

## Лабораторная работа №6

### Обработка двумерных массивов и контейнеров

ЗАДАНИЕ 1. Обработка двумерных массивов .....	2
ЗАДАНИЕ 2. Многоалфавитное шифрование/дешифрование с использованием алгоритма AES .....	10
Теоретические аспекты алгоритма AES.....	10
ЗАДАНИЕ 3. Методы решения систем линейных алгебраических уравнений.....	16
ЗАДАНИЕ 4. Логические задачи .....	25
ЗАДАНИЕ 5*. .....	27

**Цели и задачи работы:** изучение алгоритмов формирования и обработки двумерных массивов, программирование и отладка программ формирования и обработки массивов (статических и динамических) и контейнеров STL.

**Задание к работе:** Самостоятельно решить задачи в соответствии с индивидуальным вариантом на двух языках программирования.

#### Методика выполнения работы:

1. Разработать алгоритм решения задачи по индивидуальному заданию.
2. Написать и отладить программу решения задачи на двух языках (C++ и второй, по выбору).
3. Протестировать работу программы на различных исходных данных.

#### Перечень понятий к защите лабораторной работы 6.

Массив. Статический массив: определение, использование. Виды перебора элементов массива. Обработка элементов массива по два элемента.

Многомерные массивы. Хранение, описание, доступ к элементам. Пример инициализации. Передача параметров одномерного массива (виды). Передача параметров многомерного массива (виды).

Понятие указатель и динамическая память. Схема работы стек-куча при указателях. Оператор *new*. Инициализация объектов в куче. Контейнеры (динамика). Доступ к элементу в куче. Указатель на функцию. Указатель на объект. Указатель на *void*. Инициализация указателей (4 вида). Освобождение памяти. Операции с указателями. Арифметические операции с указателями. Ссылки. Взаимосвязь ссылки и указателя. Указатели и массивы. Динамические одномерные массивы. Динамические двумерные массивы: порядок создания и освобождения памяти. Контейнер *vector*: назначение, инициализация, принцип работы.

## Индивидуальные задания

### ЗАДАНИЕ 1. Обработка двумерных массивов<sup>1</sup>

#### Вариант 1

1. Определите и инициализируйте матрицу размерности  $(M \times N)$  случайными вещественными числами в диапазоне  $[0, 100]$ . Найдите и выведите средние арифметические значения элементов матрицы, а также каждой строки. Определите номер строки, у которой среднее арифметическое значение наибольшее.

2. Определите и инициализируйте квадратную матрицу порядка  $M$  ( $M > 5$ ) случайными целыми числами в диапазоне  $[-100, 100]$ . Отсортируйте чётные столбцы в порядке возрастания, а нечётные – в порядке убывания. Определите, в какой половине матрицы относительно главной диагонали, включая саму диагональ, больше положительных элементов.

3. Реализуйте клеточный автомат Джона Конвея на **ограниченной** плоскости по классическим правилам. Продемонстрируйте работу клеточного автомата на нескольких примерахдвигающихся фигур (фигур, у которых состояние повторяется, но со смещением\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

\* *Glider, Lightweight Spaceship, Middleweight Spaceship, Heavyweight Spaceship, Queen Bee Shuttle*

#### Вариант 2

1. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами в диапазоне  $[0, 50]$ . Определите, какое простое число встречается в матрице наибольшее количество раз.

2. Напишите программу, инициализирующую матрицу, как в примере. Размерность матрицы задаётся с клавиатуры. Модифицируйте вывод таким образом, чтобы вместо “0” в консоль выводилось “000”.

N = 5					N = 7						
000	103	102	101	100	000	105	104	103	102	101	100
100	000	102	101	100	100	000	104	103	102	101	100
100	101	000	101	100	100	101	000	103	102	101	100
100	101	102	000	100	100	101	102	000	102	101	100
100	101	102	103	000	100	101	102	103	000	101	100
100	101	102	103	000	100	101	102	103	104	000	100
100	101	102	103	000	100	101	102	103	104	105	000

3. В двумерном массиве размерностью  $M \times N$ , все элементы которого различны, требуется найти такие элементы, которые одновременно являются минимальными в своей строке и максимальными в своем столбце.

<sup>1</sup> Распределение вариантов по модулю 15

### Вариант 3

1. Определите и инициализируйте матрицу размера  $(M \times N)$  случайными целыми числами в диапазоне  $[10, 50]$ . Поменяйте в каждой строке местами максимальный и минимальный элементы. Определите и инициализируйте новую матрицу, первая строка которой содержит минимальные элементы, а вторая строка – максимальные.

2. Определите и инициализируйте матрицу размера  $(M \times N)$  случайными целыми числами в диапазоне  $[100, 200]$ . Найдите строку, в которой сумма цифр элементов наибольшая.

3. Реализуйте клеточный автомат Джона Конвея на **ограниченной** плоскости по классическим правилам. Продемонстрируйте работу клеточного автомата на нескольких примерах устойчивых фигур (фигур, которые остаются неизменными\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

\* Block, Beehive, Loaf, Boat

### Вариант 4

1. Определите и инициализируйте квадратную матрицу размерности  $N$  (где  $N > 6$ ) случайными целыми числами. Определите и инициализируйте новую матрицу, содержащую две строки: копии элементов диагоналей исходной матрицы.

2. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами. Замените в элементах диагоналей матрицы значения на разность наибольшего и второго наименьшего элементов матрицы.

3. Определите, по правилам ли заполнена доска для sudoku размером  $9 \times 9$ . Только заполненные ячейки должны быть проверены в соответствии со следующими правилами: каждая строка должна содержать цифры 1-9 без повторения, каждый столбец должен содержать цифры 1-9 без повторения, каждый из девяти вложенных блоков сетки размером  $3 \times 3$  должен содержать цифры 1-9 без повторения. Доска sudoku (частично заполненная) может быть валидной, но не обязательно разрешимой. Только заполненные ячейки должны быть проверены в соответствии с указанными правилами. Для упрощения достаточно реализовать для четырех квадратов, то есть  $6 \times 6$ .

Ввод						Вывод		
7		5						
	2		4			5		
6	4	3	5					1
5		4	7		6			
1	3	6	8		2	4	5	7
			4		4	9	1	6
2					3	1	7	5
		9			1		6	
						3		9

true

### Вариант 5

1. Определите и инициализируйте квадратную матрицу порядка  $N$  ( $N$  – чётное число) случайными целыми числами в диапазоне  $[100, 200]$ . Найдите и выведите сумму элементов строк, а затем столбцов матрицы.

2. Определите и инициализируйте матрицу размера  $M \times N$  случайными целыми числами в диапазоне  $[1000, 3000]$ . Найдите строку, в которой содержится самая длинная последовательность чисел, в которых нет цифр 5 и 7. Инициализируйте новый массив найденной последовательностью.

3. Реализуйте клеточный автомат Джона Конвея на замкнутой плоскости по классическим правилам. Продемонстрируйте работу клеточного автомата на примере «сорняка» («паразита») (фигуры, которая при столкновении с некоторыми фигурами дублируются\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

\* Replicators, Parasites

### Вариант 6

1. Определите и инициализируйте матрицу размерности  $M \times N$  (где  $M > 7$ ,  $N > 5$ ) случайными вещественными числами. Определите и инициализируйте массив произведения элементов столбцов матрицы.

