Министерство образования и науки Российской Федерации

ФГБОУ ВО «Пензенский государственный университет»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Пенза 2025

**Пояснительная записка**

к курсовой работе

по курсу «ПРОГРАММИРОВАНИЕ»

на тему «База данных “Телефонный справочник”»

Выполнил: студент группы 24ВВВ1

Марушкин Д.И.

Проверил: к.т.н., доцент

Токарев А.Н.

Генералова А.А

**Содержание**

[1. Постановка задачи 5](#_Toc198421621)

[2. Выбор решения 6](#_Toc198421622)

[3. Описание разработки программы 7](#_Toc198421623)

[4. Отладка и тестирование 8](#_Toc198421624)

[5. Описание программы 9](#_Toc198421625)

[5.1 Схема данных 9](#_Toc198421626)

[5.2 Схема функции main 9](#_Toc198421627)

[5.3 Описание модулей 10](#_Toc198421628)

[6. Руководство пользователя 15](#_Toc198421629)

[6.1 Информационная записка (заставка) 15](#_Toc198421630)

[6.2 Главное меню 15](#_Toc198421631)

[6.3 Вывод БД 16](#_Toc198421632)

[6.4 Добавление контакта 16](#_Toc198421633)

[6.5 Удаление контакта 17](#_Toc198421634)

[6.6 Редактирование контакта 18](#_Toc198421635)

[6.7 Сохранение БД в файл 19](#_Toc198421636)

[6.8 Загрузка БД из файла 20](#_Toc198421637)

[6.9 Сортировка по алфавиту А-Я 20](#_Toc198421638)

[6.10 Поиск контакта по номеру или по имени 21](#_Toc198421639)

[6.11 Смена языка программы 21](#_Toc198421640)

[6.12 Отображение справочной информации 22](#_Toc198421641)

[6.13 Выход из программы 22](#_Toc198421642)

[Заключение 23](#_Toc198421643)

[Список используемых материалов 24](#_Toc198421644)

[Приложение А. Листинг программы. 25](#_Toc198421645)

[Файл main.cpp 25](#_Toc198421646)

[Файл utils.h 28](#_Toc198421647)

[Файл utils.cpp 29](#_Toc198421648)

[Файл language.h 30](#_Toc198421649)

[Файл language.cpp 31](#_Toc198421650)

[Файл contact\_manager.h 37](#_Toc198421651)

[Файл contact\_manager.cpp 38](#_Toc198421652)

[Файл contact.h 49](#_Toc198421653)

[Файл contact.cpp 50](#_Toc198421654)

[Файл colors.h 51](#_Toc198421655)

[Файл assembly.h 52](#_Toc198421656)

[Файл assembly.cpp 53](#_Toc198421657)

**Введение**

Компьютерные технологии уже давно укоренились в нашей жизни. Многие привычные для нас вещи были бы невозможны без программирования.

Язык программирования Си++ - это универсальный язык с богатым набором операторов и компактным способом записи выражений. Благодаря гибкости, выразительности и компактности своих конструкций. Си++ завоевал наибольшую популярность в среде профессиональных программистов и широко используется при разработке системных и прикладных программ.

В данной работе продемонстрированы базовые возможности языка СИ++ вкупе с ассемблерными вставками. Разработка велась в среде Microsoft Visual Studio 2022.

1. Постановка задачи

Необходимо разработать программу для работы с базой данных по телефонному справочнику. Программа должна иметь интуитивно понятный интерфейс для пользователя. Необходимо определить названия пунктов меню, построить алгоритмы работы с массивами структур для фильтрации и сортировки записей по определенным критериям, изучить функции для работы с файлами, чтобы сохранять базу данных в файл и загружать из файла.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы.

Использование сложных типов данных – массивов, структур, файлов.

Режим работы видеосистемы – текстовый/графический. Для начала необходимо определиться с типом интерфейса и с элементами управления, затем необходимо изучить способы их реализации. Устройство ввода-вывода – клавиатура и мышь. Необходимо различать и идентифицировать действия, произведенные с их помощью, это облегчит использование программы.

Интерфейс должен быть построен на основе меню. Это необходимо для создания интуитивно понятного интерфейса для пользователя.

2. Выбор решения

При запуске программы открывается приветственный текст. Далее появляется главное меню, состоящее из 11 пунктов:

1. **add** (Добавить номер). Пользователь вводит данные, необходимые для нового элемента базы данных в порядке, указанном в программе. Все данные вводятся по мере их запроса.

2. **list**(Отобразить список номеров). На экране будут отображаться все введенные данные.

3. **remove**(Удалить данные телефона). Пользователь вводит **id** номера телефона, которое отображается рядом с именем, и программа находит и удаляет выбранный номер телефона из списка.

4. **edit**(Редактирование данных номера). Пользователь вводит **id** номера телефона, далее программа открывает его для повторной записи, при этом отображая старые данные.

5. **save**(Сохранение данных в файл). Программа шифрует и сохраняет данные в файл 'contacts.bin', который находится в каталоге программы. Если файла не существует, то программа создаст его.

6. **load**(Загрузка данных из файла). Программа открывает файл 'contacts.bin' из каталога программы и загружает из него данные. Если файла нет, то на экране отобразится ошибка.

7 **sort[asc/desc]**(Сортировка по алфавиту). Пользователь выбирает отображение списка номеров от А-Я или Я-А.

8 **search**(Поиск по номеру или имени). Пользователь может найти часть номера или часть имени и программа покажет совпадения. (Чувствителен к регистру)

9 **language[en/ru]**(Смена языка программы). Пользователь выбирает язык программы, при вводе дополнительного слова через пробел слова en или ru.

10 **exit**(Выход из программы). Пользователь может выйти из программы.

11 **help**(Показать описание команд). Пользователь может посмотреть справочную информацию по назначения команд в главном меню.

3. Описание разработки программы

Для написания данной программы будет использован язык программирования Си++ (C++). Этот язык является одним из самых распространённых и универсальных языков программирования. При его разработке был найден компромисс между низкоуровневым языком ассемблера и высокоуровневыми языками, что позволило объединить эффективность и контроль над ресурсами с удобством и выразительностью.

C++ предоставляет множество операций, которые напрямую поддерживаются большинством микропроцессоров, а также широкие возможности для описания алгоритмов и структур данных наиболее подходящими средствами. Язык обеспечивает поддержку как процедурного, так и объектно-ориентированного программирования, позволяя создавать масштабируемые и структурированные программные системы с возможностью модульной разработки и повторного использования кода.

Как и в языке Си, в C++ сохраняется высокая степень переносимости. Условная компиляция и гибкая работа с платформозависимым кодом позволяют адаптировать программы под различные операционные системы и аппаратные архитектуры, что делает язык особенно удобным для кроссплатформенной разработки. Таким образом, C++ является мощным инструментом для создания как системного, так и прикладного программного обеспечения.

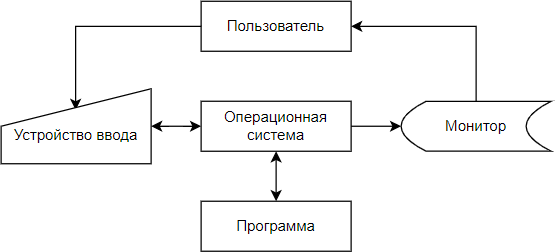
В качестве среды программирования был выбран программный продукт Visual Studio 2022.

4. Отладка и тестирование

В качестве среды разработки использовалась Microsoft Visual Studio 2022. Эта интегрированная среда предоставляет все необходимые инструменты для эффективной разработки и отладки программ. В процессе отладки применялись различные функции Visual Studio, включая установку точек останова, пошаговое выполнение кода, трассировку и просмотр содержимого переменных.

Тестирование программы проводилось как в процессе разработки, так и после её завершения. В ходе работы были обнаружены и устранены многочисленные ошибки, связанные с обработкой файлов, управлением памятью, работой с полями структур, а также с некорректными размерами массивов и другими аспектами реализации.

