

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**  
**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**  
**Кафедра вычислительных технологий**

**ОТЧЁТ О ПРОХОЖДЕНИИ УЧЕБНОЙ ПРАКТИКИ**  
**Б2.В.01.01 (У) научно-исследовательская работа (получение первичных**  
**навыков научно-исследовательской работы)**

по направлению подготовки  
02.03.02 Фундаментальная информатика и информационные технологии

период с 06.07.2022 г. по 19.07.2022 г.

Выполнил:  
студент 16 группы ОФО

\_\_\_\_\_  
(подпись)      Эзри А. А.  
(Ф.И.О. студента)

Руководитель практики:

\_\_\_\_\_  
к.ф.-м.н., доцент  
(ученое звание, должность)      \_\_\_\_\_  
(подпись)      Лапина О.Н.  
(Ф.И.О)

Оценка по итогам защиты практики: \_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_ 2022 г.

Краснодар  
2022

ФГБОУ ВО «КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
Факультет компьютерных технологий и прикладной математики  
Кафедра вычислительных технологий

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ,  
ВЫПОЛНЯЕМОЕ В ПЕРИОД ПРОВЕДЕНИЯ  
УЧЕБНОЙ ПРАКТИКИ  
Б2.В.01.01 (У) НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА**

Студент Эзри Артём Александрович  
(фамилия, имя, отчество полностью)

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Место прохождения практики: ФГБОУ ВО «Кубанский государственный университет»

Срок прохождения практики с 06.07.2022 г. по 19.07.2022 г.

Цель практики – закрепление полученных теоретических знаний по дисциплинам направления и специальным дисциплинам программы бакалавриата Фундаментальная информатика информационные технологии, овладение необходимыми профессиональными компетенциями по избранному направлению подготовки.

Формирование компетенций, регламентируемых ФГОС ВО:

Код компетенции	Содержание компетенции (или её части)
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач
УК-2	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений
УК-3	Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде
УК-4	Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах)
УК-6	Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни
ОПК-1	Способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности
ОПК-2	Способен применять компьютерные/суперкомпьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности
ПК-1	Способен понимать и применять в научно-исследовательской и прикладной деятельности современный математический аппарат, основные законы естествознания, современные языки программирования и программное обеспечение; операционные системы и сетевые технологии
ПК-2	Способен проводить под научным руководством локальные исследования на основе существующих методов в конкретной области профессиональной деятельности

**Перечень вопросов (заданий, поручений) для прохождения практики:**

1. Дана квадратная матрица  $A$  размерности  $n \times n$  и номер столбца  $k$ .  
Вычислить её определитель методом разложения по  $k$  столбцу.
2. Для выбранного Вами набора из 10 непрерывных функций построить приложение, позволяющее вычислить определённый интеграл на отрезке  $[a; b]$  с шагом  $h$  методом срединных прямоугольников. Пользователь выбирает конкретную функцию и вводит значения  $[a; b]$ ,  $h$ .
3. Дан список ДДФ в векторной форме, построить и вывести АНФ для каждой из функций.
4. Реализовать программный продукт, позволяющий хранить и отображать информацию о пользователях. При запуске программного обеспечения вся информация о пользователях считывается из файла, пользователь может выбрать одну из следующих альтернатив:
  - a. Посмотреть список пользователей
  - b. Добавить пользователя
  - c. Удалить пользователя
    - i. По фамилии-имени
    - ii. По логину
    - iii. По номеру телефона
  - d. Изменить пользователя
    - i. По фамилии-имени
    - ii. По логину
    - iii. По номеру телефона
  - e. Сохранить изменения в файл
  - f. Отправить сообщение на e-mail пользователя
    - i. По фамилии-имени
    - ii. По логину
    - iii. По номеру телефона
  - g. Отсортировать по выбранному полю
  - h. Выход

### План-график выполнения работ:

№	Этапы работы (виды деятельности) при прохождении практики	Сроки	Отметка руководителя практики о выполнении (подпись)
1	Получение задания на практику. Инструктаж по технике безопасности. <sup>1</sup>	06.07.2022	
2	Решение первой задачи летней практики на основе курса «Алгебра»	07.07.2022 – 08.07.2022	
3	Решение второй задачи летней практики на основе курсов «Дифференциальное исчисление» и «Интегральное исчисление»	09.07.2022 – 10.07.2022	
4	Решение третьей задачи летней практики на основе курса «Дискретная математика»	11.07.2022 – 12.07.2022	
5	Решение четвертой задачи летней практики на основе курса «Методы программирования»	13.07.2022 – 16.07.2022	
6	Оформление результатов выполненных заданий и их согласование с руководителем (составление отчета о прохождении производственной практики)	17.07.2022 – 18.07.2022	
7	Защита отчета	19.07.2022	

Ознакомлен

(подпись студента)

(расшифровка подписи)

« 06 » июля 2022 г.

Руководитель практики

(подпись)

Лапина О.Н.

(Ф.И.О. руководителя)

<sup>1</sup> Инструктаж по ознакомлению с требованиями охраны труда, техники безопасности, а также правилами внутреннего трудового распорядка.

**ОЦЕНОЧНЫЙ ЛИСТ**  
**результатов прохождения**  
**УЧЕБНОЙ ПРАКТИКИ**  
**(НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА)**  
**по направлению подготовки**  
**02.03.02 Фундаментальная информатика и информационные технологии**

Фамилия И.О. студента Эзри А. А.

Курс 1

№	ОБЩАЯ ОЦЕНКА (отмечается руководителем практики)	Оценка			
		5	4	3	2
1.	Уровень подготовленности студента к прохождению практики				
2.	Умение правильно определять и эффективно решать основные задачи				
3.	Степень самостоятельности при выполнении задания по практике				
4.	Оценка трудовой дисциплины				
5.	Соответствие программе практики работ, выполняемых студентом в ходе прохождения практики				

№	СФОРМИРОВАННЫЕ В РЕЗУЛЬТАТЕ УЧЕБНОЙ ПРАКТИКИ КОМПЕТЕНЦИИ (отмечается руководителем практики от университета)	Оценка			
		5	4	3	2
1.	УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач				
2.	УК-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений				
3.	УК-3 Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде				
4.	УК-4 Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах)				
5.	УК-6 Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни				
6.	ОПК-1 Способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности				
7.	ОПК-2 Способен применять компьютерные/суперкомпьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности				
8.	ПК-1 Способен понимать и применять в научно-исследовательской и прикладной деятельности современный математический аппарат, основные законы естествознания, современные языки программирования и				

	программное обеспечение; операционные системы и сетевые технологии				
9.	ПК-2 Способен проводить под научным руководством локальные исследования на основе существующих методов в конкретной области профессиональной деятельности				

Руководитель практики \_\_\_\_\_ Лапина О.Н.  
(подпись) (расшифровка подписи)

## СВЕДЕНИЯ

о прохождении инструктажа по ознакомлению с требованиями охраны труда, технике безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка

Предприятие Федеральное государственное бюджетное образовательное учреждение высшего образования «Кубанский государственный университет»  
Факультет компьютерных технологий и прикладной математики  
Кафедра вычислительных технологий

Студент Эзри Артём Александрович, 18 лет  
(ФИО, возраст)

Дата 06 июля 2022 г.

### 1. Инструктаж по требованиям охраны труда

Провел канд. физ.-мат. наук, доцент Лапина О.Н.  
(должность, ФИО сотрудника, проводившего инструктаж, подпись)

Прослушал Эзри А. А.  
(ФИО, подпись студента)

### 2. Инструктаж по технике безопасности

Провел канд. физ.-мат. наук, доцент Лапина О.Н.  
(должность, ФИО сотрудника, проводившего инструктаж, подпись)

Прослушал Эзри А. А.  
(ФИО, подпись студента)

### 3. Инструктаж по пожарной безопасности

Провел канд. физ.-мат. наук, доцент Лапина О.Н..  
(должность, ФИО сотрудника, проводившего инструктаж, подпись)

Прослушал Эзри А. А.  
(ФИО, подпись студента)

### 4. Инструктаж по правилам внутреннего трудового распорядка

Провел канд. физ.-мат. наук, доцент Лапина О.Н.  
(должность, ФИО сотрудника, проводившего инструктаж, подпись)

Прослушал Эзри А. А.  
(ФИО, подпись студента)

**ОТЗЫВ**  
**руководителя о прохождении учебной практики**  
**(НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА)**

Студент \_\_\_\_\_ Эзри Артём Александрович  
(фамилия, имя, отчество полностью)

Направление подготовки 02.03.02. Фундаментальная информатика и информационные технологии

За время прохождения учебной практики (НИР) студент Эзри Артём проявил себя как способный и самостоятельный студент.

Студентом были решены следующие задачи и получены результаты:

1. Изучены возможности языков программирования C++ и Python, позволяющие решать поставленные задачи.
2. Реализованы следующие алгоритмы: вычисление определителя матрицы методом разложения по столбцу, вычисление определённого интеграла методом серединных прямоугольников, построение полинома Жегалкина двоичной дискретной функции по её вектору.
3. Изучена информация о способах нахождения алгебраического дополнения матрицы, о способах оптимизации алгоритмов путём сохранения результатов выполнения функций для предотвращения повторных вычислений, о способах приближительного вычисления определённых интегралов, о создании графических интерфейсов.
4. Реализована БД для хранения информации о пользователях.
5. Разработаны приложения для вычисления определителя матрицы и определённого интеграла для определённых функций, для построения полинома Жегалкина для нескольких ДДФ и для управления базой данных пользователей.

Все поставленные задачи выполнены, степень проработанности материала считаю достаточной.

Результаты практики оцениваю на оценку "\_\_\_\_\_".

Руководитель практики \_\_\_\_\_ Лапина О.Н.  
(подпись) (расшифровка подписи)



## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	10
Задача 1 .....	11
Задача 2 .....	18
Задача 3 .....	25
Задача 4 .....	34
ЗАКЛЮЧЕНИЕ .....	58
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	59
ПРИЛОЖЕНИЕ .....	60

## ВВЕДЕНИЕ

**Цель практики:** получение первичных навыков самостоятельного осуществления научно-исследовательской работы; приобретение опыта в реализации программных продуктов, основанных на знаниях, полученных за время обучения на первом курсе.

**Место прохождения практики:** Россия, Краснодарский край, г. Краснодар, ФГБОУ ВО «Кубанский государственный университет».

**Продолжительность практики:** 2 недели.

**Перечень основных работ:** разработать программу, вычисляющую определитель матрицы, программу, вычисляющую определённый интеграл некоторых функций с заданным шагом, программу позволяющую построить и вывести АНФ для каждой ДДФ из списка, заданной в векторной форме, приложение, позволяющее хранить и отображать информацию о пользователях из базы данных, а также производить некоторые операции с БД, такие как добавление, удаление, редактирование, сортировка пользователей, отправка сообщений на электронную почту (не по-настоящему) и сохранение изменений в файл.

## Задача 1

### 1. Математическая постановка задачи

**Решаемая задача:** Вычислить определитель квадратной матрицы произвольного порядка методом разложения по  $k$  столбцу.

**Определителем квадратной матрицы** называется алгебраическая сумма всевозможных произведений элементов этой матрицы, взятых по одному из каждой строки и по одному из каждого столбца. Сомножители в каждом слагаемом записываются в порядке следования строк, тогда номера столбцов образуют перестановки; слагаемые, соответствующие чётным перестановкам, берутся со знаком «плюс», соответствующие нечётным – со знаком «минус».

**Минором** порядка  $n - 1$  для данного определителя называется определитель матрицы, получающейся из матрицы исходного определителя посредством вычеркивания одной строки и одного столбца.

**Алгебраическим дополнением** элемента  $a_{ij}$  матрицы  $A$  называется минор, получающийся из исходной матрицы  $A$  путём вычёркивания  $i$ -й строки и  $j$ -го столбца, взятый со знаком «+», если сумма  $i + j$  чётная и «-», если нечётная. Другими словами, это число  $A_{ij} = (-1)^{i+j} M_{ij}$ .

### 2. Описание алгоритма решения

Чтобы вычислить определитель матрицы по столбцу  $k$ , нужно каждый элемент столбца  $k$  умножить на его алгебраическое дополнение  $M_{ik}$  и найти сумму всех полученных произведений. Эта сумма и будет являться определителем матрицы  $A$ .

$$\det(A) = \sum_{i=1}^n (-1)^{i+k} a_{ik} M_{ik}, \quad k = 1, 2, \dots, n$$

Чтобы получить алгебраическое дополнение элемента  $a_{ik}$ , нужно вычислить соответствующий ему минор, то есть определитель матрицы  $n-1$  порядка. Таким образом, получается рекурсивный алгоритм. Дно рекурсии –

матрица, состоящая из одного элемента, её определитель – значение этого элемента.

Асимптотическая сложность:  $O(n!)$ . В столбце всего  $n$  элементов, для каждого из них нужно обработать  $n-1$  элемент, для каждого из  $n-1$  элементов нужно обработать  $n-2$  элемента и так далее. Таким образом, время работы алгоритма в худшем случае равно  $O(n * (n - 1) * (n - 2) * ... * 2 * 1) = O(n!)$  (худший случай – это когда в матрице нет нулей, чем больше нулей – тем быстрее работает алгоритм, так как для нулевых элементов не нужно вычислять алгебраическое дополнение).

Время работы данного алгоритма оставляет желать лучшего, особенно для матриц размерности больше 10. В процессе работы программы вычисление некоторых миноров проводится многократно, и при этом программа каждый раз «спускается» до дна рекурсии. Однако, есть возможность существенно сократить время работы алгоритма, записывая уже вычисленные значения в память. Каждый раз перед вычислением очередного минора, программа проверяет, есть ли в словаре уже готовый результат (словарь – ассоциативный контейнер, представляющий собой множество пар вида «ключ–значение»). Если результат нашёлся, то не нужно продолжать ветку рекурсии. Если же результат не нашёлся – вычисляем его рекурсивно и записываем в словарь. Такой приём называется *кэшированием* или *мемоизацией*.

Недостатком такого подхода является повышенное использование памяти. Чтобы уменьшить занимаемое программой место в оперативной памяти компьютера, можно задать максимальный размер кэша, при достижении которого из него удаляются старые значения. Это своего рода компромисс между временной и пространственной эффективностью. В моей программе это ограничение не реализовано.

### **3. Техническое описание программного продукта**

Язык программирования: C++.

Работу программы можно описать так:

1. Создаётся пустая матрица – двумерный массив вещественных чисел с использованием структуры данных «вектор».
2. Вызывается функция *read\_matrix*, в которой из текстового файла построчно считывается матрица. Из каждой строки извлекаются значения, разделённые пробелом, обрабатываются функцией *float\_from\_string* и записываются в соответствующую ячейку матрицы. В этой функции также проводятся проверки на корректность входного файла и выводятся сообщения об ошибках.
3. На экран выводится матрица функцией *print\_matrix* и дальнейшие указания. Пользователь вводит номер столбца, по которому выполняется разложение и после этого вызывается функция *determinant*, возвращающая значение определителя.
4. Перед непосредственным вычислением создаётся статическая переменная *cache* типа *map* – ассоциативный контейнер, который запоминает результаты функции для всех использованных комбинаций входных данных.
5. Создаётся переменная *det* (определитель), равная нулю.
6. Берётся первый элемент в столбце (используется выбранный пользователем столбец, если он существует, иначе – первый столбец).
7. На этом шаге находится значение минора текущего элемента. Сначала проверяется кэш на наличие готового значения, и если оно не найдено, то минор вычисляется путём рекурсивного вызова функции *determinant* (шаги 6-8) от значения функции *submatrix* (функция, возвращающая матрицу без *del\_row* строки и *del\_col* столбца, на пересечении которых находится текущий элемент). Каждый вычисленный минор добавляется в *cache*. Если размер матрицы равен 1, функция *determinant* возвращает единственное значение этой матрицы.
8. К переменной *det* прибавляется произведение текущего элемента, взятого со знаком  $(-1)^{i+j}$ , и его минора.

9. Берётся следующий элемент столбца и для него выполняются шаги 7-8.  
Если больше элементов нет – функция *determinant* возвращает переменную *det*.
10. После возвращения в *main* на экран выводится значение определителя.

**Примечание.** В процессе вычисления определителя может произойти потеря точности из-за особенностей представления чисел с плавающей запятой в памяти компьютера. Это означает, что в некоторых случаях данной программой может быть получено приблизительное значение, то есть получение точного результата не гарантировано.

#### 4. Инструкция по эксплуатации

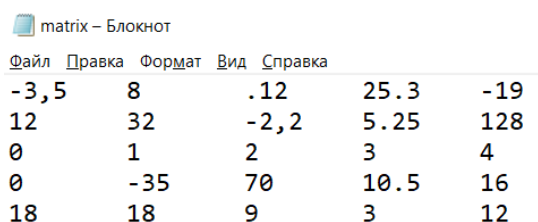
Данные на вход подаются из файла `matrix.txt`, который находится в той же директории, что и сама программа. При отсутствии входного файла он автоматически создаётся и на экран выводится полный путь к нему:

Файл 'C:\Users\ARTEZON\Desktop\Летняя практика\Летняя практика, Эзри Артём, 1 курс\Программы\Скомпилированные программы\Эзри Артём. Алгебра. Задача 10\matrix.txt' создан.  
Пожалуйста, заполните его матрицей, затем снова запустите программу.

