10. Типы данных.

Когда решается задача, прежде всего встает проблема представления данных. А для этого нужно четко знать диапазон и структуру представляемой информации.

Турбо-Паскаль предоставляет довольно широкие возможности для представления данных сложной структуры. Эта возможность обеспечивается определением *Типа* элементов программы. Тип переменной предопределяет диапазон ее значений, а также для сложных, т.н. структурных элементов – распределение значений по составным частям.

По организации представления данных различаются два вида типов: *скалярные* и *структурированные*. Переменная скалярного типа может принимать строго одно значение из допустимой области, определенной данным типом, тогда как переменная структурированного типа представляет определенный набор значений величин одинаковых или различных типов. Все типы, которые мы рассматривали до сих пор, были скалярными – Integer, Real, Char, Boolean. Это т.н. *стандартные* типы данных. Диапазон их значений определяется самим компилятором. Кроме перечисленных типов, в Турбо-Паскале определены другие стандартные типы для представления целых и действительных величин, только они предоставляют различный диапазон значений в зависимости от вида данных или требуемой точности. Ниже приводится таблица, где перечисляются все типы с указанием количества отведенных байтов для хранения в оперативной памяти и диапазоном допустимых значений.

Тип	Длина	Диапазон допустимых значений
Целочисленные		
Byte	1 Б	0. 255
ShortInt	1 Б	–128 .127
Word	2 Б	0 65 535
Integer	2 Б	–32 768… 32 767
LongInt	4 Б	–2 147 483 648… 2 147 483 647

Действительные	Значащие цифры	Диапазон десятичной степени
Single	4 Б 7..8	- 45.. + 38
Real	6 Б 11..12	-39.. +38
Double	8 Б 15..16	-324.. +308
Extended	10 Б 19..20	-4 951.. +4 932
Comp	8 Б 19..20	$-2^{63} + 1.. +2^{63} - 1$

Кроме стандартных типов, пользователь может сам определять множество или диапазон значений для своих структур данных. В этом случае он сам определяет новый, т.н. пользовательский тип данных, который может быть как скалярным, так и структурированным. На Турбо-Паскале это можно реализовать в блоке объявления типов (в разделе описаний), который начинается с зарезервированного слова Type. Самые простые типы пользователя – *перечисляемый* и *интервальный* . Оба типа являются *скалярными* и *порядковыми.*

Скалярным называется тип, который представляет одно-единственное значение, а структурированный тип представляет либо набор значений одинакового или различного типа, либо какую-нибудь сложную структуру, как в случае файлов. В таблице 10.1 дается классификационная картина всех стандартных типов, определенных в Турбо-Паскале.

Таблица 10.1.

Скалярные типы		Структурированные
Порядковые	Непрерывный	
Integer	Real	String
Char		Array
Boolean		Set
Интервальный		Record
Перечисляемый		File
		Pointer

Коротко охарактеризуем структурированные типы:

String — последовательность символов, заключенная в апострофы. Максимальная длина — 255. В нулевом элементе содержится информация о действительной длине строки;

Array — упорядоченный набор однотипных элементов фиксированного количества. Каждый элемент массива определяется своим индексом, с помощью которого можно к нему обращаться. Массив может быть одномерным или многомерным;

Set — неупорядоченный набор однотипных элементов, объединенных по какому-либо признаку, например, множество четных или нечетных чисел, множество гласных или согласных букв и т.д.

Record — набор элементов различного или одинакового типа. Количество элементов фиксировано. Каждый компонент-элемент именуется полем;

File — позволяет работать с файлами различного типа (создавать, модифицировать файл, считывать информацию из нее и др).

Text — позволяет обрабатывать с текстовыми файлами.

Pointer — позволяет использовать дополнительную динамическую память и манипулировать данными, структура и количество которых предварительно неизвестны.

10.1. Перечисляемый тип.

Перечисляемый (Enumerated) тип данных определяет множество допустимых значений, упорядоченных перечислением символьных значений. Имена этих величин следуют правилам формирования идентификаторов. После определения типа нужно объявить переменные данного типа, которые смогут принимать значения из перечисленных в заголовке описания типа. Например:

```
Type
weekDays=(Monday, Tuesday, Wednesday, Thursday, Friday,
Saturday, Sunday);

Var workDay, holiday : weekDays;
```

Здесь мы определили новый тип weekdays. Допустимые значения для него перечислены в круглых скобках, причем набор этих значений упорядочен, т.е. первому элементу соответствует номер 0, второму — 1 и т.д. С помощью этой неявной нумерации можно обращаться к отдельным элементам. Например,

```
If(workDay>Sunday) And (workDay< Saturday)
    Then Writeln(' Сегодня выходной!');
If workDay = Sunday
    Then Writeln(' Сегодня Воскресенье!');
```

Значения перечисляемых типов нельзя считывать с клавиатуры или выводить на экран, их можно использовать только для обработки данных, массивов и т.д. Таким образом, следующие перечеркнутые операторы нелегальны:

```
~~Writeln (workday);~~
~~Readln (holiday);~~
```

В следующем примере продемонстрировано использование переменных перечисляемого типа:

```
Type drinks = (water, coffee, tea, cola, soda, milk);
    taste = (sweet, neutral, hot);
Var bacal: drinks;
    taste_for_me, taste for_ you:_ taste;

    …
Begin
    …
    bacal := cola;
            taste_for_me:= neutral;
    …
    If bacal = cola Then taste_for_you:= sweet;
    …
End.
```

В Турбо-Паскале определены три функции, позволяющие оперировать перечисляемыми данными. Эти функции:

Ord(symbol) – (сокращенное от ordinal) возвращает номер символа в последовательности перечисления. Например, значением функции Ord(tuesday) будет единица.

