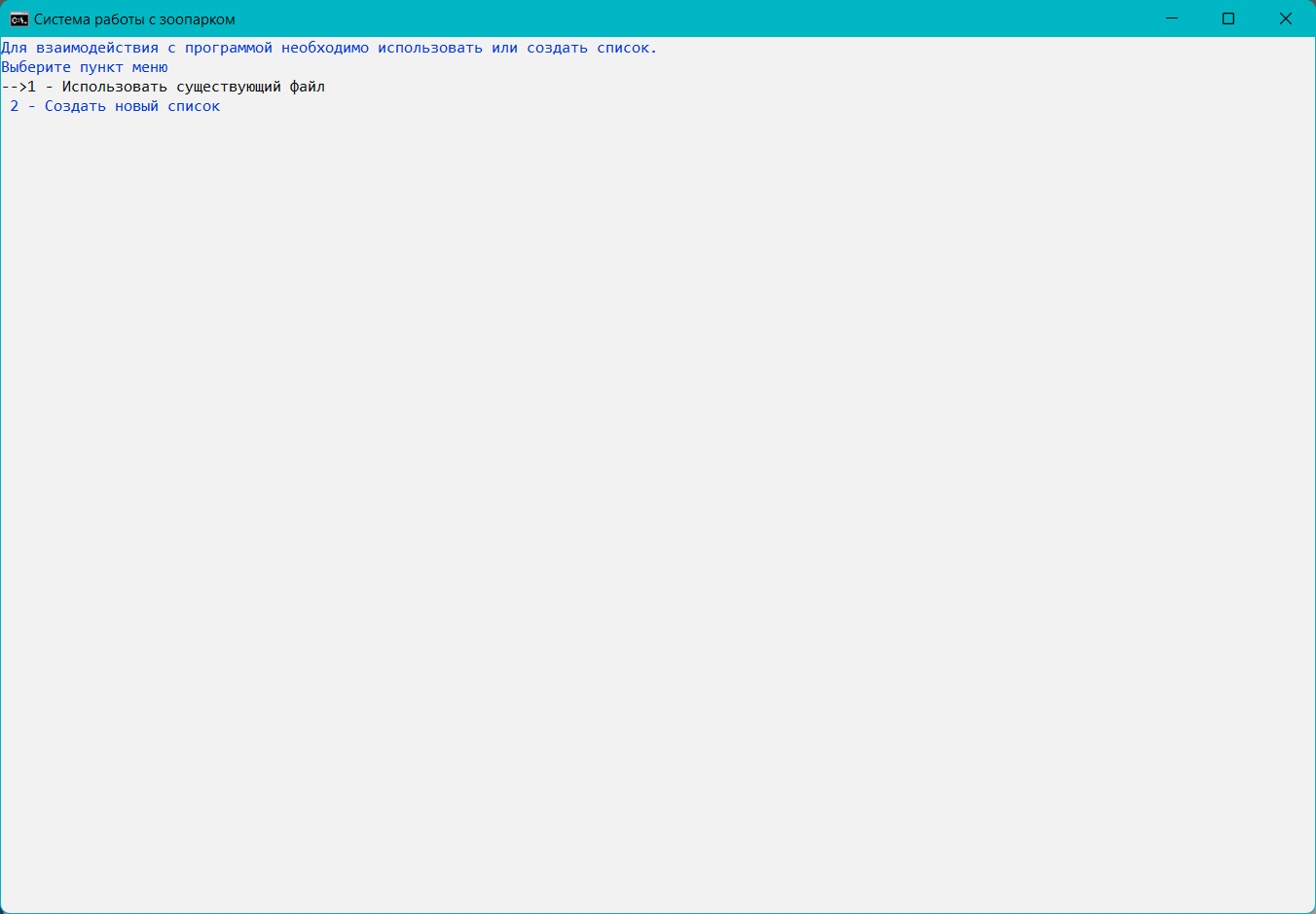


Рисунок 3.4 – Меню выбора создания или использования списка.

В случае выбора второго пункта – программа объявит о создании нового списка и направит пользователя в главное меню программы. В случае выбора первого пункта ­– программа попросит ввести имя файла , и определить расширение файла , что представленно на рисунках 3.5 и 3.6

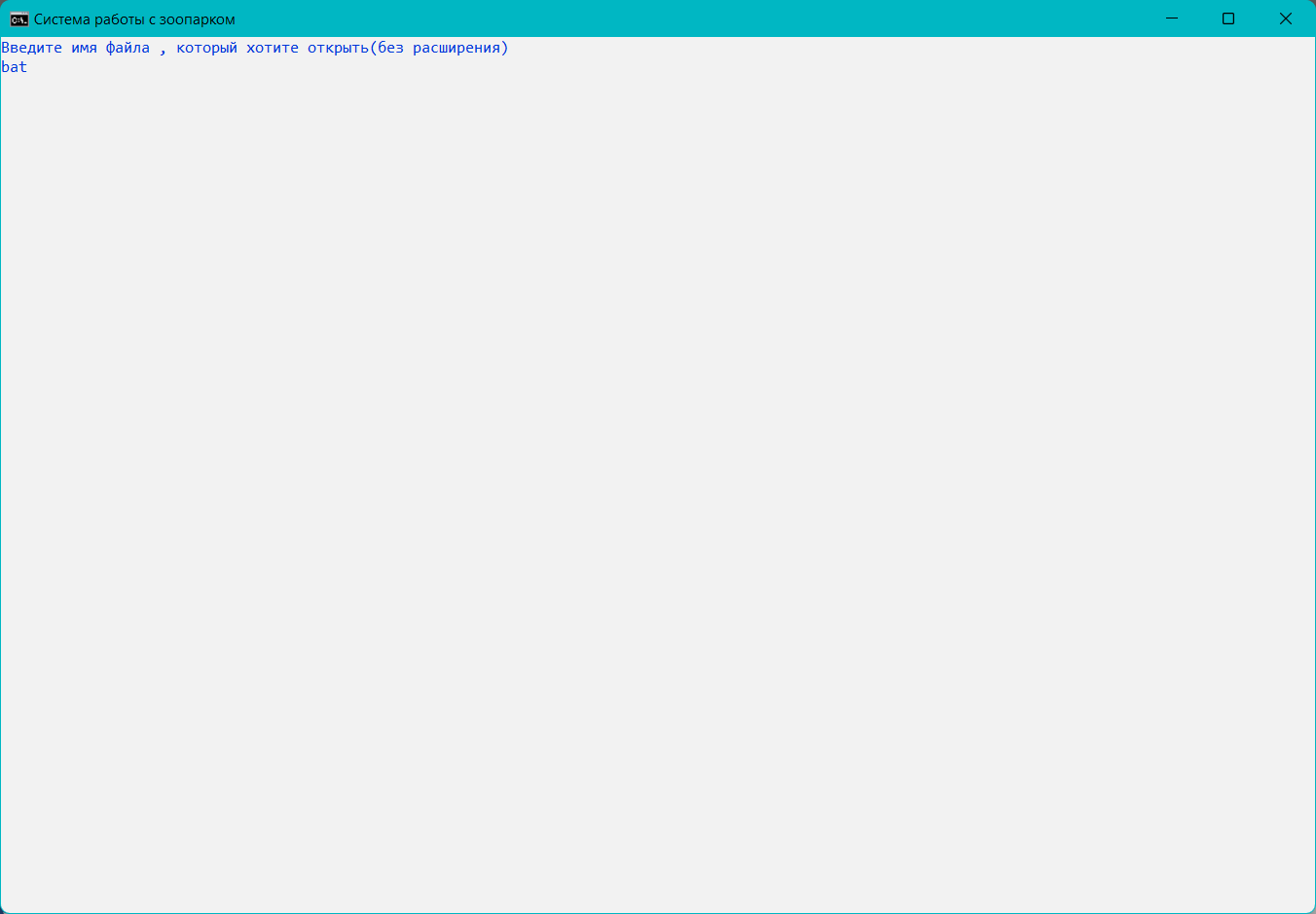


Рисунок 3.5 – Ввод имени файла

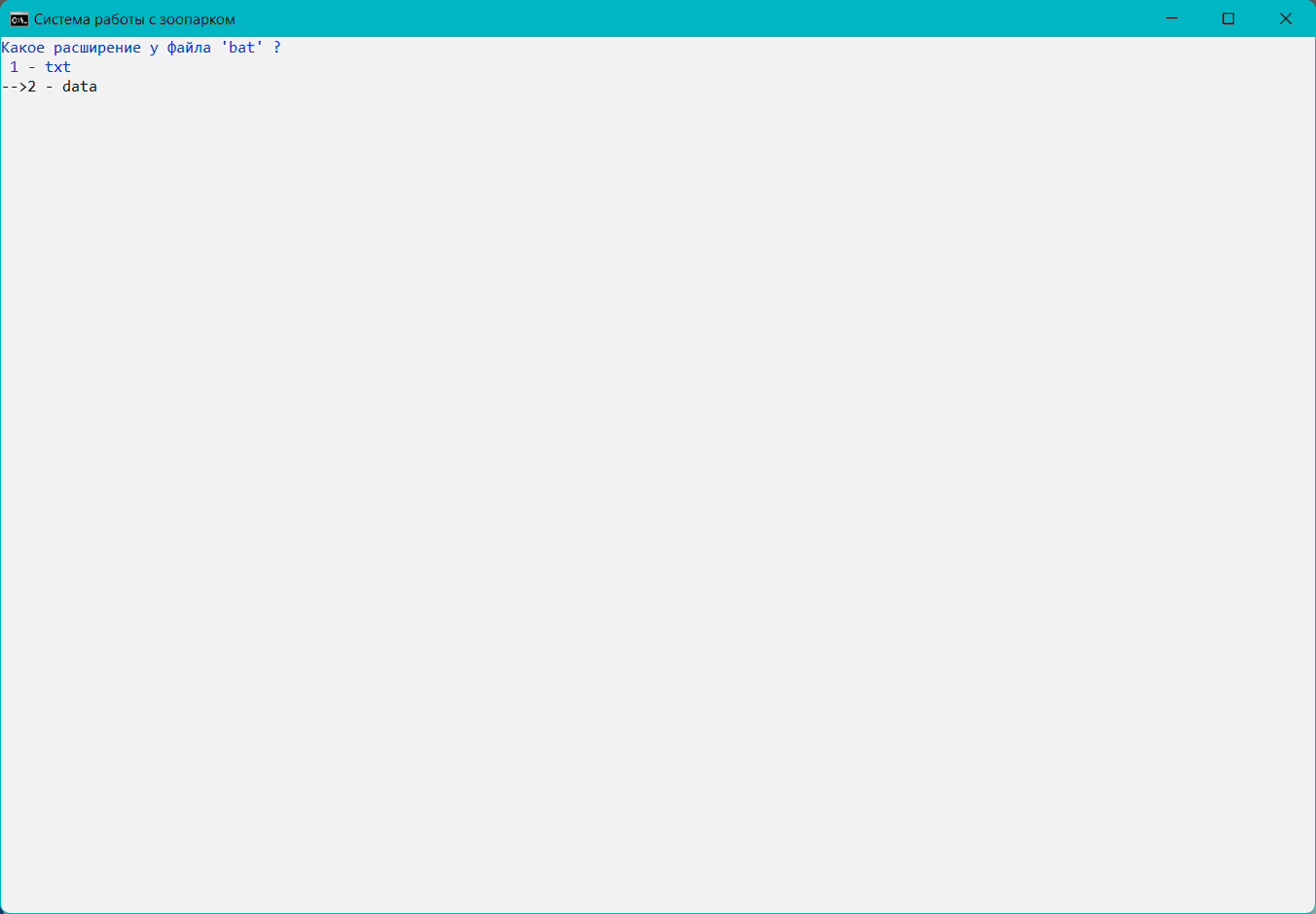


Рисунок 3.6 – Выбор расширения файла

В случае удачного считывания информации из файла пользователю будет выведено сообщение о количестве считанных записей , что продемонстриванно на рисунке 3.7

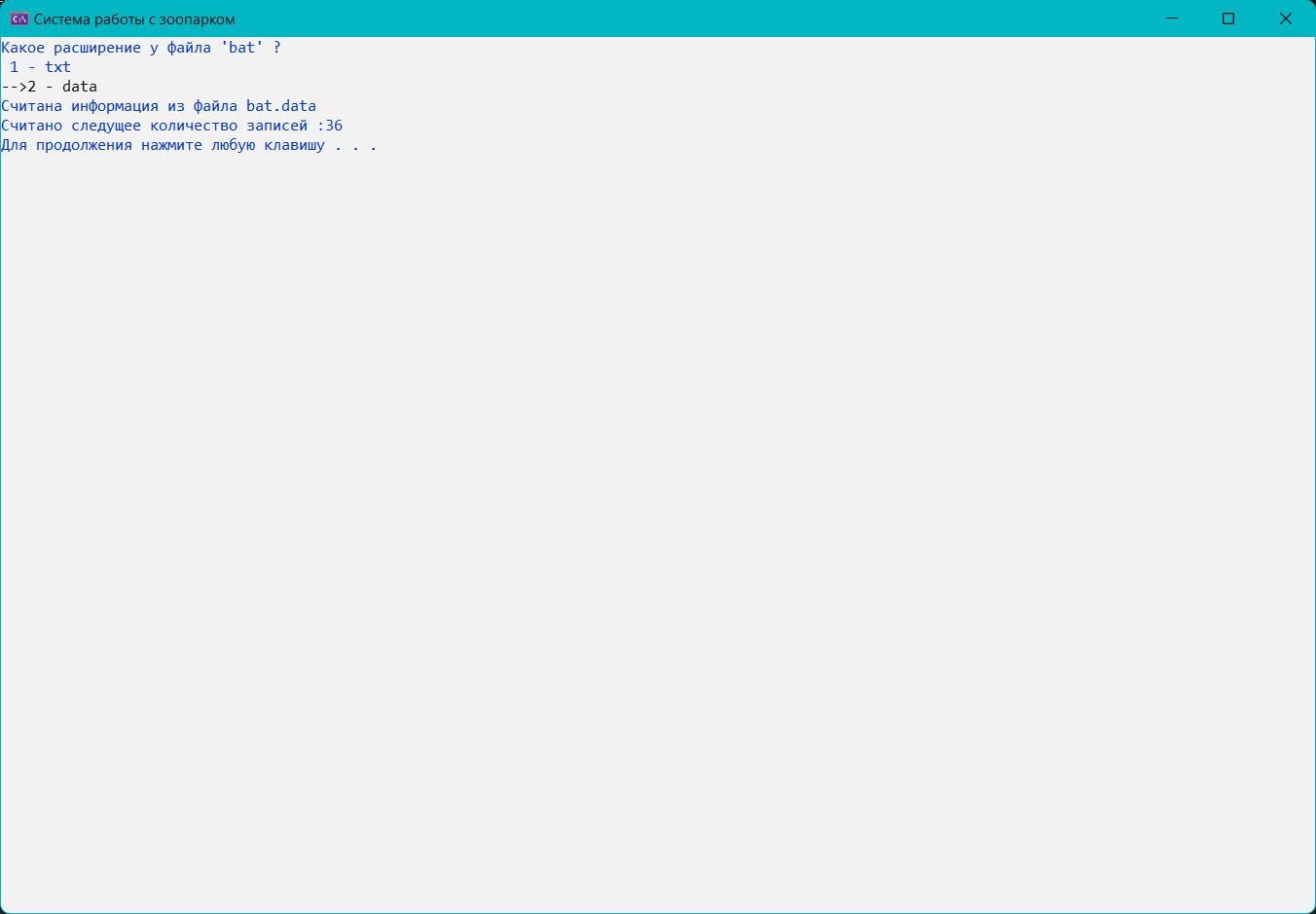


Рисунок 3.7 – Сообщение о успешно считанной информации

Далее пользователь попадает в главное меню программы. Навигация по меню осуществляется с помощью стрелочек. Подтверждение выбора производиться с помощью клавиши ENTER. Для выхода из программы/подпрограммы (где возможно) используется клавиша ESC

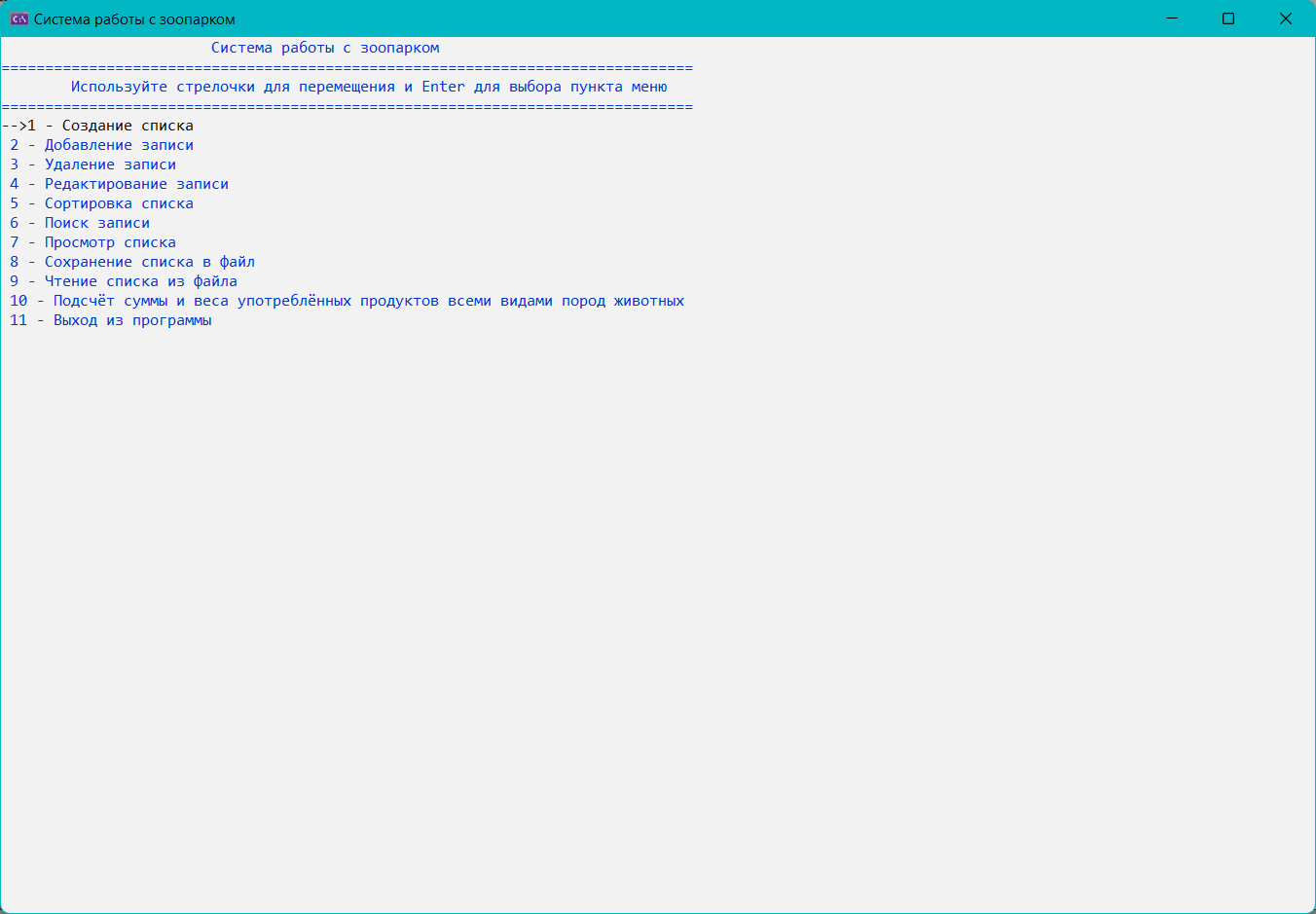


Рисунок 3.9 – Главное меню программы

На рисунке 3.10 и 3.11 представлено добавление элемента.

