



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика»

Программирование в Delphi: записи

Методические указания к лабораторной работе № 9
по курсам «Информатика», «Алгоритмические языки
и программирование»

Автор
Е.Н. Ладоша, Д.С. Цымбалов, О.В. Яценко,
А.П. Мул, Ю.Г. Зуева

Ростов-на-Дону, 2018

Аннотация

Описывается создание и преобразование записей в Object Pascal. Целью работы ставится рационализация обработки структурированной гетерогенной информации средствами Delphi. Предназначены для студентов всех специальностей факультета «Информатика и вычислительная техника».

Автор

Доцент, к.т.н.
Ладоса Е.Н.

Старший преподаватель кафедры
«Электроника и электротехника»
Цымбалов Д.С.

Доцент, к.ф.-м.н.
Яценко О.В.

Старший преподаватель кафедры
«Прикладная математика»
Мул А.П.

Студент ДГТУ
Зуева Ю.Г.



Цель работы

Цель работы – изучить функциональность Delphi в части обработки структурированных гетерогенных данных, приобрести навыки создания, преобразования и сопоставления записей. Рассмотреть основные средства работы с записями и их практическое применение.

Запись

Запись – это тип данных, определенный пользователем. Она содержит группу связанных данных.

Неделимый элемент информации в записи называется полем. Например, в адресной книге каждая адресная карточка является записью. Таким образом, каждая запись в нашем примере состоит из полей, содержащих имя, фамилию, улицу и номер дома, город, область, индекс и номер телефона.

Определение типа записи задает имя этого типа, а также имена и типы каждого поля записи. В определении используется ключевое слово `record` (запись). Синтаксис определения типа записи в Object Pascal имеет вид

`type`

```
имя_типа_записи = record
    список_полей1: тип_данных1;
    список_полей2: тип_данных2;
    ...
    список_полейN: тип_данныхN;
end;
```

В этом синтаксисе *имя_типа_записи* должно быть любым правильным идентификатором. Список полей должен быть любым правильным идентификатором или списком идентификаторов. *Тип_данных* задает тип каждого поля в соответствующем списке полей. Последняя точка с запятой перед ключевым словом `end` необязательна.

В следующем фрагменте кода определяется тип записи `StudentInfo` и объявляется статический массив `students` элементов типа `StudentInfo`:

`const`

```
MAX_RECORDS = 25;
```

`type`

```
StudentInfo = record
    lastName: String;
    firstName: String;
    age: Integer;
```

```
end;
```

```
var
```

```
    students: array[1..MAX_RECORDS] of StudentInfo;
```

Для ссылки на конкретное поле записи нужно привести имена записи и поля, разделенные точкой. В Object Pascal символ "точка" отделяет также объект от свойства или метода. Таким образом, выражение

имя_записи.имя_поля

ссылается на заданное поле в заданной записи. В качестве имени записи можно использовать также элемент массива записей или любое выражение, ссылающееся на запись. Например, присвоить значения полям первой записи массива `students` можно с помощью следующих операторов:

```
students[1].lastName := 'Smith';
```

```
students[1].firstName := 'John';
```

```
students[1].age := 19;
```

Для доступа к полям записи можно также воспользоваться ключевым словом `with`. Например, предыдущий фрагмент кода может быть записан так:

```
with students[1] do begin
```

```
    lastName := 'Smith';
```

```
    firstName := 'John';
```

```
    age := 19;
```

```
end;
```

Таким образом, оператор `with` позволяет опускать имя записи во всем блоке. Перед каждым упоминанием элемента записи, указанной в операторе `with`, компилятор автоматически подставляет имя этой записи. При этом пространство имен, видимых в блоке, расширяется именами элементов записи, указанной в операторе `with`. Синтаксис оператора `with` имеет вид

```
with объект do begin
```

```
    [операторы];
```

```
end;
```

Оператор `with` используется для квалификации не только элементов записей, но и элементов объектов. Если в операторе `with` приведено несколько объектов или записей, то он неявно заменяется компилятором на последовательность вложенных операторов `with`. Таким образом, оператор

```
with объект1, объект2, ..., объектN do begin
```

```
    [операторы];
```

```
end;
```

эквивалентен оператору

```
with объект1 do begin
```

```
with объект2 do begin
...
    with объектN do begin
        [операторы;]
    end;
end;
end;
```

В этом синтаксисе компилятор пытается сослаться на каждую переменную внутри блока как на элемент N-го объекта; если это невозможно – то как на элемент предыдущего объекта с номером N-1 и так далее вплоть до 1-го объекта. Если и после этого имя не найдено, то только теперь компилятор ищет его среди имен локальных и глобальных переменных. То же относится и к именам объектов. Например, если *объектN* является элементом 1-го и 2-го объектов, то компилятор ссылается на него как на *объект2.объектN*.