5. Описание программы

5.1 Схема данных

5.2 Схема функции main

5.3 Описание модулей

1. **main.cpp** – основной файл программы.  
    1.1) **main()** – функция, реализующая главное меню, обработку пользовательских команд и взаимодействие с классом ContactManager.
2. **contact\_manager.h** – заголовочный файл с объявлением класса ContactManager, его методов и подключением необходимых заголовков (contact.h, utils.h, language.h, assembly.h).
3. **contact\_manager.cpp** – реализация методов класса ContactManager для управления контактами.  
    3.1) **addContact()** – добавляет новый контакт (имя, телефон, описание).  
    3.2) **removeContact()** – удаляет контакт по индексу.  
    3.3) **editContact()** – редактирует данные существующего контакта.  
    3.4) **displayContacts()** – выводит список всех контактов с учетом текущей сортировки.  
    3.5) **saveToFile()** – сохраняет контакты в файл, шифруя поле описания.  
    3.6) **loadFromFile()** – загружает контакты из файла с расшифровкой описания.  
    3.7) **setSortOrder()** – задаёт порядок сортировки (возрастание или убывание).  
    3.8) **searchContacts()** – ищет контакты по имени или телефону без учёта регистра.  
    3.9) **sortContactsIfNeeded()** – приватный метод для сортировки контактов с учётом локали и порядка.
4. **contact.h** – заголовочный файл с объявлением класса Contact и его методов.
5. **contact.cpp** – реализация методов класса Contact.  
    5.1) **Contact()** – конструктор, инициализирующий имя, телефон и описание.  
    5.2) **edit()** – изменяет данные контакта.  
    5.3) **getUpperFirstCharOfName()** – возвращает первую букву имени в верхнем регистре, используя ассемблер для MSVC.  
    5.4) **display()** – выводит данные контакта с форматированием и цветами.
6. **language.h** – заголовочный файл с объявлением перечисления Language, глобальных переменных и функций локализации.
7. **language.cpp** – реализация функций локализации сообщений.  
    7.1) **getMessage()** – возвращает локализованное сообщение по идентификатору.  
    7.2) **initMessages()** – инициализирует словарь сообщений для английского и русского языков.
8. **utils.h** – заголовочный файл с объявлениями вспомогательных функций и глобальных констант.
9. **utils.cpp** – реализация утилитарных функций.  
    9.1) **clearConsole()** – очищает консоль с помощью ANSI-последовательностей.  
    9.2) **isValidPhoneNumber()** – проверяет корректность номера телефона (допустимые символы и минимальное количество цифр).  
    9.3) **getStringInput()** – запрашивает строковый ввод с цветным промптом.  
    9.4) **getIndexInput()** – запрашивает числовой индекс с валидацией и цветным промптом.
10. **colors.h** – заголовочный файл с макросами ANSI-последовательностей для цветного вывода в консоль.
11. **assembly.h** – заголовочный файл с объявлениями функций с ассемблерными вставками.
12. **assembly.cpp** – реализация ассемблерных функций для 32-битного MSVC.  
     12.1) **toUpper\_cp1251\_asm()** – преобразует символ в верхний регистр с учётом кодировки CP1251 (латиница и кириллица).  
     12.2) **xorEncryptDecrypt\_asm()** – шифрует/дешифрует данные методом XOR с ключом.

12.1) **toUpper\_cp1251\_asm()** – функция преобразует символ из нижнего регистра в верхний с учетом кодировки CP1251. Это необходимо для корректной обработки латинских и русских букв в именах контактов при сортировке или отображении.

char toUpper\_cp1251\_asm(char c)

{

char upper\_c = c;

\_\_asm

{

mov al, c

cmp al, 'a'

jl check\_russian

cmp al, 'z'

jg check\_russian

sub al, 32

jmp end\_conversion

check\_russian:

cmp al, 0xE0

jl check\_russian\_r

cmp al, 0xF0

jg check\_russian\_r

sub al, 0x20

jmp end\_conversion

check\_russian\_r:

cmp al, 0xF2

jl check\_russian\_yo

sub al, 0x20

jmp end\_conversion

check\_russian\_yo:

cmp al, 0xB1

jne end\_conversion

sub al, 0x10

end\_conversion:

mov upper\_c, al

}

return upper\_c;

}

12.2) **xorEncryptDecrypt\_asm()** – функция выполняет шифрование или дешифрование строки методом XOR с использованием ключа. Это необходимо для защиты поля описания контакта при сохранении в файл и его расшифровки при загрузке.

void xorEncryptDecrypt\_asm(char\* data, size\_t data\_size, const char\* key, size\_t key\_size)

{

if (data\_size == 0 || key\_size == 0) return;

\_\_asm

{

mov esi, data

mov ecx, data\_size

mov edi, key

mov edx, key\_size

xor ebx, ebx

encrypt\_loop:

cmp ecx, 0

jle end\_loop

cmp ebx, edx

jl key\_index\_ok

xor ebx, ebx

key\_index\_ok:

mov al, [edi + ebx]

xor [esi], al

inc esi

inc ebx

dec ecx

jmp encrypt\_loop

end\_loop:

}

}

6. Руководство пользователя

Программа phoneList.exe предназначена для хранения и управления информацией о контактах. Программа имеет интуитивно понятный консольный интерфейс и поддерживает такие операции, как добавление новой записи в базу данных, удаление записи из базы данных, изменение записи в базе данных, поиск записи в базе данных, сортировка базы данных, сохранение существующей базы данных в файл, импорт базы данных из файла, сортированный вывод записей базы данных и переключение языка интерфейса. Дополнительно программа обеспечивает шифрование данных при сохранении и использует ассемблерные вставки для оптимизации обработки символов и шифрования.

Далее БД – база данных

6.1 Информационная записка (заставка)

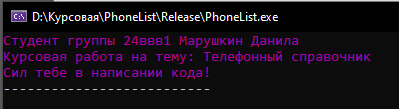
****

Рисунок 1. Заставка

6.2 Главное меню

Главное меню.

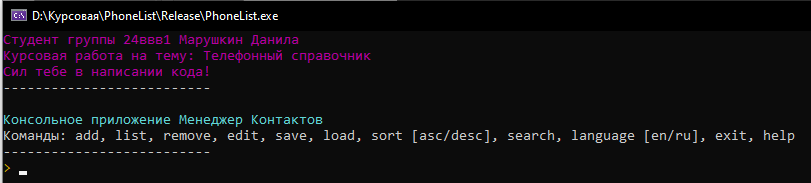
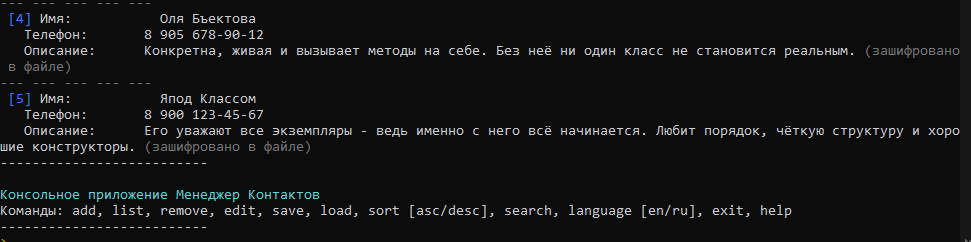


Рисунок 2. Главное меню

6.3 Вывод БД

Для открытия и вывода на экран существующего БД пользователю необходимо выбрать пункт меню «list»

Рисунок 3. Вывод БД

6.4 Добавление контакта

Для добавления записи в каталог пользователю необходимо выбрать пункт меню «add».

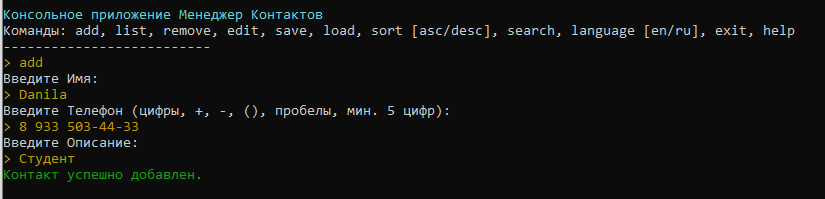


Рисунок 4. Добавление контакта

Проверяем с помощью пункта «list».

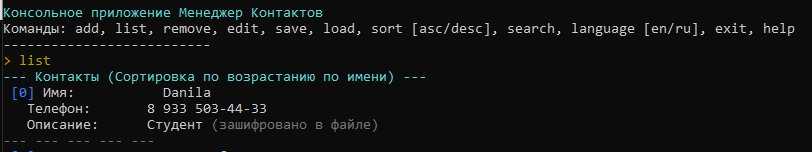


Рисунок 5. Полученный результат

6.5 Удаление контакта

Для удаления записи из каталога пользователю необходимо выбрать пункт меню «remove». Далее необходимо ввести личный номер студента удаляемой записи и нажать клавишу Enter.

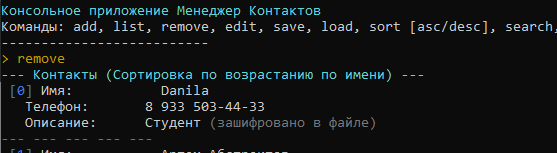
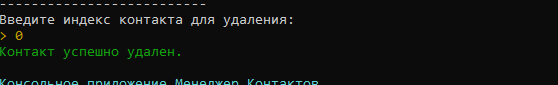


Рисунок 6. Удаление контакта

Проверяем с помощью пункта «list».

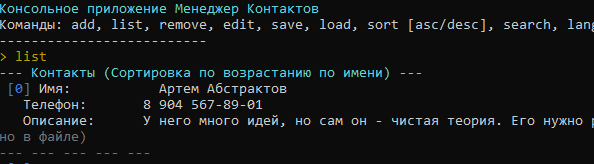
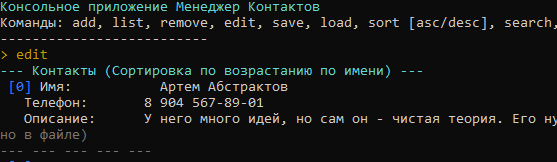
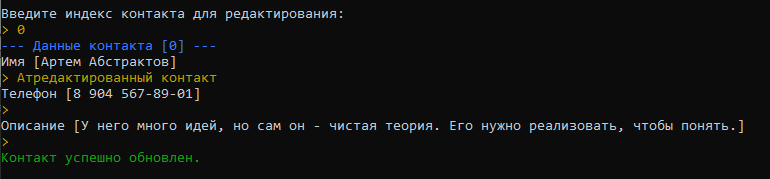


Рисунок 7. Полученный результат

6.6 Редактирование контакта

Для редактирования контакта необходимо выбрать пункт меню «edit». Далее необходимо ввести личный номер студента удаляемой записи и нажать клавишу Enter. 

