



И. В. Заика

Таганрогский институт имени А. П. Чехова (филиал) Ростовского государственного экономического университета, Россия

РАБОТА С ДИНАМИЧЕСКИМИ МАССИВАМИ И ТЕКСТОВЫМИ ФАЙЛАМИ В ВИЗУАЛЬНОЙ СРЕДЕ РАЗРАБОТКИ ПРОГРАММ DELPHI

Аннотация

Ни один язык программирования не обходится без описания таких структурированных типов данных, как массивы. В статье рассматривается возможность работы с динамическими массивами и текстовыми файлами в Delphi 7. Разработана программа «Записная книга» в визуальной среде разработки программ Delphi 7, которая может быть использована при изучении объектно-ориентированного программирования в школьном курсе информатики. Представлены процедуры и функции для работы с динамическими массивами в Delphi 7. Излагаются необходимые теоретические сведения для работы с записями и текстовыми файлами в Delphi 7. Приведены листинги программного кода с подробными пояснениями. Программа «Записная книга» может быть использована для получения и закрепления теоретических знаний о динамических массивах, а также для получения практических навыков работы с динамической памятью в Delphi 7.

Ключевые слова: объектно-ориентированное программирование, Delphi, динамические массивы, текстовый файл.

DOI: 10.32517/2221-1993-2020-19-4-44-48

В школьном курсе информатики при изучении структурированных типов данных рассматриваются одномерные массивы, которые объявляются в Delphi следующим образом:

A: array [1..n] of string;

Здесь:

A — имя массива;

[1..*n*] — длина массива (диапазон);

n — количество элементов в массиве;

string — тип данных.

В данном случае все элементы массива *A* имеют строковый тип.

Работать с такими массивами возможно, если изначально задан конечный диапазон [1..*n*]. Но существуют задачи, в которых не всегда заранее известна конечная длина массива. Примером таких задач может быть составление списков, картотек, телефонных справочников, в которых изначально ничего не известно о диапазоне массива, причем в ходе программной реализации массив может менять свою текущую длину. При решении таких задач используют **динамические массивы**. Ди-

намический массив — это массив, память для которого выделяется динамически. Размерность динамического массива на момент компиляции неизвестна [9].

Процедуры и функции для работы с динамическими массивами в Delphi

В разделе описания переменных и констант при объявлении динамического массива в Delphi применяется обычное описание, но без указания размерности, например: **A: array of string**.

Для работы с динамическими массивами в Delphi существуют следующие процедуры и функции [8]:

- **SetLength(A, n)** — процедура, которая в массиве *A* устанавливает текущую длину *n* (нумерация элементов начинается с нуля);
- **Length(A)** — функция, которая возвращает количество элементов в массиве *A*;
- **SetLength(A, Length(A) + 1)** — процедура, которая добавляет один элемент к массиву *A*;
- **Low(A)** — функция, которая возвращает низший индекс массива *A*;

Контактная информация

Заика Ирина Викторовна, канд. тех. наук, доцент, доцент кафедры информатики, Таганрогский институт имени А. П. Чехова (филиал) Ростовского государственного экономического университета, Россия; *адрес:* 347936, Россия, Ростовская область, г. Таганрог, ул. Инициативная, д. 48; *e-mail:* irin-zaika@yandex.ru

I. V. Zaika

Taganrog Institute named after A. P. Chekhov (branch), Rostov State University of Economics, Russia

THE WORK WITH DYNAMIC ARRAYS AND TEXT FILES IN THE VISUAL PROGRAMMING ENVIRONMENT DELPHI

Abstract

No programming language can do without a description of structured data types such as arrays. The article discusses the possibility of working with dynamic arrays and text files in Delphi 7. The Notebook program is developed in the visual programming environment Delphi 7, which can be used to study object oriented programming in a school informatics course. The procedures and functions for working with dynamic arrays in Delphi 7 are presented. The necessary theoretical information for working with records and text files in Delphi 7 is presented. The code listings are given with detailed explanations. The Notebook program can be used to obtain and consolidate theoretical knowledge about dynamic arrays, as well as to provide practical skills in working with dynamic memory in Delphi 7.

Keywords: object oriented programming, Delphi, dynamic arrays, text file.

- *High(A)* — функция, которая возвращает высший индекс массива *A*.

