Вопросы к экзамену
по дисциплине «Информатика и программирование»

Информатика. Основные понятия.

1. Информатика: понятие. Цели и задачи дисциплины.
2. Информация и данные. Единицы измерения информации. Устройства хранения информации.
3. Архитектура ЭВМ. Принципы Дж.фон Неймана.
4. Классификация программного обеспечения.
   Операционные системы: классификация, основные элементы.
   Классификация окон MS© Windows.
5. Программное обеспечение: браузеры. Основные операции с объектами.
6. Технологии обмена данными MS© Windows.
7. Текстовый редактор MS© Word.
   Основные элементы.
   Форматирование элементов.
   Стили. Шаблоны документов.
   Слияние документов.
   Установка защиты на документ.
8. Табличный редактор MS© Excel.
   Основные элементы.
   Форматирование элементов.
   Форматы ячеек. Адресация ячеек.
   Формулы.
   Установка защиты на документ.
9. Гипертекстовые документы.
   Основные элементы.
   Структура документа.
   Тэги: парные и одиночные.
   Тэги: выравнивание текста, вставка рисунков, гиперссылки, списки, таблицы, метки.

Программирование. Основные понятия.

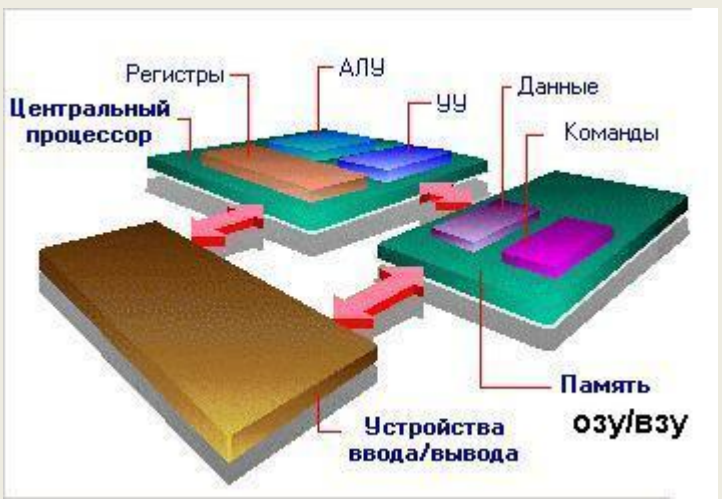
10. Алгоритмы обработки данных: виды алгоритмов, типы записей алгоритмов, обозначения ГОСТ 703.90.
11. Этапы решения задач на ЭВМ.
12. Классификация языков программирования.
13. Программирование на языке OPascal и C++. Структура программ.
14. Классификация типов данных в языке OPascal и C++. Описание типов данных в программе.
15. Выражения, арифметические и логические операции, оператор присваивания.
16. Операторы ввода–вывода. Проектирование ввода–вывода информации.
17. Условный оператор и оператор варианта. Пример с использованием блок-схемы.
18. Составной оператор. Пример с использованием блок-схемы.
19. Организация циклов. Блок-схемы. Вложенные циклы; правила работы с вложенными циклами.

1 Информатика – наука, занимающаяся автоматизированной обработкой инфы с помощью ЭВМ. Jна не существует сама по себе а призвана решать задачи в различных областях. Структура инф.:
Технические средства, программные средства, алгоритмические средства. Задачи инф.:
Исследование информац. процессов любой природы, разработка информационной техники, решение научных и инженерных проблем; создание, внедрение и обеспечение нормальной работы компьютерной техники.
Основные понятия: информация (отображение знаний и фактов), информационная модель (материальный или идеальный образ совокупности объектов) , алгоритмы (послед. действий выполняемых над исходными данными) программы , ЭВМ (электронное устройство для автоматизации процессов поиска, хранения, обработки, и передачи инфы, которые осущ. по заранее разработанным алгоритмам.

2. Информация – сведения об объектах и явлениях окружающей среды, их параметрах, свойствах, состоянии, позволяющая понизить степень неопределённости знаний об этих объектах. Данные – могут рассматриваться как признаки или записанные наблюдения, которые пока не используются а только хранится. Если есть возможность использовать их для уменьшения неопределённости о чём-либо, данные становятся информацией. Компьютер работает с данными, а пользователь-с информацией. . Информация бывает: текстовая, звуковая, числовая, графическая, видео. Измерение данных основано на способе кодирования данных - при обработке их на компьютере в двоичной системе счисления. Размер измеряется в:
бит (ноль или единица), байт=8бит, килобайт=1024 байта, мегабайт=1024 кг, гигабайт = 1024мегайта. Она хранится в памяти компа. ОЗУ – информация, используемая в данный момент, ПЗУ – неизменяемая, системная информация, ВЗУ – внешние запоминающие устройства

3 Архитектура ЭВМ. Принципы фон Неймана.
Огромный авторитет фон Неймана привел к тому, что всем базовым принципам построения ЭВМ стали приписывать его имя, а архитектура называться «фон нейманская».
Фон Нейман с соавторами выдвинули основные принципы логического устройства ЭВМ и предложили ее структуру, которая полностью воспроизводилась в течение первых двух поколений ЭВМ: 1.**Использование двоичной системы представления данных**
Авторы убедительно продемонстрировали преимущества двоичной системы для технической реализации,удобство и простоту выполнения в ней арифметических и логических операций. ЭВМ стали обрабатывать и нечисловые виды информации - текстовую, графическую, звуковую и другие, но двоичное кодирование данных по-прежнему составляет информационную основу любого современного компьютера.
2. П**ринцип хранимой программы** Первоначально программа задавалась путем установки перемычек на специальной коммунационной панели. Это было весьма трудоемким занятием. Нейман первым догадался, что программа может также храниться в виде нулей и единиц, причем в той же самой памяти, что и обрабатываемые ею числа. Отсутствие принципиальной разницы между программой и данными дало возможность ЭВМ самой формировать для себя программу в соответствии с результатами вычислений.
Фон Нейман не только выдвинул основополагающие принципы логического устройства ЭВМ , но и предложил ее структуру, которая воспроизводилась в течение первых двух поколений ЭВМ.