Pred(symbol) (сокращенное от predecessor) возвращает предыдущее значение аргумента. Например, значением Pred(wednesday) будет tuesday.

Succ(symbol) – (сокращенное от successor) возвращает следующее за аргументом значение. Например, значением Succ(wednesday) будет thursday.

Перечисляемый тип можно использовать в качестве индексов оператора For, константы варианта в операторе Case, а также для индексации массивов. Например,

```
For workday := monday to friday Do Begin ... End;
```

или

```
Case workday of
    monday : Writeln(' Понедельник трудный день недели.');
    friday : Writeln(' Приближается время пикника!!')
End;
```

10.2. Интервальный тип (тип-диапазон).

При определении нового типа можно задать какой-либо конкретный диапазон значений. Например, пусть мы хотим ввести переменную для представления числа дня месяца dayINmonth. Диапазон в данном случае не будет превышать числа 31, и, следовательно, нам не нужно описывать ее как Integer, или Word. Мы можем объявить новый тип, скажем, daysRange. На Турбо—Паскале это можно реализовать в блоке объявления типов (в разделе описаний), который начинается с зарезервированного слова Type:

```
Type daysRange=1 ..31;
Var dayINmonth : daysRange;
```

Фактически, мы определили новый скалярный тип, т.н. *интервальный,* или *тип-диапазон (Subrange).* Общий формат объявления имеет вид:

```
Type имя-типа = const1., const2;
Var имя-переменной: Имя-типа;
```

где *const1* – нижний предел допустимых значений, а *const2* – верхний предел, т.е. максимально допустимое значение. Эти константы могут быть целочисленного, символьного, либо логического типов. Приведем еще несколько примеров объявлений:

```
Type
    week_days = 1..7;
    alphabet = 'a'..'z';
    capitals = 'A'..'Z';
    hours = 0..24;
Var
    wd:week_days;
    letter:alphabet;
    caps: capitals;
    h:hours;
```

Можно создать свой перечисляемый тип и затем определить диапазон из его значений, например,

```
Type
    weekDays= (Monday , Tuesday, Wednesday, Thursday, Friday,
    Saturday, Sunday);
    drinks = (water, coffee, tea, cola, soda, milk);
Var
    workDay: Monday..Friday;
    holiday: Saturday..Sunday;
    cup: water, .tea;
```

Отметим, что нельзя определить тип-диапазон из дйствительных значений.

10.3. Преобразование типов.

Турбо-Паскаль является типизированным языком. Все операции определены для т.н. *совместимых типов.* Совместимыми считаются два типа, если:

- Это один и тот же тип;
- Один тип является диапазоном другого;
- Оба типа являются диапазонами одного типа;
- Это нетипизированный указатель и указатель любого типа.

Присваивание

переменная: = выражение;

допустимо, если типы *переменная* и *выражение* являются совместимыми, и диапазон значений типа *переменной* есть подмножество диапазона значений типа *выражения.*

В Турбо-Паскале определено т.н. *преобразование* типов. Преобразование типов может быть явным или неявным. Явное преобразование осуществляется специальными функциями, аргументы которых принадлежат к одному типу, а значения функций – другому, Например,

Функция	Тип аргумента	Тип значения
Ord	Символьный	Целочисленный
Chr	Целочисленный	Символьный
Trunc	Действительный	Целочисленный
Round	Действительный	Целочисленный

Неявное преобразование типов осуществляется в выражениях с помощью применения идентификатора имени типа, стандартного или определенного пользователем. Рассмотрим примеры таких преобразований:

```
Var i,j:Integer;
    b:Byte;
    w:Word;
    l:Boolean;
    r:Real;
    c:Char;
    s: String;
        ...
        j:=Integer('b');
        i:=Integer(b);
        c:=Char<(66);
        r:=Real(i);
        b:=Byte(i);
```

Недопустимы преобразования:

```
I:=Integer(r);
S:=String(c);
C:=Char(s);
```

А такое преобразование:

```
i:=Integer(w);
```

выдаст правильный результат только в том случае, если значение переменной *w* (типа Word) лежит в диапазоне типа Integer, хотя синтаксической ошибки не возникнет. То же самое выполняется для логических величин, например,

```
l:=Boolean(b);
```

Если *b*=0, то *l*=False, для любого другого значения логическая величина выдаст True.

