Содержание

[Лабораторная работа №4 Обработка одномерных массивов 2](#_Toc5451063)

[Задание №4.1 «Вычисления в одномерных массивах» 2](#_Toc5451064)

[Задание №4.2 «Модификации одномерных массивов» 7](#_Toc5451065)

[Задание №4.3 «Криптографические приложения»\* 10](#_Toc5451066)

[**Лабораторная работа №5. Многофайловые проекты. Стандартная библиотека шаблонов STL.** 11](#_Toc5451067)

[**ЗАДАНИЕ 5.1 «Многофайловый проект ENUM»** 11](#_Toc5451068)

[**ЗАДАНИЕ 5.2 «Комбинация контейнеров»** 14](#_Toc5451069)

[Лабораторная работа №6 Обработка двумерных массивов 20](#_Toc5451070)

[**Задание №6.1 «Матричные преобразования»** 20](#_Toc5451071)

[Задание №6.2 «Многоалфавитное шифрование методом сложной замены (подстановки). Дешифрование текста». 28](#_Toc5451072)

[**Теоретическая часть** 28](#_Toc5451073)

[**Задание 6.3** Методы решения систем линейных алгебраических уравнений 31](#_Toc5451074)

[**Теоретическая часть** 31](#_Toc5451075)

# Лабораторная работа №4 Обработка одномерных массивов

**Цели и задачи работы**: изучение алгоритмов формирования и обработки одномерных массивов, программирование и отладка программ формирования и обработки массивов.

**Задание к работе**: Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Методика выполнения работы:**

1. Разработать алгоритм решения задачи по индивидуальному заданию.

2. Написать и отладить программу решения задачи.

3. Протестировать работу программы на различных исходных данных.

## Задание №4.1 «Вычисления в одномерных массивах»

Подгруппа 1 – варианты 1..15.

Подгруппа 2 – варианты 16..30.

**Вариант 1**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Сумму отрицательных элементов.

2. Произведение элементов, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

**Вариант 2**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Сумму положительных элементов.

2. Произведение элементов, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

**Вариант 3**

В одномерном массиве, состоящем из *n* целочисленных элементов, вычислить:

1. Произведение элементов с четными номерами.

2. Сумму элементов, расположенных между первым и последним нулевыми элементами. Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом - все отрицательные (элементы, равные нулю, считать положительными).

**Вариант 4**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Сумму элементов с нечетными номерами.

2. Сумму элементов, расположенных между первым и последним отрицательными элементами. Сжать массив, удалив из него все элементы, модуль которых не превышает единицу. Освободившиеся в конце массива элементы заполнить нулями.

**Вариант 5**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Максимальный элемент массива.

2. Сумму элементов, расположенных до последнего положительного элемента.

Сжать массив, удалив из него все элементы, модуль которых находится в интервале [а, b]. Освободившиеся в конце массива элементы заполнить нулями.

**Вариант 6**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Минимальный элемент массива.

2. Сумму элементов, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом - все остальные.

**Вариант 7**

В одномерном массиве, состоящем из *n* целочисленных элементов, вычислить:

1. Номер максимального элемента массива.

2. Произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине - элементы, стоявшие в четных позициях.

**Вариант 8**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Номер минимального элемента.

2. Сумму элементов, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом - все остальные.

**Вариант 9**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Максимальный по модулю элемент.

2. Сумму элементов, расположенных между первым и вторым положительными элементами. Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

**Вариант 10**

В одномерном массиве, состоящем из *n* целочисленных элементов, вычислить:

1. Минимальный по модулю элемент.

2. Сумму модулей элементов, расположенных после первого элемента, равного нулю. Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине - элементы, стоявшие в нечетных позициях.

**Вариант 11**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Номер минимального по модулю элемента.

2. Сумму модулей элементов, расположенных после первого отрицательного элемента. Сжать массив, удалив из него все элементы, величина которых находится в интервале [а, b]. Освободившиеся в конце массива элементы заполнить нулями.

**Вариант 12**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Номер максимального по модулю элемента.

2. Сумму элементов, расположенных после первого положительного элемента. Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [а, b], а потом — все остальные.

**Вариант 13**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Количество элементов массива, лежащих в диапазоне от А до В.

2. Сумму элементов, расположенных после максимального элемента. Упорядочить элементы массива по убыванию модулей.

**Вариант 14**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Количество элементов массива, равных нулю.

2. Сумму элементов, расположенных после минимального элемента. Упорядочить элементы массива по возрастанию модулей.

**Вариант 15**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Количество элементов массива, больших С.

2. Произведение элементов, расположенных после максимального по модулю элемента. Преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом - все положительные (элементы, равные нулю, считать положительными).

**Вариант 16**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Количество отрицательных элементов.

2. Сумму модулей элементов, расположенных после минимального по модулю элемента. Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

**Вариант 17**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Количество положительных элементов.

2. Сумму элементов, расположенных после последнего элемента, равного нулю. Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает единицу, а потом - все остальные.

**Вариант 18**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Количество элементов массива, меньших С.

2. Сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20 %, а потом - все остальные:

**Вариант 19**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Произведение отрицательных элементов.

2. Сумму положительных элементов, расположенных до максимального элемента. Изменить порядок следования элементов в массиве на обратный.

**Вариант 20**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Произведение положительных элементов.

2. Сумму элементов, расположенных до минимального элемента.

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

**Вариант 21**

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

1. Максимальный элемент массива.

2. Сумму остатков получаемых от деления элементов массива на максимальный элемент. Упорядочить элементы массива по возрастанию остатков, полученных от их деления на максимальный элемент.

**Вариант 22**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Среднее значение, как отношение суммы всех элементов на их количество

2. Сумму квадратов разностей между элементами массива и полученным средним значением

Упорядочить элементы массива по убыванию их разности со средним значением: вначале расположить элементы с положительной максимальной разностью, а затем остальные по убыванию разности.

**Вариант 23**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

1. Второй по величине элемент.

2. Сумму элементов, расположенных между максимальным и вторым по величине элементами.

Упорядочить элементы массива по возрастанию остатков, полученных от их деления на второй по величине элемент.

**Вариант 24**

В одномерном массиве, состоящем из *n* вещественных элементов:

1. Вычислить максимальный и минимальный элементы (Max и Min).

2. Выполнить вставку нового значения в элемент, который расположен в середине между максимальным и минимальным элементами. Вставку выполнить следующим образом: запросить новое значение, которое должно быть в диапазоне (Min ≤ a ≤ Max); переместить вправо на одну позицию элементы массива от точки вставки (последний элемент теряется); освободившемуся элементу присвоить новое значение.

Упорядочить по возрастанию модулей разности элементов массива и нового значения:

|ai - a|

**Вариант 25**

С использованием mt19937\_64 сформировать одномерный массив, состоящий из *n* элементов в котором элементы принимают случайное значение, и выполняется условие: a1 < a2 < a3 < … < an .

1. Для заданного числа y, такого, что a1 < y < an, определить такой k, чтобы ak < y < ak+1.

2. Вычислить сумму элементов массива, расположенных до элемента ak. Упорядочить по убыванию элементы, стоящие после элемента ak.