Устройство управления (УУ) и арифметико-логическое устройство (АЛУ) в современных компьютерах объединены в один блок - процессор, являющийся преобразователем информации, поступающей из памяти и внешних устройств.
Память (ЗУ) хранит информацию (данные) и программы. Запоминающее устройство у современных компьютеров "многоярусно" и включает оперативное запоминающее устройство (ОЗУ) и внешние запоминающие устройства(ВЗУ).
ОЗУ- это устройство, хранящее ту информацию, с которой компьютер работает непосредственно в данное время (исполняемая программа, часть необходимых для нее данных, некоторые управляющие программы).
ВЗУ-устройства гораздо большей емкости, чем ОЗУ, но существенно более медленны.
3.Принцип последовательного выполнения операций Структурно основная память состоит из пронумерованных ячеек. Процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти, так, чтобы к запомненным в них значениям можно было бы впоследствии обращаться или менять их в процессе выполнения программы с использованием присвоенных имен.
4.Принцип произвольного доступа к ячейкам оперативной памяти Программы и данные хранятся в одной и той же памяти.



Поэтому ЭВМ не различает, что хранится в данной ячейке памяти - число, текст или команда. Над командами можно выполнять такие же действия, как и над данными.

4 Программа –
упорядоченная последовательность инструкций и команд для решения задач.
Программное обеспечение – совокупность программ необходимых для корректной работы компа, которые могут выполняться на компьютерах данной модели, включающая комплекты сопровождающей их технической информации. Различают:
1. системное П.О. – служит для эффективной работы аппаратуры компа (ОС, оболочки ОС, архиваторы, антивирусы)
2. Инструментальное П.О. – применяются для создания разных программ
3. Прикладное П.О. – пакет взаимосвязанных программ, обеспечивающий решение задач в какой-то конкретной области (графические, текстовые редакторы и т.д.)
ОС – совокупность программных средств, обеспечивающая управление аппаратной частью компа и прикладными прогами, а так же их взаимодействие между собой и пользователем. Разделяются по кол-ву вып. задач; кол-ву пользователей. Есть так-же сетевые ОС для обеспечения работы локальных сетей.
Окна делятся на: окно приложения, окно документа, служебное окно(модальное, немодальное)

5. Браузер.
Предназначение программы Explorer.

Программа Explorer предоставляет гораздо больше возможностей, чем простое управление файлами. С помощью программы Explorer можно управлять не только файлами, но и другими объектами. В Windows NT 4.0 файлам уже не отводится какой-либо особой роли — теперь с ними обращаются так же, как и с другими объектами (Recycle Bin — Корзина, Control Panel — Панель управления и т.п.).

Программа Explorer является своего рода пультом управления компьютером и всеми подключёнными к нему устройствами.

Принципы управления файлами, применяемые в окне программы Explorer, не отличаются от тех, которые используются в окне любой папки.
Виртуальный Рабочий стол (Desktop).

Виртуальный Рабочий стол позволяет получить доступ к любым данным, хранящимся на компьютере.

Поэтому самая верхняя пиктограмма в иерархическом дереве папок и устройств так и называется: Рабочий стол (Desktop). Достаточно выполнить на ней двойной щелчок, и отобразится в окне программы Explorer содержимое рабочего стола.

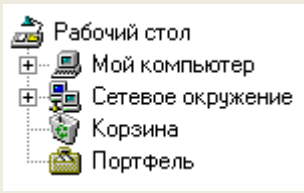


Рис. 1. Desktop

Обычно на рабочем столе расположены пиктограммы, которые представлены на рис. 1.
Переименование, копирование и перемещение файлов и папок.

Перед копированием или перемещением файла (папки) его выделяют в окне щелчком кнопки мыши по значку. Выделенный файл (папка) может быть переименован с помощью контекстного меню или после нажатия функциональной клавиши F2. Дальнейшие действия выполняют с использованием:

- Команд меню;
- Кнопок на панели инструментов;
- Перемещением значка мышью.

Копирование и перемещение с использованием команд главного меню и панели инструментов выполняется так же, как и в Windows 3.x.

Копирование с помощью мыши осуществляется нажатием кнопки мыши и транспортировкой значка файла (папки) в другую папку, программу, на диск или дискету, на рабочий стол. Можно копировать сразу несколько файлов или папок, если их предварительно выделить.

6 Технология обмена данными
Windows – многозадачная многопользовательская ОС. Обмен данных от одного объекта (доки, попки и т.д.) к другому. Осуществляется либо ч/з буфер обмена(область данных, предназначенная для временного хранения объектов) , командами «копировать, вырезать, вставить), либо путём перетаскивания объекта мышью.

11. Этапы решения задачи на ЭВМ
Программирование (programming) - теоретическая и практическая деятельность, связанная с созданием программ. Решение задач на компьютере включает в себя следующие основные этапы, часть из которых осуществляется без участия компьютера.
Постановка задачи:

- сбор информации о задаче;
- формулировка условия задачи;
- определение конечных целей решения задачи;

- определение формы выдачи результатов;
- описание данных (их типов, диапазонов величин, структуры и т. п.).

Анализ и исследование задачи, модели:

- анализ существующих аналогов;
- анализ технических и программных средств;
- разработка математической модели;
- разработка структур данных.

Разработка алгоритма:

- выбор метода проектирования алгоритма;
- выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
- выбор тестов и метода тестирования;
- проектирование алгоритма.

Программирование:

- выбор языка программирования;
- уточнение способов организации данных;
- запись алгоритма на выбранном языке программирования.

Тестирование и отладка:

- синтаксическая отладка;
- отладка семантики и логической структуры;
- тестовые расчеты и анализ результатов тестирования;
- совершенствование программы.

Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2-5.

7. Сопровождение программы:

- доработка программы для решения конкретных задач;
- составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

7. MS WORD

Объект «символ». Формат-Шрифт(размер, начертание(векторная(truetime), растровая), расстояние между символами, анимация).

Слово. Набор символов без пробела.

Предложение. слова и точка. (CTRL+ЛКМ).Формат-Регистр.

Абзац. Формат-абзац (красная строка,интервалы)

Раздел. Файл-Параметры странницы.

Документ может состоять из нескольких разделов.

Гипердокумент. Несколько доков, связанных ссылками. Может сост. из 1 дока, но с ссылками на самого себя.

Ссылка – путь к файлу.(скопировать объект в буфер – правка – спец. вставка – как гиперссылка)

При **прямом форматировании** следует сначала выделить форматированный фрагмент текста, а затем в диалоговых окнах команд меню Формат установить его параметры.

Стилевое форматирование заключается в назначении отдельным абзацам или символам определённых стилей.

Стилем называется группа параметров, имеющая уникальное имя. Стиль оформления может содержать множество различных параметров абзацев или символов и хранится совместно с документом или его шаблоном. Пользователь задаёт параметры объекта, например, шрифт и его размер, выравнивание, поля, расстояния между абзацами и обрамление, а затем назначает группу параметров выделенному фрагменту документа или всему документу.

