

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ  
И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Северо-Кавказский филиал ордена Трудового Красного Знамени  
федерального государственного бюджетного образовательного учреждения  
высшего образования  
«Московский технический университет связи и информатики»

**С.А. ШВИДЧЕНКО**

Методические указания  
для проведения лабораторных работ (II семестр)  
по дисциплине

**Б1.О.05 «ИНФОРМАТИКА»**

Кафедра **«Информатика и вычислительная техника»**

Направление подготовки **09.03.01. Информатика и вычислительная техника**

Профиль **Вычислительные машины, комплексы, системы и сети,**  
**Программное обеспечение и интеллектуальные системы**

Разработала:

*Доцент кафедры ИВТ Швидченко С.А.*

Ростов-на-Дону  
2021

Методические указания  
для проведения лабораторных работ  
по дисциплине  
«Информатика»

Составитель: Швидченко С.А., доц. каф. «ИВТ»

Рассмотрено и одобрено  
на заседании кафедры «ИВТ»  
Протокол от «30» августа 2021 г., № 1.

## Модуль 1.

### Лабораторная работа №1. Математическое моделирование. Компьютерное моделирование.

#### Тема: Линейная аппроксимация статистических данных

**Цель работы:** изучение одного из методов обработки статистических или экспериментальных данных с целью получения линейной зависимости между двумя показателями.

#### Задание и порядок выполнения работы:

- 1) Изучить теоретические материалы [1-3,12] о линейной аппроксимации данных эксперимента или статистики и ознакомиться с методическими указаниями по выполнению данной работы.
- 2) Сформулировать постановку задачи.
- 3) Изучить метод решения задачи и разработать алгоритм ее решения.
- 4) Решить задачу с помощью MS Excel или программы на алгоритмическом языке.
- 5) Построить график аппроксимирующей функции.
- 6) Оформить отчет по данной работе.

#### Методические указания по выполнению работы

**Постановка задачи.** Пусть требуется определить функциональную зависимость между двумя величинами  $X$  и  $Y$ . Они могут быть параметрами некоторого либо физического процесса, либо экономического показателя, либо других явлений природного характера. Для определения этой зависимости проводят наблюдения или эксперименты, результаты которых записывают в виде таблицы значений рассматриваемых величин. Используя эти данные, требуется определить математическую зависимость между этими величинами.

**Математическая модель задачи.** Искомая зависимость записывается в виде линейной функции  $y = a \cdot x + b$ , где  $a$  и  $b$  – неизвестные пока параметры, значения которых должны быть определены.

**Метод решения задачи.** Данная задача решается известным методом наименьших квадратов, сущность которого заключается в следующем. График аппроксимирующей функции должен проходить очень близко от статистических точек. Это условие может быть осуществлено, если следующая функция имеет наименьшее значение:

$$U(a, b) = \sum_{k=1}^n [y_k - (a \cdot x_k + b)]^2 \Rightarrow \min .$$

Здесь  $n$  – количество статистических точек,  $(x_k, y_k)$  – координаты статистических точек. Необходимым и достаточным условием минимума данной функции может быть записано:

$$\frac{\partial U}{\partial a} = 0, \quad \frac{\partial U}{\partial b} = 0.$$

Из этих условий будут определены следующие формулы для определения значений неизвестных параметров  $a$  и  $b$ :

$$a = \frac{n \cdot S_4 - S_1 \cdot S_2}{n \cdot S_3 - S_1^2}, \quad b = \frac{S_3 \cdot S_2 - S_1 \cdot S_4}{n \cdot S_3 - S_1^2},$$

где  $S_1 = \sum_{k=1}^n x_k, \quad S_2 = \sum_{k=1}^n y_k, \quad S_3 = \sum_{k=1}^n x_k^2, \quad S_4 = \sum_{k=1}^n x_k y_k.$

Алгоритм решения задачи:

- ввод массивов  $x_k, y_k, k = 1, 2, \dots, n,$  в память компьютера;
- определение значений следующих сумм  $S_1, S_2, S_3, S_4;$
- определить значения искомых параметров  $a$  и  $b$ .

Пример. Использовать статистические данные, приведенные в таблице 1.

Таблица 1- Статистические данные значений величин  $X$  и  $Y$

$x_k$	50	60	70	80	90	100	110	120	130	140
$y_k$	65	75	95	100	115	130	155	170	180	190
$x_k$	150	160	170	180	190	200	210	220	230	240
$y_k$	200	210	235	250	265	290	310	350	370	400

Результатом решения задачи является линейная аппроксимирующая функция  $y = a \cdot x + b$  и ее график в виде прямой линии.

#### Контрольные вопросы ЛР1(ОПК-3):

1. Что такое Математическое моделирование?
2. Что такое Компьютерное моделирование?
3. Создать простейшие модели объектов и процессов в виде изображений и чертежей, динамических (электронных) таблиц.
4. Провести компьютерные эксперименты с использованием готовых моделей объектов и процессов.

5. Классификация моделей.
6. Основные этапы разработки и исследования моделей на компьютере.
7. Сущность математического моделирования.
8. Сущность компьютерного моделирования.
9. Что такое аппроксимация статистических данных?
10. Какая линия является графиком линейной функции аппроксимации?
11. Почему рассматривается минимум функции  $U(a, b)$ ?
12. Почему условие равенства нулю первых производных функции  $U(a, b)$  является достаточным условием минимума?
13. В чем сущность метода наименьших квадратов?

Тема: **Основы проектирования локальных компьютерных сетей**

Цели работы Изучение базовых технологий построения локальных сетей; получение навыков конфигурирования локальной компьютерной сети в зависимости от возлагаемых на нее функций. Применение метода анализа иерархий для выбора оптимального решения.

#### Указания к выполнению работы

Перед выполнением работы необходимо повторить следующие разделы теории, изучавшиеся в предыдущих курсах:

- Основы сетевых технологий. Модель взаимодействия открытых систем OSI.
- Понятие топологии вычислительной сети. Виды топологий.
- Основные сетевые технологии: Ethernet, Token Ring, Arcnet.
- Техническое обеспечение информационно-вычислительных сетей. Коммутаторы, концентраторы, маршрутизаторы.

#### Основные этапы проектирования ЛВС

Проектирование информационно-вычислительных сетей – сложный и ответственный процесс. Известно, что любая вычислительная система по своей сути представляет комплекс технических средств, необходимых для функционирования некоторой информационной системы. Поэтому эффективность работы информационной системы во многом зависит от того, соответствует ли ей уровень используемой вычислительной системы.

В данной работе рассматриваются основы проектирования *локальных информационно-вычислительных сетей* (ЛВС). Следует помнить, что универсальных рекомендаций по проектированию, которые бы учитывали все возможные факторы и обстоятельства и давали наилучшее решение во всех случаях, не существует. Тем не менее, можно сформулировать общие подходы к проектированию локальных компьютерных сетей, использование которых хотя бы направит этот процесс в нужное русло.

Обычно процесс создания локальной сети включает в себя следующую последовательность этапов:

1. Анализ исходных данных;
2. Выбор основных сетевых решений;
3. Анализ финансовых затрат на проект и принятие окончательного решения;
4. Прокладка кабельной системы;
5. Организация силовой электрической сети;
6. Установка оборудования и сетевого программного обеспечения;
7. Конфигурирование (настройка параметров) сети.

Первые три этапа касаются непосредственно процесса проектирования и являются основополагающими. В результате их выполнения формулируется *технико-экономическое обоснование* (ТЭО), которое включает в себя анализ предметной области и обоснование необходимости создания в организации локальной информационно-вычислительной сети. Кроме того, ТЭО обязательно должно содержать расчеты экономической эффективности, а также итоговое заключение о целесообразности и получаемых перспективах от реализации проекта (в данном случае, создания ЛВС)

## Определение исходных данных

На этом этапе на основе анализа предметной области определяются те базовые требования, которым должна удовлетворять проектируемая локальная сеть.

1. Анализ предметной области необходимо начинать с определения *целей* разработки ЛВС. В качестве общих можно назвать такие цели как: обеспечение связи, совместная обработка информации, совместное использование данных и файлов, централизованное управление компьютерами, контроль за доступом к важным данным. Разумеется, в каждом конкретном случае перечень целей должен быть уточнен и дополнен. Следует помнить, что всякая цель проектирования и реализации ЛВС возникает не сама по себе, а как одна из целей функционирования некоторой информационной системы.
2. После определения списка целей необходимо выделить функционально-независимые группы пользователей локальной сети и указать для каждой из групп перечень их *функций* в ЛВС. **Например**, для пользователей группы «Клиенты туристической фирмы» можно предусмотреть функцию ознакомления с электронными презентациями новых маршрутов, а для пользователей «Менеджер туристической фирмы» – функции доступа к внутренней базе данных фирмы, подключения к глобальным сетям бронирования, связи с другими менеджерами и т.п. Следует помнить, что реализация каждой пользовательской функции должна способствовать достижению ранее заявленных целей разработки локальной сети.
3. Проведенный анализ целей и функций позволяет выдвинуть *общие требования* к проектируемой ЛВС:
  - Размер сети (количество компьютеров и расстояние между ними в настоящее время, а также в ближайшем будущем и в перспективе);
  - Структура сети (иерархия и основные части – по подразделениям, комнатам, этажам и т.п.);
  - Основные направления, характер (данные, изображения, звук, видео) и интенсивность информационных потоков;
  - Необходимость подключения к глобальным или другим локальным сетям.
  - Типовые характеристики компьютеров ЛВС.
  - Требования к программному обеспечению, устанавливаемому на компьютерах, объединяемых в сеть.

На основе выдвинутых требований проектировщик осуществляет поиск оптимального варианта ЛВС.

## Выбор основных сетевых решений

Выбор сетевых решений для локальной компьютерной сети осуществляется на основе следующих принципов:

- Сеть должна соответствовать требованиям, сформулированным на этапе анализа исходных данных.
- Проект сети должен удовлетворять условиям совместимости выбранных программных и аппаратных средств
- Предложенный вариант проекта ЛВС должен быть наиболее оптимальным с точки зрения некоторого критерия.
- Архитектура сети должна обеспечивать возможность дальнейшего развития сети.
- Управление используемым оборудованием должны быть как можно более простым.

К основным сетевым решениям, которые проектировщик должен выбрать для проектируемой компьютерной сети, относятся:

- Выбор сетевой архитектуры, что подразумевает:

- Выбор топологии сети, то есть схемы соединения компьютеров, кабельной системы и других сетевых компонентов;
- Выбор протокола передачи данных;
- Выбор типа кабельной системы;
- Выбор сетевого оборудования.
- Определение параметров серверного оборудования.
- Определение характеристик рабочих станций.
- Планирование мер по обеспечению информационной безопасности.
- Планирование мер защиты от перебоев электропитания.
- Выбор концепции совместного использования периферийных устройств.
- Выбор сетевого ПО.

Выбор топологии означает выбор схемы соединения компьютеров, кабельной системы и других сетевых компонентов. Существуют три основных вида сетевой топологии: общая шина, звезда и кольцо. Каждая из топологий имеет свои достоинства и недостатки, указанные в таблице 1.