Рисунок 8. Редактирование контакта

Проверяем с помощью пункта «list»

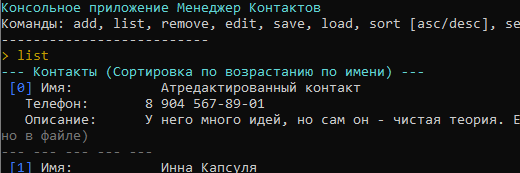


Рисунок 9. Полученный результат

6.7 Сохранение БД в файл

Для сохранения в файл использую команду «save»

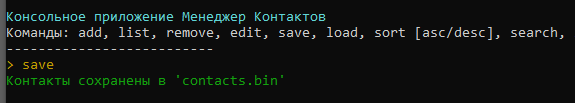


Рисунок 10. Сохранение результата в файл

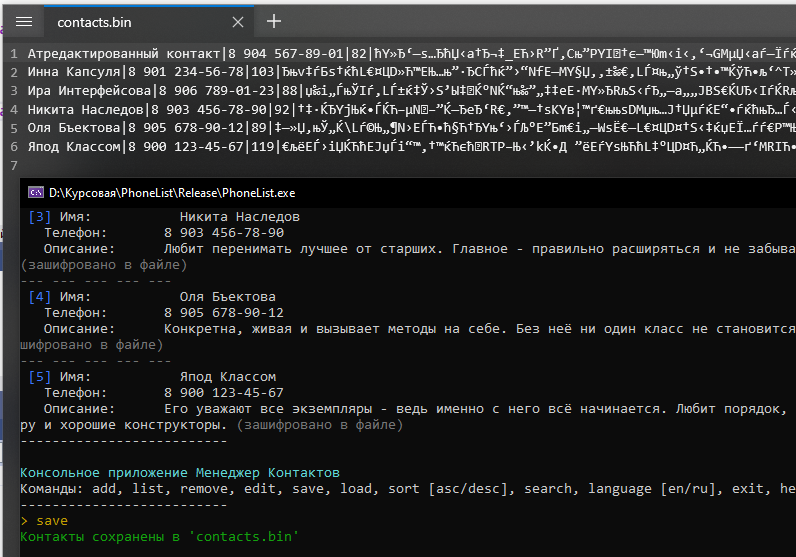


Рисунок 11. Полученный результат

6.8 Загрузка БД из файла

Для загрузки из файла использую команду «load»

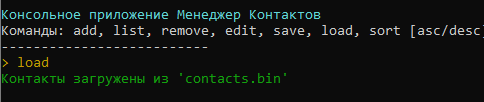
****

Рисунок 12. Загрузка контактов из файла contacts.bin

6.9 Сортировка по алфавиту А-Я

Для сортировки программы использую команду «sort»

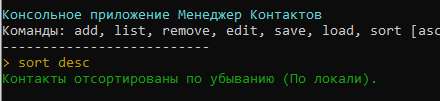


Рисунок 13. Выбор сортировки по убыванию

Для проверки результата использую команду «list»

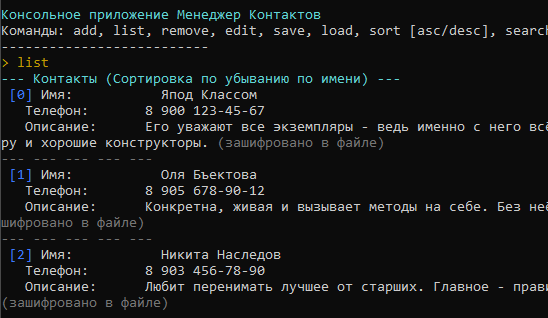


Рисунок 14. Полученный результат

6.10 Поиск контакта по номеру или по имени

Для поиска контакта использую команду «search» и указываю часть имени или номера

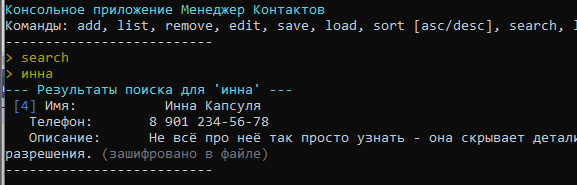


Рисунок 15. Поиск по имени.

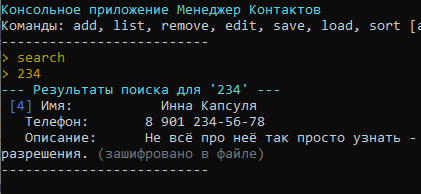


Рисунок 16. Поиск по части номера телефона

6.11 Смена языка программы

Для смены языка программы воспользуюсь командой «language» и через пробел введу желаемый язык

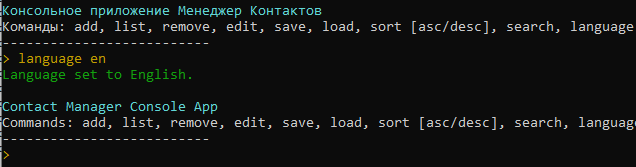


Рисунок 17. Смена языка на английский

6.12 Отображение справочной информации

Для отображения справочной информации использую команду «help»

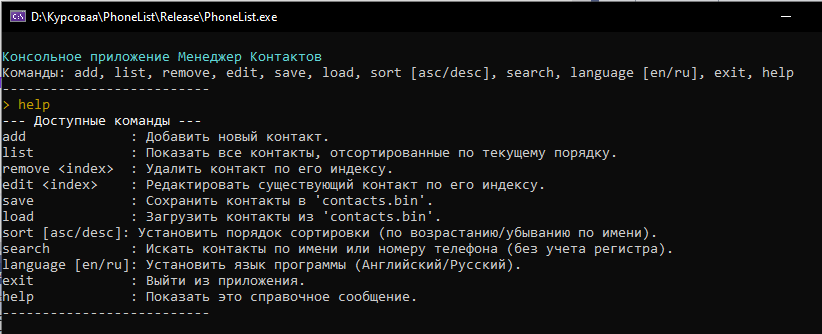


Рисунок 18. Отображение справочной информации

6.13 Выход из программы

Для выхода из программы использую команду «exit»

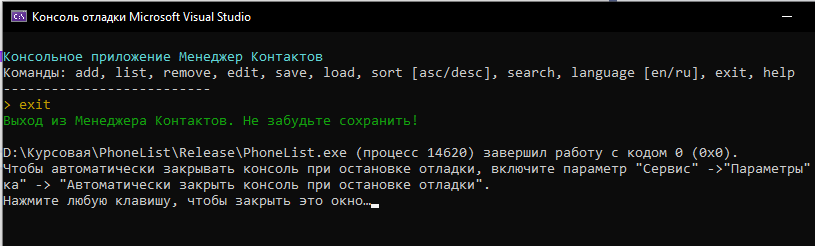


Рисунок 19. Результат работы команды «exit»

Заключение

В процессе выполнения данной курсовой работы были приобретены навыки разработки многомодульных программ на языках C++ и Assembler. Были изучены и применены методы работы с файлами, включая чтение и запись данных с использованием шифрования, а также функции для взаимодействия с консолью и управления оперативной памятью. Особое внимание было уделено реализации ассемблерных вставок для оптимизации обработки символов и шифрования данных. Кроме того, были освоены основы локализации интерфейса, отладки и тестирования программ в среде Visual Studio 2022.

В рамках курсовой работы была разработана программа phoneList.exe для управления базой данных контактов. Она включает все необходимые функции для работы с базой данных, такие как добавление, удаление, редактирование, поиск, сортировка, сохранение и загрузка данных, а также поддержку русского и английского языков интерфейса.

В дальнейшем программу можно усовершенствовать путем внедрения графического интерфейса, что упростит навигацию и сделает взаимодействие с программой более удобным за счет использования мыши. Также возможно расширение функциональности, например, добавлением фильтрации контактов по дополнительным критериям или интеграцией с другими форматами данных.

Список используемых материалов

1. YouTube канал «SimpleCode», «Основы С++. Программирование для начинающих»

<https://www.youtube.com/@SimpleCode>

2. Роберт Мартин. «Чистый код: создание, анализ и рефакторинг». 2019 г.

3. Кэти Сьерра и Берт Бейтс. «Изучаем JAVA» 2022 г.

4. Б. Страуструп. «Программирование. Принципы и практика с использованием C++»

5. Скотт Мейерс. «Эффективный и современный C++»

6. **Документация языка C++ от Microsoft (MSVC)**  
 <https://learn.microsoft.com/cpp>

Приложение А. Листинг программы.

