МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт информационных технологий и управления в технических системах

Кафедра/департамент «Информационные системы»

**КУРСОВАЯ РАБОТА / КУРСОВОЙ ПРОЕКТ**

по дисциплине

Алгоритмизация и программирование

на тему «Программа расчёта стоимости и массы употреблённых животными продуктов»

Выполнил: обучающийся

группы: ИС/б-20-2-о

Филозоп А.Н.

« \_\_» \_\_\_\_\_\_\_20 21 г.

Научный руководитель:

«\_\_» \_\_\_\_\_\_\_\_20 21 г.

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

« \_\_» \_\_\_\_\_\_\_\_\_ 20 21 г.

Севастополь 2021

# АННОТАЦИЯ

Пояснительная записка содержит описание программы для работы с табличными данными, а именно, записями о студентах, разработанной в рамках курсового проектирования, целью которого закрепление и углубление знаний в области основ структурного и модульногопрограммирования. Также целью является приобретение практических навыков разработки приложений с использованием wchar\_t-строк, сторонних открытых библиотек и примитивной терминальной графики.

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc91046475)

[1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ 6](#_Toc91046476)

[2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ 7](#_Toc91046477)

[2.1 Постановка задачи на разработку программы 7](#_Toc91046478)

[2.2 Применяемые математические методы 8](#_Toc91046479)

[2.3 Описание и обоснование выбора метода организации входных, выходных и промежуточных данных 8](#_Toc91046480)

[2.4 Обоснование выбора языка и среды программирования 10](#_Toc91046481)

[2.5 Разработка модульной структуры программы 11](#_Toc91046482)

[2.5.1 Дополнительные типы данных 11](#_Toc91046483)

[2.5.2 Дополнительные константы, массивы и перечисления 11](#_Toc91046484)

[2.6 Описание алгоритмов функционирования программы 13](#_Toc91046485)

[2.7 Обоснование состава технических и программных средств 22](#_Toc91046486)

[3 ВЫПОЛНЕНИЕ ПРОГРАММЫ 23](#_Toc91046487)

[3.1 Условия выполнения программы 23](#_Toc91046488)

[3.2 Загрузка и запуск программы 23](#_Toc91046489)

[3.3 Проверка работоспособности программы 26](#_Toc91046490)

[ЗАКЛЮЧЕНИЕ 32](#_Toc91046491)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 33](#_Toc91046492)

[ПРИЛОЖЕНИЕ А 35](#_Toc91046493)

[ПРИЛОЖЕНИЕ Б 46](#_Toc91046494)

# ВВЕДЕНИЕ

В рамках настоящего курсового проектирования ведется разработка программы по теме «Программа обработки информации о потреблении продуктов животными зоопарка» на основании документа – техническое задание – и в рамках организации – Севастопольский государственный университет. Дата выдачи задания: 07.09.2020.

С появлением и широким распространением табличных процессоров, одним из самых известных представителей которых является Microsoft Exel, и иных программ, позволяющих обрабатывать большие объёмы табличных данных без необходимости знания пользователем языков программирования, написание узкоспециализированных программ, примером которых является разрабатываемая программа, утратило свою актуальность, что, тем не менее, никак не сказалось на возможности использования разработки программы, преимуществом которой является простота и крайне малое потребление ресурсов компьютера.

Целью курсового проектирования является систематизация, закрепление и углубление знаний в области основ процедурного программирования и совершенствование практических навыков разработки программ на языке Си на примере разработки программы «расчёта стоимости и массы употреблённых животными продуктов», представляющей собой упрощённое подобие базы данных и позволяющей выполнять различные операции над записями.

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;

Также должны быть разработаны программные документы в соответствии с действующими стандартами.

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ

Программа предназначена для организации, хранения, поиска и модификации данных о потреблённых продуктов животными через пользователь-ориентированный терминальный интерфейс.

Области применения программы: парки и зоопарки, ветеринарные клиники и центры, находящиеся в городе Севастополь и в его округе. Одной из возможностей программы является импорт данных, сохранённых в формате «CSV», что позволяет работать с данными как из других программ, так и с данными, изначально созданными с помощью табличных процессоров, например, MS Excel и LibreOffice.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ

## Постановка задачи на разработку программы

Был выдан 13 вариант задания.

Входной файл имеет следующую структуру:

* порядковый номер записи;
* номер вольера;
* имя животного;
* порода животного;
* ареал животного;
* тип употребляемого продукта;
* вес употребляемого продукта;
* общая стоимость употреблённого продукта;
* дата употребления животным продукта.

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

* создание и добавление элементов в таблицу;
* просмотр существующих записей;
* поиск элемента/группы элементов по ключевому 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);
* реализовать выполнение требуемого по варианту задания;
* корректное завершение работы программы.

В случае обработки записей по варианту программа должна вывести общую сумму и общий вес употреблённых продуктов всеми видами пород животных за указанный период.

## Применяемые математические методы

В ходе разработки применялись следующие математические макросы и функции:

1. Поиск количества элементов в массиве. Находится как частное всего размера всего массива и размера первого элемента.

#define len(arr) (sizeof(arr) / sizeof((arr)[0]))

1. Получить максимальное число из двух.

#define getmax(a, b) (((a) > (b)) ? (a) : (b))

1. Получить минимальное число из двух.

#define getmin(a, b) (((a) < (b)) ? (a) : (b))

1. Сравнить два числа.

#define cmp\_nums(a, b) (((a) == (b)) ? (0) : (((a) > (b)) ? (1) : (-1)))

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

Для хранения данных в оперативной памяти было принято решение использовать двунаправленный список, потому что двунаправленный список позволяет ускорить поиск требуемых элементов и постраничный вывод данных на экран.

Были определены дополнительные структуры данных.

Структура хранения даты:

typedef struct

{

int Y, // Год

M, // Месяц

D; // День

}

udate\_t;

Структура хранения пользовательских данных:

typedef struct list\_data\_t

{

list\_id\_t cage\_num; // Номер вольера.

struct { // Информация о животном.

wchar\_t areal[ANIMAL\_AREAL\_MAX\_LEN + 1]; // Ареал обитания животного.

wchar\_t breed[ANIMAL\_BREED\_MAX\_LEN + 1]; // Порода животного.

wchar\_t name[ANIMAL\_NAME\_MAX\_LEN + 1]; // Имя животного.

} animal;

struct { // Информация о продуктах.

wchar\_t type[PRODUCT\_TYPE\_MAX\_LEN + 1]; // Тип.

weight\_t weight; // Вес.

cost\_t cost; // Стоимость.

} products;

udate\_t date; // Дата.

}

list\_data\_t;

Элемент списка:

typedef struct list\_elem\_t

{

list\_id\_t id; // Уникальный номер элемента в списке

list\_data\_t data; // Хранимые данные

/\* Указатели на следующий и предыдущий элементы списка \*/

struct list\_elem\_t \*next, \*prev;

}

list\_elem\_t;

Объект списка:

typedef struct list\_obj\_t

{

list\_elem\_t \*head;

length\_t \_length\_;

list\_id\_t \_last\_id\_;

}

list\_obj\_t;

Пункты каждого существующего в программе меню.

const menu\_t MAIN\_MENU = {

L" Начать работу ",

L" О программе ",

L" Выйти из программы ",

NULL

};

const menu\_t SAVE\_OR\_LOAD\_MENU = {

L" Загрузить данные из файла ",

L" Сохранить данные в файл ",

L" Отменить действие ",

NULL

};

const menu\_t FILE\_FORMATS\_MENU = {

L" .bin ",

L" .csv ",

L" Отменить действие ",

NULL

};

const menu\_t LIST\_TYPES\_MENU = {

L" Все данные ",

L" Результат поиска ",

NULL

};

const menu\_t SELECT\_SEARCH\_FIELD\_MENU = {

L" По номеру позиции ",

L" По ID ",

L" По номеру волера ",

L" По ареалу ",

L" По названию породы ",

L" По имени животного ",

L" По типу продукта ",

L" По весу продукта ",

L" По стоимости продукта ",

L" По дате поступления ",

L" Отменить действие ",

NULL

};

const menu\_t SORT\_ORDER\_LIST = {

L" По возрастанию [или A→Z] ",

L" По убыванию [или Z→A] ",

NULL

};

const menu\_t DELETE\_ELEMENT\_MENU = {

L" Не удалять ",

L" Не удалять ",

L" Не удалять ",

L" Удалить ",

L" Не удалять ",

NULL

};

## Обоснование выбора языка и среды программирования

Исходный код программы был написан на языке программирования C, стандарт С99. Причина выбора:

* скорость выполнения функций;
* эффективное потребление памяти;

Для разработки программы была создана следующая среда разработки:

* компилятор: GCC v. 12.0.0. Основные причины выбора: быстрая компиляция программы; отслеживание ошибок в коде;
* текстовый редактор: Microsoft Visual Studio Code. Основные причины выбора: легковесность редактора; возможность установки дополнений из предоставленной библиотеки;
* система сборки проекта: GNU make v.4.3. Причина использования: простая и быстрая сборка проекта, состоящего из множества файлов.
* ситема контроля версий кода: git.

В качестве основной операционной системы была выбрана GNU/Linux Ubuntu 20.04.

## Разработка модульной структуры программы

В основу организации программы был положен принцип событийного управления. До начала разработки было принято решение разделить программу на 3 части:

* главный файл, содержащий основной сценарий выполнения программы;
* ядро программы, содержащее основные функции программы;
* графический интерфейс, содержащий функции отрисовки окон и таблиц на экране.

### Дополнительные типы данных

typedef unsigned short field\_num\_t; // Условный номер поля

typedef unsigned long list\_id\_t; // ID элемента в списке

typedef unsigned long length\_t; // Длина чего-либо

typedef long double weight\_t; // Вес

typedef long double cost\_t; // Стоимость

typedef wchar\_t\* menu\_t[]; // Тип массива из пунктов меню

### Дополнительные константы, массивы и перечисления

Коды полей в структуре данных.

enum DATA\_FIELDS

{

LIST\_POSITION = 0, // Позиция элемента в списке

LIST\_ID, // Id элемента в списке

LIST\_CAGE\_NUMBER, // a.cage\_num

LIST\_ANIMAL\_AREAL, // a.animal.areal

LIST\_ANIMAL\_BREED, // a.animal.breed

LIST\_ANIMAL\_NAME, // a.animal.name

LIST\_PRODUCT\_TYPE, // a.products.type

LIST\_PRODUCT\_WEIGHT, // a.products.weight

LIST\_PRODUCT\_COST, // a.products.cost

LIST\_DATE // a.date

};

Длины строк в структуре данных.

#define ANIMAL\_AREAL\_MAX\_LEN 32 // Максимальная длина Ареал обитания

#define ANIMAL\_BREED\_MAX\_LEN 32 // Максимальная длина Порода

#define ANIMAL\_NAME\_MAX\_LEN 32 // Максимальная длина Имя животного

#define PRODUCT\_TYPE\_MAX\_LEN 32 // Максимальная длина Тип продукта

Длины чисел при выводе на экран.

#define WEIGHT\_DGT\_CNT 7

#define COST\_DGT\_CNT 7

#define CAGE\_NUM\_DGT\_CNT 7

#define POSITION\_DGT\_CNT 4

#define ID\_DGT\_CNT 4

Порядок сортировки элементов в списке.

#define SORT\_DESCENDING\_ORDER (-1)

#define SORT\_ASCENDING\_ORDER (1)

Коды, возвращаемые функциями взаимодействия с файлами.

enum FS\_RETURN {

FS\_SUCCESS = 0, /\* Успешное завершение работы \*/

FS\_FILE\_NO\_READ\_ACCESS, /\* Ошибка: нельзя прочитаться данные из файла \*/

FS\_FILE\_NO\_WRITE\_ACCESS, /\* Ошибка: нельзя записать данные в файл \*/

FS\_NO\_LIST\_PTR, /\* Ошибка: передан пустой указатель на список \*/

FS\_NO\_LIST\_DATA, /\* Ошибка: передан список без данных \*/

FS\_BROKEN\_DATA /\* Ошибка: данные в файле повреждены \*/

};

Коды специальных клавиш.

enum MKEYBOARD\_KEYS

{

MKEY\_NULL = 0,

MKEY\_CTRL\_A = 1,

MKEY\_CTRL\_B,

MKEY\_CTRL\_C,

MKEY\_CTRL\_D,

MKEY\_TAB = 9,

MKEY\_ENTER = 13,

MKEY\_ESCAPE= 27,

MKEY\_SPACE = 32,

MKEY\_ARROW\_UP = 65,

MKEY\_ARROW\_DOWN,

MKEY\_ARROW\_RIGHT,

MKEY\_ARROW\_LEFT,

MKEY\_END = 70,

MKEY\_HOME = 72,

MKEY\_F1 = 80,

MKEY\_F2,

MKEY\_F3,

MKEY\_F4,

MKEY\_F5,

MKEY\_F6,

MKEY\_F7,

MKEY\_F8,

MKEY\_F9,

MKEY\_F10,

MKEY\_F11,

MKEY\_F12,

MKEY\_DELETE = 126,

MKEY\_BACKSPACE

};

## Описание алгоритмов функционирования программы

Функция создания объекта списка. Возвращает указатель на созданный объект.

list\_obj\_t \* lists\_CreateNewListObject (void);

Функция удаления объекта списка и всех связанных с ним элементов из памяти. В указатель на объект списка автоматически записывается NULL.

void lists\_DeleteListObject (list\_obj\_t \*\*);

Функция удаления всех элементов, связанных с указанным объектом списка.

void lists\_CleanListObject (list\_obj\_t \*);

Функция создания элемента списка на основе переданных данных. Вернуть указатель на созданный в памяти элемент.

list\_elem\_t \* lists\_CreateNewElement (list\_data\_t \*);

Функция вставки элемента, переданного первым аргументом, после элемента, переданного вторым элементом.

void lists\_InsertAfterElement (list\_elem\_t \*, list\_elem\_t \*);

Функция безопасного удаления элемента из списка.

void lists\_DeleteElement (list\_elem\_t \*);

Функция безопасного извлечения элемента из списка.

void list\_DetachElement (list\_elem\_t \*);

Функция вставки элемента (второй аргумент) в начало объекта списка (первый аргумент).

void lists\_InsertAsListsHead (list\_obj\_t \*, list\_elem\_t \*);

Функция вставки элемента (второй аргумент) в конец объекта списка (первый аргумент).

void lists\_InsertAsListsTail (list\_obj\_t \*, list\_elem\_t \*);

Функция поиска в объекте списка (первый аргумент) первого элемента, соответствующего указанным критериям (второй и третий аргументы). Возвращает указатель на найденный элемент или NULL, если элемент не был найден.

list\_elem\_t \* lists\_SearchElementByField (list\_obj\_t \*, field\_num\_t, void \*);

Функция поиска в последовательности элементов списка, начиная с текущего (первый аргумент), первого элемента, соответствующего указанным критериям (второй и третий аргументы). Возвращает указатель на найденный элемент или NULL, если элемент не был найден.

list\_elem\_t \* lists\_SearchElementByFieldFromThisElement (list\_elem\_t \*, field\_num\_t, void \*);

Функция сравнения двух элементов списка только по заданному полю (третий аргумент). Вернуть 0, если элементы равны; -1, если первый элемент «меньше» второго, иначе 1.

int lists\_CompareElementsByOneField (const list\_elem\_t \*, const list\_elem\_t \*, const field\_num\_t);

Функция сравнения двух элементов списка. В качестве третьего аргумента передаётся приоритетное поле, по которому будет проводится сравнение в первую очередь. Вернуть 0, если элементы равны; -1, если первый элемент «меньше» второго, иначе 1.

int lists\_CompareElements (const list\_elem\_t \*, const list\_elem\_t \*, const field\_num\_t);

Функция сортировки объекта списка (первый аргумент) по заданному полю (третий аргумент) по возрастанию/убыванию (второй аргумент, определено макросами SORT\_DESCENDING\_ORDER и SORT\_ASCENDING\_ORDER).

void lists\_SortListByField (list\_obj\_t \*, const int, const field\_num\_t);

Функция подсчёта суммы употреблённых животными всех пород продуктов за указанный период. Возвращает подсчитанную сумму.

cost\_t lists\_GetCostForPeriod (const list\_obj\_t \*, const udate\_t \*, const udate\_t \*);

Функция подсчёта веса употреблённых животными всех пород продуктов за указанный период. Возвращает подсчитанный вес.

weight\_t lists\_GetWeightForPeriod (const list\_obj\_t \*, const udate\_t \*, const udate\_t \*);

Функция проверки доступности файла для чтения. Имя файла передаётся в виде массива символов char. Возвращает один из кодов FS\_RETURN.

int is\_exist (const char \*);

Функция проверки доступности файла для чтения. Имя файла передаётся в виде массива символов wchar\_t. Возвращает один из кодов FS\_RETURN.

int is\_exist\_wcs (const wchar\_t \*);

Функция сохранения переданного объекта списка list\_obj\_t в текстовый типизированный файл. Возвращает один из кодов FS\_RETURN.

int save\_as\_bin (const list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs);

Функция чтения данных из текстового типизированного файла. Возвращает один из кодов FS\_RETURN.

int save\_as\_bin (const list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs);

Функция сохранения переданного объекта списка list\_obj\_t в текстовый файл. Возвращает один из кодов FS\_RETURN.

int save\_as\_csv (const list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs);

Функция чтения данных из текстового файла. Возвращает один из кодов FS\_RETURN.

int read\_from\_csv (list\_obj\_t \* list, const wchar\_t \* fileName);

Функция проверки соответствия данного символа заданной маске. Вернуть 0, если символ не подошёл маске, иначе вернуть неотрицательное число.

int check\_wchar (const wint\_t, const checker\_mode\_t);

Функция конвертирования переданной даты (первый аргумент) в строку (второй аргумент). В качестве разделителя использовать указанный символ (третий аргумент). Строку сформировать по указанному шаблону (четвёртый аргумент).

void date2wcsdp (const udate\_t \*, wchar\_t \*, const wchar\_t, const wchar\_t \*);

Для функции выше с целью упрощения кода были определены следующие три макроса:

* Преобразовать дату в строку. Разделитель и шаблон предопределены

#define date2wcs(date, string) date2wcsdp(date, string, L'/', L"%02d%lc%02d%lc%04d")

* Преобразовать дату в строку. Использовать данный разделитель. Шаблон предопределён.

#define date2wcsd(date, string, delimiter) date2wcsdp(date, string, delimiter, L"%02d%lc%02d%lc%04d")

* Преобразовать дату в строку. Использовать данный шаблон. Разделитель предопределён.

#define date2wcsp(date, string, pattern) date2wcsdp(date, string, L'/', pattern)

Функция конвертирования строки (второй аргумент) в дату (первый аргумент).

void wcs2date (udate\_t \*, const wchar\_t \*);

Функция сравнения двух дат. Вернуть 0, если даты равны; -1, если первая дата раньше второй; 1, если первая дата позже второй.

int datecmp (const udate\_t \*, const udate\_t \*);

Функция конвертирования из строки (массив символов wchar\_t) в целое число. Возвращает получившееся число.

long long wcs2int (const wchar\_t \*);

Функция конвертирования из строки (массив символов wchar\_t) в число с плавающей точкой. Возвращает получившееся число.

long double wcs2float (const wchar\_t \*);

Функция преобразования целого числа (первый аргумент) в строку (второй аргумент, массив символов wchar\_t).

void int2wcs (const long long, wchar\_t \*);

Функция преобразования числа с плавающей точкой (первый аргумент) в строку (второй аргумент, массив символов wchar\_t).

void float2wcs (const long double, wchar\_t \*);

Функция изменения фактического размера строки с помощью удаления лишних или добавления указанного заполнителя (второй аргумент) в конец.

void fitwcs (wchar\_t \*, const wchar\_t \*, const int);

Функция подсчёта фактического количества строк в переданном сообщении (первый аргумент), с учётом символов перехода на новую строку, символов табуляции и максимальной ширины окна (второй аргумент). Возвращает подсчитанное количество строк.

int get\_message\_lines\_count (const wchar\_t \*, const int max\_width);

Функция подсчёта количества пунктов меню. Возвращает количество пунктов меню.

int tui\_get\_menu\_item\_count (const menu\_t);

Функция отрисовки вертикального меню.

void tui\_draw\_vmenu (WINDOW \*, const int, const int, const int, const menu\_t, const int);

Функция отображения окна с текстовым сообщением.

void tui\_draw\_popup\_text\_message (const wchar\_t \*, const wchar\_t \*);

Функция отображения окна с пунктами меню. Возвращает порядковый номер выбранного пункта меню.

int tui\_draw\_popup\_select (const wchar\_t \*, const wchar\_t \*, const menu\_t);

Функция отображения окна с полем для ввода сообщения. Возвращает фактическое количество введённых пользователем символов.

int tui\_draw\_popup\_select (const wchar\_t \*, const wchar\_t \*, const menu\_t);

Функция отображения окна с вводом даты.

void tui\_draw\_popup\_date (const wchar\_t \*, const wchar\_t \*, udate\_t \*);

Функция отображения окна с информацией о выбранном элементе таблицы. Редактирование разрешено.

int tui\_popup\_edit\_element\_data (const wchar\_t \*, list\_data\_t \*, int);

Функция отображения окна с информацией о выбранном элементе таблицы. Редактирование запрещено.

void tui\_popup\_show\_only\_element (const wchar\_t \*, const list\_elem\_t \*);

Функция отрисовки заднего фона главного окна.

void draw\_win\_background (WINDOW \*);

Функция подтверждения действия через ввод числового четырёхзначного кода. Возвращает 1, если пользователь отменил действие; 0, если был введён верный код.

int tui\_confirm\_action ();

Рисунок А.1 – структурная схема функции проверки соответствия символа маске.

Поблочное описание:

* блок 1 – если символ принадлежит группе «Символы нижнего регистра»;
* блок 2 – вернуть значение «VMASK\_LOWER\_CHARS»;
* блок 3 – если символ принадлежит группе «Символы верхнего регистра»;
* блок 4 – вернуть значение «VMASK\_UPPER\_CHARS»;
* блок 5 – если символ принадлежит группе «Цифры»;
* блок 6 – вернуть значение «VMASK\_DIGITS»;
* блок 7 – если символ принадлежит группе «Пробельные символы»;
* блок 8 – вернуть значение «VMASK\_SPACES»;
* блок 9 – если символ является запятой или точкой;
* блок 10 – вернуть значение «VMASK\_PUNCTS»;
* блок 11 – если символ двоеточием, слешем, подчёркиванием или тире;
* блок 12 – вернуть значение «VMASK\_ADD\_FS\_CHARS»;
* блок 13 – вернуть значение «0».

Рисунок А.2 – Подпрограммы перевода даты в строку – структурная схема функции перевода даты в строку по заданному шаблону и с использованием заданного разделителя.

Поблочное описание:

* блок 1 – если был передан пустой указатель на структуру с датой;
* блок 2 – использовать встроенную функцию для преобразования строки в дату.

Рисунок А.3 – структурная схема функции перевода строки в дату.

Поблочное описание:

* блок 1 – если передан пустой указатель на структуру с датой;
* блок 2 – присвоить offset значение 1;
* блок 3 – если символ string[offset] является пробельным символом;
* блок 4 – инкрементировать значение offset;
* блок 5 – в каждое поле структуры записать результат преобразования из строки в символ.

Рисунок А.4 – структурная схема функции сравнения двух дат.

Поблочное описание:

* блок 1 – если был передан пустой указатель на одну из структур даты;
* блок 2 – выйти из функции, вернув значение «0»;
* блок 3 – присвоить переменной result значение «0»;
* блок 4 – записать результат сравнения чисел в result и сравнить с нулём;
* блок 5 – записать результат сравнения чисел в result и сравнить с нулём;
* блок 6 – выйти из функции, вернув значение result;
* блок 7 – выйти из функции, вернув результат сравнения двух чисел.

Рисунок А.5 – структурная схема функции поиска элемента в списке по заданному полю с заданным значением.

Поблочное описание:

* блок 1 – если был передан пустой указатель на элемент;
* блок 2 – если полученно неожиданный условный номер поля;
* блок 3 – вернуть значение null;
* блок 4 – присвоить переменным position и exit\_flag значения 1 и 0 соответственно;
* блок 5 – если e существует и не установлен флаг выхода;
* блок 6 – сравнить значения field;
* блок 7 – в флаг выхода записать результат сравнения позиций;
* блок 8 – в флаг выхода записать результат сравнения id;
* блок 9 – в флаг выхода записать результат сравнения номеров вольера;
* блок 10 – в флаг выхода записать результат сравнения ареала обитания;
* блок 11 – в флаг выхода записать результат сравнения пород животных;
* блок 12 – в флаг выхода записать результат сравнения имён животных;
* блок 13 – в флаг выхода записать результат сравнения типов продуктов;
* блок 14 – в флаг выхода записать результат сравнения массы продуктов;
* блок 15 – в флаг выхода записать результат сравнения стоимости продуктов;
* блок 16 – в флаг выхода записать результат сравнения дат;
* блок 17 – если влаг выхода равен 0;
* блок 18 – передвинуть указатель e на следующий элемент;
* блок 19– вернуть значение e.

Рисунок А.6 – структурная схема функции сравнения двух элементов списка по одному полю.

Поблочное описание:

* блок 1 – записать в переменную result значение «0»;
* блок 2 – сравнить значение переменной field;
* блок 3 – записать в result результат сравнение двух чисел;
* блок 4 – записать в result результат сравнение двух чисел;
* блок 5 – записать в result результат сравнение двух строк;
* блок 6 – записать в result результат сравнение двух строк;
* блок 7 – записать в result результат сравнение двух строк;
* блок 8 – записать в result результат сравнение двух строк;
* блок 9 – записать в result результат сравнение двух чисел;
* блок 10 – записать в result результат сравнение двух чисел;
* блок 11 – записать в result результат сравнение двух дат;
* блок 12 – выйти из функции, вернув значение result.

Рисунок А.7 – структурная схема функции сравнения элементов списка по нескольким полям, начиная с приоритетного.

Поблочное описание:

* блок 1 – записать в переменные result и i значения «0» и «list\_id + 1» соответственно;
* блок 2 – если приоритетным полем является поле с позицией элемента в списке;
* блок 3 – выйти из функции, вернув 0;
* блок 4 – записать в переменную result результат сравнения элементов по приоритетному полю;
* блок 5 – если результат сравнения не равен нулю;
* блок 6 – выйти из функции, вернув результат сравнения;
* блок 7 – если result равен нулю и i меньше либо равно list\_date;
* блок 8 – если i не равно условному номеру приоритетного поля;
* блок 9 – записать в result результат сравнения двух элементов списка по полю с порядковым номером i;
* блок 10 – вернуть результат сравнения двух элементов.

Рисунок А.8 – структурные схемы функции сортировки списка по заданному полю.

Поблочное описание:

* блок 1 – если передан пустой указатель на объект списка;
* блок 2 – если размер списка меньше 2;
* блок 3 – объявление и инициализация дополнительных переменных;
* блок 4 – если i меньше размера списка;
* блок 5 – записать в переменную e1 адрес элемента списка с порядковым номером i; увеличить значение i на 1;
* блок 6 – если j меньше длины списка плюс один;
* блок 7 – записать в переменную e2 адрес элемента списка с порядковым номером j;
* блок 8 – если элементы необходимо поменять местами;
* блок 9 – поменять элементы местами;
* блок 10– инкрементировать значение j;
* блок 11 – инкрементировать значение i.

Рисунок А.9 – структурная схема функции создания нового элемента списка.

Поблочное описание:

* блок 1 – если передан пустой указатель на структуру с данными;
* блок 2 – вернуть из функции значение null;
* блок 3 – создать новый элемент списка. Вернуть адрес этого элемента.

Рисунок А.10 – структурная схема функции очистки списка от всех элементов.

Поблочное описание:

* блок 1 – если передан пустой указатель на объект списка;
* блок 2 – если в списке нет элементов;
* блок 3 – присвоить переменной e указатель на первый элемент списка;
* блок 4 – если существует элемент, следующий за текущим;
* блок 5 – удалить текущий элемент и передвинуть указатель e на следующий;
* блок 6 – удалить последний элемент. Подготовить объект списка для следующего использования.

Рисунок А.11 – структурная схема функции вставки элемента после заданного.

Поблочное описание:

* блок 1 – если не был передан указатель на один из элементов;
* блок 2 – за первым элементом следует на второй;
* блок 3 – если за элементом e1 следует какой-то элемент;
* блок 4 – указатель на предыдущий элемент у следующего за e1 элемента теперь указывает на e1;
* блок 5 – пересвязать указатели так, чтобы e2 являлся предшествующим элементом для e1.

Рисунок А.12 – структурная схема функции получения стоимости употреблённых животными продуктов за указанный период.

Поблочное описание:

* блок 1 – если был передан пустой указатель на список;
* блок 2 – вернуть зачение -1.0;
* блок 3 – инициализация дополнительных переменных;
* блок 4 – если текущий элемиент является пустым;
* блок 5 – сравнить дату элемента с заданной датой;
* блок 6 – сравнить дату элемента с заданной датой;
* блок 7 – увеличить сумму sum на стоимость, указанную в элементе;
* блок 8 – переместить указатель текущего элемента на следующий;
* блок 9 – вернуть полученную сумму.

## Обоснование состава технических и программных средств

Для работы программы требуется использование компьютера с дисплеем для вывода данных и клавиатурой для приёма команд от пользователя. Должна быть предустановлена операционная система семейства GNU/Linux с версией ядра не ниже 4.0. Требования относительно жёсткого диска: 144 килобайта для хранения программы, дополнительное пространство для хранения файлов базы данных. Для запуска программы требуется не менее 4 килобайт оперативной памяти для 64 битных систем, а также дополнительная память для обработки базы данных.

# ВЫПОЛНЕНИЕ ПРОГРАММЫ

## Условия выполнения программы

Таблица 3.1 демонстрирует минимально необходимые системные требования для запуска приложения.

Таблица 3.1 – Системные требования

|  |  |
| --- | --- |
| Операционная система | GNU/Linux с версией ядра не ниже 4.0 |
| Процессор | Одноядерный процессор, x86\_64, не менее 1ГГц |
| ОЗУ | Не менее 16 КБ |
| Место на внешнем носителе | Не менее 256 КБ |
| Требования к терминалу | поддержка 256bit цветов;  размеры окна – не менее 107x24 символов;  поддержка кодировки UTF-8.ru\_ru |

Для запуска программы на устройствах под управлением Windows требуется использование программы «Подсистемы Windows для Linux» (WSL).

