



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

ЛАБОРАТОРНАЯ РАБОТА № 3

«Кейс "Шифровальщик сообщений" (Мессенджер)»

ДИСЦИПЛИНА: «Основы программирования»

Выполнили: студенты гр. ИУ5-14Б

_____ (Бутузов Р. А.)
(Подпись) (Ф.И.О.)

_____ (Корнеев Г. И.)
(Подпись) (Ф.И.О.)

_____ (Лимаренко Д. А.)
(Подпись) (Ф.И.О.)

_____ (Олейник В. Д.)
(Подпись) (Ф.И.О.)

_____ (Стеганцева К. М.)
(Подпись) (Ф.И.О.)

_____ (Шебордаев А. Д.)
(Подпись) (Ф.И.О.)

Проверил:

_____ (Колосов М. И.)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

2025 г.

Цель: наглядное применение транспонирования, работа с матрицей символов (char**).

Задача: реализовать простой шифр перестановки для текстового сообщения.

Подзадачи:

1. Преобразовать строку (например, "ПРИВЕТ МИР") в матрицу, записывая символы построчно. Размер матрицы выбрать так, чтобы вместить все символы.
2. Реализовать функцию encrypt (Matrix chars), которая "шифрует" сообщение, выполняя транспонирование матрицы и считывая символы по столбцам.
3. Реализовать функцию decrypt, которая выполняет обратную операцию.

Выполнение работы

Выполнено принятия задания из электронной образовательной системы МГТУ имени Н. Э. Баумана (e-learning.bmstu). Настроен VS Code для работы с GitHub, создан новый репозитория на GitHub и проведено клонирование репозитория в WSL. Результаты приведены на рисунке 1.

