Смирнов М.В. МКО ООШ п. Климковка Белохолуницкого района Кировской области



# Структурированные типы данных

Информатика и ИКТ



## Общие замечания

Структурированные типы данных определяют упорядоченную совокупность скалярных переменных и харак-теризуются типом своих компонентов.

## Общие замечания

# В Паскале допускаются следующие структурированные типы данных:

- ◆ строки
- ◆ массивы
- множества
- ◆ записи

- файлы
- указатели
- процедурные типы
- ❖ объекты

# О чем пойдет речь

# Структурированные типы данных

- Строковый (строки)
- Строковые операции функции и процедуры
- Массивы
- Множества. Операции над множествами
- ❖ Записи

**Строковая константа** – последовательность символов, заключенная в апострофы.

Максимальная длина строковой константы 255 символов

Пример: '234', 'константа'

Два следующих друг за другом апострофа ('') обозначают пустую строку, т.е. строку с нулевой длиной

# Строковые переменные

Строки в **Pascal** имеют тип **string** и состоят не более чем из 255 символов.

Под переменную S отводится 256 байт, при этом в нулевом байте хранится длина строки.

При описании var S: string;

Тип **String** и стандартный тип **Char** совместимы. Строки и символы могут использоваться в одних выражениях

Для экономии памяти предусмотрено описание вида **var** S1: **string**[n];

В этом случае под строку отводится **n+1** байт (нулевой байт - под длину строки). В случае присваивания переменной S1 строки из более чем **n** символов лишние символы отсекаются, и длина строки S1 полагается равной **n**.

Пример: var S1:string[40]

К символам в строке можно обращать-ся, используя индекс: **s[i]** обозначает **i**-тый символ в строке. Обращение к нулевому символу **s[0]** считается ошибочным.

Если индекс і выходит за пределы дли-ны строки, то сообщение об ошибке не выдается.

Индекс і принимает значения от 1 до 255, если длина строки не указана или от 1 до n, где n — максимальная длина строки, указанная при описании переменной.

## Операции над строками

# Операция сцепления (конкатенации) – (+)

Применяется для соединения нескольких строк в одну результирующую строку. Сцеплять можно как строковые константы, так и переменные.

## Пример:

'Мама '+'мыла '+'раму' = 'Мама мыла раму'.

# Длина результирующей строки не должна превышать 255 символов

## Операции над строками

Операция отношения: =, >, <, >=, <=, <>.

Позволяют производить сравнение двух строк. Результат **true** или **false**.

Сравнение строк происходит слева направо до первого несовпадающего символа, и та строка считается больше, в которой первый несовпада-ющий символ имеет больший номер в таблице символьной кодировки.

Если строки имеют разную длину и в общей части символы совпадают, то более короткая строка считается меньше.

Length (S) – определяет длину строки S.

Результат – значение целого типа

<u>Синтаксис записи</u>: a := Length(s)

Значение <b>S</b>	Выражение	Результат
'функция'	a:= Length(s)	a := 7

**Copy (S, Poz, N)** – выделяет из строки S подстроку длиной N символов, начиная с позиции Poz. Здесь N и Poz целочисленные выражения.

<u>Синтаксис записи</u>: a := Copy (S, Poz, N)

Значение <b>S</b>	Выражение	Результат
'Мама мыла раму'	a:=copy(S,6,4)	а := 'мыла'

Pos (S1,S2) – обнаруживает первое появление в строке S2 подстроки S1.

**Результат** – целое число, равное номеру позиции, где находится первый символ подстроки S1. Если в S2 подстроки S1 не обнаружено, то результат равен 0.