## Загрузка и запуск программы

Запустить программу можно двумя способами:

1. С помощью проводника (рисунок 3.1). На рисунке продемонстрирован пример запуска программы из проводника thunar 4.16.10.
2. С помощью терминала, введя название программы, как показано на рисунке 3.2.

Рисунок 3.3 показывает главное меню программы. В главном меню пользователю будет предложено выбрать одно из действий: начать работу с данными; прочитать информацию о программе или выйти из программы.

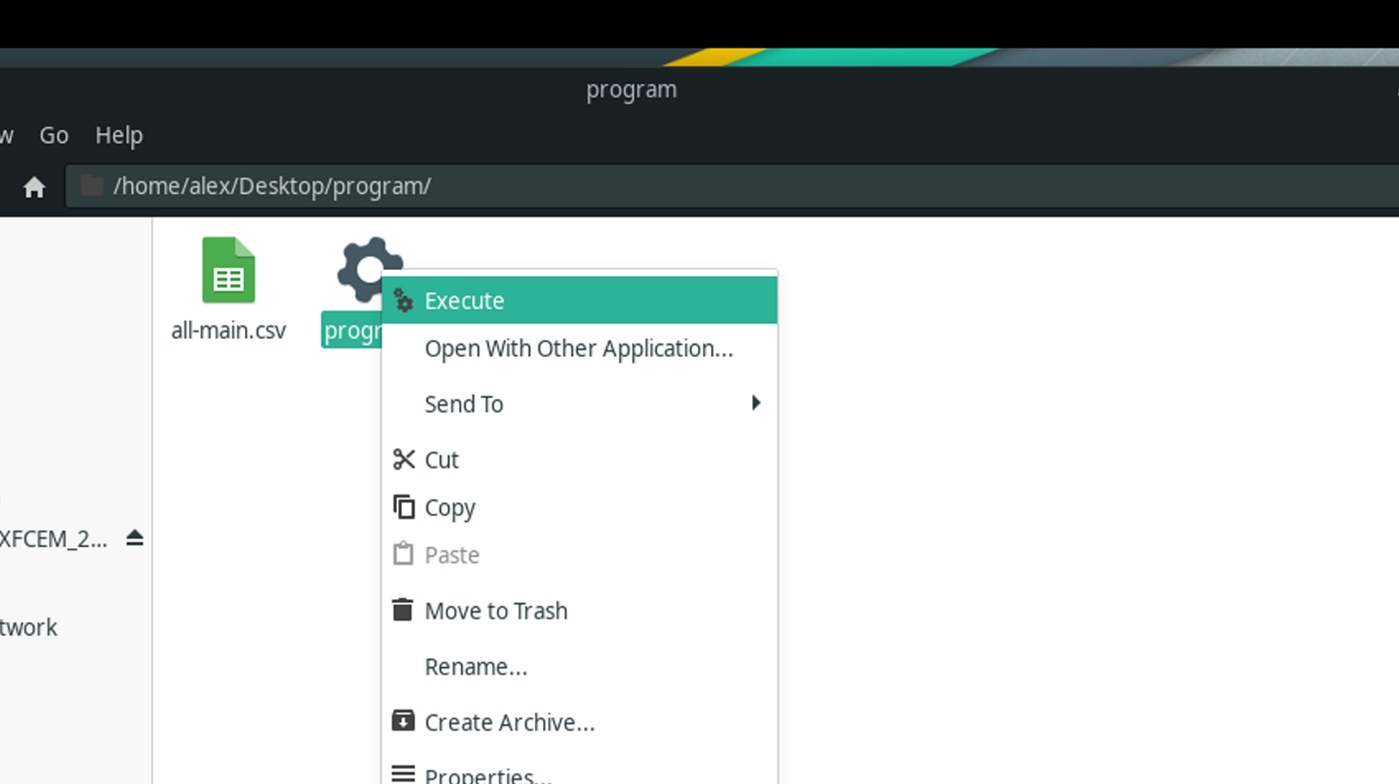


Рисунок 3.1 – Запуск программы через проводник

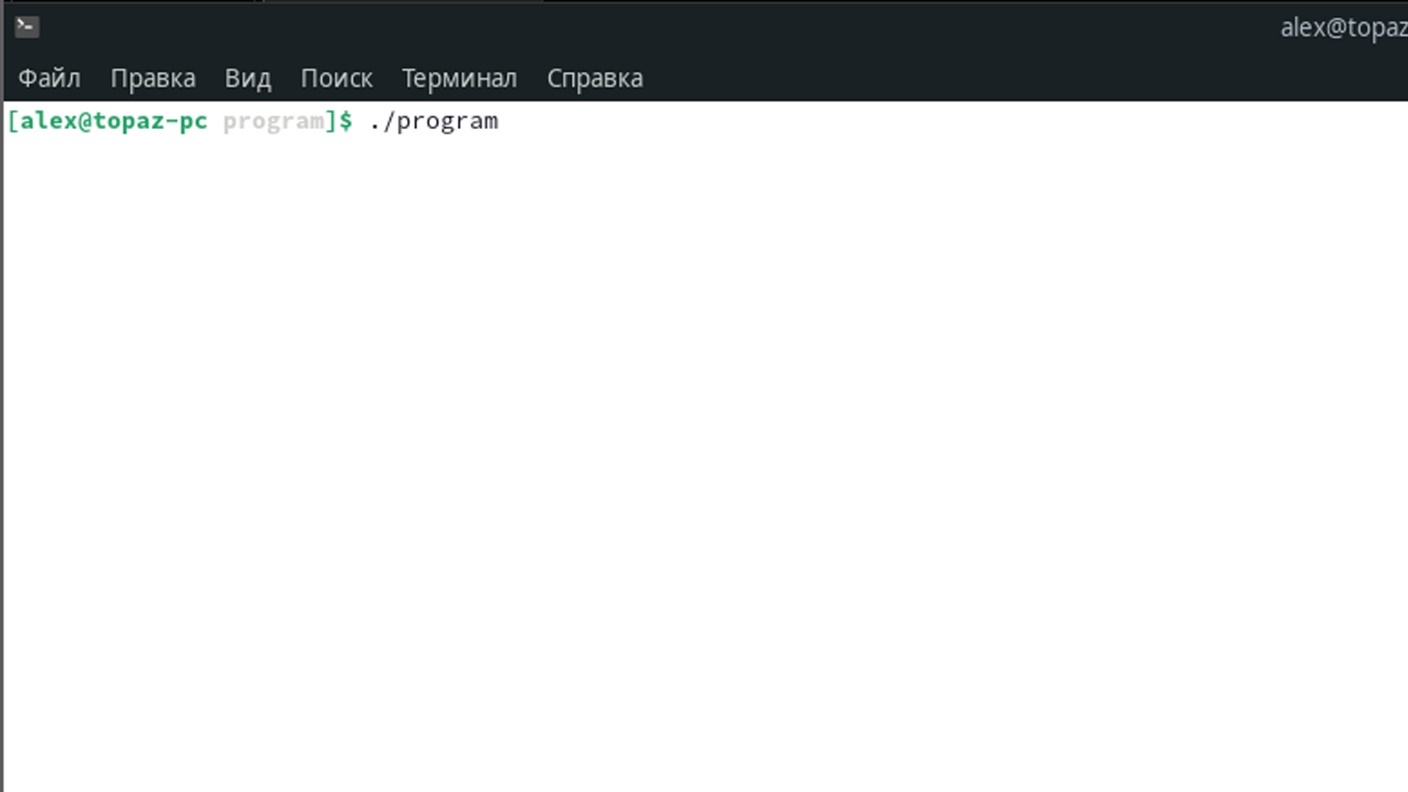


Рисунок 3.2 – Запуск программы через терминал

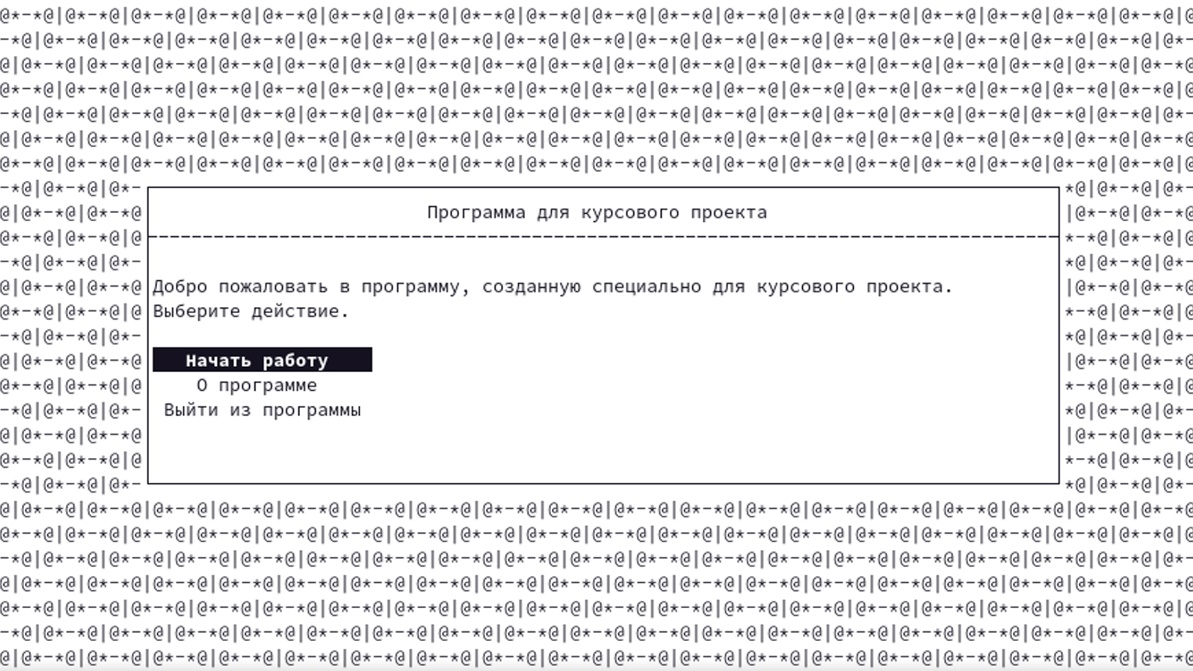


Рисунок 3.3 – Главное меню программы

Все данные отображаются в табличном виде. Для изменения выбранной строчки используются стрелочки вверх/вниз, а для изменения выбранной страницы – стрелочки влево\_вправо. Рисунок 3.4 показывает, что был выбран 7-ой элемент на 2-ой странице.

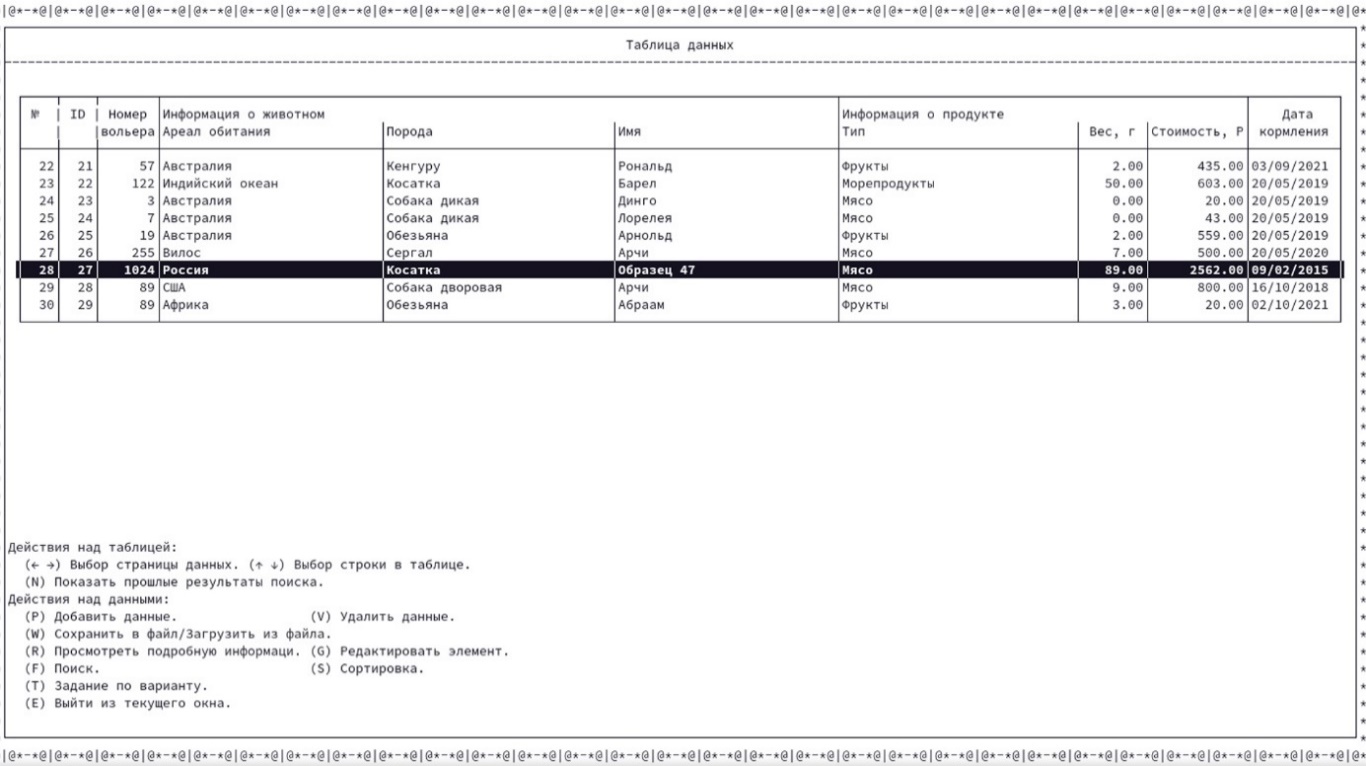


Рисунок 3.4 – Выбор строчки таблицы

## Проверка работоспособности программы

Для проверки работоспособности программы был создан файл в формате «CSV». Содержимое файла представлено ниже:

1;1;Белка;Собака дворовая;Россия;Мясо свинины;1;80.00;02/09/2021

2;2;Стрелка;Собака дворовая;Россия;Мясо свинины;1;80.00;02/09/2021

3;3;Динго;Собака дикая;Австралия;Мясо;0,5;200;02/09/2021

4;120;ДонКиХод;Кит;Индийский океан;Морепродукты;100;5000;02/09/2021

5;254;Нео;Сергал;Вилос;Мясо;10;600;02/09/2021

6;122;Барел;Косатка;Индийский океан;Морепродукты;100;4000;02/09/2021

7;7;Лорелея;Собака дикая;Австралия;Мясо кенгуру;3;900.00;02/09/2021

8;8;Абраам;Обезьяна;Африка;Фрукты;5;100;02/09/2021

9;9;Тузик;Собака дворовая;Россия;Мясо свинины;10.000;100.00;02/09/2021

10;10;Арнольд;Обезьяна;Австралия;Фрукты;9;200;02/09/2021

11;11;Динго;Собака дикая;Австралия;Мясо кенгуру;2;5000.56;02/09/2021

12;122;Барел;Косатка;Индийский океан;Морепродукты;120;5620;03/09/2021

13;120;ДонКиХод;Кит;Индийский океан;Морепродукты;95;4330,90;01/08/2021

14;9;Тузик;Собака дворовая;Россия;Мясо свинины;8,15;100.00;03/09/2021

15;1;Белка;Собака дворовая;Россия;Мясо свинины;6,15;80.00;03/09/2021

16;2;Стрелка;Собака дворовая;Россия;Мясо свинины;10.000;150,36;03/09/2021

17;3;Динго;Собака дикая;Австралия;Мясо;0,5;250;03/09/2021

18;7;Лорелея;Собака дикая;Австралия;Мясо;2,58;503,23;03/09/2021

19;10;Арнольд;Обезьяна;Австралия;Фрукты;7,023;559,25;03/09/2021

20;11;Динго;Собака дикая;Австралия;Мясо;2;5000.56;03/09/2021

21;56;Малыш;Кенгуру;Австралия;Фрукты;2,58;422,32;03/09/2021

22;57;Рональд;Кенгуру;Австралия;Фрукты;2,495;435,6;03/09/2021

23;122;Барел;Косатка;Индийский океан;Морепродукты;50,2;603;20/05/2019

24;3;Динго;Собака дикая;Австралия;Мясо;0,2;20,3;20/05/2019

25;7;Лорелея;Собака дикая;Австралия;Мясо;0,4;43,02;20/05/2019

26;19;Арнольд;Обезьяна;Австралия;Фрукты;2,03;559,25;20/05/2019

27;255;Арчи;Сергал;Вилос;Мясо;7,202;500,3;20/05/2020

28;1024;Образец 47;Косатка;Россия;Мясо;89,2;2562,8;09/02/2015

29;89;Арчи;Собака дворовая;США;Мясо;9,2;800,2;16/10/2018

30;89;Абраам;Обезьяна;Африка;Фрукты;3,28;20;02/10/2021

Рисунок 3.5 и 3.5 демонстрируют процесс импорта данных из файла в формате «CSV».



Рисунок 3.5 – Импорт данных. Выбор типа файла

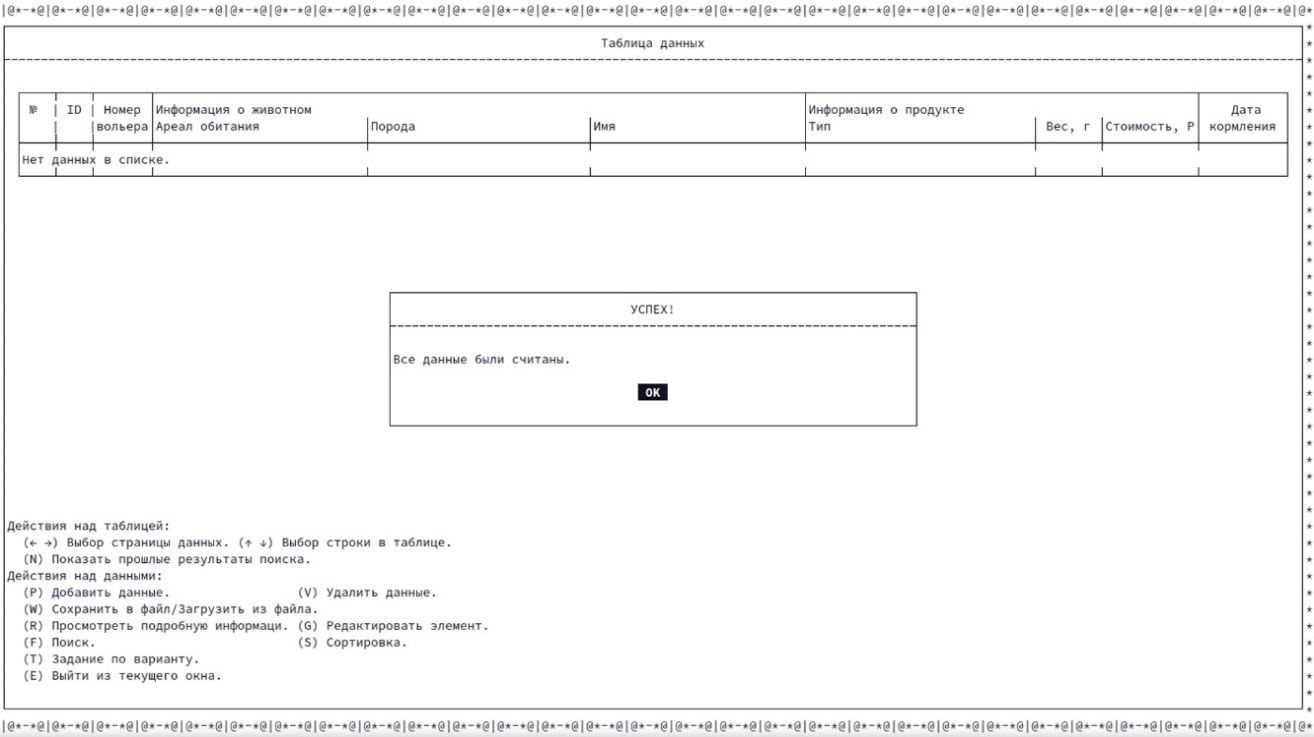


Рисунок 3.6 – Импорт данных. Завершение импортирования

По нажатию на клавиатуре кнопки «G» было открыто окно редактирования элемента (рисунок 3.7). После редактирования по нажатию кнопки «R» было открыто окно с показом подобной информации об элементе (рисунок 3.8).

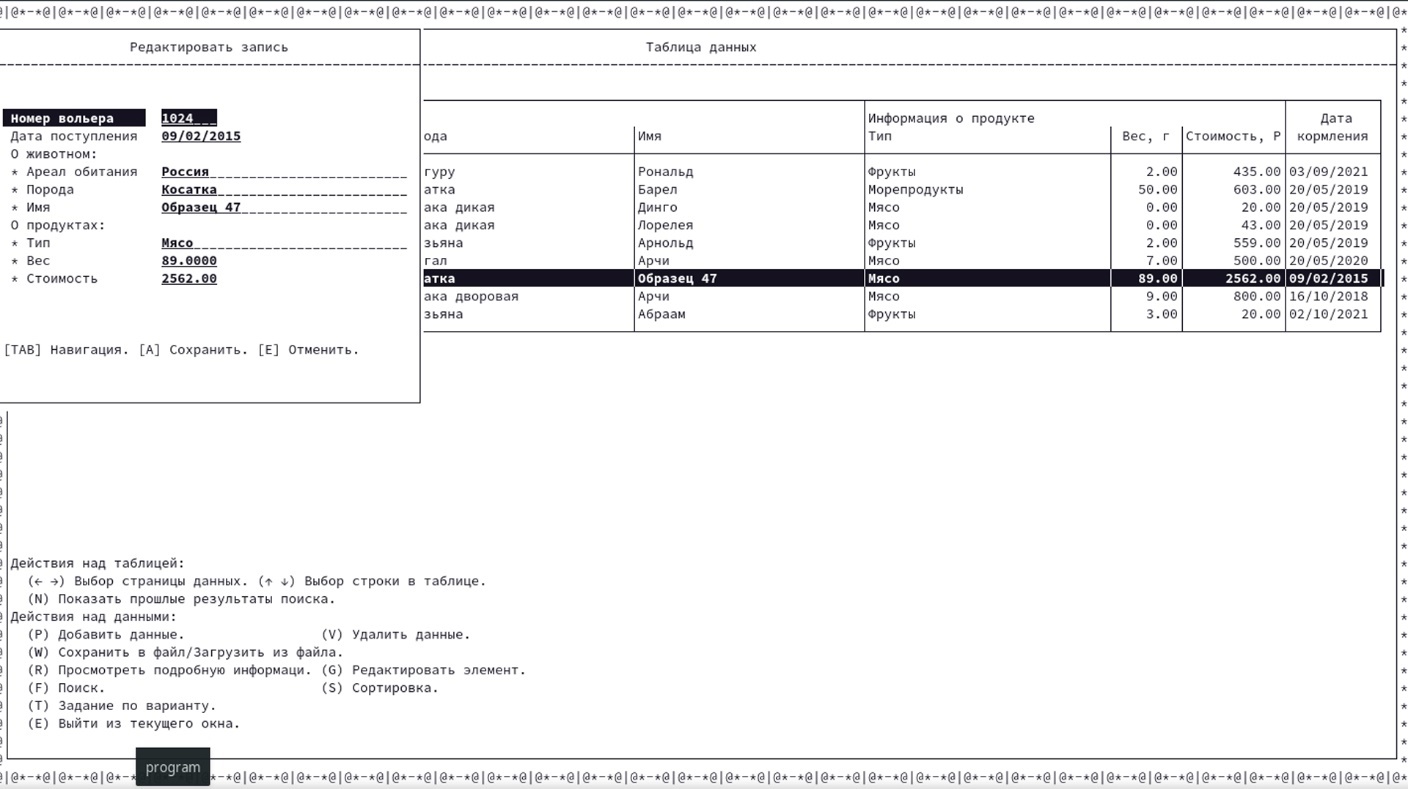


Рисунок 3.7 – Редактирование выбранного элемента

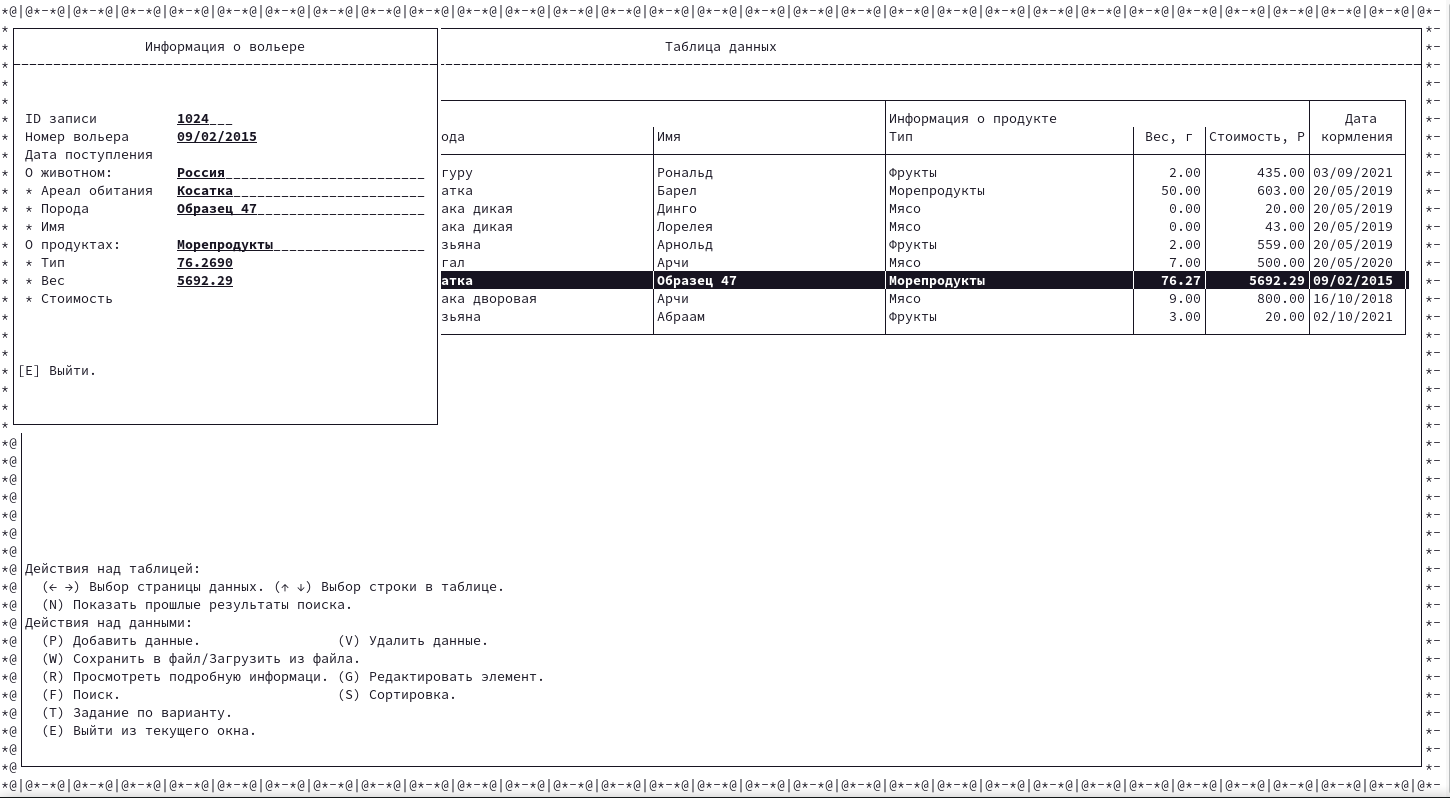


Рисунок 3.8 – Просмотр подробной информации

С помощью функции поиска были выбраны те записи, в которых в поле «Ареал» указано «Австралия» (рисунок 3.9). Все каждый элемент был удалён с помощью функции, вызываемой по нажатию кнопки «V». Была получена изменённая таблица (рисунок 3.10).