Файл main.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#define NOMINMAX

#ifdef \_WIN32

#include <Windows.h>

#endif

#include <iostream>

#include <string>

#include <locale>

#include <algorithm>

#include <cctype>

#include "contact\_manager.h"

#include "language.h"

#include "utils.h"

#include "colors.h"

int main() {

#ifdef \_WIN32

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

#endif

try {

std::locale::global(std::locale(""));

}

catch (const std::runtime\_error& e) {

std::cerr << ANSI\_YELLOW << "Warning: Could not set locale. Sorting might not work correctly for non-ASCII characters." << ANSI\_RESET << std::endl;

std::cerr << ANSI\_YELLOW << "Error details: " << e.what() << ANSI\_RESET << std::endl;

}

catch (...) {

std::cerr << ANSI\_YELLOW << "Warning: Could not set locale due to an unknown error. Sorting might not work correctly." << ANSI\_RESET << std::endl;

}

initMessages();

ContactManager manager;

manager.loadFromFile();

clearConsole();

// --- Блок вывода сплеша - С ЦВЕТАМИ ---

std::cout << ANSI\_BOLD << ANSI\_MAGENTA;

std::cout << getMessage("MSG\_SPLASH\_AUTHOR") << std::endl;

std::cout << getMessage("MSG\_SPLASH\_PROJECT") << std::endl;

std::cout << getMessage("MSG\_SPLASH\_ENCOURAGEMENT") << std::endl;

std::cout << ANSI\_RESET;

std::cout << "--------------------------" << std::endl;

std::string command;

bool running = true;

while (running) {

// --- Блок вывода приветствия и команд - С ЦВЕТАМИ ---

std::cout << std::endl << ANSI\_BOLD << ANSI\_CYAN;

std::cout << getMessage("MSG\_WELCOME") << std::endl;

std::cout << ANSI\_RESET;

std::cout << getMessage("MSG\_COMMANDS") << std::endl;

std::cout << "--------------------------" << std::endl;

// --- Запрос команды - С ЦВЕТОМ ПРОМПТА ---

std::cout << ANSI\_YELLOW << INPUT\_PREFIX;

std::cout.flush();

std::string line\_input;

std::getline(std::cin, line\_input);

std::cout << ANSI\_RESET;

command = line\_input;

std::string lowerCommand = command;

try {

const std::locale& loc = std::locale("");

std::transform(lowerCommand.begin(), lowerCommand.end(), lowerCommand.begin(),

[&](char c) { return std::tolower(c, loc); });

}

catch (...) {

std::transform(lowerCommand.begin(), lowerCommand.end(), lowerCommand.begin(), ::tolower);

}

if (lowerCommand == "add") {

manager.addContact();

}

else if (lowerCommand == "list") {

manager.displayContacts();

}

else if (lowerCommand == "remove") {

manager.removeContact();

}

else if (lowerCommand == "edit") {

manager.editContact();

}

else if (lowerCommand == "save") {

manager.saveToFile();

}

else if (lowerCommand == "load") {

manager.loadFromFile();

}

else if (lowerCommand.rfind("sort", 0) == 0) {

std::string order = lowerCommand.substr(4);

size\_t first = order.find\_first\_not\_of(' ');

if (std::string::npos == first) { order = ""; }

else {

size\_t last = order.find\_last\_not\_of(' ');

order = order.substr(first, (last - first + 1));

}

if (order.empty()) {

manager.displayContacts();

}

else {

manager.setSortOrder(order);

}

}

else if (lowerCommand == "search") {

std::string searchTerm = getStringInput("MSG\_ENTER\_SEARCH\_TERM");

manager.searchContacts(searchTerm);

}

else if (lowerCommand.rfind("language", 0) == 0) {

std::string langCode = lowerCommand.substr(8);

size\_t first = langCode.find\_first\_not\_of(' ');

if (std::string::npos == first) { langCode = ""; }

else {

size\_t last = langCode.find\_last\_not\_of(' ');

langCode = langCode.substr(first, (last - first + 1));

}

if (langCode == "en") {

currentLanguage = Language::English;

std::cout << ANSI\_GREEN << getMessage("MSG\_LANG\_SET\_TO") << getMessage("MSG\_LANG\_ENGLISH") << ANSI\_RESET << std::endl;

}

else if (langCode == "ru") {

currentLanguage = Language::Russian;

std::cout << ANSI\_GREEN << getMessage("MSG\_LANG\_SET\_TO") << getMessage("MSG\_LANG\_RUSSIAN") << ANSI\_RESET << std::endl;

}

else {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_INVALID\_LANG") << ANSI\_RESET << std::endl;

}

}

else if (lowerCommand == "exit") {

std::cout << ANSI\_GREEN << getMessage("MSG\_EXITING") << ANSI\_RESET << std::endl;

running = false;

}

else if (lowerCommand == "help") {

std::cout << ANSI\_BOLD << getMessage("MSG\_HELP\_TITLE") << ANSI\_RESET << std::endl;

std::cout << getMessage("MSG\_HELP\_ADD") << std::endl;

std::cout << getMessage("MSG\_HELP\_LIST") << std::endl;

std::cout << getMessage("MSG\_HELP\_REMOVE") << std::endl;

std::cout << getMessage("MSG\_HELP\_EDIT") << std::endl;

std::cout << getMessage("MSG\_HELP\_SAVE") << std::endl;

std::cout << getMessage("MSG\_HELP\_LOAD") << std::endl;

std::cout << getMessage("MSG\_HELP\_SORT") << std::endl;

std::cout << getMessage("MSG\_HELP\_SEARCH") << std::endl;

std::cout << getMessage("MSG\_HELP\_LANGUAGE") << std::endl;

std::cout << getMessage("MSG\_HELP\_EXIT") << std::endl;

std::cout << getMessage("MSG\_HELP\_HELP") << std::endl;

std::cout << getMessage("MSG\_HELP\_END") << std::endl;

}

else {

std::cerr << ANSI\_RED << getMessage("MSG\_UNKNOWN\_COMMAND") << ANSI\_RESET << std::endl;

}

}

return 0;

}

Файл utils.h

#ifndef UTILS\_H

#define UTILS\_H

#define NOMINMAX

#include <string>

#include <iostream>

#include <vector>

#include <algorithm>

#include <cctype>

#include <limits>

#include <stdexcept>

#include <ios>

#ifdef \_WIN32

#include <Windows.h>

#endif

#include <cstdlib>

extern const std::string DEFAULT\_FILENAME;

extern const size\_t MIN\_PHONE\_DIGITS;

extern const std::string INPUT\_PREFIX;

void clearConsole();

bool isValidPhoneNumber(const std::string& phone);

std::string getStringInput(const std::string& promptMessageId);

bool getIndexInput(const std::string& promptMessageId, size\_t maxIndex, size\_t& index);

#endif

Файл utils.cpp

#include "utils.h"

#include "language.h"

#include "colors.h"

const std::string DEFAULT\_FILENAME = "contacts.bin";

const size\_t MIN\_PHONE\_DIGITS = 5;

const std::string INPUT\_PREFIX = "> ";

void clearConsole() {

std::cout << "\033[2J\033[1;1H";

std::cout.flush();

}

bool isValidPhoneNumber(const std::string& phone) {

size\_t digitCount = 0;

for (char c : phone) {

if (std::isdigit(static\_cast<unsigned char>(c))) {

digitCount++;

}

else if (c != '+' && c != '-' && c != '(' && c != ' ') {

return false;

}

}

return digitCount >= MIN\_PHONE\_DIGITS;

}

std::string getStringInput(const std::string& promptMessageId) {

std::string input;

std::cout << ANSI\_YELLOW << INPUT\_PREFIX;

std::cout.flush();

std::getline(std::cin, input);

std::cout << ANSI\_RESET;

return input;

}

bool getIndexInput(const std::string& promptMessageId, size\_t maxIndex, size\_t& index) {

std::string input\_str;

unsigned long temp\_index;

while (true) {

std::cout << ANSI\_YELLOW << INPUT\_PREFIX;

std::cout.flush();

std::getline(std::cin, input\_str);

std::cout << ANSI\_RESET;

try {

if (input\_str.empty() || std::find\_if(input\_str.begin(), input\_str.end(), [](unsigned char c) { return !std::isdigit(c); }) != input\_str.end()) {

throw std::invalid\_argument("Not a number");

}

temp\_index = std::stoul(input\_str);

index = static\_cast<size\_t>(temp\_index);

if (index < maxIndex) {

return true;

}

else {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_INVALID\_INDEX") << ANSI\_RESET << std::endl;

}

}

catch (const std::invalid\_argument&) {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_INVALID\_INPUT\_NUMBER") << ANSI\_RESET << std::endl;

}

catch (const std::out\_of\_range&) {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_INVALID\_INPUT\_NUMBER") << " (Number too large)." << ANSI\_RESET << std::endl;

}

}

}

Файл language.h

#ifndef LANGUAGE\_H

#define LANGUAGE\_H

#include <string>

#include <map>

#include <vector>

#include <locale>

enum class Language { English, Russian };

extern Language currentLanguage;

extern std::map<std::string, std::map<Language, std::string>> messages;

std::string getMessage(const std::string& id);

void initMessages();

#endif

Файл language.cpp

#include "language.h"

#include "utils.h"

Language currentLanguage = Language::Russian;

std::map<std::string, std::map<Language, std::string>> messages;

std::string getMessage(const std::string& id) {

auto it = messages.find(id);

if (it != messages.end()) {

auto lang\_it = it->second.find(currentLanguage);

if (lang\_it != it->second.end()) {

return lang\_it->second;

}

lang\_it = it->second.find(Language::English);

if (lang\_it != it->second.end()) {

return lang\_it->second;

}

}

return "MISSING\_MESSAGE: " + id;

