Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему «Разработка программы сложной структуры методом нисходящего программирования. База данных "Телефонная книга"»

Выполнил: студент гр25ВВВ4
Джураев Б. В.

Принял:
к.т.н., доцент Токарев А. Н.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра “Вычислительная техника”

“УТВЕРЖДАЮ”

Зав. кафедрой ВТ

_____ М.А. Митрохин

«____»_____20__г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Окуневу Богдану Вячеславовичу Группа 25ВВВ4.

Тема проекта: ***Разработка программы сложной структуры методом нисходящего программирования.***

Исходные данные (технические требования) на проектирование

Тема: телефонная книга.

Требования:
- Добавление записей
- Удаление записей
- Редактирование записей
- Поиск и сортировка записей
- Фильтрация записей по определенному критерию
- Создание новой базы данных
- Сохранение текущей базы данных
- Загрузка сохраненных баз данных

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, 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 MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. **Расчетная часть**

Разработка программы.

2. **Графическая часть**

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. **Экспериментальная часть**

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "18" 02 2026

Дата защиты проекта "___" _____

Руководитель _____

Задание получил "25" 02 2026 г.

Студент Отрыв Богдан Вячеславович.

СОДЕРЖАНИЕ

Введение..... 5
Постановка задачи 8
Выбор решения..... 10
Описание разработки программы..... 12
Отладка и тестирование 14
Описание программы..... 15
Руководство пользователя..... 21
Заключение 22
Список используемых источников..... 23
Приложение А. Листинги программы**Ошибка! Закладка не определена.**
Приложение Б. Результаты работы**Ошибка! Закладка не определена.**

Введение

Информационные технологии и автоматизированные системы управления базами данных (СУБД) на современном этапе развития вычислительной техники составляют основу программного обеспечения во всех сферах человеческой деятельности. Задачи структурирования, упорядочивания, долговременного хранения и оперативной фильтрации информации являются классическими в области системного программирования. Одной из базовых, но в то же время комплексных моделей таких систем является локальная автоматизированная система ведения контактных данных типа «Телефонная книга». Проектирование программного комплекса подобного класса требует от разработчика применения фундаментальных принципов структурного программирования, нисходящей декомпозиции, модульного анализа, организации бинарных файловых потоков данных, а также глубокого понимания механизмов распределения ресурсов в адресном пространстве электронно-вычислительных машин (ЭВМ).

При проектировании программных систем, оперирующих динамически изменяющимися объёмами информации, перед разработчиком встаёт фундаментальный вопрос выбора архитектуры хранения данных в оперативной памяти (ОЗУ). Существует два принципиально различных методологических подхода: использование статических структур (массивов) и построение динамических линейных или иерархических структур данных.

Использование статических массивов накладывает на программную систему ряд жестких, зачастую неприемлемых в промышленной эксплуатации ограничений. Прежде всего, это ограничение масштабируемости: размер массива декларируется на этапе компиляции исходного кода и не может быть изменён в процессе выполнения программы без её перезапуска. Это приводит к невозможности обработки информационных объёмов, превышающих заранее заданный лимит. Кроме

того, наблюдается нерациональное использование адресного пространства кучи (heap) и стека (stack). При выделении памяти под максимальное гипотетическое количество записей (например, 10 000 элементов) ресурсы ОЗУ резервируются операционной системой монопольно. Если фактически в базе данных хранится лишь несколько записей, оставшаяся выделенная память простаивает, блокируя ресурсы для других процессов ОС.

Динамические структуры данных, в частности линейные односвязные списки, полностью лишены вышеперечисленных недостатков. В данной архитектурной модели понятие монолитного массива подменяется понятием изолированных узлов (структур), распределённых по адресному пространству памяти. Выделение памяти под каждый конкретный узел осуществляется операционной системой строго по требованию (динамически) в момент инициализации операции добавления новой записи посредством системных вызовов. Связность разрозненных в пространстве кучи блоков данных обеспечивается за счет внедрения в структуру узла специализированного адресного полей — указателя на следующий элемент. Таким образом, верхний предел количества хранящихся записей в СУБД ограничивается не искусственными рамками компилятора, а исключительно физическим объёмом оперативной памяти, доступным текущему процессу ЭВМ. При удалении элемента память мгновенно возвращается операционной системе, что гарантирует высокую степень системной интеграции и бережное отношение к ресурсам компьютера.

