

Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему «База данных Контакты»

ОПР.
22.12.22


Выполнил:

студент группы 21BO2

Кокорин А.А.

Принял:

Митрохин М.А.

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра «Вычислительная техника»

«УТВЕРЖДАЮ»

Зав. кафедрой ВТ

Митрохин М. А.

«___» _____ 2022 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Кокорину Артёму Александровичу

Группа 21ВО2

Тема проекта: Разработка программы сложной структуры методом нисходящего программирования

Исходные данные (технические требования) на проектирование

Наименование программы: База данных «Контакты».

База данных включает следующие поля:

№ Имя Фамилия Номер телефона

Задание. Программа должна выполнять следующие функции:

- Создание новой базы данных.
- Добавление/удаление контактов, их редактирование.
- Поиск , сортировка, обмен местами контактов.
- Сохранение базы данных в файл и удаление этого файла.
- Загрузка ранее сохраненной базы данных из файла.
- Аллокатор памяти

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных — списков, структур, файлов.
3. Режим работы видеосистемы — текстовый/графический.
4. Устройство ввода информации — клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система — Windows.
8. Система программирования — C/C++ на базе среды Microsoft Visual Studio 2019.
9. Язык программирования — Си.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Структура модулей программы, схема алгоритма программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

1. В соответствии с графиком выполнения курсового проекта.

2. _____

3. _____

Дата выдачи задания «1» _____ октября 2022 г

Дата защиты проекта «27» декабря 2022 г

Руководитель _____ Митрохин М. А.

Задание получил _____ «1» _____ октября 2022 г

Студент Кокорин Кокорин Артём Александрович

Содержание

Введение	4
1 Постановка задачи.....	5
2 Выбор решения.....	7
2.1 Определение необходимых модулей программы.....	7
3 Описание разработки программы	10
4 Отладка и тестирование	12
5 Описание программы	13
6 Руководство пользователя	17
Заключение	22
Список используемых источников.....	23
Приложение А	24
Приложение Б.....	32

Введение

С развитием технологий компьютеры все больше и больше вливаются в повседневную жизнь человека. Персональные компьютеры - это уже не просто машины для математических вычислений, их общедоступность значительно расширила область применения, функционал, возможности и обработки запросов.

Базы данных значительно изменились с момента их появления в начале 1960-х годов. Исходными системами, которые использовались для хранения и обработки данных, были навигационные базы данных – например, иерархические базы данных (которые опирались на древовидную модель и допускали только отношение «один-ко-многим») и базы данных с сетевой структурой (более гибкая модель, допускающая множественные отношения). Несмотря на простоту, эти ранние системы были негибкими. В 1980-х годах стали популярными реляционные базы данных, в 1990-х годах за ними последовали объектно-ориентированные базы данных. Совсем недавно вследствие роста Интернета и возникновения необходимости анализа неструктурированных данных и многих других.

В настоящее время облачные базы данных и автономные базы данных открывают новые возможности в отношении способов сбора, хранения, использования данных и управления ими. Скорость — ещё одно преимущество базы данных. База данных устроена так, что она легко и быстро находит, записывает, переписывает и снова находит данные. Всё потому, что СУБД (Система управления базами данных) всегда знает, что где лежит и по какому критерию искать. Там не будет случайных данных в случайном месте.

Скорость важна ещё и потому, что СУБД обычно обслуживает сразу много потоков: одновременно ей могут пользоваться десятки и сотни тысяч человек, поэтому ей некогда копаться. В хорошо сделанных БД всё молниеносно.

Сложность. Базы данных нужны в числе прочего для хранения сложно структурированных данных. Мы привыкли думать, что база данных — это такая таблица, где есть строки и столбцы. Но база данных при правильной организации может намного больше

В нашем случае, База данных контактов - это программа, которая позволяет хранить, управлять и обрабатывать информацию о контактах. Такая база данных может быть полезна для любого человека, который хочет организовать свои контакты и иметь возможность быстро найти интересующую информацию.

1 Постановка задачи

Задачи для решения данной программы приведены в виде списка:

1. Необходимо разработать программу, которая позволит хранить информацию о контактах (имя, фамилия, номер телефона), а также управлять ими. Программа должна предоставить пользователю удобную систему базы данных "Контакты".
Создать структуру содержащую строковые (char[]) поля "фамилия", "имя", "номер телефона".
2. В программе, в функции main, создать массив "контактов".
3. Для этого массива реализовать функции добавления и удаления контакта, учитывающие невозможность добавления контакта при заполненном массиве и невозможность удаления при пустом массиве, а также функцию вывода всех контактов в stdout.
4. Оформить меню для добавления, удаления и вывода контактов.
5. В коде, отвечающем за добавление контакта, реализовать ввод, поддерживающий строки с пробелами, и "защиту от дурака".
6. Добавить пункты меню "сохранение в файл", "загрузка из файла".
7. Имя файла задается как аргумент командной строки (argv[1]).
8. Реализовать проверку наличия аргументов командной строки.
9. Реализовать обработку ошибок, связанных с работой с файлами.
10. Все ошибки выводить в стандартный поток вывода ошибок (stderr).
11. Каждой функции должен соответствовать заголовочный файл и файл с реализацией.
12. Все заголовочные файлы должны быть защищены от повторного включения.
13. В любой из функций, где есть обработка ошибок, реализовать вывод сообщений об ошибках посредством макроса, определенного самостоятельно (__FILE__, __LINE__, __func__).
14. Контакты хранить в односвязном списке (один узел содержит в себе один контакт).
15. Реализовать функции для работы с односвязным списком.

16.Реализовать дополнительные функции программы (Изменение Контакта, Сортировка по номеру телефона, Обмен местами, Поиск контактов)

Также должно быть предоставлено меню, в котором должны быть упорядочены и предельно понятны базовые функции программы, а именно:

- 1 Для добавления контактов
- 2 Для удаления контактов
- 3 Для вывода контактов
- 4 Для сохранения контактов в файл
- 5 Для загрузки контактов из файла
- 6 Предоставление типичного Аллокатора памяти
- 7 Изменение Контакта
- 8 Сортировка по номеру телефона
- 9 Обмен местами
- 10 Поиск
- 0 Для выхода из программы

2 Выбор решения

2.1 Определение необходимых модулей программы

Для решения данной задачи было выбрано использование списка, реализованного через структуру. Этот вариант позволит удобно управлять информацией, создавать, редактировать и удалять контакты. Также были созданы Заголовки для каждого из модуля:

1. add_c.c - add_c.h
2. change.c - change.h
3. alloc.c - alloc.h
4. delete_c.c - delete_c.h
5. find.c - find.h
6. main.c - main.h
7. pos.c - pos.h
8. printc_c.c - printc_c.h
9. print_f.c - print_f.h
10. scan_err.c - scan_err.h
11. scan_f.c - scan_f.h
12. sort.c - sort.h

Заголовки в Си нужны для того, чтобы объявить функции и другие сущности, которые описаны в другом файле. Это нужно, чтобы компилятор знал, что такие функции и другие сущности существуют и что они будут определены в дальнейшем. Обычно заголовки находятся в файлах с расширением .h

Модуль add_c.c содержит функции для добавления контакта в список. Они включают в себя start_push, push, pop, free_node, а также функцию add_c, которая вызывает остальные функции и управляет процессом добавления контакта. Эти функции являются важным модулем программы, так как позволяют пользователю добавлять новые контакты в список, управляя процессом добавления.

Модуль "alloc" предназначен для тестирования различных функций управления памятью в Си. Он содержит 5 тестов, каждый из которых иллюстрирует работу различных функций:

Тест 1 представляет собой обычное успешное выделение памяти с помощью функции malloc.

Тест 2 демонстрирует освобождение одного блока из нескольких выделенных с помощью функции free.

Тест 3 показывает освобождение двух блоков из нескольких выделенных с помощью функции free.

Тест 4 иллюстрирует ситуацию, когда память закончилась, и новый регион памяти расширяет старый с помощью функции malloc.

Тест 5 представляет собой пример работы функции realloc, которая изменяет размер уже выделенного региона памяти.