2. Напишите программу, инициализирующую матрицу, как в примере. Количество строк матрицы задаётся с клавиатуры. Пример:

$N = 3$			$N = 5$				
			10				
10			12	14			
12	14		16	18	20		
16	18	20	22	24	26	28	
			30	32	34	36	38

3. Реализуйте клеточный автомат Джона Конвея на замкнутой плоскости по классическим правилам. Продемонстрируйте работу клеточного автомата на нескольких примерах периодических фигур (фигур, у которых состояние повторяется через некоторое число поколений\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

\* Blinker, Toad, Beacon, Pulsar, Pentadecathlon

### Вариант 7

1. Определите и инициализируйте матрицу размерности  $M \times N$  (где  $M > 7$ ,  $N > 5$ ) случайными целыми числами в диапазоне  $(-10, 10)$ . Найдите номер строки, содержащей наибольшее количество отрицательных чисел. Определите и инициализируйте новый массив найденной строкой.

2. Определите и инициализируйте квадратную матрицу порядка  $M$  ( $M > 5$ ) случайными целыми числами в диапазоне  $[10, 100]$ . Отсортируйте цифры в элементах обеих диагоналей в порядке убывания. Определите, сумма элементов какой из диагоналей наибольшая.

3. Дана доска для игры «морской бой» размера  $m \times n$ , где каждая ячейка представляет собой линкор "X" или пустое ".". Определите количество кораблей на доске. Корабли могут быть размещены только горизонтально или вертикально на доске. По крайней мере, ячейка разделяет два линкора (т.е. корабли не могут примыкать друг к другу).

Ввод					Вывод	
					2	
X	X			X		
				X		
				X		

### Вариант 8

1. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами в диапазоне  $[-50, 50]$ . Найдите в строках матрицы самую длинную возрастающую последовательность элементов. Инициализируйте новый массив данной последовательностью.

2. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами в диапазоне  $[100, 200]$ . Найдите и выведите среднее арифметическое чётных столбцов и среднее арифметическое нечётных строк. Если больше арифметическое чётных столбцов, то отсортируйте главную диагональ в порядке убывания, если меньше – в порядке возрастания.

3. Дана матрица символов и строка, содержащая слово. Напишите программу, определяющую, содержится ли заданное слово в матрице или не содержится. Слово может быть составлено из букв последовательно смежных ячеек, где соседние ячейки являются соседними по горизонтали или вертикали. Одна и та же буквенная ячейка не может использоваться более одного раза.

Ввод "JSPROGS",					Вывод	
Q	W	E	R	H	true	
J	S	P	E	S		
S	F	R	S	F		
A	D	O	G	S		
S	S	G	F	I		

### Вариант 9

1. Определите и инициализируйте матрицу размера  $M \times N$  случайными целыми числами в диапазоне  $[-10, 40]$ . Выведите номера строк, содержащих хотя бы три отрицательных элемента. Инициализируйте новый массив этими отрицательными элементами. Создайте новую матрицу, состоящую из найденных строк.

2. Определите и инициализируйте квадратную матрицу порядка  $N$  (где  $N$  чётное число) случайными целыми числами в диапазоне  $[10, 30]$ . Замените значения в левой нижней четверти матрицы на 0, в правой нижней части матрицы на 10, а в оставшихся четвертях поменяйте местами

3. Реализуйте клеточный автомат Джона Конвея на замкнутой плоскости по классическим правилам. Продемонстрируйте работу клеточного автомата на примере «ружья» (фигуры, у которой состояние повторяется, но дополнительно появляетсядвигающаяся фигура\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

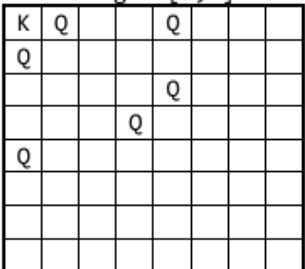
\* Gosper Glider Gun, Simkin Glider Gun, Twin Bees Shuttle

### Вариант 10

1. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами в диапазоне  $[0, 100]$ . Найдите в строках матрицы возрастающую последовательность элементов с минимальным начальным элементом. Инициализируйте новый массив найденной последовательностью.

2. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами в диапазоне  $[-50, 50]$ . Определите, сколько раз каждый уникальный элемент встречается в матрице.

3. На шахматной доске расположены несколько чёрных ферзей и один белый король. Матрица координат ферзей и пара координат короля задают их положение на шахматной доске. Определите координаты ферзей, которые атакуют короля. Программа должна автоматически заполнять поле фигурами случайным образом. Реализуйте возможность добавления на поле коней и определение координат тех, которые атакуют короля. \*

Ввод	Вывод
<pre>queens = [[0,1],[1,0],[4,0],[0,4],[3,3] ,[2,4]] king = [0,0]</pre> 	<pre>[[0,1],[1,0],[3,3]]</pre>

\* Королева не может бить короля, если на ее пути есть другая фигура.

### Вариант 11

1. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами в диапазоне  $[-50, 50]$ . Вычислите и выведите сумму всех неотрицательных элементов в чётных строках, стоящих на нечётных местах. Инициализируйте новый массив этими числами.

2. Определите и инициализируйте квадратную матрицу порядка  $N$  случайными целыми числами в диапазоне  $[100, 200]$ . Найдите сумму элементов строк матрицы. Определите и инициализируйте массив строкой со второй максимальной суммой элементов среди строк матрицы.

3. Реализуйте клеточный автомат Джона Конвея на ограниченной плоскости по классическим правилам. Продемонстрируйте работу клеточного автомата на примере «паровоза» (двигающиеся фигуры, которая оставляет за собой следы в виде устойчивых или периодических фигур\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

\* Switch Engine, Fumarole Puffer, Low-Flux Puffer, Space Rake

### Вариант 12

1. Определите и инициализируйте квадратную матрицу порядка  $N$  случайными целыми числами в диапазоне  $[-100, 100]$ . Найдите строку, содержащую наибольшее количество положительных чисел. Инициализируйте новый массив найденной строкой.

2. Определите и инициализируйте матрицу размерности  $M \times N$  случайными целыми числами в диапазоне  $[100, 150]$ . Определите число, которое встречается в матрице наибольшее количество раз. Если такого числа нет, то выведите соответствующее сообщение.

3. Реализуйте клеточный автомат Джона Конвея на замкнутой плоскости по классическим и оригинальным (своим) правилам. Продемонстрируйте работу клеточного автомата на нескольких примерахдвигающихся фигур (фигур, у которых состояние повторяется, но со смещением. Например, глайдер\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Понаблюдайте за изменениями в эволюции. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

\* *Glider, Lightweight Spaceship, Middleweight Spaceship, Heavyweight Spaceship, Queen Bee Shuttle*

### Вариант 13

1. Определите и инициализируйте квадратную матрицу порядка  $N$  ( $N$  - четное число) случайными вещественными числами в диапазоне  $[0, 100]$ . Определите и инициализируйте новую матрицу, состоящую из левой верхней четверти исходной матрицы.

