Федеральное агентство по рыболовству
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Астраханский государственный технический университет»

Институт информационных технологий и коммуникаций
Направление подготовки 09.03.04 Программная инженерия
Профиль «Разработка программно-информационных систем»
Кафедра «Автоматизированные системы обработки информации и управления»

КУРСОВОЙ ПРОЕКТ

Учебно-демонстрационная программа «Оптимизация умножения матриц методом динамического программирования»

по дисциплине «Алгоритмы и структуры данных»

Допущен к защите
«__» _____20__г.
Руководитель
_____

Проект выполнен
обучающимся группы ДИПРб-21
Иргалиев А. А.
_____

Оценка, полученная на защите
«_____»

Руководитель
доцент Лаптев В.В.

Члены комиссии:

_____ Лаптев В.В.

Астрахань – 2019

ФЕДЕРАЛЬНОЕ АГЕНТСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ ПО РЫБОЛОВСТВУ

АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ
Заведующий кафедрой
к.т.н., доцент
С.В. Белов_____
«____»_____201 __ г.

Кафедра
«Автоматизированные системы
обработки информации и управления»

ЗАДАНИЕ

на выполнение курсового проекта

Обучающийся ***Иргалиев Амин Альбертович***

Группа ***ДИПРб-21***

Дисциплина ***Алгоритмы и структуры данных***

Тема ***Учебно-демонстрационная программа «Оптимизация умножения матриц методом динамического программирования»***

Дата получения задания «_____»_____201___г.

Срок представления обучающимся КП на кафедру «_____»_____201___г.

Руководитель ***доцент_____ Лаптев В.В.*** «_____»_____201___г.

должность, степень, звание подпись ФИО

Обучающийся _____ ***Иргалиев А.А.*** «_____»_____201___г.

подпись ФИО

Задачи

Разработка программного продукта, который

- предоставляет пользователю теоретический материал по теме «Оптимизация умножения матриц методом динамического программирования»;
- визуализирует поиск оптимального количества скалярных операций, для вычисления их произведения, путем расставления скобок;
- предоставляет пользователю возможность тестирования по теме «Оптимизация умножения матриц методом динамического программирования»;
- позволяет очищать файл с результатами тестирования.

Список рекомендуемой литературы

1. Кормен Т., Лайзерсон Э.Ч.,Ривест Р.Л., Алгоритмы: построение и анализ / Пер. с англ. Под ред. А. Шеня. – М.: МЦНМО, 2002. – 960 с.: 263 ил.
2. Лаптев В.В., С++. Экспресс-курс. – СПб.: БХВ-Петербург, 2004. – 512 с.: ил..
3. Шлее М., Qt 5.10. Профессиональное программирование на C++ - СПБ.: БХВ-Петербург, 2018 — 1072 с.: ил. - (В подлиннике)

УТВЕРЖДАЮ

Заведующий кафедрой

к.т.н., доцент

С.В. Белов_____

«____»_____20 ___г.

К заданию на курсовой проект
по дисциплине
«Алгоритмы и структуры данных»

КАЛЕНДАРНЫЙ ГРАФИК
курсового проектирования

№ п/п	Разделы, темы и их содержание, графический материал	Дата сдачи	Объем,%
1	Выбор темы	15.09.2019	1
2	Техническое задание	30.09.2019	3
3	Разработка модели, проектирование системы ▪ *введение,* ▪ *технический проект,* ▪ *программа и методика испытаний,* ▪ *литература*	31.10.2019	25
4	Программная реализация системы ▪ *работающая программа,* ▪ *рабочий проект* ▪ *скорректированное техническое задание (при необходимости)*	30.11.2019	40
5	Тестирование и отладка системы, эксперименты ▪ *работающая программа с внесёнными изменениями,* ▪ *окончательные текстывсех разделов*	15.12.2019	50
6	Компоновка текста Подготовка презентации и доклада ▪ *пояснительная записка* ▪ *презентация* ▪ *электронный носитель с текстом пояснительной записки, исходным кодом проекта, презентацией и готовым программным продуктом*	20.12.2019	59
7	Защита курсового проекта	27.12.2019	60-100

С графиком ознакомлен «____» _____ 20 ___г.

ИргалиевА.А. _____, обучающийся группы ДИПРб-21
(фамилия, инициалы, подпись)

График курсового проектирования выполнен
без отклонений / с незначительными отклонениями / со значительными отклонениями

нужное подчеркнуть

Руководитель курсового проекта_____ доцент Лаптев В.В.
подпись, ученая степень, звание, фамилия, инициалы

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время существует всё больше процессов, которые необходимо оптимизировать. В процессе проектирования ставится обычно задача определения наилучших, в некотором смысле, структуры или значений параметров объектов. Такая задача называется оптимизационной. Многим известно, что динамическое программирование разделяет более сложную задачу на несколько легких подзадач, тем самым в этом становится легче разобраться и, к тому же, это оптимизирует программу. Это подобно методу «разделяй и властвуй». Алгоритм, основанный на динамическом программировании, решает каждую из подзадач единожды и запоминает ответы в специальной таблице. Это позволяет не вычислять заново ответ к уже встречавшейся подзадаче.

Задача о порядке перемножения матриц — классическая задача динамического программирования. С помощью динамического программирования можно оптимизировать данную задачу путем расставления скобок. При правильной расстановке скобок, время выполнения перемножения данной последовательности матриц уменьшится, как и количество скалярных операций.

Для обучения студентов и повышения их качества знаний необходимо создать программный продукт, демонстрирующий работу программы, которая оптимизирует умножение матриц методом динамического программирования, предоставляющий теорию по данной теме, а также проводящий тестирование по ней.

Целью создания учебно-демонстрационной программы «Оптимизация умножения матриц методом динамического программирования» является автоматизация обучения и контроля знаний по данной теме.

Назначение программы — повышение уровня знаний по данной теме и снижение нагрузки на преподавателя.

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Анализ предметной области

1.1.1 Динамическое программирования

Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

1.1.2 Порядок перемножения матриц

Произведение матриц — ассоциативная операция, которая принимает на вход две матрицы размером $k \times m$ и $m \times n$ и возвращает матрицу размером $k \times n$, потратив на это $k*m*n$ операций умножения.

Задача о порядке перемножения матриц — классическая задача динамического программирования, в которой дана последовательность матриц и требуется минимизировать количество скалярных операций для вычисления их произведения. Матрицы предполагаются совместимыми по отношению к матричному умножению (то есть количество столбцов одной матрицы совпадает с количеством строк другой).

Когда матрицы велики по одному измерению и малы по другому, количество скалярных операций может серьёзно зависеть от порядка перемножений матриц.

Чтобы проиллюстрировать, как расстановка скобок при перемножении нескольких матриц влияет на количество выполняемых операций, рассмотрим пример, в котором перемножаются 3 матрицы A_1, A_2, A_3 размерами, соответственно, 10×100, 100×5 и 5×50. Существует 2 способа их перемножения (расстановки скобок): $((A_1, A_2), A_3)$ *и* $(A_1 (A_2 A_3))$. В первом случае нам потребуется $10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 = 7500$ скалярных умножений, а во втором случае $100 \cdot 5 \cdot 50 + 10 \cdot 100 \cdot 50 = 75000$ умножений — разница налицо. Поэтому может быть выгоднее потратить некоторое время на предобработку, решив, в каком порядке лучше всего умножать, чем умножать сразу в лоб.

Таким образом, даны n матриц $A_1 : p_0 x\ p_1$, $A_2 : p_1 x\ p_2$, …, $A_n : p_{n-1} x\ p_n$. Требуется определить, в каком порядке перемножать их, чтобы количество операций умножения было минимальным.

1.1.3 Описание структуры оптимального решения

Для задачи об умножении последовательности матриц поиск оптимальных решений методом динамического программирования будет выглядеть следующим образом. Обозначим для удобства через $A_{i..j}$ матрицу, являющуюся произведением матриц $A_iA_{i+1} \dots A_j$. Оптимальная расстановка скобок в произведении $A_{i..j}$ разрывает последовательность между A_k и A_{k+1} для некоторого k, удовлетворяющего неравенству $i \le k < j$. Иными словами, при вычислении произведения, диктуемом этой расстановкой скобок, мы сначала вычисляем произведения $A_{1..k}$ и $A_{k+1..j}$, а затем перемножаем их и получаем окончательный ответ $A_{1..j}$. Стало быть, стоимость этой оптимальной расстановки равна стоимости вычисления матрицы $A_{1..k}$, плюс стоимость вычисления матрицы $A_{k+1..j}$, плюс стоимость перемножения этих двух матриц.

Чем меньше умножений нам потребуется для вычисления $A_{i..k}$ и $A_{k+1..j}$, тем меньше будет общее число умножений. Стало быть, оптимальное решение задачи о перемножении последовательности матриц содержит оптимальные решения задач о перемножении её частей. Это и позволяет применить динамическое программирование.

1.1.4 Рекурсивное нахождение оптимального решения

В задаче о перемножении последовательности матриц в качестве вспомогательной задачи выбирается задача об оптимальной расстановке скобок в последовательности $A_{i..j}$ при $1 \le i \le j \le n$. Пусть m[i][j] – минимальная стоимость, необходимая для вычисления умножения матрицы $A_{i..j}$. Тогда m[1][n] – минимальное количество скалярных умножений матрицы $A_{1..n}$.