Важным научно-практическим аспектом данной курсовой работы является исследование механизмов межъязыкового взаимодействия (interoperability) между языками программирования различных уровней abstraction. Интеграция высокоуровневого языка структурного программирования C с низкоуровневым языком Ассемблер (Assembler architecture x86) позволяет успешно сочетать высокую скорость разработки, читаемость и модульность прикладного программного кода со

сверхвысоким быстродействием низкоуровневых вставок. Вынесение критических с точки зрения вычислительной сложности или частоты вызовов операций (таких как инкремент глобального счетчика базы данных) на аппаратный уровень — с прямой манипуляцией регистрами центрального процессора (ЦП) — представляет собой классический приём оптимизации системного программного обеспечения.

Обоснование необходимости выполнения работы:

– Формирование профессиональных компетенций. В процессе выполнения работы приобретаются и закрепляются навыки ручного управления динамической памятью (malloc, free), углублённо изучается адресная арифметика, механизмы работы с многомерными указателями, а также принципы сквозной компиляции и линковки многомодульных систем. Осваивается синтаксис и специфика выполнения машинных команд встроенного ассемблера в современных интегрированных средах разработки (IDE).

– Практическая и технологическая ценность. Спроектированная в рамках курсовой работы архитектура локальной СУБД на базе связного списка и бинарной файловой подсистемы представляет собой готовый отчуждаемый программный каркас. Данные решения могут быть непосредственно интегрированы в программное обеспечение встраиваемых систем, умных устройств, контроллеров автоматизации и систем реального времени, где развёртывание промышленных реляционных СУБД технически невозможно из-за жестких аппаратных ограничений.

Постановка задачи

Основной технической и научно-методической целью данной курсовой работы является проектирование, многомодульная программная реализация, комплексная отладка и верификационное тестирование консольного прикладного программного обеспечения «Автоматизированная локальная база данных «Телефонная книга»». Программный продукт должен функционировать в рамках концепции динамического распределения памяти, использовать механизмы низкоуровневой оптимизации вычислений и обеспечивать персистентность (сохраняемость) информации.

Для достижения детерминированного результата проектирования необходимо формализовать и реализовать следующие функциональные требования и технические условия, составляющие полное техническое задание (ТЗ):

1. Динамическое представление информационной структуры. Каждая отдельная запись должна быть описана структурой фиксированных символьных массивов для предотвращения неопределенного поведения памяти. Набор полей включает: ФИО, номер телефона, email, город и категорию группы. Количество записей должно быть динамическим, без жестких ограничений, что обязывает использовать структуру линейного односвязного списка.

2. Алгоритм добавления с верификацией входных данных. Ключевым идентификатором выступает номер телефона. Для исключения дублирования информации перед выделением памяти под новый узел должен запускаться сквозной поиск по всей базе. При обнаружении совпадения операция добавления блокируется.

3. Разработка низкоуровневой ассемблерной подсистемы. Для оперативного учета объема базы данных используется глобальная переменная-счетчик. Операция инкремента данного счетчика должна

выполняться внешней ассемблерной процедурой, использующей регистры общего назначения ЦП.

4. Алгоритм направленного удаления элементов. Программа должна удалять любую выбранную запись по ключевому полю. Алгоритм обязан корректно обрабатывать перестройку ссылок для головного, промежуточного и единственного узлов списка с обязательным освобождением памяти через free.

5. Модификация (редактирование) данных. Возможность изменения полей существующего контакта, найденного по телефону. Интерфейс должен поддерживать функцию сохранения данных по умолчанию при вводе пустой строки.

6. Многокритериальный контекстный поиск. Механизм поиска подстрок должен осуществлять фильтрацию базы данных по частичному совпадению текстового запроса с полями ФИО или номера телефона контактов.

7. Алгоритмическая алфавитная сортировка. Разработка подсистемы упорядочивания элементов списка по лексикографическому возрастанию поля ФИО внутренними средствами программы.