1.3.2. Назначение стилей.

Прямое форматирование целесообразно, если необходимо быстро выделить слово или выровнять абзац. Это удобный способ оформления небольших документов разового использования. При оформлении больших документов с разнообразными абзацами лучше выполнить стилевое оформление.

Стилевое оформление делает процесс форматирования документов более простым, поскольку пользователь оперирует уже готовыми стилями, позволяет сэкономить время и достигнуть унификации оформления всех документов, используемых в определённой организации.

Использование стилей позволяет автоматизировать создание структуры документа. Например создание оглавления, при этом работа с документом в электронном режиме отображения документа становится более удобной: появляется возможность быстро перемещаться к нужным пунктам (с помощью оглавления), перемещаться по всему документу (при прокрутке документа с помощью вертикальной полосы прокрутки рядом с поплавком будет появляться названия разделов документа).

1.3.3. Диалоговое окно стили.

В списке Стили диалогового окна команды Стили меню Формат перечислены все стили, хранящиеся в шаблоне активного документа. Рядом с именем стиля абзаца находится маркер абзаца, рядом с именем стиля символов — подчёркнутая литера а. В этом окне можно изменять существующие стили и создавать новые, копировать, удалять, переименовывать, а также назначать стили выделенным фрагментам текста.

Стили знаков (символов) акцентируют на отдельные слова в тексте, фразы в абзацах. Создав и используя стили знаков, можно упростить и впоследствии упростить применение и изменение оформление особого формата шрифта для выделения специальных терминов, особо важных названий или заголовков.

Чтобы создать стиль знака, сначала введите какой-нибудь текст, а затем подберите для него необходимый шрифт. Обозначив правильно оформленный специальный текст, выберите из Формат команду Стиль, в появившемся диалоговом окне щелкнуть по «Создать ».

Шаблон документа – особый документ, в котором установлены размеры полей страниц, определены форматы и заготовлены стандартные фрагменты теста для составления какой-либо деловой бумаги. Достоинство шаблонов заключается в их способности соединять воедино параметры и методы создания, форматирования документов. В шаблонах хранятся задаваемые по умолчанию текст, графические изображения, элементы автотекста, стили, макросы, наборы модифицированных команд меню, конфигурации меню инструментов и созданные панели инструментов.

Существуют шаблоны общие и дополнительные.

По умолчанию новые документы основываются на шаблоне Normal, хранимом в файле Normal.dot. При этом существует возможность загружать дополнительные шаблоны – для совместного использования с Normal.

Макрокоманды, изменения определений меню, клавиш, хранящиеся в специализированных шаблонах документа, имеют больший приоритет над установками Normal и других общих шаблонов. Параметры дополнительного шаблона заменяют параметры общих шаблонов. Если в дополнительном шаблоне какие-то параметры отсутствуют, Word использует параметры из общих шаблонов.

Возможности дополнительных шаблонов можно будет оценить при работе с документами различных типов: факс, служебная записка, резюме, отчеты и другие. При переключении из одного документа в другой будет изменяться внешний вид редактора Word: команды меню, настройки панелей инструментов, наборы стилей – будет происходить переключение их одной области со своими настройками – в другую.

Создание документа на основе существующего шаблона: меню Файл, команда Создать… Далее из списка зарегистрированных шаблонов выбирается наиболее подходящий.

Создание шаблона. В меню Файл, команда Создать… позволяет указать тип документа – шаблон. Далее создаются стили, набирается стационарный текст, вставляются таблицы и остальные объекты. Созданный шаблон сохраняется и затем на его основе создается документ

СЛИЯНИЕ

В текстовом редакторе существует возможность автоматизировать создание документов, писем с помощью слияния (см. справочную систему – раздел «Обзор слияния»).

Слияние – процесс, позволяющий на основе содержания двух документов сформировать целый ряд новых.

Основной документ — документ, содержащий текст и графические изображения, повторяющиеся во всех создаваемых документах. Они должны содержать заголовок, текст стереотипного сообщения, заключительную часть, помещаемые во все письма.

Источник данных — документ, содержащий таблицу с текстом и изображениями, которые будут использоваться в каждом создаваемом документе, ни разу не повторяясь.

В текстовом редакторе процессом слияния управляет функция «Ассистент слияния».

8. EXCEL

Execute Cell – вып. ячейка.

Элементы *интерфейса пользователя*:

1. панель заголовка, как и у любого окна в среде Windows, указывает на программу, которая запущена, и документ MSOffice, с которым происходит работа;
2. панель меню содержит ниспадающие меню, предлагающие различные команды;
3. панели инструментов — набор кнопок для быстрого выполнения нужной команды;
4. панель формул показывает адрес текущей ячейки (поле имени) и её содержимое;
5. окно рабочей книги электронной таблицы;
6. строка состояния информирует о выполняемом действии, или о готовности к новой команде.
7. строка формул

Объект «Ячейка».

Содержимым ***ячейки*** может быть как *текст, числовые данные*, так и *формула*.

Так как колонки озаглавлены буквами, а столбцы — цифрами, то у каждой ячейки есть свой – *только её* – уникальный адрес, состоящий из буквы и цифры. Например, на рис.1 у выбранной ячейки адрес —

К21. Адрес ячейки (во время заполнения, составления формул) можно увидеть в строке формул.
Существуют ***три вида адресации***:

Адресация ячейки	Относительная	Абсолютная	Смешанная
Пример	B12, G34:H40	$B$12, $G$34:$H$40, Количество, а	$B12, B$12

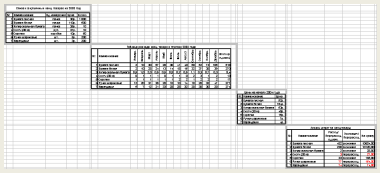
Относительная – меняется при размножении ячейки.
Для присваивания ячейке (группе ячеек) уникального имени необходимо выделить ячейку и выполнить команду Присвоить пункта Имя меню Вставить. В имени нельзя использовать пробелы.
Объект «Лист»
Объект «Рабочая книга».
Электронная таблица напоминает бумажную таблицу. Все таблицы помещены в ***Рабочую книгу***, которая появляется при запуске Excel.
Рабочая книга — файл данного приложения. С ней работают как с обычным файлом.
Помимо этого, MS Excel позволяет оформить таблицу, отформатировать данные, построить диаграммы, работать с базой данных (данные типа Paradox, dBase), распечатать итоговый документ, работать с базой данных (данные типа Paradox, dBase), обрабатывать данные из разных файлов, поддерживает технологию обмена данными между приложениями **OLE 2.0**.