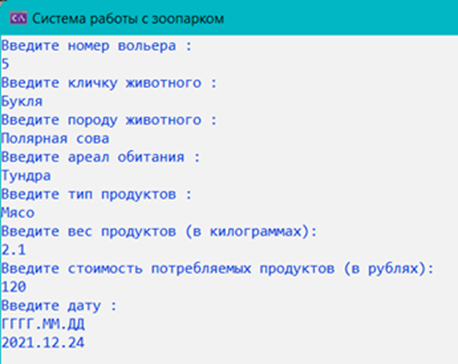


Рисунок 3.10 – Добавление элемента



Рисунок 3.11 – Вывод добавленного элемента

На рисунке 3.12 представлен результат вызова функции печати после добавления одного элемента

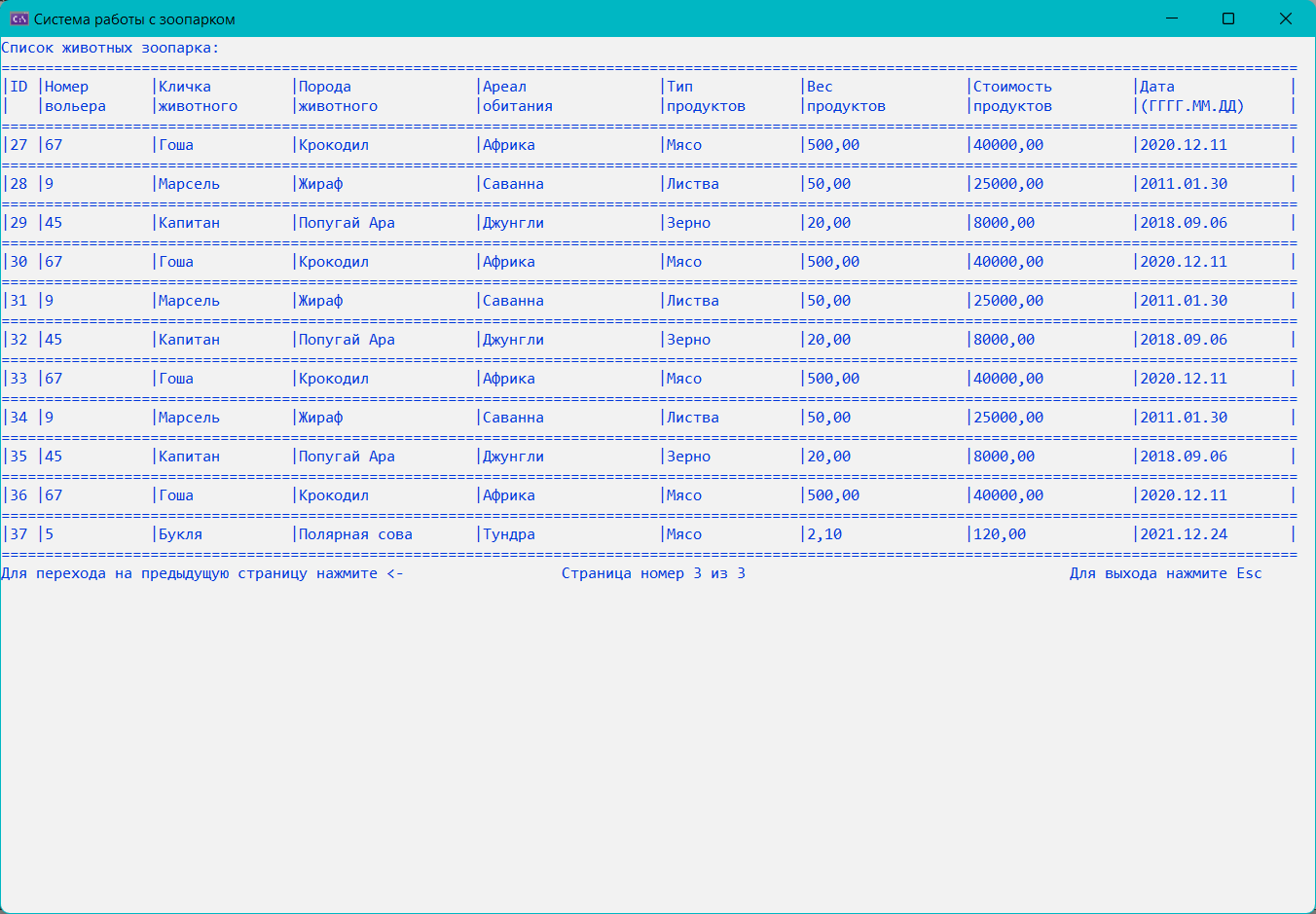


Рисунок 3.12– Результат добавления элемента и вызова функции печати

На рисунках 3.13-3.17 представлено меню редактирования существующей записи. Редактирование производится путем ввода id элемента. Далее у пользователя уточняется, действительно ли он хочет изменить данный элемент. Если пользователь соглашается – ему необходимо уточнить, какое именно поле элемента он хочет изменить. После ввода новой информации – изменённый элемент выводится на экран.



Рисунок 3.13 – Ввод id элемента для изменения

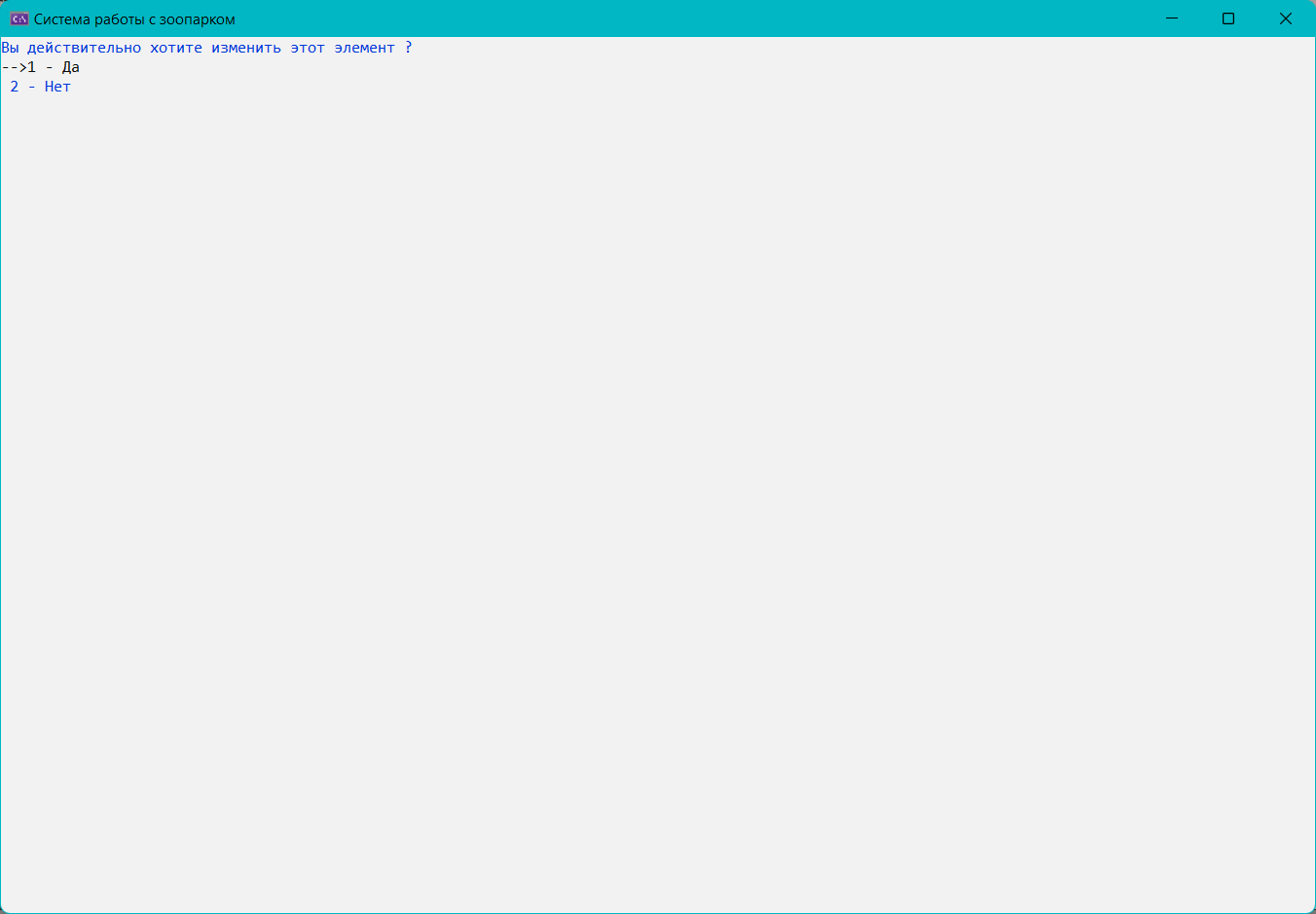


Рисунок 3.14 – Уточнение об изменении элемента

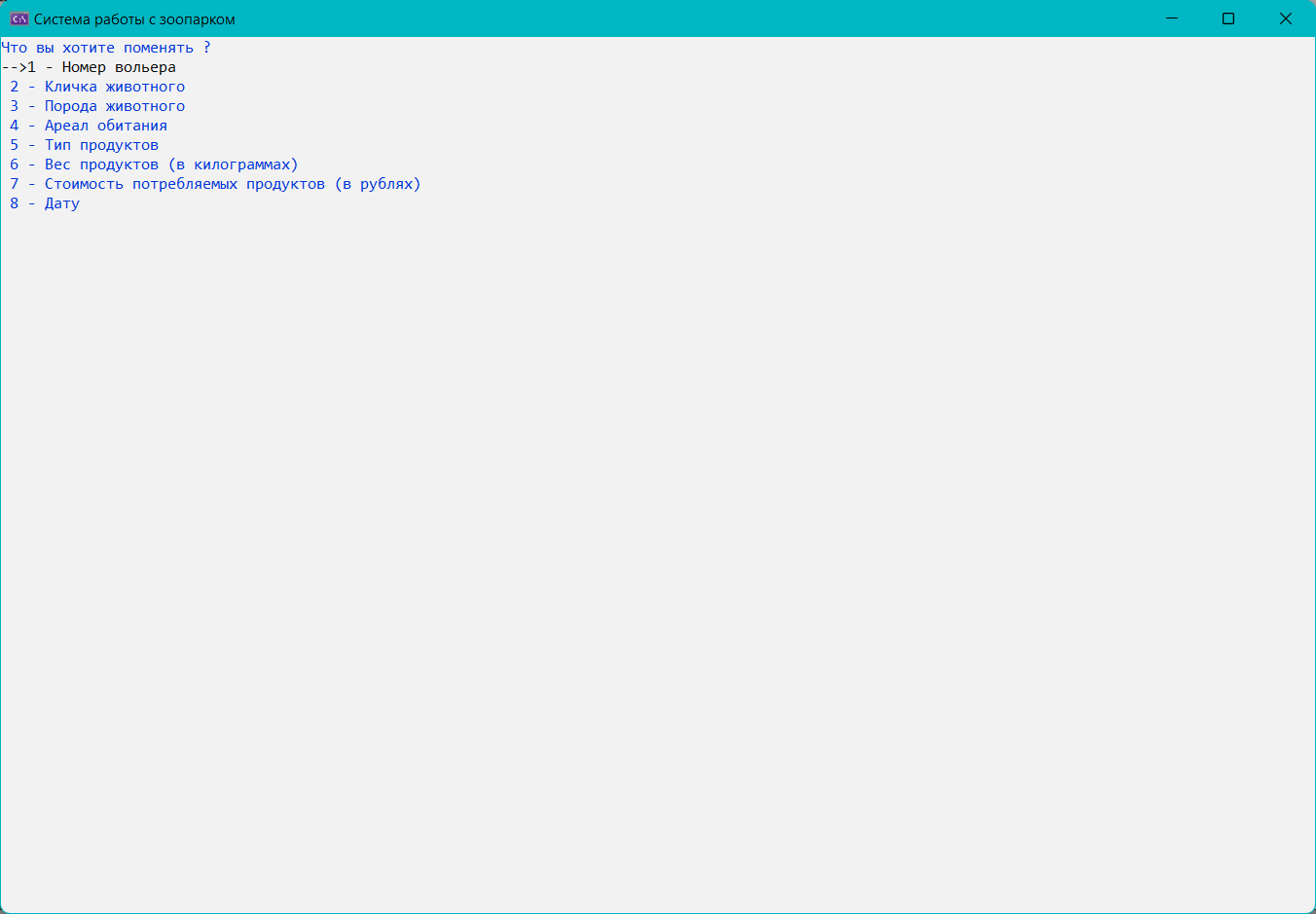


Рисунок 3.15 – Выбор поля для изменения

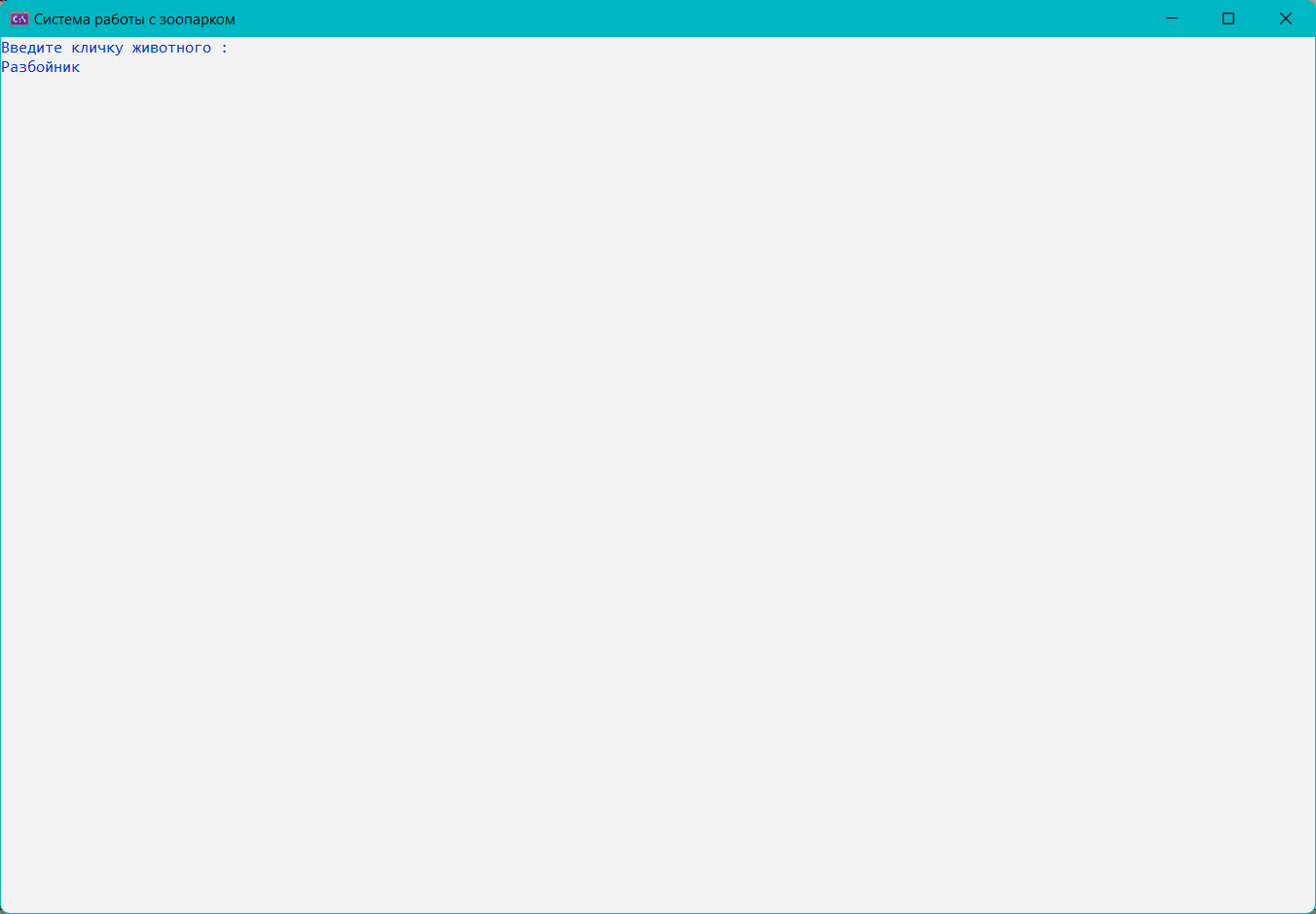


Рисунок 3.16 – Ввод новой информации для выбранного поля

Рисунок 3.17 – Вывод отредактированного элемента на экран

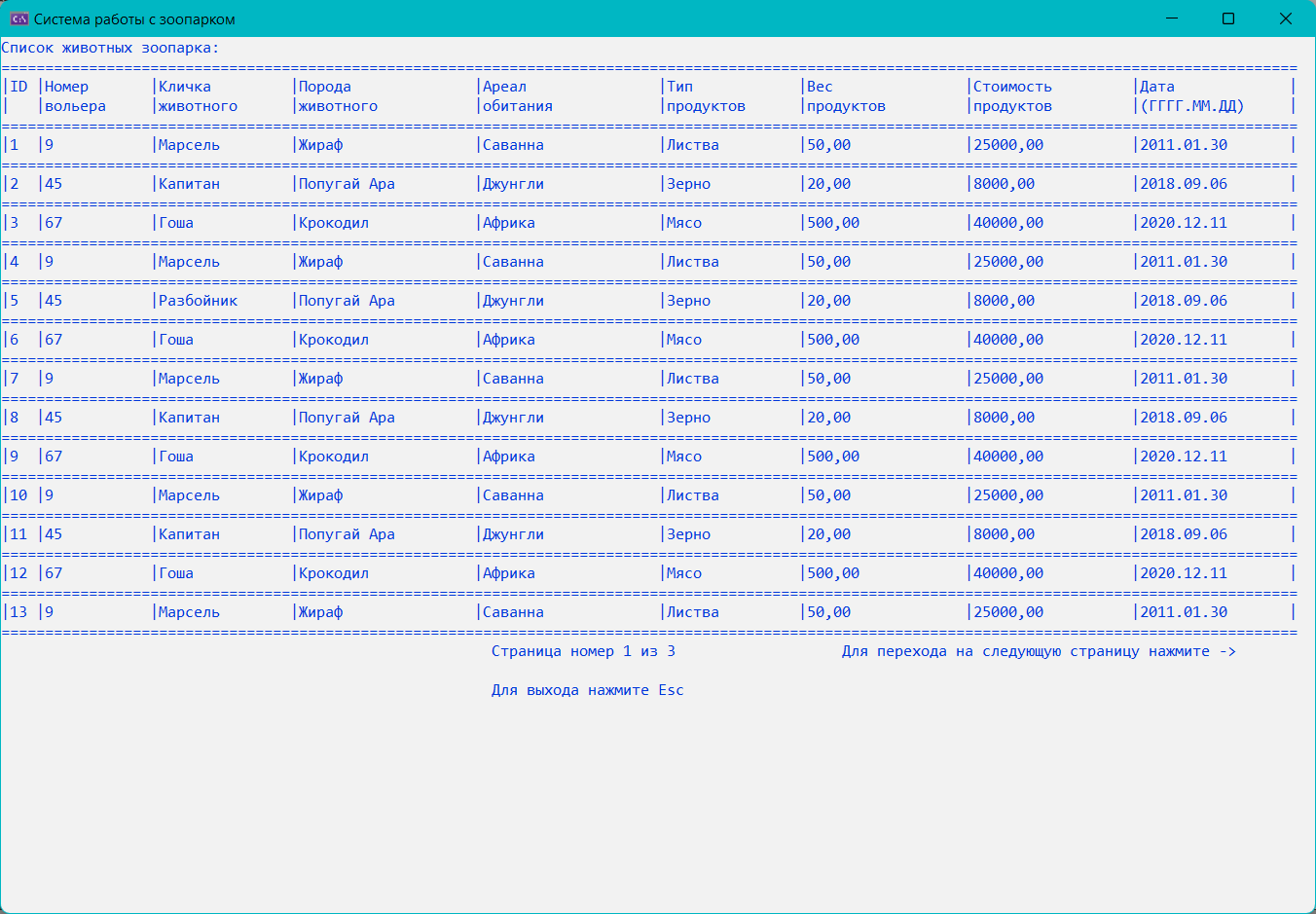
На рисунке 3.18 представлен результат редактирования записи. 

Рисунок 3.18 – Результат редактирования полей

На рисунках 3.19-3.21 представлено меню удаления элемента. При вызове данного меню программа задаёт вопрос о том , что именно пользователь хочет удалить – весь список или конкретный элемент. В случае выбора второго пункта – программа просит ввести id элемента и затем выводит его на экран. Далее пользователю задаётся вопрос о том , действительно ли он хочет удалить данный элемент. В случае положительного ответа – элемент выводится на экран и удаляется из списка .