}

void initMessages() {

// === Английские сообщения ===

messages["MSG\_SPLASH\_AUTHOR"][Language::English] = "Student of group 24vvv1 Marushkin Danila";

messages["MSG\_SPLASH\_PROJECT"][Language::English] = "Course work on the topic: Phone Directory";

messages["MSG\_SPLASH\_ENCOURAGEMENT"][Language::English] = "Good luck with coding!";

messages["MSG\_WELCOME"][Language::English] = "Contact Manager Console App";

messages["MSG\_COMMANDS"][Language::English] = "Commands: add, list, remove, edit, save, load, sort [asc/desc], search, language [en/ru], exit, help";

messages["MSG\_ENTER\_COMMAND"][Language::English] = "Enter command";

messages["MSG\_ENTER\_NAME"][Language::English] = "Enter Name:";

messages["MSG\_ENTER\_PHONE"][Language::English] = "Enter Phone (digits, +, -, (), spaces allowed, min " + std::to\_string(MIN\_PHONE\_DIGITS) + " digits):";

messages["MSG\_ENTER\_DESCRIPTION"][Language::English] = "Enter Description:";

messages["MSG\_CONTACT\_ADDED\_SUCCESS"][Language::English] = "Contact added successfully.";

messages["MSG\_ERR\_NAME\_EMPTY"][Language::English] = "Error: Contact name cannot be empty.";

messages["MSG\_ERR\_PHONE\_INVALID"][Language::English] = "Error: Invalid phone number format or too few digits.";

messages["MSG\_ENTER\_INDEX\_REMOVE"][Language::English] = "Enter index of contact to remove:";

messages["MSG\_CONTACT\_REMOVED\_SUCCESS"][Language::English] = "Contact removed successfully.";

messages["MSG\_ERR\_INVALID\_INDEX"][Language::English] = "Error: Invalid contact index.";

messages["MSG\_ERR\_INVALID\_INPUT\_NUMBER"][Language::English] = "Invalid input. Please enter a number.";

messages["MSG\_ENTER\_INDEX\_EDIT"][Language::English] = "Enter index of contact to edit:";

messages["MSG\_CONTACT\_UPDATED\_SUCCESS"][Language::English] = "Contact updated successfully.";

messages["MSG\_ERR\_COULD\_NOT\_RETRIEVE\_CONTACT"][Language::English] = "Error: Could not retrieve contact data for editing.";

messages["MSG\_ERR\_FILE\_SAVE"][Language::English] = "Error: Could not open file '" + DEFAULT\_FILENAME + "' for saving.";

messages["MSG\_CONTACTS\_SAVED"][Language::English] = "Contacts saved to '";

messages["MSG\_ERR\_FILE\_LOAD"][Language::English] = "Error: Could not open file '" + DEFAULT\_FILENAME + "' for loading. File might not exist.";

messages["MSG\_WARN\_MALFORMED\_LINE"][Language::English] = "Warning: Skipping malformed line in file: ";

messages["MSG\_WARN\_MALFORMED\_ENCRYPTED\_DATA"][Language::English] = "Warning: Skipping malformed encrypted data in file line: ";

messages["MSG\_CONTACTS\_LOADED"][Language::English] = "Contacts loaded from '";

messages["MSG\_ENTER\_SORT\_ORDER"][Language::English] = "Enter sort order (asc/desc):";

messages["MSG\_SORTED\_ASC"][Language::English] = "Contacts sorted in ascending order (Locale default).";

messages["MSG\_SORTED\_DESC"][Language::English] = "Contacts sorted in descending order (Locale default).";

messages["MSG\_ERR\_INVALID\_SORT\_ORDER"][Language::English] = "Invalid sort order. Use 'asc' or 'desc'.";

messages["MSG\_WARN\_LOCALE\_SORT"][Language::English] = "Warning: Current locale might not support proper character collation. Sorting might be incorrect for some characters.";

messages["MSG\_ERR\_LOCALE\_SORT"][Language::English] = "Locale error during sorting: ";

messages["MSG\_ERR\_UNKNOWN\_SORT"][Language::English] = "An unknown error occurred during sorting.";

messages["MSG\_ENTER\_SEARCH\_TERM"][Language::English] = "Enter search term (name or phone):";

messages["MSG\_NO\_CONTACTS\_FOUND\_SEARCH"][Language::English] = "No contacts found matching '";

messages["MSG\_SEARCH\_RESULTS\_FOR"][Language::English] = "--- Search Results for '";

messages["MSG\_EXITING"][Language::English] = "Exiting Contact Manager. Don't forget to save!";

messages["MSG\_UNKNOWN\_COMMAND"][Language::English] = "Unknown command. Type 'help' for a list of commands.";

messages["MSG\_HELP\_TITLE"][Language::English] = "--- Available Commands ---";

messages["MSG\_HELP\_ADD"][Language::English] = "add : Add a new contact.";

messages["MSG\_HELP\_LIST"][Language::English] = "list : Display all contacts, sorted by current order.";

messages["MSG\_HELP\_REMOVE"][Language::English] = "remove <index> : Remove a contact by its index.";

messages["MSG\_HELP\_EDIT"][Language::English] = "edit <index> : Edit an existing contact by its index.";

messages["MSG\_HELP\_SAVE"][Language::English] = "save : Save contacts to '" + DEFAULT\_FILENAME + "'.";

messages["MSG\_HELP\_LOAD"][Language::English] = "load : Load contacts from '" + DEFAULT\_FILENAME + "'.";

messages["MSG\_HELP\_SORT"][Language::English] = "sort [asc/desc]: Set sort order (ascending/descending by name).";

messages["MSG\_HELP\_SEARCH"][Language::English] = "search : Search contacts by name or phone (case-insensitive).";

messages["MSG\_HELP\_LANGUAGE"][Language::English] = "language [en/ru]: Set program language (English/Russian).";

messages["MSG\_HELP\_EXIT"][Language::English] = "exit : Выйти из приложения."; // Пропущено в оригинале, исправил.

messages["MSG\_HELP\_HELP"][Language::English] = "help : Show this help message.";

messages["MSG\_HELP\_END"][Language::English] = "--------------------------";

messages["MSG\_LANG\_SET\_SUCCESS"][Language::English] = "Language set successfully.";

messages["MSG\_LANG\_SET\_TO"][Language::English] = "Language set to ";

messages["MSG\_LANG\_ENGLISH"][Language::English] = "English.";

messages["MSG\_LANG\_RUSSIAN"][Language::English] = "Russian.";

messages["MSG\_ERR\_INVALID\_LANG"][Language::English] = "Invalid language code. Use 'en' for English or 'ru' for Russian.";

messages["MSG\_CONTACTS\_TITLE\_SORT"][Language::English] = "--- Contacts (Sorted ";

messages["MSG\_CONTACTS\_TITLE\_SORT\_ASC"][Language::English] = "Ascending";

messages["MSG\_CONTACTS\_TITLE\_SORT\_DESC"][Language::English] = "Descending";

messages["MSG\_CONTACTS\_TITLE\_BY\_NAME"][Language::English] = " by Name) ---";

messages["MSG\_NO\_CONTACTS\_FOUND\_LIST"][Language::English] = "No contacts found.";

messages["MSG\_ASM\_CASE\_CONVERSION\_TEST"][Language::English] = "ASM CP1251 Case Conversion Test";

messages["MSG\_ENCRYPTED"][Language::English] = " (encrypted in file)";

messages["MSG\_INDEX"][Language::English] = "Index";

messages["MSG\_CONTACT\_DETAILS"][Language::English] = "Contact Details";

messages["MSG\_FIELD\_NAME"][Language::English] = "Name";

messages["MSG\_FIELD\_PHONE"][Language::English] = "Phone";

messages["MSG\_FIELD\_DESC"][Language::English] = "Description";

// === Русские сообщения === (скопированы из оригинала)

messages["MSG\_SPLASH\_AUTHOR"][Language::Russian] = "Студент группы 24ввв1 Марушкин Данила";

messages["MSG\_SPLASH\_PROJECT"][Language::Russian] = "Курсовая работа на тему: Телефонный справочник";

messages["MSG\_SPLASH\_ENCOURAGEMENT"][Language::Russian] = "Сил тебе в написании кода!";

messages["MSG\_WELCOME"][Language::Russian] = "Консольное приложение Менеджер Контактов";

messages["MSG\_COMMANDS"][Language::Russian] = "Команды: add, list, remove, edit, save, load, sort [asc/desc], search, language [en/ru], exit, help";

messages["MSG\_ENTER\_COMMAND"][Language::Russian] = "Введите команду";

messages["MSG\_ENTER\_NAME"][Language::Russian] = "Введите Имя:";

messages["MSG\_ENTER\_PHONE"][Language::Russian] = "Введите Телефон (цифры, +, -, (), пробелы, мин. " + std::to\_string(MIN\_PHONE\_DIGITS) + " цифр):";

messages["MSG\_ENTER\_DESCRIPTION"][Language::Russian] = "Введите Описание:";

messages["MSG\_CONTACT\_ADDED\_SUCCESS"][Language::Russian] = "Контакт успешно добавлен.";

messages["MSG\_ERR\_NAME\_EMPTY"][Language::Russian] = "Ошибка: Имя контакта не может быть пустым.";

messages["MSG\_ERR\_PHONE\_INVALID"][Language::Russian] = "Ошибка: Неверный формат номера телефона или слишком мало цифр.";

messages["MSG\_ENTER\_INDEX\_REMOVE"][Language::Russian] = "Введите индекс контакта для удаления:";