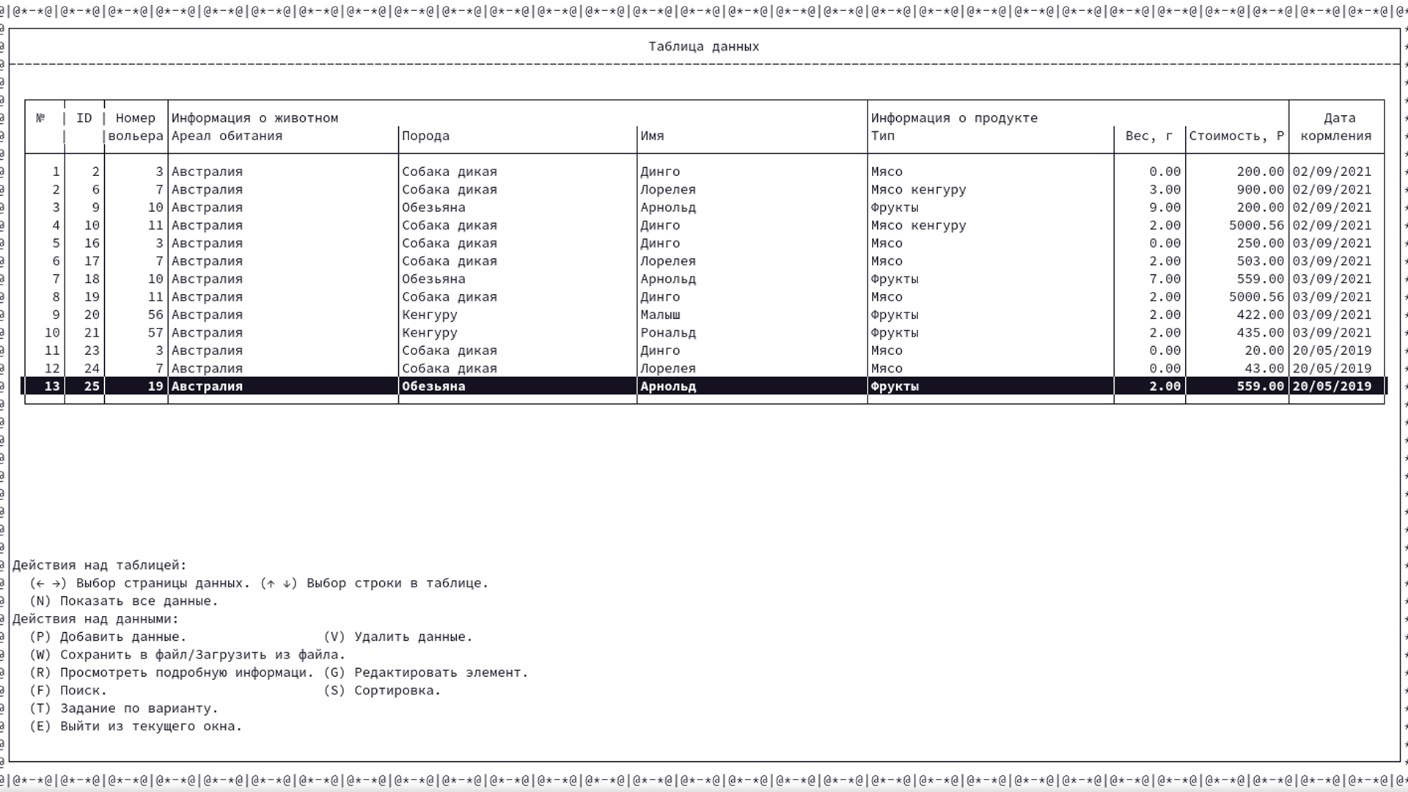


Рисунок 3.9 – Результаты поиска по ключевому полю

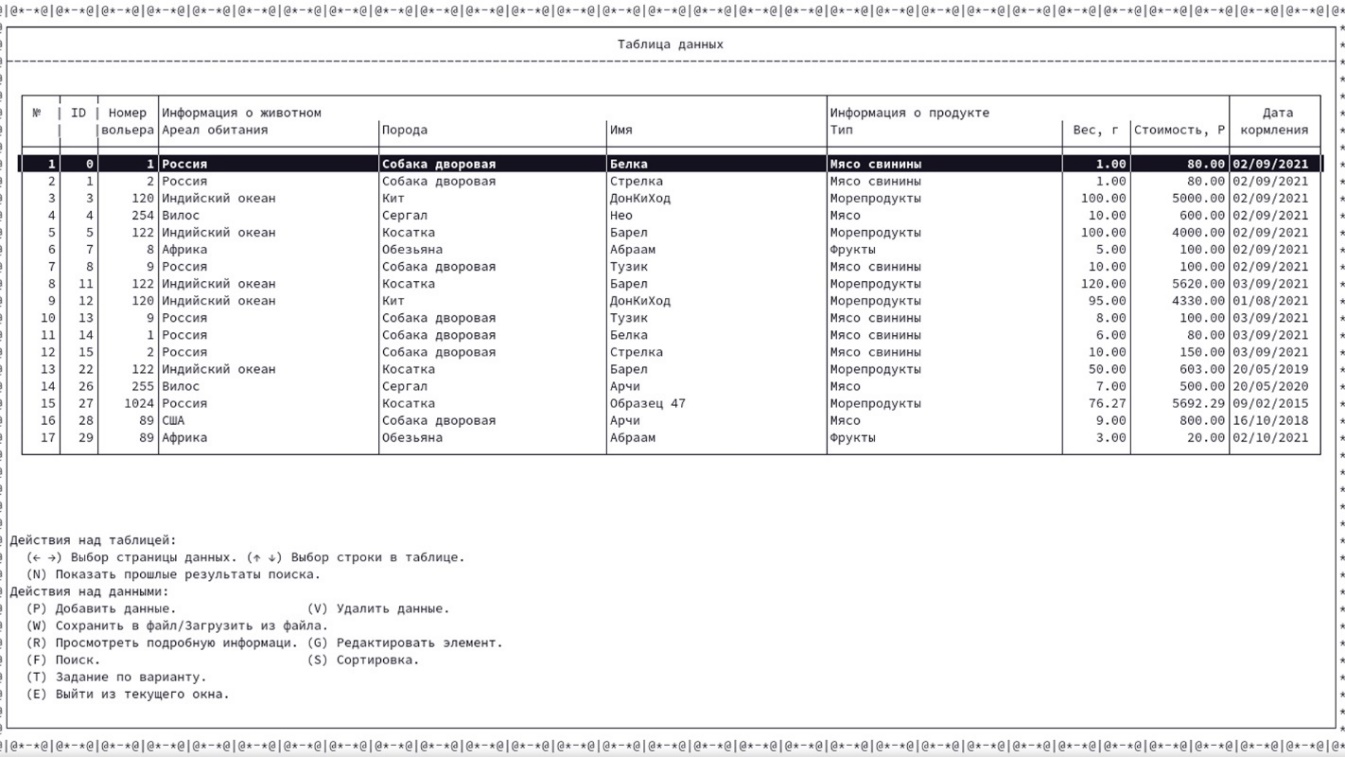


Рисунок 3.10 – Модифицированная таблица (были удалены некоторые данные)

Данные в таблице были отсортированы по значениям в поле «Имя» по алфавиту (процесс выполнения – рисунок 3.11; результат выполнения – рисунок 3.12). После чего по нажатию кнопки «T» была запущена функция, заданная по варианту. Рисунок 3.13 демонстрирует результат выполнения функции для периода 01.01.2021–01.01.2022.

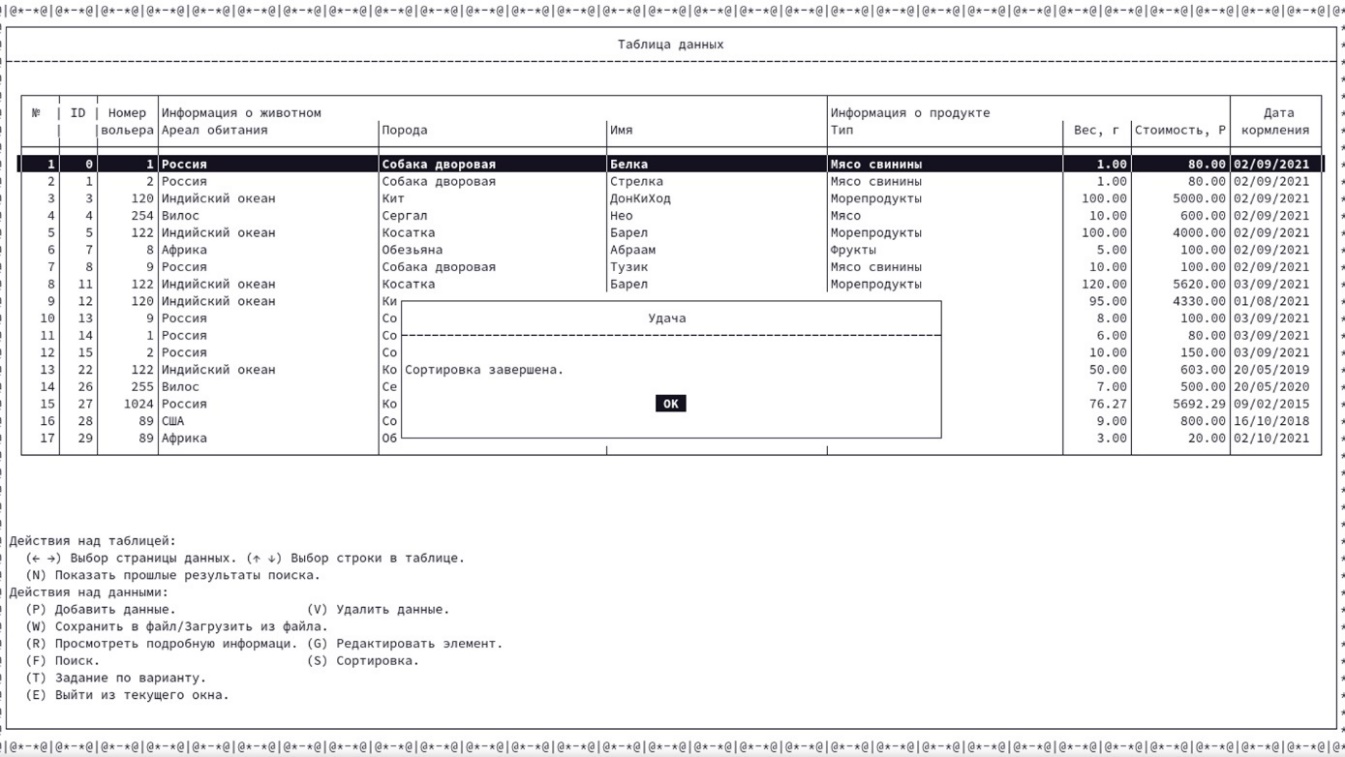


Рисунок 3.11 – Процесс сортировки данных

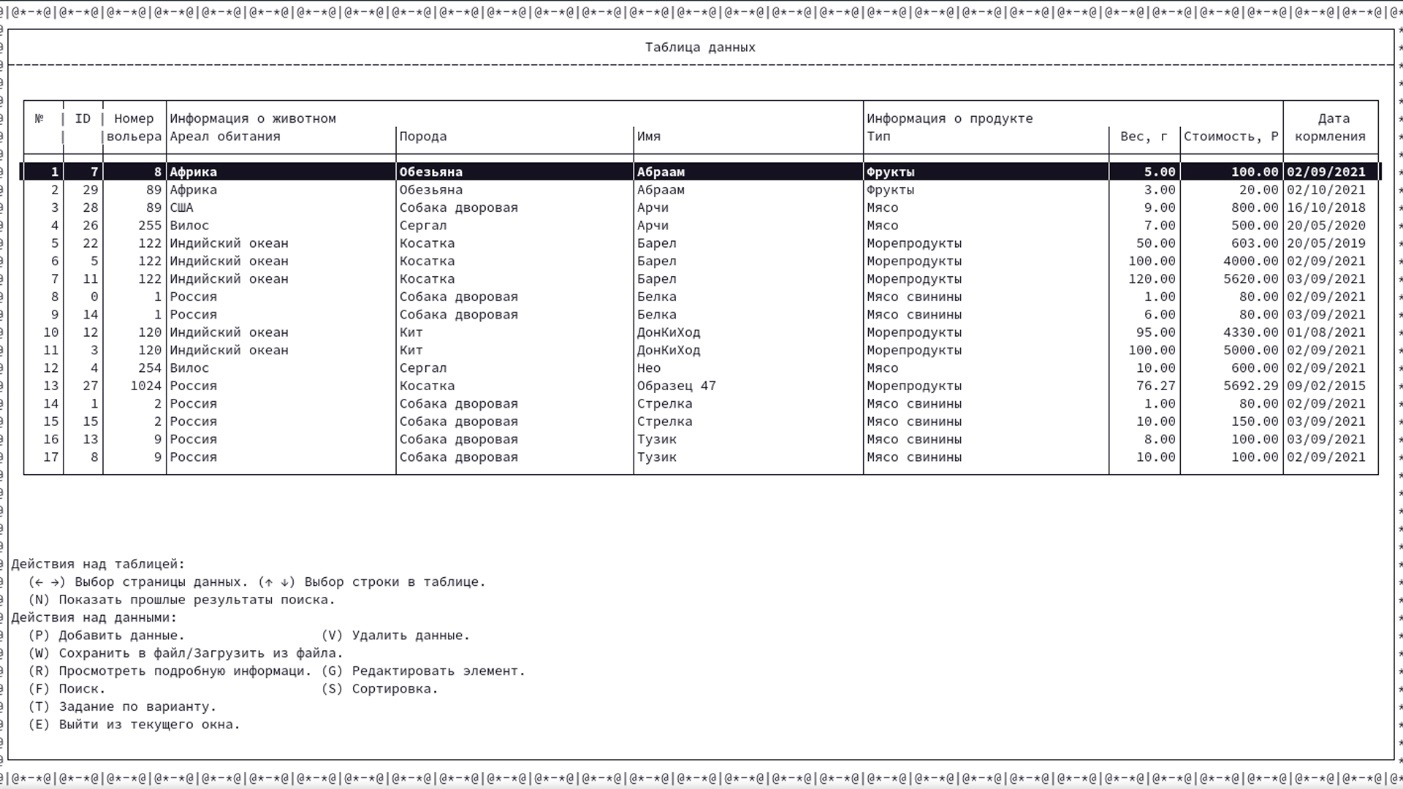


Рисунок 3.12 – Результат применения над данными функции сортировки

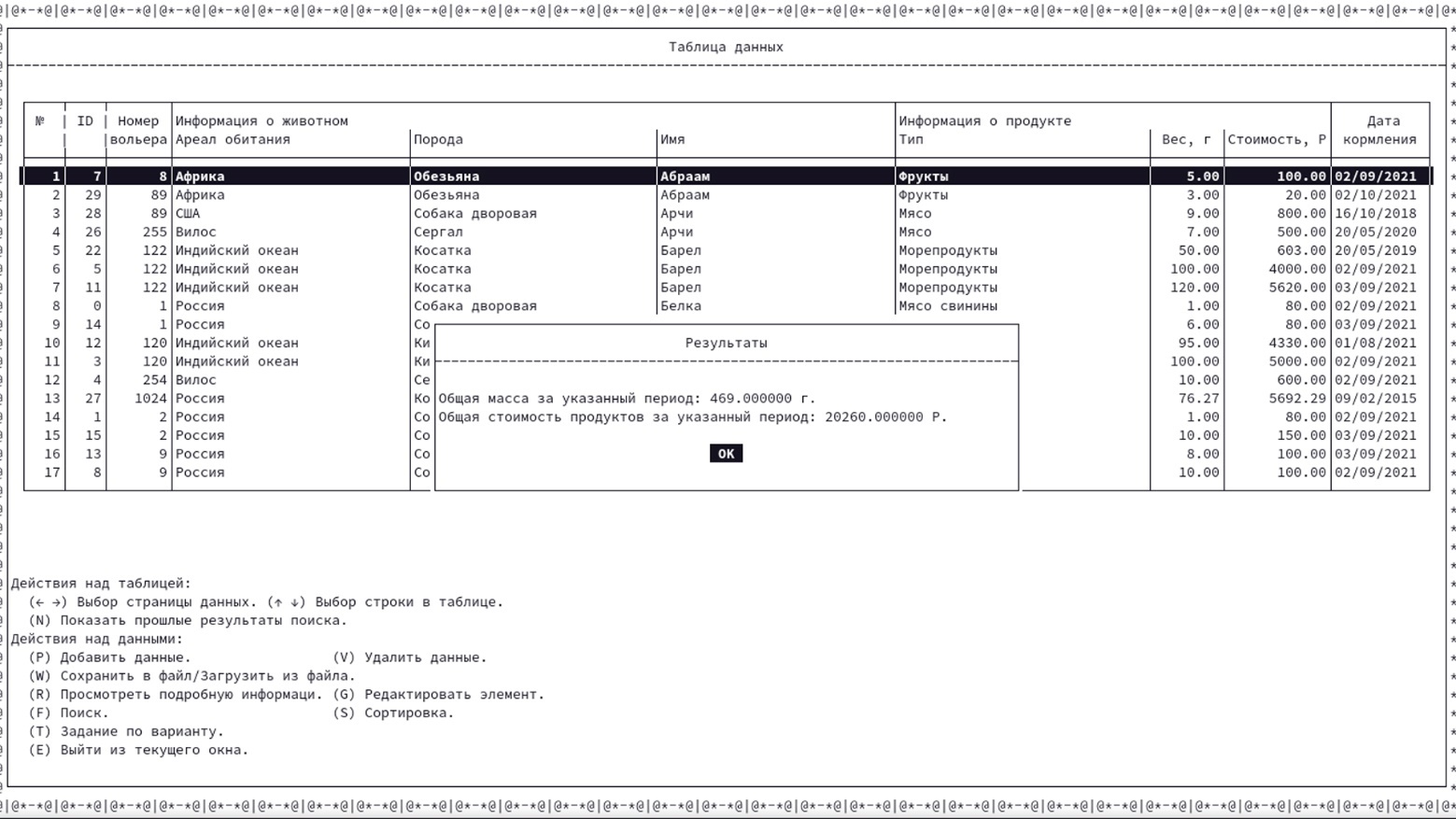


Рисунок 3.13 – Результат работы функции по варианту

По завершению всех действий данные были сохранены в файл в формате «CSV». Содержимое файла:

1;8;Абраам;Обезьяна;Африка;Фрукты;5.000;100.00;02/09/2021

2;89;Абраам;Обезьяна;Африка;Фрукты;3.000;20.00;02/10/2021

3;89;Арчи;Собака дворовая;США;Мясо;9.000;800.00;16/10/2018

4;255;Арчи;Сергал;Вилос;Мясо;7.000;500.00;20/05/2020

5;122;Барел;Косатка;Индийский океан;Морепродукты;50.000;603.00;20/05/2019

6;122;Барел;Косатка;Индийский океан;Морепродукты;100.000;4000.00;02/09/2021

7;122;Барел;Косатка;Индийский океан;Морепродукты;120.000;5620.00;03/09/2021

8;1;Белка;Собака дворовая;Россия;Мясо свинины;1.000;80.00;02/09/2021

9;1;Белка;Собака дворовая;Россия;Мясо свинины;6.000;80.00;03/09/2021

10;120;ДонКиХод;Кит;Индийский океан;Морепродукты;95.000;4330.00;01/08/2021

11;120;ДонКиХод;Кит;Индийский океан;Морепродукты;100.000;5000.00;02/09/2021

12;254;Нео;Сергал;Вилос;Мясо;10.000;600.00;02/09/2021

13;1024;Образец 47;Косатка;Россия;Морепродукты;76.269;5692.29;09/02/2015

14;2;Стрелка;Собака дворовая;Россия;Мясо свинины;1.000;80.00;02/09/2021

15;2;Стрелка;Собака дворовая;Россия;Мясо свинины;10.000;150.00;03/09/2021

16;9;Тузик;Собака дворовая;Россия;Мясо свинины;8.000;100.00;03/09/2021

17;9;Тузик;Собака дворовая;Россия;Мясо свинины;10.000;100.00;02/09/2021

Проверка работоспособности программы была проведена успешно: программа корректно считала данные из одного файла, изменила их и записала изменения в указанный пользователем другой файл.

# ЗАКЛЮЧЕНИЕ

В соответствии с вариантом задания разработана программа, в основу алгоритма которой положена структура данных в виде двунаправленного списка, позволяющего выполнять просмотр данных в двух направлениях. Особенностями программы являются: возможность производить поиск группы элементов по всем полям; сортировка элементов конкретной таблицы по всем полям как по возрастанию (в алфавитном порядке), так и по убыванию (не в алфавитном порядке); таблица результатов поиска полностью аналогична таблице со всеми данными, а значит над ней можно производить все те же операции, что и над всеми данными.

Разработка велась на базе дистрибутива GNU/Linux Manjaro в редакторе Visual Studio Code с использованием GCC для компиляции, GDB для дебагинга, make для сборки проекта и git для контроля версий кода.

В процессе разработки возникало несколько проблем.

Во-первых, для хранения строк было принято решение использовать массив символов wchar\_t. С одной стороны, это дало возможность хранить строки в кодировке UTF-8, а также корректное отображение нелатинских символов. С другой стороны, был усложнён процесс разработки ввиду того, что значительное большинство стандартных функций не поддерживает работу с wchar\_t-строками, и как следствие пришлось писать свои функции, поддерживающие работу с данными строками.

Во-вторых, ввиду громосткости и наполненности графическими элементами отлаживать программу было тяжелее.

Таким образом были достигнуты цели курсового проектирования: углублены знания языка Си; получен навык разработки программ с использованием методологии структурного программирования, а также получены практические навыки разработки приложений с использованием стронних открытых библиотек. Полученные навыки помогут разрабатывать пользователь-ориентированные приложения в терминальной среде.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Г. Шилдт C++ для начинающих. Серия "Шаг за шагом" / Г. Шилдт; пер. с англ. - М.: ЭКОМ Паблишерз, 2013. - 643 с.
2. 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.
3. 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.
4. 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.
5. Павловская Т. А. Паскаль. Программирование на языке высокого уровня: практикум / Т. А. Павловская. – СПб.: Питер, 2006. – 317 с.
6. Павловская Т. А. Паскаль. Программирование на языке высокого уровня: учеб. для вузов / Т. А. Павловская. – СПб.: Питер, 2008. – 393 с.
7. 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.
8. Вирт Н. Алгоритмы и структуры данных / Н. Вирт; пер. с англ. – М.: Мир,1989. – 360 с.
9. Белецкий Я. Энциклопедия языка Си / Я. Белецкий; пер. с польск. – М.: Мир, 1992. – 687 с.
10. 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

