|  |  |  |
| --- | --- | --- |
|  | ***Федеральное агентство по рыболовству***  ***Федеральное государственное бюджетное образовательное учреждение***  ***высшего образования***  ***«Астраханский государственный технический университет»***  **Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована DQS**  **по международному стандарту ISO 9001:2015** | |
| 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**  **«Решение систем линейных уравнений методом Гаусса»**  по дисциплине «Программирование и информатика» | | |
| Допущен к защите  «\_\_\_» \_\_\_\_\_\_\_\_202\_\_г.  Руководитель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Оценка, полученная на защите «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_» | | Проект выполнен  обучающимся группы ДИПРб-11 Лиджигоряевым В.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Руководитель  ст. преп. Толасова В.В. |
| Члены комиссии:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |  |

Астрахань – 2020

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ ПО РЫБОЛОВСТВУ**

**АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |  |
| --- | --- |
| **УТВЕРЖДАЮ** | Кафедра «Автоматизированные системы  обработки информации и управления» |
| Заведующий кафедрой  д.т.н., проф.  Т.В. Хоменко\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202 \_\_ г. |

**ЗАДАНИЕ**

**на выполнение Курсового проекта**

Обучающийся ***Лиджигоряев Владимир Алексеевич***

Группа ***ДИПРб-11***

Дисциплина ***Программирование и информатика***

Тема ***Учебно-демонстрационная программа «Решение систем линейных***

***уравнений методом Гаусса»***

Дата получения задания «21» сентября 2020 г.  
Срок представления обучающимся КП на кафедру «14»  декабря  2020 г.

Руководитель ***ст. преподаватель***\_\_\_\_\_\_\_\_\_ ***Толасова В.В.*** «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_202\_\_г.

должность, степень, званиеподписьФИО

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ***Лиджигоряев В.А.*** «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_202\_\_г.

подпись ФИО

**Задачи**

Разработка программного продукта, который

* предоставляет теоретический материал по теме “Решение систем линейных уравнений методом Гаусса”;
* демонстрирует работу метода Гаусса;
* тестирует пользователя по изученному материалу;
* позволяет редактировать теоретический материал, вопросы теста и настраивать тестирование.

**Список рекомендуемой литературы**

1. Булычева Ю.А., Васильева Т.В. Учебное пособие “Алгебра (часть 2)” для студентов специальности 090105.65 “Комплексное обеспечение информационной безопасности автоматизированных систем” – АГТУ, Астрахань 2019. – 88 с.
2. Прата С. Язык программирования С++. Лекции и упражнения, 6-е изд. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2012. – 1248 с. : ил.

|  |  |
| --- | --- |
| **УТВЕРЖДАЮ** | К заданию на курсовой проектпо дисциплине  «Программирование и информатика» |
| Заведующий кафедрой  д.т.н., проф.  Т.В. Хоменко\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202 \_\_ г. |

**КАЛЕНДАРНЫЙ ГРАФИК**

Курсового проектирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Разделы, темы и их содержание, графический материал | Дата сдачи | Объем, % |
| 1 | Выбор темы | 15.02.2021 | 1 |
| 2 | Техническое задание | 25.02.2021 | 3 |
| 3 | Разработка модели, проектирование системы   * *введение,* * *технический проект,* * *программа и методика испытаний, в том числе – данные для тестирования* * *литература* | 31.01.2021 | 25 |
| 4 | Программная реализация системы   * *работающая программа,* * *рабочий проект* * *скорректированное техническое задание (при необходимости)* | 30.04.2021 | 40 |
| 5 | Тестирование и отладка системы, эксперименты   * *работающая программа с внесёнными изменениями,* * *окончательные тексты всех разделов* | 14.05.2021 | 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* | 24.05.2021 | 59 |
| 7 | Защита творческого проекта | 21.12.2021-01.06.2021 | 60-100 |

С графиком ознакомлен «21» сентября 2020 г.

Лиджигоряев В.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, обучающийся группы ДИПРб-11

(фамилия, инициалы, подпись)

**График творческого проектирования выполнен**

без отклонений / с незначительными отклонениями / со значительными отклонениями

нужное подчеркнуть

Руководитель Курсового проекта\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель Толасова В.В.

подпись, ученая степень, звание, фамилия, инициалы

**СОДЕРЖАНИЕ**

[Введение](#__RefHeading___Toc531184546) 7

[1 Технический проект](#__RefHeading___Toc531184547) 8

[1.1 Анализ предметной области](#__RefHeading___Toc531184548) 8

[1.1.1 Система линейных уравнений](#__RefHeading___Toc531184549) 8

[1.1.2 Совместность системы, равносильность систем](#__RefHeading___Toc531184550) 8

[1.1.3 Элементарные преобразования системы линейных уравнений](#__RefHeading___Toc531184550) 9

[1.1.4 Миноры матрицы. Ранг матрицы](#__RefHeading___Toc531184550) 10

[1.1.5 Метод Гаусса. Критерий совместимости. Критерий определенности](#__RefHeading___Toc531184550) 10

[1.1.6 Тестирование и проверка знаний](#__RefHeading___Toc531184551) 12

[1.2 Технология обработки информации](#__RefHeading___Toc531184552) 13

[1.2.1 Форматы данных](#__RefHeading___Toc531184553) 14

[1.2.2 Алгоритм получения глав в тексте с теорией из файла с теорией](#__RefHeading___Toc531184554) 14

[1.2.3 Алгоритм вывода строки теории](#__RefHeading___Toc531184554) 15

[1.2.4 Алгоритм вывода главы теории](#__RefHeading___Toc531184554) 16

[1.2.5 Алгоритм вывода теоретического материала на экран](#__RefHeading___Toc531184554) 16

[1.2.6 Алгоритм заполнения матрицы случайными числами](#__RefHeading___Toc531184554) 17

[1.2.7 Алгоритм заполнения матрицы с клавиатуры](#__RefHeading___Toc531184554) 18

[1.2.8 Алгоритм заполнения матрицы из файла](#__RefHeading___Toc531184554) 19

[1.2.9 Алгоритм заполнения матрицы](#__RefHeading___Toc531184554) 20

[1.2.10 Алгоритм вывода строки матрицы на экран](#__RefHeading___Toc531184554) 21

[1.2.11 Алгоритм вывода матрицы на экран](#__RefHeading___Toc531184554) 22

[1.2.12 Алгоритм перемены двух строк местами в матрице](#__RefHeading___Toc531184554) 23

[1.2.13 Алгоритм умножения строки на некоторое число](#__RefHeading___Toc531184554) 23

[1.2.14 Алгоритм очищения главной диагонали](#__RefHeading___Toc531184554) 24

[1.2.15 Алгоритм прямого хода метода Гаусса](#__RefHeading___Toc531184554) 25

[1.2.16 Алгоритм проверки определенности системы](#__RefHeading___Toc531184554) 28

[1.2.17 Алгоритм метода Гаусса](#__RefHeading___Toc531184554) 28

[1.2.18 Алгоритм вывода решения системы линейных уравнений](#__RefHeading___Toc531184554) 31

[1.2.19 Алгоритм получения перемешанного массива строк](#__RefHeading___Toc531184553) 31

[1.2.20 Алгоритм получения тестовых вопросов из файла](#__RefHeading___Toc531184553) 32

[1.2.21 Алгоритм подготовки тестовых заданий к выводу](#__RefHeading___Toc531184553) 34

[1.2.22 Алгоритм получения случайных вопросов и ответов из теста](#__RefHeading___Toc531184553) 35

[1.2.23 Алгоритм получения вопросов и ответов для определенного уровня сложности.](#__RefHeading___Toc531184553) 36

[1.2.24 Алгоритм получения нужных вопросов и ответов](#__RefHeading___Toc531184553) 37

[1.2.25 Алгоритм тестирования](#__RefHeading___Toc531184553) 37

[1.2.26 Алгоритм добавления новой заголовочной строки](#__RefHeading___Toc531184553) 39

[1.2.27 Алгоритм добавления новой текстовой строки](#__RefHeading___Toc531184553) 40

[1.2.28 Алгоритм добавления новой главы](#__RefHeading___Toc531184553) 40

[1.2.29 Алгоритм выбора главы для изменения](#__RefHeading___Toc531184553) 41

[1.2.30 Алгоритм выбора строки главы для добавления на ее место новой строки](#__RefHeading___Toc531184553) 42

[1.2.31 Алгоритм выбора строк главы для их изменения](#__RefHeading___Toc531184553) 43

[1.2.32 Алгоритм получения типа строки](#__RefHeading___Toc531184553) 43

[1.2.33 Алгоритм удаления строки из главы](#__RefHeading___Toc531184553) 44

[1.2.34 Алгоритм изменения строк](#__RefHeading___Toc531184553) 45

[1.2.35 Алгоритм изменения главы](#__RefHeading___Toc531184553) 46

[1.2.36 Алгоритм изменения файла с теорией](#__RefHeading___Toc531184553) 47

[1.2.37 Алгоритм изменения теории](#__RefHeading___Toc531184553) 48

[1.2.38 Алгоритм создания переходов на новую строку](#__RefHeading___Toc531184554) 49

[1.2.39 Алгоритм разделения строки](#__RefHeading___Toc531184554) 49

[1.2.40 Алгоритм анализа деструкторов](#__RefHeading___Toc531184554) 50

[1.2.41 Алгоритм добавления тестового вопроса](#__RefHeading___Toc531184554) 50

[1.2.42 Алгоритм удаления тестового вопроса](#__RefHeading___Toc531184554) 51

[1.2.43 Алгоритм изменения тестового вопроса](#__RefHeading___Toc531184554) 51

[1.2.44 Алгоритм изменения файла с тестами](#__RefHeading___Toc531184554) 53

[1.2.45 Алгоритм вывода всех тестовых заданий из файла](#__RefHeading___Toc531184554) 54

[1.2.46 Алгоритм изменения тестирования](#__RefHeading___Toc531184554) 54

[1.3 Входные и выходные данные](#__RefHeading___Toc531184557) 57

[1.4 Системные требования](#__RefHeading___Toc531184558) 58

[2 Рабочий проект](#__RefHeading___Toc531184559) 59

[2.1 Общие сведения о работе системы](#__RefHeading___Toc531184560) 59

[2.2 Функциональное назначение программного продукта](#__RefHeading___Toc531184561) 59

[2.3 Инсталляция и выполнение программного продукта](#__RefHeading___Toc531184562) 59

[2.4 Описание программы](#__RefHeading___Toc531184563) 60

[2.5 Разработанные меню и интерфейсы](#__RefHeading___Toc531184564) 67

[2.6 Сообщения системы](#__RefHeading___Toc531184565) 78

[3 Программа и методика испытаний](#__RefHeading___Toc531184566) 79

[3.1 Проверка работоспособности показа теории](#__RefHeading___Toc531184567) 79

[3.2 Проверка работоспособности демонстрации работы метода Гаусса](#__RefHeading___Toc531184568) 79

[3.3 Проверка работоспособности тестирования](#__RefHeading___Toc531184569) 79

[3.4 Проверка работоспособности администрирования файла с теорией](#__RefHeading___Toc531184567) 80

[3.5 Проверка работоспособности администрирования файла с тестами](#__RefHeading___Toc531184567) 81

[Заключение](#__RefHeading___Toc531184570) 83

[Список использованных источников](#__RefHeading___Toc531184571) 84

[Придожение 1 Техническое задание](#__RefHeading___Toc531184566) 85

[Приложение 2 База вопросов](#__RefHeading___Toc531184572) 88

# ВВЕДЕНИЕ

Уметь решать системы линейных уравнений – очень важный навык любого человека. В школе изучаются линейные уравнения, а также решение систем с этими уравнениями. Решение многих задач разного спектра и из разных областей наук, будь то физика, химия или что-то другое, часто сводится к решению системы линейных уравнений.

Задачи экономического содержания зачастую приводят к решению систем линейных уравнений. Данные задачи могут быть оптимизационного характера, например, для оптимизации какого-нибудь бизнес-процесса или выбора лучшей тактики ведения бизнеса, расчета расходов. Даже тривиальные жизненные задачи, подобные расчету максимального количества разных товаров, которые человек может купить на определенную сумму, приводят к решению системы линейных уравнений. Существует несколько основных методов решения систем линейных уравнений и метод Гаусса является одним из них, это классический метод решения систем линейных уравнений, который помогает человеку решать системы любого вида. Поэтому очень важно уметь решать системы линейных уравнений с помощью метода Гаусса.

Целью создания учебно-демонстрационной программы «Решение систем линейных уравнений методом Гаусса» является автоматизация процесса обучения студентов навыку решения систем линейных уравнений методом Гаусса.

Назначение программы – повышение успеваемости и качества знаний студентов, снижение нагрузки на преподавателя.

# 1 ТЕХНИЧЕСКИЙ ПРОЕКТ

## 1.1 Анализ предметной области

### 1.1.1 Система линейных уравнений с n неизвестными над полем P

Системой линейных уравнений над полем P с n переменными называется система вида:

, (1)  
где

.

Элементы поля называют коэффициента при неизвестных, а элементы – свободными членами.

Примером такой системы может быть:

Решением системы уравнений (1) называется упорядоченный набор элементов из поля P, при подстановке которых в уравнения вместо соответственно неизвестных , все уравнения системы (1) превращаются в верные равенства между элементами поля P. [1].

Систему уравнений (1) можно также записать в следующем виде:

,

где

– основная матрица системы (1);

, – столбец неизвестных и столбце свободных членов системы (1) соответственно.

Данный вид записи системы называется матричным видом.

### 1.1.2 Совместность системы, равносильность систем.

Система уравнений над полем P называется совместной, если она имеет хотя бы одно решение. В противном случае – система несовместна.

Систему, имеющую единственное решение, называют определенной. Систему, имеющую более одного решения, называют неопределенной.

Две системы уравнений над полем P с одними и теми же неизвестными называются равносильными, если множества их решений совпадают.

Для нахождения решений системы обычно стремятся преобразовать ее к какой-либо более простой системе, равносильной исходной.

### 1.1.3 Элементарные преобразования системы линейных уравнений

Существуют следующие элементарные преобразования систем линейных уравнений:

1. перемена мест двух уравнений системы;
2. умножение любого уравнения на обратимый элемент поля P;
3. прибавление к любому уравнению системы другого уравнения, умноженного на произвольный элемент поля P.

Каждое элементарное преобразование системы приводит исходную систему к

равносильной ей системе.

Пример:

Дана система линейных уравнений:

Следующие преобразования данной системы не изменяют множества решений системы:

или

или

### 1.1.4 Миноры матрицы. Ранг матрицы

Определителем квадратной матрицы *А п*-ого порядка над кольцом *К* называется элемент кольца *К,* равный алгебраической сумме *п*! произведений, составленных из элементов, взятых в точности по одному из каждойстроки и каждого столбца матрицы *А*. В алгебраическойсумме произведение берется со знаком «+» или «-» в соответствии с тем, какой знак имеет подстановка вида:

,(2)   
 Обозначение:

Пример нахождения определителя:

Пусть n = 2, , матрица А задана над кольцом действительных чисел.

Тогда существует 2! всевозможных подстановок вида (2) для этой матрицы:

и

Первая подстановка является четной, поэтому имеет знак «+». Вторая подстановка – нечетная, поэтому имеет знак «-». Поэтому произведение элементов матрицы , соответствующее первой подстановке, берем со знаком «+», а произведение, составленное по второй подстановке - со знаком «-». Т.е.

Пусть дана матрица над полем P, где

Выберем в матрице А произвольные k строк и k столбцов (). Элементы, стоящие на пересечении этих строк и столбцов, составляют квадратную матрицу k-ого порядка. Определитель этой матрицы называется минором к-ого порядка матрицы А.

Обозначение: .

Пример нахождения минора:

Дан определитель матрицы А над :

Тогда минор элемента, стоящего в первой строке, первом столбце матрицы А будет следующим:

Ранг матрицы – наивысший порядок отличных от нуля минором ненулевой матрицы А.

Обозначение: r(A), rang(A).

### 1.1.5 Метод Гаусса. Критерий совместимости. Критерий определенности

Расширенная матрица – матрица, содержащая в себе основную матрицу и столбце свободных членов системы линейных уравнений.

Критерий совместимости: система линейных уравнений над полем совместна тогда и только тогда, когда ранг ее основной матрицы равен рангу расширенной матрицы.

Критерий определенности: Система линейных уравнений над полем имеет единственное решение тогда и только тогда, когда ранги основной и расширенной матриц системы равны числу ее неизвестных.

Метод Гаусса заключается в постепенном исключении неизвестных.

Пусть дана система m линейных уравнений с n неизвестными:

,

Если = 0, добиваемся с помощью перестановкой уравнений, чтобы этот коэффициент не был равен 0.

С помощью элементарных преобразований, исключая неизвестное из всех уравнений, кроме первого, систему (3) можно привести к системе (4), равносильной исходной:

,

Для этого к уравнения с номером m системы (3) прибавили первое уравнение, умноженное на элемент поля .

Аналогичным образом, считая , исключаем неизвестное из всех уравнений системы (4), начиная с третьего и т.д.

В ходе таких преобразований могут возникнуть следующие ситуации:

1. Появится уравнение вида:

,

где . В этом случае система (3) не имеет решений.

1. Получили систему из s линейных уравнений с n неизвестными вида:

, (5)  
где

.

Если s = n, то у системы (5) треугольный вид. Выражаем из последнего уравнения. Подставляя его в предпоследнее уравнение мы найдем однозначно определенное значение для неизвестного . Продолжая так далее, мы найдем, что система (5), а поэтому и система (3), обладают единственным решением, т.е. совместны и определены. Описанный алгоритм последовательного нахождения значений всех переменных представляет собой обратный ход метода Гаусса.

Если s < n, то система (5) имеет вид «трапеции». В этом случае часть неизвестных становятся главными, остальные неизвестные – свободными. Пусть *, ,…,* – главные неизвестные. Остальные члены - *, ,…,* в каждом уравнении системы (5) перенесем в другую часть. Таким образом, неизвестные *, ,…,* становятся свободными (они принимают любые значения поля P). В результате получим систему треугольного вида и из нее сначала найдем , потом и так до включительно.

Пример решения системы линейных уравнений методом Гаусса:

Дана следующая система линейных уравнений:

,

Прибавим ко второму уравнений первое уравнение, умноженное на 3:

Из последнего уравнения выражаем переменную y:

Выражаем переменную x из первого уравнения, подставляя найденное значение переменной y:

Таким образом, система имеет единственное решение: x = 0, y = 0.5

### 1.1.5 Тестирование и проверка знаний

Вопросы, задаваемые пользователю, выбираются из базы вопросов. Количество задаваемых вопросов и сами вопросы зависят от уровня сложности тестирования.

Всего уровней сложности тестирования 4. Каждый вопрос в базе вопросов имеет свой уровень сложности, определяемый от 1 до 3. На первом уровне сложности пользователю выводятся только задания первого уровня сложности, на втором – задания 2 уровня сложности, на третьем – 3 уровня сложности. 4 уровень сложности – случайный подбор вопросов из базы вопросов, вне зависимости от их уровня сложности.

Ответом пользователя на вопрос является только один из вариантов ответов, предложенные пользователю. Пользователь в качестве ответа вводит номер одного из вариантов ответов, который ему кажется правильным.

Если пользователь дает правильный ответ на задание, то выводится сообщение об этом, и серия правильных ответов увеличивается на 1, если же дает неправильный ответ, то выводится сообщение о том, что ответ неправильный, и серия неправильных ответов увеличивается на 1. Пользователь может отказаться от ответа на вопрос, введя 0, как ответ. В этом случае серия правильных ответов не изменится, а серия неправильных ответов увеличится и будет произведен переход к следующему вопросу.

После каждого вопроса пользователю показывается текущий результат по тестированию. По окончании тестирования пользователю показывается итоговый результат по тестированию.

## 1.2 Технология обработки информации

Анализ предметной области показал, что программа рассчитана на одного пользователя. На рис. 1.1. показана диаграмма вариантов использования.