2. Определите и инициализируйте матрицу размера  $M \times N$  случайными целыми числами в диапазоне  $[1000, 5000]$ . Найдите строку, в которой сумма цифр элементов наименьшая.

3. Реализуйте клеточный автомат Джона Конвея на ограниченной плоскости по классическим правилам. Пропридемонстрируйте работу клеточного автомата на примере пожирателя (устойчивой фигуры, которая может пережить столкновения с некоторыми двигающимися фигурами\*) и на развитии колоний клеток, сгенерированных в случайном порядке. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени. Таким образом, получается непрерывная анимация.

\* Eater

### Вариант 14

1. Определите и инициализируйте матрицу размера  $M \times N$  случайными вещественными числами в диапазоне  $[-50, 50]$ . Перезапишите строки в матрице в обратном порядке (дополнительный массив не используйте). Найдите в матрице второй минимальный и максимальный элементы.

2. Напишите программу, инициализирующую матрицу, как в примере. Количество строк матрицы задаётся с клавиатуры. Пример:

$N = 5$	$N = 9$
	100
	105 110
100	115 120 125
105 110	130 135 140 145
115 120 125	150 155 160 165 170
105 110	130 135 140 145
100	115 120 125 115
	105 110
	100

3. Реализуйте клеточный автомат Джона Конвея на ограниченной плоскости по классическим и оригинальным (своим) правилам. Пропридемонстрируйте работу клеточного автомата на примере устойчивой фигуры (фигуры, которые остаются неизменными\*), а затем на этой же фигуре с измененными правилами, и на развитии колоний клеток, сгенерированных в случайном порядке. Понаблюдайте за изменениями в эволюции. Реализовать работу графического отображения клеточного автомата можно таким образом, чтобы живые клетки изображались единицами, а мёртвые – нулями, либо живые – нулями, а мёртвые пробелами, либо другим возможным вариантом. Каждое новое поколение выводится на очищенное окно консоли через некоторый промежуток времени, например, 250 мс. Таким образом, получается непрерывная анимация.

\* Block, Beehive, Loaf, Boat



### Вариант 15

1. Определите и инициализируйте квадратную матрицу порядка  $N$  ( $N$  – чётное число) случайными целыми числами в диапазоне  $[-50, 50]$ . Модифицируйте матрицу таким образом, чтобы в чётных столбцах не было чисел из диапазона  $[-10, 10]$ .

2. Определите и инициализируйте матрицу размера  $M \times N$  случайными целыми числами в диапазоне  $[-100, 100]$ . Определите, каких чисел в столбце больше, если положительных, то отсортируйте первую половину столбца в порядке возрастания, если меньше, то возведите в квадрат все элементы в столбце, а если равно, то замените все числа на ноль.

3. Дана матрица размера  $M \times N$  из единиц и нулей. Определите, сколько квадратных подматриц составляют единицы, а также сколько и какого размера.

Ввод	Вывод
<pre>0 1 1 1 1 1 1 1 0 1 1 1</pre>	<pre>Всего найдено 15 подматриц: 10 подматриц со стороной 1, 4 подматрицы со стороной 2, 1 подматрица со стороной 3.</pre>

## **ЗАДАНИЕ 2. Многоалфавитное шифрование/дешифрование с использованием алгоритма AES<sup>2</sup>**

### **Вариант 1**

а) Реализовать систему шифрования AES128 (OFB) для преобразования исходного текста;

б) Реализовать систему шифрования AES128 (OFB) для преобразования зашифрованного текста в исходный.

### **Вариант 2**

а) Реализовать систему шифрования AES128 (CFB) для преобразования исходного текста;

б) Реализовать систему шифрования AES128 (CFB) для преобразования зашифрованного текста в исходный.

### **Вариант 3**

а) Реализовать систему шифрования AES128 (CBC) для преобразования исходного текста;

б) Реализовать систему шифрования AES128 (CBC) для преобразования зашифрованного текста в исходный.

Для всех вариантов, вывести все сгенерированные ключи, промежуточные результаты *State*, а также вектор инициализации, добавить генерацию случайного ключа и запись ключа в файл. Реализация должна работать с любым языком, как русским, так и английским.

## **Теоретические аспекты алгоритма AES**

AES (Advanced Encryption Standard) – симметричный блочный шифр, выбранный Национальным Институтом Стандартов и Технологий (NIST) США в 2001 году в качестве стандарта замены DES (Data Encryption Standard). Алгоритм разработан так, чтобы обеспечивать эффективную и безопасную криптографическую защиту [1].

### **Основные характеристики AES:**

- Размер блока: 128 бит;
- Размер ключа: 128, 192 или 256 бит (в зависимости от версии AES);
- Количество раундов шифрования: 10 раундов для 128-битного ключа, 12 раундов для 192-битного ключа и 14 раундов для 256-битного ключа.
- Есть различные виды реализации, такие как: ECB (Electronic Codebook) – каждый блок текста шифруется одним и тем же ключом; CBC – использует расширение ключа, использует IV (вектор инициализации для первого раунда обработки, состоит из 16 символов); CFB (Cipher Feedback) – шифрует с использованием ранее зашифрованного текста и так же используется IV; OFB – шифрует текст опираясь на незашифрованные данные.

### **Алгоритм состоит из нескольких основных этапов:**

- *AddRoundKey* (Добавление раундового ключа): для каждого байта блока данных применяется операция XOR с соответствующим байтом раундового ключа.
- *SubBytes* (Подстановка байтов): Каждый байт блока данных заменяется соответствующим значением из S-блока (таблицы подстановки).

---

<sup>2</sup> Распределение вариантов по модулю 3

- *ShiftRows* (Сдвиг строк): Байты в каждой строке блока данных циклически сдвигаются влево.
  - *MixColumns* (Смешивание столбцов): Каждый столбец блока данных подвергается линейному преобразованию.
- Эти этапы повторяются в зависимости от числа раундов, и процесс завершается финальными операциями, включая добавление последнего раундового ключа.

## Реализация

### 1. Начальная перестановка

Начальная перестановка представляет собой первый шаг в процессе шифрования. В AES выполняется при обработке блока данных в методах *encrypt\_block* и *decrypt\_block*. В частности, метод *BytesToMatrix* преобразует входные данные в матрицу, состоящую из 4-байтовых слов, что соответствует начальной перестановке данных.

### 2. Преобразование ключа

Преобразование ключа (*key expansion*) выполняется методом *expand\_key*. Этот метод принимает мастер-ключ и количество раундов шифрования, а затем расширяет ключ, генерируя подключи для каждого раунда. Расширение ключа осуществляется в соответствии с алгоритмом *Key Schedule*, используемым в AES.

### 3. Расширяющая перестановка

Расширяющая перестановка (*expand\_key*), как описано в предыдущем пункте, генерирует подключи для каждого раунда шифрования на основе мастер-ключа. Эти подключи затем используются в процессе шифрования для добавления раундовых ключей.