8. Категорийная фильтрация. Программа должна осуществлять проход по списку и отображать на экране исключительно те записи, поле группы которых полностью эквивалентно заданному пользователем критерию.

9. Бинарная файловая подсистема. Обеспечение сохранения текущего состояния динамического списка в бинарный файл на жестком диске и последующего циклического считывания с полной деаллокацией старой сессии.

10. Гарантированная деаллокация ресурсов при выходе. При выходе из программы должен запускаться каскадный деструктор динамической памяти, последовательно очищающий кучу от всех созданных узлов во избежание утечек.

Выбор решения

Для реализации сложной многокомпонентной системы «Телефонная книга» применен метод нисходящего проектирования и программирования. Сущность данного подхода заключается в первоначальном определении глобальной архитектуры приложения и верхнеуровневых интерфейсов управления, с последующим пошаговым спуском к детальной реализации конкретных функций манипуляции ссылками.

1 Архитектура главного цикла и механизм управления

Управление программным комплексом реализовано в рамках парадигмы событийно-ориентированного консольного интерфейса. Логика распределения вычислений сосредоточена в модуле main.c. Для обеспечения поддержки кириллицы на этапе инициализации выполняются вызовы SetConsoleCP(1251), SetConsoleOutputCP(1251) и setlocale(LC_ALL, "Russian"). Центральным диспетчером является бесконечный цикл while(1), внутри которого развернута многовариантная структура switch-case. Важным архитектурным решением является интеграция защитной функции clear_input(), выполняющей принудительное чтение и уничтожение остаточных символов в буфере потока ввода, что предотвращает бесконечное зацикливание при вводе некорректных данных.

2 Схема данных и структура динамического списка

Базовым элементом информационной системы является линейный односвязный список. Логическая структура узла описывается пользовательским типом данных Contact, содержащим символьные буферы фиксированной длины для ФИО, телефона, email, города, группы, а также указатель struct Contact* next. Глобальный контроль осуществляется через указатель начала head и счетчик db_count. Топология связей имеет вид

однонаправленной цепочки (таблица 1). Новый элемент всегда вставляется в позицию головы списка, что оптимизирует скорость операции добавления до константного времени O(1).

Таблица 1 – Схема динамической структуры данных в оперативной памяти

head	—>	Contact 1 [data | next]	—>	Contact 2 [data |NULL]

Описание разработки программы

Разработка программного комплекса осуществлялась на принципах строгого модульного проектирования, что позволило изолировать контексты компиляции и распределить обязанности между компонентами.

1 Модульная структура проекта

– Интерфейсный заголовочный файл phonebook.h – включает макроопределения, описание шаблона структуры Contact и объявления глобальных переменных с модификатором extern для предотвращения ошибок дублирования символов линкера.

– Управляющий модуль main.c – содержит точку входа main(), выполняет инициализацию параметров консоли и бесконечный цикл меню.

– Модуль бизнес-логики phonebook.c – ядро базы данных, содержащее физическую реализацию алгоритмов управления списком, поиска, сортировки и бинарного ввода-вывода.

– Низкоуровневый модуль asm.c – специализированный модуль, содержащий процедуру asm_increment, написанную с использованием встроенного ассемблера для архитектуры x86.

2 Реализация ключевых подсистем и алгоритмов

Для хранения данных был выбран динамический список, реализованный на основе динамического выделения памяти с помощью функции malloc. Такой подход обеспечивает высокую гибкость структуры: объем используемой памяти строго соответствует количеству записей в базе данных, что позволяет эффективно использовать ресурсы системы и динамически изменять размер БД без необходимости предварительного резервирования фиксированного объема памяти.

При добавлении записи данные считываются с помощью безопасной функции read_line (на базе fgets), заменяющей символ новой строки на нулевой

терминатор. Ключевой особенностью реализации счетчика общего количества записей является использование ассемблерной вставки asm_increment. Данная оптимизация позволяет выполнять операцию инкремента переменной непосредственно на уровне регистров центрального процессора (используя инструкцию inc регистра EAX), минуя промежуточные стадии компиляции стандартных высокоуровневых операций, что демонстрирует глубокий подход к управлению низкоуровневыми ресурсами ЭВМ.