Если i = j, то задача становится тривиальной, то есть последовательность матриц состоит всего из одной матрицы, следовательно, $A_{i..j} = A_j$. Таким образом, при i = 1, 2, …, n элементы главной диагонали, то есть m[i][i] = 0. Предположим, что в результате оптимальной расстановки скобок последовательность $A_iA_{i+1} \dots A_j$ разбивается между матрицами $A_{i..k}$ и $A_{k+1..j}$. Тогда величина m[i][j] равна минимальной стоимости вычисления частных произведений $A_{i..k} * A_{k+1..j}$ плюс стоимость умножения этих матриц друг на друга. Если вспомнить, что каждая матрица A_i имеет размеры p_{i-1} на p_i, то нетрудно понять, что для вычисления произведения матриц $A_{i..k} * A_{k+1..j}$ понадобится $p_{i-1} * p_k * p_j$ скалярных умножений. Таким образом получаем:

$$m[i][j] = m[i][k] + m[k+1][j] + p_{i-1}p_kp_j \qquad 1$$

Поскольку в оптимальной расстановке скобок необходимо использовать одно из значений k, а именно i, i + 1, …, j – 1 – всё, что нужно сделать, - проверить все возможности и выбрать среди них наилучшую. Таким образом, рекурсивное определение оптимальной расстановки скобок в произведении $A_{i..j}$ принимает вид:

$$m[i][j] = \begin{cases} 0 & \text{при } i = j, \\ \min\limits_{i \le k < j} \{m[i][k] + m[k+1][j] + p_{i-1} p_k p_j\} & \text{при } i < j. \end{cases} \quad (2)$$

Величины m[i][j] равны минимальных количествам скалярных умножений вспомогательных задач (подзадач), а величина m[1][n] равна минимальной стоимости матрицы $A_{i..j}$. Чтобы легче было проследить за процессом построения оптимального решения, обозначим через s[i][j] значение k, при котором последовательность $A_{i..j}$ разбивается на две подпоследовательности в процессе оптимальной расстановки скобок.

1.1.5 Тренировка и проверка знаний

Чтобы удостовериться, что студент знает теоретический материал в достаточной мере, ему необходимо пройти тестирование по данной теме.

Студенту будут предоставляться вопросы с одним правильным ответом. При этом желательно, чтобы количество вопросов было достаточным для того, чтобы полагать, что данный материал усвоен студентом, но и не избыточным, чтобы это занятие не показалось ему утомительным. Также для проверки качества знаний студента будет лучше, если ему будут предоставляться вопросы открытого типа.

При этом в базе будут содержаться всего десять вопросов, пять из которых будут выбраны случайным образом и добавлены в тест. Варианты ответов каждого вопроса, как и сами вопросы, при повторном прохождении теста перемешиваются.

Каждый правильный ответ на вопрос будет равен одному баллу. Таким образом, если пользователь ввел 5 правильных ответов, то будет выставляться оценка 5, 4 правильных ответа – 4, 3 правильных ответа – 3, если меньше, то 2.

1.2 Технология обработки информации

1.2.1 Диаграмма вариантов

Анализ предметной области показал, что программа рассчитана на двух пользователей: пользователь и администратора.

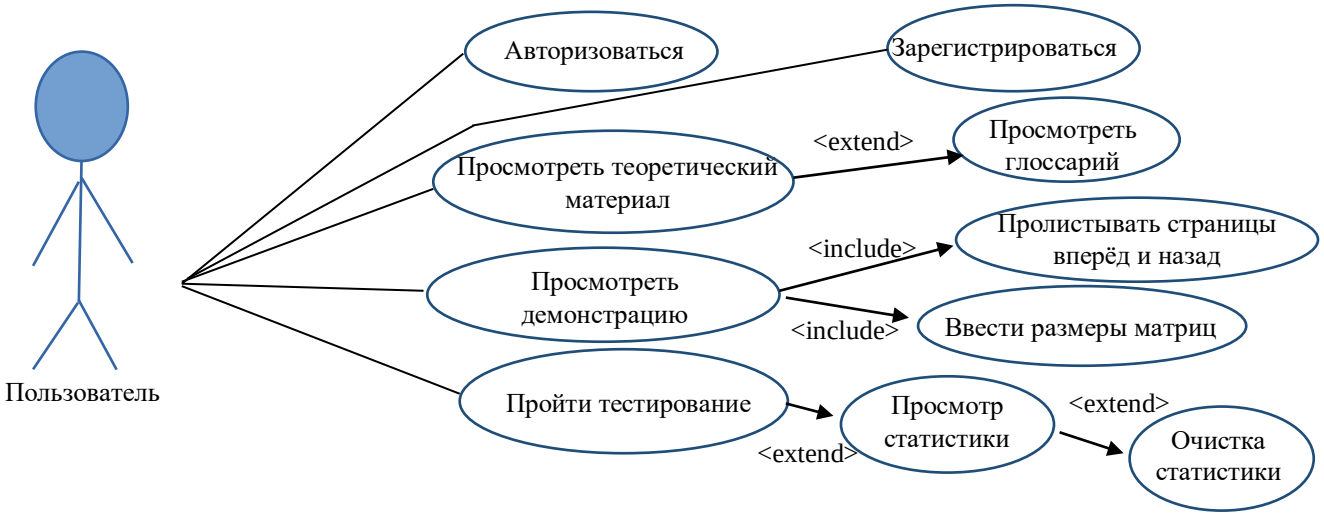

Рисунок 1.1 – Диаграмма вариантов использования

Пользователь может авторизоваться, если нет аккаунта – зарегистрироваться; просмотреть теоретический материал, пролистывая страницы вперёд или назад; просмотреть демонстрацию работы, с возможностью выхода в меню; пройти тест на заданную тему, после чего посмотреть свою статистики и очистить её.

1.2.2 Диаграмма классов

В ходе анализа предметной области была разработана диаграмма классов. Данная диаграмма представлена в Приложении 3.

Главным классом выступает menu, он включает в себя:

- демонстрацию работы программы;
- тестирование test;
- теория theory;

Тестирование включает в себя класс testview, в котором хранятся структуры теста (вопросы и правильные ответы), и структура ответа пользователя на вопросы.

Демонстрация работы алгоритма использует класс demo для формирования демонстрации. В нем реализован алгоритм поиска минимального количества скалярных умножений последовательности матриц и расстановки скобок для этого методом динамического программирования.

Класс theory предназначен для вывода теории и глоссария на экран.

1.2.3 Форматы данных

Тексты титульного листа, теории, глоссария, статистики оформлены с использованием HTML (.html). Файлы с тестом и данными пользователей оформлены как текстовые (.txt).

Текст теории разделен на файлы (страницы) и находится в файлах с именами «Theory.html», «SecondTheory.html», «ThirdyTheory.html», а глоссарий в файле «Dictionary.html». Статистика находится в файле «Statistic.html», в ней дата и время окончания теста, время тестирования и процент правильных ответов.

Материалы для тестирования находятся в файле «Test.txt». Для оформления текста используются теги, с помощью которых определяется вопрос это или ответ, а также тип вопроса и правильность ответов.

Вопрос закрытого типа: тег «?» указывает на то, что на следующей строке файла находится текст вопроса. После текста вопроса размещаются строки с ответами на вопрос (количество строк - 4). Каждая строка вначале имеет свой тег, тег «-» означает, что встречен дистрактор, а тег «+» означает, что встречен правильный ответ.

Вопрос открытого типа: если же после тега «?» последовал не текст вопроса, а тег «!» значит, что встречен вопрос открытого типа. После тега «!» - текст вопроса. На следующей строке после текста вопроса размещается правильный ответ на вопрос открытого типа.

Пример файла:

?

В чем суть динамического программирования?

-Решение задач путем выделения динамической памяти

-Оптимизация операции умножения

+Решение задач путем разделения на подзадачи

-Оптимизация хранения больших объемов данных

?

!

Метод динамического программирования называют методом "Разделяй и ..."

Властвуй

1.2.4 Алгоритмы работы демонстрации

1.2.4.1 Алгоритм вычисления оптимальной стоимости и расстановки скобок

Дано:p – множество размеров матриц, где length[**p**] = **n**+1,

m – матрица оптимальных стоимостей,

s – матрица расстановки скобок

n- Количество матриц – присвоить length[**p**] – 1

Присвоить главной диагонали матрицы m нули

Пока **l** = 2 меньше **n**

| Пока **i** = 0 меньше **n – l + 1**

| | **j = i + l – 1**

| | **m[i][j] =** inf

| | Пока **k** = **i**меньше **j**

| | | **q = m[i][k] + m[k+1][j] + p[i]*p[k+1]*p[j+1]**

| | | Если **q<m[i][j]**

| | | | **m[i][j] = q**;

| | | | **s[i][j] = k**;

| | | Конец ветвления

| | Конец цикла

| Конец ветвления

Конец ветвления

Вернуть **m, s**

1.2.4.2 Алгоритм рекурсии для расставления скобок

Дано: start – открывающая скобка,

end– закрывающая скобка

Если **Start** !=**End**

|**End++**

| Если **start**, то вывести открывающую скобку перед **start + 1** матрицей

| Конец ветвления

| Если **end**, то вывести закрывающую скобку после **end** матрицей

| Конец ветвления

Конец ветвления

1.2.4.3 Алгоритм рекурсии для расставления скобок

Дано: s – матрица расстановки скобок,

i– индекс открывающей скобки,

j – количество столбцов

Если j>i

| Выполнить алгоритм **1.2.4.2** с входными данными **i, s[i][j]**

| Выполнить алгоритм **1.2.4.2** с входными данными **s[i][j] + 1, j**

| Выполнить алгоритм **1.2.4.3** с входными данными **s, i, s[i][j]**

| Выполнить алгоритм **1.2.4.3** с входными данными **s, s[i][j] + 1, j**

Конец ветвления

1.2.5 Алгоритм входа в программу

Дано: логин **login**,

пароль **password**;