### 4. Подстановка с помощью S-блоков

Подстановка с использованием S-блоков (*sub\_bytes*) выполняется над состоянием матрицы данных в каждом раунде шифрования. Она применяется к каждому байту матрицы, заменяя его значением из S-блока.

### 5. Перестановка с помощью P-блока

Перестановка с использованием P-блока (*shift\_rows*) выполняет циклический сдвиг строк матрицы данных в различных направлениях. Этот шаг предназначен для обеспечения дополнительной диффузии данных в процессе шифрования.

### 6. Заключительная перестановка

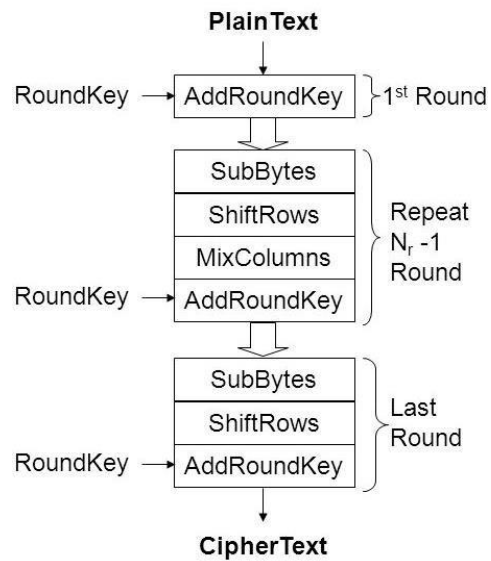
Заключительная перестановка (*final permutation*) выполняет окончательные преобразования данных перед завершением процесса шифрования.

#### Примечание:

Стоит не забывать о том, что текст должен быть кратен 16. В случае, если символов не хватает, стоит обработать это как ошибку или сделать заполнение недостающих символов при шифровании, и убирать их соответственно при расшифровании.

Схема для функций шифрования и расшифрования представлена на рисунке 2.1. Определения представлены в таблице 2.1.

### Шифрование



### Расшифрование

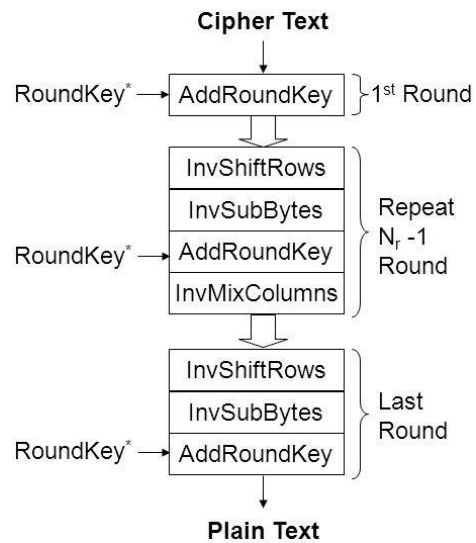


Рисунок 2.1 - Схема для функций шифрования и расшифрования

**RoundKey\*** – означает что используются те же раундовые ключи, что и при шифровании, только в обратном порядке

Таблица 2.1 – Определения алгоритма AES

Обозначение	Расшифровка / Значение
Block	Последовательность бит, из которых состоит input, output, State и Round Key. Также под Block можно понимать последовательность байтов.
Cipher key	Секретный криптографический ключ, который используется Key Expansion процедурой, чтобы произвести набор ключей для раундов (Round Keys); может быть представлен как прямоугольный массив байтов, имеющий четыре строки и $N_k$ колонок.
Ciphertext	Выходные данные алгоритма шифрования.
Key Expansion	Процедура генерации Round Keys из Cipher Key.
Round key	Round Keys получаются из Cipher Key использованием процедуры Key Expansion. Они применяются к State при шифровании и расшифровании.
State	Промежуточный результат шифрования, который может быть представлен как прямоугольный массив байтов, имеющий 4 строки и $N_b$ колонок.
S-Box	Нелинейная таблица замен, используемая в нескольких трансформациях замены байтов и в процедуре Key Expansion для взаимнооднозначной замены значения байта. Предварительно рассчитанный S-box можно увидеть ниже.
$N_b$	Число столбцов (32-битных слов), составляющих State. Для AES $N_b = 4$
$N_k$	Число 32-битных слов, составляющих шифроключ. Для AES $N_k = 4, 6$ , или
$N_r$	8 (в зависимости от длины ключа 16/24/32 символа)
Rcon[ ]	Массив, который состоит из битов 32-разрядного слова и является постоянным для данного раунда.

S\_box (рисунок 2.2):

```
Sbox = array{
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
    0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
    0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
    0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
    0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
    0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
    0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
    0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
    0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
    0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
    0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
    0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
    0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
    0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
    0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16
};
```

Рисунок 2.2 – Таблица подстановки S\_box

Обратный S-box для процедуры InvSubBytes (рисунок 2.3):

```
InvSbox = array{
    0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
    0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
    0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
    0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
    0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
    0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
    0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
    0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
    0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
    0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
    0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
    0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
    0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
    0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
    0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d
};
```

Рисунок 2.3 – Таблица подстановки Inv\_S\_box

**R\_con** – может отличаться, в зависимости от ЯП и реализации, поэтому стоит вычислять в отдельной функции

#### Список использованных источников

1. <https://www.cryptosoft.de/docs/Rijndael.pdf>
2. <https://www.geeksforgeeks.org/galois-fields-and-its-properties/> - для тех, кто хочет лучше узнать математическую часть и как работает поле Галуа ( $GF(2^8)$ ), которое активно используется в AES
3. <https://www.geeksforgeeks.org/advanced-encryption-standard-aes/>
4. [https://csrc.nist.gov/external/nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIP S.197.pdf](https://csrc.nist.gov/external/nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIP%20S.197.pdf)

### ЗАДАНИЕ 3. Методы решения систем линейных алгебраических уравнений<sup>3</sup>

Из таблицы 3.1 выбрать данные для системы линейных уравнений. Найти решение этой системы прямым и приближенным методами с точностью до  $\varepsilon=10^{-3}$ .

Четные варианты – решить систему уравнений методом Гаусса с выбором главного элемента.

Нечетные варианты – решить систему уравнений методом LU-разложения.

Исходная система линейных уравнений:

$$\begin{cases} Mx_1 - 0.04x_2 + 0.21x_3 - 18x_4 = -1.24, \\ 0.25x_1 - 1.23x_2 + Nx_3 - 0.09x_4 = P, \\ -0.21x_1 + Nx_2 + 0.8x_3 - 0.13x_4 = 2.56, \\ 0.15x_1 - 1.31x_2 + 0.06x_3 + Px_4 = M. \end{cases}$$

Таблица 3.1 – Данные для исходной системы линейных уравнений по вариантам

№ варианта	M	N	P	№ варианта	M	N	P
1	-1.19	-0.21	1.21	9	-0.77	0.16	1.12
2	0.89	0.12	-1.15	10	1.09	-0.16	0.84
3	1.08	0.22	-1.16	11	0.89	0.08	-1.21
4	-0.88	0.1	0.91	12	0.87	-0.19	1.08
5	1.25	-1.14	-1.09	13	-1.21	0.2	0.88
6	0.79	0.18	-0.86	14	0.93	0.07	-0.84
7	-1.13	0.14	0.87	15	-1.14	-0.17	0.95
8	0.91	-0.23	-1.04	16	1.08	0.22	-1.16

#### Примерный порядок выполнения работы

1. Составьте методику решения системы линейных уравнений прямым методом согласно варианту.