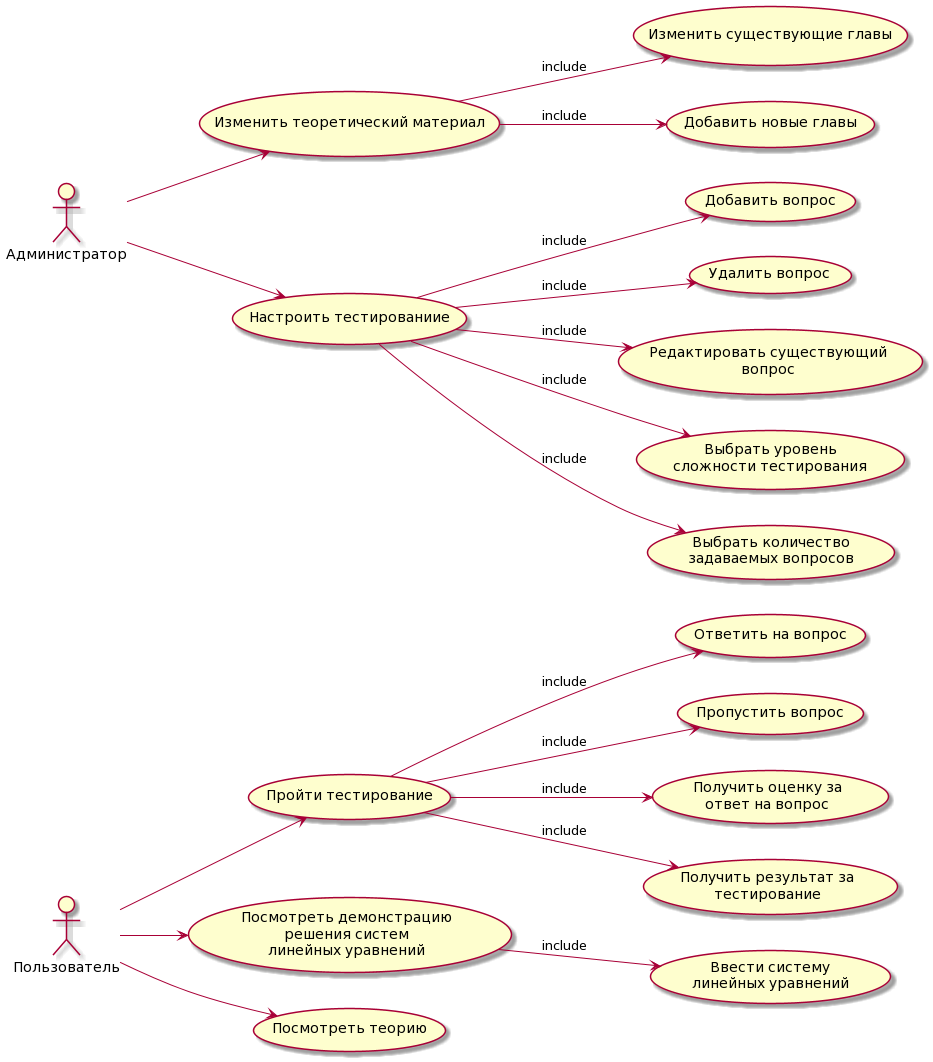


Рисунок 1.1 – Диаграмма вариантов использования

<include>

<include>

Получить результат тестирования

Вариант использования «Пройти тестирование» включает одну из обязательных функций («Ответить на вопрос», «Пропустить вопрос», «Получить оценку ответа на вопрос в тесте», «Получить результат тестирования»).

Вариант использования «Посмотреть демонстрацию решения системы линейных уравнений методом Гаусса» включает обязательную функцию «Ввести систему линейных уравнений».

Вариант использования «Изменить теоретический материал» включает 2 функции («Изменить существующие главы», «Добавить новые главы).

Вариант использования «Настроить тестирование» включает одну из обязательных функций («Добавить вопрос», «Удалить вопрос», «Редактировать существующий вопрос», «Выбрать уровень сложности тестирования», «Выбрать количество задаваемых вопросов»).

### 1.2.1 Форматы данных

Теоретический материал будет находиться в текстовом файле.

Материал разбит на главы, каждая глава заканчивается символом «#». Заголовки и определения до и после отделяются от остального текста символом «&».

База вопросов для тестирования будет храниться в текстовом файле.

Файл имеет следующий формат:

В первой строке записываются через пробел количество вопросов, задаваемых при тестировании на 4 уровне сложности (случайно подбираемые вопросы), и текущий уровень сложности тестирования.

На нижних строках записываются вопросы: сначала записывается вопрос на необходимо количество строк, вопрос обязательно должен оканчиваться символом «#», строкой ниже следуют дистракторы, они занимают одну строку и разделяются друг от друга знаком «&», строкой ниже следует правильный ответ на вопрос, он занимает одну строку, под правильным ответом следует уровень сложности вопроса, который также занимает одну строку.

### 1.2.2 Алгоритм получения глав в тексте с теорией из файла с теорией

1. В программе страницы теоретического материала хранятся в двумерном массиве ***chapters*** строкового типа на ***N*** элементов, где ***N*** – целочисленная переменная, хранящая количество глав в файле. Элементами массива ***chapters*** являются массивы строк, которые хранят в себе строки соответствующей главы, их количество элементов определяется количеством строк в соответствующей главе.
2. Объявить двумерный массив строкового типа chapters.
3. Объявить строку row.
4. Присвоить целочисленной переменной index 0.
5. Присвоить целочисленной переменной rowIndex 0.
6. Пока ***не конец файла***, повторять

| Изменить размер массива chapters на index+1.

| Присвоить символьной переменной lastSymbolOfRow последний символ строки

| row.

| Изменить размер массива chapters[index] на rowsIndex + 1.

| Если lastSymbolOfRow = ‘#’, то

| | Удалить последний символ строки row.

| | Прибавить к chapters[index][rowsIndex] row.

| | Прибавить к chapters[index][rowsIndex] ‘\n’.

| | Увеличить index на единицу.

| | Обнулить rowsIndex.

| иначе

| | Прибавить к chapters[index][rowsIndex] row.

| | Прибавить к chapters[index][rowsIndex] ‘\n’.

| | Увеличить rowsIndex на единицу.

| конец ветвления

конец цикла

1. Вернуть pages.
2. Конец алгоритма.

### 1.2.3 Алгоритм вывода строки теории

row – константная строка

1. Присвоить целочисленной переменной s значение 0.
2. Пока s < rowSize, повторять

| Присвоить ch pages[i][j][s].

| Если ch не равно ‘&’, то

| | Вывести ch.

| иначе

| | Присвоить termin противоположное значение termin.

| | Если termin, то

| | | Установить красный цвет шрифта для вывода на консоль.

| | иначе

| | | Установить черный цвет шрифта для вывода на консоль.

| | конец ветвления

| конец ветвления

| Увеличить s на единицу.

конец цикла

1. Конец алгоритма.

### 1.2.4 Алгоритм вывода главы теории

chapters – константный двумерный массив строкового типа

chap – константная целочисленная переменная

1. Вывести “Глава ”.
2. Вывести i+1.
3. Вывести “: ” и перейти на новую строку.
4. Присвоить целочисленной переменной rowsCountOfPage количество элементов
5. pages[i].
6. Присвоить целочисленной переменной j значение 0.
7. Пока j < rowsCountOfPage, повторять

| Присвоить целочисленной переменной rowsLeft (rowsCountOfPage – j).

| Объявить целочисленной переменную outRowsCount.

| Если rowsLeft >= 20, то

| | Присвоить outRowsCount 20.

| иначе

| | Присвоить outRowsCount rowsLeft.

| конец ветвления

| Присвоить целочисленной переменной k значение j.

| Пока (k - j) < outRowsCount, повторять

| | Вызвать вспомогательный алгоритм вывода строки теории 1.2.4

| | Увеличить k на единицу.

| конец цикла

| Увеличить j на outRowsCount.

конец цикла

1. Конец алгоритма.

### 1.2.5 Алгоритм вывода теоретического материала на экран

chapters – константный двумерный массив строкового типа

1. Записать в целочисленную переменную chapsCount количество элементов chapters.
2. Присвоить целочисленной переменной i значение 0.
3. Пока ***i < chapsCount***, повторять

| Вызвать вспомогательный алгоритм вывода главы теории 1.2.4.

| Присвоить булевой переменной changed значение false.

| Объявить целочисленную переменную temp.

| Пока changed = false, повторять

| | Повторять, пока temp не равно коду символа стрелки влево или вправо

| | | Вывести “Нажмите стрелку влево, чтобы вернуться на одну страницу назад,

| | | стрелку вправо – чтобы пролистнуть на одну страницу вперед: ”.

| | | Ввести temp.

| | конец цикла

| | Если temp = код символа стрелки влево, то

| | | Если i = 0, то

| | | | Вывести “Нельзя пролистнуть назад, так как это первая страница”.

| | | иначе

| | | | Уменьшить i на единицу.

| | | | Присвоить переменной changed значение true.

| | | конец ветвления

| | иначе

| | | Увеличить i на единицу.

| | | Присвоить переменной changed значение true.

| | конец ветвления

| конец цикла

| Увеличить i на единицу.

конец цикла

1. Вывести “Показ теории завершен.”.
2. Конец алгоритма.

### 1.2.6 Алгоритм заполнения матрицы случайными числами

Дано:

v – ссылка на двумерный массив вещественного типа

rows – ссылка на целочисленную переменную

cols – ссылка на целочисленную переменную

min - константная переменная целочисленного типа

max - константная переменная целочисленного типа

1. Изменить количество элементов массива v на rows.
2. Присвоить целочисленной переменной i значение 0.
3. Пока i < rows, повторять

| Изменить количество элементов массива v[i] на cols.

| Увеличить i на единицу.

конец цикла.

1. Присвоить целочисленной переменной diapason (max – min + 1).
2. Присвоить целочисленной переменной i значение 0.
3. Пока i < rows, повторять

| Присвоить целочисленной переменной j значение 0.

| Пока j < cols, повторять

| | Присвоить v[i][j] (min + остаток от деления сгенерированного случайного числа

| | на diapason).

| | Увеличить j на единицу.

| конец цикла.

| Увеличить i на единицу.

конец цикла.

1. Конец алгоритма.

### 1.2.7 Алгоритм заполнения матрицы с клавиатуры

v – ссылка на двумерный массив вещественного типа

rows – ссылка на целочисленную переменную

cols – ссылка на целочисленную переменную

1. Изменить размер массива v на rows.
2. Присвоить целочисленной переменной i значение 0.
3. Пока i < rows, повторять

| Изменить количество элементов массива v[i] на cols.

| Увеличить i на единицу.

конец цикла.

1. Присвоить целочисленной переменной i значение 0.
2. Пока i < rows, повторять

| Присвоить целочисленной переменной j значение 0.

| Пока j < cols, повторять

| | Вывести “Введите элемент ”.

| | Вывести i+1.

| | Вывести “ строки ”.

| | Вывести j+1.

| | Вывести “ столбца матрицы: ”.

| | Ввести v[i][j].

| | Увеличить j на единицу.

| конец цикла.

| Увеличить i на единицу.

конец цикла.

1. Конец алгоритма.

### 1.2.8 Алгоритм заполнения матрицы из файла

v – ссылка на двумерный массив вещественного типа

path – константная переменная строкового типа

rows – ссылка на целочисленную переменную

cols – ссылка на целочисленную переменную

1. Если файл с названием path открылся на чтение, то

| Присвоить целочисленной переменной count 0.

| Объявить вещественную переменную temp.

| Пока не конец файла с именем path, повторять

| | Считать число в переменную temp из файла.

| | Увеличить count на единицу.

| конец цикла

| Если count = 0, то

| | Вернуть ложь.

| иначе

| | Установить позицию чтения файла с названием path в начало файла.

| | Присвоить целочисленной переменной space\_count 0.

| | Присвоить символьной переменной symbol ‘ ’.

| | Пока не конец файла с названием path и symbols не равно ‘\n’, повторять

| | | Считать символ из файла в переменную symbol.

| | | Если symbol = ‘ ’, то

| | | | Увеличить space\_count на единицу.

| | | конец ветвления

| | конец цикла

| | Установить позицию чтения файла с названием path в начало файла.

| | Присвоить целочисленной переменной row значение count / (space\_count+1).

| | Присвоить целочисленной переменной cols значение space\_count+1.

| | Если rows < 2 или rows > 6 или rows не равно (cols-1), то

| | | Вернуть ложь.

| | иначе

| | | Изменить размер массива v на rows.

| | | Присвоить целочисленной переменной i 0.

| | | Пока i < rows, повторять

| | | | Изменить размер v[i] на cols.

| | | | Увеличить i на единицу.

| | | конец цикла

| | | Присвоить целочисленной переменной i 0.

| | | Пока i < rows, повторять

| | | | Присвоить целочисленной переменной j 0.

| | | | Пока j < cols, повторять

| | | | | Считать число из файла в v[i][j].

| | | | | Увеличить j на единицу.

| | | | конец цикла

| | | | Увеличить i на единицу.

| | | конец цикла

| | | Вернуть истину.

| | конец ветвления

| конец ветвления

конец ветвления

1. Вернуть ложь.
2. Конец алгоритма.

### 1.2.9 Алгоритм заполнения матрицы

Дано:

v – ссылка на двумерный массив вещественного типа

method – константная переменная целочисленного типа

rows – ссылка на целочисленную переменную

cols – ссылка на целочисленную переменную

1. Если method = 1 или method = 2, то

| Повторять, пока rows < 2 или rows > 6

| | Вывести “Введите количество уравнений (2-6): ”.

| | Ввести rows.

| конец цикла

| Присвоить cols значение rows+1.

| Если method = 1, то

| | Вызвать вспомогательный алгоритм заполнения матрицы с клавиатуры 1.2.7

| | с v, rows и cols, как параметрами.

| иначе

| | Объявить целочисленные переменные min и max.

| | Вывести “Введите нижнюю границу генерации случайных чисел (целое число):”.

| | Ввести min.

| | Повторять, пока max < min

| | | Вывести “Введите верхнюю границу генерации случайных чисел, ”.

| | | Вывести “должна быть не ниже нижней границы (целое число): ”.

| | | Ввести max.

| | конец цикла

| | Вызвать вспомогательный алгоритм заполнения матрицы случайными

| | числами 1.2.6 с v, rows, cols, min и max, как параметрами.

| конец ветвления

иначе

| Объявить строковую переменную path.

| Вывести “Введите название файла с расширением: ”.

| Ввести path.

| Если вспомогательный алгоритм заполнения матрицы из файла 1.2.8

| с v и path, как параметрами, возвращает ложь, то

| | Вернуть ложь.

| конец ветвления

конец ветвления

1. Вернуть истину.
2. Конец алгоритма.

### 1.2.10 Алгоритм вывода строки матрицы на экран

Дано:

mat – константный двумерный массива вещественного типа

row – константная целочисленная переменная

cols – константная целочисленная переменная

1. Присвоить целочисленной переменной j значение 0.
2. Пока j < cols-1, повторять

| Если абсолютное значение mat[row][j] < , то

| | Вывести 0 и “\*x\_”.

| | Вывести j.

| иначе если j > 0 и mat[row][j] < 0, то

| | Вывести –mat[row][j] и “\*x\_”.

| | Вывести j.

| иначе

| | Вывести mat[row][j] и “\*x\_”.

| | Вывести j.

| конец ветвления

| Если j < (cols-2) и mat[row][j+1] >= 0, то

| | Вывести “+”.

| иначе если j < (cols-2) и mat[row][j+1] < 0

| | Вывести “-”.

| конец ветвления

| Увеличить j на единицу.

конец цикла

1. Вывести “ = ”.
2. Если абсолютное значение mat[row][cols-1] < , то

| Вывести 0 и перейти на новую строку.

иначе

| Вывести mat[row][cols-1] и перейти на новую строку.

конец ветвления

1. Конец алгоритма.

### 1.2.11 Алгоритм вывода матрицы на экран

Дано:

mat – константный двумерный массива вещественного типа

rows – константная целочисленная переменная

cols – константная целочисленная переменная

1. Присвоить целочисленной переменной i значение 0.
2. Пока i < rows, повторять

| Вызвать вспомогательный алгоритм вывода строки матрицы на экран

| с mat, i и cols, как параметрами.

| Увеличить i на единицу.

конец цикла

1. Перейти на новую строку.
2. Конец алгоритма.

### 1.2.12 Алгоритм перемены двух строк местами в матрице

Дано:

mat – двумерный массива вещественного типа

cols – константная целочисленная переменная

i – константная целочисленная переменная

j – константная целочисленная переменная

1. Присвоить целочисленной переменной k значение 0.
2. Пока k < cols, повторять

| Присвоить вещественной переменной temp mat[i][k].

| Присвоить mat[i][k] значение mat[j][k].

| Присвоить mat[j][k] значение temp.

| Увеличить k на единицу.

конец цикла

1. Конец алгоритма.

### 1.2.13 Алгоритм умножения строки на некоторое число

Дано:

mat – двумерный массива вещественного типа

row – константная целочисленная переменная

num – константная целочисленная переменная

cols – константная целочисленная переменная

1. Присвоить целочисленной переменной j значение 0.
2. Пока j < cols, повторять

| Домножить mat[row][j] на num.

| Увеличить j на единицу.

конец цикла

1. Конец алгоритма.

### 1.2.14 Алгоритм очищения главной диагонали

Дано:

mat – двумерный массива вещественного типа

rows – константная целочисленная переменная

cols – константная целочисленная переменная

1. Присвоить целочисленной переменной go\_on значение 1.
2. Присвоить целочисленной переменной i значение 0.
3. Пока i < rows, повторять

| Вывести “Перед очищением ”.

| Вывести i+1 и “ строки: ”.

| Перейти новую строку.

| Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat, rows,

| cols, как параметрами.

| Присвоить вещественной переменной cleaner 1/mat[i][i].

| Вывести “Находим коэффициент очищения: ” и перейти на новую строку.

| Вывести “Коэффициент = 1 / элемент”.

| Вывести i+1 и “ строки”.

| Вывести i+1 и “ столбца = ”.

| Вывести cleaner и перейти на новую строку.

| Вывести “Умножаем ” и i+1.

| Вывести “ строку на этот коэффициент: ” и перейти на новую строку.

| Вызвать вспомогательный алгоритм умножения строки матрицы на некоторое

| число 1.2.13 с mat, i, cleaner, cols, как параметрами.

| Вывести “После очищения ” и i+1.

| Вывести “ строки: ” и перейти на новую строку.

| Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat, rows,

| cols, как параметрами.

| Остановить выполнение программы.

| Присвоить go\_on значение 1.

| Очистить экран консоли.

| Увеличить i на единицу.

конец цикла

1. Конец алгоритма.

### 1.2.15 Алгоритм прямого хода метода Гаусса

Дано:

mat – двумерный массива вещественного типа

rows – константная целочисленная переменная

cols – константная целочисленная переменная

Для демонстрации работы метода Гаусса используется двумерный массив вещественного типа ***matr***, в нем хранятся коэффициенты при неизвестных и свободные коэффициенты системы линейных уравнений. Число строк массива ***matr*** – натуральное число в диапазоне [2; 6], оно обозначает число уравнений в системе. Число столбцов массива ***matr*** на единицу больше числа строк этого массива. Коэффициенты при неизвестных и свободные коэффициенты системы линейных уравнений – целые числа в диапазоне [-50; 50]. В одномерном динамическом массиве ***x*** на количество элементов, равное количество строк в массиве ***matr***, хранится решение системы линейных уравнений. Каждый элемент массива ***x*** – соответствующая переменная в системе линейных уравнений, индекс элемента массива ***x*** соответствует индексу неизвестной.

1. Вывести “Выполнение прямого хода метода Гаусса:” и перейти на новую строку.
2. Вывести “Система линейных уравнений: ” и перейти на новую строку.
3. Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat, rows,

cols, как параметрами.

1. Присвоить целочисленной переменной ***k*** значение 0.
2. Пока k < rows, повторять

| Присвоить целочисленной переменной go\_on значение 1.

| Вывести “Находим максимальный элемент в” и k+1.

| Вывести “ столбце среди элементов, находящихся не выше главной диагонали: ”.

| Перейти на новую строку.

| Присвоить целочисленной переменной i\_max значение k.

| Присвоить вещественной переменной v\_max значение mat[i\_max][k].

| Присвоить целочисленной переменной i значение k+1.

| Пока i < rows

| | Если абсолютное значение mat[i][k] > v\_max, то

| | | Присвоить v\_max mat[i][k].

| | | Присвоить i\_max i.

| | конец ветвления

| | Увеличить i на единицу.

| конец цикла

| Если абсолютное значение mat[k][i\_max] меньше , то

| | Вернуть ложь.

| конец ветвления

| Вывести “Максимальный элемент = ” и v\_max.

| Вывести “, он расположен в ” и i\_max+1.

| Вывести “ строке.” и перейти на новую строку.

| Если i\_max не равно k, то

| | Вывести “Меняем ”.

| | Вывести k+1.

| | Вывести “ и ”.

| | Вывести i\_max +1.

| | Вывести “ строки местами.” и перейти на новую строку.

| | Вывести “Перед изменениями: ” и перейти на новую строку.

| | Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat,

| | rows, cols, как параметрами.

| | Вызвать вспомогательный алгоритм перемены двух строк матрицы местами

| | в матрице 1.2.12 с mat, cols, k, i\_max, как параметрами.

| | Вывести “После изменений: ” и перейти на новую строку.

| | Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat,

| | rows, cols, как параметрами.

| | Пока go\_on не равно 0, повторять

| | | Вывести “Введите 0, чтобы продолжить”.

| | | Ввести go\_on.

| | конец цикла

| | Очистить экран консоли.

| конец ветвления

| Присвоить go\_on значение 1.

| Пока i < rows, повторять

| | Вывести “Находим коэффициент для обнуления элемента в ”.

| | Вывести i+1.

| | Вывести “ строке ”.

| | Вывести k+1.

| | Вывести “ столбце” и перейти на новую строку.

| | Присвоить вещественной переменной f значение (mat[i][k] / mat[k][k].

| | Вывести “коэффициент = элемент ”.

| | Вывести i+1.

| | Вывести “ строки ”.

| | Вывести k+1.

| | Вывести “ столбца / элемент”.

| | Вывести k+1.

| | Вывести “ строки ”.

| | Вывести k+1.

| | Вывести “ столбца = ”.

| | Вывести f и перейти на новую строку.

| | Вывести “Из ”.

| | Вывести i+1.

| | Вывести “ строки вычитаем ”.

| | Вывести k+1.

| | Вывести “ строку, умноженную на коэффициент, равный ”.

| | Вывести f и перейти на новую строку.

| | Вывести “Перед изменениями в ”.

| | Вывести i+1.

| | Вывести “ строке: ” и перейти на новую строку.

| | Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat,

| | rows, cols, как параметрами.

| | Присвоить целочисленной переменной j значение (k+1).

| | Пока j < k+1, повторять

| | | Вычесть из mat[i][j] mat[k][j] \* f.

| | | Увеличить j на единицу.

| | конец цикла

| | Обнулить mat[i][k].

| | Вывести “После изменений в ”.

| | Вывести i+1.

| | Вывести “ строке: ” и перейти на новую строку.

| | Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat,

| | rows, cols, как параметрами.

| | Пока go\_on не равно 0, повторять

| | | Вывести “Введите 0, чтобы продолжить: ”.

| | | Ввести go\_on.

| | конец цикла

| | Очистить экран консоли.

| | Присвоить go\_on 1.

| | Увеличить i на единицу.

| конец цикла

| Увеличить k на единицу.

