bhv

Никита Культин

Основы программирования в Delphi 7

Самоучитель

Основы, теория и практика программирования

Язык программирования Delphi

Графика, мультимедиа и базы данных

Создание справочной системы в Microsoft HTML Help Workshop

Дискета содержит примеры проектов

Весь процесс создания программ

УДК 681.3.06
ББК 32.973.26-018.1
К90

Культин Н. Б.

К90 Основы программирования в Delphi 7. — СПб.: БХВ-Петербург, 2007. — 608 с.: ил.

ISBN 978-5-94157-269-4

Книга является руководством по программированию в среде Delphi 7. Описывается весь процесс разработки программы: от конструирования диалогового окна до организации справочной системы и создания установочного CD-ROM. Материал включает ряд тем, которые, как правило, остаются за рамками книг, адресованных начинающим — обработка символьной информации, использование динамических структур, работа с файлами. Рассматриваются вопросы работы с графикой, мультимедиа и базами данных. Приведено описание процесса создания анимации, а также справочной системы при помощи программы Microsoft HTML Help Workshop, установочного CD-ROM в InstallShield Express. Книга отличается доступностью изложения, большим количеством наглядных примеров. Адресована студентам, школьникам старших классов и всем изучающим программирование.

Прилагаемый к книге диск содержит тексты программ, которые приведены в книге в качестве примеров.

Для начинающих программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Григорий Добин*
Компьютерная верстка	*Татьяны Олоновой*
Корректор	*Виктория Голуб*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.03.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 49,02.
Доп. тираж 2000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-94157-269-4

Содержание

Глава 1

Основы программирования

Программа

Программа, работающая на компьютере, нередко отождествляется с самим компьютером, т. к. человек, использующий программу, "вводит в компьютер" исходные данные, как правило, при помощи клавиатуры, а компьютер "выдает результат" на экран, на принтер или в файл. На самом деле, преобразование исходных данных в результат выполняет процессор компьютера. Процессор преобразует исходные данные в результат по определенному алгоритму, который, будучи записан на специальном языке, называется программой. Таким образом, чтобы компьютер выполнил некоторую работу, необходимо разработать последовательность команд, обеспечивающую выполнение этой работы, или, как говорят, написать программу.

Этапы разработки программы

Выражение "написать программу" отражает только один из этапов создания компьютерной программы, когда разработчик программы (программист) действительно пишет команды (инструкции) на бумаге или при помощи текстового редактора.

Программирование — это процесс создания (разработки) программы, который может быть представлен последовательностью следующих шагов:

1. Спецификация (определение, формулирование требований к программе).
2. Разработка алгоритма.
3. Кодирование (запись алгоритма на языке программирования).
4. Отладка.
5. Тестирование.

6. Создание справочной системы.
7. Создание установочного диска (CD-ROM).

Спецификация

Спецификация, определение требований к программе — один из важнейших этапов, на котором подробно описывается исходная информация, формулируются требования к результату, поведение программы в особых случаях (например, при вводе неверных данных), разрабатываются диалоговые окна, обеспечивающие взаимодействие пользователя и программы.

Разработка алгоритма

На этапе разработки алгоритма необходимо определить последовательность действий, которые надо выполнить для получения результата. Если задача может быть решена несколькими способами и, следовательно, возможны различные варианты алгоритма решения, то программист, используя некоторый критерий, например, скорость решения алгоритма, выбирает наиболее подходящее решение. Результатом этапа разработки алгоритма является подробное словесное описание алгоритма или его блок-схема.

Кодирование

После того как определены требования к программе и составлен алгоритм решения, алгоритм записывается на выбранном языке программирования. В результате получается *исходная* программа.

Отладка

Отладка — это процесс поиска и устранения ошибок. Ошибки в программе разделяют на две группы: синтаксические (ошибки в тексте) и алгоритмические. Синтаксические ошибки — наиболее легко устраняемые. Алгоритмические ошибки обнаружить труднее. Этап отладки можно считать законченным, если программа правильно работает на одном-двух наборах входных данных.

Тестирование

Этап тестирования особенно важен, если вы предполагаете, что вашей программой будут пользоваться другие. На этом этапе следует проверить, как ведет себя программа на как можно большем количестве входных наборов данных, в том числе и на заведомо неверных.

