Комбинированный тип данных (записи)

Подобно массиву, запись представляет собой совокупность родственных данных. Однако, в отличие от массива, запись может содержать элементы, принадлежащие различным типам. Этим обуславливаются различия в механизмах доступа к отдельным элементам. Место элемента в массиве определяется его индексом, и элементы массива можно различным образом сортировать. Что касается записей, то здесь каждый элемент имеет собственное имя, а операция сортировки для элемента записи смысла не имеет.

Запись (или комбинированный тип данных) очень хорошо подходит для объединения разнородной информации о каком-либо объекте. Это могут быть технические характеристики некоторого устройства, экономические данные предприятия или сведения о человеке. Элементы, образующие запись, называются полями.

Описание записи имеет вид:

```
Type
   a=record
   x,y:m;
   . . .
   z:n
   end;
```

Здесь *A* – имя объявляемого комбинированного типа.

RECORD и *END* – зарезервированные слова, имеющие смысл *запись* и *конец*;

X, Y, Z – имена полей;

M и *N* – типы, которым принадлежат те или иные поля.

Правила объявления типа данных RECORD

- После зарезервированного слова *RECORD* точка с запятой не ставится. Описания отдельных полей (или групп полей, принадлежащих одному типу) завершаются точкой с запятой (кроме последнего поля перед служебным словом *END*). Описания полей записи похожи на описания обычных переменных. Поля могут принадлежать любым типам – как простым, так и структурированным, как стандартным, так и определимым пользователем. Например, допустимы записи, поля которых представляют собой также записи или, например, массивы.
- Количество полей в записи фиксировано и определяется описанием записи. Имена полей в пределах записи не должны повторяться. Если в программе объявлены несколько комбинированных типов, имена полей, принадлежащих разным типам, могут совпадать; конфликта имен при этом не будет, поскольку обращение к отдельным полям производится с оказанием имени записи (об этом далее). Как и с другими идентификаторами в программах на *Delphi*, следует стремиться, чтобы имена полей соответствовали смыслу информации в том или ином поле. А информация, в свою очередь, должна соответствовать типу своего поля.

Объявление записи в программе

Организация данных о людях – один из наиболее типичных случаев применения записей. Например, если существует информация о какой-то группе людей (предположим, объединенных в производственный коллектив), то эту информацию лучше всего организовать в виде набора записей, где для каждого члена коллектива предусмотрена своя запись. При этом все записи будут принадлежать одному типу, структуру которого определяет характер данных. Вот как может выглядеть объявление подобного типа в разделе описаний программы:

```
Type
Employee=record
    ID:word;  {Идентификатор (личный номер)}
   SurName, FirstName, SecondName : string[20];   {Имя, фамилия, отчество}
    Standing: byte; {Стаж}
    Salary:real   {Зарплата}
    end;
 var
    Assistant:Employee;
    m:array[1..50] of Employee; {Массив записей}
    f:file of Employee; {файл записей}
```

Описанный выше комбинированный тип *Employee* включает шесть полей, три из них – *FirstName* (Имя), *SecondName* (Отчество) и *SurName* (Фамилия) – представляют собой строки по 20 символов каждая. Для остальных полей – *ID* (Идентификатор), *Standing* (Стаж) и *Salary* (Зарплата) – выбраны типы, также подходящие для соответствующей информации.

Обращение к полям записи

Содержимое одной из записей, принадлежащих типу *Employee*, выглядит так:

ID	*SurName*	*FirstName*	*SecondName*	*Standing*	*Salary*
14873	Петров	Иван	Кузьмич	10	15000

Доступ к отдельным полям проводится с помощью составных имен, включающих имя записи и имя поля, разделенные точкой. Вот как можно обратиться к различным полям переменной *Assistant* из примера выше:

```
Assistant.ID:=19876;
 Edit1.text:=Assistant.FirstName;
Assistant.Surname:='Петров';
 a:=Assistant.Standing;
 Assistant.Salary:=15000;
```

Иными словами, манипулировать полями записи можно так же, как переменными, с учетом типа, которому принадлежит то или иное поле.

