**Ссылочный тип данных. Динамические объекты.**

**1. Природа динамических объектов и способы их реализации.**

 Все объекты, представляющие данные в программе и которые рассматривали до сих пор, были статические в том смысле, что все их параметры, размеры были известны до выполнения программы. Следовательно, ресурсы для них можно было заранее спланировать и выделить.

 Существуют задачи, для которых характерно наличие данных: - фактическое появление которых возможно, но не обязательно; - время жизни этих объектов меньше времени исполнения программы. Такие объекты называют динамическими объектами.

 Например, если нам надо выбрать из входного потока данных, совокупности данных, обладающих определенными свойствами. Встретим или нет мы такие совокупности это вопрос. Поэтому выделение ресурсов для их хранения заранее вряд ли разумно. Более того мы не знаем как велика будет такая совокупность. Затем, если собранную совокупность мы должны передать по линиям связи, например, на другую машину, то в нашей программе логично было бы ресурсы, занимаемые переданной совокупностью, освободить для других нужд. (Ресурсов не хватает всегда - это закон.)

 Для работы со статическими объектами в языках программирования используется хорошо известный механизм имен. Pascal здесь не исключение. Однако, этот механизм вряд ли нам подходит для представления и манипуляции динамическими объектами. Дело в том, что имя должно быт известно до выполнения программы - это во-первых. Во вторых, порождение всякого именованного объекта связано с выделением памяти. Раз объекты возникают динамически, то заранее мы не знаем сколько их будет. Следовательно не можем заранее выделить (породить, написать, придумать) нужное количество имен. Далее, не ясно чему соответствует в памяти имя не существующего объекта. Когда объект стал не нужен мы не можем уничтожить имя. Нет таких средств в языке. С другой стороны, уже при написании программы нам надо как-то описывать действия над динамическими объектами.

 Для решения этой проблемы в программировании был предложен механизм ссылок. Идея его состоит в том, что динамические объекты представлены не явно, а через некоторую статическую переменную, которая указывает на место расположения динамического объекта, если он существует, либо содержит специальное значение - объект отсутствует. Что значит указывает? Это значит что ее значение - "имя" динамического объекта. Это специальное имя, называемое ссылкой (саму переменную, при этом, часто называют указателем) и которое указывает где размещается, как найти объект и получить доступ к значению.

 В случае Pascal такими именем является адрес в памяти где размещается динамический объект. Каждый раз, когда порождается динамический объект ему выделяется место в памяти. Адрес начала выделенной области памяти полагается в качестве значения ссылочной переменной, представляющей динамический объект. При уничтожении динамического объекта занимаемая им память считается свободной, а соответствующая ссылочная переменная принимает специальное значение - нет объекта. Все действия над динамическими объектами в программе описываются как действия над значениями ссылочных переменных. Каждая ссылочная переменная указывает, представляет всегда только один динамический объект.

 К недостаткам такого решения можно отнести следующее. Если значение ссылочной переменной (ссылка на динамический объект) будет утеряно - этот объект будет безвозвратно утерян. Ведь его "имени" мы явно не знаем. Оно есть значение ссылочной переменной. Если несколько ссылочных переменных указывают на один и тот же объект и этот объект будет уничтожен с помощью одной из них, то все манипуляции с другими ссылочными переменными станут некорректными.

**2. Описание ссылочных переменных и их семантика**

Синтаксис задания ссылочного типа:

 <задание ссылочного типа>::= ^<имя типа>

^ - признак ссылочного типа; <имя типа> - имя стандартного либо описанного ранее типа. Это тип динамических объектов, которые может представлять переменная ссылочного типа. Надо подчеркнуть , что здесь может быть только имя типа.

 Сами переменные ссылочного типа вводятся обычным образом.

type
 массив = array [1..100] of integer;
 массивссылок= array [1..100]of ^integer;
var
 q,p:^integer;
 c:^char;
 матрица:^массив;
 чудо:массивссылок;

 Связь указателя (ссылочной переменной) с объектом графически можно проиллюстрировать так:

 Значение ссылочной переменной, соответствующее отсутствию динамического объекта - nil (зарезервированное слово).

Следует уяснить, что описание вида

 var v:^T;

лишь вводит статическую переменную (под которую отводится память). Однако, никакого объекта типа Т при этом не появляется. Для порождения объекта служит стандартная процедура new. Эта процедура имеет один фактический параметр - ссылочную переменную. В результате выполнения оператора процедуры new порождается динамический объект надлежащего типа, а ссылочной переменной-параметру присваивается ссылка на этот только что порожденный объект.

 При этом созданному динамическому объекту никакого значения не присваивается. Поэтому конструкция new(v) равносильна описанию статической переменной типа Т.

 Для работы с динамическими объектами используется переменная с указателем:

<переменная с указателем>::=<ссылочная переменная>^

здесь ссылочная переменная - это та ссылочная переменная, которая представляет в программе соответствующий динамический объект. Стрелка после имени ссылочной переменной свидетельствует о том, что речь идет о значении динамического объекта, который эта переменная представляет, а не о самой переменной.