Таблица 1. Сравнительная характеристика базовых сетевых топологий

Характеристики	«Звезда»	«Кольцо»	«Шина»
Стоимость организации	Средняя	Высокая	Низкая
Надежность передачи данных	Средняя	Высокая	Низкая
Масштабируемость	Высокая	Средняя	Низкая
Защищенность от прослушивания	Хорошая	Хорошая	Плохая
Удобство и простота обслуживания	Хорошее	Среднее	Плохое

На практике очень редко удастся организовать локальную сеть на базе единственной топологии. Чтобы сеть работала эффективно, сначала необходимо спроектировать *структуру сети*, то есть определить способ ее разделения на части (*сегменты*) и схему соединения этих частей между собой. Определение структуры сети должно производиться с использованием сведений, полученных на этапе определения исходных данных: физическое расположение компьютеров по комнатам и этажам, взаимное расположение комнат, относящихся к одному подразделению, направления, характер и объемы информационных потоков внутри и между подразделениями. Идеальным вариантом является ситуация, когда рабочие места сотрудников, занимающихся одной задачей, находятся в одной или рядом расположенных комнатах. В этом случае структура сети будет соответствовать структуре здания (или комплекса зданий) организации. После определения структуры сети, проектировщик принимает решение о выборе топологии – либо общей для всей сети, либо отдельно для каждого сегмента.

Выбор согласованных протоколов для передачи данных (выбор сетевой технологии) – одна из важнейших и наиболее сложных задач, возникающих в процессе проектирования ЛВС. В зависимости от метода доступа к передающей среде (каналу передачи данных), различают следующие сетевые технологии:

- Технология Ethernet;
- Технология Token Ring;
- Технология Arcnet.

Указанные технологии реализованы на базе международных стандартов Института Инженеров по Электротехнике и Радиоэлектронике (IEEE) и являются широко распространенными в настоящее



время (в последнее время использование технологии Arcnet значительно уменьшилось). Их сравнительную характеристику можно увидеть в таблице 2.

Таблица 2. Сравнительная характеристика основных сетевых технологий

Характеристика	Ethernet	Token Ring	Arcnet
Используемые топологии	Шина, Звезда	Кольцо, Звезда	Шина, Звезда
Кабельная система	Коаксиальный кабель, неэкранированная и экранированная витая пара, волоконно-оптический кабель, радио и инфракрасные каналы	Экранированная и неэкранированная витая пара, волоконно-оптический кабель	Коаксиальный кабель, неэкранированная и экранированная витая пара, волоконно-оптический кабель, радио и инфракрасные каналы
Стоимость	Низкая	Высокая	Средняя
Макс. скорость передачи данных	До 1 Гбит/с	До 200 Мбит/с	До 20 Мбит/с
Надежность передачи данных	Низкая	Высокая	Средняя
Масштабируемость	Низкая	Средняя	Средняя
Удобство и простота обслуживания	Средняя	Низкая	Высокая

На скорость и надежность передачи данных, а также на максимальный размер сети существенное влияние оказывает и выбор кабельной системы, используемой для соединения сегментов сети и отдельных компьютеров. В настоящее время используют такие типы кабельной системы как экранированная и неэкранированная витая пара, толстый и тонкий коаксиальный кабель, одномодовый и многомодовый волоконно-оптический кабель, радио и инфракрасные каналы. Сравнительные характеристики различных типов кабелей приведены в таблице 3.

Таблица 3. Сравнительная характеристика основных типов кабельных систем

Характеристика	Неэкранированная витая пара	Экранированная витая пара	Коаксиальный кабель	Волоконно-оптический кабель	Радио и инфракрасный канал
Стоимость	Низкая	Средняя	Выше средней	Высокая	Выше средней
Скорость передачи данных	До 1 Гбит/с	До 1 Гбит/с	До 50 Мбит/с	До 1 Гбит/с	До 50 Мбит/с
Защита от помех	Низкая	Средняя	Выше средней	Высокая	Низкая
Размер линии связи	Низкий	Низкий	Средний	Высокий	Средний
Удобство	Выше средней	Ниже средней	Ниже средней	Низкая	Высокая

прокладки и обслуживания					
Мобильность	Средняя	Низкая	Низкая	Низкая	Высокая

При выборе сетевого оборудования необходимо учитывать многие факторы, в том числе:

- Требования к скорости и интенсивности передачи данных в проектируемой ЛВС (по сети в целом и по отдельным сегментам);
- Требования к структуре сети и возможный выбор сетевых топологий;
- Выбранную сетевую технологию (Ethernet, Token Ring, Arcnet и т.п.);
- Выбранные типы кабеля сети, требования к максимальному размеру сети (в том числе отдельных соединяющих сегментов) и защищенности от помех.
- Стоимость и технические характеристики конкретных аппаратных средств (сетевых адаптеров, повторителей, концентраторов, коммутаторов, мостов, маршрутизаторов и др.);
- Уровень стандартизации оборудования и его совместимость с наиболее распространенными программными средствами;

Следует помнить, что все рассмотренные аспекты выбора сетевой архитектуры должны рассматриваться не в отрыве друг от друга, а комплексно.

При определении характеристик серверного оборудования и оборудования рабочих компьютеров сети следует ориентироваться на требования, выдвинутые в процессе анализа исходных данных. Кроме того, следует принять решение относительно выбора организации управления в ЛВС. В настоящее время по данному основанию разделяют следующие виды компьютерных сетей:

- Одноранговые сети (сети с децентрализованным управлением);
- Серверные сети с «толстым» клиентом (сети с централизованным управлением, прикладное программное обеспечение размещено и на клиенте, и на сервере);
- Серверные сети с «тонким» клиентом (сети с централизованным управлением, прикладное программное обеспечение размещено только на сервере);

В таблице 4 рассмотрены некоторые характеристики указанных видов ЛВС.

Таблица 4. Сравнительная характеристика ЛВС с разной организацией управления

Характеристика	Одноранговая сеть	Серверная сеть с «толстым» клиентом	Серверная сеть с «тонким» клиентом
Стоимость серверного оборудования	Отсутствует	Высокая	Очень высокая
Стоимость рабочих станций	Высокая	Средняя	Низкая
Макс. размер сети	Низкий	Высокий	Высокий
Защита информации	Низкая	Выше средней	Высокая
Удобство управления	Низкое	Высокое	Высокое

Планирование мер по обеспечению информационной безопасности и защиты от сбоев электропитания заключается в выборе дополнительных аппаратных или программных средств, в том числе таких, как:

- Организация межсетевых брандмауэров;
- Применение механизмов шифрования данных;
- Использование электронной цифровой подписи;
- Применение средств контроля и подстановки трафика;
- Использование сверхнадежных RAID-систем для хранения информации на сервере;
- Использование источников бесперебойного питания для обеспечения надежной работы серверных и иных сетевых устройств.

Каждая из приведенных выше мер позволяет повысить соответствующий «показатель качества» проектируемой компьютерной сети, однако стоимость ЛВС при этом также возрастает.

При выборе программного обеспечения для проектируемой сети особое значение имеет выбор *сетевой операционной системы* (СОС). В настоящее время широкое распространение получили СОС Novel Netware и СОС Microsoft Windows (Server) (разумеется, это не единственные возможные варианты). Многие специалисты указывают, что при примерно равных затратах на покупку ПО, сетевая операционная система обеспечивает более высокий уровень защиты данных от несанкционированного доступа и быстродействия при данном типе сетевого оборудования. Кроме того, эксплуатационные расходы при использовании СОС Novell заметно ниже аналогичных расходов при использовании СОС Microsoft Windows (особенно для больших ЛВС). С другой стороны, СОС Microsoft Windows обеспечивают более высокий уровень совместимости с программным обеспечением рабочих компьютеров сети, что положительно сказывается на эффективности работы ЛВС. Поэтому для небольших и средних компьютерных сетей использование СОС Microsoft Windows является вполне оправданным.

### Выбор оптимального варианта ЛВС

Обычно для заданной предметной области можно составить несколько вариантов конфигурации локальной компьютерной сети, каждый из которых удовлетворяет требованиям, выдвинутым на этапе определения исходных данных. Между собой эти варианты могут сильно различаться по стоимости реализации, уровню быстродействия и надежности передачи данных и т. д. Для выбора оптимального проекта проводится системная оценка всех вариантов ЛВС по восьми основным критериям:

- Быстродействие (скорость передачи данных);
- Надежность (защищенность передачи данных от искажений и помеховню быстродействия и надежности передачи данных и т. иямлисти и условиям программной и аппаратной совместимос);
- Информационная безопасность (защищенность от несанкционированного доступа к информации, защищенность информации от возможных потерь);
- Мобильность (как один из показателей эффективности использования ЛВС);
- Стоимость организации и эксплуатации сети;
- Масштабируемость (возможность увеличения размера сети в будущем);
- Удобство организации и обслуживания ЛВС.

Очевидно, что нахождение оптимального варианта зависит от того, какие критерии из перечисленных являются приоритетными. Из множества методов решения поставленной задачи в данной работе предлагается рассмотреть и использовать *метод анализа иерархий* Саати.

### Метод анализа иерархий

Метод анализа иерархий (МАИ) был разработан известным американским специалистом Т. Саати (T. Saaty) специально для задач принятия решений. В настоящее время указанный метод широко используется в самых разных предметных областях от оценки недвижимости до выбора кандидата на замещение вакантной должности. Суть метода заключается в иерархической декомпозиции исходной

проблемы на все более простые составляющие части и последующего экспертного сравнения этих частей для определения приоритетности имеющихся альтернатив. Рассмотрим общий алгоритм метода более подробно.

- *Первым этапом* применения МАИ является структурирование проблемы выбора в виде иерархии или сети. В вершине иерархии, используемой в МАИ, располагается основная цель, далее, со второго по предпоследний уровень — подцели, и, наконец, на самом нижнем уровне — альтернативы, среди которых производится выбор. Цель, подцели и альтернативы обычно называют *объектами* или *элементами иерархии*. В нашем случае целью, очевидно, является выбор оптимального варианта ЛВС. Поскольку решение о выборе наилучшего проекта зачастую принимается группой лиц (экспертов), каждый из которых имеет собственное суждение относительно имеющихся вариантов, то за объекты иерархии второго уровня целесообразно принять мнения каждого из этих лиц. Например, в принятии решения о выборе варианта ЛВС организации могут участвовать технический финансовый и генеральный директоры. Эксперты не зависимо друг от друга выбирают оптимальный вариант исходя из указанного выше набора критериев (быстродействие, надежность, информационная безопасность и т.д.) — которые образуют множество объектов иерархии третьего уровня. Наконец, на последнем, четвертом уровне должны находиться имеющиеся альтернативы — варианты построения локальной компьютерной сети. Построенная иерархия довольно точно отражает реальную ситуацию, в которой принимается решение — во всяком случае, с точки зрения влияющих на него факторов.
- *На втором этапе* применения МАИ выясняется интенсивность взаимодействия элементов иерархии. Определение интенсивности взаимодействия позволяет вычислить величину воздействия низших уровней иерархии на высшие уровни и, тем самым, решить задачу выбора наилучшей альтернативы. На каждом уровне интенсивность взаимодействия объектов может быть интерпретирована по-разному (рассмотрим интерпретацию для поставленной задачи):
  - Для второго уровня она показывает, насколько мнение одного эксперта относительно остальных для принятия окончательного решения.
  - Для второго уровня — насколько важен с точки зрения каждого из экспертов тот или иной критерий по отношению к остальным при выборе оптимального варианта.
  - Для третьего уровня — насколько предпочтительнее, по мнению каждого из экспертов и с точки зрения каждого из используемых критериев, один из имеющихся вариантов ЛВС по отношению к остальным.