2. Составить программу решения системы уравнений, вывести результаты прямого и обратного хода метода.

3. Преобразовать систему к каноническому виду с выполнением условия сходимости итерационной последовательности.

4. Составьте методику решения системы уравнений любым итерационным методом.

5. Составьте программу решения системы уравнений с точностью до  $\varepsilon$ , выводящую результаты в таблицу:

N	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	ε <sub>n</sub>
...	...	...	...	...	...

6. Найдите все корни системы уравнений и выпишите их с верными знаками.

<sup>3</sup> Распределение вариантов по модулю 2



## Теоретическая часть

### 1. Постановка задачи

Рассмотрим линейную неоднородную задачу для систем линейных алгебраических уравнений, которая записывается в виде:

$$Ax = b \quad (1)$$

где  $A$  – действительная матрица,  $b$  – вектор столбец.  $X$  – вектор неизвестных, принадлежат  $R^n$  –  $n$ -мерному евклидовому пространству.

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \quad (2)$$

Требуется найти решение  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  из  $R^n$  системы (2), подстановка которого в (2) приводит к верному равенству  $Ax^* = b$ .

#### Замечание

Из линейной алгебры известно, что решение задачи (2) существует и единственно, если детерминант матрицы  $A$  отличен от нуля.

#### Особенности задачи:

- а) линейна и неоднородна;
- б) количество неизвестных равно количеству уравнений;
- в) количество  $n$  для некоторых практических задач велико;
- г) трудно найти обратную матрицу при больших  $n$ .

Характер задачи и точность получаемого решения в большой степени зависят от ее обусловленности, являющейся важным математическим понятием, влияющим на выбор метода решения.

Более строго обусловленность задачи характеризуется числом обусловленности  $\nu(A) = \|A\| \cdot \|A^{-1}\|$ , где  $\|A\|$  – норма матрицы  $A$ ,  $\|A^{-1}\|$  – норма обратной матрицы. Чем больше это число, тем хуже обусловленность системы. В качестве нормы  $A$  можно принять число, являющееся максимальным из сумм (по модулю) элементов строк этой матрицы.

**!!! Реализация хорошей или плохой обусловленности в задачах напрямую связана с численной устойчивостью и неустойчивостью.**

В численном анализе используются два класса численных методов решения систем линейных уравнений:

Прямые методы, позволяющие найти решение за определенное число операций. К прямым методам относятся: метод Гаусса и его модификации, метод LU – разложения и др.

Итерационные методы, основанные на использовании повторяющегося процесса и позволяющие получить решение в результате последовательных приближений. Операции, входящие в повторяющийся процесс, составляют итерацию. К итерационным методам относятся: метод простых итераций, метод Зейделя и др.

### 2. Прямые методы решения системы линейных уравнений

#### 1. Метод Гаусса

Метод Гаусса состоит в исключении слагаемых системы путем ее равносильного преобразования. Метод разбивается на две совокупности операций, которые разбиваются условно на прямой и обратный ход.

а) Прямой ход состоит в исключении элементов, расположенных ниже элементов, соответствующих главной диагонали матрицы  $A$ . Матрица  $A$  преобразуется к верхнетреугольному виду с единицами на главной диагонали.

б) Обратный ход, из матрицы  $A^*$  определяем последовательно  $x_n, \dots, x_1$ .

Надо решить систему алгебраических уравнений  $Ax = b$ :

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

...

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

Нужно преобразовать к треугольной матрице. Сделаем последовательно:

1 шаг:

$$x_1 = \frac{(-1)}{a_{11}} (a_{12}x_2 + \dots + a_{1n}x_n) + \frac{b_1}{a_{11}}$$

Исключаем из  $x_1$  всех уравнений по симплекс - правилу:

$$\begin{bmatrix} p & q \\ r & s \end{bmatrix} \rightarrow \begin{bmatrix} p & q \\ 0 & s - \frac{qr}{p} \end{bmatrix}$$

где  $p$  - ведущий элемент;  $q$  - элемент ведущей строки;  $r$  - элемент ведущего столбца;  $s$  - произвольный элемент.

Ведущий элемент выбирается по главной диагонали матрицы; из произвольного элемента вычитается произведение элементов ведущей строки на ведущий столбец, деленное на ведущий элемент.

$$\text{При } i=1 \text{ имеем } a_{ij}^{(1)} = a_{ij} - \frac{a_{1j}a_{1i}}{a_{11}} \quad b_i^{(1)} = b_i - \frac{a_{1i}b_1}{a_{11}}, \quad j=i, \dots, n$$

$$\text{В общем случае,} \quad a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}},$$

$$b_i^{(k)} = b_i^{(k-1)} - \frac{a_{ik}^{(k-1)}b_k^{(k-1)}}{a_{kk}^{(k-1)}},$$

где  $k$ -номер шага  $i=k+1, \dots, n-1, j=i, 2 \dots n$

(3)

Получим

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & \dots & \\ 0 & 0 & a_{33}^{(2)} & \vdots \\ 0 & 0 & 0 & \ddots \\ 0 & 0 & 0 & 0 & a_{nn}^{(n-1)} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2^{(1)} \\ \vdots \\ b_n^{(n-1)} \end{pmatrix}$$

Обратный ход, начиная с последнего уравнения, последовательно определяем  $x_n, x_{n-1}, \dots, x_1$ :

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$$

$$x_k = \frac{b_k^{(k-1)} - \sum_{j=k+1}^n a_{kj}^{(k-1)} x_j}{a_{kk}^{(k-1)}} \quad (4)$$

**Замечание!**

На основе прямого хода путем перемножения ведущих элементов вычисляется определитель матрицы  $A$ .

Изложенный метод имеет ограничение, связанное с тем, что ведущие элементы  $a_{11}, a_{22}, \dots, a_{nn}$  и т.д. должны быть отличны от нуля и не должны быть малыми по модулю, поскольку погрешности вычислений будут большими.

Таким образом, порядок последовательности исключения неизвестных может сильно сказаться на результатах расчетов. Уменьшить опасность такого рода позволяют модифицированные варианты метода Гаусса. Наиболее надежным является метод исключения с выбором главного элемента.

В методе Гаусса-Жордана (с выбором главного элемента) в качестве ведущих элементов выбираются максимальное по модулю  $a_{ii}$  путем перебора этих элементов по столбцу, соответствующему этому ведущему элементу или по всем столбцам.

Метод Гаусса с выбором главного элемента по столбцу

Перед исключением  $x_1$  отыскивается  $\max |a_{i1}|$  по  $i$ . Допустим, максимум соответствует  $i=i_0$ . Тогда первое уравнение в исходной системе меняем местами с  $i_0$  уравнением. После этого осуществляем первый шаг исключения. Затем перед исключением  $x_2$  из оставшихся уравнений отыскиваем  $\max |a_{i2}|$ , где  $2 \leq i \leq n$ , осуществляется соответствующая перестановка уравнений.