При запуске программы потребуется ввод логина и пароля

Ввести логин и пароль

FlagEnter = false

Открыть файл Users.txt на чтение

Если файл открыт

| Пока не конец файла

| | Считать логин и пароль с файла

| | Если логин и пароль с файла равны введенным логину и паролю

| | | Присвоить FlagEnter = true

| | Конец ветвления

| Конец цикла

Конец ветвления

Если FlagEnter

| Вывести сообщение о удачной авторизации

| Открыть меню

Иначе вывести сообщение о неудачной авторизации и попробовать снова

Конец ветвления

1.2.6 Алгоритм тестирования

1.2.6.1 Алгоритм считывания теста с файла

Создать переменную **tmp** типа string;

Открыть на чтение текстовый файл «Test.txt»;

Если файл открыт

| Пока не конец файла

| | Прочитать строку из файла в **tmp**

| | Пока**tmp** != «?»

| | | Прочитать строку из файла в **tmp**

| | | Записать в переменную тип вопроса **tmp**

| | | Если тип вопроса == «!»

| | | | Прочитать строку из файла в **tmp**

| | | | Переменной текст вопроса присвоить **tmp**

| | | | Прочитать строку из файла в **tmp**

| | | | Переменной эталонный ответ на вопрос открытого типа присвоить **tmp**

| | | Иначе переменной текст вопроса присвоить **tmp**

| | | Конец ветвления

| | | Прочитать строку из файла в **tmp**

| | | Если тип вопроса != «!»

| | | | Пока **tmp** != «?» и не конец файла

| | | | | Записать ответы на вопрос закрытого типа в вектор, а также их корректность

| | | | | Прочитать строку из файла в **tmp**

| | | | Конец цикла

| | | Конец ветвления

| | | Сохранить готовый вопрос в вектор вопросов

| | | Очистить вектор ответов на вопрос закрытого типа

| | Конец цикла

| Конец цикла

Конец ветвления

Вернуть тест

1.2.6.2 Алгоритм вывода вопроса

Вывести вопрос

Если тип вопроса == «!»

| Вывести варианты ответа

Иначе

| Очистить строку для ввода ответа на вопрос открытого типа

Конец ветвления

1.2.6.3 Алгоритм проверки ответа

Если тип вопроса != «!»

|Присвоить переменной Truвариант ответа пользователя на данный вопрос

| Если правильно

| | В вектор правильных ответов добавить true

| | Счётчику правильных ответов прибавить 1

| | Вывести сообщение о том, что пользователь ответил правильно

| Иначе

| | В вектор правильных ответов добавить false

| | Вывести сообщение о том, что пользователь ответил неправильно и вывести правильный Ответ

| Конец ветвления

Иначе

| Считать из строки ответ пользователя на вопрос открытого типа и сравнить с эталонным

| Если правильно

| | В вектор правильных ответов добавить true

| | Счётчику правильных ответов прибавить 1

| | Вывести сообщение о том, что пользователь ответил правильно

| Иначе

| | В вектор правильных ответов добавить false

| | Вывести сообщение о том, что пользователь ответил неправильно и вывести правильный Ответ

| Конец ветвления

Конец ветвления

1.2.6.4 Алгоритм начала тестирования

Дано: number– счётчик вопроса (какой по счёту идёт вопрос)

Выполнить алгоритм **1.2.6.2**

Если **number** < 5

| number++

| Выполнить алгоритм **1.2.6.3**

Иначе

| Перестать выводить вопросы

| Вычислить процент правильных ответов

| Получить оценку тестирования

| Вывести подсвеченную статистику ответов на экран (зеленый – правильный, красный - нет)

| Записать статистику тестирования в файл

Конец ветвления

1.3 Входные и выходные данные

Входные данные:

- размеры матриц;
- логин и пароль пользователя;
- ответ пользователя на вопрос тестирования;

Выходные данные:

- прочитанный из файла текст;
- демонстрация работы алгоритма;
- таблица оптимальных стоимостей;
- таблица расставления скобок;
- тестовые задания;
- результат тестирования.

1.4 Системные требования

Рекомендуемая конфигурация:

- Intel-совместимый процессор с частотой не менее 1,6 ГГц;
- не менее 512 МБ ОЗУ;
- не менее 40 МБ свободного места на диске;
- дисковод CD-ROM/DVD-ROM.

Операционная система: Windows 7 (x86).

2 РАБОЧИЙ ПРОЕКТ

2.1 Общие сведения о работе системы

Программный продукт разработан в интегрированной среде QtCreator 4.10.2, язык C++ с библиотекой Qt 5.14.0. Программа работает под управлением операционной системы Windows 7 (x86) и более поздними. Стартовый файл Example.exe.

2.2 Функциональное назначение программного продукта

Данный программный продукт разработан для ознакомления с методом динамического программирования для оптимизации умножения матриц.

Программа имеет следующие функциональные возможности:

- предоставление любому пользователю демонстрации работы поиска оптимальных стоимостей при умножении нескольких матриц и при этом расстановки скобок;
- предоставление любому пользователю теории на тему «Оптимизация умножения матриц методом динамического программирования»;
- предоставление любому пользователю тест на тему «Оптимизация умножения матриц методом динамического программирования».

Программа имеет следующие функциональные ограничения:

- количество матриц, участвующих в демонстрации не больше 4;
- количество вариантов ответа на вопрос закрытого типа 4;

2.3 Инсталляция и выполнение программного продукта

Для выполнения программы необходимо:

1. Скопировать на жесткий диск компьютера папку Example.
2. Запустить исполняемый файл Example.exe.

Все файлы, лежащие в папке «Example», представлены в таблице 2.1.

Таблица 2.1 – Файлы из папки «Example»

Название	Описание
«Example.exe»	Основной исполняемый файл программы
libgcc_s_seh-1.dll	Модуль библиотеки Qt
libstdc++-6.dll	Модуль библиотеки Qt
libwinpthread-1.dll	Модуль библиотеки Qt
Qt5Core.dll	Основные классы из библиотеки Qt
Qt5Gui.dll	Классы пользовательского интерфейса из библиотеки Qt
Qt5Widgets.dll	Классы виджетов из библиотеки Qt
Dictionary.html	Файл, содержащий глоссарий

Продолжение таблицы 2.1

Название	Описание
Theory.html SecondTheory.html ThirdyTheory.html	Файлы, содержащие теорию по данной теме
Statistic	Файл, содержащий статистику
Test	Файл, содержащий тест (вопросы с ответами)
User	Файл, содержащий логины и пароли зарегистрированных пользователей
platforms	Папка, содержащая модуль из библиотеки Qt «qwindows.dll»

2.4 Описание программы

В таблице 2.2 приведена структура Question, используемая в программе.

Таблица 2.2 – Описание структуры Question

Поле	Тип	Назначение
TextQues	QString	Текст вопроса
QuestionType	QString	Тип вопроса (открытый, закрытый)

В таблице 2.3 приведена структура Answer, используемая в программе.

Таблица 2.3 – Описание структуры Answer

Поле	Тип	Назначение
TextAnsw	QString	Текст вопроса
Correct	bool	Правильность вопроса

В таблице 2.4 приведена структура Task, используемая в программе.

Таблица 2.4 – Описание структуры Task

Поле	Тип	Назначение
TaskQuest	Question	Вопрос (текст вопроса и его тип)
CloseAnsw	QVector<Answer>	Варианты ответа вопроса закрытого типа
OpenAnswer	QString	Правильный ответ на вопрос открытого типа

Структура Test включает в себя поле AllTest типа QVector<Task>, где находятся все вопросы с ответами, считанные с файла.

В таблице 2.5 приведены функции, используемые в программе (модуль MainWindow).

Таблица 2.5 – Функции модуля MainWindow

Прототип	**Назначение**
void on_Cancel_clicked()	Выход из программы
void on_Enter_clicked()	Попытка авторизации
void on_Reg_clicked()	Открытие окна с регистрацией
void on_EnterReg_clicked()	Попытка регистрации
void on_CancelInAutho_clicked()	Вернуться к авторизации

В таблице 2.6 приведены функции, используемые в программе (модуль menu).

Таблица 2.6 – Функции модуля menu

Прототип	**Назначение**
void on_theory_clicked()	Переход к теории
void on_demo_clicked()	Переход к демонстрации
void on_test_clicked()	Переход к тесту
void on_Exit_clicked()	Выход из программы
void ReturnMenu()	Для возврата из других модулей в меню

В таблице 2.7 приведены функции, используемые в программе (модуль theory).

Таблица 2.7 – Функции модуля theory

Прототип	**Назначение**
voidon_NextPageTheory_clicked()	Переход на следующую страницу теории
voidon_PreviousPageTheory_clicked()	Переход на предыдущую страницу теории
void on_BackMenu_clicked()	Выход в меню
void on_pushButton_clicked()	Переход с глоссария в теорию
void on_TextTheo_anchorClicked(constQUrl&arg1)	Переход по ссылке из теории в глоссарий
void on_pushButton_Dict_clicked()	Переход из теории в глоссарий

В таблице 2.8 приведены функции, используемые в программе (модуль demo).

