МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

ст. преподаватель, программист.		Фоменкова А.А.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

Разработка программы «Список контактов»

по дисциплине: Основы программирования

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № М011		Борисов С.И.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2021

Содержание:

1. ПОСТАНОВКА ЗАДАЧИ 3 стр
2. ОПИСАНИЕ СТРУКТУР ДАННЫХ 9 стр
3. ОПИСАНИЕ АЛГОРИТМА И ФУНКЦИЙ 11 стр
4. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 20 стр
5. ЗАКЛЮЧЕНИЕ 34 стр

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 35 стр

ПРИЛОЖЕНИЕ 36 стр

1 ПОСТАНОВКА ЗАДАЧИ

Первичная постановка задачи

Вариант №6

Предметная область – «Список контактов»

Данные о человеке хранятся в структуре с именем NOTE, содержащей следующие поля:

- имя;
- номер телефона;
- дата рождения (массив из трёх чисел).

Задание на поиск: найти информацию о человеке, номер телефона.

Цель и задачи приложения

Разработать консольное приложение, которое осуществляет работу с базой данных (БД) списка контактов. Приложение позволяет пользователю добавлять и удалять контакты в БД, отображать содержимое БД, а также осуществлять поиск контактов в БД по номеру телефона или имени, задаваемому пользователем.

Сценарии использования

Предусловие.

Создается структура NOTE, содержащая следующие поля:

- имя;
- номер телефона;
- дата рождения (массив из трёх чисел).

В БД вводится несколько контактов в формате описанной структуры.

Сценарий: Запуск приложения.

1. Пользователь запускает приложение.
2. На экране отображается пронумерованный список пунктов меню и предложение ввести номер одного из пунктов.
3. Пользователь вводит номер пункта меню. Дальнейшее поведение приложения описано в последующих сценариях.

Сценарий: Добавление новой записи в БД.

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.

2. На экране отображается текст с предложением добавить информацию о новом контакте в БД приложения. С новой строки предлагается ввести данные о контакте.

3. Пользователь вводит имя нового контакта и нажимает Enter.

4. Приложение проверяет введенные данные. В зависимости от результата проверки:

a. Если пользователь ввел недопустимые символы, приложение выводит сообщение об ошибке и предлагает повторить ввод.

5. Если пользователь ввел данные корректно, приложение выводит сообщение с предложением ввести номер телефона нового контакта.

6. Пользователь вводит номер телефона и нажимает Enter.

7. Приложение проверяет введенные данные. В зависимости от результата проверки:

a. Если пользователь ввел недопустимые символы или сочетания символов, приложение выводит сообщение об ошибке и предлагает повторить ввод.

b. Если пользователь ввел данные корректно, приложение выводит сообщение с предложением ввести дату рождения нового контакта.

8. Пользователь вводит дату рождения и нажимает Enter.

9. Приложение проверяет введенные данные. В зависимости от результата проверки:

a. Если пользователь ввел недопустимую дату, приложение выводит сообщение об ошибке и предлагает повторить ввод.

b. Если пользователь ввел дату корректно, переход к следующему шагу.

10. Приложение проверяет, существует ли уже в БД запись с полями «Номер телефона» равными введенным пользователем:

a. Если такая запись существует, приложение выводит сообщение об ошибке и предлагает повторить ввод.

b. Если запись не найдена, то запись должна быть добавлена в БД. Приложение выводит сообщение о том, что контакт успешно добавлен в БД. При нажатии любой кнопки, пользователю отображается главное меню.

Сценарии: Редактирование контакта

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.
2. В консоль выводится пронумерованный список контактов. Предлагается ввести имя или номер контакта.
3. Пользователь вводит имя контакта.
    a. При правильном вводе имени выводиться контакт для редактирования. Пользователю демонстрируется меню с разными пунктами, предлагающими отредактировать отдельные элементы контакта (имя, номер телефона, дату рождения). Далее данные редактируются по выбранному пункту меню пользователем.
    b. При неправильном вводе контакта выводиться сообщение об ошибке.
    c. Если контакт введен правильно, но его нет в БД, то выводиться сообщение об этом и предлагается выйти в главное меню.
4. Пользователь вводит номер телефона.
    a. При правильном вводе номера телефона выводиться контакт для редактирования. Пользователю демонстрируется меню с разными пунктами, предлагающими отредактировать отдельные элементы контакта (имя, номер телефона, дату рождения). Далее данные редактируются по выбранному пункту меню пользователем.
    b. При неправильном вводе номера телефона выводиться сообщение об ошибке.
    c. Если контакт введен правильно, но его нет в БД, то выводиться сообщение об этом и предлагается выйти в главное меню.

Сценарий: Удаление записи из БД.

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.

2. На экране отображается БД и меню с выбором удалить запись по номеру в списке или данным контакта. С новой строки предлагается ввести пункт меню.

3. Пользователь вводит пункт меню 1 и нажимает Enter.

4. Пользователю предлагается ввести номер в списке.

Если такая запись существует, приложение выводит удаляемый контакт и сообщение о его удалении.

Если ввод осуществлен не корректно, то выводится сообщение о повторении ввода.

5. Пользователь вводит пункт меню 2 и нажимает Enter.

6. Пользователь вводит номер телефона или имя и нажимает Enter.

a. Если пользователь ввел недопустимые символы или сочетания символов, приложение выводит сообщение об ошибке и предлагает повторить ввод.

b. Если такая запись существует, приложение выводит удаляемый контакт и сообщение о его удалении.

c. Если такой записи не существует, приложение выводит сообщение об этом и предлагает выйти ы главное меню.

Сценарий: Сортировка БД контакты.

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.

2. В консоли выводится меню с предложенными сортировками (1 – сортировка по имени; 2 – сортировка по номеру телефона)

3. Сортировка по имени (в алфавитном порядке)

a. Пользователь вводит пункт меню – 1 и нажимает Enter.

b. В консоль выводится список контактов до сортировки.

c. В консоль выводится список контактов после сортировки.

d. Пользователю предлагается сохранить отсортированный список.

4. Сортировка по номеру телефона (осуществляется по убыванию)

a. Пользователь вводит пункт меню – 2 и нажимает Enter.

b. В консоль выводится список контактов до сортировки.

c. В консоль выводится список контактов после сортировки.

d. Пользователю предлагается сохранить отсортированный список.

Сценарий: Поиск контакта.

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.

2. На экране отображается БД контакты и меню с выбором найти контакт по имени или номеру телефона. С новой строки предлагается ввести пункт меню, по которому будет осуществляться поиск.

3. Пользователь вводит 1 и нажимает Enter.

a. Если пользователь ввел недопустимые символы или сочетания символов, приложение выводит сообщение об ошибке и предлагает повторить ввод.

4. Пользователь вводит номер телефона

a. Если ввод корректен, осуществляется вывод найденного контакта

b. Если ввод не корректен, приложение уведомляет об этом пользователя. Так же дает возможность ввести номер телефона заново.

5. Пользователь вводит пункт меню 2 и нажимает Enter.

a. Если пользователь ввел недопустимые символы или сочетания символов, приложение выводит сообщение об ошибке и предлагает повторить ввод.

6. Пользователь вводит имя контакта

a. Если ввод корректен, осуществляется вывод найденного контакта

b. Если ввод не корректен, приложение уведомляет об этом пользователя. Так же дает возможность ввести номер телефона заново.

Сценарий: Вывод на экран содержимого БД

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.

2. На экране отображается пронумерованный список содержимого БД. При нажатии любой кнопки, пользователю отображается главное меню.

Сценарий: Сохранения БД контакты в файл формата txt.

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.

2. Осуществляется проверка формата БД контакты

3. БД контакты сохраняется в файл.

Сценарий: Загрузка БД контакты из файла формата txt.

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню.

2. Осуществляется проверка формата БД контакты

3. БД контакты загружается из файла.

Сценарий: Выход из приложения

1. Пользователь запускает приложение и вводит соответствующий номер пункта меню. Приложение заканчивает работу.

2 ОПИСАНИЕ СТРУКТУР ДАННЫХ

1.1 Описание структуры данных для хранения информации о контактах.

База данных контакты организованна в виде двунаправленного списка. Каждая запись списка представляет собой:

```
struct ListNOTE
{
	NOTE* Head; // Указатель на голову
	NOTE* Tail; // Указатель на хвост
	int size;
};
```

Данные о контактах хранятся в структуре NOTE.

```
struct NOTE
{
	int num;
	string Name; // Имя контакта
	string Phone; // Номер телефона контакта
	int birthday[3]; // День рождения контакта

	NOTE* next; // Указатель на следующий элемент
};
```

Структура NOTE содержит следующие поля:

- Name – имя контакта. Поле может содержать буквы латинского и русского алфавитов. Для хранения используется тип string, что позволяет автоматически регулировать размер строки.
- Phone – номер телефона контакта. Поле может содержать натуральные числа от 0 до 9 в количестве 11.
- birthday[3] – дата дня рождения контакта, являющейся массивом из 3 чисел (год, месяц, день). Тип данных использован int. Поле хранит натуральные числа.

База данных (БД) контакты хранится на диске в виде текстового файла в формате TXT.

1.2 Описание данных, запрашиваемых у пользователя.

Разрабатываемая программа предполагает текстовый интерфейс взаимодействия с пользователем. Введенные пользователем данные проходят проверки в соответствии с таблицей 1.1.

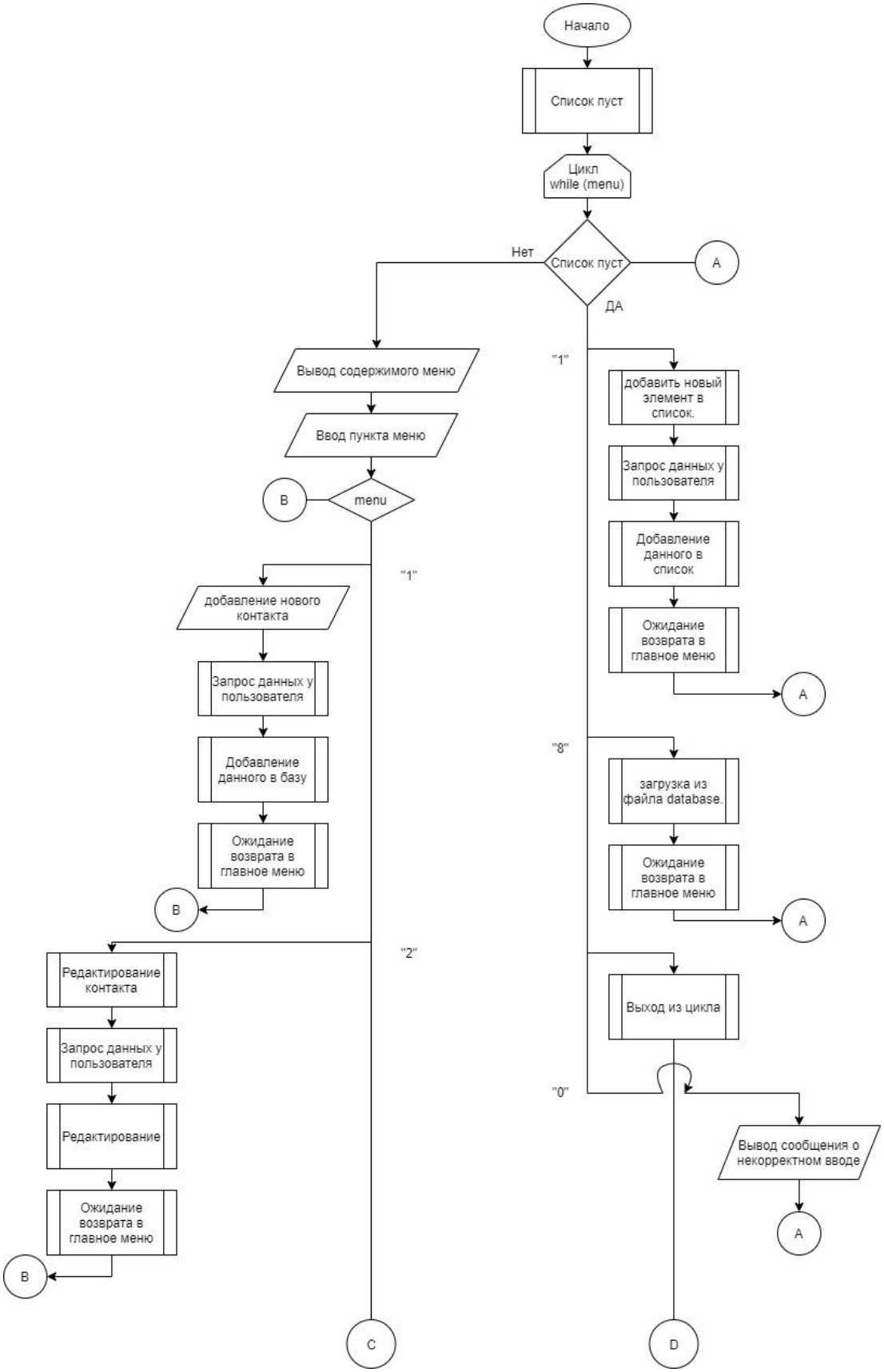
Таблица 1.1- Описание данных, вводимых пользователем

