МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«МИРЭА – Российский технологический университет»**
**РТУ МИРЭА**

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

# ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4
**по дисциплине**
**«**Структуры и алгоритмы обработки данных**»**

## Тема: «Структуризация многоэлементных

## структур средствами struct»

Выполнил студент группы ИКБО-68-23 Зуев Д.А.

Принял преподаватель Сартаков М.В.

Самостоятельная работа выполнена «__»_____202__г. *(подпись студента)*

«Зачтено» «__»_____202__г. *(подпись руководителя)*

Москва 2025

# Практическая работа 4

## 1. Цель

Получение навыков по реализации многоэлементных структур данных задачи C++ и навыков по реализации многоэлементных данных средствами структуры данных – таблица.

## 2. Постановка задачи

1. Разработать набор операций для управления таблицей, созданной на основе динамического массива. Структура записи элемента таблицы определена вариантом индивидуального задания.

   a. Создать проект, добавить заголовочный файл. Включить в заголовочный файл:

      i. Разработанный тип данных, определяющий структуру элемента таблицы (записи).

      ii. Определенная на базе типа struct структура хранения данных по таблице, содержащая: размер таблицы – n, массив из n элементов типа записи варианта.

      iii. Реализованная функция вывода таблицы на экран.

      iv. Операции варианта задания, оформленные как функции с параметрами.

   b. Для заполнения отдельной записи с клавиатуры разработать функцию, которая принимает в качестве параметра пустую запись, а возвращает заполненную.

   c. Наполнение таблицы данными должна выполнять функция вставки или добавления записи, включенная в список операций варианта. Эта функция должна приять через параметр заполненную запись, выполнить действие по вставке или добавления, согласно алгоритму варианта этой

операции.

2. Разработать программу, демонстрирующую работу всех функций с массивом или записей. Программа должна позволять пользователю непрерывно выполнять операции над таблицей в произвольном порядке.

3. Разработать набор тестовых данных по наполнению таблицы. Включить в меню программы возможность автоматического ввода разработанных тестовых данных.

Вариант 29. Разработка тестовых вопросов. Структура записи вопроса: номер темы, текст вопроса, балл за правильный ответ, список вариантов ответов, номер правильного ответа.

Операции

1. Заполнить запись по одному вопросу с клавиатуры.
2. Вставить запись в таблицу, упорядочивая по номеру темы (сортировку не использовать).
3. Сформировать тест, включая в него по два вопроса из каждой темы, выбранных случайно.
4. Удалить вопросы по одной теме.

## 3. Решение

В процессе выполнения данной практической работы были разработаны несколько файлов программы, которые полностью реализуют поставленную задачу.

### Файл main.cpp

Этот файл содержит основную логику программы, в том числе меню для взаимодействия с пользователем. Функция menu предоставляет пользователю различные варианты действий, такие как добавление вопросов, формирование теста и удаление вопросов по теме. Пользователь может выбрать одну из опций, а затем программа вызывает соответствующую функцию из файла

table_operations.cpp для выполнения выбранного действия.

Функция menu отвечает за взаимодействие с пользователем, предлагая различные опции для управления таблицей вопросов. Она использует switch для обработки пользовательских выборов, вызывая соответствующие функции для выполнения задач, таких как добавление вопросов или удаление вопросов по теме. Функция также заботится об очистке памяти, освобождая ресурсы перед выходом из программы.