Таблица 2.8 – Функции модуля demo

Прототип	**Назначение**
void on_BackMenu_clicked()	Выход в меню
void FillTableS(int Row, int Col)	Вывод таблицы s, заполненной 0
void FillTableM(int Row, int Col)	Вывод таблицы m, заполненной «inf»
void FillTableA(int Row, int Col)	Вывод таблицы (матрицы) A
void FillTableB(int Row, int Col)	Вывод таблицы (матрицы) B

Продолжение таблицы 2.8

Прототип	**Назначение**
void FillTableC(int Row, int Col)	Вывод таблицы (матрицы) B
void FillTableD(int Row, int Col)	Вывод таблицы (матрицы) B
void Fill_p()	Заполнение множества размеров матриц
void on_ShowPage1_clicked()	Вывод всех таблиц (матриц) на экран
void on_previousPage_2_clicked()	Переход с 3 на 2 страницу демонстрации
void on_previousPage_1_clicked()	Переход с 2 на 1 страницу демонстрации
void on_nextPage_2_clicked()	Переход с 1 на 2 страницу демонстрации
void on_nextPage_3_clicked()	Переход с 2 на 3 страницу демонстрации
void TableMS(size_t q, size_t k, size_t i, size_t j)	Вывод пошагово таблиц s и m
void Matrix_Chain_Rekursiya(QVector<QVector<size_t> > s, size_ti, size_t j)	Рекурсивная функция для расставления скобок
void rastavlenie_scobok(size_t Start, size_t End)	Расставление скобок

В таблице 2.9 приведены функции, используемые в программе (модуль test).

Таблица 2.9 – Функции модуля test

Прототип	**Назначение**
void on_BackMenu_clicked()	Выход в меню
void DisplayTask()	Вывод вопроса теста (с вариантами ответов, если закрытый тип)
void on_radioButton_clicked()	Выбор первого варианта ответа
void on_radioButton_2_clicked()	Выбор второго варианта ответа
void on_radioButton_3_clicked()	Выбор третьего варианта ответа
void on_radioButton_4_clicked()	Выбор четвертого варианта ответа
void on_pushButton_Enter_clicked()	Принятие выбранного варианта ответа
void CheckAnswer()	Проверка ответа пользователя
QStringRightAnswer()	Возвращает правильный ответ на заданный вопрос закрытого типа
Test MakeTest(std::string FileName)	Считывание теста с файла (генерация)
void TotalRes()	Вывод и подсветка решенных заданий(вопросов)
void TotalMark()	Итоговая оценка
void StatInFile(double percent)	Запись статистики тестирования в файл
void on_pushButton_Stat_clicked()	Вывод статистики на экран

Продолжение таблицы 2.9

Прототип	**Назначение**
void on_pushButton_ clicked()	Очистка файла со статистикой
void SetLog(QStringuserLogin)	Получает логин пользователя для сохранения в статистику
void show()	Перегрузка оператора show()

2.5 Разработанные меню и интерфейсы

После запуска программы будет предоставлено окно авторизации (рис.2.1).

Рисунок 2.1 – Окно с авторизацией

1. Строки для ввода пароля и логина
2. Кнопка для подтверждения введенных данных
3. Кнопка для вывода окна с регистрацией
4. Кнопка для выхода

Если у пользователя нет логина и пароля – он может зарегистрироваться (рис. 2.2).

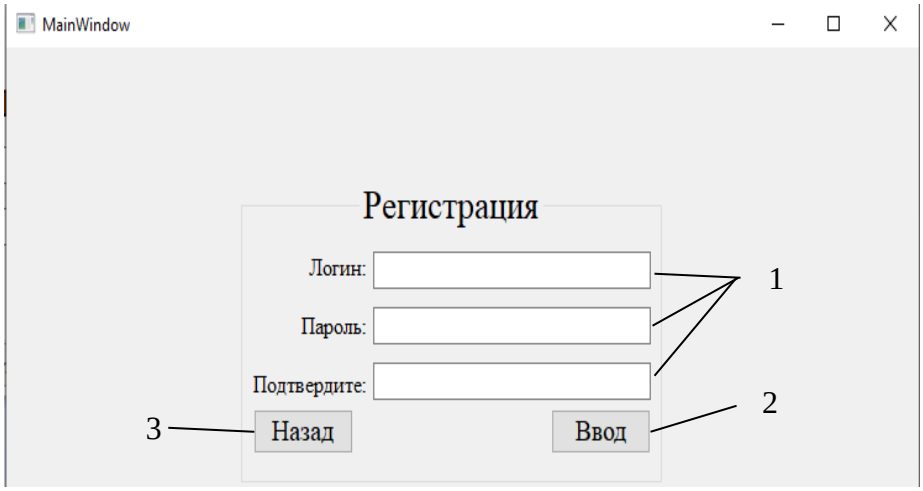

Рисунок 2.2 – Окно с регистрацией

1. Строки для ввода данных при регистрации
2. Кнопка для подтверждения введенных данных
3. Кнопка для выхода с регистрации в окно с авторизацией

При удачной попытке авторизации или регистрации будет выведено меню (рис. 2.3).

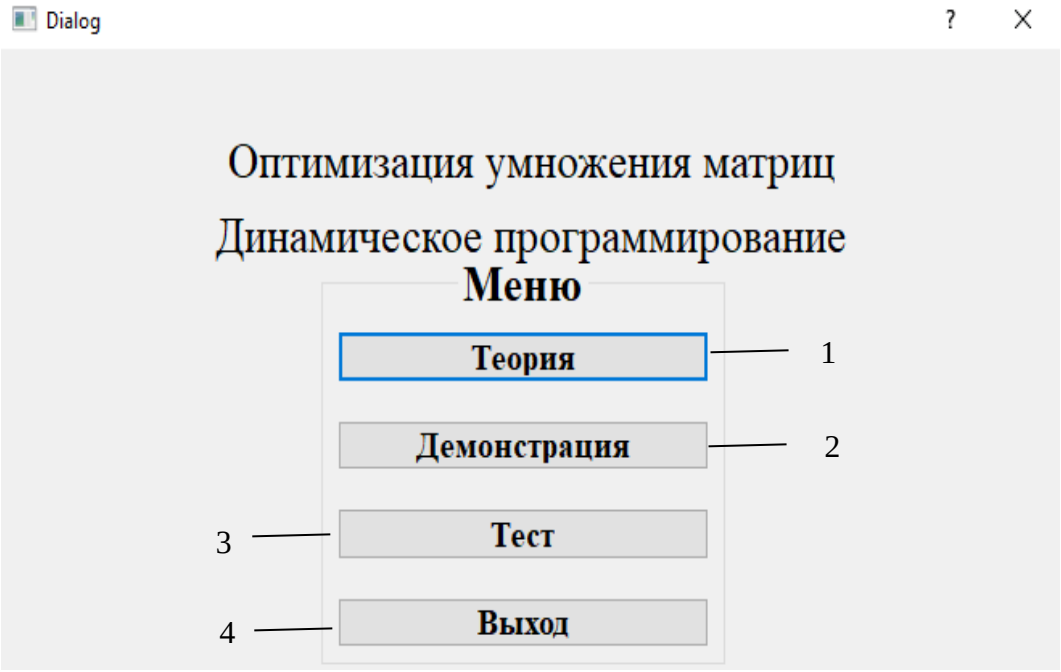

Рисунок 2.3 – Окно с меню

1. Кнопка для вывода теории
2. Кнопка для вывода демонстрации
3. Кнопка для вывода теста
4. Кнопка для выхода в окно с авторизацией

После нажатия на кнопку «Теория» на экран будет выведено теория на тему «Оптимизация умножения матриц методом динамического программирования» (рис. 2.4).

