Глава 6. Процедуры и функции

Часто, работая над программой, программист замечает, что некоторая последовательность инструкций встречается в разных частях программы несколько раз. Например, в листинге 6.1 приведена программа пересчета веса из фунтов в килограммы. Обратите внимание, что инструкции, обеспечивающие ввод исходных данных из полей редактирования, расчет и вывод результата (в листинге они выделены фоном), есть как в процедуре обработки события на кнопке **Вычислить,** так и в процедуре обработки события OnKeyPress В поле Editl.

Листинг 6.1. Пересчет веса из фунтов в килограммы

```
unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants,

Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type

TForm1 = class(TForm)

Label1: TLabel; // пояснительный текст

Edit1: TEdit; // поле ввода веса в фунтах

Button1: TButton; // кнопка Вычислить

Label2: TLabel; // поле вывода результата

procedure ButtonlClick(Sender: TObject);

procedure EditlKeyPress(Sender: TObject;

var Key: Char); private

{ Private declarations } public

{ Public declarations }

end;

var

Form1: TForm1 ;
```

```
implementation

{$R *.dfm}

// щелчок на кнопке Вычислить

procedure TForml.Button1Click(Sender: TObject);

var

f : real; // вес в фунтах

kg : real; // вес в килограммах

begin

f := StrToFloat(Edit1.Text);

kg := f; * 0.4059;

Label2.Caption := Edit1.Text + ' ф. — это ' +

FloatToStrF(kg, ffGeneral, 4, 2} + 'кг.'; end;

// нажатие клавиши в поле ввода исходных данных

procedure TForml.Edit1KeyPress(Sender: TObject; var Key: Char);

var

f : real; // вес в фунтах kg : real; // вес в килограммах

begin

if Key = Char(VK_RETURN) then

begin

f: = . StrToFloat(Editl.Text) ;

kg := f * 0.4059;

Label2.Caption := Editl.Text + ' ф. - это ' +

FloatToStrF(kg, ffGeneral, 4, 2) + 'кг.'[1].;

end;

end;
```

end.

Можно избежать дублирования кода в программе. Для этого надо оформить инструкции, которые встречаются в программе несколько раз, как подпрограмму, и заменить инструкции, оформленные в виде подпрограммы, инструкцией вызова подпрограммы.

В листинге 6.2 приведена программа пересчета веса из фунтов в килограммы, в которой ввод исходных данных, вычисления и вывод результата объединены в подпрограмму, реализованную как функция.

Листинг 6.2. Пересчет веса из фунтов в килограммы (использование процедуры)

unit Onit1; **interface**

uses

Windows, Messages, SysUtils, Variants,

Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type

TForm1= class(TForm)

Label1: TLabel; // пояснительный текст

Edit1: TEdit; // поле ввода веса в фунтах

Button1: TButton; // кнопка Вычислить

Label2: TLabel; // поле вывела результата

procedure Button1Click(Sender: TObject);

procedure EditlKeyPress(Sender: TObject;

var Key: Char); private

{ Private declarations } public

{ Public declarations } end;

var

Form1: TForm1;

implementation

```
{$R *.dfm}

// процедура программиста

procedure FuntToKg;

var

f : real; // вес в фунтах

kg : real; // вес в килограммах

begin

f := StrToFloat(Form1.Edit1.Text);

kg := f * 0.4059;

Forml.Label2.Caption := Forml.Edit1.Text + ' ф. — это ' +

FloatToStrF(kg, ffGeneral, 4, 2) + 'кг.';

end;

// щелчок на кнопке Вычислить

procedure TForml.ButtonlClick(Sender: TObject);

begin

FuntToKg; // вызов процедуры FuntToKg end;

// нажатие клавиши в поле ввода исходных данных

procedure TForm1.EditlKeyPress(Sender: TObject;

var Key: Char);

begin

if Key = Char(VK_RETURN)

then FuntToKg; // вызов процедуры FuntToKg end;

end.
```

Преимущества использования подпрограмм очевидны. Во-первых, в программе нет дублирования кода, что сокращает трудоемкость создания программы, делает более удобным процесс отладки и внесения изменений. Представьте, что нужно изменить пояснительный текст,

выводимый программой пересчета веса из фунтов в килограммы. В программе, не использующей подпрограмму, нужно просмотреть весь текст и сделать необходимые изменения. Если программа использует подпрограмму, то изменения надо внести только в текст подпрограммы. Во-вторых, значительно повышается надежность программы. Следует обратить внимание, что подпрограммы используют не только тогда, когда нужно избежать дублирования кода. Удобно большую задачу разделить на несколько подзадач и оформить каждую задачу как подпрограмму. В этом случае значительно улучшается "читаемость" программы и, как следствие, существенно облегчается процесс отладки.