messages["MSG\_CONTACT\_REMOVED\_SUCCESS"][Language::Russian] = "Контакт успешно удален.";

messages["MSG\_ERR\_INVALID\_INDEX"][Language::Russian] = "Ошибка: Неверный индекс контакта.";

messages["MSG\_ERR\_INVALID\_INPUT\_NUMBER"][Language::Russian] = "Неверный ввод. Пожалуйста, введите число.";

messages["MSG\_ENTER\_INDEX\_EDIT"][Language::Russian] = "Введите индекс контакта для редактирования:";

messages["MSG\_CONTACT\_UPDATED\_SUCCESS"][Language::Russian] = "Контакт успешно обновлен.";

messages["MSG\_ERR\_COULD\_NOT\_RETRIEVE\_CONTACT"][Language::Russian] = "Ошибка: Не удалось получить данные контакта для редактирования.";

messages["MSG\_ERR\_FILE\_SAVE"][Language::Russian] = "Ошибка: Не удалось открыть файл '" + DEFAULT\_FILENAME + "' для сохранения.";

messages["MSG\_CONTACTS\_SAVED"][Language::Russian] = "Контакты сохранены в '";

messages["MSG\_ERR\_FILE\_LOAD"][Language::Russian] = "Ошибка: Не удалось открыть файл '" + DEFAULT\_FILENAME + "' для загрузки. Файл может не существовать.";

messages["MSG\_WARN\_MALFORMED\_LINE"][Language::Russian] = "Предупреждение: Пропущена некорректная строка в файле: ";

messages["MSG\_WARN\_MALFORMED\_ENCRYPTED\_DATA"][Language::Russian] = "Предупреждение: Пропущены некорректные зашифрованные данные в строке файла: ";

messages["MSG\_CONTACTS\_LOADED"][Language::Russian] = "Контакты загружены из '";

messages["MSG\_ENTER\_SORT\_ORDER"][Language::Russian] = "Введите порядок сортировки (asc/desc):";

messages["MSG\_SORTED\_ASC"][Language::Russian] = "Контакты отсортированы по возрастанию (По локали).";

messages["MSG\_SORTED\_DESC"][Language::Russian] = "Контакты отсортированы по убыванию (По локали).";

messages["MSG\_ERR\_INVALID\_SORT\_ORDER"][Language::Russian] = "Неверный порядок сортировки. Используйте 'asc' или 'desc'.";

messages["MSG\_WARN\_LOCALE\_SORT"][Language::Russian] = "Предупреждение: Текущая локаль может не поддерживать корректное сопоставление символов. Сортировка может быть неверной.";

messages["MSG\_ERR\_LOCALE\_SORT"][Language::Russian] = "Ошибка локали при сортировке: ";

messages["MSG\_ERR\_UNKNOWN\_SORT"][Language::Russian] = "Произошла неизвестная ошибка при сортировке.";

messages["MSG\_ENTER\_SEARCH\_TERM"][Language::Russian] = "Введите поисковый запрос (имя или телефон):";

messages["MSG\_NO\_CONTACTS\_FOUND\_SEARCH"][Language::Russian] = "Контакты, соответствующие запросу '";

messages["MSG\_SEARCH\_RESULTS\_FOR"][Language::Russian] = "--- Результаты поиска для '";

messages["MSG\_EXITING"][Language::Russian] = "Выход из Менеджера Контактов. Не забудьте сохранить!";

messages["MSG\_UNKNOWN\_COMMAND"][Language::Russian] = "Неизвестная команда. Введите 'help' для списка команд.";

messages["MSG\_HELP\_TITLE"][Language::Russian] = "--- Доступные команды ---";

messages["MSG\_HELP\_ADD"][Language::Russian] = "add : Добавить новый контакт.";

messages["MSG\_HELP\_LIST"][Language::Russian] = "list : Показать все контакты, отсортированные по текущему порядку.";

messages["MSG\_HELP\_REMOVE"][Language::Russian] = "remove <index> : Удалить контакт по его индексу.";

messages["MSG\_HELP\_EDIT"][Language::Russian] = "edit <index> : Редактировать существующий контакт по его индексу.";

messages["MSG\_HELP\_SAVE"][Language::Russian] = "save : Сохранить контакты в '" + DEFAULT\_FILENAME + "'.";

messages["MSG\_HELP\_LOAD"][Language::Russian] = "load : Загрузить контакты из '" + DEFAULT\_FILENAME + "'.";

messages["MSG\_HELP\_SORT"][Language::Russian] = "sort [asc/desc]: Установить порядок сортировки (по возрастанию/убыванию по имени).";

messages["MSG\_HELP\_SEARCH"][Language::Russian] = "search : Искать контакты по имени или номеру телефона (без учета регистра).";

messages["MSG\_HELP\_LANGUAGE"][Language::Russian] = "language [en/ru]: Установить язык программы (Английский/Русский).";

messages["MSG\_HELP\_EXIT"][Language::Russian] = "exit : Выйти из приложения.";

messages["MSG\_HELP\_HELP"][Language::Russian] = "help : Показать это справочное сообщение.";

messages["MSG\_HELP\_END"][Language::Russian] = "--------------------------";

messages["MSG\_LANG\_SET\_SUCCESS"][Language::Russian] = "Язык успешно установлен.";

messages["MSG\_LANG\_SET\_TO"][Language::Russian] = "Язык установлен на ";

messages["MSG\_LANG\_ENGLISH"][Language::Russian] = "Английский.";

messages["MSG\_LANG\_RUSSIAN"][Language::Russian] = "Русский.";

messages["MSG\_ERR\_INVALID\_LANG"][Language::Russian] = "Неверный код языка. Используйте 'en' для английского или 'ru' для русского.";

messages["MSG\_CONTACTS\_TITLE\_SORT"][Language::Russian] = "--- Контакты (Сортировка ";

messages["MSG\_CONTACTS\_TITLE\_SORT\_ASC"][Language::Russian] = "по возрастанию";

messages["MSG\_CONTACTS\_TITLE\_SORT\_DESC"][Language::Russian] = "по убыванию";

messages["MSG\_CONTACTS\_TITLE\_BY\_NAME"][Language::Russian] = " по имени) ---";

messages["MSG\_NO\_CONTACTS\_FOUND\_LIST"][Language::Russian] = "Контактов не найдено.";

messages["MSG\_ASM\_CASE\_CONVERSION\_TEST"][Language::Russian] = "Тест преобразования регистра (ASM)";

messages["MSG\_ENCRYPTED"][Language::Russian] = " (зашифровано в файле)";

messages["MSG\_INDEX"][Language::Russian] = "Индекс";

messages["MSG\_CONTACT\_DETAILS"][Language::Russian] = "Данные контакта";

messages["MSG\_FIELD\_NAME"][Language::Russian] = "Имя";

messages["MSG\_FIELD\_PHONE"][Language::Russian] = "Телефон";

messages["MSG\_FIELD\_DESC"][Language::Russian] = "Описание";

}

Файл contact\_manager.h

#ifndef CONTACT\_MANAGER\_H

#define CONTACT\_MANAGER\_H

#include <vector>

#include <string>

#include <fstream>

#include <algorithm>

#include <locale>

#include <iostream>

#include <stdexcept>

#include "contact.h"

#include "utils.h"

#include "language.h"

#include "assembly.h"

class ContactManager {

private:

std::vector<Contact> contacts;

bool sortByNameAscending = true;

std::string encryption\_key = "MyVerySimpleAndInsecureEncryptionKey!";

void sortContactsIfNeeded();

public:

void addContact();

void removeContact();

void editContact();

void displayContacts();

void saveToFile(const std::string& filename = DEFAULT\_FILENAME);

void loadFromFile(const std::string& filename = DEFAULT\_FILENAME);

void setSortOrder(const std::string& order);

void searchContacts(const std::string& searchTerm);

};

#endif

Файл contact\_manager.cpp

#include "contact\_manager.h"

#include "contact.h"

#include "utils.h"

#include "language.h"

#include "assembly.h"

#include "colors.h" // <-- Добавляем включение файла с цветами

#include <iomanip> // Нужен для std::setw (используется в Contact::display)

#include <iostream> // std::cerr, std::cout, std::endl

#include <algorithm> // std::transform, std::sort, std::find\_if

#include <locale> // std::locale, std::collate, std::use\_facet

#include <fstream> // std::ifstream, std::ofstream

#include <vector> // std::vector

#include <stdexcept> // Стандартные исключения

#include <cstddef> // size\_t

// Ключ шифрования определен в заголовочном файле как const член класса

// Реализация addContact (Сообщения о успехе/ошибке с цветом)

void ContactManager::addContact() {

std::string name, phone, description;

// --- Ввод Имени ---

while (true) {

std::cout << getMessage("MSG\_ENTER\_NAME") << std::endl;

name = getStringInput("MSG\_ENTER\_NAME");

if (!name.empty()) break;

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_NAME\_EMPTY") << ANSI\_RESET << std::endl;

}

// --- Ввод Телефона ---

while (true) {

std::cout << getMessage("MSG\_ENTER\_PHONE") << std::endl;

phone = getStringInput("MSG\_ENTER\_PHONE");

if (isValidPhoneNumber(phone)) break;

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_PHONE\_INVALID") << ANSI\_RESET << std::endl;