Структурные схемы подпрограмм

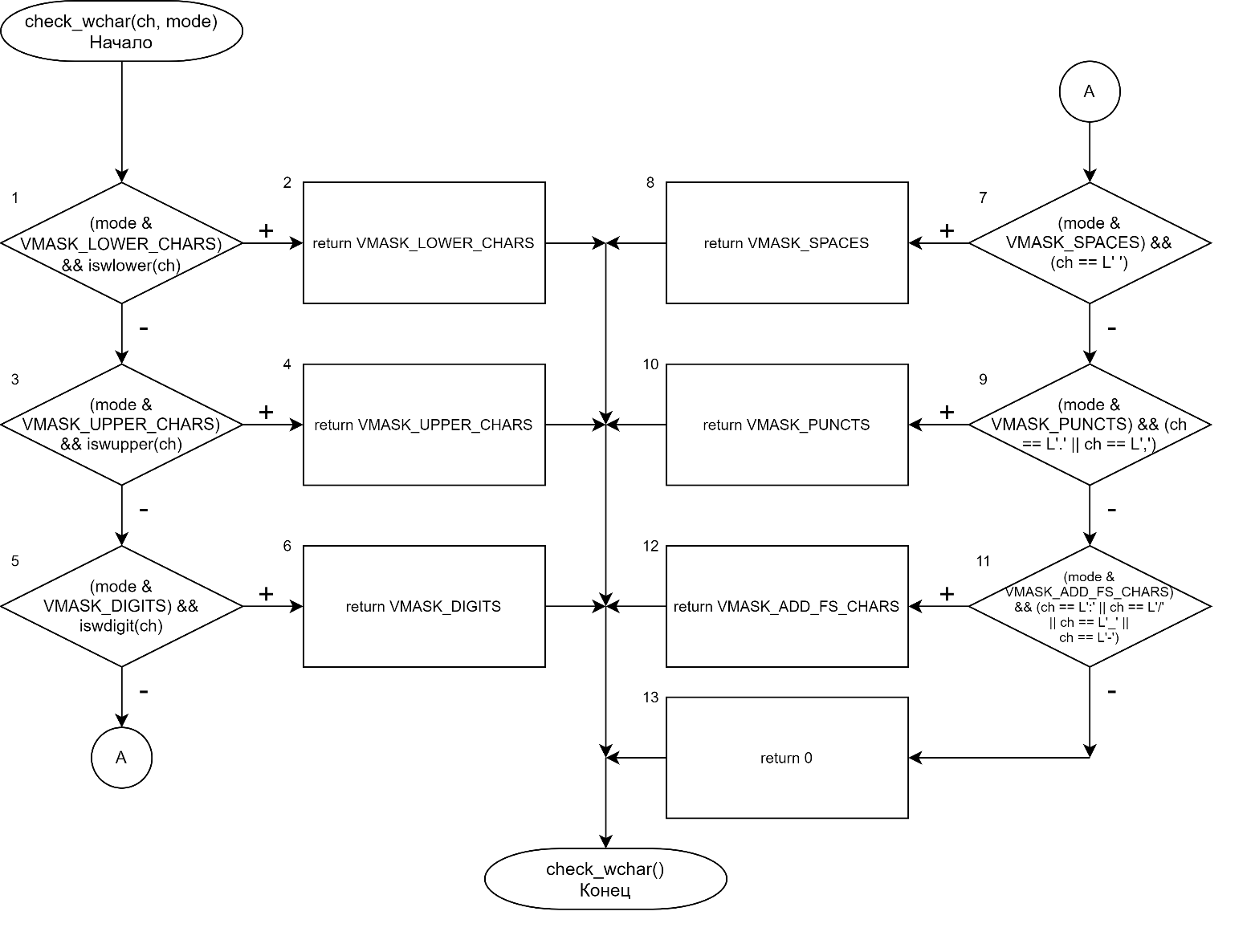


Рисунок А.1 – Подпрограмма проверки соответствия символа маске

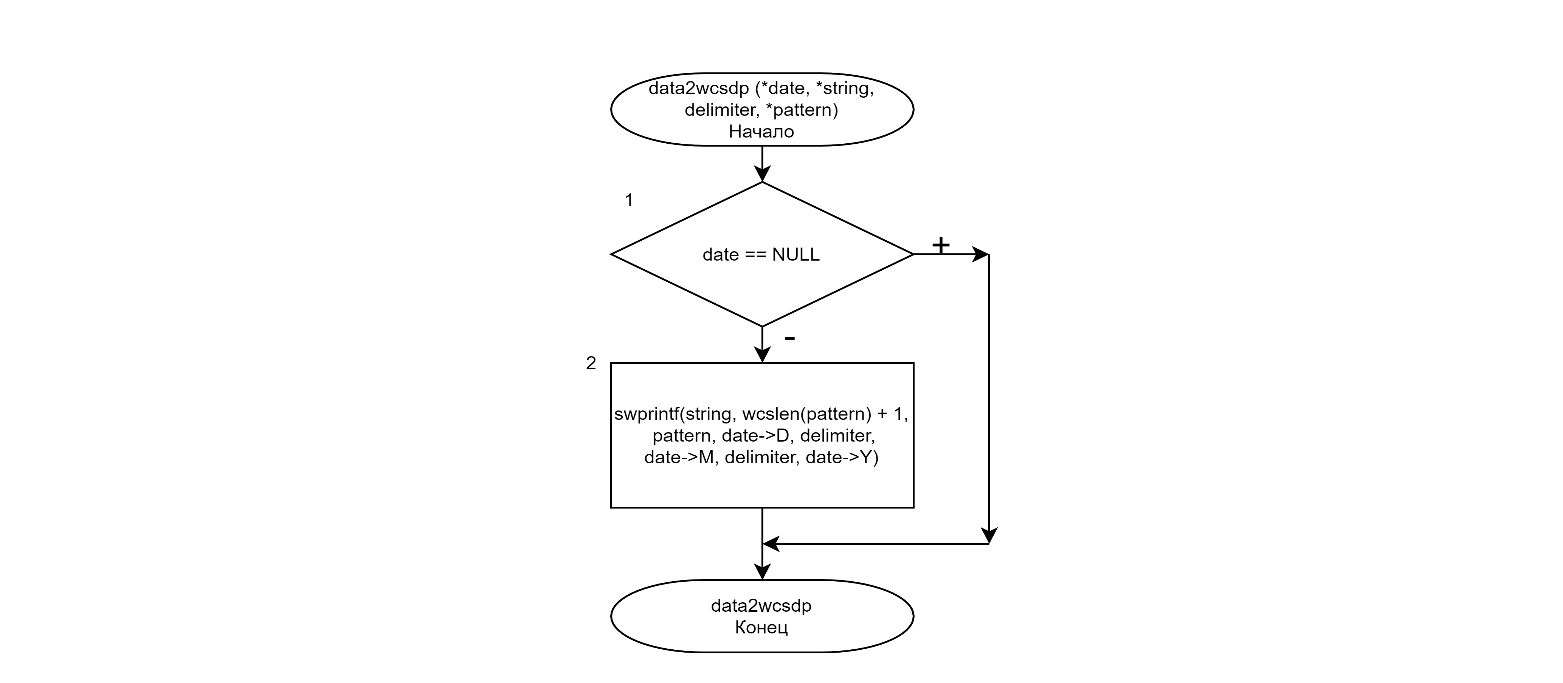


Рисунок А.2 – Подпрограммы перевода даты в строку

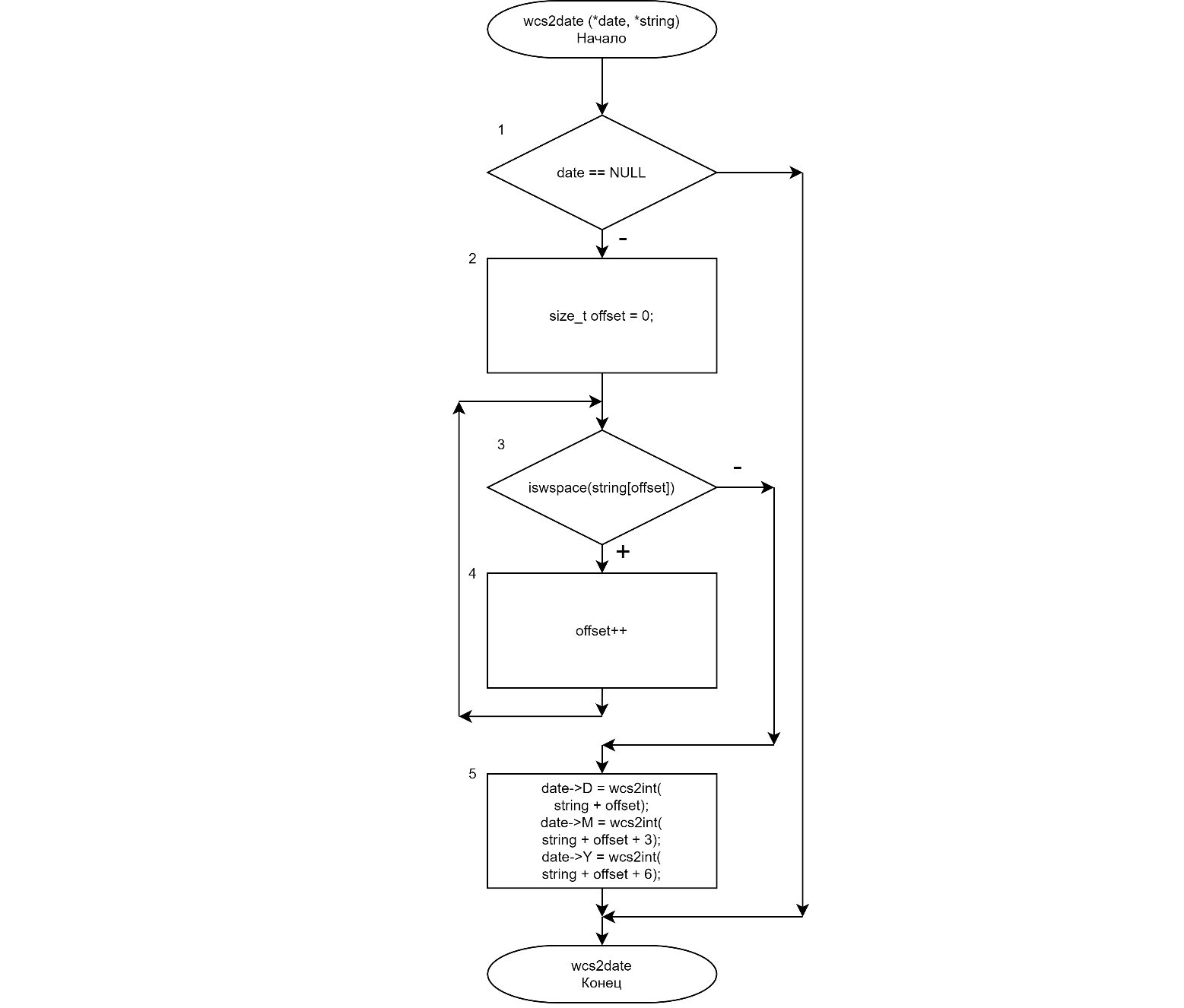


Рисунок А.3 – Подпрограмма перевода строки в дату

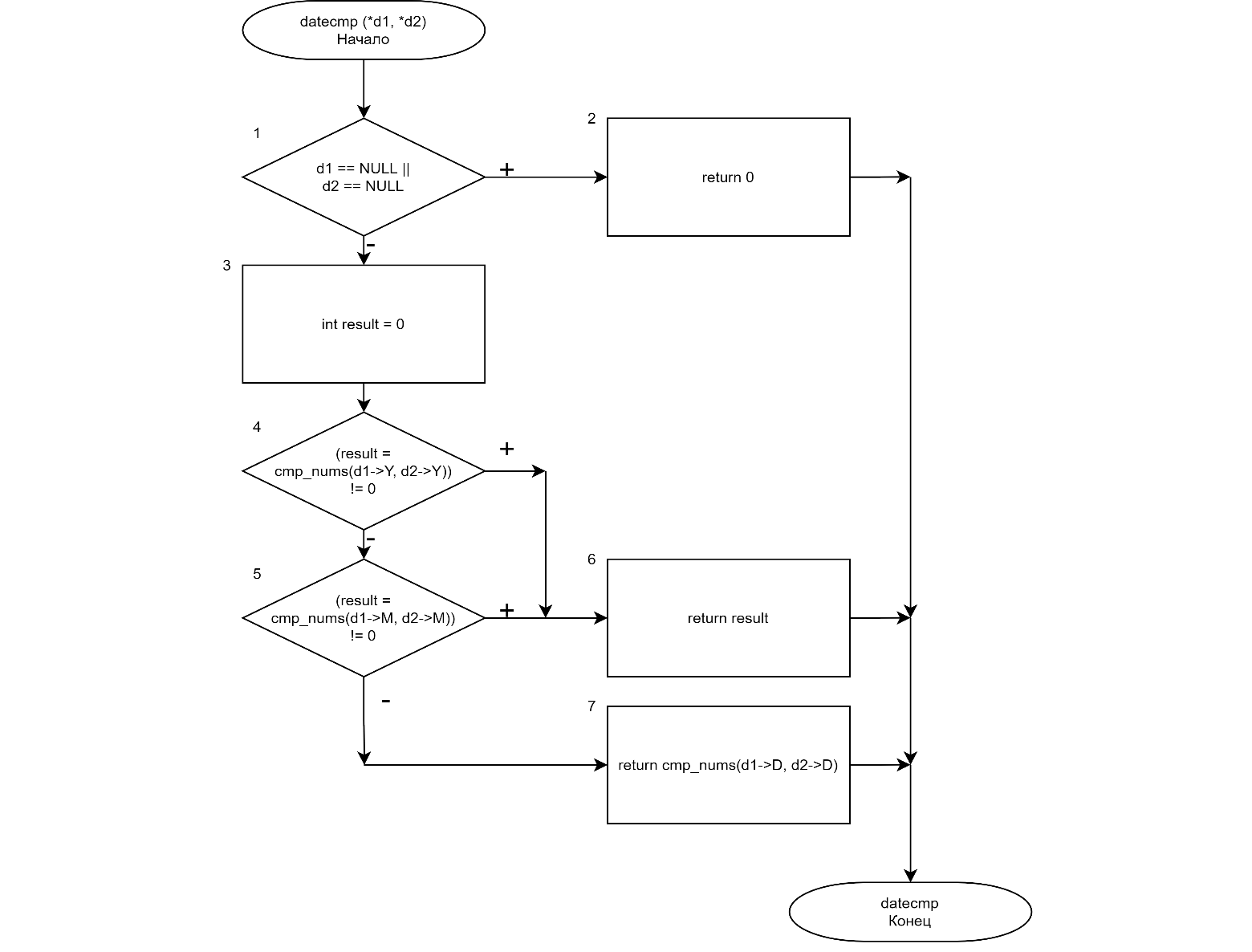


Рисунок А.4 – Подпрограмма сравнения дат

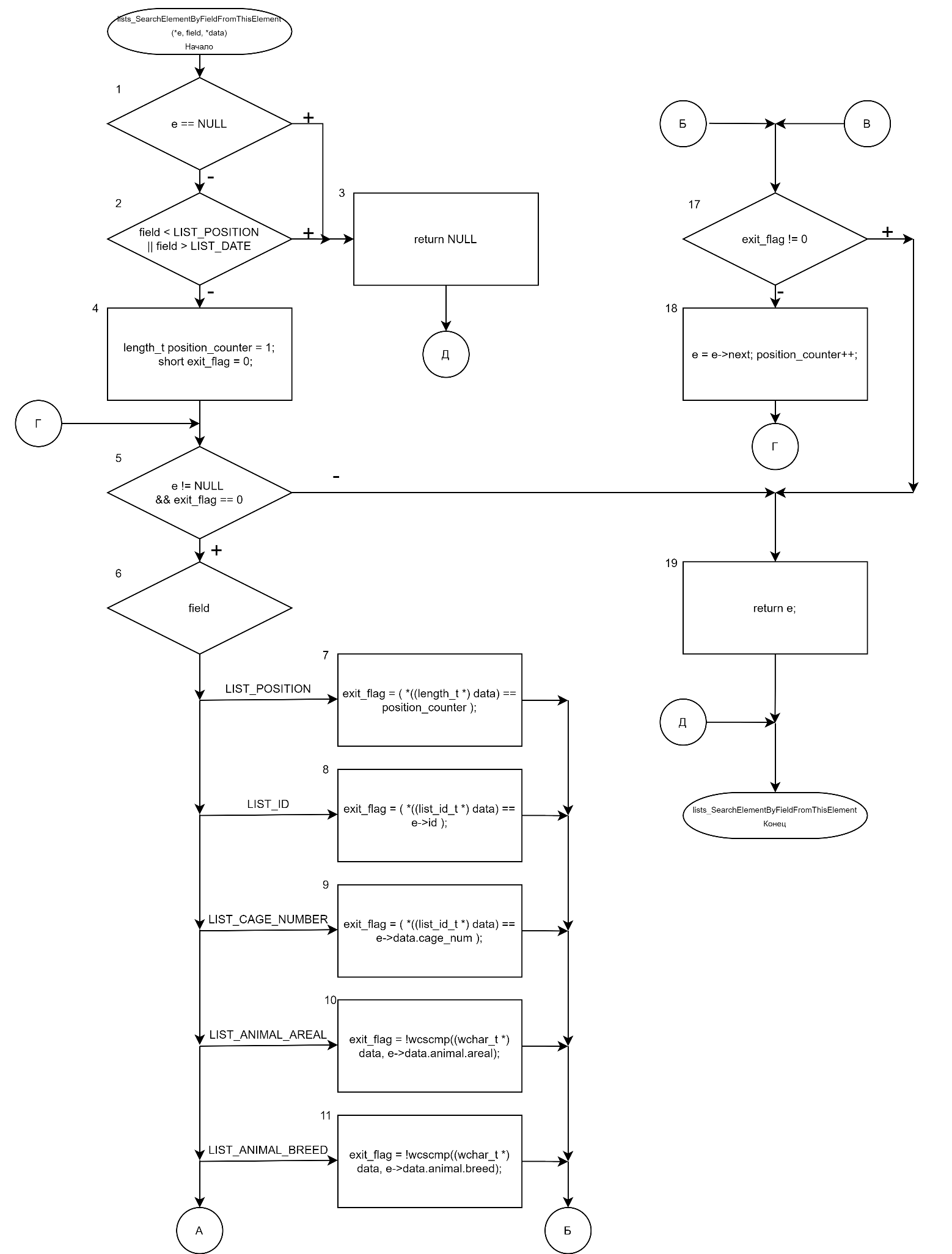


Рисунок А.5 – Подпрограмма поиска элемента списка по заданному значению поля

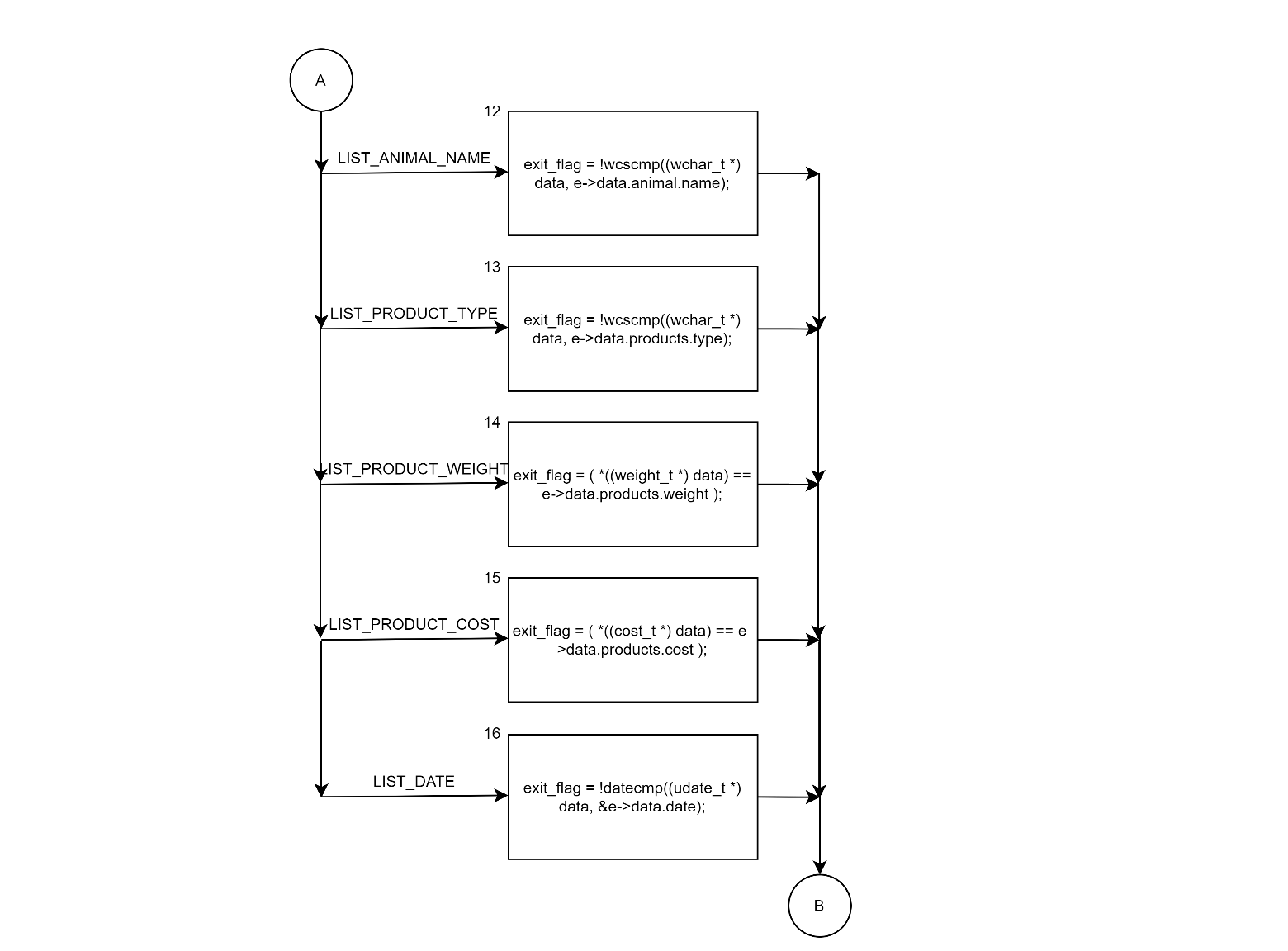


Рисунок А.5 – Лист 2

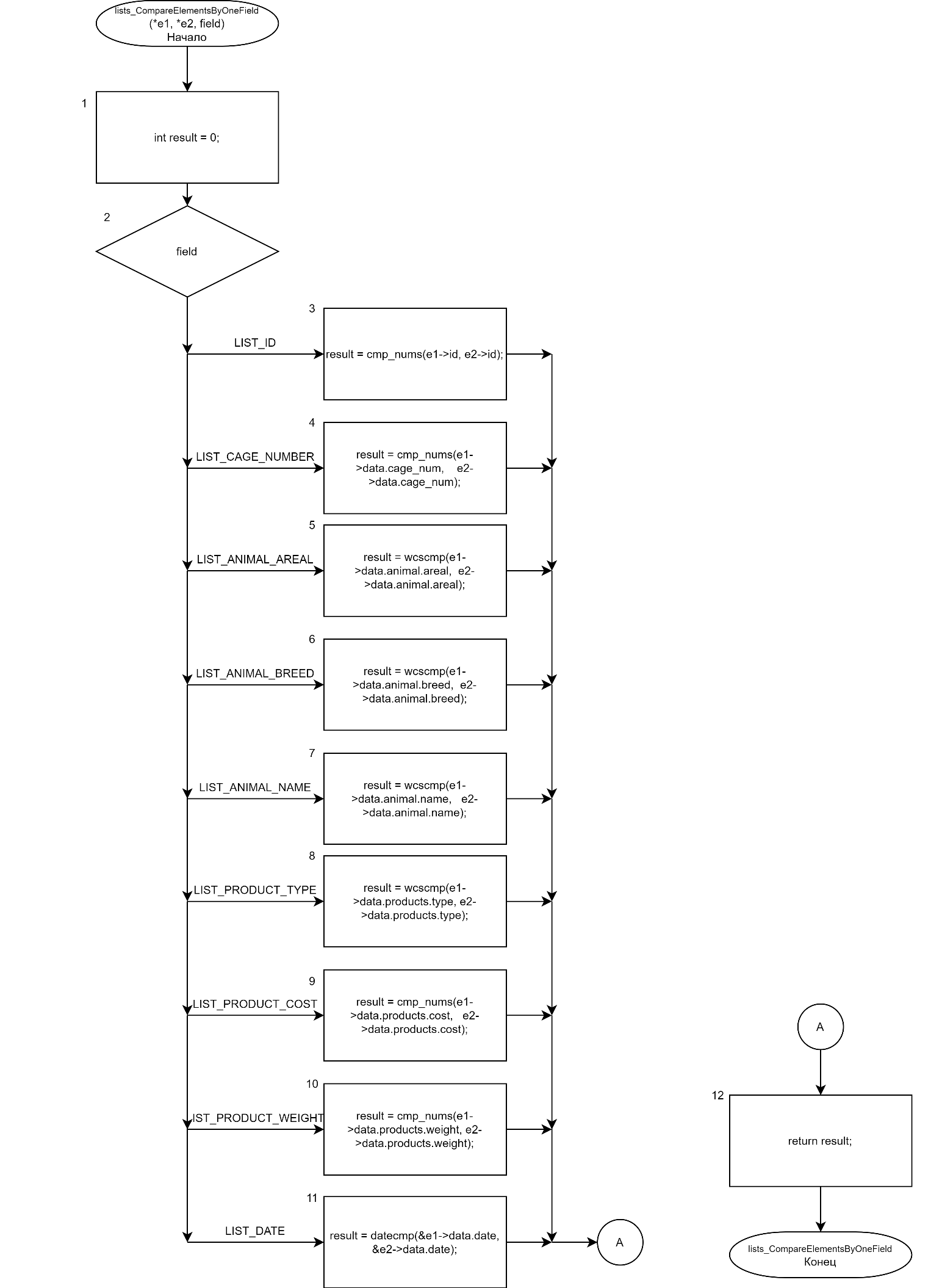


Рисунок А.6 – Подпрограмма сравнения элементов списка по одному полю

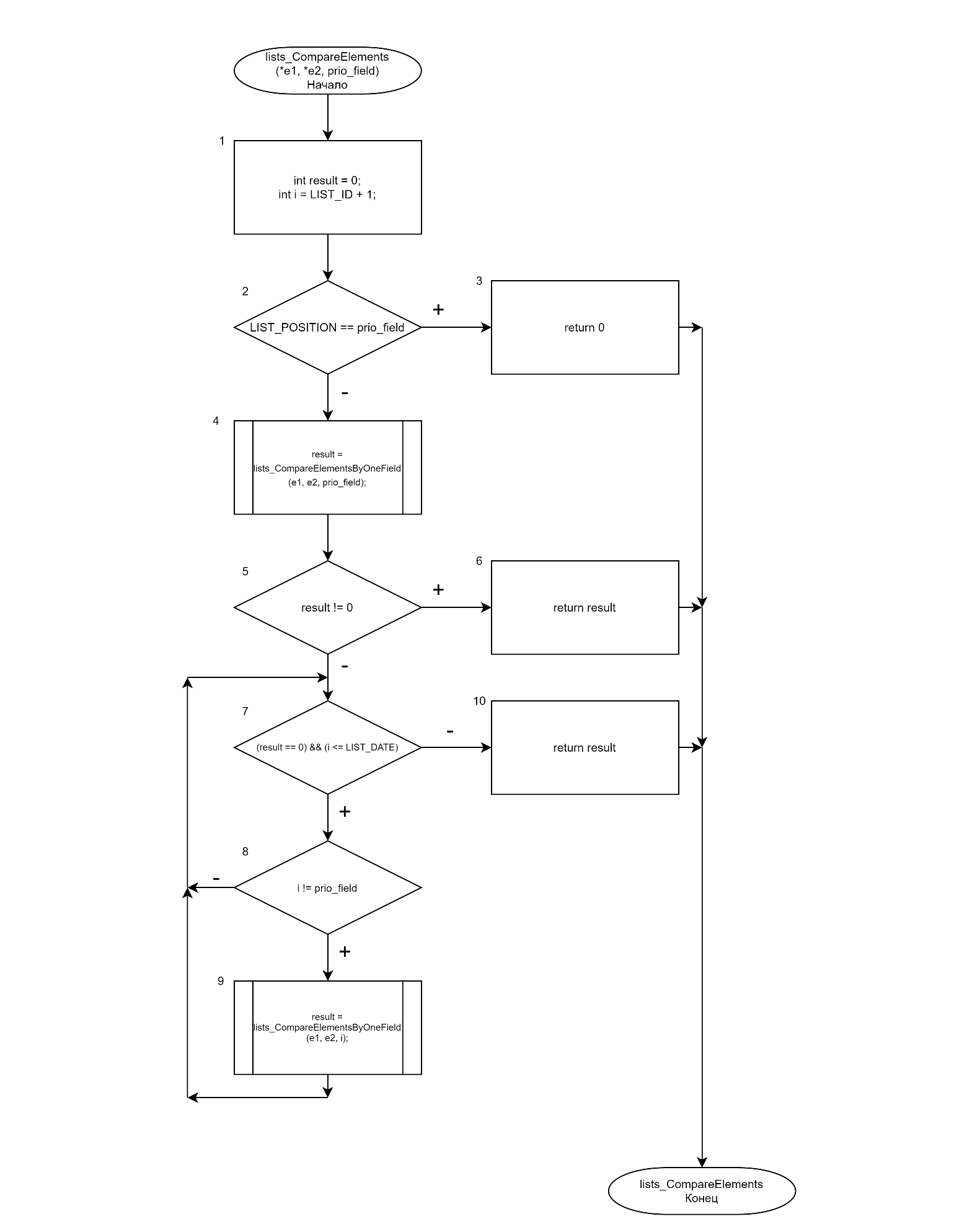


Рисунок А.7 – Подпрограмма сравнения элементов списка по нескольким полям

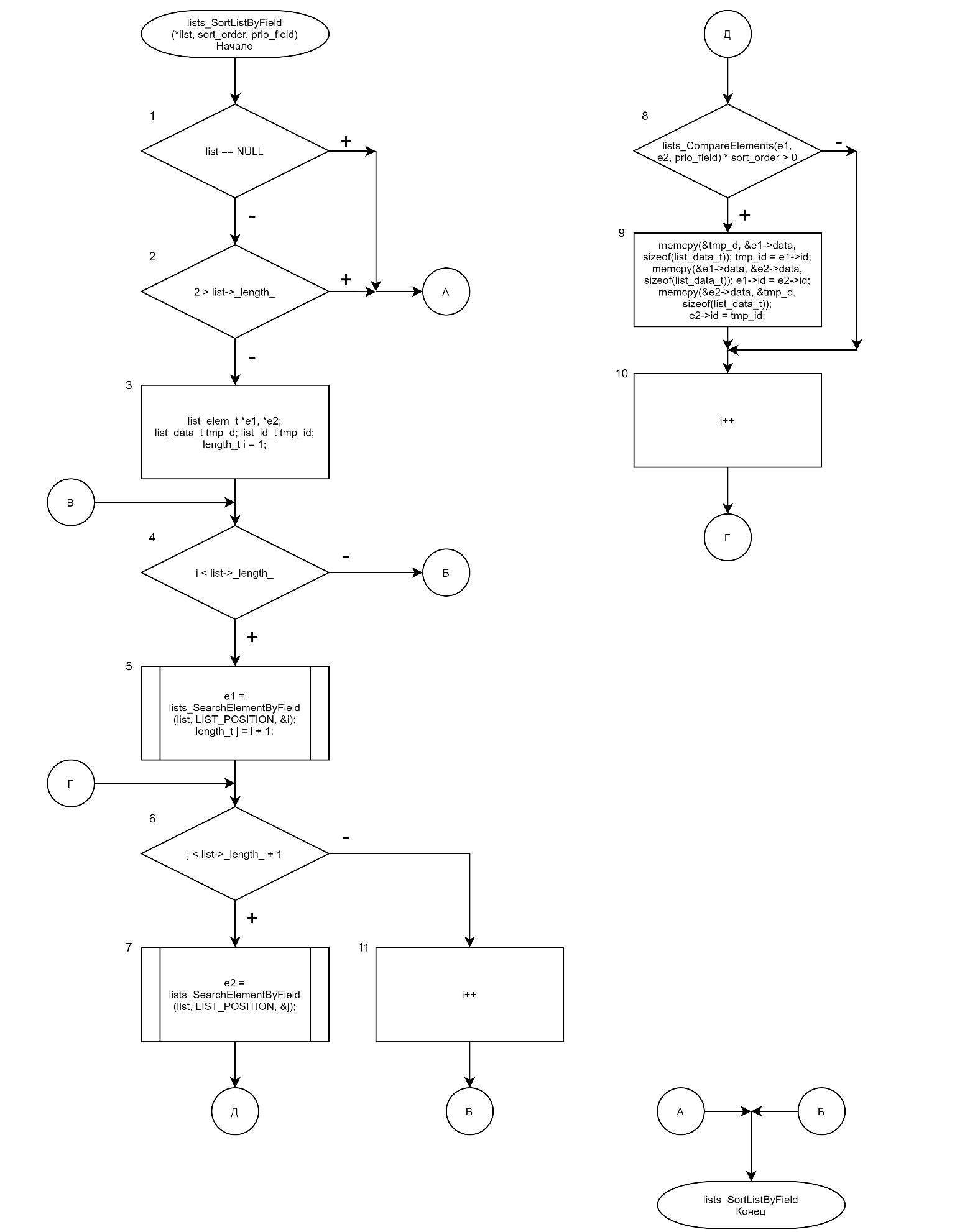


Рисунок А.8 – Подпрограмма сортировки списка по заданному полю

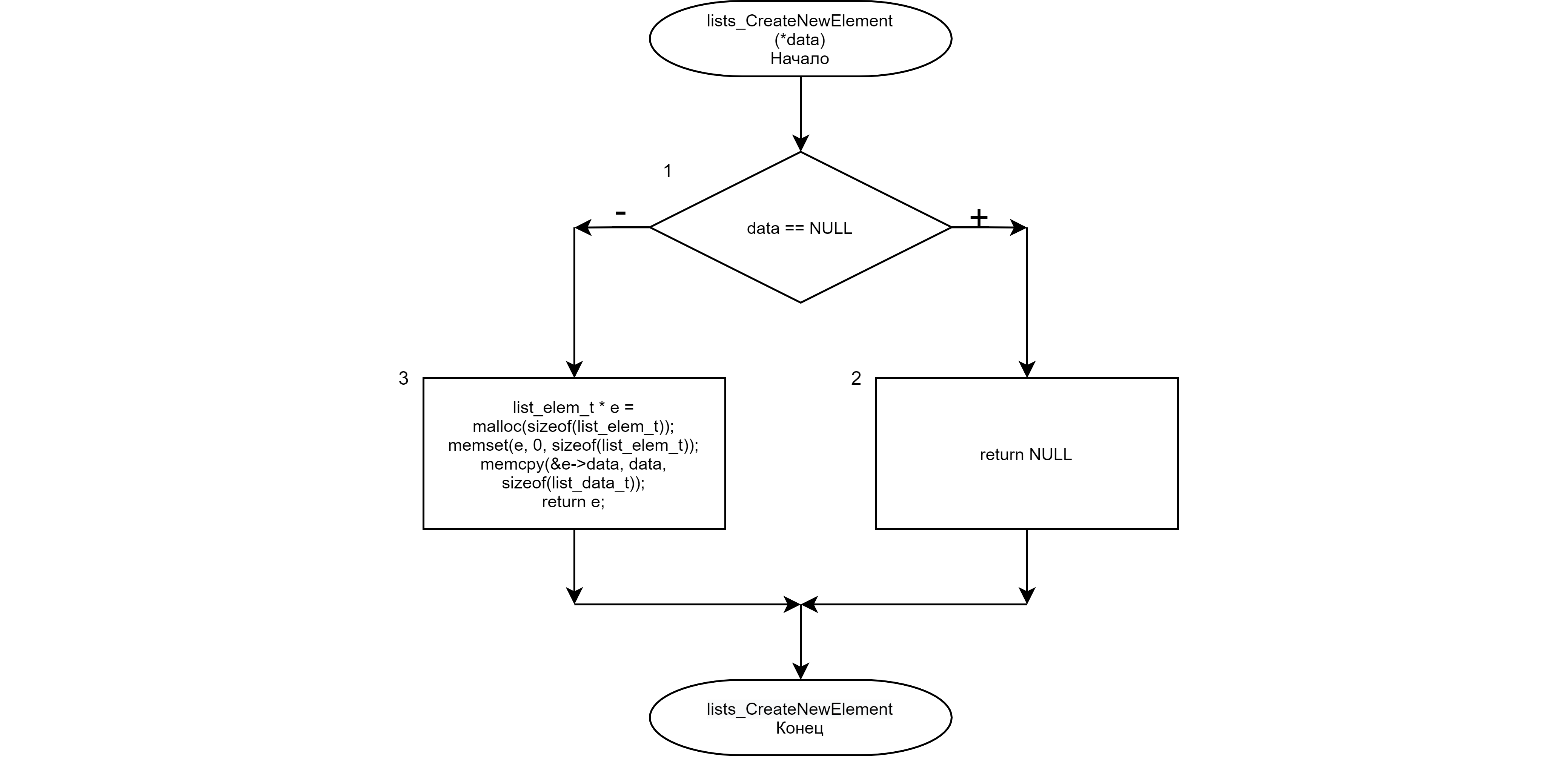


Рисунок А.9 – Подпрограмма создания элемента списка

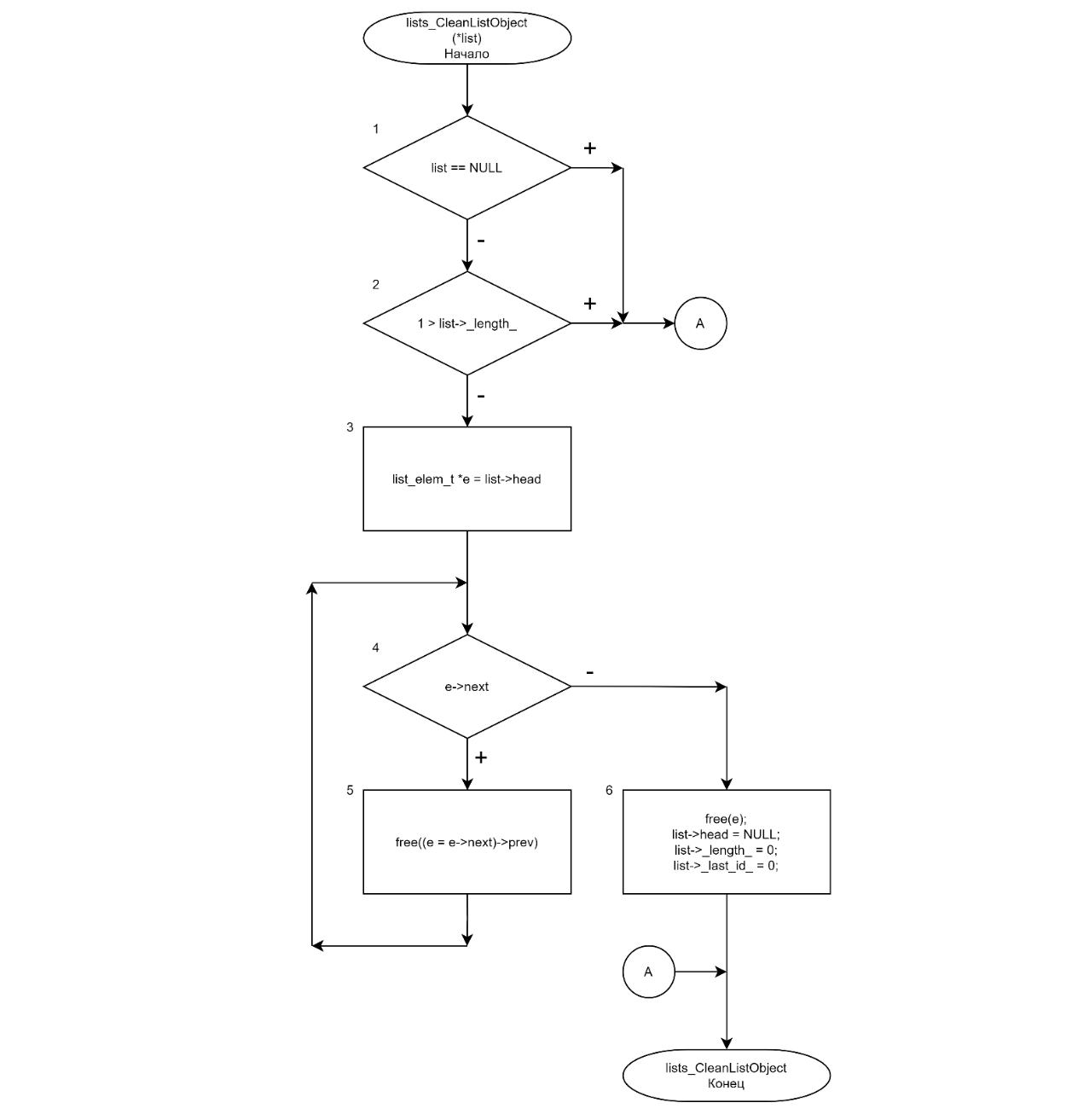


Рисунок А.10 – Подпрограмма удаления всех элементов списка

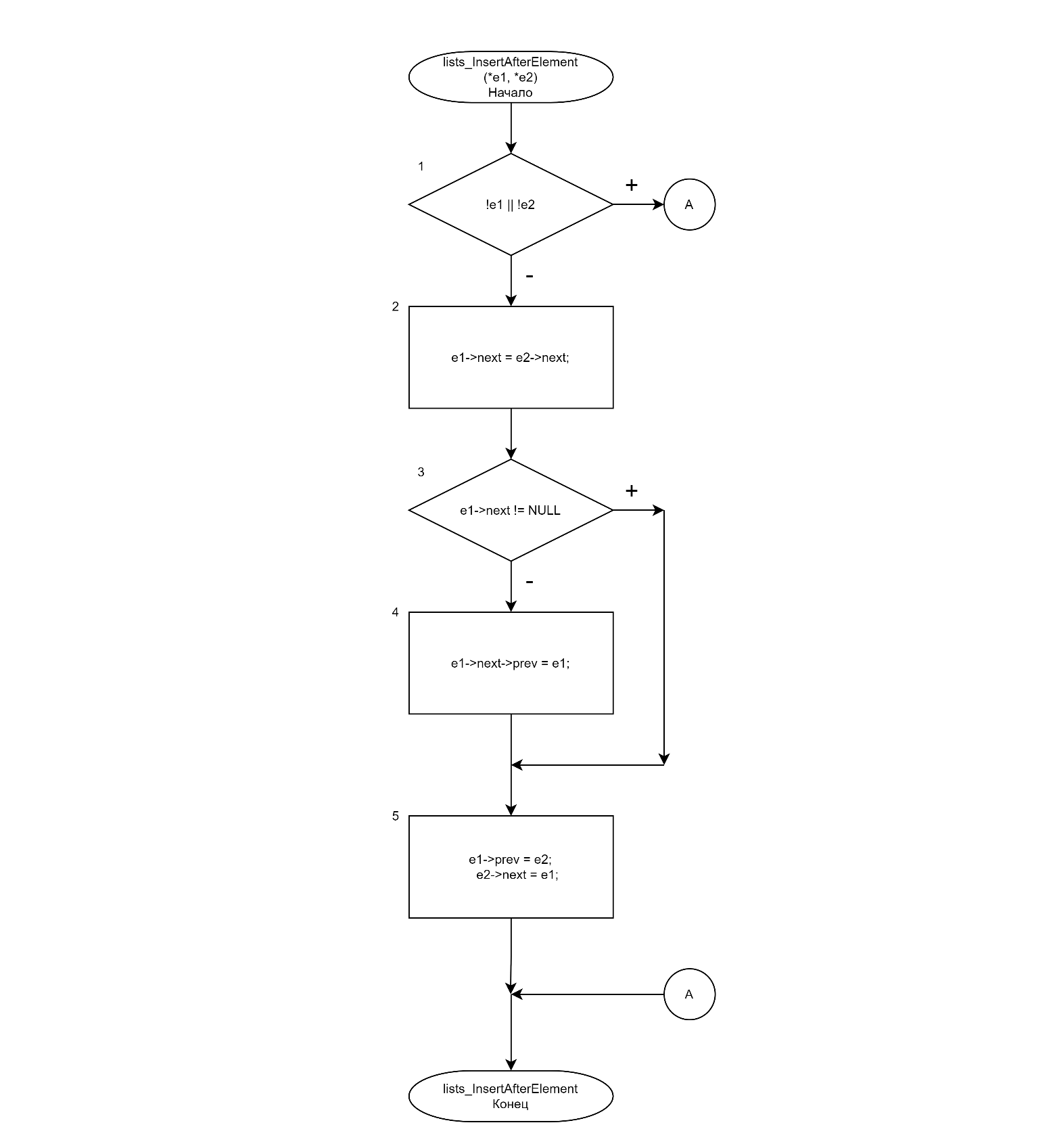


Рисунок А.11 – Подпрограмма вставки элемента после заданного элемента

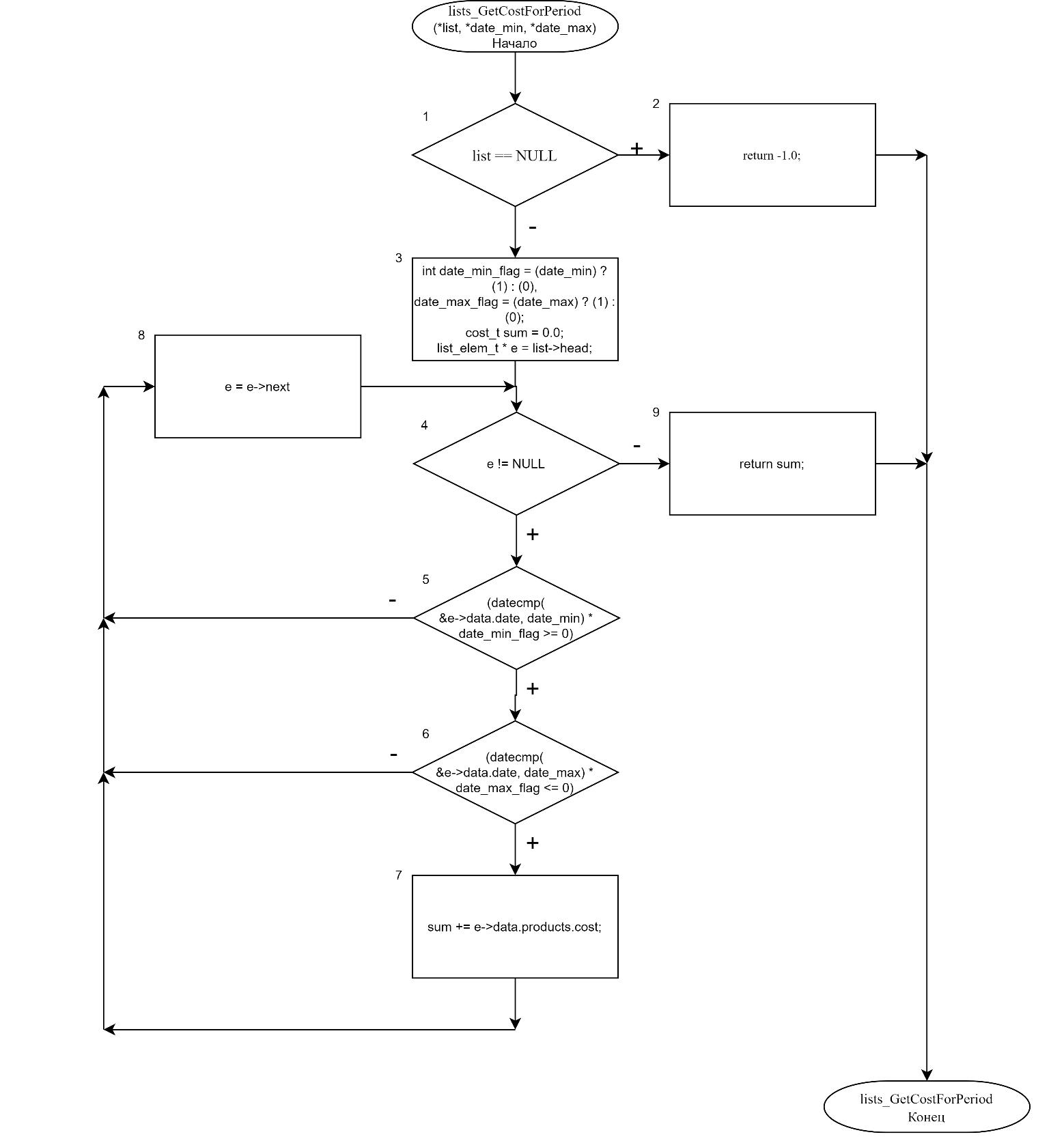


Рисунок А.12 – Подпрограмма получения стоимости употреблённых продуктов за период

# ПРИЛОЖЕНИЕ Б Листинг программы

Листинг Б.1 – Исходный код программы

**Исходный код header-файла «additional\_task.h»**

#ifndef \_ADDITIONAL\_TASK\_H

#define \_ADDITIONAL\_TASK\_H

#include "lists.h"

/\* Получить сумму употреблённых продуктов за указанный период \*/

cost\_t lists\_GetCostForPeriod (const list\_obj\_t \*, const udate\_t \*, const udate\_t \*);

/\* Получить вес употреблённых продуктов за указанный период \*/

weight\_t lists\_GetWeightForPeriod (const list\_obj\_t \*, const udate\_t \*, const udate\_t \*);

#endif // !\_ADDITIONAL\_TASK\_H

**Исходный код header-файла «checker.h»**

#ifndef \_VERIFICATORS\_H

#define \_VERIFICATORS\_H

#include <wchar.h>

#include <wctype.h>

typedef int checker\_mode\_t;

#define VMASK\_LOWER\_CHARS (0b0000001)

#define VMASK\_UPPER\_CHARS (0b0000010)

#define VMASK\_ANY\_CHAR (VMASK\_LOWER\_CHARS | VMASK\_UPPER\_CHARS)

#define VMASK\_DIGITS (0b0001000)

#define VMASK\_SPACES (0b0010000)

#define VMASK\_PUNCTS (0b0100000)

#define VMASK\_ADD\_FS\_CHARS (0b1000000)

/\*

Проверить соответствие символа маске.

Вернуть 0, если символ не соответствует маске.

\*/

int check\_wchar (const wint\_t, const checker\_mode\_t);

#endif //! \_VERIFICATORS\_H

**Исходный код header-файла «cmp\_funcs.h»**

#ifndef \_CMP\_FUNCS\_H

#define \_CMP\_FUNCS\_H

/\*

Сравнить два числа.

Вернуть 0, если числа равны; -1, если первое число меньше второго; 1, если

первое число больше второго.

\*/

#define cmp\_nums(a, b) (((a) == (b)) ? (0) : (((a) > (b)) ? (1) : (-1)))

/\* Вернуть максимум из двух чисел. \*/

#define getmax(a, b) (((a) > (b)) ? (a) : (b))

/\* Вернуть минимум из двух чисел. \*/

#define getmin(a, b) (((a) < (b)) ? (a) : (b))

#endif // !\_CMP\_FUNCS\_H

**Исходный код header-файла «date.h»**

#ifndef \_DATE\_H

#define \_DATE\_H

#include <wchar.h>

#include <wctype.h>

#include "wcs\_conv.h"

#include "cmp\_funcs.h"

/\* Тип данных: дата \*/

typedef struct

{

int Y, // Год

M, // Месяц

D; // День

}

udate\_t;

/\*

Конвертировать дату в строку, используя указанный разделитель и заданный

шаблон.

\*/

void date2wcsdp (const udate\_t \*, wchar\_t \*, const wchar\_t, const wchar\_t \*);

/\*Конвертировать дату в строку \*/

#define date2wcs(date, string) date2wcsdp(date, string, L'/', L"%02d%lc%02d%lc%04d")

/\* Конвертировать дату в строку, используя указанный разделитель \*/

#define date2wcsd(date, string, delimiter) date2wcsdp(date, string, delimiter, L"%02d%lc%02d%lc%04d")

/\* Конвертировать дату в строку, используя указанный шаблон \*/

#define date2wcsp(date, string, pattern) date2wcsdp(date, string, L'/', pattern)

/\* Конвертировать строку в дату \*/

void wcs2date (udate\_t \*, const wchar\_t \*);

/\*

Сравнить две даты.

Вернуть 0, если даты равны; отрицательное число, если первая дата меньше

второй; положительное число, если первая дата больше второй.

\*/

int datecmp (const udate\_t \*, const udate\_t \*);

#endif // !\_DATE\_H

**Исходный код header-файла «files.h»**

#ifndef \_FILES\_H

#define \_FILES\_H

#include <stdio.h>

#include <string.h>

#include <wchar.h>

#include <wctype.h>

#include <locale.h>

#include "lists.h"

#define CSV\_DELIMITER L';'

enum FS\_RETURN {

FS\_SUCCESS = 0, /\* Успешное завершение работы \*/

FS\_FILE\_NO\_READ\_ACCESS, /\* Ошибка: нельзя прочитаться данные из файла \*/

FS\_FILE\_NO\_WRITE\_ACCESS, /\* Ошибка: нельзя записать данные в файл \*/

FS\_NO\_LIST\_PTR, /\* Ошибка: передан пустой указатель на список \*/

FS\_NO\_LIST\_DATA, /\* Ошибка: передан список без данных \*/

FS\_BROKEN\_DATA /\* Ошибка: данные в файле повреждены \*/

};

/\* Определить, доступен ли файл для чтения (имя файла в виде MBS) \*/

int is\_exist (const char \*);

/\* Определить, доступен ли файл для чтения (имя файла в виде WCS) \*/

int is\_exist\_wcs (const wchar\_t \*);

/\* Сохранить данные в бинарный файл \*/

int save\_as\_bin (const list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs);

/\* Считать данные из бинарного файла \*/

int read\_from\_bin (list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs);

/\* Сохранить данные объекта структуры в CSV-файл \*/

int save\_as\_csv (const list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs);

/\* Считать данные из CSV-файла \*/

int read\_from\_csv (list\_obj\_t \* list, const wchar\_t \* fileName);

#endif // !\_FILES\_H

**Исходный код header-файла «keyboard\_keys.h»**

typedef unsigned int mkey\_t;

enum MKEYBOARD\_KEYS

{

MKEY\_NULL = 0,

MKEY\_CTRL\_A = 1,

MKEY\_CTRL\_B,

MKEY\_CTRL\_C,

MKEY\_CTRL\_D,

MKEY\_TAB = 9,

MKEY\_ENTER = 13,

MKEY\_ESCAPE= 27,

MKEY\_SPACE = 32,

MKEY\_ARROW\_UP = 65,

MKEY\_ARROW\_DOWN,

MKEY\_ARROW\_RIGHT,

MKEY\_ARROW\_LEFT,

MKEY\_END = 70,

MKEY\_HOME = 72,

MKEY\_F1 = 80,

MKEY\_F2,

MKEY\_F3,

MKEY\_F4,

MKEY\_F5,

MKEY\_F6,

MKEY\_F7,

MKEY\_F8,

MKEY\_F9,

MKEY\_F10,

MKEY\_F11,

MKEY\_F12,

MKEY\_DELETE = 126,

MKEY\_BACKSPACE

};

**Исходный код header-файла «lists.h»**

#ifndef \_LISTS\_H

#define \_LISTS\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <wchar.h>

#include "date.h"

#include "cmp\_funcs.h"

/\* Способы сортировки \*/

#define SORT\_DESCENDING\_ORDER (-1)

#define SORT\_ASCENDING\_ORDER (1)

/\* Условная привязка чисел к полю структуры элемента списка \*/

enum DATA\_FIELDS

{

LIST\_POSITION = 0, // Позиция элемента в списке

LIST\_ID, // Id элемента в списке

LIST\_CAGE\_NUMBER, // a.cage\_num

LIST\_ANIMAL\_AREAL, // a.animal.areal

LIST\_ANIMAL\_BREED, // a.animal.breed

LIST\_ANIMAL\_NAME, // a.animal.name

LIST\_PRODUCT\_TYPE, // a.products.type

LIST\_PRODUCT\_WEIGHT, // a.products.weight

LIST\_PRODUCT\_COST, // a.products.cost

LIST\_DATE // a.date

};

/\* Максимальные длины массивов в структуре данных \*/

#define ANIMAL\_AREAL\_MAX\_LEN 32 // Максимальная длина Ареал обитания

#define ANIMAL\_BREED\_MAX\_LEN 32 // Максимальная длина Порода

#define ANIMAL\_NAME\_MAX\_LEN 32 // Максимальная длина Имя животного

#define PRODUCT\_TYPE\_MAX\_LEN 32 // Максимальная длина Тип продукта

/\* Максимальное количество цифр в числе \*/

#define WEIGHT\_DGT\_CNT 7

#define COST\_DGT\_CNT 7

#define CAGE\_NUM\_DGT\_CNT 7

#define POSITION\_DGT\_CNT 4

#define ID\_DGT\_CNT 4

/\* Вспомогательные типы данных \*/

typedef unsigned short field\_num\_t; // Условный номер поля

typedef unsigned long list\_id\_t; // ID элемента в списке

typedef unsigned long length\_t; // Длина чего-либо

typedef long double weight\_t; // Вес

typedef long double cost\_t; // Стоимость

/\* Тип данных: структура с данными \*/

typedef struct list\_data\_t

{

list\_id\_t cage\_num; // Номер вольера.

struct { // Информация о животном.

wchar\_t areal[ANIMAL\_AREAL\_MAX\_LEN + 1]; // Ареал обитания животного.

wchar\_t breed[ANIMAL\_BREED\_MAX\_LEN + 1]; // Порода животного.

wchar\_t name[ANIMAL\_NAME\_MAX\_LEN + 1]; // Имя животного.

} animal;

struct { // Информация о продуктах.

wchar\_t type[PRODUCT\_TYPE\_MAX\_LEN + 1]; // Тип.

weight\_t weight; // Вес.

cost\_t cost; // Стоимость.

} products;

udate\_t date; // Дата.

}

list\_data\_t;

/\* Тип данных: элемент списка \*/

typedef struct list\_elem\_t

{

list\_id\_t id; // Уникальный номер элемента в списке

list\_data\_t data; // Хранимые данные

/\* Указатели на следующий и предыдущий элементы списка \*/

struct list\_elem\_t \*next, \*prev;

}

list\_elem\_t;

/\* Тип данных: объект списка \*/

typedef struct list\_obj\_t

{

list\_elem\_t \*head;

length\_t \_length\_;

list\_id\_t \_last\_id\_;

}

list\_obj\_t;

/\* Создать новый объект списка \*/

list\_obj\_t \* lists\_CreateNewListObject (void);

/\* Полное удаление объекта списка из памяти \*/

void lists\_DeleteListObject (list\_obj\_t \*\*);

/\* Удалить все данные из объекта списка \*/

void lists\_CleanListObject (list\_obj\_t \*);

/\* Создать элемент списка на основе переданных данных \*/

list\_elem\_t \* lists\_CreateNewElement (list\_data\_t \*);

/\* Вставить элемент (первый аргумент) после элемента (второй аргумент) \*/

void lists\_InsertAfterElement (list\_elem\_t \*, list\_elem\_t \*);

/\* Удалить элемент (пересвязка делается автоматически) \*/

void lists\_DeleteElement (list\_elem\_t \*);

/\* Безопасно извлечь элемент из списка \*/