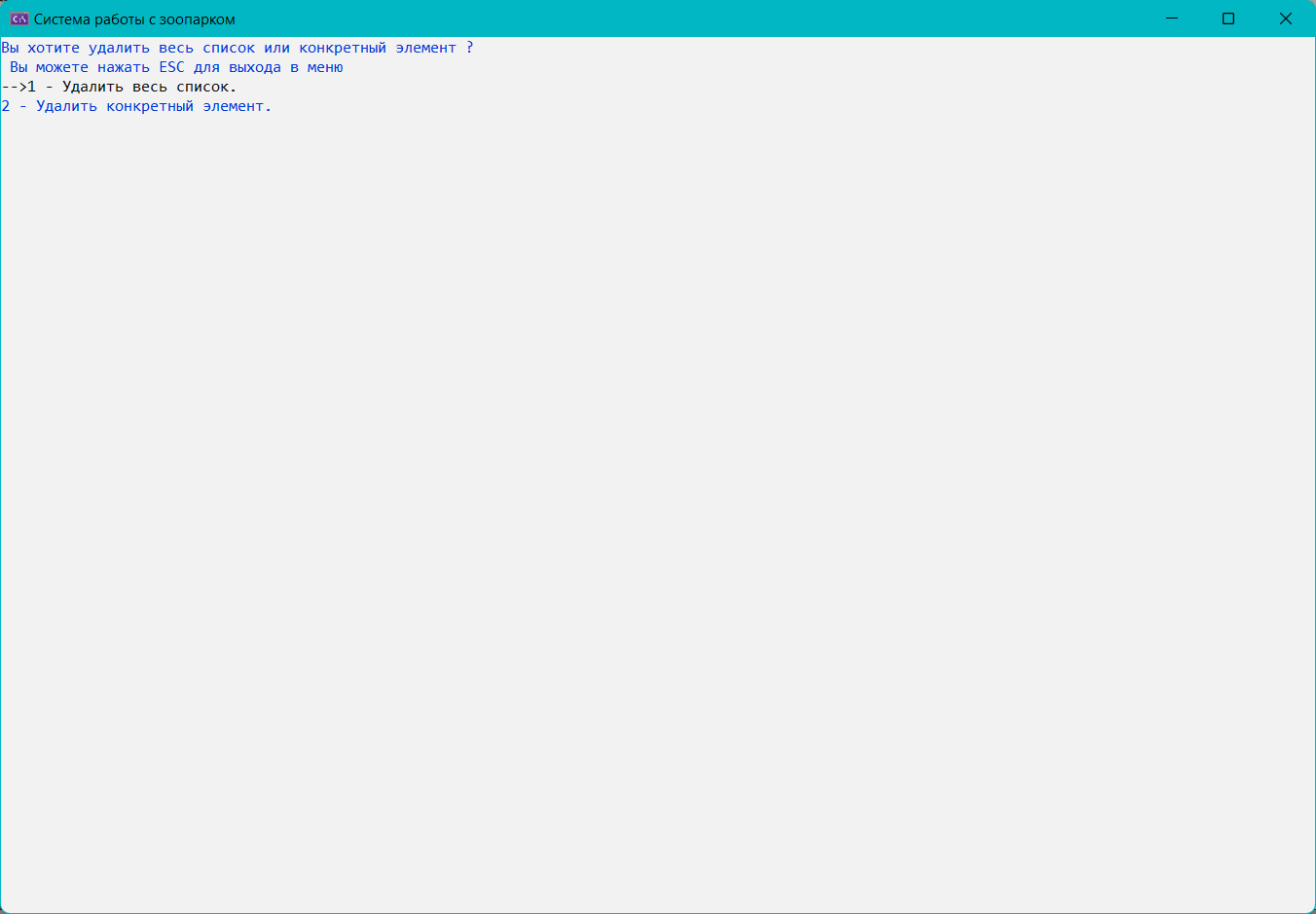


Рисунок 3.19 – Меню удаления



Рисунок 3.20 – Ввод id элемента и последующий вывод элемента на экран



Рисунок 3.21 – Подтверждение удаления элемента

На рисунке 3.22 представлен результат удаления элемента

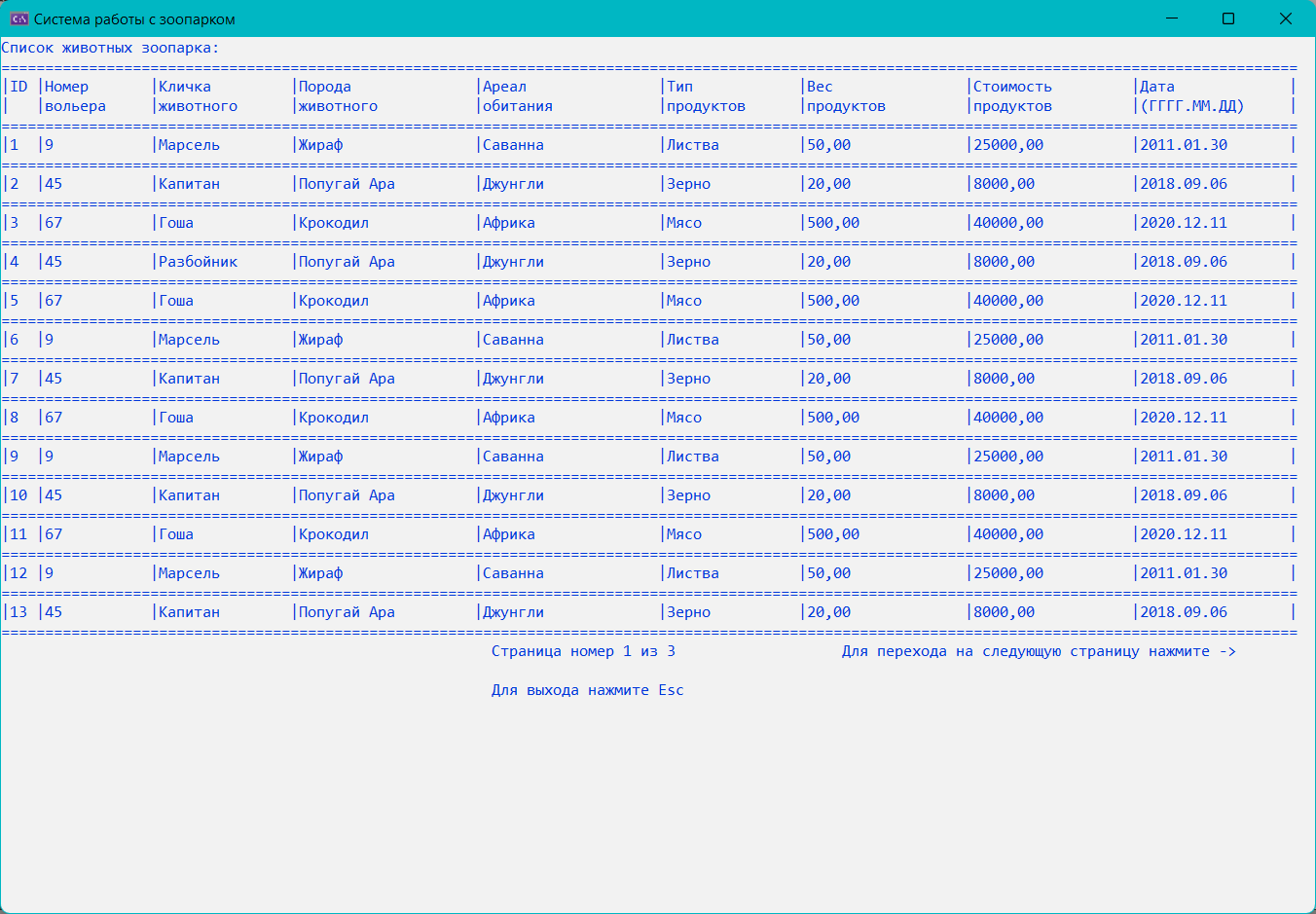


Рисунок 3.22 – Результат удаления элемента

На рисунках 3.23 и 3.24 представлен вызов функции сохранения списка.

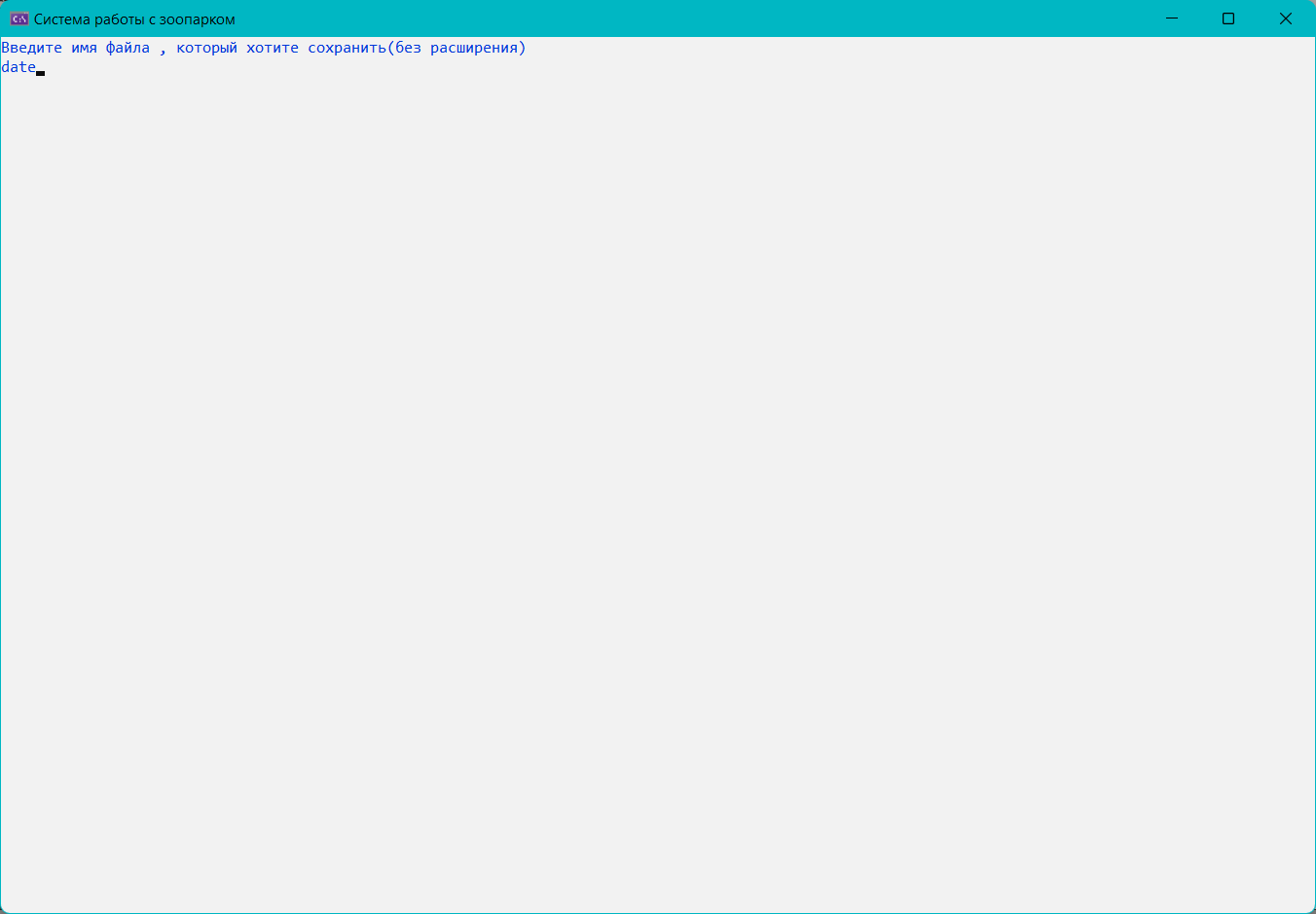


Рисунок 3.23 – Ввод имени сохраняемого файла

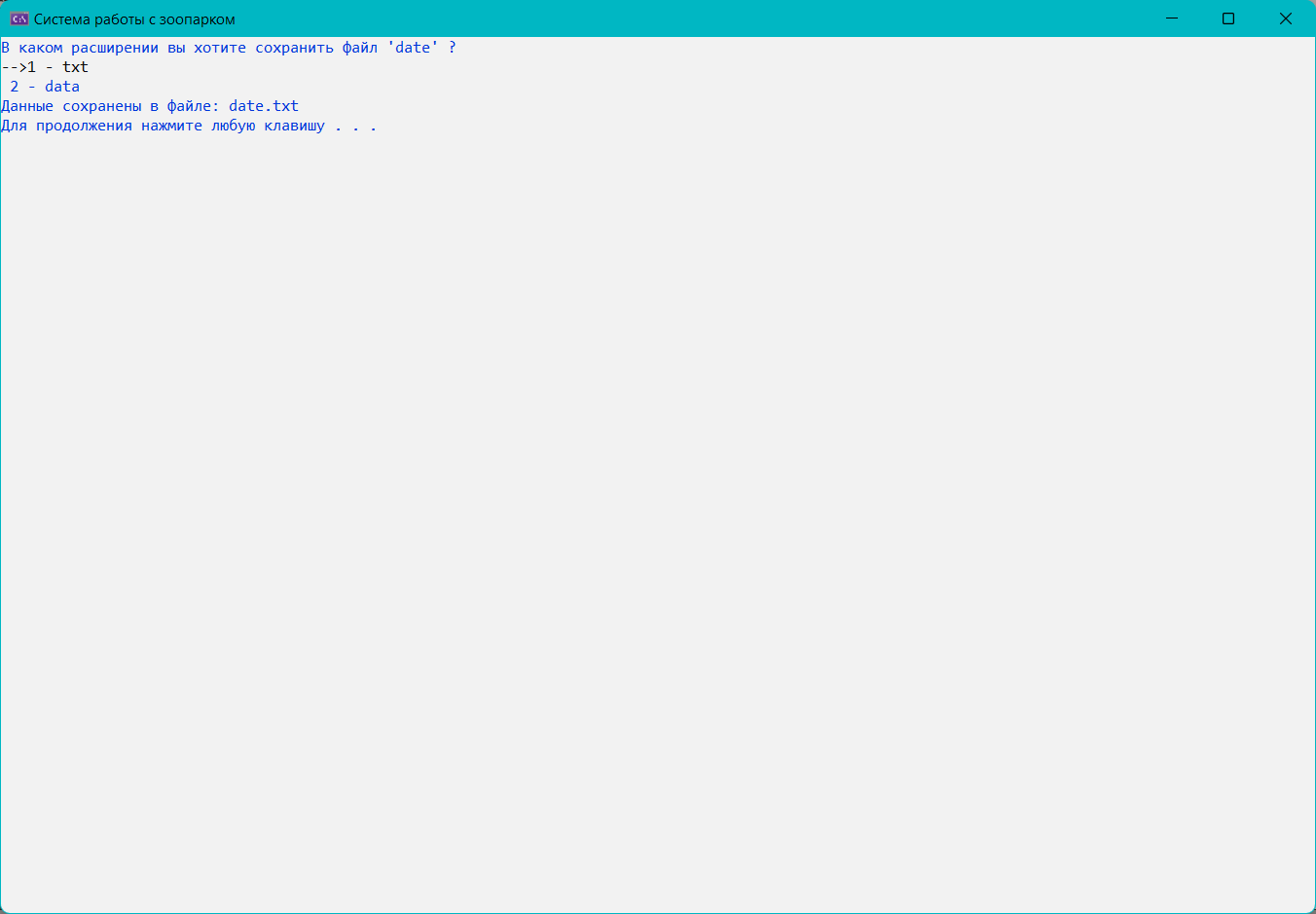


Рисунок 3.24 – Выбор расширения и сохранение файла «date.txt»

На рисунках 3.25-3.27 представлен результат вызова функции загрузки данных. После ввода имени и выбора расширения файла, пользователю необходимо выбрать добавить список из файла к текущему списку или заменить текущий список.

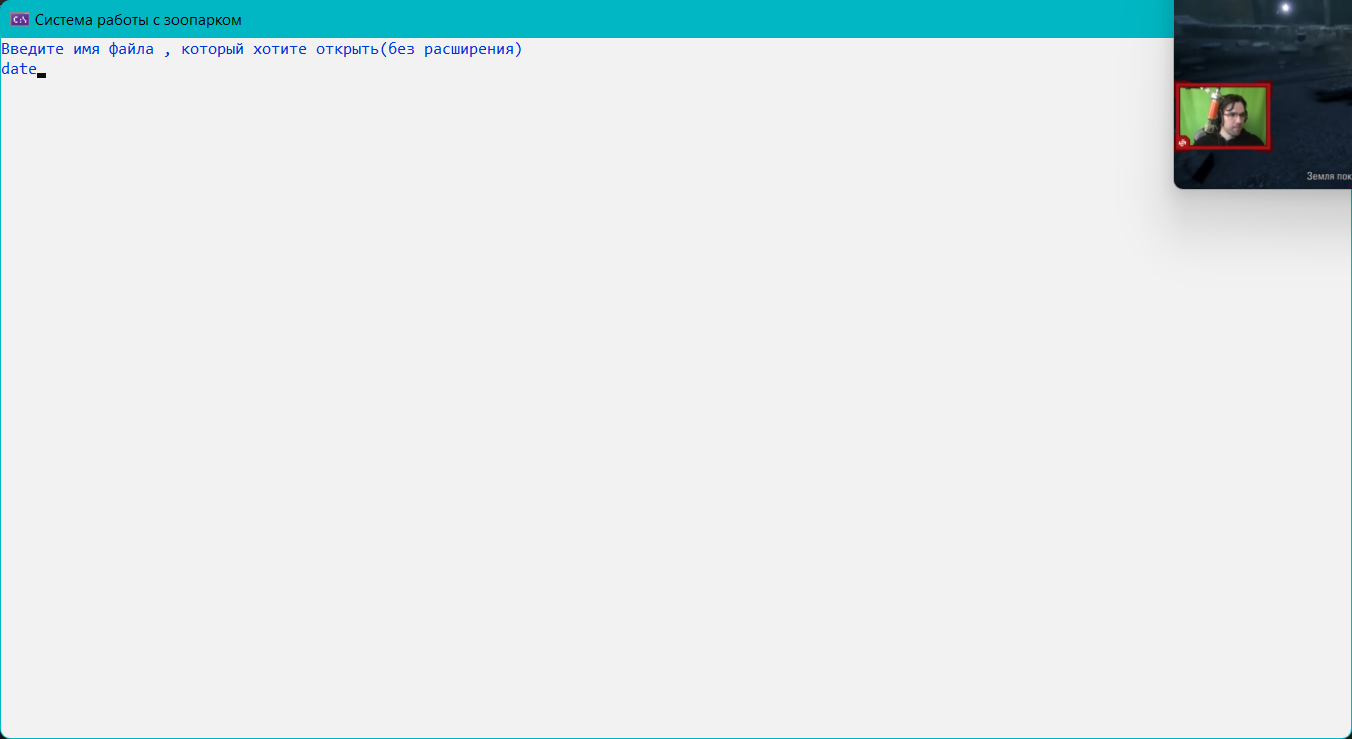


Рисунок 3.25 – Ввод имени файла для открытия

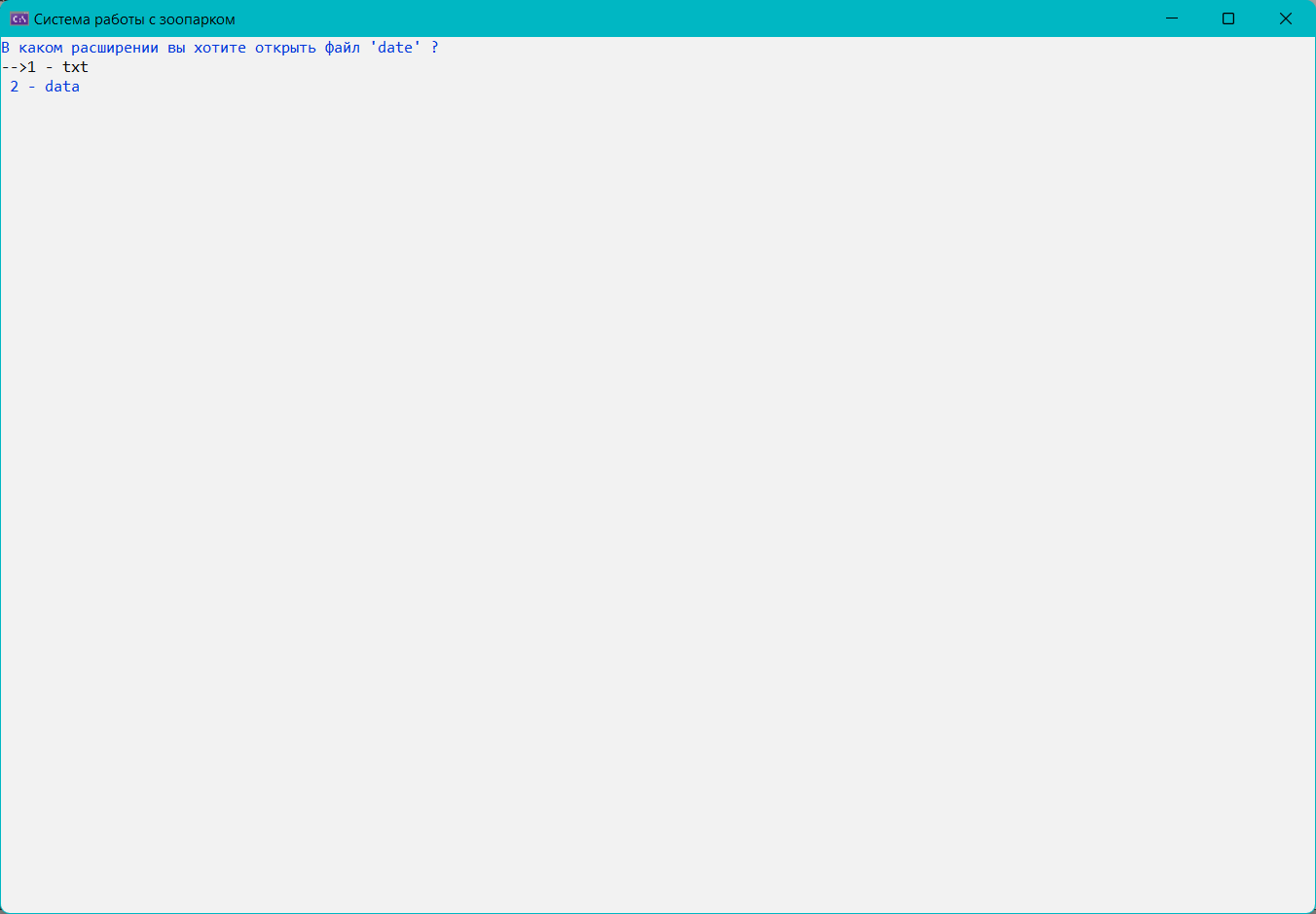


Рисунок 3.26 – Выбор расширения файла

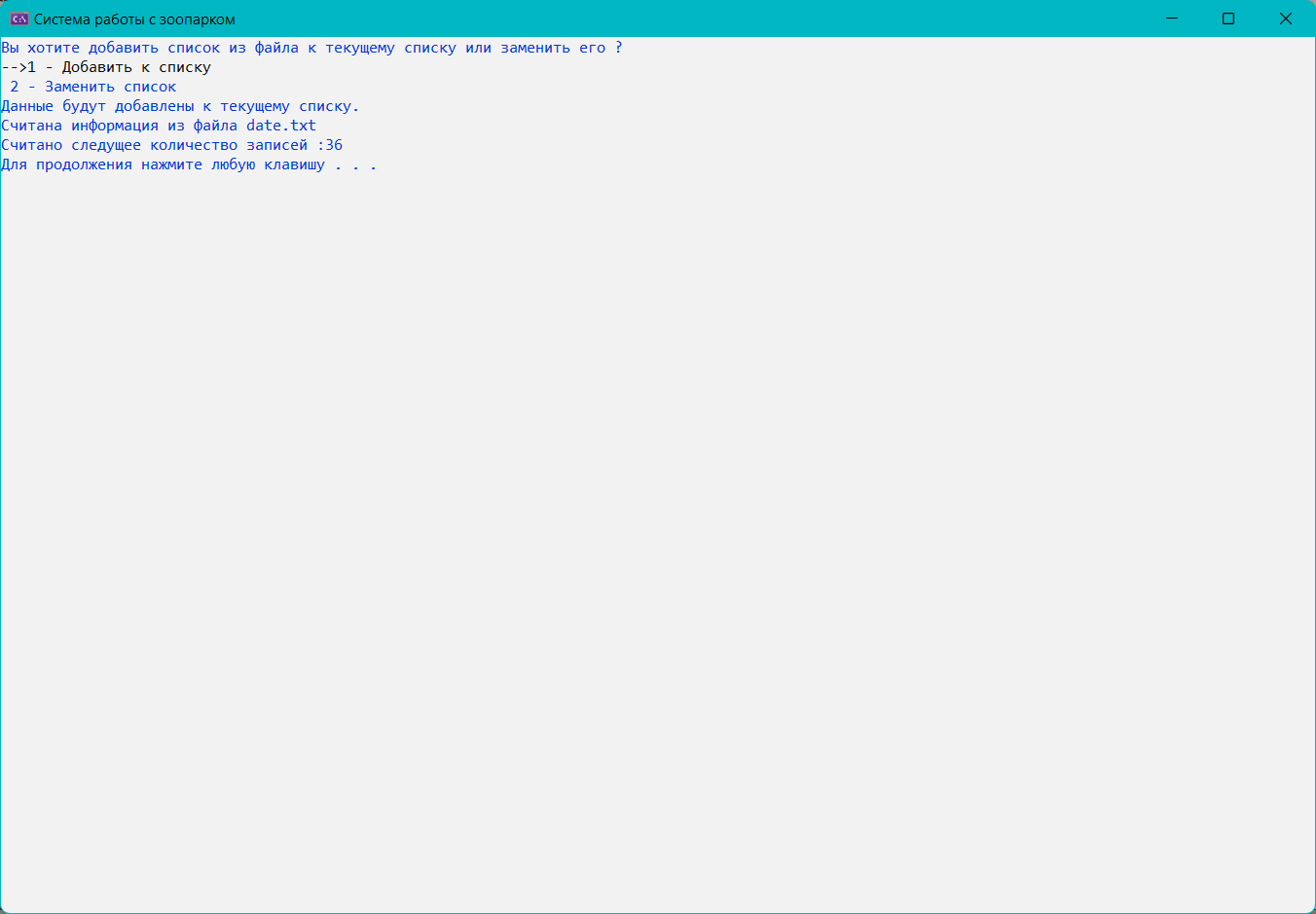


Рисунок 3.27 – Меню выбора добавления списка к текущему или его полная замена

На рисунке 3.28 представлен результат загрузки данных из файла.