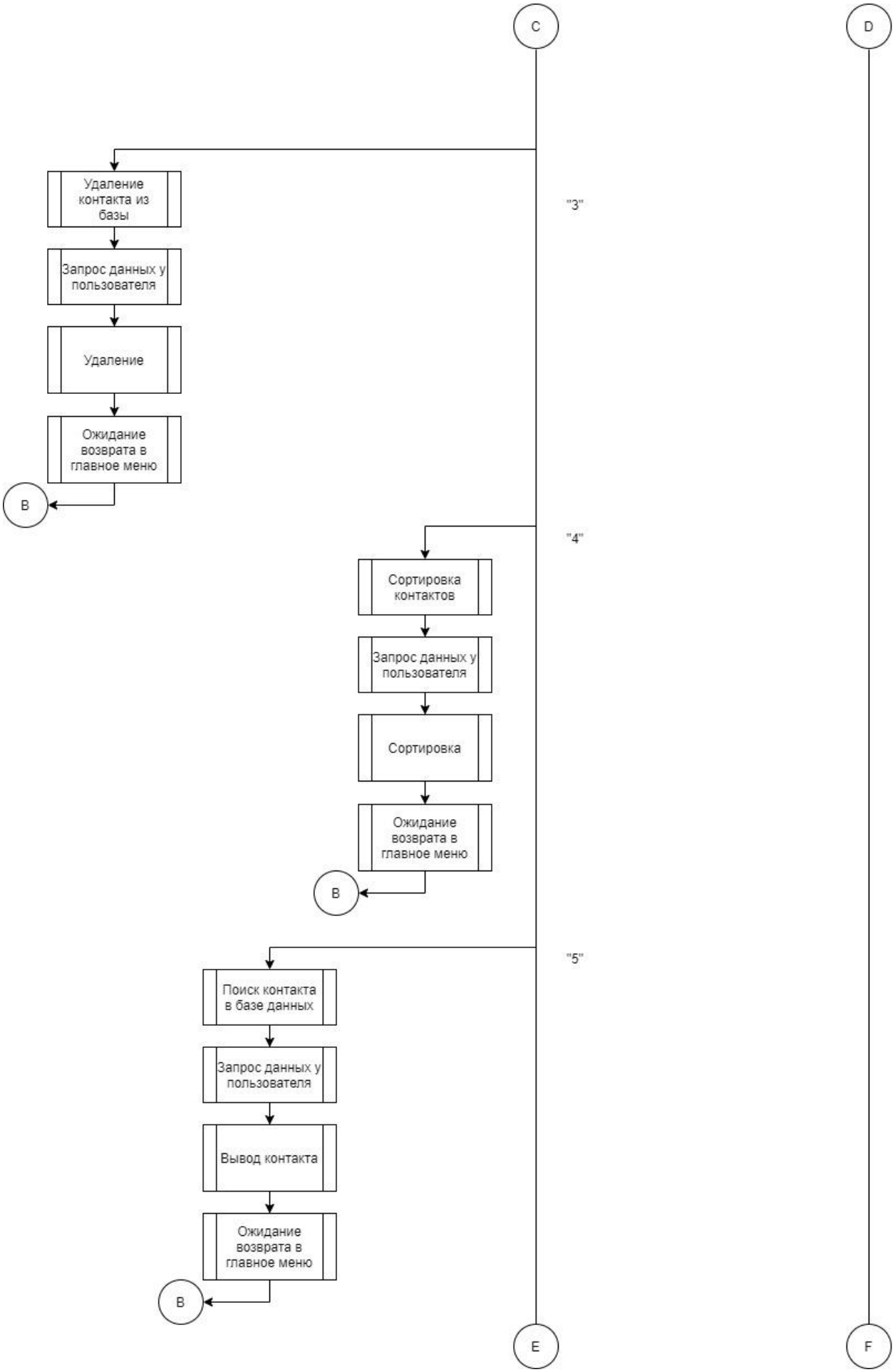
Наименование переменной	Тип данных	Семантика	Описание проводимых проверок

menu	int	Основное меню	Натуральное число от 0 до 8
Name	string	Имя контакта	буквы латинского и русского алфавитов
Phone	string	Номер телефона контакта	Натуральные числа от 0 до 9. Размер строки составляет 11 чисел.
birthday	int	Дата рождение контакта	Год – натуральное число в диапазоне [1900;2021] Месяц – натуральное число в диапазоне [1;12] День – натуральное число. Диапазон вводимого числа зависит от введенных ранее параметром Года и Месяца.

3 ОПИСАНИЕ АЛГОРИТМА И ФУНКЦИЙ

Точкой входа в программу является функция main(), организующая вывод меню пользователя и вызов остальных функций.




C
D
Удаление контакта из базы
Запрос данных у пользователя
Удаление
Ожидание возврата в главное меню
B
"3"
"4"
Сортировка контактов
Запрос данных у пользователя
Сортировка
Ожидание возврата в главное меню
B
"5"
Поиск контакта в базе данных
Запрос данных у пользователя
Вывод контакта
Ожидание возврата в главное меню
B
E
F

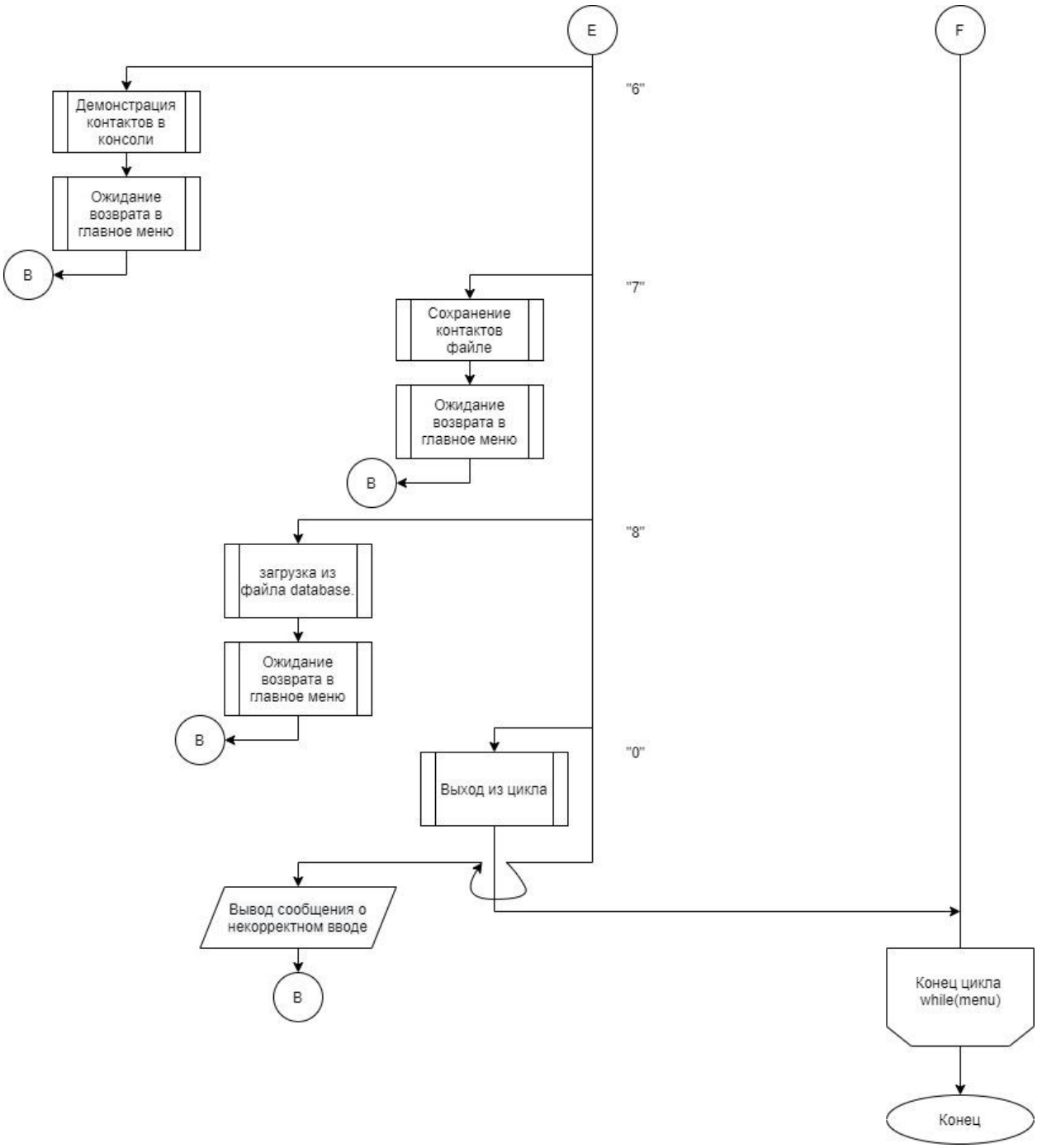


Рисунок 3.1 – Блок-схема алгоритма работы функции main()

В программе выполняется функция вывода на консоль имеющихся в БД контактов. Если в базе хранятся данные о контактах, то эта функция сможет их вывести, иначе будет выведено сообщение о пустом массиве. Блок-схема алгоритма представлена на рис. 3.2.

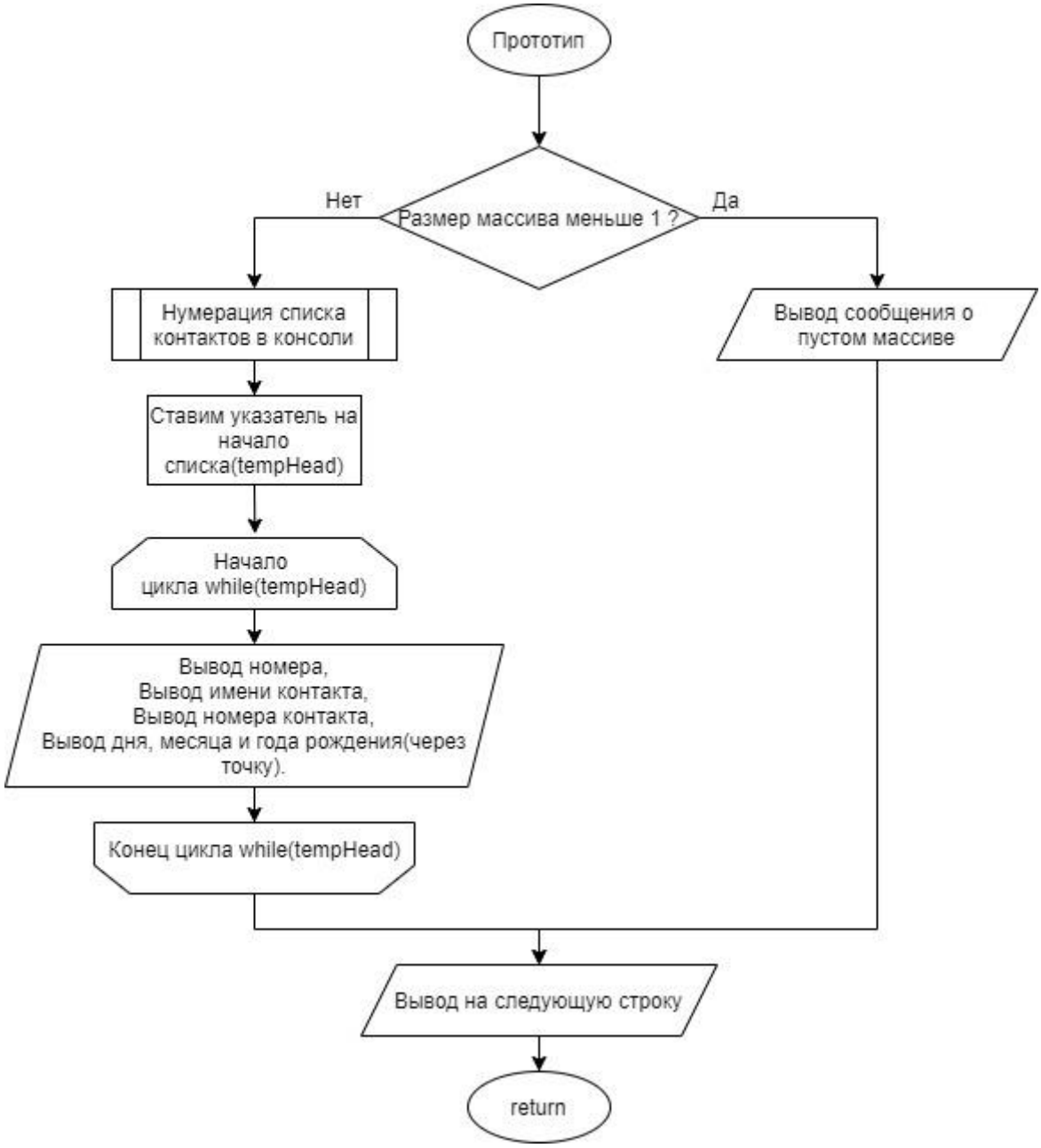


Рисунок 3.2 – Блок-схема алгоритма работы функции Show()

Для реализации поиска контактов по указанному пользователем номеру телефона разработана функция Search_by_Phone, которая выполняет поиск методом последовательного перебора. Блок-схема функции Search_by_Phone представлена на рис.3.3

