ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
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

«ИНФОРМАТИКА»

Кафедра **«Информатика и вычислительная техника»**

Направление подготовки **связи** **11.03.02. Инфокоммуникационные технологии и системы**

Профили: **Многоканальные телекоммуникационные системы, Сети связи и системы коммутации, Системы радиосвязи и радиодоступа, Защищенные системы и сети связи**

Разработала:
Доцент кафедры ИВТ Швидченко С.А.

Ростов-на-Дону
2019

Методические указания
для проведения практических занятий
по дисциплине
«Информатика»

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

Рассмотрено и одобрено
на заседании кафедры «ИВТ»
Протокол от «26» августа 2019 г., № 1.

Модуль 3.

Лабораторная работа №13. Классификация и формы представления моделей. Информационная модель объекта.

В среде MS Excel создать информационную модель простейшей ЛВС.

Исходные данные:

1.1 Число серверов – 1.

1.2 Число отделов – 4.

1.3 Число ПЭВМ (рабочих станций, персональных компьютеров - PC) в каждой рабочей группе равно 4+i.

1.4 Занимаемая полоса пропускания линии связи при придаче данныхв процессе общения пары клиент – сервер равна:

- в направлении ПЭВМ – сервер – 1,35 МБит/с;
- в направлении сервер – ПЭВМ – 5МБит/с.

1.5Занимаемая полоса пропускания линии связи при придаче данныхв процессе общения пары клиент – сервер равна:

- в направлении ПЭВМ -Internet - 1,9 МБит/с;
- в направлении Internet – ПЭВМ – 3,8 МБит/с.

1.6Занимаемая полоса пропускания линии связи при придаче данныхмежду компьютерами – 1,5МБит/с.

Для представленных данных необходимо:

1. Построить схему сети;
2. Создать шаблон позволяющий рассчитывать значения трафика в каждой линии связи.
3. Вывести на график пропускные способности в различных линиях связи.
4. Увеличивая параметры занимаемой полосы пропускания на 30%, с шагом 0.1, представить изменение пропускной способности в точках подключения к серверу и выходу в интернет.
5. Сделать выводы о возможности процессов и целесообразности этого процесса.

Контрольные вопросы ЛР13(УК-1):

1. Классификация моделей.
2. Основные этапы разработки и исследования моделей на компьютере.
3. Сущность математического моделирования.
4. Сущность компьютерного моделирования.
5. Создать простейшие модели объектов и процессов в виде изображений и чертежей, дина-мических (электронных) таблиц.
6. Провести компьютерные эксперименты с использованием готовых моделей объектов и процессов.

Лабораторная работа №14. 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 $x_k, y_k, k = 1,2,\cdots,n,$ 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 таблице1.

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 ЛР14(УК-1):

1. Что такое Математическое моделирование?
2. Что такое Компьютерное моделирование?
3. Создать простейшие модели объектов и процессов в виде изображений и чертежей, дина-мических (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?

Лабораторная работа №15. Вычислительные сети. Основные понятия. Построение и компоненты. Основные топологии.

.

Цель работы: знать назначение и классификацию программного обеспечения вычислительных сетей, основные возможности сетевых операционных сред, уметь использовать некоторые сетевые прикладные программные пакеты для решения сетевых задач.

1.2 Указания по оформлению отчета:

Отчет должен содержать: титульный лист, цель работы; ответы на контрольные вопросы; выводы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Компьютерной сетью называют совокупность узлов (компьютеров, терминалов, периферийных устройств), имеющих возможность информационного взаимодействия друг с другом с помощью специального коммуникационного оборудования и программного обеспечения.

Средства передачи и обработки информации ориентированы в ней на коллективное использование общесетевых ресурсов – информационных, программных, аппаратных.

Компьютерные сети могут работать в различных режимах: обмена данными между абонентами сети, запроса и выдачи информации, сбора информации пакетной обработки данных по запросам пользователей с удаленных терминалов, в диалоговых режимах.

Таким образом, с появлением сетей ЭВМ разрешены две очень важные проблемы:

1) обеспечение в принципе неограниченного доступа к ЭВМ пользователей независимо от территориального расположения,

2) возможность оперативного перемещений больших массивов информации на любые расстояния, позволяющий своевременно получать данные для принятия тех или иных решений.

Использование вычислительных сетей дает предприятию следующие возможности:

1. Разделение дорогостоящих ресурсов;
2. Улучшение доступа к информации;
3. Быстрое и качественное принятие решений;
4. Совершенствование коммуникаций;
5. Свобода в территориальном размещении компьютеров.

Программное обеспечение сетей ЭВМ в расширенном варианте составляют:

1) сетевые операционные системы;

2) сетевые драйвера, протоколы, службы и другое дополнительное программное обеспечение сетевых интерфейсов;

3) прикладное сетевое программное обеспечение.

Под сетевыми операционными системами понимают такие операционные системы, которые обеспечивают пользователям распределенный доступ к сетям ЭВМ.

Во вторую группу входит большой круг всевозможного программного обеспечения в основном изготовителя данного интерфейса (сетевой платы, модема и т.п.) для обеспечения правильной работы сетевого устройства.

При этом под драйвером понимается программа, непосредственно взаимодействующая с интерфейсом - сетевым адаптером и операционной системой (ОС). Драйвер сетевого адаптера

взаимодействует с ОС через систему протоколов и служб, которые могут находится как в самих ОС, так и поставляться вместе с устройством.

При этом под сетевым протоколом понимается набор правил поведения сетевых узлов при передаче-приеме информации.

Под сетевыми службами понимается набор программного обеспечения сетевого обеспечения узкоспециального назначения, например:

- клиенты сетей - позволяют подключаться, обозревать и пользоваться сетевыми ресурсами соответствующих сетей,
- службы контроля трафика сетей,
- службы использования доступа к разделяемым ресурсам,
- доменные службы и др.

Круг прикладного сетевого программного обеспечения составляют всевозможные сетевые приложения.

Каждый компьютер работает под управлением собственной операционной системы, а какая-либо «общая» операционная система, распределяющая работу между компьютерами сети, отсутствует. Взаимодействие между компьютерами сети про-исходит за счет передачи сообщений через сетевые адаптеры и каналы связи. С помощью этих сообщений один компьютер обычно запрашивает доступ к локальным ресурсам другого компьютера. Такими ресурсами могут быть как данные, хранящиеся на диске, так и разнообразные периферийные устройства — принтеры, модемы, факс-аппараты и т.д. Разделение локальных ресурсов каждого компьютера между всеми пользователями сети — основная цель создания вычислительной сети.

Каким же образом сказывается на пользователе тот факт, что его компьютер подключен к сети? Прежде всего, он может пользоваться не только файлами, дисками, принтерами и другими ресурсами своего компьютера, но и аналогичными ресурсами других компьютеров, подключенных к той же сети. Правда, для этого недостаточно снабдить компьютеры сетевыми адаптерами и соединить их кабельной системой. Необходимы еще некоторые добавления к операционным системам этих компьютеров. На тех компьютерах, ресурсы которых должны быть доступны всем пользователям сети, необходимо добавить модули, которые постоянно будут находиться в режиме ожидания запросов, поступающих по сети от других компьютеров.

Обычно такие модули называются программными серверами (server), так как их главная задача — обслуживать (serve) запросы на доступ к ресурсам своего компьютера. На компьютерах, пользователи которых хотят получать доступ к ресурсам других компьютеров, также нужно добавить к операционной системе некоторые специальные программные модули, которые должны вырабатывать запросы на доступ к удаленным ресурсам и передавать их по сети на нужный компьютер. Такие модули обычно называют программными клиентами (client).

Собственно же сетевые адаптеры и каналы связи решают в сети достаточно простую задачу — они передают сообщения с запросами и ответами от одного компьютера к другому, а основную

работу по организации совместного использования ресурсов выполняют клиентские и серверные части операционных систем.

Пара модулей «клиент – сервер» обеспечивает совместный доступ пользователей к определенному типу ресурсов, например к файлам. В этом случае говорит, что пользователь имеет дело с файловой службой (service). Обычно сетевая операционная система поддерживает несколько видов сетевых служб для своих пользователей — файловую службу, службу печати, службу электронной почты, службу удаленного доступа и т. п.

Термины «клиент» и «сервер» используются не только для обозначения про-граммных модулей, но и компьютеров, подключенных к сети. Если компьютер предоставляет свои ресурсы другим

компьютерам сети, то он называется сервером, а если он их потребляет — клиентом. Иногда один и тот же компьютер может одновременно играть роли и сервера, и клиента.

Сетевые службы всегда представляют собой распределенные программы, состоящие из нескольких взаимодействующих частей, причем каждая часть, как правило, выполняется на отдельном компьютере сети.