Подпрограмма — это небольшая программа, которая решает часть общей задачи. В языке Delphi есть два вида подпрограмм — процедура и функция.

У каждой подпрограммы есть имя, которое используется в программе для вызова подпрограммы (процедуры).

Отличие функции от процедуры состоит в том, что с именем функции связано значение, поэтому функцию можно использовать в качестве операнда выражения, например, инструкции присваивания.

Как правило, подпрограмма имеет параметры. Различают формальные и фактические параметры.

Параметры, которые указываются в объявлении функции, называются формальными. Параметры, которые указываются в инструкции вызова процедуры, называются фактическими.

Параметры используются:

- для передачи данных в подпрограмму;
- для получения из результата подпрограммы.

В общем случае в качестве фактического параметра процедуры можно использовать выражение, тип которого должен совпадать с типом соответствующего формального параметра.

Функция

Функция — это подпрограмма, т. е. последовательность инструкций, имеющая имя.

Процесс перехода к инструкциям функции называется вызовом функции или обращением к функции. Процесс перехода от инструкций функции к инструкциям программы, вызвавшей функцию, называется возвратом из функции.

В общем виде инструкция обращения к функции выглядит так:

Переменная := Функция (Параметры) ;

где:

- переменная — имя переменной, которой надо присвоить значение, вычисляемое функцией;
- Функция — имя функции, значение которой надо присвоить переменной;
- Параметры — список формальных параметров, которые применяются для вычисления значения функции. В качестве параметров обычно используют переменные или константы.

Следует обратить внимание на то, что:

- каждая функция возвращает значение определенного типа, поэтому тип переменной, которой присваивается значение функции, должен соответствовать типу функции;
- тип и количество параметров для каждой конкретной функции строго определены.

Объявление функции

Объявление функции в общем виде выглядит так:

function Имя (параметр1 : тип1, ..., параметрК : типК) : Тип; **var**

// здесь объявления локальных переменных **begin**

// здесь инструкции функции

Имя := Выражение; **end;**

где:

- function — зарезервированное слово языка Delphi, обозначающее, что далее следуют инструкции, реализующие функцию программиста;
- имя — имя функции. Используется для перехода из программы к инструкциям функции;
- параметр — это переменная, значение которой используется для вычисления значения функции. Отличие параметра от обычной переменной состоит в том, что он объявляется не в разделе объявления переменных, который начинается словом var, а в заголовке функции. Конкретное значение параметр получает во время работы программы в результате вызова функции из основной программы;
- тип — тип значения, которое функция возвращает в вызвавшую ее программу.

Следует обратить внимание, что последовательность инструкций, реализующих функцию, завершается инструкцией, которая присваивает значение имени функции. Тип выражения, определяющего значение функции, должен совпадать с типом функции, указанным в ее объявлении.

В качестве примера в листинге 6.3 приведены функции isint и isFioat. Функция isint проверяет, является ли символ, соответствующий клавише, нажатой во время ввода целого числа в поле редактирования, допустимым. Предполагается, что допустимыми являются цифры, клавиши <Enter> и <Backspace>. Функция IsFloat решает аналогичную задачу, но для дробного числа. У функции IsFloat два параметра: код нажатой клавиши и строка символов, которая уже введена в поле редактирования.