Рассмотрим некоторые возможности работы с динамическими массивами на практике. Напишем **программу «Записная книга» в визуальной среде разработки программ Delphi 7** [1, 5], которая может быть использована при изучении объектно-ориентированного программирования в школьном курсе информатики.

Задание.

Создать записную книгу, в которой будут содержаться данные друзей: фамилия и имя, номер телефона, дата рождения. Фамилии и имена друзей должны быть сохранены в выпадающем списке *ComboBox*. Форму создать по образцу (рис. 1).

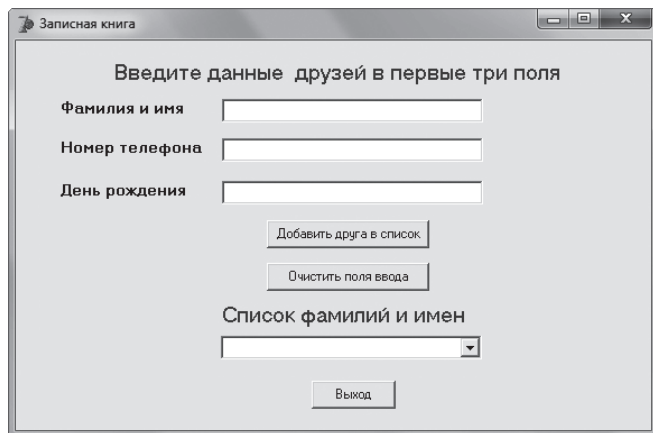


Рис. 1. Внешний вид приложения «Записная книга»

Решение.

1. Запустить Delphi 7.
2. Создать новый проект (программу). Для создания нового проекта необходимо:
 - выбрать в меню *File* команду *New, Other*;
 - в открывшемся окне *New Items* выбрать *Application*;
 - щелкнуть мышью на кнопке OK. Откроется форма *Form1*;
 - дать заголовку формы *Form1* имя «Записная книга». Для этого в меню *View* открыть окно *Object Inspector*, во вкладке *Properties* в свойстве *Caption* заменить имя заголовка *Form1* на новое имя «Записная книга».
3. В меню *View* открыть панель инструментов *Toolbars*, выбрать вкладку *Standard*.
На форме разместить:
 - три компоненты *Edit* для ввода данных друзей:
 - *Edit1* — фамилия и имя;
 - *Edit2* — номер телефона;
 - *Edit3* — дата рождения;
 - три кнопки *Button* с надписями:
 - *Button1* — «Добавить друга в список»;
 - *Button2* — «Очистить поля ввода»;
 - *Button3* — «Выход»;
 - один компонент *ComboBox1* для формирования выпадающего списка фамилий и имен;
 - пять компонент *Label*; каждую из пяти компонент переименовать по образцу (рис. 1), например:
 - *Label1* — «Введите данные друзей в первые три поля»;

- *Label2* — «Фамилия и имя» и т. д.
4. Сохранить проект. Для этого:
 - открыть меню *File*, выбрать в нем команду *Save As*;
 - дать имена: модулю (unit) — *Kniga* и проекту (project) — *ZapishnayaKniga*.

5. После того как проект *ZapishnayaKniga* сохранен, необходимо открыть редактор для описания собственного программного кода. При нажатии на клавиатуре клавиши F12 откроется редактор кода модуля *Kniga*.

Задание динамического массива в разделе описания глобальных переменных

Чтобы задать переменные, в которых будут храниться данные друзей, организуется глобальная запись — **record** (структурированный комбинированный тип данных, состоящий из фиксированного числа полей разного типа [9]). Объявление глобальной записи — *ZKniga* представлено в листинге 1.

```
{1} type ZKniga=record
{2} NomerTel: string[15];
{3} FamilImya: string[30];
{4} Data: string[10];
{5} end;
{6} var Form1: TForm1;
{7} massiv: array of ZKniga;
```

Листинг 1. Объявление динамического массива в разделе описания переменных

Рассмотрим подробно программный код, представленный в листинге 1.

Строки 2–4: запись *ZKniga* имеет три поля — *NomerTel*, *FamilImya*, *Data*, в которых будут храниться строковые типы данных: номер телефона, фамилия и имя, день рождения друга.

В строке 7 происходит объявление динамического массива записи *ZKniga*, где *massiv* — имя динамического массива.