Для определения интенсивности взаимодействия элементов иерархии в МАИ используются *парные сравнения элементов*. Все элементы иерархии одного уровня сравниваются парами с точки зрения их важности и влияния на принятие решения. Сравнение происходит с использованием следующей шкалы:

1	Равная важность/предпочтительность
3	Умеренное превосходствл одного над другим
5	Существенное превосходство одного над другим
7	Значительное превосходство одного над другим
9	Очень сильное превосходство одного над другим
2, 4, 6, 8	Промежуточные значения шкалы

Результаты попарного сравнения элементов заносятся в *матрицу сравнения* размерности  $n \times n$ , где  $n$  — число сравниваемых элементов. Элемент  $a_{ij}$  указанной матрицы выражает результат сравнения элементов  $i$  и  $j$ . Если при сравнении элементов  $i$  и  $j$  получено  $a_{ij}=b$ , то результатом сравнения элементов  $j$  и  $i$  должно быть  $a_{ji}=1/b$ . Очевидно, что диагональные элементы матрицы равны 1. Сравнение элементов проводится на всех уровнях иерархии, начиная со второго. В случае выбора оптимального варианта ЛВС сначала проводится

сравнение авторитетности мнений экспертов, участвующих в принятии решений. После этого каждый эксперт должен, во-первых, провести попарное сравнение важности используемых критериев оценки, а затем выполнить попарное сравнение имеющихся альтернатив с точки зрения каждого из критериев. Таким, образом, каждый эксперт должен получить в результате своей работы 1 матрицу сравнения размером  $8 \times 8$  (для третьего уровня иерархии) и 8 матриц размером  $3 \times 3$  (для трех возможных вариантов ЛВС). Общее количество матриц сравнения для рассматриваемой задачи:  $1 + m \times (1+8) = 1+9 \times m$ , где  $m$  – количество участвующих экспертов.

Если  $a_{1j}, \dots, a_{n(j)j}$  – обозначения элементов иерархии  $j$ -того уровня, а  $n(j)$  – их количество ( $j = 2, 3, \dots, k$ ), то матрицы сравнения  $j$ -того уровня можно обозначить как  $A_{ij} = (a_{pq}^{ij})$ , где  $a_{pq}^{ij} = \frac{a_{pj}}{a_{qj}}$ ,  $p, q = 1, \dots, n(j)$  – каждая матрица соответствует фиксированному набору  $n(j)$  элементов иерархии вышеразмещенных уровней.

- На третьем этапе происходит обработка полученных данных и синтез вектора приоритетов, который ранжирует рассматриваемые альтернативы с точки зрения их предпочтительности. Для этого прежде всего находят *векторы локальных приоритетов* для каждой из полученных матриц сравнения. Искомый вектор локальных приоритетов  $w$  будет равен собственному вектору для максимального собственного значения соответствующей матрицы, нормализованному к единице. Т. Саати предложил упрощенную процедуру вычисления вектора  $w$ . Пусть  $v$  – вектор *геометрических средних* строк некоторой матрицы сравнения:

$$v_j = \sqrt[n(j)]{a_{1j} a_{2j} \dots a_{n(j)j}} \quad (1)$$

Тогда вектор  $w$  будет определяться следующим образом:

$$w_j = \frac{v_j}{\sum_{j=1}^{n(j)} v_j} \quad (v_1, \dots, v_{n(j)} - \text{элементы вектора } v) \quad (2)$$

Вектор локальных приоритетов составляется для каждой матрицы сравнения и характеризует относительную силу влияния каждого отдельного объекта на данном уровне иерархии без учета информации с других уровней. После определения локальных векторов приоритета для всех матриц сравнения производится синтез *общих векторов приоритетов*  $W$ , характеризующих степень влияния каждого объекта на данном уровне иерархии с учетом информации вышестоящих уровней. Процедура синтеза проводится по иерархии объектов снизу вверх и может быть записана в виде следующего алгоритма:

1. На нижнем ( $k$ -том) уровне иерархии вектор локальных приоритетов и общий векторы приоритетов совпадают:  $w_k = v_k$ .

2. На  $j$ -том уровне иерархии ( $2 \leq j < k$ ) для всех наборов элементов иерархии вышестоящих уровней  $\{x_1, x_2, \dots, x_{n(j)}\}$  можно составить матрицу  $\{a_{ij}^{(j)}\}$ , размера  $n(k) \times n(j)$ , столбцами которой являются общие векторы приоритетов следующего ( $j+1$ -ого) уровня:

$$\{a_{ij}^{(j)}\} = \{a_{ij}^{(j+1)}\} \cdot \{w_{ij}^{(j+1)}\} \quad (3)$$

В этом случае общие векторы приоритетов  $j$ -того уровня будут вычисляться как произведение матрицы  $\{a_{ij}^{(j)}\}$  на соответствующий вектор локальных приоритетов:

$$\{w_{ij}^{(j)}\} = \{a_{ij}^{(j)}\} \cdot \{w_{ij}^{(j+1)}\} \quad (4)$$

Размерность вектора  $\{w_{ij}^{(j)}\}$  равна  $n(k)$ .

3. В результате на 2-ом уровне иерархии получим *глобальный вектор приоритетов*  $W$  размерности  $n(k)$ , элементы которого показывает относительную предпочтительность выбора той или иной альтернативы  $k$ -того уровня.

## Пример

Рассмотрим пример использования метода анализа иерархий для принятия решений. Пусть задача принятия решения состоит в выборе телевизора в квартиру. Анализируя предметную область задачи, можно получить следующие данные:

- Лица, принимающие решения (эксперты): Муж, Жена.
- Критерии, по которым выбирается телевизор: качество изображения, стоимость, внешний вид (дизайн).
- Альтернативы: телевизор А, телевизор В.

Иерархия объектов, отражающая структуру решаемой задачи, приведена на рисунке 1.

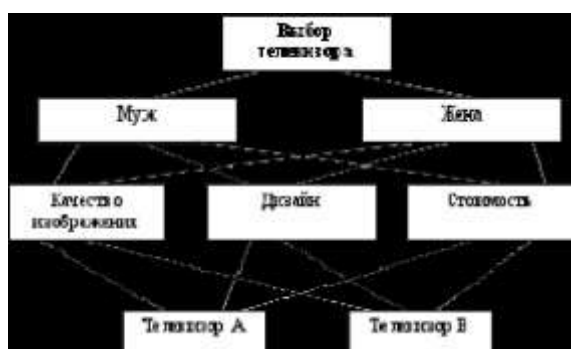


Рис. 1. Пример иерархии объектов для задачи принятия решения

Выясним интенсивность взаимодействия элементов иерархии на каждом уровне. На втором уровне единственная матрица сравнения показывает влияние мнения каждого из экспертов на принятие окончательного решения:

$$M(1,1) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad M(1,2) = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

В данном случае предполагается, что муж и жена равноправно участвуют в выборе телевизора.

На следующем уровне каждый из экспертов должен установить свои приоритеты для критериев, по которым будет выбираться телевизор:

$$M(2,1) = \begin{bmatrix} \text{качество изображения} & \text{дизайн} & \text{стоимость} \\ \text{дизайн} & 1 & 1 \\ \text{стоимость} & 1 & 1 \end{bmatrix}, \quad M(2,2) = \begin{bmatrix} \text{качество изображения} & \text{дизайн} & \text{стоимость} \\ \text{дизайн} & 1 & 1 \\ \text{стоимость} & 1 & 1 \end{bmatrix}$$

Матрица сравнения  $M(1,1)$  составляется первым экспертом, матрица  $M(1,2)$  – вторым. Из матрицы, составленной мужем, видно, что качество изображения имеет, по его мнению, значительное превосходство над таким критерием, как дизайн, а внешний вид и стоимость одинаково важны. Жена считает, что качество изображения и стоимость – одинаково важны при выборе, но дизайн является существенно более важной характеристикой, чем стоимость (хотя качество изображение – существенно важнее дизайна).

Соответствующие матрицам сравнения векторы локальных приоритетов находятся следующим образом:

$$M(1,1) = \begin{bmatrix} 1 & 1/7 \\ 7 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 3,287 \\ 0,309 \end{bmatrix} \rightarrow \begin{bmatrix} 3,287 + 0,309 + 0,385 \\ 0,309 \\ 3,287 + 0,309 + 0,385 \\ 0,385 \\ 3,287 + 0,309 + 0,385 \end{bmatrix} \rightarrow \begin{bmatrix} 0,757 \\ 0,119 \\ 0,134 \end{bmatrix}$$

$$M(2,1) = \begin{bmatrix} 1 & 1/2 \\ 2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1,709 \\ 0,585 \end{bmatrix} \rightarrow \begin{bmatrix} 1,709 \\ 1 \\ 1,709 - 1 - 0,585 \\ 1,709 \\ 1,709 - 1 - 0,585 \end{bmatrix} \rightarrow \begin{bmatrix} 0,519 \\ 0,304 \\ 0,177 \end{bmatrix}$$

После парного сравнения критериев каждый эксперт составляет матрицы сравнения для имеющихся альтернатив (элементов третьего уровня), то есть определяет, насколько предпочтительнее является один телевизор по отношению к другому с точки зрения того или иного критерия.

*Матрицы сравнения эксперта 1 (мужа):*

- По критерию 1 (качество изображения);

$$M(3,1) = \begin{bmatrix} \text{A1} & \text{A2} \\ \text{A2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1/2 \\ 2 & 1 \end{bmatrix}, \quad M(3,2) = \begin{bmatrix} \text{A1} & \text{A2} \\ \text{A2} & 1 \end{bmatrix} = \begin{bmatrix} 0,507 & 0,333 \\ 0,333 & 1 \end{bmatrix}$$

- По критерию 2 (дизайн);

$$w(1,2) = \frac{1}{2/3} = 0.75, \quad w(1,3) = \frac{0.75}{0.75} = 0.95$$

- По критерию 3 (стоимость).

$$w(2,1) = \frac{1}{2/4} = 0.2, \quad w(2,3) = \frac{0.2}{0.6} = 0.33$$

По мнению эксперта 1, телевизор А обладает более предпочтительным дизайном, несколько лучшим качеством изображения, но имеет заметно более высокую стоимость.