void list\_DetachElement (list\_elem\_t \*);

/\* Вставить элемент в самое начало списка \*/

void lists\_InsertAsListsHead (list\_obj\_t \*, list\_elem\_t \*);

/\* Вставить элемент в конец списка \*/

void lists\_InsertAsListsTail (list\_obj\_t \*, list\_elem\_t \*);

/\* Поиск элемента в объекте списка по значению поля \*/

list\_elem\_t \* lists\_SearchElementByField (list\_obj\_t \*, field\_num\_t, void \*);

/\* Поиск элемента, начиная с текущего и включая в поиск текущий \*/

list\_elem\_t \* lists\_SearchElementByFieldFromThisElement (list\_elem\_t \*, field\_num\_t, void \*);

/\* Сравнить элементы по одному полю \*/

int lists\_CompareElementsByOneField (const list\_elem\_t \*, const list\_elem\_t \*, const field\_num\_t);

/\* Сравнить элементы по всем полям (вначале по приоритетному) \*/

int lists\_CompareElements (const list\_elem\_t \*, const list\_elem\_t \*, const field\_num\_t);

/\* Сортировать список по возрастанию или по убыванию по приоритетному полю \*/

void lists\_SortListByField (list\_obj\_t \*, const int, const field\_num\_t);

/\* Получить общую стоимость всех продуктов за определённый период \*/

cost\_t lists\_GetCostForPeriod (const list\_obj\_t \*, const udate\_t \*, const udate\_t \*);

/\* Получить общую массу продуктов за определённый период \*/

weight\_t lists\_GetWeightForPeriod (const list\_obj\_t \*, const udate\_t \*, const udate\_t \*);

#endif // !\_LISTS\_H

**Исходный код header-файла «main.h»**

#include <locale.h>

#include "files.h"

#include "tui.h"

#define FILENAME\_MAX\_LENGTH 32

const menu\_t MAIN\_MENU = {

L" Начать работу ",

L" О программе ",

L" Выйти из программы ",

NULL

};

const menu\_t SAVE\_OR\_LOAD\_MENU = {

L" Загрузить данные из файла ",

L" Сохранить данные в файл ",

L" Отменить действие ",

NULL

};

const menu\_t FILE\_FORMATS\_MENU = {

L" .bin ",

L" .csv ",

L" Отменить действие ",

NULL

};

const menu\_t LIST\_TYPES\_MENU = {

L" Все данные ",

L" Результат поиска ",

NULL

};

const menu\_t SELECT\_SEARCH\_FIELD\_MENU = {

L" По номеру позиции ",

L" По ID ",

L" По номеру волера ",

L" По ареалу ",

L" По названию породы ",

L" По имени животного ",

L" По типу продукта ",

L" По весу продукта ",

L" По стоимости продукта ",

L" По дате поступления ",

L" Отменить действие ",

NULL

};

const menu\_t SORT\_ORDER\_LIST = {

L" По возрастанию [или A→Z] ",

L" По убыванию [или Z→A] ",

NULL

};

const menu\_t DELETE\_ELEMENT\_MENU = {

L" Не удалять ",

L" Не удалять ",

L" Не удалять ",

L" Удалить ",

L" Не удалять ",

NULL

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Перечисление дополнительны функций, которые должны быть здесь просто потому что. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Нарисовать задний фон для заданного окна.

\*/

void draw\_win\_background (WINDOW \*);

/\* Подтвердить действие. Вернуть 0, если введён верный код \*/

int tui\_confirm\_action ();

**Исходный код header-файла «tui.h»**

#ifndef \_TUI\_POPUPS\_H

#define \_TUI\_POPUPS\_H

#include <stdio.h>

#include <stdlib.h>

#include <wchar.h>

#include <wctype.h>

#include <ncursesw/ncurses.h>

#include <ncursesw/panel.h>

#include "keyboard\_keys.h"

#include "cmp\_funcs.h"

#include "checker.h"

#include "lists.h"

#define TUI\_HIGHLIGHT\_ITEM\_STYLE ((A\_REVERSE) | (A\_BOLD))

#define TUI\_ADD\_ELEMENT\_CACELLED 0b00000

#define TUI\_ADD\_ELEMENT\_FLAG 0b00001

#define TUI\_AT\_BEGIN\_FLAG 0b00100

#define TUI\_AT\_END\_FLAG 0b00000

/\* Тип массива из пунктов меню \*/

typedef wchar\_t\* menu\_t[];

/\* Получить количество строк, занимаемых переданным текстом. Также учитываются символы перехода на новую строку \*/

int get\_message\_lines\_count (const wchar\_t \*, const int max\_width);

/\* Получить количество пунктов в меню \*/

int tui\_get\_menu\_item\_count (const menu\_t);

/\* Отрисовать вертикальное меню в заданном месте с заданным расстоянием между элементами \*/

void tui\_draw\_vmenu (WINDOW \*, const int, const int, const int, const menu\_t, const int);

/\* Нарисовать границы и заголовок для окна \*/

void tui\_draw\_popup\_header (WINDOW \*, const wchar\_t \*);

/\* Отобразить окно с обычным текстом \*/

void tui\_draw\_popup\_text\_message (const wchar\_t \*, const wchar\_t \*);

/\* Отобразить окно с выбором \*/

int tui\_draw\_popup\_select (const wchar\_t \*, const wchar\_t \*, const menu\_t);

/\* Отобразить окно с формой \*/

int tui\_draw\_popup\_form (const wchar\_t \*, const wchar\_t \*, const wchar\_t \*, wchar\_t \*, int, checker\_mode\_t);

/\* Отобразить окно с вводом даты \*/

void tui\_draw\_popup\_date (const wchar\_t \*, const wchar\_t \*, udate\_t \*);

/\* Отобразить окно с редактированием информации \*/

int tui\_popup\_edit\_element\_data (const wchar\_t \*, list\_data\_t \*, int);

/\* Отобразить окно с информацией об элементе \*/

void tui\_popup\_show\_only\_element (const wchar\_t \*, const list\_elem\_t \*);

/\*

\* Нарисовать таблицу в выбранном окне, отступив от начала списка на num\_page страниц.

\*/

int tui\_draw\_table\_in\_window (WINDOW \*, const list\_obj\_t, const length\_t, const int);

void tui\_print\_element\_info\_window (WINDOW \*, const int, const list\_elem\_t \*);

#endif // ! \_TUI\_POPUPS\_H

**Исходный код header-файла «wcs\_conv.h»**

#ifndef \_WCS\_CONV\_H

#define \_WCS\_CONV\_H

#include <wchar.h>

#include <wctype.h>

/\* Преобразовать WCS в целое число \*/

long long wcs2int (const wchar\_t \*);

/\* Преобразовать WCS в вещественное число \*/

long double wcs2float (const wchar\_t \*);

/\*

Преобразовать целое число в WCS (назначение должно вмещать не менее

20 символов)

\*/

void int2wcs (const long long, wchar\_t \*);

/\*

Преобразовать вещественное число в WCS (назначение должно вмещать не менее

20 символов)

\*/

void float2wcs (const long double, wchar\_t \*);

/\*

\* Нормализовать строку и Изменить размер строки до указанного.

\*/

void fitwcs (wchar\_t \*, const wchar\_t \*, const int);

#endif // !\_WCS\_CONV\_H

**Исходный код header-файла «\_\_forms\_\_.h»**

#ifndef \_\_\_FORMS\_\_\_H

#define \_\_\_FORMS\_\_\_H

#include <wchar.h>

#include <wctype.h>

#include <ncursesw/ncurses.h>

#include "cmp\_funcs.h"

#include "checker.h"

/\*

Нарисовать поле для формы.

\*/

void tui\_draw\_form\_field (WINDOW \*, int, int, const wchar\_t \*, const wchar\_t \*, int);

#endif //! \_\_\_FORMS\_\_\_H

**Исходный код header-файла «\_\_popups\_\_.h»**

#ifndef \_\_\_POPUPS\_\_\_H

#define \_\_\_POPUPS\_\_\_H

#include "tui.h"

// Нарисовать окно для попапов

void tui\_draw\_popup\_header (WINDOW \*, const wchar\_t \*);

#endif // ! \_\_\_POPUPS\_\_\_H

**Исходный код файла «main.c»**

#include "main.h"

int main ()

{

/\* Включить нужную локализацию \*/

setlocale(LC\_ALL, "");

/\* Задать начальные настройки ncurses \*/

initscr();

raw();

noecho();

keypad(stdscr, 1);

/\* Начало выполнения основной программы \*/

const int LAST\_CURS\_STATE = curs\_set (0);

wchar\_t tmp\_wcs[1025]; // Переменная для хранения временных строк

mkey\_t selected\_key = 0;

list\_obj\_t \* main\_list = lists\_CreateNewListObject (), // Список, содержащий все данные

\* save\_list = NULL, // Список, который нужно сохранить в файл

\* selected\_table\_list = main\_list, // Список, который нужно отображить в таблице

\* search\_results\_list = lists\_CreateNewListObject (); // Список с результатами поиска

int curr\_selected\_page = 1,

curr\_selected\_table\_row = 1,

drawed\_rows = 0;

draw\_win\_background (stdscr);

refresh ();

while (

(selected\_key = tui\_draw\_popup\_select (L"Программа для курсового проекта", L"Добро пожаловать в программу, созданную специально для курсового проекта.\nВыберите действие.", MAIN\_MENU))

!= 3

)

{

switch (selected\_key)

{

// Выбрано "Работа с программой"

case 1:

{

/\* Создать окна и панели для главного окна \*/

// Само окно

WINDOW \* win\_main = newwin (LINES - 2, COLS - 4, 1, 2);

PANEL \* panel\_main = new\_panel (win\_main);

update\_panels ();

// Таблица

WINDOW \* win\_table = derwin (win\_main, getmaxy(win\_main) - (3 + 1 + 1 + 6 + 3 + 1 + 1), getmaxx (win\_main) - 4, 4, 2);

PANEL \* panel\_table = new\_panel (win\_table);

/\* Отрисовать всё и вся \*/

tui\_draw\_popup\_header (win\_main, L"Таблица данных");

/\* Отрисовать подсказки ввода \*/

mvwprintw (win\_main, getmaxy(win\_main) - 12, 1, "Действия над таблицей: ");

mvwprintw (win\_main, getmaxy(win\_main) - 11, 1, " (← →) Выбор страницы данных. (↑ ↓) Выбор строки в таблице. ");

mvwprintw (win\_main, getmaxy(win\_main) - 9, 1, "Действия над данными: ");

mvwprintw (win\_main, getmaxy(win\_main) - 8, 1, " (P) Добавить данные. (V) Удалить данные. ");

mvwprintw (win\_main, getmaxy(win\_main) - 7, 1, " (W) Сохранить в файл/Загрузить из файла. ");

mvwprintw (win\_main, getmaxy(win\_main) - 6, 1, " (R) Просмотреть подробную информаци. (G) Редактировать элемент. ");

mvwprintw (win\_main, getmaxy(win\_main) - 5, 1, " (F) Поиск. (S) Сортировка. ");

mvwprintw (win\_main, getmaxy(win\_main) - 4, 1, " (T) Задание по варианту. ");

mvwprintw (win\_main, getmaxy(win\_main) - 3, 1, " (E) Выйти из текущего окна. ");

selected\_key = MKEY\_NULL;

do

{

/\* ---------------------------------------Обработка нажатой клавиши--------------------------------------- \*/

switch (selected\_key)

{

// Действие "Добавить элемент"

case L'P': case L'p': case L'З': case L'з':

{

if (selected\_table\_list == search\_results\_list)

{

tui\_draw\_popup\_text\_message (

L"Предупреждение",

L"Невозможно добавить элемент в результаты поиска. Переключитель на таблицу со всеми данными."