конец цикла

1. Вернуть истину.
2. Конец алгоритма.

### 1.2.16 Алгоритм проверки определенности системы

Дано:

mat – двумерный массива вещественного типа

rows – константная целочисленная переменная

cols – константная целочисленная переменная

1. Если вспомогательный алгоритм прямого хода метода Гаусса 1.2.15 с mat, rows, и cols, как параметрами, возвращает ложь, то

| Вернуть ложь.

конец ветвления

1. Вернуть истину.
2. Конец алгоритма.

### 1.2.17 Алгоритм метода Гаусса

Дано:

mat – двумерный массива вещественного типа

rows – константная целочисленная переменная

cols – константная целочисленная переменная

countOfX – ссылка на целочисленную переменную

1. Присвоить целочисленной переменной countOfX значение rows.
2. Объявить вещественный одномерный массив x.
3. Присвоить целочисленной переменной go\_on значение 1.
4. Изменить размер массива x на countOfX.
5. Вывести “Перед очищением главной диагонали: ” и перейти на новую строку.
6. Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat, rows,

cols, как параметрами.

1. Вызвать вспомогательный алгоритм очищения главной диагонали 1.2.14 с mat, rows,

cols, как параметрами.

1. Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat, rows,

cols, как параметрами.

1. Вывести “Прямой ход метода Гаусса завершен.” и перейти на новую строку.
2. Пока go\_on не равен 0, повторять

| Вывести “Введите 0, чтобы продолжить” и перейти на новую строку.

| Ввести go\_on.

конец цикла

1. Очистить экран консоли.
2. Присвоить целочисленной переменной go\_on значение 1.
3. Вывести “Находим значения переменных:” и перейти на новую строку.
4. Вывести “Система:” и перейти на новую строку.
5. Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat, rows,

cols, как параметрами.

1. Вывести “Значение х”.
2. Вывести countOfX -1.
3. Вывести “ находим из ”.
4. Вывести countOfX .
5. Вывести “ строки ”.
6. Вывести countOfX .
7. Вывести “ строка: ” и перейти на новую строку.
8. Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat,

countOfX-1, cols, как параметрами.

1. Присвоить x[countOfX – 1] значение mat[countOfX-1][countOfX].
2. Вывести “x”.
3. Вывести countOfX-1 .
4. Вывести “ равно: ”.
5. Вывести x[countOfX-1] и перейти на новую строку.
6. Пока go\_on не равно 0, повторять

| Вывести “Введите 0, чтобы продолжить” и перейти на новую строку.

| Ввести go\_on.

конец цикла

1. Присвоить go\_on значение 1.
2. Очистить экран консоли.
3. Присвоить целочисленной переменной i значение countOfX – 2.
4. Пока i >= 0, повторять

| Вывести “Система: ” и перейти на новую строку.

| Вызвать вспомогательный алгоритм вывода матрицы на экран 1.2.11 с mat, rows,

| cols, как параметрами.

| Вывести “Значение х”.

| Вывести i.

| Вывести “ находим из ”.

| Вывести i+1.

| Вывести “ строки: ” и перейти на новую строку.

| Вывести i+1 .

| Вывести “ строка: ” и перейти на новую строку.

| Вызвать вспомогательный алгоритм вывода строки матрицы на экран 1.2.10

| с mat, i, cols, как параметрами.

| Вывести “ При этом зная, что: ” и перейти на новую строку.

| Присвоить целочисленной переменной j значение countOfX-1.

| Пока j > 1, повторять

| | Вывести “ х”.

| | Вывести j.

| | Вывести “ = ”.

| | Вывести x[j].

| | Вывести “, ”.

| | Уменьшить j на единицу.

| конец цикла

| Вывести “х”.

| Вывести i+1.

| Вывести “ = ”.

| Вывести x[i+1] и перейти на новую строку.

| Присвоить x[i] значение mat[i][countOfX].

| Присвоить целочисленной переменной j значение i+1.

| Пока j < countOfX, повторять

| | Уменьшить x[i] на mat[i][j] \* x[j].

| | Увеличить j на единицу.

| конец цикла

| Вывести “Получаем” и перейти на новую строку.

| Вывести “х”.

| Вывести i.

| Вывести “ = ”.

| Вывести x[i] и перейти на новую строку.

| Остановить выполнение программы и ожидать нажатия пользователя.

| Очистить экран консоли.

| Уменьшить i на единицу.

конец цикла

1. Вернуть x.
2. Конец алгоритма.

### 1.2.18 Алгоритм вывода решения системы линейных уравнений

Дано:

x – одномерный массив вещественного типа

countOfX – константная целочисленная переменная

1. Присвоить целочисленной переменной i значение 0.
2. Пока i < countOfX, повторять

| Вывести “x\_”.

| Вывести i.

| Вывести “ = ”.

| Вывести x[i] и перейти на новую строку.

конец цикла

1. Конец алгоритма.

### 1.2.19 Алгоритм получения перемешанного массива строк

Дано:

A – константный одномерный массив строкового типа

Asize – константная целочисленная переменная

1. Объявить одномерный целочисленный массив mentioned.
2. Объявить одномерный строкового массив mixed.
3. Присвоить целочисленной переменной i значение 0.
4. Пока i < Asize, повторять

| Объявить целочисленную переменную generatedIndex.

| Повторять, пока в массиве mentioned есть элемент, равный i

| | Присвоить generatedIndex остаток от деления сгенерированного целого числа

| | на Asize.

| конец цикла

| Присвоить mixed[i] A[generatedIndex].

| Присвоить mentioned[i] generatedIndex.

| Увеличить i на единицу.

конец цикла

1. Вернуть mixed.
2. Конец алгоритма.

### 1.2.20 Алгоритм получения тестовых вопросов из файла

Дано:

testPath - константная строковая переменная

QA –ссылка на двумерный массив строкового типа

correctAns –ссылка одномерный массив строкового типа

countOfAllQ – ссылка на целочисленную переменную

countOfAskingQ – ссылка на целочисленную переменную

lvlSelection – ссылка на строковую переменную

В двумерном массиве строк ***allQA*** хранятся все вопросы, варианты ответов и уровни сложности из базы вопросов. В первой строке массива ***allQA*** хранятся вопросы, во второй строке – варианты ответов, в третьей строке – уровни сложности. Строки вариантов ответов и уровня сложности для определенного вопроса хранятся в соответствующих строках, но в том же столбце, что и строка вопроса. Количество столбцов массива ***allQA*** определяется количеством вопросов в базе вопросов.

В одномерном массиве ***allCorrectAns*** строкового типа на 10 элементов хранятся правильные ответы на вопросы, каждый элемент массива ***allCorrectAns*** соответствует определенному стобцу массива ***allQA***, номер столбца массива ***allQA*** – номер определенного вопроса, вариантов ответов к нему и его уровня сложности, а элемент массива ***allCorrectAns,*** определенный номером столбца массива ***allCorrectAns*** – правильный ответ для соответствующего вопроса.

В двумерном массиве строк ***QAOutVer*** хранятся вопросы, варианты ответов и уровни сложности к ним из базы вопросов для выдачи пользователю. В первой строке массива ***QAOutVer*** хранятся вопросы, во второй строке – варианты ответов, в третьей – уровни сложности. Строки вариантов ответов и уровня сложности для определенного вопроса хранится в том же столбце, что и строка вопроса. Количество столбцов массива ***QAOutVer*** составляет зависит от выбранного уровня сложности.

В одномерном массиве ***correctAnsOutVer*** хранятся правильные ответы на вопросы, которые будут предлагаться пользователю. Каждый элемент массива ***correctAnsOutVer*** соответствует определенному стобцу массива ***QAOutVer***, номер столбца массива ***QAOutVer*** – номер определенного вопроса, а элемент массива ***correctAnsOutVer,*** определенный номером столбца массива ***QAOutVer*** – правильный ответ для соответствующего вопроса.

1. Изменить размер массива QA на 3.
2. Объявить строку counterNLevel.
3. Считать строку из файла в counterNLevel.
4. Считать до первого пробела из counterNLevel в countOfAskingQ.
5. Считать оставшееся из counterNLevel в lvlSelection.
6. Объявить строку row.
7. Объявить строку variants.
8. Объявить строку corAns.
9. Присвоить целочисленной переменной index значение 0.
10. Пока не конец файла, повторять

| Считать строку из файла в row.

| Изменить размер массива QA[0] на index+1.

| Изменить размер массива QA[1] на index+1.

| Изменить размер массива QA[2] на index+1.

| Присвоить строковой переменной lastSymbolOfRow последний символ строки

| row.

| Если lastSymbolOfRow = ‘#’, то

| | Считать строку из файла в variants.

| | Считать строку из файла в corAns.

| | Занести в массив correctAns corAns на последнюю позицию.

| | Прибавить к QA[1][index] variants.

| | Прибавить к QA[0][index] row.

| | Считать строку из файла в currentLvl.

| | Присвоить QA[2][index] currentLvl.

| | Увеличить index на единицу.

| иначе

| | Прибавить к QA[0][index] row.

| | Прибавить к QA[1][index] ‘\n’.

| конец ветвления

конец цикла

1. Конец алгоритма.

### 1.2.21 Алгоритм подготовки тестовых заданий к выводу

Дано:

testPath - константная строковая переменная

QA –ссылка на двумерный массив строкового типа

correctAns –ссылка одномерный массив строкового типа

countOfAllQ - константная целочисленная переменная

1. Присвоить целочисленной переменной i значение 0.
2. Пока i < countOfAllQ, повторять

| Объявить одномерный массив строк variantsTemp.

| Присвоить строке variants QA[1][i].

| Присвоить строке delim “&”.

| Присвоить целочисленной переменной prev 0.

| Присвоить целочисленной переменную next индекс первого появления

| строки delim в строке variants с индекса prev.

| Присвоить целочисленной переменной delta 1.

| Пока next != -1, повторять

| | Присвоить строке tmp подстроку строки varints, начинающуюся.

| | c prev и заканчивающуюся (next – prev).

| | Занести в массив variantsTemp tmp на последнюю позицию.

| | Прибавить к prev (next + delta).

| конец цикла

| Присвоить строке tmp подстроку строки varints, начинающуюся с prev.

| Занести в массив variantsTemp tmp на последнюю позицию.

| Присвоить variants “”.

| Присвоить целочисленной переменной variantsCount размер variantsTemp.

| Присвоить булевой переменной numOfCorAnsFound false.

| Присвоить целочисленной переменной s 0.

| Пока s < variantsCount и НЕ numsOfCorAnsFound, повторять

| | Если variantsTemp[s] == correctAns[i], то

| | | Присвоить numOfCorAnsFound true.

| | | Присвоить correctAns[i] s+1.

| | конец ветвления

| | Увеличить s на единицу

| конец цикла

| Присвоить целочисленной переменной k 0.

| Пока k < variantsCount, повторять

| | Присвоить строке tmp i+1.

| | Прибавить к variants temp.

| | Прибавить к variants “. ”.

| | Прибавить к variants variantsTemp[i].

| | Прибавить к variants ‘\n’.

| | Увеличить k на единицу

| конец цикла

| Присвоить QA[1][i] variants.

| Изменить размер QA[0][i] на (размер QA[0][i] - 1).

| Прибавить к QA[0][i] ‘\n’.

| увеличить i на единицу.

конец цикла

1. Конец алгоритма.

### 1.2.22 Алгоритм получения случайных вопросов и ответов из теста

Дано:

allQA – константный двумерный массив строкового типа

allcorrectAns – константный одномерный массив строкового типа

resultQA – ссылка на двумерный массив строкового типа

resultcorrectAns – ссылка на одномерный массив строкового типа

qCount – константная целочисленная переменная

1. Изменить размер массива resultQA на 3.
2. Присвоить целочисленной переменной QAcount размер allQA[0].
3. Объявить целочисленный массив mentioned.
4. Присвоить целочисленной переменной i 0.
5. Пока i < qCount, повторять

| Объявить целочисленную переменную generatedIndex.

| Повторять, пока в mentioned есть элемент, равный i

| | Присвоить generatedIndex остаток от деления сгенерированного целого числа

| | на Asize.

| конец цикла

| Занести на последнюю позицию mentioned generatedIndex.

| Занести на последнюю позицию resultQA[0] allQA[0][generatedIndex].

| Занести на последнюю позицию resultQA[1] allQA[1][generatedIndex].

| Занести на последнюю позицию resultQA[2] allQA[2][generatedIndex].

| Занести на последнюю позицию resultCorrectAns allCorrectAns[generatedIndex].

| Увеличить i на единицу.

конец цикла

1. Конец алгоритма.

### 1.2.23 Алгоритм получения вопросов и ответов для определенного уровня сложности

Дано:

allQA – константный двумерный массив строкового типа

allcorrectAns – константный одномерный массив строкового типа

resultQA – ссылка на двумерный массив строкового типа

resultcorrectAns – ссылка на одномерный массив строкового типа

lvl – константная строковая переменная

1. Изменить размер массива resultQA на 3.
2. Присвоить целочисленной переменной QAcount размер allQA[0].
3. Присвоить целочисленной переменной i 0.
4. Пока i < QAcount, повторять

| Если allQA[2][i] == lvl, то

| | Занести на последнюю позицию resultQA[0] allQA[0][i].

| | Занести на последнюю позицию resultQA[1] allQA[1][i].

| | Занести на последнюю позицию resultQA[2] allQA[2][i].

| | Занести на последнюю позицию resultCorrectAns allCorrectAns[i].

| | конец ветвления

конец цикла

1. Конец алгоритма.

### 1.2.24 Алгоритм получения нужных вопросов и ответов

Дано:

allQA – константный двумерный массив строкового типа

allcorrectAns – константный одномерный массив строкового типа

resultQA – ссылка на двумерный массив строкового типа

resultcorrectAns – ссылка на одномерный массив строкового типа

qCount – константная целочисленная переменная

lvlSelection – константная строковая переменная

1. Если lvlSelection == “4”, то

| Вызвать вспомогательный алгоритм получения случайных вопросов и ответов из

| теста 1.2.22 с allQA, allcorrectAns, resultQA, resultcorrectAns, qCount в качестве

| параметров.

иначе

| Вызвать вспомогательный алгоритм вопросов и ответов для определенного

| уровня сложности 1.2.23 с allQA, allcorrectAns, resultQA, resultcorrectAns,

| lvlSelection в качестве параметров.

| конец ветвления

конец ветвления

1. Конец алгоритма.

### 1.2.25 Алгоритм тестирования

Дано:

allQA – константный двумерный массив строкового типа

allcorrectAns – константный одномерный массив строкового типа

resultQA – ссылка на двумерный массив строкового типа

resultcorrectAns – ссылка на одномерный массив строкового типа

qCount – константная целочисленная переменная

lvlSelection – константная строковая переменная

1. Объявить двумерный массив строкового типа QAOutVer.
2. Объявить одномерный массив строкового типа CorrectAnsOutVer.
3. Вызвать вспомогательный алгоритм получения нужных вопросов и ответов 1.2.24 с allQA, allCorrectAns, QAOutVer, CorrectAnsOutVer, countOfAskingQ, lvlSelection, как параметрами.
4. Присвоить целочисленной переменной realAskingQ размер correctAnsOutVer.
5. Вызвать вспомогательный алгоритм подготовки тестовых заданий к выводу 1.2.21 с QAOutVer, CorrectAnsOutVer, realAskingQ.
6. Если QAOutVer не пустой, то

| Присвоить countOfAskingQ размер QAOutVer[0].

| Вывести “Выбран ”.

| Если lvlSelection = 4, то

| | Вывести “уровень со случайно выбираемыми заданиями.”/

| иначе

| | Вывести lvlSelection.

| | Вывести “ уровень заданий.” и перейти новую строку.

| конец ветвления

| Вывести инструкцию по тестированию.

| Присвоить целочисленным переменным correctAnsCount, wrongAnsCount,

| go\_on, i значение 0.

| Пока i < realAskingQ, повторять

| | Вывести “Вопрос:” и перейти на новую строку.

| | Вывести QAOutVer[0][i] и перейти на новую строку.

| | Вывести “Варианты ответов:” и перейти на новую строку.

| | Вывести QAOutVer[1][i] и перейти на новую строку.

| | Объявить строку userAns.

| | Вывести “Введите ответ на вопрос (вводить нужно сам ответ, а не его

| | номер)”.

| | Перейти на новую строку.

| | Ввести userAns.

| | Если userAns не равно “0”, то

| | | Если correctAns[i] = userAns, то

| | | | Вывести “Вы ответили правильно.”.

| | | | Увеличить correctAnsCount на единицу.

| | | иначе

| | | | Вывести “Вы ответили неправильно.”.

| | | | Увеличить wrongAnsCount на единицу.

| | | конец ветвления

| | иначе

| | | Увеличить wrongAnsCount на единицу.

| | конец ветвления

| | Вывести “Текущий результат: ” и перейти на новую строку.

| | Вывести “Количество правильных ответов: ”.

| | Вывести correctAnsCount и перейти на новую строку.

| | Вывести “Количество неправильных ответов: ”.

| | Вывести wrongAnsCount и перейти на новую строку.

| | Остановить выполнение программы до нажатия клавиши.

| | Очистить экрана консоли.

| конец цикла

| Вывести “Тестирование завершено. Итоговый результат: ”

| Вывести “Количество правильных ответов: ”.

| Вывести correctAnsCount и перейти на новую строку.

| Вывести “Количество неправильных ответов: ”.

| Вывести wrongAnsCount и перейти на новую строку.

| Если correctAnsCount >= wrongAnsCount, то

| | Вывести “Тестирование прошло успешно.”.

| иначе

| | Вывести “Тестирование прошло неуспешно.” и перейти на новую строку.

| конец ветвления

иначе

| Вывести “Не нашлось вопроса с выбранными настройками.”.

конец ветвления

1. Конец алгоритма.

### 1.2.26 Алгоритм добавления новой заголовочной строки

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

1. Вывести “Введите заголовок главы: ”.
2. Ввести строку headerContent.
3. Присвоить строке resultHeader “\t&”.
4. Прибавить к resultHeader headerContent.
5. Прибавить к resultHeader ‘&’.
6. Занести в chapters[changingChap] на последнюю позицию resultHeader.
7. Конец алгоритма.

### 1.2.27 Алгоритм добавления новой текстовой строки

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

rowNum – константная целочисленная переменная

1. Вывести “Введите новую строку для добавления: ”.
2. Ввести строку newRow.
3. Вставить в chapters[changingChap] на позицю rowNum newRow.
4. Конец алгоритма.

### 1.2.28 Алгоритм добавления новой главы

chapters – константный двумерный массив строкового типа

addingChapNum – константная целочисленная переменная

1. Объявить одномерный массив newChapter.
2. Вставить в chapters на позицю addingChapNum newChapter.
3. Вызвать вспомогательный алгоритм добавления новой заголовочной строки 1.2.26 с chapters и addingChapNum в качестве параметров.
4. Присвоить целочисленной переменной go\_on 1.
5. Вывести “Введите 0, чтобы закончить изменения новой главы, любое другое целое число – чтобы продолжить: ”.
6. Ввести go\_on.
7. Присвоить целочисленной переменной addingRow 1.
8. Пока go\_on не равно 0, повторять

| Изменить размер chapters[addingChapNum] на (addingRow + 1).

| Вызвать вспомогательный алгоритм добавления новой текстовой строки 1.2.27 с

| chapters, addingChapNum и addingRow в качестве параметров.

| Увеличить addingRow на единицу.

| Вывести “Введите 0, чтобы закончить изменения новой главы, любое другое

| целое число – чтобы продолжить: ”.

| Ввести go\_on.

конец цикла

1. Прибавить к chapters[addingChapNum][addingRow – 1] ‘#’.
2. Вывести “Глава после изменений: ” и перейти на новую строку.
3. Вызвать вспомогательный алгоритм вывода главы теории 1.2.4 с chapters, addingChapNum в качестве параметров.
4. Конец алгоритма.

### 1.2.29 Алгоритм выбора главы для изменения

chapters – константный двумерный массив строкового типа

1. Записать в целочисленную переменную chapsCount количество элементов chapters.
2. Присвоить целочисленной переменной i значение 0.
3. Пока ***i < chapsCount***, повторять

| Вызвать вспомогательный алгоритм вывода главы теории 1.2.4 с chapters и i, как

| параметрами

| Присвоить булевой переменной changed значение false.

| Объявить целочисленную переменную temp.

| Пока changed = false, повторять

| | Повторять, пока temp не равно коду символа стрелки вправо, или стрелки влево,

| | или символа возврата каретки

| | | Вывести “Нажмите стрелку влево, чтобы вернуться на одну страницу назад,

| | | стрелку вправо – чтобы пролистнуть на одну страницу вперед, ENTER –

| | | чтобы выбрать главу для изменения или добавления новой на ее месте: ”.

| | | Ввести temp.

| | конец цикла

| | Если temp = код символа стрелки влево, то

| | | Если i = 0, то

| | | | Вывести “Нельзя пролистнуть назад, так как это первая глава”.

| | | иначе

| | | | Уменьшить i на единицу.

| | | | Присвоить переменной changed значение true.

| | | конец ветвления

| | иначе если temp = код символа стрелки вправо, то

| | | Если i = (pagesCount - 1), то

| | | | Вывести “Нельзя перейти на следующую главу, так как это

| | | | последняя глава.” и перейти на новую строку.

| | | иначе

| | | | Увеличить i на единицу.

| | | | Присвоить переменной changed значение true.

| | | конец ветвления

| | иначе

| | | Вернуть i.

| | конец ветвления

| конец цикла

| Увеличить i на единицу.