Матрицы сравнения эксперта 2 (жены):

- По критерию 1 (качество изображения);

$$w(2,1) = \frac{1}{1/1} = 0.5, \quad w(2,3) = \frac{0.5}{0.9} = 0.56$$

- По критерию 2 (дизайн);

$$w(2,1) = \frac{1}{1/3} = 0.875, \quad w(2,3) = \frac{0.875}{0.723} = 1.21$$

- По критерию 3 (стоимость).

$$w(2,1) = \frac{1}{1/3} = 0.25, \quad w(2,3) = \frac{0.25}{0.78} = 0.32$$

Эксперт 2 не заметил особой разницы в качестве изображения телевизоров, однако считает, что телевизор А имеет значительно более привлекательный внешний вид, несмотря на несколько более высокую стоимость.

После завершения экспертных сравнений можно переходить к синтезу глобального вектора приоритетов. Общий вектор приоритетов для эксперта 1 вычисляется следующим образом:

$$w(1) = [w(1,1) \quad w(1,2) \quad w(1,3)] = [w(1,1) \quad w(1,2) \quad w(1,3)] = \begin{bmatrix} 0.667 \\ 0.333 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}$$

$$w(1) = \frac{1}{1+w(1,1)} = \frac{0.667}{0.667+0.333} = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}$$

Аналогично вычисляем общий вектор приоритетов для эксперта 2:

$$w(2) = [w(2,1) \quad w(2,2) \quad w(2,3)] = [w(2,1) \quad w(2,2) \quad w(2,3)] = \begin{bmatrix} 0.5 \\ 0.9 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix}$$



$$\begin{bmatrix} 0.275 & 0.275 \\ 0.275 & 0.275 \end{bmatrix} - \begin{bmatrix} 0.275 & 0.275 \\ 0.275 & 0.275 \end{bmatrix} = \begin{bmatrix} 0.275 & 0.275 \\ 0.275 & 0.275 \end{bmatrix} - \begin{bmatrix} 0.275 & 0.275 \\ 0.275 & 0.275 \end{bmatrix} = \begin{bmatrix} 0.275 & 0.275 \\ 0.275 & 0.275 \end{bmatrix}$$

Используя  $C(1)$  и  $C(2)$  можно вычислить глобальный вектор приоритетов:

$$\begin{bmatrix} 0.592 \\ 0.408 \end{bmatrix} = \begin{bmatrix} 0.592 \\ 0.408 \end{bmatrix} - \begin{bmatrix} 0.592 \\ 0.408 \end{bmatrix} = \begin{bmatrix} 0.592 \\ 0.408 \end{bmatrix}$$

Таким образом, вариант «телевизор А» является более предпочтительным ( $0.592 > 0.408$ ).

### Задание к лабораторной работе

1. Для проектирования ЛВС провести анализ предметной области, указанной в варианте задания:
  - Выделить основные подразделения исследуемой организации с указанием их основных задач и функций;
  - Сформулировать основные цели внедрения локальной вычислительной сети исходя из нужд исследуемой организации;
  - Выделить функционально-независимые группы пользователей ЛВС и указать для каждой из них перечень функций, которые должна обеспечивать компьютерная сеть.
  - Сформулировать общие требования, которым должна удовлетворять проектируемая локальная сеть (размер, структура, направление, характер и интенсивность информационных потоков и т.д.).
2. Предложить 3 различных варианта ЛВС, удовлетворяющих выдвинутым требованиям. Предложенные проекты могут отличаться по следующим параметрам:
  - Базовая топология сети или сегментов (шина, звезда, кольцо);
  - Применяемая сетевая технология (Ethernet, Token Ring);
  - Используемые каналы связи (витая пара, коаксиальный кабель, волоконно-оптический кабель, беспроводные каналы связи);
  - Метод организации управления ЛВС (одноранговая сеть, серверная сеть с «толстым» клиентом, серверная сеть с «тонким» клиентом);
  - Принимаемые меры по обеспечению информационной безопасности и защиты ЛВС от перебоев электропитания.
  - Используемая сетевая операционная система (Novel Netware, Windows Server).
3. Используя метод анализа иерархий провести оценку предложенных проектов ЛВС и выбрать оптимальный вариант.
  1. Назначить каждому члену бригады, выполняющей лабораторную работу, одну из ролей:
    - Технический директор – согласовывает с генеральным директором финансирование проектов, связанных с технической модернизацией, отвечает за эффективную работу технических и программных средств, осуществляет стратегическое планирование в соответствующей области;
    - Системный администратор – обеспечивает бесперебойную работу компьютерного и программного обеспечения, отвечает за информационную безопасность и сохранность данных, осуществляет тактическое планирование в соответствующей области;

- Разработчик информационных систем (для бригад из трех человек) – обеспечивает эффективную работу пользователей, отвечает за быстрый и надежный доступ к информации, осуществляет планирование развития информационных систем организации.
1. Построить иерархическую модель поставленной задачи принятия решения. Для определения критериев оценки ЛВС использовать указания к выполнению лабораторной работы.
  2. Задать матрицу сравнения, характеризующую степень относительного влияния мнения каждого эксперта на принятие окончательного решения.
  3. Каждому члену бригады, в соответствии с выбранной ролью, задать матрицу сравнения, характеризующую относительную важность используемых критериев. Для каждого критерия выполнить сравнение альтернативных вариантов ЛВС, используя информацию из указаний к выполнению лабораторной работы (в частности, таблицы 1 – 4).
  4. Для полученных матриц сравнения вычислить векторы соответствующих локальных приоритетов.
  5. В соответствии с алгоритмом МАИ синтезировать вектор глобальных приоритетов и определить оптимальный вариант ЛВС.
2. В отчете к лабораторной работе подробно отразить ход выполнения работы, в том числе иерархическую модель задачи принятия решений. Обязательно изложить сделанные выводы.

#### Варианты заданий

1. Информационная система для факультета университета.
2. Информационная система для филиала банка.
3. Информационная система для небольшого торгового предприятия.
4. Информационная система для поликлиники.
5. Информационная система для больницы.
6. Информационная система железнодорожной станции.
7. Информационная система для школы.
8. Информационная система для библиотеки.
9. Информационная система для юридической фирмы.

#### Контрольные вопросы

1. Понятие информационно-вычислительной сети. Виды ЛВС.
2. Основные этапы проектирования ЛВС. Принципы проектирования ЛВС.
3. Понятие и виды топологий.
4. Что такое одноранговая сеть?
5. Основные критерии оценки локальных вычислительных сетей.
6. Метод анализа иерархий.

#### Контрольные вопросы ЛР2(ОПК-3):

14. Что такое информационно-вычислительная сеть?
15. Что такое топология сети? Какие основные виды топологий сетей существуют?
16. Каким образом составляются различные конфигурации сетей? Какие сетевые устройства это реализуют?
17. Каким образом информация распространяется в сети? Для чего используется команда ping?
18. Как соотносятся понятия «сеть», «корпоративная сеть» и «подсеть»?
19. Что такое маска подсети и как она задается?
20. Что такое шлюзы подсети и для чего они используются?
21. Что такое DNS сервер подсети и для чего он используется?
22. Из чего состоит IP адрес конкретного ПК и как он задается?
23. Что такое рабочая группа, как ее создать и на что это влияет?

**Задание 1: «Защита ПК от несанкционированного доступа»**

Цель работы: Закрепление теоретического материала по изучению особенностей защиты ПК от несанкционированного доступа (НСД). Изучение способов и систем защиты ПК. Приборы и оборудование: Персональный компьютер ОС MS Windows 7 (MS Windows 10), MS Office, Браузер Microsoft Internet Explorer (Edge)

Пояснения к работе и задание:

Технические, организационные и программные средства обеспечения сохранности и защиты от несанкционированного доступа

Существует четыре уровня защиты компьютерных и информационных ресурсов: 1. Предотвращение предполагает, что только авторизованный персонал имеет доступ к защищаемой информации и технологии. 2. Обнаружение предполагает раннее раскрытие преступлений и злоупотреблений, даже если механизмы защиты были обойдены. 3. Ограничение уменьшает размер потерь, если преступление все-таки произошло, несмотря на меры по его предотвращению и обнаружению. 4. Восстановление обеспечивает эффективное воссоздание информации при наличии документированных и проверенных планов по восстановлению. 5. Меры защиты - это меры, вводимые руководством, для обеспечения безопасности информации. К мерам защиты относят разработку административных руководящих документов, установку аппаратных устройств или дополнительных программ, основной целью которых является предотвращение преступлений и злоупотреблений.

1. Аутентификация пользователей. Данная мера требует, чтобы пользователи выполняли процедуры входа в компьютер, используя это как средство для идентификации в начале работы. Для аутентификации личности каждого пользователя нужно использовать уникальные пароли, не являющиеся комбинациями личных данных пользователей, для пользователя. Необходимо внедрить меры защиты при администрировании паролей, и ознакомить пользователей с наиболее общими ошибками, позволяющими совершиться компьютерному преступлению. Если в компьютере имеется встроенный стандартный пароль, его нужно обязательно изменить.

2. Правила соблюдения защиты пароля Следующие правила полезны для защиты пароля: · нельзя делиться своим паролем ни с кем; · пароль должен быть трудно угадываемым; · для создания пароля нужно использовать строчные и прописные буквы, а еще лучше позволить компьютеру самому сгенерировать пароль; · не рекомендуется использовать пароль, который является адресом, псевдонимом, именем родственника, телефонным номером или чем-либо очевидным; · предпочтительно использовать длинные пароли, так как они более безопасны, лучше всего, чтобы пароль состоял из 6 и более символов; · пароль не должен отображаться на экране компьютера при его вводе; · пароли должны отсутствовать в распечатках; · нельзя записывать пароли на столе, стене или терминале, его нужно держать в памяти; · пароль нужно периодически менять и делать это не по графику; · на должности администратора паролей должен быть самый надежный человек; · не рекомендуется использовать один и тот же пароль для всех сотрудников в группе; · когда сотрудник увольняется, необходимо сменить пароль; · сотрудники должны расписываться за получение паролей.

3. Процедуры авторизации В организации, имеющей дело с критическими данными, должны быть разработаны и внедрены процедуры авторизации, которые определяют, кто из пользователей должен иметь доступ к той или иной информации и приложениям. В организации должен быть установлен такой порядок, при котором для использования компьютерных ресурсов, получения разрешения доступа к информации и приложениям, и получения пароля требуется разрешение тех или иных начальников. Если информация обрабатывается на большом вычислительном центре, то необходимо контролировать физический доступ к вычислительной технике. Могут оказаться уместными такие методы, как журналы, замки и пропуска, а также

охрана. Ответственный за информационную безопасность должен знать, кто имеет право доступа в помещения с компьютерным оборудованием и выгонять оттуда посторонних лиц.