);

break;

}

list\_data\_t d;

switch (tui\_popup\_edit\_element\_data (L"Данные элемента", &d, 1))

{

case TUI\_ADD\_ELEMENT\_FLAG | TUI\_AT\_BEGIN\_FLAG:

lists\_InsertAsListsHead (main\_list, lists\_CreateNewElement(&d));

break;

case TUI\_ADD\_ELEMENT\_FLAG | TUI\_AT\_END\_FLAG:

lists\_InsertAsListsTail (main\_list, lists\_CreateNewElement(&d));

break;

}

}

break;

// Действие "Подробная информация"

case L'R': case L'r': case L'К': case L'к':

{

length\_t position = (curr\_selected\_page - 1) \* (getmaxy(win\_table) - 5) + curr\_selected\_table\_row;

list\_elem\_t \* element = lists\_SearchElementByField (selected\_table\_list, LIST\_POSITION, (void \*) &position);

if (element)

tui\_popup\_show\_only\_element (L"Информация о вольере", element);

else

tui\_draw\_popup\_text\_message (L"ОШИБКА", L"ОШИБКА: не удалось найти элемент.\nСкорее всего, это произошло потому, что отсутствуют данные.");

}

break;

// Действие "Удалить элемент таблицы"

case L'V': case L'v': case L'М': case L'м':

{

if (selected\_table\_list->\_length\_ < 1)

{

tui\_draw\_popup\_text\_message (L"Предупреждение", L"Невозможно удалить несуществующую строку таблицы.");

break;

}

length\_t position = (curr\_selected\_page - 1) \* (getmaxy(win\_table) - 5) + curr\_selected\_table\_row;

list\_elem\_t

\* tmp\_element = lists\_SearchElementByField (selected\_table\_list, LIST\_POSITION, (void \*) &position),

\* tmp\_element\_2 = NULL;

if (tui\_draw\_popup\_select (L"Опасное действие", L"ВНИМАНИЕ: вы хотите удалить элемент таблицы. Восстановить удалённые данные будет невозможно!\nВаше окончательное решение?", DELETE\_ELEMENT\_MENU) != 4 || tui\_confirm\_action () != 0)

{

tui\_draw\_popup\_text\_message (L"Предупреждение", L"Действие было отменено пользователем.");

break;

}

if (selected\_table\_list == main\_list)

{

if ((tmp\_element\_2 = lists\_SearchElementByField (search\_results\_list, LIST\_ID, &tmp\_element->id)))

{

if (search\_results\_list->head == tmp\_element\_2)

{

search\_results\_list->head = search\_results\_list->head->next;

if (search\_results\_list->head) search\_results\_list->head->prev = NULL;

}

lists\_DeleteElement (tmp\_element\_2);

search\_results\_list->\_length\_--;

}

}

else

{

if ((tmp\_element\_2 = lists\_SearchElementByField (main\_list, LIST\_ID, &tmp\_element->id)))

{

if (main\_list->head == tmp\_element\_2)

{

main\_list->head = main\_list->head->next;

if (main\_list->head) main\_list->head->prev = NULL;

}

lists\_DeleteElement (tmp\_element\_2);

main\_list->\_length\_--;

}

}

if (selected\_table\_list->head == tmp\_element)

{

selected\_table\_list->head = selected\_table\_list->head->next;

if (selected\_table\_list->head) selected\_table\_list->head->prev = NULL;

}

lists\_DeleteElement (tmp\_element);

selected\_table\_list->\_length\_--;

curr\_selected\_table\_row = 1;

if (drawed\_rows <= 1)

curr\_selected\_page--;

}

break;

// Действие "Редактировать элемент"

case L'G': case L'g': case L'П': case L'п':

{

length\_t position = (curr\_selected\_page - 1) \* (getmaxy(win\_table) - 5) + curr\_selected\_table\_row;

list\_elem\_t \* element = lists\_SearchElementByField (selected\_table\_list, LIST\_POSITION, (void \*) &position);

if (element)

{

if (tui\_confirm\_action () == 1)

{

tui\_draw\_popup\_text\_message (L"Предупреждение", L"Действие отменено пользователем.");

break;

}

tui\_popup\_edit\_element\_data (L"Редактировать запись", &element->data, 0);

if (selected\_table\_list == search\_results\_list)

{

lists\_SearchElementByField(main\_list, LIST\_ID, &element->id)->data = element->data;

}

}

else

tui\_draw\_popup\_text\_message (L"ОШИБКА", L"ОШИБКА: не удалось найти элемент.\nСкорее всего, это произошло потому, что в таблице отсутствуют данные.");

}

break;

// Действие "Сортировка"

case L'S': case L's': case L'Ы': case L'ы':

{

if (selected\_table\_list->\_length\_ < 1)

{

tui\_draw\_popup\_text\_message (

L"Ошибка",

L"Невозможно отсортировать пустую таблицу данных."

);

break;

}

int sort\_order = 0,

prio\_field = 0;

prio\_field = tui\_draw\_popup\_select (L"Выбор приоритетного поля", L"Выберите приоритетное поле (поле, по которому в первую очередь будет выполнена сортировка).", SELECT\_SEARCH\_FIELD\_MENU) - 1;

if (prio\_field > LIST\_DATE)

{

tui\_draw\_popup\_text\_message (L"Предупреждение", L"Сортировка была отменена пользователем.");

break;

}

if (tui\_draw\_popup\_select (L"Выбор порядка сортировки", L"Выберите порядок сортировки.", SORT\_ORDER\_LIST) == 1)

sort\_order = SORT\_ASCENDING\_ORDER;

else

sort\_order = SORT\_DESCENDING\_ORDER;

lists\_SortListByField (selected\_table\_list, sort\_order, prio\_field);

tui\_draw\_popup\_text\_message (L"Удача", L"Сортировка завершена.");

}

break;

// Действие "Поиск"

case L'F': case L'f': case L'А': case L'а':

{

if (selected\_table\_list->\_length\_ < 1)

{

tui\_draw\_popup\_text\_message (L"ОШИБКА", L"В выбранной таблице отсутствуют данные. Заполните или смените таблицу.");

break;

}

selected\_key = tui\_draw\_popup\_select (

L"Выбор поля для поиска",

L"Выберите поле для поиска.\nПосле чего необходимо будет ввести конкретное значений.",

SELECT\_SEARCH\_FIELD\_MENU

) - 1;

void \* universal\_ptr = NULL;

list\_id\_t tmp\_id = 0;

length\_t tmp\_int = 0;

weight\_t tmp\_weight = 0.0;

cost\_t tmp\_cost = 0.0;

udate\_t tmp\_date = {.D = 1, .M = 9, .Y = 2021};

list\_elem\_t \* tmp\_element\_ptr = NULL,

\* tmp\_created\_element = NULL;

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

switch (selected\_key)

{

case LIST\_POSITION:

tui\_draw\_popup\_form (

L"Поиск по позиции",

L"Введите требуемый номер позиции.",

L"POS",

tmp\_wcs,

POSITION\_DGT\_CNT,

VMASK\_DIGITS

);

tmp\_int = wcs2int (tmp\_wcs);

universal\_ptr = &tmp\_int;

break;

case LIST\_ID:

tui\_draw\_popup\_form (

L"Поиск по ID",

L"",

L"ID",

tmp\_wcs,

ID\_DGT\_CNT,

VMASK\_DIGITS

);

tmp\_id = wcs2int (tmp\_wcs);

universal\_ptr = &tmp\_id;

break;

case LIST\_CAGE\_NUMBER:

tui\_draw\_popup\_form (

L"Поиск по номеру вольера",

L"",

L"вольер",

tmp\_wcs,

CAGE\_NUM\_DGT\_CNT,

VMASK\_DIGITS

);

tmp\_int = wcs2int (tmp\_wcs);

universal\_ptr = &tmp\_int;

break;

case LIST\_ANIMAL\_AREAL:

tui\_draw\_popup\_form (L"Поиск по ареалу", L"Введите ареал обитания животного.", L"ареал", tmp\_wcs, ANIMAL\_AREAL\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

universal\_ptr = tmp\_wcs;

break;

case LIST\_ANIMAL\_BREED:

tui\_draw\_popup\_form (L"Поиск по породе", L"Введите породу животного.", L"порода", tmp\_wcs, ANIMAL\_BREED\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

universal\_ptr = tmp\_wcs;

break;

case LIST\_ANIMAL\_NAME:

tui\_draw\_popup\_form (L"Поиск по имени", L"Введите имя животного.", L"имя", tmp\_wcs, ANIMAL\_NAME\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

universal\_ptr = tmp\_wcs;

break;

case LIST\_PRODUCT\_TYPE:

tui\_draw\_popup\_form (L"Поиск по типу", L"Введите тип продукта.", L"тип", tmp\_wcs, PRODUCT\_TYPE\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

universal\_ptr = tmp\_wcs;

break;

case LIST\_PRODUCT\_WEIGHT:

tui\_draw\_popup\_form (L"Поиск по весу", L"Введите вес.", L"вес", tmp\_wcs, WEIGHT\_DGT\_CNT, VMASK\_DIGITS | VMASK\_PUNCTS);

tmp\_weight = wcs2float (tmp\_wcs);

universal\_ptr = &tmp\_weight;

break;

case LIST\_PRODUCT\_COST:

tui\_draw\_popup\_form (L"Поиcк по стоимоти", L"Введите стоимость.", L"стоимос", tmp\_wcs, COST\_DGT\_CNT, VMASK\_DIGITS | VMASK\_PUNCTS);

tmp\_cost = wcs2float (tmp\_wcs);

universal\_ptr = &tmp\_cost;

break;

case LIST\_DATE:

tui\_draw\_popup\_date (

L"Поиск по дате",

L"Введите дату.",

&tmp\_date

);

universal\_ptr = &tmp\_date;

break;

default:

tui\_draw\_popup\_text\_message (

L"Предупреждение",

L"Поиск данных был отменён пользователем."

);

break;

}

if (selected\_key > LIST\_DATE)

break;

// Если предстоит уточнить поиск, т.е. найти какие-то элементы в результатах поиска, ...

if (selected\_table\_list == search\_results\_list)

// ... то поудалять неподходящие элементы из результатов поиска без изменения исходных данных ...

{

for (tmp\_element\_ptr = selected\_table\_list->head; tmp\_element\_ptr;)

{

if (tmp\_element\_ptr != lists\_SearchElementByFieldFromThisElement (tmp\_element\_ptr, selected\_key, universal\_ptr))

{

if (selected\_table\_list->head == tmp\_element\_ptr)

{

selected\_table\_list->head = tmp\_element\_ptr->next;

lists\_DeleteElement (tmp\_element\_ptr);

tmp\_element\_ptr = selected\_table\_list->head;

}

else

{

if (tmp\_element\_ptr->next)

{

tmp\_element\_ptr = tmp\_element\_ptr->next;

lists\_DeleteElement (tmp\_element\_ptr->prev);

}

else

{

lists\_DeleteElement (tmp\_element\_ptr);

tmp\_element\_ptr = NULL;

}

}

selected\_table\_list->\_length\_--;

}

else

{

tmp\_element\_ptr = tmp\_element\_ptr->next;

}

}

}

else

// ... иначе надо создать новый список с результатами поиска

{

lists\_CleanListObject (search\_results\_list);

tmp\_element\_ptr = selected\_table\_list->head;

while (tmp\_element\_ptr = lists\_SearchElementByFieldFromThisElement (tmp\_element\_ptr, selected\_key, universal\_ptr))

{

tmp\_created\_element = lists\_CreateNewElement (&tmp\_element\_ptr->data);

lists\_InsertAsListsTail (search\_results\_list, tmp\_created\_element);

tmp\_created\_element->id = tmp\_element\_ptr->id;

tmp\_element\_ptr = tmp\_element\_ptr->next;

}

selected\_table\_list = search\_results\_list;

}

curr\_selected\_table\_row = 1;

curr\_selected\_page = 1;

}

break;

// Действие "Задание по варианту"

case L'T': case L't': case L'Е': case L'е':

{

/\*

if (selected\_table\_list == search\_results\_list)

{

tui\_draw\_popup\_text\_message (

L"Предупреждение",

L"Невозможно выполнить задание по варианту в отношении результатов поиска."

);

break;

}

\*/

if (selected\_table\_list->\_length\_ < 1)

{

tui\_draw\_popup\_text\_message (

L"ОШИБКА",

L"Запрос невозможно выполнить для пустого списка."

);

break;

}

udate\_t d\_1, d\_2;

weight\_t tmp\_weight = 0.0;

cost\_t tmp\_cost = 0.0;

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

tui\_draw\_popup\_date (

L"Ввод даты",

L"Введите раннюю дату.",

&d\_1

);

tui\_draw\_popup\_date (

L"Ввод даты",

L"Введите позднюю дату.",

&d\_2

);

tmp\_weight = lists\_GetWeightForPeriod (selected\_table\_list, &d\_1, &d\_2);

tmp\_cost = lists\_GetCostForPeriod (selected\_table\_list, &d\_1, &d\_2);

swprintf (tmp\_wcs, sizeof(tmp\_wcs), L"Общая масса за указанный период: %Lf г.\nОбщая стоимость продуктов за указанный период: %Lf Р.", tmp\_weight, tmp\_cost);

tui\_draw\_popup\_text\_message (L"Результаты", tmp\_wcs);

}

break;

// Изменение выбранной строки

case MKEY\_ARROW\_DOWN:

if (curr\_selected\_table\_row < drawed\_rows)

curr\_selected\_table\_row++;

break;

case MKEY\_ARROW\_UP:

if (curr\_selected\_table\_row > 1)

curr\_selected\_table\_row--;

break;

// Изменение текущей страницы

case MKEY\_ARROW\_LEFT:

if (curr\_selected\_page > 1)

{

curr\_selected\_page--;

curr\_selected\_table\_row = 1;

}

break;

case MKEY\_ARROW\_RIGHT:

if (curr\_selected\_page < selected\_table\_list->\_length\_ / (getmaxy(win\_table) - 5) + !!(selected\_table\_list->\_length\_ % (getmaxy(win\_table) - 5)))

{

curr\_selected\_page++;

curr\_selected\_table\_row = 1;

}

break;

// Изменение выбранной таблицы

case L'N': case L'n': case L'Т': case L'т':

if (selected\_table\_list == search\_results\_list)

selected\_table\_list = main\_list;

else

selected\_table\_list = search\_results\_list;

curr\_selected\_table\_row = 1;

curr\_selected\_page = 1;

break;

// Действие "Работа с файлами"

case L'W': case L'w': case L'Ц': case L'ц':

// Выбор действия на файлами

selected\_key = tui\_draw\_popup\_select (

L"Сохранить/Загрузить",

L"Вы хотите ЗАГРУЗИТЬ данные из файла или СОХРАНИТЬ данные в файл?",

SAVE\_OR\_LOAD\_MENU

);

switch (selected\_key)

{

// Загрузить данные

case 1:

/\* Выбрать тип загружаемого файла \*/

selected\_key = tui\_draw\_popup\_select (

L"Загрузить данные",

L"Выберите тип файла, в котором хранятся данные:\n\* .bin -> Специальный формат для хранения данных в бинарном формате. Только эта и никакая другая программа может расшифровать сохранённые данные.\n\* .csv -> Текстовый формат хранения данных. Таким образом вы можете внести данные как из Текстового редактора, так и импортировать из других программ, например, MS Excel.",

FILE\_FORMATS\_MENU

);

if (selected\_key == 3)

{

tui\_draw\_popup\_text\_message (

L"Предупреждение",

L"Загрузка данных из файла отменена пользователем."

);

break;

}

/\* Обнулить значение tmp\_wcs и Запросить имя загружаемого файла файла \*/

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

tui\_draw\_popup\_form (

L"Имя файла",

L"Осталось ввести имя файла.",

L" Введите имя файла",

tmp\_wcs,

FILENAME\_MAX\_LENGTH,

VMASK\_ANY\_CHAR | VMASK\_DIGITS | VMASK\_SPACES | VMASK\_PUNCTS | VMASK\_ADD\_FS\_CHARS

);

/\* Проверить на существование файл \*/

if (is\_exist\_wcs (tmp\_wcs))

{

tui\_draw\_popup\_text\_message (

L"ОШИБКА!",

L"Невозможно получить доступ к файлу.\nВозможно файл не существует или его нельзя прочитать.\nВ следующий раз введите достижимый файл."

);

break;

}

/\* Всё ок. Очистить данные в старых списках и загрузить новые данные \*/

lists\_CleanListObject (main\_list);

lists\_CleanListObject (search\_results\_list);

switch (selected\_key)

{

case 1:

read\_from\_bin (main\_list, tmp\_wcs);

break;

case 2:

read\_from\_csv (main\_list, tmp\_wcs);

break;

}

selected\_table\_list = main\_list;

curr\_selected\_table\_row = 1;

curr\_selected\_page = 1;

tui\_draw\_popup\_text\_message (

L"УСПЕХ!",

L"Все данные были считаны."

);

break;

// Сохранить данные

case 2:

/\* Какие данные нужно сохранить \*/

selected\_key = tui\_draw\_popup\_select (

L"Сохранить ...",

L"Выберите список, который хотите сохранить.",

LIST\_TYPES\_MENU

);

/\* Выбрать список для сохранения и проверить список на наличие данных \*/

save\_list = (selected\_key == 1) ? (main\_list) : (search\_results\_list);

if (save\_list->\_length\_ < 1)

{

tui\_draw\_popup\_text\_message (

L"ОШИБКА!",

L"Похоже, что данные отсутствуют, а значит нечего сохранять.\nДля начала добавьте какие-то данные или смените сохраняемую таблицу."

);

break;

}

/\*Всё ок. Выбрать тип сохраняемого файла \*/

selected\_key = tui\_draw\_popup\_select (

(save\_list == main\_list) ? L"Сохранить все данные как ..." : L"Сохранить результаты поиска как ...",

L"Выберите тип файла для сохранения:\n\* .bin -> Специальный формат для хранения данных в бинарном формате. Злоумышленнику не сможет скомпрометировать данные.\n\* .csv -> Текстовый формат хранения данных. Возможность как редактирования Текстовыми редакторами, так и экспорта в другие программы, например, MS Excel.",

FILE\_FORMATS\_MENU

);

/\* Выйти, если пользователь запросил отмену \*/

if (selected\_key == 3)

{

tui\_draw\_popup\_text\_message (L"Предупреждение", L"Отмена сохранения данных в файл.");

break;

}

/\* Обнулить значение tmp\_wcs и Запросить имя файла для сохранения \*/

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

tui\_draw\_popup\_form (

L"Имя файла",

L"Осталось ввести имя файла.",

L" Введите имя файла",

tmp\_wcs,

FILENAME\_MAX\_LENGTH,

VMASK\_ANY\_CHAR | VMASK\_DIGITS | VMASK\_SPACES | VMASK\_PUNCTS | VMASK\_ADD\_FS\_CHARS

);

if (!wcslen (tmp\_wcs))

{

tui\_draw\_popup\_text\_message (L"Предупреждение", L"Данные не были введены. Отмена сохранения.");

break;

}

switch (selected\_key)

{

// Сохранить как бинарный файл

case 1:

save\_as\_bin (save\_list, tmp\_wcs);

break;

// Сохранить как текстовый файл

case 2:

save\_as\_csv (save\_list, tmp\_wcs);

break;

}

if (!is\_exist\_wcs (tmp\_wcs))

tui\_draw\_popup\_text\_message (

L"УСПЕХ!",

L"Все данные были сохранены."

);

else

tui\_draw\_popup\_text\_message (

L"НЕУДАЧА!",

L"Данные не были сохранены. Пожалуйста, выберите другой путь."

);

break;

}

break;

}

wclear (win\_table);

drawed\_rows = tui\_draw\_table\_in\_window (win\_table, \*selected\_table\_list, curr\_selected\_page, curr\_selected\_table\_row);

if (selected\_table\_list != search\_results\_list)

mvwprintw (win\_main, getmaxy(win\_main) - 10, 1, " (N) Показать прошлые результаты поиска. ");

else

mvwprintw (win\_main, getmaxy(win\_main) - 10, 1, " (N) Показать все данные. ");

update\_panels ();

doupdate ();

}

while ((selected\_key = getwchar()) != L'E' && selected\_key != L'e');

/\* Удалить панели и окна \*/

del\_panel (panel\_table);

del\_panel (panel\_main);

delwin (win\_table);

delwin (win\_main);

}

break;

// Выбрано "О программе"

case 2:

tui\_draw\_popup\_text\_message (

L"О программе",

L"Программа была написана для курсового проекта для предмета \"АиП\" (3 семестр).\n\* Создатель: Филозоп Алексей.\n\* Исходный код: github.com/Cory7666/3semester\_AaP\_Coursework\_new.git"

);

break;

}

}

/\* Завершить выполнение программы \*/

lists\_DeleteListObject (&main\_list);

curs\_set (LAST\_CURS\_STATE);

endwin();

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Блок дополнительных функций. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void draw\_win\_background (WINDOW \* win)

{

wmove (win, 0, 0);

for (size\_t i = 0; i < (int) (getmaxx(win) \* getmaxy(win) / 6); i++)

{

wprintw (win, "\*@|@\*-");

}

}

int tui\_confirm\_action ()

{

int control\_code = 0;

wchar\_t tmp\_message[100],

tmp\_answer[10];

while (1)

{

memset (tmp\_message, 0, sizeof(tmp\_message));

memset (tmp\_answer, 0, sizeof(tmp\_answer));

control\_code = rand() % 8999 + 1000;

swprintf (tmp\_message, sizeof(tmp\_message), L"Для продолжения введите код ниже:\n %4d\nДля отмены просто введите:\n 0.", control\_code);

tui\_draw\_popup\_form (L"", tmp\_message, L"код", tmp\_answer, 4, VMASK\_DIGITS);

if (wcs2int (tmp\_answer) == 0)

return 1;

else if (wcs2int (tmp\_answer) == control\_code)

return 0;

}

}

**Исходный код файла «date.c»**

#include "date.h"

void date2wcsdp (const udate\_t \* date, wchar\_t \* string, const wchar\_t delimiter, const wchar\_t \* pattern)

{

/\* Выйти, если дата не была передана \*/

if (!date) return;

swprintf(string, wcslen(pattern) + 1, pattern, date->D, delimiter, date->M, delimiter, date->Y);

}

void wcs2date (udate\_t \* date, const wchar\_t \* string)

{

/\* Выйти, если дата не была передана \*/

if (!date) return;

size\_t offset = 0;

for (; iswspace(string[offset]); offset++);

date->D = wcs2int(string + offset);

date->M = wcs2int(string + offset + 3);

date->Y = wcs2int(string + offset + 6);

}

int datecmp (const udate\_t \* d1, const udate\_t \* d2)

{

/\* Выйти, если одна из дат не была передана \*/

if (!(d1 && d2)) return 0;

int result = 0;

// Сначала сравнить по полю Год

if (result = cmp\_nums(d1->Y, d2->Y))

return result;

else

// Если равны, то сравнить по полю Месяц

if (result = cmp\_nums(d1->M, d2->M))

return result;

else

// Если предыдущие поля равны, сравнить по полю день и незамедлительно вернуть результат

return cmp\_nums(d1->D, d2->D);

}

**Исходный код файла «bin\_files.c»**

#include "files.h"

int save\_as\_bin (const list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs)

{

/\* Мгновенный выход из подпрограммы, если передан пустой указатель на объект списка \*/

if (!list) return FS\_NO\_LIST\_PTR;

/\* Мгновенный выход из подпрограммы, если передан объект списка без данных \*/

if (1 > list->\_length\_) return FS\_NO\_LIST\_DATA;

/\* Перевести имя файла из WCS в MBS \*/

char file\_name\_mbs[(wcslen(file\_name\_wcs) + 1) \* sizeof(wchar\_t)];

wcstombs(file\_name\_mbs, file\_name\_wcs, wcslen(file\_name\_wcs) \* sizeof(wchar\_t));

FILE \* f = fopen(file\_name\_mbs, "w");

/\* Мгновенно выйти из подпрограммы, если невозможно создать файл \*/

if (!f) return FS\_FILE\_NO\_WRITE\_ACCESS;

/\* Перебор элементов и запись каждого в файл \*/

for (list\_elem\_t \* e = list->head; e; e = e->next)

fwrite(&(e->data), sizeof(list\_data\_t), 1, f);

/\* Закрыть файл и выйти \*/

fclose(f);

return FS\_SUCCESS;

}

int read\_from\_bin (list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs)

{

/\* Выйти, если файл не существует или к нему нет доступа \*/

if (is\_exist\_wcs(file\_name\_wcs) != FS\_SUCCESS) return FS\_FILE\_NO\_READ\_ACCESS;

/\* Перевод имени файла в MBS и открытие файла \*/

char file\_name\_mbs[(wcslen(file\_name\_wcs) + 1) \* sizeof(wchar\_t)];

wcstombs(file\_name\_mbs, file\_name\_wcs, wcslen(file\_name\_wcs) \* sizeof(wchar\_t));

FILE \* f = fopen(file\_name\_mbs, "r");

/\* Проверка целостности содержащихся данных \*/

size\_t currpos = ftell(f); fseek(f, 0, SEEK\_END);

if ( ((ftell(f) - currpos) % sizeof(list\_data\_t)) != 0 ) { fclose(f); return FS\_BROKEN\_DATA; }

fseek(f, currpos, SEEK\_SET);

/\* Всё ок, значит можно удалить данные в списке \*/

lists\_CleanListObject (list);

/\* Поэлементное считывание данных из файла и добавление в объект списка \*/

list\_data\_t d;

while(fread(&d, sizeof(list\_data\_t), 1, f)) lists\_InsertAsListsTail(list, lists\_CreateNewElement(&d));

/\* Закрыть файл и выйти из подпрограммы \*/

fclose(f);

return FS\_SUCCESS;

}

**Исходный код файла «csv\_files.c»**

#include "files.h"

int save\_as\_csv (const list\_obj\_t \* list, const wchar\_t \* file\_name\_wcs)

{

/\* Мгновенный выход из подпрограммы, если передан пустой указатель на объект списка \*/

if (!list) return FS\_NO\_LIST\_PTR;

/\* Мгновенный выход из подпрограммы, если передан объект списка без данных \*/

if (1 > list->\_length\_) return FS\_NO\_LIST\_DATA;

/\* Всё хорошо, продолжить выполнение \*/

/\* Ввести дополнительные переменные \*/

length\_t position = 1; // Порядковый номер элемента в списке

char file\_name\_mbs[(wcslen(file\_name\_wcs) + 1) \* sizeof(wchar\_t)]; // Имя файла как MBS

wchar\_t date\_wcs[20]; // MBS для хранения даты

/\* Перевести из WCS в MBS \*/

wcstombs(file\_name\_mbs, file\_name\_wcs, sizeof(file\_name\_mbs));

/\* Открыть файл для записи \*/

FILE \* f = fopen(file\_name\_mbs, "w");

/\* Разместить заголовок таблицы \*/

//fprintf(f, "Порядковый номер%cНомер клетки%cИмя животного%cПорода животного%cАреал обитания животного%cТип продуктов%cВес продуктов%cСтоимость продуктов%cДата\n", CSV\_DELIMITER, CSV\_DELIMITER, CSV\_DELIMITER, CSV\_DELIMITER, CSV\_DELIMITER, CSV\_DELIMITER, CSV\_DELIMITER, CSV\_DELIMITER);

/\* Сохранение данных в файл \*/

for (list\_elem\_t \* e = list->head; e; e = e->next, position++)

{

/\* Перевод даты в MBS \*/

date2wcs(&e->data.date, date\_wcs);

/\* Записать данные в файл \*/

fprintf(f, "%lu%c%lu%c%ls%c%ls%c%ls%c%ls%c%.3Lf%c%.2Lf%c%ls\n",

position, CSV\_DELIMITER,

e->data.cage\_num, CSV\_DELIMITER,

e->data.animal.name, CSV\_DELIMITER,

e->data.animal.breed, CSV\_DELIMITER,

e->data.animal.areal, CSV\_DELIMITER,

e->data.products.type, CSV\_DELIMITER,

e->data.products.weight, CSV\_DELIMITER,

e->data.products.cost, CSV\_DELIMITER,

date\_wcs

);

/\* Обновить данные в файле \*/

fflush(f);

}

/\* Закрыть файл и выйти из подпрограммы \*/

fclose(f);

return FS\_SUCCESS;

}

int read\_from\_csv (list\_obj\_t \* list, const wchar\_t \* fileName)

{

if (is\_exist\_wcs(fileName) != FS\_SUCCESS) return FS\_FILE\_NO\_READ\_ACCESS; // Если файл нельзя открыть.

setlocale(LC\_ALL, "");

// Открытие файла для чтения.

char fileName\_char[(wcslen(fileName) + 1) \* sizeof(wchar\_t)];

wcstombs(fileName\_char, fileName, wcslen(fileName) \* sizeof(wchar\_t));

FILE \* f = fopen(fileName\_char, "r");

// Очистка списка, если в нём были какие-то данные

if (list->head) { lists\_DeleteListObject(&list); list = lists\_CreateNewListObject(); }

size\_t tmp\_line\_size = 10;

char \*tmp\_line = calloc(tmp\_line\_size, sizeof(char)), // Содержит строку из файла

tmp\_str[2048], // Содержит обрабатываемое слово

\*pos1 = NULL, \*pos2 = NULL; // Позиции при обработке строки

wchar\_t tmp\_wstr[2048];

list\_data\_t d;

while (!feof(f))