3.1.2. Форматирование ячеек.

Форматировать данные можно как в одной ячейке, так и диапазоне ячеек.

Диапазон — это прямоугольная группа ячеек.

Форматирование может быть уловным – в зависимости от выполнения или невыполнения условия.
Выделение групп ячеек происходит так же, как описано в **лаб. раб.№1 Explorer** для выделения группы файлов.
Для установки параметров форматирования небходимо выделить ячейку (группу ячеек) и вызвать диалоговое окно Формат ячеек (пункт меню Формат, команда Ячейки…).
Поскольку на одном листе рабочей книги могут находиться несколько таблиц, каждая из которых имеет свой формат (высота и ширина ячеек), то лучше их располагать «со смещением» (см. пример на листе **Ссуда** рабочей книги **Работа.xls**).



В ячейке могут находится: числовые данные, дата и время, текст, формула (отображается результат вычисления). Формула всегда начинается со знака равно.Формулы представляют собой выражения, по которым выполняются вычисления на странице. Формула также может включать следующие элементы: функции, ссылки, операторы и константы.

9. Web-страница — текстовый файл.

Исходный текст Web-страницы написан на языке описания структуры страницы **HTML** (**H**yper**T**ext **M**arkup **L**anguage — язык гипертекстовой разметки). HTML позволяет форматировать обычный текст в абзацы, заголовки, списки и другие структуры, позволяет создавать ссылки на связанные страницы, т.е. позволяет определять назначение фрагментов текста. HTML есть попытка создать единый стандарт для Mac, PC и для Unix. Язык постоянно развивается, появляются новые стандарты, но не все нововведения могут использоваться, так как они реализованы в web-браузерах по-разному. Поэтому предпочтительно использовать базовые возможности языка HTML.

Код HTML-документа можно писать в любом текстовом редакторе и просматривать результат в web-браузере. При этом необходимо не забывать **обновлять** полученную web-страницу после внесения изменений в исходный код.

HTML-файл — текстовый файл с расширением htm (или html), в котором использованы инструкции по форматированию — теги (tags).

Тег — набор символов. Все теги заключаются в угловых скобках: символы "меньше" (<) и "больше" (>). После открывающей скобки идет ключевое слово, определяющее тег.

Каждый тег имеет специальное значение. Теги не чувствительны к регистру, но принято их набирать прописными буквами.

Существуют **парные** и **одиночные** теги. Парные теги воздействуют на фрагменты текста (открывающие создают эффект, закрывающие прекращают его действие), одиночные дают разовый эффект в месте их появления. Закрывающие теги начинаются с символа "слеш" (/).

парные теги:		одиночные теги:
<HTML>	**</HTML>**	** **
<B>	**</B>**	**<HR>**
<HEAD>	**</HEAD>**	**<META>**
<H3>	**</H3>**	**<BASEFONT>**
<ADDRESS>	**</ADDRESS>**	**<FRAME>**

<LI>	<LI>	<INPUT>

Структура документа.

HTML-документ имеет строго заданную структуру. Он состоит из основного **текста** документа и **тегов разметки**, сам документ — это один контейнер с именем "**HTML**":

```
<HTML> Содержание документа </HTML>
```

HTML-документ должен начинаться с тега **<HTML>** и заканчиваться закрывающем тегом **</HTML>**. Эта пара тегов сообщает браузеру, что данный документ — документ HTML.

Документ HTML состоит из **раздела заголовка** (**HEAD**) и **тела документа** (**BODY**). Раздел заголовка заключен между парными тегами и содержит информацию о документе в целом. Этот раздел должен содержать теги **<TITLE>** и **</TITLE>** — контейнер заголовка, текст которого отображается в строке заголовка окна браузера. Пропуск данного парного тега не вызывает ошибки в современных браузерах, но опускать его не рекомендуется. Пример структуры документа представлен ниже:

```
<HTML>
    <HEAD>
        <TITLE> Содержание заголовка</TITLE>
    </HEAD>
    <BODY>
        Содержание тела документа
    </BODY>
</HTML>
```

Язык HTML предназначен для описания функциональных разделов документов. В обычных документе основными функциональными разделами являются заголовки и абзацы При форматировании документа необходимо взять за правило не ставить лишние пробелы и пустые строки, так как они игнорируются. Тег тела документа **<BODY>** может содержать большое количество атрибутов. Они определяю внешний облик документа.

Большинство фоновых рисунков малы, и их копии заполняют все окно браузера. Можно использовать стандартные текстурные изображения, эмблемы или логотипы. Формат изображения долен быть одним из трех: GIF, JPEG или PNG [**Ошибка! Источник ссылки не найден.**].

Абзацы.

Для обозначения обычных абзацев используют парный тег **<P>** и **</P>**. Перед параграфом автоматически вставляется пустая строка. Закрывающий тег не обязателен, но желательно его не опускать.

Текст в абзаце может быть выровнен по левому краю, правому, по центру, по ширине. Для этого используется атрибут **ALIGN** тега **<P>.**

Заголовки.

Заголовки используются не для простого определения размера и начертания текста,а для создания структуры документа.

HTML поддерживает шесть уровней внутренних заголовков документа, помечаемых тегами от **<H1>** и **</H1>** до **<H6>** и **</H6>**. В окне браузера все заголовки отображаются шрифтами разного размера. Все они могут иметь атрибут выравнивания **ALIGN.**

Например, следующий фрагмент организует заголовки 1-го и 2-го уровней:

```
<HTML>
…
    <BODY>
        <H1>заголовок 1 уровня </H1>
        <H2>заголовок 2 уровня </H2>
        текст
…
    </BODY>
</HTML>
```

Списки.

Списки - одна из разновидности оформления абзаца. Наиболее используемыми являются два варианта списков.

Ненумерованный список (**u**nordered **l**ist) создается двумя тегами: **<UL>** и **<LI>**. Парный тег **<UL>** сдвигает абзац вправо, а одиночный тег **<LI>** — ставит маркер (bullet).

Нумерованный список (**o**rdered **l**ist) задается тегами: **<OL>** и **<LI>**. Парный тег **<OL>** сдвигает абзац вправо, а одиночный тег **<LI>** — ставит номер.