Практическая часть: Выполнить программу на одном из языков программирования (например, PASCAL), осуществляющую функцию защиты файла паролем. Ход выполнения задания: 1. Составить алгоритм 2. Использовать условные операторы 3. Создать необходимые циклы, один из которых использует функцию сравнения пароля 1 цикл на запуск программы используя число ввода пароля до 3 4. Завершение программы неудачей, если число ввода неверного пароля превысило  $N=3$  5. Можете использовать следующие текстовые сообщения (примерные): - «ВВЕДИТЕ ПАРОЛЬ ДЛЯ ВХОДА В ПРОГРАММУ» (Начало выполнения загрузки) - «ПАРОЛЬ НЕВЕРНЫЙ! ИСПОЛЬЗУЙТЕ ЕЩЕ ОДНУ ПОПЫТКУ» (Если пароль введен некорректно) - ДОБРО ПОЖАЛОВАТЬ! (Если пароль введен корректно) - «ВЫ ПРЕВЫСИЛИ ДОПУСТИМОЕ ЧИСЛО ПОПЫТОК! ДО СВИДАНИЯ!» (Если количество неверных попыток ввода пароля превысило допустимое число  $N=3$ )

Ход выполнения лабораторной работы:

1. Изучить приведенный в методическом описании к лабораторной работе материал. 2. Ознакомиться с разделами с применением возможностей лаборатории. 3. Выполнить задание согласно практической части методического пособия. 4. Оформить отчет в установленной форме по разделам. 5. Ответить на контрольные вопросы. 6. Представить результаты работы преподавателю.

Контрольные вопросы:

1. Перечислите уровни защиты компьютерных и информационных ресурсов. 2. Сформулируйте функцию аутентификации и перечислите требования, предъявляемые к процедуре аутентификации. 3. Перечислите правила защиты пароля. Какие из них наиболее необходимы для выполнения? 4. Какие действия в рамках организационной защиты требуется выполнять для осуществления процедуры авторизации?

## **Задание 2 «Шифрование информации методом простой замены»**

Цель работы: 1. Закрепление теоретического материала на тему «Шифрование информации методом простой замены». 2. Получение шифротекста по исходным данным. 3. Получение исходного текста по заданному шифротексту и ключу.

Пояснения к работе: Сущность методов замены (подстановки) заключается в замене символов исходной информации, записанных в одном алфавите, символами из другого алфавита по определенному правилу. Самым простым является метод прямой замены. Символам  $So_i$  исходного алфавита  $A_o$ , с помощью которых записывается исходная информация, однозначно ставятся в соответствие символы  $Sl_i$  шифрующего алфавита  $A_i$ . В простейшем случае оба алфавита могут состоять из одного и того же набора символов. Например, оба алфавита могут содержать буквы алфавита кириллица. Задание соответствия между символами обоих алфавитов осуществляется с помощью преобразования числовых эквивалентов символов исходного текста  $To$ , длиной -  $K$  символов, по определенному алгоритму. Алгоритм моноалфавитной замены может быть представлен в виде последовательности шагов. Шаг 1. Формирование числового кортежа  $Loh$  путем замены каждого символа  $So_i$   $To$  ( $i=1,K$ ), представленного в исходном алфавите  $A_o$  размера  $[1xR]$ , на число  $hoi(so_i)$ , соответствующее порядковому номеру символа  $so_i$  в алфавите  $A_o$ . Шаг 2. Формирование числового кортежа  $L1h$  путем замены каждого числа кортежа  $Loh$  на соответствующее число  $h1i$  кортежа  $L1h$ , вычисляемое по формуле:

где  $k_1$  - десятичный коэффициент;  $k_2$  - коэффициент сдвига. Выбранные коэффициенты  $K_1, K_2$  должны обеспечивать однозначное соответствие чисел  $ho_i$  и  $h1_i$ , а при получении  $h1_i = 0$  выполнить замену  $h1_i = R$ . Шаг 3. Получение шифротекста  $T1$  путем замены каждого числа  $h1_i$  ( $sl_i$ ) кортежа  $L1h$  соответствующим символом  $sl_i$   $T_i$  ( $i=1,K$ ) алфавита шифрования  $A_1$  размера  $[1xR]$ . Шаг 4. Полученный шифротекст разбивается на блоки фиксированной длины  $b$ . Если

последний блок оказывается неполным, то в конец блока помещаются специальные символы-заполнители (например, символ \*).

Пример. Исходными данными для шифрования являются:

$T_0 = \langle \text{МЕТОД\_ШИФРОВАНИЯ} \rangle$ ;

$A_0 = \langle \text{АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩ ЪЫЬЭЮЯ} \rangle$ ;

$A_1 = \langle \text{ОРЩЬЯТЭ\_ЖМЧХАВДЫФКСЕЗПИЦГНЛТЬШБУЮ} \rangle$ ;

$R=32$ ;  $k_1=3$ ;  $k_2=15$ ,  $b=4$ .

Пошаговое выполнение алгоритма приводит к получению следующих результатов.

Шаг 1.  $L_{0h} = \langle 12, 6, 18, 14, 5, 32, 24, 9, 20, 16, 14, 3, 1, 13, 9, 31 \rangle$ .

Шаг 2.  $L_{1h} = \langle 19, 1, 5, 25, 30, 15, 23, 10, 11, 31, 25, 24, 18, 22, 10, 12 \rangle$ .

Шаг 3.  $T_1 = \langle \text{С О Я Г Б Д И М Ч У Г Ц К П М Х} \rangle$ .

Шаг 4.  $T_2 = \langle \text{СОЯГ БДИМ ЧУГЦ КПМХ} \rangle$ .

При расшифровании сначала устраняется разбиение на блоки. Получается непрерывный шифротекст  $T_i$  длиной  $K$  символов. Расшифрование осуществляется путем решения целочисленного уравнения:

При известных целых величинах  $k_i$ ,  $k_2$ ,  $h_{1i}$  и  $R$  величина  $h_{0i}$  вычисляется методом перебора  $n$ . Последовательное применение этой процедуры ко всем символам шифротекста приводит к его расшифрованию. По условиям приведенного примера может быть построена таблица замены, в которой взаимозаменяемые символы располагаются в одном столбце (табл. 1).

#### Таблица 1. Таблица замены

Использование таблицы замены значительно упрощает процесс шифрования. При шифровании символ исходного текста сравнивается с символами строки  $so_i$  таблицы. Если произошло совпадение в  $i$ -м столбце, то символ исходного текста заменяется символом из строки  $sl_j$ , находящегося в том же столбце  $i$  таблицы. Расшифрование осуществляется аналогичным образом, но вход в таблицу производится по строке  $sl_i$ .

Ход выполнения лабораторной работы:

1. Изучить приведенные в методическом описании к лабораторной работе материал и пример шифрования методом простой замены. 2. Получить у преподавателя исходный текст и ключ для шифрования. 3. Выполнить по шагам процедуру шифрования, полученный шифротекст представить в виде блоков информации. 4. Представить результаты преподавателю. 5. Получить у преподавателя шифротекст и ключ для расшифрования. 6. Выполнить по шагам процедуру расшифрования, полученный исходный текст представить преподавателю для проверки. 7. Оформить отчет в установленной форме. 8. Представить результаты работы преподавателю и защитить работу ответами на контрольные вопросы.

Контрольные вопросы:

1. Определение метода шифрования (шифра) 2. Понятие атаки на шифр (криптоанализа). 3. Понятие криптостойкости и требования, предъявляемые к криптостойкости. 4. Понятие и особенности метода простой замены. 5. Недостатки метода простой замены.

Контрольные вопросы ЛР3(ОПК-3):

1. Что такое защита информации на ПК и в информационной сети?
2. Назовите основные методы защиты информации на ПК и в информационных сетях?
3. Какие существуют основные (распространенные) методы криптографической защиты информации на ПК и в информационных сетях?
4. Как шифрование позволяет повысить безопасность компьютера?

5. Какие виды шифров вы знаете?
6. Приведите пример слабого шифра.
7. Симметричные криптосистемы: шифры перестановки.
8. Симметричные криптосистемы: шифры простой замены.
9. Симметричные криптосистемы: шифры сложной замены.
10. Симметричные криптосистемы: гаммирование.
11. Асимметричные криптосистемы, схема шифрования RSA, Диффи-Хеллмана, Эль-Гамаля

## Модуль 2.

### **Лабораторная работа №4. Алгоритмы разветвляющейся структуры. Алгоритмы циклической структуры. Изображение блок-схемы алгоритма согласно ГОСТ РФ.**

**Операции** - стандартные действия, разрешенные для переменных того или иного базового типа данных.

**Замечание:** Все перечисленные ниже операции (за исключением унарных '-' и 'not') требуют двух операндов.

1. Логические операции (and, or, not, xor) применимы только к значениям типа boolean. Их результатом также служат величины типа boolean. Приведем таблицы значений для этих операций:

not		and	true	false	or	true	false	xor	true	false
true	false	true	true	false	true	true	true	true	false	true
false	true	false	false	false	false	true	false	false	true	false

2. Операции сравнения (=, <, >, <=, >=) применимы ко всем базовым типам. Их результатами также являются значения типа boolean.

3. Операции целочисленной арифметики применимы, как легко догадаться, только к целым типам. Их результат - целое число, тип которого зависит от типов операндов.

**a div b** - деление a на b нацело (не нужно, наверное, напоминать, что деление на 0 запрещено, поэтому в таких случаях операция выдает ошибку). Результат будет принадлежать к типу данных, общему для тех типов, к которым принадлежат операнды. Например, (shortint div byte = integer). Пояснить это можно так: integer - это минимальный тип, подмножествами которого являются одновременно и byte, и shortint.

**a mod b** - взятие остатка при делении a на b нацело. Тип результата, как и в предыдущем случае, определяется типами операндов, а 0 является запрещенным значением для b. В отличие от математической операции mod, результатом которой всегда является неотрицательное число, знак результата "программистской" операции mod определяется знаком ее первого операнда. Таким образом, если в математике  $(-2 \bmod 5) = 3$ , то у нас  $(-2 \bmod 5) = -2$ .

**a shl k** - сдвиг значения a на k битов влево (это эквивалентно умножению значения переменной a на  $2^k$ ). Результат операции будет иметь тот же тип, что и первый ее операнд (a).

**a shr k** - сдвиг значения a на k битов вправо (это эквивалентно делению значения переменной a на  $2^k$  нацело). Результат операции будет иметь тот же тип, что и первый ее операнд (a).

**and, or, not, xor** - операции двоичной арифметики, работающие с битами двоичного представления целых чисел, по тем же правилам, что и соответствующие им логические операции.

4. Операции общей арифметики (+, -, \*, /) применимы ко всем арифметическим типам. Их результат принадлежит к типу данных, общему для обоих операндов (исключение составляет только операция дробного деления /, результат которой всегда относится к вещественному типу данных).

### Стандартные арифметические функции

К арифметическим операциям примыкают и стандартные арифметические функции. Их список с кратким описанием мы приводим в таблице.

	Описание	Тип аргумента	Тип результата
abs(x)	Абсолютное значение (модуль) числа	Арифметический	Совпадает с типом аргумента
arctan(x)	Арктангенс (в радианах)	Арифметический	Вещественный
cos(x)	Косинус (в радианах)	Арифметический	Вещественный
exp(x)	Экспонента ( $e^x$ )	Арифметический	Вещественный
frac(x)	Взятие дробной части числа	Арифметический	Вещественный
int(x)	Взятие целой части числа	Арифметический	Вещественный
ln(x)	Натуральный логарифм (по основанию e)	Арифметический	Вещественный
odd(x)	Проверка нечетности числа	Целый	boolean
pi	Значение числа	-	Вещественный
round(x)	Округление к ближайшему целому	Арифметический	Целый
trunc(x)	Округление "вниз" - к ближайшему меньшему целому	Арифметический	Целый
sin(x)	Синус (в радианах)	Арифметический	Вещественный
sqr(x)	Возведение в квадрат	Арифметический	Вещественный
sqrt(x)	Извлечение квадратного корня	Арифметический	Вещественный

### Арифметические выражения

Все арифметические операции можно сочетать друг с другом - конечно, с учетом допустимых для их операндов типов данных.