<u>Синтаксис записи</u>: а := **Pos (S1,S2)** 

Значение <b>S2</b>	Выражение	Результат		
'Мама мыла раму'	a:=pos('мыла',S2)	a := 6		

Concat (S1,S2, ...,Sn) – выполняет сцепление (конкатенацию) строк S1, S1, ... Sn в одну строку

<u>Синтаксис записи</u>: а := **concat (S1,S2**, ...,**Sn)** 

Выражение	Результат			
Concat('Мама ','мыла ','раму')	а := 'Мама мыла раму'			

IntToStr(N) – преобразует целое число к строке

<u>Синтаксис записи</u>: S := IntToStr(N)

Значение N	Выражение	Результат
12	S:= IntToStr(12)	S := '12'

StrToInt(S) – преобразует строку в целое число

<u>Синтаксис записи</u>: N := StrToInt(S)

Значение S	Выражение	Результат		
'12'	N:= StrToInt('12')	N := 12		

**Trim(S)** – возвращает копию строки S с удаленными лидирующими и заключительными пробелами

<u>Синтаксис записи</u>: S1 := **Trim(S)** 

Значение S	Выражение	Результат		
' привет '	S1:= Trim(S)	S1:='привет'		

**Val(S, v, code)** - преобразует строку S к числовому представлению и записывает результат в переменную v. Если преобразование возможно, то в переменной соde возвращается 0, если невозможно, то в соde возвращается ненулевое значение. Переменные V и соde должны иметь тип **integer.** 

Синтаксис записи: Val(S, v, code)

Значение	e S	Оператор	Результат
S:='12	' Va	al('12',v,code)	V:=12, code<>0

**Str(x, s)** - преобразует число x к строковому представлению

Переменная X может иметь тип integer или real

Синтаксис записи: Str(x, S)

Значение Х	Оператор	Результат S
x:=12.4	Srt(12,s)	s:='12.4'

**Delete (S, Poz, N)** – удаление N символов из строки S, начиная с позиции Poz.

Здесь N и Роz целочисленные выражения.

<u>Синтаксис записи</u>: delete (S, Poz, N)

Исходное значение S	Оператор	Конечное значение S		
'Мама мыла раму'	delete(S,6,4)	'Мама раму'		

Insert (S1, S2,Poz) – вставка строки S1 в строку S2, начиная с позиции Poz.

Синтаксис записи: Insert (S1, S2, Poz)

**Пример:** S1:='мыла'

Исходное значение S2	Оператор	Конечное значение S2
'Мама раму'	Insert(S1,S2,6)	'Мама мыла раму'

<u>назад</u>

**МАССИВ** – это пронумерованная конечная последовательность однотипных величин. Величины входящие в состав массива называют элементами массива.

Каждый элемент массива имеет два атрибута: индекс и значение.

Индекс элементов меняться не может в отличии от значения элемента массива.

Обращение к элементам массива происходит по его имени и индексу.

Имя элемента массива **совпадает** с именем массива

**Например** : a[5]; b[1,7]

**ИНДЕКС** – константа, переменная или выражение целого типа, определяющее порядковый номер элемента в массиве.

**Линейные** (одномерные) массивы — это массивы элементу у которых — простые переменные.

В одномерных массивах хранят значения линейных таблиц.

Месяц	1	2	3	4	5	6	7	8	9	10	11	12
T,C	-25	-20	-5	5.6	10	18	22.2	24	17	5.4	-7	-18

Значения температуры в таблице – это элементы массива: **Т [1], Т[2], ... Т[12]** 

#### Описание линейных массивов

#### Var

```
<имя массива>: array [нижняя граница массива]. верхняя граница массива] of
```

```
Var a: array [1..10] of integer;
b: array [1..12] of real;
c: array ['a'..'z'] of real;
```

Двумерные массивы — структура данных, хранящая прямоугольную матрицу (таблицу). В матрице каждый элемент определяется ном-ером строки и номером столбца, на пересече-нии которых он расположен.