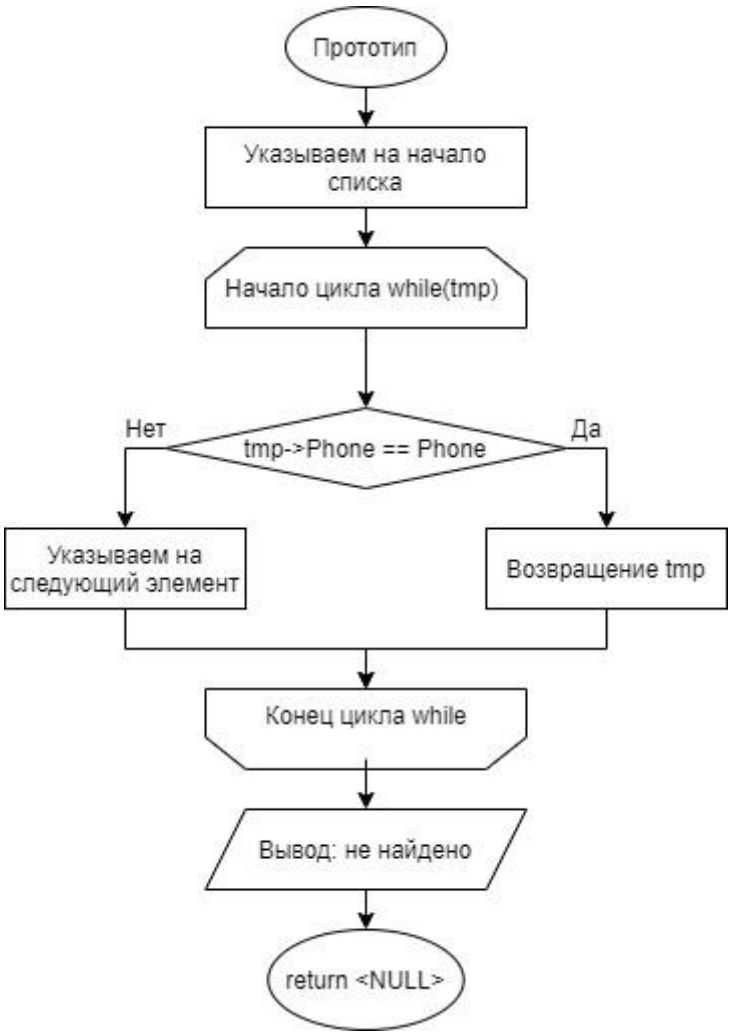


Рисунок 3.3 – Блок-схема алгоритма работы функции Search_by_Phone

В таблице 3.1 приведен список всех разработанных для реализации программы функций и процедур.

Таблица 3.1 – Перечень разработанных функций и процедур

Прототип	**Входные параметры**	**Выходные параметры**	**Описание функционала**
`int inputint()`		Целочисленное число x	Проверка на целочисленный формат.
`void AddList(ListNOTE*& list, NOTE*& ya4eeka)`	list – структура списка,	-	Добавление контакта в список.

	ya4eeka- указатель ячейку списка		
void numbering(ListNOTE*& list)	list – структура списка	-	Нумерация контактов в консоли.
void Show(ListNOTE* list)	list – структура списка	-	Демонстрация списка контактов в консоли.
void Delete(ListNOTE*& list, NOTE* ya4eeka)	list – структура списка, ya4eeka- указатель на ячейку списка	-	Удаление контакта из списка.
NOTE* Search_by_Phone(ListNOTE*& list, string Phone)	list – структура списка, Phone- поле списка, содержащее номер телефона	Ячейка списка с найденным номером телефона	Поиск по номеру телефона.
NOTE* Search_by_Name(ListNOTE*& list, string Name)	list – структура списка, Name- поле списка, содержащее Имя	Ячейка списка с найденным именем контакта	Поиск по имени контакта.
void Search_by_NameALL(ListNOTE*& list, string Name)	list – структура списка, Name- поле списка, содержащее Имя	-	Вывод похожих имен из списка.
void littleprint(NOTE* tmp)	tmp- указатель на ячейку списка	-	Вывод контакта.

`string nameconvert(string name)`	Строка name	Возвращает name	Преобразование имени при выводе из файла.
`string nameunconvert(string name)`	Строка name	Возвращает name	Преобразование имени при вводе в файл.
`bool namecheck(string name)`	Строка name	Возвращает или правду или ложь	Проверка на имя.(Русские или английские, большие или маленькие буквы, написание через дефис”-”)
`bool phonecheck(string Phone)`	Phone- поле списка, содержащее номер телефона	Возвращает или правду или ложь	Проверка номера телефона(11 цифр)
`bool phonesamecheck(ListNOTE*& list, string Phone)`	list – структура списка, Phone- поле списка, содержащее номер телефона	Возвращает или правду или ложь	Проверка номера телефона в списке (номера тел. Не должно быть больше одного)
`bool daycheck(int* BIRTHDAY)`	BIRTHDAY-указатель на день рождения	Возвращает или правду или ложь	Проверка на дату рождения(День[от 1 до корректной цифры в ранее введенном месяце] Месяц[от 1 до 12] Год[от 1900 до 2021])
`string inputstring(int flag)`	Flag-переменная (принимает на вход 1 либо 2)	Возвращает строку	Проверка данных вводимых пользователем. (1

			– проверка имени. 2- проверка номера телефона)
`bool ultimatecheck(NOTE*& tmp, ListNOTE*& list)`	list – ссылается на список, tmp- ссылается на ячейку списка	Возвращает или правду или ложь	Окончательная проверка (для вывода данных из файл)
`void dobavlenie_elementa_v_spisok(ListNOTE*& list)`	list – структура списка	-	Добавление элемента в список.
`int nalichie_imen_v_spiske(ListNOTE*& list, string name)`	list – структура списка, name- поле списка, содержащее имя	Количество найденных имен	Проверка на наличие имен в списке.
`int nalichie_nomera_telepona_v_spiske(ListNOTE*& list, string Phone)`	list – структура списка, Phone- поле списка, содержащее номер телефона	1 – если телефон есть в списке. 0 – если телефона нет в списке.	Проверка на наличие номера тел в списке.
`void redactirovanie_elementa_v_spiske(ListNOTE*& list)`	list – структура списка	-	Редактирование элемента в списке.
`void notedelete(ListNOTE*& list)`	list – структура списка	-	Удаление контакта по данным(Имя или телефон)
`void fileout(ListNOTE*& list)`	list – структура списка	-	Сохранение данных в файл.
`void filein(ListNOTE*& list)`	list – структура списка	-	Загрузка данных из файла.

int comp_name(NOTE* num1, NOTE* num2)	num1- указатель на имя num2- указатель на имя	Возвращает или правду или ложь	Функция проходит по полю списка имен контактов, сравнивая их.
ListNOTE* data_sort_name(ListNOTE*& list)	list – структура списка	Возвращает список (отсортированный или не отсортированный)	Сортировка по имени(в алфавитном порядке)
int comp_phone(NOTE* num1, NOTE* num2)	num1- указатель на номер num2- указатель на номер	Возвращает или правду или ложь	Функция проходит по полю списка номеров телефонов, сравнивая их.
ListNOTE* data_sort_phone(ListNOTE*& list)	list – структура списка	Возвращает список (отсортированный или не отсортированный)	Сортировка по номеру телефона(по убыванию)

4 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Для проверки корректности работы приложения были использованы тестовые данные, приведенные в таблице 3.1.

Таблица 3.1 – Тестовые данные

№	*Название этапа тестирования*	*Тестовые данные*	*Ожидаемый результат*
1	**Главное меню**	-	Вывод на экран окна меню
1.1	Некорректный ввод пункта меню	qwe	Вывод сообщения: Повторите ввод.
		4.5	Вывод сообщения: Повторите ввод.
2	**Добавление новой записи в БД**	Выбор пункта меню 1	Вывод сообщения о предложении ввода имени контакта.
2.1	Ввод имени контакта	Луня	Переход к вводу номера телефона
2.1.1	Некорректный ввод имени контакта	Луня3	Вывод сообщения: 2 - Недопустимый символ! Некорректный ввод. Пожалуйста, повторите:
2.2	Ввод номера телефона	55555555555	Переход к вводу даты рождения
2.2.1	Некорректный ввод номера телефона	уиав	Вывод сообщения: Некорректный ввод. Пожалуйста, повторите:
2.2.2	Если номер уже есть в БД	11111111111	Некорректный ввод (такой номер уже есть в базе). Повторите:
2.3	Ввод даты рождения	-	Вывод сообщения об успешной загрузке в БД
2.3.1	Ввод года	2002	Переход к вводу месяца
2.3.2	Некорректный ввод даты рождения	«2222» или «уиав»	Вывод сообщения: Некорректный ввод. Повторите:
2.3.3	Ввод месяца	10	Переход к вводу дня
2.3.4	Некорректный ввод даты рождения	«2222» или «уиав»	Вывод сообщения: Некорректный ввод. Повторите:
2.3.5	Ввод дня	10	Вывод сообщения об успешном добавлении контакта в БД
2.3.6	Некорректный ввод даты рождения	«2222» или «уиав»	Вывод сообщения: Некорректный ввод. Повторите:
3	**Редактирование БД контакты**	Выбор пункта меню 2	Вывод БД и предложении редактировать контакт по имени или номеру телефона.
3.1	Ввод имени	Петр	Вывод меню с предложением редактирования имени, номера телефона, даты рождения
3.1.1	Некорректный ввод имени	Петр3	Выводиться в консоли: : Петр3 Проверяю имя ли это? 3 - Недопустимый символ!

			Проверяю номер ли это? Некорректный ввод. Повторите:
3.1.2	Ввод несуществующего имени в БД	Ирина	Выводиться в консоли: Ирина Это имя! Ищу в базе: Не найдено: Для продолжения нажмите любую клавишу
3.2	Ввод номера телефона	66666666666	Вывод меню с предложением редактирования имени, номера телефона, даты рождения
3.2.1	Некорректный ввод номера телефона	1111111111з	Проверяю имя ли это? 1 - Недопустимый символ! Проверяю номер ли это? Некорректный ввод. Повторите:
3.2.2	Ввод несуществующего номера телефона в БД	11111111186	Проверяю имя ли это? 1 - Недопустимый символ! Проверяю номер ли это? Это номер! Ищу в базе: Не найдено:
4	**Удаление записи из БД**	Выбор пункта меню 3	Вывод БД и меню, сообщающее о предложении удалить контакт по номеру в списке или данным контакта.
4.1	Ввод пункта меню	Ввод пункта меню: – 1 (удалить по номеру в списке) Ввод пункта меню – 2 (удалить по данным) Другая цифра – выход в главное меню	При вводе - 1 предлагается ввести номер в списке При вводе - 2 предлагается ввести имя или номер телефона При вводе другой цифры – осуществляется выход в главное меню
4.1.1	Некорректный ввод пункта меню	аочр	Вывод сообщения: Повторите ввод:
4.2	Удаление по номеру в списке	25	Вывод удаляемого контакта. Вывод сообщения: Удалено
4.2.1	Некорректный ввод номера списка	30	Вывод сообщения: Повторите ввод:
4.3	Удаление по данным контакта	-	Вывод сообщения о вводе имени или номера тел. контакта.
4.3.1	Ввод имени	Петр	Вывод сообщения: Удалено. Для продолжения нажмите любую клавишу
4.3.2	Некорректный ввод данных	Луня3	Вывод сообщения: 3 - Недопустимый символ! Проверяю номер ли это? Пользователь повторяет ввод.
4.3.3	Ввод несуществующего контакта в БД	Ирина	Вывод сообщения: Этого имени абонента нет.

			Для продолжения нажмите любую клавишу
4.4	Ввод номера телефона	66666666666	Вывод удаляемого контакта. Вывод сообщения: Удалено
4.4.1	Некорректный ввод номера телефона	8кив5964888	8 - Недопустимый символ! Пользователь повторяет ввод.
4.4.2	Ввод несуществующего номера телефона в БД	22222222222	2 - Недопустимый символ! Проверяю номер ли это? Это номер! Ищу в базе: не найдено
4.4.3	Если есть одинаковые имена	Борисов Сергей	Происходит вывод контактов с одинаковыми именами. Приложение предлагает ввести номер телефона так как он уникален
5	**Сортировка БД**	Выбор пункта меню 4	Вывод БД и меню, сообщающее о предложении отсортировать контакт по номеру телефона или имени.
5.1	Сортировка по имени	Выбор пункта меню 1	Вывод отсортированного списка в алфавитном порядке и предложении о сохранении его в таком виде
5.2	Сортировка по номеру телефона	Выбор пункта меню 2	Вывод отсортированного списка номеров телефонов по убыванию и предложении о сохранении его в таком виде
6	**Поиск контакта**	Выбор пункта меню 5	Вывод БД и меню, сообщающее о предложении найти контакт по номеру телефона или имени контакта.
6.1	Ввод пункта меню	Ввод пункта меню: – 1 (поиск по номеру телефона) Ввод пункта меню – 2 (поиск по имени) Другая цифра – выход в главное меню	При вводе - 1 предлагается ввести номер телефона При вводе - 2 предлагается ввести имя При вводе другой цифры – осуществляется выход в главное меню
6.1.1	Некорректный ввод пункта меню	аочр	Вывод сообщения: Повторите ввод:
6.2	Ввод номера телефона	66666666666	Осуществляется вывод найденного контакта
6.2.1	Некорректный ввод номера телефона	6з666666666	Вывод сообщения: Телефон не найден или некорректен! Проверьте правильность ввода! (для выхода введите 0) Пользователь может ввести номер заново или выйти в главное меню

6.3	Ввод имени	Луня	Осуществляется вывод найденного контакта
6.3.1	Некорректный ввод пункта номера телефона	Луня3	Вывод сообщения: 3 - Недопустимый символ! Некорректный ввод. Повторите:
6.4	Ввод несуществующего контакта	Ирина	Вывод сообщения: Данного контакта НЕТ! Для продолжения нажмите любую клавишу
7	**Вывод на экран БД**	Выбор пункта меню 6	Осуществляется вывод БД контакты
8	**Сохранение БД контакты в файл**	-	Осуществляется сохранение в файл «database.txt»
8.1	Файл существует	-	В консоли выводиться сообщение об успешной загрузке. В файле находиться список группы М011
8.2	Файла не существует	-	Создается файл и выводиться сообщение об успешной загрузке. В файле находиться: Борисов.Сергей 77777777777 1 1 2020
9	**Загрузка БД контакты из файла**	Выбор пункта меню 8	Осуществляется загрузка из файла «database.txt».
9.1	Файл существует	-	В консоли выводиться сообщение об успешной загрузке. В файле находиться список группы М011
9.2	В файле ничего нет или его не существует.	-	Пользователю выводиться сообщение об этом и предлагается добавить новый контакт.
10	**Выход из приложения**	Выбор пункта меню 0	Осуществляется выход из приложения

Результаты тестирования приложения по данным таблицы 3.1 приведены ниже.