Работа с компонентом ComboBox. Формирование выпадающего списка

Для выпадающего списка *ComboBox1*, который будет состоять из фамилий и имен друзей, организуется процедура *ObnovlenieSpiska* (см. листинг 2).

```
{1} implementation
{2} {$R *.dfm}
{3} procedure ObnovlenieSpiska;
{4} var k: integer;
{5} begin
{6} Form1.ComboBox1.Items.Clear;
{7} for k:=0 to High(massiv) do
{8} Form1.ComboBox1.Items.Add(massiv[k].
    FamilImya);
{9} end;
```

Листинг 2. Процедура *ObnovlenieSpiska*

Рассмотрим процедуру *ObnovlenieSpiska* подробно. Процедура пользователя *ObnovlenieSpiska* задается вручную в редакторе кода модуля *Kniga* после строки 2 директивы *{\$R *.dfm}*.

Предварительно в строке 6 очищается поле компонента *ComboBox1*.

Строки 7–8 содержат основной цикл. В строке 8 организуется вывод фамилии и имени из динамического массива *massiv* в поле компонента *ComboBox1*. Таким образом, происходит обновление выпадающего списка *ComboBox1*, состоящего из фамилий и имен друзей.

Чтобы перейти из редактора кода к конструктору формы, необходимо еще раз нажать на клавиатуре кнопку F12. Откроется форма «Записная книга» (см. рис. 1).

Для кнопки *Button1* («Добавить друга в список») создается обработчик события *OnClick*. После двойного щелчка пользователя на кнопке *Button1* среда Delphi автоматически сгенерирует процедуру с именем *TForm1.Button1Click*.

Процедура *TForm1.Button1Click* обработчика события для кнопки *Button1* представлена в листинге 3.

```
{1} procedure TForm1.Button1Click(Sender:
      TObject);
{2} var k:integer;
{3} begin
{4} if Edit1.Text='' then begin
{5} ShowMessage('Введите фамилию и имя
      друга!');
{6} Exit;
{7} end;
{8} if Edit2.Text='' then begin
{9} ShowMessage('Введите телефон друга!');
{10} Exit;
{11} end;
{12} if Edit3.Text='' then begin
{13} ShowMessage('Введите день рождения
      друга!');
{14} Exit;
{15} end;
{16} if length(massiv)>0 then
{17} for k:=Low(massiv) to High(massiv) do
{18} if massiv[k].NomerTel=Edit2.Text then
      begin
{19} ShowMessage('Номер уже существует,
      введите новый');
{20} Exit;
{21} end;
{22} SetLength(massiv, Length(massiv)+1);
{23} massiv[High(massiv)].Familiya:=
      Edit1.Text;
{24} massiv[High(massiv)].NomerTel:=Edit2.
      Text;
{25} massiv[High(massiv)].Data:=Edit3.Text;
{26} ObnovlenieSpiska;
{27} end;
```

Листинг 3. Процедура *TForm1.Button1Click*

Рассмотрим подробно код процедуры *TForm1.Button1Click*.

В строках 4, 8 и 12 в начале организуется проверка полей *Edit1*, *Edit2* и *Edit3*. Если, например, поле *Edit1* не заполнено и пользователь преждевременно нажмет на кнопку *Button1*, в строке 5 процедура *ShowMessage* покажет сообщение: «Введите фамилию и имя друга!»

В строках 6, 10, 14 с помощью директивы *Exit* происходит завершение процедуры *ShowMessage*.

Аналогично в строках 9 и 13 процедура *ShowMessage* выводит сообщения «Введите телефон друга!» и «Введите день рождения друга!» в случае, если в соответствующих полях *Edit2* или *Edit3* отсутствуют данные.

Чтобы пользователь не смог в массив сохранить дважды номер телефона, в строках 16–21 организуется проверка. Если длина динамического массива отлична от нуля (строка 16), то в строках 17–21 организуется цикл. В строке 18 происходит проверка каждого элемента динамического массива *massiv*. Например, в случае, если в поле *Edit2* будет повторно введен номер телефона, то в строке 19 процедура *ShowMessage* выведет сообщение: «Номер уже существует, введите новый!» Далее в строке 20 директива *Exit* производит досрочное завершение процедуры *ShowMessage*, если такой номер телефона уже существует.

