1. **Синтаксис** - правила, определяющие допустимые конструкции языка, построенные из символов алфавита, его форму.
2. **Семантика** – правила, определяющие смысл синтаксически корректных предложений, его содержание.
3. **Алфавит включает:**
   1. Латинские буквы без различия строчных и прописных a, b, c…
   2. Арабские цифры 1, 2, 3
   3. Шеснадцатеричные цифры 1, 2, 3, a, b, c
   4. Специальные символы + - \* / = : ^
   5. Служебные слова do while begin end
4. **Синтаксические диаграммы** – диаграммы, отображающие правила построения конструкций в более наглядной форме

2

1. **Константы** – данные, не изменяемые в процессе выполнения программы.

Константы в тексте программы – **литералы**. Виды литералов:

* 1. Числовые
  2. Шеснадцатеричные
  3. Логические
  4. Символьные
  5. Строковые
  6. Адресные

1. **Переменные** - поименованные данные, которые могут изменяться в процессе выполнения программы
2. Классификация скалярных типов
   1. Порядковый (конечный набор чисел)
      1. Стандартный
         1. ***Integer*** Целое
         2. ***Smallint*** Короткое целое
         3. ***Longint*** Длинное целое
         4. ***Word*** Слово
         5. ***Byte*** Байт
         6. ***Bolean*** Булевский
         7. ***Char*** Символ
      2. Перечисление ***Type day := (m,t,w,th,f,s,su)***
      3. Отрезок ***Type date := 1..31***
   2. Вещественный(условно бесконечный набор чисел)
      1. ***Real*** Вещественный
      2. ***Single*** Одинарной точности
      3. ***Double*** Двойной точности
      4. ***Extended*** Расширенный
      5. ***Comp*** Большое целое
3. Функции для порядковых типов
   1. ***Ord*** Номер значения по порядку
   2. ***Pred*** Предыдущее значение
   3. ***Dec*** Значение, уменьшенное на n
   4. ***Succ*** Следующее значение
   5. ***Inc*** Значение увеличенное на n
   6. ***High*** Самое большое значение типа
   7. ***Low*** Самое маленькое значение типа

3

1. **Совместимые типы**
   1. Целые между собой
   2. Вещественные между собой
   3. Отрезок базового типа и базовый тип
   4. Два отрезка одного и того же базового типа
   5. Символ и строка
2. **Неявное автоматическое преобразование** выполняется, если типы результата и переменной не совпадают, но совместимы.
3. **Явное преобразование** требуется, когда типы несовместимы
   1. ***Trunc*** Преобразует вещественное в целое, отбрасывая дробную часть
   2. ***Round*** Преобразует вещественное в целое по правилам арифметики.
   3. ***Ord*** Преобразует значение в его номер
   4. ***Chr*** Преобразует номер в значение

4

1. Оператор присваивания ***:=*** используется для изменения значений переменных. Переменные должны иметь одинаковые или совместимые значения типов.
2. Условный оператор ***if … else*** используется при обработке вариантов вычислений и реализует ветвление
3. Оператор выбора ***case … of*** позволяет программировать несколько вариантов решения
4. Составной оператор записывается в операторных скобках ***Begin … End***

5

1. Счетный цикл  ***for i := 0 to n do*** – цикл, количество повторений которого известно или его можно посчитать. Вывод по счетчику
2. Итерационный цикл ***while … do; repeat … until*** - цикл, количество повторений которого неизвестно или считается неизвестным при построении цикла. Выход по условию.

6

7

1. Массив ***array […] of …***  - упорядоченная совокупность однотипных данных.

Тип индекса определяет доступ к элементу

Тип элемента может быть любым, кроме файла

1. Операции
   1. Присваивание ***:=***
   2. Доступ к элементу ***mas[i]***
   3. Ввод-вывод поэлементно в цикле

8

1. Строка – последовательность символов
2. Операции над строками
   1. Присваивание ***:=***
   2. Обращение к элементу ***string[i]***
   3. Конкатенация ***string[i] := ‘A’ + ‘B’***
   4. Отношение ***‘A’ > ‘B’ => false***
   5. Ввод – вывод ***readln(string);writeln(string);***
3. Процедуры и функции
   1. ***Length()*** Возвращает длину строки
   2. ***Delete(st, a, n)*** Удаляет n символов, начиная с a
   3. ***Insert(st2, st1, i)*** Вставляет st2 в st1 начиная с i
   4. ***Str(x, st)*** Преобразует x в st
   5. ***Val(st, x, code)*** Преобразует st в x; code - номер ошибки
   6. ***Copy(st, a, n)*** Копирует n символов, начиная с a
   7. ***Pos(st2,st1)*** Возвращает где st2 в st1
   8. ***Upcase(ch)*** Возвращает большой символ

9

1. Множество - неупорядоченная совокупность неповторяющихся элементов.
2. Операции над множествами
   1. Присваивание ***:=***
   2. Объединение, пересечение, дополнение ***+ \* -***
   3. Отношение ***= <> > <***
   4. Проверка на вхождение ***in***

10

1. Запись – структура данных, образованная фиксированным числом разнотипных компонентов, называемых полями записи.
2. Операции над записями
   1. Присваивание записей одного типа ***:=***
   2. Доступ к полям записи ***rec.info***
   3. Ввод - вывод по полям ***readln(rec.info); writeln(rec.info)***

11

1. Процедуры и функции ***Procedure Function*** – самостоятельные фрагменты программы, соответствующим образом оформленные и вызываемые по имени.