Нажмите любую клавишу, чтобы выйти.

Рисунок 1 – Информация о созданном файле

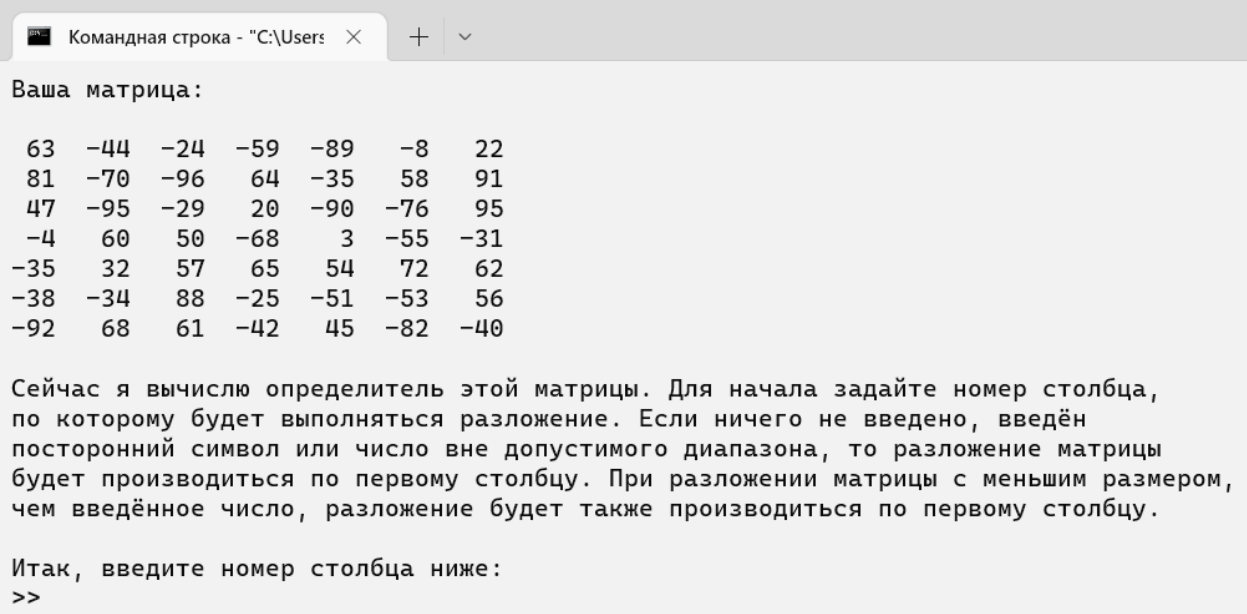
Сам файл должен содержать  $n$  строк, в каждой строке по  $n$  чисел, разделённые пробелами или знаками табуляции. Для разделения целой и дробной части используется точка или запятая. Если целая часть равна нулю, а дробная – нет, то целую часть можно опустить (см. рис. 2, третье число в первой строке).



-3,5	8	.12	25.3	-19
12	32	-2,2	5.25	128
0	1	2	3	4
0	-35	70	10.5	16
18	18	9	3	12

Рисунок 2 – Пример допустимого входного файла

После заполнения и сохранения файла, запустите программу снова. Если файл заполнен правильно, в окне консоли появится матрица и дальнейшие инструкции. Вам будет предложено ввести номер столбца, по которому нужно выполнить разложение:



```
Командная строка - "C:\Users\...
Ваша матрица:

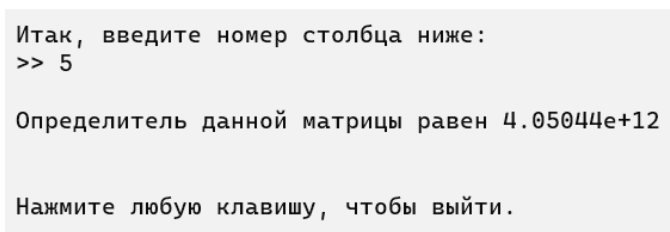
63  -44  -24  -59  -89  -8   22
81  -70  -96   64  -35   58   91
47  -95  -29   20  -90  -76   95
-4   60   50  -68   3  -55  -31
-35   32   57   65   54   72   62
-38  -34   88  -25  -51  -53   56
-92   68   61  -42   45  -82  -40

Сейчас я вычислю определитель этой матрицы. Для начала задайте номер столбца,
по которому будет выполняться разложение. Если ничего не введено, введён
посторонний символ или число вне допустимого диапазона, то разложение матрицы
будет производиться по первому столбцу. При разложении матрицы с меньшим размером,
чем введённое число, разложение будет также производиться по первому столбцу.

Итак, введите номер столбца ниже:
>>
```

Рисунок 3 – Ввод номера столбца

После нажатия Enter начнётся вычисление определителя. Результат будет выведен на экран:



```
Итак, введите номер столбца ниже:
>> 5

Определитель данной матрицы равен 4.05044e+12

Нажмите любую клавишу, чтобы выйти.
```

Рисунок 4 – Результат работы программы

В данном случае результат оказался округлённым:  $4,05044 \cdot 10^{12}$  (см. пункт 3, примечание).

## 5. Набор информации для тестирования

Таблица 1 – Примеры для тестирования программы по алгебре

	Содержимое файла (матрица из предм. обл.)	Результат работы программы
<b>1</b>	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	0
<b>2</b>	2 2 1 3 4 3 1 2 3 1 4 -1 2 4 -2 1 -1 1 1 2 4 -1 2 5 6	15
<b>3</b>	-1 -4 0 0 -2 0 1 1 5 4 3 1 7 1 0 0 0 2 0 -3 -1 0 4 2 2	996
<b>4</b>	0 0 0 2.5 3.3 12 -2 -4 -0.8 3 -15 10 1 2 3 4	33,25
<b>5</b>	-94 95 4 -16 48 85 -48 -32 33 12 38 8 69 0 0 0	-958272
<b>6</b>	-12.47 -0.59 8.96 -1.64 -4.29 8.97 -14.15 -9.57 -14.05 -3.77 5.35 13.66 5.07 0.05 12.44 -5.39	14857,7 (приблизительный результат)
<b>7</b>	0.2 7.2 1.8 2.7 6.5 2.8 8.8 4.0 6.0 0.6 6.4 1.6 7.8 5.0 0.6 4.9 5.4 1.3 3.9 3.0 5.6 6.5 4.6 6.2 3.9	1314,41 (приблизительный результат)
<b>8</b>	7 8 6 1 7 9 9 2 9 0 3 2 6 9 4 2 1 9 1 3 7 5 8 7 6 3 5 8 7 9 0 6 6 5 1 0 2 1 0 8 2 8 1 9 4 1 1 6 2 7 3 7 2 4 3 1 2 5 8 4 9 6 9 1 3 8 5 9 9 8 8 0 8 0 7 0 0 0 9 0 8 7 0 8 4 2 1 7 1 3 4 2 7 8 2 7 0 8 8 4	-5.14075e+08 (приблизительный результат)



Продолжение таблицы 1

<b>9</b>	49.4 58.7 -67.5 -92.6 39.6 -72.0 -74.8 11.6 -12.6 62.6 -79.7 -64.8 -81.5 -70.4 -96.9 12.7 12.6 -87.6 3.5 -52.6 -16.8 -67.0 78.4 -47.1 -95.6 -8.2 -20.0 -23.4 -91.8 17.1 -41.4 62.0 -64.2 65.7 -95.3 46.1 15.7 21.0 -39.4 -81.0 30.8 82.8 -5.3 23.9 -99.5 75.2 91.0 99.1 -13.9 50.6 81.0 -17.1 -63.3 57.7 -75.4 16.3 -31.1 30.8 -81.0 -66.6 16.1 -0.0 -80.8 -38.2 41.2 82.1 88.2 21.8 -6.1 88.2 -25.8 18.9 -90.4 -60.5 96.3 -27.6 75.3 36.5 -93.1 67.5 97.6	-1.16755e+19 (приблизительный результат)
<b>10</b>	2 0 -7 -3 -8 -4 6 5 -3 2 -7 -2 -6 -2 5 -2 -2 5 -1 -1 4 7 -4 -7 -9 -4 -7 -3 6 7 9 5 3 -7 -5 3 -9 -4 -4 1 8 5 -9 4 4 2 -1 -6 2 -2 -1 4 4 8 4 6 -4 1 -8 2 1 -5 1 -9 -1 2 -4 -3 1 -6 -4 6 6 -3 -7 1 -9 -1 9 0 -5 -4 -3 -9 1 -2 7 8 -2 -7 -8 -4 -2 -1 -2 8 -4 -3 -4 8 -1 -6 -5 -5 -8 8 -1 8 -8 2 3 1 9 0 -4 -3 -7 -4 2 3 -7 -5 -1 3 3 6 -4 8 6 8 1 -5 -1 -1 7 6 1 -2 -2 -8 -9 -6 -6 6 -3 -8 9 -9 8 -7 2 1 -3 7 6 3 2 -3 -1 -8 -7 -3 -1 -4 -8 -8 1 -1 2 -5 -7 6 -9 -5 -1 -6 9 -6 9 -4 -8 -9 -4 -8 9 9 0 4 2 -9 -6 9 -7 5 5 -6 -8 -8 -6 7 -9 9 -2 -4 9 -6 -4 -8 9 2 -4 2 -1 -1 1 3 -1 0 -5 -2 5 5 3 -5 -1	-2.44877e+16 (приблизительный результат)

## Задача 2

### 1. Математическая постановка задачи

**Решаемая задача:** вычислить определенный интеграл непрерывной функции на отрезке  $[a; b]$  с шагом  $h$  методом срединных прямоугольников.

#### Определённый интеграл.

Пусть функция  $f(x)$  определена на отрезке  $[a; b]$ . Разобьём этот отрезок на  $n$  произвольных частей точками:  $a = x_0 < x_1 < x_2 < \dots < x_n = b$ . Эти части называются *частичными отрезками*.

Обозначим их длины как  $\Delta x_1, \Delta x_2, \dots, \Delta x_n$ .

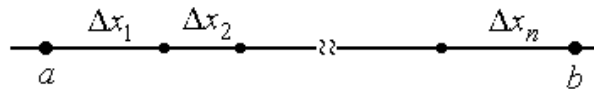


Рисунок 5 – Разбиение отрезка  $[a; b]$  на частичные отрезки

Внутри каждого промежутка выберем произвольную точку  $\xi_i \in [x_{i-1}; x_i]$  и составим *интегральную сумму*:

$$\sigma = \sum_{i=1}^n f(\xi_i) \Delta x_i$$

Обозначим через  $\lambda$  длину наибольшего частичного отрезка:

$$\lambda = \max (\Delta x_i)$$

**Определённым интегралом** от функции  $f(x)$  на отрезке  $[a; b]$  называется предел интегральной суммы при  $\lambda \rightarrow 0$ , если он существует независимо от разбиения на частичные отрезки и выбора точек  $\xi_i$ , то есть:

$$\int_a^b f(x) dx = \lim_{\lambda \rightarrow 0} \sum_{i=1}^n f(\xi_i) \Delta x_i$$

## 2. Описание алгоритма решения

Геометрический смысл определённого интеграла состоит в том, что определённый интеграл от неотрицательной функции численно равен площади фигуры, ограниченной осью абсцисс, прямыми  $x = a$  и  $x = b$  и графиком функции  $f(x)$ . То есть, интеграл равен сумме площадей прямоугольников бесконечно малой длины.

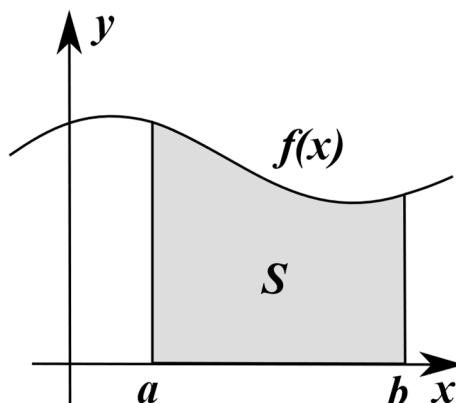


Рисунок 6 – Геометрический смысл определённого интеграла

**Метод прямоугольников** – метод численного интегрирования функции одной переменной, заключающийся в замене подынтегральной функции на многочлен нулевой степени, то есть константу, на каждом элементарном отрезке. Если рассмотреть график подынтегральной функции, то метод будет заключаться в приближённом вычислении площади под графиком суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирования, а высота — значением подынтегральной функции в этих узлах.

При вычислении интеграла **методом срединных прямоугольников** криволинейная трапеция заменяется прямоугольниками, высоты которых равны значению функции в **центрах** интервалов.

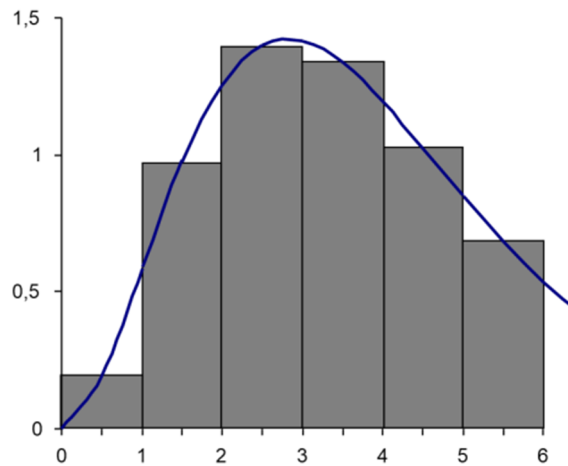


Рисунок 7 – Иллюстрация метода срединных прямоугольников

Длины оснований всех прямоугольников равны введённому пользователем шагу разбиения и вычисляются по формуле:  $h = \frac{b-a}{n}$ .

Следовательно, число прямоугольников равно  $n = \frac{b-a}{h}$ .

Формула вычисления интеграла методом срединных прямоугольников:

$$I = h * \sum_{i=1}^n f\left(a + \frac{(2i - 1) * h}{2}\right)$$

Асимптотическая сложность алгоритма равна  $O(n)$ , где  $n$  – число отрезков разбиения, так как для вычисления интеграла применяется один цикл, повторяющийся  $n$  раз и вычисляющий значения функции  $f(x)$  для середины каждого  $i$  прямоугольника. *Формула вычисления интеграла методом срединных прямоугольников*, которая находится выше, даёт наглядное представление об алгоритме.

### 3. Техническое описание программного продукта

Язык программирования: C++.

После запуска программы пользователь выбирает одну из десяти функций, вшитых в программу, и вводит номер этой функции, который записывается в переменную *func\_id* (типа *int*). Затем нужно ввести верхний и

нижний пределы интегрирования. Они сохраняются в переменных  $a$  и  $b$  соответственно (обе типа *long double*). Далее программа запрашивает желаемое значение шага интегрирования  $h$  и сохраняет его в переменной *step* типа *long double*. Чем шаг меньше, тем точнее результат, и тем дольше вычисление, т.к.  $n$  обратно пропорционально  $h$ . Оптимальным значением является 0,0001 (вводится как 0.0001).

После получения необходимой информации от пользователя, на экране появляется ASCII-рисунок интеграла для предпросмотра и вызывается функция *integral(func\_id, a, b, step)*.

В этой функции создаётся переменная  $n$  (число прямоугольников), а также переменные *sum* (сумма значений функции от середин прямоугольников, изначально равна нулю) и  $sign = 1$ . Если нижний предел интегрирования больше верхнего, то значение  $sign$  меняется на  $-1$ . Затем в цикле от 1 до  $n$  к *sum* прибавляется значение функции  $f(x, func\_id)$  от середины  $i$ -го прямоугольника.

```
for (int i = 1; i <= n; i++) {
    sum += f(a + ((2 * i - 1) * h / 2), func_id);
}
```

Для вычисления некоторых функций  $y = f(x)$  используются функции стандартной библиотеки *cmath*.

```
long double f(long double x, unsigned int func_id) {
    switch (func_id) {
        case 1: return powl(x, 2);
        case 2: return sqrtl(abs(x));
        case 3: return logl(abs(x));
        case 4: return expl(x);
        case 5: return sinl(x);
        case 6: return 2 * x * cosl(x);
        case 7: return 2 * x * x + 3 * x - 1;
        case 8: return (9 * x + 5) / (x - 8);
        case 9: return expl(3 * x) * sinl(x * x);
        case 10: return x / sqrtl(powl(x, 4) + 16);
        default: return 0;
    }
}
```

В итоге в переменную  $I$  в функции *main* возвращается вычисленное значение определённого интеграла, равное  $h * sum * sign$ . Перед выводом результата на экран он обрабатывается функцией *floating\_point*, которая возвращает строку, убирая незначащие нули. После завершения работы программа не закрывается, а возвращается в главное меню, именно для этого часть главной функции обернута в бесконечный цикл. Выйти из программы можно закрыв её окно или при помощи сочетания клавиш Ctrl+C.