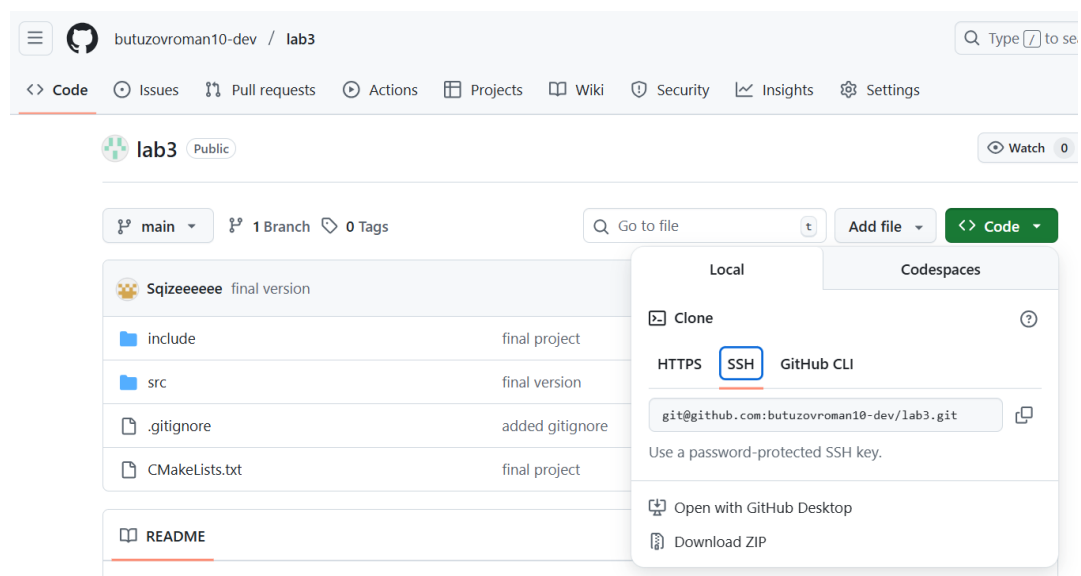


Рисунок 1 — Создание репозитория на GitHub

На рисунках 2, 3 представлен код файла main.cpp.

```
/**
 * @file main.cpp
 * @brief Основной файл приложения для тестирования шифровальщика
 * @author Ваше Имя
 * @date 2024
 *
 * Содержит тестовые функции для демонстрации работы шифровальщика
 * методом транспонирования матрицы.
 */

#include "../include/matrix.h"
#include <iostream>

/**
 * @brief Тестирует шифрование и дешифрование для заданной фразы
 *
 * Функция выполняет полный цикл:
 * 1. Преобразование текста в матрицу
 * 2. Шифрование транспонированием
 * 3. Дешифрование обратным транспонированием
 * 4. Проверка результата
 *
 * @param phrase Текст для тестирования
 * @param description Описание теста для вывода
 *
 * @see encrypt
 * @see decrypt
 * @see string_to_matrix
 */
void test_phrase(const char* phrase, const std::string& description) {
    std::cout << "\n" << std::string(50, '=') << std::endl;
    std::cout << "Тест: " << description << std::endl;
    std::cout << std::string(50, '=') << std::endl;

    int text_length = strlen(phrase);

    std::cout << "Исходный текст: '" << phrase << "' << std::endl;
    std::cout << "Длина: " << text_length << " символов" << std::endl;

    // Вычисляем размер матрицы
    int size = calculate_matrix_size(text_length);
    std::cout << "Размер матрицы: " << size << "x" << size << std::endl;

    // Создаем матрицу из строки
    Matrix original = string_to_matrix(phrase, size, size);

    std::cout << "\nИсходная матрица:" << std::endl;
    print_matrix(original);
}
```

```
// Шифруем
Matrix encrypted = encrypt(chars: original);
std::cout << "\nЗашифрованная матрица (транспонированная):" << std::endl;
print_matrix(mat: encrypted);

// Показываем шифр как в задании (чтение по столбцам исходной матрицы)
std::cout << "\nШифр (чтение по столбцам исходной матрицы): \n";
for (int j = 0; j < original.cols; j++) {
    for (int i = 0; i < original.rows; i++) {
        std::cout << original.data[i][j];
    }
    if (j < original.cols - 1) std::cout << " ";
}
std::cout << "\n" << std::endl;

// Дешифруем
std::string decrypted = decrypt(encrypted);
std::cout << "\nРасшифрованный текст: '" << decrypted << "' << std::endl;

// Проверка
if (decrypted == phrase) {
    std::cout << "✓ Шифрование и дешифрование прошли успешно!" << std::endl;
} else {
    std::cout << "✗ Ошибка в шифровании/дешифровании!" << std::endl;
    std::cout << "Ожидалось: '" << phrase << "' << std::endl;
    std::cout << "Получено: '" << decrypted << "' << std::endl;
}

// Освобождаем память
free_matrix(&m: original);
free_matrix(&m: encrypted);

/**
 * @brief Главная функция приложения
 *
 * Содержит набор тестов для демонстрации работы шифровальщика.
 *
 * @return int Код завершения программы (0 - успех, 1 - ошибка)
 */
int main() {
    try {
        std::cout << "ШИФРОВАЛЬЩИК СООБЩЕНИЙ" << std::endl;
        std::cout << "Метод: транспонирование матрицы" << std::endl;

        // Тест из задания
        //test_phrase("ПРИВЕТМИР", "Основной тест из задания");

        // Дополнительные тесты
        test_phrase("HELLO", "Тест с английским текстом");
        test_phrase("ABCDEFGHII", "Тест с 9 символами (полный квадрат)");
        test_phrase("TEST", "Тест с 4 символами");

        std::cout << "\n" << std::string(50, '=') << std::endl;
        std::cout << "Все тесты завершены!" << std::endl;
        std::cout << std::string(50, '=') << std::endl;
    } catch (const std::exception& e) {
        std::cerr << "Произошла ошибка: " << e.what() << std::endl;
        return 1;
    } catch (...) {
        std::cerr << "Произошла неизвестная ошибка" << std::endl;
        return 1;
    }

    return 0;
}
```