**Вариант 26**

С использованием mt19937\_64 сформировать одномерный массив, состоящий из n элементов в котором элементы принимают случайное значение, и выполняется условие: a1 > a2 > a3 > … > an .

1. Для заданного числа y, такого, что a1 < y < an, определить такой k, чтобы ak < y < ak+1.

2. Вычислить сумму элементов массива, расположенных после элемента ak. Упорядочить по убыванию элементы, стоящие до элемента ak.

**Вариант 27**

С использованием mt19937\_64 сформировать одномерный массив, состоящий из *n* элементов в котором элементы случайным образом принимают положительный или отрицательный знак и значение от -5 до 5. Для заданного числа y, такого, что amin < y < amax, вычислить:

1. Сумму элементов массива, значения модуля которых меньше y.

2. Произведение остальных элементов.

**Вариант 28**

С использованием mt19937\_64 сформировать одномерный массив, состоящий из *n* элементов в котором элементы случайным образом принимают положительный или отрицательный знак и значение от -10 до 10. Для заданного числа y, такого, что amin < y < amax, вычислить:

1. Произведение элементов массива, значения модуля которых больше y.

2. Сумму модулей остальных элементов.

**Вариант 29**

Координаты n точек заданы как элементы одномерного массива. Нечётные элементы - значения ординат, а чётные - абсцисс. Вычислить максимальное расстояние между ближайшими точками.

Вывести координаты всех точек, которые попадают в круг с координатами центра X0, Y0 и радиусом R.

Упорядочить положение точек по возрастанию их ординат.

**Вариант 30**

Координаты *n* точек заданы как элементы одномерного массива. Нечётные элементы - значения ординат, а чётные - абсцисс. Вычислить минимальное расстояние между ближайшими точками.

Вывести координаты всех точек, которые попадают в кольцо с координатами центра X0, Y0, внутренним радиусом R1 и внешним радиусом R2.

Упорядочить положение точек по возрастанию их абсцисс.

## Задание №4.2 «Модификации одномерных массивов»

**(вариант = номер в списке подгруппы)**

Общей частью для всех вариантов является считывание данных в массив. Сначала с клавиатуры вводится количество элементов n, после чего выделяется память под динамический массив заданного размера. После этого необходимо заполнить массив числами, при этом можно использовать два варианта: 1) все элементы массива вводятся вручную с клавиатуры; 2) все элементы массива задаются случайным образом, при этом необходимо вывести полученный массив на экран.

Для генерации чисел использовать mt19937\_64 (<https://ru.cppreference.com/w/cpp/numeric/random>).

Далее необходимо выполнить задание в соответствии с вариантом:

**Вариант 1**

С клавиатуры вводятся два числа a,b Необходимо отсортировать по неубыванию все элементы с номерами от *a*до *b*, после чего вывести на экран.

Пример.

Входные данные:

5

1 4 3 2 5

2 4

Выходные данные:

1 2 3 4 5

**Вариант 2**

Необходимо вывести весь массив чисел в обратном порядке (при этом перевернуть и сами числа, отбросив ведущие нули)

Пример.

Входные данные:

3

4327 8004 2300

Выходные данные:

32 4008 7234

**Вариант 3**

Необходимо в массиве поменять местами попарно соседние элементы, если в элементе с чётным номером количество десятков больше, чем количество сотен. Если количество элементов нечётное, последний элемент не рассматривается. Вывести на экран полученный результат.

Пример.

Входные данные:

5

231 543 434 198 57

Выходные данные:

231 543 198 434 57

**Вариант 4**

В каждом элементе массива требуется поменять местами первый и последний разряд числа, после чего отсортировать их по невозрастанию и вывести полученный массив на экран.

Пример.

Входные данные:

4

4009 132 908 771

Выходные данные:

9004 809 231 177

**Вариант 5**

Создать новый массив, в который поместить только те элементы исходного массива, которые являются простыми числами, после чего отсортировать полученный массив по возрастанию и вывести на экран.

Пример.

Входные данные:

6

122 29 13 19 77 7

Выходные данные:

7 13 19 29

**Вариант 6**

Необходимо посчитать в массиве количество различных элементов и вывести полученное число на экран.

Пример.

Входные данные:

10

1 4 9 2 4 5 8 3 2 1

Выходные данные:

7

**Вариант 7**

Будем считать, что введённый массив – это рост студентов в группе. Представим, что все эти студенты пришли на урок физкультуры и встали в шеренгу по росту (человек с меньшим ростом имеет больший номер). Однако один студент опоздал, и теперь ему необходимо определить под каким номером он будет стоять в шеренге. После ввода всех n элементов массива вводится ещё одно число x – рост опоздавшего студента. Если есть студенты одного роста с опоздавшим, то он становится в шеренгу после них. Вывести на экран найденный номер (нумерация начинается с 1).

Пример.

Входные данные:

7

188 172 163 192 166 169 178

172

Выходные данные:

5

**Вариант 8**

В этом варианте требуется вывести сначала все неотрицательные элементы массива, отсортировав их по возрастанию, после чего вывести все отрицательные элементы, отсортировав их по возрастанию модуля.

Пример.

Входные данные:

6

-13 8 0 23 -7 -44

Выходные данные:

0 8 23 -7 -13 -44

**Вариант 9**

Требуется отсортировать все элементы с чётными номерами по возрастанию, а все элементы с нечётными номерами по убыванию (нумерация начинается с 1). После чего вывести полученный массив на экран

Пример.

Входные данные:

9

87 324 95 6 234 23 401 3 87

Выходные данные:

401 3 234 6 95 23 87 324 87

**Вариант 10**

Необходимо найти максимальный и минимальный элементы в массиве (если таких несколько, то выбрать любой) и поменять их местами, после чего вывести полученный массив

Пример.

Входные данные:

6

23 1 87 5 2 99

Выходные данные:

23 99 87 5 2 1

## Задание №4.3 «Криптографические приложения»\*

Выполнить программную реализацию метода «шаг младенца, шаг великана» и алгоритма исчисления порядка и решить с помощью компьютера следующие уравнения:

1. 2*x* mod 30203 = 24322;
2. 2*x* mod 30323 = 21740;
3. 2*x* mod 30539 = 28620;
4. 2*x* mod 30803 = 16190;
5. 2*x* mod 30203 = 24322.

**Лабораторная работа №5.  
Многофайловые проекты. Стандартная библиотека шаблонов STL.**

**Цели и задачи работы**: изучение стандартной библиотеки шаблонов (STL) языка С++, программирование и отладка программ формирования и обработки контейнеров, комбинации контейнеров.

**Задание к работе**: Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Методика выполнения работы:**

1. Разработать алгоритм решения задачи по индивидуальному заданию.

2. Написать и отладить программу решения задачи.

3. Протестировать работу программы на различных исходных данных.