Создать нумерованный список можно следующим образом:

```
<OL>
<LI>1 элемент <BR>
<LI>2 элемент <BR>
```

```
</OL>
```

Тег **OL** атрибут TYPE		Тег **UL** атрибут TYPE	
TYPE="1"	Обычная нумерация.	**TYPE="DISK"**	Маркером - жирная точка.
TYPE="I"	Нумерация большими римскими цифрами	**TYPE="CIRCLE"**	Маркер - кружок.
TYPE="i"	Нумерация малыми римскими цифрами	**TYPE="SQUARE"**	Маркер - квадратик.
TYPE="A"	Нумерация большими латинскими цифрами		
TYPE="a"	Нумерация малыми латинскими цифрами		

Таблицы.

Таблицы — одно из основных средств верстки web-страницы и достижения эффектных результатов. Таблицы строятся их трех элементов: тег таблицы **<TABLE>** (1), теги строк **<TR>** (2), теги ячеек **<TD>** (3). Данные теги — **парные**, наличие закрывающих тегов **обязательно** [**Ошибка! Источник ссылки не найден.**]. Между тегами **<TABLE>** и **</TABLE>** может один раз встретиться пара тегов **<CAPTION>** и **</CAPTION>**, определяющая заголовок таблицы. Заголовок таблицы размещается над таблицей или под ней.

Внутри ячеек таблицы могут содержаться любые данные и любые теги. Ячейка таблицы может содержать вложенную таблицу.

Атрибуты и теги таблицы. **Таблица 7.**

Атрибут	Назначение
Bgcolor="#FFCC33"	цвет фона таблицы (строки, ячейки).
Background="картинка.gif"	задает фоновый рисунок для таблицы (строки, ячейки).
width="50" или **width="50%"**	ширина таблицы (строки, ячейки) в пикселях или процентах.
height="45" или **height="45%"**	высота таблицы (строки, ячейки) в пикселях или процентах.
align="center" (**right**, **left**)	выравнивает содержимое ячейки относительно ее центра (правого или левого краев).
valign="middle" (**top**, **bottom**)	вертикальное выравнивание содержимого строки (ячейки) по середине (наверху или внизу).
colspan="2"	растянуть ячейку на несколько столбцов.
rowspan="2"	растянуть ячейку на несколько рядов.
cellspacing="5"	задает пространство между ячейками.
cellpadding="5"	задает пространство внутри ячейки между ее содержимым и границами.
border="3"	задает толщину рамки таблицы.
bordercolor="#000000"	задает цвет рамки таблицы.

Гиперссылки.

В качестве гиперссылки могут выступать текст, рисунок. Для создания гиперссылки из текста необходимо его пометить парным тегом <A>(anchor – якорь).

Атрибуты гиперссылки. **Таблица 8.**

Атрибут	Назначение
HREF="адрес"	Переход к файлу, путь которого указан либо относительно, либо URL файла (адрес файла в Сети – в случае ссылки на чужой сайт). Важно в пути заменить *бэкслэш* на *слэш*.
TARGET=new	После выбора ссылки, страница будет открыта в новом окне браузера. Таким образом без задержки можно вернуться к исходной странице — не тратиться время на новое открытие исходной страницы в случае «отката».

Например, тег

```
<A HREF="../main/index.html">Содержание</A>
```

осуществляет переход к файлу index.html, расположенному в подкаталоге main надкаталога текущего каталога. Для связи с автором сайта используется тег

```
<A HREF="mailto:auther@mymail.ru"> Ждем писем! </A>
```

А следующий тег

```
<A HREF="mailto:auther@mymail.ru?subject=About solution"> Ждем писем! </A>
```

позволяет в поле темы вставить текст **«About solution».**

Метки.

С помощью меток можно перейти не только в начало страницы, но и перемещаться по тексту. Порядок создания меток и работы с ними таков:

1 шаг. Место, в которое должен происходить переход, помечается тегом

```
<A NAME="Имя Метки"> текст </A>.
```

2 шаг. В гиперссылке на место перехода указывается имя метки.

В случае перехода внутри текущей страницы:

```
<A HREF="#Имя Метки"> текст </A>.
```

В случае перехода к метке в файле index.html:

```
<A HREF="index.html#Имя Метки"> (см. Оглавление)</A>.
```

Существует возможность одновременно и устанавливать ссылку куда–либо, и присваивать имя метке, например, в теге:

```
<A HREF="index.html#Literat" NAME="Part1">(см. Литература)</A>.
```

А в случае нахождения в файле index.html гиперссылки с меткой

```
<A HREF="Glava1.html#Part1" NAME="Literat">(вернуться в Главу 1)</A>.
```

можно вернуться обратно.

Тег <IMG>

Тег **IMG** служит для внедрения графики на страницы. На данный момент поддерживаются форматы файлов GIF, JPG, JPEG, PNG. Возможно использование анимированных картинок (GIF). Чтобы картинка была одновременно и ссылкой, то поместите тег **IMG** между <img src="xakep.gif">

Атрибуты

src Обязательный атрибут, указывающий URL рисунка

align Выравнивает изображение к одной из сторон документа

alt Выводит текст к картинке. Полезно, если браузер не отображает графику на странице

border Устанавливает толщину рамки вокруг изображения в пикселах. По умолчанию этот параметр для ускорения загрузки страницы браузером

hspace Определяет размер свободного места в пикселах слева и справа от изображения, улучшает внешний вид страницы, отделяя изображение от текста

vspace Определяет размер свободного места в пикселах сверху и снизу от изображения, улучшает внешний вид страницы, отделяя изображение от текста

width Ширина картинки в пикселах или процентах. Всегда желательно явно устанавливать этот параметр для ускорения загрузки страницы браузером

20..Алгоритм – заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения некоторой задачи.

Свойства алгоритмов:

1. понятность для использования – получив алгоритм исполнитель должен понимать как его выполнить;
2. дискретность – алгоритм должен представлять процесс решения задачи в виде конечного числа законченных действий;
3. определенность алгоритма – алгоритм должен всякий раз приводить к одному и тому же результату при одних и тех же исходных данных;
4. результативность – за конечное число шагов алгоритм должен либо приводить к решению задачи, либо останавливаться из-за невозможности получить решение и выдавать соответствующие сообщения;
5. массовость – алгоритм должен быть разработан для некоторого класса задач, различающихся исходными данными, а не для одной конкретной задачи. Набор исходных данных для которых применяется алгоритм называется областью применения алгоритма.

Используются следующие формы представления алгоритма:

1. словесная – запись на естественном разговорном языке;
2. графическая – действие изображается в виде графических символов, смысл которых заранее оговорен;
3. псевдокоды – условный алгоритмический язык, включающий в себя элементы естественного разговорного языка и элементы языков программирования;
4. программа на алгоритмическом языке программирования – будет использоваться для представления алгоритмической блок-схемы.