10.4. Объявление типизированных констант.

Мы уже рассмотрели объявление и определение значений постоянных величин. При этом, тип величины явно не указывается, а определяется автоматически заданным значением. Можно объявить константу с явным определением типа. Такой механизм дает возможность объявить структурированные константы. Синтаксически определение типизированной константы осуществляется в блоке объявления Const и имеет следующий формат:

```
Const Идентификатор : Тип = Значение;
```

т.е. сначала определяется тип величины, а затем перечисляются соответствующие значения. Рассмотрим несколько примеров таких объявлений:

```
    Const
{ Целочисленная}
    n : Integer = 10;
    m : Integer = 20;

{Действительная }
    r : Real = 5.5;

{ Символьная }
    reply : Char = 'y';
    ch : Char = 'a';

{ Строковая }
    name : String[30] = 'String constant';
    st:String[15]='Hello darling!';

{ Типа диапазон }
    dayN: 1..7=4;
{ Перечисляемого типа }
    Type weekDays=(Monday, Tuesday, Wednesday, Thursday, Friday,
    Saturday, Sunday);
Const holiday1 : weekDays = Saturday;
```

```
holiday2 : weekDays = Sunday;
```

«Типизированная константа» – это особый вид величин, она фактически является переменной с инициируемым начальным значением. Поэтому ее свойства несколько отличаются от свойств обыкновенных констант.

Перечислим основные свойства типизированных констант (ТК):

• ТК объявляется с указанием типа и присваиванием значения одновременно.

• ТК имеет смысл скорее переменной с инициируемым начальным значением, а не постоянной величины.

• ТК инициируется только один раз, в момент входа в программный блок. Если ТК объявляется в подпрограмме (в процедуре или функции), где ее значение меняется, при повторном обращении к этому же блоку инициация значения заново не производится, а величина сохраняет свое последнее, измененное значение.

• ТК нельзя использовать в качестве границ диапазона ранга массивов, например,

```
Const n:Integer=10; m:Integer=20;
Var mass: Array[1…n,1…m] of Real;
```

• ТК может быть любого типа, скалярного или структурированного, кроме файлового типа, например,

```
fc:Text=fileConst;
```

• При определении значения ТК нельзя использовать вызов функции, например,

```
x:Real=Pi;
```

При объявлении структурированных типизированных констант значения составляющих компонентов заключаются в круглые скобки и перечисляются через запятую. Приведенные здесь примеры объявления структурированных типизированных констант станут более понятными после изучения структурированных типов данных

в последующих главах. Можно было бы распределить эти примеры по соответствующим темам, но, с другой стороны, такую расстановку легче использовать в качестве справочника при составлении собственных программ. И потом, всегда можно возвратиться к этой главе после того, как вы овладеете навыками манипулирования структурированными типами. Этот раздел можно использовать в качестве справочника.

Итак:

Типизированная константа-массив (одномерный):

```
digitInteger : Array[0..9] of Byte = (0,1,2,3, 4, 5, 6, 7, 8, 9);
digitChar : Array[0..9] of Char = ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9');

Type
weekDays = (Monday , Tuesday, Wednesday, Thursday, Friday,
Saturday, Sunday);

Const
wD:Array[weekDays] ofString[15] =
('Sunday', 'Monday', Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday');
```

Двумерный массив:

Определение единичной матрицы:

```
Const n=5;
Type matrixUnit=Array[1..n,1..n] of Byte;
Const mU: matrixUnit =
        ((1.0,0,0,0),
        (0,1,0,0,0),
        (0,0,1,0,0),
        (0,0,0,1,0),
        (0.0,0,0,1));
```

Следующий программный фрагмент:

```
Var ii,jj:Integer;
BEGIN
    For ii:=1 to n Do
    Begin For jj:=1 to n Do Write(mU[ii,jj]:5);
          Writeln;
    End;
END.
```

выдаст результат:

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

Типизированная константа-множество:

```
Const
    empty: Set of Byte=[ ];
    digits : Set of Byte = [0,1,2,3,4,5,6.7,8,9];
    maths : Set of Char. = ['+','-', '*',':', '='];
```

Следующий программный фрагмент:

```
Var ii,jj:Integer;
    st: String;
BEGIN
    st:='2+5-1=6';
    jj=0;
    For ii:=1 to Length(st) Do If st[ii] In maths Then Inc(jj);
    Writeln(' There are ',jj,' math symbols in the string.');
END.
```

выдаст результат:

There are 3 math symbols in the string.

Типизированная константа-запись:

```
Type
    colorSet=(red,yellow,black,white,green,silver);
    car=Record
            name: String;
            maxSpeed:real;
            color:colorSet;
    End;
Const

    volga : car = (name:'volga'; maxSpeed:120; color:red);
Type
    point = Record x,y:Real End;
    triangle = Record a,b,c:point End;
Const
    tr 1 : triangle =(a:(x: 10; y:10); b:(x:5; y:15); c:(x:15; y:15));
```