Оператор WITH

Как указывалось выше, если в операторах используются составные имена (имя записи плюс имя поля), такие операторы получаются чересчур громоздкими. Ситуация еще больше ухудшится, если несколько подобных операторов окажутся сосредоточены в одном месте программы. В этом случае неплохо бы каким-то образом вынести имя записи, над полями которого проводятся различные действия, в некоторый заголовок. Создать такой заголовок позволяет оператор *WITH* (его еще называют оператором над записями).

Оператор *WITH* имеет вид:

With p do s;

Использованные здесь зарезервированные слова *WITH* и *DO* имеют смысл *с* и *выполнить* соответственно. Идентификатор *P* – имя переменной комбинированного типа. Идентификатор *S* – оператор (часто составной).

Обращения к шести полям записи *Assistant*, представленные выше, можно преобразовать так (предполагается, что эти операторы следуют в программе один за другим):

```
with Assistant do
begin
    ID:=14873;
    SurName:='Петров';
    FirstName:='Иван';
    SecondName:='Кузьмич';
    Standing:=10;
    Salary:=15000
end;
```

Здесь имя переменной *Assistant* употребляется всего один раз –после оператора *WITH*. Затем все действия над полями (в составном операторе) выполняются только с указанием имен полей.

Иными словами, оператор *WITH* особенно полезен, когда обрабатывается несколько полей одной переменной комбинированного типа. Однако злоупотреблять использованием оператора *WITH* также не следует, поскольку в результате часто неочевидно, к чему относится тот или иной идентификатор. Например, идентификатор *Assistant.ID* более информативен, чем просто *ID*. Это особенно верно, если составной оператор, выполнение которого инициирует оператор *WITH*, включает множество операторов.

Иерархические записи

Возможно существование записей, отдельные поля которых представляют собой также записи. Для того чтобы понять, как это реализуется, попробуем добавить в описание комбинированного типа *Employee* пару полей-записей. Пусть это будут поля, содержащие информацию об адресе служащего (почтовый код, город, улица, дом, квартира) и его дате рождения (число, месяц, год). В принципе соответствующие поля (пять полей и три поля с данными соответственно об адресе и дате рождения) можно добавить непосредственно в описание комбинированного типа *Employee*. А что если эту информацию впоследствии потребуется использовать в качестве полей еще в каких-нибудь комбинированных типах? Поэтому лучше для адреса и даты рождения создать самостоятельные комбинированные типы в разделе описаний программы, которые затем добавить как поля в тип *Employee*. Описание этих типов может выглядеть так:

```
type
  Address = record
  PostCode:1.. 999999;      {Почтовый код}
  City,Street:string[20];   {Город, улица}
  House:word;               {Дом}
  Apartment:word            {Квартира}
  end;                      {address}
 Date = record
  Day:1..31;                {Число}
  Month:1..12;              {Месяц}
  Year;1900..2012           {Год}
  end;                      {date}
 NewEmployee = record
  ID:word;                  {Идентификатор (личный номер)}
  FirstName, SecondName, SurName : string[20];   {Имя, фамилия, отчество}
  Standing: byte;           {Стаж}
  Salary:real               {Зарплата}
  BirthDate:date;           {Дата рождения}
  Habitation:address        {Место жительства}
  end;
  var
  NewAssistant: NewEmployee;
```

Составные имена, с помощью которых можно обращаться к полям вложенных записей, должны включать имя основной записи, имя вложенной записи и имя поля. Например:

NewEmployee.Address.PostCode

или

NewEmployee.Date.Day

Если имеют место несколько уровней вложения записей, составные имена могут быть очень длинными. Однако ничто не мешает в этих случаях широко применять оператор *WITH*.

Действия над записями

Действия над полями записи

Поскольку поле записи трактуется как переменная, оно может фигурировать в выражениях. (Как обратиться к полю записи, мы выяснили выше – для этого достаточно указать имя записи и имя поля.) Набор операций над полями записи соответствует операциям, допустимым для типа, которому принадлежит данное поле. Какие операции и подпрограммы применимы к тому или иному простому типу, мы уже знаем.