{

// Считать строку из файла

if (getline(&tmp\_line, &tmp\_line\_size, f) < 4) continue;

if (tmp\_line[strlen(tmp\_line) - 1] == '\n') tmp\_line[strlen(tmp\_line) - 1] = '\0';

memset(&d, 0, sizeof(d));

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

// Пропуск порядкового номера

pos1 = tmp\_line; pos2 = strchr(pos1, CSV\_DELIMITER);

pos1 = pos2 + 1;

// Обработка номера вольера

pos2 = strchr(pos1, CSV\_DELIMITER);

strncpy(tmp\_str, pos1, pos2 - pos1); // Копирование фрагмента строки в другую строку

d.cage\_num = atoi(tmp\_str); // Перевод в число и запись в структуру

pos1 = pos2 + 1; // Передвинуть позицию

// Обработка имени животного

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

pos2 = strchr(pos1, CSV\_DELIMITER);

strncpy(tmp\_str, pos1, pos2 - pos1); // Копирование фрагмента строки в другую строку

mbstowcs(tmp\_wstr, tmp\_str, sizeof(tmp\_str) \* sizeof(wchar\_t)); // Перевод из MBS в WCS

wcscpy(d.animal.name, tmp\_wstr); // Копирование в соответствующее поле

pos1 = pos2 + 1; // Передвинуть позицию

// Обработка породы животного

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

pos2 = strchr(pos1, CSV\_DELIMITER);

strncpy(tmp\_str, pos1, pos2 - pos1); // Копирование фрагмента строки в другую строку

mbstowcs(tmp\_wstr, tmp\_str, strlen(tmp\_str)); // Перевод из MBS в WCS

wcscpy(d.animal.breed, tmp\_wstr); // Копирование в соответствующее поле

pos1 = pos2 + 1; // Передвинуть позицию

// Обработка ареала обитания животного

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

pos2 = strchr(pos1, CSV\_DELIMITER);

strncpy(tmp\_str, pos1, pos2 - pos1); // Копирование фрагмента строки в другую строку

mbstowcs(tmp\_wstr, tmp\_str, strlen(tmp\_str)); // Перевод из MBS в WCS

wcscpy(d.animal.areal, tmp\_wstr); // Копирование в соответствующее поле

pos1 = pos2 + 1; // Передвинуть позицию

// Обработка типа продуктов животного

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

pos2 = strchr(pos1, CSV\_DELIMITER);

strncpy(tmp\_str, pos1, pos2 - pos1); // Копирование фрагмента строки в другую строку

mbstowcs(tmp\_wstr, tmp\_str, strlen(tmp\_str)); // Перевод из MBS в WCS

wcscpy(d.products.type, tmp\_wstr); // Копирование в соответствующее поле

pos1 = pos2 + 1; // Передвинуть позицию

// Обработка веса продуктов

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

pos2 = strchr(pos1, CSV\_DELIMITER);

strncpy(tmp\_str, pos1, pos2 - pos1); // Копирование фрагмента строки в другую строку

d.products.weight = atof(tmp\_str); // Перевод в число и запись в структуру

pos1 = pos2 + 1; // Передвинуть позицию

// Обработка стоимости продуктов

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

pos2 = strchr(pos1, CSV\_DELIMITER);

strncpy(tmp\_str, pos1, pos2 - pos1); // Копирование фрагмента строки в другую строку

d.products.cost = atof(tmp\_str); // Перевод в число и запись в структуру

pos1 = pos2 + 1; // Передвинуть позицию

// Обработка даты

memset(tmp\_str, 0, sizeof(tmp\_str));

memset(tmp\_wstr, 0, sizeof(tmp\_wstr));

strcpy(tmp\_str, pos1);// Копирование фрагмента строки в другую строку

mbstowcs(tmp\_wstr, tmp\_str, strlen(tmp\_str)); // Перевод из MBS в WCS

wcs2date(&d.date, tmp\_wstr); // Перевод строки в дату и запись в структуру

lists\_InsertAsListsTail(list, lists\_CreateNewElement(&d));

}

fclose(f);

return FS\_SUCCESS;

}

**Исходный код файла «files.c»**

#include "files.h"

int is\_exist (const char \* file\_name)

{

FILE \*f = fopen(file\_name, "r");

/\* Файл не существует или прочитать его невозможно \*/

if (!f) return FS\_FILE\_NO\_READ\_ACCESS;

/\* Файл существует и доступен для чтения \*/

fclose(f);

return FS\_SUCCESS;

}

int is\_exist\_wcs (const wchar\_t \* file\_name\_wcs)

{

char file\_name\_mbs[(wcslen(file\_name\_wcs) + 1) \* sizeof(wchar\_t)];

wcstombs(file\_name\_mbs, file\_name\_wcs, wcslen(file\_name\_wcs) \* sizeof(wchar\_t));

return is\_exist(file\_name\_mbs);

}

**Исходный код файла «lists\_cmp\_elems.c»**

#include "lists.h"

int lists\_CompareElementsByOneField (const list\_elem\_t \* e1, const list\_elem\_t \* e2, const field\_num\_t field)

{

int result = 0;

switch (field)

{

case LIST\_ID:

result = cmp\_nums(e1->id, e2->id);

break;

case LIST\_CAGE\_NUMBER:

result = cmp\_nums(e1->data.cage\_num, e2->data.cage\_num);

break;

case LIST\_ANIMAL\_AREAL:

result = wcscmp(e1->data.animal.areal, e2->data.animal.areal);

break;

case LIST\_ANIMAL\_BREED:

result = wcscmp(e1->data.animal.breed, e2->data.animal.breed);

break;

case LIST\_ANIMAL\_NAME:

result = wcscmp(e1->data.animal.name, e2->data.animal.name);

break;

case LIST\_PRODUCT\_TYPE:

result = wcscmp(e1->data.products.type, e2->data.products.type);

break;

case LIST\_PRODUCT\_COST:

result = cmp\_nums(e1->data.products.cost, e2->data.products.cost);

break;

case LIST\_PRODUCT\_WEIGHT:

result = cmp\_nums(e1->data.products.weight, e2->data.products.weight);

break;

case LIST\_DATE:

result = datecmp(&e1->data.date, &e2->data.date);

break;

}

return result;

}

int lists\_CompareElements (const list\_elem\_t \* e1, const list\_elem\_t \* e2, const field\_num\_t prio\_field)

{

int result = 0;

/\* Завершить подпрогрумму, так как элементы нельзя сравнить по позиции \*/

if (LIST\_POSITION == prio\_field) return 0;

/\* Сравнить по приоритетному полю \*/

result = lists\_CompareElementsByOneField(e1, e2, prio\_field);

if (result) return result;

/\* Сравнить по остальным полям \*/

for (int i = LIST\_ID + 1; !result && i <= LIST\_DATE; i++)

if (i != prio\_field)

result = lists\_CompareElementsByOneField(e1, e2, i);

/\* Вернуть окончательный результат сравнения \*/

return result;

}

**Исходный код файла «lists\_sort.c»**

#include "lists.h"

void lists\_SortListByField (list\_obj\_t \* list, const int sort\_order, const field\_num\_t prio\_field)

{

if (!list) return;

if (2 > list->\_length\_) return;

list\_elem\_t \*e1, \*e2;

list\_data\_t tmp\_d; list\_id\_t tmp\_id;

for (length\_t i = 1; i < list->\_length\_; i++)

{

e1 = lists\_SearchElementByField(list, LIST\_POSITION, &i);

for (length\_t j = i + 1; j < list->\_length\_ + 1; j++)

{

e2 = lists\_SearchElementByField(list, LIST\_POSITION, &j);

if (lists\_CompareElements(e1, e2, prio\_field) \* sort\_order > 0)

{

memcpy(&tmp\_d, &e1->data, sizeof(list\_data\_t)); tmp\_id = e1->id;

memcpy(&e1->data, &e2->data, sizeof(list\_data\_t));

e1->id = e2->id;

memcpy(&e2->data, &tmp\_d, sizeof(list\_data\_t));

e2->id = tmp\_id;

}

}

}

}

**Исходный код файла «lists.c»**

#include "lists.h"

list\_obj\_t \* lists\_CreateNewListObject (void)

{

return ((list\_obj\_t \*) calloc(sizeof(list\_obj\_t), 1));

}

void lists\_DeleteListObject (list\_obj\_t \*\* list)

{

/\* Удаление данных из списка \*/

lists\_CleanListObject(\*list);

/\* Освобождение памяти \*/

free(\*list);

\*list = NULL;

return;

}

void lists\_CleanListObject (list\_obj\_t \* list)

{

/\* Немедленный выход из подпрограммы, если передан пустой указатель \*/

if (!list) return;

/\* Немедленный выход из подпрограммы, если объект не содержит данных \*/

if (1 > list->\_length\_) return;

list\_elem\_t \*e = list->head;

/\* Циклическое удаление данных из объекта \*/

while (e->next) free((e = e->next)->prev);

free(e);

/\* Обнуление длины списка и идентификатора \*/

list->head = NULL;

list->\_length\_ = 0;

list->\_last\_id\_ = 0;

return;

}

list\_elem\_t \* lists\_CreateNewElement (list\_data\_t \* data)

{

/\* Немедленный выход из программы, если данные не были переданы \*/

if (!data) return NULL;

/\* Выделить память для нового элемента. Обнулить память \*/

list\_elem\_t \* e = malloc(sizeof(list\_elem\_t));

memset(e, 0, sizeof(list\_elem\_t));

/\* Записать данные в элемент \*/

memcpy(&e->data, data, sizeof(list\_data\_t));

/\* Завершить выполнение подпрограммы \*/

return e;

}

void lists\_InsertAfterElement (list\_elem\_t \* e1, list\_elem\_t \* e2)

{

/\* Немедленные выход, если один из параметров NULL \*/

if (!e1 || !e2) return;

e1->next = e2->next;

if (e1->next) e1->next->prev = e1;

e1->prev = e2;

e2->next = e1;

return;

}

void lists\_DeleteElement (list\_elem\_t \* e)

{

/\* Немедленный выход, если передан пустой указатель \*/

if (!e) return;

/\* Вытащить элемент из списка и очитить память \*/

list\_DetachElement(e);

free(e);

return;

}

void list\_DetachElement (list\_elem\_t \* e)

{

/\* Выход из подпрограммы, если передан пустой указатель \*/

if (!e) return;

list\_elem\_t \*e\_left = e->prev, \*e\_right = e->next;

/\* Пересвязка \*/

if (e\_left) e\_left->next = e\_right;

if (e\_right) e\_right->prev = e\_left;

return;

}

void lists\_InsertAsListsHead (list\_obj\_t \* list, list\_elem\_t \* e)

{

/\* Немедленный выход, если передан пустой указатель на объект списка \*/

if (!list) return;

/\* Немедленно выйти, если не был передан указатель на элемент \*/

if (!e) return;

e->next = list->head;

if (e->next) e->next->prev = e;

list->head = e;

e->prev = NULL;

/\* Обновить Id элемента и длину списка\*/

e->id = list->\_last\_id\_++;

list->\_length\_++;

return;

}

void lists\_InsertAsListsTail (list\_obj\_t \* list, list\_elem\_t \* e)

{

/\* Немедленный выход, если передан пустой указатель на объект списка \*/

if (!list) return;

if (1 > list->\_length\_) lists\_InsertAsListsHead (list, e);

else

{

list\_elem\_t \* f = lists\_SearchElementByField(list, LIST\_POSITION, &list->\_length\_);

lists\_InsertAfterElement(e, f);

e->id = list->\_last\_id\_++;

list->\_length\_++;

}

return;

}

list\_elem\_t \* lists\_SearchElementByField (list\_obj\_t \* list, field\_num\_t field, void \* data)

{

return lists\_SearchElementByFieldFromThisElement (list->head, field, data);

}

list\_elem\_t \* lists\_SearchElementByFieldFromThisElement (list\_elem\_t \* e, field\_num\_t field, void \* data)

{

/\* Немедленный выход из подпрограммы, если передан пустой указатель на элемент' списка \*/

if (!e) return NULL;

/\* Немедленный выход из подпрограммы, если передан некорректный номер поля \*/

if ( field < LIST\_POSITION || field > LIST\_DATE ) return NULL;

length\_t position\_counter = 1;

short exit\_flag = 0;

/\* Пытаться найти элемент, пока не будет достигнут конец списка или искомый элемент не будет найден \*/

while (e && !exit\_flag)

{

switch (field)

{

case LIST\_POSITION:

exit\_flag = ( \*((length\_t \*) data) == position\_counter );

break;

case LIST\_ID:

exit\_flag = ( \*((list\_id\_t \*) data) == e->id );

break;

case LIST\_CAGE\_NUMBER:

exit\_flag = ( \*((list\_id\_t \*) data) == e->data.cage\_num );

break;

case LIST\_ANIMAL\_AREAL:

exit\_flag = !wcscmp((wchar\_t \*) data, e->data.animal.areal);

break;

case LIST\_ANIMAL\_BREED:

exit\_flag = !wcscmp((wchar\_t \*) data, e->data.animal.breed);

break;

case LIST\_ANIMAL\_NAME:

exit\_flag = !wcscmp((wchar\_t \*) data, e->data.animal.name);

break;

case LIST\_PRODUCT\_TYPE:

exit\_flag = !wcscmp((wchar\_t \*) data, e->data.products.type);

break;

case LIST\_PRODUCT\_WEIGHT:

exit\_flag = ( \*((weight\_t \*) data) == e->data.products.weight );

break;

case LIST\_PRODUCT\_COST:

exit\_flag = ( \*((cost\_t \*) data) == e->data.products.cost );

break;

case LIST\_DATE:

exit\_flag = !datecmp((udate\_t \*) data, &e->data.date);

break;

}

if (exit\_flag) break;

e = e->next; position\_counter++;

}

return e;

}

**Исходный код файла «checker.c»**

#include "checker.h"

int check\_wchar (const wint\_t ch, const checker\_mode\_t mode)

{

if ((mode & VMASK\_LOWER\_CHARS) && iswlower(ch)) return VMASK\_LOWER\_CHARS;

if ((mode & VMASK\_UPPER\_CHARS) && iswupper(ch)) return VMASK\_UPPER\_CHARS;

if ((mode & VMASK\_DIGITS) && iswdigit(ch)) return VMASK\_DIGITS;

if ((mode & VMASK\_SPACES) && (ch == L' ')) return VMASK\_SPACES;

if ((mode & VMASK\_PUNCTS) && (ch == L'.' || ch == L',')) return VMASK\_PUNCTS;

if ((mode & VMASK\_ADD\_FS\_CHARS)&& (ch == L':' || ch == L'/' || ch == L'\_' || ch == L'-')) return VMASK\_ADD\_FS\_CHARS;

return 0;

}

**Исходный код файла «wcs\_converters.c»**

#include "wcs\_conv.h"

long long wcs2int (const wchar\_t \* string)

{

/\* Немедленно выйти из подпрограммы, если строка не была передана \*/

if (string == NULL) return 0;

size\_t offset = 0; // Отступ от начала строки

int sign = 1; // Окончательный знак числа

long long n = 0; // Полученное число

/\* Пропустить пробелы перед числом \*/

while (iswspace(string[offset])) offset++;

/\* Определиться со знаком числа \*/

switch (string[offset])

{

case '-': sign = -1;

case '+': offset++; break;

}

/\* Преобразовать оставшёюся часть строки в число \*/

while (iswdigit(string[offset]))

n = n \* 10 + (string[offset++] - L'0');

return (n \* sign);

}

long double wcs2float (const wchar\_t \* string)

{

/\* Немедленно выйти из подпрограммы, если не была передана строка \*/

if (string == NULL) return 0;

size\_t offset = 0; // Отступ от начала строки

int sign = 1; // Итоговый знак числа

long double n = 0; // Полученное число

/\* Пропуск пробелов перед числом \*/

while (iswspace(string[offset])) offset++;

/\* Определиться со знаком числа \*/

switch (string[offset])

{

case '-': sign = -1;

case '+': offset++; break;

}

/\* Преобразовать целую часть числа \*/

for (; iswdigit(string[offset]); offset++)

n = n \* 10 + ((long double) (string[offset] - L'0'));

/\* Встречено не-число \*/

switch (string[offset])

{

// Продолжить выполнение, если встреченное не-число является

// разделителем

case '.': case ',':

offset++;

break;

// Иначе вернуть полученную целую часть числа

default:

return (n \* sign);

break;

}

int d = 1;

/\* Преобразовать дробную часть числа \*/

while (iswdigit(string[offset]))

n += ((long double) (string[offset++] - '0')) / (d \*= 10);

return (n \* sign);

}

void int2wcs (const long long number, wchar\_t \* string)

{

swprintf(string, 20, L"%lld", number);

}

void float2wcs (const long double number, wchar\_t \* string)

{

swprintf(string, 8, L"%Lf", number);

}

void fitwcs (wchar\_t \* dest, const wchar\_t \* src, const int max\_length)

{

if (!dest || !src || max\_length < 4) return;

int

curr\_length = 0,

src\_length = wcslen (src),

insert\_colon = (src\_length - max\_length > 0) ? 1 : 0;

wchar\_t

\* src\_var = src;

while (\*src\_var != 0 && max\_length > curr\_length)

{

\*dest = (max\_length - 4 < curr\_length && insert\_colon) ? (L'.') : (\*src\_var);

src\_var++;

dest++;

curr\_length ++;

}

while (curr\_length < max\_length)

(\*(dest++) = L' ') & (curr\_length ++);

return;

}

**Исходный код файла «additional\_task.c»**

#include "additional\_task.h"

cost\_t lists\_GetCostForPeriod (const list\_obj\_t \* list, const udate\_t \* date\_min, const udate\_t \* date\_max)

{

/\* Выйти, если не был передан указатель на объект списка \*/

if (!list) return -1.0;

/\* Инициализация переменных \*/

int date\_min\_flag = (date\_min) ? (1) : (0),

date\_max\_flag = (date\_max) ? (1) : (0);

cost\_t sum = 0.0;

/\* Поиск подходящих элементов и суммирование \*/

for (list\_elem\_t \* e = list->head; e; e = e->next)

if ( (datecmp(&e->data.date, date\_min) \* date\_min\_flag >= 0) && (datecmp(&e->data.date, date\_max) \* date\_max\_flag <= 0) )

sum += e->data.products.cost;

return sum;

}

weight\_t lists\_GetWeightForPeriod (const list\_obj\_t \* list, const udate\_t \* date\_min, const udate\_t \* date\_max)

{

/\* Выйти, если не был передан указатель на объект списка \*/

if (!list) return -1.0;

/\* Инициализация переменных \*/

int date\_min\_flag = (date\_min) ? (1) : (0),

date\_max\_flag = (date\_max) ? (1) : (0);

weight\_t sum = 0.0;

/\* Поиск подходящих элементов и суммирование \*/

for (list\_elem\_t \* e = list->head; e; e = e->next)

if ( (!date\_min\_flag || datecmp(&e->data.date, date\_min) >= 0) && (!date\_max\_flag || datecmp(&e->data.date, date\_max) <= 0) )

sum += e->data.products.weight;

return sum;

}

**Исходный код файла «forms.c»**

#include "\_\_forms\_\_.h"

void tui\_draw\_form\_field (WINDOW \* win, int y, int x, const wchar\_t \* placeholder, const wchar\_t \* value, int width)

{

const int value\_len = (int) wcslen (value),

placeholder\_len = (int) wcslen (placeholder);

size\_t curr\_width = (value\_len) ? (value\_len) : (placeholder\_len);

wmove (win, y, x);

wattron (win, A\_UNDERLINE);

// Если value не содержит текста, ...

if (value\_len < 1)

{

// ... то записать placeholder в форму, ...

wattron(win, A\_ITALIC);

waddnwstr(win, placeholder, width);

wattroff(win, A\_ITALIC);

}

else

{

// ... иначе вставить в поле value

wattron(win, A\_BOLD);

waddwstr(win, value);

wattroff(win, A\_BOLD);

}

wattroff(win, A\_UNDERLINE);

for (; curr\_width < width; curr\_width++) waddwstr (win, L"\_");

}

**Исходный код файла «popup\_date.c»**

#include "\_\_popups\_\_.h"

#include "\_\_popups\_\_.h"

#include "../forms/\_\_forms\_\_.h"

void tui\_draw\_popup\_date (const wchar\_t \* popup\_title, const wchar\_t \* message, udate\_t \* date)

{

mkey\_t key = MKEY\_NULL;

int last\_curs\_state = curs\_set(0),

message\_lines\_cnt = -1, // Количество строк, занимаемых сообщением

helper\_lines\_cnt = 4, // Количество строк, занимаемых помощником

form\_field\_lines\_cnt = -1, // Количество строк, занимаемых формой

selected\_field = 1; // Номер выбранного поля

int tmp\_day = 1,

tmp\_month = 1,

tmp\_year = 2021;

int win\_width = 0.4 \* getmaxx(stdscr),

win\_height = 3 // Высота заголовка с границей

+ (message\_lines\_cnt = get\_message\_lines\_count (message, win\_width)) // ...

+ 2 //

+ helper\_lines\_cnt // ...

+ 1 // Отступ от помощника

+ 1 // Высота формы для даты

+ 2, // Дополнительное пространство после поля формы

win\_start\_x = (getmaxx(stdscr) - win\_width) / 2,

win\_start\_y = (getmaxy(stdscr) - win\_height) / 2;

/\* Основное окно popup'a \*/

WINDOW \* popup\_win = newwin(win\_height, win\_width, win\_start\_y, win\_start\_x);

PANEL \* popup\_panel = new\_panel(popup\_win);

tui\_draw\_popup\_header(popup\_win, popup\_title);

/\* Создать окно для текста и поместить в окно текст \*/

WINDOW \* inner\_win\_text = derwin (popup\_win, message\_lines\_cnt, win\_width - 2, 3, 1);

PANEL \* inner\_panel\_text = new\_panel (inner\_win\_text);

mvwprintw(inner\_win\_text, 0, 0, "%ls", message);

/\* Отобразить помощника ввода \*/

mvwprintw (popup\_win, 3 + message\_lines\_cnt + 1, 1, "Осуществление ввода:");

mvwprintw (popup\_win, 3 + message\_lines\_cnt + 2, 1, "\* ARROW UP/DOWN - Увеличить/Уменьшить значение.");

mvwprintw (popup\_win, 3 + message\_lines\_cnt + 3, 1, "\* ARROW LEFT/RIGHT, TAB - Выбор поля.");

mvwprintw (popup\_win, 3 + message\_lines\_cnt + 4, 1, "\* ENTER - Подтвердить выбор.");

/\* Отрисовать кнопку выхода из попапа \*/

wattron(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, win\_height - 2, (win\_width - 18) / 2, " ENTER для выхода ");

wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

do

{

/\* Проверить нажатую клавишу \*/

switch (key)

{

// Нажат Enter или Первый проход цикла

case MKEY\_ENTER: case MKEY\_NULL:

break;

// Нажата кнопка увеличения значения

case MKEY\_ARROW\_UP:

switch (selected\_field)

{

case 1:

if (tmp\_day < 31)

tmp\_day++;

break;

case 2:

if (tmp\_month < 12)

tmp\_month++;

break;

case 3:

if (tmp\_year < 2100)

tmp\_year++;

break;

}

break;

// Нажата кнопка уменьшения значения

case MKEY\_ARROW\_DOWN:

switch (selected\_field)

{

case 1:

if (tmp\_day > 1)

tmp\_day--;

break;

case 2:

if (tmp\_month > 1)

tmp\_month--;

break;

case 3:

if (tmp\_year > 1970)

tmp\_year--;

break;

}

break;

// Нажата кнопка перехода к следующему полю

case MKEY\_TAB: case MKEY\_ARROW\_RIGHT:

if (selected\_field == 4)

selected\_field = 1;

else

selected\_field++;

break;

// Нажата кнопка перехода к предыдущему полю

case MKEY\_ARROW\_LEFT:

if (selected\_field == 1)

selected\_field = 4;

else

selected\_field--;

break;

}

/\* Обработать день, если выбран февраль в високосный год \*/

if (2 == tmp\_month)

{

if (tmp\_year % 4)

// Невисокосный год

{

if (tmp\_day > 28)

tmp\_day = 28;

}

else

// Високосный од

{

if (tmp\_day > 29)

tmp\_day = 29;

}

}

/\* Вывести форму на экран \*/

mvwprintw (popup\_win, win\_height - 3, 1, " / / ");

if (selected\_field == 1) wattron(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, win\_height - 3, 1, "%02d", tmp\_day);

if (selected\_field == 1) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

if (selected\_field == 2) wattron(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, win\_height - 3, 4, "%02d", tmp\_month);

if (selected\_field == 2) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

if (selected\_field == 3) wattron(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, win\_height - 3, 7, "%04d", tmp\_year);

if (selected\_field == 3) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

/\* Отрисовать кнопку выхода из попапа \*/

if (selected\_field == 4) wattron(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, win\_height - 2, (win\_width - 18) / 2, " ENTER для выхода ");

if (selected\_field == 4) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

/\* Обновить окна/панели \*/

update\_panels ();

doupdate ();

}

while ((key = getwchar()) != MKEY\_ENTER || selected\_field != 4);

/\* Записать данные в структуру \*/

date->D = tmp\_day;

date->M = tmp\_month;

date->Y = tmp\_year;

/\* Удаление всех созданных панелей и окон \*/

del\_panel (inner\_panel\_text);

del\_panel (popup\_panel);

delwin (inner\_win\_text);

delwin (popup\_win);

/\* Установить прошлое значение видимости курсора \*/

curs\_set(last\_curs\_state);

return;

}**Исходный код файла «popup\_element\_info.c»**

#include "\_\_popups\_\_.h"

#include "../forms/\_\_forms\_\_.h"

const menu\_t PLACE\_ELEMENT\_MENU = {

L"В начало",

L"В конец",

NULL

};

int tui\_popup\_edit\_element\_data (const wchar\_t \* title, list\_data\_t \* edata, int create\_new\_element\_mode)

{

int

exit\_flag = 0,

flags = TUI\_AT\_END\_FLAG,

selected\_field = 1;

wchar\_t tmp\_wcs[100];

mkey\_t key = MKEY\_NULL;

// Подготовка начальных данных

if (create\_new\_element\_mode)

{

memset (edata, 0, sizeof(list\_data\_t));

edata->date.Y = 2021;

edata->date.M = 11;

edata->date.D = 11;

}

WINDOW \* popup\_win = newwin (22, 22 + ANIMAL\_AREAL\_MAX\_LEN, 1, 1);

PANEL \* popup\_panel = new\_panel (popup\_win);

update\_panels ();

tui\_draw\_popup\_header (popup\_win, title);

do

{

/\* Обработать нажатую клавишу \*/

switch (key)

{

case MKEY\_TAB:

if (selected\_field < 9 - (!create\_new\_element\_mode))

selected\_field++;

else

selected\_field = 1;

break;

case MKEY\_ENTER:

switch (selected\_field)

{

case 1:

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

if (edata->cage\_num)

int2wcs (edata->cage\_num, tmp\_wcs);

tui\_draw\_popup\_form (L"Ввод # Вольера", L"Введите номер вольера.", L"№", tmp\_wcs, CAGE\_NUM\_DGT\_CNT, VMASK\_DIGITS);

edata->cage\_num = wcs2int(tmp\_wcs);

break;

case 2:

tui\_draw\_popup\_date (L"Ввод даты поступления", L"Введите дату поступления животкого в центр.", &edata->date);

break;

case 3:

tui\_draw\_popup\_form (L"Ввод ареала", L"Введите ареал обитания животного.", L"ареал", edata->animal.areal, ANIMAL\_AREAL\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

break;

case 4:

tui\_draw\_popup\_form (L"Ввод породы", L"Введите породу животного.", L"порода", edata->animal.breed, ANIMAL\_BREED\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

break;

case 5:

tui\_draw\_popup\_form (L"Ввод имени", L"Введите имя животного.", L"имя", edata->animal.name, ANIMAL\_NAME\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

break;

case 6:

tui\_draw\_popup\_form (L"Ввод типа продукта", L"Введите тип продукта.", L"тип", edata->products.type, PRODUCT\_TYPE\_MAX\_LEN, VMASK\_ANY\_CHAR | VMASK\_SPACES);

break;

case 7:

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

if (edata->products.weight)

float2wcs (edata->products.weight, tmp\_wcs);

tui\_draw\_popup\_form (L"Ввод веса продукта", L"Введите вес продукта, употребляемого животным за день.", L"вес", tmp\_wcs, WEIGHT\_DGT\_CNT, VMASK\_DIGITS | VMASK\_PUNCTS);

edata->products.weight = wcs2float(tmp\_wcs);

break;

case 8:

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

if (edata->products.cost)

float2wcs (edata->products.cost, tmp\_wcs);

tui\_draw\_popup\_form (L"Ввод стоимости продукта", L"Введите стоимость всех продуктов, потребляемых животным за день", L"стоим", tmp\_wcs, COST\_DGT\_CNT, VMASK\_DIGITS | VMASK\_PUNCTS);

edata->products.cost = wcs2float(tmp\_wcs);

break;

case 9:

switch (tui\_draw\_popup\_select (L"Выбрать размещение данных", L"Где разместить создаваемый элемент по отнашению к существующим данным?", PLACE\_ELEMENT\_MENU))

{

// Добавить в начало

case 1:

flags &= ~TUI\_AT\_END\_FLAG;

flags |= TUI\_AT\_BEGIN\_FLAG;

break;

// Добавить в конец списка

case 2:

flags &= ~TUI\_AT\_BEGIN\_FLAG;

flags |= TUI\_AT\_END\_FLAG;

break;

}

break;

}

break;

case L'A': case L'a':

flags |= TUI\_ADD\_ELEMENT\_FLAG;

exit\_flag = 1;

break;

case L'E': case L'e':

flags = TUI\_ADD\_ELEMENT\_CACELLED;

exit\_flag = 1;

break;

default:

break;

}

/\* Отрисовать список \*/

// Номер вольера

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); if (edata->cage\_num) int2wcs (edata->cage\_num, tmp\_wcs);

if (selected\_field == 1) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 5, 1, " Номер вольера "); tui\_draw\_form\_field (popup\_win, 5, 21, L"НЕТ НОМ", tmp\_wcs, 7);

if (selected\_field == 1) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

// Дата поступления

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); date2wcs (&edata->date, tmp\_wcs);

if (selected\_field == 2) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 6, 1, " Дата поступления "); tui\_draw\_form\_field (popup\_win, 6, 21, L"НЕТ ДАТЫ", tmp\_wcs, 10);

if (selected\_field == 2) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 7, 1, " О животном: ");