В роли операндов любой операции могут выступать переменные, константы, вызовы функций или выражения, построенные на основе других операций. Все вместе и называется выражением.

Примеры арифметических выражений:

$(x < 0)$  and  $(y > 0)$  - выражение, результат которого принадлежит к типу boolean;

$z$  shl abs(k) - вторым операндом является вызов стандартной функции;

$(x \bmod k) + \min(a, b) + \text{trunc}(z)$  - сочетание арифметических операций и вызовов функций;

odd(round(x/abs(x))) - "многоэтажное" выражение.

### Полнота вычислений

В общем случае вычисление сложного логического выражения прекращается в тот момент, когда его окончательное значение становится понятным (например, true or (b < 0)). Зачастую такой



подход позволяет заметно сэкономить на выполнении "лишних" действий. Скажем, если есть некоторая сложно вычисляемая функция `my_func`, вызов которой входит в состав выражения

`if (x<=0) and my_func(z+12),`

то для случая, когда `x` положительно, этих сложных вычислений можно избежать.

Однако включение директивы `{B+}` принудит компилятор завершить эти вычисления даже в таком случае. Ее выключение `{B-}` вернет обычную схему вычислений.

### Порядок вычислений

Если в выражении расставлены скобки, то вычисления производятся в порядке, известном всем еще с начальной школы: чем меньше глубина вложенности скобок, тем позже вычисляется заключенная в них операция. Если же скобок нет, то сначала вычисляются значения операций с более высоким приоритетом, затем - с менее высоким. Несколько подряд идущих операций одного приоритета вычисляются в последовательности "слева направо".

Таблица 2.1. Приоритеты (для всех) операций языка Pascal		
	Операции	Приоритет
Унарные операции	<code>+, -, not, @, ^, #</code>	Первый(высший)
Операции, эквивалентные умножению	<code>*, /, div, mod, and, shl, shr</code>	Второй
Операции, эквивалентные сложению	<code>+, -, or, xor</code>	Третий
Операции сравнения	<code>=, &lt;&gt;, &gt;, &lt;, &lt;=, &gt;=, in</code>	Четвертый

**Замечание:** Вызов любой функции имеет более высокий приоритет, чем все внешние относительно этого вызова операции. Выражения, являющиеся аргументами вызываемой функции, вычисляются в момент вызова.

Примеры выражений (с указанием последовательности вычислений) для целых чисел:

`a + b * c / d` (результат принадлежит к вещественному типу данных);

3 1 2

`a * not b or c * d = 0` (результат принадлежит к логическому типу данных);

2 1 4 3 5

`-min(a + b, 0) * (a + 1)` (результат принадлежит к целочисленному типу данных).

3 2 1 5 4

### Практическая часть

1. Написать программу вычисления объема цилиндра.
2. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей.
3. Написать программу, которая преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 12,5 должно быть преобразовано к виду 12 руб. 50 коп.

4. Написать программу вычисления силы тока в электрической цепи.
5. Написать программу вычисления площади поверхности цилиндра.
6. Написать программу пересчета веса из фунтов в килограммы (1фунт – это 405,9 грамма). Написать программу вычисления стоимости некоторого количества (по весу) яблок.
7. Написать программу вычисления стоимости некоторого количества (по объему) молока. Запишите в виде инструкции присваивания формулу вычисления объема цилиндра.
8. Составить программу, которая подсчитывает зарплату рабочего за определенный промежуток времени.
9. Проверьте, делится ли сумма трех произвольных чисел, введенных с клавиатуры, на первое число без остатка.
10. Запишите в виде инструкции присваивания формулу вычисления тока, по известным значениям напряжения и сопротивления электрической цепи.
11. Написать программу вычисления выражения  $y = 5x + 7z$ .
12. Запишите в виде инструкции присваивания формулу вычисления площади трапеции:  $s = \frac{a+b}{2}h$ , где  $a$  и  $b$  - длины оснований;  $h$  – высота трапеции.
13. Проверьте, является ли сумма четырех произвольных чисел, введенных с клавиатуры, нечетным числом.
14. Напишите программу вычисления площади прямоугольного треугольника.
15. Запишите в виде инструкции присваивания формулу вычисления площади круга:  $s = \pi r^2$ .
16. Написать программу вычисления выражения  $y = 7x + 3z$ .
17. Написать программу, которая преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 12,5 должно быть преобразовано к виду 12 руб. 50 коп.
18. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных резисторов:  $r = \frac{r_1 \cdot r_2}{r_1 + r_2}$ .

19. Проверьте, является ли произведение трех произвольных чисел, введенных с клавиатуры, уменьшенное в три раза, четным числом.
20. Написать программу, вычисляющую, скорость, с которой бегун пробежал дистанцию.
21. Написать программу вычисления стоимости покупки, состоящей из нескольких блокнотов и ручек.
22. Запишите в виде инструкции присваивания формулу вычисления сопротивления электрической цепи, состоящей из трех последовательно соединенных резисторов.
23. Написать программу вычисления расстояния между населенными пунктами, изображенными на карте.
24. Вычислить выражение  $y = 17x^3 + \frac{x-4}{25}$ , если  $12 \leq x \leq 42$ .
25. Даны два натуральных числа:  $m$ ,  $n$  (1..999999999), образующие дробь вида  $m/n$ . Сократить дробь, что бы числитель и знаменатель были взаимнопростые. Пример  $m = 256$ ;  $n = 64$ . Результат: 4 1.
26. Вычислить площадь треугольника со сторонами  $a$ ,  $b$ ,  $c$  ( $a$ ,  $b$ ,  $c > 0$ ) и полупериметром  $p$ .
27. Вычислите сдачу с покупки музыкального диска, если диск стоит  $m$  рублей, а у вас в кармане  $n$  рублей.
28. Составьте алгоритм и программу для определения сдачи после покупки в магазине товара: перчаток стоимостью  $A$  руб., портфеля стоимостью  $B$  руб., галстука стоимостью  $C$  руб. Исходная сумма, выделенная на покупку  $D$  руб. В случае нехватки денег сдача получится отрицательной.
29. Одна сторона прямоугольника на 5 см. длиннее другой, а сумма их длин равна 17 см. Найти стороны этого прямоугольника.
30. Одно число в 2 раза больше другого, а их сумма равна 93. Найти эти числа.
31. Дано число  $S$  (0..999999999), обозначающее количество секунд. Вычислить числа Hour, Minute (0..59), Second (0..59), показывающие число часов, минут и секунд соответственно в числе  $S$ .

32. Найти все целые корни уравнения  $ax^2+bx+c=0$ , где  $a, b, c$  - заданные целые числа.
33. Вычислить наибольший общий делитель двух натуральных чисел.
34. Вычислить среднее арифметическое и среднее геометрическое чисел  $a, b, c, d$ .
35. Вычислить площадь поверхности куба (длина ребра равна  $a$ ).
36. Проверьте, является ли произведение четырех произвольных чисел, введенных с клавиатуры, четным числом.

**Контрольные вопросы ЛР4(ОПК-3):**

1. Какими свойствами обладает алгоритм разветвляющейся структуры?
2. Что такое алгоритмы разветвляющейся структуры?
3. Что такое алгоритмы циклической структуры?
4. Изображение блок-схемы алгоритма согласно ГОСТ РФ.

## Теоретическая часть

Для того чтобы обработать несколько однотипных элементов, совершить несколько одинаковых действий и т.п., разумно воспользоваться оператором цикла - любым из четырех, который наилучшим образом подходит к поставленной задаче.

Оператор цикла повторяет некоторую последовательность операторов заданное число раз, которое может быть определено и динамически - уже во время работы программы.

**Замечание:** Алгоритмы, построенные только с использованием циклов, называются итеративными - от слова итерация, которое обозначает повторяемую последовательность действий.

### **for-to и for-downto**

В случае когда количество однотипных действий заранее известно (например, необходимо обработать все компоненты массива), стоит отдать предпочтение циклу с параметром (for).

### **Инкрементный цикл с параметром**

Общий вид оператора for-to:

for i:= first to last do <оператор>;

Счетчик i (переменная), нижняя граница first (переменная, константа или выражение) и верхняя граница last (переменная, константа или выражение) должны относиться к эквивалентным порядковым типам данных. Если тип нижней или верхней границы не эквивалентен типу счетчика, а лишь совместим с ним, то осуществляется неявное приведение: значение границы преобразуется к типу счетчика, в результате чего возможны ошибки.

Цикл for-to работает следующим образом:

1. вычисляется значение верхней границы last;
2. переменной i присваивается значение нижней границы first;
3. производится проверка того, что  $i \leq last$ ;
4. если это так, то выполняется <оператор>;
5. значение переменной i увеличивается на единицу;
6. пункты 3-5, составляющие одну итерацию цикла, выполняются до тех пор, пока i не станет строго больше, чем last; как только это произошло, выполнение цикла прекращается, а управление передается следующему за ним оператору.

Из этой последовательности действий можно понять, какое количество раз отработает цикл for-to в каждом из трех случаев:

- first < last: цикл будет работать last-first+1 раз;
- first = last: цикл отработает ровно один раз;
- first > last: цикл вообще не будет работать.

После окончания работы цикла переменная-счетчик может потерять свое значение. Таким образом, нельзя с уверенностью утверждать, что после того, как цикл завершил работу, обязательно окажется, что  $i = last + 1$ . Поэтому попытки использовать переменную-счетчик сразу после завершения цикла (без присваивания ей какого-либо нового значения) могут привести к непредсказуемому поведению программы при отладке.

### **Декрементный цикл с параметром**

Существует аналогичный вариант цикла `for`, который позволяет производить обработку не от меньшего к большему, а в противоположном направлении:

`for i:= first downto last do <оператор>;`

Счетчик  $i$  (переменная), верхняя граница `first` (переменная, константа или выражение) и нижняя граница `last` (переменная, константа или выражение) должны иметь эквивалентные порядковые типы. Если тип нижней или верхней границы не эквивалентен типу счетчика, а лишь совместим с ним, то осуществляется неявное приведение типов.

Цикл `for-downto` работает следующим образом:

1. переменной  $i$  присваивается значение `first`;
2. производится проверка того, что  $i \geq last$ ;
3. если это так, то выполняется `<оператор>;`
4. значение переменной  $i$  уменьшается на единицу;
5. пункты 2-4 выполняются до тех пор, пока  $i$  не станет меньше, чем `last`;

как только это произошло, выполнение цикла прекращается, а управление передается следующему за ним оператору.