Модуль "change" отвечает за изменение контакта в базе данных. Он принимает указатель на структуру типа "contact", которая является началом списка контактов, а также размер списка. Пользователь вводит номер контакта, который хочет изменить. Если введенный номер находится вне диапазона контактов, программа выводит сообщение об ошибке. В противном случае, программа находит нужный контакт в списке, считывает новые значения имени, фамилии и номера телефона с клавиатуры и заменяет старые значения этими новыми. После этого программа выводит сообщение об успешном изменении контакта.

Модуль delete_с предназначен для удаления контакта из телефонной книги. Пользователь вводит порядковый номер контакта, который хочет удалить, и если такой контакт существует, он удаляется из телефонной книги. Если контакт, который хочет удалить пользователь, является первым в списке, то указатель на начало списка смещается на следующий элемент.

find_с модуль реализует функцию поиска контакта в телефонной книге по имени. При вызове функции пользователь вводит имя контакта, которое он хочет найти. Затем происходит перебор всех контактов в телефонной книге и сравнение их имен с введенным пользователем именем. Если имя совпадает, то информация о контакте выводится на экран. После завершения перебора всех контактов программа ожидает ввода пользователем команды для продолжения работы или выхода в меню.

main модуль является основным модулем программы, в котором определена структура контакта и реализован меню, позволяющее выбирать различные операции с контактами. Операции, которые можно выполнять с контактами, включают в себя добавление, удаление, вывод, сохранение в файл, загрузку из файла, сортировку, обмен местами, поиск и изменение контакта. В меню также предусмотрена операция выхода из программы. При выборе операции из меню вызывается соответствующая функция, реализующая данную операцию. В меню также предусмотрена операция вызова аллокатора, который выводит информацию о распределении памяти в системе.

ros модуль обеспечивает возможность обмена местами двух контактов в телефонной книге. Пользователь вводит номера контактов, которые нужно обменять местами, и модуль меняет их местами, перезаписывая информацию о них.

print_с отвечает за вывод контактов на экран. Он принимает указатель на структуру "contact", которая содержит данные контакта, и указатель на переменную "size_с", которая содержит количество контактов. Модуль выводит на экран заголовок таблицы с информацией о номере, имени, фамилии и номере телефона контакта, а также информацию о каждом контакте в отдельной строке таблицы. Если контакты отсутствуют, выводится сообщение об этом. После вывода контактов на экран программа ожидает нажатия клавиши для продолжения работы

print_f модуль сохраняет информацию о контактах в файл. Он ожидает, что имя файла было передано в качестве аргумента командной строки. Он открывает файл для записи, проверяет, что файл успешно открыт, и записывает информацию о каждом контакте в файл. Если контакты отсутствуют, модуль выводит сообщение об ошибке. После записи в файл он закрывает файл.

`scan_err` модуль сканирует строку `s` и проверяет, содержит ли она символы, которые указаны в массиве `ERR`. Если строка `s` содержит хотя бы один символ из массива `ERR`, то функция выводит сообщение об ошибке и устанавливает значение переменной `f` равным 0. Если строка `s` не содержит символов из массива `ERR`, то функция устанавливает значение переменной `f` равным 1. Функция также читает и очищает буфер ввода, чтобы избежать последующих проблем с вводом данных.

`scan_f` отвечает за загрузку данных контактов из файла в память программы. Он принимает указатель на файл, указатели на начало и конец списка контактов, размер списка контактов, флаг, который указывает на наличие контактов в списке, а также количество аргументов и массив указателей на строки с аргументами. Модуль открывает файл с именем, заданным в качестве аргумента, и считывает из него данные о контактах. Если файл не был открыт, выводит сообщение об ошибке. Затем он проверяет, были ли ранее добавлены контакты в список. Если список пуст, то модуль создает новый узел, который станет началом списка.

`Sort` модуль сортирует контакты в телефонной книге по номеру телефона. Он использует алгоритм сортировки пузырьком. Он перебирает все элементы контактов и сравнивает их номера телефона с текущим элементом, если находит элемент с меньшим номером телефона, то он меняет их местами. После сортировки контакты отображаются в порядке возрастания номера телефона.

3 Описание разработки программы

Для написания программы был использован язык программирования C. В качестве среды разработки была использована программа Microsoft Visual Studio 2019.

Для начала было разработан план действий для создания базы данных , создать структуру содержащую строковые поля "фамилия", "имя", "номер телефона". Создать массив "контактов". Для этого массива реализовать функции добавления и удаления контакта, учитывающие невозможность добавления контакта при заполненном массиве и невозможность удаления при пустом массиве, а также функцию вывода всех контактов . Оформить меню для добавления, удаления и вывода контактов.

Разработать модуль, который содержит функции для добавления, удаления, изменения и просмотра информации о контактах, с помощью которых пользователь может управлять списком контактов в базе данных. Также добавить реализующую функцию изменения контакта в базе данных, принимая указатель на список контактов, размер списка и номер контакта, который требуется изменить, и считывая новые значения имени, фамилии и номера телефона с клавиатуры. Операции, которые можно выполнять с контактами, включают в себя добавление, удаление, вывод, сохранение в файл, загрузку из файла, сортировку, обмен местами, поиск и изменение контакта. Обеспечивать возможность обмена местами двух контактов в телефонной книге, сканирования, сортировку, загрузку и выгрузку контактов.

Иерархическая структура программы представлена на рисунке 1.