Можно показать, что условие диагонального преобладания остается справедливым после каждого шага исключений в процессе приведения матрицы к треугольному виду, т.е.

$$|a_{ii}^{(k)}| > \sum_{\substack{j=k \\ i \neq j}}^n |a_{ij}^{(k)}|, \text{ для всех } k=1, 2, \dots, n-1. \quad (5)$$

Это означает, что перед каждым исключением очередной неизвестной главный элемент будет находиться в «нужной позиции».

Рассмотренные модификации метода Гаусса позволяют, как правило, существенно уменьшить влияние погрешности округления на результаты расчетов.

Метод Гаусса с выбором главного элемента.

$$\begin{vmatrix} a_{11} & \dots & 0 & \dots & a_{1n} \\ & & \vdots & & \\ \vdots & & a_{pq} & & \vdots \\ & & \vdots & & \\ a_{n1} & \dots & 0 & \dots & a_{nn} \end{vmatrix}$$

При первом ходе производится выбор наибольшего по модулю элемента и выполняется перестановка строк или столбцов. Это повышает точность вычисления. Затем проводится преобразование элементов по мнемоническому правилу, т.е. запоминается главная строка, главный столбец обнуляется, получаем новую матрицу с меньшим числом строк и столбцов. Вновь определяем главный элемент. Преобразование проводим до тех пор, пока не останется строка из двух элементов, т.е. получаем треугольную матрицу, далее начинается обратный ход (4).

## 2. Метод LU-разложения

Метод опирается на возможность представления матрицы в виде произведения двух треугольных матриц:

$$A=LU.$$

$L$  - нижняя треугольная матрица;  $U$  - верхняя треугольная матрица.

Решение системы сводится к последовательному решению двух простых систем с треугольными матрицами. Если же диагональные элементы одной из матриц зафиксировать, то такое разложение единственно.

**Замечание.** Всякую квадратную матрицу  $A$ , имеющую отличные от нуля  $|a_{11}|, |a_{11} a_{12}|, |a_{21} a_{22}|, \dots$  главные миноры можно представить в виде LU-разложения, причем это разложение будет единственным. Если зафиксированы элементы матрицы  $L$ , то матрицы  $L$  и  $U$  имеют вид:

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & \dots & \dots & 1 \end{bmatrix}; U = \begin{bmatrix} u_{11} & u_{12} & \dots & \dots & u_{1n} \\ 0 & u_{22} & \dots & \dots & u_{2n} \\ 0 & 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & u_{nn} \end{bmatrix};$$

Где элементы вычисляются по формулам:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \quad (i \leq j), l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}}{u_{ij}} \quad (i > j).$$

Заметим, что в этих формулах не будет деления на нуль. Пользуясь этой теоремой, решение системы  $Ax = b$  упрощается. Имеем представление  $LUx = b$ . Обозначив  $Ux = y$ , получим  $Ly = b$ . Решение СЛАУ сводится к последовательному решению двух систем с треугольными матрицами. Поэтому их решения выписываются явно. Сперва находим решение системы  $Ly = b$ :

$$x_k = \frac{y_k - \sum_{j=k+1}^n u_{kj} x_j}{u_{kk}}, k = n, n-1, \dots, 1.$$

#### **Замечание!**

LU-метод решения СЛАУ – это тот же метод Гаусса, записанный в матричном представлении. Матрица  $U$  и вектор  $y$  есть не что иное, как полученные после прямого хода метода Гаусса треугольная матрица и вектор правой части. Проверка условия, что все главные миноры отличны от нуля

– довольно трудоемкий процесс, поэтому проверку можно выполнить по ходу вычислений. Если окажется, что на каком-то шаге  $u_{ii} = 0$ , то это будет означать, что не все главные миноры исходной матрицы были отличны от нуля и LU-метод здесь не работает. Если найдены матрицы  $L$  и  $U$ , то легко

вычислить определитель исходной матрицы  $\det A = \det L * \det U = \prod_{i=1}^n 1^{u_{ii}}$

### **3. Итерационные методы решения систем линейных уравнений**

Альтернативой прямым методам являются итерационные, основанные на многократном уточнении  $x(0)$  – приближенного решения задачи  $Ax=b$ . Верхним индексом в скобках обозначается номер итерации (совокупности повторяющихся действий).

#### **1. Метод трехточечной прогонки**

Общая постановка задачи.

Дана система линейных алгебраических уравнений с трехдиагональной матрицей с диагональным преобладанием.

$$a_i x_{i-1} - b_i x_i + c_i x_{i+1} = f_i, a_1 = c_n = 0, i = 1, 2, \dots, n \quad (8)$$

Требуется найти решения  $x = (x_1, x_2, \dots, x_n)$  системы.

Приводимый ниже алгоритм носит название метода трехточечной прогонки и является специальным случаем метода исключения Гаусса. Рассмотрим следующую линейную систему:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \quad (9)$$

Чтобы начать исключение, разделим первое уравнение этой системы на диагональный элемент  $b_1$  и используем обозначения  $p_1 = \frac{c_1}{b_1}$  и  $q_1 = \frac{d_1}{b_1}$ . Предположим, что мы исключили все нулевые поддиагональные элементы в первых  $i-1$  строках. В этом случае система преобразуется к виду:

$$\begin{bmatrix} 1 & p_1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & p_2 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ 0 & \dots & 0 & 1 & p_{i-1} & 0 & \dots & 0 \\ 0 & \dots & 0 & a_i & b_i & c_i & \dots & 0 \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & 0 & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{i-1} \\ x_i \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{i-1} \\ d_i \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \quad (10)$$

Теперь, чтобы исключить поддиагональный элемент  $a_i$  в  $i$ -й строке, умножим  $(i-1)$ -ю строку на  $a_i$  и вычтем ее из  $i$ -й строки. В результате  $i$ -я строка нашей системы преобразуется к виду

$$(b_i - a_i p_{i-1})x_i + c_i x_{i+1} = d_i - a_i q_{i-1}.$$

Чтобы получить единицу на главной диагонали матрицы, разделим  $i$ -ю строку на коэффициент  $(b_i - a_i p_{i-1})$ . В результате для элементов  $p_i$  и  $q_i$  в окончательном виде  $i$ -й строки получаем следующие формулы:

$$\begin{aligned} p_i &= \frac{c_i}{b_i - a_i p_{i-1}}, \quad i = 2, \dots, n-1, & p_1 &= \frac{c_1}{b_1}, \\ q_i &= \frac{d_i - a_i q_{i-1}}{b_i - a_i p_{i-1}}, \quad i = 2, \dots, n, & q_1 &= \frac{d_1}{b_1}. \end{aligned} \quad (11)$$