#### 4. Инструкция по эксплуатации

После запуска программы пользователя встречает главное меню:

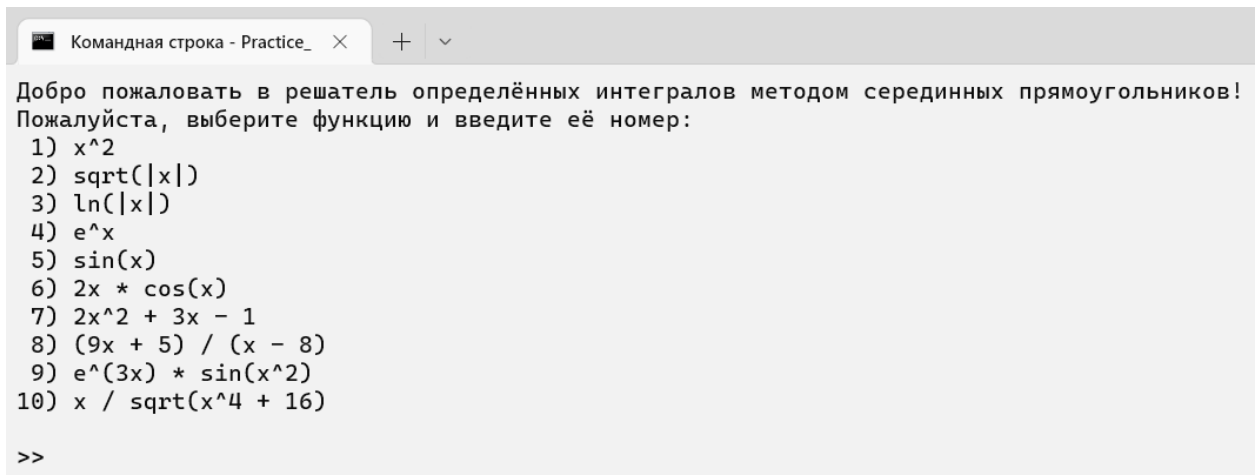


Рисунок 8 – Главное меню программы

В программе есть следующие функции:

1.  $x^2$
2.  $\sqrt{|x|}$
3.  $\ln |x|$
4.  $e^x$
5.  $\sin x$
6.  $2x * \cos x$
7.  $2x^2 + 3x - 1$
8.  $\frac{9x+5}{x-8}$
9.  $e^{3x} * \sin x^2$
10.  $\frac{x}{\sqrt{x^4+16}}$

Введите номер выбранной функции, затем пределы интегрирования, затем шаг интегрирования.

```
Добро пожаловать в решатель определённых интегралов методом срединных прямоугольников!
Пожалуйста, выберите функцию и введите её номер:
1) x^2
2) sqrt(|x|)
3) ln(|x|)
4) e^x
5) sin(x)
6) 2x * cos(x)
7) 2x^2 + 3x - 1
8) (9x + 5) / (x - 8)
9) e^(3x) * sin(x^2)
10) x / sqrt(x^4 + 16)

>> 10

Хорошо, теперь введите нижний и верхний пределы интегрирования через пробел.

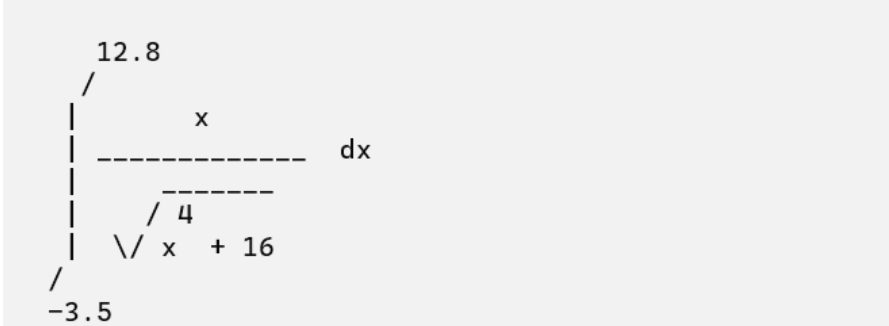
>> -3.5 12.8

Отлично. Осталось только ввести шаг интегрирования - ширину прямоугольников. Например, 0.0001

>> 0.00001
```

Рисунок 9 – Ввод данных

После нажатия Enter начнётся вычисление интеграла. Результат будет выведен на экран:



Интеграл приблизительно равен 1.283932

Нажмите любую клавишу, чтобы вернуться в главное меню.

Рисунок 10 – Результат работы программы

Чтобы посчитать ещё какой-нибудь интеграл, нажмите любую клавишу. Чтобы выйти, закройте окно программы.

## 5. Набор информации для тестирования

Таблица 2 – Примеры для тестирования программы по матанализу

	Входные данные	Предметная область	Результат
<b>1</b>	1 0 25 0.0001	$\int_0^{25} x^2$	5208.333333
<b>2</b>	2 -5 5 0.000001	$\int_{-5}^5 \sqrt{ x }$	14.90712
<b>3</b>	3 1 2.71828 0.00001	$\int_1^{2,71828} \ln  x $	0.999998
<b>4</b>	4 8 -2 0.0001	$\int_8^{-2} e^x$	-2980.822651
<b>5</b>	5 0 3.14 0.0001	$\int_0^{3,14} \sin x$	1.999999
<b>6</b>	6 -100 200 0.000001	$\int_{-100}^{200} 2x * \cos x$	-248.796053
<b>7</b>	7 3 5 0.0001	$\int_3^5 2x^2 + 3x - 1$	87.333333
<b>8</b>	8 -1 1 0.001	$\int_{-1}^1 \frac{9x + 5}{x - 8}$	-1.351191
<b>9</b>	9 5 10 0.0000001	$\int_5^{10} e^{3x} * \sin x^2$	-490054595720.179871
<b>10</b>	10 -3.5 12.8 0.00001	$\int_{-3,5}^{12,8} \frac{x}{\sqrt{x^4 + 16}}$	1.283932



### Задача 3

#### 1. Математическая постановка задачи

**Решаемая задача:** Дан список ДДФ в векторной форме, построить и вывести АНФ для каждой из функций.

**Двоичная дискретная функция (ДДФ или булева функция)** от  $n$  переменных – это отображение  $B^n \rightarrow B$ , где  $B = \{0, 1\}$  – булево множество.

**Таблица истинности** – это таблица, состоящая из двух частей: в левой части перечисляются все наборы значений аргументов (булевы векторы пространства  $B^n$ ) в естественном порядке, то есть по возрастанию значений чисел, представляемых этими векторами, а в правой части – значения булевой функции на соответствующих наборах.

Таблица 3 – Общий вид таблицы истинности

$x_1$	$x_2$	...	$x_{n-1}$	$x_n$	$f$
0	0	...	0	0	$f(0,0,...,0,0)$
0	0	...	0	1	$f(0,0,...,0,1)$
0	0	...	1	0	$f(0,0,...,1,0)$
0	0	...	1	1	$f(0,0,...,1,1)$
...	...	...	...	...	...
1	1	...	0	0	$f(1,1,...,0,0)$
1	1	...	0	1	$f(1,1,...,0,1)$
1	1	...	1	0	$f(1,1,...,1,0)$
1	1	...	1	1	$f(1,1,...,1,1)$

Так как наборы значений аргументов (строки левой части таблицы) записаны в лексикографическом порядке, то для того, чтобы задать функцию, достаточно выписать значения  $f(0,0,...,0,0)$ ,  $f(0,0,...,0,1)$ , ...,  $f(1,1,...,1,0)$ ,  $f(1,1,...,1,1)$ . Этот набор называют **вектором значений функции**.

**Алгебраическая нормальная форма (АНФ или полином Жегалкина)** – полином с коэффициентами 0 и 1, где в качестве произведения берётся *конъюнкция*, а в качестве сложения *исключающее или*. Полином Жегалкина имеет следующий вид:

$$P = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \dots \oplus a_{1\dots n} x_1 \dots x_n$$

С его помощью можно выразить любую булеву функцию, так как он строится из следующего набора функций:  $\langle \wedge, \oplus, 1 \rangle$ , который, в свою очередь, по теореме Поста является полным.

## 2. Описание алгоритма решения

Алгоритм построения полинома Жегалкина методом треугольника.

**Примечание.** На изображениях для наглядности представлен конкретный пример функции 3-х переменных. Данный алгоритм работает в общем случае для всех функций  $n$  переменных при  $n \geq 1$ .

**Шаг 1.** Строим таблицу истинности функции по её вектору (строки в таблице идут в порядке возрастания двоичных кодов).

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Рисунок 11 – Построение таблицы истинности

**Шаг 2.** Построение треугольника.

Для этого берём вектор значения функции и выписываем его напротив первой строки таблицы.

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

0	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

Рисунок 12 – Первая строка треугольника

Далее заполняем треугольник, складывая попарно соседние значения по модулю 2, результат сложения выписываем ниже. Продолжаем вычисления, пока в строке не останется лишь одна цифра.

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

0	0	0	1	0	1	1	1
0	0	1	1	1	0	0	
0	1	0	0	1	0		
1	1	0	1	1			
0	1	1	0				
1	0	1					
1	1						
0							

Рисунок 13 – Построенный треугольник

### Шаг 3. Построение полинома Жегалкина.

Нас интересует левая сторона треугольника:

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Рисунок 14 – Левая сторона треугольника

Числа на левой стороне треугольника есть коэффициенты полинома при монотонных конъюнкциях, соответствующих наборам значений переменных.

Выпишем эти конъюнкции. Конъюнкции выписываем по двоичным наборам в левой части таблицы по следующему принципу: если напротив переменной  $x_i$  стоит 1, то переменная входит в конъюнкцию; в противном случае переменная отсутствует в конъюнкции. Набору (0,0,0) соответствует константа 1.

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Рисунок 15 – Конъюнкции

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

	$x_1$	$x_2$	$x_3$	$f$
1	0	0	0	0
$x_3$	0	0	1	0
$x_2$	0	1	0	0
$x_2 x_3$	0	1	1	1
$x_1$	1	0	0	0
$x_1 x_3$	1	0	1	1
$x_1 x_2$	1	1	0	1
$x_1 x_2 x_3$	1	1	1	1

Это и есть конъюнкции, входящие в состав полинома Жегалкина. В приведённом выше примере полином равен  $x_1 + x_2 \oplus x_1 + x_3 \oplus x_2 + x_3$ .

Пусть  $n$  – число переменных функции. Алгоритм генерирует треугольник (сложность:  $O(2^{2n})$ ), вычисляет количество переменных ( $O(n)$ ), затем каждую из  $2^n$  строк таблицы истинности переводит в двоичную систему и проверяет соответствующие им конъюнкции ( $O(n)$ ). После упрощения итоговая асимптотическая сложность алгоритма равна  $O(2^n * n)$ .

Язык программирования: C++.

Сначала считывается входной текстовый файл и его содержимое проверяется на корректность в функции *read\_file*. Файл считается корректным, если состоит из одной или более строк, в каждой строке записан один вектор без пробелов, букв и знаков препинания, состоящий из  $n$  цифр (допустимы только 0 и 1), и при этом  $\log_2 n \in N$  (т.е.  $n$  является степенью 2). Массив функций в векторной форме (массив строк) записывается в переменную *func\_list* типа *vector<string>*.

Затем открывается выходной файл (переменная *out*), создаётся строка *polynomial* для текущего полинома и  $n$  – счётчик обработанных функций. В цикле каждая функция конвертируется в АНФ функцией *anf*, результат выводится на экран и добавляется в конец выходного файла, если тот был успешно открыт.

Описание остальных функций:

- *bool is\_pow\_2(int n)* – возвращает *true*, если число  $n$  является степенью двойки, иначе – *false*.
- *unsigned int log\_2(unsigned int n)* – возвращает логарифм числа по основанию 2.
- *string bin(int n, unsigned int len)* – возвращает строку, содержащую число  $n$  в двоичной системе счисления, добавляя незначащие нули, если длина строки меньше *len*.
- *vector<string> triangle(string func)* – возвращает массив (контейнер *vector*) строк треугольника, начиная с самой функции в векторной форме и заканчивая строкой из одной цифры.
- *string anf(string func)* – генерирует и возвращает АНФ в виде строки.

Сначала создаётся массив *trngl = triangle(func)*, строка *anf\_str* и переменная *var\_count = log\_2(trngl[0].size())* – количество переменных булевой функции. Затем в цикле обрабатывается каждая строка треугольника: переменной *left\_number* присваивается первая цифра слева, а строке *truth\_table\_row* – текущая строка таблицы истинности: *truth\_table\_row = bin(i, var\_count)*. Если *left\_number = 1*, то в АНФ

добавляется знак «+» (который в данном случае означает сложение по модулю два), и приведённый ниже код добавляет в АНФ конъюнкцию вида 1 или, например,  $x_1x_3$ :

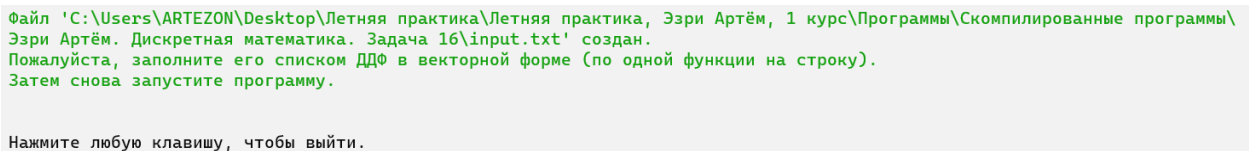
```
if (i == 0) anf_str += "1";
else {
    for (int j = 0; j < var_count; j++) {
        if (truth_table_row[j] == '1') {
            anf_str += "x" + to_string(j + 1);
        }
    }
}
```

После завершения цикла удаляется лишний знак «+» с начала строки.

Переменная *anf\_str* возвращается в функцию *main*.

#### 4. Инструкция по эксплуатации

Данные на вход подаются из файла *matrix.txt*, который находится в той же директории, что и сама программа. При отсутствии входного файла он автоматически создаётся и на экран выводится полный путь к нему:

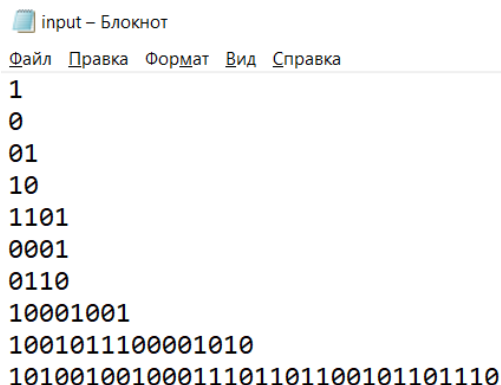


Файл 'C:\Users\ARTEZON\Desktop\Летняя практика\Летняя практика, Эзри Артём, 1 курс\Программы\Скомпилированные программы\Эзри Артём. Дискретная математика. Задача 16\input.txt' создан.  
Пожалуйста, заполните его списком ДДФ в векторной форме (по одной функции на строку).  
Затем снова запустите программу.

Нажмите любую клавишу, чтобы выйти.

Рисунок 17 – Информация о созданном файле

Сам файл должен состоять из одной или более строк, в каждой строке по одной функции в векторной форме без пробелов, букв и знаков препинания, состоящей из  $n$  цифр (0 или 1), где  $n$  является степенью числа 2.



input – Блокнот

Файл Правка Формат Вид Справка

1  
0  
01  
10  
1101  
0001  
0110  
10001001  
1001011100001010  
10100100100011101101100101101110

Рисунок 18 – Пример допустимого входного файла

После заполнения и сохранения файла, запустите программу снова. Если файл заполнен правильно, сразу начнётся вычисление всех функций и вывод результатов на экран:

```

Командная строка - Practise
Функция 1:
Введённый вектор: 1
Полином Мегаalkина: 1

Функция 2:
Введённый вектор: 0
Полином Мегаalkина: не существует

Функция 3:
Введённый вектор: 01
Полином Мегаalkина: x1

Функция 4:
Введённый вектор: 10
Полином Мегаalkина: 1 + x1

Функция 5:
Введённый вектор: 1101
Полином Мегаalkина: 1 + x1 + x1x2

Функция 6:
Введённый вектор: 0001
Полином Мегаalkина: x1x2

Функция 7:
Введённый вектор: 0110
Полином Мегаalkина: x2 + x1

Функция 8:
Введённый вектор: 10001001
Полином Мегаalkина: 1 + x3 + x2 + x2x3 + x1x2x3

Функция 9:
Введённый вектор: 1001011100001010
Полином Мегаalkина: 1 + x4 + x3 + x2 + x2x3x4 + x1 + x1x4 + x1x3 + x1x2x4 + x1x2x3x4

Функция 10:
Введённый вектор: 10100100100011101101100101101110
Полином Мегаalkина: 1 + x5 + x3 + x3x4x5 + x2x4 + x2x4x5 + x2x3 + x2x3x5 + x2x3x4 + x2x3x4x5 + x1x5 + x1x4 + x1x4x5 + x1x3 + x1x3x5 + x1x2 + x1x2x5 + x1x2x4 + x1x2x3x5 + x1x2x3x4x5

Результат(ы) записан(ы) в файл 'C:\Users\ARTEZON\Desktop\Летняя практика\Летняя практика, Эзри Артём, 1 курс\Программы\Скомпилированные программы\Эзри Артём. Дискретная математика. Задача 16\output.txt'.

Нажмите любую клавишу, чтобы выйти.

```

Рисунок 19 – Программа по дискретной математике

Программа создаст файл *output.txt* в той же директории и запишет в него функции в форме АНФ, причём каждая строка выходного файла соответствует строке из входного файла с тем же номером.