В строке 22 процедура *SetLength(massiv, Length(massiv)+1)* добавляет новый элемент к динамическому массиву *massiv*.

После чего в строках 23–25 новые данные друзей (фамилия и имя, номер телефона, день рождения) из соответствующих полей *Edit1*, *Edit2* и *Edit3* сохраняются в динамический массив *massiv*.

В строке 26 организуется вызов процедуры *ObnovlenieSpiska*, которая добавляет фамилию и имя в поле выпадающего списка *ComboBox1*.

После сохранения проекта (программы) *ZapishnayaKniga* и его компиляции (компиляция происходит при нажатии на клавишу F9) необходимо ввести данные друга в первые три поля формы «Записная книга». При нажатии на кнопку *Button1* («Добавить друга в список») программа сначала сохранит введенные данные в динамический массив *massiv*, а затем с помощью вызова процедуры *ObnovlenieSpiska* добавит фамилию и имя из динамического массива в список *ComboBox1*.

После открытия выпадающего списка можно убедиться, что фамилия и имя отображаются в поле компонента *ComboBox1* (рис. 2). Чтобы завершить компиляцию программы *ZapishnayaKniga*, необходимо закрыть форму «Записная книга», нажав на крестик в правом верхнем углу формы.

Для кнопки *Button2* («Очистить поля ввода») создается обработчик события *OnClick*. После двойного щелчка пользователя на кнопке *Button2* среда Delphi автоматически сгенерирует процедуру с именем *TForm1.Button2Click*, она представлена в листинге 4.

Рис. 2. Открытие выпадающего списка *ComboBox1*

```
{1} procedure TForm1.Button2Click(Sender:
    TObject);
{2} begin
{3} Edit1.Clear;
{4} Edit2.Clear;
{5} Edit3.Clear;
{6} end;
```

Листинг 4. Процедура *TForm1.Button2Click*

В строках 3–5 процедуры *TForm1.Button2Click* происходит очищение полей *Edit1*, *Edit2*, *Edit3*.

Далее необходимо организовать процедуры, которые все данные динамического массива записывают в текстовый файл и считывают из текстового файла. В противном случае, если такие процедуры отсутствуют, после закрытия программы *ZapishnayaKniga* все ранее введенные данные будут утеряны.

Процедуры для работы с текстовыми файлами в Delphi

В Delphi под файлом понимается область внешней памяти компьютера, которая имеет имя [2, 3]. Объявление файловых переменных происходит в разделе объявления переменных:

```
var fyle: File of <тип>
```

Здесь *fyle* — имя файловой переменной. После служебного слова *of* указывается тип данных файла.

Для работы с файлами в Object Pascal (Delphi) существуют следующие процедуры [7]:

- *AssignFile(fyle, 'filetext.txt')* — связывает переменную *fyle* с именем физического файла *filetext.txt*;
- *Reset(fyle)* — открывает файл для чтения;
- *Rewrite(fyle)* — создает и открывает файл для записи;
- *Read(fyle, a)* — считывает все данные *a* из файла *fyle*;
- *Write(fyle, a)* — записывает все данные *a* в файл *fyle*;
- *CloseFile(fyle)* — процедура закрытия файла *fyle*.

Заранее с помощью программы Блокнот создадим текстовый файл *filetext.txt* и сохраним его на любом свободном диске, например на диске d.

Для кнопки *Button3* («Выход») создается обработчик события *OnClick*. После двойного щелчка пользователя кнопкой мыши на кнопке *Button3* среда Delphi автоматически сгенерирует процедуру с именем *TForm1.Button3Click*. Процедура *TForm1.Button3Click* организует запись всех данных из динамического массива *massiv* в файл с адресом *d:filetext.txt*. Здесь: *d* — имя физического диска, *filetext* — имя файла, *txt* — расширение текстового файла.

Процедура *TForm1.Button3Click* обработчика события для кнопки *Button3* представлена в листинге 5.

Рассмотрим подробно код процедуры *TForm1.Button3Click*.

В строке 5 процедура *AssignFile(fyle, 'd:filetext.txt')* связывает имя файловой переменной *fyle* с текстовым файлом *d:filetext.txt*.

Процедура *Rewrite(fyle)* в строке 6 открывает файл для записи.