- **Двумерные массивы** структура данных, хранящая прямоугольную матрицу (таблицу). В матрице каждый элемент определяется номером строки и номером столбца, на пересечении которых он расположен.
- В Паскале двумерный массив рассматрива-ют как массив, элементами которого являются линейные массивы, т.е. массив линейных массивов

## Описание двумерных массивов

## Пример:

Var a: **array** [1..10,1..5] **of** integer; b: **array** [1..12,1..50] **of** real;

Элементы двумерного массива идентифицируются переменными в двумя индексами. Первый индекс связывают с номером строки, второй — с номером столбца матрицы (прямоугольной таблицы)

Фундаментальное отличие компонента массива от простой переменной состоит в том, что для элемента массива в квадратных скобках может стоять не только непосредственное значение индекса, но и выражение, приводящее к значению индексного типа. Таким образом реализуется косвенная адресация:

В[15] - прямая адресация;

**В[K]** - косвенная адресация через переменную K, значение которой будет использовано в качестве индекса элемента массива **B**.

Такая организация работы с такой структурой данных, как массив, позволяет использовать цикл для заполнения, обработки и распечатки его содержимого.

29

Заполнение массивов происходит поэлементно с использованием операторов цикла.

## Пример:

- 1. **Заполнение линейного массива** For i:=1 to 10 do b[i]:=i;
- 2. Заполнение двумерного массива for i:=1 to 10 do for J:=1 to 5 do a[i,i] := i+j;

<u>назад</u>

#### Основные понятия

**Множество** – это структурированный тип данных, представляющий собой набор взаимосвязанных по какому-либо признаку или группе признаков объектов, которые можно рассматривать как единое целое.

#### Основные понятия

Каждый элемент в множестве называется элементом множества.

Все элементы множества должны принадлежать одному из скалярных типов, кроме вещественного. Этот тип называется **базовым** типом множества. Базовый тип задается диапазоном или перечисле-нием.

Область значений типа множество — набор всевозможных подмножеств, составленных из элементов базового типа

#### Основные понятия

- В выражениях на языке Паскаль значения элементов множества указывается в квадратных скобках: [1,2,3,4], ['a', 'b', 'c'], [1..20].
- Если множество не имеет элементов оно называется **пустым** и обозначается как [].
- Количество элементов множества называется его мощностью.
- В Паскале отсутствуют средства ввода-вывода элементов множества.

## Описание множеств. Вариант 1

Формат записи множественных типов

## **Type**

```
<имя типа> = set of <элемент1, .., элемент N >;
```

Var <идентификатор> : <имя типа>

## Пример:

```
Type exmpl = set of 'a', 'b', 'z'; namber = set of 1..31;
```

Var a: exmpl; b: namber;

## Описание множеств. Вариант 2

Задать множественный тип можно и без предварительного описания

#### Var

```
<идентификатор> : set of <элемент1, .. элемент</li>2>
```

```
Var a: set of 'a', 'b', 'z';b: set of 1..31;
```

## Операции над множествами

Операция «равно» (=). Два множества А и В считаются равными, если они состоят из одних и тех же элементов. Порядок следования элементов в множества м.б. любым.

Значение А	Значение В	Выражение	Результат
[1,2,3,4,5]	[4,5,2,3,1]	A=B	True
[1,2,3,4,5]	[4,5,8,3,1]	A=B	False

Операция «не равно» (< >). Два множества А и В считаются не равными, если они отличаются по мощности или по значению хотя бы одного элемента

Значение А	Значение В	Выражение	Результат
[1,2,3,4,5]	[4,5,2,3,1]	A<>B	False
[1,2,3,4,5]	[4,5,8,3,1]	A<>B	True

Операция «больше или равно» (>=). Множество А >= множества В, если все элементы множества В содержатся в множестве А, т.е. множество В явля-ется частью множества А.

Значение А	Значение В	Выражение	Результат
[1,2,3,4,5]	[4,6,2]	A>=B	False
[1,2,3,4,5]	[4,5,3,1]	A>=B	True

Операция «меньше или равно» (>=). Множество А <= множества В, если все элементы множества А содержатся в множестве В, т.е. множество А является частью множества В.