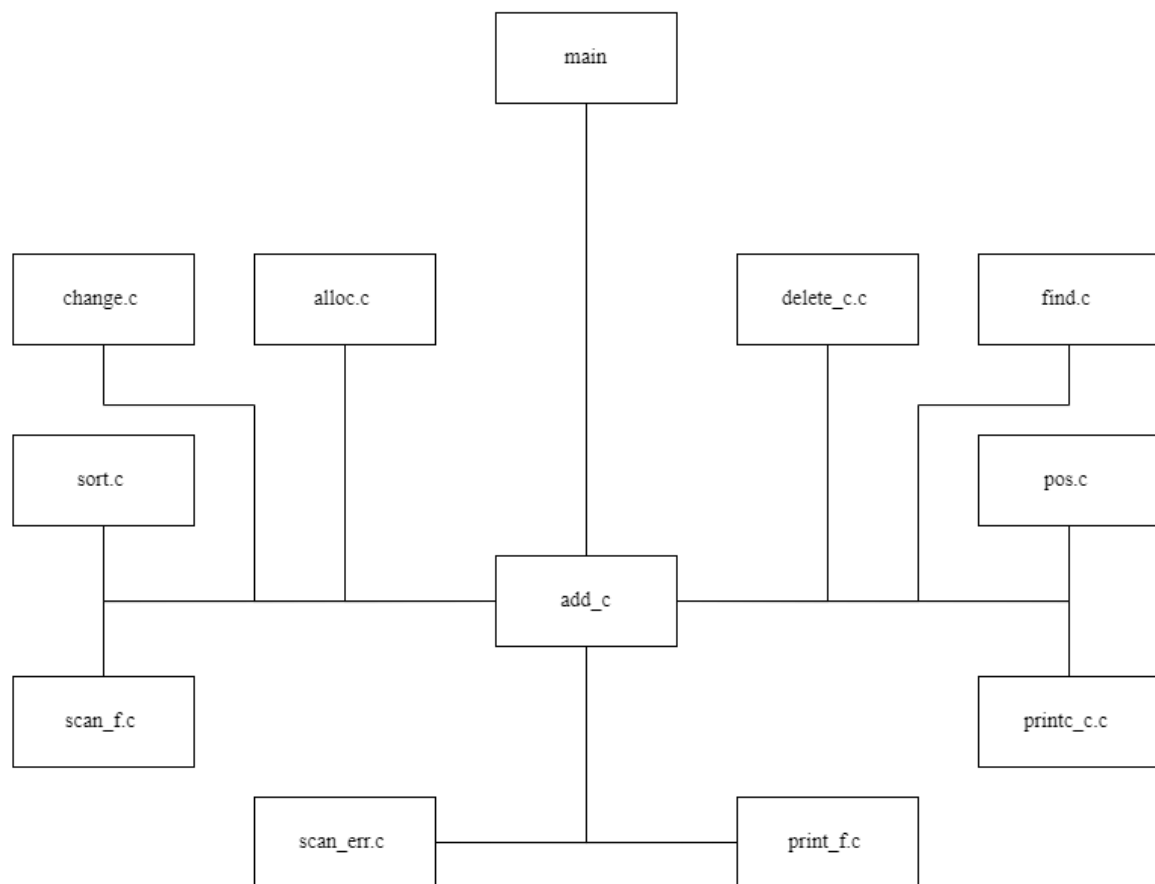


Рисунок 1- Иерархическая структур

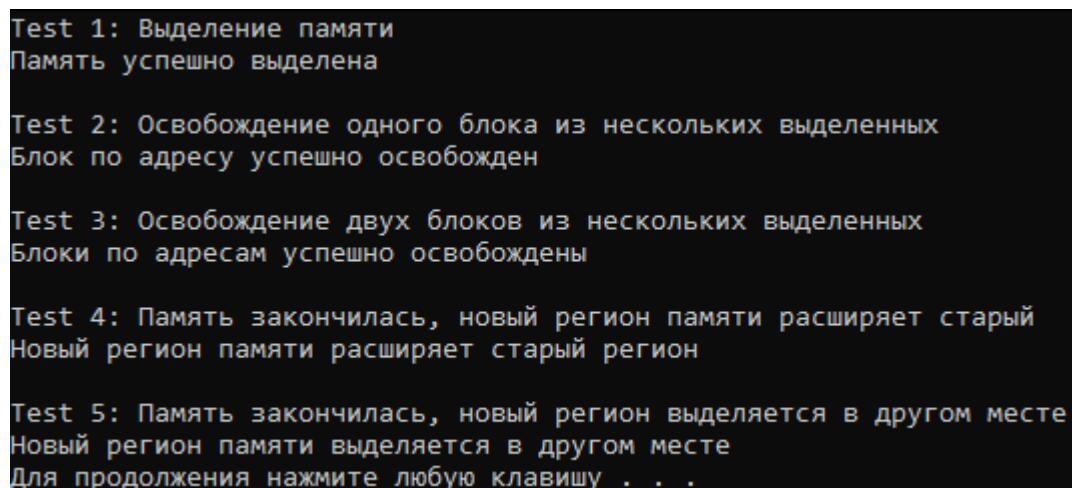
4 Отладка и тестирование

Согласно заданию, в качестве среды разработки была выбрана программа Microsoft Visual Studio 2019, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ. Для отладки программы использовались точки останова и пошаговое выполнение кода программы, анализ содержимого глобальных и локальных переменных.

В процессе отладки не было выявлено ошибок.

Тестирование было проведено несколько раз и получило положительный результат.