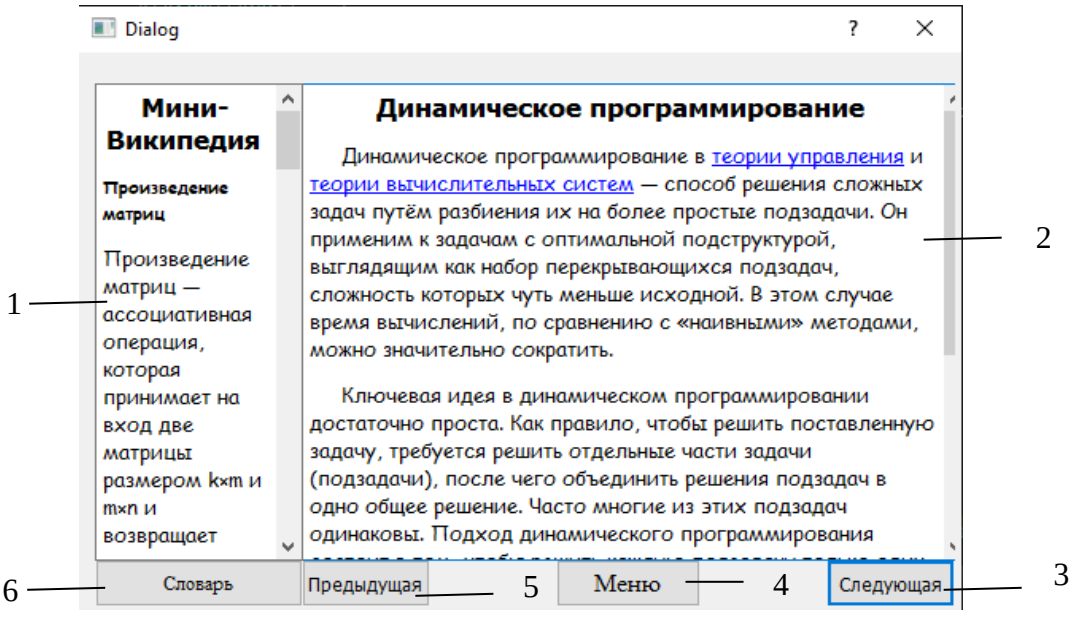

Рисунок 2.4 – Вывод теории

1. Глоссарий в маленьком окне слева
2. Тест теории какой-либо страницы
3. Кнопка для перелистывания теории на следующую страницу
4. Кнопка для выхода в меню программы
5. Кнопка для перелистывания теории на предыдущую страницу
6. Кнопка для того, чтобы открыть глоссарий в полном виде

Слева от теории расположен глоссарий для удобства работы с ссылками. Если же данное расположение неудобно, можно нажать на кнопку «Словарь», после чего на экран будет выведен глоссарий в более удобном виде (рис. 2.5).

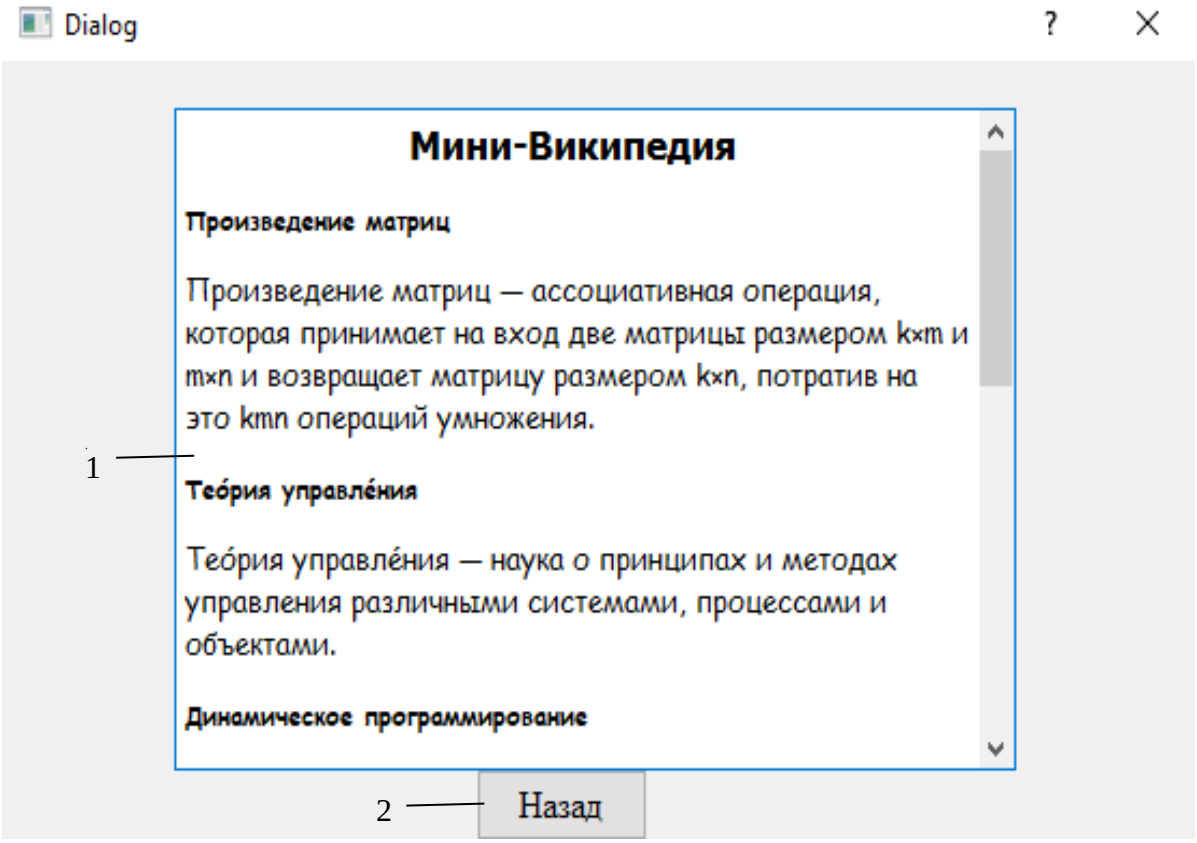

Рисунок 2.5 – Глоссарий

1. Глоссарий в большом окне
2. Кнопка для возвращение обратно в окно с теорией

Если в меню выбрать пункт «Демонстрация», то будет выведено окно с возможностью ввода размеров матрицы для вычисления оптимальных стоимостей и оптимальной расстановки скобок (рис. 2.6).

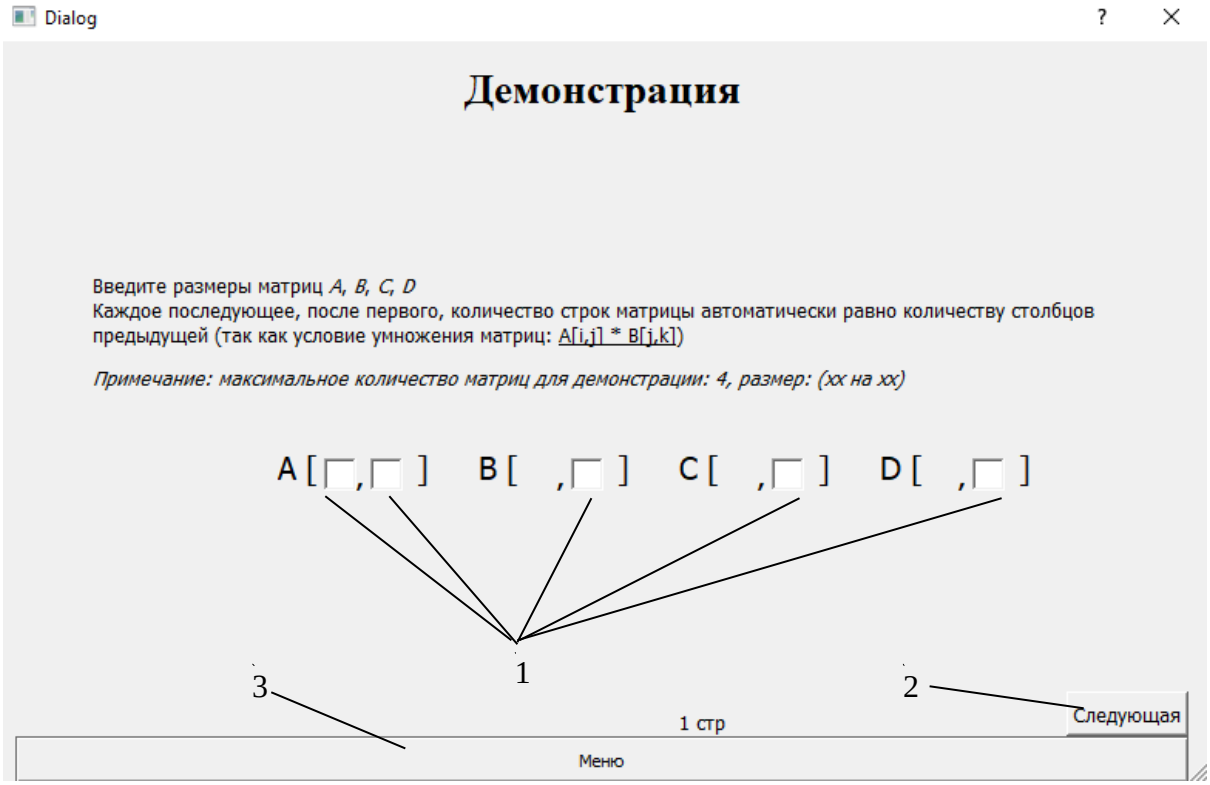

Рисунок 2.6 – Первая страница демонстрации

1. Строки для ввода размеров матриц
2. Кнопка для перехода на следующую страницу демонстрации
3. Кнопка для выхода в меню программы

На следующей странице показано пошаговое заполнение таблицы оптимальных стоимостей и таблицы расставления скобок (рис. 2.7).

Рисунок 2.7 – Шаг демонстрации заполнения таблиц

1. Матрица оптимальных стоимостей
2. Матрица расставления скобок
3. Между какими матрицами расставляются скобки и ищется минимальное количество скалярных умножений
4. Кнопка для перехода на следующую страницу демонстрации
5. Строка для ввода количества секунд, которое будет между шагами заполнения таблиц
6. Кнопка для начала пошагового заполнения таблиц
7. Количество скалярных операций между данными в пункте 3 матрицами
8. Номер матрицы, на которой разделяется последовательность матриц, данная в пункте 3, на две подзадачи
9. Кнопка для перехода на предыдущую страницу демонстрации
10. Кнопка для выхода в меню программы

Следующая страница предоставляет итог с расставленными скобками между данной последовательностью матриц (рис. 2.8).