**Замечание**: Нумерация для пользователей с единицы. Использовать только контейнеры STL и циклы Range-based for loop.

**Информационное обеспечение:**

<https://ru.cppreference.com/w/cpp/container>

<http://www.cplusplus.com/reference/stl/>

**ЗАДАНИЕ 5.1 «Многофайловый проект ENUM»**

Необходимо реализовать систему хранения и обработки информации по заданию:

|  |  |
| --- | --- |
| **Вариант** | Задание |
| **1** | График движения поездов |
| **2** | График движения трамваев г. Новосибирска |
| **3** | График движения самолетов |
| **4** | График движения троллейбусов |

Реализовать многофаловый проект, предусматривающий обработку запросов:

|  |  |  |
| --- | --- | --- |
| **Вариант** | Задание | Расшифровка |
| **1** | CREATE\_TRAIN train town1 town2 town3 …. townn | Создание поезда с именем train, который проходит через города town1 town2 town3 …. townn |
| TRAINS\_FOR\_TOWN town | Вывод всех поездов, которые проходят через город town |
| TOWNS\_FOR\_TRAIN train | Вывод всех городов, которые проезжает поезд с именем train. Для каждого города прописать, какие поезда проезжают этот город (не включая train) |
| TRAINS | Отобразить все поезда с указанием остановок |
| **2** | CREATE\_TRAM tram stop1 …. stopn | Создание трамвая с именем tram, который проходит через остановки |
| TRAMS\_IN\_STOP stop | Вывод всех трамваев, которые проходят через конкретную остановку |
| STOPS\_IN\_TRAM tram | Вывод всех остановок, которые проезжает трамвай с именем tram. Для каждой остановки прописать, какие трамваи идут через эту остановку (не включая tram) |
| TRAMS | Отобразить все трамваи с указанием остановок |
| **3** | CREATE\_PLANE plane town1 town2 town3 …. townn | Создание самолета с именем plane, который пролетает через города town1 town2 town3 …. townn |
| PLANES\_FOR\_TOWN town | Вывод всех самолетов, которые пролетают через город town |
| TOWNS\_FOR\_PLANE plain | Вывод всех городов, которые пролетает самолет с именем plane. Для каждого города прописать, какие самолеты делают остановку в этом городе (не включая plain) |
| PLANES | Отобразить все самолеты с указанием городов-остановок |
| **4** | CREATE\_TRL trl stop1 …. stopn | Создание троллейбуса с именем trl, который проходит через остановки |
| TRL\_IN\_STOP stop | Вывод всех троллейбусов, которые проходят через конкретную остановку |
| STOPS\_IN\_TRL trl | Вывод всех остановок, которые проезжает троллейбус с именем trl. Для каждой остановки прописать, какие троллейбусы идут через эту остановку (не включая trl) |
| TRLS | Отобразить все троллейбусы с указанием остановок |

В программе предусмотреть

Команды хранить в классе enum:

*Например,*

**enum** **class** Type {

CREATE\_TRAIN,

TRAINS\_FOR\_TOWN,

TOWNS\_FOR\_TRAIN,

TRAINS

};

Предусмотреть создание многофайлового проекта: основной файл cpp, header+source для реализации действий (4 команды = 4 функции).

В отчет вставить осмысленные примеры остановок-городов.

Пример:

*Трамваи:*

**Input**

TRAMS

TRAMS\_IN\_STOP Marksa

STOPS\_IN\_TRAM 18

CREATE\_TRAM 18 4 Student Marksa TVset Cosmos

CREATE\_TRAM 15 4 Student Cosmos TVset Titova

TRAMS\_IN\_STOP Student

CREATE\_TRAM 666 6 Student Gorskyi NGTU NGTUMAIN IKEA Beach

CREATE\_TRAM 999 4 NGTU SibGUTI NSU Forest

STOPS\_IN\_TRAM 999

TRAMS

**Output**

Trams is absent

Stops is absent

Trams is absent

18 15

Stop NGTU: 666

Stop SibGUTI: 0

Stop NSU: 0

Stop Forest: 0

TRAM 18: Student Marksa TVset Cosmos

TRAM 15: Student Cosmos TVset Titova

TRAM 666: Student Gorskyi NGTU NGTUMAIN IKEA Beach

TRAM 999: NGTU SibGUTI NSU Forest

**ЗАДАНИЕ 5.2 «Комбинация контейнеров»**

**Вариант 1**

Реализовать автоматизированную систему:

|  |  |  |
| --- | --- | --- |
| Запрос | Расшифровка | Комментарий |
| FRIENDS i j | Записывает *i j* как друзей | - |
| COUNT i | Подсчет количества друзей *i* | Результат - Число |
| QUESTION i j | *i* дружат с *j*? | YES/NO |

Input:

1) N – количество запросов

2) Nзапросов

Output:

Обработать COUNT и QUESTION

**придумать минимум 5 тестов в разными значениями (в том числе, повторяющимися)**

Пример

**Input**

8

FRIENDS Peter Goward

COUNT Sally

FRIENDS Goward Sally

COUNT Goward

COUNT Peter

QUESTION Goward Peter

QUESTION Peter Sally

QUESTION Jenny Jastin

**Output**

0

2

1

YES

NO

NO

**Вариант 2**

Необходимо реализовать справочник регионов России.

На вход программе поступают следующие запросы:

|  |  |  |
| --- | --- | --- |
| Запрос | Расшифровка | Output |
| CHANGE ***region*** ***new\_center*** | Создание региона ***region*** с административным центром ***new\_center***. | New region ***region*** with administrative center ***new\_center*** |
| Переименование названия административного (old\_center) центра региона ***region*** в ***new\_center***. | Region ***region*** has changed its administrative center from old\_center to ***new\_center*** |
| RENAME ***old\_region new\_region*** | переименование региона со старым названием ***old\_region*** в регион с новым названием ***new\_region*** | ***old\_region*** has been renamed to ***new\_region*** |
| ABOUT ***region*** | вывод административного центра введенного региона ***region*** | ***region*** has administrative center ……… |
| ALL | вывод всех административных центров | *Region - Center* |

В случае ошибки (переименование несуществующего региона, переименование в старое значение и т.п.) вывести в поток ошибок «Incorrect»

Input:

1) N – количество запросов

2) N запросов

Output:

Обработать CHANGE, RENAME, ABOUT, ALL.

**придумать минимум 5 тестов в разных значениях (в том числе, повторяющихся).**

Пример

**Input**

**6**

**CHANGE Sibir Novo-nikolaevksk**

**RENAME Sibir SibirSFO**

**CHANGE Sibir Novosibirsk**

**ABOUT SibirSFO**

**ABOUT Sibir**

**ALL**

**Output**

New region Sibir with administrative center **Novo-nikolaevksk**

Sibir has been renamed to SibirSFO

Region SibirSFO has changed its administrative center from **Novo-nikolaevksk** to **Novosibirsk**

SibirSFO has administrative center **Novosibirsk**

Incorrect

SibirSFO - Novosibirsk

**Вариант 3**

У каждого студента есть расписание занятий. Посещать занятия необходимо в конкретный день определенного месяца. Необходимо автоматизировать работу расписания за счет обработки операций:

|  |  |  |
| --- | --- | --- |
| Запрос | Расшифровка | Output |
| СLASS i s | Установить дисциплину *s* в день *i* текущего месяца | - |
| NEXT | Смена месяца на следующий. | - |
| VIEW *i* | Организовать вывод всех дня *i*. При отсутствии пар – выдать соответствующее сообщение | In ***i*** day ***N*** classes in university: ***class 1,…,class N.*** |
| In ***i*** day We are free! |

INPUT: N – число операций ввода и затем эти операции (NEW, NEXT, VIEW).