конец цикла

1. Конец алгоритма.

### 1.2.30 Алгоритм выбора строки главы для добавления на ее место новой строки

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

1. Объявить целочисленную переменную result.
2. Вывести “Глава ”.
3. Вывести changingChap + 1.
4. Вывести “: ” и перейти на новую строку.
5. Присвоить целочисленной переменной rowsCountOfChap количество элементов chapters[changingChap].
6. Присвоить целочисленной переменной j значение 1.
7. Пока j < rowsCountOfPage, повторять

| Вывести j+1.

| Вызвать вспомогательный алгоритм вывода строки теории 1.2.6 с

| chapters[changingChap][j], как параметром.

| Увеличить j на единицу.

конец цикла

1. Повторять, пока result < 2 или result > (rowsCountOfChap+1) или result не равно -1

| Вывести “Введите номер строки главы, на место которой хотите добавить новую

| (для добавления в конец главы, введите ”.

| Вывести rowsCountOfChap + 1.

| Вывести “)”.

| Вывести “Чтобы ничего не добавлять, введите -1: ”.

| Ввести result.

конец цикла

1. Вернуть result.
2. Конец алгоритма.

### 1.2.31 Алгоритм выбора строк главы для их изменения

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

beginRow – ссылка на целочисленную переменную

endRow – ссылка на целочисленную переменную

1. Вывести “Глава ”.
2. Вывести changingChap + 1
3. Вывести “: ” и перейти на новую строку.
4. Присвоить целочисленной переменной rowsCountOfChap количество элементов chapters[changingChap].
5. Присвоить целочисленной переменной j значение 0.
6. Пока j < rowsCountOfPage, повторять

| Вывести j+1.

| Вывести “: ” и перейти на новую строки.

| Вызвать вспомогательный алгоритм вывода строки теории 1.2.3 от

| chapters[changingChap][j].

| Увеличить j на единицу.

конец цикла

1. Повторять, пока beginRow < 1 или beginRow > rowsCountOfChap

| Вывести “Введите номер начальной строки главы для изменениями: ”.

| Ввести beginRow.

конец цикла

1. Уменьшить beginRow на единицу.
2. Повторять, пока endRow < beginRow или endRow > rowsCountOfChap

| Вывести “ Введите номер конечной строки главы для изменениями: ”.

| Ввести endRow.

конец цикла

1. Уменьшить endRow на единицу.
2. Конец алгоритма.

### 1.2.32 Алгоритм получения типа строки

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

rowNum – константная целочисленная переменная

1. Если rowNum = 0, то

| Вернуть 0.

иначе

| Присвоить целочисленной переменной rowsSize длину строки

| chapters[changingChap][rowNum].

| Присвоить символьной переменной lastSymbolOfRow

| chapters[changingChap][rowNum][rowSize-1].

| Если lastSymbolOfRow = ‘\n’, то

| | Присвоить lastSymbolOfRow chapters[changingChap][rowNum][rowSize-2].

| конец ветвления

| Если lastSymbolOfRow = ‘&’, то

| | Вернуть 1.

| иначе

| | Вернуть 2.

| конец ветвления

конец ветвления

1. Конец алгоритма.

### 1.2.33 Алгоритм удаления строки из главы

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

deletingRow – константная целочисленная переменная

rowType – константная целочисленная переменная

1. Если rowType = 0, то

| Если размер chapters[changingChap] > 2, то

| | Прибавить к chapters[changingChap][deletingRow-1] ‘#’.

| | Удалить из chapters[changingChap] последний элемент.

| иначе

| | Удалить из chapters[changingChap] последний элемент.

| конец ветвления

иначе

| Удалить из chapters[changingChap] элемент с индексом deletingRow.

конец ветвления

1. Конец алгоритма.

### 1.2.34 Алгоритм изменения строк

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

beginRow – константная целочисленная переменная

endRow – константная целочисленная переменная

1. Если beginRow не равно endRow, то

| Вывести “Строки с ”.

| Вывести beginRow + 1.

| Вывести “ по ”.

| Вывести endRow + 1.

| Вывести “ главы № ”.

| Вывести changingRow.

| Вывести “: ” и перейти на новую строку.

иначе

| Вывести “Строка №”.

| Вывести beginRow + 1.

| Вывести “ главы № ”.

| Вывести changingRow.

| Вывести “: ” и перейти на новую строку.

конец ветвления

1. Присвоить целочисленной переменной i beginRow.
2. Пока i <= endRow, повторять

| Вывести chapters[changingChap][i] и перейти на новую строку.

| Присвоить целочисленной переменной rowType значение, возвращаемое

| вспомогательным алгоритмом получения типа строки 1.2.32 с chapters,

| changingChap, i в качестве параметров.

| Вывести “Вводите текст для изменения .

| Вывести i + 1.

| Вывести “ строки: ” и перейти на новую строку.

| Ввести строку newRow.

| Если строка newRow пустая, то

| | Вызвать вспомогательный алгоритм удаления строки из главы 1.2.33 с

| | chapters, changingChap, i, rowType в качестве параметров.

| иначе

| | Если rowType = 0, то

| | | Присвоить chapters[chaningChap][i] newRow.

| | | Прибавить к chapters[chaningChap][i] ‘#’.

| | иначе если rowType = 1

| | | Присвоить chapters[chaningChap][i] “\t&”.

| | | Прибавить к chapters[chaningChap][i] newRow.

| | | Прибавить к chapters[chaningChap][i] “&\n”.

| | конец ветвления

| конец ветвления

конец цикла

1. Конец алгоритма.

### 1.2.35 Алгоритм изменения главы

chapters – константный двумерный массив строкового типа

changingChap – константная целочисленная переменная

1. Объявить целочисленную переменную changeOrAdd.
2. Повторять, пока changeOrAdd < 1 или changeOrAdd > 2

| Вывести “1. Изменить существующие строки.” и перейти на новую строку.

| Вывести “2. Добавить новые строки. ” и перейти на новую строку.

| Вывести “Введите номер действия: ”.

| Ввести changeOrAdd.

конец цикла

1. Если changeOrAdd = 1, то

| Объявить целочисленные переменный beginRow, endRow.

| Вызвать вспомогательный алгоритм выбора строк главы для их изменения 1.2.31

| с chapters, changingChap, beginRow и endRow в качестве параметров.

| Вызвать вспомогательный алгоритм изменения строк 1.2.34

| с chapters, changingChap, beginRow и endRow в качестве параметров.

иначе если changeOrAdd = 2, то

| Объявить целочисленные переменные beginRow и endRow.

| Вызвать вспомогательный алгоритм выбора строк главы для их изменения 1.2.31

| с chapters, changingChap, beginRow и endRow в качестве параметров.

| Удалить элементы chapters[changingChap] с beginRow по endRow.

иначе

| Объявить целочисленную переменную addingRowNum.

| Повторять, пока addingRowNum не равно -1

| | Присвоить addingRowNum результат вспомогательного выбора строки главы

| | для добавления на ее место новой строки 1.2.30 с chapters и changingChap

| | Если addingRowNum не равно -1, то

| | | Вызвать вспомогательный алгоритм добавления новой текстовой строки

| | | 1.2.32 с chapters, changingChap, addingRowNum в качестве параметров.

| | | Очистить консоль.

| | | Вывести “Глава после изменений: ” и перейти на новую строку.

| | | Вызвать вспомогательный алгоритм вывода главы из теории 1.2.4 с chapters

| | | и changingChap.

| | конец ветвления

| конец цикла

конец ветвления

1. Очистить консоль.
2. Вывести “Глава после изменений: ” и перейти на новую строку.
3. Вызвать вспомогательный алгоритм вывода главы из теории 1.2.4 с chapters и changingChap.
4. Конец алгоритма.

### 1.2.36 Алгоритм изменения файла с теорией

theoryPath – строковая константа

chapters – двумерный массив строкового типа

1. Если файл с именем theoryPath открылся на запись, то

| Присвоить целочисленной переменной chaptersCount размер chapters.

| Присвоить целочисленной переменной i 0.

| Пока i < chaptersCount, повторять

| | Присвоить целочисленной переменной rowsCount размер chapters[i].

| | Присвоить целочисленной переменной j 0.

| | Пока j < (rowsCount-1), повторять

| | | Вывести в файл chapters[i][j].

| | | Увеличить j на единицу.

| | конец цикла

| | Присвоить целочисленной переменной lastRowSize размер

| | chapters[i][rowsCount-1].

| | Если chapters[i][rowsCount-1][lastRowSize-1] = ‘\n’, то

| | | Присвоить chapters[i][rowsCount-1][lastRowSize-1] ‘#’.

| | | Прибавить к chapters[i][rowsCount-1] “\n”.

| | иначе

| | | Прибавить к chapters[i][rowsCount-1] “#\n”.

| | конец ветвления

| | Вывести в файл chapters[i][rowsCount-1].

| | Увеличить i на единицу.

| конец цикла

| Вернуть истину.

конец ветвления

1. Вернуть ложь.
2. Конец алгоритма.

### 1.2.37 Алгоритм изменения теории

theoryPath – строковая константа

1. Присвоить двумерному массиву строкового типа chapters результат вспомогательного алгоритма 1.2.2 с theoryPath.
2. Объявить целочисленную переменную changeOrAdd.
3. Повторять, пока changeOrAdd < 1 или changeOrAdd > 2

| Вывести “1. Изменить существующие главы.” и перейти на новую строку.

| Вывести “2. Добавить новые главы. ” и перейти на новую строку.

| Вывести “Введите номер действия: ”.

| Ввести changeOrAdd.

конец цикла

1. Очистить консоль.
2. Если changeOrAdd = 1, то

| Присвоить целочисленную переменной changingChap результат вспомогательного

| алгоритма выбора главы для изменения 1.2.29 с chapters в качестве параметра.

| Вызвать вспомогательный алгоритм изменения главы 1.2.35

| с chapters, changingChap в качестве параметров.

иначе

| Присвоить целочисленной переменной addingRowNum результат

| вспомогательного алгоритма выбора главы для изменения 1.2.29 с chapters в

| качестве параметра.

| Вызвать вспомогательный алгоритм добавления новой главы 1.2.28 с chapters

| и addingChap в качестве параметров.

конец ветвления

1. Если вспомогательный алгоритм изменения файла с теорией 1.2.36 возвращает истину, то

| вернуть истину

конец ветвления

1. Вернуть ложь.
2. Конец алгоритма.

### 1.2.38 Алгоритм создания переходов на новую строку

Q – строковая константа

1. Присвоить qSize количество символов строки Q.
2. Присвоить целочисленной переменной i 0.
3. Пока i < qSize, повторять

| Если Q[i] = ‘&’, то

| | Присвоить Q[i] ‘\n’.

| конец ветвления

конец цикла

1. Конец алгоритма.

### 1.2.39 Алгоритм разделения строки

str – строка

delim – строка

Объявить массив строк result.

Переменная prev := 0.

Переменная delta := длина строки delim.

Переменная next := индекс элемента str после подстроки delim, начиная с prev элемента.

Пока next ≠ -1, повторять

| Строковая переменная tmp := подстрока str, начиная с prev элемента, до (next –

| prev) элемента.

| Занести tmp на последнюю позицию result.

| prev := next + delta.

| next := индекс элемента str после подстроки delim, начиная с prev элемента.

конец цикла

Вернуть result.

### 1.2.40 Алгоритм анализа деструкторов

destructors – массив строк

Переменная destCount := размер destuctors.

Переменная i := 0.

Пока i < destCount, повторять

| Если destuctors[i] пустая, то

| | Вернуть ложь.

| конец ветвления.

| Увеличить i на единицу.

конец цикла

Вернуть истину.

### 1.2.41 Алгоритм добавления тестового вопроса

QA – двумерный массив строкового типа

correctAns – одномерный массив строкового типа

countOfAllQ – ссылка на целочисленную переменную

1. Увеличить countOfAllQ на единицу.
2. Изменить размер QA[0] на countOfAllQ.
3. Изменить размер QA[1] на countOfAllQ.
4. Изменить размер QA[2] на countOfAllQ.
5. Изменить размер correctAns на countOfAllQ.
6. Объявить строковые переменные Q, destructors, corAns, lvl.
7. Вывести “Введите текст вопроса (вводите & для перехода на новую строку в файле”.
8. Ввести Q.
9. Вызвать вспомогательный алгоритм создания переходов на новую строку 1.2.38 с Q.
10. Прибавить к Q ‘#’.
11. Присвоить QA[0][countOfAllQ-1] Q.
12. Объявить массив строк variantsTemp.
13. Повторять, пока размер variantsTemp < 2 или вспомогательный алгоритм анализа деструкторов 1.2.40 с variantsTemp в качестве параметра возвращает ложь

| Вывести “Введите деструкторы, разделяя их знаком &:” и перейти новую строку.

| Ввести destructors.

| Присвоить variantsTemp результат работы вспомогательного алгоритма

| разделения строки 1.2.39 с destuctors и “&” в качестве параметров.

1. Присвоить QA[1][countOfAllQ-1] destructors.
2. Повторять, пока lvl не равно “1” и lvl не равно “2” и lvl не равно “3”

| Вывести “Введите уровень сложности вопроса (1, 2 или 3): ”.

| Ввести lvl.

конец цикла

1. Присвоить QA[2][countOfAllQ-1] lvl.
2. Повторять, пока в variantsTemp нет элемента, равного corAns

| Вывести “Введите правильный ответ на данный вопрос: ”

| Ввести corAns.

конец цикла

1. Присвоить correctAns[countOfAllQ-1] corAns.
2. Конец алгоритма.

### 1.2.42 Алгоритм удаления тестового вопроса

QA – двумерный массив строкового типа

correctAns – одномерный массив строкового типа

countOfAllQ – ссылка на целочисленную переменную

countOfAllQ –целочисленная константа

1. Уменьшить countOfAllQ на единицу.
2. Удалить элемент QA[0] с индексом (deletingQ – 1).
3. Удалить элемент QA[1] с индексом (deletingQ – 1).
4. Удалить элемент QA[2] с индексом (deletingQ – 1).
5. Удалить элемент correctAns с индексом (deletingQ – 1).
6. Конец алгоритма.

### 1.2.43 Алгоритм изменения тестового вопроса

QA – двумерный массив строкового типа

correctAns – одномерный массив строкового типа

countOfAllQ – ссылка на целочисленную переменную

edditingQ –целочисленная константа

1. Объявить строковые переменные Q, destructors, corAns, lvl.
2. Объявить целочисленную переменную way.
3. Повторять, пока way < 1 или way > 5

| Вывести “1. Изменить текст вопроса.”.

| Вывести “2. Изменить варианты ответа.”.

| Вывести “3. Изменить правильный ответ.”.

| Вывести “4. Изменить уровень сложности вопроса.”.

| Вывести “5. Вернуть назад.”.

| Вывести “Введите номер действия: ”.

| Ввести way.

конец цикла

1. Если way = 1, то

| Вывести “Введите текст вопроса (вводите & для перехода на новую строку).

| Ввести Q.

| Вызвать вспомогательный алгоритм создания переходов на новую строку 1.2.43 с

| Q.

| Прибавить к Q ‘#’.

| Присвоить QA[0][ edditingQ -1] Q.

иначе если way = 2, то

| Объявить массив строк variantsTemp.

| Повторять, пока размер variantsTemp < 2 или вспомогательный алгоритм анализа

| деструкторов 1.2.40 с variantsTemp в качестве параметра возвращает ложь

| | Вывести “Введите деструкторы, разделяя их знаком &:”.

| | Ввести destructors.

| | Присвоить variantsTemp результат работы вспомогательного алгоритма

| | разделения строки 1.2.39 с destuctors и “&” в качестве параметров.

| конец цикла

| Присвоить QA[1][ edditingQ -1] destructors.

иначе если way = 3, то

| Присвоить массиву строк variantsTemp результат работы вспомогательного

| алгоритма разделения строки 1.2.39 с QA[1][edittingQ-1] и “&”.

| Повторять, пока в variantsTemp нет элемента, равного corAns

| | Вывести “Введите правильный ответ на данный вопрос: ”.

| | Ввести corAns.

| конец цикла

| Присвоить correctAns[edditingQ -1] corAns.

иначе если way = 4, то

| Повторять, пока lvl не равно “1” и lvl не равно “2” и lvl не равно “3”

| | Вывести “Введите уровень сложности вопроса (1, 2 или 3): ”.

| | Ввести lvl.

| конец цикла

| Присвоить QA[2][ edditingQ -1] lvl.

1. Конец алгоритма.

### 1.2.44 Алгоритм изменения файла с тестами

theoryPath – строковая константа

QA – двумерный массив строкового типа

correctAns – одномерный массив строкового типа

countOfAllQ – целочисленная константа

countOfAskingQ – целочисленная константа

lvlSelection – строковая константа

1. Если файл с именем theoryPath открылся на запись, то

| Присвоить строке countNLvl countOfAskingQ.

| Прибавить к строке countNLvl “ ”.

| Прибавить к строке countNLvl lvlSelection.

| Вывести в файл countNLvl и перейти новую строку в файле.

| Присвоить целочисленной переменной i 0.

| Пока i < countOfAllQ-1, повторять

| | Вывести в файл QA[0][i] и перейти новую строку в файле.

| | Вывести в файл QA[1][i] и перейти новую строку в файле.

| | Вывести в файл correctAns[i] и перейти новую строку в файле.

| | Вывести в файл QA[2][i] и перейти новую строку в файле.

| | Увеличить i на единицу.

| конец цикла

| Вывести в файл QA[0][ countOfAllQ-1] и перейти новую строку в файле.

| Вывести в файл QA[1][ countOfAllQ-1] и перейти новую строку в файле.

| Вывести в файл correctAns[countOfAllQ-1] и перейти новую строку в файле.

| Вывести в файл QA[2][ countOfAllQ-1] и перейти новую строку в файле.

| Вернуть истину.

конец ветвления

1. Вернуть ложь.
2. Конец алгоритма.

### 1.2.45 Алгоритм вывода всех тестовых заданий из файла

QA – двумерный массив строкового типа

correctAns – одномерный массив строкового типа

countOfAllQ – целочисленная константа

countOfAskingQ – целочисленная константа

lvlSelection – строковая константа

1. Вывести “Количество задаваемых вопросов на 4 уровне сложности: ”.
2. Вывести countOfAskingQ и перейти на новую строку.
3. Вывести “Текущий уровень сложности тестирования: ”.
4. Вывести lvlSelection и перейти на новую строку 2 раза.
5. Присвоить целочисленной переменной i 0.
6. Пока i < countOfAllQ

| Вывести “Вопрос №”.

| Вывести i+1.

| Вывести “. Уровень сложности: ”.

| Вывести QA[2][i] и перейти на новую строку.

| Вывести “Текст вопроса: ” и перейти новую строку.

| Вывести QA[0][i] и перейти новую строку.

| Вывести “Варианты ответов: ” и перейти новую строку.

| Вывести QA[1][i] и перейти новую строку.

| Вывести “Правильный ответ: ”.

| Вывести correctAns[i] и перейти новую строку.

1. Конец алгоритма.