Значение А	Значение В	Выражение	Результат
[1,2,6]	[1,3,2,4]	A<=B	False
[1,2,3]	[2,5,3,1]	A<=B	True

**Операция «in»**. Данная операция используется для проверки принадлежности какого-либо значения указан-ному множеству. Обычно применяется в условных операторах.

При использовании операции «in» проверяемое на принадлежность значение и множество в квадратных скобках не обязательно предварительно описывать в разделе описаний.

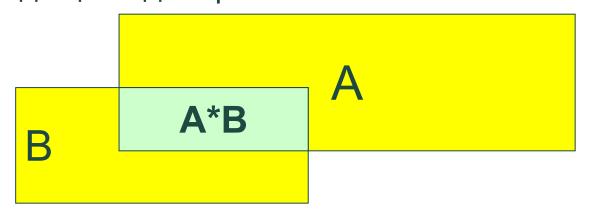
Значение А	Выражение	Результат
2	If A in [1,3,4,5] then	False
2	If A in [1,2,4,5] then	True

**Объединение множеств (+)** – объединением двух множеств является третье множество, содержащее элементы обоих множеств.

+ A
B

Значение А	Значение В	Выражение	Результат
[1,2,6]	[5,2,4]	A+B	[1,2,6,5,4]

**Пересечение множеств (\*)** – пересечением двух множеств является третье множество, которое содержит элементы, входящие одновременно в оба множества.



Значение А	Значение В	Выражение	Результат
[1,2,6,4,3,8]	[1,5,2,4]	A*B	[1,2,4]

**Разность множеств (-)** — разностью двух множеств является третье множество, которое содержит элемен-ты первого множества, не входящие во второе множество.

A-B

B

Значение А	Значение В	Выражение	Результат
[1,2,6,5,7]	[5,2,4]	A-B	[1,6,7]

<u>назад</u>

- **Запись** это структурированный тип данных, состоящий из компонентов одного или нескольких типов.
- Определение типа записи начинается идентификатором record и заканчивается зарезервированным словом end.
- Между ними заключен список компонентов, называемых **полями**, с указанием идентификато-ров полей и типа каждого поля.

## Пример описания записи:

## **Type**

Car = record

Numbe : integer; {Номер}

Marka : string[40]; {Марка авто}

FIO : string[40]; {Фамилия, имя, отчество}

Address: string[60] {Адрес владельца авто}

end;

Var M, V: Car;

Идентификатор поля д.б. **уникален** только в пределах записи, однако во избежание ошибок лучше делать его уникальным в **пределах** всей программы.

Обращение к значению поля осуществляется с помощью идентификатора переменной и иденти-фикатора поля, разделенных точкой.
Такая комбинация называется составным именем.

**Например**: M.Namber; M.FIO; M.Addres

Составное имя можно использовать везде, где допустимо применение типа **поле.** 

Примеры:

M.Namber := 163 M.Marka := ' $\Gamma$ A3-24'

Read (M.Namber, M.FIO);

Write (M.Addres, M.Marka);

В ряде задач удобно пользоваться массивами из записей. Их можно определить т.о.

```
Type
    Person = record
    Name_1 : string[20];
    Name_2 : string[20];
    Age : byte;
    Prof : string[30];
end.

Var List: array[1..59 ] of Person;
```

Обращение к полям записи имеет несколько громоздкий вид. Для решения этой проблемы в языке Паскаль предусмотрен оператор **With**, который имеет следующий формат:

```
With <переменная типа запись> do begin поле_1 записи <оператор>; поле_2 записи <оператор>; поле_3 записи <оператор>; End;
```

**Пример**: Присвоить значения полям записи Car с помощью оператора **with** 

```
With M do begin
Number := 1234;
Marka := 'ГАЗ-24';
FIO := 'Петров И.О.';
Addres := 'ул.Труда 24'
End;
```

Паскаль допускает вложение записей друг в друга (т.е. поле в записи может быть в свою очередь тоже записью). Уровень вложения не должен превышать **9**.