Продолжая исключение, получаем систему уравнений с двухдиагональной матрицей вида

$$\begin{bmatrix} 1 & p_1 & 0 & 0 & \dots & 0 \\ 0 & 1 & p_2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & 1 & p_{n-2} & 0 \\ 0 & \dots & 0 & 0 & 1 & p_{n-1} \\ 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{n-2} \\ q_{n-1} \\ q_n \end{bmatrix}. \quad (12)$$

Это позволяет записать рекуррентные формулы для вычисления неизвестных  $x_i$

$$\begin{aligned}x_n &= q_n, \\x_i &= p_i x_{i+1} + q_i, \quad i = n-1, \dots, 1.\end{aligned}\tag{13}$$

В виду опасности обращения знаменателя в ноль возникает вопрос об устойчивости решения, таким условием является диагональное преобладание в исходной матрице:  $|b_i| > |a_i| + |c_i|$  для всех  $i$ . В этом случае  $|p_i| < 1$

Док-во методом индукции: Пусть  $|p_{i-1}| < 1$ , тогда

$$|p_i| = \frac{|c_i|}{|b_i - p_{i-1}a_i|} \leq \frac{|c_i|}{\|b_i\| - \|p_{i-1}\| \|a_i\|} < \frac{|c_i|}{\|b_i\| - \|a_i\|} < \frac{|c_i|}{|c_i|} = 1.$$

Утверждение доказано. Из того, что  $|b_i - p_i a_i| > |c_i| \geq 0$ , то есть знаменатель в формулах отличен от нуля.

Отметим, что метод прогонки относится к экономичным методам. Экономичными называются методы, для которых число неизвестных требуемых арифметических операций пропорционально числу неизвестных. Этот метод требует порядка  $8n$  операций, метод Гаусса порядка  $n^3$  операций.

## 2. Метод простой итерации

Для решения систему линейных уравнений  $Ax=b$  приводим к каноническому виду:

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\&\dots \\a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n\end{aligned}\tag{14}$$

Выразим  $x_1, x_2, \dots, x_n$ :

$$\begin{aligned}x_1 &= \frac{b_1}{a_{11}} - \frac{a_{12}x_2}{a_{11}} - \dots - \frac{a_{1n}x_n}{a_{11}} \\&\dots \\x_n &= \frac{b_n}{a_{nn}} - \frac{a_{12}x_2}{a_{nn}} - \dots - \frac{a_{nn-1}x_{n-1}}{a_{nn}}\end{aligned}$$

Таким образом, получим  $x=Cx+f$ , где

$$C = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ \dots & 0 & \dots & \dots \\ \dots & \dots & 0 & \dots \\ -\frac{a_{12}}{a_{nn}} & \dots & -\frac{a_{nn-1}}{a_{nn}} & 0 \end{pmatrix}, \quad f = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \dots \\ \frac{b_n}{a_{nn}} \end{pmatrix}, \quad x^{(0)} = \begin{bmatrix} x_1^{(0)} \\ \vdots \\ x_n^{(0)} \end{bmatrix}.\tag{15}$$

Итерационный процесс запишется виде:

$$x^{(k+1)} = Cx^{(k)} + f\tag{16}$$

### Замечание!

Процесс называется параллельным итерированием, так как для вычисления  $(k+1)$ -го приближения всех неизвестных учитываются вычисленные ранее их  $k$ -е приближение. Начальное приближение  $x^{(0)}$  выбирается произвольно.

Возникают вопросы об условии сходимости, и какова погрешность. Ответ на вопрос о сходимости дают следующие две теоремы

Теорема 1. Для сходимости итераций (16) к решению системы (14) достаточно, чтобы в какой-либо норме выполнялось условие  $\|C\| \leq q < 1$ , тогда независимо от выбора  $x^{(0)}$  выполняется

$$\|x^{(k)} - x^*\| \leq q^k \|x^{(0)} - x^*\|, \text{ где } x^* - \text{точное решение (14).}$$

Доказательство:

Из (14) имеем  $x^* = Cx^* + f$ .

Вычитаем из (16)  $x(k) - x^* = C(x(k-1) - x^*)$ .

Получим оценку погрешности

$$\|x^{(k)} - x^*\| \leq \|P\| \cdot \|x^{(k-1)} - x^*\| \leq q \|x^{(k-1)} - x^*\| \leq \dots \leq q^k \|x^{(0)} - x^*\|$$

очевидно, что при  $q < 1$   $\lim_{k \rightarrow \infty} x^{(k)} = x^*$

**Замечание!**

Условие теоремы, как достаточное, является завышенным. Сходящийся процесс обладает свойством самоисправляемости.

Условие сходимости выполняется если в матрице  $A$  диагональные элементы преобладают. Иначе модули диагональных коэффициентов в каждом уравнении системы больше суммы модулей недиагональных коэффициентов. Чем меньше величина нормы матрицы  $C$ , тем быстрее сходится метод.

Теорема 2. Для сходимости итераций (16) к решению системы (14) необходимо и достаточно, чтобы все собственные значения матрицы  $C$  по абсолютной величине были меньше 1.

**Замечание!**

Хотя теорема 2 дает более общее условие сходимости метода простых итераций, однако ею воспользоваться сложнее, так как нужно вычислить границы собственных значений.

$$\begin{aligned} \sum_{j=1}^n |c_{ij}| &\leq \alpha < 1 & i = 1, 2, \dots, n \\ \sum_{i=1}^n |c_{ij}| &\leq \beta < 1 & j = 1, 2, \dots, n \end{aligned} \quad (17)$$

Для того, чтобы данный процесс сходиллся, необходимо, чтобы норма матрицы  $C$  была меньше 1 или выполнялись следующие условия:

$$\begin{aligned} |x_i - x_i^{(k)}| &\leq \frac{\alpha}{1 - \alpha} \max_{j=1, \dots, n} |x_j^{(k)} - x_j^{(k-1)}| \\ |x_j - x_j^{(k)}| &\leq \frac{\beta}{1 - \beta} \sum_{i=1}^n |x_i^{(k)} - x_i^{(k-1)}| \end{aligned} \quad (18)$$

Метод работает лучше, если диагональные элементы значительно превосходят остальные.

Метод имеет ряд преимуществ:

погрешность округления сказывается значительно меньше, чем в методе Гаусса;

метод самоисправляющийся, отдельные ошибки при определении очередного вектора могут рассматриваться как новый начальный вектор;

метод удобен в разреженных матрицах (в коэффициентах стоит много нулей);

процесс легко программируется.

Количество операций для выполнения этого метода приблизительно равно  $(2n^2)k$ , где  $k$  количество приближений. Если допустимая погрешность достигается при  $k < n/3$ , то метод итераций становится предпочтительнее метода Гаусса.

Кроме того, методы итераций могут оказаться предпочтительнее с точки зрения устойчивости вычислений.

**Методика решения задачи**

Преобразовать систему  $Ax = b$  к виду  $x = Cx + f$

Задать начальное приближение  $x^{(0)}$  и точность  $\varepsilon$ . Положить  $k = 0$ .

Вычислить приближение по формуле  $x^{(k+1)} = cx^{(k)} + f$

Если выполнено условие  $\|x^{(k+1)} - x^{(k)}\| < \varepsilon$ , то процесс завершить и положить  $x^* = x^{(k+1)}$ , иначе  $k=k+1$  перейти к пункту 3.