В Object Pascal допускается создание вариантных записей, содержащих вариантную часть. Другими словами, вариантная запись содержит поля, предназначенные для разных типов данных, причем в одном экземпляре записи никогда не используются все такие поля. Рассмотрим следующий пример:

```
type
    EmployeeData = record
        firstName: String[30];
        lastName: String[30];
        birthDate: TDate;
        phoneNumber: String[14];
        case citizen: Boolean of
            True: (birthplace: String[30];
                  SSN: String[11]);
            False: (country: String[30];
                   entryPort: String[20];
                   entryDate: TDate;
                   workID: String[20]);
    end;
```

Запись типа `EmployeeData` (данные на служащего) может содержать разные поля в зависимости от значения булева поля `citizen` (гражданство). Предполагается, что если поле `citizen` истинно, значит, служащий является гражданином данной страны. В этом случае запись содержит поля, хранящие информацию о его месте рождения и номере карточки социального страхования. В противном случае запись содержит название страны, откуда он прибыл, порт

въезда и т.д. Пример заполнения записи `anEmployee` типа `EmployeeData` приведен в следующем фрагменте кода:

```
with anEmployee do begin
    firstName := 'Andrew';
    lastName := 'Stallman';
    birthDate := strToDate('12/31/70');
    phoneNumber := '(205) 925-1363';
    citizen := True;
    birthplace := 'Skokie, IL';
    SSN := '261-17-9518';
end;
```

Синтаксис вариантной записи имеет вид

```
type
    имя_типа_записи = record
        список_полей1: тип1;
        ...
        список_полейN: типN;
        case поле_переключатель: порядковый_тип
of
    список_констант1: {вариант1};
    ...
    список_константN: (вариантN);
end;
```

Первая половина объявления (вплоть до ключевого слова `case`) совпадает со стандартным объявлением типа записи. Фрагмент от ключевого слова `case` до `end` содержит вариантную часть объявления. Вариантная часть обязательно должна быть расположена после объявления остальных полей.

Необязательный параметр *поле_переключатель* представляет собой любой правильный идентификатор. Если *поле_переключатель* опущено, то следует также опустить двоеточие. Если *поле_переключатель* приведено, то оно является невариантным полем записи указанного порядкового типа. Разделенные запятыми элементы списков констант содержат константы указанного порядкового типа. Ни одно значение константы не может находиться более чем в одном списке констант. Синтаксис варианта, ассоциированного со списком констант, имеет вид

```
список_полей1: тип1;
...
список_полейN: типN;
```

Разделенные запятыми имена в списке полей должны быть правильными идентификаторами. Все поля одного списка имеют тип, указанный после двоеточия. Точка с запятой после *until* не обязательна. Типами полей не могут быть длинные строки, динамические массивы, вариантные типы, а также любые структуры, содержащие перечисленные типы. Однако тип поля может быть указателем на данные этих типов.

Для каждого экземпляра вариантной записи компилятор выделяет память, достаточную для хранения всех полей наибольшего варианта. Поэтому экземпляр записи содержит несколько вариантов, разделяющих одну и ту же область памяти. Более того, все поля экземпляра вариантной записи всегда видимы, т.е. к любому полю любого варианта можно обращаться в любой момент времени. Однако помните: если записывать данные сначала в одном варианте, а затем в другом, то существует риск повредить данные, хранящиеся в полях первого варианта. Более подробная информация о вариантных записях содержится в справочной системе Delphi.

Контрольное задание

1. Создать запись для описания дат. Запись имеет поля – число, месяц и год. Для работы с созданным типом описать функцию **NextDay**, которая увеличивает значение даты на 1 день. Предусмотреть смену месяца (например $31.10.79 + 1 \text{ день} = 1.11.79$) и года. Не учитывать наличие високосных годов.

Контрольные вопросы

1. Что такое запись? Приведите пример определения типа записи.
2. Какую структуру данных наиболее удобно описывать через тип "запись"?
3. Может ли имя поля записи совпадать с именем самой записи?
4. Обязательно ли все имена полей записи должны быть различны?
5. Может ли запись содержать одно поле?
6. Как можно заполнить значения полей записей?
7. Как можно вывести на экран значения полей записей?

Задания для самостоятельной работы

1. Фамилии и имена 25 учеников класса записаны в двух различных табли-

цах. Напечатать фамилию и имя каждого ученика на отдельной строке.

2. Названия 20-ти футбольных клубов и городов, которые они представляют, записаны в двух различных таблицах. Напечатать название и город каждого клуба на отдельной строке.

3. Даны названия 26-ти городов и стран, в которых они находятся. Среди них есть города, находящиеся в Италии. Напечатать их названия.

4. Известны данные о 16-ти сотрудниках фирмы: фамилия и отношение в воинской службе (военнообязанный или нет). Напечатать фамилии всех военнообязанных сотрудников.