Отдельно был произведён тест аллокатора предоставленный на рисунке 2.



```
Test 1: Выделение памяти
Память успешно выделена

Test 2: Освобождение одного блока из нескольких выделенных
Блок по адресу успешно освобожден

Test 3: Освобождение двух блоков из нескольких выделенных
Блоки по адресам успешно освобождены

Test 4: Память закончилась, новый регион памяти расширяет старый
Новый регион памяти расширяет старый регион

Test 5: Память закончилась, новый регион выделяется в другом месте
Новый регион памяти выделяется в другом месте
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2 - Аллокатор памяти

5 Описание программы

Программа база данных контактов представляет собой удобное решение для хранения информации о контактах, которое позволяет пользователю легко добавлять, удалять, изменять и просматривать информацию о контактах. Программа имеет удобный интерфейс, в котором пользователь может выбрать нужную операцию с помощью меню. Меню (“Введите ”) содержит 11 кнопок и используется с помощью клавиатуры, а именно ввод цифр под определённые задачи :

«Для добавления контактов» – При нажатии на 1 вызывается функция добавления. Это один из вариантов начала программы базы данных. Также можно начать загрузку контактов из готового, заранее описанного txt файла под названием 1.txt, находящийся (D:\ProjectKURS\ProjectKURS\1.txt) (Нажатие на 1)

«Для удаления контактов» - Создана для удаления ненужных контактов и проверки загрузочных контактов из файла. (Нажатие на 2)

«Для вывода контактов» - Самая главная часть программы, реализующая вывод контактов для пользователя, показывающая данные. (Нажатие на 3)

«Для сохранения контактов в файл» - Создана и используется ,если мы хотим обновить нашу базу контактов и сохранить их. (Нажатие на 4)

«Для загрузки контактов из файла» - Создана и используется ,если мы хотим загрузить нашу базу контактов и вывести контакты из файла. (Нажатие на 5)

«Аллокатор» - Для тестирования различных функций управления памятью. Это дополнительная функция программы с 5 тестами (Нажатие на 6)

«Изменение Контакта» - Если мы хотим изменить какой-либо контакт , полностью поменяв его или редактируя. (Нажатие на 7)

«Сортировка по номеру телефона» - Мы можем сортировать контакты для более удобного прочтения их в базе . (Нажатие на 8)

«Обмен местами» - Создана для более удобного представления пользователю контактов , например для важности какого-то контакта, можно поместить его на первые места. (Нажатие на 9)

«Поиск» - Ищет нужный нам контакт в огромном списке. (Нажатие на 10)

«Для выхода из программы»-Реализует выход из программы. (Нажатие на 0)

На следующих рисунках 3-5 предоставлены блок схемы программы .

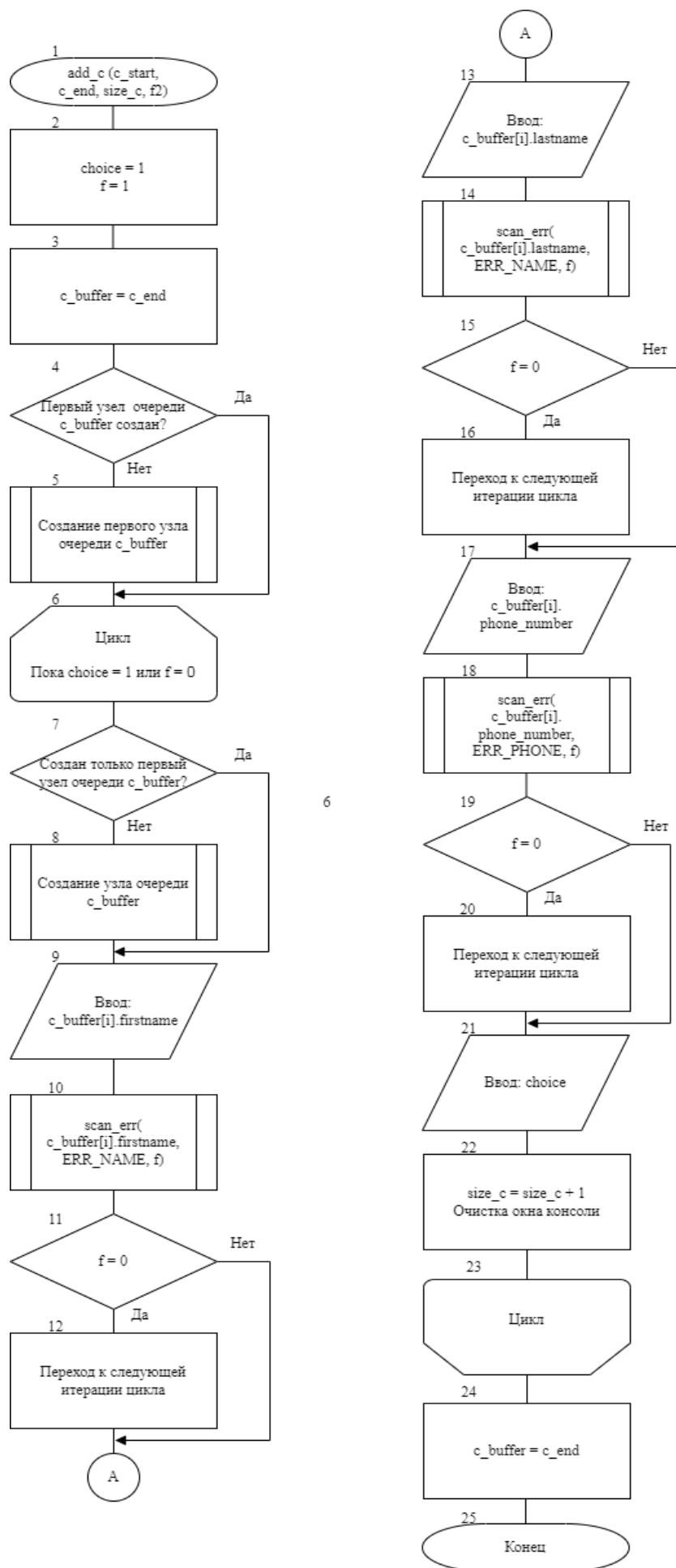


Рисунок 3 — Блок-схема алгоритма add_c

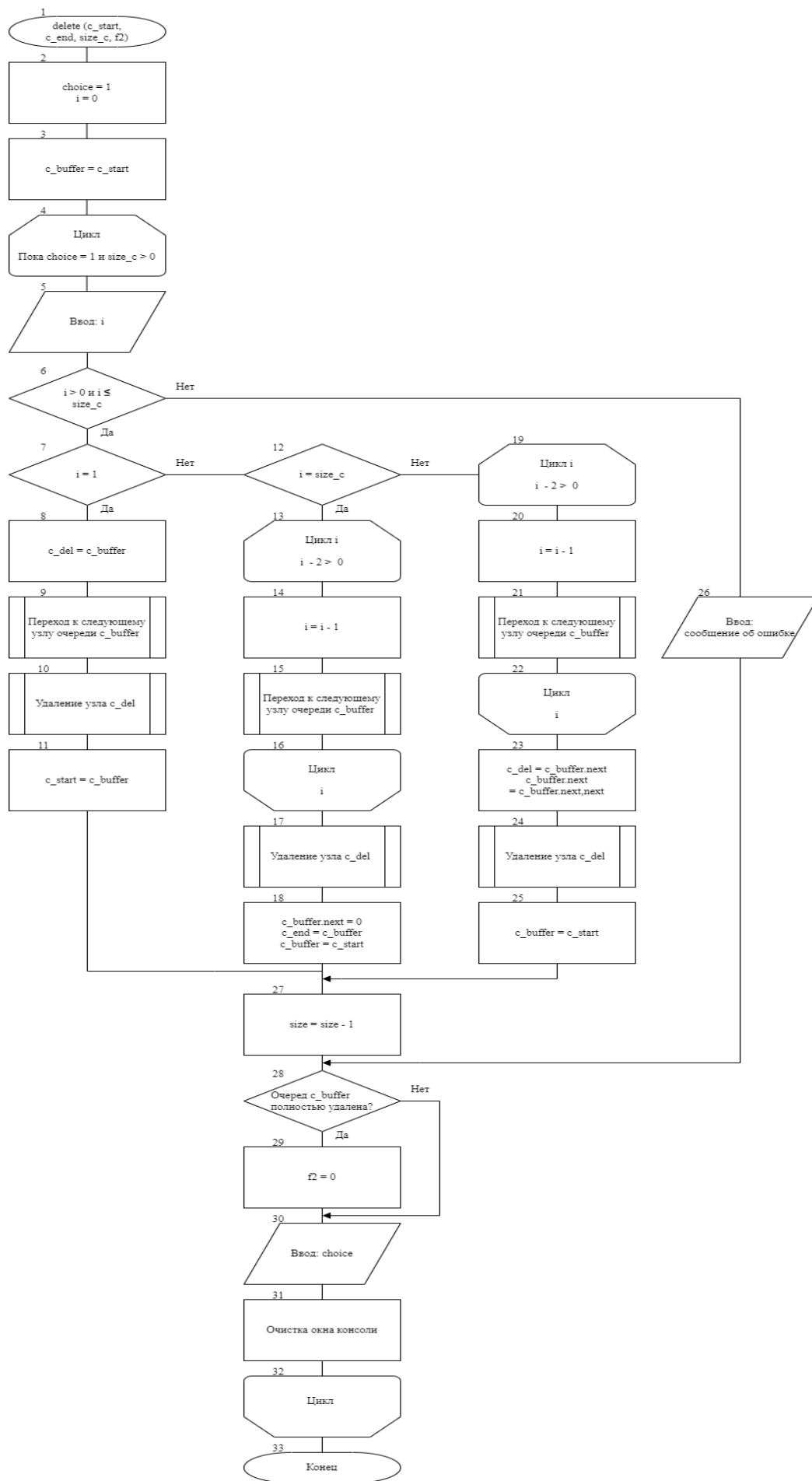


Рисунок 4 — Блок-схема алгоритма delete_c

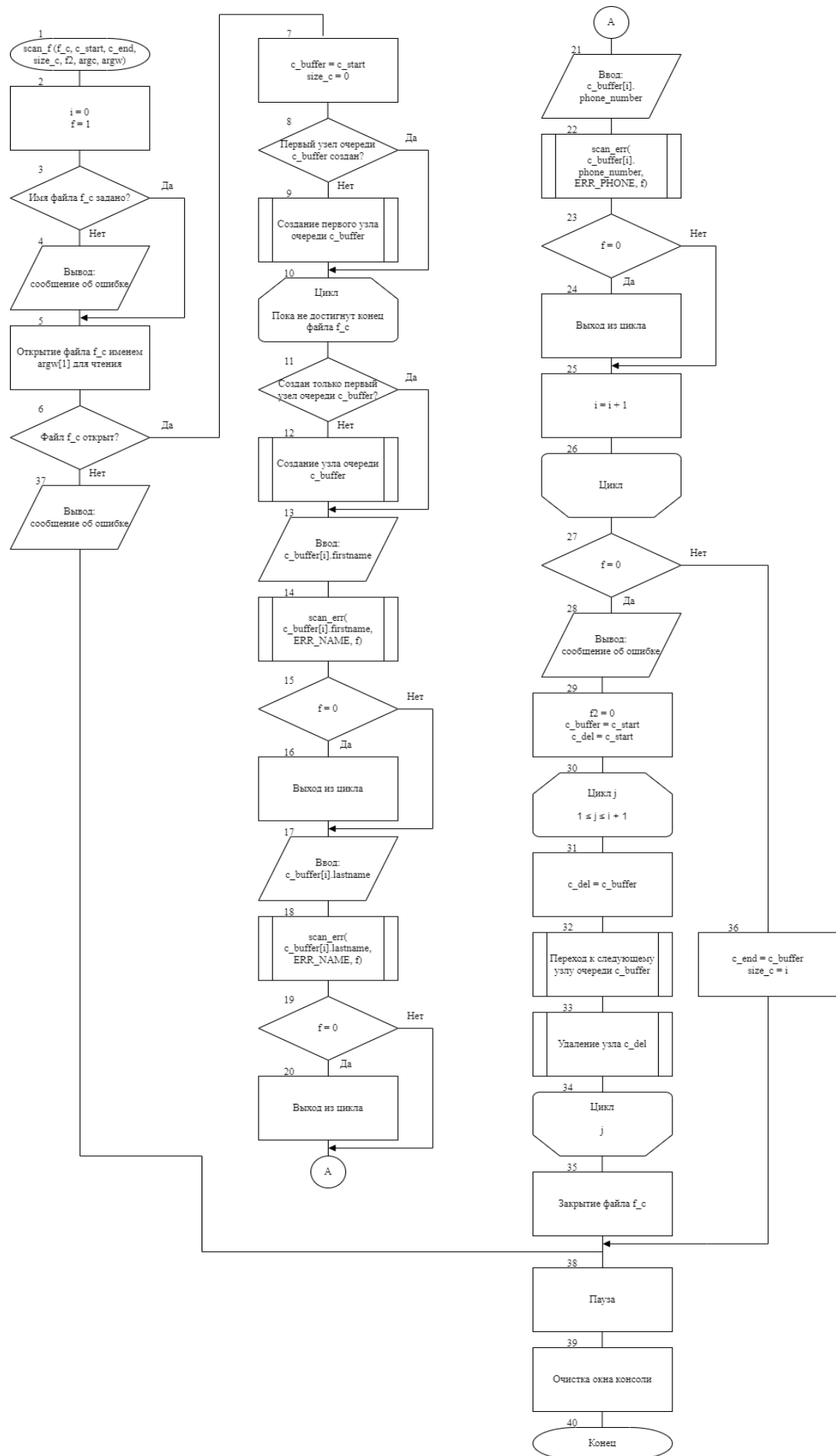
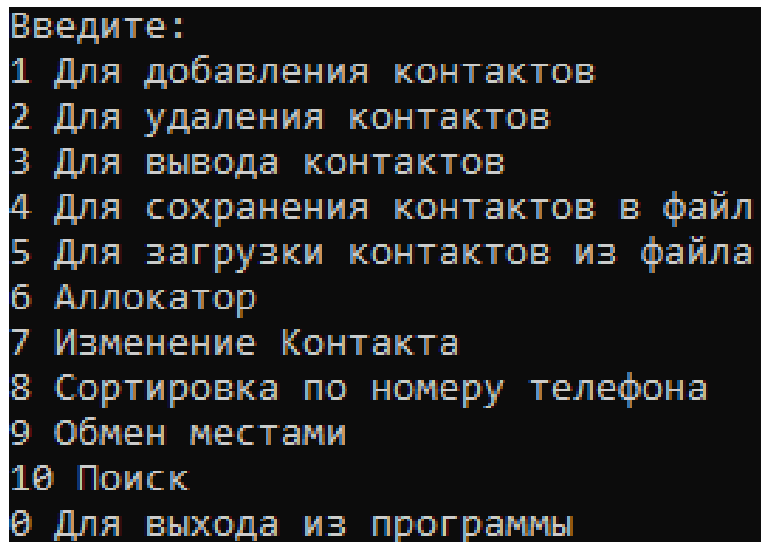


Рисунок 5 — Блок-схема алгоритма scan_f

6 Руководство пользователя

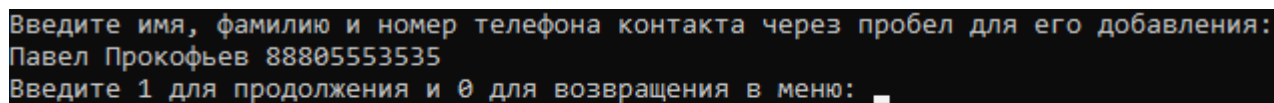
При первом запуске появляется меню. Здесь пользователь может выбрать нужный ему запрос или выйти из программы. Представлено на рисунке 6.



