

Г. Г. Рапаков  
С. Ю. Ржеуцкая

var f:

```
begin  
  x:=random; (write(x:6:3));  
  sum:=sum+x;  
  interval;  
  intx:=round(x); (write(intx:2));  
  maxlenh;  
end;
```

# Программирование на языке Pascal

- *Базовые понятия программирования*
- *Конструкции языка*
- *Структурное и модульное программирование*
- *Массивы, множества, записи, файлы*
- *Динамические структуры данных*
- *Введение в объектно-ориентированное программирование*



```
var s1,s2:integer; print:1;begin  
  randomize;  
  for k:=1 to 5 do begin  
    s1:=random[10]+2; s2:=random[10]+2;
```

Министерство образования Российской Федерации  
Вологодский государственный технический университет

**Г. Г. Рапаков**  
**С. Ю. Ржеуцкая**

# Программирование на языке **PASCAL**

*Рекомендовано УМО*

*по университетскому политехническому образованию  
для студентов высших учебных заведений, обучающихся  
по специальности 220400 – "Программное обеспечение  
вычислительной техники и автоматизированных систем"*

Санкт-Петербург

«БХВ-Петербург»

2004

УДК 681.3.068+800.92Pascal  
ББК 32.973.26-018.1я7  
P23

**Рапаков Г. Г., Ржеуцкая С. Ю.**

P23 Программирование на языке Pascal. — СПб.: БХВ-Петербург, 2004. — 480 с.: ил.

ISBN 5-94157-401-0

Учебное пособие ориентировано на широкий круг читателей, как начинающих знакомство с программированием, так и имеющих в нем достаточный опыт. Необходимое для новичков изложение азов предмета сочетается в книге с подробным и глубоким описанием тонкостей языка Pascal. Издание насыщено примерами и содержит множество полезных рекомендаций. Особое внимание уделено вопросам стиля в программировании, как линейном, так и объектно-ориентированном. Каждую главу завершают контрольные вопросы и задания, сложность которых дозированно возрастает от начальных глав к конечным. Эти материалы могут быть рекомендованы преподавателям информатики средней и высшей школ в качестве методических.

*Для школьников старших классов и студентов*

УДК 681.3.068+800.92Pascal  
ББК 32.973.26-018.1я7

### **Группа подготовки издания:**

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Людмила Еремеевская</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Галина Смирнова</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Наталья Першакова</i>
Дизайн обложки	<i>Игоря Цырульниковца</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 20.10.03.

Формат 70×100<sup>1/16</sup>. Печать офсетная. Усл. печ. л. 38,7.

Тираж 4000 экз. Заказ №

"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953 Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов  
в Академической типографии "Наука" РАН  
199034, Санкт-Петербург, 9 линия, 12.

# Содержание

<b>Введение</b> .....	<b>1</b>
Как работать с книгой.....	2
<b>Часть I. ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ. БАЗОВЫЕ ПОНЯТИЯ</b> .....	<b>5</b>
<b>Глава 1. Основы алгоритмизации</b> .....	<b>7</b>
1.1. Алгоритмизация и требования к алгоритму.....	7
1.2. Способы записи алгоритмов.....	8
1.2.1. Описательный способ .....	8
1.2.2. Блок-схемы алгоритмов.....	9
1.2.3. Пример блок-схемы алгоритма.....	11
1.2.4. Следование, ветвление, цикл .....	13
1.3. Контрольные вопросы и задания .....	13
<b>Глава 2. Язык программирования. Программа</b> .....	<b>14</b>
2.1. Язык программирования .....	14
2.1.1. Исторический обзор.....	14
2.1.2. Состав языка программирования. Синтаксис и семантика.....	19
2.1.3. Описание языка .....	21
2.1.4. Стандарт и реализации языка .....	23
2.2. Программа. Программный продукт и его характеристики .....	25
2.3. Жизненный цикл программы .....	26
2.4. Контрольные вопросы .....	28
<b>Глава 3. Система программирования</b> .....	<b>30</b>
3.1. Компиляторы и интерпретаторы .....	30
3.2. Этапы компиляции и компоновки.....	32
3.3. Отладка программы. Типы ошибок .....	33
3.4. Состав системы программирования. Понятие интегрированной среды разработки.....	34
3.5. Системы программирования на основе языка Pascal.....	35
3.6. Контрольные вопросы .....	36
<b>Глава 4. Программа и данные. Типы данных</b> .....	<b>38</b>
4.1. Основные понятия .....	38
4.1.1. Исполнение программы.....	38
4.1.2. Биты и байты .....	40