 Например:

r^:=35; p^:=r^+p^div3; матрица^[p^+3]:=25; чудо[p^+3]^:=3

 В примере матрица^[p^+3]:=25; происходит разименование ссылочной переменной матрица (выражение вида матрица^ часто называют разименованием, а операцию ^ - операцией разименования; эта операция есть во многих языках программирования). В случае чудо[p^+3]^:=3 происходит разименования не полной ссылочной переменной чудо, а лишь частичной переменной чудо[p^+3]. Операция разименование применима только к простым переменным ссылочного типа.

**3. Действия над ссылками**

 Над значениями ссылочного типа нет операций, которые бы давали значения ссылочного типа. Над значениями ссылочного типа определены только операции присваивания и сравнения на равенство и не равенство. В операторе присваивания вида p:=e ссылочным выражением может быть: - пустая ссылка nil; -ссылочная переменная; -ссылочная функция (т.е. функция чье значение - ссылка). Значения левой и правой частей должны ссылать на объекты одного и того же типа.

 Пусть var p, g :^integer. Например, p:=q; приводит к тому что и p и q указывают на один и тот же объект, но если при этом объект, на который указывала ссылочная переменная p, будет утерян, то действия с q станут некорректными!

 Неправильно было бы написать p:=q^ т.к. слева переменная ссылочного типа, а справа значение целого типа; p^:=3.0 - опять несоответствие типов (integer и real); p^:=nil - слева переменная целого типа, справа - ссылочное значение.

 Отличия использования динамических переменных:

динамические переменный представлены через статические переменные ссылочного типа;

динамическая переменная должна порождаться явно с помощью процедуры new;

для доступа к значениям динамической переменной используется переменная с указателем.

4. Пример.

 Рассмотрим следующую задачу. Есть внешний файл, представляющий учреждение, состоящий из записей о служащих учреждения. О каждом служащем известно ФИО, пол, год рождения, должность (лаборант, техник, инженер, научный сотрудник, администратор, начотдела, замдиректора, директор), специальность (математик, программист, механик, электроник, экономист, юрист, физик, химик), номер отдела (1 .. 32), где работает служащий, характеристика. Требуется определить отдел, где работает больше всего научных сотрудников. В отделе не более 50 сотрудников.

 Структура программы может быть описана так:

program выбор (учреждение, output);
begin
 for i:= 1 to 32 do
 begin{выбрать из файла всех сотрудников из одного отдела}
 {если число нс в текущем отделе больше, чем в
 результирующем, то взять в качестве нового значения
 результирующего массива значение текущего}
 end
 {вывод номера отдела}
end

 Очевидно, что эффективность этой программы будет значительно зависеть от того как мы реализуем "взять в качестве нового значения результирующего массива значение текущего". Простая "перекачка" была бы самым неудачным решением.

program выбор (учреждение, output);
const n=50;
type служащий = record имя:packed array [1..20] of char;
 возраст: 1920..1980;
 пол:(М,Ж);
 должность:(лб,тх,инж,нс,адм,завотд,замдир,дир);
 специальность:(мтмтк,пргрммст,мхнк,элктрнк,
 экнмст,фзк,хмк,юрст);
 отдел: 1..32;
 характеристика:packed array[1..1024]of char
 end;
var отделтекущий, отделрезультат,R:^array [1..50] of служащий;
 номотдела,номрезотдела:1..32;
 сотрудник:служащий;
 числонс,maxнс,i,j:integer;
 учреждение:file of служащий;

begin maxнс:=0;
 {порождение динамических массивов}
 new(отделтекущий);new(отделрезультат);
 for i:= 1 to 32 do
 begin{выбрать из файла всех сотрудников из одного отдела}
 номотдела:=i;
 {подготовка файла учреждение к очередному просмотру}
 reset(учреждение) {установили режим чтения};
 j:=0; числонс:=0;
 while not eof(учреждение) do
 begin read(учреждение,сотрудник);j:=j+1;
 if сотрудник.отдел=i then
 begin отделтекущий^[j]:=сотрудник;
 if сотрудник.должность=нсthen числонс:=числонс+1
 end
 end;
 {если число нс в текущем отделе больше, чем в результирующем, то
 взять в качестве нового значения результирующего массива значение
 текущего}
 if числонс>maxнс then
 begin {перестановка ссылок}
 maxнс:=числонс; номрезотдела:=номотдела;
 R:=отделрезультат;
 отделрезультат:=отделтекущий;
 отделтекущий:=R
 end
 end
 {вывод номера отдела}
 write(maxнс,номрезотдела); writeln
end.

 Следует обратить внимание в этой программе на

перестановку ссылок, что избавляет нас от необходимости перекачки значений, из одного массива в другой;

нельзя путать окно файла учреждение^ и ссылочную переменную на объекты типа служащий.

**5. Уничтожение динамических объектов**

 Если в ходе вычислений какой-то динамический объект стал не нужен то его можно уничтожить и освободить занимаемую им память. Уничтожение динамического объекта происходит с помощью процедуры

 dispose (<имя ссылочной переменной>)

В результате объект, на который указывала ссылочная переменная, исчезает ,а значение переменной становится неопределенным. Сама ссылочная переменная при этом сохраняется.

 С появлением процедуры dispose в Pascal становится реальная опасность, о которой мы говорили в начале лекции - уничтожение объекта, на который указывают несколько переменных.

new(p); p^:=3;

d:=p; dispose(p);

 Здесь ошибка в том, что в результате описанных действий переменная d указывает на объект, который прекратил свое существование.