```
Введите:  
1 Для добавления контактов  
2 Для удаления контактов  
3 Для вывода контактов  
4 Для сохранения контактов в файл  
5 Для загрузки контактов из файла  
6 Аллокатор  
7 Изменение Контакта  
8 Сортировка по номеру телефона  
9 Обмен местами  
10 Поиск  
0 Для выхода из программы
```

Рисунок 6 - Меню программы

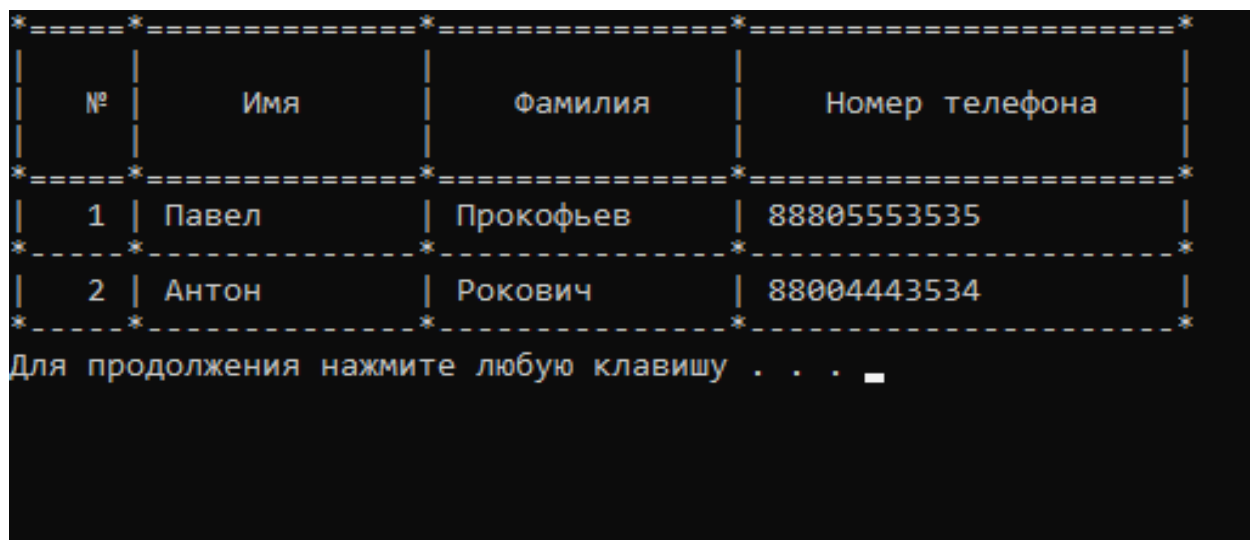
В самом меню, можно начать действие программы с нажатия клавиши 1 и добавить контакт. Также нас встретит строка после ввода, которая предложит продолжить или вернуться в главное меню . Представлено на рисунке 7.



```
Введите имя, фамилию и номер телефона контакта через пробел для его добавления:  
Павел Прокофьев 88805553535  
Введите 1 для продолжения и 0 для возвращения в меню: _
```

Рисунок 7 - Для добавления контактов

Попробуем добавить новый контакт . Для этого можно сразу ввести цифру 1 или выйти в меню. После этого попробуем проверить контакты в базе, нажав на цифру 3 в главном меню. Представлено на рисунке 8.

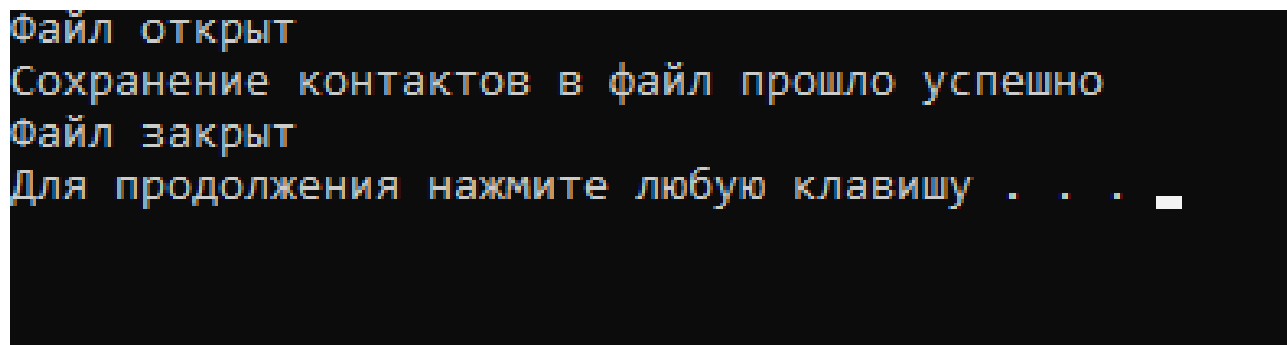


№	Имя	Фамилия	Номер телефона
1	Павел	Прокофьев	88805553535
2	Антон	Рокович	88004443534

Для продолжения нажмите любую клавишу . . .

Рисунок 8 – Вывод контактов

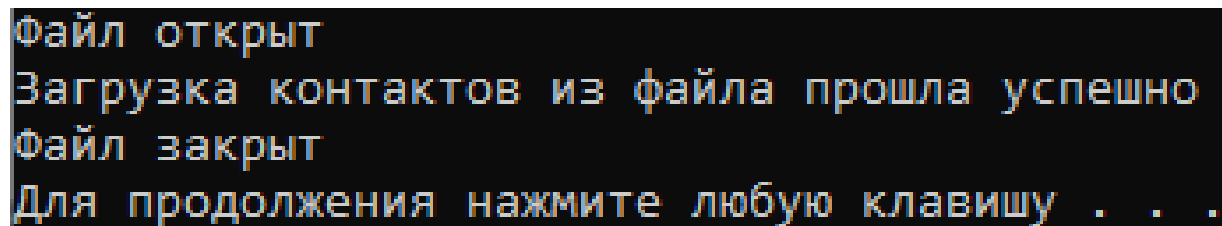
Дальше можно воспользоваться всеми удобными способами размещения контактов, редактирования и поиска. Пользователь может сохранить в txt файл свои заготовки и вывести позже. Представлено на рисунке 9.



Файл открыт
Сохранение контактов в файл прошло успешно
Файл закрыт
Для продолжения нажмите любую клавишу . . .

Рисунок 9 – Сохранение в файл

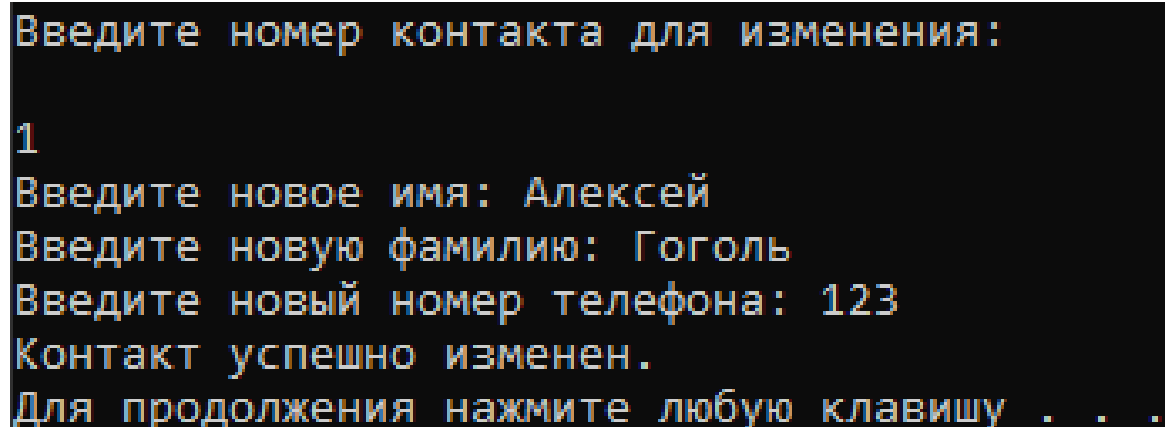
После успешного сохранения можно попробовать выйти из программы через меню и нажатия клавиши 0 . Перезаходим в программу и пробуем загрузить из файла наши контакты . Представлено на рисунке 10.