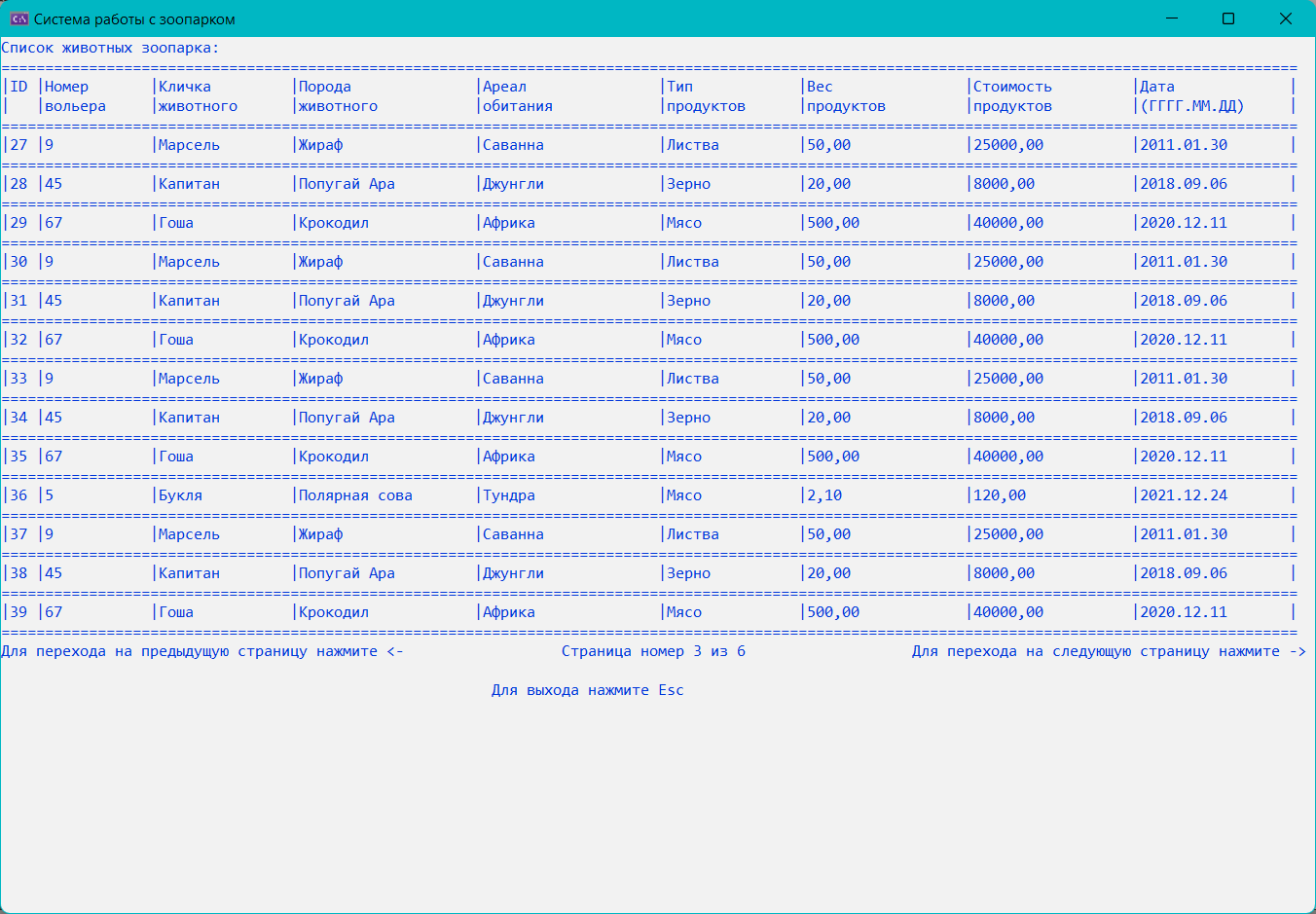
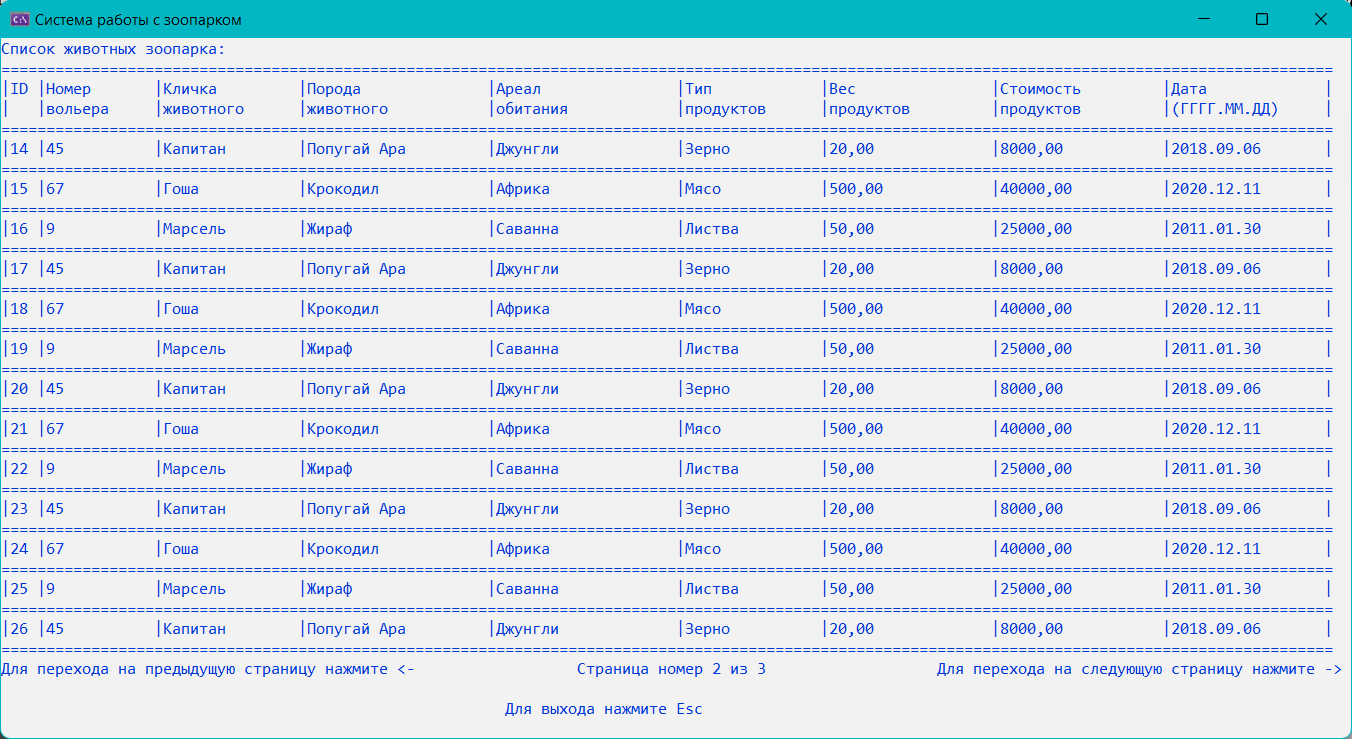


Рисунок 3.28 – Результат загрузки данных из файла

### 3.3 Проверка работоспособности программы

Для проверки работоспособности программы было произведено тестирование функций программы. Результаты представленны на рисунках ниже.

Был произведен вывод элементов в режиме скроллинга. Вывод программы при данном режиме представлен на рисунке 3.29.

Рисунок 3.29 – Результаты скроллинга на 1 страницу вправо

Для демонстрации работы функции поиска было выбран пункт «Поиск записи». Для этого была выбрана порода животного «Полярная сова». Результат поиска представлен на рисунках 3.30-3.32 .

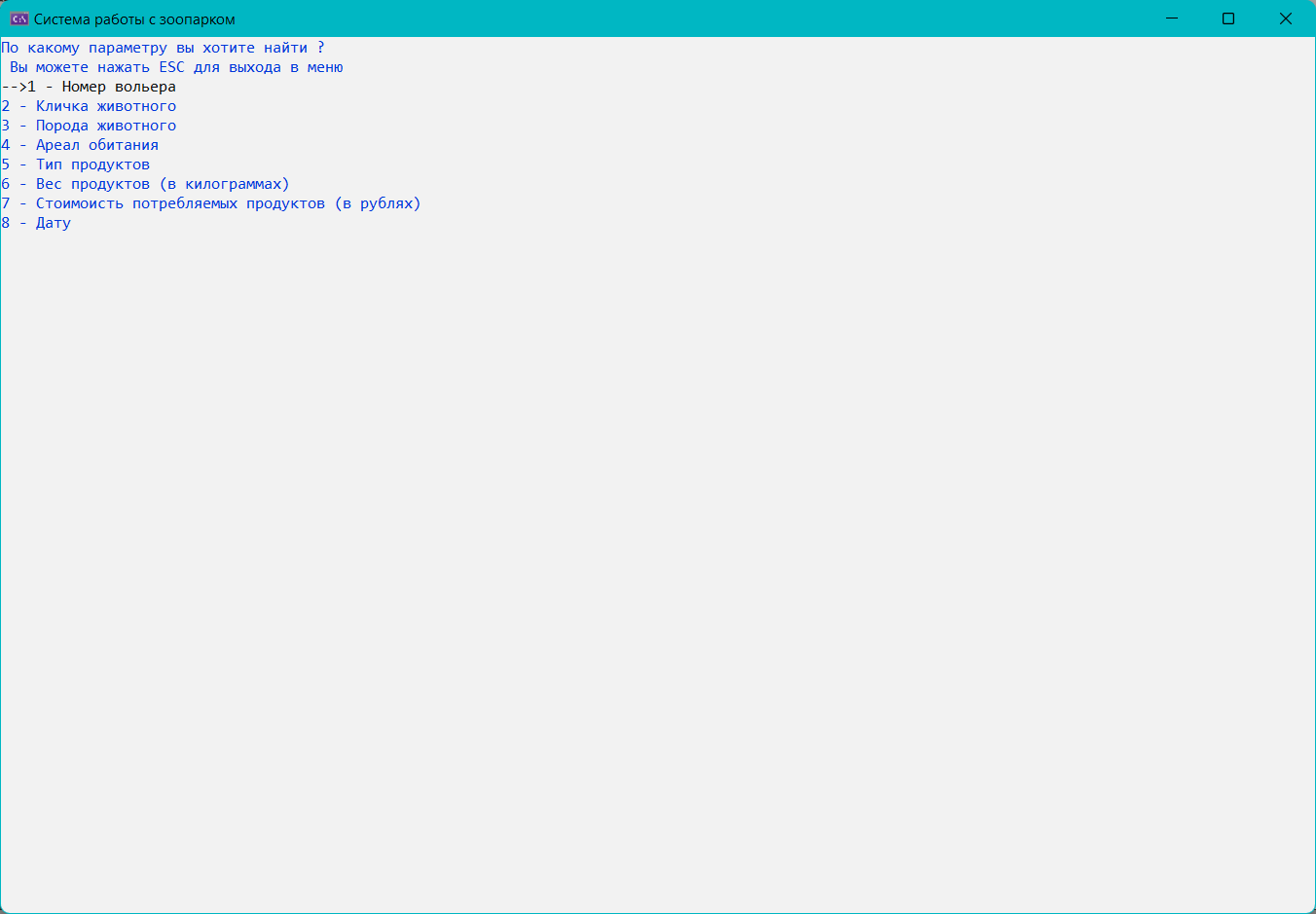


Рисунок 3.30 – Меню выбора поля для поиска

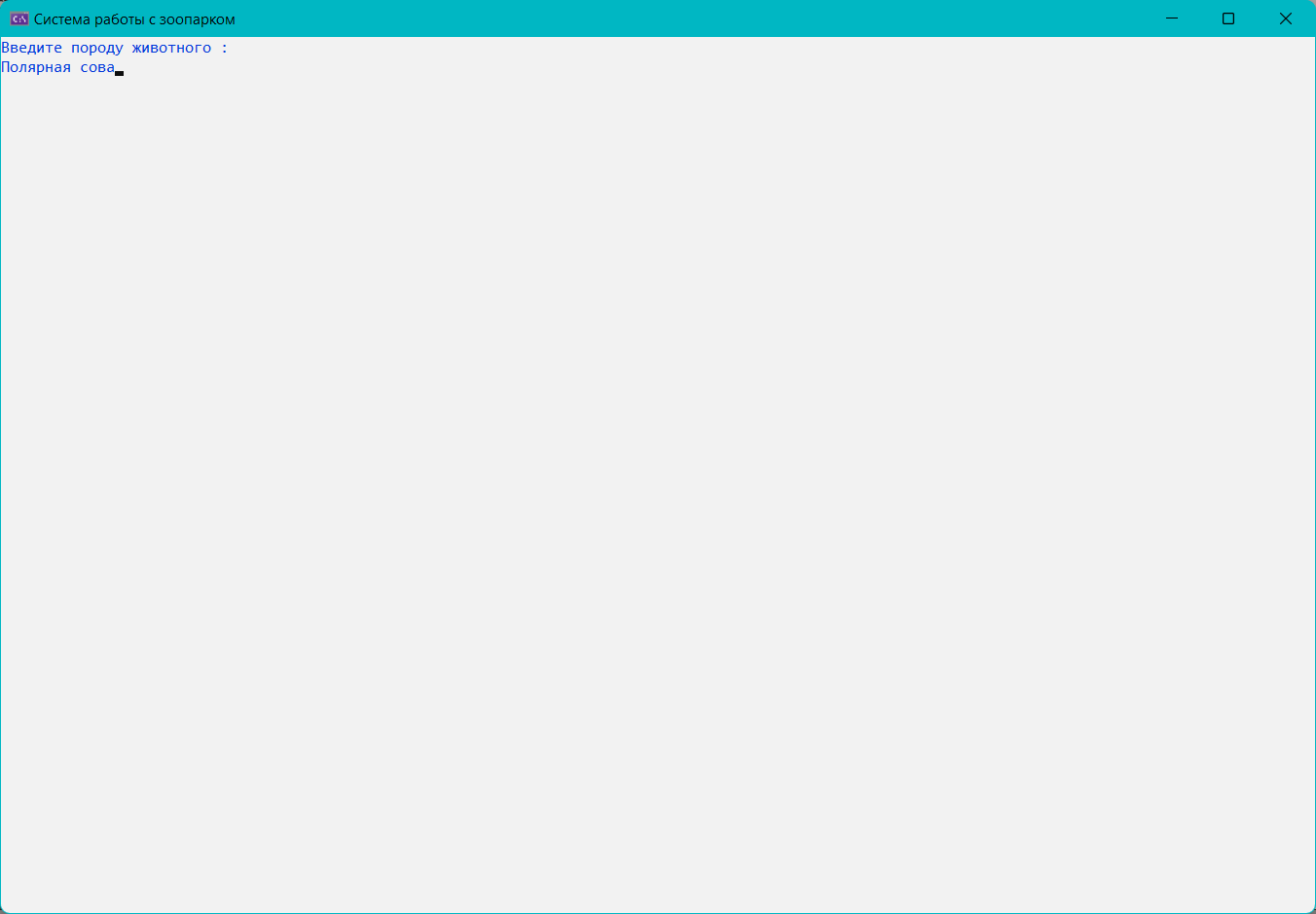


Рисунок 3.31 – Ввод данных для поиска



Рисунок 3.32 – Результат поиска

Был выбран пункт меню «сортировка». Работа программы в данном режиме продемонстрирована на рисунках 3.33-3.35