4.2. Константы и переменные.....	41
4.3. Типы данных и способы внутреннего представления данных.....	42
4.3.1. Концепция типов данных.....	42
4.3.2. Внутреннее представление данных в памяти компьютера.....	43
4.3.3. Классификация типов.....	48
4.4. Контрольные вопросы и задания .....	49
<b>Часть II. Основы программирования на языке PASCAL.....</b>	<b>51</b>
<b>Глава 5. Типы данных и константы языка Pascal .....</b>	<b>53</b>
5.1. Классификация типов. Стандартные типы: порядковые и вещественные .....	54
5.2. Целые типы. Логический тип .....	55
5.3. Символьный тип. Строки символов .....	56
5.4. Вещественные типы .....	58
5.5. Типы данных, определяемые программистом .....	59
5.6. Перечисляемый и интервальный типы.....	60
5.7. Контрольные вопросы и задания .....	62
<b>Глава 6. Программа на языке Pascal.....</b>	<b>64</b>
6.1. Алфавит языка .....	64
6.2. Лексика языка .....	65
6.3. Структура программы .....	69
6.3.1. Общие сведения.....	69
6.3.2. Раздел <i>uses</i> .....	71
6.3.3. Раздел описания меток .....	71
6.3.4. Раздел описания констант.....	71
6.3.5. Раздел описания типов данных.....	73
6.3.6. Раздел описания переменных .....	74
6.3.7. Раздел описания процедур и функций .....	75
6.3.8. Раздел операторов .....	76
6.4. Контрольные вопросы и задания .....	77
<b>Глава 7. Простые (линейные) программы.....</b>	<b>79</b>
7.1. Пример учебной программы.....	79
7.2. Ввод и вывод данных.....	80
7.2.1. Инструкции ввода и вывода.....	80
7.2.2. Формат вывода.....	83
7.3. Оператор присваивания.....	84
7.4. Арифметические выражения.....	85
7.4.1. Арифметические операции.....	85
7.4.2. Побитовые операции.....	88
7.4.3. Приоритет операций .....	90
7.4.4. Стандартные арифметические функции .....	91

7.4.5. Типы в арифметических выражениях. Автоматическое преобразование типов .....	93
7.4.6. Явное преобразование типов. Переполнение.....	94
7.5. Полезные формулы .....	97
7.6. Примеры программ .....	97
7.6.1. Вычисления по формулам .....	97
7.6.2. Выделение цифр целого числа.....	98
7.6.3. Перестановка значений переменных .....	99
7.6.4. Определение размера ячейки памяти.....	100
7.7. Контрольные вопросы и задания .....	101
<b>Глава 8. Программирование разветвлений .....</b>	<b>104</b>
8.1. Логические выражения.....	104
8.2. Составной оператор .....	107
8.3. Условный оператор .....	108
8.4. Оператор выбора .....	112
8.5. Оператор безусловного перехода. Пустой оператор.....	114
8.6. Примеры программ .....	117
8.6.1. Разные задачи .....	117
8.6.2. День недели по заданной дате .....	118
8.7. Контрольные вопросы и задания .....	119
<b>Глава 9. Циклические программы .....</b>	<b>123</b>
9.1. Цикл с постусловием .....	124
9.2. Цикл с предусловием .....	129
9.3. Цикл с параметром .....	132
9.4. Вложенные циклы.....	137
9.5. Примеры программ .....	138
9.5.1. Выделение цифр целого числа.....	138
9.5.2. Обработка числовых данных.....	139
9.5.3. Формирование числовых последовательностей .....	140
9.5.4. Вычисление сумм бесконечных убывающих последовательностей. Особенности арифметики с плавающей точкой.....	141
9.5.5. График функции на текстовом экране .....	145
9.5.6. Задачи на перебор всех вариантов.....	146
9.5.7. "Римские цифры" .....	148
9.6. Контрольные вопросы и задания .....	151
<b>Часть III. СТРУКТУРНОЕ И МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ .....</b>	<b>155</b>
<b>Глава 10. Процедуры и функции .....</b>	<b>157</b>
10.1. Общие сведения о подпрограммах .....	157
10.2. Процедуры.....	158
10.3. Механизм передачи параметров .....	161
10.3.1. Параметры-значения .....	162
10.3.2. Параметры-переменные.....	163
10.3.3. Параметры-константы.....	164