До сих пор речь шла о системных распределенных программах. Однако в сети могут выполняться и распределенные пользовательские программы - приложения. Распределенное приложение также состоит из нескольких частей, каждая из которых выполняет какую-то определенную законченную работу по решению прикладной задачи. Например, одна часть приложения, выполняющаяся на компьютере пользователя, может поддерживать специализированный графический интерфейс, вторая - работать на мощном выделенном компьютере и заниматься статистической обработкой введенных пользователем данных, а третья - заносить полученные результаты в базу данных на компьютере с установленной стандартной СУБД. Распределенные приложения в полной мере используют потенциальные возможности распределенной обработки, предоставляемые вычислительной сетью, и поэтому часто называются сетевыми приложениями.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Охарактеризовать сетевые операционные системы по следующей схеме:
1) платность,
2) доступ к исходному коду,
3) многоплатформенность,
4) мультизадачность,
5) количество пользователей,
6) функции управления сетью,
7) интерфейс работы,
8) потребляемые ресурсы.

Контрольные вопросы ЛР 15(УК-1):

1. Что такое вычислительная сеть?
2. Что такое компоненты вычислительных сетей?
3. Что такое построение вычислительных сетей?
4. Что такое информационно-вычислительная сеть?
5. Что такое топология сети? Какие основные виды топологий сетей существуют?
6. Каким образом составляются различные конфигурации сетей? Какие сетевые устройства это реализуют?
7. Каким образом информация распространяется в сети? Для чего используется команда ping?
8. Как соотносятся понятия «сеть», «корпоративная сеть» и «подсеть»?
9. Что такое маска подсети и как она задается?
10. Что такое шлюзы подсети и для чего они используются?
11. Что такое DNS сервер подсети и для чего он используется?
12. Из чего состоит IP адрес конкретного ПК и как он задается?
13. Что такое рабочая группа, как ее создать и на что это влияет?

Лабораторная работа №16. Изучение построения и компонент вычислительных сетей.

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 топологи: 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 удается 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 брандмауеров;
- 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. иямласти 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 находится 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 эксперто*в 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 *попарные 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×n, где n – число сравниваемых элементов. Элемент $a_{ij} = a(i, j)$ указанной матрицы выражает результат сравнения элементов *i* и *j*. Если при сравнении элементов *i* и *j* получено $a(i,j)=b$, то результатом сравнения элементов *j* и *i* должно быть $a(j,i)=1/b$. 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 8×8 (для третьего уровня иерархии) и 8 матриц размером 3×3 (для трех возможных вариантов ЛВС). Общее количество матриц сравнения для рассматриваемой задачи: 1 + m×(1+8) = 1+9×m, где m – количество участвующих экспертов.

Если E_1^j, …, $E_{n(j)}^j$ – обозначения элементов иерархии j-того уровня, а $n(j)$ – их количество (j = 2, 3, …, k), то матрицы сравнения j-того уровня можно обозначить как $M(i_2, i_3 \ldots, i_{j-2}, i_{j-1})$, где $i_2 \in [1, n(2)]$,…, $i_{j-2} \in [1, n(j-2)]$, $i_{j-1} \in [1, n(j-1)]$ – каждая матрица соответствует фиксированному набору $(i_2, i_3, \ldots, i_{j-2}, i_{j-1})$ элементов иерархии вышерасположенных 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:

$$v = \begin{bmatrix} \sqrt[n]{a(1,1) \times \ldots a(1,n)} \\ \ldots \\ \sqrt[n]{a(n,1) \times \ldots a(n,n)} \end{bmatrix} \quad (1)$$

Тогда вектор w будет определяться следующим образом:

$$w = \begin{bmatrix} \frac{v_1}{\sum_{i=1}^{n} v_n} \\ \ldots \\ \frac{v_n}{\sum_{i=1}^{n} v_n} \end{bmatrix} \quad (v_1,\ldots,v_n \text{ – элементы вектора } v) \quad (2)$$

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: $W(i_2, i_3, \ldots, i_{k-2}, i_{k-1}) = w(i2, i_3, \ldots, i_{k-2}, i_{k-1})$ $\forall\, i_2 = \overline{1, n(2)}, i_3 = \overline{1, n(3)}, \ldots, i_{k-2} = \overline{1, n(k-2)}, i_{k-1} = \overline{1, n(k-1)}$.

2. На *j*-том уровне иерархии ($2 \le j < k$) для всех наборов элементов иерархии вышестоящих уровней $(i_2, i_3, \ldots, i_{j-2}, i_{j-1})$ можно составить матрицу $C(i_2, i_3, \ldots, i_{j-2}, i_{j-1})$, размера $n(k) \times n(j)$, столбцами которой являются общие векторы приоритетов следующего (j+1-ого) уровня:

$$C(i_2, i_3, \ldots, i_{j-2}, i_{j-1}) = \left| W(i_2, i_3, \ldots, i_{j-2}, i_{j-1}, 1) \quad \ldots \quad W(i_2, i_3, \ldots, i_{j-2}, i_{j-1}, n(j)) \right| \tag{3}$$

В этом случае общие векторы приоритетов *j*-того уровня будут вычисляться как произведение матрицы $C(i_2, i_3, \ldots, i_{j-2}, i_{j-1})$ на соответствующий вектор локальных приоритетов:

$$W(i_2, i_3, \ldots, i_{j-2}, i_{j-1}) = C(i_2, i_3, \ldots, i_{j-2}, i_{j-1}) \times w(i_2, i_3, \ldots, i_{j-2}, i_{j-1}) \tag{4}$$

Размерность вектора $W(i_2, i_3, \ldots, i_{j-2}, i_{j-1})$ 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.

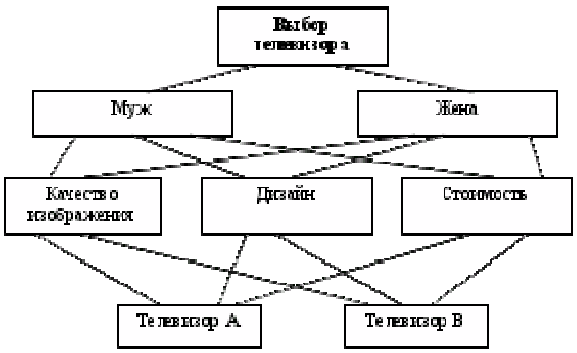


Рис. 1. Пример иерархии объектов для задачи принятия решения

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

$$M = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad w = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

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:

$$M(1) = \begin{matrix} \\ \textit{кач.изображения} \\ \textit{дизайн} \\ \textit{стоимость} \end{matrix} \begin{bmatrix} 1 & 7 & 5 \\ 1/7 & 1 & 1 \\ 1/5 & 1 & 1 \end{bmatrix} \quad M(2) = \begin{matrix} \\ \textit{кач.изображения} \\ \textit{дизайн} \\ \textit{стоимость} \end{matrix} \begin{bmatrix} 1 & 5 & 1 \\ 1/5 & 1 & 5 \\ 1 & 1/5 & 1 \end{bmatrix}$$

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:

$$v(1) = \begin{bmatrix} \sqrt[3]{35} \\ \sqrt[3]{1/7} \\ \sqrt[3]{0.2} \end{bmatrix} = \begin{bmatrix} 3{,}267 \\ 0{,}523 \\ 0{,}585 \end{bmatrix} \quad w(1) = \begin{bmatrix} \dfrac{3{,}267}{3{,}267+0{,}523+0{,}585} \\ \dfrac{0{,}523}{3{,}267+0{,}523+0{,}585} \\ \dfrac{0{,}585}{3{,}267+0{,}523+0{,}585} \end{bmatrix} \approx \begin{bmatrix} 0{,}747 \\ 0{,}119 \\ 0{,}134 \end{bmatrix}$$

$$v(2) = \begin{bmatrix} \sqrt[3]{5} \\ \sqrt[3]{1} \\ \sqrt[3]{0.2} \end{bmatrix} = \begin{bmatrix} 1{,}709 \\ 1 \\ 0{,}585 \end{bmatrix} \quad w(2) = \begin{bmatrix} \dfrac{1{,}709}{1{,}709+1+0{,}585} \\ \dfrac{1}{1{,}709+1+0{,}585} \\ \dfrac{0{,}585}{1{,}709+1+0{,}585} \end{bmatrix} \approx \begin{bmatrix} 0{,}519 \\ 0{,}304 \\ 0{,}177 \end{bmatrix}$$

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);

$$M(1,1) = \begin{matrix} \\ \textit{телевизор } A \\ \textit{телевизор } B \end{matrix} \begin{bmatrix} 1 & 2 \\ 1/2 & 1 \end{bmatrix}, \quad w(1,1) = \begin{bmatrix} 0.667 \\ 0.333 \end{bmatrix}$$

- По критерию 2 (дизайн);

$$M(1,2)=\begin{bmatrix}1 & 3\\ 1/3 & 1\end{bmatrix},\quad w(1,2)=\begin{bmatrix}0.75\\ 0.25\end{bmatrix}$$

- По критерию 3 (стоимость).

$$M(1,3)=\begin{bmatrix}1 & 1/4\\ 4 & 1\end{bmatrix},\quad w(1,3)=\begin{bmatrix}0.2\\ 0.8\end{bmatrix}$$

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);

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

- По критерию 2 (дизайн);

$$M(2,2)=\begin{bmatrix}1 & 7\\ 1/7 & 1\end{bmatrix},\quad w(2,2)=\begin{bmatrix}0.875\\ 0.125\end{bmatrix}$$