Если при этом

- `first < last`, то цикл вообще не будет работать;
- `first = last`, то цикл отработает один раз;
- `first > last`, то цикл будет работать  $first - last + 1$  раз.

Замечание о неопределенности значения счетчика после окончания работы цикла справедливо и в этом случае.

### **while и repeat-until**

Если заранее неизвестно, сколько раз необходимо выполнить тело цикла, то удобнее всего пользоваться циклом с предусловием (`while`) или циклом с постусловием (`repeat-until`).

Общий вид этих операторов таков:

`while <условие_1> do <оператор>;`

`repeat <операторы> until <условие_2>;`

Условие окончания цикла может быть выражено переменной, константой или выражением, имеющим логический тип.

**Замечание:** Обратите внимание, что на каждой итерации циклы `for` и `while` выполняют только по одному оператору (либо группу операторов, заключенную в операторные скобки `begin-end` и потому воспринимаемую как единый составной оператор). В отличие от них, цикл `repeat-until` позволяет выполнить сразу несколько операторов: ключевые слова `repeat` и `until` сами служат операторными скобками.

Так же, как циклы `for-to` и `for-downto`, циклы `while` и `repeat-until` можно назвать в некотором смысле противоположными друг другу.

Последовательности действий при выполнении этих циклов таковы:

Для <b>while</b> :	Для <b>repeat-until</b> :
1. Проверяется, истинно ли <code>&lt;условие_1&gt;</code> .	1. Выполняются <code>&lt;операторы&gt;</code> .
2. Если это так, то выполняется <code>&lt;оператор&gt;</code> .	2. Проверяется, ложно ли <code>&lt;условие_2&gt;</code>
3. Пункты 1 и 2 выполняются до тех пор, пока <code>&lt;условие_1&gt;</code> не станет ложным.	3. Пункты 1 и 2 выполняются до тех пор, пока <code>&lt;условие_2&gt;</code> не станет истинным.

Таким образом, если `<условие_1>` изначально ложно, то цикл `while` не выполнится ни разу. Если же `<условие_2>` изначально истинно, то цикл `repeat-until` выполнится один раз.

### **break и continue**

Существует возможность прервать выполнение цикла (или одной его итерации), не дожидаясь конца его (или ее) работы.

**break** прерывает работу всего цикла и передает управление на следующий за ним оператор.

**continue** прерывает работу текущей итерации цикла и передает управление следующей итерации (цикл `repeat-until`) или на предшествующую ей проверку (циклы `for-to`, `for-downto`, `while`).

**Замечание:** При прерывании работы циклов `for-to` и `for-downto` с помощью функции `break` переменная цикла (счетчик) сохраняет свое текущее значение, не "портится".

### **Оператор безусловного перехода goto**

Возвращаясь к сказанному об операторе `goto`, необходимо отметить, что при всей его нежелательности все-таки существует ситуация, когда предпочтительно использовать именно этот оператор - как с точки зрения структурированности текста программы, так и с точки зрения логики ее построения, и уж тем более с точки зрения уменьшения трудозатрат программиста. Эта ситуация - необходимость передачи управления изнутри нескольких вложенных циклов на самый верхний уровень.

Дело в том, что процедуры `break` и `continue` прерывают только один цикл - тот, в теле которого они содержатся. Поэтому в упомянутой выше ситуации пришлось бы заметно усложнить текст программы, вводя много дополнительных прерываний. А один оператор `goto` способен заменить их все.

Сравните, например, два программно-эквивалентных отрывка:

```

write('Матрица ');
for i:=1 to n do
begin
  flag:=false;
  for j:=1 to m do
    if a[i,j]>a[i,i]
    then begin flag:=true;
           write('не ');
           break;
        end
  if flag then break;
end;
writeln('обладает свойством
        диагонального
        преобладания.');
```

```

write('Матрица ');
for i:=1 to n do
  for j:=1 to m do
    if a[i,j]>a[i,i]
    then begin
      write('не ');
      goto 1;
    end;
1: writeln('обладает
           свойством
           диагонального
           преобладания.');
```

### Пример использования циклов

**Задача.** Вычислить интеграл в заданных границах  $a$  и  $b$  для некоторой гладкой функции  $f$  от одной переменной (с заданной точностью).

**Алгоритм.** Метод последовательных приближений, которым мы воспользуемся для решения этой задачи, состоит в многократном вычислении интеграла со все возрастающей точностью, - до тех пор, пока два последовательных результата не станут различаться менее чем на заданное число (скажем,  $\text{eps} = 0,001$ ). Количество приближений нам заранее неизвестно (оно зависит от задаваемой точности), поэтому здесь годится только цикл с условием (любой из них).

Вычислять одно текущее значение для интеграла мы будем с помощью метода прямоугольников: разобьем отрезок  $[a,b]$  на несколько мелких частей, каждую из них дополним (или урежем - в зависимости от наклона графика функции на данном участке) до прямоугольника, а затем просуммируем получившиеся площади. Количество шагов нам известно, поэтому здесь удобнее всего воспользоваться циклом с параметром.

На нашем рисунке изображена функция  $f(x) = x^2$  (на отрезке  $[1,2]$ ). Каждая из криволинейных трапеций будет урезана (сверху) до прямоугольника: высотой каждого из них послужит значение функции на левом конце участка. График станет "ступенчатым".

### Реализация

```

step:= 1;
h:= b-a;
s_nov:= f(a)*h;
repeat
  s_star:= s_nov;
  s_nov:= 0;
  step:= step*2;
  h:= h/2;
  for i:= 1 to step do
    s_nov:= s_nov+f(a+(step-1)*h);
  s_nov:= s_nov*h;
until abs(s_nov - s_star)<= eps;
writeln(s_nov);
```



## Практическая часть

Написать программы, используя циклы **while**, **repeat-until** и цикл с параметром.

1. Вычислить выражение  $Y = X/2 + 3X/4 + 5X/6 + \dots + 17X/18$ .

2. Вывести на печать целые положительные кратные 9, числа пока истинно условие  $m \leq 100$ .

3. Дано натуральное число N. Вычислить сумму первых N- слагаемых

4. Вычислить  $1/1 + 3/2 + 5/3 + 7/4 + \dots$

5. Вывести 10 раз на печать Фамилию Имя Отчество.

6. Вычислить выражение  $y = \sum_{k=1}^5 5 + k$ .

7. Вводится последовательность ненулевых чисел, 0-конец последовательности.

Найти наименьшее число.

8. Вычислить выражение  $y = \sum_{k=2}^5 5 * k$ .

9. Вывести на печать целые положительные кратные 5 числа пока истинно условие  $Z \leq 100$ .

10. Вычислить выражение  $y = \sum_{x=1}^5 \ell^{5x}$ .

11. Составить программу, которая вводит 20 символов.

12. Вычислить выражение  $y = x + \frac{x^2}{2} + \frac{x^3}{3}$ .

13. Вводятся числа до тех пор, пока не введено число 10. Найти сумму чисел.

14. Вычислить выражение  $y = 100 + 5 \sum_{k=1}^{10} k^2$ .

15. Вывести на печать целые положительные кратные 10 числа пока истинно условие  $k \leq 100$ .

16. Вычислить выражение  $y = \sum_{m=2}^{20} (2m - 40)$ .

17. Вывести на печать целые положительные кратные 8 числа пока истинно условие  $k \leq 100$ .

18. Вычислить выражение  $y = \sum_{x=1}^{10} (\operatorname{tg} x + 5)$ .

19.Вычислить значение функции  $Y = (X + 3)$  в точках от 1 до 5 с шагом 0,5.

20.Вычислить выражение  $y = \sum_{x=1}^4 5x^2$ .

21.Вводятся целые положительные числа, пока их сумма меньше 100. Найти количество целых чисел.

22.Вычислить выражение  $y = 50 - \sum_{c=5}^{10} (5 - c)$ .

23.Вычислить выражение  $y = \sum_{z=1}^5 (45 + z^3)$ .

24.Вывести на печать целые положительные числа кратные 3, пока истинно условие  $X \leq 100$ .

25.Вычислить выражение  $y = \sum_{k=2}^5 \frac{5 * k^2}{4}$ .

26.Вычислить  $Y = 4a + b$ , если  $A = 1, 2, \dots, 10$  и  $b$  вводится пользователем.

27.Вычислить выражение  $y = \sum_{k=1}^5 (50 + k)$ .

28.Вводятся числа до тех пор, пока не введено отрицательное число. Найти максимальное число.

29.Вычислить выражение  $y = \sum_{x=1}^{10} (x + 5)^2$ .

30.Вывести на печать простые положительные числа, пока истинно условие  $X \leq 20$ .

31.Вычислить выражение  $y = \sum_{x=1}^5 \frac{(x + 3)^3}{10}$ .

32.Вводится последовательность ненулевых чисел, 0-конец последовательности. Найти максимальное число.

33.Вычислить  $n! = 1 * 2 * 3 * \dots * n$ .  $N$  – вводится.

34.Вычислить  $y = 4(1000 - \sum (k^2 - 1))$ .

35.Вычислить  $y = 8 \sum_{i=1}^8 i + 8$ .

36.Вычислить  $y = 10 + \sum_{k=1}^5 k^3$ .

**Контрольные вопросы ЛР5(ОПК-3):**

5. Что значит Ввод/вывод данных?
6. Что значит программная реализация линейных алгоритмов в интегрированной среде разработки?
7. Что значит отладка (тестирование) программы?

## Задание 1: Программирование одномерных массивов (векторов).

1. Ознакомиться с теоретическими основами составления линейных алгоритмов и программ.
2. Выбрать вариант задания.
3. Используя методику решения задач на ЭВМ (прил.1) составить и отладить программу на языке высокого уровня (ЯВУ) Turbo Pascal согласно выбранному варианту задания.
4. Выполнить несколько вычислений с помощью составленной программы и оценить устойчивость решений к вариации исходных данных и ограничений точности полученных решений.
5. Заполнить отчет по лабораторной работе и представить его для защиты (проверки).

## Содержание отчета

- Тема, цель выполняемой работы.
- Формулировка задания согласно варианту.
- Составить блок-схему решаемой задачи.
- Записать код программы, реализующей вариант задания.
- Привести результаты решения задачи и сделать выводы по использованным конструкциям ЯВУ Turbo Pascal и сложности составленной программы, а также по устойчивости и достоверности полученных результатов.

## Теоретические основы выполнения задания

# Массивы

**Массив** представляет собой структуру, состоящую из фиксированного числа компонент одного типа. В качестве компонент можно использовать как ранее описанные типы, так и следующие: массивы, записи, множества, указатели и т.п. Число элементов в массиве фиксируется при описании и далее при выполнении программы не меняется.

Определение типа, значения которого являются массивами, выполняется следующим образом:

**TYPE** <имя типа> = **ARRAY**[<диапазон первого индекса>, ...,  
                                  <диапазон n-го индекса>] **OF** <тип компонент>;

Количество индексов  $n$  определяет размерность массива, а сами индексы разделяются запятыми и заключаются в квадратные скобки.