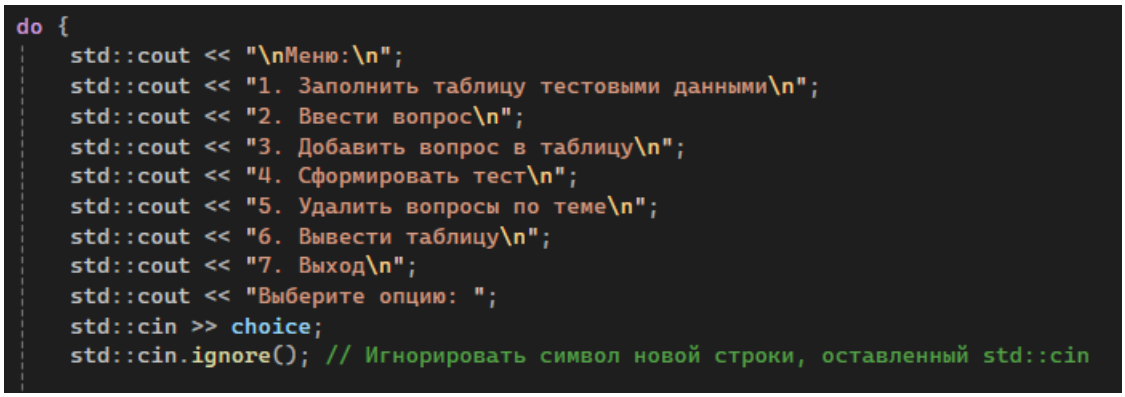


**Рисунок 1 – Часть функции menu**

**Файл table_operations.cpp**

Этот файл содержит реализацию функций для работы с таблицей вопросов, таких как printTable, addRecord, deleteQuestionsByTopic, и другие. Функции в этом файле выполняют задачи, связанные с добавлением, удалением и отображением вопросов в таблице, а также с созданием теста на основе существующих вопросов.

Функция printTable. Эта функция выводит содержимое таблицы на экран. Она перебирает все вопросы в таблице и выводит информацию о каждом вопросе, включая текст, варианты ответов и номер правильного ответа. Функция помогает визуально оценить состояние таблицы и ознакомиться со всеми её вопросами.

```
void printTable(const Table& table) {
    setlocale(0, "rus");
```

```
    for (int i = 0; i < table.size; ++i) {
        const Question& q = table.data[i];
        std::cout << "Номер темы: " << q.topicNumber << std::endl;
        std::cout << "Вопрос: " << q.questionText << std::endl;
        std::cout << "Баллы: " << q.points << std::endl;
        std::cout << "Варианты ответов: ";
        for (const std::string& option : q.options) {
            std::cout << option << " ";
        }
        std::cout << std::endl;
        std::cout << "Номер правильного ответа: " << q.correctAnswerIndex << std::endl;
        std::cout << "--------------------------------" << std::endl;
    }
}
```

Функция createQuestion создает новый вопрос, инициализируя его полями, такими как номер темы, текст вопроса, варианты ответов и индекс правильного ответа. Возвращаемое значение — объект типа Question, который можно добавлять в таблицу. Функция упрощает процесс создания вопроса, предоставляя готовую структуру.

```
Question createQuestion(int topicNumber, const std::string& questionText, int points, const std::vector<std::string>& options, int correctAnswerIndex) {
    Question q;
    q.topicNumber = topicNumber;
    q.questionText = questionText;
    q.points = points;
    q.options = options;
    q.correctAnswerIndex = correctAnswerIndex;
    return q;
}
```

Функция addRecord добавляет новый вопрос в таблицу, вставляя его на нужную позицию в зависимости от номера темы. Она выделяет новую память для увеличенного массива вопросов, копирует в него старые данные и добавляет новый вопрос. В конце функция освобождает память, занятую старым массивом.

```
void addRecord(Table& table, const Question& question) {
    // Найти точку вставки
    int pos = 0;
    while (pos < table.size && table.data[pos].topicNumber <
```

```
question.topicNumber) {
        ++pos;
    }

    // Выделить новый массив
    Question* newData = new Question[table.size + 1];
    for (int i = 0; i < pos; ++i) {
        newData[i] = table.data[i];
    }
    newData[pos] = question;
    for (int i = pos; i < table.size; ++i) {
        newData[i + 1] = table.data[i];
    }

    delete[] table.data;
    table.data = newData;
    ++table.size;
}
```