Создание справочной системы

Если разработчик предполагает, что программой будут пользоваться другие, то он обязательно должен создать справочную систему и обеспечить пользователю удобный доступ к справочной информации во время работы с программой. В современных программах справочная информация представляется в форме CHM- или HLP-файлов. Помимо справочной информации, доступ к которой осуществляется из программы во время ее работы, в состав справочной системы включают инструкцию по установке (инсталляции) программы, которую оформляют в виде Readme-файла в одном из форматов: TXT, DOC или HTM.

Процесс создания справочной системы и механизмы доступа к справочной информации описаны в *гл. 14*.

Создание установочного диска

Установочный диск или CD-ROM создаются для того, чтобы пользователь мог самостоятельно, без помощи разработчика, установить программу на свой компьютер. Обычно помимо самой программы на установочном диске находятся файлы справочной информации и инструкция по установке программы (Readme-файл). Следует понимать, что современные программы, в том числе разработанные в Delphi, в большинстве случаев (за исключением самых простых программ) не могут быть установлены на компьютер пользователя путем простого копирования, так как для своей работы требуют специальных библиотек и компонентов, которых может и не быть у конкретного пользователя. Поэтому установку программы на компьютер пользователя должна выполнять специальная программа, которая помещается на установочный диск. Как правило, установочная программа создает отдельную папку для устанавливаемой программы, копирует в нее необходимые файлы и, если надо, выполняет настройку операционной системы путем внесения дополнений и изменений в реестр.

Процесс создания установочного диска (CD-ROM) при помощи входящей в состав Delphi утилиты InstallShield Express описан в *гл. 18*.

Алгоритм и программа

На первом этапе создания программы программист должен определить последовательность действий, которые необходимо выполнить, чтобы решить поставленную задачу, т. е. разработать алгоритм. *Алгоритм* — это точное

предписание, определяющее процесс перехода от исходных данных к результату.

Алгоритм решения задачи может быть представлен в виде словесного описания или графически — в виде блок-схемы. При изображении алгоритма в виде блок-схемы используются специальные символы (рис. 1.1).

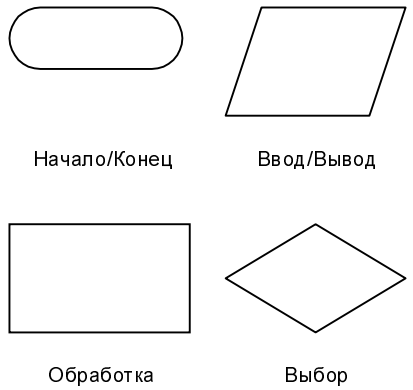

Рис. 1.1. Основные символы, используемые для представления алгоритма в виде блок-схемы

Представление алгоритма в виде блок-схемы позволяет программисту уяснить последовательность действий, которые должны быть выполнены для решения задачи, убедиться в правильности понимания поставленной задачи.

При программировании в Delphi алгоритм решения задачи представляет собой совокупность алгоритмов *процедур обработки событий*.

В качестве примера на рис. 1.2 приведена совокупность алгоритмов программы **Стоимость покупки**, а на рис. 1.3 — ее диалоговое окно. После разработки диалогового окна и алгоритмов обработки событий можно приступить к написанию программы. Ее текст приведен в листинге 1.1.

Листинг 1.1. Программа **Стоимость покупки**

```
unit pokupka_1;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;
```

```
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Label3: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

// подпрограмма
procedure Summa;
var
    cena: real;       // цена
    kol: integer;     // количество
    s: real;          // сумма
    mes: string[255]; // сообщение
begin
    cena := StrToFloat(Form1.Edit1.Text);
    kol := StrToInt(Form1.Edit2.Text);
    s := cena * kol;
    if s > 500 then
```

```
      begin
          s := s * 0.9;
          mes := 'Предоставляется скидка 10%' + #13;
      end;
    mes := mes+ 'Стоимость покупки: '
          + FloatToStrF(s,ffFixed,4,2) +' руб.';
    Form1.Label3.Caption := mes;
end;

// щелчок на кнопке Стоимость
procedure TForm1.Button1Click(Sender: TObject);
begin
       Summa; // вычислить сумму покупки
end;

// нажатие клавиши в поле Количество
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
    case Key of
      '0' .. '9',#8: ;     // цифры и клавиша <Backspace>
      #13:        Summa;   // вычислить стоимость покупки
      else Key := Chr(0);  // символ не отображать
    end;
end;

// нажатие клавиши в поле Цена
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    case Key of
       '0' .. '9', #8 :    ;      // цифры и клавиша <Backspace>

        #13: Form1.Edit2.SetFocus; // клавиша <Enter>

        '.', ',':
             begin
                if  Key = '.'
                    then Key:=',';
                if Pos(',',Edit1.Text) <> 0
```

```
                    then  Key:= Chr(0);
                end;
            else    // все остальные символы запрещены
                Key := Chr(0);
    end;
end;

end.
```