1 Главное меню:

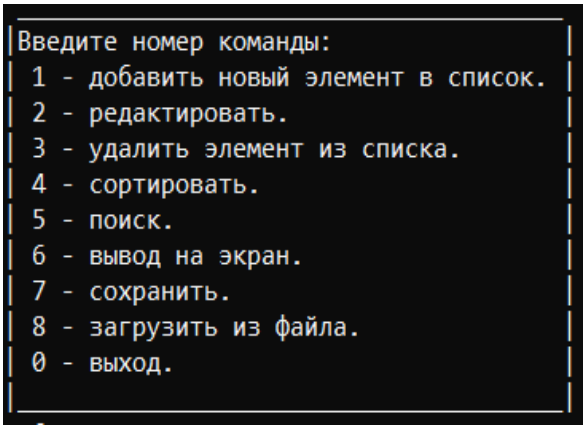


Рисунок 1 – Главное меню

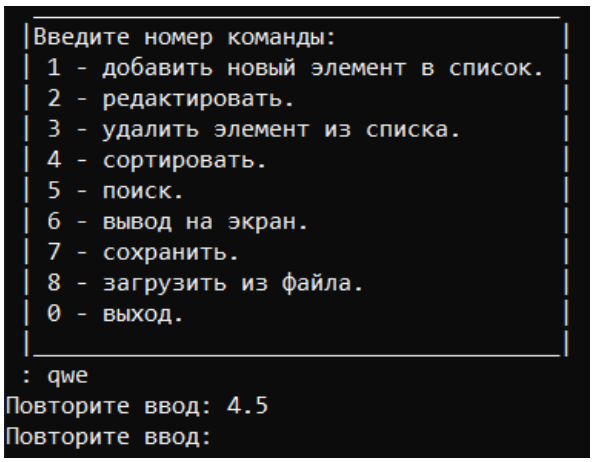


Рисунок 1.1 – Некорректный ввод пункта меню

2 Добавление новой записи в БД:

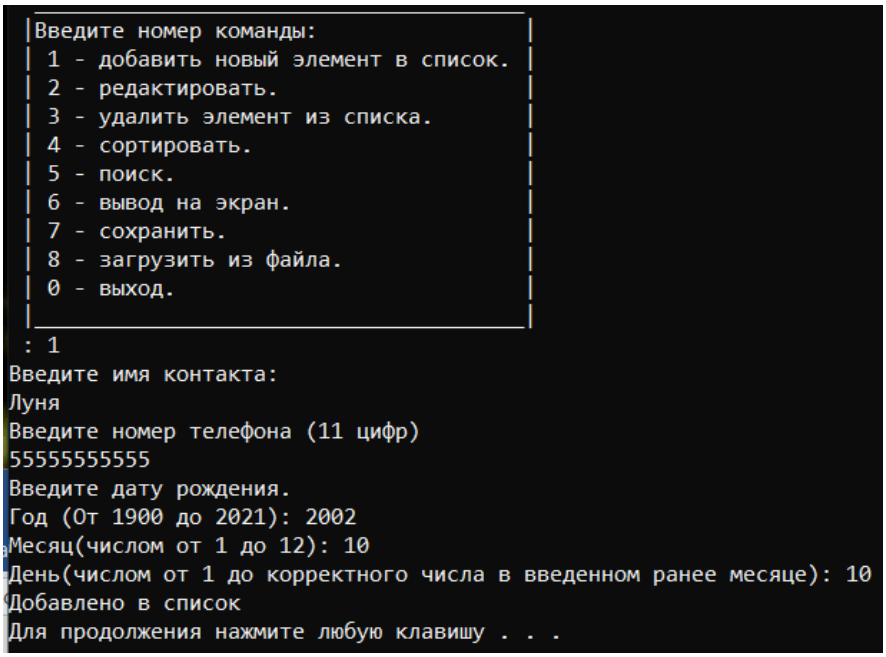


Рисунок 2 включающий пункты: 2.1 2.2 2.3 2.3.1 2.3.3 2.3.5 – Добавление корректный данных в БД

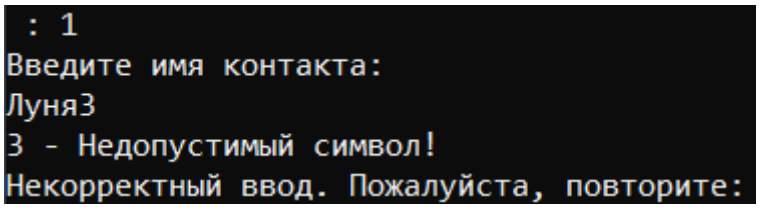


Рисунок 2.1.1 – Некорректный ввод имени контакта

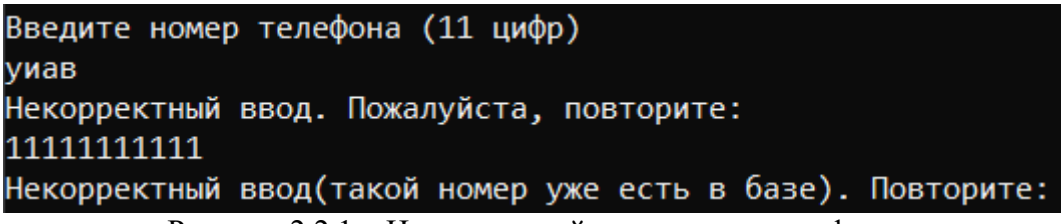


Рисунок 2.2.1 – Некорректный ввод номера телефона

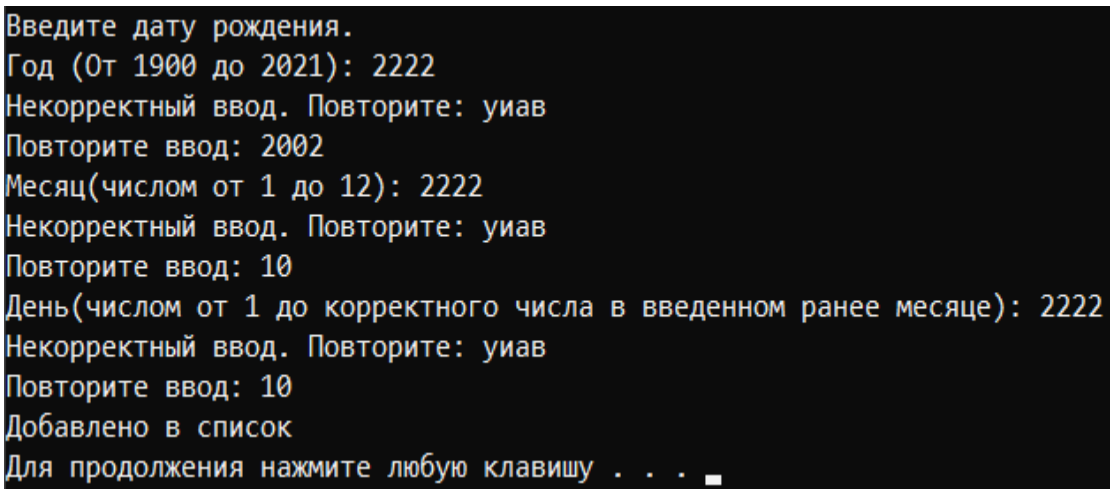


Рисунок 2.3 включающий пункты: 2.3.2 2.3.4 2.3.6 – Некорректный ввод даты рождения

3 Редактирование БД контакты:

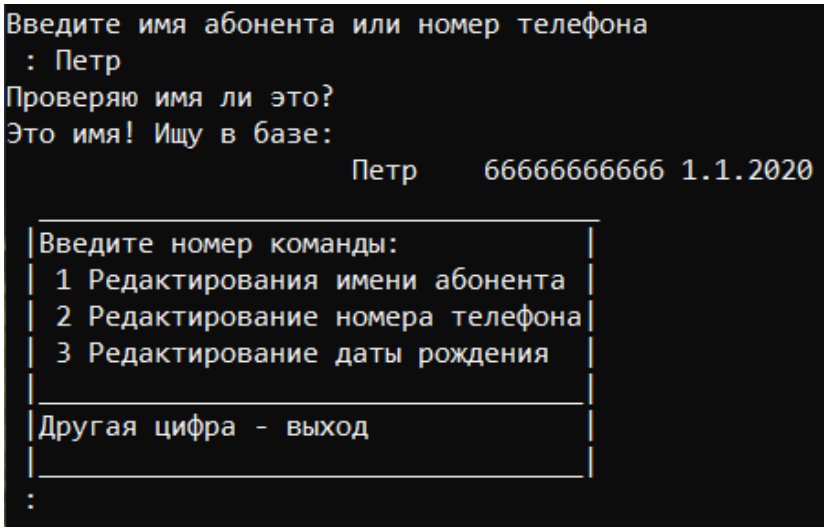


Рисунок 3.1 – Ввод имени

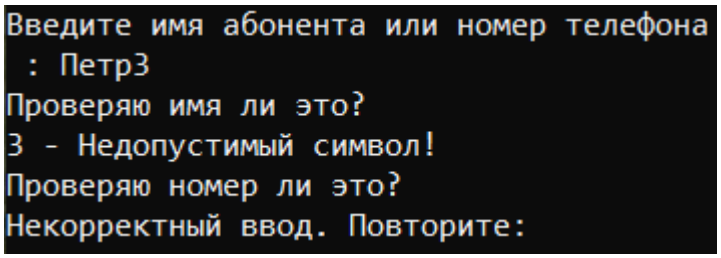


Рисунок 3.1.1 – Некорректный ввод имени

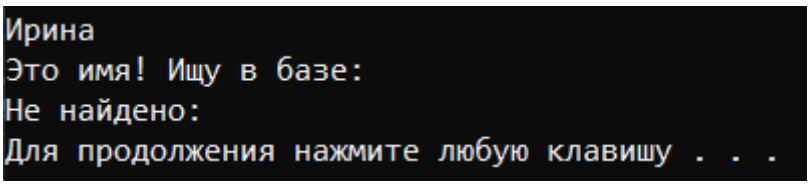


Рисунок 3.1.2 – Ввод несуществующего имени в БД

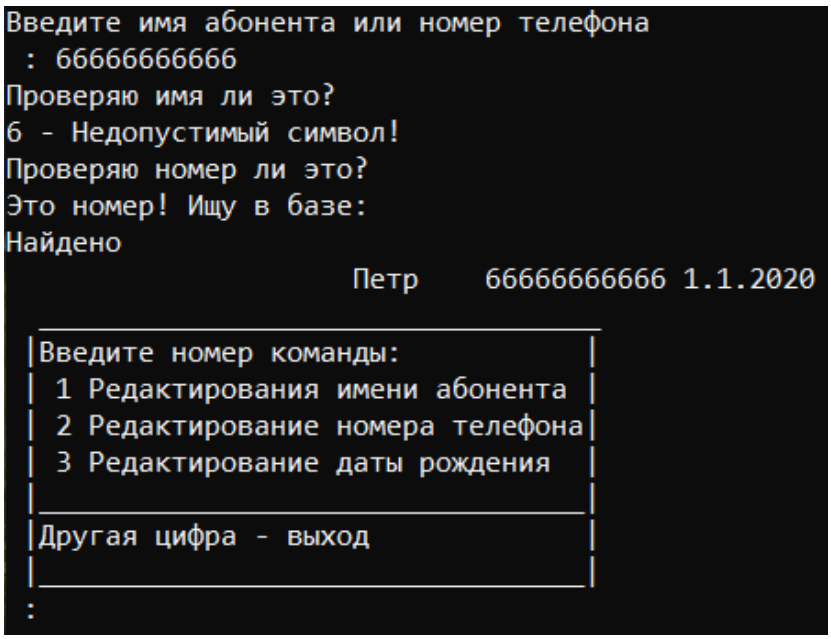


Рисунок 3.2 – Ввод номера телефона

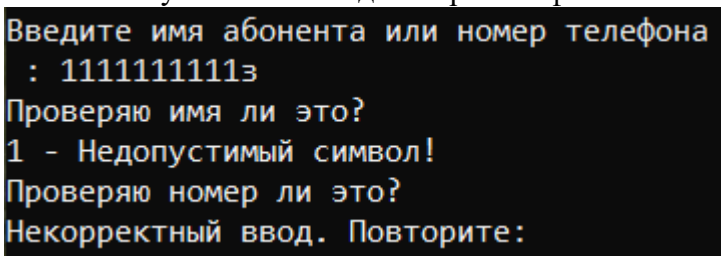


Рисунок 3.2.1 – Некорректный ввод номера телефона

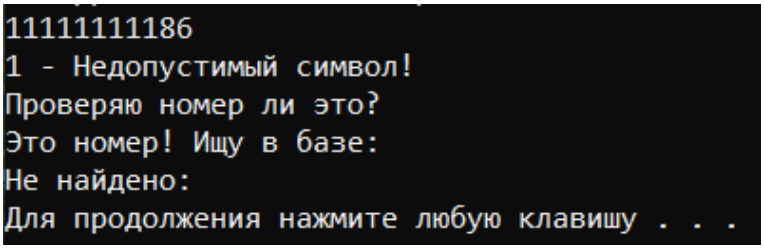


Рисунок 3.2.2– Ввод несуществующего номера телефона в БД

4 Удаление записи из БД:

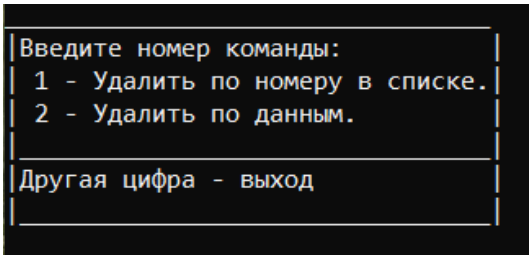


Рисунок 4.1 – Ввод пункта меню

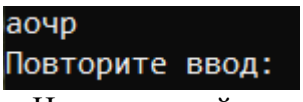


Рисунок 4.1.1 – Некорректный ввод пункта меню

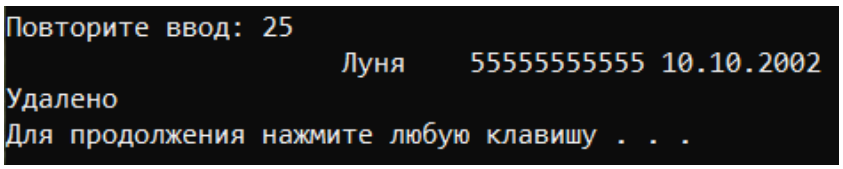


Рисунок 4.2 – Удаление по номеру в списке

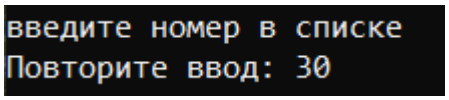


Рисунок 4.2.1 – Некорректный ввод номера списка

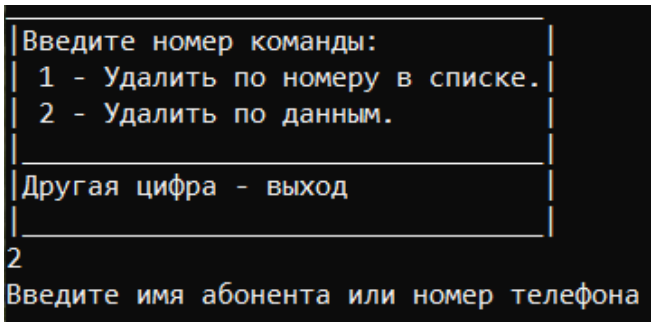


Рисунок 4.3 – Удаление по данным контакта

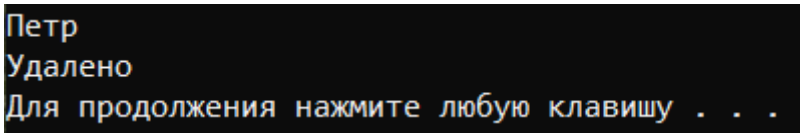


Рисунок 4.3.1 – Ввод имени

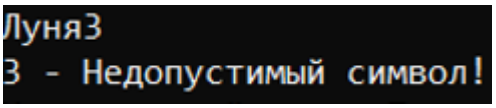


Рисунок 4.3.2 – Некорректный ввод данных

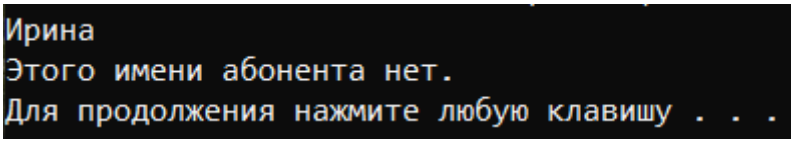


Рисунок 4.3.3 – Ввод несуществующего контакта в БД

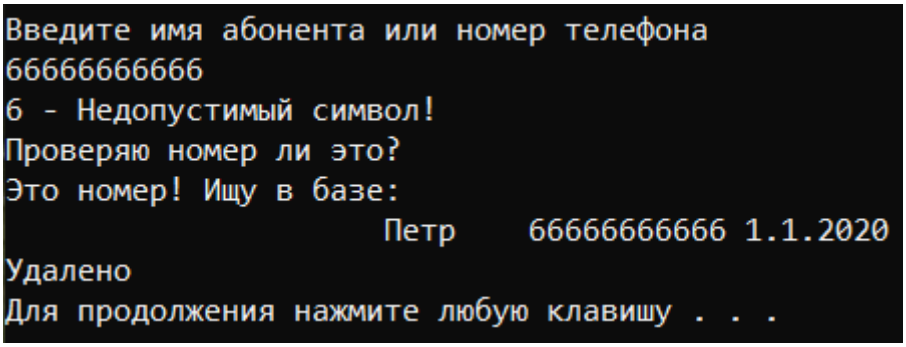


Рисунок 4.4 – Ввод номера телефона

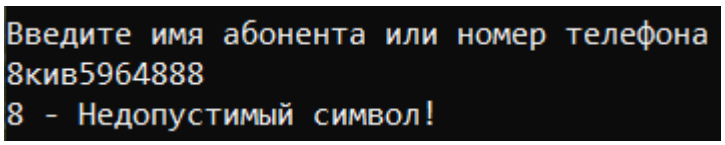


Рисунок 4.4.1 – Некорректный ввод номера телефона

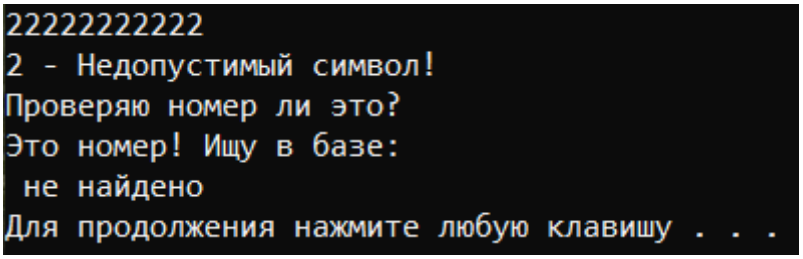


Рисунок 4.4.2 – Ввод несуществующего номера телефона в БД

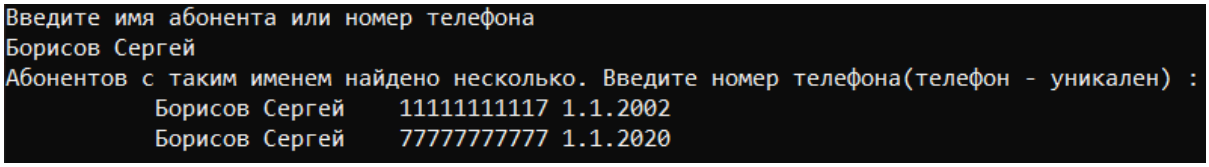


Рисунок 4.4.3 –Если есть одинаковые имена

5 Сортировка БД:

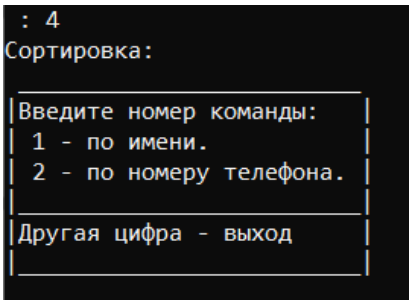


Рисунок 5 – Вывод БД и меню сортировки

Рисунок 5.1 – Сортировка по имени

Рисунок 5.2 – Сортировки по номеру телефона

6 Поиск контакта:

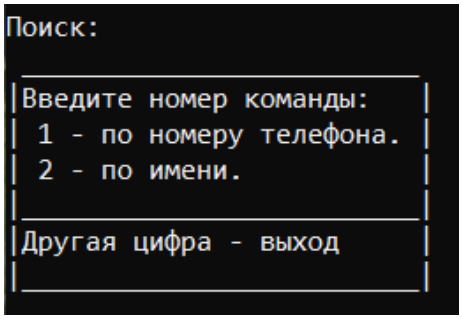


Рисунок 6.1 – Ввод пункта меню

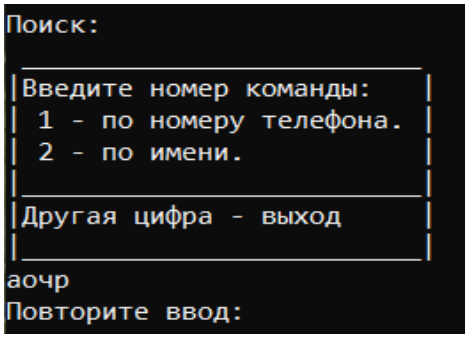


Рисунок 6.1.1 – Некорректный ввод пункта меню

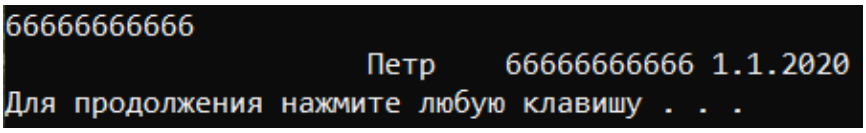


Рисунок 6.2 – Ввод номера телефона

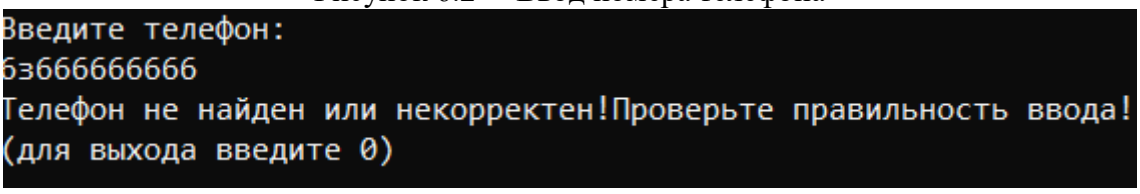


Рисунок 6.2.1 – Некорректный ввод номера телефона

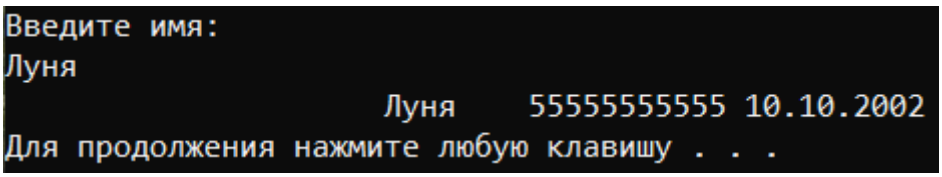


Рисунок 6.3 – Ввод имени

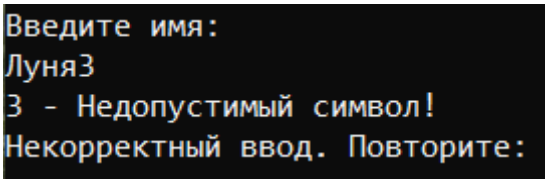


Рисунок 6.3.1 – Некорректный ввод пункта номера телефона

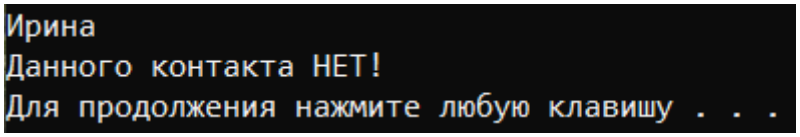


Рисунок 6.4 – Ввод несуществующего контакта

7 Вывод на экран БД контакты:

```
1 Аксененко Богдан    11111111111 8.4.2002
2 Анисимов Игорь    11111111112 16.5.2002
3 Арутюнян Вячеслав    11111111113 15.3.2002
4 Батищев Кирилл    11111111114 1.1.2003
5 Берзин Дмитрий    11111111115 27.11.2002
6 Бовыкина Полина    11111111116 5.2.2002
7 Борисов Сергей    11111111117 1.1.2002
8 Высоцкая Анастасия    11111111118 17.9.2001
9 Горлов Никита    11111111119 8.11.2002
10 Долгова Майя    11111111120 13.5.2002
11 Жвакин Андрей    11111111121 3.11.2002
12 Заякина Ульяна    11111111122 7.6.2002
13 Казакова Полина    11111111123 23.11.2001
14 Калашников Илья    11111111124 1.1.2002
15 Лиходедова Валерия    11111111125 24.7.2002
16 Лючков Дмитрий    11111111126 1.1.2002
17 Михайлов Данил    11111111127 1.1.2002
18 Мищенко Александр    11111111128 1.1.2002
19 Наговицына Софья    11111111129 1.1.2002
20 Подграмский Иван    11111111130 5.7.2002
21 Портнягин Максим    11111111131 15.8.2002
22 Соболева Алиса    11111111132 13.11.2002
23 Уланский Никита    11111111133 1.1.2002
24 Шиловский Степан    11111111134 19.8.2001
```