Исполнитель алгоритма – некоторая абстрактная или реальная система, способная выполнить действия, предписываемые алгоритмом.В информатике исполнитель алгоритмов – компьютер.

2)ветвления. Обеспечивает в зависимости от рез-та проверки условие выбора одного из альтернативных путей работы алгоритма. При этом оба пути ведут к одному общему выходу – стр-ра ветвления сущ-ет в 4-х вариантах

цикл. Обеспечивает многократное выполнение некоторых совокупных действий, называемых телом цикла.

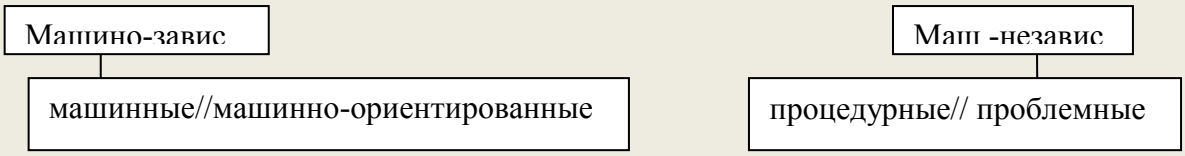
11. Этапы решения задачи на ЭВМ

23. Этапы решения задачи на компьютере:

1. постановка задачи – сбор информации о задаче; формулировка задачи; определение конечных целей задачи;
2. анализ и исследование задачи и составление модели – анализ существующих аналогов; анализ технических и программных средств, имеющихся в наличии; разработка математической модели; определение формы выдачи результатов; описание обработанных данных;
3. разработка алгоритма – выбор метода проектирования алгоритма; выбор формы записи алгоритма; проектирование алгоритма; выбор тестов и методов тестирования;
4. программирование – выбор языка программирования; уточнения способов организации данных; запись алгоритма в виде программы;
5. тестирование и отладка - синтаксическая; отладка логической структуры; тестовые расчеты и анализ результатов тестирования; совершенствование программ;
6. анализ результатов решения задачи (повторное выполнение со 2 по 5 этапы);
7. сопровождение программы – составление документации к алгоритму, к программе, к набору тестов, к использованию.

12 Классификация языков программирования

Языки различаются по зависимости от аппаратной части.



1. машинофиксированный (ассемблеры): имеют жесткую ориентацию на определенный тип компьютеров. Также называются языки низкого уровня. yt nht,e.n rjvgbkzwbb/
аторитмические: ориентированы на систему операторов, характерных для замен определенного класса алгоритмов; языки высокого уровня
Компилятор – проги, переводящие текст на машинный язык.
Стиль - совокупность правил, лежащих в основе синтаксиса и семантики языка программирования.
Различают следующие стили:

1. неструктурный;
2. структурный;
3. логический;
4. объектно-ориентированный;
5. функциональный.

Рассмотрим перечисленные стили подробнее.
Неструктурное программирование допускает использование в явном виде команды безусловного перехода (в большинстве языков GOTO). Типичные представители неструктурных языков - ранние версии Бейсика и Фортрана. Данный стиль вызван особенностями выполнения машиной программы в кодах и унаследован от программ на языке ассемблера, поскольку там команда перехода является обязательной. Однако в языках высокого уровня наличие команды перехода влечет за собой массу серьезных недостатков: программа превращается в "спагетти" с бесконечными переходами вверх-вниз, ее очень трудно сопровождать и модифицировать. Фактически неструктурный стиль программирования не позволяет разрабатывать большие проекты. Ранее широко практиковавшееся первоначальное обучение программированию на базе неструктурного языка (обычно Бейсика) приводило к огромным трудностям при переходе на более современные стили. Как отмечал известный голландский ученый Э. Дейкстра, "программисты, изначально ориентированные на Бейсик, умственно оболванены без надежды на исцеление".

Структурный стиль был разработан в середине 60-х - начале 70-х гг. В его основе лежат две идеи:
- задача разбивается на большое число мелких подзадач, каждая из которых решается своей процедурой или функцией (декомпозиция задачи). При этом проектирование программы идет по принципу сверху вниз: сначала определяются необходимые для решения программы модули, их входы и выходы, а затем уже эти модули разрабатываются. Такой подход вместе с локальными именами переменных позволяет разрабатывать проект силами большого числа программистов.
- как доказал Э. Дейкстра, любой алгоритм можно реализовать, используя лишь три управляющие конструкции: последовательное выполнение, ветвление и цикл (рисунок 1.1; обратите внимание на обозначения в соответствии с действующим стандартом). Данное обстоятельство позволяет при наличии соответствующих операторов исключить из языка команду перехода GOTO.

а) Последовательное б) Ветвление в) Цикл выполнение

Принципы структурного программирования были реализованы в языке Алгол, но наибольшую популярность завоевал язык Паскаль, созданный в 1970 швейцарским ученым Н. Виртом. Паскаль получил широчайшее распространение и может считаться образцовым языком программирования, наиболее популярным и сейчас (например, в версии Delphi фирмы Imprise).

Логическое программирование представляет собой попытку возложить на программиста только постановку задачи, а поиски путей ее решения предоставить транслятору. Логические языки (Пролог, Симула) имеют специальные конструкции для описания объектов и связей между ними. Например, если дано:

БРАТЬЯ ИМЕЮТ ОДНОГО ОТЦА
ДЖОН - ОТЕЦ ДЖЕКА
МАЙК - БРАТ ДЖЕКА
то система логического программирования должна сделать вывод: ДЖОН - ОТЕЦ МАЙКА.
Хотя работы по логическому программированию ведутся с 50-х гг., в настоящее время данное направление несколько потеряло свою актуальность в связи с отсутствием реальных результатов, поскольку большинство реализованных на принципах логического программирования систем оказались практически непригодными.

Объектно-ориентированное (ОО) программирование, разработанное в середине 70-х гг. Кернганом и Риччи и реализованное в ОО-версиях языков Си и Паскаль, представляет собой отображение объектов реального мира, их свойств (атрибутов) и связей между ними при помощи специальных структур данных. Структурное программирование подразумевает наличие ряда встроенных структур данных: целых, вещественных и строковых переменных, массивов, записей - при помощи которых и производится отображение свойств объектов реального мира. При объектно-ориентированном подходе для объекта создается своя структура данных (класс), содержащая как свойства объекта (поля), так и процедуры для управления объектом (методы).
Например, рассмотрим простейший объект - точку на экране. Она имеет как минимум три поля (координаты и цвет) и методы вроде "ChangeColor" (поменять цвет), "MoveXY" (переместиться в точку x, y) и т.д. Объектно-ориентированный подход является в настоящее время доминирующим и позволяет сократить время разработки и увеличить надежность больших проектов. Однако программы в данном стиле отличаются громоздким синтаксисом; в целом идеология объектно-ориентированного подхода весьма неочевидна часто воспринимается с трудом (особенно это характерно для языка Си, который и в своем первоначальном виде отличается крайне неудобочитаемым синтаксисом) и переход к его использованию труден или невозможен для большого числа программистов.

