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

Структурированные типы данных

Информатика и ИКТ

Pascal

Общие замечания

Структурированные типы данных определяют упорядоченную совокуп-ность 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

Строковый тип данных

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

Строковый тип данных

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

Строковый тип данных

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]

Строковый тип данных

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 **i** выходит за пределы дли-ны строки, то сообщение об ошибке не выдается.

Индекс i 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.

Строковые функции

Length (S) – определяет длину строки S.

Результат – значение целого типа

Синтаксис записи: a := **Length(s)**

Пример:

Значение **S**	Выражение	Результат
‘функция’	a:= Length(s)	a := 7

Строковые функции

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

Синтаксис записи: a := **Copy (S, Poz, N)**

Пример:

Значение **S**	Выражение	Результат
‘Мама мыла раму’	a:=copy(S,6,4)	a := ‘мыла’

Строковые функции

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.

Синтаксис записи: a := **Pos (S1,S2)**

Пример:

Значение **S2**	Выражение	Результат
‘Мама мыла раму’	a:=pos(‘мыла’,S2)	a := 6

Строковые функции

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

Синтаксис записи: a := **concat (S1,S2, …,Sn)**

Пример:

Выражение	Результат
Concat(‘Мама ‘,’мыла ‘,’раму’)	a := ‘Мама мыла раму’

Строковые функции

IntToStr(N) – преобразует целое число к строке

Синтаксис записи: S := **IntToStr(N)**

Пример:

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

Строковые функции

StrToInt(S) – преобразует строку в целое число

Синтаксис записи: N := **StrToInt(S)**

Пример:

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

Строковые функции

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

Синтаксис записи: S1 := **Trim(S)**

Пример:

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

Строковые процедуры

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

Синтаксис записи: **Val(S, v, code)**

Пример:

Значение S	Оператор	Результат
S:=’12’	Val(’12’,v,code)	V:=12, code<>0

Строковые процедуры

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

Переменная **X** может иметь тип **integer** или **real**

Синтаксис записи: **Str(x, S)**

Пример:

Значение X	Оператор	Результат S
x:=12.4	Srt(12,s)	s:=’12.4’

Строковые процедуры

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

Здесь N и Poz целочисленные выражения.

Синтаксис записи: delete **(S, Poz, N)**

Пример:

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

Строковые процедуры

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’

назад

Массивы элементов

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.

Массивы элементов

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.

Массивы элементов

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: **T [1], T[2], . . . T[12]**

Массивы элементов

Описание линейных массивов

Var

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

Пример:

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

Массивы элементов

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.

Массивы элементов

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

Массивы элементов

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)

Массивы элементов

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:

B[15] - прямая адресация;

B[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.