Рисунок 2.8 – Итоговая расстановка скобок

1. Последовательность матриц с оптимально расставленными скобками
2. Кнопка для перехода на предыдущую страницу демонстрации
3. Кнопка для выхода в меню

При выборе в меню пункта «Тест» будет доступно прохождение тестирования с вопросами закрытого типа (рис. 2.9).

Рисунок 2.9 – Вопрос закрытого типа

1. Варианты ответов на вопрос закрытого типа
2. Кнопка для подтверждения своего ответа
3. Кнопка для выхода в меню

Также при выводе вопроса может попасться вопрос открытого типа (рис. 2.10)

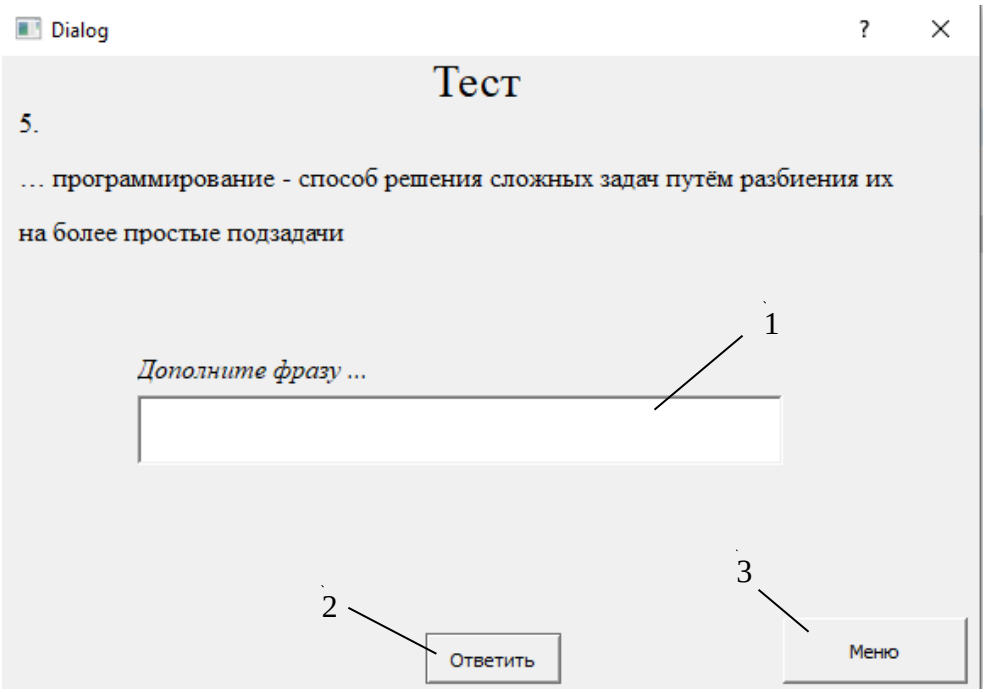

Рисунок 2.10 – Вопрос открытого типа

1. Строка для ввода ответа на вопрос открытого типа
2. Кнопка для подтверждения своего ответа
3. Кнопка для выхода в меню

После завершения тестирования будут предоставлены результаты (рис. 2.11).

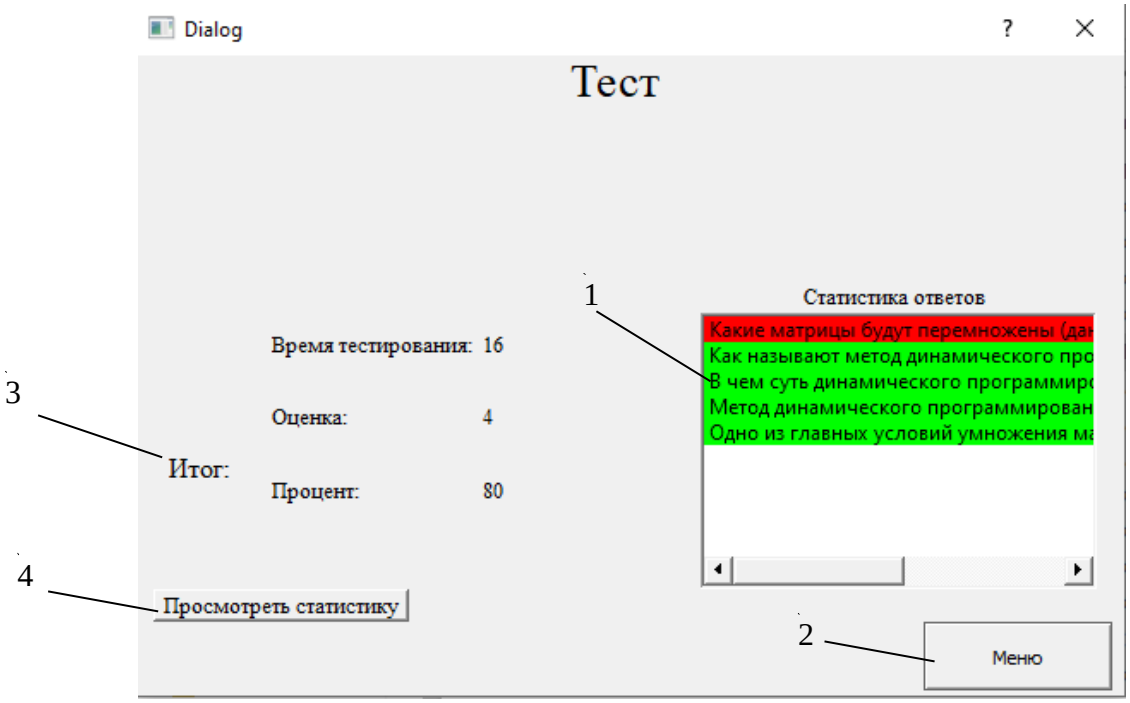

Рисунок 2.11 – Итог тестирования

1. Список ответов (красный цвет – текст вопроса, на который ответили неверно, зеленый цвет – текст вопроса, на который ответили верно)
2. Кнопка для выхода в меню
3. Итоговый результат тестирования
4. Кнопка для просмотра статистики

Также можно просмотреть статистику тестирований, нажав на кнопку «Просмотреть статистику» (рис. 2.12).

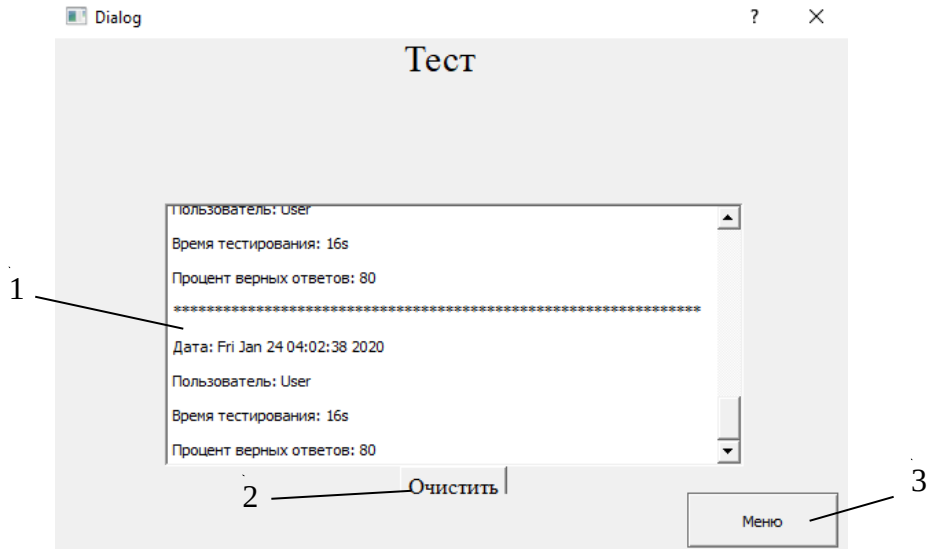

Рисунок 2.12 – Статистика тестирований

1. Статистика тестирований
2. Кнопка для очистки файла со статистикой
3. Кнопка для выхода в меню

В случае успешной авторизации на экране появится окно с сообщением (рис. 2.13)

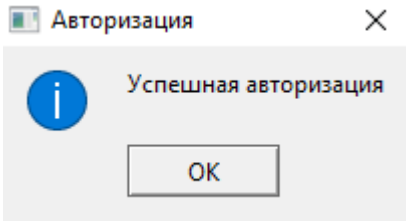

Рисунок 2.13 – Окно с сообщением об удачно авторизации

В случае неуспешной авторизации на экране появится другое окно с предупреждением (рис. 2.14)

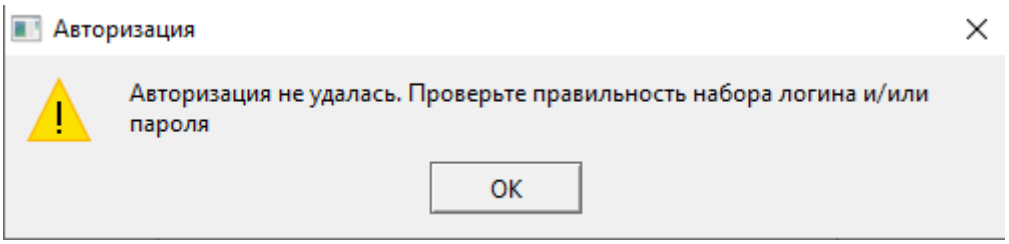

Рисунок 2.14 – Окно с предупреждением о том, что авторизация не удалась

При регистрации возникают такие случаи, когда заполняешь не все, что требуется, либо вводишь это не в корректной форме. В таком случае будет выведено окно с предупреждением (рис. 2.15)