В основе **функционального стиля** лежит понятие функции как "черного ящика", имеющего вектор параметров (аргументов) $\vec{p}$ на входе и результат *r* (скаляр) на выходе:
Допустим случай , т.е. вектор параметров отсутствует. В функциональных языках программирования отсутствуют операторы: все действия, в том числе и управляющие конструкции, выполняются при помощи вызовов функций. Поскольку каждая функция возвращает значение, ее можно подставить в качестве аргумента другой функции, что позволяет записывать сложные выражения в функциональной форме. Одним из первых функциональных языков стал язык Лисп, созданный в конце 50-х гг. как язык искусственного интеллекта.
К языкам искусственного интеллекта (сокращенно обозначается AI - artificial intelligence) относят такие языки, которые способны в зависимости от набора исходных данных модифицировать алгоритм работы, т.е. "на ходу" менять программу (поговорка гласит, что на языках искусственного интеллекта программируют те, кому не хватает интеллекта естественного).

Язык программирования - это система обозначений, служащая для точного описания программ или алгоритмов для ЭВМ. Языки программирования являются искусственными языками, в которых синтаксис и семантика строго определены. Поэтому при применении их по назначению они не допускают свободного толкования выражения, характерного для естественного языка [Толковый,1989].
Программа - это совокупность допустимых операторов конкретного языка программирования.
Программирование , в узком смысле, это запись алгоритма на конкретном языке программирования - языке, "понимаемом" компьютером. Заметим, что довольно часто в литературе можно встретить и другое более широкое определение программирования - это все технические операции, необходимые для создания программы, включая анализ требований и все стадии разработки и реализации алгоритма решения задачи с помощью компьютера.
Напомним, что алгоритм - это точное и понятное предписание (указание) исполнителю совершить последовательность действий, направленных на достижение указанной цели или на решение поставленной задачи. Существует три базовые конструкции алгоритмов: "линейная","ветвление" и "цикл".

В данном пособии Вам предлагается изучать язык программирования Pascal (диалект TURBO Pascal, версия 3.0). Pascal - это императивный язык программирования высокого уровня, имеющий строгую типизацию данных и переменных. Автором языка Pascal является профессор, директор Института информатики Швейцарской высшей политехнической школы Никлаус Вирт, работавший над созданием этого языка в 1965-1971 гг. Язык программирования Pascal был назван в честь Блеза Паскаля, выдающегося французского математика и философа, жившего в 1623-1662 гг.
Императивные языки программирования [Толковый,1989] - это класс языков программирования. Программа, написанная на императивном языке, явно указывает способ получения желаемого результата, не определяя при этом ожидаемых свойств результата. Процедура получения желаемого результата имеет вид последовательности операций, поэтому для императивных языков характерно указание порядка выполнения операторов. В основе такого программирования лежат извлечение из памяти значения некоторой переменной, совершение над ним действия и сохранение нового значения с помощью оператора присваивания. Отметим, что операторы присваивания разрушают информацию (присваиваемое значение заменяет предыдущее значение переменной) и зависят от порядка выполнения.

Система программирования включает в себя:
1. редактор текстов программ;
2. компилятор;
3. компоновщик (компонует исполняемый код программы);
4. отладчик (используется для отладки программ).

Среда программирования Turbo Pascal – интерактивная, работа с которой выполняется с помощью команд меню или управляющих клавиш.
Структура программы: программа состоит из трех частей:
1. заголовок программы(Program имя программы);
2. описательная часть – объявляются все объекты, которые будут использованы в программе с указанием имен и типов данных. Раздел описаний состоит из подразделов:
- uses – список имен, подключенных в стандартных библиотеках;
- uses crt – стирание предыдущих выводов на экран;
- label – описание меток (Label M1, M2);
- const – описание констант (const p=3,1415);
- type (a: array [1..10] of real);
- var – объявление переменных (var a, b, c: real; x, y: integer;);
- procedure – объявление процедурного пользователя;
  function – объявление функций пользователя;

3. раздел операторов (Begin … End.).

Алфавит и словарь языка Pascal. Алфавит содержит:
1. строчные и заглавные буквы латинского алфавита
2. десятичные цифры (от 0 до 10)
3. специальные символы + - * / > < = ; # ‘ . , : { } [ ] ( )

Примечание: буквы русского алфавита и другие символы можно написать только в строковых константах (‘…’).
Словарь Pascal содержит зарезервированные слова [имеют фиксированное написание и навсегда определяют смысл(begin, end, type и т.д.)].Стандартные идентификаторы используются для обозначения встроенных в язык функций, процедур и т.д.
Идентификаторы - имена тех объектов программы, которые создает и называет программист (до 127 символов).

14 Типы данных языка Pascal
Компиляторы языка Pascal требуют, чтобы сведения об объеме памяти, необходимой для работы программы, были предоставлены до начала ее работы. Для этого в разделе описания переменных (var) нужно перечислить все переменные, используемые в программе. Кроме того, необходимо также сообщить компилятору, сколько памяти каждая из этих переменных будет занимать. А еще было бы неплохо заранее условиться о различных операциях, применимых к тем или иным переменным...
Все это можно сообщить программе, просто указав тип будущей переменной. Имея информацию о типе переменной, компилятор "понимает", сколько байтов необходимо отвести под нее, какие действия с ней можно производить и в каких конструкциях она может участвовать.
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.
Типы данных. Данные в программе обрабатываются в виде констант и переменных константы, не изменяют своего значения в процессе выполнения в программе. Переменные могут принимать различные значения в процессе выполнения программы. Переменную можно трактовать как именованную область оперативной памяти, в которой можно записывать различные значения. Для описания множества допустимых значений величин используют указания типа данных. В Pascal существуют стандартные типы данных и есть возможность создавать свои. Типы данных могут быть простыми (целые, вещественные…) или составными (файлы, записи…).
Основные типы данных:
1. целочисленные: integer (-32768 - 32767), longint, byte (0 - 255), shorint (-128 - 127), word (0 - 65537);
2. вещественные: представляют данные которые могут представлять не целые значения (real, single, double, extende, comp);
3. символьный тип: char – определяет множество значений символов, каждый символ кодируется числом от 0 до 255, каждое значение такого типа один символ (var b, a: char; begin … a: = ‘1’; b: = ‘A’;);
4. логические (булевские): Boolean: определяет два возможных значения: истина (true) или ложь (false).