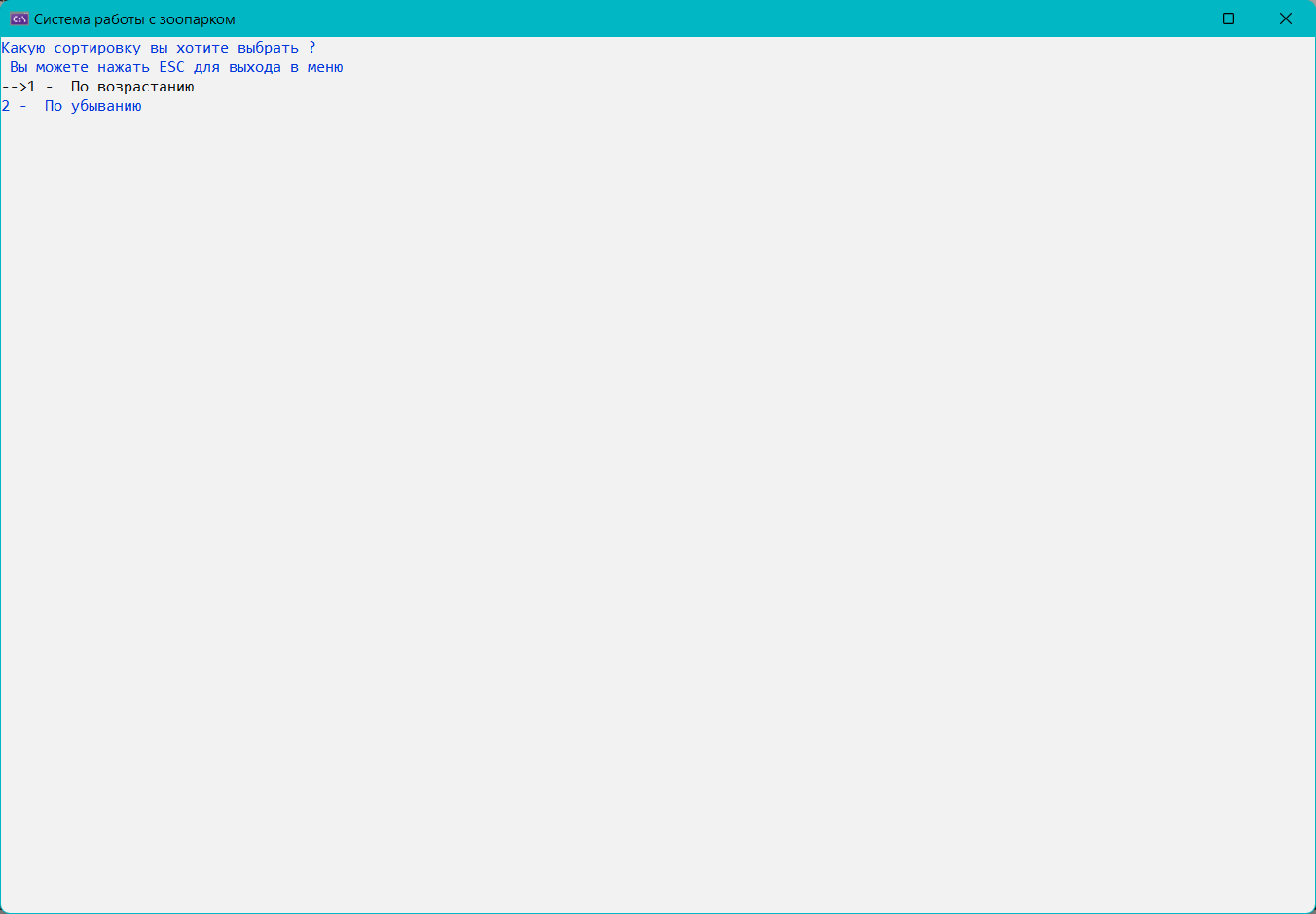


Рисунок 3.33 – Выбор режима сортировки

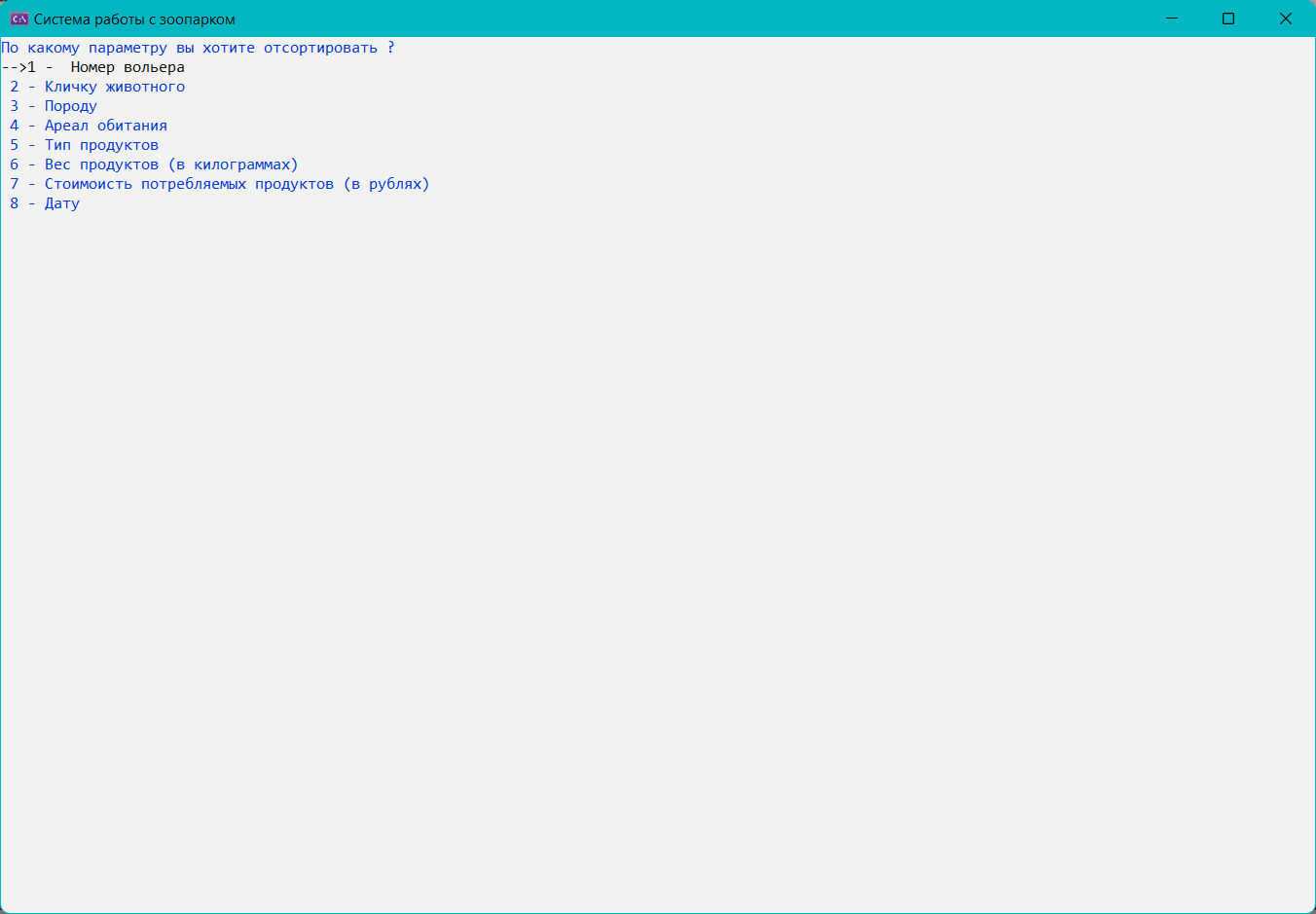


Рисунок 3.34 – Выбор поля для сортировки

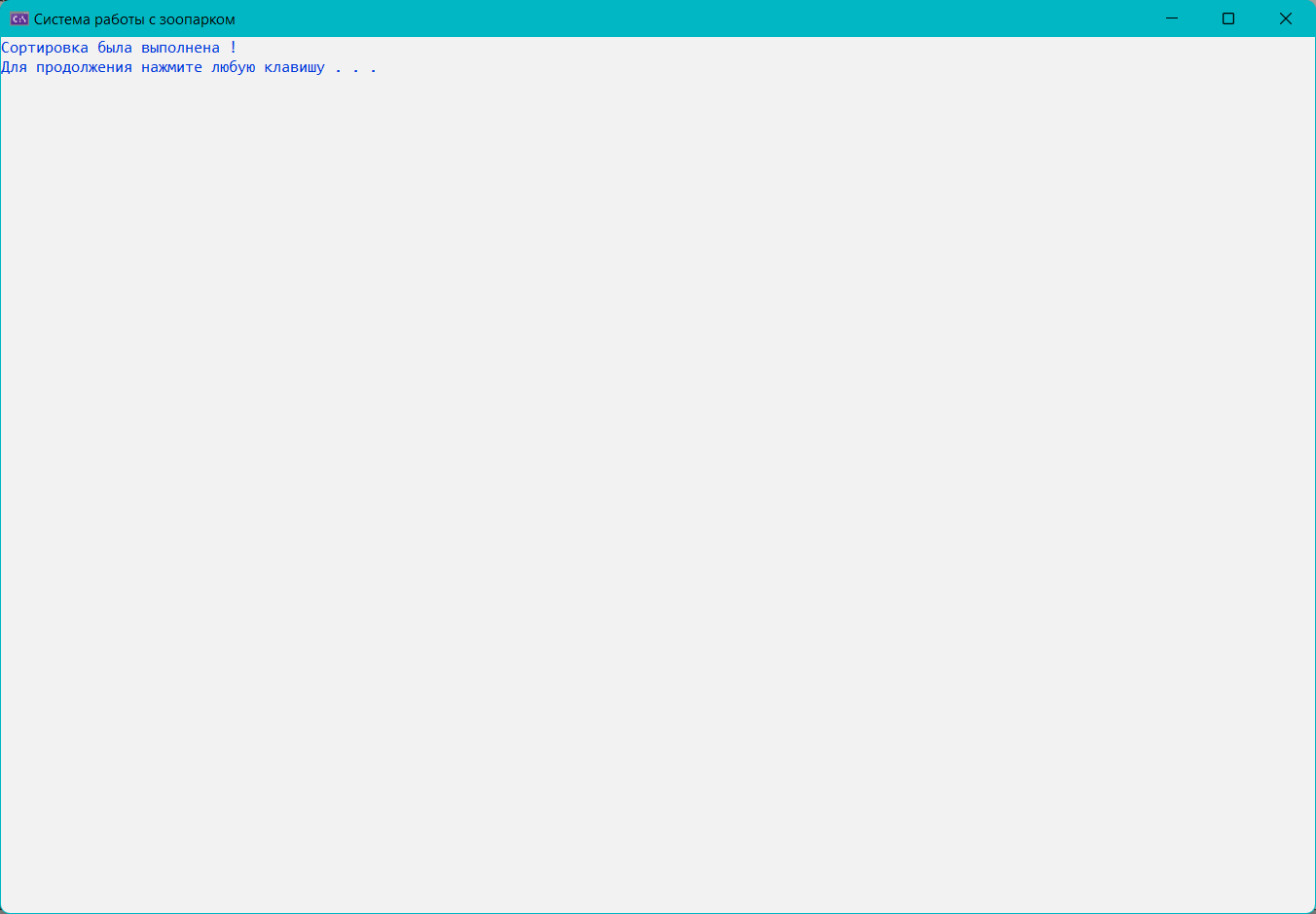


Рисунок 3.35 – Уведомления о выполнении сортировки

Был подтвержден выбор поля сортировки. Тип сортировки был выбран «по убыванию» после чего была выведена отсортированная таблица (рисунок 3.36 и 3.37).



Рисунок 3.36 – Вывод таблицы после сортировки (страница 1)



Рисунок 3.37 – Вывод таблицы после сортировки (страница 2)

Далее был выбран пункт меню «Подсчёт суммы и веса употреблённых продуктов всеми видами пород животных» после чего программа попросила пользователя ввести дату начала и окончания временного промежутка.

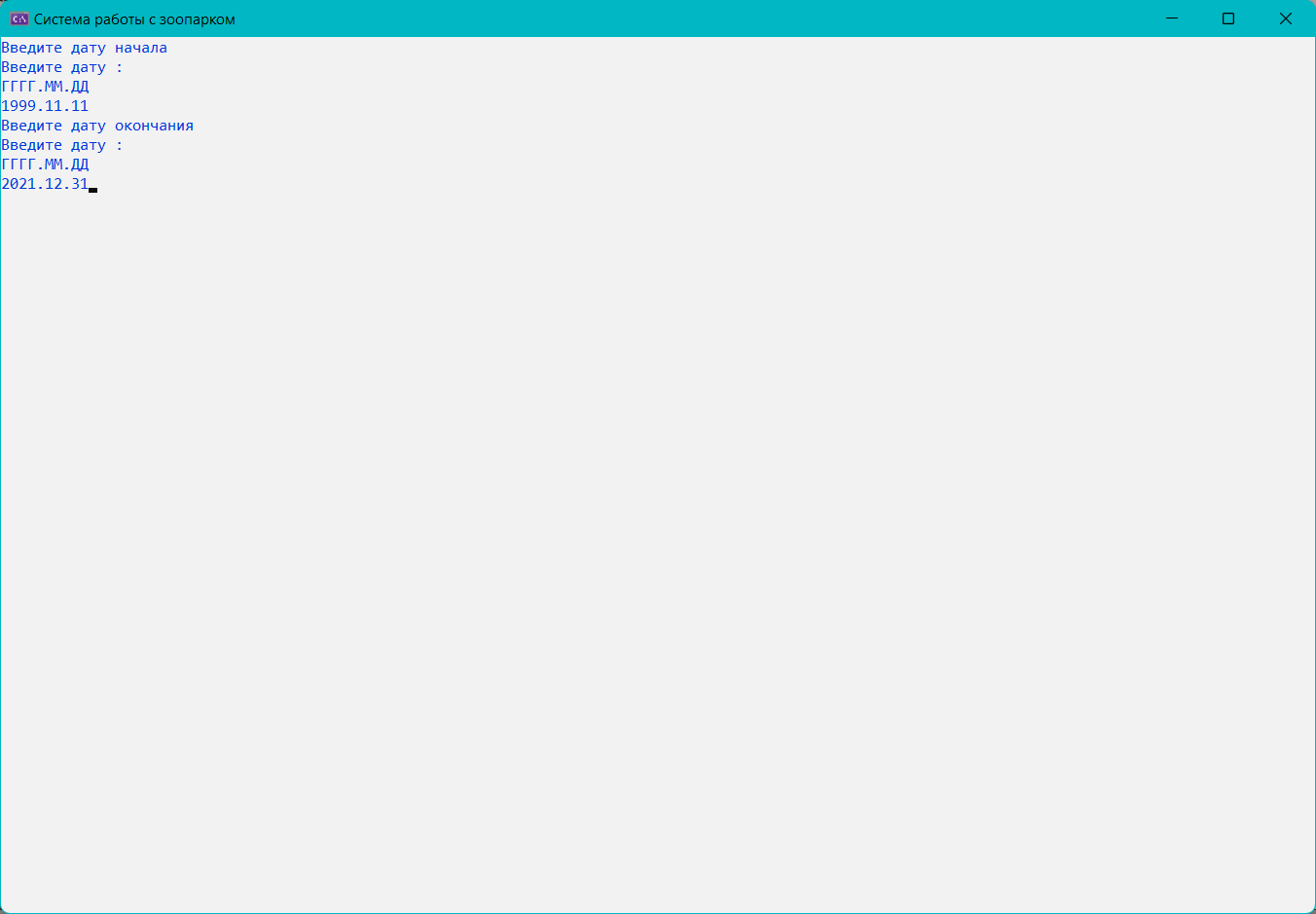


Рисунок 3.38 – Ввод даты начала и конца времменного промежутка

Далее программа подсчитывает сумму стоимости и вес употреблённых продуктов всеми видами пород животных, за указаннй пользователем период, а также просит пользователя ввести название файла, в который будут сохранены выведенные данные.



Рисунок 3.39 -Вывод данных на экран и сохранение данных в файл

На рисунке 3.40 показано содержимое результирующего файла «inf.txt»

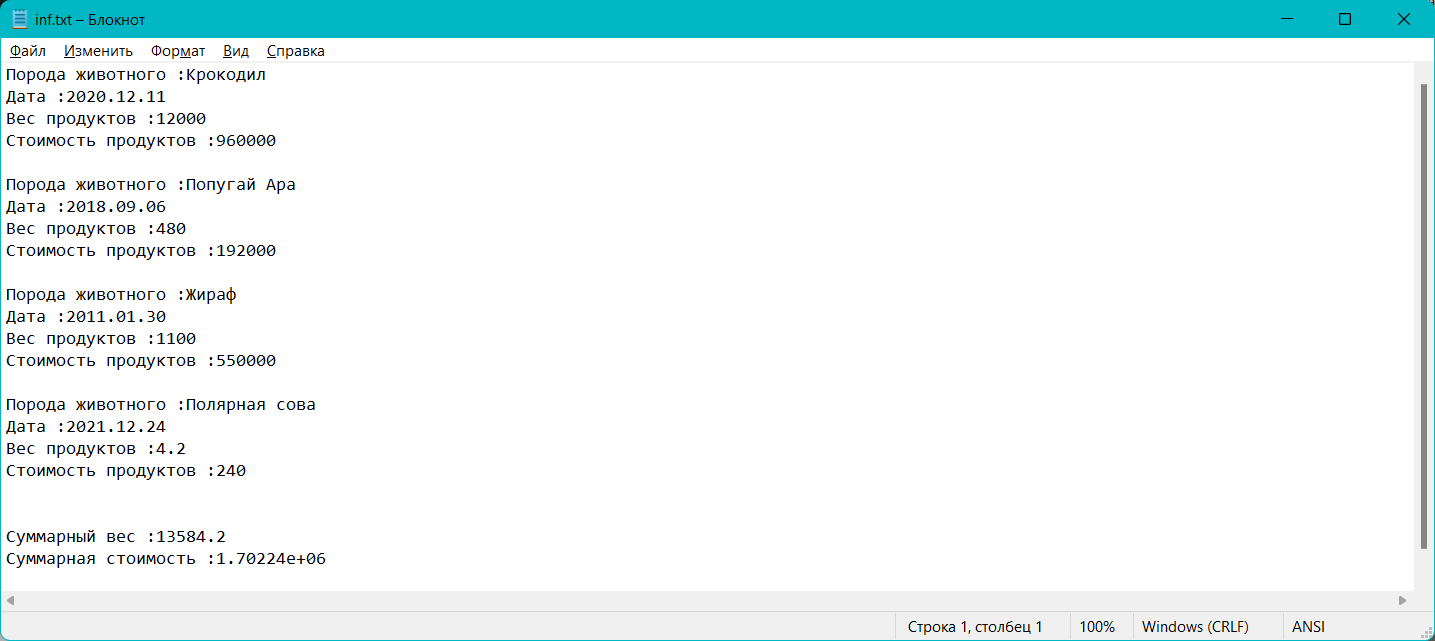


Рисунок 3.40 – Содержимое inf.txt

При выходе из программы пользователю предлагается сохранить данные. После выбора пользователем того или иного варианта, программа переходит в завершающий этап выполнения (рисунок 3.41).

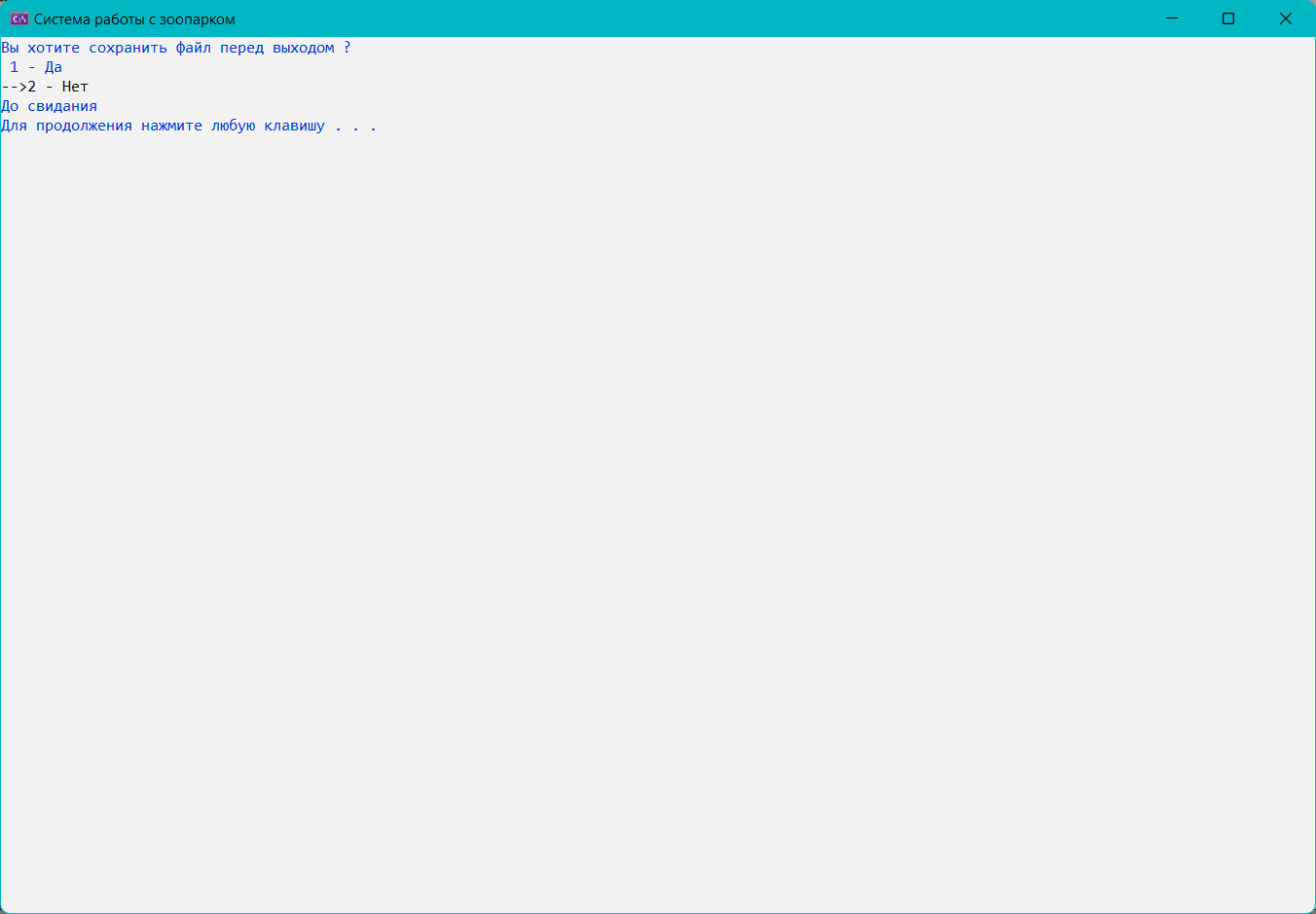


Рисунок 3.41 – Завершение программы

# ВЫВОДЫ

В соответствии с вариантом задания разработана программа, в основу алгоритма которой положена структура данных в виде двунаправленного списка, позволяющая выполнять просмотр данных в двух направлениях, что позволяет реализовать быстрые функции для скроллинга по списку, более быстрые функции сортировки, добавления и удаления данных. Особенностями программы является также возможность производить поиск и сортировку по практически полям записей, что достигается путём унификации алгоритмов сравнения и ввода, возможность печати таблицы. Разработка производилась на операционной системе Windows 11 x64 в IDE «Microsoft Visual Studio 2022». Программа использует ряд функций предоставляемых WinAPI. В процессе разработки возникла проблема с чтением строк, содержащих пробельные символы, что было исправлено замещением при сохранении всех пробельных символов нижним подчеркиванием, что гарантирует корректную работу программы независимо от количества пробельных символов. Также проблемой при разработке приложения была фильтрация нежелательных комбинаций клавиш таких как Ctrl+C, Ctrl+Z, приводивших к завершению программы. Данная проблема была решена путем фильтрации вводимых данных с помощью функции \_getch.

Следует заметить, что разработка программы была осложнена увеличением связностей отдельных функций программы, увеличением объема кода. Это приводило к усложнению отладки, модернизации программы. Данная проблема может быть решена рефакторингом программы с учетом текущего функционала.

Работа программы в различных режимах продемонстрирована ранее. Для улучшения программы можно добавить скроллинг элементов в режиме изменения данных. Также улучшение может быть произведено путем добавления нового функционала, например, подсчета количества особей, которые находятся в вольере. Помимо улучшений в функциональном плане также может быть осуществлена модернизация графического интерфейса пользователя путем добавления окон, более наглядного отображения таблицы

Таким образом, была достигнута цель курсового проектирования – углублены знания языка С/С++, получен опыт разработки программ с использованием методологии структурного программирования. Изучены алгоритмы для разработки программ, использующих скроллинг. В результате проектирования была получена простая функционалу программа, работающая с табличной информацией, превосходящая конкурентов по производительности и системным требованиям.