Строки 7–8 содержат цикл.

```
{1} procedure TForm1.Button3Click(Sender:
    TObject);
{2} var fyle:File of ZKniga; k:integer;
{3} begin
{4} try
{5} AssignFile(fyle, 'd:filetext.txt');
{6} Rewrite(fyle);
{7} for k:=Low(massiv) to High(massiv) do
{8} Write(fyle, massiv[k]);
{9} finally
{10} CloseFile(fyle);
{11} end;
{12} close;
{13} end;
```

Листинг 5. Процедура *TForm1.Button3Click*

В строке 8 команда *Write(fyle, massiv[k])* производит запись каждого элемента динамического массива *massiv[k]* в файл *fyle*.

После записи всех данных динамического массива в строках 9–10 процедура *CloseFile(fyle)* закрывает файл *fyle*.

В строке 12 метод *close* завершает работу программы *ZapishnayaKniga*.

Таким образом, процедура *TForm1.Button3Click* осуществляет запись всех данных из динамического массива *massiv* в текстовый файл *filetext.txt*.

Теперь необходимо организовать процедуру, которая при каждой компиляции программы *ZapishnayaKniga* будет отображать все ранее сохраненные данные друзей из файла *filetext.txt* в поле компонента *ComboBox1*.

Для формы «Записная книга» создадим обработчик события *onCreate*. После двойного щелчка пользователя кнопкой мыши по форме среда Delphi автоматически сгенерирует процедуру с именем *TForm1.FormCreate*. Эта процедура представлена в листинге 6.

Рассмотрим подробно код процедуры *TForm1.FormCreate*.

В строке 5 процедура *AssignFile(fyle, 'd:filetext.txt')* связывает имя файловой переменной *fyle* с текстовым файлом *d:filetext.txt*.

Процедура *Reset(fyle)* в строке 6 открывает файл для чтения.

Строки 7–10 содержат основной цикл программы. Цикл *while* работает до тех пор, пока не будет достигнут конец файла *fyle*. В строке 8 процедура *SetLength(massiv, Length(massiv)+1)* добавляет новый элемент к динамическому массиву *massiv*. В строке 9 команда *Read(fyle, massiv[High(massiv)])* считывает все данные динамического массива *massiv[High(massiv)]* из файла *fyle*.

Процедура *CloseFile(fyle)* в строках 11–12 закрывает файл *fyle*.

После вызова процедуры с именем *ObnovlenieSpiska* в строке 14 происходит добавление фамилии и имени в поле компонента *ComboBox1*. Теперь каждый раз при новой компиляции программы *ZapishnayaKniga* с помощью процедуры *TForm1.FormCreate* данные из файла *fyle* будут отображаться в поле компонента *ComboBox1*.

После сохранения проекта (программы) *ZapishnayaKniga* и его компиляции можно ввести данные всех


```

{1} procedure TForm1.FormCreate(Sender:
    TObject);
{2} var fyle:File of Zkniga;
{3} begin
{4} try
    {связываем имя файловой переменной fyle
    с текстовым файлом filetext.txt}
{5} AssignFile(fyle, 'd:filetext.txt');
    {открытие файла для чтения}
{6} Reset(fyle);
    {считываем все данные массива из файла fyle}
{7} while not Eof(fyle) do begin
    {ввод нового элемента массива}
{8} SetLength(massiv, Length(massiv)+1);
    {считываем все данные массива из файла fyle}
{9} Read(fyle, massiv[High(massiv)]);
{10} end;
{11} finally
{12} CloseFile(fyle);
{13} end;
    {обновление списка фамилий и имен
    в ComboBox1}
{14} ObnovlenieSpiska;
{15} end;

```

Листинг 6. Процедура *TForm1.FormCreate*

друзей в первые три поля формы «Записная книга». В выпадающий список фамилии и имена должны загружаться в *ComboBox1*.

Чтобы завершить компиляцию программы *ZapishnayaKniga*, необходимо закрыть форму «Записная книга», нажав на кнопку *Button3* («Выход»).

Для того чтобы все данные друзей отображались на форме «Записная книга», в полях ввода записей необходимо организовать для компоненты *ComboBox1* процедуру *TForm1.ComboBox1Change* обработчика события *OnChange*. Эта процедура представлена в листинге 7.

```

{1} procedure TForm1.ComboBox1Change(Sender:
    TObject);
{2} begin
{3} Edit1.Text:=massiv[ComboBox1.ItemIndex].
    FamilImya;
{4} Edit2.Text:=massiv[ComboBox1.ItemIndex].
    NomerTel;
{5} Edit3.Text:=massiv[ComboBox1.ItemIndex].
    Data;
{6} end;

```

Листинг 7. Процедура *TForm1.ComboBox1Change*

Рассмотрим подробно код процедуры *TForm1.ComboBox1Change*.