**Примечание.** Операция «исключающее или» представлена в программе знаком «+», т.к. символ  $\oplus$  в кодировке ASCII отсутствует. По той же причине подстрочные индексы выводятся обычными символами (x1 вместо  $x_1$ ).

## 5. Набор информации для тестирования

Таблица 4 – Примеры для тестирования программы по ДМ

	Содержимое файла (соответствует предметной области)	Результат (в предметной области вместо знака + используется $\oplus$ )
1	1	1
2	0	не существует
3	01	x1
4	10	1 + x1



Продолжение таблицы 4

<b>5</b>	1101	$1 + x_1 + x_1x_2$
<b>6</b>	0001	$x_1x_2$
<b>7</b>	0110	$x_2 + x_1$
<b>8</b>	10001001	$1 + x_3 + x_2 + x_2x_3 + x_1x_2x_3$
<b>9</b>	1001011100001010	$1 + x_4 + x_3 + x_2 + x_2x_3x_4 + x_1 + x_1x_4 + x_1x_3 + x_1x_2x_4 + x_1x_2x_3x_4$
<b>10</b>	10100100100011101101100101101110	$1 + x_5 + x_3 + x_3x_4x_5 + x_2x_4 + x_2x_4x_5 + x_2x_3 + x_2x_3x_5 + x_2x_3x_4 + x_2x_3x_4x_5 + x_1x_5 + x_1x_4 + x_1x_4x_5 + x_1x_3 + x_1x_3x_5 + x_1x_2 + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_3x_5 + x_1x_2x_3x_4x_5$

Все 10 примеров, представленные в таблице, можно записать в один файл и обработать все сразу, тогда входной и выходной файлы примут вид:

Таблица 5 – Примеры в одном файле

<b>input.txt</b>	<b>output.txt</b>
1	1
0	не существует
01	$x_1$
10	$1 + x_1$
1101	$1 + x_1 + x_1x_2$
0001	$x_1x_2$
0110	$x_2 + x_1$
10001001	$1 + x_3 + x_2 + x_2x_3 + x_1x_2x_3$
1001011100001010	$1 + x_4 + x_3 + x_2 + x_2x_3x_4 + x_1 + x_1x_4 + x_1x_3 + x_1x_2x_4 + x_1x_2x_3x_4$
10100100100011101101100101101110	$1 + x_5 + x_3 + x_3x_4x_5 + x_2x_4 + x_2x_4x_5 + x_2x_3 + x_2x_3x_5 + x_2x_3x_4 + x_2x_3x_4x_5 + x_1x_5 + x_1x_4 + x_1x_4x_5 + x_1x_3 + x_1x_3x_5 + x_1x_2 + x_1x_2x_5 + x_1x_2x_4 + x_1x_2x_3x_5 + x_1x_2x_3x_4x_5$

## Задача 4

### 1. Постановка задачи

**Решаемая задача:** Реализовать программный продукт, позволяющий хранить и отображать информацию о пользователях. При запуске программного обеспечения вся информация о пользователях считывается из файла, пользователь может выбрать одну из следующих альтернатив:

- a. Посмотреть список пользователей
- b. Добавить пользователя
- c. Удалить пользователя
  - i. По фамилии-имени
  - ii. По логину
  - iii. По номеру телефона
- d. Изменить пользователя
  - i. По фамилии-имени
  - ii. По логину
  - iii. По номеру телефона
- e. Сохранить изменения в файл
- f. Отправить сообщение на e-mail пользователя
  - i. По фамилии-имени
  - ii. По логину
  - iii. По номеру телефона
- g. Отсортировать по выбранному полю
- h. Выход

- 1) Для каждого пользователя хранится следующая информация: фамилия, имя, отчество, номер телефона, e-mail, логин, пароль.
- 2) Пароль должен храниться и отображаться в форме хэш-кода.
- 3) Фамилия, имя и отчество должны быть записаны с большой буквы.

- 4) Номер телефона и e-mail должны быть корректны.
- 5) При вводе информации необходимо проверять корректность ввода полей с помощью регулярных выражений. ФИО должны меняться в соответствии с правилом 3), если телефон записан корректно, то сохранять его в следующей форме: +7-(222)-222-22-22.
- 6) Пароль должен состоять из как минимум 8 символов, содержать как минимум одну заглавную букву, одну строчную, одну цифру, один специальный символ. Проверка осуществляется с помощью регулярных выражений.
- 7) ФИО не должны содержать цифр.
- 8) Если ФИО, телефон, e-mail или пароль введены неправильно, предлагается ввести их заново.
- 9) Для изменения или удаления записи необходимо найти пользователя по введенным полям, если он не найден, сообщить об этом.
- 10) Если пользователь найден, необходимо ввести его пароль и проверить на корректность.
- 11) После завершения одной операции программа не закрывается, а предлагает пользователю заново одну из описанных выше альтернатив.

## **2. Техническое описание программного продукта**

Язык программирования: Python 3.10.

Использованы следующие библиотеки:

- *os, sys* – системные функции, работа с файлами.
- *re* – работа с регулярными выражениями.
- *json* – чтение и запись файлов в формате JSON.
- *hashlib* – алгоритмы для хэширования, например, SHA256.
- *datetime* – получение даты и времени компьютера.
- *PyQt5* – библиотека Python для работы с фреймворком Qt, который позволяет организовать оконный графический интерфейс.

Все действия с данными в программе осуществляются через окно программы с графическим интерфейсом.

Пользовательский интерфейс был создан в программе Qt Creator, которая включает в себя редактор кода и визуальные средства разработки интерфейса.

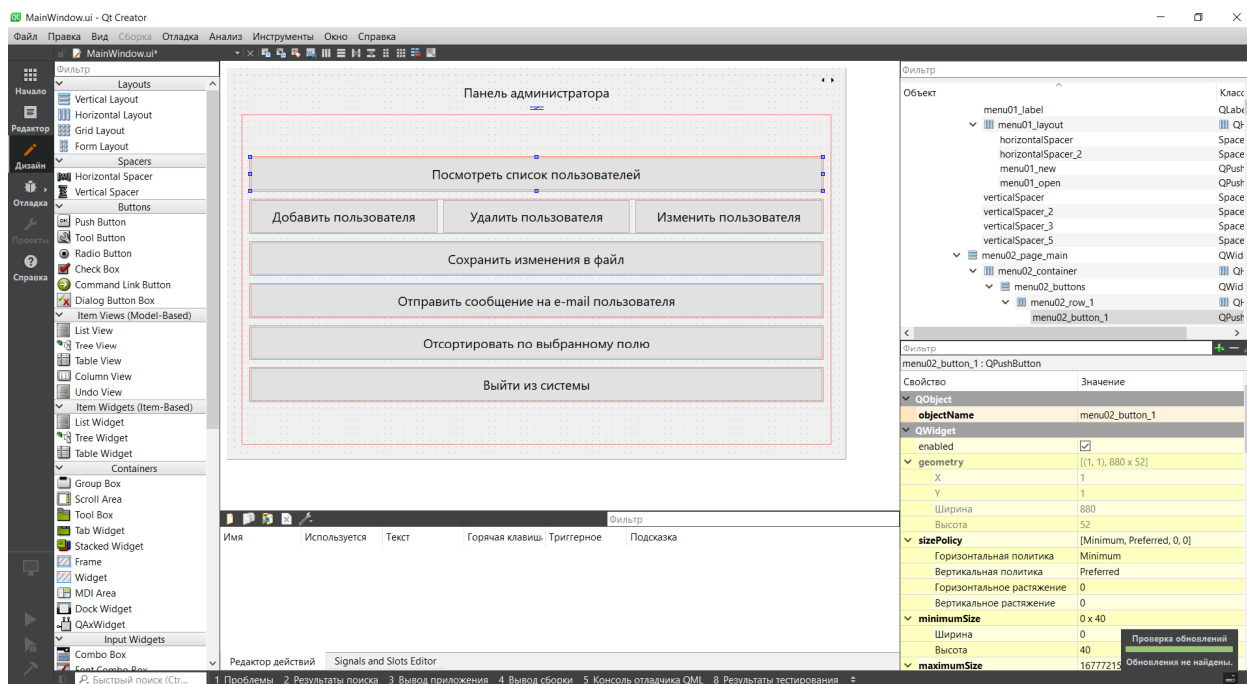


Рисунок 20 – Qt Creator в режиме дизайна

Qt Creator позволяет создавать окна, редактировать их макет и дизайн, добавлять в них виджеты – так называемые исходные элементы для создания пользовательского интерфейса в Qt, изменять их свойства, такие как название, размер, расположение, текст, форму и цвет.

Структура окон сохраняется в файл с расширением *.ui*. Затем этот файл автоматически преобразовывается в исходный код на языке Python, который импортируется в основной файл с названием *main.py*. Во время выполнения программы можно динамически редактировать свойства, добавлять или удалять виджеты, а также делать их интерактивными.

В основном файле при запуске программы инициализируются классы *Data* и *MainWindow*. Класс *Data* содержит переменные состояния программы

и саму базу данных – файл открыт (bool), путь к файлу (string), база данных (dict), файл сохранён (bool). Класс *MainWindow* содержит объекты главного окна, импортированные из файла *gui\MainWindow.py*, и их методы.

Например, метод *connect\_signals()* назначает действия для интерактивных виджетов, то есть задаёт функции, которые нужно вызвать по нажатию какой-либо кнопки, изменению текстового поля или другому действию пользователя.

При запуске программы пользователя встречает окно приветствия с двумя кнопками – создать и открыть файл, которые вызывают функции *new\_file()* и *open\_file()* соответственно. При открытии файла его содержимое считывается и записывается в переменную *data.db* и на экран выводится главное меню программы.

В главном меню есть несколько кнопок, которые позволяют перейти к действиям, указанным в задании.

### 1. Посмотреть список пользователей.

При нажатии вызывается функция *view()*, которая выводит таблицу *mainWindow.ui.menu03\_tableWidget* на экран и заполняет её данными из базы.

```
for person in range(len(data.db["users"])):
    user_info = data.db["users"][person]
    table.setItem(person, 0, QTableWidgetItem(str(user_info["id"])))
    table.setItem(person, 1, QTableWidgetItem(user_info["surname"]))
    table.setItem(person, 2, QTableWidgetItem(user_info["name"]))
    table.setItem(person, 3, QTableWidgetItem(user_info["patronym"] if user_info["patronym"] else "-"))
    table.setItem(person, 4, QTableWidgetItem(user_info["phone"]))
    table.setItem(person, 5, QTableWidgetItem(user_info["e-mail"]))
    table.setItem(person, 6, QTableWidgetItem(user_info["login"]))
    table.setItem(person, 7, QTableWidgetItem(user_info["password"]))
```

Рисунок 21 – Цикл заполнения таблицы

Для возвращения в главное меню предусмотрена кнопка, вызывающая функцию *back\_to\_main\_menu()*. Кнопка для возвращения назад есть во всех меню, так что о ней далее упоминаться не будет.

## 2. Добавить пользователя.

При нажатии вызывается функция `add_user()`, которая открывает форму для регистрации, содержащую поля для ввода данных, кнопку для подтверждения и для отмены. При отмене вызывается `cancel_registration()`. При вводе пароля символы скрываются (это одно из свойств текстовых полей в Qt), но если нажать на иконку в виде глаза, функция `toggle_pass_visible()` переключит видимость пароля. Функция `show_pass_strength()` показывает сложность пароля, определяемую несколькими регулярными выражениями.

Наконец, при нажатии кнопки «Продолжить» вызывается `register()`. Здесь считывается информация с полей ввода в переменные `surname`, `name`, `patronym`, `phone`, `email`, `login`, `password`. Затем каждая переменная последовательно проверяется на соответствие своему регулярному выражению и при первом несовпадении показывается сообщение об ошибке и программа возвращается к этапу ввода данных. Также выполняется проверка на уникальность e-mail адреса и логина.

```
def register():
    surname = mainWindow.ui.menu04_field_surname.text()
    name = mainWindow.ui.menu04_field_name.text()
    patronym = mainWindow.ui.menu04_field_patronym.text()
    phone = mainWindow.ui.menu04_field_phone.text()
    email = mainWindow.ui.menu04_field_email.text()
    login = mainWindow.ui.menu04_field_login.text()
    password = mainWindow.ui.menu04_field_password.text()

    dialog = QMessageBox

    name_regular = r"^[А-ЯЁ]{1}[а-яё]+|[А-З]{1}[a-z]+|[А-ЯЁ]{1}[а-яё]+\-[А-ЯЁ]{1}[а-яё]+|[А-З]{1}[a-z]+\-[А-З]{1}[a-z]+$"
    phone_regular = r"^(+7)\(\d{3}\)\d{3}\d{2}\d{2}$"
    email_regular = r"^[a-zA-Z0-9_+~]+(?:\.[a-zA-Z0-9_+~]+)*@[a-zA-Z0-9_+~]+(?:\.[a-zA-Z0-9_+~]+)+$"
    login_regular = r"[\w\._,.$!%*&]{3,}"
    pass_regular = r"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d\_\.@!%*?&]{8,}$"

    if not re.fullmatch(name_regular, surname):
        if len(surname) > 0:
            message = "Фамилия введена неверно. Пожалуйста, укажите настоящую фамилию.\n\nФИО должны начинаться с большой буквы"
        else:
            message = "Поле \"Фамилия\" является обязательным."
        dialog.warning(mainWindow, "Некорректные данные", message)
        return
    if not re.fullmatch(name_regular, name):
```

Рисунок 22 – Начало функции `register()`

Если проверки пройдены, вычисляется хэш пароля алгоритмом SHA256 и записывается в переменную `pass_hash`. Номер телефона преобразуется в формат +7-(XXX)-XXX-XX-XX, ставится флаг `saved` в

положение false и в базу данных добавляется пользователь. Затем выводится сообщение об успешной регистрации.

```
pass_hash = hashlib.sha256(password.encode("utf-8")).hexdigest()

number = re.fullmatch(phone_regular, phone)
phone = number.group(1) + "-" + number.group(2) + "-" + number.group(3) + "-" + number.group(4) + "-" + number.group(5)

data.saved = False
data.db["last_id"] += 1
data.db["users"].append({"id": data.db["last_id"],
                        "surname": surname,
                        "name": name,
                        "patronym": patronym,
                        "phone": phone,
                        "e-mail": email,
                        "login": login,
                        "password": pass_hash})

mainWindow.ui.menu07_label.setText("<html><head><body><p align='center'><span style=' font-size:12pt;'>Завершение пер
mainWindow.ui.menu07_message.setText("<html><head><body><p align='center'><span style=' font-size:16pt;'>✔</span></
mainWindow.ui.menu07_button_add.setText("Добавить ещё одного пользователя")
mainWindow.ui.menu07_button_add.clicked.connect(add_user)
mainWindow.ui.stackedWidget.setCurrentWidget(mainWindow.ui.menu07_page_success)
```

Рисунок 23 – Конец функции register()

### 3. Удалить пользователя.

При нажатии вызывается функция delete\_user(), открывающая меню, в котором пользователь выбирает критерий поиска (по фамилии-имени, логину или номеру телефона) и вводит поисковой запрос. При изменении критерия вызывается функция search\_criteria\_change(), меняющая поля для ввода, а при редактировании поля вызывается search\_query\_changed(). После ввода при нажатии на кнопку «Поиск» функция search() ищет пользователей и выводит список совпадений, где можно выбрать конкретного пользователя и нажать кнопку удаления. Если поиск не дал результатов, выводится сообщение «Нет пользователей, удовлетворяющих условиям поиска».

```
for user in enumerate(data.db["users"]):
    if user[1]["login"] == login:
        found_users.append((user[1]["login"] + " (" + user[1]["e-mail"] + ")", user[0]))
```

Рисунок 24 – Алгоритм поиска. Поиск по логину

Перед удалением программа запрашивает пароль в функции ask\_for\_password(), проверяет его в check\_password() и если хэши паролей совпадают, то переходит к функции action\_delete().

```
def action_delete(user_string, user_index):
    del data.db["users"][user_index]
    data.saved = False
    mainWindow.ui.menu07_label.setText("<html><head/><body><p align=\\\"center\\\"><span
mainWindow.ui.menu07_message.setText("<html><head/><body><p align=\\\"center\\\"><spa
mainWindow.ui.menu07_button_add.setText("Удалить ещё одного пользователя")
mainWindow.ui.menu07_button_add.clicked.connect(delete_user)
mainWindow.ui.stackedWidget.setCurrentWidget(mainWindow.ui.menu07_page_success)
```

Рисунок 25 – Удаление пользователя

#### 4. Изменить пользователя.

При нажатии вызывается функция `edit_user()`, которая по аналогии с `delete_user()` открывает меню поиска, но в данном случае `search_action = 2` (действия для поиска: 1 – удаление, 2 – изменение, 3 – отправка почты). Кнопка поиска также вызывает функции `search()`, `ask_for_password()` и `check_password()`, только вместо функции `action_delete()` выполняется процедура `edit_screen_show()`, выводящая на экран меню редактирования, в котором все поля заполнены и доступны для изменения, кроме пароля: для его изменения предусмотрена кнопка, открывающая отдельное окно. За показ окна отвечает `edit_pass_window()`, за изменение пароля – `edit_pass()`.