Рисунок 7 – Вывод на экран БД контакты

8 Сохранение БД контакты в файл:

```
Введите номер команды:
 1 - добавить новый элемент в список.
 2 - редактировать.
 3 - удалить элемент из списка.
 4 - сортировать.
 5 - поиск.
 6 - вывод на экран.
 7 - сохранить.
 8 - загрузить из файла.
 0 - выход.
: 7
Сохранено!
Для продолжения нажмите любую клавишу . . .
```

database – Блокнот

Файл Правка Формат Вид Справка

```
Аксененко.Богдан 11111111111 8 4 2002
Анисимов.Игорь 11111111112 16 5 2002
Арутюнян.Вячеслав 11111111113 15 3 2002
Батищев.Кирилл 11111111114 1 1 2003
Берзин.Дмитрий 11111111115 27 11 2002
Бовыкина.Полина 11111111116 5 2 2002
Борисов.Сергей 11111111117 1 1 2002
Высоцкая.Анастасия 11111111118 17 9 2001
Горлов.Никита 11111111119 8 11 2002
Долгова.Майя 11111111120 13 5 2002
Жвакин.Андрей 11111111121 3 11 2002
Заякина.Ульяна 11111111122 7 6 2002
Казакова.Полина 11111111123 23 11 2001
Калашников.Илья 11111111124 1 1 2002
Лиходедова.Валерия 11111111125 24 7 2002
Лючков.Дмитрий 11111111126 1 1 2002
Михайлов.Данил 11111111127 1 1 2002
Мищенко.Александр 11111111128 1 1 2002
Наговицына.Софья 11111111129 1 1 2002
Подграмский.Иван 11111111130 5 7 2002
Портнягин.Максим 11111111131 15 8 2002
Соболева.Алиса 11111111132 13 11 2002
Уланский.Никита 11111111133 1 1 2002
Шиловский.Степан 11111111134 19 8 2001
```

Рисунок 8.1 – Файл существует

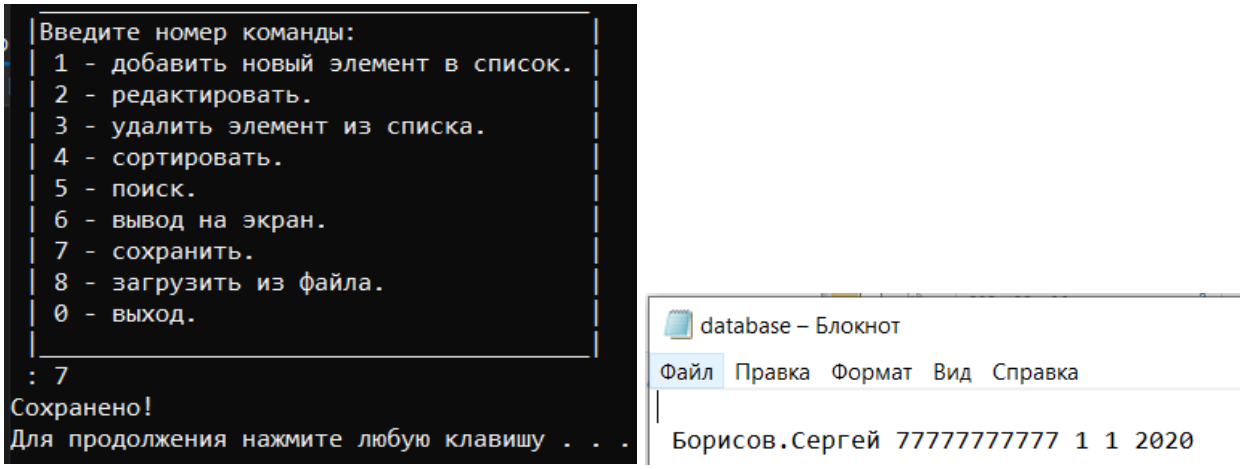


Рисунок 8.1 – Файл не существует

9 Загрузка БД контакты из файла:

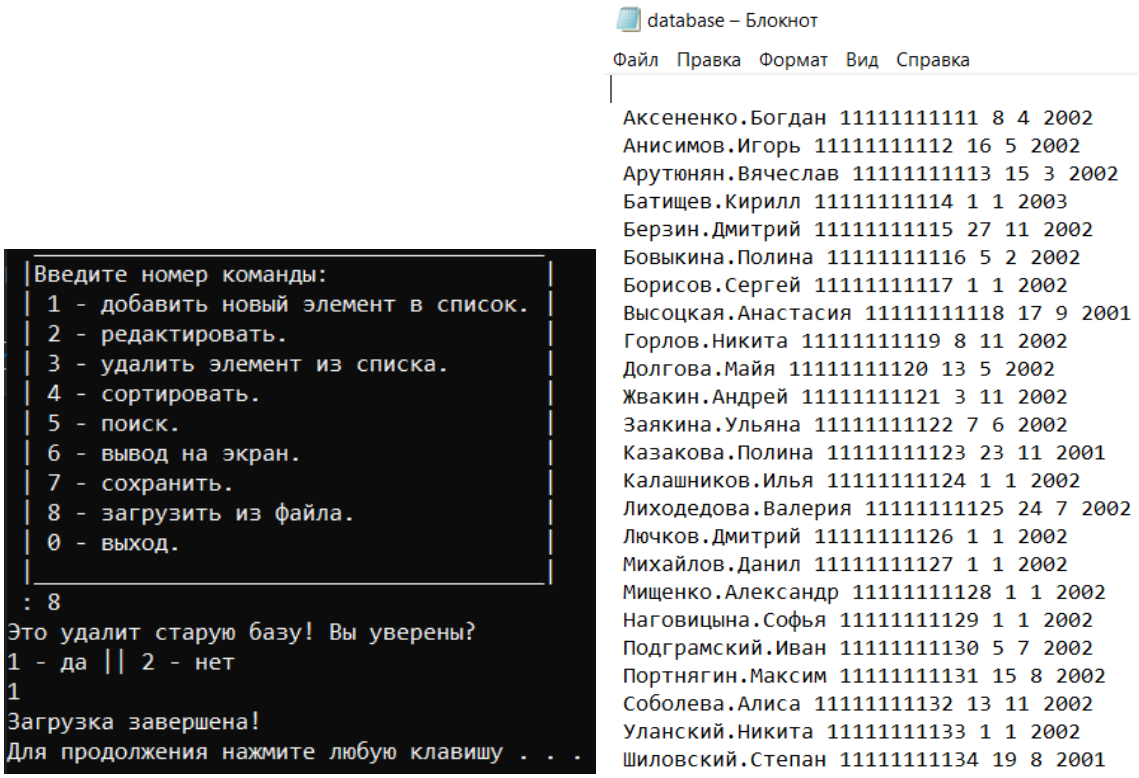


Рисунок 9.1 – Файл существует.

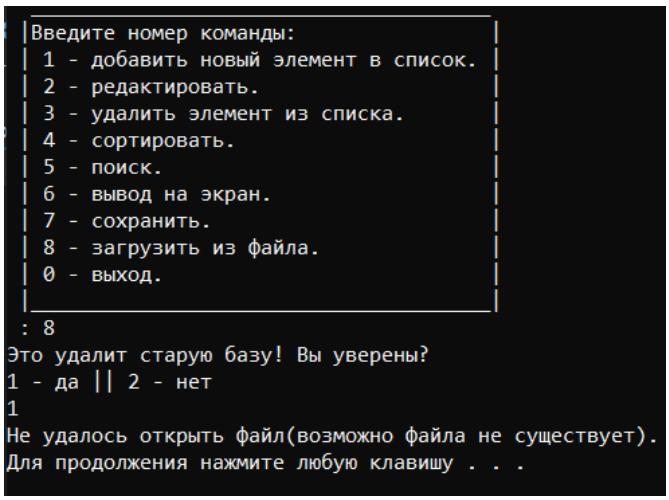


Рисунок 9.2 – В файле ничего нет или его не существует.

10 Выход из приложения:

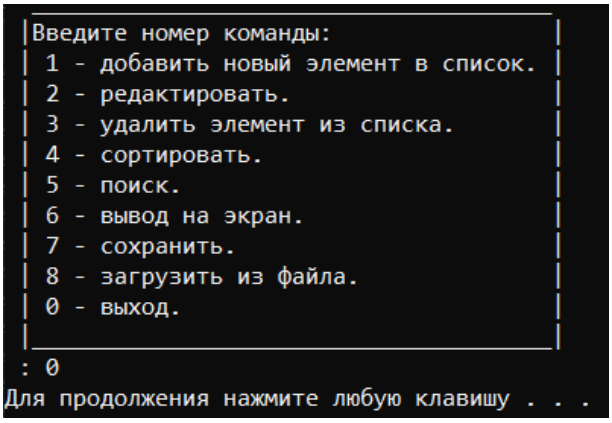


Рисунок 10 – Выход из приложения

5 ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта было разработано консольное приложение «Список контактов». Оно позволяет пользователю добавлять и удалять контакты в БД, отображать содержимое БД, а также осуществлять поиск контактов в БД по номеру телефона или имени, задаваемому пользователем.

Достоинства программы:

1. В приложении осуществляются различные проверки ввода данных.
2. Данное приложение простое и понятное в использовании.
3. Контакты могут быть записаны как латиницей, так и кириллицей.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Ключарев А.А., Матьяш В.А., Щекин С.В. Структуры и алгоритмы обработки данных: Учебное пособие / СПбГУАП. СПб., 2004.
2. *Колдаев В.Д.* Основы алгоритмизации и программирования: Учебное пособие / Колдаев В.Д; Под ред. проф.Л.Г. Гагариной - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2016. - 416 с.
3. *Павловская Т. А*. C/C++. Программирование на языке высокого уровня: учебник. СПб. : ПИТЕР, 2007. - 461 с