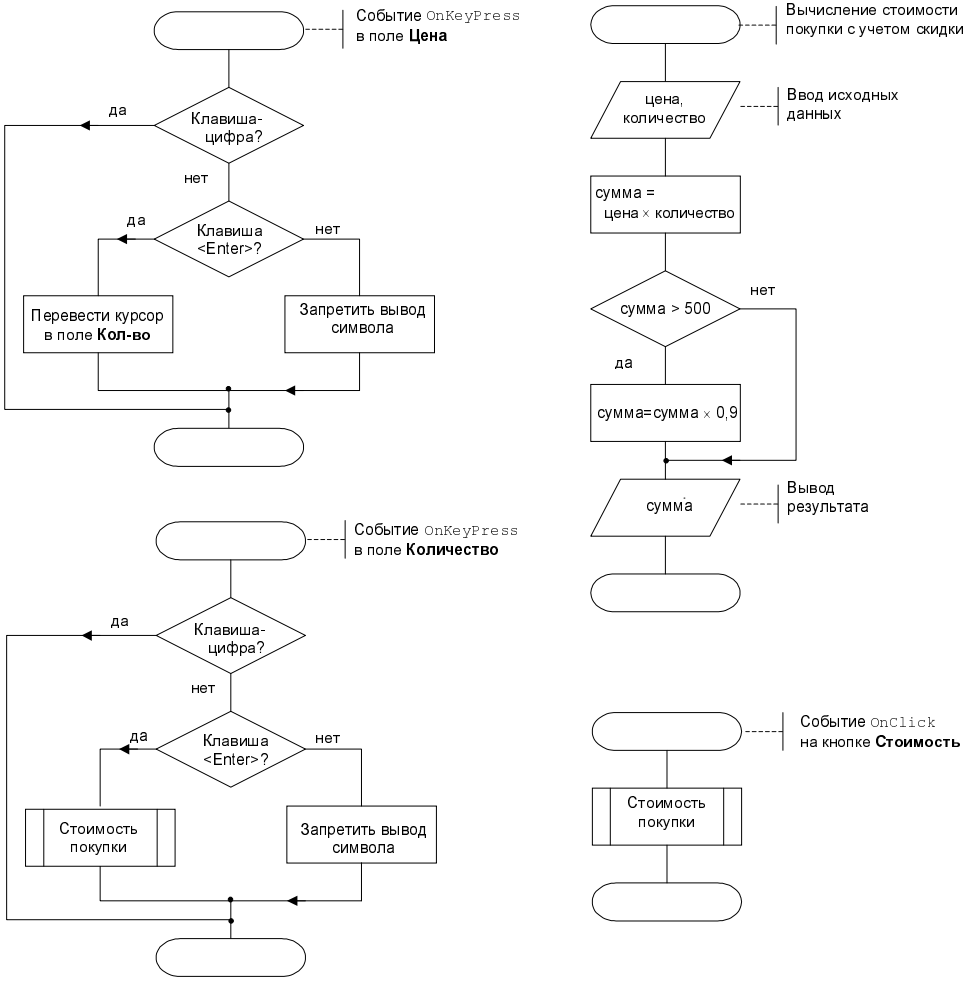

Рис. 1.2. Алгоритм программы вычисления стоимости покупки — совокупность алгоритмов обработки событий на компонентах формы