Наконец, после подтверждения изменения срабатывает функция `action_edit()`, которая почти идентична процедуре `register()`: также проводит проверку регулярными выражениями, хэширует пароль, форматирует номер телефона. Только вместо добавления нового пользователя, вносит изменения в выбранную ранее запись.

#### 5. Сохранить изменения в файл.

При нажатии вызывается функция `save()`, которая конвертирует базу данных в json формат и сохраняет файл по пути `data.path`, флаг `data.saved` ставит в положение `true`. Возвращает 1, если сохранение прошло успешно, иначе 0.



```
def save(event=None, confirmation=True):
    dialog = QMessageBox
    try:
        json.dump(data.db, open(data.path, "w", encoding="utf-8"), ensure_ascii=False, indent=4)
        data.saved = True
        mainWindow.ui.menu02_button_5.setStyleSheet("")
        if confirmation:
            dialog.information(mainWindow, "Сохранение", f"Файл {data.path} сохранён.")
        return 1
    except:
        dialog.critical(mainWindow, "Ошибка", "Не удалось сохранить базу данных.")
        return 0
```

Рисунок 26 – Сохранение базы данных в файл

При закрытии окна вызывается функция `check_unsaved()`. Если есть несохранённые изменения, т.е. флаг `saved` в положении `false` – программа предложит перед выходом сохранить файл.

#### **6. Отправить сообщение на e-mail пользователя.**

При нажатии вызывается функция `write_email()`, которая открывает форму отправки письма с 3 полями: «кому», «тема» и «текст письма». По нажатии кнопки «Выбрать пользователя» выполняется процедура `select_user_button()`, переменной `search_action` присваивается значение 3, открывается меню выбора, также вызывается `search()` и при выборе пользователя вызывается `set_recipient()`, где его e-mail адрес копируется в поле «кому».

При нажатии на кнопку «Отправить» выполняется подпрограмма `send()`. Проверяется электронный адрес, в директории рядом с программой создаётся папка «Почта», если её не существует, и в ней папка с адресом получателя. В неё помещается текстовый файл с письмом. Его имя находится в переменной `filename` и состоит из текущей даты и времени, а содержимое файла генерируется по шаблону и помещается в переменную `content`. Проигрывается анимация отправки и файл сохраняется.

```

path = os.path.join(os.getcwd(), "Почта", email)
os.makedirs(path, exist_ok=True)

time = datetime.now()
filename = "inbox_" + time.strftime("%Y-%m-%d_%H-%M-%S") + ".txt"

content = f"Входящее письмо\n\nДата:          {time.strftime('%d.%m.%Y')} \nВремя:          {time.strftime('%H:%M:%S')} \nПолучат

spinner = QMovie(":/gif/Spinner-1s-500px.gif")
spinner.setScaledSize(QSize(100, 100))
mainWindow.ui.menu10_spinner.setMovie(spinner)
spinner.start()
mainWindow.ui.stackedWidget.setCurrentWidget(mainWindow.ui.menu10_page_sending)

loop = QEventLoop()
QTimer.singleShot(1234, loop.quit)
loop.exec()

try:
    open(os.path.join(path, filename), "w", encoding="utf-8").write(content)
except Exception as e:
    spinner.stop()
    mainWindow.ui.stackedWidget.setCurrentWidget(mainWindow.ui.menu06_page_send_email)
    QMessageBox.critical(mainWindow, "E-mail ", "Письмо не было доставлено.\n\nИнформация об ошибке: " + str(e))
    return

```

Рисунок 27 – Фрагмент функции send(). Сохранение файла с письмом

## 7. Отсортировать по выбранному полю.

При нажатии вызывается функция sort\_ui(), которая открывает меню сортировки. В нём пользователь выбирает направление сортировки и поле (ключ) сортировки. Сортировка выполняется в процедуре do\_sorting() посредством встроенной функции sort().

```

def do_sorting():
    reverse = mainWindow.ui.menu11_group1_button2.isChecked()

    if mainWindow.ui.menu11_group2_button1.isChecked(): field = "id"
    elif mainWindow.ui.menu11_group2_button2.isChecked(): field = "surname"
    elif mainWindow.ui.menu11_group2_button3.isChecked(): field = "name"
    elif mainWindow.ui.menu11_group2_button4.isChecked(): field = "patronym"
    elif mainWindow.ui.menu11_group2_button5.isChecked(): field = "phone"
    elif mainWindow.ui.menu11_group2_button6.isChecked(): field = "e-mail"
    else: field = "login"

    data.db["users"].sort(reverse=reverse, key=lambda record: record[field])
    data.saved = False

```

Рисунок 28 – Фрагмент функции do\_sorting(). Сортировка

## 8. Выход.

При нажатии вызывается функция `logout()`, которая вызывает `check_unsaved()` для проверки сохранения и закрывает файл, переходя в меню приветствия. После этого можно начать работу с другим файлом или закрыть окно.

```
def logout():
    if check_unsaved() == 1:
        data.opened = False
        data.saved = False
        mainWindow.ui.menu02_button_5.setStyleSheet("")
        data.path = ""
        data.db = {}
        mainWindow.ui.stackedWidget.setCurrentWidget(mainWindow.ui.menu01_page_welcome)
        mainWindow.ui.menu01_label.setText(mainWindow.ui.menu01_label.text().replace("Добро пожаловать", "Вы вышли из системы."
```

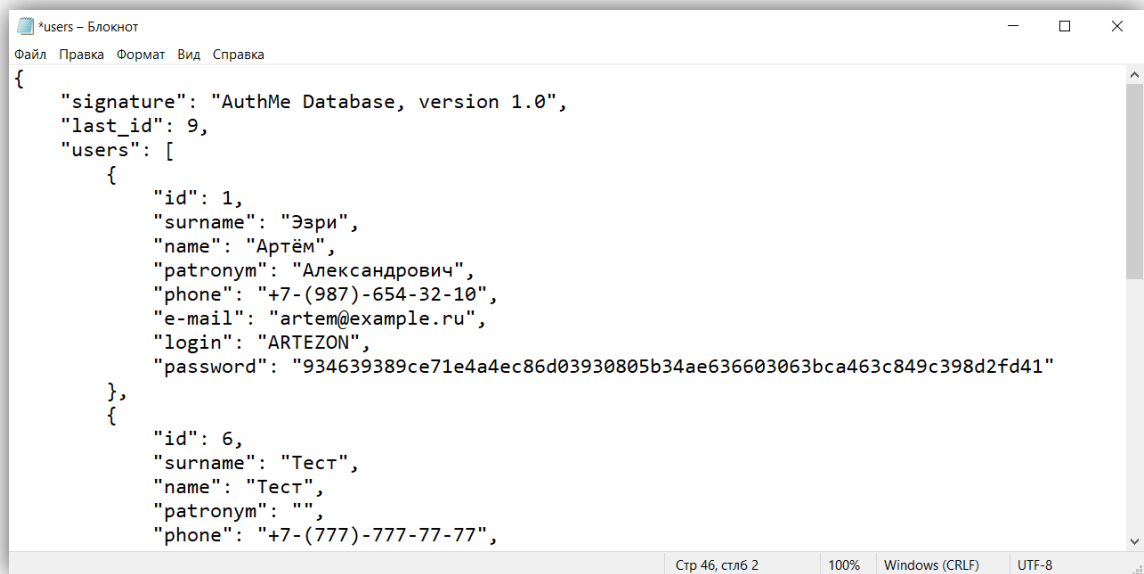
Рисунок 29 – Заккрытие файла и выход из системы

## 3. Описание применяемых алгоритмов и решений

Для хранения информации обо всех пользователях используется словарь – встроенная структура данных в языке Python, позволяющая идентифицировать её элементы не по порядковому номеру (индексу), а по произвольному ключу, например, по строке. Словарь может содержать в себе разные типы одновременно, в том числе и другие словари.

Файл базы данных представляет собой текстовый файл в формате JSON (JavaScript Object Notation). Этот формат легко читается и программой, и человеком. При открытии файла он конвертируется в словарь посредством функции `json.load()` из встроенной библиотеки `json`.

При сохранении файла происходит обратное действие – преобразование словаря в текст и сохранение в json-файл. Для этого используется функция `json.dump()`.



```
{
  "signature": "AuthMe Database, version 1.0",
  "last_id": 9,
  "users": [
    {
      "id": 1,
      "surname": "Ээри",
      "name": "Артём",
      "patronym": "Александрович",
      "phone": "+7-(987)-654-32-10",
      "e-mail": "artem@example.ru",
      "login": "ARTEZON",
      "password": "934639389ce71e4a4ec86d03930805b34ae636603063bca463c849c398d2fd41"
    },
    {
      "id": 6,
      "surname": "Тест",
      "name": "Тест",
      "patronym": "",
      "phone": "+7-(777)-777-77-77",
    }
  ]
}
```

Рисунок 30 – Структура файла users.json

Для сортировки пользователей была использована встроенная в Python функция `sort()`. Она использует алгоритм под названием Timsort.

**Timsort** – гибридный алгоритм сортировки, сочетающий сортировку вставками и сортировку слиянием, опубликованный в 2002 году Тимом Петерсом.

Асимптотическая сложность данного алгоритма сортировки:  $O(n * \log(n))$ .

Поиск пользователей по заданному полю имеет асимптотическую сложность  $O(n)$ , поскольку осуществляется просмотр поля для каждого пользователя в базе данных.

Для проверки правильности ввода данных используются регулярные выражения, соответствующие условиям задания.

- Регулярное выражение для имени, фамилии и отчества:

$$^([A-ЯЁ][a-яё]+|[A-Z][a-z]+|[A-ЯЁ][a-яё]+\-[A-ЯЁ][a-яё]+|[A-Z][a-z]+\-[A-Z][a-z]+)\$$$

Выражение допускает одну из четырёх альтернатив (разделены вертикальной чертой «или»):

- [А-ЯЁ][а-яё]+  
Одна заглавная русская буква, одна или более прописных русских букв. Буква «Ё» указана отдельно, т.к. движок регулярных выражений не включает её в диапазон А-Я.
- [А-Z][а-z]+  
Одна заглавная латинская буква, одна или более прописных латинских букв.
- [А-ЯЁ][а-яё]+\-[А-ЯЁ][а-яё]+  
Одна заглавная русская буква, одна или более прописных русских букв, затем тоже самое через дефис, например, Салтыков-Щедрин.
- [А-Z][а-z]+\-[А-Z][а-z]+  
То же самое, только для латинского алфавита.
- Регулярное выражение для телефона:  

$$^(\backslash+7)([\backslash- ]?(?(\backslash d\{3})\backslash)?[\backslash- ]?(?(\backslash d\{3})[\backslash- ]?(?(\backslash d\{2}))[\backslash- ]?(?(\backslash d\{2}))\$$$
  - ^ Начало строки
  - (\+7) Группа 1. +7
  - [\ - ]? Необязательный дефис или пробел
  - \(? Необязательная открывающая скобка
  - (\d{3}) Группа 2. Три цифры
  - \)? Необязательная закрывающая скобка
  - [\ - ]? Необязательный дефис или пробел
  - (\d{3}) Группа 3. Три цифры
  - [\ - ]? Необязательный дефис или пробел
  - (\d{2}) Группа 4. Две цифры
  - [\ - ]? Необязательный дефис или пробел
  - (\d{2}) Группа 5. Две цифры
  - \$ Конец строки

- Регулярное выражение для почты:

$^{\wedge}([a-zA-Z0-9\_+ - ]+(?:\.[a-zA-Z0-9\_+ - ]+)^*@[a-zA-Z0-9\_+ - ]+(?:\.[a-zA-Z0-9\_+ - ]+)^+)\$$

Имя почтового ящика может содержать 1 или более латинских букв, цифр и символов «\_», «+», «-», может содержать точки, но не в начале и не в конце, также нельзя ставить две точки подряд.

Домен должен состоять как минимум из 2 частей, разделённых точкой.

В этих частях (уровнях домена) разрешены латинские буквы, цифры и знаки «\_».

- Регулярное выражение для логина:

$[\backslash w\_ = . , \$ ! \% * ? \& ] \{ 3 , \}$

Разрешены все буквы ( $\backslash w$ ) и символы  $\_ = . , \$ ! \% * ? \& .$

Длина логина  $\geq 3$ .

- Регулярное выражение для пароля:

$^{\wedge}(?=.[a-z])(?=.[A-Z])(?=.\d)(?=.[@\$!\%*?&])[A-Za-z\d\_ - \. @$!%*?& ] \{ 8 , \} \$$

- $[A-Za-z\d\_ - \. @$!%*?& ]$  определяет множество разрешённых символов.
- $\{ 8 , \}$  определяет длину ( $\geq 8$ ).
- $(?=.[a-z])(?=.[A-Z])(?=.\d)(?=.[@\$!\%*?& ])$  – опережающая проверка. Первая скобка проверяет, что в пароле есть сколь угодно символов, после которых стоит буква a-z, т.е. встречается буква a-z, вторая проверяет наличие буквы A-Z, третья – наличие цифры, четвёртая – наличие хотя бы одного из следующих спец. символов: @\$!%\*?& .

Пароли пользователей хранятся в формате хэш-кодов. Для реализации хранения в данной форме используется хэш-функция.

**Хэш-функция** – это функция, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку

установленной длины, выполняемое определённым алгоритмом.

**Хэш-код** – это результат хэш-функции.

Для получения хэш-кода подключается встроенная библиотека *hashlib*.  
Используемый алгоритм – SHA256. Результат работы этого алгоритма – строка размером 256 бит (64 шестнадцатеричные цифры, т.е. 64 символа).

Реализация хеширования:

```
pass_hash = hashlib.sha256(password.encode("utf-8")).hexdigest()
```

#### 4. Инструкция по эксплуатации

При запуске программы открывается экран приветствия, в котором можно создать новую базу данных или открыть существующую.

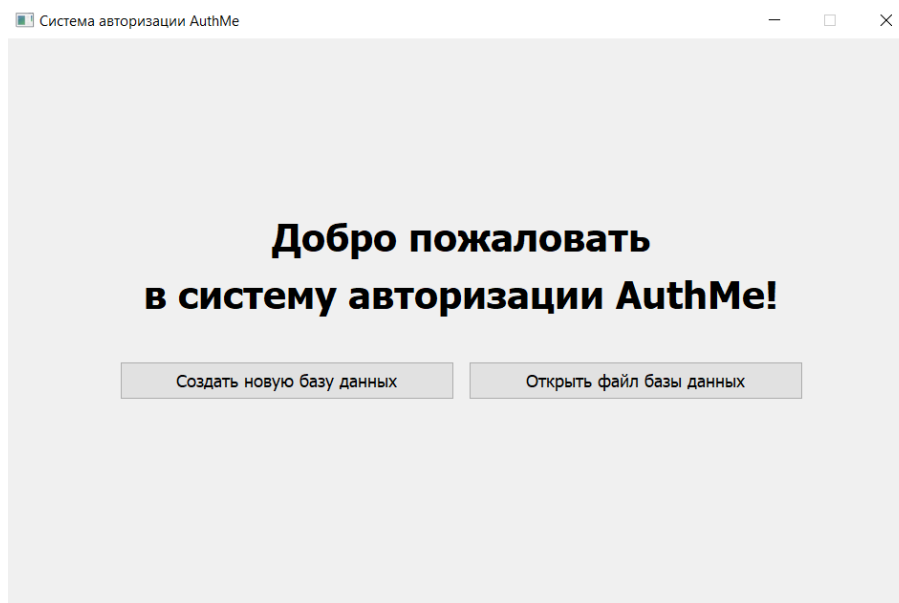


Рисунок 31 – Экран приветствия

После успешного открытия файла пользователь попадает в главное меню, которое предлагает выбрать одну из нескольких альтернатив.

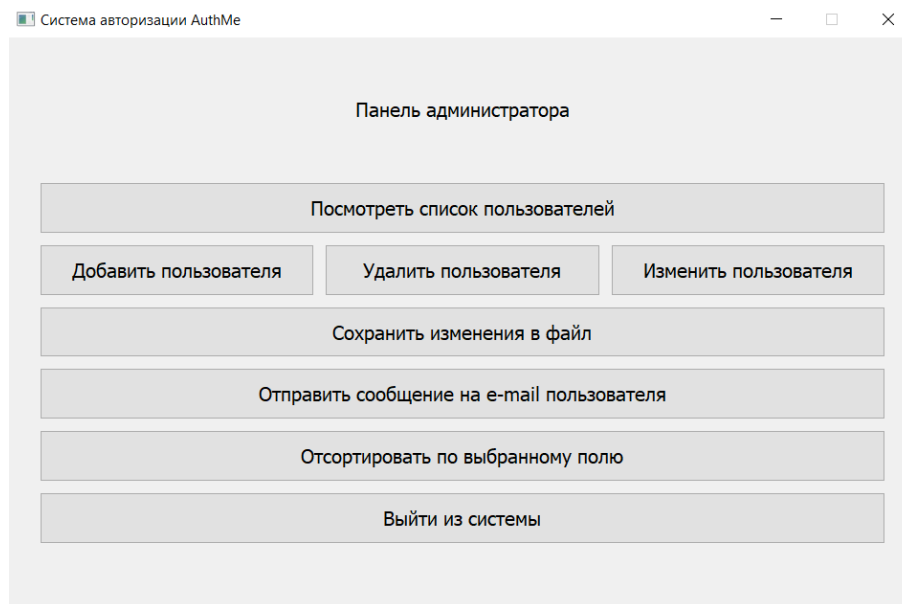


Рисунок 32 – Главное меню программы

Чтобы посмотреть список пользователей, нужно нажать на первую кнопку.