10.4. Функции .....	165
10.5. Область видимости и время жизни переменной .....	168
10.5.1. Подробнее о распределении памяти .....	168
10.5.2. Вложенные процедуры и функции .....	170
10.5.3. Глобальные, автоматические и статические переменные. Определения .....	171
10.5.4. Побочные эффекты вызова подпрограмм .....	173
10.5.5. Рекомендации по разработке программ .....	174
10.6. Рекурсия .....	174
10.7. Дополнительные сведения о процедурах и функциях .....	176
10.7.1. Опережающее объявление .....	176
10.7.2. Параметры-процедуры и параметры-функции .....	177
10.7.3. Нетипизированные параметры-переменные .....	179
10.8. Примеры программ .....	180
10.8.1. Числовые последовательности .....	181
10.8.2. Численные методы решения математических задач .....	182
10.8.3. Случайные числа .....	186
10.8.4. Игра "Ханойские башни" .....	188
10.9 Контрольные вопросы и задания .....	189
<b>Глава 11. Структуризация в программировании .....</b>	<b>192</b>
11.1. Теорема структуры .....	192
11.2. Основы структурного программирования .....	193
11.2.1. Методы структурного программирования .....	193
11.2.2. Пример разработки программы .....	195
11.3. Контрольные вопросы и задания .....	200
<b>Глава 12. Библиотечные модули .....</b>	<b>203</b>
12.1. Компиляция и компоновка программы .....	203
12.2. Понятие модуля. Преимущества модульного программирования .....	204
12.3. Структура модуля .....	205
12.4. Снова о глобальных, локальных и статических переменных .....	208
12.5. Пример модуля .....	209
12.6. Раздельная компиляция модулей .....	212
12.6.1. Компиляция модулей в Turbo Pascal .....	212
12.6.2. Управление модулями в Delphi. Понятие проекта .....	213
12.6.3. Распространение готовых модулей .....	214
12.7. Стандартные модули .....	214
12.8. Контрольные вопросы и задания .....	216
<b>Часть IV. СТРУКТУРЫ ДАННЫХ И АЛГОРИТМЫ ОБРАБОТКИ .....</b>	<b>219</b>
<b>Глава 13. Массивы .....</b>	<b>221</b>
13.1. Общие сведения .....	221
13.2. Описание массивов .....	222
13.2.1. Описание массива в разделе <i>var</i> .....	223

13.2.2. Предварительное описание типа массива.....	223
13.2.3. Инициализация массивов начальными значениями .....	224
13.2.4. Подробнее о выделении памяти .....	225
13.3. Действия над массивами.....	227
13.3.1. Заполнение массива данными.....	227
13.3.2. Вывод элементов массива.....	229
13.3.3. Обработка одномерных массивов .....	229
13.3.4. Действия с двумерными массивами .....	233
13.3.5. Перестановки элементов в массиве.....	236
13.3.6. Сортировка массива .....	237
13.3.7. Быстрый поиск в упорядоченных массивах .....	241
13.3.8. Удаление и вставка элементов в массив .....	242
13.3.9. Умножение матриц.....	244
13.4. Массивы как параметры подпрограмм .....	247
13.5. Примеры программ .....	249
13.5.1. Перевод числа в двоичную систему .....	249
13.5.2. Решение системы линейных уравнений .....	249
13.6. Контрольные вопросы и задания .....	252
<b>Глава 14. Обработка строк текста .....</b>	<b>256</b>
14.1. Типы данных <i>char</i> и <i>string</i> .....	256
14.1.1. Символьный тип.....	256
14.1.2. Строковый тип.....	257
14.2. Операции над строками.....	261
14.2.1. Операция сцепления (+).....	261
14.2.2. Операции отношения.....	263
14.3. Строковые процедуры и функции.....	264
14.3.1. Процедуры удаления и вставки символов .....	265
14.3.2. Функции для работы со строками.....	266
14.3.3. Процедуры преобразования типов .....	267
14.4. Примеры программ .....	268
14.4.1. Вставка, удаление и замена фрагментов текста .....	268
14.4.2. Преобразование строчных букв в заглавные .....	270
14.4.3. Удаление комментариев из строки программы.....	271
14.4.4. Вычисление арифметического выражения .....	274
14.5. Контрольные вопросы и задания .....	278
<b>Глава 15. Множества .....</b>	<b>281</b>
15.1. Понятие множества.....	281
15.2. Операции над множествами.....	283
15.3. Примеры программ .....	287
15.3.1. Формирование случайных неповторяющихся чисел .....	287
15.4. Контрольные вопросы и задания .....	289
<b>Глава 16. Записи.....</b>	<b>291</b>
16.1. Работа с записями. Примеры.....	291
16.1.1. Определение типа "запись". Правила работы с записями.....	291