}

// --- Ввод Описания ---

std::cout << getMessage("MSG\_ENTER\_DESCRIPTION") << std::endl;

description = getStringInput("MSG\_ENTER\_DESCRIPTION");

contacts.emplace\_back(name, phone, description);

std::cout << ANSI\_GREEN << getMessage("MSG\_CONTACT\_ADDED\_SUCCESS") << ANSI\_RESET << std::endl;

}

void ContactManager::removeContact() {

if (contacts.empty()) {

std::cout << getMessage("MSG\_NO\_CONTACTS\_FOUND\_LIST") << std::endl;

return;

}

displayContacts();

size\_t index;

std::cout << getMessage("MSG\_ENTER\_INDEX\_REMOVE") << std::endl;

if (getIndexInput("MSG\_ENTER\_INDEX\_REMOVE", contacts.size(), index)) {

contacts.erase(contacts.begin() + index);

std::cout << ANSI\_GREEN << getMessage("MSG\_CONTACT\_REMOVED\_SUCCESS") << ANSI\_RESET << std::endl;

}

}

void ContactManager::editContact() {

if (contacts.empty()) {

std::cout << getMessage("MSG\_NO\_CONTACTS\_FOUND\_LIST") << std::endl;

return;

}

displayContacts();

size\_t index;

std::cout << getMessage("MSG\_ENTER\_INDEX\_EDIT") << std::endl;

if (!getIndexInput("MSG\_ENTER\_INDEX\_EDIT", contacts.size(), index)) {

return;

}

Contact& contactToEdit = contacts[index];

std::cout << ANSI\_BOLD << ANSI\_BLUE << "--- " << getMessage("MSG\_CONTACT\_DETAILS") << " [" << index << "] ---" << ANSI\_RESET << std::endl;

std::string name = contactToEdit.name;

std::string input\_line;

std::cout << getMessage("MSG\_FIELD\_NAME") << " [" << contactToEdit.name << "]" << std::endl;

input\_line = getStringInput("MSG\_ENTER\_NAME");

if (!input\_line.empty()) { name = input\_line; }

if (name.empty()) {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_NAME\_EMPTY") << " (Name not changed)." << ANSI\_RESET << std::endl;

name = contactToEdit.name;

}

std::string phone = contactToEdit.phone;

while (true) {

std::cout << getMessage("MSG\_FIELD\_PHONE") << " [" << contactToEdit.phone << "]" << std::endl;

input\_line = getStringInput("MSG\_ENTER\_PHONE");

if (input\_line.empty()) { phone = contactToEdit.phone; break; }

else {

if (isValidPhoneNumber(input\_line)) { phone = input\_line; break; }

else {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_PHONE\_INVALID") << ANSI\_RESET << std::endl;

}

}

}

std::string description = contactToEdit.description;

std::cout << getMessage("MSG\_FIELD\_DESC") << " [" << contactToEdit.description << "]" << std::endl;

input\_line = getStringInput("MSG\_ENTER\_DESCRIPTION");

if (!input\_line.empty()) { description = input\_line; }

contactToEdit.edit(name, phone, description);

std::cout << ANSI\_GREEN << getMessage("MSG\_CONTACT\_UPDATED\_SUCCESS") << ANSI\_RESET << std::endl;

}

void ContactManager::saveToFile(const std::string& filename) {

std::ofstream outFile(filename, std::ios::binary);

if (!outFile) {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_FILE\_SAVE") << ANSI\_RESET << std::endl;

return;

}

for (const auto& contact : contacts) {

std::string description\_to\_encrypt = contact.description;

#ifdef \_MSC\_VER

xorEncryptDecrypt\_asm(

const\_cast<char\*>(description\_to\_encrypt.data()),

description\_to\_encrypt.size(),

encryption\_key.data(),

encryption\_key.size()

);

#else

if (!description\_to\_encrypt.empty() && !encryption\_key.empty()) {

for (size\_t i = 0; i < description\_to\_encrypt.size(); ++i) {

description\_to\_encrypt[i] ^= encryption\_key[i % encryption\_key.size()];

}

}

#endif

outFile << contact.name << "|" << contact.phone << "|" << description\_to\_encrypt.size() << "|";

outFile.write(description\_to\_encrypt.data(), description\_to\_encrypt.size());

outFile << "\n";

}

outFile.close();

std::cout << ANSI\_GREEN << getMessage("MSG\_CONTACTS\_SAVED") << filename << "'" << ANSI\_RESET << std::endl;

}

void ContactManager::loadFromFile(const std::string& filename) {

std::ifstream inFile(filename, std::ios::binary);

if (!inFile) {

std::cerr << ANSI\_YELLOW << getMessage("MSG\_ERR\_FILE\_LOAD") << ANSI\_RESET << std::endl;

return;

}

contacts.clear();

std::string line;

size\_t line\_number = 0;

while (std::getline(inFile, line)) {

line\_number++;

if (!line.empty() && line.back() == '\r') { line.pop\_back(); }

if (line.empty()) continue;

size\_t pos1 = line.find('|');

size\_t pos2 = (pos1 == std::string::npos) ? std::string::npos : line.find('|', pos1 + 1);

size\_t pos3 = (pos2 == std::string::npos) ? std::string::npos : line.find('|', pos2 + 1);

if (pos1 == std::string::npos || pos2 == std::string::npos || pos3 == std::string::npos) {

std::cerr << ANSI\_YELLOW << getMessage("MSG\_WARN\_MALFORMED\_LINE") << "(Line " << line\_number << "): " << line << ANSI\_RESET << std::endl;

continue;

}

std::string name = line.substr(0, pos1);

std::string phone = line.substr(pos1 + 1, pos2 - pos1 - 1);

std::string encrypted\_len\_str = line.substr(pos2 + 1, pos3 - pos2 - 1);

size\_t encrypted\_len = 0;

try {

if (encrypted\_len\_str.empty() || std::find\_if(encrypted\_len\_str.begin(), encrypted\_len\_str.end(), [](unsigned char c) { return !std::isdigit(c); }) != encrypted\_len\_str.end()) {

throw std::invalid\_argument("Invalid length format");

}

encrypted\_len = static\_cast<size\_t>(std::stoul(encrypted\_len\_str));

}

catch (const std::exception& e) {

std::cerr << ANSI\_YELLOW << getMessage("MSG\_WARN\_MALFORMED\_ENCRYPTED\_DATA") << "Invalid length '" << encrypted\_len\_str << "' (Line " << line\_number << "): " << line << ". Error: " << e.what() << ANSI\_RESET << std::endl;

continue;

}

size\_t data\_start\_pos = pos3 + 1;

if (data\_start\_pos + encrypted\_len > line.size()) {

std::cerr << ANSI\_YELLOW << getMessage("MSG\_WARN\_MALFORMED\_ENCRYPTED\_DATA") << "Declared length " << encrypted\_len << " > data size " << (line.size() - data\_start\_pos) << " (Line " << line\_number << "): " << line << ANSI\_RESET << std::endl;

continue;

}

std::string encrypted\_data = line.substr(data\_start\_pos, encrypted\_len);

#ifdef \_MSC\_VER

xorEncryptDecrypt\_asm(

const\_cast<char\*>(encrypted\_data.data()),

encrypted\_data.size(),

encryption\_key.data(),

encryption\_key.size()

);

#else

if (!encrypted\_data.empty() && !encryption\_key.empty()) {

for (size\_t i = 0; i < encrypted\_data.size(); ++i) {

encrypted\_data[i] ^= encryption\_key[i % encryption\_key.size()];

}

}

#endif

std::string description = encrypted\_data;

contacts.emplace\_back(name, phone, description);

}

inFile.close();

std::cout << ANSI\_GREEN << getMessage("MSG\_CONTACTS\_LOADED") << filename << "'" << ANSI\_RESET << std::endl;

}

void ContactManager::setSortOrder(const std::string& order) {

if (order == "asc") {

sortByNameAscending = true;

std::cout << ANSI\_GREEN << getMessage("MSG\_SORTED\_ASC") << ANSI\_RESET << std::endl;

}

else if (order == "desc") {

sortByNameAscending = false;

std::cout << ANSI\_GREEN << getMessage("MSG\_SORTED\_DESC") << ANSI\_RESET << std::endl;

}

else {

std::cerr << ANSI\_RED << getMessage("MSG\_ERR\_INVALID\_SORT\_ORDER") << ANSI\_RESET << std::endl;

}

}

void ContactManager::sortContactsIfNeeded() {

try {

const std::locale& loc = std::locale("");

const std::collate<char>& coll = std::use\_facet<std::collate<char>>(loc);

std::sort(contacts.begin(), contacts.end(),

[&](const Contact& a, const Contact& b) {

int result = coll.compare(a.name.c\_str(), a.name.c\_str() + a.name.length(),

b.name.c\_str(), b.name.c\_str() + b.name.length());

return sortByNameAscending ? (result < 0) : (result > 0);

});

}

catch (const std::runtime\_error& e) {

std::cerr << ANSI\_YELLOW << getMessage("MSG\_ERR\_LOCALE\_SORT") << e.what() << ANSI\_RESET << std::endl;

std::cerr << ANSI\_YELLOW << getMessage("MSG\_WARN\_LOCALE\_SORT") << ANSI\_RESET << std::endl;

}

catch (...) {

std::cerr << ANSI\_YELLOW << getMessage("MSG\_ERR\_UNKNOWN\_SORT") << ANSI\_RESET << std::endl;