Рисунки 2, 3 — код файла main.cpp

На рисунках 4, 5 представлен код файла `matrix.cpp`.

```
#include "../include/matrix.h"

Matrix create_matrix(int rows, int cols) {
    Matrix mat;
    mat.rows = rows;
    mat.cols = cols;
    mat.data = new char*[rows];

    for (int i = 0; i < rows; i++) {
        mat.data[i] = new char[cols];
        // Инициализируем пробелами
        for (int j = 0; j < cols; j++) {
            mat.data[i][j] = ' ';
        }
    }

    return mat;
}

void free_matrix(Matrix& mat) {
    if (mat.data != nullptr) {
        for (int i = 0; i < mat.rows; i++) {
            delete[] mat.data[i];
        }
        delete[] mat.data;
        mat.data = nullptr;
        mat.rows = 0;
        mat.cols = 0;
    }
}

Matrix string_to_matrix(const char* text, int rows, int cols) {
    Matrix mat = create_matrix(rows, cols);

    int text_length = strlen(text);
    int pos = 0;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (pos < text_length) {
                mat.data[i][j] = text[pos++];
            }
            // Остальные ячейки остаются с пробелами
        }
    }

    return mat;
}
```

```
Matrix encrypt(Matrix chars) {
    // При транспонировании строки становятся столбцами и наоборот
    Matrix result = create_matrix(rows: chars.cols, cols: chars.rows);

    for (int i = 0; i < chars.rows; i++) {
        for (int j = 0; j < chars.cols; j++) {
            result.data[j][i] = chars.data[i][j];
        }
    }

    return result;
}

std::string decrypt(Matrix encrypted) {
    std::string result = "";

    // Для обратного преобразования создаем временную матрицу
    // Меняем размеры обратно
    Matrix temp = create_matrix(rows: encrypted.cols, cols: encrypted.rows);

    // Транспонируем обратно
    for (int i = 0; i < encrypted.rows; i++) {
        for (int j = 0; j < encrypted.cols; j++) {
            temp.data[j][i] = encrypted.data[i][j];
        }
    }

    // Считываем построчно из временной матрицы
    for (int i = 0; i < temp.rows; i++) {
        for (int j = 0; j < temp.cols; j++) {
            if (temp.data[i][j] != ' ') {
                result += temp.data[i][j];
            }
        }
    }

    free_matrix(&mat: temp);
    return result;
}

bool is_perfect_square(int n) {
    if (n < 0) return false;
    int root = static_cast<int>(std::sqrt(n));
    return root * root == n;
}

void print_matrix(Matrix mat) {
    for (int i = 0; i < mat.rows; i++) {
        std::cout << "| ";
        for (int j = 0; j < mat.cols; j++) {
            std::cout << mat.data[i][j] << " ";
        }
        std::cout << "|" << std::endl;
    }
}

int calculate_matrix_size(int text_length) {
    int size = static_cast<int>(std::ceil(std::sqrt(text_length)));
    return size;
}
```

Рисунки 4, 5 — код файла `matrix.cpp`

На рисунке 6 представлен код файла CMakeLists.txt.

```
cmake_minimum_required(VERSION 3.10)
project(MatrixProject)

# Устанавливаем стандарт C++
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

# Включаем директории с исходниками
include_directories(include)

# Добавляем исполняемый файл
add_executable(matrix_app
    src/main.cpp
    src/matrix.cpp
)
```

Рисунок 6 – код файла CMakeLists.txt

На рисунках 7, 8 приведено выполнение команды ./matrix_app.

```
[gregory@HP build]$ ./matrix_app
ШИФРОВАЛЬЩИК СООБЩЕНИЙ
Метод: транспонирование матрицы

=====
Тест: Тест с английским текстом
=====
Исходный текст: 'HELLO'
Длина: 5 символов
Размер матрицы: 3x3

Исходная матрица:
| H E L |
| L O   |
|      |

Зашифрованная матрица (транспонированная):
| H L |
| E O |
| L   |

Шифр (чтение по столбцам исходной матрицы): "HL EO L "

Расшифрованный текст: 'HELLO'
✓ Шифрование и дешифрование прошли успешно!

=====
Тест: Тест с 9 символами (полный квадрат)
=====
Исходный текст: 'ABCDEFGH I'
Длина: 9 символов
Размер матрицы: 3x3
```

```
Исходная матрица:
| A B C |
| D E F |
| G H I |

Зашифрованная матрица (транспонированная):
| A D G |
| B E H |
| C F I |

Шифр (чтение по столбцам исходной матрицы): "ADG BEN CFI"

Расшифрованный текст: 'ABCDEFGH I'
✓ Шифрование и дешифрование прошли успешно!

=====
Тест: Тест с 4 символами
=====
Исходный текст: 'TEST'
Длина: 4 символа
Размер матрицы: 2x2

Исходная матрица:
| T E |
| S T |

Зашифрованная матрица (транспонированная):
| T S |
| E T |

Шифр (чтение по столбцам исходной матрицы): "TS ET"

Расшифрованный текст: 'TEST'
✓ Шифрование и дешифрование прошли успешно!

=====
Все тесты завершены!
```

Рисунки 7, 8 – выполнение команды ./matrix_app

На рисунке 9 представлена документация doxygen.