### 1.2.46 Алгоритм изменения тестирования

theoryPath – строковая константа

1. Объявить двумерный строковый массив QA.
2. Объявить одномерный строковый массив corAns.
3. Объявить целочисленные переменные allQ, askingQ.
4. Объявить строковую переменную lvlSelection.
5. Вывести “Файл с тестовыми заданиями: ” и перейти на новую строку.
6. Вызвать вспомогательный алгоритм получения тестовых вопросов из файла 1.2.20 с testPath, QA, corAns, allQ, askingQ, lvlSelection в качестве параметров.
7. Скопировать в двумерный массив строк QAOutVer значения из QA.
8. Скопировать в одномерный массив строк corAnsOutVer значения из corAns.
9. Вызвать вспомогательный алгоритм подготовки тестовых заданий к выводу 1.2.21 с QAOutVer, corAnsOutVer, allQ в качестве параметров.
10. Вызвать вспомогательный алгоритм вывода всех тестовых заданий из файла 1.2.46 с QAOutVer, corAnsOutVer, allQ, askingQ, lvlSelection в качестве параметров.
11. Объявить целочисленную переменную way.
12. Повторять, пока way не равно 0

| Повторять, пока way < 0 или way > 5

| | Вывести “1. Добавить вопрос.” и перейти новую строку.

| | Вывести “2. Удалить вопрос.” и перейти новую строку.

| | Вывести “3. Редактировать существующий вопрос.” и перейти новую строку.

| | Вывести “4. Выбрать уровень сложности тестирования.” и перейти новую

| | строку.

| | Вывести “5. Выбрать количество задаваемых вопросов (только для 4 уровня

| | сложности.” и перейти новую строку.

| | Вывести “0. Завершить изменения файла с тестированием.” и перейти новую

| | строку.

| | Ввести way.

| конец цикла

| Если way = 1, то

| | Вызвать вспомогательный алгоритм добавления тестового вопроса 1.2.41 с

| | QA, corAns, allQ в качестве параметров.

| иначе если way = 2, то

| | Если allQ = 0, то

| | | Вывести “0 вопросов в файле.” и перейти новую строку.

| | иначе

| | | Объявить целочисленную переменную deletingQ.

| | | Повторять, пока deletingQ < 1 или deletingQ > allQ

| | | | Вывести “Введите номер удаляемого вопроса: ”.

| | | | Ввести deletingQ.

| | | конец цикла

| | | Вызвать вспомогательный алгоритм удаления тестового вопроса 1.2.42 с

| | | QA, corAns, allQ, deletingQ в качестве параметров.

| | конец ветвления

| иначе если way = 3, то

| | Если allQ = 0, то

| | | Вывести “0 вопросов в файле.” и перейти новую строку.

| | иначе

| | | Объявить целочисленную переменную editingQ.

| | | Повторять, пока editingQ < 1 или editingQ > allQ

| | | | Вывести “Введите номер редактируемого вопроса: ”.

| | | | Ввести editingQ.

| | | конец цикла

| | | Вызвать вспомогательный алгоритм изменения тестового вопроса 1.2.43 с

| | | QA, corAns, allQ, editingQ в качестве параметров.

| | конец ветвления

| иначе если way = 4, то

| | Вывести “Текущий уровень сложности тестирования: ”.

| | Вывести lvlSelection и перейти на новую строку 2 раза.

| | Повторять, пока lvlSelection не равно “1” и lvlSelection не равно “2” и

| | lvlSelection не равно “3” и lvlSelection не равно “4”.

| | | Вывести “Введите уровень сложности тестирования (1, 2, 3 или 4): ”.

| | | Ввести lvlSelection.

| | конец цикла

| иначе если way = 5, то

| | Если allQ < 5, то

| | | Вывести “Меньше 5 вопросов в файле. Добавьте вопросы, чтобы

| | | выбирать количество задаваемых вопросов” и перейти новую строку.

| | иначе

| | | Вывести “Количество задаваемых вопросов на 4 уровне сложности: ”.

| | | Вывести askingQ и перейти новую строку.

| | | Повторять, пока askingQ < 5 или askingQ > allQ

| | | | Вывести “Введите количество задаваемых вопросов (не менее 5): ”.

| | | | Ввести askingQ.

| | | конец цикла

| | конец ветвления

| конец ветвления

| Очистить консоль.

| Если way не равно 0, то

| | Если вспомогательный алгоритм изменения файла с тестами 1.2.44 с

| | testPath, QA, corAns, allQ, askingQ, lvlSelection в качестве параметров.

| | | Вывести “Файл тестирования обновлен.” и перейти новую строку.

| | | Очистить массив QA.

| | | Очистить массив corAns.

| | | Вывести “Файл с тестовыми заданиями: ” и перейти новую строку.

| | | Вызвать вспомогательный алгоритм получения тестовых вопросов из

| | | файла 1.2.20 с testPath, QA, corAns, allQ, askingQ, lvlSelection в качестве

| | | параметров.

| | | Скопировать в QAOutVer значения из QA.

| | | Скопировать в corAnsOutVer значения из corAns.

| | | Вызвать вспомогательный алгоритм подготовки тестовых заданий к

| | | выводу 1.2.21 с QAOutVer, corAnsOutVer, allQ в качестве

| | | параметров.

| | | Вызвать вспомогательный алгоритм вывода всех тестовых заданий из

| | | файла 1.2.46 с QAOutVer, corAnsOutVer, allQ, askingQ, lvlSelection

| | | в качестве параметров.

| | иначе

| | | Вывести “Ошибка открытия файла с тестированием.” и перейти на новую

| | | строку.

| | конец ветвления

| конец ветвления

конец цикла

1. Конец алгоритма.

## 1.3 Входные и выходные данные

Входные данные:

* целое число: выбор пункта меню;
* двумерный вещественный массив: матрица системы;
* двумерный строковый массив: страницы теоритического материала;
* двумерный строковый массив: все вопросы и вариантов ответов к ним;
* одномерный строковый массив: все правильные ответы;
* строка: ответ пользователя на вопрос в тесте.

Выходные данные:

* одномерный вещественный массив: решение системы.
* двумерный строковый массив: 5 вопросов и варианты ответов к ним;
* одномерный строковый массив: 5 правильных ответов к 5 вопросам;

## 1.4 Системные требования

Рекомендуемая конфигурация:

* Intel-совместимый процессор с частотой не менее 1,6 ГГц;
* не менее 2 Гб ОЗУ;
* не менее 20 MБ свободного места на диске;
* дисковод CD-ROM/DVD-ROM.

Операционная система: Windows 10.

# 2 РАБОЧИЙ ПРОЕКТ

## 2.1 Общие сведения о работе системы

Программный продукт разработан в интегрированной среде Visual Studio Community 2019 (версия 16.9.4) на языке C++. Программа работает под управлением операционной системы Windows 10.

## 2.2 Функциональное назначение программного продукта

Разработанный программный продукт предназначен для отработки навыков решения систем линейных уравнений методом Гаусса. Программа имеет следующие функциональные возможности:

* предоставление пользователю просмотра теории по теме “Решение систем линейных уравнений методом Гаусса”;
* предоставление пользователю ввода системы линейных уравнений для демонстрации работы метода Гаусса;
* предоставление пользователю просмотра демонстрации работы метода Гаусса;
* предоставление пользователю прохождения тестирования по теме “Решение систем линейных уравнений методом Гаусса”;
* проверка правильности ответа пользователем на задание в тесте;
* вывод текущего результата после каждого ответа на задание;
* прекращение тестирования после ответов на все задания;
* вывод итогового результата после ответов на все задания;

Программа имеет следующие функциональные ограничения:

* + количество неизвестных в системе для демонстрации должно быть в диапазоне [2; 6]..
  + коэффициенты при неизвестных и свободные коэффициенты в уравнениях системы  
     должны быть в диапазоне [-50; 50].

## 2.3 Инсталляция и выполнение программного продукта

Для выполнения программы необходимо:

1. Скопировать на жесткий диск компьютера папку gauss, содержащую исполняемый файл gauss.exe и текстовые файлы: theory.txt и test.txt.
2. Запустить на выполнение файл gauss.exe.

## 

## 2.4 Описание программы

На рисунке 2.1 показана диаграмма взаимодействия модулей программы

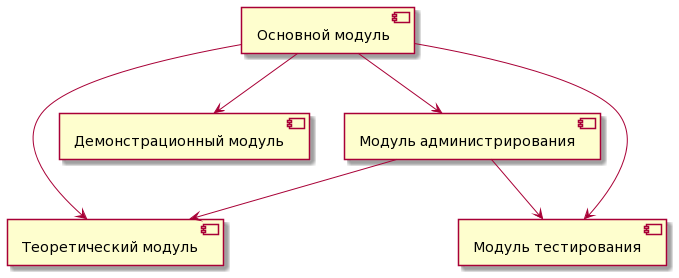


Рисунок 2.1 – Диаграмма взаимодействия модулей программы

В программе модуль main является основным модулем, который вызывает остальные модули. Модуль theory – модуль, отвечающий за показ теории. Модуль demonstration отвечает за демонстрирование работы метода Гаусса. Модуль test отвечает за тестирование. Модуль admin – модуль, отвечающий за редактирование файлов с теорией и тестами

В таблице 2.1 приведены функции теоретического модуля программы.

Таблица 2.1 – Функции теоретического модуля программы

| **Прототип** | **Назначение** |
| --- | --- |
| vector<vector<string>> GetChapters(const string theoryPath) | Возвращает массив глав из файла с теоретическим материалом |
| void OutRow(const wstring row) | Выводит строку row из теории |
| void OutTheoryChap(const vector<vector<wstring>>& chapters, const int chap) | Выводит главу номер chap из теории |
| void ReadTheory(const vector<vector<string>>&chapters) | Выводит главы теории на экран, позволяет менять просматриваемую главу |

В таблице 2.2 приведены функции демонстрационного модуля программы.

Таблица 2.2 – Функции демонстрационного модуля программы

| **Прототип** | **Назначение** |
| --- | --- |
| int GetMethodOfFilling() | Возвращает номер метода заполнения системы линейных уравнений для демонстрационной части |
| void FillWithRandomNums(vector<vector<double>>& v, const int rows, const int cols, const int min, const int max) | Заполняет матрицу v с rows строками и cols столбцами случайными числами в диапазоне [min; max] |
| void FillByKeyboard(vector<vector<double>>& v, const int rows, const int cols) | Заполняет матрицу v с rows строками и cols столбцами с клавиатуры |
| bool FillFromFile(vector<vector<double>>& v, const string path) | Заполняет матрицу v из файла с именем path. Если файл открылся, то возвращает истину,  иначе – ложь. |
| bool FillMatrix(vector<vector<double>>& v, const int method) | Заполняет матрицу v. Если матрица заполнена, возвращает истину, иначе – ложь. |
| void Copy(vector<vector<double>>& In, vector<vector<double>>& Out) | Копирует матрицу In в матрицу Out |
| void printRow(vector<vector<double>> mat, const int row) | Выводит на экран row строку матрицы mat |
| void print(vector<vector<double>> mat) | Выводит на экран матрицу mat |
| void swap\_rows(vector<vector<double>>& mat, int i, int j) | Меняет местами i-ую и j-ую строки матрицы mat местами |
| void mulRowOnSomeNum(vector<vector<double>>& mat, const int row, const double num) | Умножает row строку матрицы mat на число num |
| void cleanLeadings(vector<vector<double>>& mat) | Образует единицы на главной диагонали матрицы mat, деля каждый элемент строки на элемент главной диагонали |
| void ShowAnswer(vector<double> x) | Выводит решение системы на экран |

Продолжение таблицы 2.2

| bool forwardElim(vector<vector<double>>& mat) | Выполняет прямой ход метода Гаусса для матрицы mat. Возвращает истину, если система имеет единственное решения, иначе – ложь. |
| --- | --- |
| bool CanBeSolved(vector<vector<double>>& mat) | Проверяет может ли быть решена система. Если да, то возвращает истину, иначе - ложь |
| vector<double> gaussianElimination(vector<vector<double>>& mat) | Выводит решение системы, возвращает массив решений |

В таблице 2.3 приведены функции модуля тестирования программы.

Таблица 2.3 – Функции модуля тестирования программы

| **Прототип** | **Назначение** |
| --- | --- |
| void GetTests(const string testPath, vector<vector<wstring>>& QA, vector<wstring>& correctAns, int& countOfAllQ, int& countOfAskingQ, wstring& lvlSelection) | Записывает все вопросы с альтернативами на задания в QA, все правильные ответы на задания в correctAns, количество всех вопросов в countOfAllQ, количество задаваемых вопросов в countOfAskingQ и уровень сложности тестирования в lvlSelection из файла с базой вопросов и ответов |
| void GetRandomQ(const vector<vector<wstring>>& allQA, const vector<wstring>& allCorrectAns, vector<vector<wstring>>& resultQA, vector<wstring>& resultCorrectAns, const int qCount) | Случайно записывает в resultQA и resultCorrectAns только qCount вопросов с вариантами ответов и qCount правильных ответов к ним из allQA и allCorrectAns соответственно |
| void PrepareTestsForOut(vector<vector<wstring>>& QA, vector<wstring>& correctAns, const int countOfAllQ) | Редактирует элементы QA и correctAns для вывода на экран |

Продолжение таблицы 2.3

| **Прототип** | **Назначение** |
| --- | --- |
| void GetSomeLvlQ(const vector<vector<wstring>>& allQA, const vector<wstring>& allCorrectAns, vector<vector<wstring>>& resultQA, vector<wstring>& resultCorrectAns, const wstring lvl) | Записывает в resultQA и resultCorrectAns вопросы с вариантами ответов уровня сложности lvl и правильные ответы к ним из allQA и allCorrectAns соответственно |
| void GetNecessaryQ(const vector<vector<wstring>>& allQA, const vector<wstring>& allCorrectAns, vector<vector<wstring>>& resultQA, vector<wstring>& resultCorrectAns, const int qCount, wstring lvlSelection) | Записывает в resultQA и resultCorrectAns вопросы с вариантами ответов и правильные ответы к ним из allQA и allCorrectAns соответственно в зависимости от выбранного уровня сложности тестирования lvlSelection |
| vector<string> Mix(const vector<string> A) | Перемешивает элементы массива A. Возвращает перемешанный массив А |
| bool Has(const vector<int> A, const int num) | Проверяет есть ли в массиве А число num. Если да, то возвращает истину, иначе – ложь |

В таблице 2.4 приведены функции модуля администрирования в программе.

Таблица 2.4 – Функции модуля администрирования в программе

| **Прототип** | **Назначение** |
| --- | --- |
| int AdminMenu() | Выводит меню администрирования. Возвращает выбранный пункт меню администрирования |
| vector<string> Mix(const vector<string> A) | Перемешивает элементы вектора A. Возвращает перемешанный массив А |
| void ReadTheoryChap(const vector<vector<wstring>>& chapters, const int chap) | Выводит на экран chap элемент вектора chapters |
| void AddHeaderRow(vector<vector<wstring>>& chapters, const int addingChap) | Добавляет заголовок в главе addingChap для вектора chapters |

Продолжение таблицы 2.4

| **Прототип** | **Назначение** |
| --- | --- |
| void AddTextRow(vector<vector<wstring>>& chapters, const int addingChap, const int rowNum) | Вставляет обычную строку в главе addingChap под номером rowNum для вектора chapters |
| void ChooiceRowOfChapForChange(const vector<vector<wstring>> chapters, const int changingChap, int& beginRow, int& endRow) | Записывает номер начальной строки в главе changingChap в chapters для изменения в beginRow, конечной строки -–в endRow |
| int GetRowType(const vector<vector<wstring>>& chapters, const int changingChap, const int rowNum) | Возвращает тип строки под номером rowNum в главе changingChap в chapters |
| void AddChapter(vector<vector<wstring>>& chapters, const int addingChapNum) | Вставляет новую главу под номером addingChapNum для вектора chapters |
| int ChooiceChapter(const vector<vector<wstring>> chapters) | Возвращает индекс выбранной главы в chapters для изменения или добавления на ее месте новой главы |
| int ChooiceRowOfChapForAdd(const vector<vector<wstring>> chapters, const int changingChap) | Возвращает номер строки в главе changingChap в chapters для добавления на ее месте новой строки |
| void DeleteRow(vector<vector<wstring>>& chapters, const int changingChap, const int deletingRow, const int rowType) | Удаляет строки под номером deletingRow в главе changingChap в chapters |
| void ChangeRows(vector<vector<wstring>> &chapters, const int changingChap, const int beginRow, const int endRow) | Изменяет строки с beginRow по endRow в главе changingChap в chapters |
| void ChangeChapter(vector<vector<wstring>> &chapters, const int changingChap) | Изменяет главу changingChap в chapters |

Продолжение таблицы 2.4

| **Прототип** | **Назначение** |
| --- | --- |
| bool ChangeTheoryFile(const string theoryPath, vector<vector<wstring>> chapters) | Записывает в файл с именем theoryPath вектор с измененными главами chapters. Если файл на запись открылся, то возвращает истину, иначе - ложь |
| bool ChangeTheory(const string theoryPath) | Изменяет теоретический материал в файле с именем theoryPath. Если ChangeTheoryFile(const string theoryPath, vector<vector<wstring>> chapters) возвращает истину, то возвращает истину, иначе - ложь |
| void AddQ(vector<vector<wstring>>& QA, vector<wstring>& correctAns, int& countOfAllQ) | Добавляет новый вопрос, его уровень сложности, варианты ответов к нему в c QA, добавляет правильный ответ к этому вопросу в correctAns, увеличивает countOfAllQ на единицу |
| void DeleteQ(vector<vector<wstring>>& QA, vector<wstring>& correctAns, int& countOfAllQ, const int deletingQ) | Удаляет deletingQ элемент QA[0], QA[1], QA[2] и correctAns, уменьшает countOfAllQ на единицу |
| void DeleteQ(vector<vector<wstring>>& QA, vector<wstring>& correctAns, int& countOfAllQ, const int deletingQ) | Удаляет deletingQ элемент QA[0], QA[1], QA[2] и correctAns, уменьшает countOfAllQ на единицу |
| void MakeNewLines(wstring& Q) | Добавляет в строку вопроса Q символ перехода на новую строку, если такой отсутствовал в конце строки |
| void EditQ(vector<vector<wstring>>& QA, vector<wstring>& correctAns, const int countOfAllQ, const int editingQ) | Редактирует editingQ вопрос, варианты ответов к нему, его уровень сложности и правильный ответ к нему |
| void OutAllTest(vector<vector<wstring>> QA, vector<wstring> correctAns, const int countOfAllQ, const int countOfAskingQ, const wstring lvlSelection) | Выводит количество всех вопросов в файле тестирования, задаваемых вопросов на 4 уровне сложности и все вопросы, варианты ответов, правильные ответы, уровни сложности к ним |

Продолжение таблицы 2.4

| **Прототип** | **Назначение** |
| --- | --- |
| bool OutToFile(const string testPath, const vector<vector<wstring>> QA, const vector<wstring> correctAns, const int countOfAllQ, const int countOfAskingQ, const wstring lvlSelection) | Записывает в файл с именем testPath измененныые QA, correctAns, countOfAskingQ и lvlSelection |
| void ChangeTest(const string testPath) | Изменяет файл тестирования с именем testPath |

В таблице 2.5 приведены функции основного модуля.

Таблица 2.5 – Функции основного модуля

| **Прототип** | **Назначение** |
| --- | --- |
| int Menu() | Выводит главное меню. Возвращает выбранный пункт меню |
| int main() | Главная функция программы. Вызывается остальные функции. Если программа завершает работу корректно, возвращает 0, иначе – 1. |

В таблице 2.5 приведены важнейшие переменные, используемые в программе.

Таблица 2.5 – Важнейшие переменные программы

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение** |
| chooice | целочисленный | Номер выбранного пользователем задания |
| chapters | вектор векторов строк | Главы теоретического материала |
| matr | вектор векторов вещественных чисел | Расширенная матрица системы для демонстрации работы метода Гаусса |

|  |  |  |
| --- | --- | --- |
| x | вектор вещественных чисел | Решение системы |

Продолжение таблицы 2.5

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение** |
| QA | вектор векторов строк | 5 вопросов и варианты ответов к ним |
| correctAns | одномерный вектор строк | Правильные ответов к 5 вопросам |
| userAns | строковый | Ответ пользователя на задание |
| correctAnsCount | целочисленный | Количество правильных ответов пользователя |
| wrongAnsCount | целочисленный | Количество неправильных ответов пользователя |

## 2.5 Разработанные меню и интерфейсы

После запуска программы на выполнение в консольном окне появится главное меню с выбором действия (рис.2.2).

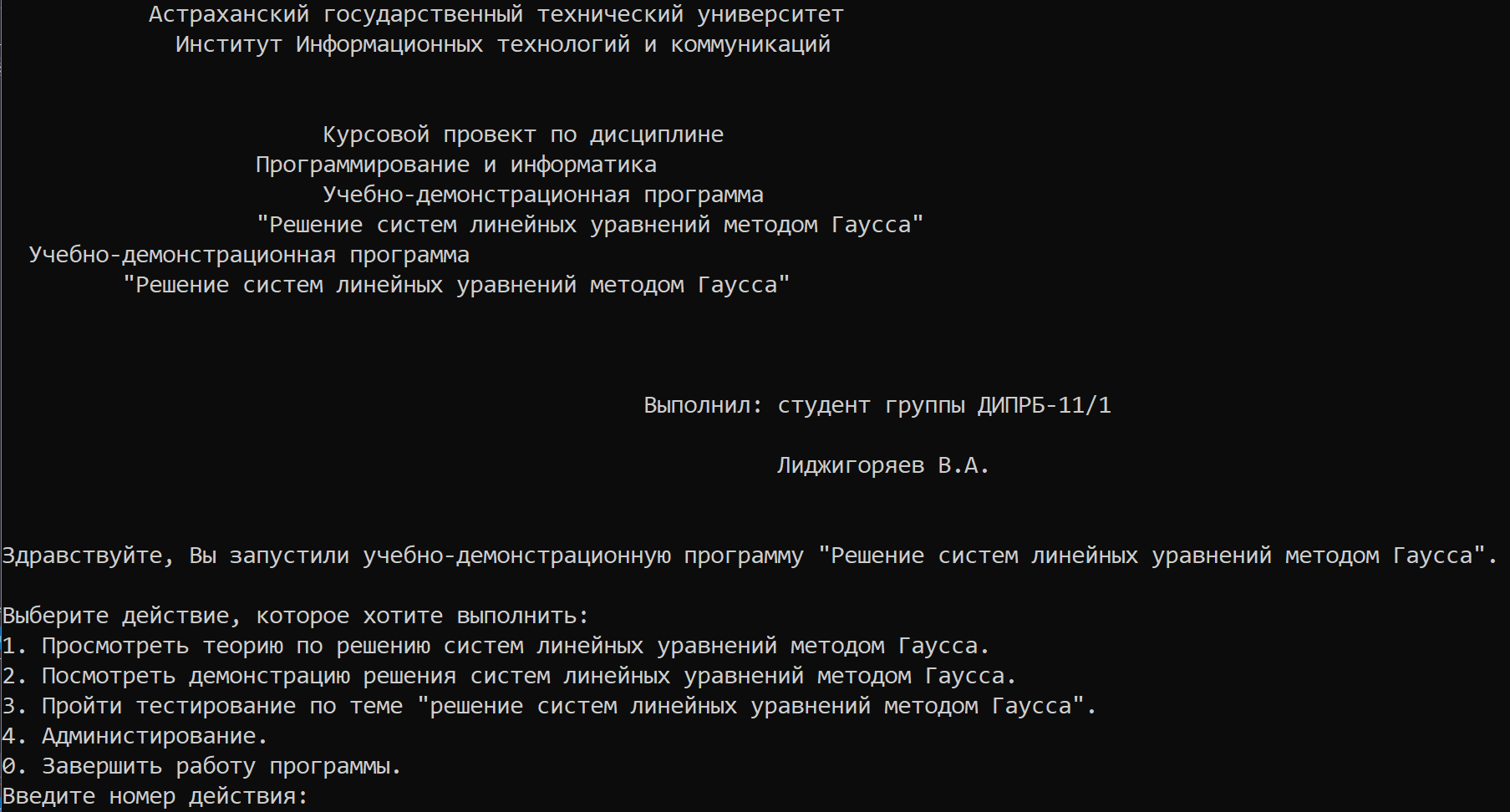


Рисунок 2.2 – Консоль программы с главным меню

При выборе пункта 1 пользователю будет предоставлена возможность просмотреть

теоретический материал с пролистыванием вперед или назад (рис. 2.3).