В функции сортировки применен метод «пузырька». С целью повышения надежности реализован метод обмена полезным содержимым узлов (payload swap) с помощью memcpy без изменения указателей связей next, что предотвращает потерю целостности списка при перестановке элементов.

.

Отладка и тестирование

1 Методы и средства отладки

Процесс отладки выполнялся в среде Microsoft Visual Studio с использованием точек останова (Breakpoints), окон инспекции Watch и анализа кучи. Особое внимание уделялось предотвращению утечек памяти при деаллокации (контролировалось сохранение адреса следующего узла во временную переменную перед очисткой текущего) и верификации ассемблерного кода в режиме Disassembly View с контролем регистров.

2 Результаты тестирования граничных условий

Для проверки отказоустойчивости проведен ряд функциональных тестов (таблица 2).

Таблица 2 – Результаты функционального тестирования модулей программы

Название теста	**Входные данные**	**Ожидаемый результат**	**Статус**
Тест пустой структуры	head == NULL	Сообщение 'База пустая', блокировка обхода	Успешно
Тест уникальности ключа	Существующий номер телефона	Блокировка транзакции, вызов free()	Успешно
Тест отказоустойчивости ввода	Буквенная строка вместо цифры	Очистка буфера stdin, сброс зацикливания	Успешно
Тест дефекта файлов	Отсутствие файла на диске	fopen() возвращает NULL, защита сессии	Успешно

Описание программы

1 Функциональное назначение и системные требования

Программа предназначена для автоматизации процессов учета, поиска, сортировки и долговременного хранения контактных данных абонентов в локальном режиме. Системные требования: ОС Windows XP и выше, процессор архитектуры x86, ОЗУ от 16 МБ, свободное дисковое пространство от 1 МБ.

2 Спецификация программных модулей и функций

– clear_input() – посимвольная очистка входного буфера потока stdin до знака переноса строки.

– wait_and_clear() – организация задержки экрана до нажатия Enter и последующий вызов очистки консоли.

– read_line() – безопасное чтение строковых данных с помощью fgets() с контролем размера буфера.

– find_by_phone() – функция линейного поиска узла по ключевому полю номера телефона.

– add_record() – подсистема динамического добавления элементов с контролем уникальности ключа.

– delete_record() – удаление записи по номеру телефона с корректной перелинковкой указателей списка.

– sort_records() – алфавитная пузырьковая сортировка списка методом поблочного копирования памяти.

– save_db() / load_db() – бинарная сериализация и десериализация информационной базы на диск.

– free_db() – каскадный деструктор полной очистки динамической памяти связного списка.

3 Текстовое описание ключевых алгоритмов программы

3.1 Алгоритм главной управляющей функции main()

1. Блок «Пуск»: Запуск процесса выполнения программы операционной системой.

2. Блок «Процесс»: Вызов функций SetConsoleCP(1251) и SetConsoleOutputCP(1251) для кодовых страниц.

3. Блок «Процесс»: Инициализация локали стандартной библиотеки вызовом setlocale(LC_ALL, "Russian").

4. Начало цикла while(1) (Точка возврата диспетчера интерфейса меню).

5. Блок «Процесс»: Вызов подпрограммы menu(), отображающей перечень доступных команд.

6. Блок «Ввод/Вывод»: Считывание числового значения в переменную choice через поток stdin.

7. Блок «Процесс»: Вызов защитной процедуры очистки входного буфера clear_input().

8. Блок «Решение» (Оператор switch-case для выбора пункта меню 0-10): при выборе 1-10 вызываются соответствующие подпрограммы с последующим break и возвратом к шагу 4. При выборе '0' осуществляется переход к шагу 9.

9. Блок «Процесс»: Вызов деструктора динамической памяти списка free_db() перед закрытием.

10. Блок «Останов»: Возврат кода 0, завершение работы процесса и закрытие консольного окна.

Графическая схема алгоритма функции main представлена на рисунке 1.