Функция generateTest. Эта функция создает набор вопросов для теста, выбирая случайные вопросы из таблицы по каждой теме. Она перебирает темы и, если в таблице есть достаточно вопросов по теме, случайным образом выбирает два из них, добавляя их в итоговый набор. Возвращает вектор вопросов, которые можно использовать для проведения теста.

```
std::vector<Question> generateTest(const Table& table) {
    std::vector<Question> testQuestions;
    std::vector<int> selectedTopics;

    std::srand(std::time(0));
    for (int topic = 1; topic <= 10; ++topic) {
        std::vector<Question> topicQuestions;
        for (int i = 0; i < table.size; ++i) {
            if (table.data[i].topicNumber == topic) {
                topicQuestions.push_back(table.data[i]);
            }
        }
        if (topicQuestions.size() >= 2) {
            int index1 = std::rand() % topicQuestions.size();
            int index2;
            do {
                index2 = std::rand() % topicQuestions.size();
            } while (index2 == index1);
            testQuestions.push_back(topicQuestions[index1]);
            testQuestions.push_back(topicQuestions[index2]);
        }
```

```
    }

    return testQuestions;
}
```

Функция deleteQuestionsByTopic удаляет все вопросы из таблицы по заданной теме. Она создает новый массив, исключая из него вопросы по указанной теме, и освобождает память от старого массива. Функция помогает эффективно управлять таблицей, позволяя удалять группы вопросов одним вызовом.

```
void deleteQuestionsByTopic(Table& table, int topicNumber) {
    int newSize = 0;
    for (int i = 0; i < table.size; ++i) {
        if (table.data[i].topicNumber != topicNumber) {
            ++newSize;
        }
    }

    Question* newData = new Question[newSize];
    int j = 0;
    for (int i = 0; i < table.size; ++i) {
        if (table.data[i].topicNumber != topicNumber) {
            newData[j++] = table.data[i];
        }
    }

    delete[] table.data;
    table.data = newData;
    table.size = newSize;
}
```

Функция fillTableWithTestData. Эта функция заполняет таблицу заранее заданными тестовыми данными, которые можно использовать для демонстрации или тестирования. Она инициализирует таблицу массивом вопросов, заполняя его значениями по умолчанию. Это упрощает начальную настройку таблицы, позволяя быстро создать базу вопросов.

```
void fillTableWithTestData(Table& table) {
    setlocale(0, "rus");

    // Пример тестовых данных
    Question testData[] = {
        {1, "Сколько будет 2 + 2?", 5, {"3", "4", "5", "6"}, 1},
        {1, "Сколько будет 3 + 5?", 5, {"7", "8", "9", "10"}, 1},
```

```
        {2, "Какой столица Франции?", 10, {"Берлин", "Мадрид", "Париж", "Рим"},
2},
        {2, "Какой самый большой океан?", 10, {"Атлантический", "Индийский",
"Арктический", "Тихий"}, 3}
        // Добавьте больше вопросов по необходимости
    };

    int numTestData = sizeof(testData) / sizeof(testData[0]);
    table.size = numTestData;
    table.data = new Question[numTestData];
    for (int i = 0; i < numTestData; ++i) {
        table.data[i] = testData[i];
    }
}
```

**Файл table_operations.h**

Заголовочный файл содержит объявления структур Question и Table, а также прототипы функций для работы с таблицей вопросов. Структуры представляют данные, такие как текст вопросов и варианты ответов, в то время как функции позволяют управлять таблицей, добавляя, удаляя и отображая вопросы.

Структура Question представляет собой структуру, содержащую поля для хранения данных одного вопроса: номер темы, текст вопроса, количество баллов, список вариантов ответов и индекс правильного ответа. Эта структура позволяет компактно хранить и передавать данные, связанные с вопросом, внутри программы.

```
// Структура для записи элемента таблицы
struct Question {
    int topicNumber; // Номер темы
    std::string questionText; // Текст вопроса
    int points; // Баллы за правильный ответ
    std::vector<std::string> options; // Список вариантов ответов
    int correctAnswerIndex; // Номер правильного ответа
};
```

**Рисунок 2 – Структура Question**

Структура Table — структура, представляющая таблицу вопросов, содержащая массив записей типа Question и поле size, указывающее на количество записей. Это позволяет удобно управлять данными, обеспечивая возможность добавлять и удалять вопросы и получать доступ к ним по индексу.

```
// Структура хранения данных по таблице
struct Table {
    int size; // Размер таблицы
    Question* data; // Динамический массив записей
};
```