```
Файл открыт
Загрузка контактов из файла прошла успешно
Файл закрыт
Для продолжения нажмите любую клавишу . . .
```

Рисунок 10 – Выгрузка из файлов

Попробуем изменить контакт , для этого нажмём на клавишу 7 в главном меню. Представлено на рисунке 11.



```
Введите номер контакта для изменения:
1
Введите новое имя: Алексей
Введите новую фамилию: Гоголь
Введите новый номер телефона: 123
Контакт успешно изменен.
Для продолжения нажмите любую клавишу . . .
```

Рисунок 11 – Изменение

Удалим контакт с помощью нажатия 2. Оставим только контакт Алексей, введя его порядковый номер. Представлено на рисунке 12.

```
Введите порядковый номер контакта для его удаления:
2
Введите 1 для продолжения и 0 для возвращения в меню:
```

Рисунок 12 – Удаление

Как мы убедились все функции были выполнены, для следующих функций меню, попробуем создать несколько контактов с короткими номерами. Дальше отсортируем их и выведем. Представлено на рисунке 13.

```
*=====*
```

№	Имя	Фамилия	Номер телефона
1	l	l	1
2	f	f	2
3	g	g	3

```
*=====*
```

Для продолжения нажмите любую клавишу . . .

Рисунок 13 – Сортировка

Реализуем обмен местами несколько раз. Представлено на рисунке 14.

```
Введите номера контактов для изменения местами: 1 3
Контакты успешно изменены местами.
Для продолжения нажмите любую клавишу . . .
```

Рисунок 14 – Обмен местами

Осуществим поиск контакта через меню. Представлено на рисунке 15.

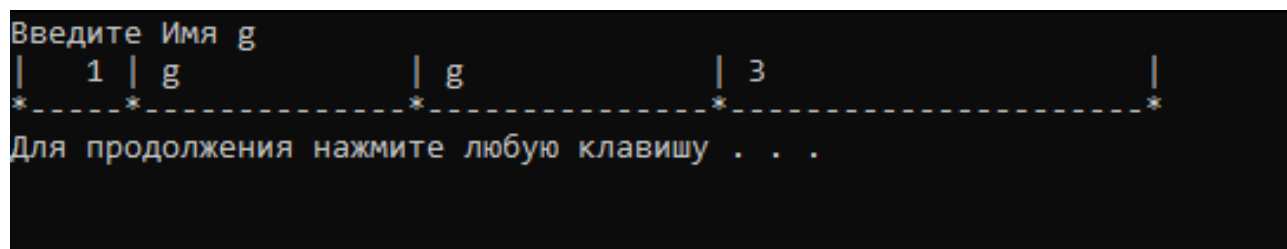


Рисунок 15 – Поиск

При условиях прошлой функции, мы поменял 1 и 3 контакт местами, значит контакт g находится на первом месте. Далее по поиску мы нашли данный контакт в базе данных, используя его имя.

Руководство пользователя закончена. Чтобы выйти из программы нажмите в меню на клавишу 0 .

Заключение

При выполнении данной курсовой работы были улучшены базовые навыки программирования на языке C .

Изучены основные возможности среды программирования Microsoft VisualStudio 2019. Получены навыки отладки и тестирования программ. Были освоены приемы создания и подключения модулей программы. Получены навыки работы со сложными типами данных, такие как структуры и файлы.

В результате была разработана База данных Контакты, с приятным текстовым оформлением и простым для использования интерфейсом, меню.

Список используемых источников

1. <https://ru.wikipedia.org/wiki/>
2. <https://pnzgu.ru>
<https://www.cyberforum.ru/c/>
3. <https://github.com>
4. <https://habr.com/ru/post/505632/>
5. <https://itnan.ru/post.php?c=1&p=590415>
6. <https://learn.microsoft.com/ru-ru/cpp/standard-library/allocator-class?view=msvc-160&viewFallbackFrom=vs-2019>

Приложение А

Основной код в модулях

Файл add_c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"
#include "scan_err.h"

void add_c(struct contact** c_start, struct contact** c_end, int* size_c, int* f2) {
    int choice = 1, f1 = 1;
    struct contact* c_buffer = *c_end;
    if (*f2 == 0)
        start_push(&c_buffer, c_start, f2);
    while ((choice == 1) || (!f1)) {
        if (*size_c > 0)
            push(&c_buffer);
        printf("Введите имя, фамилию и номер телефона контакта через пробел для его добавления:\n");
        scanf("%s", c_buffer->firstname);
        scan_err(c_buffer->firstname, ERR_NAME, &f1);
        if (!f1)
            continue;
        scanf("%s", c_buffer->lastname);
        scan_err(c_buffer->lastname, ERR_NAME, &f1);
        if (!f1)
            continue;
        scanf("%s", c_buffer->phone_number);
        scan_err(c_buffer->phone_number, ERR_PHONE, &f1);
        if (!f1)
            continue;
        printf("Введите 1 для продолжения и 0 для возвращения в меню: ");
        scanf("%d", &choice);
        ++*size_c;
        system("cls");
    }
    *c_end = c_buffer;
}

void start_push(struct contact** head, struct contact** c_start, int* f2) {
    struct contact* tmp = (struct contact*)malloc(sizeof(struct contact));
    if (tmp == NULL)
        exit(1);
    tmp->firstname = (char*)malloc(30 * sizeof(char));
    if (tmp->firstname == NULL)
        exit(1);
    tmp->lastname = (char*)malloc(30 * sizeof(char));
    if (tmp->lastname == NULL)
        exit(1);
    tmp->phone_number = (char*)malloc(30 * sizeof(char));
    if (tmp->phone_number == NULL)
        exit(1);
    tmp->next = NULL;
    *head = tmp;
    *c_start = *head;
    *f2 = 1;
}

void push(struct contact** head) {
    struct contact* tmp = (struct contact*)malloc(sizeof(struct contact));
    if (tmp == NULL)
        exit(1);
    tmp->firstname = (char*)malloc(30 * sizeof(char));
    if (tmp->firstname == NULL)
        exit(1);
    tmp->lastname = (char*)malloc(30 * sizeof(char));
```

```

        if (tmp->lastname == NULL)
            exit(1);
        tmp->phone_number = (char*)malloc(30 * sizeof(char));
        if (tmp->phone_number == NULL)
            exit(1);
        tmp->next = NULL;
        (*head)->next = tmp;
        *head = tmp;
    }
    void pop(struct contact** head) {
        if ((*head) == NULL)
            exit(1);
        (*head) = (*head)->next;
    }
    void free_node(struct contact** del) {
        free((*del)->firstname);
        free((*del)->lastname);
        free((*del)->phone_number);
        free((*del));
    }
}