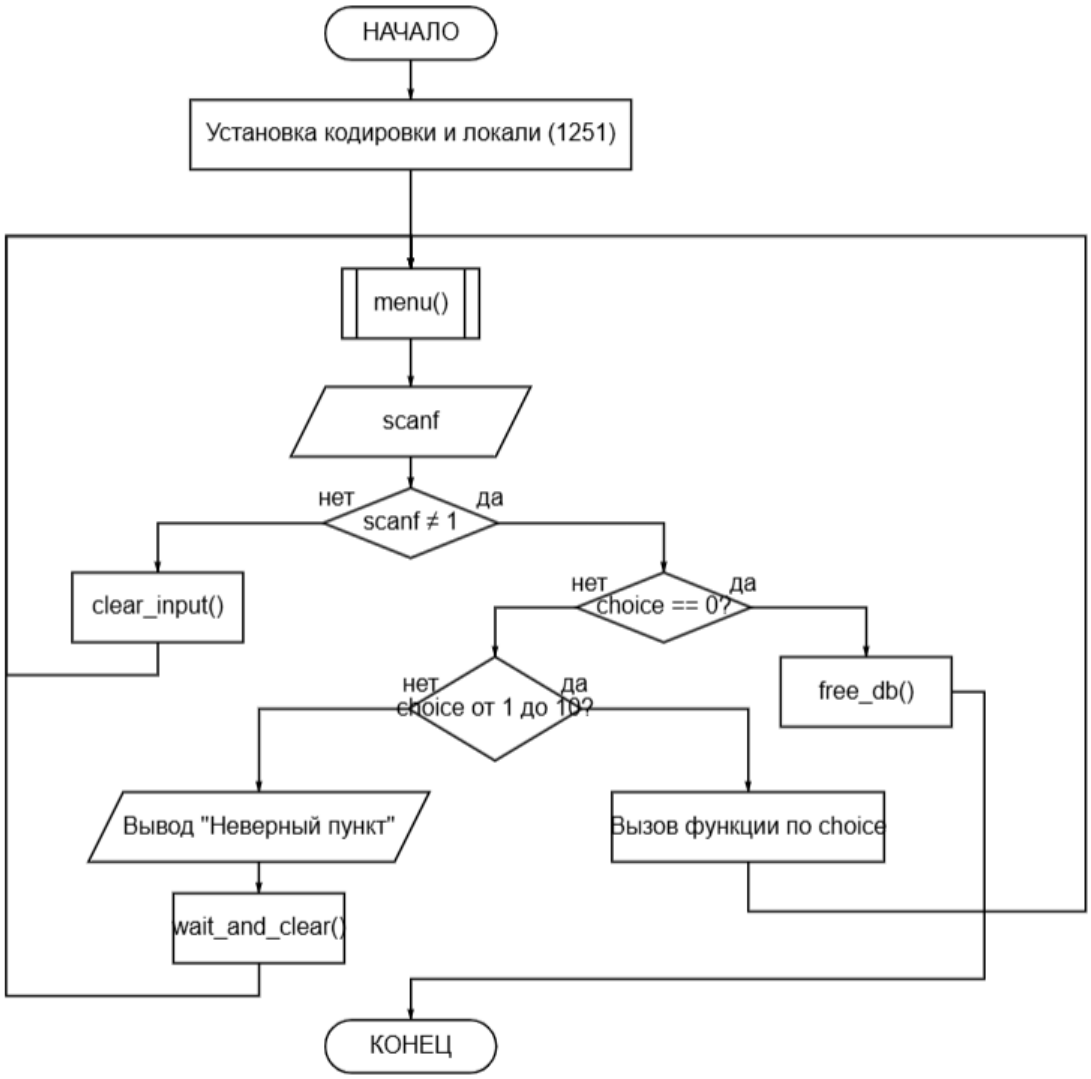

Рисунок 1 – Блок-схема функции main()

3.2 Алгоритм подсистемы добавления записи add_record()

1. Блок «Пуск»: Инициализация подпрограммы добавления контакта.

2. Блок «Процесс»: Выделение памяти под новый узел: new_node = malloc(sizeof(Contact)).

3. Блок «Решение»: Если new_node == NULL, то переход к шагу 9 (Выход), иначе переход к шагу 4.