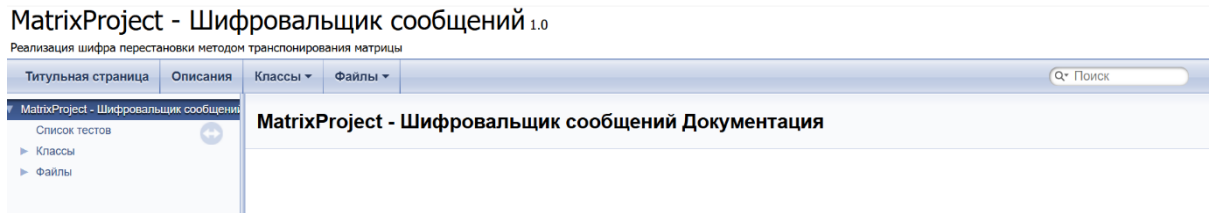


Рисунок 9 — создание документации doxygen

На этапе реализации шифровальщика сообщений (Мессенджер) возникли вопросы, требующие уточнения. Для их оперативного разрешения и проверки предполагаемых решений был использован AI-ассистент. Скриншоты диалога, демонстрирующие процесс решения проблем, представлены на рисунках 13 и 14.

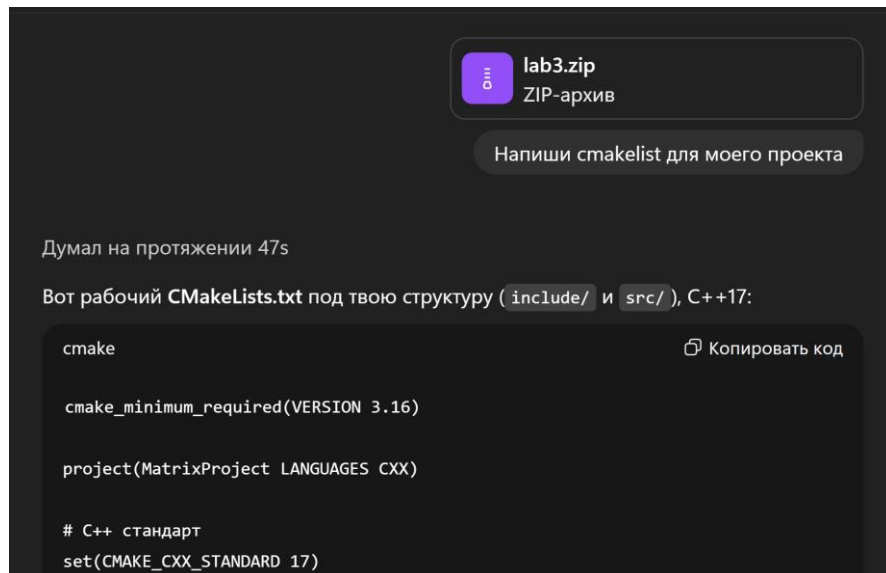


Рисунок 10 — Пример взаимодействия с AI-ассистентом

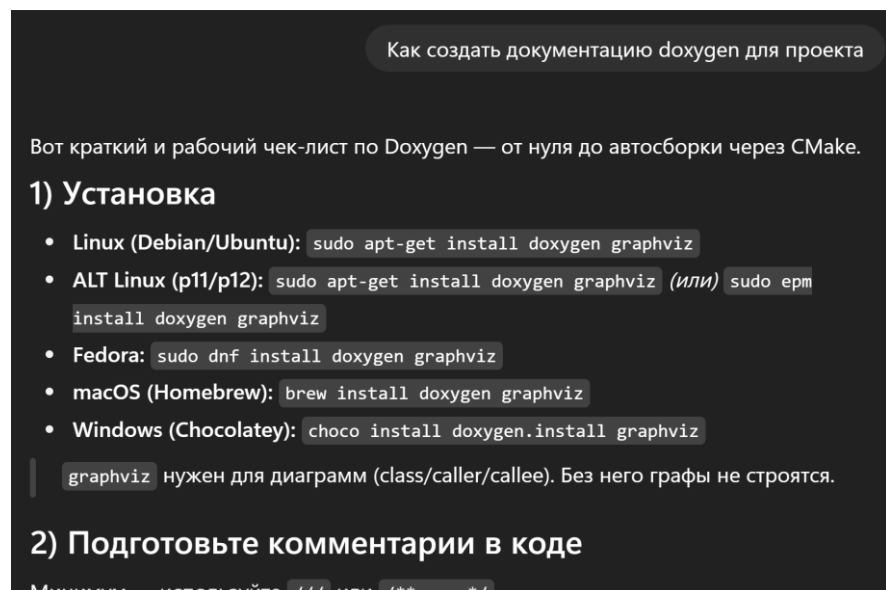


Рисунок 11 — Пример взаимодействия с AI-ассистентом

Вывод: в результате выполнения работы был создан шифровальщик сообщений (Мессенджер), с наглядным применением транспонирования, работой с матрицей символов (char**) и с акцентом на использование ИИ-ассистентов для решения задач.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Быков, А. Ю. Решение задач на языках программирования Си и Си++ : методические указания / А. Ю. Быков. — Москва : МГТУ им. Н.Э. Баумана, 2017. — 248 с. — ISBN 978-5-7038-4577-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/103505>
2. Каширин, И. Ю. От Си к Си++ : учебное пособие / И. Ю. Каширин, В. С. Новичков. — 2-е изд., стер. — Москва : Горячая линия-Телеком, 2012. — 334 с. — ISBN 978-5-9912-0259-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/5161>
3. Быков А. Ю. Решение задач на языках программирования Си и Си++ : метод. указания к выполнению лаб. работ / Быков А. Ю. ; МГТУ им. Н. Э. Баумана. - М. : Изд-во МГТУ им. Н. Э. Баумана, 2017. - 244 с. : ил. - ISBN 978-5-7038-4577-6.
4. Иванова Г. С., Ничушкина Т. Н. Объектно-ориентированное программирование : учебник для вузов / Иванова Г. С., Ничушкина Т. Н. ; общ. ред. Иванова Г. С. - М. : Изд-во МГТУ Н.Э.Баумана, 2002. <http://progbook.ru/technologiya-programmirovaniya/582-ivanova-tehnologiya-programmirovaniya.html>
5. Иванова Г. С., Ничушкина Т. Н. Объектно-ориентированное программирование : учебник для вузов / Иванова Г. С., Ничушкина Т. Н. ; общ. ред. Иванова Г. С. - М. : Изд-во МГТУ им. Н. Э. Баумана, 2014. - 455 с. : ил. - Библиогр.: с. 450. - ISBN 978-5-7038-3921-8.