ПРИЛОЖЕНИЕ

Главный код.

```
#include <iostream>
#include <string>
#include <cctype>
#include <fstream>
#include <iomanip>
#include <windows.h>
#include <vector>
#include "InputAndCheck.h"
#include "List.h"
#include <algorithm>

using namespace std;


void littleprint(NOTE* tmp) // Вывод контакта
{
	if (tmp)
		cout << setw(25) << tmp->Name << setw(15) << tmp->Phone << " " << tmp-
>birthday[0] << "." << tmp->birthday[1] << "." << tmp->birthday[2] << endl;
}

string nameconvert(string name) // Преобразование имени при выводе из файла
{
	for (size_t i = 0; i < name.length(); i++)  // .length()-Возвращает длину строки name
		if (name[i] == ' ')
			name[i] = '.';
	return name;
}

string nameunconvert(string name) // Преобразование имени при вводе в файл
{
	for (size_t i = 0; i < name.length(); i++)
		if (name[i] == '.')
			name[i] = ' ';
	return name;
}

bool namecheck(string name) // Проверка на имя
{
	if (name.size() == 0)
	{
		return false;
	}
	int check = 0;
	for (size_t i = 0; i < name.length(); i++)
	{
```

```
			//
			if (int('a') <= int(name[i]) && int(name[i]) <= int('z') ||
				int('A') <= int(name[i]) && int(name[i]) <= int('Z') ||
				int('A') <= int(name[i]) && int(name[i]) <= int('Я') ||
				int('a') <= int(name[i]) && int(name[i]) <= int('я') ||
				int(name[i]) == int(' ') ||
				int(name[i]) == int('ё') ||
				int(name[i]) == int('Ё') ||
				int(name[i]) == int('-'))
			{
				//
			}
			else
			{
				cout << name[i] << " - Недопустимый символ!" <<endl;
				return false;
			}
		}
			return true;
}

bool phonecheck(string Phone) // Проверка номера телефона
{
	if (Phone.length() != 11)
		return false;
	for (size_t i = 0; i < Phone.length(); i++)
	{
		if (!iswdigit(Phone[i])) // проверка на соответствие знакам (от 0 до 9)
			return false;
	}
	return true;
}

bool phonesamecheck(ListNOTE*& list, string Phone)// Проверка номера телефона в списке
{
	if (list)
	{
		NOTE* tmp = list->Head;
		int count = 0;
		for (int i = 0; i < list->size; i++)
		{
			if (Phone == tmp->Phone)
				count++;
			tmp = tmp->next;
		}
		if (count < 1)
			return true;
		else
			return false;
	}
}
```

```
bool daycheck(int* BIRTHDAY) // Проверка на дату рождения (День[0] Месяц[1] Год[2])
{
	if (BIRTHDAY[0] < 1 || BIRTHDAY[0] > 31)
		return false;
	if ((BIRTHDAY[1] == 4 || BIRTHDAY[1] == 6 || BIRTHDAY[1] == 9 || BIRTHDAY[1]
== 11) && BIRTHDAY[0] > 30 && BIRTHDAY[0] < 1)
		return false;
	if (BIRTHDAY[1] == 2 && BIRTHDAY[0] > 29)
		return false;
	if (BIRTHDAY[1] == 2 && BIRTHDAY[0] == 29) // Проверка на високосный год
	{
		int temp = BIRTHDAY[2] - 1000;
		while (temp > 0)
			temp -= 4;
		if (temp != 0)
			return false;
	}
	return true;
}

string inputstring(int flag) // Ввод строки
{
	string str;
	getline(cin, str); // Ввод данных с клавиатуры и запись в строку

	if (flag == 1)
		while (!namecheck(str))
		{
			cout << "Некорректный ввод. Пожалуйста, повторите: " << endl;
			getline(cin, str);
		}
	if (flag == 2)
		while (!phonecheck(str))
		{
			cout << "Некорректный ввод. Пожалуйста, повторите: " << endl;
			getline(cin, str);
		}

	int i = str.size();
	while (str[i - 1] == ' ') // является ли символ слева пробелом
	{
		str[i - 1] = '\0'; // конец строки
		i--;
	}
	for (int j = 0; j < i;
++j)////////////////////////////////////////////////////////////////////////////////////////////////
	{
		while (str[j] == ' ' && str[j + 1] == ' ')
		{
			for (int k = unsigned(j); k < str.size() - 1; ++k)   // unsigned() - содержит
только положительные числа
```

```
                                str[k] = str[k + 1];                    // str.size() - возвращает
размер str
                        i--;
                }
        }
        if (str[0] == ' ')
        {
                for (int k = 0; k < str.size() - 1; ++k)
                        str[k] = str[k + 1];
                i--;
        }
        str.resize(i); //  изменяет размер строки в длинну из i символов  Если меньше -
удаляет
        return str;    //                                  Если больше - добавляет
}

bool ultimatecheck(NOTE*& tmp, ListNOTE*& list)  //  Окончательная проверка (для
вывода данных из файла)
{
        if (!namecheck(tmp->Name) || !phonecheck(tmp->Phone))
        {
                cout << "Номера или Именна контактов не соответсвуют формату" <<endl;
                return false;
        }
        if (!phonesamecheck(list, tmp->Phone))
        {
                cout << "Номера телефонов повторяются" << endl;
                return false;
        }//////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
        if (tmp->birthday[1] < 1 && tmp->birthday[1] > 12 || tmp->birthday[2] < 1900 && tmp-
>birthday[2] > 2021 || (!daycheck(tmp->birthday)))
        {
                cout << "Даты имеют не верный формат" << endl;
                return false;
        }
                return true;
}

void dobavlenie_elementa_v_spisok(ListNOTE*& list) // Добавление элемента в список
{
        NOTE* newya4eeka = new NOTE;

        cout << "Введите имя контакта:" << endl;
        string name = inputstring(1);
        newya4eeka->Name = name;

        cout << "Введите номер телефона (11 цифр)" << endl;
        string Phone = inputstring(2);

        while (!phonesamecheck(list, Phone))
        {
```

```
			cout << "Некорректный ввод(такой номер уже есть в базе). Повторите: " << endl;
			Phone = inputstring(2);
		}
		newya4eeka->Phone = Phone;

		cout << "Введите дату рождения.\nГод (От 1900 до 2021): ";
		newya4eeka->birthday[2] = inputint();

		while (newya4eeka->birthday[2] < 1900 || newya4eeka->birthday[2] > 2021)
		{
			cout << "Некорректный ввод. Повторите: ";
			newya4eeka->birthday[2] = inputint();
		}

		cout << "Месяц(числом от 1 до 12): ";
		newya4eeka->birthday[1] = inputint();

		while (newya4eeka->birthday[1] < 1 || newya4eeka->birthday[1] > 12)
		{
			cout << "Некорректный ввод. Повторите: ";
			newya4eeka->birthday[1] = inputint();
		}

		cout << "День(числом от 1 до корректного числа в введенном ранее месяце): ";
		newya4eeka->birthday[0] = inputint();

		while (!daycheck(newya4eeka->birthday))
		{
			cout << "Некорректный ввод. Повторите: ";
			newya4eeka->birthday[0] = inputint();
		}

		AddList(list, newya4eeka); // добавление списка
		cout << "Добавлено в список" << endl;
}

int nalichie_imen_v_spiske(ListNOTE*& list, string name) // Проверка на наличие имен в списке
{
		int count = 0;
		NOTE* tmp = list->Head;
		for (int i = 0; i < list->size; i++)
		{
			if (tmp->Name == name)
				count++;
			tmp = tmp->next;
		}
		return count;
}
```

```
int nalichie_nomera_telepona_v_spiske(ListNOTE*& list, string Phone) // Проверка на наличие номера тел в списке
{
	NOTE* tmp = list->Head;
	for (int i = 0; i < list->size; i++)
	{
		if (tmp->Phone == Phone)
			return 1;
		tmp = tmp->next;
	}
	return 0;
}

void redactirovanie_elementa_v_spiske(ListNOTE*& list) // редактирование елемента в списке

{
	cout << "Введите имя абонента или номер телефона \n : ";
	string NameOrPhone;
	string Name = "";
	string Phone = "";
	NOTE* tmp = new NOTE;
	getline(cin, NameOrPhone);
	cout << "Проверяю имя ли это?" << endl;
	while (!namecheck(NameOrPhone))
	{
		cout << "Проверяю номер ли это?" << endl;
		if (phonecheck(NameOrPhone))
		{
			cout << "Это номер! Ищу в базе: " << endl;
			Phone = NameOrPhone;
			break;
		}
		cout << "Некорректный ввод. Повторите: " << endl;
		getline(cin, NameOrPhone);
	}
	if (Phone == "") //если есть имя
	{
		Name = NameOrPhone;
		cout << "Это имя! Ищу в базе: " << endl;
		if (nalichie_imen_v_spiske(list, Name) < 1)
		{
			cout << "Не найдено: " << endl;
			return;
		}
		else
		{
			//если имя одно
			if (nalichie_imen_v_spiske(list, Name) == 1)
			{
				tmp = Search_by_Name(list, Name);
				littleprint(tmp);
```

```
                }
                //если имён много
                else
                {
                        cout << "Найдено несколько людей \n";
                        NOTE* tmp = list->Head;
                        int q = 0;
                        while (tmp)
                        {
                                if (tmp->Name == Name)
                                {
                                        cout << q << ' ';
                                        littleprint(tmp);
                                        q++;
                                }
                                tmp = tmp->next;
                        }
                        cout << "Введите номер телефона: \n";
                        getline(cin, Phone);
                        while (!phonecheck(Phone) || !Search_by_Phone(list, Phone))
                        {
                                cout << "Некорректный ввод. Повторите: " <<    endl;
                                getline(cin, Phone);
                        }
                }
        }
}

if (Phone != "") //если есть телефон
{
        if (!nalichie_nomera_telepona_v_spiske(list, Phone))
        {
                cout << "Не найдено: " << endl;
                        return;
        }
        else
        {
                tmp = Search_by_Phone(list, Phone);
                cout << "Найдено " << endl;
                littleprint(tmp);
        }
}

        cout << "  _____" << endl;
        cout << " |Введите номер команды:          |" << endl;
        cout << " | 1 Редактирования имени абонента |" << endl;
        cout << " | 2 Редактирование номера телефона|" << endl;
        cout << " | 3 Редактирование даты рождения  |" << endl;
        cout << " |_____|" << endl;
        cout << " |Другая цифра - выход            |" << endl;
        cout << " |_____|" << endl;
        cout << " :";
```

```
		int menu = inputint();

			switch (menu)
			{
			case 0:
				break;
			case 1:
			{
				cout << "Введите имя абонента: " << endl;
				string nam2 = inputstring(1);
				while (!namecheck(nam2))
				{
					cout << "Некорректный ввод. Повторите: " << endl;
					string nam2 = inputstring(1);
				}
				tmp->Name = nam2;
				cout << "Отредактированно" << endl;
				break;
			}
			case 2:
			{
				cout << "Введите номер телефона " << endl;
				string pho2 = inputstring(2);
				while (!phonesamecheck(list, pho2))
				{
					cout << "Этот номер уже существует. Повторите: " <<
endl;
					pho2 = inputstring(2);
				}
				tmp->Phone = pho2;
				cout << "Отредактированно" << endl;
				break;
			}
			case 3:
			{
				cout << "Введите дату рождения.\nГод (от 1900 до 2021): ";
				tmp->birthday[2] = inputint();
				while (tmp->birthday[2] < 1900 || tmp->birthday[2] > 2021)
				{
					cout << "Некорректный ввод. Повторите: ";
					tmp->birthday[2] = inputint();
				}
				cout << "Месяц(числом от 1 до 12): ";
				tmp->birthday[1] = inputint();
				while (tmp->birthday[1] < 1 || tmp->birthday[1] > 12)
				{
					cout << "Некорректный ввод. Повторите: ";
					tmp->birthday[1] = inputint();
				}
				cout << "День(числом от 1 до корректного числа в введенном
ранее месяце): ";
```

```
                        tmp->birthday[0] = inputint();
                        while (!daycheck(tmp->birthday))
                        {
                                cout << "Некорректный ввод. Повторите: ";
                                tmp->birthday[0] = inputint();
                        }
                        cout << "Отредактированно" << endl;
                }
                }
}

void notedelete(ListNOTE*& list) // Удаление контакта по данным
{
        cout << "Введите имя абонента или номер телефона " << endl;
        string NameOrPhone;
        string Phone;
        string name;
        getline(cin, NameOrPhone);
        while (!namecheck(NameOrPhone))
        {
                cout << "Проверяю номер ли это?" << endl;
                if (phonecheck(NameOrPhone))
                {
                        cout << "Это номер! Ищу в базе: " << endl;//////////добавил Удалено
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                        Phone = NameOrPhone;
                        NOTE* tmp = Search_by_Phone(list, Phone);
                        if (tmp)
                        {
                                littleprint(tmp);
                                cout << "Удалено" << endl;
                                Delete(list, tmp);
                        }
                        return;
                }
                getline(cin, NameOrPhone);
        }
        name = NameOrPhone;
        if (nalichie_imen_v_spiske(list, name) > 0)
        {
                if (nalichie_imen_v_spiske(list, name) > 1)
                {
                        cout << "Абонентов с таким именем найдено несколько. Введите
номер телефона(телефон - уникален) : " << std::endl;
                                Search_by_NameALL(list, name);
                        string Phone;
                        getline(cin, Phone);
                        while (!phonecheck(Phone))
                        {
                                cout << "Некорректный ввод. Повторите: " << endl;
                                getline(cin, Phone);
                        }
```

```
                Delete(list, Search_by_Phone(list, Phone));
                return;
            }
            else
            {

                cout << "Удалено" << endl;
                Delete(list, Search_by_Name(list, name));
                return;
            }
        }
        else
        {
            cout << "Этого имени абонента нет." << endl;
        }
}

void fileout(ListNOTE*& list) // сохранить в файл
{
        ofstream fout("database.txt");//////////////////////////////////////////////////////////

            NOTE * tmp = list->Head;
        while (tmp)
        {
            string name = nameconvert(tmp->Name);
            fout << '\n';
            fout << ' ' << name;
            fout << ' ' << tmp->Phone;
            fout << ' ' << tmp->birthday[0];
            fout << ' ' << tmp->birthday[1];
            fout << ' ' << tmp->birthday[2];
            tmp = tmp->next;
        }
        fout.close();
}

void filein(ListNOTE*& list) // загрузить из файла
{
        ifstream fin("database.txt");
        if (!fin.is_open())
        {
            cout << "Не удалось открыть файл(возможно файла не существует)." << endl;
        }
        else
        {
            bool flag = true;
            if (list)
                list = NULL;
            while (!fin.eof())  // файл не пуст
            {
                NOTE* tmp = new NOTE;
                string name;
```

```
                        fin >> name >> tmp->Phone >> tmp->birthday[0] >> tmp->birthday[1] >>
tmp -> birthday[2];
                        tmp->Name = nameunconvert(name);
                        if (!ultimatecheck(tmp, list) || tmp->Name == "")
                        {
                                cout << "Файл не коректен." << endl;
                                list = NULL;
                                flag = false; break;
                        }
                        AddList(list, tmp);
                }
                if (flag)
                {
                        cout << "Загрузка завершена!" << endl;
                }
                fin.close();
        }
}

int comp_name(NOTE* num1, NOTE* num2)
{
        if (num1->Name < num2->Name)
                return true;
        else
                return false;
}

ListNOTE* data_sort_name(ListNOTE*& list)
{
        cout << "Сортирую" << endl;
        NOTE* tmp = list->Head;
        NOTE* newtmp = list->Head;
        ListNOTE* newlist = NULL;

        vector<NOTE*> v;
        while (tmp)
        {
                        v.push_back(tmp);               // Добавление в конец вектора
                tmp = tmp->next;
        }

        sort(begin(v), end(v), comp_name);      // сортировка для comp_name.
                                // begin(v)-возвращает итератор, который указывает на
первый элемент в "v".
        for (int i = 0; i < v.size(); i++)      // end(v)-возвращает итератор, который указывает на
конец "v".
        {                       // v.size()- возращает размер длины вектора "v".
                littleprint(v[i]);
        }

        cout << endl;
        cout << "Сортировка окончена!" << endl;
```

```
	cout << "Сохранить отсортированный вид?" << endl;
	cout << "1 - да сохранить || другое - нет " << endl;
	int x = inputint();
	if (x == 1)
	{
		for (int i = 0; i < v.size(); i++)
		{
			v[i]->next = NULL;
			AddList(newlist, v[i]);
		}
		Show(newlist);
		return newlist;
	}
	else
	{
		return list;
	}

}

int comp_phone(NOTE* num1, NOTE* num2)
{
	if (num1->Phone > num2->Phone)
		return true;
	else
		return false;
}

ListNOTE* data_sort_phone(ListNOTE*& list)
{
	cout << "Сортирую по убыванию" << endl;
	NOTE* tmp = list->Head;
	NOTE* newtmp = list->Head;
	ListNOTE* newlist = NULL;

	 vector<NOTE*> v;
	while (tmp)
	{
		v.push_back(tmp);
		tmp = tmp->next;
	}

	sort(begin(v), end(v), comp_phone);     // сортировка(в обратном направлении) для
comp_name.
                                                                        // begin(v)-
возвращает итератор, который указывает на первый элемент в "v".
	for (int i = 0; i < v.size(); i++)      // end(v)-возвращает итератор, который указывает
на конец "v".
	{                                  // v.size()- возращает размер длины вектора "v".
		littleprint(v[i]);
	}
```

```
	cout << endl;
	cout << "Сортировка окончена!" << endl;
	cout << "Сохранить отсортированный вид?" << endl;
	cout << "1 - да сохранить || другое - нет " << endl;
	int x = inputint();
	if (x == 1)
	{
		for (int i = 0; i < v.size(); i++)
		{
			AddList(newlist, v[i]);
		}
		return newlist;
	}
	else
	{
		return list;
	}
}

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	setlocale(LC_CTYPE, "rus");
	system("color 0F");

	ListNOTE* list = NULL;

	int menu;
	menu = 1;
	while (menu)
	{
		if (list == NULL)
		{
			cout << endl;
			cout << "  _____ " << endl;
			cout << " |Введите номер команды:          |\n";
			cout << " | 1 добавить новый элемент в список. |\n";
			cout << " | 8 загрузить из файла database.     |\n";
			cout << " | 0 выход.                           |\n";
			cout << " |_____| " << endl;
			cout << " : ";
		}
		else
		{
			system("pause");
			system("cls");
			cout << "  _____" << endl;
			cout << " |Введите номер команды:          |\n";
			cout << " | 1 - добавить новый элемент в список. |\n";
			cout << " | 2 - редактировать.                 |\n";
```

```
        cout << " | 3 - удалить элемент из списка.       |\n";
        cout << " | 4 - сортировать.                     |\n";
        cout << " | 5 - поиск.                           |\n";
        cout << " | 6 - вывод на экран.                  |\n";
        cout << " | 7 - сохранить.                       |\n";
        cout << " | 8 - загрузить из файла.              |\n";
        cout << " | 0 - выход.                           |\n";
        cout << " |_____________________________________| " << endl;
        cout << " : ";
}
menu = inputint();

if (menu == 0)
        break;

else if (menu == 1)
        dobavlenie_elementa_v_spisok(list);

else if (menu == 2 && list)
{
        Show(list);
        redactirovanie_elementa_v_spiske(list);
}

else if (menu == 3 && list)
{
        if (list)
        {
                Show(list);
                cout << "_____________________________ "<< endl;
                cout << "|Введите номер команды:      |" << endl;
                cout << "| 1 - Удалить по номеру в списке.|" << endl;
                cout << "| 2 - Удалить по данным.     |" << endl;
                cout << "|____________________________|" << endl;
                cout << "|Другая цифра - выход        |" << endl;
                cout << "|____________________________|" << endl;
                int choice = inputint();

                if (choice == 1)
                {
                        cout << "введите номер в списке" << endl;
                        int num = 0;
                        while (num == 0 || num > list->size)
                        {
                                cout << "Повторите ввод: ";
                                num = inputint();
                        }
                        NOTE* tmp = list->Head;
                        while (tmp->num != num)
                        {
                                tmp = tmp->next;
                        }
```

```
                littleprint(tmp);
                cout << "Удалено" << endl;
                Delete(list, tmp);
            }
            else if (choice == 2)
            {
                notedelete(list);
            }

        }
    }
    else if (menu == 4 && list)
    {
        cout << "Сортировка:" << endl;
        cout << " _____ " << endl;
        cout << "|Введите номер команды:   |" << endl;
        cout << "| 1 - по имени.           |" << endl;
        cout << "| 2 - по номеру телефона. |" << endl;
        cout << "|_____|" << endl;
        cout << "|Другая цифра - выход     |" << endl;
        cout << "|_____|" << endl;
        int choice = inputint();
        if (choice == 1)
        {
            Show(list);
            list = data_sort_name(list);
        }
        else if (choice == 2)
        {
            Show(list);
            list = data_sort_phone(list);
        }
    }

    else if (menu == 5 && list)
    {
        Show(list);
        cout << "Поиск:" << endl;
        cout << " _____ " << endl;
        cout << "|Введите номер команды:   |" << endl;
        cout << "| 1 - по номеру телефона. |" << endl;
        cout << "| 2 - по имени.           |" << endl;
        cout << "|_____|" << endl;
        cout << "|Другая цифра - выход     |" << endl;
        cout << "|_____|" << endl;
        int choice = inputint();
        if (choice == 1)
        {
            cout << "(для отмены поиска введите 0)" << endl;
            cout << "Введите телефон: \n";
            string Phone;
            getline(cin, Phone);
```

```
                while (!phonecheck(Phone) || phonesamecheck(list, Phone))
                {
                        if (Phone == "0")
                                break;

                                cout << "Телефон не найден или
некорректен!Проверьте правильность ввода!" << endl;
                                cout << "(для выхода введите 0)" << endl;
                        getline(cin, Phone);
                }
                if (Phone != "0")
                {
                        NOTE* tmp = Search_by_Phone(list, Phone);
                        littleprint(tmp);
                }
        }
        else if (choice == 2)
        {
                cout << "Введите имя: " << endl;
                string name;
                getline(cin, name);
                while (!namecheck(name))
                {
                        cout << "Некорректный ввод. Повторите: " << endl;
                        getline(cin, name);
                }
                if (nalichie_imen_v_spiske(list, name) < 1) ///
                {
                        cout << "Данного контакта НЕТ! \n";
                }
                //если имён много
                else
                {
                        if (nalichie_imen_v_spiske(list, name) == 1) ///
                        {
                                littleprint(Search_by_Name(list, name));
                        }
                        else
                        {
                                cout << "Найдено несколько людей \n";
                                NOTE* tmp = list->Head;
                                int q = 0;
                                while (tmp)
                                {
                                        if (tmp->Name == name)
                                        {
                                                cout << q << ' ';
                                                littleprint(tmp);
                                                q++;
                                        }
                                        tmp = tmp->next;
                                }
```

```
                                        cout << "Введите телефон \n";
                                        string Phone;
                                        getline(cin, Phone);
                                        while (!phonecheck(Phone))
                                        {
                                                getline(cin, Phone);
                                                cout << "Некорректный ввод. Повторите: "
<< endl;
                                        }
                                        tmp = Search_by_Phone(list, Phone);
                                        littleprint(tmp);
                                }
                        }
                }
        }
        else if (menu == 6 && list)
                Show(list);
        else if (menu == 7 && list)
        {
                if (list)
                        fileout(list);
                cout << "Сохранено!" << endl;
        }
        else if (menu == 8)
    {
                int choice = 1;
                if (list)
                {
                        cout << "Это удалит старую базу! Вы уверены?" << endl;
                        cout << "1 - да || 2 - нет\n";
                        choice = inputint();
                }
                if (choice == 1)
                {
                        filein(list);
                }
                else if (choice == 2);
        }
        else
                cout << "Такого пункта нет в меню" << endl;
      }
      system("pause");
}
```