- По критерию 3 (стоимость).

$$M(2,3)=\begin{bmatrix}1 & 1/3\\ 3 & 1\end{bmatrix},\quad w(2,3)=\begin{bmatrix}0.25\\ 0.75\end{bmatrix}$$

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:

$$C(1)=[W(1,1)\quad W(1,2)\quad W(1,3)]=[w(1,1)\quad w(1,2)\quad w(1,3)]=\begin{bmatrix}0.667 & 0.75 & 0.2\\ 0.333 & 0.25 & 0.8\end{bmatrix}$$

$$W(1)=C(1)\times w(1)=\begin{bmatrix}0.667 & 0.75 & 0.2\\ 0.333 & 0.25 & 0.8\end{bmatrix}\cdot\begin{bmatrix}0.747\\ 0.119\\ 0.134\end{bmatrix}=\begin{bmatrix}0.615\\ 0.385\end{bmatrix}$$

Аналогично вычисляем общий вектор приоритетов для эксперта 2:

$$C(2)=[W(2,1)\quad W(2,2)\quad W(2,3)]=[w(2,1)\quad w(2,2)\quad w(2,3)]=\begin{bmatrix}0.5 & 0.875 & 0.25\\ 0.5 & 0.125 & 0.75\end{bmatrix}$$

$$W(2) = C(2) \times w(2) = \begin{bmatrix} 0.5 & 0.875 & 0.25 \\ 0.5 & 0.125 & 0.75 \end{bmatrix} \cdot \begin{bmatrix} 0.519 \\ 0.304 \\ 0.177 \end{bmatrix} = \begin{bmatrix} 0.569 \\ 0.431 \end{bmatrix}$$

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

$$C = [W(1) \quad W(2)] = \begin{bmatrix} 0.615 & 0.569 \\ 0.385 & 0.431 \end{bmatrix}$$

$$W = C \times w = \begin{bmatrix} 0.615 & 0.569 \\ 0.385 & 0.431 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.592 \\ 0.408 \end{bmatrix}$$

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 ЛР16(ОПК-3):

1. Что такое информационно-вычислительная сеть?
2. Что такое топология сети? Какие основные виды топологий сетей существуют?
3. Каким образом составляются различные конфигурации сетей? Какие сетевые устройства это реализуют?
4. Каким образом информация распространяется в сети? Для чего используется команда ping?
5. Как соотносятся понятия «сеть», «корпоративная сеть» и «подсеть»?
6. Что такое маска подсети и как она задается?
7. Что такое шлюзы подсети и для чего они используются?
8. Что такое DNS сервер подсети и для чего он используется?
9. Из чего состоит IP адрес конкретного ПК и как он задается?
10. Что такое рабочая группа, как ее создать и на что это влияет?

Лабораторная работа №17. Основные методы защиты информации в компьютерных сетях. Виды электронной подписи.

Задание 1 Защита документов, созданных в Microsoft Word.

Цель работы: Изучение возможностей современных средств защиты документов, созданных в среде приложений OS Microsoft. Закрепление теоретического материала. Изучение способов и систем защиты файлов и файловых систем.

Учебно-наглядные пособия и ТСО: ПЭВМ, ОС Windows, пакет Microsoft Office, методическое пособие. ЗАДАНИЕ

Создать документ в приложении Word офисного пакета Microsoft Office Используя свойства и возможности приложения Word, защитить созданный файл паролем. Используя настройки атрибутов файла, ограничить доступ к файлу пользователей сети. Используя настройки атрибутов файла, сделать файл «скрытым».

Задание 2 Защита документов, созданных в Microsoft Excel.

Цель работы: Изучение возможностей современных средств защиты документов, созданных в среде приложений OS Microsoft. Закрепление теоретического материала. Изучение способов и систем защиты файлов и файловых систем.

Учебно-наглядные пособия и ТСО: ПЭВМ, ОС Windows, пакет Microsoft Office, методическое пособие. ЗАДАНИЕ
Создать документ в приложении Excel офисного пакета Microsoft Office Используя свойства и возможности приложения Excel, защитить созданный файл паролем. Используя настройки атрибутов файла, ограничить доступ к файлу пользователей сети. Используя настройки атрибутов файла, сделать файл «скрытым».

Ход выполнения лабораторной работы:
1. Изучить приведенный в методическом описании к лабораторной работе материал. 2. Ответить на контрольные вопросы. 3. Представить отчет по лабораторной работе преподавателю.

Контрольные вопросы:
1. Перечислите свойства файлов (документов), создаваемых в MS Office. 2. В чем заключаются особенности файлов (документов), создаваемых в MS Word? 3. Перечислите порядок установки пароля к созданному файлу в MS Word.

Задание 3 Защита документов, созданных в Microsoft Access. Защита файла паролем.

Цель работы: Изучение возможностей современных средств защиты документов, созданных в среде приложений OS Microsoft. Закрепление теоретического материала. Изучение способов и систем защиты файлов и файловых систем.

Учебно-наглядные пособия и ТСО: ПЭВМ, ОС Windows, пакет Microsoft Office, методическое пособие. ЗАДАНИЕ 1
Создать документ в приложении Access пакета Microsoft Office Используя свойства и возможности приложения Access, защитить созданный файл паролем. Используя настройки атрибутов файла, ограничить доступ к файлу пользователей сети. Используя настройки атрибутов файла, сделать файл «скрытым».

ЗАДАНИЕ 2

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. Выполнить распечатку кода программы, оформить отчет в установленной форме по разделам. 3. Ответить на контрольные вопросы. 4. Представить программу по разделу 2 и отчет по лабораторной работе преподавателю.

Контрольные вопросы:

1. Перечислите свойства файлов (документов), создаваемых в MS Office. 2. В чем заключаются особенности файлов (документов), создаваемых в MS Access? 3. Перечислите порядок установки пароля к созданному файлу в MS Access. 4. Назовите назначение основных модулей в блок-схеме алгоритма разработанной Вами программы. 5. При каких условиях в общем случае может быть обеспечен доступ к сетевому ресурсу?

Контрольные вопросы ЛР 17(ОПК-3):

1. Основные методы защиты информации в компьютерных сетях.
2. Виды электронной подписи.
3. Что такое методы защиты информации в компьютерных сетях?
4. Симметричное и асимметричное кодирование.
5. Открытый и закрытый коды.
6. Что такое электронная подпись?
7. Виды электронной подписи.

Лабораторная работа №18. Изучение методов защиты информации в компьютерных сетях. Электронная подпись.

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 делится 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 Soi исходного алфавита Ao, с помощью которых записывается исходная информация, однозначно ставятся в соответствие символы S1i шифрующего алфавита Ai. 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 Soi To (i=l,K), представленного в исходном алфавите Ao размера [lxR], на число hoi(soi), соответствующее порядковому номеру символа soi в алфавите Ao. Шаг 2. Формирование числового кортежа L1h путем замены каждого числа кортежа Loh на соответствующее число h1i кортежа L1h, вычисляемое по формуле:

где ki - десятичный коэффициент; к2 - коэффициент сдвига. Выбранные коэффициенты K1, K2 должны обеспечивать однозначное соответствие чисел hoi и h1i, а при получении h1i = 0 выполнить замену h1i = R. Шаг 3. Получение шифротекста T1 путем замены каждого числа h1i(s1i) кортежа L1h соответствующим символом s1i Ti (i=l,K) алфавита шифрования A1 размера [1XR]. Шаг 4. Полученный шифротекст разбивается на блоки фиксированной длины b. Если

последний блок оказывается неполным, то в конец блока помещаются специальные символызаполнители (например, символ *).

Пример. Исходными данными для шифрования являются:

T0 = <МЕТОД_ШИФРОВАНИЯ>;

Ao = <АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩ ЪЫЬЭЮЯ >;

A1 = <ОРЩЬЯТЭ_ЖМЧХАВДЫФКСЕЗПИЦГНЛЪШБУЮ>;

R=32; k1=3; k2=15, b=4.

Пошаговое выполнение алгоритма приводит к получению сле¬дующих результатов.

Шаг1. L0h = <12,6,18,14,5,32,24,9,20,16,14,3,1,13,9,31>.

Шаг 2. L1h = <19,1,5,25,30,15,23,10,11,31,25,24,18,22,10,12>.

Шаг 3. T1 =<С О Я Г Б Д И М Ч У Г Ц К П М Х >.

Шаг4. T2 = <СОЯГ БДИМ ЧУГЦ КПМХ>.

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

При известных целых величинах ki, k2, h1i и R величина hoi 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 soi таблицы. Если произошло совпадение в i-м столбце, то символ исходного текста заменяется символом из строки s1j, находящегося в том же столбце i таблицы. Расшифрование осуществляется аналогичным образом, но вход в таблицу производится по строке s1i.

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 шифровании, 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 ЛР 18 (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 4.

Лабораторная работа №19. Разработка алгоритмов линейной, разветвляющейся, циклической структуры. Графическая реализация. ЕСПД.