6. Подбельский В.В. Язык Си++: Учебное пособие. — М.: Финансы и статистика, 2003. <http://progbook.ru/c/737-podbelskii-programmiovanie-na-yazyke-si.html>.
7. Акулов О.А., Медведев Н.В. Информатика. Базовый курс. — М.: Омега-Л, 2006. <http://razym.ru/naukaobraz/obrazov/151874-akulov-oa-medvedev-nv-informatika-bazovyy-kurs.html>
8. Вычислительные методы и программирование. МГУ им. М.В. Ломоносова. ISSN 1726-3522. Журнал входит в 1-й уровень Белого списка научных журналов Минобрнауки России. <https://num-meth.ru/index.php/journal/index>

Дополнительные материалы

1. Иванова Г.С. Технология программирования: Учебник для вузов. – М.: Изд. МГТУ им. Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы. – М., Вильямс, 2003. <http://razym.ru/naukaobraz/obrazov/181547-aho-a-ulman-d-hopkroft-d-struktury-dannyh-i-algoritmy.html>
2. Дейтел Х.М., Дейтел П.Дж. Как программировать на С++. – М.: Бином, 2001.
3. Т. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М. МЦНМО, 2005. <http://rutracker.org/forum/viewtopic.php?t=533181>
4. Джосьютис Н. С++ Стандартная библиотека для профессионалов. – СПб.: Питер, 2004. http://progbook.ru/c/178-dzhosyutis_c_standartnaya_biblioteka.html
5. Подбельский В.В. Стандартный Си++: Учебное пособие. – М.: Финансы и статистика, 2008.
6. Объектно-ориентированное программирование в С++: пер. с англ. / Лафоре Р. - 4-е изд. - СПб.: Питер, 2004. - 923 с. - (Классика computer science). - ISBN 5-94723-302-9.
7. Т. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М. МЦНМО, 2005.
8. Г. Шилдт. С++. Базовый курс, 3-е издание: Пер с англ. – М.: Издательский дом «Вильямс», 2011. – 624 с.
9. Павловская Т. А. С/С++. Программирование на языке высокого уровня: учебник для вузов / Павловская Т. А. - СПб.: Питер, 2003. - 460 с. - (Учебник для вузов). - ISBN 5-94723-568-4.
10. Бесплатные образовательные программы партнера (VK): <https://education.vk.company/students>