12

1. Способы передачи данных
   1. Неявно – с использованием средства доступности глобальных переменных из подпрограмм.
   2. Явно – через параметры.
2. Неявная передача данных в подпрограмму

Недостатки:

* 1. Жестко связывает подпрограмму и данные
  2. Приводит к большому количеству ошибок

1. Явная передача данных в подпрограмму

Способы передачи параметров

* 1. Передача по значению
  2. Передача по ссылке
  3. Передача константы

13

1. Глобальные переменные – переменные, обьявленные внутри основной программмы
   1. Время жизни от запуска до завершения программы
   2. Доступность из любого места программы, включая подпрограммы
2. Локальные переменные – переменные, объявленные в подпрограмме
   1. Время жизни от запуска до завершения подпрограммы
   2. Доступность из любого места подпрограмммы и подпрограмм вызываемых данной

14

1. Формальные параметры – параметры, которые передаются в программы. (параметры-значения)
2. Фактические параметры– параметры, адрес которых передается в программу. (параметры-переменные, параметры-константы)

15

1. Структурные типы параметров (массивы, строки, множества, записи, указатели, файлы) должны быть предварительно объявлены.

16

**Открытый массив** – конструкция описания типа массива без указания типа индексов. Используется при объявлении формальных параметров.

Индексы открытого массива всегда начинаются с 0.

Размер можно:

* 1. передать через дополнительный параметр;
  2. получить, используя функцию High(<Идентификатор массива>).

17

1. ***Нетипизированные параметры*** – параметры-переменные, тип которых при объявлении не указан.

Для приведения нетипизированного параметра к определенному типу можно использовать:

* 1. автоопределенное преобразование типов
  2. наложенное описание переменной определенного типа

1. Параметры процедурного типа используются для передачи в подпрограмму имен процедур и функций.
   1. Для объявления процедурного типа используется заголовок подпрограммы, в котором отсутствует имя
   2. Значениями переменных процедурных типов являются идентификаторы процедур и функций с соответствующими заголовками

18

19

1. **Рекурсия** – организация вычислений, при которой процедура или функция обращаются к самим себе.
   1. При явной рекурсии – в теле подпрограммы существует вызов самой себя
   2. При косвенной – вызов осуществляется в подпрограммах, вызываемых из рассматриваемой.

20

1. *Получение адреса* ***@****.* Результат операции – указатель типа pointer – может присвоен любому указателю.
2. Операция разыменования ***^****.* Полученное значение имеет тип, совпадающий с базовым типом данных указателя.

*Нетипизированные указатели разыменовывать нельзя.*

1. Процедура ***new*** –выделяет память для размещения переменной, размер определяется типом указателя.
2. Процедура ***dispose*** –освобождает выделенную память.

21

1. Списки - способ организации данных, предполагающий использование указателей для определения следующего элемента.

Элемент списка состоит из 2 частей

* 1. Информационный содержит поля данных
  2. Адресный включает от одного до n указателей, содержащих адреса следующих элементов. Количество указателей определяет его связность: односвязный, двусвязный… n-связный.

Виды списков

* 1. Линейный
  2. Кольцевой
  3. Древовидный
  4. N-связный

1. Основные приемы работы со списком
   1. Создание элемента
   2. Добавление элемента к списку
   3. Удаление элемента из списка
2. Область применения замена массивов, если не известно кол-во элементов.

22

***Файл*** – поименованная последовательность элементов данных (компонентов файла), хранящихся, как правило, во внешней памяти.

Как исключение данные файла могут не храниться, а вводиться с внешних устройств (ВУ), например клавиатуры или выводиться на ВУ, например экран.

Полное имя файла включает:

<Имя диска>:<Список имен каталогов>\<Имя файла>.<Расширение>

Имя файла в Windows составляют из строчных и прописных букв латинского и русского алфавитов, арабских цифр и некоторых специальных символов, например, символов подчеркивания «\_» или доллара «$»

***Файл языка Pascal*** – последовательность однотипных компонентов: файл записей, файл целых чисел, файл строк.

В зависимости от типа компонентов различают три типа файлов: *типизированные*, *текстовые* и *нетипизированные*.

Количество компонентов файла при объявлении файловой переменной не указывается.

Максимальный размер файла определяется свободным пространством на устройстве, например, диске.

Физически операции ввода-вывода с файлами выполняются с использованием буфера.

Для файлов принципиально возможен не только последовательный, но и произвольный доступ, при котором чтение информацииосуществляется из указанного места.