Листинг 6.3. Примеры функций

```
// проверяет, является ли символ допустимым

// во время ввода целого числа

function Islnt(ch : char) : Boolean;

begin

if (ch >= '0'} and (ch <= '9') // цифры

or (ch = 113) // клавиша <Enter>

or (ch = #8) // клавиша <Backspace>

then Islnt := True // символ допустим

else Islnt := False; // недопустимый символ

end;

// проверяет, является ли символ допустимым

// во время ввода дробного числа

function IsFloat(ch : char; st: string) : Boolean;

begin

if (ch >= '0') and (ch <= '9') // цифры

or (ch = #13) // клавиша <Enter>

or (ch = #8) // клавиша <Backspace>
```

```
then

begin

IsFloat := True; // символ верный

Exit; // выход из функции

end;

case ch of

'-': if Length(st) = 0

then IsFloat := True; ',':

if (Pos(',',st) = 0)

and (st[Length(st)]'>= '0') and (st[Length(st)] <= '9')

then // разделитель можно ввести только после цифры // и если он еще не введен

IsFloat := True; else // остальные символы запрещены

IsFloat := False;

end;

end;
```

Использование функции

Если вы собираетесь использовать в программе свою функцию, то в простейшем случае ее объявление следует поместить в текст программы, перед подпрограммой, которая применяет эту функцию.

Рис. 6.1. Окно программы **Поездка на дачу**

Следующая программа (ее текст приведен в листинге 6.4, а вид диалогового окна на рис. 6.1) вычисляет стоимость поездки на дачу. Исходными данными для программы являются: расстояние, цена одного литра бензина и потребление бензина на 100 км пути. Для ввода исходных данных применяются поля Edit1, Edit2 и Edit3. Функции обработки события OnKeyPress

используют функцию IsFioat для фильтрации вводимых в эти поля символов, во время работы программы в полях ввода отображаются только допустимые символы.

Листинг 6.4. Пример использования функций программиста

```
unit fazenda_;

interface

Windows, Messages, SysUtils, Variants,

Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type

TForm1 = class(TForm)

Edit1: TEdit; // расстояние

Edit2: TEdit; // цена литра бензина
```

```
Edit3: TEdit; // потребление бензина на 100 км
CheckBox1: TCheckBox; // True - поездка туда и обратно
Button1: TButton; // кнопка Вычислить
Label4: TLabel; // поле вывода результата расчета
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
procedure EditlKeyPress(Sender: TObject;
var Key: Char);
procedure Edit2KeyPress(Sender: TObject;
var Key: Char);
procedure Edit3KeyPress(Sender: TObject;
var Key: Char);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.dfm}
// проверяет, является ли символ допустимым
// во время ввода дробного числа
```

```
function IsFloat(ch : char; st: string) : Boolean;
begin
if (ch >= '0') and (ch <= '9') // цифры
or (ch = #13) // клавиша <Enter>
or (ch = #8) // клавиша <Backspace>
then
begin
IsFloat := True; // символ верный
Exit; // выход из функции
end; case ch of
'-': if Length(st) = 0 then IsFloat := True; ', ':
if (Pos(',',st) = 0)
and (st[Length(st)] >= '0') and (st[Length(st)] <= '9')
then // разделитель можно ввести только после цифры
// и если он еще не введен
IsFloat := True/else // остальные символы запрещены
IsFloat := False;
end;
end;
// нажатие клавиши в поле Расстояние
procedure TForm1.EditlKeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit2.SetFocus // переместить курсор в поле Цена
```

```
else
If not IsFloat(Key,Edit2.Text) then Key := Chr(O);
end;
// нажатие клавиши в поле Цена
procedure TForml.Edit2KeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Edit3.SetFocus // переместить курсор в поле Потребление
else If not IsFloat(Key,Edit2.Text)
then Key := Chr (0);
end;
// нажатие клавиши в поле Потребление
procedure TForml.Edit3KeyPress(Sender: TObject;
var Key: Char);
begin
if Key = Char(VK_RETURN)
then Buttonl.SetFocus // // сделать активной кнопку Вычислить
else If not IsFloat(Key,Edit2.Text) then Key := Chr (0);
end;
// щелчок на кнопке Вычислить
procedure TForm1.ButtonlClick(Sender: TObject);
var
rast : real; // расстояние
cena : real; // цена
```

```
potr : real; // потребление на 100 км
summ : real; // сумма
mes: string;
begin
rast := StrToFloat(Edit1.Text);
cena := StrToFloat(Edit2.Text);
potr := StrToFloat(Edit3.Text);
summ := rast / 100 * potr * cena;
if CheckBoxl.Checked then summ := summ * 2;
mes := 'Поездка на дачу';
if CheckBoxl.Checked then mes := mes + ' и обратно';
mes := mes + 'обойдется в '
+ FloatToStrF(summ,ffGeneral,4,2) + ' руб.';
Label4.Caption := mes;
end;
end.
```