Линейный алгоритм. Ввод и вывод информации

Теоретическая часть

Алгоритм - это последовательность действий, которые необходимо выполнить, чтобы решить поставленную задачу.

Программа же представляет собой набор команд на языке, понятном исполнителю, реализующий некоторый алгоритм. В нашем случае исполнителем является компьютер, а языком программирования будет язык высокого уровня Pascal. К сожалению, любой язык высокого уровня удобен только человеку, пишущему или отлаживающему программу, но совершенно непонятен компьютеру. Программа на таком языке называется исходным текстом и хранится во внешнем файле с расширением .pas.

Для перевода программы на язык низкого уровня, понятный исполнителю-компьютеру, существуют специальные программы-переводчики - компиляторы. Результатом работы компилятора (иными словами, результатом процесса компиляции) является исполняемый код, который записывается в файл с расширением .exe.

Линейным принято называть вычислительный процесс, в котором этапы вычислений выполняются в линейной последовательности и каждый этап выполняется только один раз. На схеме блоки размещаются сверху вниз в порядке их выполнения. Для таких процессов характерно, что направление вычислений не зависит от исходных данных или промежуточных результатов.

Линейные процессы имеют место, например, при вычислении арифметических выражений.

В состав среды разработчика Turbo Pascal входят:

- текстовый редактор, в котором можно набирать тексты программ;
- компилятор, превращающий исходные тексты в исполняемый код;
- отладчик, помогающий обнаруживать и исправлять ошибки в программе

Из многочисленных возможностей, предоставляемых средой Turbo Pascal, мы упомянем лишь самые важные - те, без которых написание программ становится совсем уж затруднительным.

- Нажатие клавиш F1, Alt+F1, Ctrl+F1 открывает экранную подсказку.
- Нажатие клавиши F2 позволяет сохранить исходный текст программы.
- Нажатие клавиши F3 открывает диалог выбора нужного файла (по умолчанию, отображаются только файлы с расширением .pas).
- Нажатие клавиши Alt+F5 показывает консоль (см. п. "Ввод и вывод: консоль" ниже) с результатами работы программы.

- Нажатие клавиши Ctrl+F9 начинает процесс выполнения программы Если она еще не была откомпилирована, предварительно будет вызван компилятор
- Клавиши F7 и F8 обеспечивают трассировку - пошаговое выполнение программы, позволяющее проследить за процессом ее выполнения.
- Дополнительное окно Debug/Watch показывает текущее состояние выбранных переменных.

Любая Pascal-программа может состоять из следующих **блоков** (напомним, что квадратными скобками здесь и далее помечены необязательные части):

```
program <имя_программы>;
[ uses <имена_подключаемых_модулей>;]
[ label <список_меток>;]
[ const <имя_константы> = <значение_константы>;]
[ type <имя_типа> = <определение_типа>;]
[ var <имя_переменной> : <тип_переменной>;]
[ procedure <имя_процедуры> <описание_процедуры>;]
[ function <имя_функции> <описание_функции>;]
begin   {начало основного тела программы}
<операторы>
end.  (* конец основного тела программы *)
```

Сразу же необходимо сделать важную оговорку: поздние версии компиляторов языка Pascal уже не требуют указывать название программы, то есть строку

```
program <имя_программы>;
```

вообще говоря, можно опустить. Но это возможно только в том случае, если вся программа содержится в одном модуле-файле. Если же программа состоит из нескольких самостоятельных кусков - модулей, то каждый из них должен иметь заголовок (program или unit).

Комментарии

Помимо отступов, большие логически замкнутые блоки программы удобно разделять строками-комментариями, содержащими информацию о смысле последующего блока. Комментарий - это строка (или несколько строк) из произвольных символов, заключенная в фигурные скобки:

```
{ комментарий }
```

Другой вариант оформления комментария:

```
(* комментарий *)
```

Внутри самого комментария символы } или *) встречаться не должны.

Во время компилирования программы комментарии игнорируются. Следовательно, их можно добавлять в любом месте программы. Можно даже разорвать оператор вставкой комментария. Кроме того, все, что находится после ключевого слова end., завершающего текст программы, компилятор тоже воспринимает как комментарий.

Переменные и типы данных

Переменная - это программный объект, значение которого может изменяться в процессе работы программы.

Тип данных - это характеристика диапазона значений, которые могут принимать переменные, относящиеся к этому типу данных.

Все используемые в программе переменные должны быть описаны в специальном разделе **var** по следующему шаблону:

var <имя_переменной_1> [, <имя_переменной_2, _>] : <имя_типа_1>;

<имя_переменной_3> [, <имя_переменной_4, _>] : <имя_типа_2>;

Для удобства программистов в языке Pascal существует множество стандартных типов данных и плюс к тому возможность создавать новые типы.

Конструируя новые типы данных на основе уже имеющихся (стандартных или опять-таки определенных самим программистом), нужно помнить, что любое здание должно строиться на хорошем фундаменте. Поэтому сейчас мы и поговорим об этом "фундаменте".

На основании базовых типов данных строятся все остальные типы языка Pascal, которые так и называются: конструируемые.

Разделение на базовые и конструируемые типы данных в языке Pascal показано в таблице:

<table>
<tr><td rowspan="2">Базовые типы данных</td><td colspan="3">Порядковые (дискретные) типы данных</td><td rowspan="2">Адресные типы данных</td><td rowspan="2">Структурированные типы данных</td></tr>
<tr><td rowspan="4">Логический
<code>boolean</code></td><td rowspan="4">Символьный (литерный)
<code>char</code></td><td>Арифметические типы данных</td></tr>
<tr><td>Целые
<code>shortint
byte
integer
word
longint</code></td><td>Вещественные
<code>real
single
double
extended
comp</code></td></tr>
<tr><td rowspan="5">Конструируемые типы</td><td>Нетипизированный указатель
<code>pointer</code></td><td></td></tr>
<tr><td colspan="2" rowspan="2">Перечисляемый
<code>week = (su, mo, tu,
we, th, fr,sa);</code></td><td rowspan="4">Типизированный указатель
<code>^<тип></code></td><td>Массив array
Строка string</td></tr>
<tr><td>Запись record</td></tr>
<tr><td colspan="2" rowspan="2">Интервал (диапазон)
<code>budni = mo..fr;</code></td><td>Файл
<code>text
file</code></td></tr>
<tr><td>Процедурный
Объектный</td></tr>
<tr><td colspan="5">Типы данных, конструируемые программистом</td></tr>
</table>

Типы данных, конструируемые программистом, описываются в разделе type по следующему шаблону:

type <имя_типа> = <описание_типа>;

Например:

type lat_bukvy = 'a'..'z','A'..'Z';

Базовые типы данных являются стандартными, поэтому нет нужды описывать их в разделе type. Однако при желании это тоже можно сделать, например, дав длинным определениям короткие имена. Скажем, введя новый тип данных

type int = integer;

можно немного сократить текст программы.

Порядковые типы данных

Среди базовых типов данных особо выделяются порядковые типы. Такое название можно обосновать двояко:

1. Каждому элементу порядкового типа может быть сопоставлен уникальный (порядковый) номер. Нумерация значений начинается с нуля. Исключение - типы данных shortint, integer и longint. Их нумерация совпадает со значениями элементов.

2. Кроме того, на элементах любого порядкового типа определен порядок (в математическом смысле этого слова), который напрямую зависит от нумерации. Таким образом, для любых двух элементов порядкового типа можно точно сказать, который из них меньше, а который - больше.

Стандартные подпрограммы, обрабатывающие порядковые типы данных

Только для величин порядковых типов определены следующие функции и процедуры:

1. Функция **ord(x)** возвращает порядковый номер значения переменной x (относительно того типа, к которому принадлежит переменная x).
2. Функция **pred(x)** возвращает значение, предшествующее x (к первому элементу типа неприменима).
3. Функция **succ(x)** возвращает значение, следующее за x (к последнему элементу типа неприменима).
4. Процедура **inc(x)** возвращает значение, следующее за x (для арифметических типов данных это эквивалентно оператору x:=x+1).
5. Процедура **inc(x,k)** возвращает k-е значение, следующее за x (для арифметических типов данных это эквивалентно оператору x:=x+k).
6. Процедура **dec(x)** возвращает значение, предшествующее x (для арифметических типов данных это эквивалентно оператору x:=x-1).
7. Процедура **dec(x,k)** возвращает k-е значение, предшествующее x (для арифметических типов данных это эквивалентно оператору x:=x-k).

Целочисленные типы данных

Тип данных	Количество		Диапазон	
	байтов	битов		
shortint	1	8	-128..127	$-2^7..2^7-1$

byte	1	8	0..255	$0..2^8-1$
integer	2	16	-32768..32767	$-2^{15}..2^{15}-1$
word	2	16	0..65535	$0..2^{16}-1$
longint	4	32	-2147483648..2147483647	$-2^{31}..2^{31}-1$

Вещественные типы данных

Напомним, что эти типы данных являются арифметическими, но не порядковыми.