# ПЕРЕЧЕНЬ ССЫЛОК

1. Белецкий Я. Энциклопедия языка Си / Я. Белецкий; пер. с польск. — М.: Мир, 1992. — 687 с.

2. Вирт Н. Алгоритмы и структуры данных / Н. Вирт; пер. с англ. — М.: Мир,1989. — 360 с.

3. Керниган Б., Ритчи Д. Язык программирования СИ: пер. с англ./Под ред. и спредисл. В.С. Штаркмана. —2-е изд., перераб. и доп. — М.; СПб. ; К. :

Вильямс, 2006. —272с.

4. Павловская Т.А. C/ C++. Программирование на языке высокого уровня: учеб.для студ. вузов, обуч. по напр. «Информатика и вычислительная техника» / Т. А. Павловская. – СПб.: Питер, 2009. – 461 с.

5. Павловская Т.А. С/С++. Структурное программирование: практикум / Т.А.

Павловская, Ю.А. Щупак. — СПб.: Питер, 2004.—239 с.

6. Павловская Т. А. Паскаль. Программирование на языке высокого уровня: практикум / Т. А. Павловская. — СПб.: Питер, 2006. — 317 с.

7. Павловская Т. А. Паскаль. Программирование на языке высокого уровня: учеб. для вузов / Т. А. Павловская. — СПб.: Питер, 2008. — 393 с.

8. Методические указания к лабораторным работам по дисциплине«Информатика» для студентов дневной и заочной форм обучения направления 09.03.02 — «Информационные системы и технологии», часть 1 /

Сост. В.Н.Бондарев, Т.И. Сметанина. — Севастополь: Изд-во СевГУ, 2014. — 44 с.

9. Методические указания к лабораторным работам по дисциплине«Информатика» для студентов дневной и заочной форм обучения направления 09.03.02 — «Информационные системы и технологии», часть 2 / Сост. В.Н.Бондарев, Т.И. Сметанина — Севастополь: Изд-во СевГУ, 2014. — 64с.

10. Структурное программирование на языке С/С++: методические указания клабораторным работам по дисциплине «Основы программирования и алгоритмические языки» для студентов дневной и заочной форм обучения направления 09.03.02 — «Информационные системы и технологии», часть 1 / Сост. В.Н. Бондарев, Т.И. Сметанина.— Севастополь: Изд-во СевГУ, 2015. — 60 с.

11. Структурное программирование на языке С/С++: методические указания клабораторным работам по дисциплине «Основы программирования и алгоритмические языки» для студентов дневной и заочной форм обучения направления 09.03.02 — «Информационные системы и технологии», часть 2 / Сост. В.Н. Бондарев, Т.И. Сметанина.— Севастополь: Изд-во СевГУ, 2015. — 60 с.

12. Разработка САПР: в 10 кн. Кн. 3. Проектирование программногообеспечения САПР: практ. пособие / Б. С. Федоров, Н. Б. Гуляев; под ред. А.В. Петрова. — М.: Высш. шк., 1990. — 159с.

# ПРИЛОЖЕНИЕ А

Код zoo.h

#ifndef ZOO\_H

#define ZOO\_H

#pragma once

enum Keys {

NoPressed,

Enter = 13,//Клавиша "Enter"

Esc = 27,//Клавиша "ESC"

Up = 72,//Клавиша "Up"

Left = 75,//Клавиша "Left"

Right = 77,//Клавиша "Right"

Down = 80,//Клавиша "Down"

};

enum ConsoleColor {

Black = 0,

Blue = 1,

Green = 2,

Cyan = 3,

Red = 4,

Magenta = 5,

Brown = 6,

LightGray = 7,

DarkGray = 8,

LightBlue = 9,

LightGreen = 10,

LightCyan = 11,

LightRed = 12,

LightMagenta = 13,

Yellow = 14,

White = 15

};

/\*константы нужные для выделения памяти\*/

const int n\_n = 16,//кличка

n\_s = 21,//порода

n\_a = 21,//ареал

n\_p = 16,//тип продуктов

n\_d = 11,//дата

print = 13;//константа для количества элементов на странице

using namespace std;//использование пространства имён

/\*Подсчитать сумму и вес употреблённых продуктов всеми видами пород животных (за указанный период).\*/

struct zoo //структура с информацией

{

unsigned int id,//ключевое поле скрытое от пользователя

cage;//номер воль #ifndef ZOO\_H

#define ZOO\_H

#pragma once

enum Keys {

NoPressed,

Enter = 13,//Клавиша "Enter"

Esc = 27,//Клавиша "ESC"

Up = 72,//Клавиша "Up"

Left = 75,//Клавиша "Left"

Right = 77,//Клавиша "Right"

Down = 80,//Клавиша "Down"

};

enum ConsoleColor {

Black = 0,

Blue = 1,

Green = 2,

Cyan = 3,

Red = 4,

Magenta = 5,

Brown = 6,

LightGray = 7,

DarkGray = 8,

LightBlue = 9,

LightGreen = 10,

LightCyan = 11,

LightRed = 12,

LightMagenta = 13,

Yellow = 14,

White = 15

};

/\*константы нужные для выделения памяти\*/

const int n\_n = 16,//кличка

n\_s = 21,//порода

n\_a = 21,//ареал

n\_p = 16,//тип продуктов

n\_d = 11,//дата

print = 13;//константа для количества элементов на странице

using namespace std;//использование пространства имён

/\*Подсчитать сумму и вес употреблённых продуктов всеми видами пород животных (за указанный период).\*/

struct zoo //структура с информацией

{

unsigned int id,//ключевое поле скрытое от пользователя

cage;//номер вольера

char name[n\_n],//кличка животного

specie[n\_s],//порода

area[n\_a],//ареал обитания

p\_type[n\_p],//тип продуктов

date[n\_d];// дата

float p\_weight,//вес продуктов

p\_money;//стоимость потребляемых продуктов

struct zoo\* next;//следущий элемент

struct zoo\* prev;//предыдущий элемент

};

#endifера

char name[n\_n],//кличка животного

specie[n\_s],//порода

area[n\_a],//ареал обитания

p\_type[n\_p],//тип продуктов

date[n\_d];// дата

float p\_weight,//вес продуктов

p\_money;//стоимость потребляемых продуктов

struct zoo\* next;//следущий элемент

struct zoo\* prev;//предыдущий элемент

};

#endif

Код Change.h

#ifndef CHANGE\_H

#define CHANGE\_H

#pragma once

void change(zoo\* beg, zoo\* end, int id\_num);//корректировка

#endif

Код Fileworks.h

#ifndef FILEWORKS\_H

#define FILEWORKS\_H

#pragma once

int read\_file(string filename, zoo\*\* beg, zoo\*\* end);//считывание из файла

int write\_in\_file(string filename, zoo\* temp);//запись в файл

#endif

Код Input.h

#ifndef INPUT\_H

#define INPUT\_H

#pragma once

int inp\_int(int n, const char\* inf);//проверка ввода полей типа int

char\* inp\_date(char\* date);//проверка ввода даты

bool check\_date(char\* date);//проверка ввода даты на правильность

float inp\_float(float temp, const char\* inf);//проверка ввода полей типа float

char\* inp\_char(char\* temp, const char\* inf, int size);//проверка ввода полей типа char

string inp\_filename(string temp, const char\* inf);

void cursor\_off\_on(bool inf);

bool check\_char(char\* temp);

char\* delete\_space(char\* temp, int size);

zoo input();

#endif

Код Menu.h

#ifndef MENU\_H

#define MENU\_H

#pragma once

#include <iostream>

#include <string>

#include <setjmp.h>

using namespace std;

int menu(string menuElems[], int size, string inf);//меню

int menu\_category(string menuElems[], int size, string inf, jmp\_buf env);

int start\_menu();

void goto\_x\_y(short x, short y);

#endif

Код Output.h

#ifndef OUTPUT\_H

#define OUTPUT\_H

#pragma once

#include "zoo.h"

void print\_on\_the\_screen(const zoo& z);//просмотр элемента

void see\_the\_information(zoo\* beg, zoo\* end);//просмотр всей информации

void header();//заголовок

void header\_proccessing();//заголовок для обработки

void print\_processing(const zoo& z);//Вывод элемента обработки на экран

void show\_processing(zoo\* beg);//Вывод списка элементов обработки

#endif

Код Processing.h

#ifndef PROCESSING\_H

#define PROCESSING\_H

#pragma once

void processing(zoo\* beg);//обработка согласно варианту

float count\_money(zoo\* beg\_p);//Подсчёт всей суммы денег для обработки

float count\_weight(zoo\* beg\_p);//Подсчёт всего веса для обработки

int processing\_write\_in\_file(string filename, zoo\* temp);

#endif

Код Queue.h

#ifndef QUEUE\_H

#define QUEUE\_H

#pragma once

#include "zoo.h"

zoo\* add\_first(const zoo& z);//добавление первого элемента

zoo\* add(zoo\* end, const zoo& z);//добавление элемента

zoo\* dellete\_id\_beg(zoo\* beg);//подфункции

zoo\* dellete\_id\_end(zoo\* beg, zoo\* end);//нужные

zoo\* dellete\_id(zoo\* beg, zoo\* end, int num);//для

void dellete\_all(zoo\* beg);//удаления

#endif

Код Search.h

#ifndef SEARCH\_H

#define SEARCH\_H

#pragma once

#include <setjmp.h>

#include "zoo.h"

void search(zoo\* beg,jmp\_buf env);//поиск не по ключевому полю

void search\_cage(zoo\* beg, int num\_cage);//поиск по номеру клетки

void search\_name(zoo\* beg, char\* pet\_name);//поиск по кличке

void search\_specie(zoo\* beg, char\* pet\_specie);//поиск по породе

void search\_area(zoo\* beg, char\* pet\_are);//поиск по ареалу обитания

void search\_p\_type(zoo\* beg, char\* p\_type);//поиск по типу продуктов

void search\_p\_weight(zoo\* beg, float p\_weight);//поиск по весу продуктов

void search\_p\_money(zoo\* beg, float sum\_money);//поиск по сумме потраченной на продукты

void search\_date(zoo\* beg, char\* data);//поиск по дате

void search\_id(zoo\* beg, int id);

#endif

Код Search.h

#ifndef SORT\_H

#define SORT\_H

#pragma once

#include <setjmp.h>

#include "zoo.h"

void sort\_by\_field(zoo\* beg,jmp\_buf env);//сортировка

void sort\_cage(zoo\* beg, int how\_sort);//сортировка по номеру вольера

void sort\_name(zoo\* beg, int how\_sort);//сортировка по кличке

void sort\_specie(zoo\* beg, int how\_sort);//сортировка по породе

void sort\_area(zoo\* beg, int how\_sort);//сортировка по ареалу обитания

void sort\_p\_type(zoo\* beg, int how\_sort);//сортировка по типу продуктов

void sort\_p\_weight(zoo\* beg, int how\_sort);//сортировка по весу продуктов

void sort\_p\_money(zoo\* beg, int how\_sort);//сортировка по стоимости потребляемых продуктов

void sort\_date(zoo\* beg, int how\_sort);//сортировка по дате

void sort\_replace(zoo\* left, zoo\* right, zoo\* temp);//функция для смены местами элементов функции

#endif#pragma once

Код Change.cpp

#include <iostream>

#include <Windows.h>

#include "zoo.h"

#include "Menu.h"

#include "Input.h"

#include "Output.h"

HANDLE ChangeHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

/\*Функция корректировки\*/

void change(zoo\* beg, zoo\* end, int id\_num)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

if (beg != 0)

{

int number = 0;

char templ\_n[n\_n], templ\_s[n\_s],

templ\_a[n\_a], templ\_p[n\_p], data[n\_d];

float weight = 0, money = 0;

strcpy(templ\_n, "");

strcpy(templ\_s, "");

strcpy(templ\_a, "");

strcpy(templ\_p, "");

strcpy(data, "");

string change\_menu[] = {

"Номер вольера",

"Кличка животного",

"Порода животного" ,

"Ареал обитания" ,

"Тип продуктов" ,

"Вес продуктов (в килограммах)" ,

"Стоимость потребляемых продуктов (в рублях)" ,

"Дату"

};

zoo\* temp = beg;

while (temp)

{

if (temp->id == id\_num)

{

switch (menu(change\_menu, sizeof(change\_menu), "Что вы хотите поменять ? "))

{

SetConsoleTextAttribute(ChangeHandle, (WORD)((White << 4) | Blue));

case 0:

system("cls");

cin.ignore();

cursor\_off\_on(TRUE);

number = inp\_int(number, "Введите номер вольера :");

temp->cage = number;

system("cls");

cout << "Номер вольера был изменён на " << temp->cage << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

case 1:

system("cls");

cin.ignore();

cursor\_off\_on(TRUE);

inp\_char(templ\_n, "Введите кличку животного :", n\_n);

strcpy(temp->name, templ\_n);

system("cls");

cout << "Кличка животного была изменена на " << temp->name << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

case 2:

system("cls");

cin.ignore();

cursor\_off\_on(TRUE);

inp\_char(templ\_s, "Введите породу животного :", n\_s);

strcpy(temp->specie, templ\_s);

system("cls");

cout << "Порода животного была изменена на " << temp->specie << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

case 3:

system("cls");

cin.ignore();

cursor\_off\_on(TRUE);

inp\_char(templ\_a, "Введите ареал обитания :", n\_a);

strcpy(temp->area, templ\_a);

system("cls");

cout << "Ареал обитания был изменён на " << temp->area << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

case 4:

system("cls");

cin.ignore();

cursor\_off\_on(TRUE);

inp\_char(templ\_p, "Введите тип продуктов :", n\_p);

strcpy(temp->p\_type, templ\_p);

system("cls");

cout << "Тип продуктов был изменён на " << temp->p\_type << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

case 5:

system("cls");

cin.ignore();

weight = inp\_float(weight, "Введите вес продуктов (в килограммах):");

temp->p\_weight = weight;

system("cls");

cout << "Вес потребляемых продуктов был изменён на " << temp->p\_weight << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

case 6:

system("cls");

cin.ignore();

cursor\_off\_on(TRUE);

money = inp\_float(money, "Введите стоимость потребляемых продуктов (в рублях):");

temp->p\_money = money;

system("cls");

cout << "Стоимость потребляемых продуктов была изменена на " << temp->p\_money << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

case 7:

system("cls");

cin.ignore();

cursor\_off\_on(TRUE);

inp\_date(data);

strcpy(temp->date, data);

system("cls");

cout << "Дата была изменена на " << temp->date << endl;

cout << "Отредактированный элемент :" << endl;

header();

print\_on\_the\_screen(\*temp);

break;

}

break;

}

else

{

temp = temp->next;

}

}

}

else

{

cout << "Невозможно использовать функцию ! Отсутствуют элементы очереди!" << endl;

}

cursor\_off\_on(FALSE);

SetConsoleCP(866);

SetConsoleOutputCP(866);

}

Код Fileworks.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

#include <cstdlib>

#include "zoo.h"

#include "Queue.h"

HANDLE FileworksHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

/\*Чтение из файла\*/

int read\_file(string filename, zoo\*\* beg, zoo\*\* end)

{

SetConsoleTextAttribute(FileworksHandle, (WORD)((White << 4) | Blue));

string type = ".data";

int count=0;

if (filename.find(type) != std::string::npos)

{

zoo temp;

const char\* file = filename.c\_str();

FILE\* fin = fopen(file, "rb");

if(!fin) { cout << "Файл " << filename << " не найден!" << endl; cin.get(); return 1; }

else

{

while (fread(&temp.cage, sizeof(int), sizeof(temp.cage), fin))

{

fread(&temp.name, sizeof(char), n\_n, fin);

fread(&temp.specie, sizeof(char), n\_s, fin);

fread(&temp.area, sizeof(char), n\_a, fin);

fread(&temp.p\_type, sizeof(char), n\_p, fin);

fread(&temp.date, sizeof(char), n\_d, fin);

fread(&temp.p\_weight, sizeof(float), sizeof(temp.p\_weight), fin);

fread(&temp.p\_money, sizeof(float), sizeof(temp.p\_money), fin);

if (\*beg == 0)//если список ещё не создан

{

\*beg = add\_first(temp); \*end = \*beg;//создаём список

}

else//иначе

{

\*end = add(\*end, temp);//добавляем к концу списка

}

count++;

}

}

fclose(fin);

}

else

{

ifstream fin(filename, ios::\_Nocreate);

if (!fin) { cout << "Файл " << filename << " не найден!" << endl; cin.get(); return 1; }//файл не найден

zoo d;

while (fin.getline(d.name, n\_n))//пока в файле есть структуры

{//считываем кличку

fin.getline(d.specie, n\_s);//считываем породу

fin.getline(d.area, n\_a);//считываем среду обитания

fin.getline(d.p\_type, n\_p);//считываем тип продуктов

fin.getline(d.date, n\_d);//считываем дату

fin >> d.cage;//считываем номер вольера

fin >> d.p\_weight;//считываем вес продуктов

fin >> d.p\_money;//считываем сумму потраченную на продукты

fin.get();

if (\*beg == 0)//если список ещё не создан

{

\*beg = add\_first(d); \*end = \*beg;//создаём список

}

else//иначе

{

\*end = add(\*end, d);//добавляем к концу списка

}

count++;

}

fin.close();

}

cout << "Считана информация из файла " << filename << endl;

cout << "Считано следущее количество записей :" << count << endl;

return 0;

}

/\*Запись в файл\*/

int write\_in\_file(string filename, zoo\* temp)

{

SetConsoleTextAttribute(FileworksHandle, (WORD)((White << 4) | Blue));

string type = ".data";

if (filename.find(type) != std::string::npos)

{

const char\* file = filename.c\_str();

FILE\* fout = fopen(file, "wb");

if (!fout) { cout << "Невозможно открыть файл " << filename << " для записи" << endl; return 1; }

else

{

zoo\* templ = temp;

while (templ)

{

fwrite(&templ->cage, sizeof(int), sizeof(templ->cage), fout);

fwrite(&templ->name, sizeof(char), n\_n, fout);

fwrite(&templ->specie, sizeof(char), n\_s, fout);

fwrite(&templ->area, sizeof(char), n\_a, fout);

fwrite(&templ->p\_type, sizeof(char), n\_p, fout);

fwrite(&templ->date, sizeof(char), n\_d, fout);

fwrite(&templ->p\_weight, sizeof(float), sizeof(templ->p\_weight), fout);

fwrite(&templ->p\_money, sizeof(float), sizeof(templ->p\_money), fout);

templ = templ->next;

}

}

fclose(fout);

}

else

{

ofstream fout(filename);

if (!fout) { cout << "Невозможно открыть файл " << filename << " для записи" << endl; return 1; }//если поток

//открыть невозможно - вывести сообщение об ошибке

while (temp)//пока список не закончился

{

fout << temp->name << endl;//запись клички животного

fout << temp->specie << endl;//запись вида животного

fout << temp->area << endl;//запись ареала обитания

fout << temp->p\_type << endl;//запись типа продуктов

fout << temp->date << endl;//запись даты

fout << temp->cage << endl;//запись номера вольера

fout << temp->p\_weight << endl;//запись веса продуктов

fout << temp->p\_money << endl;//запись суммы денег потраченных на продукты

temp = temp->next;//переход к следующей структуре

}

fout.close();

}

cout << "Данные сохранены в файле: " << filename << endl;

return 0;

}

Код Input.cpp

#include <iostream>

#include <string>

#include <Windows.h>

#include "zoo.h"

#include "Input.h"

HANDLE InputHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

/\*Ввод информации\*/

zoo input()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

zoo z;

z.cage = 0;

strcpy(z.name, "");

strcpy(z.specie, "");

strcpy(z.area, "");

strcpy(z.p\_type, "");

z.p\_weight = 0;

z.p\_money = 0;

strcpy(z.date, "");

z.cage = inp\_int(z.cage, "Введите номер вольера :");

cin.ignore();

inp\_char(z.name, "Введите кличку животного :", n\_n);

inp\_char(z.specie, "Введите породу животного :", n\_s);

inp\_char(z.area, "Введите ареал обитания :", n\_a);

inp\_char(z.p\_type, "Введите тип продуктов :", n\_p);

z.p\_weight = inp\_float(z.p\_weight, "Введите вес продуктов (в килограммах):");

cin.ignore();

z.p\_money = inp\_float(z.p\_money, "Введите стоимость потребляемых продуктов (в рублях):");

cin.ignore();

inp\_date(z.date);

cin.clear();

SetConsoleCP(866);

SetConsoleOutputCP(866);

return z;

}

/\*Проверка ввода перменных типа int\*/

int inp\_int(int n, const char\* inf)

{

bool int\_ = true;

system("cls");

cout << inf << endl;

cin >> noskipws >> n;

do

{

if (!cin ||n<0)

{

int\_ = false;

system("cls");

cin.clear();

cin.get();

cout << "Похоже вы неправильно ввели информацию. Повторите попытку." << endl;

cout << inf << endl;

cin >> noskipws >> n;

}

else int\_ = true;

} while (!int\_);

return n;

}

/\*Проверка ввода переменных типа char\*/

char\* inp\_char(char\* temp, const char\* inf, int size)

{

cout << inf << endl;

cin.getline(temp,size);

temp=delete\_space(temp, size);

while (strcmp(temp, "") == 0 || strcmp(temp, "\0") == 0||!check\_char(temp))

{

cout << "Похоже вы неправильно ввели информацию. Повторите попытку." << endl;

cout << inf << endl;

cin.getline(temp, size);

}

return temp;

}

bool check\_char(char\* temp)

