Структурные типы данных

Пользовательские типы данных

В Object Pascal имеется возможность создавать собственные типы данных на основе уже имеющихся, совмещая их, или комбинируя. Например, для создания упорядоченного списка однотипных данных используют массивы (**arrays**), а для объединения нескольких типов в один - записи (**records**).

Другим аспектом применения собственных типов данных, упрощающих процесс программирования и делающий его более понятным человеку, является использование **множеств** - именованного набора из нескольких возможных значений.

Создание того или иного типа данных всегда начинается с декларации, или описания нового типа данных. Делается это в заголовочной части программы или модуля и начинается с ключевого слова **type**. После того, как новый тип данных определен, можно создавать переменные нового типа - точно так же, как и любого простого.

МНОЖЕСТВА

Ограниченный неупорядоченный набор различных элементов одинакового типа, логически связанных друг с другом. Количество элементов, входящих в **множество**, может меняться в пределах от 0 до 255.

Для определения **множества** используется ключевое слово **set**:

Set of <тип элементов>

диапазон из строчных латинских символов можно определить следующим образом:

```
type Letters = set of Char;
var a: Letters;
...
a := [a..z];
```

Множества

Пример: есть **подмножество** символов от a до z:

type SmLetter = a..z;

Переменные типа SmLetter не смогут принимать значения, выходящие за пределы указанного диапазона:

var a: SmLetter;

...

a := b; // здесь все правильно, т.к. малая b входит в подмножество a..z

a := B; // ошибка! Прописная B не входит в подмножество a..z

Перечисление

Перечисление — это упорядоченный список идентификаторов. Они определяются следующим образом:

<Имя типа> = (<название значения1>, ... < названиеN>);

Пример: Радуга

type TRainbow=(Red, Orange, Yellow, Green, Azure, Blue, Violet);

Указатели

Указатели (pointers) - это такой тип переменных, которые хранят адрес в памяти компьютера, по которому расположена другая переменная. Фактически, указатель не содержит значение, а ссылается на него.

Указатели можно задать двумя принципиально разными способами.

Во-первых, можно использовать специальный тип - **Pointer**. При этом будет создан нетипизированный указатель, под который всякий раз надо будет принудительно выделять память, используя функцию **GetMem**.

Другой, как правило более предпочтительный способ, состоит в том, что сразу же создается указатель нужного типа. Делается это при помощи символа "^", предшествующего названию типа:

После того, как указатель создан, можно связать его с переменной подходящего типа, используя операцию @:

```
var P: ^integer;
x: integer;
...
P := @x;
```

Теперь к переменной x можно обращаться как непосредственно, так и через ее указатель. В случае обращения через указатель так же используют символ "^":

```
x := 10;
P^ := 10;
```

Другим вариантом использования указателя, помимо связывания с существующей переменной, является ***выделение для него памяти и дальнейшая работа как с ссылкой на некую абстрактную переменную*** (фактически, непосредственно на область в памяти, выделенную для хранения данных). В таком случае, код будет выглядеть так:

```
var P: ^integer;
...
New(P); // выделение памяти, необходимой для хранения данных типа Integer
P^ := 10; // занесение данных в выделенный блок памяти
Dispose(P); // освобождение памяти
```

Запись

Запись – это структурированный тип данных, состоящий из фиксированного числа компонентов одного или нескольких типов. Формат записи:

```
type
<имя типа> = record
<идентификатор поля>:<тип компонента>;
<идентификатор поля>:<тип компонента>
end;
```

```
var
<идентификатор, ...> : <имя типа>;
```

Пример записи:

```
type Car = record
Number : integer;          {Номер}
Marka : string[20];        {Марка автомобиля}
FIO : string[40];          {Фамилия, инициалы владельца}
Address : string[60]    {Адрес владельца}
end;
var M, V : Car;
```

Например, чтобы получить доступ к полям записи Car, надо записать:

M.Number
M.Marka
M.FIO
M.Address

Пример:

M.Number := 1678;

M.Marka := ТАЗ - 24';

M.FIO := 'Демьяшкин В.А.1';

M.Address := 'ул. Пушкина 12 - 3Г';

Составные имена можно использовать, в частности, в операторах ввода-вывода:

```
Read(M.Number, M.Marka, M.FIO, M.Address);
Write(M.Number:4, M.Marka:7, M.FI0:12, M.Address:25);
```

Допускается применение оператора присваивания и к записям в целом, если они имеют одинаковый тип. Например,

```
V := M;
```

ПРИМЕР

- Храним информацию о людях, заполнивших анкету на поступление на работу.
- Нас интересуют: ФИО человека, возраст, образование (среднее/высшее), владение компьютером, владение иностранными языками.

```
type
    TPerson = record
        Name: String;
        Age: Byte;
        Education,PC: Boolean;
        Foreign: set of TForeignLanguages;
    end;
```

Для *ФИО* вполне подходит текстовая строка типа *String*. 255 символов.

Для хранения *возраста* целесообразно выбрать тип данных *Byte* (число от 0 до 255).

Для поля "*образование*" выбран *логический* тип данных. Условимся, что True - это высшее образование, False - высшего нет (т.е. среднее).

PC - владение компьютером, здесь всё понятно.

Для хранения *иностранных языков* здесь используется *множество*.

```
type
    TForeignLanguages = (flEnglish, flGerman, flFrench);
```

Специальные типы данных

TDateTime - этот тип является, фактически, вещественным числом, данные хранятся в нем следующим образом: целая часть числа определяет дату, за которую берется количество дней, прошедших с 31 декабря 1899 года, а дробная определяет время в миллисекундах, прошедших с начала текущего дня.

Преимуществом же типа TDateTime является то, что для него предусмотрен целый набор готовых функций, позволяющих работать с датами и временем.

Функция	Описание
Now	Возвращает текущие дату и время
Date	Возвращает текущую дату (целую часть TDateTime)
Time	Возвращает текущее время (дробную часть TDateTime)
DateTimeToStr	Преобразует дату и время в строку на основе системных настроек
DateTimeToString	Копирует дату и время в указанную строковую переменную
DateToStr	Преобразует дату в строку
TimeToStr	Преобразует время в строку
FormatDateTime	Преобразует дату и время в указанный формат
StrToDateTime	Преобразует строку, содержащую написанную надлежащим способом дату и время, в переменную типа TDateTime
StrToDate	Преобразует строку в дату в формате TDateTime
StrToTime	Преобразует строку во время в формате TDateTime
DayOfWeek	Возвращает номер дня недели (от 1 до 7) для указанной даты. Учитывайте, что 1-й день недели – воскресенье
DecodeDate	Раскладывает значение типа TDateTime на 3 целых, представляющих собой год, месяц и день месяца
DecodeTime	Раскладывает значение типа TDateTime на 4 целых, представляющих собой часы, минуты, секунды и миллисекунды
EncodeDate	Объединяет 3 целых, представляющих собой год, месяц и день, в одно значение типа TDateTime
EncodeTime	Объединяет 4 целых, представляющих собой часы, минуты, секунды и миллисекунды? в одно значение типа TDateTime

Типичный пример использования функций для работы с датами, может выглядеть примерно таким образом:

```
var today, yesterday: TDateTime;
s: string;
...
today := Now();
yesterday := today - 1;
s := TateToStr(yesterday);
```

Здесь переменной s будет назначено значение, соответствующее вчерашнему дню в формате, принятому в системе (например, "10.02.2017").