Тип	Количество байтов	Диапазон (абсолютной величины)
single	4	$1.5*10^{-45}..3.4*10^{38}$
real	6	$2.9*10^{-39}..1.7*10^{38}$
double	8	$5.0*10^{-324}..1.7*10^{308}$
extended	10	$3.4*10^{-4932}..1.1*10^{4932}$
comp	8	$-2^{63}+1..2^{63}-1$

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

Операторы

Оператором называется (минимальная) структурно законченная единица программы.

Важно! Все операторы языка Pascal должны заканчиваться знаком ";" (точка с запятой), и ни один оператор не может разрываться этим знаком. Единственная возможность не ставить после оператора ";" появляется в том случае, когда сразу за этим оператором следует ключевое слово end.

К простейшим операторам языка Pascal относятся:

1. a:= b; - присваивание переменной a значения переменной b. В правой части присваивания может находиться переменная, константа, арифметическое выражение или вызов функции.

2. ; - пустой оператор, который можно вставлять куда угодно, а также вычеркивать откуда угодно, поскольку на целостность программы это никак не влияет.

3. Операторные скобки, превращающие несколько операторов в один:

4. begin

5. <несколько операторов>

end;

Везде далее, где в записи конструкций языка Pascal мы будем использовать обозначение <один_оператор>, его следует понимать как "один оператор или несколько операторов, заключенные в операторные скобки begin - end".

Ввод и вывод: консоль

Любой алгоритм должен быть результативным. В общем случае это означает, что он должен сообщать результат своей работы потребителю: пользователю-человеку или другой программе (например, программе управления принтером). Мы не будем описывать здесь внутренние автоматические процессы, использующие сигналы непрерывно функционирующих программ, а сосредоточим внимание на взаимодействии программы и человека, то есть на процессах ввода информации с клавиатуры и вывода ее на экран.

В программировании существует специальное понятие консоль, которое обозначает клавиатуру при вводе и монитор при выводе.

Ввод с консоли

Для того чтобы получить данные, вводимые пользователем вручную (то есть с консоли), применяются команды

read(<список_ввода>) и readln(<список_ввода>).

Первая из этих команд считывает все предложенные ей данные, оставляя курсор в конце последней строки ввода, а вторая - сразу после окончания ввода переводит курсор на начало следующей строки. В остальном же их действия полностью совпадают.

Список ввода - это последовательность имен переменных, разделенных запятыми. Например, при помощи команды

```
readln(k,x,c,s);     {k:byte; x:real; c:char; s:string}
```

программа может получить с клавиатуры данные сразу для четырех переменных, относящихся к различным типам данных.

Вводимые значения необходимо разделять пробелами, а завершать ввод - нажатием клавиши Enter. Ввод данных заканчивается в тот момент, когда последняя переменная из списка ввода получила свое значение. Следовательно, вводя данные при помощи приведенной выше команды, вы можете нажать Enter четыре раза - после каждой из вводимых переменных, - либо же только один раз,

предварительно введя все четыре переменные в одну строчку (обязательно нужно разделить их пробелами).

Типы вводимых значений должны совпадать с типами указанных переменных, иначе возникает ошибка. Поэтому нужно внимательно следить за правильностью вводимых данных.

Вообще, вводить с клавиатуры можно только данные базовых типов (за исключением логического). Если же программе все-таки необходимо получить с консоли значение для boolean-величины, придется действовать более хитро: вводить оговоренный символ, а уже на его основе присваивать логической переменной соответствующее значение. Например:

```
repeat
  writeln('Согласны ли Вы с этим утверждением? y - да, n - нет');
  readln(c);          {c:char}
  case c of
    'y': b:= true;
    'n': b:= false;
     else writeln('Ошибка!');
   end;
until (c='n')or(c='y');
```

Второе исключение: строки, хотя они и не являются базовым типом, вводить тоже разрешается. Признаком окончания ввода строки является нажатие клавиши Enter, поэтому все следующие за нею переменные необходимо вводить с новой строчки.

Вывод на консоль

Сделаем одно важное замечание: ожидая от человека ввода с клавиатуры, не нужно полагать, что он окажется ясновидящим и просто по мерцанию курсора на черном экране догадается, какого типа переменная нужна ожидающей программе. Старайтесь всегда придерживаться правила: "лысый" ввод недопустим! Перед тем как считывать что-либо с консоли, необходимо сообщить пользователю, что именно он должен ввести: смысл вводимой информации, тип данных, максимальное и минимальное допустимые значения и т.п.

Примером неплохого приглашения служит, скажем, такая строчка:

Введите два вещественных числа (0.1<x,y<1000000) - длины катетов.

Впрочем, и ее можно улучшить, сообщив пользователю не только допустимый диапазон ввода, но и ожидаемую точность (количество знаков после запятой).

Средства, позволяющие организовать выдачу информации на экран, мы здесь и рассмотрим.

Для того чтобы вывести на экран какое-либо сообщение, воспользуйтесь процедурой write(<список_вывода>) или writeln(<список_вывода>).

Первая из них, напечатав на экране все, о чем ее просили, оставит курсор в конце выведенной строки, а вторая переведет его в начало следующей строчки.

Список вывода может состоять из нескольких переменных, записанных через запятую; все эти переменные должны иметь тип либо базовый, либо строчный. Например, writeln(a,b,c);

Форматный вывод

Если для вывода информации воспользоваться командой, приведенной в конце предыдущего пункта, то выводимые символы окажутся "слепленными". Чтобы этого не случилось, нужно либо позаботиться о пробелах между выводимыми переменными:

writeln(a,' ',b,' ',c);

либо задать для всех (или хотя бы для некоторых) переменных формат вывода:

writeln(a:5,b,c:20:5);

Первое число после знака ":" обозначает количество позиций, выделяемых под всю переменную, а второе - под дробную часть числа. Десятичная точка тоже считается отдельным символом.

Если число длиннее, чем отведенное под него пространство, количество позиций будет автоматически увеличено. Если же выводимое число короче заданного формата, то спереди к нему припишутся несколько пробелов. Таким образом можно производить вывод красивыми ровными столбиками, а также следить за тем, чтобы переменные не сливались.

Например, если a = 25, b = 'x', a c = 10.5, то после выполнения команды writeln(a:5,' ',b,c:10:5) на экране или в файле будет записано следующее (подчерки в данном случае служат лишь для визуализации пробелов):

_ _ _25_x_ _10.50000

Особенно важен формат при выводе вещественных переменных. К примеру, если не указать формат, то число 10.5 будет выведено как 1.0500000000E+0001. Такой формат называется записью с плавающей точкой.

Если же задать только общую длину вещественного числа, не указывая длину дробной части, то оно будет занимать на экране заданное количество символов (в случае надобности, спереди будет добавлено соответствующее количество пробелов), но при этом останется в формате плавающей точки. Минимальной длиной для вывода вещественных чисел является 10 (при формате _x.xE+yyyy). Первая позиция зарезервирована под знак "-".

Необходимо помнить, что в случае недостаточной длины вывода число будет автоматически округлено, например (подчерк служит для визуализации пробела):

Оператор форматного вывода	Результат вывода на экран
write (125.2367:10);	_1.3E+0002
write (125.2367:11);	_1.25E+0002
write (125.2367:12);	_1.252E+0002
write (125.2367:13);	_1.2524E+0002
write (125.2367:14);	_1.25237E+0002
write (125.2367:15);	_1.252367E+0002
write (125.2367:16);	_1.2523670E+0002

Пример простейшей программы на языке Pascal

```
program start;
var s: string;
begin
 write('Пожалуйста, введите Ваше имя: ');
 readln(s);
 writeln('Мы рады Вас приветствовать, ',s,'!');
end.
```

Во время работы этой программы на экране появится примерно следующее:

Пожалуйста, введите Ваше имя: Иван Иваныч

Мы рады Вас приветствовать, Иван Иваныч!

Практическая часть

1. Пусть даны длины сторон треугольника. Вычислите его площадь.

2. Вычислите расстояние между двумя точками на плоскости с данными координатами (x1,y1) и (x2,y2).

3. Введите положительное число **a.** Вычислите:

площадь равностороннего треугольника со стороной **a;**

площадь квадрата со стороной **a;**

площадь круга с диаметром **a.**

4. Вычислите длину окружности, площадь круга, объем шара заданного радиуса.

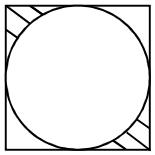
5. Пусть даны числа a, b, c. Найти площадь треугольника, две стороны которого равны a, b, а угол между этими сторонами равен c. Считать, что c – это: радианная мера; градусная мера.

6. Пусть известны длины сторон a,b,c треугольника. Вычислите высоты этого треугольника по формулам: $h_a = \frac{2}{a}\sqrt{p(p-a)(p-b)(p-c)}$ где $p = \frac{a+b+c}{2}$

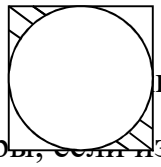
$h_b = \frac{2}{b}\sqrt{p(p-a)(p-b)(p-c)}$ $h_c = \frac{2}{c}\sqrt{p(p-a)(p-b)(p-c)}$

7. Даны координаты вершин некоторого треугольника. Вычислить его периметр.