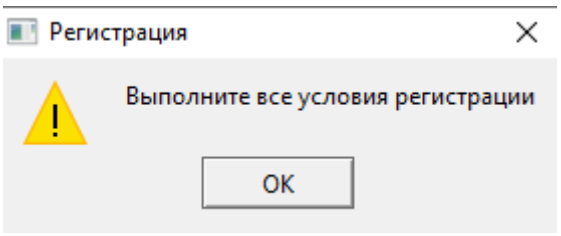

Рисунок 2.15 – Окно с предупреждением о не выполненных условиях регистрации

При прохождении теста может быть выбран неправильный ответ. При данных обстоятельствах будет выведено соответствующее предупреждение (рис. 2.16)

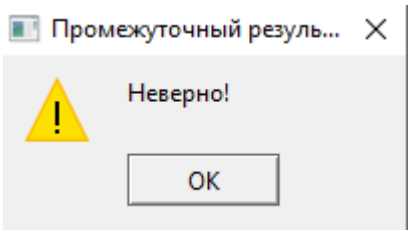

Рисунок 2.16 – Окно с предупреждением о том, что на вопрос был дан неправильный ответ

При неправильном ответе на поставленный вопрос система выведет правильный ответ (рис. 2.17)

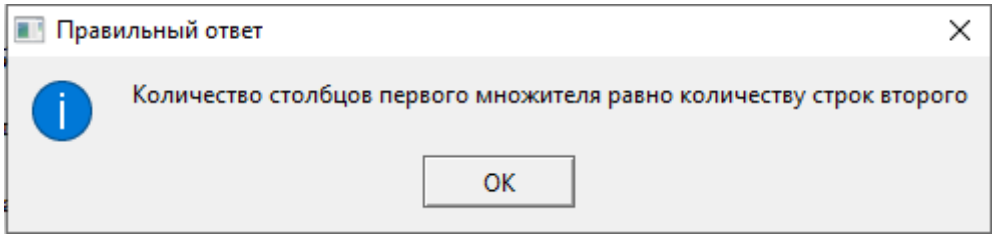

Рисунок 2.17 – Окно с правильным ответов на поставленный вопрос

Если же на вопрос был дан верный ответ система выведет сообщение об удачном промежуточном результате (рис. 2.18)

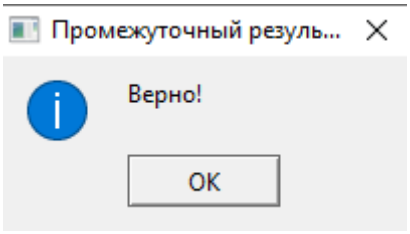

Рисунок 2.18 – Окно с удачным промежуточным результатом тестирования

В случае появления других сообщений следует обратиться к разработчику.

3 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

3.1 Проверка работоспособности авторизации

1. Запустить программу на выполнение. Появится окно авторизации (см. рис.2.1).
2. Ввести логин «Username» и пароль «user» и нажать «Ввод». Убедиться, что появится окно с сообщением (см. рис. 2.14).
3. Нажать «Ок». Убедиться, что окно закрылось.
4. Ввести логин «Username» и пароль «User» и нажать «Ввод». Убедиться, что выведется окно сообщение программы (см. рис. 2.13) и меню программы (см. рис. 2.3).

3.2 Проверка работоспособности регистрации

1. Запустить программу на выполнение. Появится окно авторизации (см. рис. 2.1).
2. Нажать «Регистрация». Убедиться, что выведено окно с регистрацией (см. рис. 2.2).
3. Ввести логин и пароль, подтвердить пароль неправильно, нажать «Ввод» Убедиться, что будет выведено окно с предупреждением (см. рис. 2.15)
4. Ввести логин и пароль заново, подтвердить пароль правильно, нажать «Ввод». Убедиться, что будет выведено меню программы (см. рис. 2.3).

3.3 Проверка работоспособности вывода теории

1. Запустить программу на выполнение. Появится окно авторизации (см. рис.2.1).
2. Ввести логин «Username» и пароль «User» и нажать «Ввод». Убедиться, что будет выведено меню программы (см. рис. 2.3).
3. Нажать на кнопку «Теория». Убедиться, что появится текст теории (см. рис. 2.4).
4. Нажать на гиперссылку. Убедиться, что словарь открылся на соответствующем пункте.
5. Нажать на «Словарь» и убедиться, что глоссарий открылся в удобном виде (см. рис. 2.5).

3.4 Проверка работоспособности тестирования

1. Запустить программу на выполнение. Появиться окно авторизации (см. рис. 2.1).
2. Ввести логин «Username» и пароль «User» и нажать «Ввод». Убедиться, что будет выведено меню программы (см. рис. 2.3).
3. Нажать на кнопку «Тест». Убедиться, что появится первый вопрос теста (см. рис. 2.9) или (см. рис. 2.10).
4. Выбрать вариант ответа и нажать на кнопку «Ответить».

5. Убедиться, что выведется окно с сообщением об удачном промежуточном результате (см. рис. 2.18), либо окно с предупреждением о неверном ответе (см. рис. 2.16), а затем правильный ответ на неправильно отвеченный вопрос (см. рис. 2.17).
6. Пройти тестирование, вводя ответы. Убедиться, что появится результат вашего тестирования (см. рис. 2.11).
7. Нажать на кнопку «Просмотреть статистику». Убедиться, что появится список результатов тестирования (см. рис. 2.12), и что результат прошедшего теста в этой статистике.

3.5 Проверка работоспособности демонстрации

1. Запустить программу на выполнение. Появится окно авторизации (см. рис.2.1).
2. Ввести логин «Username» и пароль «User» и нажать «Ввод». Убедиться, что будет выведено меню программы (см. рис. 2.3).
3. Нажать на кнопку «Демонстрация». Убедиться, что будет выведена первая страница демонстрации, где нужно ввести размеры матриц (см. рис. 2.6).
4. После ввода размеров нажать на кнопку «Следующая» и убедиться, что выведется пошаговое заполнение матриц оптимальных стоимостей и расстановки скобок (см. рис. 2.7).
5. Нажать на кнопку «Следующая» и убедиться, что будет выведена последовательность матриц с оптимально расставленными скобками (см. рис. 2.8).

ЗАКЛЮЧЕНИЕ

В результате курсового проектирования разработана учебно-демонстрационная программа «Оптимизация умножения матриц путём динамического программирования». Программа предоставляет теоретический материал по теме «Оптимизация умножения матриц путём динамического программирования», визуализирует поиск матрицы оптимальных стоимостей и матрицы расставления скобок, позволяет пройти тестирование, по окончании которого выдаётся сообщение о том, насколько успешно был выполнен тест.

Программа отвечает поставленным требованиям и может быть использована студентами, изучающими динамическое программирование.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кормен Т., Лайзерсон Э.Ч.,Ривест Р.Л., Алгоритмы: построение и анализ / Пер. с англ. Под ред. А. Шеня. – М.: МЦНМО, 2002. – 960 с.: 263 ил.
2. Лаптев В.В., C++. Экспресс-курс. – СПб.: БХВ-Петербург, 2004. – 512 с.: ил..
3. Шлее М., Qt 5.10. Профессиональное программирование на C++ - СПБ.: БХВ-Петербург, 2018 — 1072 с.: ил. - (В подлиннике)

ПРИЛОЖЕНИЕ 1

Техническое задание
на разработку учебно-демонстрационной программы
«Оптимизация умножения матриц путем динамического программирования»

по дисциплине «Алгоритмы и структуры данных»

Направление 09.03.04 Программная инженерия

Исполнитель: студент гр. ДИПРб21 **Иргалиев А.А.**

1 Назначение, цели и задачи разработки

Цель разработки – автоматизация обучения и контроля знаний по теме «Оптимизация умножения матриц путем динамического программирования».

Назначение разработки:

- повышение качества знаний пользователей;
- снижение нагрузки на преподавателя.

Основные задачи, решаемые разработчиком в процессе выполнения курсового проекта:

- анализ предметной области;
- разработка программного продукта в соответствии с требованиями;
- **документирование проекта в соответствии с установленными требованиями.**

2 Характер разработки: прикладная квалификационная работа.

3 Основания для разработки

- Учебный план направления 09.03.04 «Программная инженерия» 2018 года набора.
- Рабочая программа дисциплины «Программирование и информатика».
- Распоряжение по кафедре АСОИУ №____ от «___»_____ 2019 г.

4 **Плановые сроки выполнения – осенний семестр 2019/20 учебного года:**

Начало «15» сентября 2019 г.

Окончание «27» декабря 2019 г.

5 Требования к проектируемой системе

5.1 Требования к функциональным характеристикам

Проектируемая система представляет собой оконное приложение и должна обеспечивать выполнение следующих основных функций:

- Предоставление пользователю теоретического материала по теме «Оптимизация умножения матриц путем динамического программирования»
  - ✓ текст с теоретическим материалом разбит на абзацы и структурирован;
  - ✓ определения, прописанные в теоретическом материале, должны быть собраны в отдельный словарь и в тексте теоретического материала с гиперссылками.
  - ✓ текст хранится в файле.