Массивы элементов

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,j] := i+j;
```

назад

Множества

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.

Множества

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

Множества

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.

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 ;

Множества

Описание множеств. Вариант 2

Задать множественный тип можно и без предварительного описания

Var

<идентификатор> : set of <элемент1, .. элемент 2>

Пример:

Var **a**: set of ‘a’, ‘b’, ‘z’;

b: set of 1..31;

Операции над множествами

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

Значение A	Значение B	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

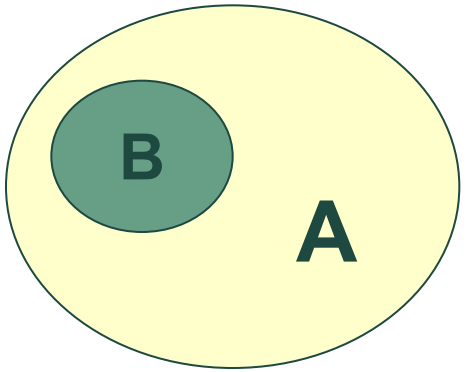
Операции над множествами

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

Значение A	Значение B	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

Операции над множествами

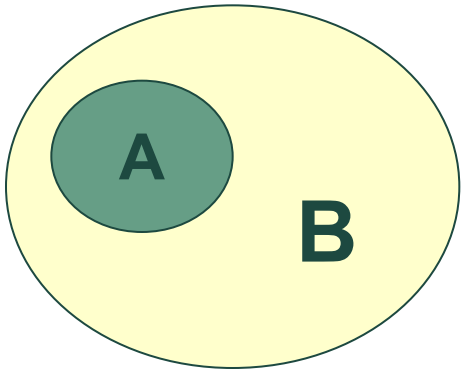
Операция «больше или равно» (**>=**). Множество A >= множества B, если все элементы множества B содержатся в множестве A, т.е. множество B явля-ется частью множества A.



Значение A	Значение B	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

Операции над множествами

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



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

Операции над множествами

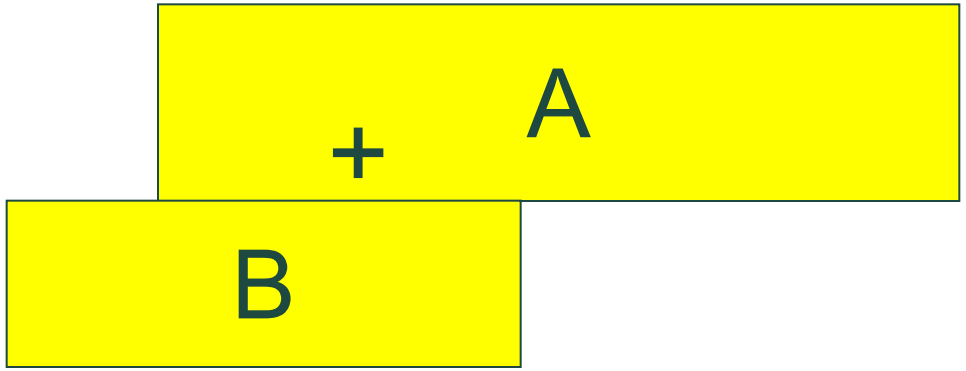
Операция «in». Данная операция используется для проверки принадлежности какого-либо значения указан-ному множеству. Обычно применяется в условных операторах.

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

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

Операции над множествами

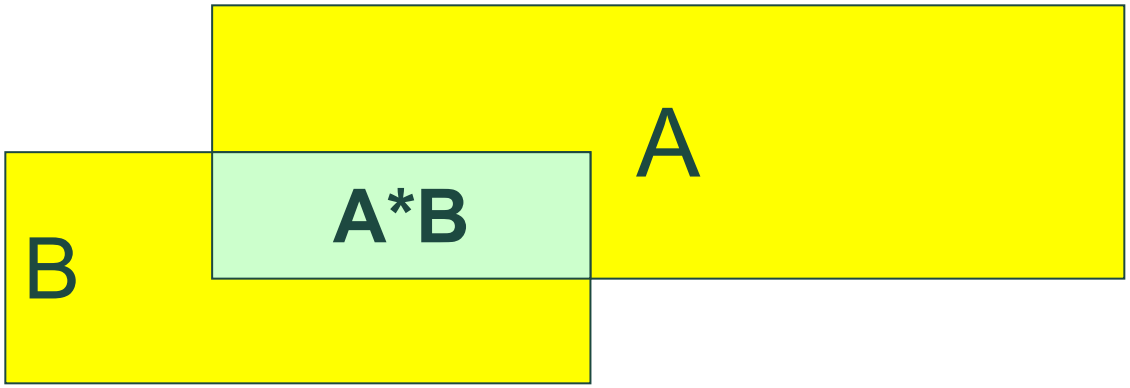
Объединение множеств (+) – объединением двух множеств является третье множество, содержащее элементы обоих множеств.



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

Операции над множествами

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



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

Операции над множествами

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

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

назад

Записи

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.

Записи

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.

Записи

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

Записи

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

Примеры:

M.Namber := 163 M.Marka := ‘ГАЗ-24’

Read (M.Namber, M.FIO);

Write (M.Addres, M.Marka);

Записи

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;
```

Записи

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;

Записи

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**.

назад