// Ареал обитания

if (selected\_field == 3) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 8, 1, " \* Ареал обитания "); tui\_draw\_form\_field (popup\_win, 8, 21, L"НЕТ АРЕАЛА", edata->animal.areal, ANIMAL\_AREAL\_MAX\_LEN - 1);

if (selected\_field == 3) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

// Порода

if (selected\_field == 4) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 9, 1, " \* Порода "); tui\_draw\_form\_field (popup\_win, 9, 21, L"НЕТ ПОРОДЫ", edata->animal.breed, ANIMAL\_BREED\_MAX\_LEN - 1);

if (selected\_field == 4) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

// Имя

if (selected\_field == 5) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 10, 1, " \* Имя "); tui\_draw\_form\_field (popup\_win, 10, 21, L"НЕТ ИМЕНИ", edata->animal.name, ANIMAL\_NAME\_MAX\_LEN - 1);

if (selected\_field == 5) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 11, 1, " О продуктах: ");

// Тип

if (selected\_field == 6) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 12, 1, " \* Тип "); tui\_draw\_form\_field (popup\_win, 12, 21, L"НЕТ ТИПА", edata->products.type, PRODUCT\_TYPE\_MAX\_LEN - 1);

if (selected\_field == 6) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

// Вес

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); if (edata->products.weight) float2wcs (edata->products.weight, tmp\_wcs);

if (selected\_field == 7) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 13, 1, " \* Вес "); tui\_draw\_form\_field (popup\_win, 13, 21, L"НЕТ ВЕС", tmp\_wcs, 7);

if (selected\_field == 7) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

// Стоимость

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); if (edata->products.cost) float2wcs (edata->products.cost, tmp\_wcs);

if (selected\_field == 8) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 14, 1, " \* Стоимость "); tui\_draw\_form\_field (popup\_win, 14, 21, L"НЕТ ЦЕН", tmp\_wcs, 7);

if (selected\_field == 8) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

// Куда добавлять элемент - в начало или в конец

if (create\_new\_element\_mode)

{

if (selected\_field == 9) wattron (popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, 16, 1, " Добавить в "); mvwprintw (popup\_win, 16, 21, (flags & TUI\_AT\_BEGIN\_FLAG) ? "В начало" : "В конец");

if (selected\_field == 9) wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

}

mvwprintw (popup\_win, 18, 1, "[TAB] Навигация. [A] Сохранить. [E] Отменить.");

update\_panels ();

doupdate ();

key = getwchar ();

}

while (!exit\_flag);

del\_panel (popup\_panel);

delwin (popup\_win);

return flags;

}

void tui\_popup\_show\_only\_element (const wchar\_t \* title, const list\_elem\_t \* elem)

{

wchar\_t tmp\_wcs[100];

mkey\_t key = MKEY\_NULL;

WINDOW \* popup\_win = newwin (23, 22 + ANIMAL\_AREAL\_MAX\_LEN, 1, 1);

PANEL \* popup\_panel = new\_panel (popup\_win);

update\_panels ();

tui\_draw\_popup\_header (popup\_win, title);

/\* Отрисовать список \*/

// ID элемента

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); if (elem->id) int2wcs (elem->id, tmp\_wcs);

mvwprintw (popup\_win, 5, 1, " ID записи "); tui\_draw\_form\_field (popup\_win, 5, 21, L"НЕТ ID", tmp\_wcs, 7);

// Номер вольера

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); if (elem->data.cage\_num) int2wcs (elem->data.cage\_num, tmp\_wcs);

mvwprintw (popup\_win, 6, 1, " Номер вольера "); tui\_draw\_form\_field (popup\_win, 5, 21, L"НЕТ НОМ", tmp\_wcs, 7);

// Дата поступления

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); date2wcs (&elem->data.date, tmp\_wcs);

mvwprintw (popup\_win, 7, 1, " Дата поступления "); tui\_draw\_form\_field (popup\_win, 6, 21, L"НЕТ ДАТЫ", tmp\_wcs, 10);

mvwprintw (popup\_win, 8, 1, " О животном: ");

// Ареал обитания

mvwprintw (popup\_win, 9, 1, " \* Ареал обитания "); tui\_draw\_form\_field (popup\_win, 8, 21, L"НЕТ АРЕАЛА", elem->data.animal.areal, ANIMAL\_AREAL\_MAX\_LEN - 1);

// Порода

mvwprintw (popup\_win, 10, 1, " \* Порода "); tui\_draw\_form\_field (popup\_win, 9, 21, L"НЕТ ПОРОДЫ", elem->data.animal.breed, ANIMAL\_BREED\_MAX\_LEN - 1);

// Имя

mvwprintw (popup\_win, 11, 1, " \* Имя "); tui\_draw\_form\_field (popup\_win, 10, 21, L"НЕТ ИМЕНИ", elem->data.animal.name, ANIMAL\_NAME\_MAX\_LEN - 1);

mvwprintw (popup\_win, 12, 1, " О продуктах: ");

// Тип

mvwprintw (popup\_win, 13, 1, " \* Тип "); tui\_draw\_form\_field (popup\_win, 12, 21, L"НЕТ ТИПА", elem->data.products.type, PRODUCT\_TYPE\_MAX\_LEN - 1);

// Вес

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); if (elem->data.products.weight) float2wcs (elem->data.products.weight, tmp\_wcs);

mvwprintw (popup\_win, 14, 1, " \* Вес "); tui\_draw\_form\_field (popup\_win, 13, 21, L"НЕТ ВЕС", tmp\_wcs, 7);

// Стоимость

memset (tmp\_wcs, 0, sizeof(tmp\_wcs)); if (elem->data.products.cost) float2wcs (elem->data.products.cost, tmp\_wcs);

mvwprintw (popup\_win, 15, 1, " \* Стоимость "); tui\_draw\_form\_field (popup\_win, 14, 21, L"НЕТ ЦЕН", tmp\_wcs, 7);

mvwprintw (popup\_win, 19, 1, "[E] Выйти.");

update\_panels ();

doupdate ();

while ((key = getwchar ()) != L'E' && key != L'e');

del\_panel (popup\_panel);

delwin (popup\_win);

return;

}

**Исходный код файла «popup\_form.c»**

#include "\_\_popups\_\_.h"

#include "../forms/\_\_forms\_\_.h"

int tui\_draw\_popup\_form (const wchar\_t \* popup\_title, const wchar\_t \* message, const wchar\_t \* placeholder, wchar\_t \* value, int max\_width, checker\_mode\_t mode)

{

if (!mode) mode = VMASK\_ANY\_CHAR;

int helper\_items\_count = 0;

for (checker\_mode\_t tmp = mode; tmp; tmp >>= 1) helper\_items\_count += ((tmp & 1) ? 1 : 0);

mkey\_t key = MKEY\_NULL;

int last\_curs\_state = curs\_set(2),

message\_len = (int) wcslen(message), // Длина сообщения

last\_char\_pos = (int) wcslen(value) - 1, // Позичия последнего символа в value

message\_lines\_cnt = -1, // Количество строк, занимаемых сообщением

helper\_lines\_cnt = 1 + helper\_items\_count, // Количество строк, занимаемых помощником

form\_field\_lines\_cnt = -1; // Количество строк, занимаемых формой

int win\_width = 0.4 \* getmaxx(stdscr),

win\_height = 3 // Высота заголовка с границей

+ (message\_lines\_cnt = get\_message\_lines\_count (message, win\_width)) // ...

+ 2 //

+ helper\_lines\_cnt // ...

+ 1 // Отступ от помощника

+ (form\_field\_lines\_cnt = max\_width / (win\_width - 2) + ( (max\_width % (win\_width - 2)) ? 1 : 0 )) // ...

+ 2, // Дополнительное пространство после поля формы

win\_start\_x = (getmaxx(stdscr) - win\_width) / 2,

win\_start\_y = (getmaxy(stdscr) - win\_height) / 2;

/\* Основное окно popup'a \*/

WINDOW \* popup\_win = newwin(win\_height, win\_width, win\_start\_y, win\_start\_x);

PANEL \* popup\_panel = new\_panel(popup\_win);

tui\_draw\_popup\_header(popup\_win, popup\_title);

/\* Создать окно для текста и поместить в окно текст \*/

WINDOW \* inner\_win\_text = derwin (popup\_win, message\_lines\_cnt, win\_width - 2, 3, 1);

PANEL \* inner\_panel\_text = new\_panel (inner\_win\_text);

mvwprintw(inner\_win\_text, 0, 0, "%ls", message);

{

int tmp\_helper\_counter = 1;

mvwprintw (popup\_win, 3 + message\_lines\_cnt + tmp\_helper\_counter++, 1, "Ограничения на ввод:");

if (mode & VMASK\_LOWER\_CHARS)

mvwprintw (popup\_win, 3 + message\_lines\_cnt + tmp\_helper\_counter++, 1, "\* Буквы нижнего регистра.");

if (mode & VMASK\_UPPER\_CHARS)

mvwprintw (popup\_win, 3 + message\_lines\_cnt + tmp\_helper\_counter++, 1, "\* Буквы верхнего регистра.");

if (mode & VMASK\_DIGITS)

mvwprintw (popup\_win, 3 + message\_lines\_cnt + tmp\_helper\_counter++, 1, "\* Цифры.");

if (mode & VMASK\_PUNCTS)

mvwprintw (popup\_win, 3 + message\_lines\_cnt + tmp\_helper\_counter++, 1, "\* Точки и запятые.");

if (mode & VMASK\_ADD\_FS\_CHARS)

mvwprintw (popup\_win, 3 + message\_lines\_cnt + tmp\_helper\_counter++, 1, "\* \"\_\", \"/\", \":\" и \"-\".");

}

/\* Создать окно для формы \*/

WINDOW \* inner\_win\_form = derwin (popup\_win, form\_field\_lines\_cnt, win\_width - 2, win\_height - 3 - form\_field\_lines\_cnt, 1);

PANEL \* inner\_panel\_form = new\_panel (inner\_win\_form);

/\* Отрисовать кнопку выхода из попапа \*/

wattron(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (popup\_win, win\_height - 2, (win\_width - 18) / 2, " ENTER для выхода ");

wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

do

{

/\* Проверить нажатую клавишу \*/

switch (key)

{

// Нажат Enter или первый проход цикла

case MKEY\_ENTER: case MKEY\_NULL:

break;

// Нажато Backspace

case MKEY\_BACKSPACE:

if (last\_char\_pos >= 0)

value[last\_char\_pos--] = 0;

break;

// Нажатадругая кнопка/комбинация кнопок

default:

if (check\_wchar(key, mode) && last\_char\_pos < max\_width - 1)

value[++last\_char\_pos] = key;

break;

}

/\* Вывести форму на экран и обновить окна/панели \*/

mvwprintw (popup\_win, win\_height - 4 - form\_field\_lines\_cnt, 1, "Ввод (%3d символов из %3d):", last\_char\_pos + 1, max\_width);

tui\_draw\_form\_field (inner\_win\_form, 0, 0, placeholder, value, max\_width);

update\_panels ();

doupdate ();

}

while ((key = getwchar()) != MKEY\_ENTER);

/\* Удаление всех созданных панелей и окон \*/

del\_panel (inner\_panel\_form);

del\_panel (inner\_panel\_text);

del\_panel (popup\_panel);

delwin (inner\_win\_form);

delwin (inner\_win\_text);

delwin (popup\_win);

/\* Установить прошлое значение видимости курсора \*/

curs\_set(last\_curs\_state);

return last\_char\_pos + 1;

}

**Исходный код файла «popup\_select.c»**

#include "\_\_popups\_\_.h"

int tui\_draw\_popup\_select (const wchar\_t \* popup\_title, const wchar\_t \* message, const menu\_t menu)

{

int menu\_item\_count = tui\_get\_menu\_item\_count(menu);

int win\_width = getmax(0.4 \* getmaxx(stdscr), wcslen(menu[0])),

message\_lines\_count = get\_message\_lines\_count (message, win\_width),

win\_height = 8 + message\_lines\_count + menu\_item\_count,

win\_start\_x = (getmaxx(stdscr) - win\_width) / 2,

win\_start\_y = (getmaxy(stdscr) - win\_height) / 2;

int selected\_menu\_item = 1;

unsigned int key = MKEY\_NULL;

WINDOW \* popup\_win = newwin(win\_height, win\_width, win\_start\_y, win\_start\_x);

PANEL \* popup\_panel = new\_panel(popup\_win);

tui\_draw\_popup\_header(popup\_win, popup\_title);

/\* Создать окно для текста и поместить в окно текст \*/

WINDOW \* inner\_win\_text = derwin (popup\_win, message\_lines\_count, win\_width - 2, 4, 1);

PANEL \* inner\_panel\_text = new\_panel (inner\_win\_text);

mvwprintw(inner\_win\_text, 0, 0, "%ls", message);

/\* Создание окна для набора кнопок \*/

WINDOW \* inner\_win\_buttons = derwin (popup\_win, menu\_item\_count, win\_width - 2, win\_height - 3 - menu\_item\_count, 1);

PANEL \* inner\_panel\_buttons = new\_panel(inner\_win\_buttons);

do {

switch (key)

{

case MKEY\_ARROW\_UP:

if (selected\_menu\_item > 1)

selected\_menu\_item--;

break;

case MKEY\_ARROW\_DOWN:

if (selected\_menu\_item < menu\_item\_count)

selected\_menu\_item++;

break;

}

tui\_draw\_vmenu(inner\_win\_buttons, 0, 0, 0, menu, selected\_menu\_item);

update\_panels();

doupdate();

key = getwchar();

} while (key != MKEY\_ENTER);

/\* Удаление окон и панелей \*/

del\_panel(inner\_panel\_buttons);

del\_panel(inner\_panel\_text);

del\_panel(popup\_panel);

delwin(inner\_win\_buttons);

delwin(inner\_win\_text);

delwin(popup\_win);

return selected\_menu\_item;

}

**Исходный код файла «popup\_text\_message.c»**

#include "\_\_popups\_\_.h"

void tui\_draw\_popup\_text\_message (const wchar\_t \* popup\_title, const wchar\_t \* message)

{

int win\_width = 0.4 \* getmaxx(stdscr),

message\_lines\_count = get\_message\_lines\_count (message, win\_width),

win\_height = 8 + message\_lines\_count,

win\_start\_x = (getmaxx(stdscr) - win\_width) / 2,

win\_start\_y = (getmaxy(stdscr) - win\_height) / 2;

WINDOW \* popup\_win = newwin(win\_height, win\_width, win\_start\_y, win\_start\_x);

PANEL \* popup\_panel = new\_panel(popup\_win);

tui\_draw\_popup\_header(popup\_win, popup\_title);

/\* Создание окна для текста \*/

WINDOW \* inner\_win = derwin (popup\_win, win\_height - 8, win\_width - 2, 4, 1);

PANEL \* inner\_panel = new\_panel (inner\_win);

mvwprintw(inner\_win, 0, 0, "%ls", message);

/\* Печать кнопки \*/

wattron(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw(popup\_win, win\_height - 3, (win\_width - 4) / 2, " OK ");

wattroff(popup\_win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

update\_panels();

doupdate();

while (getwchar() != MKEY\_ENTER);

/\* Удаление окон и панелей \*/

del\_panel(inner\_panel);

delwin(inner\_win);

del\_panel(popup\_panel);

delwin(popup\_win);

return;

}

**Исходный код файла «table.c»**

#include "tui.h"

int tui\_draw\_table\_in\_window (WINDOW \* win, const list\_obj\_t list\_obj, const length\_t page\_num, const int selected\_row)

{

const int TABLE\_TOTAL\_WIDTH = getmaxx(win), // Итоговая ширина таблицы

POS\_COL\_WIDTH = 4, // Ширина колонки с порядковым номером

ID\_COL\_WIDTH = 4, // Ширина колонки с ID

CAGE\_NUM\_COL\_WIDTH = 7, // Ширина колонки с Номером вольера

PRODUCT\_WEIGHT\_COL\_WIDTH = 8, // Ширина колонки с Весом продукта

PRODUCT\_COST\_COL\_WIDTH = 12, // Ширина колонки со Стоимостью продукта

DATE\_COL\_WIDTH = 11, // Ширина колонки с Датой

// Ширина колонки с Информацией о животном (с учётом внутренних границ)

ANIMAL\_TOTAL\_COL\_WIDTH = getmax(25, (int) (TABLE\_TOTAL\_WIDTH - POS\_COL\_WIDTH - ID\_COL\_WIDTH - CAGE\_NUM\_COL\_WIDTH - DATE\_COL\_WIDTH - PRODUCT\_WEIGHT\_COL\_WIDTH - PRODUCT\_COST\_COL\_WIDTH - 9) / 4 \* 3),

// Ширина колонки с Типом продукта

PRODUCT\_TYPE\_COL\_WIDTH = getmax(3, (int) (TABLE\_TOTAL\_WIDTH - POS\_COL\_WIDTH - ID\_COL\_WIDTH - CAGE\_NUM\_COL\_WIDTH - DATE\_COL\_WIDTH - PRODUCT\_WEIGHT\_COL\_WIDTH - PRODUCT\_COST\_COL\_WIDTH - 9 - ANIMAL\_TOTAL\_COL\_WIDTH)),

// Ширина колонки с Информацией о продукте (с учётом внутренних границ)

PRODUCT\_TOTAL\_COL\_WIDTH = PRODUCT\_COST\_COL\_WIDTH + 1 + PRODUCT\_WEIGHT\_COL\_WIDTH + 1 + PRODUCT\_TYPE\_COL\_WIDTH,

ANIMAL\_NAME\_COL\_WIDTH = getmax(5, (int) ((ANIMAL\_TOTAL\_COL\_WIDTH - 2) / 3)), // Ширина колонки с Именем животного

ANIMAL\_ARIAL\_COL\_WIDTH = getmax(14, (int) ((ANIMAL\_TOTAL\_COL\_WIDTH - 2) / 3)), // Ширина колонки с Ареалом обитания

ANIMAL\_BREED\_COL\_WIDTH = getmax(6, ANIMAL\_TOTAL\_COL\_WIDTH - 2 - ANIMAL\_NAME\_COL\_WIDTH - ANIMAL\_ARIAL\_COL\_WIDTH); // Ширина колонки с Породой животного

const int DATA\_ROWS\_CNT = getmaxy(win) - 4 - 1; // Количество строк с данными в таблице

int offset\_from\_header = 0,

some\_counter = 0;

wchar\_t \* tmp\_wcs = calloc (sizeof(wchar\_t), 256);

wmove(win, 0, 0);

/\* отрисовать верхнюю границу шапки таблицы \*/

waddstr(win, "┌");

for (int i = 0; i < POS\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┬");

for (int i = 0; i < ID\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┬");

for (int i = 0; i < CAGE\_NUM\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┬");

for (int i = 0; i < ANIMAL\_TOTAL\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┬");

for (int i = 0; i < PRODUCT\_TOTAL\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┬");

for (int i = 0; i < DATE\_COL\_WIDTH; i++) waddstr(win, "─");

waddstr(win, "┐");

/\* Отрисовка шапки, 1 ряд \*/

mvwprintw(

win, 1, 0,

"│%\*ls│%\*ls│%\*ls│%\*ls│%\*ls│%\*ls│",

POS\_COL\_WIDTH, L" № ",

ID\_COL\_WIDTH, L" ID ",

CAGE\_NUM\_COL\_WIDTH, L" Номер ",

ANIMAL\_TOTAL\_COL\_WIDTH, L"Информация о животном",

PRODUCT\_TOTAL\_COL\_WIDTH, L"Информация о продукте",

DATE\_COL\_WIDTH, L"Дата"

);

mvwprintw (win, 1, 0, "│ № | ID | Номер ");

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, L"Информация о животном", ANIMAL\_TOTAL\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, L"Информация о продукте", PRODUCT\_TOTAL\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

wprintw (win, "│ Дата │");

mvwprintw (win, 2, 0, "│ | |вольера");

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, L"Ареал обитания", ANIMAL\_ARIAL\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, L"Порода", ANIMAL\_BREED\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, L"Имя", ANIMAL\_NAME\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, L"Тип", PRODUCT\_TYPE\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

wprintw (win, "│ Вес, г │Стоимость, P│ кормления │");

/\* Отрисовать нижнюю границу шапки таблицы \*/

wmove(win, 3, 0);

waddstr(win, "├");

for (int i = 0; i < POS\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < ID\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < CAGE\_NUM\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < ANIMAL\_ARIAL\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < ANIMAL\_BREED\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < ANIMAL\_NAME\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < PRODUCT\_TYPE\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < PRODUCT\_WEIGHT\_COL\_WIDTH; i++)waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < PRODUCT\_COST\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┼");

for (int i = 0; i < DATE\_COL\_WIDTH; i++) waddstr(win, "─");

waddstr(win, "┤");

/\* Получить первый элемент page\_num страницы \*/

length\_t first\_element\_position = (page\_num - 1) \* DATA\_ROWS\_CNT + 1;

list\_elem\_t \* elem = lists\_SearchElementByField(&list\_obj, LIST\_POSITION, (void \*) &first\_element\_position);

if (!elem)

{

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, L"Нет данных в списке.", TABLE\_TOTAL\_WIDTH - 2);

mvwprintw (win, 4 + offset\_from\_header++, 0, "│%ls│", tmp\_wcs);

}

else

{

wchar\_t date\_tmp[40]; memset (date\_tmp, 0, sizeof(date\_tmp));

int element\_position = first\_element\_position;

for (; elem && element\_position < first\_element\_position + DATA\_ROWS\_CNT; element\_position++, elem = elem->next, offset\_from\_header++)

{

date2wcs(&elem->data.date, date\_tmp);

if (offset\_from\_header + 1 == selected\_row) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

wmove(win, 4 + offset\_from\_header, 0);

wprintw(win, "│%\*d", POS\_COL\_WIDTH, element\_position);

wprintw(win, "│%\*lu", ID\_COL\_WIDTH, elem->id);

wprintw(win, "│%\*lu", CAGE\_NUM\_COL\_WIDTH, elem->data.cage\_num);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, elem->data.animal.areal, ANIMAL\_ARIAL\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, elem->data.animal.breed, ANIMAL\_BREED\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, elem->data.animal.name, ANIMAL\_NAME\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, elem->data.products.type, PRODUCT\_TYPE\_COL\_WIDTH);

wprintw (win, "│%ls", tmp\_wcs);

wprintw(win, "│%\*.2Lf", PRODUCT\_WEIGHT\_COL\_WIDTH, elem->data.products.weight);

wprintw(win, "│%\*.2Lf", PRODUCT\_COST\_COL\_WIDTH, elem->data.products.cost);

memset (tmp\_wcs, 0, sizeof(wchar\_t) \* 256);

fitwcs (tmp\_wcs, date\_tmp, DATE\_COL\_WIDTH);

wprintw (win, "│%ls│", tmp\_wcs);

if (offset\_from\_header + 1 == selected\_row) wattroff(win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

some\_counter++;

}

}

wmove(win, 4 + offset\_from\_header++, 0);

waddstr(win, "└");

for (int i = 0; i < POS\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < ID\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < CAGE\_NUM\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < ANIMAL\_ARIAL\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < ANIMAL\_BREED\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < ANIMAL\_NAME\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < PRODUCT\_TYPE\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < PRODUCT\_WEIGHT\_COL\_WIDTH; i++)waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < PRODUCT\_COST\_COL\_WIDTH; i++) waddstr(win, "─"); waddstr(win, "┴");

for (int i = 0; i < DATE\_COL\_WIDTH; i++) waddstr(win, "─");

waddstr(win, "┘");

free (tmp\_wcs);

return some\_counter;

}

**Исходный код файла «show\_element\_info\_window.c»**

#include "tui.h"

#include "../forms/\_\_forms\_\_.h"

void tui\_print\_element\_info\_window (WINDOW \* win, const int selected\_field, const list\_elem\_t \* elem)

{

if (!elem || !win) return;

const int LABEL\_MAX\_WIDTH = 26; // Максимальная ширина ярлыка

int curr\_offset = 1; // Отступ от начала окна

wchar\_t tmp\_wcs[200]; // Строка хранения временного значения

/\* Нарисовать свой ярлык к каждому полю информации и нарисовать соответствующее поле \*/

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

int2wcs (elem->id, tmp\_wcs);

if (selected\_field == 1) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, 0, 0, " Уникальный ID ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", tmp\_wcs, LABEL\_MAX\_WIDTH);

if (selected\_field == 1) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

int2wcs (elem->data.cage\_num, tmp\_wcs);

if (selected\_field == 2) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Номер вольера ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", tmp\_wcs, LABEL\_MAX\_WIDTH);

if (selected\_field == 2) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

if (selected\_field == 3) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Животное. Ареал обитания ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", elem->data.animal.areal, ANIMAL\_AREAL\_MAX\_LEN);

if (selected\_field == 3) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

if (selected\_field == 4) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Животное. Порода ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", elem->data.animal.breed, ANIMAL\_BREED\_MAX\_LEN);

if (selected\_field == 4) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

if (selected\_field == 5) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Животное. Имя ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", elem->data.animal.name, ANIMAL\_NAME\_MAX\_LEN);

if (selected\_field == 5) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

if (selected\_field == 6) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Продукты. Тип ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", elem->data.products.type, PRODUCT\_TYPE\_MAX\_LEN);

if (selected\_field == 6) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

float2wcs (elem->data.products.weight, tmp\_wcs);

if (selected\_field == 7) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Продукты. Вес ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", tmp\_wcs, LABEL\_MAX\_WIDTH);

if (selected\_field == 7) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

float2wcs (elem->data.products.cost, tmp\_wcs);

if (selected\_field == 8) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Продукты. Стоимость ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", tmp\_wcs, LABEL\_MAX\_WIDTH);

if (selected\_field == 8) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

memset (tmp\_wcs, 0, sizeof(tmp\_wcs));

date2wcs (&elem->data.date, tmp\_wcs);

if (selected\_field == 9) wattron (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

mvwprintw (win, curr\_offset++, 0, " Дата поступления ");

tui\_draw\_form\_field (win, curr\_offset++, 0, L"НЕТ ЗНАЧЕНИЯ", tmp\_wcs, LABEL\_MAX\_WIDTH);

if (selected\_field == 9) wattroff (win, TUI\_HIGHLIGHT\_ITEM\_STYLE);

return;

}

**Исходный код файла для сборщика проекта «Makefile»**

.PHONY: all clean cleanall lines

EXEC\_PATH = exec

INCLUDE\_PATH = include

SRC\_PATH = src

SRC\_CORE\_PATH = $(SRC\_PATH)/core

SRC\_TUI\_PATH = $(SRC\_PATH)/tui

CORE\_FILES = $(addprefix $(SRC\_CORE\_PATH)/,\

$(addprefix date/,\

date.c \

)\

$(addprefix file\_system/,\

files.c \

csv\_files.c \

bin\_files.c \

text\_files.c \

)\

$(addprefix lists/,\

lists.c \

lists\_cmp\_elems.c \

lists\_sort.c \

)\

$(addprefix non-standart\_funcs/,\

wcs\_converters.c \

cmp\_functions.c \

checker.c \

)\

additional\_task.c \

)

TUI\_FILES = $(addprefix $(SRC\_TUI\_PATH)/,\

$(addprefix popups/,\

pattern\_functions.c \

popup\_text\_message.c \

popup\_select.c \

popup\_form.c \

popup\_date.c \

popup\_element\_info.c \

)\

$(addprefix menu/,\

menu.c \

)\

$(addprefix forms/,\

forms.c \

)\

$(addprefix table/,\

table.c \

)\

$(addprefix windows/,\

show\_element\_info\_window.c \

)\

)

PROG\_NAME = program

GCC = gcc

CFLAGS = -I $(INCLUDE\_PATH) -g

CLIBS = -lpanelw -lncursesw

MAIN\_FILE = main.c

all: $(PROG\_NAME)

clean:

find . -name "\*.o" | while read -r file; do rm "$$file"; done

cleanall: clean

find . -name "\*.a" | while read -r file; do rm "$$file"; done

if [ -f "$(PROG\_NAME)" ]; then rm -rf "$(PROG\_NAME)"; fi

lines:

wc --lines `find . -name "\*.h"` `find . -name "\*.c"`

# $(PROG\_NAME): $(SRC\_PATH)/$(subst .c,.o,$(MAIN\_FILE)) $(SRC\_PATH)/libcore.a $(SRC\_PATH)/libtui.a

# $(GCC) $(CFLAGS) -o $@ $^ $(CLIBS)

$(PROG\_NAME): $(subst .c,.o,$(SRC\_PATH)/$(MAIN\_FILE) $(CORE\_FILES) $(TUI\_FILES))

$(GCC) $(CFLAGS) -o $@ $^ $(CLIBS)

%.o: %.c

$(GCC) -c $(CFLAGS) -o $@ $^ $(CLIBS)

$(SRC\_PATH)/libcore.a: $(subst .c,.o,$(CORE\_FILES))

ar rc $@ $^

ranlib $@

$(SRC\_PATH)/libtui.a: $(subst .c,.o,$(TUI\_FILES))

ar rc $@ $^

ranlib $@

**Исходный код файла «show\_element\_info\_window.c»**

#include "\_\_popups\_\_.h"

void tui\_draw\_popup\_header (WINDOW \* win, const wchar\_t \* popup\_name)

{

box (win, 0, 0);

mvwaddstr (win, 2, 0, "├");

for (size\_t i = 0; i < getmaxx(win) - 2; i++)

waddstr(win, "╌");

waddstr(win, "┤");

mvwprintw(win, 1, (getmaxx(win) - getmin(wcslen(popup\_name), getmaxx(win) - 2)) / 2, "%ls", popup\_name);

}

int get\_message\_lines\_count (const wchar\_t \* string, const int max\_width)

{

wint\_t \* curr\_symbol = string; // Символ строки

int lines\_cnt = 0, // Итоговое количество строк

char\_on\_line\_counter = 0; // Количество символов на строке

for (; (\*curr\_symbol != L'\0'); curr\_symbol++)

{

switch (\*curr\_symbol)

{

case L'\0': break;

case L'\n':

char\_on\_line\_counter = 0;

lines\_cnt++;

break;

default:

char\_on\_line\_counter++;

if (char\_on\_line\_counter >= max\_width)

{

lines\_cnt += 1;

char\_on\_line\_counter = 0;

}

break;

}

}

if (char\_on\_line\_counter % max\_width) lines\_cnt++;

return lines\_cnt;

}