Пример:

```
TYPE MATR=ARRAY[1..2,1..12] OF REAL;
VAR A, B, C: MATR;
```

Массив можно описать в разделе VAR следующим образом:

<идентификатор>: ARRAY [<диапазон первого индекса>, ..., <диапазон n-го индекса>] OF <тип компонент>;

Пример:

VAR A, B, C: ARRAY[1..10] OF INTEGER;

Для обращения к элементам массива используются конкретные значения индексов. Индекс представляет собой выражение любого простого (скалярного) типа (кроме REAL). К примеру, оператор  $B[3] := 10$ ; присваивает третьему элементу одномерного массива с именем B значение 10.

**Пр и м е р.**

Пусть двумерный массив описан следующим образом:

$\text{VAR } A : \text{ARRAY}[1..2, 1..4] \text{ OF INTEGER}$ ; а в памяти ЭВМ записана таблица чисел, представляющая этот массив:

17	11	4	5
22	8	16	12

Все элементы в таблице имеют тип integer. При обращении к элементам матрицы A первый индекс указывает номер строки таблицы (изменяется в данном случае от 1 до 2), второй – номер столбца (в нашем примере изменяется от 1 до 4). Если задать оператор присваивания в виде  $X := A[2,3]$ ; то после его выполнения значение некоторой переменной X будет равно 16.

Ввод и вывод значений элементов массива производится поэлементно.

Рассмотрим несколько типичных задач, связанных с применением массивов.

1. { Программа, позволяющая найти сумму элементов одномерного массива }

```

program msg1;
const n=15; {число элементов массива}
var      a : array [1..n] of real;
        summa: real;

begin
  summa := 0;
  for i:=1 to n do
    begin
      readln (a[i]);
      summa := summa+a[i]
    end;
  writeln ('сумма ', n, ' элементов массива равна ', summa)
end.

```

## **Варианты заданий**

1. Найти N элементов массива X, в котором  $X_1 = X_2 = X_3 = 2$ ; а все последующие элементы вычисляются по формуле:  $X_k = X_{k-2} - X_{k-3} + 1/K$ .

2. Вычислить значения элементов массива Z по формуле :  
 $Z = \cos X + \operatorname{tg} X$ , где X меняется на отрезке [1;15] с шагом 0,92

3. Вычислить и напечатать значения функции  $Y = A_k^2 + A_k - \sin A_k$  где элементы массива A вводятся с клавиатуры .

4. Рассчитать N значений элементов массива B по формуле :

$$B_k = \begin{cases} K + \cos(K-1) ; & \text{ïðå } K \text{ íà } \text{òòîí} \\ \sin K + 3 ; & \text{ïðå } \hat{E} \text{ ð } \text{òòîí} \end{cases}$$

5. Найти сумму положительных значений элементов массива W, вводимого с клавиатуры.

6. Составить массив из положительных значений функции  $Z = \cos X * \sin X$  для X, изменяющегося на отрезке [-5,10] с шагом 0,67.



24. Составить массив В из отрицательных значений функции  $Z = \cos(X) / \sin(X - 2)$  для X, изменяющегося на отрезке  $[5; -10]$  с шагом 0,67 и найти его сумму

25. Вычислить последовательность N чисел Фибоначчи  $F_0 = F_1 = 1$ ;  $F_{i+1} = F_i + F_{i-1}$  и записать ее в массив. Найти сумму чисел с нечетными номерами.

26. Вычислить N элементов массива X,  $X_k = X_{k-1} + (1/2)X_{k-2}$   $X_1 = 3, X_2 = 0,2$  и найти их сумму.

27. Написать программу нахождения элементов массивов X и Y, пользуясь формулами:  $X_k = 3X_{k-1} + K$ ,  $Y_k = X_{k-1} + Y_{k-1}$ ,  $X_0 = Y_0 = 1$  и найти их сумму.

28. Найти N элементов массива  $X_1 = X_2 = X_3 = 1$ ;  $X_k = X_{k-1} + X_{k-3} - 1/K$  и найти их сумму.

## Задание 2: Вложенные циклы. Двумерные массивы (матрицы).

1. Ознакомиться с теоретическими основами составления линейных алгоритмов и программ.
2. Выбрать вариант задания.
3. Используя методику решения задач на ЭВМ (прил.1) составить и отладить программу на языке высокого уровня (ЯВУ) Turbo Pascal согласно выбранному варианту задания.
4. Выполнить несколько вычислений с помощью составленной программы и оценить устойчивость решений к вариации исходных данных и ограничений точности полученных решений.
5. Заполнить отчет по лабораторной работе и представить его для защиты (проверки).

## Содержание отчета

- Тема, цель выполняемой работы.
- Формулировка задания согласно варианту.
- Составить блок-схему решаемой задачи.
- Записать код программы, реализующей вариант задания.
- Привести результаты решения задачи и сделать выводы по использованным конструкциям ЯВУ Turbo Pascal и сложности составленной программы, а также по устойчивости и достоверности полученных результатов.

### **Теоретические основы выполнения задания**

## Массивы

**Массив** представляет собой структуру, состоящую из фиксированного числа компонент одного типа. В качестве компонент можно использовать как ранее описанные типы, так и следующие: массивы, записи, множества, указатели и т.п. Число элементов в массиве фиксируется при описании и далее при выполнении программы не меняется.

Определение типа, значения которого являются массивами, выполняется следующим образом:

**TYPE** <имя типа> = **ARRAY**[<диапазон первого индекса>, ...,  
                                         <диапазон n-го индекса>] **OF** <тип компонент>;

Количество индексов n определяет размерность массива, а сами индексы разделяются запятыми и заключаются в квадратные скобки.

2. { Программа поиска наибольшего элемента одномерного массива и его порядкового номера }

```

program msg;
const n=20;
var      a:=array [1..n] of real;
         amax: real;
         i, ne: integer;
begin
write ('введите элемент 1 '); readln (a[1]);
amax := a[1]; ne:=1;
for i:=2 to n do
    begin
    write ('введите элемент № ', i);    readln (a[i]);
    if a[i] > amax then begin
        amax:=a[i]; ne:=i
    end
    end;
writeln ('максимальный элемент равен ', amax);
writeln ('и имеет порядковый номер ', ne)
end.

```

3. { Программа нахождения произведения двух матриц

*A* размером  $M*N$  и *B* размером  $N*K$  . Элементы результирующей матрицы *C* размером  $M*K$

рассчитываются по формуле  $C_{ij} = \sum_{k=1}^N a_{ik} \cdot b_{kj}$  . Значения  $M, N, K$  не превышают 10}

```

program msg3;
const em=10;
type matr=array[1..em, 1...em] of real;
var a, b, c: matr;
    m, n, k, i, j, t: integer;
begin
write('введите значения m, n, k '); readln (m, n, k);
{ ввод элементов матрицы a }
for i:=1 to m do
    for j:=1 to n do
        readln (a[i, j]);
{ ввод элементов матрицы b }
for i:=1 to n do
    for j:=1 to k do
        readln (b[i, j]);
{ умножение матриц }
for i:=1 to m do
    for j:=1 to k do
        begin
            c[i, j]:=0;
            for t:=1 to n do
                c[i, j]:=c[i, j]+a[i, t]* b[t, j];
            end
        end
    end
end

```



end.

В Паскале разрешается присваивать значения одной переменной массива другой (если элементы массива имеют один тип и одинаковую размерность). К примеру, если массивы A и B имеют одинаковую размерность и тип элементов REAL, то допустимо присваивание: A := B.

### Варианты заданий

1. Вычислить сумму элементов каждого столбца матрицы A(M,N).

$$Z = \sum_{i=1}^{10} \sum_{k=1}^N \frac{\sin^k x(i)}{k}, \text{ где } X(I) \text{ заданы массивом}$$

2. Вычислить значение функции

(X(1), X(2), ..., X(10)), K=1, 2, ..., N.

3. Вычислить значение функции 
$$Z(j) = \prod_{i=1}^{20} (1 + 1/e^i + x(j)),$$

где X(J) заданы массивом (x(1), x(2), ..., x(N)). Результаты запомнить в массиве Z.

4. Вычислить сумму элементов матрицы A(N,N), расположенных над главной диагональю.

5. Найти сумму положительных элементов каждого столбца матрицы X(M,N)

6. Вычислить сумму элементов матрицы A(N,N), расположенных под главной диагональю.

7. Из матрицы X(M,N) построить матрицу Y, поменяв местами строки и столбцы.

8. Определить количество положительных и отрицательных элементов матрицы A(M,N).

9. Определить количество положительных элементов каждого столбца матрицы A(M,N) и запомнить их в массиве R.

10. Переписать первые элементы каждой строки матрицы a(M,N) в массив B

11. Даны элементы массива A, состоящего из n элементов. Вычислить  $S = A_1^1 + A_2^2 + \dots + A_n^n$  без операций возведения в степень

12. Вычислить значение функции 
$$Z = \sum_{i=1}^M \sum_{j=1}^N \frac{1}{i + j^2}.$$

13. Вычислить значение функции 
$$Z = \sum_{i=1}^M \sum_{j=1}^N \sin(i^3 + j^4).$$

14. Задана матрица A(M,M). Разделить элементы каждой строки матрицы A на соответствующий диагональный элемент.

15. Вычислить значение функции 
$$Z = \sum_{i=1}^M \sum_{j=1}^N \frac{1}{2j + i}.$$

16. Дано натуральное число N. Вычислить 
$$\sum_{K=1}^N K(K+1) \dots (K+K).$$

17. Определить количество положительных элементов каждой строки матрицы A(M,N) и запомнить их в массиве B.

18. Дано натуральное число N. Вычислить 
$$\sum_{k=1}^N \frac{1}{2^{(k)}}.$$

19. Дано натуральное число N. Вычислить  $\sum_{k=1}^N (-1)^k (2k^2 + 1)!$ .
20. Вычислить суммы элементов каждой строки матрицы X(N,N), и записать их в массив Y(N).
21. Даны натуральное число N, действительное число x. Вычислить  $\frac{1}{N!} \sum_{k=1}^N (-1)^k \frac{x^k}{(k! + 1)!}$ .
22. Даны натуральное число N, действительное число X. вычислить  $\sum_{k=1}^N k^k x^{2k-1}$  без операции возведения в степень
23. Даны натуральное число N, действительное число x. Вычислить  $\sum_{k=1}^N \sum_{m=k}^N \frac{x + k}{m}$ .
24. Заданы матрица A(5,6) и вектор B(5). Разделить каждый элемент k - ой строки матрицы A на элемент B(K).
25. Заданы матрицы A(m,m) и B(m,m). Получить матрицу X(M,2M), состоящую из M столбцов матрицы A и M столбцов матрицы B.
26. Вычислить значение функции  $Z = \sum_{i=1}^M \sum_{j=1}^N \frac{j - 1 + 1}{i + j}$
27. Найти сумму положительных элементов каждой строки матрицы X(M,N).

### Контрольные вопросы ЛР6(ОПК-3):

1. Программная реализация алгоритмов циклической и смешанной структуры.
2. Отладка (тестирование) программы.
3. Особенности программной реализации алгоритмов циклической и смешанной структуры.
4. Что такое отладка программ?
5. Что такое контрольный вариант расчета?