```

Файл alloc

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"

void alloc() {
    printf("Test 1: Выделение памяти\n");
    int* ptr1 = (int*)malloc(sizeof(int));
    if (ptr1 != NULL) {
        printf("Память успешно выделена\n");
    }
    else {
        printf("Ошибка при выделении памяти\n");
    }
    printf("\nTest 2: Освобождение одного блока из нескольких выделенных\n");
    int* ptr2 = (int*)malloc(sizeof(int));
    int* ptr3 = (int*)malloc(sizeof(int));
    free(ptr2);
    printf("Блок по адресу успешно освобожден\n", ptr2);
    printf("\nTest 3: Освобождение двух блоков из нескольких выделенных\n");
    int* ptr4 = (int*)malloc(sizeof(int));
    int* ptr5 = (int*)malloc(sizeof(int));
    free(ptr4);
    free(ptr5);
    printf("Блоки по адресам успешно освобождены\n", ptr4, ptr5);
    printf("\nTest 4: Память закончилась, новый регион памяти расширяет старый\n");
    int* ptr6 = (int*)malloc(sizeof(int));
    if (ptr6 != NULL) {
        printf("Новый регион памяти расширяет старый регион\n");
    }
    else {
        printf("Ошибка выделения памяти\n");
    }
    printf("\nTest 5: Память закончилась, новый регион выделяется в другом месте\n");
    int* ptr7 = (int*)malloc(sizeof(int));
    if (ptr7 != NULL) {
        printf("Новый регион памяти выделяется в другом месте\n");
    }
    else {
        printf("Ошибка выделения памяти\n");
    }
    system("pause");
    system("cls");
}

```

Файл change

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"
void change(struct contact** c_start, int size_c) {
    int contact_number;
    printf("Введите номер контакта для изменения: ");
    scanf("%d", &contact_number);
    if (contact_number < 1 || contact_number > size_c) {
        printf("Неверный номер контакта.\n");
        return;
    }
    struct contact* current = *c_start;
    for (int i = 1; i < contact_number; i++) {
        current = current->next;
    }
    char firstname[100], lastname[100], phone_number[100];
    printf("Введите новое имя: ");
    scanf("%s", firstname);
    printf("Введите новую фамилию: ");
    scanf("%s", lastname);
    printf("Введите новый номер телефона: ");
    scanf("%s", phone_number);
    strcpy(current->firstname, firstname);
    strcpy(current->lastname, lastname);
    strcpy(current->phone_number, phone_number);
    printf("Контакт успешно изменен.\n");
    system("pause");
    system("cls");
}

```

Файл delete_c

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"
void delete_c(struct contact** c_start, struct contact** c_end, int* size_c, int* f2) {
    int choice = 1, i = 0;
    struct contact* c_buffer = *c_start;
    struct contact* c_del = NULL;
    while ((choice == 1) && (*size_c > 0)) {
        printf("Введите порядковый номер контакта для его удаления: \n");
        scanf("%d", &i);
        if (i > 0 && i <= *size_c) {
            if (i == 1) {
                c_del = c_buffer;
                pop(&c_buffer);
                free_node(&c_del);
                *c_start = c_buffer;
            }
            else if (i == *size_c) {
                while (i - 2 > 0) {
                    --i;
                    pop(&c_buffer);
                }
                free_node(&c_buffer->next);
                c_buffer->next = NULL;
                *c_end = c_buffer;
                c_buffer = *c_start;
            }
            else {
                while (i - 2 > 0) {
                    --i;
                    pop(&c_buffer);
                }
            }
        }
    }
}

```

```

        c_del = c_buffer->next;
        c_buffer->next = c_buffer->next->next;
        free_node(&c_del);
        c_buffer = *c_start;
    }
    --* size_c;
}
else
    fprintf(stderr, "Контакта под данным номером не существует\n");
if (*size_c == 0)
    *f2 = 0;
printf("Введите 1 для продолжения и 0 для возвращения в меню: ");
scanf("%d", &choice);
system("cls");
}
}

```

Файл find

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <windows.h>
#include "main.h"
#include "add_c.h"
#include <string.h>

void find(struct contact* c_start, int* size_c) {
    struct contact* c_buffer = c_start;
    printf("Введите Имя ");
    char find[] = "a";
    scanf("%s", find);
    for (int i = 0; i < *size_c; i++) {
        if (strcmp(find, c_buffer->firstname) == 0) {
            printf("| %3d | %-12s | %-13s | %-20s |\n", i + 1, c_buffer->firstname, c_buffer->lastname, c_buffer->phone_number);
            printf("*-----*-----*-----*-----*\n");
        }
        pop(&c_buffer);
    }
    system("pause");
    system("cls");
}

```

Файл main

```

#define _CRT_SECURE_NO_WARNINGS
#define _CRTDBG_MAP_ALLOC
#include <windows.h>
#include <crtdbg.h>
#include <stdlib.h>
#include <stdio.h>
#include "add_c.h"
#include "delete_c.h"
#include "print_c.h"
#include "print_f.h"
#include "scan_f.h"

struct contact {
    char* firstname;
    char* lastname;
    char* phone_number;
    struct contact* next;
};

int main(int argc, int** argv) {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
    FILE* f_c = NULL;
    int size_c = 0, f2 = 0, choice = 1;
    struct contact* c_start = NULL, *c_end = NULL;
    do {

```

printf("Введите: \n1 Для добавления контактов\n2 Для удаления контактов\n3 Для вывода контактов\n4 Для сохранения контактов в файл\n5 Для загрузки контактов из файла\n6 Аллокатор\n7 Изменение Контакта\n8 Сортировка по номеру телефона\n9 Обмен местами\n10 Поиск\n0 Для выхода из программы\n");

```
scanf("%d", &choice);
switch (choice) {
case 1: system("cls");
        add_c(&c_start, &c_end, &size_c, &f2);
        break;
case 2: system("cls");
        delete_c(&c_start, &c_end, &size_c, &f2);
        break;
case 3: system("cls");
        print_c(c_start, &size_c);
        break;
case 4: system("cls");
        print_f(&f_c, c_start, &size_c, argc, argv);
        break;
case 5: system("cls");
        scan_f(&f_c, &c_start, &c_end, &size_c, &f2, argc, argv);
        break;
case 6: system("cls");
        alloc();
        break;
case 7: system("cls");
        change(&c_start, size_c);
        break;
case 8: system("cls");
        sort(&c_start, size_c);
        break;
case 9: system("cls");
        pos(&c_start, size_c);
        break;
case 10: {
        system("cls");
        find(c_start, &size_c);
        break;
}
case 0: {
        struct contact* c_del = c_start;
        for (int i = 0; i < size_c; i++) {
            c_del = c_start;
            pop(&c_start);
            free_node(&c_del);
        }
        break;
}
default:
        break;
}
} while (choice != 0);

return 0;
}
```

Файл pos

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"
```

```
void pos(struct contact** c_start, int size_c) {
    int contact_number1, contact_number2;
    printf("Введите номера контактов для изменения местами: ");
    scanf("%d %d", &contact_number1, &contact_number2);
    if (contact_number1 < 1 || contact_number1 > size_c || contact_number2 < 1 || contact_number2 > size_c) {
        printf("Неверный номер контакта.\n");
        return;
    }
}
```