Замечание, пары сохраняются при смене месяцев строго по дням и могут повторяться. Пропавшие в связи с переходом на новый календарный месяц пары никуда не пропадают, а переходят на предпоследний день месяца.

Нумерация для пользователей с единицы.

Пример:

|  |  |
| --- | --- |
| 12  CLASS 5 INFORMATICA  CLASS 31 YAP  CLASS 30 PHISICS  NEXT  VIEW 5  VIEW 27  NEXT  VIEW 31  VIEW 30  VIEW 27  CLASS 27 ENGLISH  VIEW 27 | In  In ***5*** day ***1*** classes in university: INFORMATICA  In 27day 2 classes in university: PHISICS, YAP  In 31day We are free!  In 30day We are free!  In 27day 2 classes in university: PHISICS, YAP  In 27day 3 classes in university: PHICICS, YAP, ENGLISH |
| 3  CLASS 4 A  CLASS 4 A  VIEW 4 | In 4day ***1*** classes in university: A |

**Вариант 4**

Отчисление студента в процессе обучения – процесс весьма трудоемкий для руководства ВУЗа. Представим поток факультета АВТФ.

Будем считать, что студенты находятся в списке учащихся, и ежегодно данный список пополняется. Причем новые студенты попадают в конец списка всегда. В зависимости от ситуации в долгами, некоторые студенты могут быть кандидатами на отчисления. Но при ликвидации задолженностей могут и уйти из списка на отчисление.

Необходимо реализовать обработку следующих запросов:

|  |  |  |
| --- | --- | --- |
| Запрос | Расшифровка | Output |
| NEW\_STUDENTS ***number*** | Добавить в конец очереди студентов в количестве ***number*** | >0: Welcome ***number*** clever students!  <0: GoodBye ***number*** clever students! |
| SUSPICIOUS ***number\_student*** | Студент с порядковым номером ***number\_student*** является крайне подозрительным и входит в топ-лист на отчисление | The suspected student ***number\_student*** |
| IMMORTAL ***number\_student*** | Студент с порядковым номером ***number\_student*** является неприкасаемым и из топ-листа на отчисление уходит. Такого студента никто и ничто не может отчислить | Student ***number\_student*** is immortal! |
| TOP-LIST | Вывод отсортированного списка студентов, входящих в топ-лист на отчисление | List of students for expulsion: Student 1,  ….., Student *N* |
| SCOUNT | Вывод количества студентов, входящих в топ-лист на отчисление | List of students for expulsion consists of ***N*** students |

В случае ошибки (удаление числа студентов, превышающих их число в текущий момент, ввод данных, выходящих за диапазон) вывести в поток ошибок «Incorrect»

**Input**

**6**

**NEW\_STUDENTS 20**

**SUSPICIOUS 10**

**NEW\_STUDENTS 5**

**SUSPICIOUS 15**

**IMMORTIAL 10**

**TOP-LIST**

**SCOUNT**

**Output**

Welcome 20 clever students!

The suspected student 10

Welcome 5 clever students!

The suspected student 15

Student 10 is immortal!

List of students for expulsion: Student 15

List of students for expulsion consists of 1 students

# Лабораторная работа №6 Обработка двумерных массивов

**Цели и задачи работы**: изучение алгоритмов формирования и обработки двумерных массивов, программирование и отладка программ формирования и обработки массивов.

**Задание к работе**: Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Методика выполнения работы:**

1. Разработать алгоритм решения задачи по индивидуальному заданию.

2. Написать и отладить программу решения задачи.

3. Протестировать работу программы на различных исходных данных.

**Задание №6.1 «Матричные преобразования»**

Задание:

1. Реализовать вычисления через динамические матрицы.

2. Вывести адрес расположения строк динамического массива в памяти.

3. Реализовать вычисления через vector<vector>. Циклы Range-based for loop.

**Вариант 1**

1. Даны целые положительные числа *M* и *N*. Сформировать целочисленную матрицу размера *M×N*, у которой все элементы *i*-й строки имеют значение 1010*‧i* (*i*= 1, . . ., *M*).

2. Дана матрица размера *M×N*. Поменять местами строки, содержащие минимальный и максимальный элементы матрицы.

3. Вводится число n. Необходимо создать матрицу размером n×n и заполнить её случайными числами в диапазоне [1,5]. После чего необходимо проверить, является ли матрица симметричной относительно главной и побочной диагоналей. (Для каждой диагонали написать в отдельной строке YES или NO)

Пример. Входные данные:

3

1 0 2

0 3 3

2 3 1

Выходные данные:

YES

NO

**Вариант 2**

1. Даны целые положительные числа *M* и *N*. Сформировать целочисленную матрицу размера *M×N*, у которой все элементы *j*-го столбца имеют значение 105‧*j*(*j*= 1, . . ., *N*).

2. Дана матрица размера M× N. Поменять местами столбцы, содержащие минимальный и максимальный элементы матрицы.

3. Даны два числа n и m. Создать матрицу размером n×m и заполнить случайными числами в диапазоне [-10,10]. Необходимо найти в каждой строке максимальный по модулю элемент и отсортировать строки матрицы по возрастанию этого значения.

Пример. Входные данные:

3 3

9 9 3

3 3 -8

-4 0 7

Выходные данные:

-4 0 7

3 3 -8

9 9 3

**Вариант 3**

1. Даны целые положительные числа *M*, *N* и набор из *M* чисел. Сформировать матрицу размера *M×N*, у которой в каждом столбце содержатся все числа из исходного набора (в том же порядке).

2. Дана матрица размера *M×N*. Поменять местами столбец с номером 1 и последний из столбцов, содержащих только положительные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

3. Задаётся число строк двумерного массива n. После чего следующие n чисел задают размеры каждой строки. Необходимо создать массив с данными размерами, отсортировать его по возрастанию размера строки и заполнить неповторяющимися числами по убыванию так, чтобы последнее число было равно 1.

Пример. Входные данные:

4

3 5 2 2

Выходные данные:

12 11

10 9

8 7 6

5 4 3 2 1

**Вариант 4**

1. Даны целые положительные числа *M, N* и набор из *N* чисел. Сформировать матрицу размера *M×N*, у которой в каждой строке содержатся все числа из исходного набора (в том же порядке).

2. Дана матрица размера M× N. Поменять местами столбец с номером N и первый из столбцов, содержащих только отрицательные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

3. Дано число n. Создать и вывести на экран матрицу размером n×n, заполненную случайными числами в диапазоне [-10,10]. После чего посчитать сумму элементов на каждой диагонали матрицы и вывести эти суммы на экран, отсортировав их по убыванию.

