**Лабораторная работа №2.**

Поиск и сортировка информации

**Вариант 10.** Разработка информационной подсистемы учета тестирований учащихся по дисциплинам.

Для разработки информационной подсистемы необходимо решить следующие подзадачи:

1. **Моделирование данных:** создать структуры данных для хранения информации об учащихся, тестах, дисциплинах и результатах.
2. **Инициализация данных:** заполнить систему тестовыми данными для демонстрации работы программы.
3. **Обработка запроса пользователя:** получить от пользователя название дисциплины.
4. **Анализ данных по дисциплине:**
   * + **Подзадача 4.1:** получить все тесты по введенной дисциплине и подсчитать, сколько раз каждый тест был использован.
   * **Подзадача 4.2:** отсортировать список тестов от наиболее используемых к наименее используемым.
   * **Подзадача 4.3:** для каждого учащегося, проходившего тесты по данной дисциплине, вычислить средний балл.
   * **Подзадача 4.4:** найти учащегося с самым высоким средним баллом по этой дисциплине.
5. **Формирование результата:** вывести на экран отсортированный список тестов и информацию о самом успешном учащемся.

**Б) Описание входных/выходных данных для каждой подзадачи**

#### ****Подзадача 1 (Моделирование данных):****

* **Входные данные:** Отсутствуют
* **Выходные данные:** Определены структуры классов Student и TestResult
* **Результат исполнения:** Создана модель данных для хранения информации

#### ****Подзадача 2 (Инициализация данных):****

* **Функция:** TestResult[] InitializeTestData()
* **Входные данные:** Отсутствуют
* **Выходные данные:**

TestResult с тестовыми данными

* **Результат исполнения:** Система заполнена тестовыми данными

#### ****Подзадача 3 (Обработка запроса пользователя):****

* **Функция:** string GetUserInput()
* **Входные данные:** отсутствуют (ввод с клавиатуры)
* **Выходные данные:**

Возвращает string - название дисциплины, введенное пользователем

* **Результат исполнения:** Получена целевая дисциплина для анализа

#### ****Подзадача 4.1 (Подсчет использования тестов):****

* **Функция** void CalculateTestUsageStatistics(TestResult[] allResults, string discipline, out string[] testNames, out int[] usageCounts, out int actualCount)
* **Входные данные:**

allResults - массив всех результатов тестов

discipline - название дисциплины для поиска

* **Выходные данные:**

testNames: Выходной параметр - массив названий тестов

usageCounts: Выходной параметр - массив количеств использований

actualCount: Выходной параметр - фактическое количество найденных тестов

* **Результат исполнения:** Определен самый часто используемый тест

#### ****Подзадача 4.2 (Сортировка тестов):****

* **Реализовано внутри функции** void SortTestsByUsage(string[] testNames, int[] usageCounts, int count)
* Входные данные:

testNames: Массив названий тестов (входной и выходной)

usageCounts: Массив количеств использований (входной и выходной)

* **Выходные данные:**

testNames: Массив названий тестов (входной и выходной)

usageCounts: Массив количеств использований (входной и выходной)

* **Результат исполнения:** найден наиболее популярный тест (фактически отсортированный список из одного элемента)

#### ****Подзадача 4.3 (Вычисление среднего балла по учащимся):****

* **Реализовано внутри функции**void CalculateAverageScores (TestResult[] allResults, string discipline, out Student[] students, out double[] averageScores, out int studentCount)
* **Входные данные:**

allResults: Массив всех результатов тестов

discipline: Название дисциплины для поиска

* **Выходные данные:**

students: массив учащихся

averageScores: массив средних баллов

studentCount: фактическое количество учащихся

* **Результат исполнения:** Рассчитаны средние баллы для каждого студента по дисциплине

#### ****Подзадача 4.4 (Поиск лучшего учащегося):****

**Функция:** bool FindTopStudent(Student[] students, double[] averageScores, int studentCount, out Student topStudent, out double topScore)

