Массивы

---

1. Структурные типы данных

Мы различаем основные и структурные операторы. Аналогичным образом следует различать ***основные*** и ***структурные типы данных***. К основным относятся типы: Integer, Real, Boolean и Char. К структурным типам относятся: массивы, записи, множества и файлы. Строго говоря, так называются переменные соответствующих структурных типов, которые следовало бы называть "тип массивов", "тип записей" и т.д. Из контекста должно быть ясно, о чём идёт речь. Каждый из указанных типов строится на базе более простых типов и в конечном итоге на базе основных типов. Для каждого структурного типа в АЯ Pascal определена ***конструкция описания типа*** и ***конструкция обращения к элементам переменных типа***. Для вновь создаваемых типов всегда выполнима операция присваивания. При этом тип переменной, которой присваивается значение, и тип присваиваемого значения должны совпадать. Это положение получило название ***совместимости типов по присваиванию***. Возможности введения новых операций для вновь создаваемых типов, как правило, ограничиваются применением процедур. Кратко охарактеризуем структурные типы.

Массив - совокупность однородных элементов базового типа, обращение к которым выполняется с помощью индексов ординального типа.

Запись - совокупность неоднородных элементов базовых типов, обращение к которым выполняется с помощью имён полей.

Множество - совокупность однотипных элементов базового типа соответствующая понятию множества в математике с набором свойственных операций.

Файл - совокупность однородных элементов базового типа, доступ к которым осуществляется последовательно. В отличие от других структурных типов, переменные которых хранятся в ОП, файлы хранятся на ВУ. В этой связи возникает необходимость в выполнении процедур открытия и закрытия файла, в ходе выполнения которых происходит установление или разрыв связей с ВУ. Доступ к элементам файлов осуществляется с помощью стандартных процедур Read и Write.

Базовыми типами для структурных типов могут быть любые типы, кроме файлов.

2. Структурный тип массивов

Задача. S := A + B + C + D + E; При увеличении количества переменных увеличивается размер программы.

В математике используются переменные с индексами: A[i], где i = 1..n .

Это позволяет использовать для вычисления суммы рекуррентное соотношение

S[1] = A[1], S[i] = S[i-1] + A[i], где i = 2..n . Здесь S[i] - сумма первых i членов последовательности переменных. Эта идея реализована в программе

```
S := 0; i := 1;
while i <= n do begin S := S + A[i]; i := i + 1 end;
```

При этом размер программы никак не зависит от количества переменных.

Описание типа:

array [<Тип индекса>] **of** <Имя или описание базового типа>

Тип индекса - любой ординальный тип. Обычно используется отрезок типа Cmin..Cmax, где Cmin и Cmax - константы, определяющие соответственно минимальное и максимальное значения диапазона изменения индекса.

Описание структурного типа размещается на месте имени типа в описании переменной. Иногда удобнее ввести имя вновь создаваемого структурного типа. Для этого предназначен специальный раздел программы для описания типов, который имеет следующий синтаксис:

<Раздел описания типов> ::=
{ **type** <Имя типа> = <Имя типа или его описание>;
{ <Имя типа> = <Имя типа или его описание>; *} }*

Раздел описания типов размещается в программе перед разделом описания переменных.

Обращение к элементам массива:

<Имя массива>[<Выражение типа индекса массива>] .

Пример. Программа для определения суммы элементов массива.

```
program SumArr1;
type TArr = array[1..5] of Real;
var A: TArr;
    i: Integer;
    S: Real;
begin
Write('A[1..5]: ');
i := 1;
while i <=5 do begin Read(A[i]); i := i + 1 end;
S := 0;
i := 1;
while i <= 5 do begin S := S + A[i]; i := i + 1 end;
WriteLn('S = ', S);
end.
```

3. Оператор цикла for

Особенно удобен при работе с массивами. Структура:

for i := e1 **to** e2 **do** S , где

i - параметр цикла, простая переменная ординального типа; e1, e2 - выражения, определяющие соответственно начальное и конечное значения параметра i. S - оператор, выполняемый в цикле. Типы i, e1 и e2 должны совпадать. Семантика:

```
begin
r1 := e1; r2 := e2;
if r1 <= r2 then
    begin
    i := r1; S;
    while i <> r2 do begin i := Succ(i); S end
    end
end
```

Если необходимо, чтобы параметр цикла уменьшался в ходе выполнения оператора цикла, следует применить вариант оператора **for** с **downto** вместо **to**. Его семантика:

```
begin
r1 := e1; r2 := e2;
if r1 >= r2 then
    begin
    i := r1; S;
    while i <> r2 do begin i := Pred(i); S end
    end
end
```

Пример.

```
S := 0;
for i := 1 to 5 do S := S + A[i];
```

Пример.

```
for a := False to True do
    for b := False to True do
        WriteLn(Ord(a), Ord(b), Ord(a<=b), Ord(not a or b));
```

4. Параметризация программ

Перед разделом описания типов в программе на языке Pascal можно разместить раздел определения именованных констант

const <Имя константы> = <Константное выражение>;
{ <Имя константы> = <Константное выражение>; *}*

Константное выражение - выражение соответствующего типа, которое может быть вычислено в ходе компиляции программы.