Пример. Входные данные:

4

Выходные данные:

1 -2 8 2

-3 -1 0 5

5 2 2 9

-7 3 9 1

13 8 8 7 3 2 -7

**Вариант 5**

1. Даны целые положительные числа *M, N*, число *D* и набор из *M* чисел. Сформировать матрицу размера *M×N*, у которой первый столбец совпадает с исходным набором чисел, а элементы каждого следующего столбца равны сумме соответствующего элемента предыдущего столбца и числа *D* (в результате каждая строка матрицы будет содержать элементы арифметической прогрессии).

2. Дана матрица размера *M×N* (*M*— четное число). Поменять местами верхнюю и нижнюю половины матрицы.

3. Даны два числа nи m. Создать и вывести на экран матрицу размером n×m и заполнить случайными числами в диапазоне [-100, 100]. После чего отсортировать четные столбцы матрицы по возрастанию, а нечётные по убыванию.

Пример. Входные данные:

3 4

Выходные данные:

-23 4 91 0

42 -93 1 18

19 -77 45 61

42 4 91 61

19 -77 45 18

-23 -93 1 0

**Вариант 6**

1. Даны целые положительные числа *M, N*, число *D* и набор из *N* чисел. Сформировать матрицу размера *M×N*, у которой первая строка совпадает с исходным набором чисел, а элементы каждой следующей строки равны соответствующему элементу предыдущей строки, умноженному на *D* (в результате каждый столбец матрицы будет содержать элементы геометрической прогрессии).

2. Дана матрица размера *M×N* (*N* — четное число). Поменять местами левую и правую половины матрицы.

3. Дано нечётное число n>1. Необходимо создать матрицу размера n×n и заполнить по спирали возрастающими числами, начиная с 1.

Пример. Входные данные:

5

Выходные данные:

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

**Вариант 7**

1. Дана матрица размера *M×N* и целое число *K* (1 ≤ *K* ≤ *M*). Вывести элементы *K*-й строки данной матрицы.

2. Дана матрица размера *M×N* (*M* и *N*— четные числа). Поменять местами левую верхнюю и правую нижнюю четверти матрицы.

3. Даны числа n и m. После чего задаётся матрица размером n×m из 0 и 1. Будем считать, что эта матрица задаёт карту лабиринта, при этом 0 является пустым пространством, а 1 стеной. В данном лабиринте будем считать, что точкой входа является клетка с координатами (1,1), а точкой выхода (n,m). Необходимо определить, возможно ли из точки входа добраться до точки выхода, если можно двигаться только в четырёх возможных направлениях (север, юг, запад, восток) и перемещаться можно только в свободную клетку.

Пример. Входные данные:

4 5

0 0 0 1 1

0 0 1 1 0

0 1 1 0 0

0 0 0 0 0

Выходные данные:

YES

**Вариант 8**

1. Дана матрица размера *M×N* и целое число *K* (1 ≤ *K* ≤ *N*). Вывести элементы *K*-го столбца данной матрицы.

2. Дана матрица размера *M×N* (*M* и *N*— четные числа). Поменять местами левую нижнюю и правую верхнюю четверти матрицы.

3. Дано число n. Задаётся верхняя треугольная матрица размером n×(n+1) (все элементы ниже главной диагонали равны 0). Необходимо решить СЛАУ, которая задана с помощью этой матрицы, и вывести значения всех x.

Пример. Входные данные:

3

1 -2 1 -3

0 2 -3 2

0 0 4 8

Выходные данные:

3 4 2

**Вариант 9**

1. Дана матрица размера *M×N*. Вывести ее элементы, расположенные в строках с четными номерами (2, 4, . . .). Вывод элементов производить по строкам, условный оператор не использовать.

2. Дана матрица размера *M×N*. Зеркально отразить ее элементы относительно горизонтальной оси симметрии матрицы (при этом поменяются местами строки с номерами 1 и *M*, 2 и *M*−1 и т.д.).

3. Даны числа nи m. Необходимо создать матрицу размером n×m, заполнить её случайными числами в диапазоне от [-10,10] и вывести на экран. После чего необходимо в каждом столбце найти минимальный элемент и переупорядочить столбцы матрицы по возрастанию этой величины.

Пример. Входные данные:

3 4

Выходные данные:

-4 6 0 7

8 -9 -3 1

0 2 5 -5

6 7 -4 0

-9 1 8 -3

2 -5 0 5

**Вариант 10**

1. Дана матрица размера *M×N*. Вывести ее элементы, расположенные в столбцах с нечетными номерами (1, 3, . . .). Вывод элементов производить по столбцам, условный оператор не использовать.

2. Дана матрица размера *M×N*. Зеркально отразить ее элементы относительно вертикальной оси симметрии матрицы (при этом поменяются местами столбцы с номерами 1 и *N*, 2 и *N*− 1 и т.д.).

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

**Вариант 11**

1. Дана матрица размера *M×N*. Вывести ее элементы в следующем порядке: первая строка слева направо, вторая строка справа налево, третья строка слева направо, четвертая строка справа налево и т.д.

2. Дана матрица размера *M×N* и целое число *K* (1 ≤ *K* ≤ *M*). Удалить строку матрицы с номером *K*.

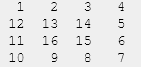
3. В двумерном массиве размерностью N × M, все элементы которого различны, требуется найти такие элементы, которые одновременно являются максимальными в своей строке и минимальными в своем столбце.

**Вариант 12**

1. Дана матрица размера *M×N*. Вывести ее элементы в следующем порядке: первый столбец сверху вниз, второй столбец снизу вверх, третий столбец сверху вниз, четвертый столбец снизу вверх и т. д.

2. Дана матрица размера *M×N* и целое число *K* (1 ≤ *K* ≤ *N*). Удалить столбец матрицы с номером *K*.

3. По введенным значениям N, M (1 ≤ n ≤ 20, 1 ≤ m ≤ 20) заполните массив размерностью N × M числами от 1 до M\*N, расположив их по спирали, закрученной по часовой стрелке, так, как показано в примере.

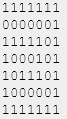


**Вариант 13**

1. Дана матрица размера *M×N* и целое число *K* (1 ≤ *K* ≤ *M*). Найти сумму и произведение элементов *K*-й строки данной матрицы.

2. Дана матрица размера *M×N*. Удалить строку, содержащую минимальный элемент матрицы.

3. Требуется заполнить массив размера N × N единичками по спирали (начиная с верхнего левого угла по часовой стрелке, как на рисунке).

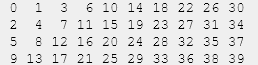


**Вариант 14**

1. Дана матрица размера *M×N* и целое число *K* (1 ≤ *K* ≤ *N*). Найти сумму и произведение элементов *K*-го столбца данной матрицы.

2. Дана матрица размера *M×N*. Удалить столбец, содержащий максимальный элемент матрицы.

3. Создайте матрицу размером *M×N*. и заполните ее, как показано на примере.



**Вариант 15**

1. Дана матрица размера *M×N*. Для каждой строки матрицы найти сумму ее элементов.

2. Дана матрица размера *M×N*. Удалить ее первый столбец, содержащий только положительные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.