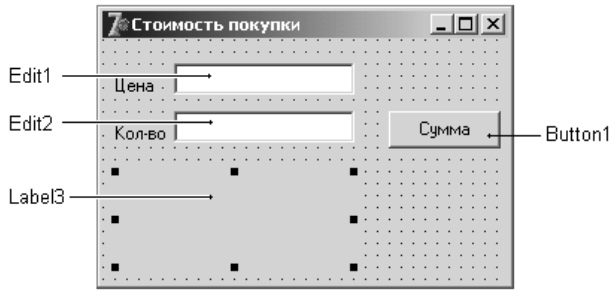

Рис. 1.3. Окно (форма) программы **Стоимость покупки**

Компиляция

Программа, представленная в виде инструкций языка программирования, называется исходной программой. Она состоит из инструкций, понятных человеку, но не понятных процессору компьютера. Чтобы процессор смог выполнить работу в соответствии с инструкциями исходной программы, исходная программа должна быть переведена на машинный язык — язык команд процессора. Задачу преобразования исходной программы в машинный код выполняет специальная программа — компилятор.

Компилятор, схема работы которого приведена на рис. 1.4, выполняет последовательно две задачи:

1. Проверяет текст исходной программы на отсутствие синтаксических ошибок.
2. Создает (генерирует) исполняемую программу — машинный код.

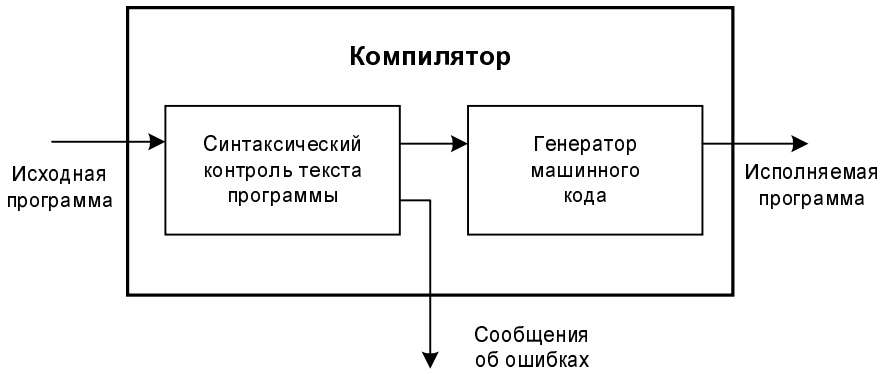

Рис. 1.4. Схема работы компилятора

Следует отметить, что генерация исполняемой программы происходит только в том случае, если в тексте исходной программы нет синтаксических ошибок.

Генерация машинного кода компилятором свидетельствует лишь о том, что в тексте программы нет синтаксических ошибок. Убедиться, что программа работает правильно можно только в процессе ее тестирования — пробных запусках программы и анализе полученных результатов. Например, если в программе вычисления корней квадратного уравнения допущена ошибка в выражении (формуле) вычисления дискриминанта, то, даже если это выражение будет синтаксически верно, программа выдаст неверные значения корней.

Язык программирования Delphi

В среде программирования Delphi для записи программ используется язык программирования Delphi. Программа на Delphi представляет собой последовательность *инструкций*, которые довольно часто называют *операторами*. Одна инструкция от другой отделяется точкой с запятой.

Каждая инструкция состоит из *идентификаторов*. Идентификатор может обозначать:

- инструкцию языка (`:=`, `if`, `while`, `for`);
- переменную;
- константу (целое или дробное число);
- арифметическую (`+`, `-`, `*`, `/`) или логическую (`and`, `or`, `not`) операцию;
- подпрограмму (процедуру или функцию);
- отмечать начало (`procedure`, `function`) или конец (`end`) подпрограммы или блока (`begin`, `end`).

Тип данных

Программа может оперировать данными различных типов: целыми и дробными числами, символами, строками символов, логическими величинами.

Целый тип

Язык Delphi поддерживает семь целых типов данных: `Shortint`, `Smallint`, `Longint`, `Int64`, `Byte`, `Word` и `Longword`, описание которых приведено в табл. 1.1.

Таблица 1.1. *Целые типы*

Тип	Диапазон	Формат
`Shortint`	–128 — 127	8 битов
`Smallint`	–32 768 — 32 767	16 битов
`Longint`	–2 147 483 648 — 2 147 483 647	32 бита
`Int64`	-2^{63} — $2^{63}-1$	64 бита
`Byte`	0 — 255	8 битов, беззнаковый
`Word`	0 — 65 535	16 битов, беззнаковый
`Longword`	0 — 4 294 967 295	32 бита, беззнаковый

Object Pascal поддерживает и наиболее универсальный целый тип — `Integer`, который эквивалентен `Longint`.