{

int i = 0,count=0;

while (temp[i])

{

if(isdigit(temp[i])||ispunct(temp[i]))count++;

i++;

}

if (count == 0)return true;

else return false;

}

char\* delete\_space(char\* temp, int size)

{

int i = 0;

while (temp[i])

{

if (isspace(temp[0]))

for (i = 0; i < sizeof(temp); i++)

{

temp[i] = temp[i + 1];

}

i++;

}

return temp;

}

/\*Проверка ввода имени файла\*/

string inp\_filename(string temp, const char\* inf)

{

cin.clear();

cout << inf << endl;

cin >> noskipws >> temp;

while (temp.empty())

{

cin.clear();

cin.get();

cout << "Похоже вы неправильно ввели имя файла. Повторите попытку." << endl;

cout << inf << endl;

cin >> noskipws >> temp;

}

return temp;

}

/\*Проверка ввода переменных типа float\*/

float inp\_float(float temp, const char\* inf)

{

bool float\_ = true;

cin.clear();

cout << inf << endl;

cin >> noskipws >> temp;

do

{

if (!cin ||temp<0)

{

float\_ = false;

cin.clear();

cin.ignore();

cin.get();

cout << "Похоже вы неправильно ввели информацию. Повторите попытку." << endl;

cout << inf << endl;

cin >> noskipws >> temp;

}

else float\_ = true;

} while (!float\_);

return temp;

}

/\*Проверка ввода даты\*/

char\* inp\_date(char\* date)

{

cout << "Введите дату :" << endl;

cout << "ГГГГ.ММ.ДД" << endl;

cin.getline(date, n\_d);

while (strcmp(date, "") == 0 || strcmp(date, "\0") == 0 || !check\_date(date))

{

cout << "Похоже вы неправильно ввели информацию. Повторите попытку." << endl;

cout << "Введите дату :" << endl;

cout << "ГГГГ.ММ.ДД" << endl;

cin.getline(date, n\_d);

}

return date;

}

/\*Проверка правильности даты\*/

bool check\_date(char\* date)

{

int year = 0, month = 0, day = 0, count = 0;

for (int i = 0; i < 5; i++)

{

if (isdigit(date[i])) count++;

}

if (count != 4) return false;

count = 0;

for (int i = 0; i < 3; i++)

{

if (isdigit(date[i + 5])) count++;

}

if (count < 1 || count > 2) return false;

count = 0;

for (int i = 0; i < 2; i++)

{

if (isdigit(date[i + 8])) count++;

}

if (count < 1 || count >2) return false;

year += atoi(&date[0]);

month += atoi(&date[5]);

day += atoi(&date[8]);

if (year < 1900 || year>2021) return false;

if (month < 1 || month>12) return false;

switch (month)

{

case 1:

if (day < 1 || day>31) return false;

break;

case 2:

if (year % 100 == 0)

{

if (year % 400 == 0)

{

if (day < 1 || day>29) return false;

}

else { if (day < 1 || day>28) return false; }

}

else

{

if (year % 4 == 0) { if (day < 1 || day>29) return false; }

else { if (day < 1 || day>28) return false; }

}

break;

case 3:

if (day < 1 || day>31) return false;

break;

case 4:

if (day < 1 || day>30) return false;

break;

case 5:

if (day < 1 || day>31) return false;

break;

case 6:

if (day < 1 || day>30) return false;

break;

case 7:

if (day < 1 || day>31) return false;

break;

case 8:

if (day < 1 || day>31) return false;

break;

case 9:

if (day < 1 || day>30) return false;

break;

case 10:

if (day < 1 || day>31) return false;

break;

case 11:

if (day < 1 || day>30) return false;

break;

case 12:

if (day < 1 || day>31) return false;

break;

}

return true;

}

/\*Фнукция отображения курсора\*/

void cursor\_off\_on(bool inf)

{

void\* handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO structCursorInfo;

GetConsoleCursorInfo(handle, &structCursorInfo);

structCursorInfo.bVisible = inf;

SetConsoleCursorInfo(handle, &structCursorInfo);

}

Код Main.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <conio.h>

#include <cstdlib>

#include <Windows.h>

#include <cctype>

#include <setjmp.h>

#include "zoo.h"

#include "Change.h"

#include "Fileworks.h"

#include "Input.h"

#include "Menu.h"

#include "Output.h"

#include "Processing.h"

#include "Queue.h"

#include "Search.h"

#include "Sort.h"

/\*Главная функция\*/

int main()

{

setlocale(LC\_ALL, "rus");

HANDLE out\_handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD maxWindow; // размер самого большого возможного консольного окна

maxWindow.X = 150;

maxWindow.Y = 35;

SMALL\_RECT srctWindow = { 0, 0, maxWindow.X - 1, maxWindow.Y - 1 };

SMALL\_RECT minWindow = { 0, 0, 0, 0 };

SetConsoleWindowInfo(out\_handle, true, &minWindow);

SetConsoleScreenBufferSize(out\_handle, maxWindow);

SetConsoleWindowInfo(out\_handle, true, &srctWindow);

system("title Система работы с зоопарком");

GWL\_STYLE;

start\_menu();

zoo\* beg = 0,//начало и

\* end = 0;//конец равны нулю

string filename = "", ques = "",

hello[] = {

"Использовать существующий файл", //0

"Создать новый список", //1

},

yes\_no[] = {

"Да",

"Нет"

},

add\_new[]={

"Добавить к списку",

"Заменить список"

},

type[] = {

"txt",//0

"data" //1

},

menuElems[] = {

"Создание списка", //0

"Добавление записи", //1

"Удаление записи", //2

"Редактирование записи", //3

"Сортировка списка", //4

"Поиск записи", //5

"Просмотр списка",//6

"Сохранение списка в файл" ,//7

"Чтение списка из файла" ,//8

"Подсчёт суммы и веса употреблённых продуктов всеми видами пород животных",//9

"Выход из программы" ,//10

},

delete\_e[] = {

"Удалить весь список.",

"Удалить конкретный элемент."

};

int count\_input = 0,count\_id\_del=0,count\_id\_change=0;

jmp\_buf menu\_choice;

cursor\_off\_on(FALSE);

switch (menu(hello, sizeof(hello), "Для взаимодействия с программой необходимо использовать или создать список.\nВыберите пункт меню"))

{

case 0:

cin.clear();

system("cls");

cursor\_off\_on(TRUE);

filename = inp\_filename(filename, "Введите имя файла , который хотите открыть(без расширения)");

cursor\_off\_on(FALSE);

ques = "Какое расширение у файла '" + filename + "' ?";

switch (menu(type, sizeof(type),ques))

{

case 0:

filename += ".txt";

break;

case 1:

filename += ".data";

break;

}

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

read\_file(filename, &beg, &end);

system("pause");

break;

case 1:

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

cout << "Был создан новый список" << endl;

beg = end = 0;

system("pause");

break;

}

do

{

setjmp(menu\_choice);

cursor\_off\_on(FALSE);

switch (menu(menuElems, sizeof(menuElems), "\t\t\tСистема работы с зоопарком\n===============================================================================\n\tИспользуйте стрелочки для перемещения и Enter для выбора пункта меню\n==============================================================================="))

{

case 0:

{

system("cls");

if (beg != 0)

{

switch (menu\_category(yes\_no, sizeof(yes\_no), "Для создание нового списка требуется удалить старый . Вы хотите сохранить старый список ?\n Вы можете нажать ESC для выхода в меню", menu\_choice))

{

SetConsoleTextAttribute(out\_handle, (WORD)((Black << 4) | Yellow));

case 0:

cursor\_off\_on(TRUE);

cin.ignore();

filename = inp\_filename(filename, "Введите имя файла , который хотите сохранить(без расширения)");

cursor\_off\_on(FALSE);

switch (menu(type, sizeof(type), "В каком расширении вы хотите сохранить файл ?"))

{

case 0:

filename += ".txt";

break;

case 1:

filename += ".data";

break;

}

SetConsoleTextAttribute(out\_handle, (WORD)((Black << 4) | Yellow));

write\_in\_file(filename, beg);

cin.get();

dellete\_all(beg);

beg = end = 0;

break;

case 1:

SetConsoleTextAttribute(out\_handle, (WORD)((Black << 4) | Yellow));

dellete\_all(beg);

beg = end = 0;

break;

}

}

else

{

beg = end = 0;

}

cout << "Был создан новый список" << endl;

system("pause");

break;

}

case 1:

{

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

system("cls");

cursor\_off\_on(TRUE);

if (beg)

{

if (filename != "" && count\_input == 0) { cin.ignore(); count\_input++; }

end = add(end, input());

}//если существует хотя бы один элемент- добавляем новый

else

{

beg = add\_first(input());//если не существует ни одного элемента - создаём первый

end = beg;//приравниваем конец к началу

}

system("cls");

cursor\_off\_on(FALSE);

cout << "Вы создали следующий элемент : " << endl;

header();

print\_on\_the\_screen(\*end);

cout << endl;

system("pause");

break;

cin.clear();

}

case 2:

{

system("cls");

if (!beg) { cout << "Невозможно выполнить операцию ! Очередь пуста!" << endl; }

else

{

switch (menu\_category(delete\_e, sizeof(delete\_e), "Вы хотите удалить весь список или конкретный элемент ?\n Вы можете нажать ESC для выхода в меню",menu\_choice))

{

case 0:

{

system("cls");

switch (menu(yes\_no, sizeof(yes\_no), "Вы действительно хотите удалить все элементы ?"))

{

case 0:

dellete\_all(beg);

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

cout << "Все элементы были удалены" << endl;

beg = end = 0;

break;

case 1:

break;

}

break;

}

case 1:

{

cursor\_off\_on(TRUE);

system("cls");

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

unsigned int num = 0;

num = inp\_int(num, "Введите ID элемента");

cursor\_off\_on(FALSE);

if (num<1 || num>end->id)

{

cout << "Невозможно выполнить удаление!" << endl;

cout << "Введенноё ID не находится в диапазоне от " << beg->id << " до " << end->id << endl;

break;

}

else

{

cout << "Выбранный вами элемент :" << endl;

search\_id(beg, num);

system("cls");

switch (menu(yes\_no, sizeof(yes\_no), "Вы действительно хотите удалить этот элемент ?"))

{

case 0:

if (num == beg->id) { beg = dellete\_id\_beg(beg); }

else

{

if (num == end->id) { end = dellete\_id\_end(beg, end); }

else { dellete\_id(beg, end, num); }

}

break;

case 1:

break;

}

}

cursor\_off\_on(FALSE);

break;

}

}

}

system("pause");

break;

}

case 3:

{

system("cls");

if (beg)

{

unsigned int num = 0;

cursor\_off\_on(TRUE);

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

if (filename != "" && count\_id\_change == 0) { cin.ignore(); count\_id\_change++; }

num = inp\_int(num, "Введите ID элемента");

cursor\_off\_on(FALSE);

if (num<1 || num>end->id)

{

cout << "Невозможно выполнить изменение!" << endl;

cout << "Введенноё ID не находится в диапазоне от " << beg->id << " до " << end->id << endl;

}

else

{

cout << "Выбранный вами элемент :" << endl;

search\_id(beg, num);

system("cls");

switch (menu(yes\_no, sizeof(yes\_no), "Вы действительно хотите изменить этот элемент ?"))

{

case 0:

change(beg, end, num);

break;

case 1:

break;

}

}

}

else { cout << "Невозможно выполнить операцию ! Очередь пуста!" << endl; }

system("pause");

break;

}

case 4:

{

system("cls");

sort\_by\_field(beg,menu\_choice);

system("pause");

break;

}

case 5:

{

system("cls");

search(beg,menu\_choice);

system("pause");

break;

}

case 6:

{

system("cls");

see\_the\_information(beg,end);

break;

}

case 7:

{

system("cls");

cursor\_off\_on(TRUE);

ques = "";

if (!filename.empty()) cin.ignore();

filename = inp\_filename(filename, "Введите имя файла , который хотите сохранить(без расширения)");

cursor\_off\_on(FALSE);

ques = "В каком расширении вы хотите сохранить файл '" + filename + "' ?";

switch (menu(type, sizeof(type), ques))

{

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

case 0:

filename += ".txt";

break;

case 1:

filename += ".data";

break;

}

write\_in\_file(filename, beg);

system("pause");

break;

}

case 8:

{

system("cls");

cursor\_off\_on(TRUE);

ques = "";

filename = inp\_filename(filename, "Введите имя файла , который хотите открыть(без расширения)");

cursor\_off\_on(FALSE);

ques = "В каком расширении вы хотите открыть файл '" + filename + "' ?";

switch (menu(type, sizeof(type), ques))

{

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

case 0:

filename += ".txt";

break;

case 1:

filename += ".data";

break;

}

if (beg)

{

switch (menu(add\_new, sizeof(add\_new), "Вы хотите добавить список из файла к текущему списку или заменить его ?"))

{

case 0:

cout << "Данные будут добавлены к текущему списку." << endl;

break;

case 1:

cout << "Текущий список будет заменён." << endl;

dellete\_all(beg);

beg = end = 0;

break;

}

}

read\_file(filename, &beg, &end);

system("pause");

break;

}

case 9:

{

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

system("cls");

if (beg) processing(beg);

else cout << "Невозможно выполнить операцию ! Очередь пуста!" << endl;

system("pause");

break;

}

case 10:

{

system("cls");

if (beg)

{

switch (menu(yes\_no, sizeof(yes\_no), "Вы хотите сохранить файл перед выходом ?"))

{

case 0:

{

cursor\_off\_on(TRUE);

if (beg) { cin.ignore(); }

ques = "";

filename = inp\_filename(filename, "Введите имя файла , который хотите сохранить(без расширения)");

cursor\_off\_on(FALSE);

ques = "В каком расширении вы хотите сохранить файл '" + filename + "' ?";

switch (menu(type, sizeof(type), ques))

{

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

case 0:

filename += ".txt";

break;

case 1:

filename += ".data";

break;

}

write\_in\_file(filename, beg);

system("pause");

break;

}

case 1:

{

break;

}

}

}

SetConsoleTextAttribute(out\_handle, (WORD)((White << 4) | Blue));

cout << "До свидания" << endl;

system("pause");

system("cls");

return 0;

}

}

} while (1);

}

Код Menu.cpp

#include <iostream>

#include <string>

#include <conio.h>

#include <Windows.h>

#include <csetjmp>

#include "zoo.h"

#include "Input.h"

using namespace std;

HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO csbInfo;

COORD coord;

void GetBufferInfo(COORD\* coord)

{

GetConsoleScreenBufferInfo(consoleHandle, &csbInfo);

coord->X = (csbInfo.srWindow.Right - csbInfo.srWindow.Left + 1);

coord->Y = (csbInfo.srWindow.Bottom - csbInfo.srWindow.Top + 1);

}

void goto\_x\_y(short x, short y)

{

SetConsoleCursorPosition(consoleHandle, { x, y });

}

int menu(string menuElems[], int size, string inf)

{

COORD coord1;

GetBufferInfo(&coord);

GetBufferInfo(&coord1);

int pointer = 0;

int code;

while (true)

{

coord1 = coord;

SetConsoleTextAttribute(consoleHandle, (WORD)((White << 4) | Blue));

system("cls");

cout << inf << endl;

for (int i = 0; i < size / sizeof(menuElems[0]); i++)

{

if (pointer == i)

{

SetConsoleTextAttribute(consoleHandle, (WORD)((White << 4) | Black));

cout << "-->";

}

else

{

cout << " ";

SetConsoleTextAttribute(consoleHandle, (WORD)((White << 4) | Blue));

}

cout << i + 1 << " - " << menuElems[i] << endl;

coord1.Y++;

}

code = \_getch();

switch (code)

{

case Down:

{

pointer++;

if (pointer >= size / sizeof(menuElems[0]))

{

pointer = 0;

}

break;

}

case Up:

{

pointer--;

if (pointer < 0)

{

pointer = size / sizeof(menuElems[0]) - 1;

}

break;

}

case Enter:

{

SetConsoleCursorPosition(consoleHandle, coord1);

return pointer;

}

default:

break;

}

}

}

int menu\_category(string menuElems[], int size, string inf,jmp\_buf env)

{

COORD coord1;

GetBufferInfo(&coord);

GetBufferInfo(&coord1);

int pointer = 0;

int code;

while (true)

{

coord1 = coord;

SetConsoleTextAttribute(consoleHandle, (WORD)((White << 4) | Blue));

system("cls");

cout << inf << endl;

for (int i = 0; i < size / sizeof(menuElems[0]); i++)

{

SetConsoleCursorPosition(consoleHandle, coord1);

if (pointer == i)

{

SetConsoleTextAttribute(consoleHandle, (WORD)((White << 4) | Black));

cout << "-->";

}

else

{

SetConsoleTextAttribute(consoleHandle, (WORD)((White << 4) | Blue));

}

cout<< i + 1 << " - " << menuElems[i] << endl;

coord1.Y++;

}

code = \_getch();

switch (code)

{

case Down:

{

pointer++;

if (pointer >= size / sizeof(menuElems[0]))

{

pointer = 0;

}

break;

}

case Up:

{

pointer--;

if (pointer < 0)

{

pointer = size / sizeof(menuElems[0]) - 1;

}

break;

}

case Enter:

{

SetConsoleCursorPosition(consoleHandle, coord1);

return pointer;

}

case Esc:

{

longjmp(env,0);

}

default:

break;

}

}

}

int start\_menu()

{

cursor\_off\_on(FALSE);

string main\_menu\_buttons[] = { "Начать","О программе","Выход" };

string about[] = { "Программа была разработана для работы с базой данных зоопарка в рамках третьего семестра по дисциплине 'Алгоритмизация и Программирование'\n",

"Полный код программы вы можете найти по ссылке https://github.com/PridetheF1rst/AnP-3-Semestr-Zoo-13.git \n","\t\t\t\tВерсия 1.0" };

start:

switch (menu(main\_menu\_buttons, sizeof(main\_menu\_buttons), "Добро пожаловать в программу работы с зоопарком !\n================================================="))

{

case 0:

return 0;

case 1:

system("cls");

for (int i = 0; i < 3; i++)

{

goto\_x\_y(5 + i \* 17, 15 + i);

cout << about[i];

}

cout << endl;

goto\_x\_y(50, 19);

system("pause");

goto start;

case 2:

system("cls");

SetConsoleTextAttribute(consoleHandle, (WORD)((White << 4) | Blue));

cout << "До свидания" << endl;

system("pause");

exit(0);

}

}

Код Output.cpp

#include <iostream>

#include <conio.h>

#include "zoo.h"

#include "Processing.h"

#include "Output.h"

/\*Просмотр списка\*/

void see\_the\_information(zoo\* beg,zoo\* end)

{

if (!beg) { cout << "Информация отсутствует !" << endl; system("pause"); return; }

else

{

int key;

bool go = false;

int page,count=0;

float page\_end;

if (end->id % print == 0) page\_end = end->id / print;

else page\_end = (end->id / print) + 1;

zoo\* temp = beg;//указатель на начало списка

cout << "Список животных зоопарка:" << endl;

header();

do//пока список не кончился

{

print\_on\_the\_screen(\*temp);

if (temp->next == nullptr)

{

if (temp->id>13) cout << endl << "Для перехода на предыдущую страницу нажмите <-";

else cout <<endl<< "\t\t\t\t\t";

cout<<"\t\t\t" << "Страница номер " << page\_end << " из " << page\_end << "\t\t\t\t\t Для выхода нажмите Esc " << endl;

button\_end:

key = \_getch();

switch (key)

{

case Left:

{

int i = 0;

while (temp->prev != nullptr && temp->id != beg->id && temp->prev->id % 13 != 0 )

{

temp = temp->prev;

i++;

}

i = 0;

while (i < 12 && temp->prev != nullptr && temp->id!=beg->id)

{

temp = temp->prev;

i++;

}

system("cls");

cout << "Список животных зоопарка:" << endl;

header();

break;

}

case Esc:

goto end;

default:

goto button\_end;

}

}

else

{

if (temp->next && temp->next->id % print == 1)//если следущий элемент - 14 ,

{

page = temp->id / print;

if (page == page\_end ) go = true;

else

{

if (page == 1) cout << endl << "\t\t\t\t\t\t\t" << "Страница номер " << page << " из " << page\_end << "\t\t\tДля перехода на следующую страницу нажмите ->" << endl;

else cout << endl << "Для перехода на предыдущую страницу нажмите <-\t\t\t" << "Страница номер " << page << " из " << page\_end << "\t\t\tДля перехода на следующую страницу нажмите ->" << endl;

cout << endl << "\t\t\t\t\t\t\tДля выхода нажмите Esc " << endl;

}

button:

key = \_getch();

switch (key)

{

case Right:

{

if (temp->next != nullptr)

{

temp = temp->next;

}

else system("pause"); break;

}

case Left:

{

if (temp->id != 13)

{

int i = 0;

while (i < 2 \* print - 1)

{

temp = temp->prev;

i++;

}

}

else temp = beg;

break;

}

case Esc:

goto end;

default:

goto button;

}

system("cls");

cout << "Список животных зоопарка:" << endl;

header();

}

else temp = temp->next;

}

} while (!go);

cout << endl;

end:

return;

}

}

/\*Заголовок для списка\*/

void header()

{

for (int i = 0; i < 148; i++) cout << "=";

printf("\n|%-3s|", "ID");

printf("%-12s|", "Номер ");

printf("%-15s|", "Кличка");

printf("%-20s|", "Порода");

printf("%-20s|", "Ареал ");

printf("%-15s|", "Тип ");

printf("%-18s|", "Вес");

printf("%-18s|", "Стоимость ");

printf("%-17s|", "Дата");

printf("\n|%-3s|", "");

printf("%-12s|", "вольера");

printf("%-15s|", "животного");

printf("%-20s|", "животного");

printf("%-20s|", "обитания");

printf("%-15s|", "продуктов");

printf("%-18s|", "продуктов");

printf("%-18s|", "продуктов");

printf("%-17s|\n", "(ГГГГ.ММ.ДД)");

for (int i = 0; i < 148; i++) cout << "=";