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.
Стандартные конструируемые типы также можно не описывать в разделе type. Однако в некоторых случаях это все равно приходится делать из-за требований синтаксиса. Например, в списке параметров процедур или функций конструкторы типов использовать нельзя
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[2)].

15.Арифметич и логич выраж:
Выражение - синтаксическая единица языка, определяет способ выч нек-ого знач. Знач арифметич выраж являетсяся число. Арифметич выраж строится из констант, перемен и ф-ций числовых типов, а также арифметич действий и круглых скобок.
Приоритеты действий:
Функции, not.
Мультипликативные операции: * , / ,div , mod , and
Аддитивные операции: + , - , or
Отношения: = , <> , > , < , >= , <= , in
Операции одного приоритета вычисляются слева направо. Это соответствует группировке скобок в бесскобочном выражении влево.
Логич выраж имеет знач истина или ложь. Может состоять из логич констант, перемен, опреций отнош-я, логич операций(<,>,=,<=,>=,<>). Применяются к логич выраж. Логич выраж может включать в себя арифметич. Сложные логич выраж сост из логич операций.
NOT-отрицание
AND-логич умножение или конъюкция
OR-сложение или дизъюнкйия(или)
XOR-исключ
Стандартные матем ф-ции:
Frac(x)-дробная часть
Int(x)-целая часть числа, рез-т вещ-ого типа
Trunc(x)-ближ целое, не привыш x по модулю, рез-т целого типа
Round(x)-округление до ближ целого Random(x)-дает целое в диапазоне[0;x]

Random-случ число [0;1] Sin(x)-sinx Cos(x)-cosx Arctg(x)-arctx Exp(x)-е в степени x Ln(x)-lnx Sqr(x)-x в кв Sqrt(x)-корень

17 - 18 Составной оператор- совокупность любого числа операторов, ограниченные операторными скобками «begin...end», т.е. оператор имеет вид
Begin
Оператор1; оператор2;…оператор n; end;
Сост оператор исп в том месте программы, где синтаксис требует наличия одного оператора, а алгоритм требует вып неск действий.
Условные операторы: исп для выбора одного из двух возможных действий в зависимости от нек-ого усл.
1)if..then...else
формат: if усл then оп1 else оп2
Оператор реализует алгоритмич стр-ру полная развилка. Оператор выч знач логич выраж(усл). Оно м.б. либо=истина, либо=ложь. Если усл вып выраж=истина, то вып оп1, иначе вып оп2.
2) if...then
Реализует алгоритмич стр-ру неполная развилка. Если усл вып, то то вып оператор после слова then, иначе ничего не вып., а осущ переход на следующ оператор программы. Операторы внутри оператора могут быть простыми и составиными.
Оператор case: (оп множественного выбора). Исп для выбора одного из нескольких возможных действий.
Формат:
Case k of
Const1:оп1;…
Const n: оп n else оп (n+1); end;
k- выраж-селектор целого, символьного или логич типа
Const1,Const n-того же типа что и селектор.
Этот оп реализует алгоритмич стр-ру «выбор-если» и работает следующ образом:выч знач выраж k. Если в группе const 1...n есть знач, равное вычисляемому, то вып соотв этой константе оператор с последующим выходом за пределы оп case. Если такой конст не нашлось, то вып оператор n+1, далее выход; часть else может отсутствовать. Все конст должны иметь различ знач.
Н:сост программу, к-ая по введенному номеру месяца выводит на экран назв времени года.
Program pr1;Var k:byte; Begin Writeln(‘’); Readln(k); Case k of 12,1,2:writeln(‘зима’); 3,4,5:writeln(‘весна’); 6,7,8:writeln(‘лето’);
9,10,11:writeln(‘осень’); else writeln(‘неверный параметр’) end; end

19 Оператор цикла с параметром while-это оператор цикла с предусловием. Реализует алгоритмическую структуру- цикл «пока» с предусловием . Общий вид оператора: **while** условие **do** оператор; оператор м.б.как простым, так и составным. Оператор-это тело цикла. Перед каждым выполнением тела цикла проверяется условие, если результат истина, то тело цикла выполняется, иначе происходит выход из цикла. Возможен случай,когда тело цикла не выполняется ни разу.

```
ProgramPrim3;
Const n=10;
Var
k,x,s: integer;
begin
k:=0;
s:=0;
while k<n do
begin
k:=k+1;
writeln (‘Введите число,’k);
readln(x);
s:=s+x;
end;
writeln(‘Сумма=’s);
end
```

Оператор цикла с параметром repeat- оператор цикла с постусловием, реализует алгоритмическую структуру(цикл «пока» с постусловием). Общий вид оператора:

```
Repeat
Оператор1;
Оператор2;
…….;
Оператор n;
Until условие;
```

Выполняется тело цикла, затем проверяется условие выхода из цикла. Если условие выполняется => выход из цикла, иначе опять вып-ся тело цикла. Для оператора **Repeat Until** тело цикла вып-ся хотя бы 1раз.

Оператор цикла с параметром For.

Структура:

For <пар_цикл> := <нач_знач> To <кон_знач> Do <оператор>.

<пар_цикл> - параметр цикла – переменная цикла Integer;

<нач_знач> - начальное значение – выражение того же типа;

<кон_знач> - конечное значение - выражение того же типа.

При выполнении оператора FOR вначале вычисляется выражение <нач_знач> и осуществляется присваивание <пар_цикл> := <нач_знач>.

После этого циклически повторяется:

- Проверка условия <пар_цик> <= <кон_знач>; если условие не выполняется, оператор For завершает работу;
- Выполнение оператора <оператор>;
- Наращивание переменной <пар_цик> на еденицу.

```
s:=0
for k:=1 to n do begin
writeln('введите',k);
Readln(x);
S:=s+x;
End;
Writeln(s);
End.
```

Сущ. другая форма оператора:

For <пар.цикл>:=<нач.знач> Downto <кон.знач> Do <оператор>

Замена зарезервированного слова To на Downto означает, что шаг наращивания параметра цикла равен (-1), а управляющее условие приобретает вид <пар.цикл> = <кон.знач>. Теперь можно подсчитывать и отрицательные суммы.

```
For k:=10 down to 1 do
Begin
y:=k*k
writeln(y);
end;
```