**Рисунок 3 – Структура Table**

## 4. Тестирование

```
Меню:
1. Заполнить таблицу тестовыми данными
2. Ввести вопрос
3. Добавить вопрос в таблицу
4. Сформировать тест
5. Удалить вопросы по теме
6. Вывести таблицу
7. Выход
Выберите опцию: 3
Введите номер темы: 3
Введите текст вопроса: Что такое арбуз?
Введите количество баллов: 3
Введите количество вариантов ответов: 3
Введите вариант ответа 1: Овощ
Введите вариант ответа 2: Ягода
Введите вариант ответа 3: Фрукт
Введите индекс правильного ответа (с нуля): 1

Меню:
1. Заполнить таблицу тестовыми данными
2. Ввести вопрос
3. Добавить вопрос в таблицу
4. Сформировать тест
5. Удалить вопросы по теме
6. Вывести таблицу
7. Выход
Выберите опцию: 6
Номер темы: 3
Вопрос: Что такое арбуз?
Баллы: 3
Варианты ответов: Овощ Ягода Фрукт
Номер правильного ответа: 1
---------------------------------
```

**Рисунок 4 – Заполнили запись по одному вопросу**

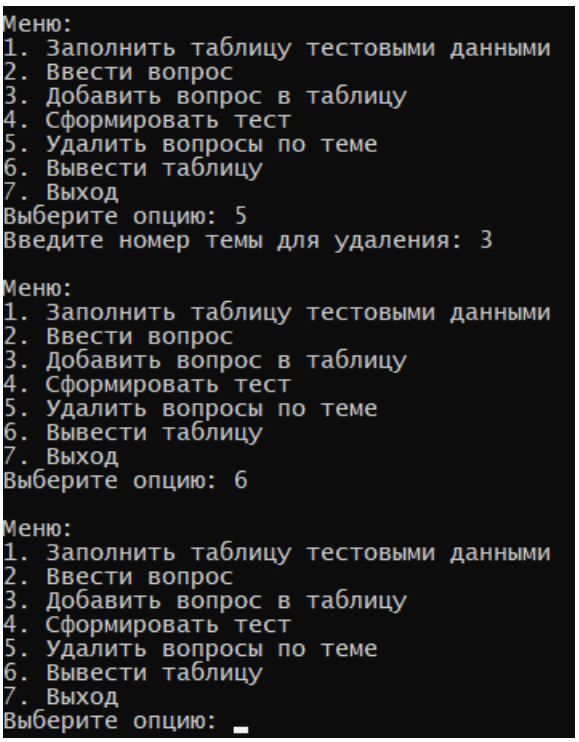


**Рисунок 5 – Удаление из таблицы вопросов по теме 3**

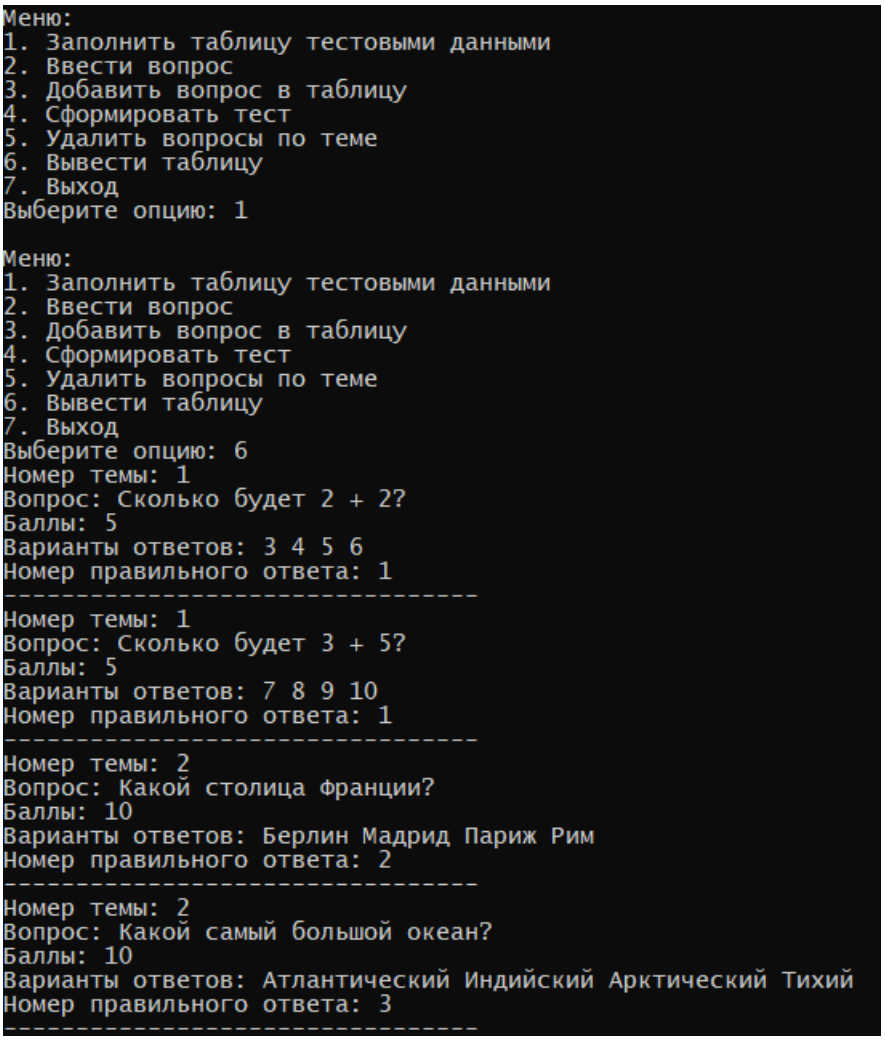


**Рисунок 8 – Заполнение таблицы тестовыми данными**

## 5. Вывод

В ходе работы был разработан набор операций для управления таблицей, созданной на основе динамического массива, для хранения тестовых вопросов. Были реализованы функции для заполнения записи с клавиатуры, вставки записи в таблицу с упорядочением по номеру темы, формирования теста путем случайного выбора вопросов, а также удаления вопросов по одной теме. Программа позволяет пользователю непрерывно выполнять операции над таблицей, включая автоматический ввод тестовых данных.