```

struct contact* current1 = *c_start;
for (int i = 1; i < contact_number1; i++) {
    current1 = current1->next;
}
struct contact* current2 = *c_start;
for (int i = 1; i < contact_number2; i++) {
    current2 = current2->next;
}
char temp_firstname[100], temp_lastname[100], temp_phone_number[100];
strcpy(temp_firstname, current1->firstname);
strcpy(temp_lastname, current1->lastname);
strcpy(temp_phone_number, current1->phone_number);
strcpy(current1->firstname, current2->firstname);
strcpy(current1->lastname, current2->lastname);
strcpy(current1->phone_number, current2->phone_number);
strcpy(current2->firstname, temp_firstname);
strcpy(current2->lastname, temp_lastname);
strcpy(current2->phone_number, temp_phone_number);
printf("Контакты успешно изменены местами.\n");
system("pause");
system("cls");
}

```

Файл print_c

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <windows.h>
#include "main.h"
#include "add_c.h"

void print_c(struct contact* c_start, int* size_c) {
    struct contact* c_buffer = c_start;
    if (*size_c > 0) {
        printf("*****\n|
        |\n| %3s|  % -9s|  % -11s|  % -18s\n| | | |
        |\n*****\n", "№", "Имя", "Фамилия", "Номер
        телефона");
        for (int i = 0; i < *size_c; i++) {
            printf("| %3d | % -12s | % -13s | % -20s \n", i + 1, c_buffer->firstname, c_buffer->lastname, c_buffer-
            >phone_number);
            printf("*****\n");
            pop(&c_buffer);
        }
    }
    else
        fprintf(stderr, "Контакты отсутствуют \n");
    system("pause");
    system("cls");
}

```

Файл print_f

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"
void print_f(FILE* f_c, struct contact* c_start, int* size_c, int argc, int** argw) {
    f_c = NULL;
    int f = 1;
    struct contact* c_buffer = c_start;
    if (argc < 2)
        fprintf(stderr, "Имя файла не задано\n");
    f_c = fopen(argw[1], "w");
    if (f_c == NULL)
        fprintf(stderr, "Не получилось открыть файл\n");
    else {
        printf("Файл открыт\n");
        if (*size_c > 0) {
            for (int i = 0; i < *size_c; i++) {

```

```

        fprintf(f_c, "%s\t\t%s\t\t%s", c_buffer->firstname, c_buffer->lastname, c_buffer->
>phone_number);

        pop(&c_buffer);
        if (i != *size_c - 1)
            fprintf(f_c, "\n");
    }
    printf("Сохранение контактов в файл прошло успешно\n");
}
else {
    fprintf(stderr, "Контакты отсутствуют\n");
    fprintf(stderr, "Ошибка при сохранении контактов в файл\n");
}
fclose(f_c);
printf("Файл закрыт\n");
}
system("pause");
system("cls");
}

```

Файл scan_err

```

#include <stdio.h>
#include <string.h>

```

```

void scan_err(char s[], char ERR[], int* f) {
    if (strpbrk(s, ERR) != NULL) {
        fprintf(stderr, "Ошибка ввода\n");
        *f = 0;
        while (getchar() != '\n');
    }
    else
        *f = 1;
}

```

Файл scan_f

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"
#include "scan_err.h"

```

```

void scan_f(FILE* f_c, struct contact** c_start, struct contact** c_end, int* size_c, int* f2, int argc, int** argv) {
    f_c = NULL;
    int i = 0, f = 1;
    if (argc < 2)
        fprintf(stderr, "Имя файла не задано\n");
    f_c = fopen(argv[1], "r");
    if (f_c == NULL)
        fprintf(stderr, "Не получилось открыть файл\n");
    else {
        struct contact* c_buffer = *c_start;
        *size_c = 0;
        if (*f2 == 0)
            start_push(&c_buffer, c_start, f2);
        printf("Файл открыт\n");
        while (feof(f_c) == 0) {
            if (i > 0)
                push(&c_buffer);
            fscanf(f_c, "%s", c_buffer->firstname);
            scan_err(c_buffer->firstname, ERR_NAME, &f);
            if (!f)
                break;
            fscanf(f_c, "%s", c_buffer->lastname);
            scan_err(c_buffer->lastname, ERR_NAME, &f);
            if (!f)
                break;
            fscanf(f_c, "%s", c_buffer->phone_number);
            scan_err(c_buffer->phone_number, ERR_PHONE, &f);
            if (!f)
                break;

```

```

        i++;
    }
    if (!f) {
        fprintf(stderr, "Ошибка при загрузке контактов из файла\n");
        *f2 = 0;
        c_buffer = *c_start;
        struct contact* c_del = *c_start;
        for (int j = 0; j < i + 1; j++) {
            c_del = c_buffer;
            pop(&c_buffer);
            free_node(&c_del);
        }
    }
    else {
        printf("Загрузка контактов из файла прошла успешно\n");
        *c_end = c_buffer;
        *size_c = i;
    }
    fclose(f_c);
    printf("Файл закрыт\n");
}
system("pause");
system("cls");
}

```

Файл sort

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "main.h"
#include "add_c.h"
void sort(struct contact** c_start, int size_c) {
    struct contact* current = *c_start;
    for (int i = 0; i < size_c - 1; i++) {
        struct contact* next = current->next;
        for (int j = i + 1; j < size_c; j++) {
            if (strcmp(current->phone_number, next->phone_number) > 0) {
                char temp_firstname[100], temp_lastname[100], temp_phone_number[100];
                strcpy(temp_firstname, current->firstname);
                strcpy(temp_lastname, current->lastname);
                strcpy(temp_phone_number, current->phone_number);
                strcpy(current->firstname, next->firstname);
                strcpy(current->lastname, next->lastname);
                strcpy(current->phone_number, next->phone_number);
                strcpy(next->firstname, temp_firstname);
                strcpy(next->lastname, temp_lastname);
                strcpy(next->phone_number, temp_phone_number);
            }
            next = next->next;
        }
        current = current->next;
    }
    system("pause");
    system("cls");
}

```

Приложение Б

Заголовок add_c

```
#pragma once
extern struct contact;
void add_c(struct contact** c_start, struct contact** c_end, int* size_c, int* f2);
void start_push(struct contact** head, struct contact** c_start, int* f2);
void push(struct contact** head);
void pop(struct contact** head);
void free_node(struct contact** del);
```

Заголовок alloc

```
#pragma once
extern struct contact;

void alloc(struct contact* c_start, int* size_c);
```

Заголовок change

```
#pragma once
extern struct contact;

void change(struct contact* c_start, int* size_c);
```

Заголовок delete_c

```
#pragma once
extern struct contact;
void delete_c(struct contact** c_start, struct contact** c_end, int* size_c, int* f2);
```

Заголовок find

```
#pragma once
extern struct contact;

void find_c(struct contact* c_start, int* size_c);
```

```
#pragma once
```

```
struct contact {
    char* firstname;
    char* lastname;
    char* phone_number;
    struct contact* next;
};
```

Заголовок search

```
#pragma once
extern struct contact;

void search(struct contact* c_start, int* size_c);
```

Заголовок print_c

```
#pragma once
extern struct contact;
void print_c(struct contact* c_start, int* size_c);
```

Заголовок print_f

```
#pragma once
extern struct contact;
void print_f(FILE* f_c, struct contact* c_start, int* size_c, int argc, int** argw);
```

Заголовок scan_err

```
#pragma once
#define ERR_PHONE
"qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNMйцукенгшщзхъфывапролджэячсмитьбюёЙЦУКЕНГ  
ШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЁ[{}];|<>.,~/~`@#$$%^&*()_+~№"
#define ERR_NAME "1234567890"
void scan_err(char s[], char ERR[], int* f);
```

Заголовок scan_f

```
#pragma once
extern struct contact;
void scan_f(FILE* f_c, struct contact** c_start, struct contact** c_end, int* size_c, int* f2, int argc, int** argw);
```

Заголовок sort

```
#pragma once
extern struct contact;
void sort(struct contact* c_start, int* size_c);
```