Действия над записями

Здесь, вероятно, уместно повторно упомянуть оператор *WITH*, который так и называется: оператор над записями.

Кроме того, как и с массивами, значения переменных, принадлежащих комбинированному типу, можно присваивать другим переменным того же типа. Например, если переменные *X* и *Y* представляют собой записи, принадлежащие одному комбинированному типу, то допустим оператор присваивания

```
x:=y;
```

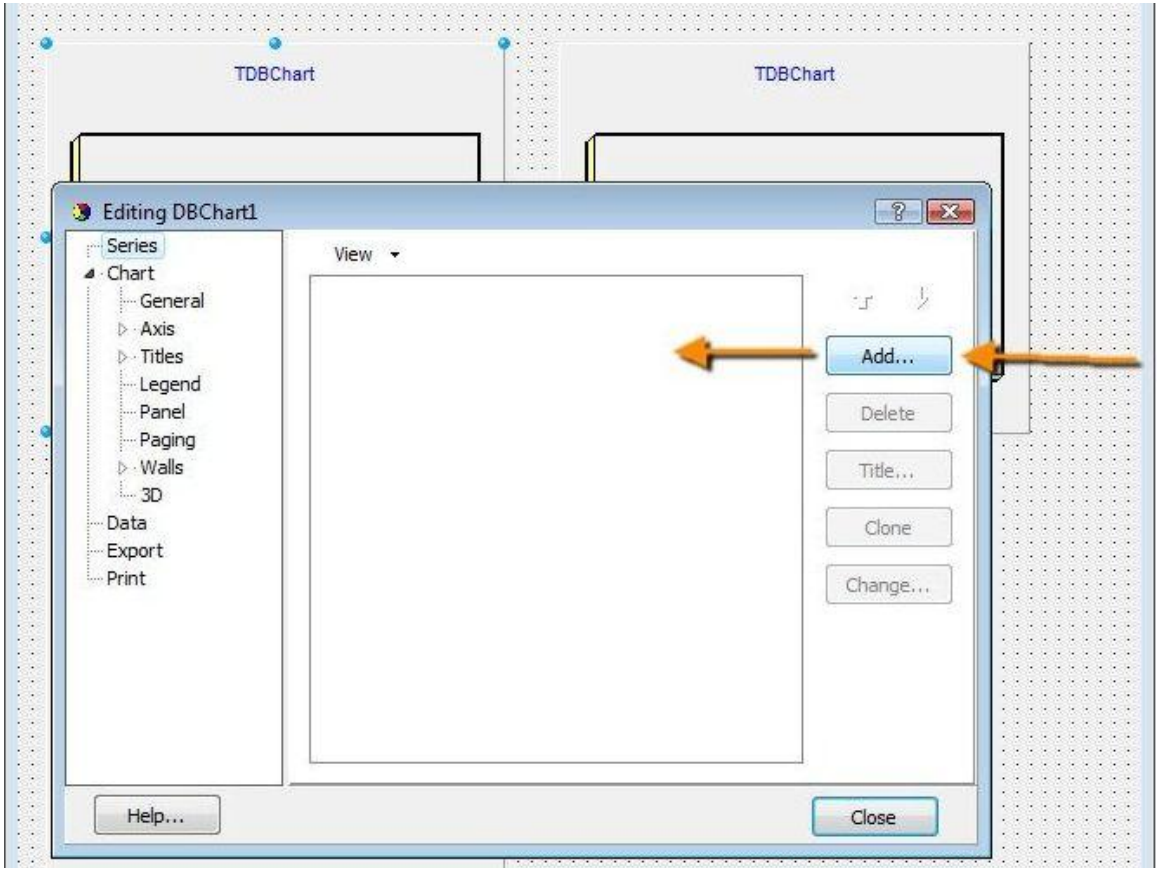
В результате значения всех полей записи *Y* окажутся присвоены полям записи *X*.