}

}

void ContactManager::displayContacts() {

sortContactsIfNeeded();

if (contacts.empty()) {

std::cout << ANSI\_YELLOW << getMessage("MSG\_NO\_CONTACTS\_FOUND\_LIST") << ANSI\_RESET << std::endl;

return;

}

std::cout << ANSI\_BOLD << ANSI\_CYAN;

std::cout << getMessage("MSG\_CONTACTS\_TITLE\_SORT");

if (sortByNameAscending) {

std::cout << getMessage("MSG\_CONTACTS\_TITLE\_SORT\_ASC");

}

else {

std::cout << getMessage("MSG\_CONTACTS\_TITLE\_SORT\_DESC");

}

std::cout << getMessage("MSG\_CONTACTS\_TITLE\_BY\_NAME") << std::endl;

std::cout << ANSI\_RESET;

// --- ЦИКЛ ВЫВОДА КОНТАКТОВ - ДОБАВЛЕНО РАЗДЕЛЕНИЕ С ЦВЕТОМ ---

for (size\_t i = 0; i < contacts.size(); ++i)

contacts[i].display(i);

if (i < contacts.size() - 1) {

std::cout << ANSI\_BRIGHT\_BLACK << "--- --- --- --- ---" << ANSI\_RESET << std::endl; // Короткая разделительная линия - темно-серым

}

}

// --- КОНЕЦ ЦИКЛА ---

std::cout << "--------------------------" << std::endl;

}

void ContactManager::searchContacts(const std::string& searchTerm) {

if (contacts.empty()) {

std::cout << ANSI\_YELLOW << getMessage("MSG\_NO\_CONTACTS\_FOUND\_LIST") << ANSI\_RESET << std::endl;

return;

}

std::string lowerSearchTerm = searchTerm;

try {

const std::locale& loc = std::locale("");

std::transform(lowerSearchTerm.begin(), lowerSearchTerm.end(), lowerSearchTerm.begin(),

[&](char c) { return std::tolower(c, loc); });

}

catch (...) {

std::transform(lowerSearchTerm.begin(), lowerSearchTerm.end(), lowerSearchTerm.begin(), ::tolower);

}

std::vector<size\_t> found\_indices;

for (size\_t i = 0; i < contacts.size(); ++i) {

std::string lowerName = contacts[i].name;

std::string lowerPhone = contacts[i].phone;

try {

const std::locale& loc = std::locale("");

std::transform(lowerName.begin(), lowerName.end(), lowerName.begin(),

[&](char c) { return std::tolower(c, loc); });

std::transform(lowerPhone.begin(), lowerPhone.end(), lowerPhone.begin(),

[&](char c) { return std::tolower(c, loc); });

}

catch (...) {

std::transform(lowerName.begin(), lowerName.end(), lowerName.begin(), ::tolower);

std::transform(lowerPhone.begin(), lowerPhone.end(), lowerPhone.begin(), ::tolower);

}

if (lowerName.find(lowerSearchTerm) != std::string::npos ||

lowerPhone.find(lowerSearchTerm) != std::string::npos) {

found\_indices.push\_back(i);

}

}

if (found\_indices.empty()) {

std::cout << ANSI\_YELLOW << getMessage("MSG\_NO\_CONTACTS\_FOUND\_SEARCH") << searchTerm << "'." << ANSI\_RESET << std::endl;

}

else {

std::cout << ANSI\_BOLD << ANSI\_CYAN << getMessage("MSG\_SEARCH\_RESULTS\_FOR") << searchTerm << "' ---" << ANSI\_RESET << std::endl;

for (size\_t index : found\_indices) {

contacts[index].display(index);

}

std::cout << "--------------------------" << std::endl;

}

}

Файл contact.h

#ifndef CONTACT\_H

#define CONTACT\_H

#include <string>

#include <iostream>

#include "assembly.h"

#include "language.h"

class Contact {

public:

std::string name;

std::string phone;

std::string description;

Contact(const std::string& \_name, const std::string& \_phone, const std::string& \_description);

void edit(const std::string& \_name, const std::string& \_phone, const std::string& \_description);

char getUpperFirstCharOfName() const;

void display(size\_t index) const;

};

#endif

Файл contact.cpp

#include "contact.h"

#include "language.h"

#include "assembly.h"

#include "colors.h"

#include <iomanip>

#include <iostream>

#include <string>

#include <cstddef>

Contact::Contact(const std::string& \_name, const std::string& \_phone, const std::string& \_description)

: name(\_name), phone(\_phone), description(\_description) {

}

void Contact::edit(const std::string& \_name, const std::string& \_phone, const std::string& \_description) {

name = \_name;

phone = \_phone;

description = \_description;

}

char Contact::getUpperFirstCharOfName() const {

if (name.empty()) return '\0';

#ifdef \_MSC\_VER

return toUpper\_cp1251\_asm(static\_cast<unsigned char>(name[0]));

#else

return std::toupper(static\_cast<unsigned char>(name[0]));

#endif

}

void Contact::display(size\_t index) const {

const int label\_width = 15;

std::cout << ANSI\_BRIGHT\_BLUE << " [" << index << "] " << ANSI\_RESET;

std::cout << ANSI\_WHITE << std::left << std::setw(label\_width) << (getMessage("MSG\_FIELD\_NAME") + ":") << ANSI\_RESET << name << std::endl;

std::cout << " ";

std::cout << ANSI\_WHITE << std::left << std::setw(label\_width) << (getMessage("MSG\_FIELD\_PHONE") + ":") << ANSI\_RESET << phone << std::endl;

std::cout << " ";

std::cout << ANSI\_WHITE << std::left << std::setw(label\_width) << (getMessage("MSG\_FIELD\_DESC") + ":") << ANSI\_RESET << description

<< ANSI\_BRIGHT\_BLACK << getMessage("MSG\_ENCRYPTED") << ANSI\_RESET << std::endl;

}

Файл colors.h

#ifndef COLORS\_H

#define COLORS\_H

// Макросы для ANSI Escape кодов

#define ANSI\_RESET "\033[0m" // Сброс всех атрибутов к стандартным

#define ANSI\_BOLD "\033[1m" // Жирный текст (или яркий)

#define ANSI\_UNDERLINE "\033[4m" // Подчеркнутый текст

// Цвета текста (Foreground Colors)

#define ANSI\_BLACK "\033[30m"

#define ANSI\_RED "\033[31m"

#define ANSI\_GREEN "\033[32m"

#define ANSI\_YELLOW "\033[33m" // Может быть коричневым в некоторых терминалах

#define ANSI\_BLUE "\033[34m"

#define ANSI\_MAGENTA "\033[35m"

#define ANSI\_CYAN "\033[36m"

#define ANSI\_WHITE "\033[37m" // Может быть светло-серым в некоторых терминалах

// Яркие цвета текста (Bright Foreground Colors)

#define ANSI\_BRIGHT\_BLACK "\033[90m" // Темно-серый

#define ANSI\_BRIGHT\_RED "\033[91m"

#define ANSI\_BRIGHT\_GREEN "\033[92m"

#define ANSI\_BRIGHT\_YELLOW "\033[93m" // Ярко-желтый

#define ANSI\_BRIGHT\_BLUE "\033[94m"

#define ANSI\_BRIGHT\_MAGENTA "\033[95m"

#define ANSI\_BRIGHT\_CYAN "\033[96m" // Ярко-голубой

#define ANSI\_BRIGHT\_WHITE "\033[97m" // Ярко-белый (настоящий белый)

#endif

Файл assembly.h

#define ASSEMBLY\_H

#include <cstdint>

#include <string>

#ifdef \_MSC\_VER

char toUpper\_cp1251\_asm(char c);

void xorEncryptDecrypt\_asm(char\* data, size\_t data\_size, const char\* key, size\_t key\_size);

#endif

Файл assembly.cpp

#include "assembly.h"

#include <cstdint>

#include <string>

#ifdef \_MSC\_VER

char toUpper\_cp1251\_asm(char c) {

char upper\_c = c;

\_\_asm {

mov al, c

cmp al, 'a'

jl check\_russian

cmp al, 'z'

jg check\_russian

sub al, 32

jmp end\_conversion

check\_russian:

cmp al, 0xE0

jl check\_russian\_r

cmp al, 0xF0

jg check\_russian\_r

sub al, 0x20

jmp end\_conversion

check\_russian\_r:

cmp al, 0xF2

jl check\_russian\_yo

sub al, 0x20

jmp end\_conversion

check\_russian\_yo:

cmp al, 0xB1

jne end\_conversion

sub al, 0x10

end\_conversion:

mov upper\_c, al

}

return upper\_c;

}

void xorEncryptDecrypt\_asm(char\* data, size\_t data\_size, const char\* key, size\_t key\_size) {

if (data\_size == 0 || key\_size == 0) return;

\_\_asm {

mov esi, data

mov ecx, data\_size

mov edi, key

mov edx, key\_size

xor ebx, ebx

encrypt\_loop:

cmp ecx, 0

jle end\_loop

cmp ebx, edx

jl key\_index\_ok

xor ebx, ebx

key\_index\_ok:

mov al, [edi + ebx]

xor [esi], al

inc esi

inc ebx

dec ecx

jmp encrypt\_loop

end\_loop:

}

}

#endif