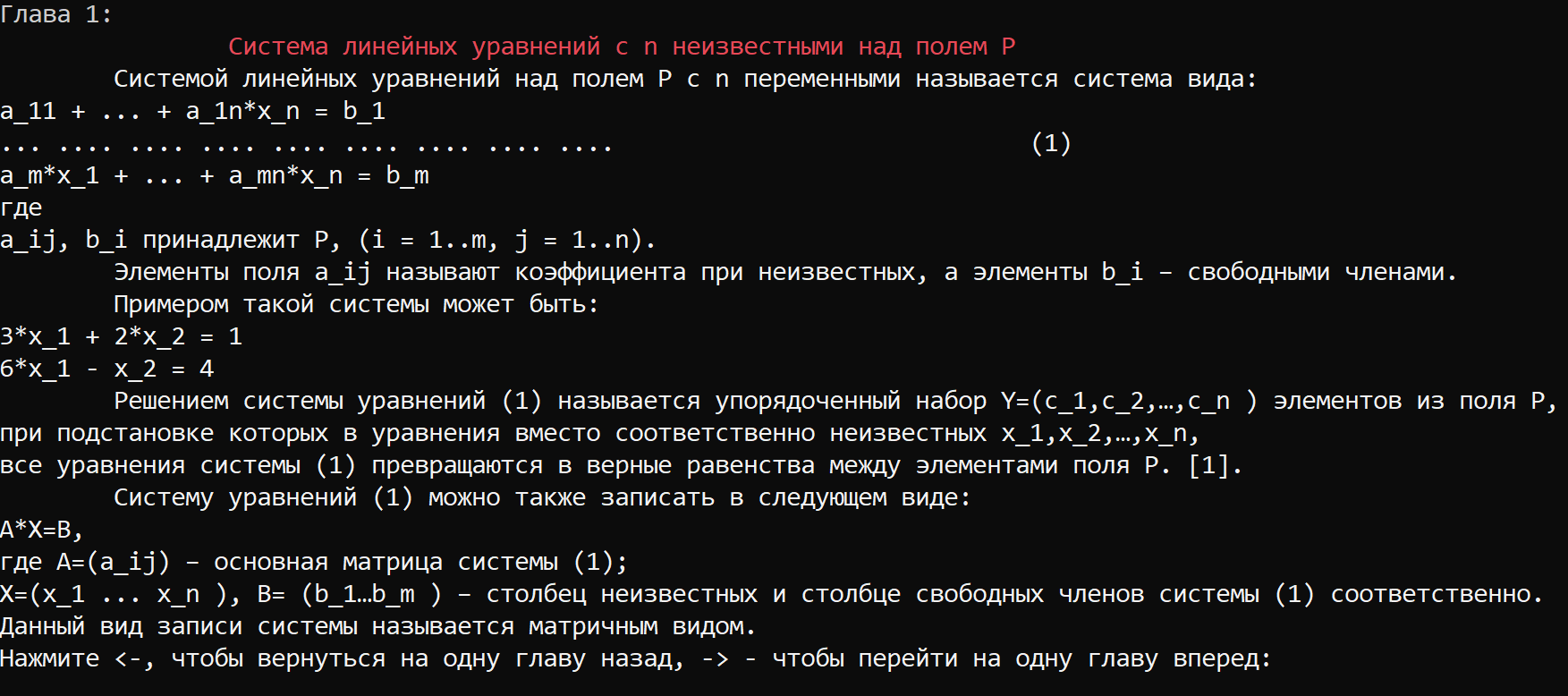


Рисунок 2.3 – Консоль программы с теоретическим материалом

Пользователь может пролистнуть вперед, нажав стрелку вправо, или пролистнуть назад, нажав стрелку влево. Если пользователь просматривает первую страницу материала и нажимает стрелку влево, то появится предупреждение о том, что назад пролистнуть нельзя, так как это первая страница, оно показано на рисунке 2.4.

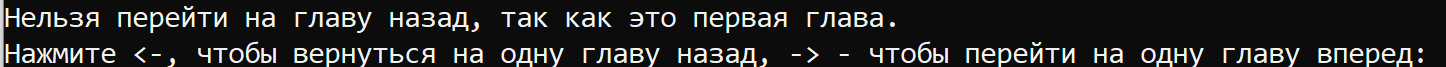


Рисунок 2.4 – Предупреждение о том, что назад пролистнуть нельзя

При нажатии стрелки вправо материал пролистывается на одну страницу вперед   
(рис 2.5).

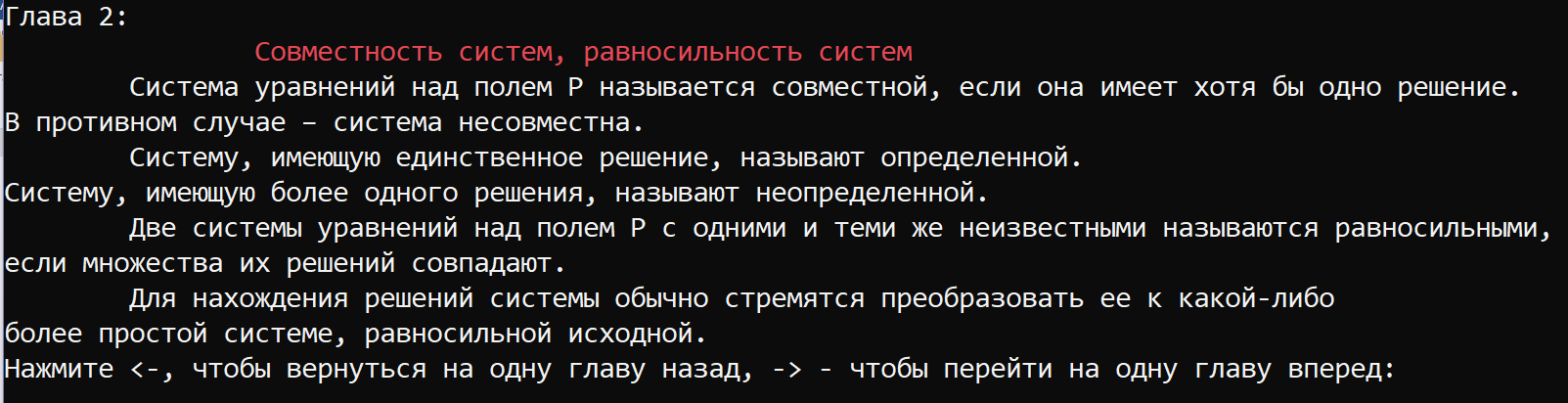


Рисунок 2.5 – Вторая страница теории

После того, как пользователь на последней странице нажмет стрелку вправо, выведутся уведомление о том, что просмотр теоретического материала завершен, и главное меню, которые показаны на рисунке 2.6.

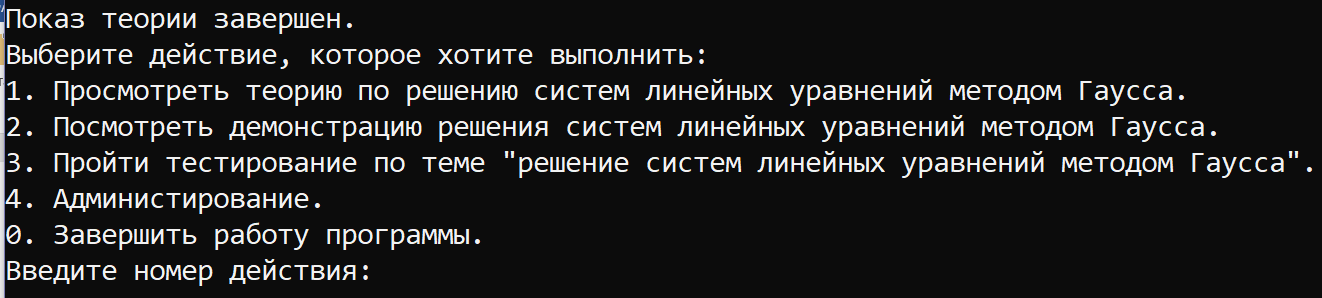


Рисунок 2.6 – Уведомление о завершении показа теории

При выборе пункта 2 главного меню (см. рис. 2.2) пользователю будет предложено выбрать способ ввода системы линейных уравнений в матричной форме и на примере нее посмотреть работу метода Гаусса(рис. 2.7).

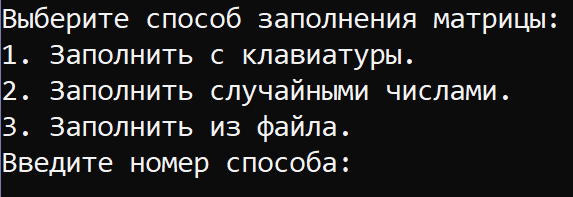


Рисунок 2.7 – Выбор способа ввода системы линейных уравнений

При выборе пункта 1 пользователь должен ввести количество уравнений в системе и элементы матрицы (рис. 2.8).

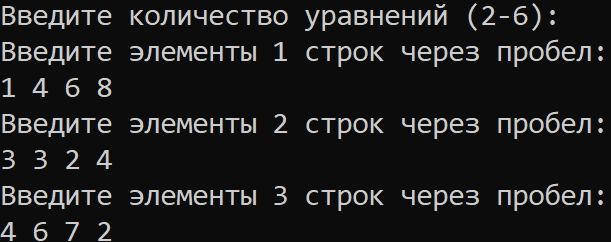


Рисунок 2.8 – Заполнение матрицы с клавиатуры

Далее показывается решение введенной системы линейных уравнений методом Гаусса, строки меняются местами.

На рисунке 2.9 показан процесс изменения строк местами.

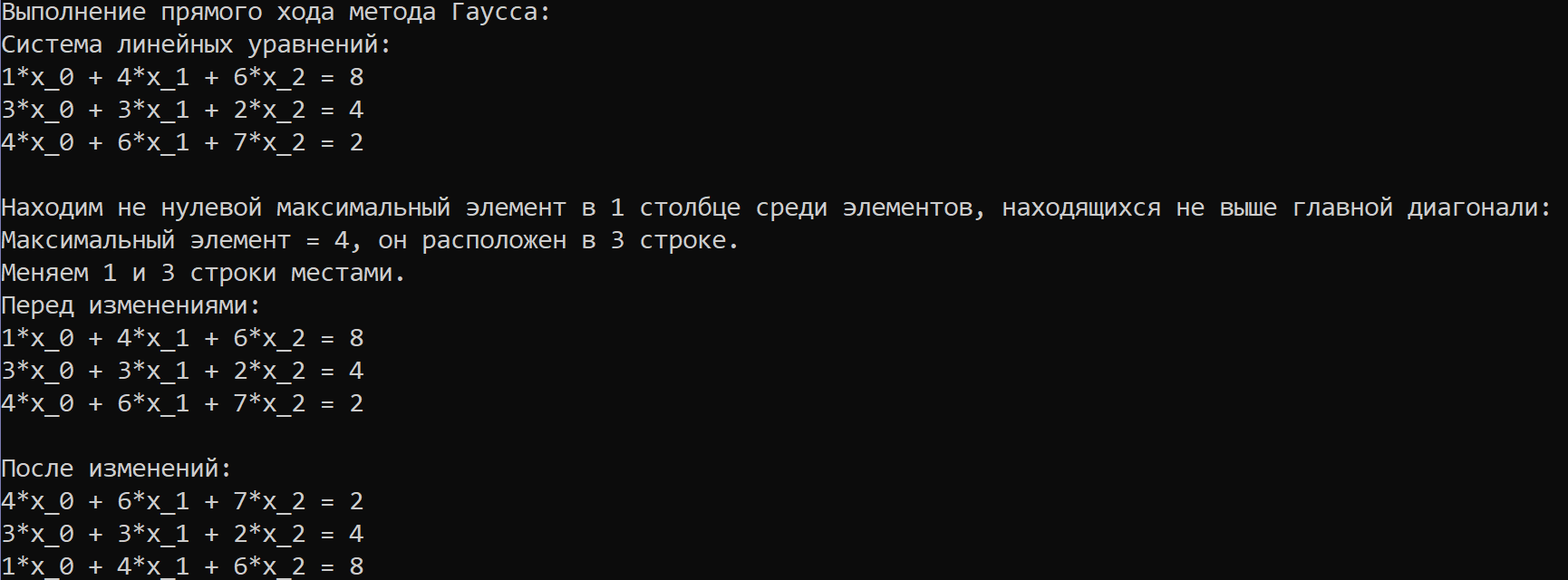


Рисунок 2.9 – Строки меняются местами

На рисунке 2.10 находится коэффициент для обнуления второй строки первого столбца и вторая строка делится на него.

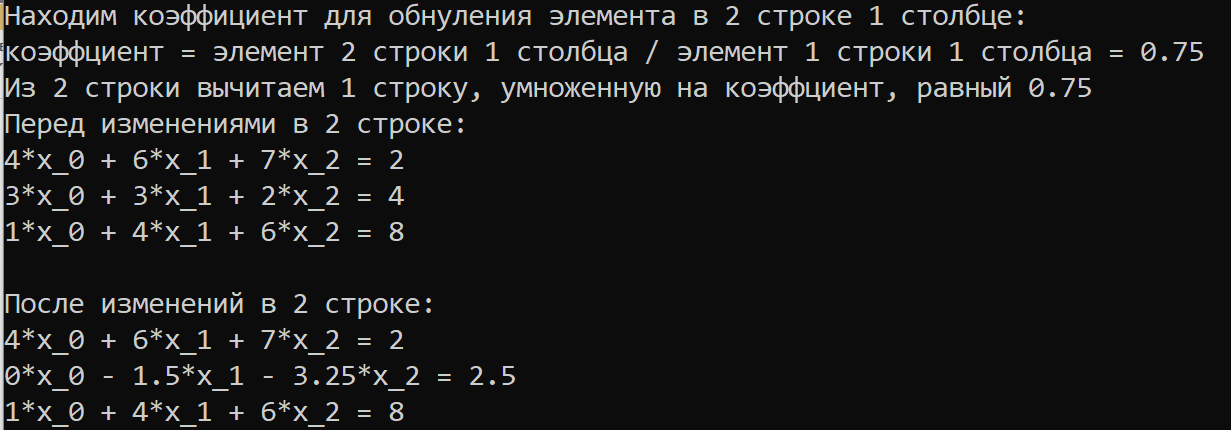


Рисунок 2.10 – Изменение 2 строки

На рисунке 2.11 находится коэффициент для обнуления третьей строки первого столбца и третья строка делится на него.

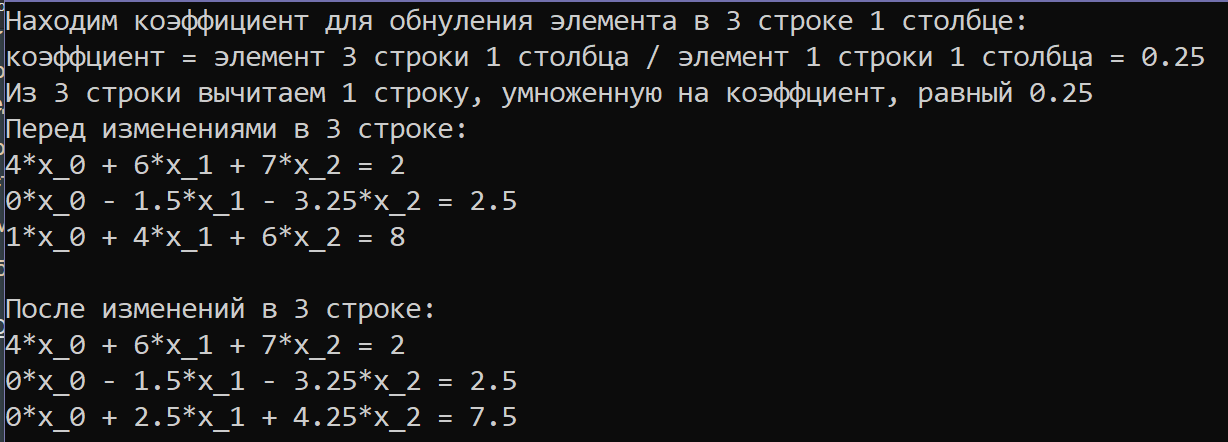


Рисунок 2.11 – Изменение 3 строки

Таким образом, была исключена переменная х\_0 из второго и третьего уравнений. Аналогично будет происходить и с другими и переменными.

Когда не нужно будет переставлять строки, будет происходить очищение главной диагонали. На рисунке 2.12 находится коэффициент для изменения первой строки.

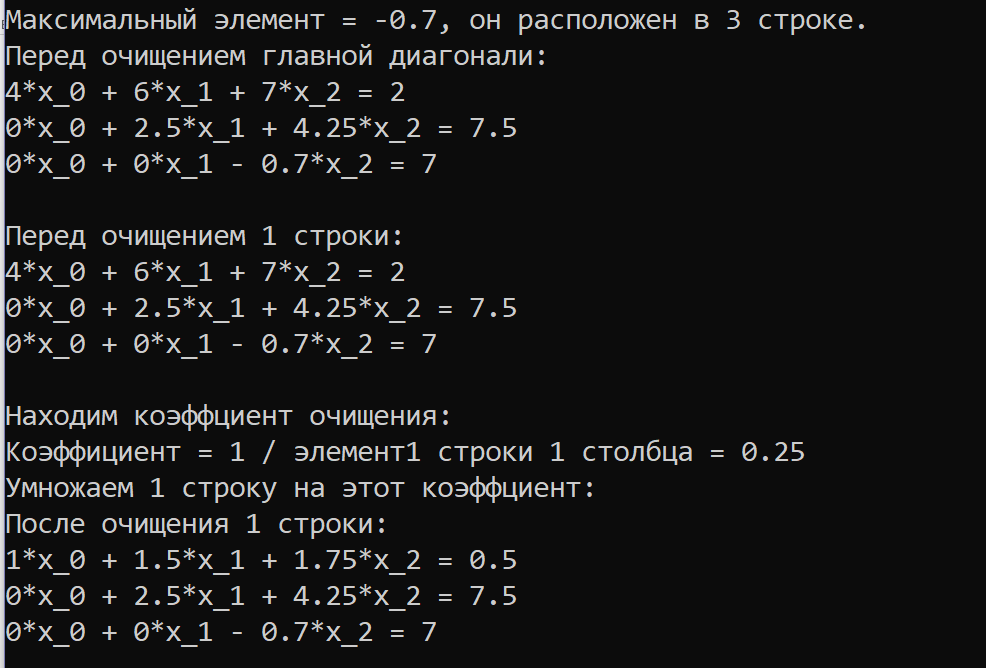


Рисунок 2.12 – Очищение 1 строки

На рисунке 2.13 находится коэффициент для очищения второй строки.

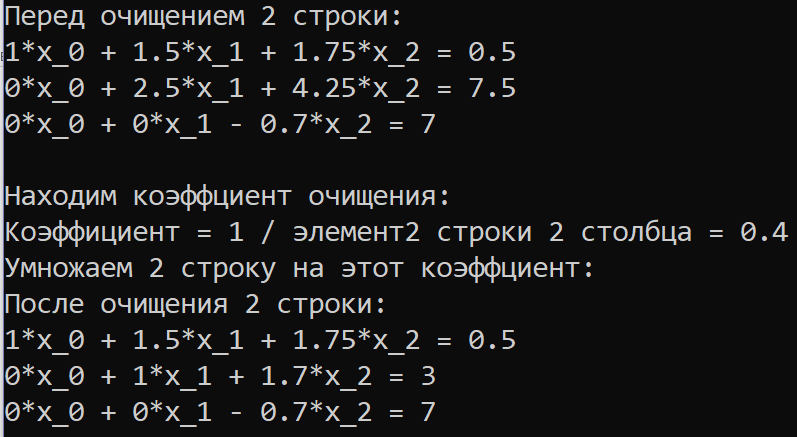


Рисунок 2.13 – Очищение 2 строки

На рисунке 2.14 находится коэффициент для очищения третьей строки.