**Примечание.** Все данные о пользователях сгенерированы случайным образом и никак не связаны с реальными людьми.

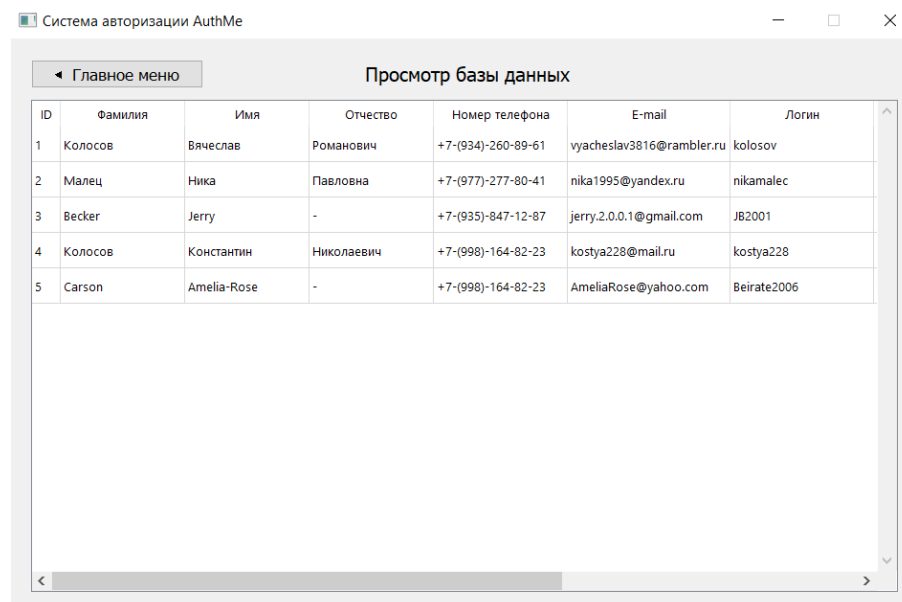


Рисунок 33 – Просмотр базы данных

Чтобы вернуться в главное меню, нажмите на левую верхнюю кнопку.



Для регистрации нового пользователя воспользуйтесь второй опцией в главном меню. Перед Вами откроется форма регистрации. Обязательные поля будут помечены звёздочкой. Номер заполняется в формате +7-(XXX)-XXX-XX-XX, при этом скобки и дефисы можно не вводить. Пароль должен иметь статус «Надёжный пароль». Наведите курсор мыши на поле ввода пароля, чтобы увидеть подсказку.

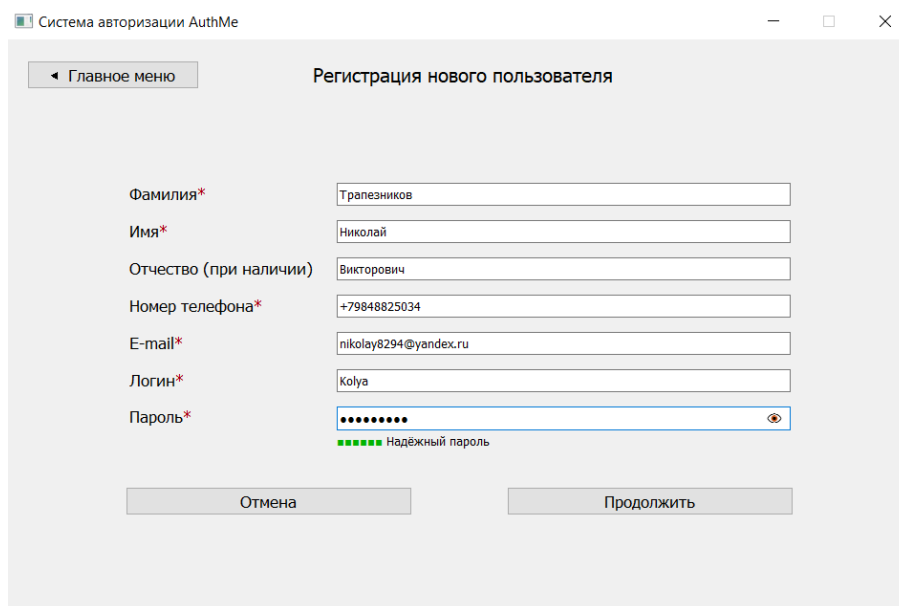


Рисунок 34 – Заполненная форма регистрации

После ввода данных нажмите «Продолжить». Если все поля заполнены правильно, пользователь будет зарегистрирован, иначе программа сообщит, что именно нужно исправить.

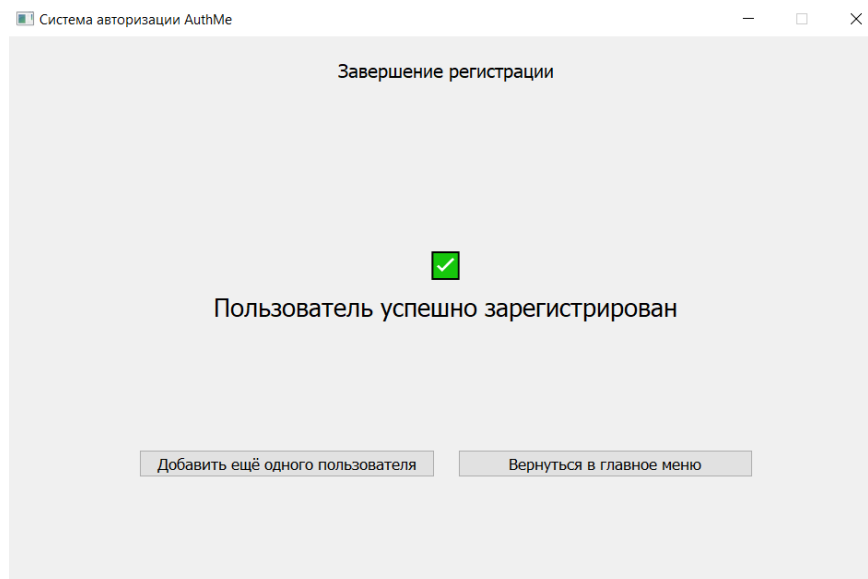


Рисунок 35 – Сообщение об успешной регистрации

При наличии несохранённых изменений кнопка сохранения будет подсвечиваться зелёным, а при закрытии программы появится диалоговое окно с подтверждением.

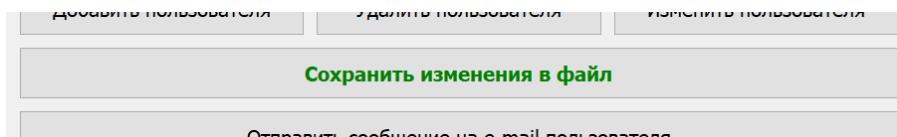


Рисунок 36 – Индикация несохранённых изменений

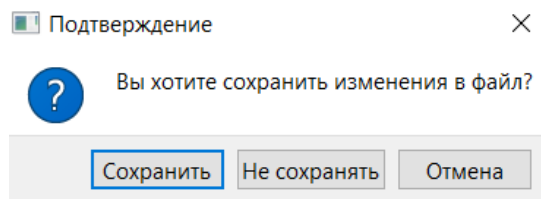


Рисунок 37 – Подтверждение перед выходом

Чтобы удалить или изменить пользователя, нужно найти его по фамилии, имени, логину или телефону. Если ввести только имя или только фамилию, найдутся все пользователи с таким именем или фамилией. Если ввести и имя, и фамилию, то будут найдены те люди, у которых совпадает и имя, и фамилия.

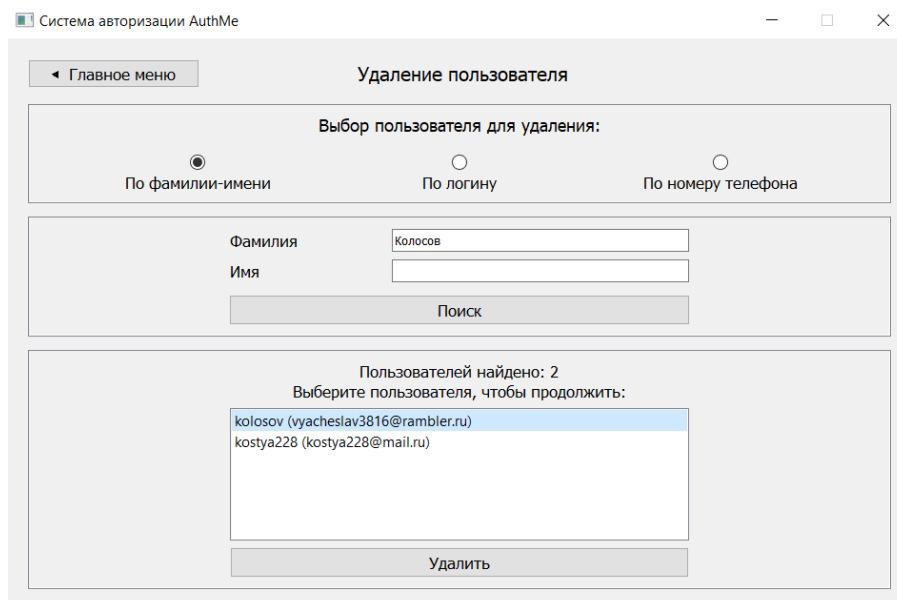


Рисунок 38 – Поиск пользователей

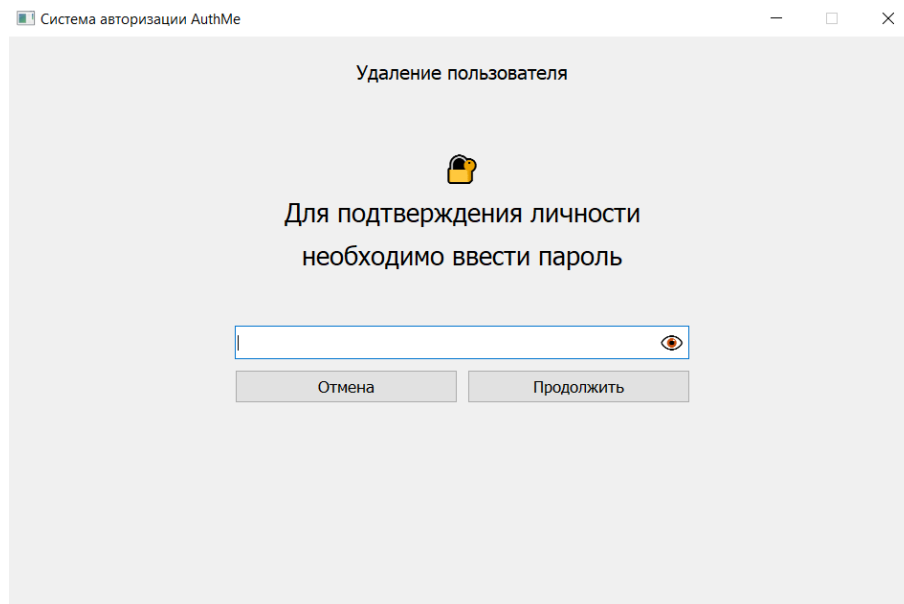


Рисунок 39 – Подтверждение личности

После регистрации 6-го и удаления 1-го пользователя база данных выглядит так:

Система авторизации AuthMe

Главное меню

Просмотр базы данных

ID	Фамилия	Имя	Отчество	Номер телефона	E-mail	Логин
2	Малец	Ника	Павловна	+7-(977)-277-80-41	nika1995@yandex.ru	nikamalec
3	Becker	Jerry	-	+7-(935)-847-12-87	jerry.2.0.0.1@gmail.com	JB2001
4	Колосов	Константин	Николаевич	+7-(998)-164-82-23	kostya228@mail.ru	kostya228
5	Carson	Amelia-Rose	-	+7-(998)-164-82-23	AmeliaRose@yahoo.com	Beirate2006
6	Трапезников	Николай	Викторович	+7-(984)-882-50-34	nikolay8294@yandex.ru	Kolya

Рисунок 40 – Изменённая база данных

Вот что выводится на экран, если пользователь не найден:

Рисунок 41 – Пользователь не найден

В случае редактирования, после подтверждения пароля появится заполненная форма. Поменяйте нужные значения.

Рисунок 42 – Редактирование. Изменено отчество

Чтобы отправить электронное письмо, выберите соответствующий пункт в главном меню.

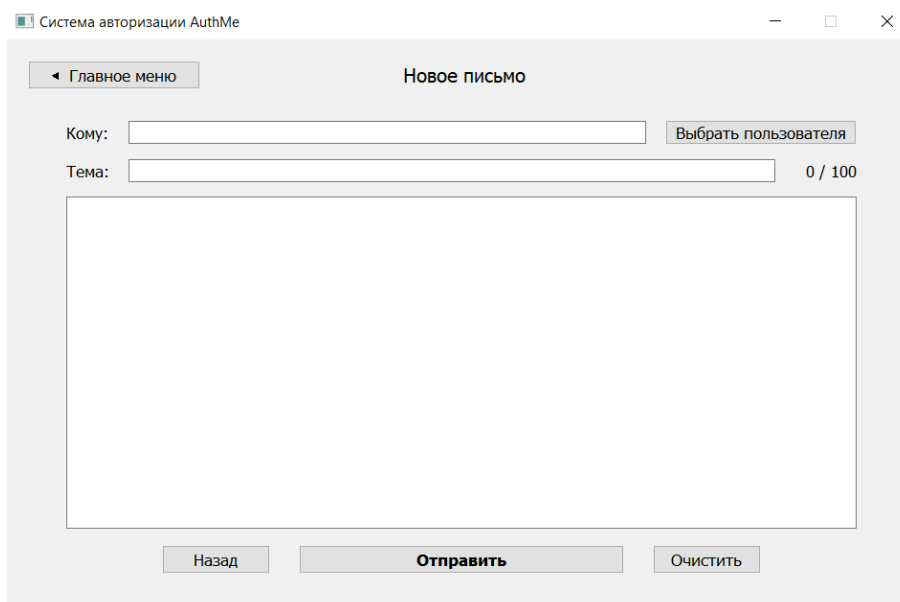


Рисунок 43 – Отправка почты

Впишите адрес вручную или воспользуйтесь функцией «Выбрать пользователя». В последнем случае откроется такой же интерфейс поиска, как и при удалении и изменении и адрес вставится автоматически.

Заполните тему (необязательно) и текст письма.