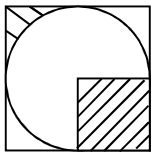
8. В квадрат вписана окружность. Определить площадь заштрихованной части фигуры, если известна длина сторон квадрата.



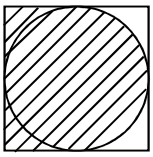
9. В квадрат вписана окружность. Определить площадь заштрихованной части фигуры, если известен радиус окружности.



вадрат вписана окружность. Определить площадь заштрихованной части фигуры, если известна длина стороны квадрата.



11. В квадрат вписана окружность. Определить площадь заштрихованной части фигуры, если известна длина стороны квадрата.



12. Даны два ненулевых числа. Найти их сумму, разность, произведение и частное.

13. Даны два числа. Найти среднее арифметическое их квадратов и среднее арифметическое их модулей.

14. Найти периметр и площадь прямоугольного треугольника, если даны длины его катетов a и b.

15. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

16. Найти длину окружности и площадь круга заданного радиуса R. В качестве значения Pi использовать 3.14.

17. Найти площадь кольца, внутренний радиус которого равен R1, а внешний радиус равен R2 (R1 < R2). В качестве значения Pi использовать 3.14.

18. Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей.

19. Дана площадь круга. Найти длину окружности, ограничивающей этот круг. В качестве значения Pi использовать 3.14.

20. Найти периметр и площадь равнобедренной трапеции с основаниями a и b (a > b) и углом alpha при большем основании (угол дан в радианах).

21. Найти периметр и площадь прямоугольной трапеции с основаниями a и b (a > b) и острым углом alpha (угол дан в радианах).

22. Определить объём и площадь боковой поверхности цилиндра с заданными радиусом основания R и высотой H.

23. Вычислите длину окружности, площадь круга и объём шара одного и того же заданного радиуса.

24. По координатам трёх вершин некоторого треугольника найдите его площадь и периметр.

25. Вычислите периметр и площадь прямоугольного треугольника по длинам двух его катетов

26. Зная два катета прямоугольного треугольника, найти гипотенузу и углы треугольника.

27. Известна гипотенуза ***c*** и прилежащий угол α прямоугольного треугольника. Найти площадь треугольника.

28. Диагональ квадрата равна ***d***. Вычислить площадь и периметр квадрата.

29. Площадь квадрата равна S. Вычислить сторону квадрата ***a***, диагональ ***d*** и площадь S1 описанного вокруг квадрата круга.

30. В равнобедренном треугольнике известно основание ***c*** и угол при нем α. Найти площадь треугольника S и величину боковой стороны ***a***.

31. Известна диагональ ромба. Вычислить его площадь и периметр.

32. Задан периметр квадрата. Вычислить его сторону, диагональ и площадь.

33. В равнобедренном треугольнике известно основание ***c*** и высота ***h***. Найти площадь и периметр треугольника.

34. Известна гипотенуза ***c*** и противолежащий угол β прямоугольного треугольника. Найти периметр треугольника.

35. В прямоугольном треугольнике известен катет ***b*** и площадь ***S***. Вычислить периметр треугольника.

36. в прямоугольном треугольнике известен катет ***b*** и площадь ***S***. Вычислить величину гипотенузы ***c*** и второго катета ***a***.

Контрольные вопросы ЛР 19(УК-1):

1. Какими свойствами обладает алгоритм?
2. Что понимается под результативностью алгоритма?
3. В чем заключается свойство массовости?
4. Что означает свойство определенности алгоритма?
5. Какими элементами задается алгоритм?
6. Что представляет собой «исполнитель алгоритма»?
7. Что является основной характеристикой исполнителя?
8. Что относится к среде исполнителя?
9. Поясните общую методику записи содержания алгоритма.
10. Какие существуют формы представления алгоритмов?
11. Что представляет собой словесный способ записи алгоритма?
12. Что называется блок-схемой алгоритма?
13. Поясните особенности составления блок-схемы алгоритма.
14. Когда используется межстраничный соединитель?
15. Что представляет собой псевдокод?

Лабораторная работа №21. Понятия об основных методах ввода/вывода данных. Разработка программ линейных алгоритмов. Понятия об основных методах отладки (тестирования) программы.

Условные конструкции

Теоретическая часть

К операторам, позволяющим из нескольких возможных вариантов выполнения программы (ветвей) выбрать только один, относятся if и case.

Условный оператор if

Оператор if выбирает между двумя вариантами развития событий:

```
if <условие>
  then <один_оператор>
  [else <один_оператор>];
```

Обратите внимание, что перед словом else (когда оно присутствует, конечно же) символ ";" не ставится - ведь это разорвало бы оператор на две части.

Условный оператор if работает следующим образом:

1. Сначала вычисляется значение <условия> - это может быть любое выражение, возвращающее значение типа boolean.
2. Затем, если в результате получена "истина" (true), то выполняется оператор, стоящий после ключевого слова then, а если "ложь" (false) - без дополнительных проверок выполняется оператор, стоящий после ключевого слова else. Если же else-ветвь отсутствует, то не выполняется ничего.

Что же произойдет, если написать несколько вложенных операторов if?

В случае, когда каждый оператор if имеет собственную else-ветвь, все будет в порядке. А вот если некоторые из них этой ветви не имеют, может возникнуть ошибка. Компилятор языка Pascal всегда считает, что else относится к самому ближнему оператору if. Таким образом, если написать

```
if i>0 then if s>2
         then s:= 1
       else s:= -1;
```

подразумевая, что else-ветвь относится к внешнему оператору if, то компилятор все равно воспримет эту запись как

```
if i>0 then if s>2
         then s:= 1
         else s:= -1
     else;
```

Ясно, что таким образом правильного результата получить не удастся.

Для того чтобы избежать подобных ошибок, стоит всегда (или по крайней мере при наличии нескольких вложенных условных операторов) указывать оба ключевых слова, даже если одна из

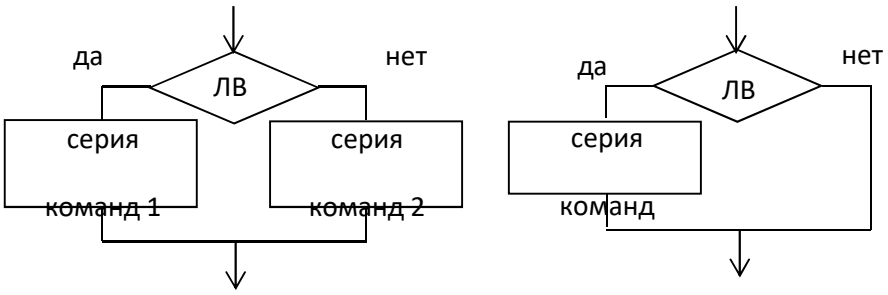
ветвей будет пустовать. Так вы застрахуетесь от одной из частых "ошибок по невнимательности", которые очень сложно найти в процессе отладки программы.

Итак, исходный вариант нужно переписать следующим образом:

```
if i>0 then if s>2
        then s:=1
        else
     else s:=-1;
либо так:
if i>0 then begin if s>2
         then s:=1
      end
     else s:=-1;
```

Вообще же, если есть возможность переписать несколько вложенных условных операторов как один оператор выбора, это стоит сделать.

Изображение условного оператора в виде блок-схемы выглядит следующим образом: (слева – полное ветвление *если-то-иначе*, справа – неполный вариант ветвления *если-то*)



Практическая часть

1. Даны три целых числа. Возвести в квадрат отрицательные числа и в третью степень — положительные (число 0 не изменять).
2. Из трех данных чисел выбрать наименьшее.
3. Из трех данных чисел выбрать наибольшее.
4. Из трех данных чисел выбрать наименьшее и наибольшее.
5. Перераспределить значения переменных X и Y так, чтобы в X оказалось меньшее из этих значений, а в Y — большее.
6. Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной сумму этих значений, а если равны, то присвоить переменным нулевые значения.

7. Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной максимальное из этих значений, а если равны, то присвоить переменным нулевые значения.

8. Даны три переменные: X, Y, Z. Если их значения упорядочены по убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное.

9. Даны три переменные: X, Y, Z. Если их значения упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное.

10. Даны целочисленные координаты точки на плоскости. Если точка не лежит на координатных осях, то вывести 0. Если точка совпадает с началом координат, то вывести 1. Если точка не совпадает с началом координат, но лежит на оси OX или OY, то вывести соответственно 2 или 3.

11. Даны вещественные координаты точки, не лежащей на координатных осях OX и OY. Вывести номер координатной четверти, в которой находится данная точка.

12. На числовой оси расположены три точки: A, B, C. Определить, какая из двух последних точек (B или C) расположена ближе к A, и вывести эту точку и ее расстояние от точки A.

13. Даны четыре целых числа, одно из которых отлично от трех других, равных между собой. Вывести порядковый номер этого числа.

14. Дан номер некоторого года (положительное целое число). Вывести соответствующий ему номер столетия, учитывая, что, к примеру, началом 20 столетия был 1901 год.