3. В кинотеатре n рядов по m мест в каждом. В соответствующем двумерном массиве хранится информация о проданных билетах на определенный сеанс (единицы означают, что на данные места билеты уже проданы, нули – что данные места еще свободны). Поступил запрос на продажу k билетов на соседние места в одном ряду. Определите, можно ли удовлетворить такой запрос.

**Вариант 16**

1. Дана матрица размера *M×N*. Для каждого столбца матрицы найти произведение его элементов.

2. Дана квадратная матрица *A* порядка *M*. Найти минимальный элемент для каждой ее диагонали, параллельной главной (начиная с одноэлементной диагонали).

3. Треугольник Паскаля строится следующим образом. Первая строка состоит из одной единицы. Каждая следующая содержит на одно число больше, чем предыдущая. Первое и последнее из этих чисел равны 1, а все остальные вычисляются как сумма числа, стоящего в предыдущей строке над ним, и числа, стоящего в предыдущей строке слева от него.

**Вариант 17**

1. Дана матрица размера *M×N*. Для каждой строки матрицы с нечетным номером (1, 3, . . .) найти среднее арифметическое ее элементов. Условный оператор не использовать.

2. Дана квадратная матрица *A* порядка *M*. Найти максимальный элемент для каждой ее диагонали, параллельной побочной (начиная с одноэлементной диагонали).

3. Требуется в каждую клетку квадратной таблицы размером N x N поставить ноль или единицу так, чтобы в любом квадрате размера K x K было ровно S единиц.

**Вариант 18**

1. Дана матрица размера *M×N*. Для каждого столбца матрицы с четным номером (2, 4, . . .) найти сумму его элементов. Условный оператор не использовать.

2. Дана квадратная матрица порядка *M*. Обнулить элементы матрицы, лежащие ниже главной диагонали. Условный оператор не использовать.

3. Магическим квадратом порядка N называется квадратная матрица размера *N × N*, составленная из чисел 1, 2, ..., *N*2 так, что суммы по каждому столбцу, каждой строке и каждой из двух больших диагоналей равны между собой. Напишите программу, которая строит магический квадрат заданного порядка *N*.

**Вариант 19**

1. Дана матрица размера *M×N*. Найти максимальный среди минимальных элементов ее строк.

2. Дана квадратная матрица порядка *M*. Обнулить элементы матрицы, лежащие выше побочной диагонали. Условный оператор не использовать.

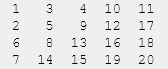
3. На карте некоторого участка планеты, которая хранится в виде прямоугольной таблицы, нули обозначают море, а единицы -– сушу. Все острова имеют форму прямоугольников и не соприкасаются. Написать программу, которая по готовой карте определяет количество островов.

**Вариант 20**

1. Дана матрица размера *M×N*. Найти минимальный среди максимальных элементов ее столбцов.

2. Дана квадратная матрица порядка *M*. Обнулить элементы матрицы, лежащие на побочной диагонали и ниже нее. Условный оператор не использовать.

3. Напишите программу, которая заполняет матрицу из N строк и M столбцов натуральными числами змейкой по диагонали, как показано в примере.



**Вариант 21**

1. Дана матрица размера *M×N*. В каждой ее строке найти количество элементов, меньших среднего арифметического всех элементов этой строки.

2. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие на главной диагонали и выше нее. Условный оператор не использовать.

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

**Вариант 22**

1. Дана матрица размера *M×N*. В каждом ее столбце найти количество элементов, больших среднего арифметического всех элементов этого столбца.

2. Дана квадратная матрица порядка *M*. Обнулить элементы матрицы, лежащие одновременно выше главной диагонали и выше побочной диагонали. Условный оператор не использовать.

3. Даны числа n и m. После чего задаётся матрица размером n×m из 0 и 1. Будем считать, что эта матрица задаёт карту лабиринта, при этом 0 является пустым пространством, а 1 стеной. В данном лабиринте будем считать, что точкой входа является клетка с координатами (1,1), а точкой выхода (n,m). Необходимо определить, возможно ли из точки входа добраться до точки выхода, если можно двигаться только в четырёх возможных направлениях (север, юг, запад, восток) и перемещаться можно только в свободную клетку.

Пример. Входные данные:

4 5

0 0 0 1 1

0 0 1 1 0

0 1 1 0 0

0 0 0 0 0

Выходные данные:

YES

**Вариант 23**

1. Дана матрица размера *M×N*. Найти номера строки и столбца для элемента матрицы, наиболее близкого к среднему значению всех ее элементов.

2. Дана квадратная матрица порядка *M*. Обнулить элементы матрицы, лежащие одновременно выше главной диагонали и ниже побочной диагонали. Условный оператор не использовать.

3. Задаётся число строк двумерного массива n. После чего следующие n чисел задают размеры каждой строки. Необходимо создать массив с данными размерами, отсортировать его по возрастанию размера строки и заполнить неповторяющимися числами по убыванию так, чтобы последнее число было равно 1.

Пример. Входные данные:

4

3 5 2 2

Выходные данные:

12 11

10 9

8 7 6

5 4 3 2 1

**Вариант 24**

1. Дана целочисленная матрица размера *M×N*. Найти номер последней из ее строк, содержащих только четные числа. Если таких строк нет, то вывести 0.

2. Дана квадратная матрица порядка *M*. Обнулить элементы матрицы, лежащие одновременно ниже главной диагонали (включая эту диагональ) и выше побочной диагонали (также включая эту диагональ). Условный оператор не использовать.

3. Даны два числа n и m. Создать матрицу размером n×m и заполнить случайными числами в диапазоне [-10,10]. Необходимо найти в каждой строке максимальный по модулю элемент и отсортировать строки матрицы по возрастанию этого значения.

Пример. Входные данные:

3 3

9 9 3

3 3 -8

-4 0 7

Выходные данные:

-4 0 7

3 3 -8

9 9 3

**Вариант 25**

1. Дана целочисленная матрица размера *M×N*. Найти номер первого из ее столбцов, содержащих только нечетные числа. Если таких столбцов нет, то вывести соответствующее сообщение.

2. Дана квадратная матрица порядка *M*. Обнулить элементы матрицы, лежащие одновременно выше главной диагонали (включая эту диагональ) и выше побочной диагонали (также включая эту диагональ). Условный оператор не использовать.

3. Вводится число n. Необходимо создать матрицу размером n×n и заполнить её случайными числами в диапазоне [1,5]. После чего необходимо проверить, является ли матрица симметричной относительно главной и побочной диагоналей. (Для каждой диагонали написать в отдельной строке YES или NO)

Пример. Входные данные:

3

1 0 2

0 3 3

2 3 1

Выходные данные:

YES

NO

## Задание №6.2 «Многоалфавитное шифрование методом сложной замены (подстановки). Дешифрование текста».

**Цели и задачи работы**: изучение алгоритмов формирования и обработки двумерных массивов, программирование и отладка программ формирования и обработки массивов при реализации многоалфавитного шифрования методом простой замены.