* **Входные данные:**

students: Массив учащихся

averageScores: Массив средних баллов

studentCount: Количество учащихся в массивах

* **Выходные данные:**

topStudent: лучший учащийся

topScore: лучший средний балл

* **Результат исполнения:** Найден студент с максимальным средним баллом

#### ****Подзадача 5 (Формирование результата):****

* **Функция:** void PrintResults(string discipline, string[] testNames, int[] usageCounts, int testCount, Student topStudent, double topScore, bool hasTopStudent)
* **Входные данные:**

discipline: Название дисциплины

testNames: Массив названий тестов

usageCounts: Массив количеств использований

testCount: Количество тестов в массивах

topStudent: Лучший учащийся

topScore: Лучший средний балл

hasTopStudent: Флаг наличия лучшего учащегося

* **Выходные данные:** отсутствуют (данные выводятся на экран)
* **Результат исполнения:** Пользователь видит результаты анализа

#### ****B) Определение функций для каждой подзадачи****

1. **Функция:** TestResult[] InitializeTestData()
   * **Назначение:** создаёт и возвращает массив с тестовыми данными о результатах тестирований.
   * **Возвращает:** Массив объектов TestResult.
   * **Модуль:** DataInitializer.cs
2. **Функция:** string GetUserInput()
   * **Назначение:** запрашивает и возвращает у пользователя название дисциплины.
   * **Возвращает:** Строку с названием дисциплины.
   * **Модуль:**UserInputHelper.cs
3. **Функция:** void CalculateTestUsageStatistics (TestResult[] allResults, string discipline, out string[] testNames, out int[] usageCounts, out int actualCount)
   * **Назначение:** подсчитывает, сколько раз каждый тест использовался в рамках заданной дисциплины.
   * **Параметры:**

allResults: Массив всех результатов тестов

discipline: Название дисциплины для поиска

testNames: Выходной параметр - массив названий тестов

usageCounts: Выходной параметр - массив количеств использований

actualCount: Выходной параметр - фактическое количество найденных тестов

* + **Модуль:** AnalysisModule.cs

1. **Функция:** void SortTestsByUsage(string[] testNames, int[] usageCounts, int count)
   * **Назначение:** сортирует массивы тестов и количеств использований по убыванию популярности.
   * **Параметры:**

testNames: Массив названий тестов (входной и выходной)

usageCounts: Массив количеств использований (входной и выходной)

count: Количество элементов для сортировки

* + **Модуль:** AnalysisModule.cs

1. **Функция:** void CalculateAverageScores (TestResult[] allResults, string discipline, out Student[] students, out double[] averageScores, out int studentCount)
   * **Назначение:** для каждого учащегося вычисляет средний балл по всем тестам в заданной дисциплине.
   * **Параметры:**

allResults: Массив всех результатов тестов

discipline: Название дисциплины для поиска

students: Выходной параметр - массив учащихся

averageScores: Выходной параметр - массив средних баллов

studentCount: Выходной параметр - фактическое количество учащихся

* + **Модуль:** AnalysisModule.cs

1. **Функция:** bool FindTopStudent (Student[] students, double[] averageScores, int studentCount, out Student topStudent, out double topScore)
   * **Назначение:** находит учащегося с максимальным средним баллом.
   * **Параметры:**

students: Массив учащихся

averageScores: Массив средних баллов

studentCount: Количество учащихся в массивах

topStudent: Выходной параметр - лучший учащийся

topScore: Выходной параметр - лучший средний балл

* + **Возвращает:** true если найден лучший учащийся, false если массив пуст
  + **Модуль:** AnalysisModule.cs

1. **Функция:** void PrintResults (string discipline, string[] testNames, int[] usageCounts, int testCount, Student topStudent, double topScore, bool hasTopStudent)
   * **Назначение:** выводит на экран отсортированный список тестов и информацию о лучшем учащемся.
   * **Параметры:**

discipline: Название дисциплины

testNames: Массив названий тестов

usageCounts: Массив количеств использований

testCount: Количество тестов в массивах

topStudent: Лучший учащийся

topScore: Лучший средний балл

hasTopStudent: Флаг наличия лучшего учащегося

* + **Модуль:** ReportModule.cs