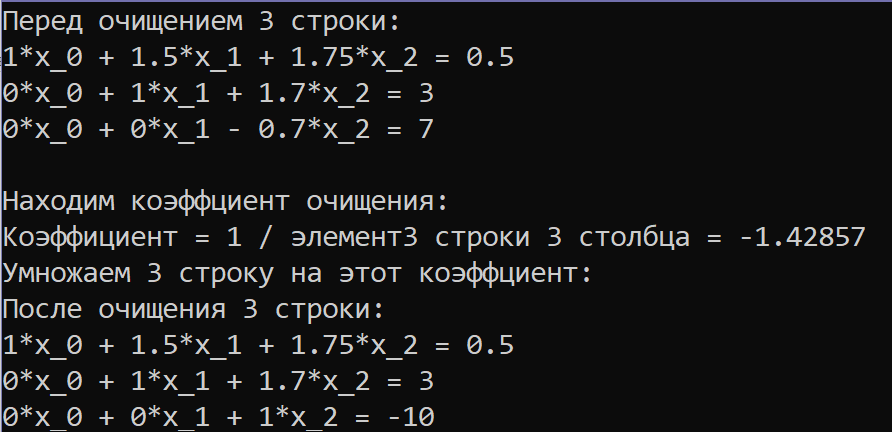


Рисунок 2.14 – Очищение 3 строки

Если введенная система имеет единственное решение, то по окончании показа решения системы методом Гаусса будет показан ответ, вид которого показан на рисунке 2.15, и выведено главное меню. Если же введенная система не имеет единственного решения, то будет выведено уведомление об этом на этапе решения.

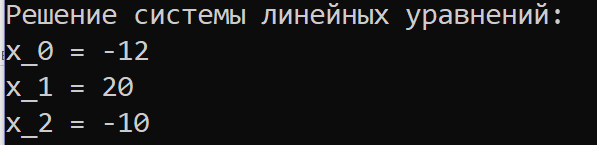


Рисунок 2.15 – Вывод решения системы линейных уравнений

Уведомление о том, что система не имеет единственного решения, показано на   
рисунке 2.16.

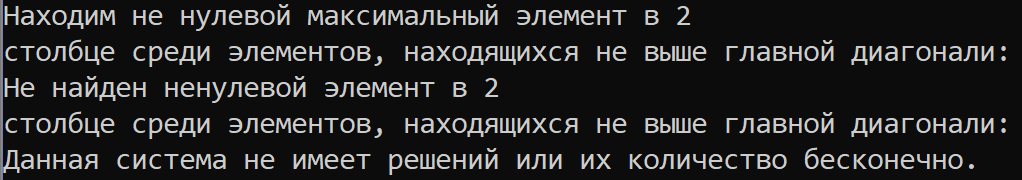


Рисунок 2.16 – Уведомление о том, что нет единственного решения

При выборе пункта 2 меню выбора способа заполнения матрицы (см. рис. 2.7) пользователь должен ввести количество уравнений в системе и границы генерации случайных   
чисел.

На рисунке 2.17 показано заполнение матрицы случайными числами.

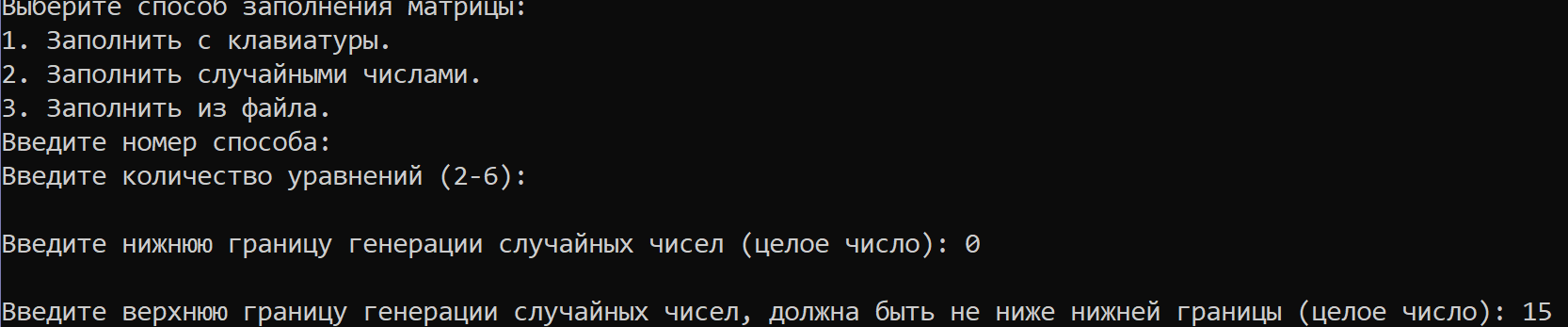


Рисунок 2.17 – Заполнение матрицы случайными числами

Система заполнится случайными числа и далее показывается решение введенной системы линейных уравнений методом Гаусса, вид решения показан на рисунке 2.7.

При выборе пункта 3 меню выбора способа заполнения матрицы (см. рис. 2.15) пользователь должен ввести название текстового файла с матрицей (2.18).

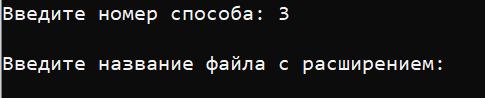


Рисунок 2.18 – Заполнение матрицы из файла

Далее будет аналогичный вывод решения для данной системы.

При выборе пункта 3 главного меню (см. рис. 2.2) пользователю будет выведена инструкция по тестированию, показанная на рисунке 2.19.

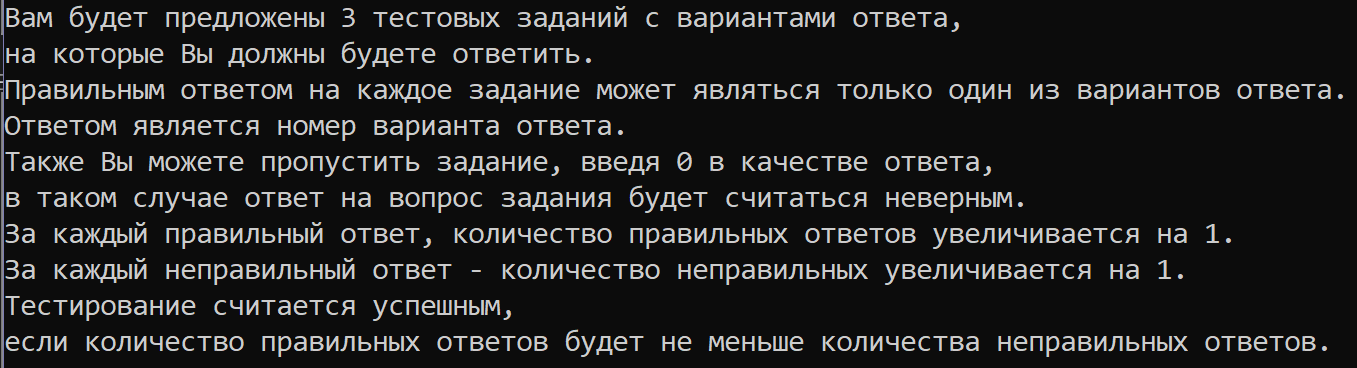


Рисунок 2.19 – Инструкция по тестированию

После того, как пользователь введёт ответ, ему будут сообщены результаты проверки. Если пользователь правильно ответил на вопрос, ему будет выведено уведомление о том, что он решил задание правильно и выведен текущий результат.

Уведомление о правильности решения задания и текущий результат тестирования показаны на рисунке 2.20

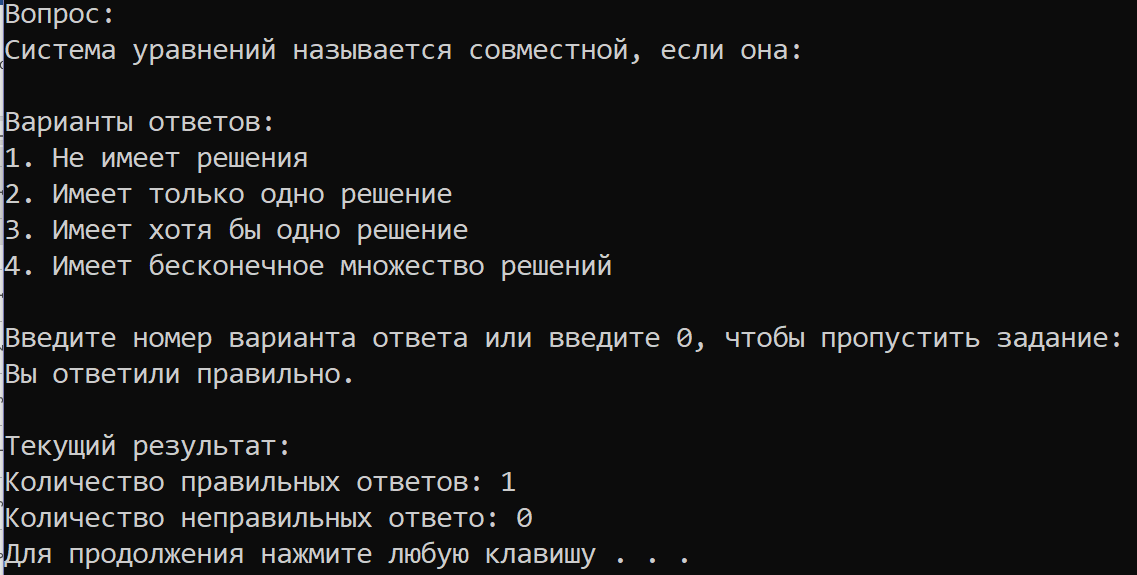


Рисунок 2.20 – Уведомление о правильном ответе

Если пользователь ошибся или ввел 0, ему будет выведено уведомление о неправильности решения задания и текущий результат (рис. 2.21).

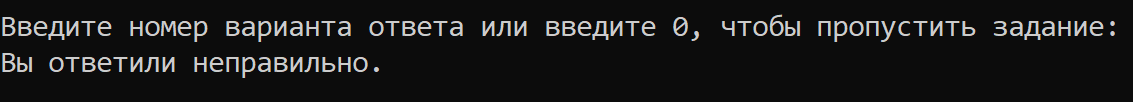


Рисунок 2.21 – Уведомление о неправильном ответе

Тестирование будет продолжаться до тех пор, пока не будут выведены все задания по данному уровню тестирования и не будут введены ответы к ним. По окончании тестирования пользователю выведется итоговый результат по тесту. Если количество правильных ответов не меньше количества неправильных, то будет выведено также уведомление об успешном тестировании, которое показано на рисунке 2.22.

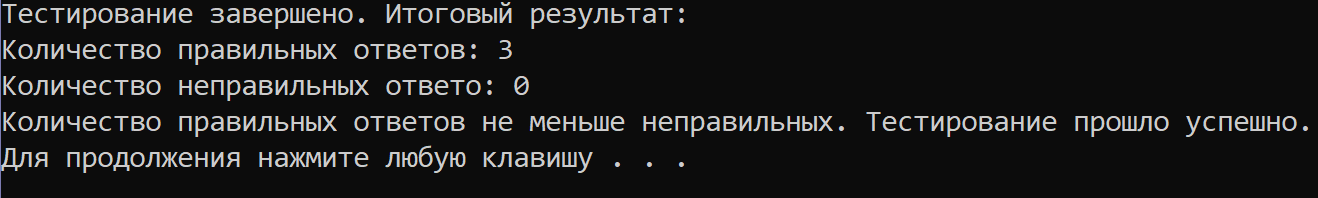


Рисунок 2.22 – Итоговый результат и уведомление об успешном тестировании

Если правильных ответов меньше неправильных, то будет выведено уведомление о неуспешном тестировании

При выборе пункта 4 главного меню (см. рис. 2.2) пользователю будет ввести пароль. Если ввести “admin123”, то произойдет вход и будет выведено меню администратора.

Меню администратора показано на рисунке 2.23

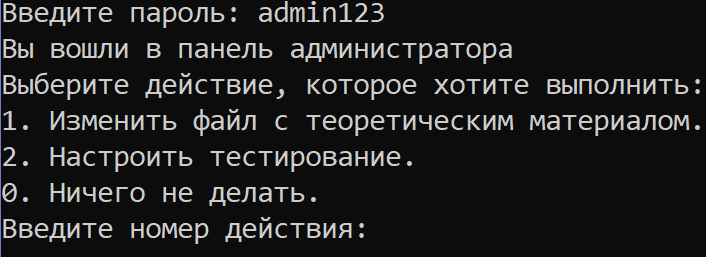


Рисунок 2.23 – Меню администратора

При выборе пункта 1 в меню администратора появится меню изменения теории   
(рис. 2.24).

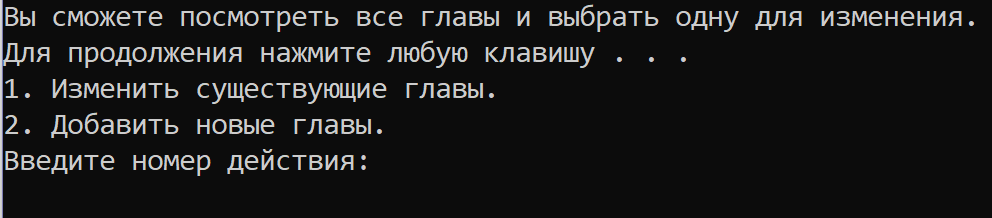


Рисунок 2.24 – Меню изменения теории

При выборе любого из пунктов пользователю будет предоставлена возможность просмотреть теоретический материал с пролистыванием вперед или назад и выбором главы для изменения или добавления на ее месте новой нажатием клавиши ENTER (рис. 2.25).

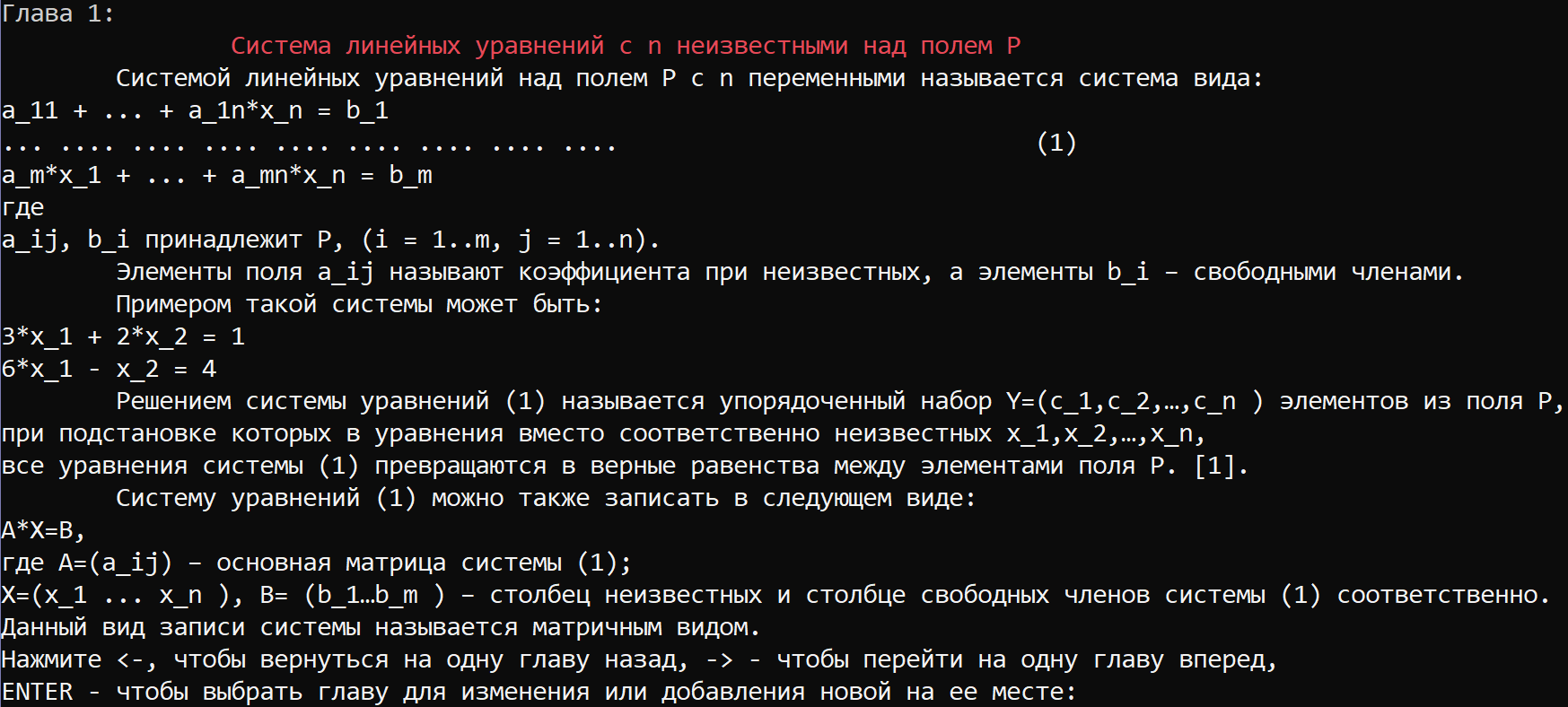


Рисунок 2.25 – Выбор главы для изменения или добавления

Если был выбран пункт 1 в меню изменения теории (см. рис 2.24), то появится меню изменения главы.

На рисунке 2.26 показано меню изменения главы.

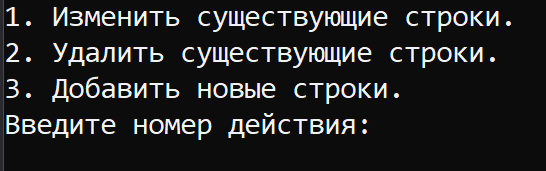


Рисунок 2.26 – Меню изменения главы

При выборе любого из пунктов администратору требуется ввести номера начальной строки для изменения и конечной строки для изменения (рис. 2.27).

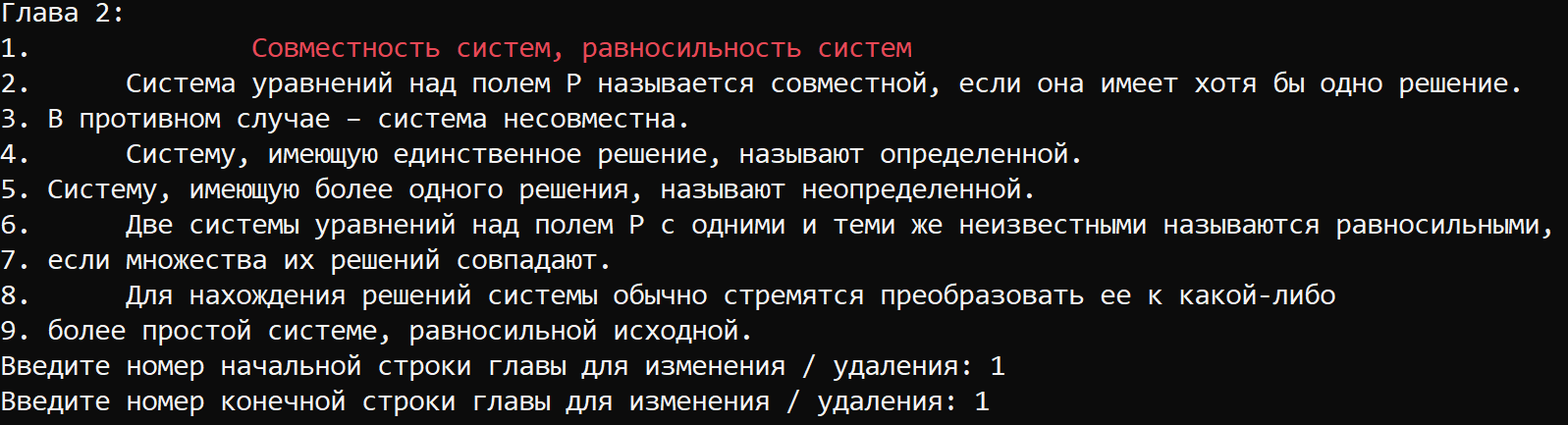


Рисунок 2.27 – Выбор строк для изменения

Если был выбран пункт 2 меню изменения главы (см. рис. 2.26), то выбранные строки будут удалены, иначе – администратору будет предоставлена возможность ввести строки для изменения или добавления на их месте новых. После этого будет выведена измененная глава, аналогично простому выводу главы (см. рис. 2.25).

При выборе пункта 2 в меню администратора (см. рис. 2.23) выведутся все вопросы из базы вопросов и появится меню изменения тестирования (рис. 2.28).

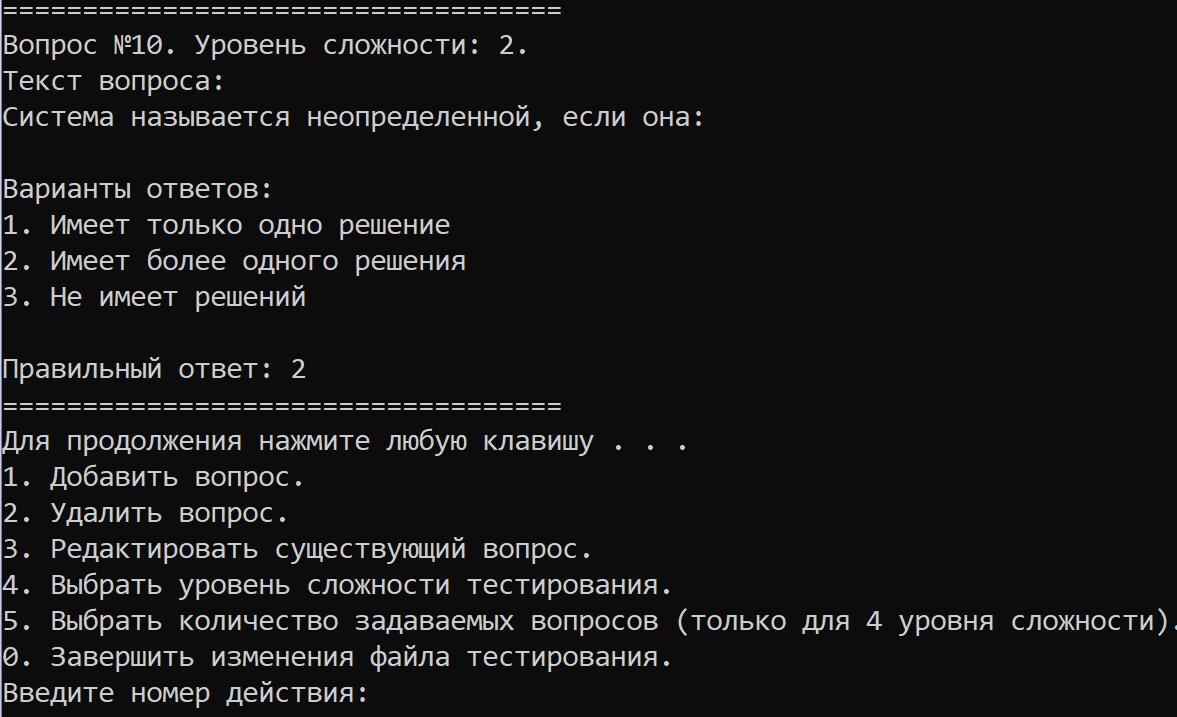


Рисунок 2.28 – Меню изменения тестирования

При выборе пункта 1 в меню изменения тестирования (см. рис. 2.28) будет предоставлена возможность ввода текста нового вопроса, деструкторов для него, его уровня сложности и правильного ответа для него (рис. 2.29).