15. Дан номер некоторого года (положительное целое число). Вывести число дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

16. Для данного x вычислить значение следующей функции f, вещественные значения: –1 если x <= , 0, f(x x· если 0 < x)= x, < 2, 4, если x >= 2.

17. Для данного x вычислить значение следующей функции f, принимающей значения целого типа: 0, если x < 0, f(x 1, если x принадлежит [0,1),)= [2,3), ... , –1 если x принадлежит [1,2), , [3,4), ...

18. Дано целое число, лежащее в диапазоне от –999 до 999. Вывести строку — словесное описание данного числа вида "отрицательное двузначное число", "нулевое число", "положительное однозначное число" и т.д.

19. Дано целое число, лежащее в диапазоне от 1 до 9999. Вывести строку — словесное описание данного числа вида "четное двузначное число", "нечетное четырехзначное число" и т.д.

20. Два прямоугольника заданы длинами сторон. Определите, можно ли первый прямоугольник целиком разместить во втором.

21. Решите квадратное уравнение $ax^2 + bx + c = 0$

22. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: данная тройка натуральных чисел a, b, c не является тройкой Пифагора, т.е. $c^2 = a^2 + b^2$

23. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: данные числа x, y являются координатами точки, лежащей в первой координатной четверти.

24. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: данная тройка натуральных чисел a, b, c является тройкой Пифагора, т.е. $c^2 = a^2 + b^2$.

25. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: число c является средним арифметическим чисел a и b.

26. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: число c является средним геометрическим чисел a и b.

27. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: сумма двух натуральных чисел кратна 2.

28. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: произведение натуральных чисел a и b кратно числу c.

29. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: данное натуральное число a кратно числу b, но не кратно числу c.

30. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: числа a и b выражают длины катетов одного прямоугольного треугольника, c и d — другого. Эти треугольники являются подобными.

31. Составить программу, печатающую значение true, если указанное высказывание является истинным, и false в противном случае: данные числа c и d являются соответственно квадратом и кубом числа a.

32. Определите, какая из точек M1(x1,y1) или M2(x2,y2) расположена ближе к началу координат. Укажите координаты этой точки.

33. Определите, какая из двух фигур (круг или квадрат) имеет большую площадь. Известно, что сторона квадрата равна a, радиус круга r. Выведите название большей фигуры и её площадь.

34. Определите, попадает ли точка M(x,y) в круг радиусом R с центром в точке (x_o,y_o).

35. Заданы площади круга и квадрата. Определите, поместится ли квадрат в круге.

36. Заданы площади круга и квадрата. Определите, поместится ли круг в квадрате.

Контрольные вопросы ЛР 21(УК-1):

1. Понятия об основных методах ввода/вывода данных.
2. Разработка программ линейных алгоритмов.
3. Понятия об основных методах отладки (тестирования) программы.
4. Что такое ввод данных?
5. Что такое вывод данных?
6. Что такое область значений данных?
7. Что такое тип данных?
8. Что такое отладка программ?
9. Что такое контрольный вариант расчета ?

Лабораторная работа №23 Разработка программ алгоритмов разветвляющейся и циклической структуры.

Оператор case позволяет сделать выбор между несколькими вариантами:

```
case <переключатель> of
    <список_констант> : <один_оператор>;
    [<список_констант> : <один_оператор>;]
    [<список_констант> : <один_оператор>;]
    [else <один_оператор>;]
end;
```

Замечание: Обратите внимание, что после else двоеточие не ставится.

Существуют дополнительные правила, относящиеся к структуре этого оператора:

1. Переключатель должен относиться только к порядковому типу данных, но не к типу longint.
2. Переключатель может быть переменной или выражением.
3. Список констант может задаваться как явным перечислением, так и интервалом или их объединением.
4. Повторение констант не допускается.
5. Тип переключателя и типы всех констант должны быть совместимыми.

Пример оператора выбора:

```
case symbol(* :char *) of
  'a'..'z', 'A'..'Z' : writeln('Это латинская буква');
  'a'..'я', 'A'..'Я' : writeln('Это русская буква');
  '0'..'9' :      writeln('Это цифра');
  ' ',#10,#13,#26   :   writeln('Это пробельный символ');
  else                writeln('Это служебный символ');
end;
```

Выполнение оператора case происходит следующим образом:

1. вычисляется значение переключателя;
2. полученный результат проверяется на принадлежность к тому или иному списку констант;
3. если такой список найден, то дальнейшие проверки уже не производятся, а выполняется оператор, соответствующий выбранной ветви, после чего управление передается оператору, следующему за ключевым словом end, которое закрывает всю конструкцию case.
4. если подходящего списка констант нет, то выполняется оператор, стоящий за ключевым словом else. Если else-ветви нет, то не выполняется ничего.

Практическая часть

1. По заданному номеру месяца m вывести на печать его название.

2. Дан номер месяца (1 — январь, 2 — февраль, ...). Вывести название соответствующего времени года ("зима", "весна" и т.д.).

3. Дан номер месяца (1 — январь, 2 — февраль, ...). Вывести число дней в этом месяце для невисокосного года.

4. Дано целое число в диапазоне 0 – 9. Вывести строку — название соответствующей цифры на русском языке (0 — "ноль", 1 — "один", 2 — "два", ...).

5. Дано целое число в диапазоне 1 – 5. Вывести строку — словесное описание соответствующей оценки (1 — "плохо", 2 — "неудовлетворительно", 3 — "удовлетворительно", 4 — "хорошо", 5 — "отлично").

6. Вычислить значение функции *y* по одной из формул: x^{10}/g, $g^2 \cdot x + 6 \cdot x$; $g^3 \cdot (\cos x + g - x)^2$, причем x и g вводятся с клавиатуры

7. Арифметические действия над числами пронумерованы следующим образом: 1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление. Дан номер действия и два числа A и B (B не равно нулю). Выполнить над числами указанное действие и вывести результат.

8. Единицы длины пронумерованы следующим образом: 1 — дециметр, 2 — километр, 3 — метр, 4 — миллиметр, 5 — сантиметр. Дан номер единицы длины и длина отрезка L в этих единицах (вещественное число). Вывести длину данного отрезка в метрах.

9. Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы и масса тела M в этих единицах (вещественное число). Вывести массу данного тела в килограммах.

10. Вычислить значение функции *y* по одной из формул: $\sqrt[4]{x}$, x^6; $\cos x + x^7$, причем x и g вводятся с клавиатуры.

11. Автобус остановился на развилке дорог. Он может поехать в четырех направлениях (1 — север, 2 — запад, 3 — юг, 4 — восток). Дано число N — заданное направление. Вывести, куда поедет автобус.

12. Элементы окружности пронумерованы следующим образом: 1 — радиус (R), 2 — диаметр (D), 3 — длина (L), 4 — площадь круга (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данной окружности (в том же порядке). В качестве значения Pi использовать 3.14.

13. Вычислить значение функции *y* по одной из формул: x^5-10, x^{9x}; $\sqrt[8]{x^2} - \sqrt[3]{x}$, причем x вводится с клавиатуры.

14. Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 — катет (a), 2 — гипотенуза (c), 3 — высота, опущенная на гипотенузу (h), 4 — площадь (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

15. Элементы равностороннего треугольника пронумерованы следующим образом: 1 — сторона (a), 2 — радиус вписанной окружности (R1), 3 — радиус описанной окружности (R2), 4 — площадь (S). Дан номер одного из этих

элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

16. Вычислить значение функции *y* по одной из формул: x^2, $2 \cdot x + 3 \cdot x^2$; $(\sin x + \cos x)^2$, причем x вводится с клавиатуры.

17. При введении времени года вывести общее количество дней данного времени года.

18. Дано целое число в диапазоне 20 – 30, определяющее возраст (в годах). Вывести строку — словесное описание указанного возраста, обеспечив правильное согласование числа со словом "год", например: 20 — "двадцать лет", 21 — "двадцать один год", 22 — "двадцать два года".

19. Определить количество дней в месяце високосного года по номеру месяца.

20. По заданному названию времени года вывести на печать название месяцев для данного времени года.

21. Вычислить значение функции *y* по одной из формул: g^3, $2 \cdot g \cdot e^x - h$; $g \cdot (\cos x + h)^2$, причем x, h и g вводятся с клавиатуры.

22. Написать программу, которая анализирует данные о возрасте и относит человека к одной из четырех групп: дошкольник, ученик, работник, пенсионер. Возраст вводится с клавиатуры.

23. Написать программу, которая по номеру дня недели (целому числу от 1 до 7) выдает в качестве результата количество уроков в вашем классе в этот день.

24. Написать программу, которая по вводимому числу от 1 до 11 (номеру класса) выдает соответствующее значения «Привет, k-классник». Например, если k=4, то «Привет, четвероклассник».

25. Вычислить значение функции *y* по одной из формул: g; $g \cdot x$; $g \cdot \sqrt{|x|}$, причем x и g вводятся с клавиатуры.

26. Вычислить значение функции *y* по одной из формул: g^2, $g \cdot e^x + h$; $g \cdot (\sin x + h)^2$, причем x, h и g вводятся с клавиатуры.