## 6. Исходный код

### Файл main.cpp:

```
#include <iostream>
#include "table_operations.h"
#include <Windows.h>

void menu() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);


    Table table = { 0, nullptr };
    int choice;

    do {
        std::cout << "\nМеню:\n";
        std::cout << "1. Заполнить таблицу тестовыми данными\n";
        std::cout << "2. Ввести вопрос\n";
        std::cout << "3. Добавить вопрос в таблицу\n";
        std::cout << "4. Сформировать тест\n";
        std::cout << "5. Удалить вопросы по теме\n";
        std::cout << "6. Вывести таблицу\n";
        std::cout << "7. Выход\n";
        std::cout << "Выберите опцию: ";
        std::cin >> choice;
        std::cin.ignore(); // Игнорировать символ новой строки, оставленный
std::cin

        switch (choice) {
            case 1:
                fillTableWithTestData(table);
                break;
```

```
            case 2: {
                // Ввод данных для создания вопроса
                int topicNumber;
                std::string questionText;
                int points;
                int numOptions;
                std::vector<std::string> options;
                int correctAnswerIndex;

                std::cout << "Введите номер темы: ";
                std::cin >> topicNumber;
                std::cin.ignore();  //  Очистка  символа  новой  строки,
оставленного std::cin

                std::cout << "Введите текст вопроса: ";
                std::getline(std::cin, questionText);

                std::cout << "Введите количество баллов: ";
                std::cin >> points;

                std::cout << "Введите количество вариантов ответов: ";
                std::cin >> numOptions;
                std::cin.ignore(); // Очистка символа новой строки

                options.resize(numOptions);
                for (int i = 0; i < numOptions; ++i) {
                    std::cout << "Введите вариант ответа " << (i + 1) << ":
";
                    std::getline(std::cin, options[i]);
                }

                std::cout << "Введите индекс правильного ответа (с нуля):
";
                std::cin >> correctAnswerIndex;

                Question  q  =  createQuestion(topicNumber,  questionText,
points, options, correctAnswerIndex);
                std::cout << "Вопрос заполнен.\n";
                break;
            }
            case 3: {
                // Добавить вопрос в таблицу
                // Вставить сюда ввод данных, аналогично case 2
                int topicNumber;
                std::string questionText;
                int points;
                int numOptions;
                std::vector<std::string> options;
                int correctAnswerIndex;
```

```
                std::cout << "Введите номер темы: ";
                std::cin >> topicNumber;
                std::cin.ignore();

                std::cout << "Введите текст вопроса: ";
                std::getline(std::cin, questionText);

                std::cout << "Введите количество баллов: ";
                std::cin >> points;

                std::cout << "Введите количество вариантов ответов: ";
                std::cin >> numOptions;
                std::cin.ignore();

                options.resize(numOptions);
                for (int i = 0; i < numOptions; ++i) {
                    std::cout << "Введите вариант ответа " << (i + 1) << ":
";
                    std::getline(std::cin, options[i]);
                }

                std::cout << "Введите индекс правильного ответа (с нуля):
";
                std::cin >> correctAnswerIndex;

                Question q = createQuestion(topicNumber, questionText,
points, options, correctAnswerIndex);
                addRecord(table, q);
                break;
            }
        std::vector<Question> test = generateTest(table);
            std::cout << "Сформированный тест:\n";
            for (const Question& q : test) {
                std::cout << "Номер темы: " << q.topicNumber << std::endl;
                std::cout << "Вопрос: " << q.questionText << std::endl;
                std::cout << "Баллы: " << q.points << std::endl;
                std::cout << "Варианты ответов: ";
                for (const std::string& option : q.options) {
                    std::cout << option << " ";
                }
                std::cout << std::endl;
                std::cout << "Номер правильного ответа: " <<
q.correctAnswerIndex << std::endl;
                std::cout << "---------------------------------" <<
std::endl;
            }
            break;
        }
        case 5: {
            int topicNumber;
```

```
                std::cout << "Введите номер темы для удаления: ";
                std::cin >> topicNumber;
                deleteQuestionsByTopic(table, topicNumber);
                break;
            }
            case 6:
                printTable(table);
                break;
            case 7:
                std::cout << "Выход из программы.\n";
                break;
            default:
                std::cout << "Неверный выбор. Попробуйте снова.\n";
                break;
        }
    } while (choice != 7);

    // Очистка ресурсов
    delete[] table.data;
}

int main() {
    menu();
    return 0;
}
```