**Задание к работе**: Написать программу решения задачи в соответствии с индивидуальным вариантом.

Методика выполнения работы:

**1. Разработать алгоритм решения задачи по индивидуальному заданию.**

**2. Написать и отладить программу решения задачи.**

**3. Протестировать работу программы на различных исходных данных.**

**Вариант 1**

а) систему шифрования Вижинера для преобразования исходного текста на английском языке;

б) систему шифрования Вижинера для преобразования зашифрованного текста в исходный.

**Вариант 2**

а) систему шифрования Вижинера для преобразования исходного текста на русском языке;

б) систему шифрования Вижинера для преобразования зашифрованного текста в исходный.

**Теоретическая часть**

Квадрат Виженера или таблица Виженера, так же известная как tabula recta, может быть использована для заширования и расшифрования. В XVI веке французский дипломат Блез де Виженер предложил модификацию шифра замен, которая впоследствии получила его имя. В данном шифре ключ задается фразой из *d* букв. Ключевая фраза подписывается с повторением под сообщением. Букву шифротекста необходимо находить на пересечении столбца, определяемого буквой открытого текста, и строки, определяемой буквой ключа:

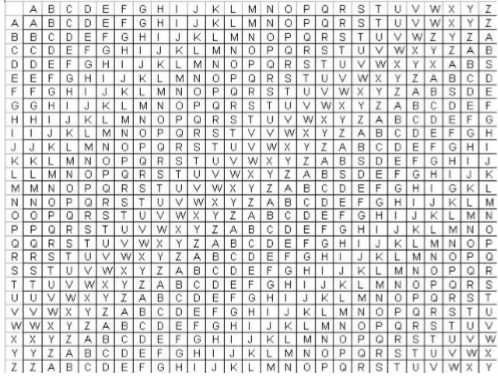
*Vigd*(*mi*)=(*mi*+*ki mod d*)(mod *n*),

где *mi, ki, Vigd*(*mi*)- порядковые номера в алфавите очередных символов открытого текста, ключа и шифротекста соответственно. Обратное преобразование выглядит следующим образом:

*Vigd*-1(*mi*)=(*mi-ki mod d*+*n*)(mod *n*).

Шифр Вижинера для латинского алфавита использует таблицу Б.3.

Таблица Б.3 – Таблица Виженера для латинского алфавита



Пример использования шифра Виженера приведен на рис. (алфавит дополнен пробелом, порядковый номер которого принят за 34, соответственно, мощность алфавита *n*=34).

Принципиальным отличием данного шифра от всех предыдущих является то, что он относится к классу многоалфавитных алгоритмов – как нетрудно заметить, одной и той же букве шифротекста могут соответствовать различные символы открытого текста в зависимости от того, каким символом ключа они были замаскированы (в приведенном примере буква ‘г’ шифрует в одном случае букву ‘ш’, а в другом – букву ‘е’). Это делает бессмысленным подсчет частоты встречаемости символов в шифротексте. Для криптоанализа шифра Виженера можно использовать *метод Казиски.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Порядковый № буквы в алфавите | 26 | 10 | 22 | 18 | 34 | 3 | 10 | 8 | 6 | 15 | 6 | 18 | 1 |
| Открытый текст | ш | и | ф | р |  | в | и | ж | е | н | е | р | а |

Рисунок Б.1 - Пример шифрования с использованием шифра Виженера

ключевая фраза – ‘ключ’

d=4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Порядковый № буквы в алфавите | 12 | 13 | 32 | 25 | 12 | 13 | 32 | 25 | 12 | 13 | 32 | 25 | 12 |
| Ключевая фраза | к | л | ю | ч | к | л | ю | ч | к | л | ю | ч | к |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Порядковый № буквы в алфавите | 4 | 23 | 20 | 9 | 12 | 16 | 8 | 33 | 18 | 28 | 4 | 9 | 13 |
| Шифротекст | г | х | т | з | к | о | ж | я | р | ь | г | з | л |

В середине XIX века немецкий математик Казиски предложил определять длину парольной фразы по расстоянию между одинаковыми фрагментами шифротекста. Допустим, найдены два одинаковых фрагмента шифротекста, расстояние между которыми составляет 20 символов. Это может означать, что два одинаковых фрагмента открытого текста были зашифрованы с одной и той же позиции ключа. Это позволяет предположить, что парольная фраза имеет длину 4, 5, 10 или 20 символов. Узнав (или угадав) длину парольной фразы *l*, можно осуществить частотный криптоанализ шифротекста для выборки каждого *l* –го символа шифротекста.

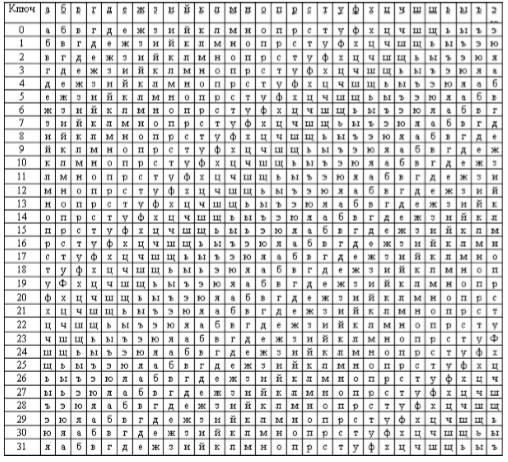
Для увеличения длины парольной фразы и затруднения криптоанализа можно применить составной шифр Виженера, который представляет собой многократное шифрование по Виженеру с различными парольными фразами. Он имеет уравнение

*Vig*\*(*mi*)=(*mi* + *ki mod dk* + *li mod dl* + ... + *si mod ds*) (mod *n*),

где *ki,  li, ..., si* – символы различных парольных фраз. Для повышения криптостойкости парольные фразы должны иметь различные взаимно простые периоды *dk, dl, ..., ds*. Это существенно затруднит для криптоаналитика поиск величины периода повторения общей парольной фразы, которая будет равна произведению длин всех парольных фраз.

Б.12 Шифр Вижинера (для русских букв) Шифр Вижинера (см. тему Б.11) для русского алфавита использует таблицу Б.4:

Таблица Б.4 – Таблица Виженера для русского алфавита



**Задание 6.3** Методы решения систем линейных алгебраических уравнений

**Задание:**

Из табл.6.3.1 выбрать данные для системы линейных уравнений. Найти решение этой системы прямым и приближенным методами с точностью до ε=10-3 . Варианты нечетные решить систему уравнений методом Гаусса с выбором главного элемента или методом простых итераций, четные варианты – методом Холесского или методом простых итераций.