Вещественный тип

Язык Delphi поддерживает шесть вещественных типов: `Real48`, `Single`, `Double`, `Extended`, `Comp`, `Currency`. Типы различаются между собой диапазоном допустимых значений, количеством значащих цифр и количеством байтов, необходимых для хранения данных в памяти компьютера (табл. 1.2).

Таблица 1.2. *Вещественные (дробные) типы*

Тип	Диапазон	Значащих цифр	Байтов
`Real48`	2.9×10^{-39} — 1.7×10^{38}	11—12	06
`Single`	1.5×10^{-45} — 3.4×10^{38}	7—8	04
`Double`	5.0×10^{-324} — 1.7×10^{308}	15—16	08
`Extended`	3.6×10^{-4951} — 1.1×10^{4932}	19—20	10
`Comp`	$-2^{63}+1$ — $2^{63}-1$	19—20	08
`Currency`	–922 337 203 685 477.5808 — –922 337 203 685 477.5807	19—20	08

Язык Delphi поддерживает и наиболее универсальный вещественный тип — `Real`, который эквивалентен `Double`.

Символьный тип

Язык Delphi поддерживает два символьных типа: `AnsiChar` и `WideChar`:

- тип `AnsiChar` — это символы в кодировке ANSI, которым соответствуют числа в диапазоне от 0 до 255;
- тип `WideChar` — это символы в кодировке Unicode, им соответствуют числа от 0 до 65 535.

Object Pascal поддерживает и наиболее универсальный символьный тип — `Char`, который эквивалентен `AnsiChar`.

Строковый тип

Язык Delphi поддерживает три строковых типа: `ShortString`, `LongString` и `WideString`:

- тип `ShortString` представляет собой статически размещаемые в памяти компьютера строки длиной от 0 до 255 символов;
- тип `LongString` представляет собой динамически размещаемые в памяти строки, длина которых ограничена только объемом свободной памяти;
- тип `WideString` представляет собой динамически размещаемые в памяти строки, длина которых ограничена только объемом свободной памяти. Каждый символ строки типа `WideString` является Unicode-символом.

В языке Delphi для обозначения строкового типа допускается использование идентификатора `String`. Тип `String` эквивалентен типу `ShortString`.

Логический тип

Логическая величина может принимать одно из двух значений `True` (истина) или `False` (ложь). В языке Delphi логические величины относят к типу `Boolean`.

Переменная

Переменная — это область памяти, в которой находятся данные, которыми оперирует программа. Когда программа манипулирует с данными, она, фактически, оперирует содержимым ячеек памяти, т. е. переменными.

Чтобы программа могла обратиться к переменной (области памяти), например, для того, чтобы получить исходные данные для расчета по формуле или сохранить результат, переменная должна иметь имя. Имя переменной придумывает программист.

В качестве имени переменной можно использовать последовательность из букв латинского алфавита, цифр и некоторых специальных символов. Первым символом в имени переменной должна быть буква. Пробел в имени переменной использовать нельзя.

Следует обратить внимание на то, что компилятор языка Delphi не различает прописные и строчные буквы в именах переменных, поэтому имена `SUMMA`, `Summa` и `summa` обозначают одну и ту же переменную.

Желательно, чтобы имя переменной было логически связано с ее назначением. Например, переменным, предназначенным для хранения коэффициентов и корней квадратного уравнения, которое в общем виде традиционно записывают

```
ax2 + bx + c = 0
```

вполне логично присвоить имена `a`, `b`, `c`, `x1` и `x2`. Другой пример. Если в программе есть переменные, предназначенные для хранения суммы покупки и величины скидки, то этим переменным можно присвоить имена `TotalSumm` и `Discount` или `ObSumma` и `Skidka`.

В языке Delphi каждая переменная перед использованием должна быть объявлена. С помощью объявления устанавливается не только факт существования переменной, но и задается ее тип, чем указывается и диапазон допустимых значений.

В общем виде инструкция объявления переменной выглядит так:

Имя : *тип*;

где:

- *Имя* — имя переменной;
- *тип* — тип данных, для хранения которых предназначена переменная.

Пример:

```
a : Real;
b : Real;
i : Integer;
```

В приведенных примерах объявлены две переменные типа `real` и одна переменная типа `integer`.

