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

**С.А. ШВИДЧЕНКО**

# Методические указания для проведения практических занятий (II семестр) по дисциплине

# Б1.О.05 «ИНФОРМАТИКА»

Кафедра **«Информатика и вычислительная техника»**

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

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

Разработала:
***Доцент кафедры ИВТ Швидченко С.А.***

Ростов-на-Дону
2021

Методические указания
для проведения практических занятий
по дисциплине
«Информатика»

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

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

**Модуль 1.**

***Практическое занятие №1. 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 ПЗ1(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, динамических (электронных) таблиц.
6. Провести компьютерные эксперименты с использованием готовых моделей объектов и процессов.

***Практическое занятие №2. 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 ПЗ2(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?

***Практическое занятие №3. 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 ПЗ3(УК-1):

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.

**Модуль 2.**

***Практическое занятие №4. 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="3">Базовые типы данных</td><td colspan="5">Порядковые (дискретные) типы данных</td></tr>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">Арифметические типы данных</td><td>Адресные типы данных</td><td>Структурированные типы данных</td></tr>
<tr><td>Целые</td><td>Вещественные</td><td></td><td></td></tr>
<tr><td>Логический<br><code>boolean</code></td><td>Символьный (литерный)<br><code>char</code></td><td><code>shortint<br>byte<br>integer<br>word<br>longint</code></td><td><code>real<br>single<br>double<br>extended<br>comp</code></td><td>Нетипизированный указатель<br><code>pointer</code></td><td></td></tr>
<tr><td rowspan="7">Конструируемые типы</td><td rowspan="7"></td><td>Перечисляемый<br><code>week = (su, mo, tu,<br>we, th, fr, sa);</code></td><td rowspan="7"></td><td rowspan="7"></td><td rowspan="7">Типизированный указатель<br><code>^<тип></code></td><td>Массив array</td></tr>
<tr><td></td><td></td><td></td><td>Строка string</td></tr>
<tr><td></td><td></td><td></td><td>Запись record</td></tr>
<tr><td>Интервал (диапазон)<br><code>budni = mo..fr;</code></td><td></td><td></td><td>Файл<br><code>text<br>file</code></td></tr>
<tr><td></td><td></td><td></td><td>Процедурный</td></tr>
<tr><td></td><td></td><td></td><td>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;**

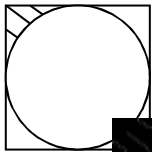
площадь круга с диаметром **a.**

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 $p = \frac{a+b+c}{2}$

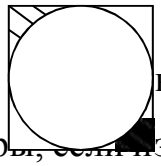
$h_b = \frac{2}{b}\sqrt{p(p-a)(p-b)(p-c)}$ $h_c = \frac{2}{c}\sqrt{p(p-a)(p-b)(p-c)}$

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

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

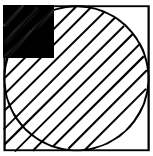


9. В квадрат вписана окружность. Определить площадь заштрихованной части фигуры, если известен радиус окружности.



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 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 . 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 ПЗ4(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?

***Практическое занятие №5. 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:

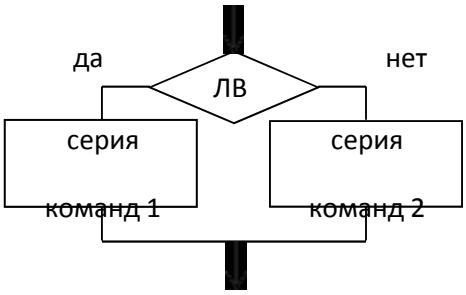
```
if i>0 then if s>2
        then s:=1
        else
     else s:=-1;
```

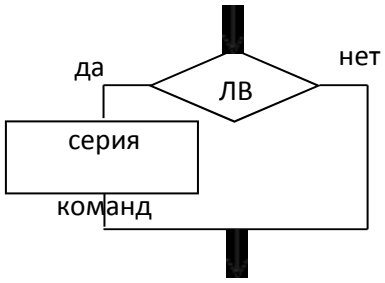
либо так:

```
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 ПЗ5(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 ?

***Практическое занятие №6Разработка 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, $x^{9x}$; $\sqrt[3]{x^2} - \sqrt[3]{x}$, 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: $g$; $g \cdot x$; $g \cdot$ [redacted], 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 ПЗ6(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?