Однако нельзя использовать с записями операции сравнения. Неверно, например, было бы к записям *X* и *Y* применить оператор

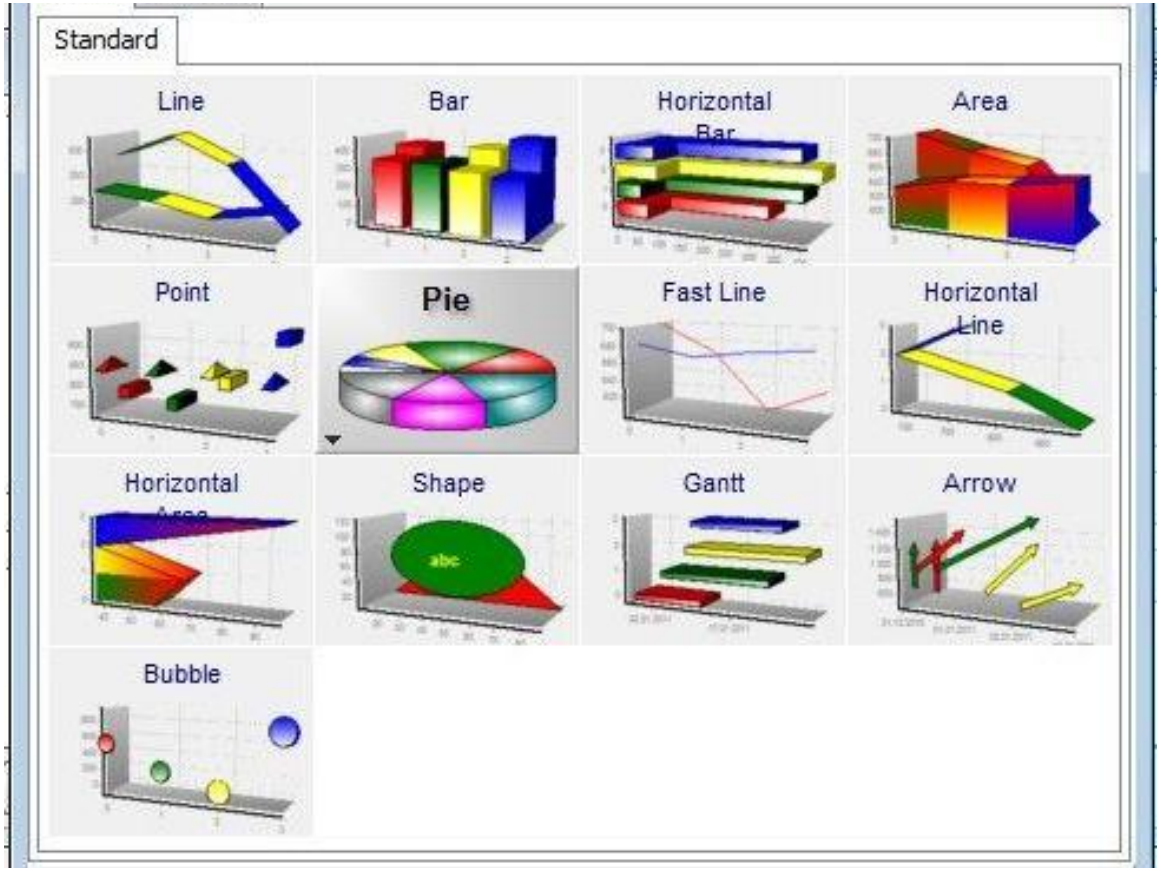
```
while x=y do...
```

Тем не менее, если такая необходимость существует, можно сравнить по очереди каждое поле одной записи с полями другой.