4. Блок «Ввод/Вывод»: Интерактивный последовательный запрос и считывание всех полей структуры.

5. Блок «Процесс»: Вызов верификатора уникальности ключа: target = find_by_phone(new_node->phone).

6. Блок «Решение»: Если target != NULL (дубликат), вывод ошибки, вызов free(new_node) и переход к шагу 9.

7. Блок «Процесс»: Интеграция узла в начало связного списка: new_node->next = head; head = new_node;.

8. Блок «Процесс»: Передача управления в ассемблерный модуль: db_count = asm_increment(db_count).

9. Блок «Останов»: Вызов wait_and_clear() и завершение работы подпрограммы добавления.

Графическая схема алгоритма функции добавления записи представлена на рисунке 2.

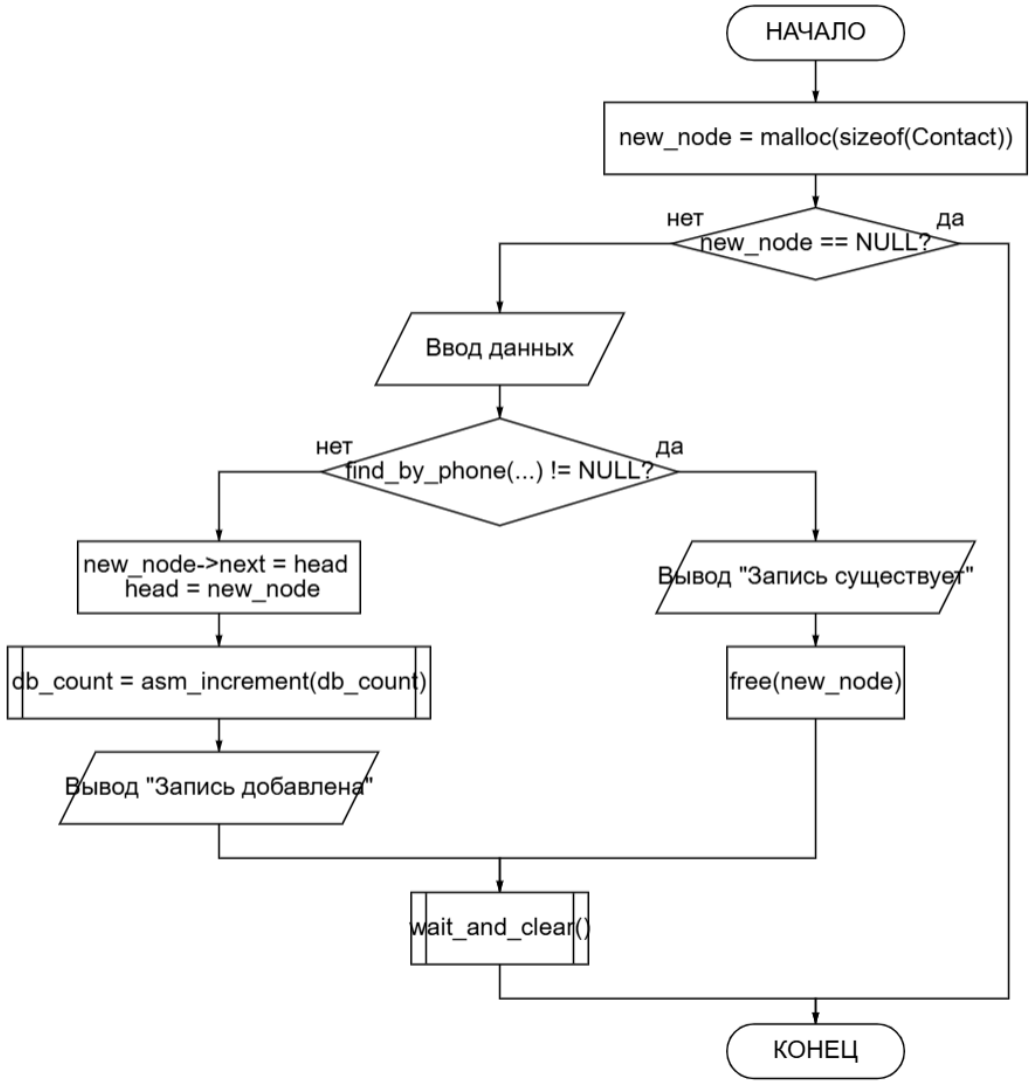

Рисунок 2 – Блок-схема функции add_record()

3.3 Алгоритм функции последовательного удаления записи delete_record()

1.Блок «Пуск»: Инициализация подпрограммы удаления контакта.