### 3. Метод Зейделя

Является модифицированным методом простой итерации, дает лучшую сходимость и экономит память. При нахождении  $(k+1)$ - приближения сразу же используются найденные значения  $k$  компонент приближения с меньшим номером.

$$\begin{aligned} x_1^{(k)} &= \frac{1}{a_{11}} (f_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \dots - a_{1n}x_n^{(k-1)}) \\ x_2^{(k)} &= \frac{1}{a_{22}} (f_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k-1)} - \dots - a_{2n}x_n^{(k-1)}) \\ x_m^{(k)} &= \frac{1}{a_{mm}} (f_m - a_{m1}x_1^{(k)} - \dots - a_{m,m-1}x_{m-1}^{(k)} - a_{m,m+1}x_{m+1}^{(k-1)} - \dots - a_{mn}x_n^{(k-1)}) \end{aligned} \quad (19)$$

#### **Замечание!**

1) Метод Зейделя гарантированно сходится:

а) выполняется условие диагонального преобладания, является достаточным условием, но не является необходимым;

б) для матриц симметричных и положительно определенных.

2) В одинаковых условиях метод Зейделя сходится примерно в два раза быстрее метода простых итераций. Метод Зейделя может сходиться, если расходится метод простых итераций и наоборот. Также является само- исправляющимся, удобным в пользовании на ЭВМ, экономит память.



## ЗАДАНИЕ 4. Логические задачи<sup>4</sup>

### 1. Шифрование

Носорожик, Галчонок, Тушканчик и Уточка придумали новый способ шифрования. Для каждой из букв задается шифр – целое число, которым она заменяется (причем у некоторых букв шифры могут совпадать). Чтобы зашифровать слово, каждую его букву заменяют соответствующим числом. Очевидно, что при таком способе шифрования некоторые слова могут иметь одинаковые шифры. Определите количество слов длины  $K$ , которые можно однозначно расшифровать, если нет слов другой длины.

Программа принимает на вход два разделенных пробелом целых числа  $N$  и  $K$ ,  $2 \leq N \leq 10$  – число букв в алфавите,  $2 \leq K \leq 6$  – длина слов. Далее  $N$  строк – шифры букв, содержащие от 1 до 5 цифр, каждая из которых находится в диапазоне от 1 до 9. В результате, программа должна вывести одно целое число – количество слов длины  $K$ , которые можно однозначно расшифровать.

Пример:

Ввод  
3 2  
1  
2  
3

Вывод  
9

### 2. Площадь

Городская площадь имеет размер  $n \times m$  и покрыта квадратной плиткой размером  $1 \times 1$ . При плановой замене плитки выяснилось, что новой плитки недостаточно для покрытия всей площади, поэтому было решено покрыть плиткой только дорожку по краю площади, а в центре площади разбить прямоугольную клумбу. При этом дорожка должна иметь одинаковую ширину по всем сторонам площади. Определите максимальную ширину дорожки, которую можно выложить из имеющихся плиток.

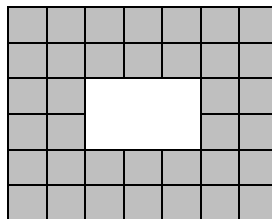
Первая и вторая строки входных данных содержат по одному числу  $n$  и  $m$  ( $3 \leq n \leq 2 \times 10^9$ ,  $3 \leq m \leq 2 \times 10^9$ ) – размеры площади. Третья строка содержит количество имеющихся плиток  $t$ ,  $1 \leq t < nm$ .

Программа должна вывести единственное число – максимальную ширину дорожки, которую можно выложить из имеющихся плиток.

Пример:

Ввод  
6  
7  
38

Вывод  
2



Как можно заметить в примере, площадь имеет размеры  $6 \times 7$ , из 38 плиток можно выложить дорожку шириной в 2 плитки остается две нетронутых плитки.

\* Обратите внимание, что значение  $t$  может быть больше, чем возможное значение 32-битной целочисленной переменной, поэтому необходимо использовать 64-битные числа.

<sup>4</sup> Распределение вариантов по модулю 4

### 3. Клетчатый удав

Мартышка, Слононок и Попугай двигают фишку по Удаву, на котором они по всей длине нарисовали полосу из клеток. Изначально фишка стоит на голове удава, где расположена первая клетка, и имеет нулевую скорость. Перед тем, как сделать ход, друзья могут либо оставить скорость фишки прежней, либо увеличить ее на 1, либо уменьшить ее на 1, если при этом скорость останется положительной. После этого фишка перемещается по клеткам Удава от головы к хвосту на количество клеток, равное текущей скорости. Зная номера клеток, в которых фишка должна оказаться, определите необходимое для этого число ходов

На вход, программа получает одно целое положительное число  $K$ ,  $K \leq 104$  – количество клеток, в которых должна побывать фишка. Во второй строке  $K$  целых чисел в порядке возрастания через пробел – номера клеток, в которых должна побывать фишка. Клетки нумеруются последовательно, начиная с 1. Известно, что номер клетки на кончике хвоста Удава не превышает  $10^4$ . В результате, должно получиться одно целое неотрицательное число – минимальное количество ходов, необходимое для посещения всех указанных клеток. Считается, что клетка посещена, если в ней начался или закончился какой-либо ход.

Пример:

Ввод	Вывод
3	5
4 8 11	

### 4. Максимум бананов

Мартышка, Слононок, Удав и Попугай выкладывают из кокосов и бананов разные числа в двоичной системе счисления, используя кокос вместо 0 и банан вместо 1. Слононок предложил усложнить задачу и выкладывать не само число, а два неотрицательных числа, сумма которых равна  $N$ , причем сделать это поочередно всеми возможными способами. Выведите те два числа, для выкладывания которых потребуется больше всего бананов. На вход, программа получает одно целое число  $N$ ,  $0 \leq N \leq 264 - 1$ . В результате, программа должна вывести в порядке возрастания через пробел два целых неотрицательных числа, сумма которых равна  $N$  и для выкладывания которых в двоичной системе счисления потребуется больше всего бананов. Если таких пар чисел несколько, выведите ту, разность чисел в которой наибольшая.

Пример:

Ввод	Вывод
5	2 3

## ЗАДАНИЕ 5\*.

### Доска с монетами

Вам дан прямоугольный щит размером  $N \times M$  клеток, в каждой из которых лежит монета либо орлом, либо решкой вверх. За один ход разрешается выбрать любую пару соседних по стороне монет (смежных по горизонтали или вертикали) и перевернуть обе сразу. Необходимо выяснить, каким минимальным числом таких ходов можно добиться того, чтобы на доске образовался «шахматный» рисунок (орлы и решки чередовались, причём существует ровно два способа такого расположения), либо же сообщить, что достичь этого невозможно.

Определите и инициализируйте данный прямоугольный щит случайными значениями (либо орел, либо решка).

#### Например

Задана следующая матрица

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

(1) ->

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

(2) ->

По итогу получаем исходную матрицу:

1	0	1
0	1	0

Следовательно, ответом данного примера будет 2.