В строках 3–5 в каждую из трех компонент *Edit* присваиваются данные из динамического массива *massiv*. Каждый раз при выборе любой фамилии в выпадающем списке *ComboBox1* отобразятся записи с данными друзей в соответствующих полях: фамилия и имя, номер телефона, день рождения. Например, при выборе в списке *ComboBox1* «Иванова Ирина» в полях «Фамилия и имя», «Номер телефона», «День рождения» отобразятся соответствующие данные (рис. 3).

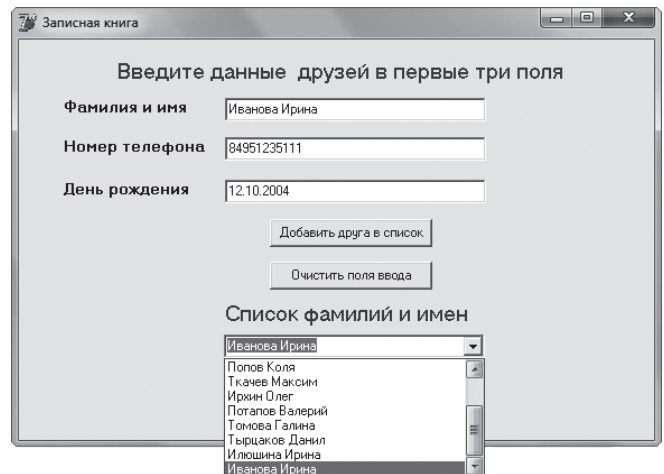


Рис. 3. Выбор фамилии и имени из выпадающего списка *ComboBox1*

Цель разработки программы «Записная книга» — расширить навыки учащихся при изучении динамических массивов и их реализации на персональном компьютере, углубить знания по предмету «Информатика», реализовать интерес к предмету.

Разработка прикладного приложения «Записная книга» поможет начинающему пользователю разобраться с особенностями работы с динамическим массивом и текстовыми файлами в Delphi [5–7, 10].

Визуализация данных находит применение в самых разных сферах человеческой деятельности. Использование программирования позволяет развивать творческие, алгоритмические, логические способности школьников.

Список использованных источников

1. Архангельский А. Я. Программирование в Delphi для Windows. Версии 2006, 2007, Turbo Delphi. М.: Бинум-Пресс, 2013. 1240 с.
2. Белов В. В., Чистякова В. И. Программирование в Delphi: процедурное, объектно-ориентированное, визуальное: учебное пособие для вузов. М.: РиС, 2014. 241 с.
3. Гниденко И. Г., Павлов Ф. Ф., Федоров Д. Ю. Технология разработки программного обеспечения: учебное пособие для СПО. М.: Издательство Юрайт, 2018. 235 с.
4. Зыков С. В. Программирование. Объектно-ориентированный подход: учебник и практикум для академического бакалавриата. М.: Издательство Юрайт, 2018. 155 с.
5. Комлев Н. Ю. Объектно Ориентированное Программирование. Хорошая книга для Хороших Людей. М.: Солон-Пресс, 2014. 770 с.
6. Лесневский А. С. Объектно-ориентированное программирование для начинающих. М.: БИНОМ. Лаборатория знаний, 2010. 232 с.
7. Нагаева И. А., Кузнецов И. А. Программирование: Delphi: учебное пособие для среднего профессионального образования / под ред. И. А. Нагаевой. М.: Издательство Юрайт, 2019. 302 с.
8. Осипов Д. Delphi. Профессиональное программирование. СПб.: Символ-плюс, 2015. 1056 с.
9. Трофимов В. В., Павловская Т. А. Алгоритмизация и программирование: учебник для академического бакалавриата / под ред. В. В. Трофимова. М.: Издательство Юрайт, 2018. 137 с.
10. Тузовский А. Ф. Объектно-ориентированное программирование: учебное пособие для прикладного бакалавриата. М.: Издательство Юрайт, 2018. 206 с.