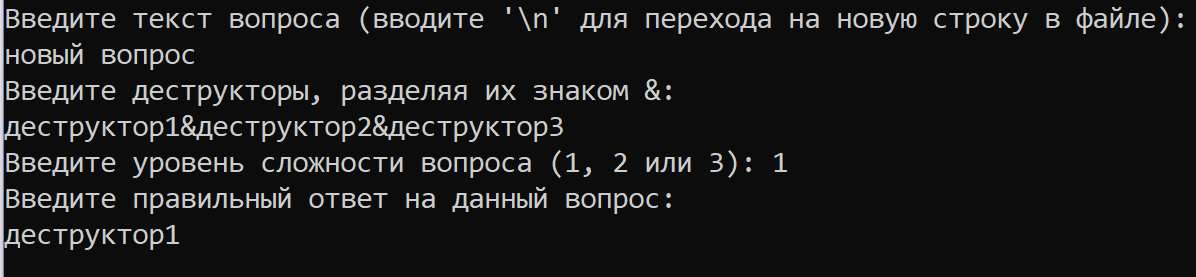


Рисунок 2.29 – Добавление нового вопроса

После добавления появится уведомление об обновлении файла тестирования и администратору будет предложено просмотреть измененный файл (рис. 2.30).

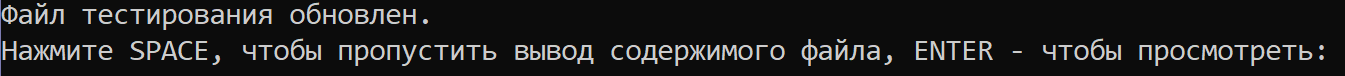


Рисунок 2.30 – Уведомление об обновлении файла тестирования

Если администратор нажмет ENTER, то будет выведены обновленный файл тестирования и меню изменения тестирования (см. рис. 2.28). Новый вопрос будет добавлен в конец файла.

При выборе пункта 2 в меню изменения тестирования (см. рис. 2.28) будет предоставлена возможность ввода номера удаляемого вопроса (рис. 2.31).



Рисунок 2.31 – Ввод номера удаляемого вопроса

При выборе пункта 3 в меню изменения тестирования (см. рис. 2.28) будет предоставлена возможность ввода номера редактируемого вопроса и номера действия для редактирования вопроса (рис. 2.32).

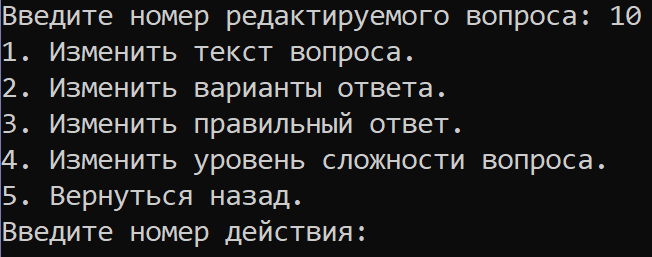


Рисунок 2.32 – Выбор редактируемого вопроса и действия для редактирования

При выборе пункта 4 в меню изменения тестирования (см. рис. 2.28) будет предоставлена возможность ввода уровня тестирования (рис. 2.33).



Рисунок 2.33 – Выбор уровня тестирования

При выборе пункта 5 в меню изменения тестирования (см. рис. 2.28) будет выведено текущее количество задаваемых вопросов на 4 уровне сложности и предоставлена возможность изменения этого количества (рис. 2.34).

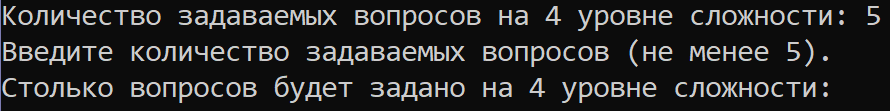


Рисунок 2.34 – Изменение количества задаваемых вопросов на 4 уровне сложности

## 2.6 Сообщения системы

В таблице 2.6 приведены сообщения системы.

Таблица 2.6 – Сообщения системы

|  |  |  |
| --- | --- | --- |
| **№ п\п** | **Сообщение** | **Причина возникновения,  способ устранения** |
| 1 | Показ теории завершен. | Все страницы теоретического материала просмотрены. |
| 2 | Нельзя пролистнуть назад, так как это первая страница. | Попытка перейти на страницу, предыдущую первой страницы |
| 3 | Вы ответили правильно. | Пользователь ответил правильно на вопрос в тесте. |
| 4 | Вы ответили неправильно. | Пользователь допустил ошибку при ответе на вопрос в тесте. |
| 5 | Количество правильных ответов меньше неправильных. Тестирование прошло неуспешно. | Тестирование завершено. Количество правильных ответов меньше 3. |
| 6 | Количество правильных ответов не меньше 3. Тестирование прошло успешно. | Тестирование завершено. Количество правильных ответов не меньше 3. |

В случае появления других сообщений следует обратиться к разработчику. **3 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ**

## 3.1 Проверка работоспособности показа теории

1. Запустить программу на выполнение. Появится меню (см. рис. 2.2).
2. Выбрать пункт 1, убедиться, что выведена первая страница теоретического   
   материала (см. рис.2.3).
3. Нажать стрелку влево, убедиться, что получено сообщение 2 (см. табл. 2.6).
4. Нажать стрелку вправо, убедиться, что выведена вторая глава теории (см. рис. 2.5).
5. Нажать стрелку влево, убедиться, что выведена первая глава теоретического   
   материала (см. рис. 2.3).
6. Нажать стрелку вправо пять раз, убедиться, что получено сообщение 1 (см. табл. 2.6).

## 3.2 Проверка работоспособности демонстрации работы метода Гаусса

1. Запустить программу на выполнение, выбрать в меню пункт 2 и убедиться, что предложено ввести номер способа заполнения матрицы (см. рис. 2.7).
2. Нажать 1, убедиться, что предложено ввести количество уравнений (см. рис. 2.8).
3. Нажать 2, убедиться, что предложено ввести элементы матрицы (см. рис. 2.8).
4. Заполнить матрицу по рис. 2.8.
5. Убедиться, что по окончании работы метода Гаусса показано решение введенной системы линейных уравнений (см. рис. 2.15).

## 3.3 Проверка работоспособности тестирования

1. Запустить программу на выполнение, выбрать в меню пункт 3 и убедиться, что выведена инструкция по тестированию (см. рис. 2.19).
2. Ввести правильный ответ, убедиться, что получено сообщение 3 (см. табл. 2.6).
3. Ввести 0 и убедиться, что выведен вопрос с вариантами ответов и предлагается ввести ответ на вопрос.
4. Ввести правильные ответы на остальные вопросы, убедиться, что получено сообщение 6 (см. табл. 2.6) и выводится итоговый результат (см. рис. 2.22).
5. Запустить программу на выполнение, выбрать в меню пункт 3 и убедиться, что выведена инструкция по тестированию (см. рис. 2.19).
6. Ответить на все вопросы неправильно, убедиться, что получено сообщение 5 (см. табл. 2.6) и выводится итоговый результат.
7. Запустить программу на выполнение, выбрать в меню пункт 3 и убедиться, что выведена инструкция по тестированию (см. рис. 2.19).
8. Выбирая из вариантов ответа, вести 0 в качестве ответа на все вопросы, убедиться, что получено сообщение 5 (см. табл. 2.6) и выводится итоговый результат (см. рис. 2.22).

## 3.4 Проверка работоспособности администрирования файла с теорией

1. Запустить программу на выполнение, выбрать в меню пункт 4 и убедиться, что предложено ввести пароль (см. рис. 2.23).
2. Ввести пароль, отличный от “admin123”, убедиться, что выведено уведомление о том, что осталась одна попытка на ввод правильного пароля.
3. Ввести пароль, отличный от “admin123”, убедиться, что выведено уведомление о том, что введенный пароль неверный и выведено меню (см. рис. 2.2).
4. Выбрать в меню пункт 4 и убедиться, что предложено ввести пароль (см. рис. 2.23).
5. Ввести пароль “admin123”, убедиться, что выведено меню администратора (см. рис. 2.23).
6. Нажать 1, убедиться, что выведено меню изменения теории (см. рис. 2.24).
7. Нажать 1, убедиться, что предложено выбрать главу для изменения (см. рис. 2.25).
8. Нажать стрелку вправо и нажать ENTER, убедиться, что появилось меню для изменения главы (см. рис. 2.26).
9. Выбрать пункт 1 в меню, убедиться, что предлагается выбрать начальную и конечную строки для изменения (см. рис. 2.27).
10. Ввести 1 для начальной и конечной строк для изменения, убедиться, что предложено ввести измененную строку для 1 строки.
11. Ввести “тест” в качестве измененной строки, убедиться, что выведена измененная глава с измененным заголовком и выведено меню администратора (см. рис. 2.23)
12. Выйти из программы, найти файл theory.txt, убедиться, что время последнего изменения схоже с временем изменения файла в программе.
13. Открыть файл theory.txt, убедиться, что заголовок второй главы называется “тест”.
14. Запустить программу на выполнение, выбрать в меню пункт 4 и убедиться, что предложено ввести пароль (см. рис. 2.23).
15. Выбрать в меню пункт 4 и убедиться, что предложено ввести пароль (см. рис. 2.23).
16. Ввести пароль “admin123”, убедиться, что выведено меню администратора (см. рис. 2.23).
17. Нажать 1, убедиться, что выведено меню изменения теории (см. рис. 2.24).
18. Нажать 1, убедиться, что предложено выбрать главу для изменения (см. рис. 2.25).
19. Нажать стрелку вправо и нажать ENTER, убедиться, что предложено ввести заголовок главы.
20. Ввести “заголовок”, убедиться, что выведено предложение завершить изменение новой главы, нажать 5.
21. Убедиться, что предлагается ввести новую строку для добавления, ввести “текст”.
22. Убедиться, что выведено предложение завершить изменение новой главы, нажать 0.
23. Убедиться, что выведена измененная глава с измененным заголовком и выведено меню администратора (см. рис. 2.23).
24. Выйти из программы, найти файл theory.txt, убедиться, что время последнего изменения схоже с временем изменения файла в программе.
25. Открыть файл theory.txt, убедиться, что заголовок второй главы называется “заголовок”, а текст 2 главы состоит из слова “текст”.

## 3.5 Проверка работоспособности администрирования файла с тестами

1. Запустить программу на выполнение, выбрать в меню пункт 4 и убедиться, что предложено ввести пароль (см. рис. 2.23).
2. Ввести пароль “admin123”, убедиться, что выведено меню администратора (см. рис. 2.23).
3. Нажать 2, убедиться, что выведено меню изменения тестирования (см. рис. 2.28).
4. Нажать 1, убедиться, что предложено ввести текст вопроса (см. рис. 2.29).
5. Ввести текст вопроса, убедиться, что предложено ввести деструкторы (см. рис. 2.29).
6. Ввести 1 деструктор, убедиться, что выведено предупреждение о том, что должно быть минимум 2 деструктора для вопроса.
7. Ввести 2 деструктора, убедиться, что предложено ввести уровень сложности вопроса (см. рис. 2.29).
8. Ввести 0, убедиться, что выведено предупреждение о том, что введенный уровень сложности некорректен.
9. Ввести 1, убедиться, что предложено ввести правильны ответ на вопрос.
10. Ввести правильный ответ на вопрос, убедиться, что выведено уведомлении об обновлении файла тестирования (см. рис. 2.30).
11. Выйти из программы, открыть файл test.txt, убедиться, что в конце файла добавлен новый вопрос.
12. Запустить программу на выполнение, выбрать в меню пункт 4 и убедиться, что предложено ввести пароль (см. рис. 2.23).
13. Ввести пароль “admin123”, убедиться, что выведено меню администратора (см. рис. 2.23).
14. Нажать 2, убедиться, что выведено меню изменения тестирования (см. рис. 2.28).
15. Нажать 2, убедиться, что предложено ввести номер вопроса для удаления.
16. Ввести номер последнего вопроса, убедиться, что выведено уведомлении об обновлении файла тестирования (см. рис. 2.30).
17. Выйти из программы, открыть файл test.txt, убедиться, что недавно добавленный вопрос удален.

# ЗАКЛЮЧЕНИЕ

В результате творческого проектирования разработана программа-тренажёр «Системы уравнений с двумя неизвестными». Тренажёр предлагает задания на умение решать системы с уравнениями второй степени с двумя неизвестными и проверяет правильность ответа, в случае ошибки сообщает правильный ответ. Тренировка продолжается до получения серии из пяти правильных либо трёх неправильных ответов, по окончании выдаётся сообщение о том, насколько успешно были выполнены задания.

Программа отвечает поставленным требованиям и может быть использована для обучения школьников старших классов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Булычева Ю.А., Васильева Т.В. Учебное пособие “Алгебра (часть 2)” для студентов специальности 090105.65 “Комплексное обеспечение информационной безопасности автоматизированных систем” – АГТУ, Астрахань 2019. – 88 с.
2. Прата С. Язык программирования С++. Лекции и упражнения, 6-е изд. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2012. – 1248 с. : ил.

# ПРИЛОЖЕНИЕ 1

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

**на разработку учебно-демонстрационной программы  
«Решение систем линейных уравнений методом Гаусса»**

по дисциплине «Программирование и информатика**»**

Направление 09.03.04 – Программная инженерия

Исполнитель: обучающийся гр. ДИПРб11 **Лиджигоряев В.А.**

1. **Назначение, цели и задачи разработки**

**Цель разработки –** автоматизация обучения решению систем линейных уравнений методом Гаусса.

**Назначение разработки:**

* + повышение качества знаний пользователей;
  + снижение нагрузки на преподавателя.

**Основные задачи,** **решаемые разработчиком в процессе выполнения творческого проекта**:

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

1. **Характер разработки:** прикладная квалификационная работа.
2. **Основания для разработки**
   * Учебный план направления 09.03.04 «Программная инженерия» 2020 года набора.
   * Рабочая программа дисциплины Программирование и информатика».
   * Распоряжение по кафедре АСОИУ №\_\_\_\_ от «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.
3. **Плановые сроки выполнения –** весенний семестр 2020/21 учебного года:

Начало «15» февраля 2021 г.

Окончание «25» мая 2021 г.

1. **Требования к проектируемой системе**

**5.1 Требования к функциональным характеристикам**

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

* + Предоставление пользователю теоретического материала по теме «Решение системы линейных уравнений методом Гаусса»
    - текст разбит на страницы не менее 10 и не более 20 строк по 60 символов в строке, перенос осуществляется по словам, выравнивание по ширине;
    - программа должна обеспечивать пролистывание как в прямом, так и в обратном направлении;
    - текст с теоретическим материалом разбит на абзацы и структурирован;
    - текст хранится в текстовом файле.
  + Визуализация решения системы линейных уравнений методом Гаусса:
    - предоставление пользователю ввода количества неизвестных в системе и коэффициентов при неизвестных в системе;
    - количество неизвестных – целое число в диапазоне [2; 6], коэффициенты при неизвестных и свободные коэффициенты – вещественные числа в диапазоне   
      [-50; 50];
    - система является квадратной, количество строк основной матрицы равно количеству столбцов и равно количеству неизвестных, столбец свободных членов состоит из элементов, количество которых равно количеству неизвестных, таким образом максимально возможный размер основной матрицы – 6х6, минимально возможный – 2х2;
    - получившая система отображается на экране, должны быть показаны коэффициенты при неизвестных, сами неизвестные, столбец свободных членов;
    - пошаговая демонстрация работы метода Гаусса для решения данной системы линейных уравнений с пояснением порядка выполняемых действий по изменению изначальной матрицы (с помощью элементарных преобразований), исследованию системы линейных уравнений, получению решений системы;
    - Решение выводится на экран только в том случае, если оно единственно.
  + Тестирование пользователя по теме «Решение системы линейных уравнений методом Гаусса»:
    - вопросы закрытого типа с выбором альтернатив, количество предлагаемых вариантов ответа не менее 2 и не более 4, правильный ответ не менее 1 и не более 3, при показе пользователю вопроса альтернативы должны быть перемешаны;
    - при тестировании пользователю предлагается 5 вопросов, общее количество вопросов в базе не менее 10, база представляет собой текстовый файл;
    - вопросы разрабатываются исполнителем самостоятельно;
    - перед тестированием пользователю должна быть предоставлена инструкция о проведении теста;
    - программа должна подсчитывать количество правильных и неправильных ответов; пользователь вводит номер (номера) выбранных им альтернатив с клавиатуры через пробел в произвольном порядке; ответ считается верным, если выбраны все альтернативы, отмеченные как правильные, и не выбрана ни одна, отмеченная как ошибочная;
    - если ответ правильный, то к количеству правильных ответов добавляется 1 балл, в противном случае 1 балл добавляется к количеству неправильных;
    - пользователь может пропустить вопрос посредством ввода значения 0, вернуться к пропущенному вопросу нельзя;
    - после ввода ответа должно быть выведено сообщение о текущем результате, по завершении тестирования – общий итог;
    - тестирование считается успешным, если количество правильных ответов не менее 3.
  + **Интерфейс программы:** текст русский, шрифт кириллический, заголовки, термины и другая важная информация выделены цветом.

Система должна предусматривать:

* + количество переменных должно быть целым числом из диапазона [2; 6];
  + коэффициенты при неизвестных и свободные коэффициенты должны быть  
    вещественными числами из диапазона [-50; 50];
  + программа не должна обеспечивать редактирование теоретического материала и базы вопросов.

1. **Литература**
2. Булычева Ю.А., Васильева Т.В. Учебное пособие “Алгебра (часть 2)” для студентов специальности 090105.65 «Комплексное обеспечение информационной безопасности автоматизированных систем» - АГТУ, Астрахань 2019. – 88 с.
3. Прата С. Язык программирования С++. Лекции и упражнения, 6-е изд. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2012. – 1248 с. : ил.

# ПРИЛОЖЕНИЕ 2

# База вопросов

1. **Если**(x0;y0) –**решение системы линейных уравнений**

**, тогда**x0+ y0  **равно…**

1. **5,5**
2. **–1,5**
3. **–5,5**
4. **1,5**

  Правильный ответ:   d)

1. **Метод Гаусса для решения систем линейных уравнений заключается ...**
2. в нахождении обратной матрицы
3. в последовательном исключении переменных
4. в последовательном исключении свободных членов
5. в вычислении вспомогательных определителей системы

Правильный ответ:   c)

1. **Система**
2. Имеет только одно решение
3. Имеет бесконечное множество решений
4. Не имеет решений

Правильный ответ: c)

1. **Если**(;) –**решение системы линейных уравнений**

***,* тогда =**

1. 0
2. 1
3. 3
4. 7

Правильный ответ:   b)

1. Система уравнений называется совместной, если она:
2. Не имеет решения
3. Имеет только одно решение
4. Имеет хотя бы одно решение
5. Имеет бесконечное множество решений

Правильный ответ:   с)

1. Система уравнений называется определенной, если она:
2. Не имеет решения
3. Имеет хотя бы одно решение
4. Имеет только одно решение
5. Имеет бесконечное количество решений

Правильный ответ:   с)

1. Справедливо утверждение:
2. Если система однородная, то она совместна
3. Если система совместна, то она однородная
4. Если в системе все свободные члены не равны нулю, то она однородная
5. Если система однородная, то она несовместная

Правильный ответ: a)

1. Система линейных уравнений имеет единственное решение тогда и только тогда, когда
2. Ранг расширенной матрицы не равен рангу основной матрицы
3. Ранг основной матрицы больше ранга расширенной матрицы
4. Ранг основной матрицы равен рангу расширенной матрицы и равен количеству неизвестных

Правильный ответ: с)

1. Выберите ответ, который нужно поставить на место пропуска.

Квадратная система линейных уравнений является определенной тогда и только тогда, когда определитель системы, составленный из коэффициентов при неизвестных \_\_\_ (рассматривается над полем вещественных чисел).

1. =0
2. ≠0
3. >0
4. <0
5. =1

Правильный ответ: b)

1. Система называется неопределенной, если она:
2. Имеет только одно решение
3. Имеет более одного решения
4. Не имеет решений

Правильный ответ: b)