В тексте программы объявление каждой переменной, как правило, помещают на отдельной строке.

Если в программе имеется несколько переменных, относящихся к одному типу, то имена этих переменных можно перечислить в одной строке через запятую, а тип переменных указать после имени последней переменной через двоеточие, например:

```
a,b,c : Real;
x1,x2 : Real;
```

Константы

В языке Delphi существует два вида констант: *обычные* и *именованные*.

Обычная константа — это целое или дробное число, строка символов или отдельный символ, логическое значение.

Числовые константы

В тексте программы числовые константы записываются обычным образом, т. е. так же, как числа, например, при решении математических задач. При записи дробных чисел для разделения целой и дробных частей используется точка. Если константа отрицательная, то непосредственно перед первой цифрой ставится знак "минус".

Ниже приведены примеры числовых констант:

```
123
0.0
-524.03
0
```

Дробные константы могут изображаться в виде числа с плавающей точкой. Представление в виде числа с плавающей точкой основано на том, что любое число может быть записано в алгебраической форме как произведение числа, меньшего 10, которое называется мантиссой, и степени десятки, именуемой порядком.

В табл. 1.3 приведены примеры чисел, записанных в обычной форме, в алгебраической форме и форме с плавающей точкой.

Таблица 1.3. *Примеры записи дробных чисел*

Число	Алгебраическая форма	Форма с плавающей точкой
1 000 000	1×10^{6}	1,0000000000E+06
−123.452	$-1{,}23452\times10^{2}$	−1,2345200000E+02
0,0056712	$5{,}6712\times10^{-3}$	5,6712000000E-03

Строковые и символьные константы

Строковые и символьные константы заключаются в кавычки. Ниже приведены примеры строковых констант:

```
'Язык программирования Delphi'
'Delphi 7'
```

```
'2.4'
'Д'
```

Здесь следует обратить внимание на константу `'2.4'`. Это именно символьная константа, т. е. строка символов, которая изображает число "две целые четыре десятых", а не число 2,4.

Логические константы

Логическое высказывание (выражение) может быть либо истинно, либо ложно. Истине соответствует константа `True`, значению "ложь" — константа `False`.

Именованная константа

Именованная константа — это имя (идентификатор), которое в программе используется вместо самой константы.

Именованная константа, как и переменная, перед использованием должна быть объявлена. В общем виде инструкция объявления именованной константы выглядит следующим образом:

константа = *значение*;

где:

- *константа* — имя константы;
- *значение* — значение константы.

Именованные константы объявляются в программе в разделе объявления констант, который начинается словом `const`. Ниже приведен пример объявления именованных констант (целой, строковой и дробной).

```
const
    Bound = 10;
    Title = 'Скорость бега';
    pi = 3.1415926;
```

После объявления именованной константы в программе вместо самой константы можно использовать ее имя.

В отличие от переменной, при объявлении константы тип явно не указывают. Тип константы определяется ее видом, например:

- `125` — константа целого типа;
- `0.0` — константа вещественного типа;
- `'Выполнить'` — строковая константа;
- `'\'` — символьная константа.

Инструкция присваивания

Инструкция присваивания является основной вычислительной инструкцией. Если в программе надо выполнить вычисление, то нужно использовать инструкцию присваивания.

В результате выполнения инструкции присваивания значение переменной меняется, ей присваивается значение.

В общем виде инструкция присваивания выглядит так:

```
Имя := Выражение;
```

где:

- *Имя* — переменная, значение которой изменяется в результате выполнения инструкции присваивания;
- := — символ инструкции присваивания.
- *Выражение* — выражение, значение которого присваивается переменной, имя которой указано слева от символа инструкции присваивания.

Пример:

```
Summa := Cena * Kol;
Skidka := 10;
Found := False;
```

Выражение

Выражение состоит из операндов и операторов. Операторы находятся между операндами и обозначают действия, которые выполняются над операндами. В качестве операндов выражения можно использовать: переменную, константу, функцию или другое выражение. Основные алгебраические операторы приведены в табл. 1.4.

Таблица 1.4. *Алгебраические операторы*

Оператор	Действие
+	Сложение
-	Вычитание
*	Умножение
/	Деление
DIV	Деление нацело
MOD	Вычисление остатка от деления