2. Блок «Ввод/Вывод»: Запрос у пользователя номера телефона для удаления.

3. Блок «Процесс»: Инициализация временных указателей поиска: curr = head, prev = NULL.

4. Начало цикла while (curr != NULL).

5. Блок «Решение»: Проверка условия равенства строк. Если истинно — переход к шагу 6, если ложно — к шагу 7.

6. Блок «Решение»: Проверка удаления из головы списка. Если истинно: head = curr->next. Если ложно: prev->next = curr->next. Переход к шагу 8.

7. Блок «Процесс»: Сдвиг указателей по списку: prev = curr; curr = curr->next;. Возврат к шагу 4.

8. Блок «Процесс»: Освобождение памяти удаленного узла free(curr) и декремент счетчика записей.

9. Блок «Останов»: Вывод сообщения об успешном удалении и выход из функции.

Графическая схема алгоритма функции удаления записи представлена на рисунке 3.

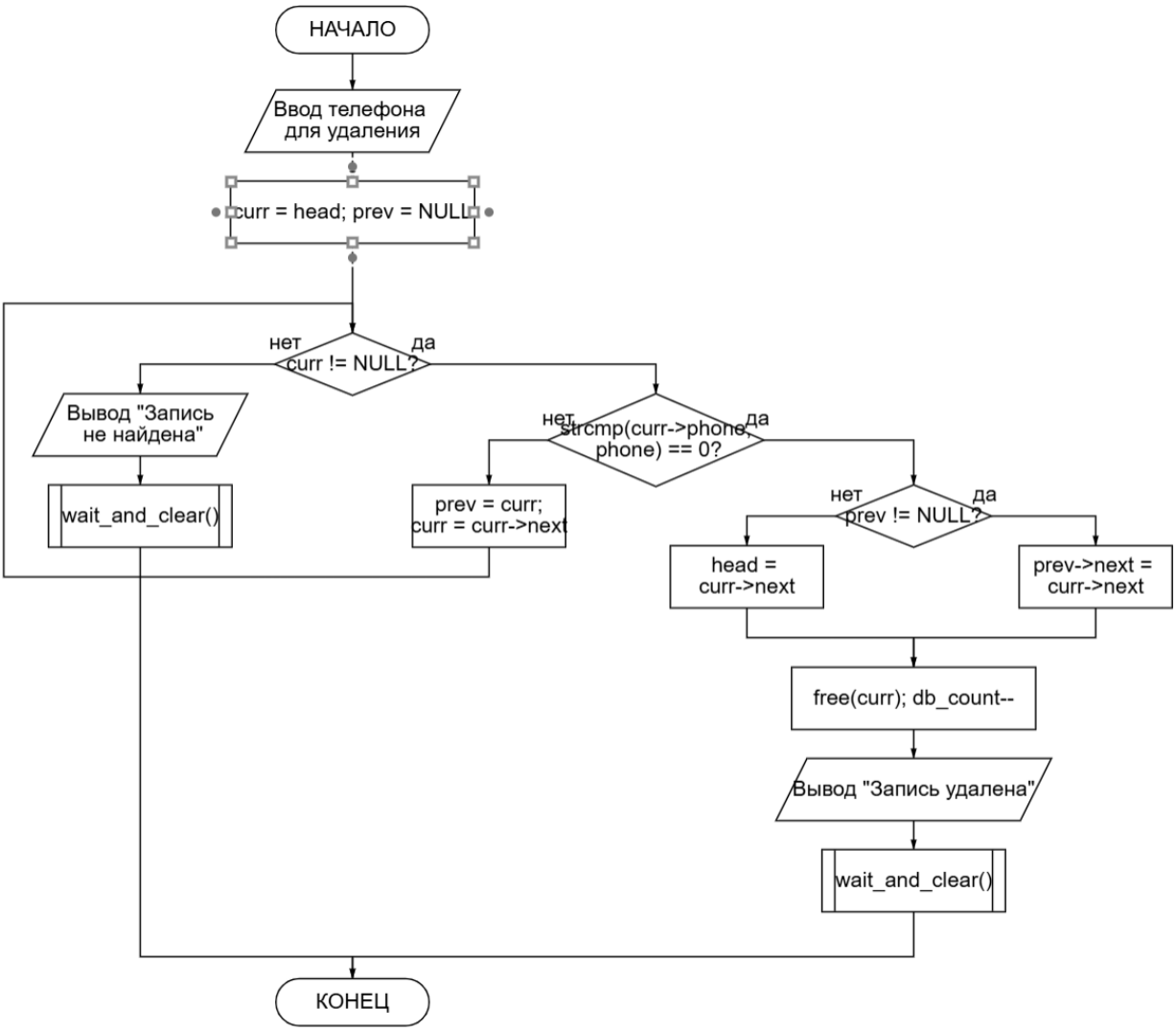

Рисунок 3 – Блок-схема функции delete_record()

Руководство пользователя

1 Порядок запуска приложения

Запуск осуществляется открытием исполняемого скомпилированного файла phonebook.exe из графической оболочки проводника Windows либо посредством командной строки терминала.

2 Инструкция по выполнению основных операций

Взаимодействие оператора с базой данных построено через интерактивный ввод номеров пунктов меню:

– Добавление контакта (Пункт 1) – последовательный ввод ФИО, телефона, email, города и группы. Номер телефона должен быть уникальным.

– Удаление контакта (Пункт 2) – запрос уникального ключевого поля (номера телефона) и автоматическая перестройка указателей с очисткой памяти.

– Редактирование (Пункт 3) – модификация полей выбранного по телефону контакта. Оставление поля пустым сохраняет старое значение.