Модуль со структурой ячейки и списка. (List.h)

```
using namespace std;
//структура ячейки
struct NOTE
{
	int num;
	string Name;
	string Phone;
	int birthday[3];
	NOTE* next;
};
struct ListNOTE
{
	NOTE* Head; //Первый элемент
	NOTE* Tail; // последний
	int size;
};

void AddList(ListNOTE*& list, NOTE*& ya4eeka);
void Show(ListNOTE* list);
void Delete(ListNOTE*& list, NOTE* ya4eeka);
NOTE* Search_by_Phone(ListNOTE*& list, string Phone);
NOTE* Search_by_Name(ListNOTE*& list, string Name);
void Search_by_NameALL(ListNOTE*& list, string Name);
```

Модуль с функцией проверки на целочисленный формат. (inputAndCheck.h)

```
#include <iostream>
using namespace std;

int inputint()//проверка на целочисленный формат
{
	int x; int i = 0;
	cin >> x;
	while (cin.fail() || cin.get() != '\n')
	{
		cin.clear();
		cin.ignore(cin.rdbuf()->in_avail());
		cout << "Повторите ввод: ";
		cin >> x;
	}
	return x;
}
```

Модуль с функциями. (List1.cpp)

```
#include <iostream>
#include <string>
#include <iomanip>
#include "List.h"
using namespace std;
void AddList(ListNOTE*& list, NOTE*& ya4eeka) // Добавление ячейки в список
{
	if (ya4eeka != NULL)
	{
		NOTE* newelem = ya4eeka; //Выделение памяти для нового элемента списка
		if (!list)
		{
			list = new ListNOTE;
			list->Head = newelem;
			list->Tail = newelem;
			list->Head->next = NULL;
			list->size = 1; //При каждом добавлении элемента увеличиваем число
элементов в списке
				return;
		}

		NOTE* temp = list->Head; //Выделение памяти для нового элемента списка
		while (temp->next)
		{
			temp = temp->next;
		}
		temp->next = newelem;
		list->Tail = newelem;
		list->Tail->next = NULL;
		list->size++; //При каждом добавлении элемента увеличиваем число
элементов в списке
	}
}
void numbering(ListNOTE*& list) //Нумерация в консоли
{
	NOTE* tmp = list->Head; //Указываем на голову
	int i = 1;
	while (tmp) //Пока не выполнен признак прохода по всему списку
	{
		tmp->num = i;
		i++;
		tmp = tmp->next; //Указываем, что нужен следующий элемент
	}
}
void Show(ListNOTE* list) // Демонстрация списка
{
	if (list->size < 1)
	{
		cout << "Массив пуст" << endl;
		return;
```

```
	}
	numbering(list);
	NOTE* tempHead = list->Head; //Указываем на голову
	while (tempHead) //Пока не выполнен признак прохода по всему списку
	{
		cout << tempHead->num << " " << tempHead->Name << setw(15);
		cout << tempHead->Phone << " ";
		cout << tempHead->birthday[0] << "." << tempHead->birthday[1] << "."
<<tempHead->birthday[2] << endl; // день мес год

		tempHead = tempHead->next; //Указываем, что нужен следующий элемент
	}
	cout << "\n";
}
void Delete(ListNOTE*& list, NOTE* ya4eeka) //Удаление элемента
списка.....................................................................................
{
	if (ya4eeka != NULL)
	{
		NOTE* tmp = list->Head;
		if (ya4eeka == list->Head)
		{
			list->Head = list->Head->next;
			list->size--;
			return;
		}
		while (tmp->next != ya4eeka)
		{
			tmp = tmp->next; //Указываем, что нужен следующий элемент
		}
		tmp->next = tmp->next->next;
		list->size--;
	}
}
NOTE* Search_by_Phone(ListNOTE*& list, string Phone) // Поиск по номеру
{
	NOTE* tmp = list->Head; //Указываем на голову
	while (tmp) //Пока не выполнен признак прохода по всему списку
	{
		if (tmp->Phone == Phone)
		{
			return tmp;
		}
		tmp = tmp->next; //Указываем, что нужен следующий элемент
	}
	cout << " не найдено " << endl;
	return NULL;
}
NOTE* Search_by_Name(ListNOTE*& list, string Name) // Поиск по имени
{
	NOTE* tmp = list->Head; //Указываем на голову
	while (tmp) //Пока не выполнен признак прохода по всему списку
```

```
	{
		if (tmp->Name == Name)
		{
			return tmp;
		}
		tmp = tmp->next; //Указываем, что нужен следующий элемент
	}
	cout << " не найдено " << endl;
	return NULL;
}
void Search_by_NameALL(ListNOTE*& list, string Name) // Если в списке несколько похожих имен, то фу-ция выводит их
{
	NOTE* tmp = list->Head; //Указываем на голову
	while (tmp) //Пока не выполнен признак прохода по всему списку
	{
		if (tmp->Name == Name)
		{
			cout << setw(25) << tmp->Name << setw(15) << tmp -> Phone << " " << tmp->birthday[0] << "." << tmp->birthday[1] << "." << tmp->birthday[2] << endl;
		}
		tmp = tmp->next; //Указываем, что нужен следующий элемент
	}
}
```