27. Дан номер месяца (1 — январь, 2 — февраль, ...). Вывести число дней в этом месяце для високосного года.

28. С клавиатуры вводится символ. Определить что это за символ (заглавная буква, строчная буква или цифра).

29. Дан номер курса k (1 – первый курс, 2 – второй курс, …). Вывести на экран, сколько лет вам осталось учиться в нашем колледже. Например, если k=1, то «Осталось учиться 3 года».

30. Вычислить значение функции *y* по одной из формул: x^3+1, x^{2x}; $\sqrt{x} + \sqrt[3]{x}$, причем x вводится с клавиатуры.

31. При введении времени года вывести его месяцы и количество дней в каждом.

32. Даны номера названий времен года (1 — зима, 2 — весна, 3 — лето, 4 — осень). Вывести названия месяцев, соответствующих каждому времени года.

33. Вычислить значение функции *y* по одной из формул: g^4, $g \cdot e^{2x} + 2 \cdot h$; $4 \cdot g \cdot (\sin x + h)^2$, причем x, h и g вводятся с клавиатуры.

34. Написать программу, которая по вводимому числу от 1 до 4 (номеру курса) выдает соответствующее значения «Привет, ты учишься k-м курсе, до получения диплома осталось 4-k года (лет)». Например, если k=1, то «Привет, ты учишься 1-м курсе, до получения диплома осталось 3 года (лет)».

35. Арифметические действия над числами пронумерованы следующим образом: 1 — возведение в степень 2, 2 — вычисление квадратного корня, 3 — умножение на 5, 4 — деление на 10. Дан номер действия и число A. Выполнить указанное действие и вывести результат.

36. Написать программу, которая по номеру дня недели (целому числу от 1 до 7) выдает в качестве результата перечень уроков в вашем классе в этот день.

Контрольные вопросы ЛР 23(УК-1):

1. Какими свойствами обладает алгоритм?
2. Что понимается под результативностью алгоритма?
3. В чем заключается свойство массовости?
4. Что означает свойство определенности алгоритма?
5. Какими элементами задается алгоритм?
6. Что представляет собой «исполнитель алгоритма»?
7. Что является основной характеристикой исполнителя?
8. Что относится к среде исполнителя?
9. Поясните общую методику записи содержания алгоритма.
10. Какие существуют формы представления алгоритмов?
11. Что представляет собой словесный способ записи алгоритма?
12. Что называется блок-схемой алгоритма?
13. Поясните особенности составления блок-схемы алгоритма.
14. Когда используется межстраничный соединитель?
15. Что представляет собой псевдокод?

Лабораторная работа №20. 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			or			xor		
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 mod 5)=3, то у нас (-2 mod 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 mod k) + min(a,b) + 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 вычислимая 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	+, -, not, @, ^, #	Первый(высший)
Операции, эквивалентные умножению	*, /, div, mod, and, shl, shr	Второй
Операции, эквивалентные сложению	+,-, or, xor	Третий
Операции сравнения	=, <>, >, <, <=, >=, in	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 + 3 z.

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 $12 \le x \le 42$.

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 А руб., портфеля стоимостью В 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 ЛР20(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.

Лабораторная работа №22. Ввод/вывод данных. Программная реализация линейных алгоритмов в интегрированной среде разработки. Отладка (тестирование) программы.

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<=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>=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 while:	Для repeat-until:
1. Проверяется, истинно ли <условие_1>.	1. Выполняются <операторы>.
2. Если это так, то выполняется <оператор>.	2. Проверяется, ложно ли <условие_2>
3. Пункты 1 и 2 выполняются до тех пор, пока <условие_1> не станет ложным.	3. Пункты 1 и 2 выполняются до тех пор, пока <условие_2> 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 дождавшись 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, 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+...+17X/18.
2. Вывести на печать целые положительные кратные 9, числа пока истинно условие m<=100.
3. Дано натуральное число N. Вычислить сумму первых N- слагаемых
4. Вычислить 1/1+3/2+5/3+7/4+…
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<=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<=100.
16. Вычислить выражение $y = \sum_{m=2}^{20} (2m - 40)$.
17. Вывести на печать целые положительные кратные 8 числа пока истинно условие k<=100.
18. Вычислить выражение $y = \sum_{x=1}^{10} (tgx + 5)$.
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<=100.

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

26. Вычислить Y=4a+b, если A=1,2,...,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<=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*…*n. N – вводится.

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

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

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

Контрольные вопросы ЛР22 (ОПК-3):

1. Что значит Ввод/вывод данных?
2. Что значит программная реализация линейных алгоритмов в интегрированной среде разработки?
3. Что значит отладка (тестирование) программы?

Лабораторная работа №24. Программная реализация алгоритмов разветвляющейся, циклической и смешанной структуры в интегрированной среде разработки.

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.

П р и м е р:
```
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>;

П р и м е р:
```
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.

П р и м е р.

Пусть двумерный массив описан следующим образом:

VAR A : ARRAY[1..2,1..4] 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 + 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. Расчитать N значений элементов массива B по формуле :

$$B_k = \begin{cases} K + \cos(K-1); & \text{ïðè K íå÷åòíîì} \\ \sin K + 3; & \text{ïðè Ê ÷åòíîì} \end{cases}$$

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.

7. Ввести с клавиатуры информацию о температуре воздуха за 2 недели. Записать в массив. Определить, сколько раз за это время она была ниже ноля.

8. Рост студентов представить в виде массива. Рост девушек закодировать со знаком "-",а рост юношей со знаком "+". Определить средний рост мальчиков.

9.Рассчитать N значений элементов массива B по формуле:

$$B_k = \begin{cases} \sin K + 3 & \text{при } 8 < K <= N \\ K + \cos(K-1) & \text{при } 3 < K <= 8 \\ K & \text{при } K <= 3 \end{cases}$$

10. Составить массив B из отрицательных значений функции Z=cos(x)/sin(x-2) для x, изменяющегося на отрезке [5;-10] с шагом 0,67.

11. Вычислить последовательность N чисел Фибоначчи и записать ее в массив $F_0 = F_1 = 1$; $F_{i+1} = F_i + F_{i-1}$

12. Вычислить N элементов массива X: $X_k = X_{k-1} + (1/2)X_{k-2}$ $X_1 = 0$, $X_2 = 0{,}25$

13. Написать программу нахождения N элементов массивов X и Y, пользуясь формулами: $X_k = 3X_{k-1} + k$, $Y_k = X_{k-1} + Y_{k-1}$, $X_0 = 1$, $Y_0 = 2$.

14. Найти N элементов массива $X_1 = X_2 = X_3 = 1$; $X_k = X_{k-1} + X_{k-3} - 1/K$.

15. Найти сумму N элементов массива $X_1 = X_2 = X_3 = 2$; $X_k = X_{k-2} - X_{k-3} + 1/K$

16. Вычислить значения элементов массива Z по формуле :
Z= cos X + ln X, где X меняется на отрезке [1;15] с шагом 0,92 и найти их сумму

17. Вычислить сумму значений функции

$$Y_k = A_k^2 + A_k - \text{SIN } A_k ,$$

где элементы массива A вводятся с клавиатуры .

18. Рассчитать сумму N значений элементов массива B, формуле :

$$B_k = \begin{cases} \sin K + 3 \ ; \text{ïðè } \hat{E} \text{ íå÷åòíîì} \\ K + \cos(K-1) \ ; \text{ïðè } \hat{E} \text{ ÷åòíîì} \end{cases}$$

19. Найти сумму отрицательных значений элементов массива W, вводимого с клавиатуры.

20. Найти сумму значений элементов массива W с четными индексами вводимого с клавиатуры.

21. Ввести с клавиатуры информацию о температуре воздуха за 2 недели. Определить, сколько раз за это время она была ниже ноля.

22. Найти сумму значений элементов массива A с нечетными индексами вводимого с клавиатуры.

23. Рассчитать сумму N значений элементов массива B, по формуле:

$$B_k = \begin{cases} \sin K + 3 & \text{при } K > 3 \\ K + \cos(K-1) & \text{при } K = 2 \\ K & \text{при } K < 3 \end{cases}$$

24. Составить массив B 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 n do
        for j:= 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.

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 стролбца матрицы A(M,N).

2. Вычислить значение функции $Z = \sum_{i=1}^{10} \sum_{k=1}^{N} \frac{\sin^k x(i)}{k}$, где X(I) заданы массивом (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.Вчислить 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.Оределить 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. Дны элементы массива A, состоящего из n элементов. Вычислить $S = A_1{}^1 + A_2{}^2 + \ldots + A_n{}^n$ 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 $Z = \sum_{i=1}^{M} \sum_{j=1}^{i} \frac{1}{2j + i}.$

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

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

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

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 $\frac{1}{N!}\sum(-1)^k\frac{x^k}{(k!+1)!}$.

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 к - ой строки матрицы A на элемент B(K).

25.Заданы матрицы A(м,м) и B(м,м). 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 ЛР24(ОПК-3):

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