ЛАБОРАТОРНАЯ РАБОТА №3
«КЕЙС «ШИФРОВАЛЬЩИК СООБЩЕНИЙ (МЕССЕНДЖЕР)»

Выполнили студенты группы ИУ5-14Б:
Бутузов Роман
Корнеев Григорий
Лимаренко Дарья
Олейник Владимир
Стеганцева Ксения
Шебордаев Андрей

Задача

Задача: Реализовать простой шифр перестановки для текстового сообщения.

Подзадачи:

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

СТРУКТУРА ПРОЕКТА

→ main.cpp

Основная логика программы, ввод/вывод данных и управление процессом шифрования/дешифрования.

→ matrix.cpp / matrix.h

Модуль с реализацией функций для создания, заполнения, транспонирования и очистки матрицы символов.

→ CMakeLists.txt

Файл для автоматизации сборки проекта с помощью системы CMake.

→ Doxygen

Средство для автоматической генерации технической документации по исходному коду.

ФАЙЛ «main.cpp»

```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);
```

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

    // Показываем шифр как в задании (чтение по столбцам исходной матрицы)
    std::cout << "\nШифр (чтение по столбцам исходной матрицы): \"";
    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 << "\"" << 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(original);
    free_matrix(encrypted);


**
* @brief Главная функция приложения
*
* Содержит набор тестов для демонстрации работы шифровальщика.
*
* @return int Код завершения программы (0 - успех, 1 - ошибка)
*
* @test Тест с фразой "ПРИВЕТМИР" из задания
* @test Тест с английским текстом
* @test Тест с разной длиной текста
*/
```

```cpp
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("ABCDEFGHI", "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;
}
```

ФАЙЛ «matrix.cpp»

```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;
}
```

```cpp
Matrix encrypt(Matrix chars) {
    // При транспонировании строки становятся столбцами и наоборот
    Matrix result = create_matrix(chars.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(encrypted.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(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;
}
```

```cpp
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;
}
```

ФАЙЛ СБОРКИ ПРОЕКТА: CMakeLists.txt

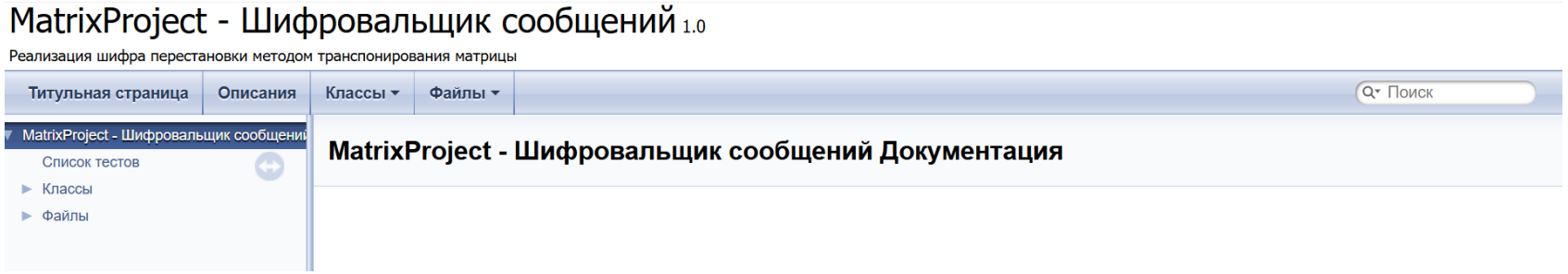
CMakeLists.txt

```
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
)
```

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: 'ABCDEFGHI'
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 BEH CFI"

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

================================================
Тест: Тест с 4 символами
================================================
Исходный текст: 'TEST'
Длина: 4 символов
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!
```

ДОКУМЕНТАЦИЯ (DOXYGEN)



ВЫВОДЫ

- Создана программа «Шифровальщик сообщений (Мессенджер)», реализующая шифр перестановки на основе транспонирования матрицы символов (char**).
- Реализованы функции обеспечивающие корректное шифрование и расшифровку текста.
- Освоены приёмы работы с указателями, матрицами и функциями в языке C++.
- Использованы инструменты VS Code, GitHub, CMake и Doxygen для разработки и документирования проекта
- При решении задач применялись ИИ-ассистенты, что повысило скорость и качество выполнения работы.