– Поиск и Фильтр (Пункты 4 и 6) – контекстный поиск подстроки в ФИО/телефоне или выборка контактов по точному совпадению группы.

– Сохранение и Загрузка (Пункты 8 и 9) – бинарный ввод-вывод структуры данных на жесткий диск с контролем дескрипторов файлов.

Заключение

В ходе выполнения курсовой работы были полностью решены задачи проектирования локальной СУБД на базе динамических структур данных. Реализованная модель односвязного списка показала высокую масштабируемость и эффективное использование оперативной памяти. Интеграция высокоуровневого кода C с низкоуровневым Ассемблером позволила успешно оптимизировать критические вычислительные операции. Разработанный комплекс обладает модульностью, отказоустойчивостью интерфейса и полностью готов к эксплуатации.

Список используемых источников

1. Керниган, Б. В. Язык программирования С [Текст] / Б. В. Керниган, Д. М. Ритчи. – М. : Финансы и статистика, 1985. – 279 с.

2. Зубков, С. В. Assembler для DOS, Windows и UNIX [Текст] / С. В. Зубков. – М. : ДМК Пресс, 2004. – 608 с.

3. Прата, С. Язык программирования С. Лекции и упражнения [Текст] : пер. с англ. / С. Прата. – М. : ООО «И.Д. Вильямс», 2015. – 928 с.

4. Юров, В. И. Assembler: учебник для вузов [Текст] / В. И. Юров. – СПб. : Питер, 2003. – 637 с..

5. Таненбаум, Э. Архитектура компьютера [Текст] / Э. Таненбаум, Т. Остин. – СПб. : Питер, 2013. – 816 с..

6. Microsoft Learn. Встроенный ассемблер (C) [Электронный ресурс] : официальная документация Microsoft по компилятору MSVC. – Режим доступа: https://learn.microsoft.com/ru-ru/cpp/c-language/inline-assembler-c

7. ГОСТ 19.003-80. Единая система программной документации. Схемы алгоритмов и программ. Обозначения условные графические [Текст]. – Введ. 01.01.1981. – М. : Изд-во стандартов, 2005. – 12 с.