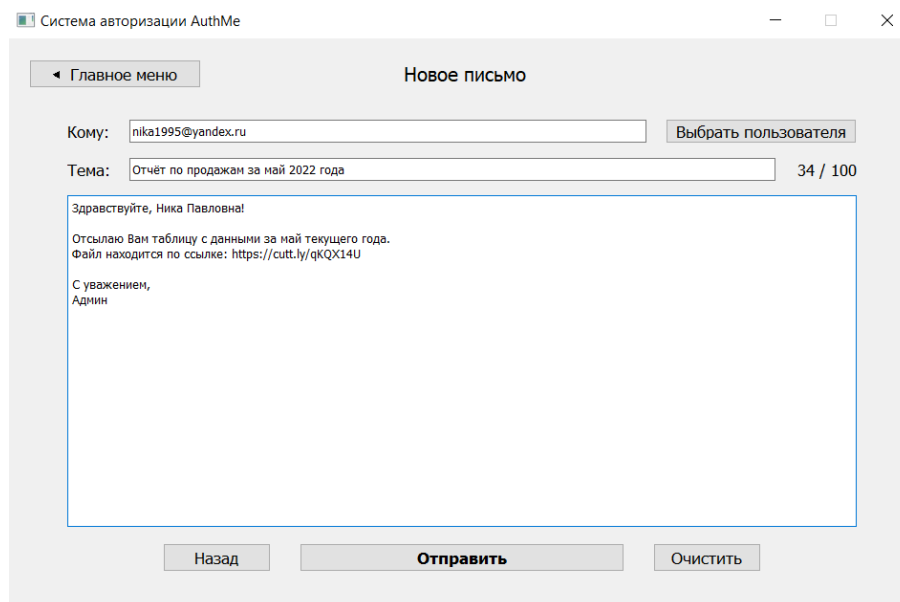


Рисунок 44 – Пример письма

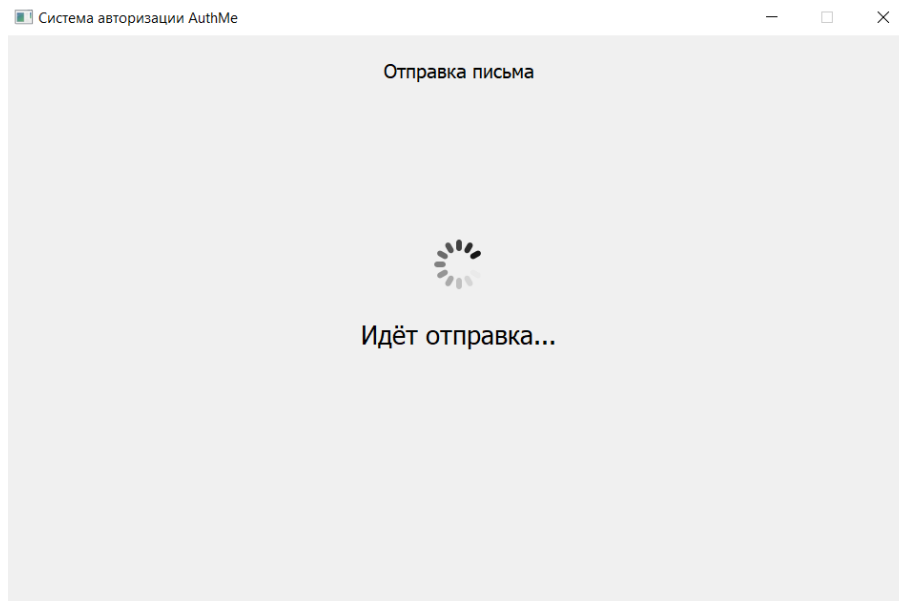


Рисунок 45 – Анимация отправки

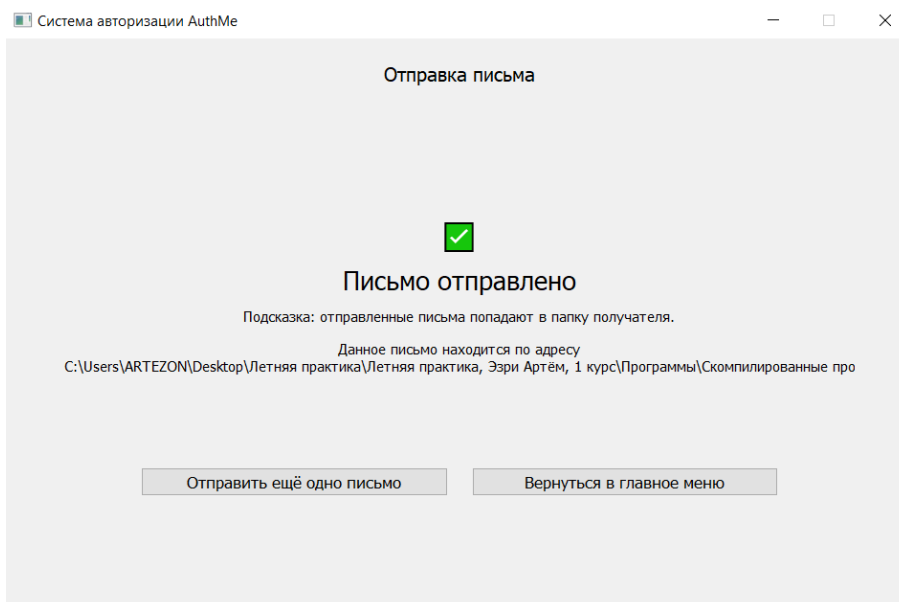


Рисунок 46 – Письмо отправлено

Так как письма отправляются «понарошку», они сохраняются в файл на компьютере, как будто помещаются в папку «Входящие» получателя.

Путь к файлу: Почта\<адрес\_получателя>\inbox\_<дата\_и\_время>.txt  
(относительно файла программы).

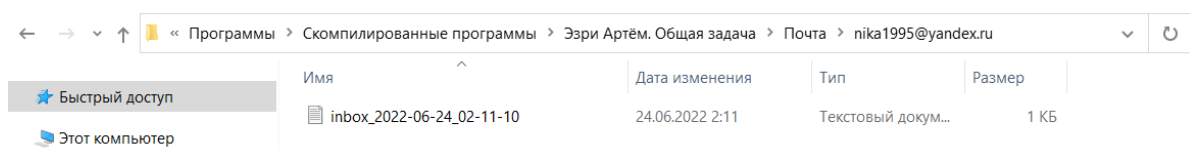


Рисунок 47 – Файл письма

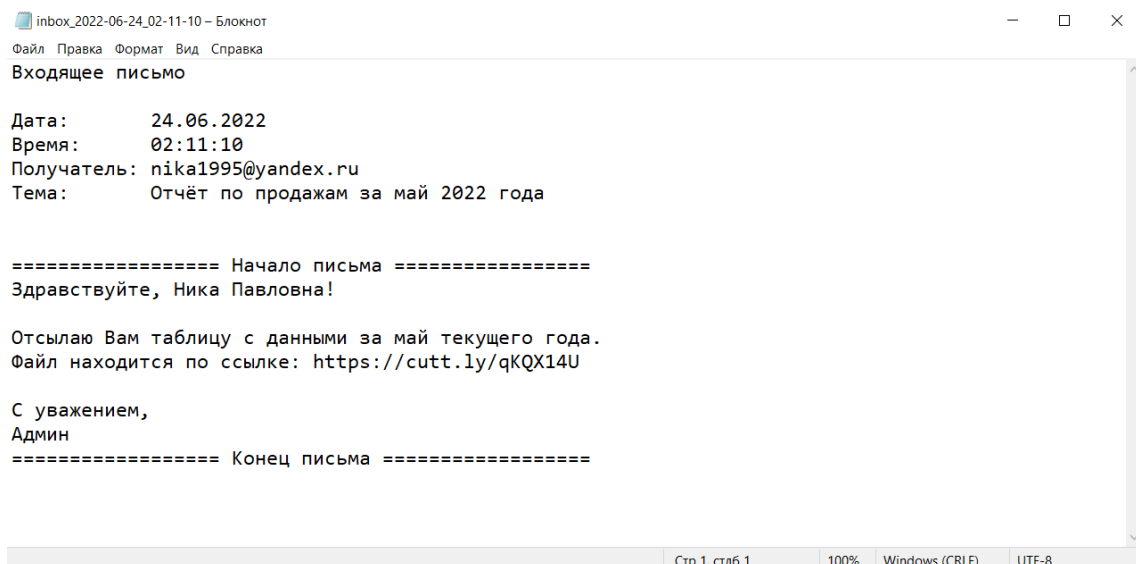


Рисунок 48 – Содержимое файла

И последнее – сортировка.

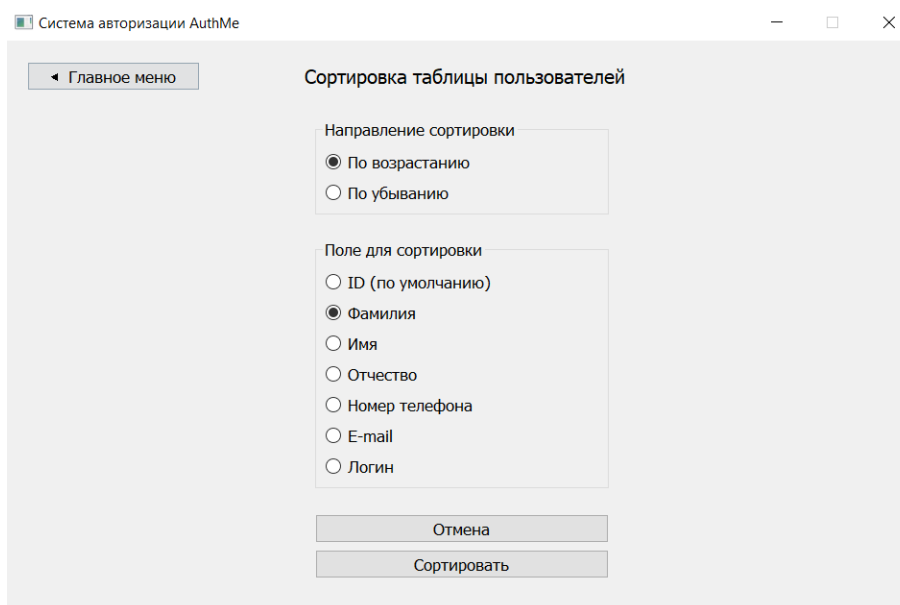


Рисунок 49 – Сортировка

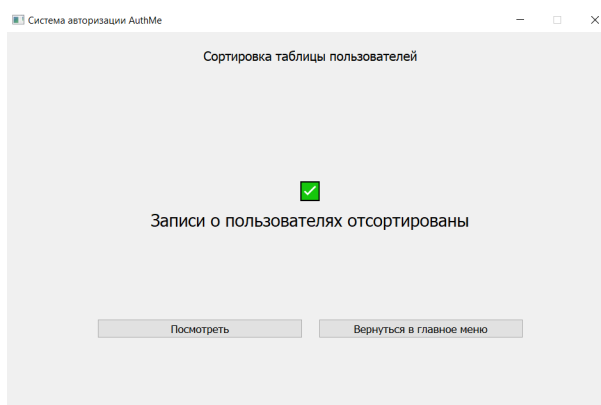
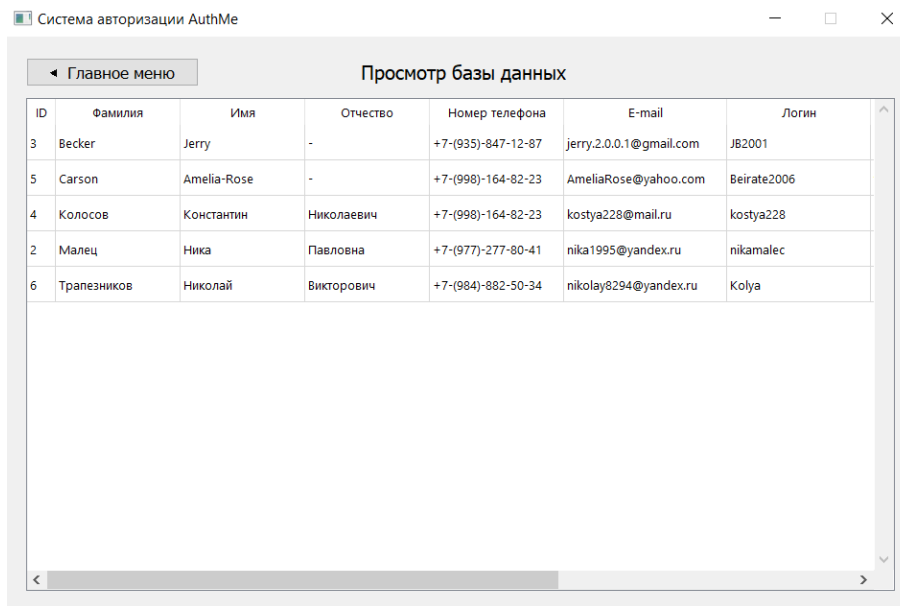


Рисунок 50 – Сообщение о сортировке

Записи отсортированы по возрастанию алфавита (от А до Я) по полю «фамилия». Сначала идёт английский алфавит, затем русский.



Система авторизации AuthMe

Главное меню

Просмотр базы данных

ID	Фамилия	Имя	Отчество	Номер телефона	E-mail	Логин
3	Becker	Jerry	-	+7-(935)-847-12-87	jerry.2.0.0.1@gmail.com	JB2001
5	Carson	Amelia-Rose	-	+7-(998)-164-82-23	AmeliaRose@yahoo.com	Beirate2006
4	Колосов	Константин	Николаевич	+7-(998)-164-82-23	kostya228@mail.ru	kostya228
2	Малец	Ника	Павловна	+7-(977)-277-80-41	nika1995@yandex.ru	nikamalec
6	Трапезников	Николай	Викторович	+7-(984)-882-50-34	nikolay8294@yandex.ru	Kolya

Рисунок 51 – Отсортированная по фамилии таблица

После выхода из системы (последняя кнопка в меню) файл закрывается и пользователь снова попадает на первоначальный экран.

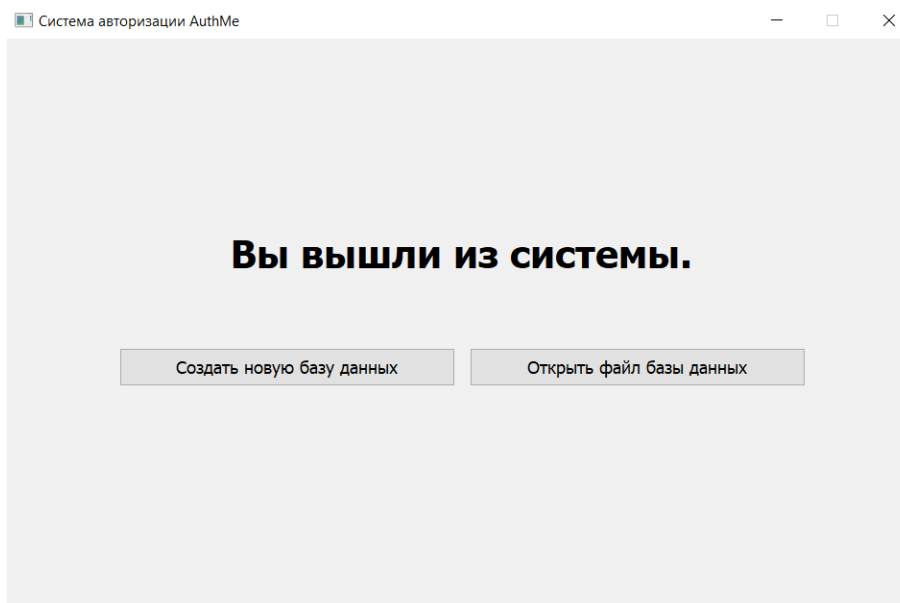


Рисунок 52 – Завершение работы с программой



## 5. Набор информации для тестирования

```
{
  "signature": "AuthMe Database, version 1.0",
  "last_id": 5,
  "users": [
    {
      "id": 1,
      "surname": "Колосов",
      "name": "Вячеслав",
      "patronym": "Романович",
      "phone": "+7-(934)-260-89-61",
      "e-mail": "vyacheslav3816@rambler.ru",
      "login": "kolosov",
      "password": "bcf1afed4262f6405bc59982fa2bd3bbac3ac948a332f008dced6c3301ae4f4b"
    },
    {
      "id": 2,
      "surname": "Малец",
      "name": "Ника",
      "patronym": "Павловна",
      "phone": "+7-(977)-277-80-41",
      "e-mail": "nika1995@yandex.ru",
      "login": "nikamalec",
      "password": "1bb16b90279c54e91591183a26a4253aef3a97b7a6cf0ad3e0f295f5ee69d211"
    },
    {
      "id": 3,
      "surname": "Becker",
      "name": "Jerry",
      "patronym": "",
      "phone": "+7-(935)-847-12-87",
      "e-mail": "jerry.2.0.0.1@gmail.com",
      "login": "JB2001",
      "password": "c718a36ef12646b6e9400d41942dbacf104b001a60cdbdc12cebbf078965f921"
    },
    {
      "id": 4,
      "surname": "Колосов",
      "name": "Константин",
      "patronym": "Николаевич",
      "phone": "+7-(998)-164-82-23",
      "e-mail": "kostya228@mail.ru",
      "login": "kostya228",
      "password": "5f815dfd72fddbb3080ac21c67ac236ddb851af197694165c27865beb6316277"
    },
    {
      "id": 5,
      "surname": "Carson",
      "name": "Amelia-Rose",
      "patronym": "",
      "phone": "+7-(998)-164-82-23",
      "e-mail": "AmeliaRose@yahoo.com",
      "login": "Beirate2006",
      "password": "f6a731b6347941f2a0bfada2214fb618232a8e8fc1a61511ee6aeea93f4611c0"
    }
  ]
}
```

## **ЗАКЛЮЧЕНИЕ**

Приобретён опыт в реализации программных продуктов, основанных на знаниях, полученных за время обучения на первом курсе: разработана программа, вычисляющая определитель матрицы, разработан программный продукт, вычисляющий определённый интеграл некоторых непрерывных функций, создано приложение для построения и вывода АНФ для булевых функций, а также разработана база данных для хранения и изменения данных о пользователях.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фаддеев Д. К. – Лекции по алгебре
2. Шипачев В. С. – Основы высшей математики
3. [https://neerc.ifmo.ru/wiki/index.php?title=Определение\\_булевой\\_функции](https://neerc.ifmo.ru/wiki/index.php?title=Определение_булевой_функции)
4. <https://habr.com/ru/post/275527/> (*метод треугольника*)
5. <https://en.cppreference.com> (*справочник по языку C++*)
6. <https://www.pythonguis.com/pyqt5-tutorial/> (*руководство по созданию графических интерфейсов в Python*)
7. <https://ru.wikipedia.org/wiki/Хеш-функция> (*определение хэш-функции*)