- Визуализация работы с алгоритмом динамического программирования для поиска оптимальной расстановки скобок для нахождения минимального количества стоимостей:
  - ✓ предоставление пользователю возможность ввести размеры матриц;
  - ✓ ход поиска оптимальной стоимости показан в матрице;
  - ✓ расставление скобок, при котором достигается оптимальная стоимость.
- Тестирование пользователя по теме «Оптимизация умножения матриц путем динамического программирования»:
  - ✓ вопросы закрытого типа, количество предлагаемых вариантов ответа 4, правильных ответов 1;
  - ✓ вопросы открытого типа с одним правильным ответом;
  - ✓ при тестировании пользователю предлагается 5 вопросов, общее количество вопросов в базе не менее 10;
  - ✓ формат файла и вопросы разрабатываются исполнителем самостоятельно;
  - ✓ программа должна подсчитывать количество правильных ответов; пользователь выбирает один ответ из заданных четырёх, нажатием на него; ответ считается верным, если выбран правильный ответ;
  - ✓ при закрытом вопросе программа должна подсчитывать количество правильных ответов;
  - ✓ если ответ правильный, то процент и балл правильных ответов, данных им, прибавляется;
  - ✓ после завершении тестирования должен быть выведен общий итог;
  - ✓ тестирование считается успешным, если результат тестирования 5 баллов.
- Интерфейс программы: текст русский, шрифт кириллический, заголовки.

Система имеет функциональные ограничения:

- количество матриц, участвующий в умножении — 4:
- количество вариантов ответа на вопрос закрытого типа — 4.

5.2 Требования к эксплуатационным характеристикам

Программа не должна аварийно завершаться при любых действиях пользователя.

Время реакции программы на действия пользователя не должно превышать 10 секунд.

5.3 Требования к программному обеспечению:

Программный продукт разработан в интегрированной среде QtCreator, язык Qt.

Операционная система: Windows 7 (x86).

5.4 Требования к аппаратному обеспечению:

Рекомендуемая конфигурация:

- Intel-совместимый процессор с частотой не менее 1,6 ГГц;
- не менее 512 МБ ОЗУ;
- не менее 40 МБ свободного места на диске;
- дисковод CD-ROM/DVD-ROM.

6 Стадии и этапы разработки

6.1 Эскизный проект (ЭП)

- Анализ предметной области.
- Подготовка проектной документации.

6.2 Технический проект (ТП)

- Разработка структур и форм представления данных.
- Разработка структуры программного комплекса.
- Подготовка пояснительной записки.

6.3 Рабочий проект (РП)

- Программная реализация.
- Тестирование и отладка программы.
- Подготовка программной и эксплуатационной документации.

6.4 Эксплуатация (Э)

Описание и анализ результатов проведенного исследования.

7 Требования к документированию проекта

К защите курсового проекта должны быть представлены следующие документы:

- Пояснительная записка к курсовому проекту:
- Презентация доклада.
- Программа, презентация и пояснительная записка к курсовому проекту на оптическом носителе.

Требования к структуре документов определены соответствующими стандартами ЕСПД.

Требования к оформлению определены соответствующими методическими указаниями.

8 Порядок контроля и приемки

Контроль выполнения курсового проекта проводится руководителем поэтапно в соответствии с утвержденным графиком выполнения проекта.

На завершающем этапе руководитель осуществляет нормоконтроль представленной исполнителем документации и принимает решение о допуске (недопуске) проекта к защите.

Защита курсового проекта проводится комиссией в составе не менее двух человек, включая руководителя проекта.

В процессе защиты проекта исполнитель представляет документацию, делает краткое сообщение по теме разработки и демонстрирует ее программную реализацию.

При выставлении оценки учитывается:

- степень соответствия представленной разработки требованиям технического задания;
- качество программной реализации, документации и доклада по теме проекта;
- соблюдение исполнителем графика выполнения курсового проекта.

9 Литература

1. Кормен Т., Лайзерсон Э.Ч.,Ривест Р.Л., Алгоритмы: построение и анализ / Пер. с англ. Под ред. А. Шеня. – М.: МЦНМО, 2002. – 960 с.: 263 ил.
2. Лаптев В.В. C++. Экспресс-курс. – СПб.: БХВ-Петербург, 2004. – 512 с.: ил..
3. Шлее М., Qt 5.10. Профессиональное программирование на C++ - СПБ.: БХВ-Петербург, 2018 — 1072 с.: ил. - (В подлиннике)

ПРИЛОЖЕНИЕ 2

База тестовых вопросов
(содержание файла Test.txt с выделенными жирным шрифтом вопросами и подчеркнутыми правильными ответами)

1) Как называют метод динамического программирования?

"Разделяй и властвуй"

"Дели и празднуй"

"Быстрее"

"Лучшее высшее"

2) Одно из главных условий умножения матриц?

Количество столбцов первого множителя равно количеству строк второго

Одинаковые по размеру

Разные по размеру

Единицы на главной диагонали

3) Как связаны между собой подзадачи при динамическом программировании?

Рекуррентно

Никак

Перпендикулярно

Родственно

4) В чем суть динамического программирования?

Решение задач путем разделения на подзадачи

Решение задач путем выделения динамической памяти

Оптимизация операции умножения

Оптимизация хранения больших объемов данных

5) Метод динамического программирования называют методом "Разделяй и ..."

Властвуй

6) Что не оптимизирует динамическое программирование?

Работу конвейера

Работу мозга

Числа Фибоначчи

Задачу о рюкзаке

7) … программирование - способ решения сложных задач путём разбиения их на более простые подзадачи

Динамическое

8) Какие матрицы будут перемножены (даны размеры матриц)?

2x2 и 4x4

3x4 и 3x4

2x3 и 3x4

2x2 и 4x2

9) Какая матрица (таблица) вычисляется путём оптимизации умножения матриц с помощью метода динамического программирования?

Матрица смежности

Обратная матрица

Матрица оптимальных стоимостей

Матрица конкуренции М. Портера

10) Время работы поиска минимального количества умножений (таблицы оптимальных стоимостей и таблицы расставления скобок)?

$O(n^3)$

$O(n^2)$

$O(n)$

$O(n^{10})$

ПРИЛОЖЕНИЕ 3

Диаграмма классов

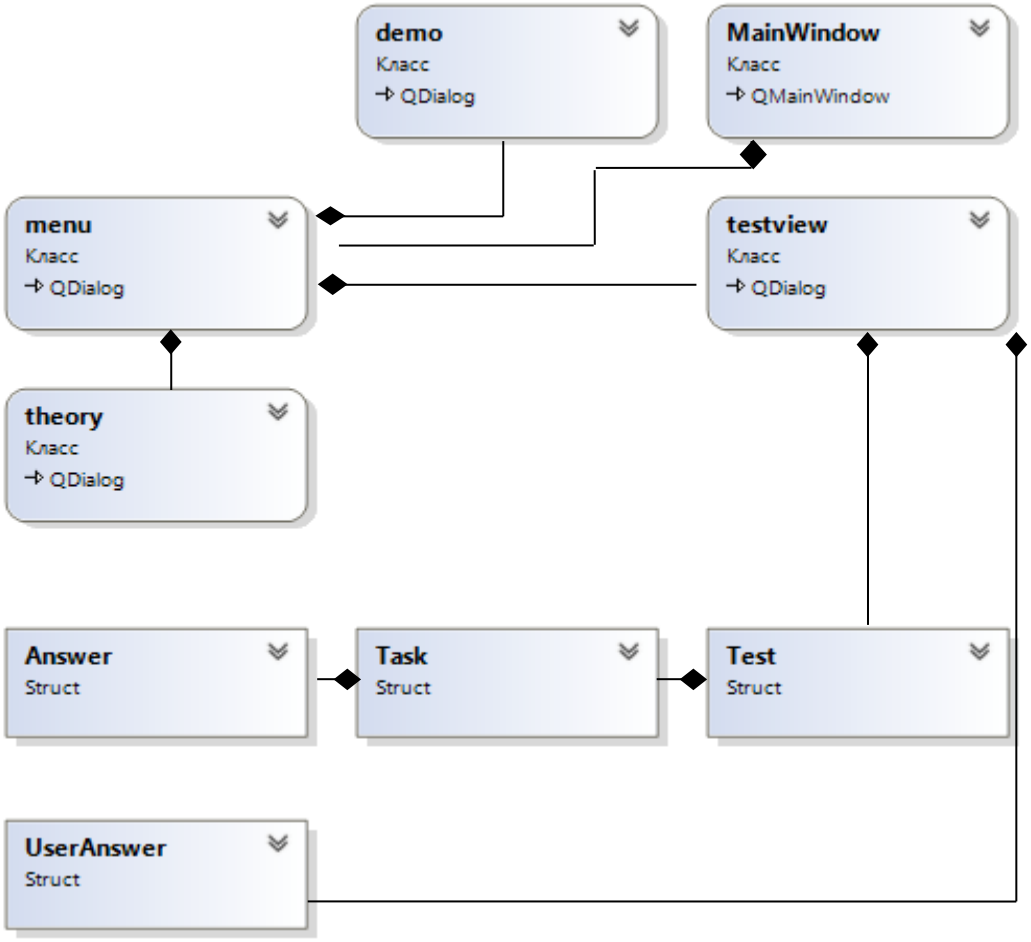

Рисунок П.3 – Диаграмма классов