

The background of the slide is a blue-tinted image of a chalkboard. On the board, there are faint, handwritten mathematical formulas, including what appears to be a binomial expansion  $(a+b)^n = a^n + n a^{n-1} b + \dots + b^n$ . A single piece of white chalk lies horizontally on the lower part of the board.

Лекция №3

# Типы данных

# План лекции

1. Концепция типа данных
2. Типы данных в языке Паскаль
3. Базовые и конструируемые типы
4. Раздел описания типов
5. Порядковые типы данных
  - 5.1. Функции и процедуры применяемые к порядковым типам
  - 5.2. Целочисленные типы данных
  - 5.3. Логический тип данных
  - 5.4. Символьный тип данных
  - 5.5. Перечисляемые типы данных
  - 5.6. Интервальные типы данных
6. Вещественные типы данных
  - 6.1. Вещественные типы
  - 6.2. Запись вещественных чисел
7. Форматный вывод данных
8. Конструируемые типы данных
9. Преобразование типов
  - 9.1. Неявное преобразование типов
  - 9.2. Явное преобразование типов
  - 9.3. Функции изменяющие тип данных

# Концепция типа данных

Для временного хранения информации в операторах памяти машины в языке Паскаль используются константы и переменные. Они могут быть различных типов.

Все данные, используемые в программе, **должны быть предварительно определены.**

Для каждого данного надо обозначить

- имя;
- характер и диапазон изменения значений;
- требуемую память для размещения;
- набор допустимых к ним операций.

# Концепция типа данных

Тип данных определяет:

- возможные значения переменных, констант, функций, выражений, принадлежащих к данному типу;
- внутреннюю форму представления данных в ЭВМ;
- операции и функции, которые могут выполняться над величинами, принадлежащими к данному типу.

# Типы данных в языке Паскаль

- Типы данных
  - Простые
    - Порядковые
      - Целые
    - Логические
    - Символьные
    - Перечисляемые
  - Вещественные
- Структурированные
  - Массивы
  - Множества
  - Записи
  - Файлы
- Указатели
- Строки
- Процедурные
- Объекты



# Базовые и конструируемые типы

**Базовые типы** – типы, определяемые в языке программирования.

**Конструируемые типы** – типы, которые задаются программистом.

# Базовые и конструируемые типы

Типы данных

Базовые

Конструируемые

Целые Литералы в отдельности и в составе массива Массивы Целые Литералы в отдельности и в составе массива

# Базовые и конструируемые типы

Например переменные базовых типов могут быть определены в разделе описания переменных

Var

```
a, b : real;  
d: integer;
```

Конструируемые типы так же могут быть описаны в разделе описания переменных

Var

```
s : string;
```



# Раздел описания типов

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

Type

<ИМЯ\_ТИПА> = <описание\_типа>;

Например:

Type

lat\_bukvy = 'a'..'z','A'..'Z';

# Раздел описания типов

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;
```

Тогда можно описать переменные

Var

```
x, y : int;
```

# Порядковые типы данных

## Целые:

shortint

byte

integer

word

longint

## Логические:

boolean

## Символьные:

char;

## Перечисляемые:

задаются перечислением значений и/или диапазонами значений.

# Функции применяемые к порядковым типам

`ord(x)` возвращает порядковый номер значения переменной `x` (относительно того типа, к которому принадлежит переменная `x`).

`pred(x)` возвращает значение, предшествующее `x` (к первому элементу типа неприменима).

`succ(x)` возвращает значение, следующее за `x` (к последнему элементу типа неприменима).

# Процедуры применяемые к порядковым типам

$\text{inc}(x)$  возвращает значение, следующее за  $x$  (для арифметических типов данных это эквивалентно оператору  $x := x + 1$ ).

$\text{inc}(x, k)$  возвращает  $k$ -е значение, следующее за  $x$  (для арифметических типов данных это эквивалентно оператору  $x := x + k$ ).

$\text{dec}(x)$  возвращает значение, предшествующее  $x$  (для арифметических типов данных это эквивалентно оператору  $x := x - 1$ ).

$\text{dec}(x, k)$  возвращает  $k$ -е значение, предшествующее  $x$  (для арифметических типов данных это эквивалентно оператору  $x := x - k$ ).

# Целочисленные типы данных

| Тип данных | Количество бит | Диапазон                |                     |
|------------|----------------|-------------------------|---------------------|
| shortint   | 8              | -128..127               | $-2^7..2^7-1$       |
| byte       | 8              | 0..255                  | $0..2^8-1$          |
| integer    | 16             | -32768..32767           | $-2^{15}..2^{15}-1$ |
| word       | 16             | 0..65535                | $0..2^{16}-1$       |
| longint    | 32             | -2147483648..2147483647 | $-2^{31}..2^{31}-1$ |

Над целыми типами определены такие операции:

+   -   \*   /   mod   div



# Логический тип данных

Логический тип `boolean` имеет два значения:  
`false` и `true`

Над операндами логического типа определены такие операции:

`or`, `and`, `not`, `xor`

Для логического типа выполняются следующие равенства:

`ord(false)=0`, `ord(true)=1`, `false<true`,  
`pred(true)=false`, `succ(false)=true`,  
`inc(true)=false`, `inc(false)=true`,  
`dec(true)=false`, `dec(false)=true`.

# Символьный тип данных

В символьный тип `char` входит 256 символов расширенной таблицы ASCII

Например,

`'a', 'b', 'я', '7', '&'`

Номер символа, возвращаемый функцией `ord()`, совпадает с номером этого символа в таблице ASCII.

# Символьный тип данных

Пример описания символьной переменной:

```
Var
    simb1, simb2 : char;
Begin
    simb1:='R'; simb2:=#65; { С помощью # производится
    перевод целого числа в
соответствующий символ                данного ASCII-кода }
    write (simb1,simb2);
End.
```

Результат работы программы

RA

# Перечисляемые типы данных

**Перечисляемые** типы данных задаются в разделе **Типе** явным перечислением их элементов.

Например:

Типе

```
week =(sun,mon,tue,wed,thu,fri,sat)
```

Напомним, что для этого типа данных:

```
inc(sat) = sun, dec(sun) = sat.
```

# Интервальные типы данных (диапазоны)

**Интервальные** типы данных задаются только границами своего диапазона.

Например:

Type

```
month = 1..12;
```

Программист может создавать и собственные типы данных, являющиеся комбинацией нескольких стандартных типов.

Например:

Type

```
valid_for_identifiers = 'a'..'z','A'..'Z','_', '0'..'9';
```

# Вещественные типы данных

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

Вещественные типы данных являются арифметическими, но не порядковыми.

Следовательно для этих типов данных выполняются арифметические операции (за исключением операций с целыми числами) и стандартные математические функции.



# Запись вещественных чисел

| Математическая запись  | Запись на Паскале         |
|------------------------|---------------------------|
| $4 \cdot 10^{-4}$      | 4E -4                     |
| $0,62 \cdot 10^5$      | 0.62E+5    либо    .62E+5 |
| $-10,88 \cdot 10^{12}$ | -10.88E12                 |

# Форматный вывод данных

Пусть

```
a:=7; b:='x'; c:=-10.5;
```

Если для вывода информации воспользоваться командой

```
write (a,b,c);
```

то выводимые символы окажутся "склепленными".

```
7x-1.0500000000000000E+1
```

# Форматный вывод данных

Используем следующую запись оператора вывода

```
write(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.

Получим

```
_____7 x _____ -10.50000
```

(подчерк служит для визуализации пробела)

# Конструируемые типы данных

Конструируемые

Массивы | Ножесткие записи | Файлы | **Числа** | Задачи | Строки | Процедуры | Объекты

Конструируемые типы данных будут изучены на последующих лекциях.

# Совместимость типов данных

В общем случае при выполнении арифметических (и любых других) операций компилятору требуется, чтобы типы операндов совпадали.

Нельзя, например, сложить массив и множество, нельзя передать вещественное число переменной, ожидающей целый аргумент, и т.п.

В то же время, любая переменная, в расчете на вещественные значения, сможет работать и с целыми числами.

# Неявное преобразование типов

Тип результата арифметических операций (а следовательно, и выражений) может отличаться от типов исходных операндов.

Пример:

Var

a,b : integer;

d : real;

Begin

read (a,b);

r:=a/b;

write (r);

End.



# Неявное преобразование типов

Если в некоторой операции присваивания участвуют два типа данных совместимых, но не совместимых по присваиванию, то тип присваиваемого выражения автоматически заменяется на подходящий.

Пример:

Var

a : byte;

Begin

a:=10;

a:=-a;

write (a);

End.

На экране мы увидим не -10, а 246 ( $246 = 256 - 10$ ).

# Явное преобразование типов

Тип значения можно изменить и **явным** способом:  
просто указав новый тип выражения.

Пример:

```
a:= byte(b);
```

В этом случае переменной *a* будет присвоено значение, полученное новой интерпретацией значения переменной *b*.

Скажем, если *b* имеет тип `shortint` и значение `-23`,  
то в *a* запишется `233 (= 256 - 23)`.

# Функции изменяющие тип данных

Функции округления:

**trunc**          real -> integer

**round**          real -> integer

Функция преобразования строки в число

**val**            string -> byte/integer/real

Получение символа по заданному ASCII-коду

**chr**            byte -> char

Преобразование порядковых типов

**ord**            <порядковый\_тип> -> longint