Пример.

```
const N = 10;
      M = N div 2;
```

Именованные константы можно применять наряду с обычными. Особенно они удобны в качестве параметров программы, например, для задания размеров массивов. Пример.

```
program SumArr2;
const N = 5;
type TArr = array[1..N] of Real;
var A: TArr;
    i: Integer;
    S: Real;
begin
Write('A[1..5]: ');
for i := 1 to N do Read(A[i]);
S := 0;
for i := 1 to N do S := S + A[i];
WriteLn('S = ', S);
end.
```

5. Типовые алгоритмы для работы с массивами

```
var A: array[1..N] of Integer;
```

5.1. Максимальный (минимальный) элемент массива

M[1] = A[1], M[i] = max(M[i-1], A[i]), где i = 2..N .

```
M := A[1];
for i := 2 to N do
   if A[i] > M then M := A[i];
```

5.2. Произведение элементов массива

P[1] := A[1], P[i] = P[i-1] * A[i], где i = 2..N .

```
P := 1;
for i := 1 to N do P := P * A[i];
```

5.3. Определение индекса элемента массива с заданным значением (линейный поиск)

```
Flag := False;
for i := 1 to N do
    if A[i] = Arg then
        begin
        Flag := True;
        Index := i;
        Break;
        end;
if Flag then
    WriteLn('Индекс искомого элемента: ', Index)
else
    WriteLn('В массиве нет элемента с заданным значением.');
```

Break и Continue это стандартные процедуры АЯ TPascal 7.0. Break - прекращает выполнение оператора цикла, а Continue - прекращает выполнение тела цикла и выполняет переход к следующей итерации.

6. Многомерные массивы

Матрица - двумерный массив. Описание типа:

```
type TVector = array[1..N] of Real;
     TMatrix = array[1..M] of TVector;
var mat: TMatrix;
```

Обращение:

`mat` - вся матрица размером (M строк по N элементов);
`mat[i]` - i-я строка матрицы (N элементов);
`mat[i][j]` - j-ый элемент в i-ой строке матрицы (1 элемент).

Пример. `mat[3] := mat[2];` .

Мы видим как просто манипулировать строками матрицы. К сожалению, так просто нельзя манипулировать колонками матрицы.

Предусмотрен и упрощённый способ определения многомерных массивов. Описания

```
type TMatrix = array[1..M] of
                  array[1..N] of Real;
```

и

```
type TMatrix = array[1..M, 1..N] of Real;
```

эквивалентны.

Кроме того, эквивалентны обращения `mat[i][j]` и `mat[i,j]` .

Пример. Пусть M производителей обеспечивают однородной продукцией N потребителей. Задана матрица C[i,j], i=1..M, j=1..N объёмов поставок i-го производителя j-му потребителю. Определить объёмы поставок по производителям, по потребителям и общий объем поставок.

```
program ProCons;
const M = 3;        { Количество производителей }
      N = 4;        { Количество потребителей }

type TRow = array[1..N] of Real;
     TCol = array[1..M] of Real;
     TMat = array[1..M] of TRow;

var  C:   TMat;     { Матрица поставок }
     Pro: TCol;     { Вектор поставок по производителям }
     Con: TRow;     { Вектор поставок по потребителям }
     i:   Integer;  { Индекс производителя }
     j:   Integer;  { Индекс потребителя }
     V:   Real;     { Общий объём поставок }

begin

{ Ввод данных }
WriteLn('Введите данные (', M, ' строк по ', N, ' элементов):');
for i := 1 to M do
    begin
    for j := 1 to N do Read(C[i,j]);
    ReadLn;
    end;
```

```
{ Определение вектора Pro }
for i := 1 to M do
    begin
    Pro[i] := 0;
    for j := 1 to N do Pro[i] := Pro[i] + C[i,j];
    end;

{ Определение вектора Con }
for j := 1 to N do
    begin
    Con[j] := 0;
    for i := 1 to M do Con[j] := Con[j] + C[i,j];
    end;

{ Определение общего объёма поставок }
V := 0;
for i := 1 to M do V := V + Pro[i];

{ Вывод данных }
WriteLn;
for j := 1 to N do Write(j:6);
WriteLn;
for i := 1 to M do
    begin
    Write(i:2);
    for j := 1 to N do Write(C[i,j]:6:1);
    WriteLn(Pro[i]:8:1);
    end;
Write('  ');
for j := 1 to N do Write(Con[j]:6:1);
WriteLn(V:8:1);

end.
```