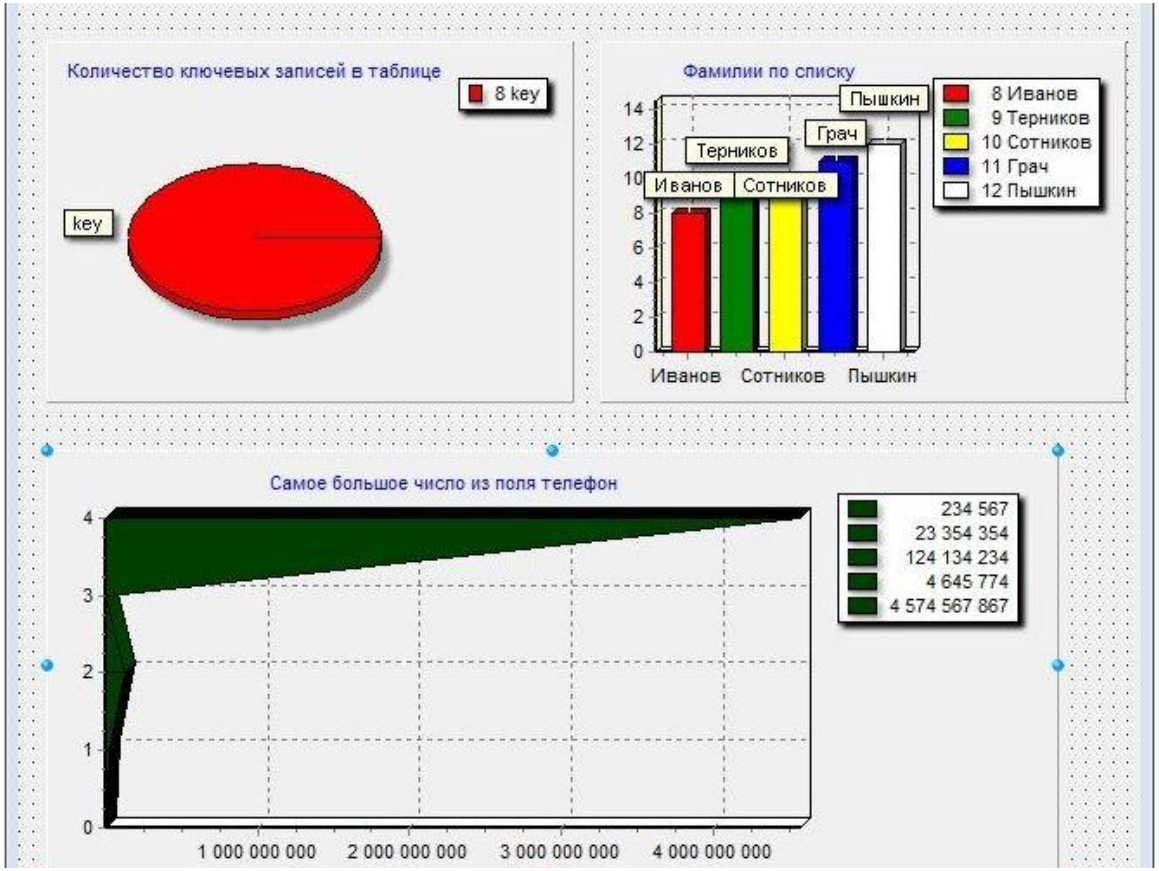
Совершенно не применимы к записям (в целом) арифметические и логические операции.



Добавление диаграмм и графиков



Выбор диаграмм и графиков



Примеры диаграмм и графиков

Построение диаграмм и графиков (компонент TChart)

```
Procedure TForm1.N1Click(Sender: Tobject);
Begin
Chart1.Title.Text.Add('Название диаграммы');
Chart1.Series[0].Add( «значение», «название», «цвет»);
End;

Procedure TForm1.N4Click(Sender: Tobject);  {ПОСТРОЕНИЕ ДИАГРАММЫ}
Begin
Chart1.Title.Text.Add('Диски');
Chart1.Series[0].Add( n1, 'Rock', clgreen);
Chart1.Series[0].Add( n2, 'Pop', clred);
Chart1.Series[0].Add( n3, 'Jazz', clblue);
End;

Procedure TForm1.N5Click(Sender: Tobject);  {ОЧИСТКА ДИАГРАММЫ}
Begin
Chart1.Title.Text.Clear;
Chart1.Series[0].Clear;
End;
```