## ПРИЛОЖЕНИЕ

### 1. Исходный код программы по алгебре:

```
#include <windows.h>
#include <direct.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <conio.h>
#include <vector>
#include <string>
#include <tuple>
#include <map>

using namespace std;

pair<bool, long double> float_from_string(string s) {
    bool has_point = 0;
    for (int i = 0; i < s.size(); i++) {
        if (!((s[i] >= '0' && s[i] <= '9') || s[i] == '-' || s[i] == '.' || s[i] == ',')) return { 0, 0 };
        else {
            if (s[i] == '-' && i != 0) return { 0, 0 };
            if (s[i] == '.' || s[i] == ',') {
                if (has_point) return { 0, 0 };
                else {
                    has_point = 1;
                    if (i == 0) {
                        s.insert(0, 1, '0');
                        i++;
                    }
                    else if (i == s.size() - 1) {
                        s.pop_back();
                        has_point = 0;
                    }
                    if (s[i] == ',') s[i] = '.';
                }
            }
        }
    }
    return { 1, stof(s) };
}

vector<vector<long double>> read_matrix(string file_path) {
    vector<vector<long double>> matrix;
    ifstream f(file_path);
    if (!f.is_open()) {
        f.close();
        ofstream newfile(file_path, fstream::app);
        newfile.close();
        ifstream check(file_path);
        if (!check.is_open()) {
            check.close();
            cout << "\033[31m" "[ОШИБКА] Отказано в доступе к файлу \' " <<
file_path << "\'.\n" "\033[30m";
            cout << "\n\nНажмите любую клавишу, чтобы выйти.";
            _getch();
            return {};
        }
        check.close();
        cout << "\033[32m" "Файл \' " << file_path << "\' создан.\nПожалуйста,
заполните его матрицей, затем снова запустите программу.\n" "\033[30m";
    }
}
```

```

        cout << "\n\nНажмите любую клавишу, чтобы выйти.";
        _getch();
        return {};
    }

    int size = 0, row = 0;
    string line, entry;
    stringstream line_stream;

    while (getline(f, line)) {
        row++;
        matrix.push_back({});

        line_stream.clear();
        line_stream.str(line);

        while (line_stream >> entry) {
            auto temp = float_from_string(entry);
            if (!temp.first) {
                cout << "\033[31m" "[ОШИБКА] Не удалось прочитать число \'' <<
entry << "\' из " << row << "-й строки.\n";
                cout << "Пожалуйста, проверьте вашу матрицу ещё раз.\n"
"\033[30m";

                cout << "\n\nНажмите любую клавишу, чтобы выйти.";
                _getch();
                return {};
            }
            else matrix[row - 1].push_back(temp.second);
        }
        if (size == 0) size = matrix[0].size();
        if (matrix[row - 1].size() != size) {
            cout << "\033[31m" "[ОШИБКА] Количество чисел в " << row << "-й
строке не совпадает с предыдущими.\n";
            cout << "Пожалуйста, проверьте вашу матрицу ещё раз.\n" "\033[30m";
            cout << "\n\nНажмите любую клавишу, чтобы выйти.";
            _getch();
            return {};
        }
    }

    if (size == 0) {
        cout << "\033[31m" "[ОШИБКА] Файл \'' << file_path << "\' пуст.\n"
"\033[30m";
        cout << "\n\nНажмите любую клавишу, чтобы выйти.";
        _getch();
        return {};
    }

    if (row != size) {
        cout << "\033[31m" "[ОШИБКА] Определитель можно вычислить только для
квадратной матрицы. Размерность вашей матрицы: " << row << "x" << size << ".\n";
        cout << "Пожалуйста, проверьте вашу матрицу ещё раз.\n" "\033[30m";
        cout << "\n\nНажмите любую клавишу, чтобы выйти.";
        _getch();
        return {};
    }

    return matrix;
}

void print_matrix(vector<vector<long double>> matrix) {
    size_t size = matrix.size();

    size_t max_number_length = 0;
    for (int i = 0; i < size; i++) {

```

```

        for (int j = 0; j < size; j++) {
            stringstream temp;
            temp << matrix[i][j];
            size_t this_length = temp.str().size();
            if (this_length > max_number_length) max_number_length =
this_length;
        }
    }

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            cout.width(max_number_length);
            cout << matrix[i][j];
            cout << " ";
        }
        cout << "\n";
    }
}

vector<vector<long double>> submatrix(vector<vector<long double>> matrix, int
del_row, int del_col) {
    size_t new_size = matrix.size() - 1;
    vector<vector<long double>> new_matrix(new_size, vector<long
double>(new_size));
    int i = 0, j = 0;
    for (int row = 1; row <= matrix.size(); row++) {
        if (row != del_row) {
            for (int col = 1; col <= matrix.size(); col++) {
                if (col != del_col) {
                    new_matrix[i][j] = matrix[row - 1][col - 1];
                    j++;
                }
            }
            i++;
            j = 0;
        }
    }
    return new_matrix;
}

long double determinant(vector<vector<long double>> matrix, int column) {
    static map<tuple<vector<vector<long double>>, int, int>, long double> cache;

    if (matrix.empty()) return 0;
    if (matrix.size() == 1) return matrix[0][0];

    long double det = 0;
    int size = matrix.size(), sign;
    if (size < column) column = 1;
    for (int row = 1; row <= size; row++) {
        if (matrix[row - 1][column - 1] != 0) {
            if ((row + column) % 2 == 0) sign = 1;
            else sign = -1;

            long double submatr_det;
            auto cache_iter = cache.find({ matrix, row, column });
            if (cache_iter != cache.end()) {
                submatr_det = cache_iter->second;
            }
            else {
                submatr_det = determinant(submatrix(matrix, row, column),
column);
                cache[{ matrix, row, column }] = submatr_det;
            }
        }
    }
}

```

```

        det += sign * matrix[row - 1][column - 1] * submatr_det;
    }
}
return det;
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    system("color F0");

    vector<vector<long double>> matrix;
    string filename = "matrix.txt";
    string cwd = _getcwd(NULL, 0);
    matrix = read_matrix(cwd + "\\ " + filename);
    if (matrix.empty()) return 0;

    cout << "Ваша матрица:\n\n";
    print_matrix(matrix);

    cout << "\n";
    cout << "Сейчас я вычислю определитель этой матрицы. Для начала задайте номер столбца,\n";
    cout << "по которому будет выполняться разложение. Если ничего не введено, введён\n";
    cout << "посторонний символ или число вне допустимого диапазона, то разложение матрицы\n";
    cout << "будет производиться по первому столбцу. При разложении матрицы с меньшим размером,\n";
    cout << "чем введённое число, разложение будет также производиться по первому столбцу.\n";
    cout << "\n";
    cout << "Итак, введите номер столбца ниже:\n";
    cout << ">> ";
    unsigned int column;
    cin >> column;
    if (column < 1 || column > matrix.size()) column = 1;
    cout << "\n";

    cout << "Идёт подсчёт...";
    long double det = determinant(matrix, column);
    cout << "\r0 определитель данной матрицы равен " << det << "\n";

    cout << "\n\nНажмите любую клавишу, чтобы выйти.";
    _getch();
}

```

## 2. Исходный код программы по математическому анализу:

```

#include <Windows.h>
#include <iostream>
#include <iomanip>
#include <conio.h>
#include <string>
#include <cmath>

using namespace std;

long double f(long double x, unsigned int func_id) {
    switch (func_id) {
        case 1: return powl(x, 2);
        case 2: return sqrtl(abs(x));
        case 3: return logl(abs(x));
    }
}

```

```

    case 4: return expl(x);
    case 5: return sinl(x);
    case 6: return 2 * x * cosl(x);
    case 7: return 2 * x * x + 3 * x - 1;
    case 8: return (9 * x + 5) / (x - 8);
    case 9: return expl(3 * x) * sinl(x * x);
    case 10: return x / sqrtl(powl(x, 4) + 16);
    default: return 0;
}
}

long double integral(unsigned int func_id, long double a, long double b, long
double h) {
    unsigned long long int n = abs(b - a) / h;
    long double sum = 0;
    int sign = 1;
    if (a > b) {
        swap(a, b);
        sign = -sign;
    }

    for (int i = 1; i <= n; i++) {
        sum += f(a + ((2 * i - 1) * h / 2), func_id);
    }

    return h * sum * sign;
}

string floating_point(long double d) {
    string s = to_string(d);
    while (!s.empty() && *(s.end() - 1) == '0') s.pop_back();
    if (!s.empty() && *(s.end() - 1) == '.') s.pop_back();
    return s;
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    system("color F0");

    int func_id;
    long double a, b, step;

    while (1) {
        system("cls");
        func_id = 0;
        cout << "Добро пожаловать в решатель определённых интегралов методом
серединных прямоугольников!\n"
            "Пожалуйста, выберите функцию и введите её номер:\n"
            " 1) x^2\n"
            " 2) sqrt(|x|)\n"
            " 3) ln(|x|)\n"
            " 4) e^x\n"
            " 5) sin(x)\n"
            " 6) 2x * cos(x)\n"
            " 7) 2x^2 + 3x - 1\n"
            " 8) (9x + 5) / (x - 8)\n"
            " 9) e^(3x) * sin(x^2)\n"
            "10) x / sqrt(x^4 + 16)\n";
        while (func_id < 1 || func_id > 10) {
            cout << "\n>> ";
            cin >> func_id;
            cout << "\n";
            if (func_id <= 0) {

```



```

        cout << "\033[31m" "Я вас не понимаю. Пожалуйста, введите число
от 1 до 10.\n" "\033[30m";
        cin.clear();
        cin.ignore(INT_MAX, '\n');
    }
    if (func_id > 10) {
        cout << "\033[31m" "У меня нет функции под номером " << func_id
<< ". Пожалуйста, введите число от 1 до 10.\n" "\033[30m";
        cin.clear();
        cin.ignore(INT_MAX, '\n');
    }
}

cout << "Хорошо, теперь введите нижний и верхний пределы интегрирования
через пробел.\n";
while (1) {
    cout << "\n>> ";
    cin >> a >> b;
    cout << "\n";
    if (cin.fail()) {
        cout << "\033[31m" "Это не число :(\n" "\033[30m";
        cin.clear();
        cin.ignore(INT_MAX, '\n');
    }
    else break;
}

cout << "Отлично. Осталось только ввести шаг интегрирования – ширину
прямоугольников. Например, 0.0001\n";
while (1) {
    cout << "\n>> ";
    cin >> step;
    cout << "\n";
    if (cin.fail()) {
        cout << "\033[31m" "Это не число :(\n" "\033[30m";
        cin.clear();
        cin.ignore(INT_MAX, '\n');
    }
    else break;
}

```

```

string ascii_image[10];
ascii_image[0] = R"(
    /
    |  2
    | x  dx
    |
    /
)";
ascii_image[1] = R"(
    /
    |  \sqrt{x}  dx
    |
    /
)";
ascii_image[2] = R"(
    /
    |  ln|x|  dx
    |
    /
)";
ascii_image[3] = R"(
    /
    |  x
    | e  dx
    |
    /
)";
ascii_image[4] = R"(
    /
    |
)";

```

```

        R"(      | sinx dx      )" "\n"
        R"(      |          )" "\n"
        R"(      |          )" "\n";
ascii_image[5] = R"(      |          )" "\n";
        R"(      |          )" "\n"
        R"(      | 2 x cosx dx )" "\n"
        R"(      |          )" "\n"
        R"(      |          )" "\n";
ascii_image[6] = R"(      |          )" "\n";
        R"(      |          )" "\n"
        R"(      | /      2      \ )" "\n"
        R"(      | | 2 x  + 3 x - 1 | dx )" "\n"
        R"(      | \          / )" "\n"
        R"(      |          )" "\n";
ascii_image[7] = R"(      |          )" "\n";
        R"(      |          )" "\n"
        R"(      | 9 x + 5          )" "\n"
        R"(      | ----- dx )" "\n"
        R"(      | x - 8          )" "\n"
        R"(      |          )" "\n";
ascii_image[8] = R"(      |          )" "\n";
        R"(      |          )" "\n"
        R"(      | 3 x      2          )" "\n"
        R"(      | e      sin x dx )" "\n"
        R"(      |          )" "\n"
        R"(      |          )" "\n";
ascii_image[9] = R"(      |          )" "\n";
        R"(      |          )" "\n"
        R"(      |          )" "\n"
        R"(      |          )" "\n"
        R"(      |          )" "\n"
        R"(      |          )" "\n";

system("cls");
cout << "\n" << floating_point(b) << "\n";
cout << ascii_image[func_id - 1];
cout << " " << floating_point(a) << "\n";

cout << "\nВычисляю интеграл...";

long double I = integral(func_id, a, b, step);
string I_str = floating_point(I);

cout << "\rИнтеграл приблизительно равен " << fixed <<
setprecision(numeric_limits<double>::max_digits10) << I_str << endl;

cout << "\n\nНажмите любую клавишу, чтобы вернуться в главное меню.";
_getch();
}
}

```

### 3. Исходный код программы по дискретной математике:

```

#include <windows.h>
#include <direct.h>
#include <iostream>
#include <fstream>
#include <conio.h>
#include <string>
#include <vector>

using namespace std;

bool is_pow_2(int n) {
    return (n > 0 && ((n & (n - 1)) == 0));
}

```

```

unsigned int log_2(unsigned int n) {
    int log = 0;
    while (n >= 2) {
        log++;
        n /= 2;
    }
    return log;
}

vector<string> read_file(string path) {
    vector<string> func_list;
    ifstream f(path);
    if (!f.is_open()) {
        f.close();
        ofstream newfile(path, fstream::app);
        newfile.close();
        ifstream check(path);
        if (!check.is_open()) {
            check.close();
            cout << "\033[31m" "[ОШИБКА] Отказано в доступе к файлу \'' << path
<< "\'.'.\\n" "\033[30m";
            cout << "\n\nНажмите любую клавишу, чтобы выйти.";
            _getch();
            return {};
        }
        check.close();
        cout << "\033[32m" "Файл \'' << path << "\' создан.\n"
        "Пожалуйста, заполните его списком ДДФ в векторной форме (по одной
функции на строку).\n"
        "Затем снова запустите программу.\n" "\033[30m";
        cout << "\n\nНажмите любую клавишу, чтобы выйти.";
        _getch();
        return {};
    }

    int count = 0;
    string func;
    while (getline(f, func)) {
        if (!func.empty()) {
            count++;
            for (char ch : func) {
                if (ch != '0' && ch != '1') {
                    cout << "\033[31m" "[ОШИБКА] В " << count << "-й строке
найден недопустимый символ: \'' << ch << "\'\\n";
                    cout << "ДДФ в векторной форме может содержать только нули и
единицы.\n" "\033[30m";
                    cout << "\n\nНажмите любую клавишу, чтобы выйти.";
                    _getch();
                    return {};
                }
            }
            if (!is_pow_2(func.size())) {
                cout << "\033[31m" "[ОШИБКА] Длина вектор функции в " << count
<< "-й строке равна " << func.size() << ", а должна быть равна 2^n.\n"
"\033[30m";
                cout << "\n\nНажмите любую клавишу, чтобы выйти.";
                _getch();
                return {};
            }
            func_list.push_back(func);
        }
    }
    if (count == 0) {

```

```

        cout << "\033[31m" "[ОШИБКА] Файл \'' << path << "\' пуст.\n"
"\033[30m";
        cout << "\n\nНажмите любую клавишу, чтобы выйти.";
        _getch();
        return {};
    }
    f.close();
    return func_list;
}

string bin(int n, unsigned int len) {
    string b;
    while (n) {
        b.insert(0, to_string(n & 1));
        n >>= 1;
    }
    int fill_zeros = len - b.size();
    if (fill_zeros > 0) b.insert(0, fill_zeros, '0');
    return b;
}

vector<string> triangle(string func) {
    vector<string> trngl;
    trngl.push_back(func);
    string new_layer;
    while (func.size() > 1) {
        for (int ch = 0; ch < func.size() - 1; ch++) {
            bool a = bool(func[ch] - '0');
            bool b = bool(func[ch + 1] - '0');
            if (a != b) new_layer.push_back('1');
            else new_layer.push_back('0');
        }
        func = new_layer;
        trngl.push_back(func);
        new_layer.clear();
    }
    return trngl;
}

string anf(string func) {
    vector<string> trngl = triangle(func);
    string anf_str;
    unsigned int var_count = log_2(trngl[0].size());
    for (int i = 0; i < trngl.size(); i++) {
        bool left_number = bool(trngl[i][0] - '0');
        string truth_table_row = bin(i, var_count);
        if (left_number) {
            anf_str += " + ";
            if (i == 0) anf_str += "1";
            else {
                for (int j = 0; j < var_count; j++) {
                    if (truth_table_row[j] == '1') anf_str += "x" + to_string(j
+ 1);
                }
            }
        }
    }
    anf_str.erase(0, 3);
    if (anf_str.empty()) anf_str = "не существует";
    return anf_str;
}

int main()
{
    SetConsoleCP(1251);

```

```

SetConsoleOutputCP(1251);
system("color F0");

string filename_in = "input.txt";
string filename_out = "output.txt";
string cwd = _getcwd(NULL, 0);

vector<string> func_list = read_file(cwd + "\\\" + filename_in);
if (func_list.empty()) return 0;

ofstream out(cwd + "\\\" + filename_out);

string polynomial;
unsigned int n = 0;
for (auto it = func_list.begin(); it != func_list.end(); it++) {
    n++;
    polynomial = anf(*it);
    cout << "Функция " << n << ":\n\tВведённый вектор:\t" << *it <<
"\n\tПолином Жегалкина:\t" << polynomial << "\n\n";
    if (out.is_open()) {
        if (n != 1) out << '\n';
        out << polynomial;
    }
}
if (out.is_open()) cout << "\033[32m" "Результат(ы) записан(ы) в файл \'' <<
cwd + "\\\" + filename_out << "\'.\n" "\033[30m";
else cout << "\033[31m" "[ОШИБКА] Не удалось открыть файл \'' << cwd + "\\\"
+ filename_out << "\' для записи.\n" "\033[30m";
out.close();

cout << "\n\nНажмите любую клавишу, чтобы выйти.";
_getch();
}

```