5. Известны сведения о высоте над уровнем моря 15-ти горных вершин. Все значения выражены в метрах. Напечатать названия вершин, чья высота превышает 3000 м над уровнем моря.

6. Известны фамилии, адреса и телефоны 25-ти человек. Найти фамилии и адреса людей, чей телефон начинается с цифры 3. Рассмотреть два случая:

- а) телефон задан в виде семизначного числа;
- б) телефон задан в виде, аналогичном следующему: 268-50-59.

7. Известны данные о 25-ти учениках класса: фамилия, имя, отчество, адрес и домашний телефон, если он есть. Вывести на экран фамилию, имя и адрес учеников, у которых нет домашнего телефона. Рассмотреть два случая:

- а) телефон задан в виде семизначного числа;
- б) телефон задан в виде, аналогичном следующему: 268-50-59.

8. Известна информация о 30-ти клиентах пункта проката: фамилия, имя, отчество, адрес и домашний телефон. Известно также название предмета, взятого каждым из них напрокат (в виде: т — телевизор, х — холодильник и т. п.). Вывести на экран фамилию, имя и адрес клиентов, взявших напрокат телевизор.

9. Известны фамилии 25-ти человек, их семейное положение: женат (замужем) или нет, и сведения о наличии детей (есть или нет). Определить фамилии женатых (замужних) людей, имеющих детей.

10. Известны данные о 30-ти учениках: фамилия, класс и оценка по информатике. Определить фамилии учеников 9-х классов, имеющих оценку "5".

11. Известна информация о 20-ти сотрудниках фирмы: фамилия, имя, отчество, адрес и дата поступления на работу (месяц, год). Напечатать фамилию, имя, отчество и адрес сотрудников, которые на сегодняшний день проработали в фирме не менее трех лет. День месяца не учитывать (при совпадении месяца поступления и месяца сегодняшнего дня считать, что прошел полный год).

12. Известны максимальные скорости 20-ти моделей легковых автомобилей. Все значения выражены в км/ч. Напечатать названия моделей, у которых максимальная скорость превышает 180 км/ч.

13. Известна информация о 25-ти моментах времени одних и тех же суток: ча-

сы (значения от 0 до 23) и минуты (от 0 до 59). Составить программу, сравнивающую два любых момента времени по их условному порядковому номеру (определяющую, какой из моментов был в эти сутки раньше).

14. Даны даты каждого из 20-ти событий, произошедших после 1930 года: год, номер месяца и число. Составить программу, сравнивающую два любых события по времени (определяющую, какое из событий произошло позже). Событие может быть представлено:

- а) условным порядковым номером;
- б) в виде текста.

15. Известна информация о 24-х моментах времени одних и тех же суток: часы (значения от 0 до 23), минуты (от 0 до 59) и секунды (от 0 до 59). Составить программу, сравнивающую два любых момента времени (определяющую, какой из моментов был в эти сутки раньше).

16. Известны фамилии всех 30 сотрудников фирмы и их адреса. Определить, работают ли в фирме люди с одной из фамилий: Кузин, Куравлев, Кудин, Кульков или Кубиков, В случае положительного ответа напечатать их адреса.

17. Даны названия 20-ти стран и частей света, в которых они находятся. Определить, есть ли среди них страны, находящиеся в Африке или в Азии. В случае положительного ответа напечатать их названия.

18. Известны данные о 20-ти учениках класса: фамилии, имена, отчество, дата рождения (год, номер месяца и число). Определить, есть ли в классе ученики, у которых сегодня день рождения, и если да, то напечатать их имя и фамилию.

19. В записной книжке указаны фамилии и номера телефонов 30-ти человек. Составить программу:

- а) которая определяет, есть ли в записной книжке телефон некоторого человека, и, если есть, печатает номер его телефона;
- б) которая определяет, есть ли в записной книжке информация о человеке с заданным номером телефона, и, если есть, печатает фамилию этого человека.

20. Известна информация о 28-ми учениках нескольких школ, занимающихся в районном Доме творчества учащихся (фамилия, имя, адрес номер школы и класс). Фамилию, имя и адрес тех учеников, которые учатся в данной школе в старших (10—11 классах), записать в отдельный массив с элементами типа "Запись".

Список использованной литературы

1. Фаронов В.В. Delphi 3. Учебный курс. М.: «Нолидж», 1998. 400 с.
2. Галисеев Г.В. Программирование в среде Delphi 8 for .NET. М.: Издательский дом «Вильямс», 2004. 304 с.



3. *Павловска Т.А.* Паскаль. Программирование на языке высокого уровня. СПб.: Питер, 2003. 393 с.
4. *Абрамов С.А. и др.* Задачи по программированию. М.: Наука, 1988. 224 с.