**Исходная система линейных уравнений:**

****

Таблица 6.3.1. Данные для исходной системы линейных уравнений по вариантам

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № варианта | M | N | P | № варианта | M | N | P |
| 1 | -0.77 | 0.16 | 1.12 | 9 | -1.13 | 0.14 | 0.87 |
| 2 | 0.93 | 0.07 | -0.84 | 10 | 0.91 | -0.23 | -1.04 |
| 3 | -1.14 | -0.17 | 0.95 | 11 | -0.88 | 0.1 | 0.91 |
| 4 | 1.08 | 0.22 | -1.16 | 12 | 1.25 | -1.14 | -1.09 |
| 5 | 0.87 | -0.19 | 1.08 | 13 | 0.79 | 0.18 | -0.86 |
| 6 | -1.21 | 0.2 | 0.88 | 14 | -1.19 | -0.21 | 1.21 |
| 7 | 1.09 | -0.16 | 0.84 | 15 | 0.89 | 0.12 | -1.15 |
| 8 | 0.89 | 0.08 | -1.21 | 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 методом простых итераций.
5. Составьте программу решения системы уравнений с точностью до ε, выводящую результаты в таблицу:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | х1 | х2 | х3 | х4 | ε n |
| … | … | … | … | … | … |

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

**Теоретическая часть**

1. Постановка задачи

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

*Ax = b* (1)

где А – действительная матрица , b – вектор столбец. X – вектор неизвестных, принадлежат Rn – n-мерному евклидовому пространству.

*a11x1 + a12x2 + … + a1nxn = b1*

*…*

*an1x1 + an2x2 + … + annxn = bn* (2)

Требуется найти решение  из Rn 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 А 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 , где  – норма матрицы A,  – норма обратной матрицы. Чем больше это число, тем хуже обусловленность системы. В качестве нормы А 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

2.1. 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 А. Матрица А преобразуется к верхнетреугольному виду с единицами на главной диагонали.

б) Обратный ход, из матрицы А\* определяем последовательно xn,…,x1.

Надо решить систему алгебраических уравнений *Ax = b:*

*a11x1 + a12x2 + … + a1nxn = b1*

*…*

*an1x1 + an2x2 + … + annxn = bn*

Нужно преобразовать к треугольной матрице.

Сделаем последовательно:

1 шаг:



Исключаем из *x1* всех уравнений по симплекс – правилу:



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.

При i=1 имеем  , j=i,…,n

В общем случае, ,

,

где *k*-номер шага *i=k+1,…n-1, j=i,2…n*  (3)

Получим



Обратный ход, начиная с последнего уравнения, последовательно определяем xn, xn-1, …, x1:



 (4)

***Замечание!***

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

Изложенный метод имеет ограничение, связанное с тем, что ведущие элементы *а11,а22,…,ann* 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 *аii* путем перебора этих элементов по столбцу, соответствующему этому ведущему элементу или по всем столбцам.

Метод Гаусса с выбором главного элемента по столбцу

Перед исключением х1 отыскивается мах|аi1| по i. Допустим, максимум соответствует i=i0. Тогда первое уравнение в исходной системе меняем местами с i0 уравнением. После этого осуществляем первый шаг исключения. Затем перед исключением x2 из оставшихся уравнений отыскиваем max |аi2| , где 2<=i<=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.

, для всех *k=1,2,…,n-1*. (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.



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.о. получаем треугольную матрицу, далее начинается обратный ход (5.4).

2.2. Ме*т*од Холесского

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

*A=LU.*

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

Решение системы сводится к последовательному решению двух простых систем с треугольными матрицами.

***Замечание.*** Всякую квадратную матрицу А, имеющую отличные от нуля  главные миноры можно представить в виде LU-разложения, причем это разложение будет единственным.

Из соотношения *A=LU* следует:



Получим для первой строки и столбца:



Аналогично для i>=j и i<j:

 (6)

Обратный ход:

 (7)

***Замечание!***

При большом числе уравнений прямые методы решения систем линейных алгебраических уравнений становятся трудно реализуемыми на компьютере, прежде всего из-за сложности хранения промежуточных результатов и операций с большими матрицами.

Существуют различные способы представления матрицы А в виде

А=А1\*А2\* …\*Аm,

Где каждая Ai имеет удобную форму для решения системы линейных уравнений. Как правило m<=5. Тогда в результате решения последовательности систем

А1b1=b0, А2b2=b1, …, Ambm=bm-1 можно найти искомое решение.

3. Итерационные методы решения систем линейных уравнений

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

3.1. Метод трехточечной прогонки.

Общая постановка задачи.

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

 (8)

Требуется найти решения *х=(х1,х2,…,хn)* системы.

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

 (9)

Чтобы начать исключение, разделим первое уравнение этой системы на диагональный элемент  и используем обозначения  и . Предположим, что мы исключили все нулевые поддиагональные элементы в первых  строках. В этом случае система преобразуется к виду:

 (10)

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



Чтобы получить единицу на главной диагонали матрицы, разделим -ю строку на коэффициент . В результате для элементов  и  в окончательном виде -й строки получаем следующие формулы:

 (11)

Продолжая исключение, получаем систему уравнений с двухдиагональной матрицей вида

 (12)

Это позволяет записать рекуррентные формулы для вычисления неизвестных 

 (13)

В виду опасности обращения знаменателя в ноль возникает вопрос об устойчивости решения, таким условием является диагональное преобладание в исходной матрице: для всех i. В этом случае *|pi|<1*

Док-во методом индукции:

Пусть *|pi-1|<1,* тогда

**.

Утверждение доказано. Из того, что , 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 n3 операций.

3.2. Метод простой итерации.

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

*a11x1 + a12x2 + … + a1nxn = b1*

*…* (14)

*an1x1 + an2x2 + … + annxn = bn*

Выразим :



Таким образом , получим *x=Сx+f ,* где

, ,   *.* (15)

Итерационный процесс запишется виде:

 (16)

***Замечаниe!***

Процесс называется параллельным итерированием, так как для вычисления (к+1)-го приближения всех неизвестных учитываются вычисленное ранее их к-е приближение. Начальное приближение х(0) выбирается произвольно.

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

*Теорема1 Для сходимости итераций (5.16) к решению системы (5.14) достаточно, чтобы в какой-либо норме выполнялось условие  тогда независимо от выбора х(0) выполняется*

*, где х\* - точное решение (5.14).*

Док-во:

Из (14) имеем х\*=Сх\*+f .

Вычитаем из (16) х(k)-x\*=C(x(k-1) – x\*).

Получим оценку погрешности



очевидно, что при q<1 .

***Замечаниe!***

Условие теоремы, как достаточное, является завышенным.

Сходящийся процесс обладает свойством самоисправляемости.

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

*Теорема 2. Для сходимости итераций (5.16) к решению системы (5.14) необходимо и достаточно, чтобы все собственные значения матрицы С 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.

 (17)

Для того, чтобы данный процесс сходился, необходимо, чтобы норма матрицы С была меньше 1 или выполнялись следующие условия:



 (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 разрешенных матрицах (в коэффициентах стоит много нулей);

- процесс легко программируется.

Количество операций для выполнения этого метода приблизительно равно (2n2)k, где k количество приближений. Если допустимая погрешность достигается при , то метод итераций становится предпочтительнее метода Гаусса.

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

***Методика решения задачи***

Преобразовать систему *Ах=b* к виду *х=сх+f*

Задать начальное приближение *х(0)*и точность ε. Положить *k=0*.

Вычислить приближение по формуле *х(к+1)=сх(к)+f*

Если выполнено условие , то процесс завершить и положить *х\*=х(к+1*), иначе *k=k+1* перейти к пункту 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.



 (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.