}

/\*Вывод элемента на экран\*/

void print\_on\_the\_screen(const zoo& z)

{

printf("\n|%-3d|", z.id);

printf("%-12d|", z.cage);

printf("%-15s|", z.name);

printf("%-20s|", z.specie);

printf("%-20s|", z.area);

printf("%-15s|", z.p\_type);

printf("%-18.2f|", z.p\_weight);

printf("%-18.2f|", z.p\_money);

printf("%-17s|\n", z.date);

for (int i = 0; i < 148; i++) cout << "=";

}

/\*Заголовок для обработки\*/

void header\_proccessing()

{

for (int i = 0; i < 79; i++) cout << "=";

printf("\n|%-30s|", "Порода животного");

printf("%-20s|", "Вес продуктов");

printf("%-25s|\n", "Стоимость продуктов");

for (int i = 0; i < 79; i++) cout << "=";

}

/\*Вывод элемента обработки на экран\*/

void print\_processing(const zoo& z)

{

printf("\n|%-30s|", z.specie);

printf("%-20.2f|", z.p\_weight);

printf("%-25.2f|\n", z.p\_money);

for (int i = 0; i < 79; i++) cout << "=";

}

/\*Вывод списка элементов обработки\*/

void show\_processing(zoo\* beg)

{

zoo\* temp = beg;//указатель на начало списка

cout << "Сумма и вес употреблённых продуктов всеми видами пород животных(за указанный период) :" << endl;

header\_proccessing();

while (temp)//пока список не кончился

{

print\_processing(\*temp);

if (temp->next && temp->next->id % print == 1)//если следущий элемент - 13 ,

{

cout << endl;

system("pause");

system("cls");

cout << "Сумма и вес употреблённых продуктов всеми видами пород животных(за указанный период) :" << endl;

header\_proccessing();

}

temp = temp->next;//переход к следующей структуре

}

cout << endl;

printf("%-31s|", " ");

printf("%-20s|", "Суммарный вес");

printf("%-25s|\n", "Суммарная стоимость");

printf("%31s", " ");

for (int i = 0; i < 48; i++) cout << "=";

printf("\n%-31s|", " ");

printf("%-20.2f|", count\_weight(beg));

printf("%-25.2f|\n", count\_money(beg));

printf("%31s", " ");

for (int i = 0; i < 48; i++) cout << "=";

cout << endl;

}

Код Processing.cpp

#include <iostream>

#include <fstream>

#include <Windows.h>

#include "zoo.h"

#include "Input.h"

#include "Queue.h"

#include "Processing.h"

#include "Output.h"

#include "Menu.h"

HANDLE ProcessingHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

using namespace std;

/\*Подсчёт сумму и вес употреблённых продуктов всеми видами пород животных (за указанный период)\*/

void processing(zoo\* beg)

{

zoo\* temp = beg;

zoo\* beg\_p = nullptr, \* end\_p = nullptr;

char time1[n\_d] = "", time2[n\_d] = "";

string filename\_processing;

system("cls");

cursor\_off\_on(TRUE);

cout << "Введите дату начала" << endl;

inp\_date(time1);

cout << "Введите дату окончания" << endl;

inp\_date(time2);

while (strcmp(time1, time2) > 0)

{

system("cls");

cout << "Неверно введена дата! Дата окончания не может быть раньше даты начала !" << endl;

cout << "Введите дату начала" << endl;

inp\_date(time1);

cout << "Введите дату окончания" << endl;

inp\_date(time2);

}

system("cls");

cursor\_off\_on(FALSE);

cout << "Дата начала :";

printf(time1, n\_d);

cout << endl << "Дата окончания :";

printf(time2, n\_d);

cout << endl;

while (temp)

{

if (((strcmp(temp->date, time1) > 0) || strcmp(temp->date, time1) == 0) && ((strcmp(temp->date, time2) < 0) || strcmp(temp->date, time2) == 0))

{

if (!beg\_p)

{

beg\_p = add\_first(\*temp);

end\_p = beg\_p;

}

else

{

int c = 0;

zoo\* temp1 = beg\_p;

while (temp1)

{

if (strcmp(temp->specie, temp1->specie) == 0)

{

temp1->p\_money += temp->p\_money;

temp1->p\_weight += temp->p\_weight;

c = 1;

break;

}

temp1 = temp1->next;

}

if (c == 0)

{

end\_p = add(end\_p, \*temp);

}

}

}

temp = temp->next;

}

if (beg\_p)

{

show\_processing(beg\_p);

cursor\_off\_on(TRUE);

filename\_processing = inp\_filename(filename\_processing, "Введите имя файла , который хотите сохранить(без расширения)");

filename\_processing += ".txt";

cursor\_off\_on(FALSE);

SetConsoleTextAttribute(ProcessingHandle, (WORD)((White << 4) | Blue));

processing\_write\_in\_file(filename\_processing, beg\_p);

}

else cout << "Не найдены элементы списка , которые могут соответствовать указанному периоду!" << endl;

}

/\*Подсчёт всей суммы денег для обработки\*/

float count\_money(zoo\* beg\_p)

{

zoo\* temp = beg\_p;

float count = 0;

while (temp)

{

count += temp->p\_money;

temp = temp->next;

}

return count;

}

/\*Подсчёт всего веса для обработки\*/

float count\_weight(zoo\* beg\_p)

{

zoo\* temp = beg\_p;

float count = 0;

while (temp)

{

count += temp->p\_weight;

temp = temp->next;

}

return count;

}

/\*Запись данных обработки в файл\*/

int processing\_write\_in\_file(string filename, zoo\* temp)

{

float money = count\_money(temp), weight = count\_weight(temp);

ofstream fout(filename);

if (!fout) { cout << "Не могу открыть файл для записи" << endl; return 1; }//если поток

//открыть невозможно - вывести сообщение об ошибке

while (temp)//пока список не закончился

{

fout<<"Порода животного :" << temp->specie << endl;//запись вида животного

fout<<"Дата :" << temp->date << endl;//запись даты

fout<<"Вес продуктов :" << temp->p\_weight << endl;//запись веса продуктов

fout <<"Стоимость продуктов :" << temp->p\_money << endl;//запись суммы денег потраченных на продукты

temp = temp->next;//переход к следующей структуре

fout << endl;

}

fout << endl;

fout <<"Суммарный вес :" << weight << endl;//запись всей массы

fout << "Суммарная стоимость :" << money << endl;//запись всей суммы денег

cout << "Результаты обработки сохранены в файле: " << filename << endl;

return 0;

}

Код Queue.cpp

#include <iostream>

#include "zoo.h"

#include "Input.h"

#include "Output.h"

using namespace std;//использование пространства имён

/\*Функция добавления первого элемента\*/

zoo\* add\_first(const zoo& z)

{

zoo\* beg = new zoo;

\*beg = z;

beg->id = 1;//ключевое поле равно единице на момент начало списка

beg->prev = NULL;

beg->next = 0;

return beg;

}

/\*Функция добавления в уже существующий список\*/

zoo\* add(zoo\* end, const zoo& z)

{

zoo\* newZ = new zoo;//

\*newZ = z;

newZ->prev = end;

newZ->next = 0;

end->next = newZ;

end = newZ;

newZ->id = newZ->prev->id + 1;//увеличение ключевого поля на единицу для новой структуры

return end;

}

/\*Удаление по ключевому полю , если id равно id начала\*/

zoo\* dellete\_id\_beg(zoo\* beg)

{

zoo\* temp;

if (!beg) { cout << "Невозможно выполнить операцию ! Очередь пуста!" << endl; return 0; }

else

{

temp = beg;

cout << "Выполнено удаление элемента с ID = " << beg->id << endl;

header();

print\_on\_the\_screen(\*beg);

cout << endl;

while (temp)

{

(temp->id)--;

temp = temp->next;

}

temp = beg;

beg = beg->next;

delete temp;

return beg;

}

}

/\*Удаление по ключевому полю , если id равно id конца\*/

zoo\* dellete\_id\_end(zoo\* beg, zoo\* end)

{

if (!beg) { cout << "Невозможно выполнить операцию ! Очередь пуста!" << endl; return 0; }

zoo\* temp;

temp = end->prev;

zoo\* temp\_1 = end;

temp->next = end->next;

end = temp;

cout << "Выполнено удаление элемента с ID = " << temp\_1->id << endl;

header();

print\_on\_the\_screen(\*temp\_1);

cout << endl;

delete temp\_1;

return end;

}

/\*Удаление по ключевому полю\*/

zoo\* dellete\_id(zoo\* beg, zoo\* end, int num)

{

zoo\* temp;

temp = beg;

if (!beg) { cout << "Невозможно выполнить операцию ! Очередь пуста!" << endl; return 0; }

else

{

while (temp)

{

if (temp->next->id == num)

{

zoo\* temp\_1 = temp->next;

temp->next = temp\_1->next;

temp\_1->next->prev = temp;

cout << "Выполнено удаление элемента с ID = " << num << endl;

header();

print\_on\_the\_screen(\*temp\_1);

cout << endl;

delete temp\_1;

temp\_1 = temp->next;

while (temp\_1)

{

(temp\_1->id)--;

temp\_1 = temp\_1->next;

}

break;

}

else temp = temp->next;

}

}

}

/\*Удаление всего списка\*/

void dellete\_all(zoo\* beg)

{

zoo\* temp = beg;

while (temp)

{

zoo\* temp\_1 = temp->next;

delete temp;

temp = temp\_1;

}

}

Код Search.cpp

#include <iostream>

#include <string>

#include <cstdlib>

#include <Windows.h>

#include "zoo.h"

#include "Queue.h"

#include "Input.h"

#include "Output.h"

#include "Menu.h"

#include "Search.h"

HANDLE SearchHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

/\*Функция поиска , объединяющая остальные подфункции\*/

void search(zoo\* beg,jmp\_buf env)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

if (beg)

{

int number = 0;

char name[n\_n], specie[n\_s],

area[n\_a], p\_type[n\_p], data[n\_d];

float weight = 0, money = 0;

strcpy(name, "");

strcpy(specie, "");

strcpy(area, "");

strcpy(p\_type, "");

strcpy(data, "");

string how\_searh[] = {

"Номер вольера",

"Кличка животного",

"Порода животного",

"Ареал обитания",

"Тип продуктов",

"Вес продуктов (в килограммах)",

"Стоимоисть потребляемых продуктов (в рублях)",

"Дату"

};

switch (menu\_category(how\_searh, sizeof(how\_searh), "По какому параметру вы хотите найти ? \n Вы можете нажать ESC для выхода в меню",env))

{

SetConsoleTextAttribute(SearchHandle, (WORD)((White << 4) | Blue));

case 0:

system("cls");

cursor\_off\_on(TRUE);

number = inp\_int(number, "Введите номер вольера :");

search\_cage(beg, number);

break;

case 1:

system("cls");

cursor\_off\_on(TRUE);

inp\_char(name, "Введите кличку животного :", n\_n);

search\_name(beg, name);

break;

case 2:

system("cls");

cursor\_off\_on(TRUE);

inp\_char(specie, "Введите породу животного :", n\_s);

search\_specie(beg, specie);

break;

case 3:

system("cls");

cursor\_off\_on(TRUE);

inp\_char(area, "Введите ареал обитания :", n\_a);

search\_area(beg, area);

break;

case 4:

system("cls");

cursor\_off\_on(TRUE);

inp\_char(p\_type, "Введите тип продуктов :", n\_p);

search\_p\_type(beg, p\_type);

break;

case 5:

system("cls");

cursor\_off\_on(TRUE);

weight = inp\_float(weight, "Введите вес продуктов (в килограммах):");

search\_p\_weight(beg, weight);

break;

case 6:

system("cls");

cursor\_off\_on(TRUE);

money = inp\_float(money, "Введите стоимость потребляемых продуктов (в рублях):");

search\_p\_money(beg, money);

break;

case 7:

system("cls");

cursor\_off\_on(TRUE);

inp\_date(data);

search\_date(beg, data);

break;

}

}

else

{

cout << "Невозможно использовать функцию ! Отсутствуют элементы очереди!" << endl;

}

cursor\_off\_on(FALSE);

SetConsoleCP(866);

SetConsoleOutputCP(866);

}

/\*Подфункции поиска\*/

/\*Поиск по номеру вольера\*/

void search\_cage(zoo\* beg, int num\_cage)

{

int count = 0;

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (temp->cage == num\_cage)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << num\_cage << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << num\_cage << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

/\*Поиск по кличке\*/

void search\_name(zoo\* beg, char\* pet\_name)

{

int count = 0;

system("cls");

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (strcmp(temp->name, pet\_name) == 0)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << pet\_name << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << pet\_name << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

/\*Поиск по породе\*/

void search\_specie(zoo\* beg, char\* pet\_specie)

{

int count = 0;

system("cls");

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (strcmp(temp->specie, pet\_specie) == 0)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << pet\_specie << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << pet\_specie << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

/\*Поиск по ареалу обитания\*/

void search\_area(zoo\* beg, char\* pet\_area)

{

int count = 0;

system("cls");

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (strcmp(temp->area, pet\_area) == 0)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << pet\_area << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << pet\_area << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

/\*Поиск по типу продуктов\*/

void search\_p\_type(zoo\* beg, char\* p\_type)

{

int count = 0;

system("cls");

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (strcmp(temp->p\_type, p\_type) == 0)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << p\_type << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << p\_type << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

/\*Поиск по весу\*/

void search\_p\_weight(zoo\* beg, float p\_weight)

{

int count = 0;

system("cls");

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (temp->p\_weight == p\_weight)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << p\_weight << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << p\_weight << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

/\*Поиск по сумме\*/

void search\_p\_money(zoo\* beg, float sum\_money)

{

int count = 0;

system("cls");

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (temp->p\_money == sum\_money)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << sum\_money << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << sum\_money << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

/\*Поиск по дате\*/

void search\_date(zoo\* beg, char\* data)

{

int count = 0;

system("cls");

zoo\* temp = beg, \* beg\_s = nullptr, \* end\_s = nullptr;

while (temp)

{

if (strcmp(temp->date, data) == 0)

{

if (!beg\_s)

{

beg\_s = add\_first(\*temp);

beg\_s->id = temp->id;

end\_s = beg\_s;

}

else { end\_s = add(end\_s, \*temp); end\_s->id = temp->id; }

}

temp = temp->next;

}

if (!beg\_s) { cout << "Информация , соответвующая запросу отсутствует !" << endl; }

else

{

cout << "Информация , соответвующая запросу (" << data << "):" << endl;

header();

temp = beg\_s;

while (temp)

{

if (count != 1 && (count % print) == 1)

{

cout << endl;

system("pause");

system("cls");

cout << "Информация , соответвующая запросу (" << data << "):" << endl;

header();

print\_on\_the\_screen(\*temp);

count++;

}

else

{

count++;

print\_on\_the\_screen(\*temp);

}

temp = temp->next;

}

}

cout << endl;

}

void search\_id(zoo\* beg, int id)

{

zoo\* temp = beg;

while (temp)

{

if (temp->id == id) { header(); print\_on\_the\_screen(\*temp); cout << endl; system("pause"); break; }

else temp = temp->next;

}

}

Код Sort.cpp

#include <iostream>

#include <Windows.h>

#include "zoo.h"

#include "Menu.h"

#include "Sort.h"

HANDLE sortHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

/\*Функция сортировки , объединяющая остальные подфункции\*/

void sort\_by\_field(zoo\* beg,jmp\_buf env)

{

if (beg)

{

int how\_sort;

string \_sort[] = { " По возрастанию " ," По убыванию " };

string field[] = {

" Номер вольера",

"Кличку животного" ,

"Породу" ,

"Ареал обитания",

"Тип продуктов" ,

"Вес продуктов (в килограммах)" ,

"Стоимоисть потребляемых продуктов (в рублях)" ,

"Дату"

};

how\_sort = menu\_category(\_sort, sizeof(\_sort), "Какую сортировку вы хотите выбрать ?\n Вы можете нажать ESC для выхода в меню",env);

SetConsoleTextAttribute(sortHandle, (WORD)((White << 4) | Blue));

zoo\* temp\_1, temp\_2;

switch (menu(field, sizeof(field), "По какому параметру вы хотите отсортировать ? "))

{

SetConsoleTextAttribute(sortHandle, (WORD)((White << 4) | Blue));

case 0:

system("cls");

sort\_cage(beg, how\_sort);

break;

case 1:

system("cls");

sort\_name(beg, how\_sort);

break;

case 2:

system("cls");

sort\_specie(beg, how\_sort);

break;

case 3:

system("cls");

sort\_area(beg, how\_sort);

break;

case 4:

system("cls");

sort\_p\_type(beg, how\_sort);

break;

case 5:

system("cls");

sort\_p\_weight(beg, how\_sort);

break;

case 6:

system("cls");

sort\_p\_money(beg, how\_sort);

break;

case 7:

system("cls");

sort\_date(beg, how\_sort);

break;

}

cout << "Сортировка была выполнена !" << endl;

}

else

{

cout << "Невозможно использовать функцию ! Отсутствуют элементы очереди!" << endl;

}

}

/\*Подфункции сортировки\*/

/\*Смена местами при сортировке\*/

void sort\_replace(zoo\* left, zoo\* right, zoo\* temp)

{

temp->cage = left->cage;

left->cage = right->cage;

right->cage = temp->cage;

strcpy(temp->name, left->name);

strcpy(left->name, right->name);

strcpy(right->name, temp->name);

strcpy(temp->specie, left->specie);

strcpy(left->specie, right->specie);

strcpy(right->specie, temp->specie);

strcpy(temp->area, left->area);

strcpy(left->area, right->area);

strcpy(right->area, temp->area);

strcpy(temp->p\_type, left->p\_type);

strcpy(left->p\_type, right->p\_type);

strcpy(right->p\_type, temp->p\_type);

temp->p\_weight = left->p\_weight;

left->p\_weight = right->p\_weight;

right->p\_weight = temp->p\_weight;

temp->p\_money = left->p\_money;

left->p\_money = right->p\_money;

right->p\_money = temp->p\_money;

strcpy(temp->date, left->date);

strcpy(left->date, right->date);

strcpy(right->date, temp->date);

}

/\*Сортировка по номеру вольера\*/

void sort\_cage(zoo\* beg, int how\_sort)

{

zoo\* left = beg,\* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((left->cage) > (right->cage))

{

sort\_replace(left, right, temp);

}

break;

case 1:

if ((left->cage) < (right->cage))

{

sort\_replace(left, right, temp);

}

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}

/\*Сортировка по кличке\*/

void sort\_name(zoo\* beg, int how\_sort)

{

zoo\* left = beg, \* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((strcmp(left->name, right->name)) > 0)

{

sort\_replace(left, right, temp);

}

break;

case 1:

if ((strcmp(left->name, right->name)) < 0)

{

sort\_replace(left, right, temp);

}

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}

/\*Сортировка по породе\*/

void sort\_specie(zoo\* beg, int how\_sort)

{

zoo\* left = beg, \* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((strcmp(left->specie, right->specie)) > 0)

{

sort\_replace(left, right, temp);

}

break;

case 1:

if ((strcmp(left->specie, right->specie)) < 0)

{

sort\_replace(left, right, temp);

}

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}

/\*Сортировка по ареалу обитания\*/

void sort\_area(zoo\* beg, int how\_sort)

{

zoo\* left = beg, \* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((strcmp(left->area, right->area)) > 0)

{

sort\_replace(left, right, temp);

}

break;

case 1:

if ((strcmp(left->area, right->area)) < 0)

{

sort\_replace(left, right, temp);

}

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}

/\*Сортировка по типу продуктов\*/

void sort\_p\_type(zoo\* beg, int how\_sort)

{

zoo\* left = beg, \* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((strcmp(left->p\_type, right->p\_type)) > 0)

{

sort\_replace(left, right, temp);

}

break;

case 1:

if ((strcmp(left->p\_type, right->p\_type)) < 0)

{

sort\_replace(left, right, temp);

}

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}

/\*Сортировка по весу продуктов\*/

void sort\_p\_weight(zoo\* beg, int how\_sort)

{

zoo\* left = beg, \* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((left->p\_weight) > (right->p\_weight))

{

sort\_replace(left, right, temp);

}

break;

case 1:

if ((left->p\_weight) < (right->p\_weight))

{

sort\_replace(left, right, temp);

}

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}

/\*Сортировка по количеству затраченнных денег на продукты\*/

void sort\_p\_money(zoo\* beg, int how\_sort)

{

zoo\* left = beg, \* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((left->p\_money) > (right->p\_money))

{

sort\_replace(left, right, temp);

}

break;

case 1:

if ((left->p\_money) < (right->p\_money))

{

sort\_replace(left, right, temp);

}

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}

/\*Сортировка по дате\*/

void sort\_date(zoo\* beg, int how\_sort)

{

zoo\* left = beg, \* right = beg->next, \* temp = new zoo;

while (left->next)

{

while (right)

{

switch (how\_sort)

{

case 0:

if ((strcmp(left->date, right->date)) > 0) { sort\_replace(left, right, temp); }

break;

case 1:

if ((strcmp(left->date, right->date)) < 0) { sort\_replace(left, right, temp); }

break;

}

right = right->next;

}

left = left->next;

right = left->next;

}

}