**Файл table_operations.cpp:**

```
#include "table_operations.h"

void printTable(const Table& table) {
    setlocale(0, "rus");

    for (int i = 0; i < table.size; ++i) {
        const Question& q = table.data[i];
        std::cout << "Номер темы: " << q.topicNumber << std::endl;
        std::cout << "Вопрос: " << q.questionText << std::endl;
        std::cout << "Баллы: " << q.points << std::endl;
        std::cout << "Варианты ответов: ";
        for (const std::string& option : q.options) {
            std::cout << option << " ";
        }
        std::cout << std::endl;
        std::cout << "Номер правильного ответа: " << q.correctAnswerIndex
<< std::endl;
        std::cout << "--------------------------------" << std::endl;
    }
}

Question createQuestion(int topicNumber, const std::string& questionText,
int points, const std::vector<std::string>& options, int correctAnswerIndex) {
    Question q;
    q.topicNumber = topicNumber;
    q.questionText = questionText;
    q.points = points;
    q.options = options;
    q.correctAnswerIndex = correctAnswerIndex;
    return q;
}

void addRecord(Table& table, const Question& question) {
    // Найти точку вставки
    int pos = 0;
    while (pos < table.size && table.data[pos].topicNumber <
question.topicNumber) {
        ++pos;
    }

    // Выделить новый массив
    Question* newData = new Question[table.size + 1];
    for (int i = 0; i < pos; ++i) {
        newData[i] = table.data[i];
    }
    newData[pos] = question;
    for (int i = pos; i < table.size; ++i) {
        newData[i + 1] = table.data[i];
```

```
    }

    delete[] table.data;
    table.data = newData;
    ++table.size;
}

std::vector<Question> generateTest(const Table& table) {
    std::vector<Question> testQuestions;
    std::vector<int> selectedTopics;

    // Инициализация генератора случайных чисел
    std::srand(std::time(0));

    for (int topic = 1; topic <= 10; ++topic) {
        std::vector<Question> topicQuestions;
        for (int i = 0; i < table.size; ++i) {
            if (table.data[i].topicNumber == topic) {
                topicQuestions.push_back(table.data[i]);
            }
        }
        if (topicQuestions.size() >= 2) {
            int index1 = std::rand() % topicQuestions.size();
            int index2;
            do {
                index2 = std::rand() % topicQuestions.size();
            } while (index2 == index1);

            testQuestions.push_back(topicQuestions[index1]);
            testQuestions.push_back(topicQuestions[index2]);
        }
    }

    return testQuestions;
}

void deleteQuestionsByTopic(Table& table, int topicNumber) {
    int newSize = 0;
    for (int i = 0; i < table.size; ++i) {
        if (table.data[i].topicNumber != topicNumber) {
            ++newSize;
        }
    }

    Question* newData = new Question[newSize];
    int j = 0;
    for (int i = 0; i < table.size; ++i) {
        if (table.data[i].topicNumber != topicNumber) {
            newData[j++] = table.data[i];
```

```
            }
        }

        delete[] table.data;
        table.data = newData;
        table.size = newSize;
    }

    void fillTableWithTestData(Table& table) {
        setlocale(0, "rus");

        // Пример тестовых данных
        Question testData[] = {
            {1, "Сколько будет 2 + 2?", 5, {"3", "4", "5", "6"}, 1},
            {1, "Сколько будет 3 + 5?", 5, {"7", "8", "9", "10"}, 1},
            {2, "Какой столица Франции?", 10, {"Берлин", "Мадрид", "Париж",
"Рим"}, 2},
            {2, "Какой самый большой океан?", 10, {"Атлантический",
"Индийский", "Арктический", "Тихий"}, 3}
            // Добавьте больше вопросов по необходимости
        };

        int numTestData = sizeof(testData) / sizeof(testData[0]);
        table.size = numTestData;
        table.data = new Question[numTestData];
        for (int i = 0; i < numTestData; ++i) {
            table.data[i] = testData[i];
        }
    }
```

**Файл table_operations.h:**

```
#ifndef TABLE_OPERATIONS_H
#define TABLE_OPERATIONS_H

#include <string>
#include <vector>
#include <iostream>

// Структура для записи элемента таблицы
struct Question {
    int topicNumber; // Номер темы
    std::string questionText; // Текст вопроса
    int points; // Баллы за правильный ответ
    std::vector<std::string> options; // Список вариантов ответов
    int correctAnswerIndex; // Номер правильного ответа
};

// Структура хранения данных по таблице
struct Table {
    int size; // Размер таблицы
    Question* data; // Динамический массив записей
};

// Функция для вывода таблицы на экран
void printTable(const Table& table);

// Функция для создания вопроса
Question createQuestion(int topicNumber, const std::string& questionText,
int points, const std::vector<std::string>& options, int correctAnswerIndex);

// Функция для добавления записи в таблицу
void addRecord(Table& table, const Question& question);

// Функция для удаления вопросов по теме
void deleteQuestionsByTopic(Table& table, int topicNumber);

// Функция для заполнения таблицы тестовыми данными
void fillTableWithTestData(Table& table);

// Функция для формирования теста
std::vector<Question> generateTest(const Table& table);

#endif // TABLE_OPERATIONS_H
```