16.1.2. Пример использования массива записей .....	294
16.1.3. Модуль для выполнения операций с комплексными числами .....	296
16.2. Записи с вариантами. Пример .....	299
16.3. Контрольные вопросы и задания .....	303
<b>Глава 17. Файлы.....</b>	<b>306</b>
17.1. Некоторые сведения о файловой системе .....	306
17.1.1. Имя и расширение файла.....	307
17.1.2. Каталоги.....	307
17.1.3. Устройства .....	310
17.2. Описание файлового типа .....	311
17.2.1. Виды файлов. Файловая переменная .....	311
17.2.2. Указатель. Доступ к файлам .....	311
17.3. Общая схема работы с файлами .....	312
17.3.1. Последовательность действий .....	312
17.3.2. Общие процедуры и функции .....	313
17.3.3. Процедуры для работы с каталогами .....	315
17.3.4. Процедуры и функции модуля <i>dos</i> .....	315
17.4. Текстовые файлы.....	316
17.4.1. Процедуры и функции для текстовых файлов .....	317
17.4.2. Стандартные текстовые файловые переменные <i>input</i> и <i>output</i> .....	320
17.4.3. Задачи на обработку текстовых файлов .....	321
17.5. Типизированные файлы .....	326
17.5.1. Процедуры и функции для типизированных файлов.....	327
17.5.2. Задачи на обработку типизированных файлов.....	328
17.6. Нетипизированные файлы .....	335
17.7. Контрольные вопросы и задания .....	336
<b>Глава 18. Динамические структуры данных.....</b>	<b>339</b>
18.1. Еще раз о распределении памяти .....	339
18.2. Указатели. Описание указателей .....	341
18.2.1. Указатели и адреса.....	342
18.2.2. Описание указателей .....	343
18.3. Создание и удаление динамических переменных.....	344
18.4. Динамически формируемые массивы и строки.....	346
18.5. Структуры данных на основе указателей.....	350
18.6. Связанные списки.....	351
18.6.1. Общие сведения.....	351
18.6.2. Примеры программ .....	354
18.6.3. Сравнение связанных списков и массивов.....	359
18.7. Деревья. Двоичные поисковые деревья .....	360
18.7.1. Общие сведения.....	360
18.7.2. Двоичные поисковые деревья. Индексы .....	361
18.7.3. Пример построения двоичного поискового дерева.....	363
18.8. Контрольные вопросы и задания .....	366

<b>Глава 19. Введение в объектно-ориентированное программирование .....</b>	<b>369</b>
19.1. Абстрактные типы данных .....	369
19.1.1. Понятие АТД и структур данных.....	369
19.1.2. Стандартные АТД .....	370
19.1.3. Принцип абстрагирования .....	375
19.2. Введение в ООП .....	375
19.2.1. Принцип сокрытия деталей. Идеи ООП .....	375
19.2.2. ООП и Pascal.....	377
19.2.3. Определение класса.....	377
19.2.4. Область видимости элементов класса .....	379
19.2.5. Пример описания класса.....	379
19.2.6. Создание и уничтожение объектов. Конструкторы и деструкторы .....	382
19.2.7. Механизм наследования .....	384
19.2.8. Переопределение методов базового класса в потомках .....	385
19.2.9. Абстрактные классы .....	388
19.2.10. Свойства ( <i>property</i> ) .....	389
19.2.11. Применение ООП.....	392
19.3. Контрольные вопросы и задания .....	393
<b>Заключение.....</b>	<b>395</b>

## **Часть V. ПРИЛОЖЕНИЯ .....**

<b>Приложение 1. Компиляторы и системы программирования на основе Pascal.....</b>	<b>399</b>
П1.1. Особенности языка Delphi .....	399
П1.2. Другие компиляторы и системы программирования на основе языка Pascal.....	408
П1.2.1 Free Pascal.....	409
П1.2.2. GNU Pascal .....	411
П1.2.3. TMT Pascal .....	411
П1.2.4. Virtual Pascal .....	413
<b>Приложение 2. Стандартные модули Turbo Pascal и дополнения к ним .....</b>	<b>414</b>
П2.1. Принципы формирования изображения.....	414
П2.2. Модуль <i>crt</i> .....	415
П2.2.1. Процедуры и функции управления экраном .....	415
П2.2.2. Работа с окнами.....	419
П2.2.3. Задержка при выполнении программы.....	421
П2.2.4. Управление клавиатурой.....	422
П2.2.5. Управление звуком .....	424
П2.3. Модуль <i>graph</i> .....	426
П2.3.1. Общие сведения.....	426
П2.3.2. Графические примитивы .....	429

П2.3.3. Установка цветов и стилей .....	434
П2.3.4. Окна в графическом режиме.....	436
П2.3.5. Вывод текста .....	437
П2.3.6. Сохранение и восстановление битовых образов изображений .....	438
П2.4. Модуль DOS .....	440
П2.4.1. Работа с системной датой и временем.....	441
П2.4.2. Функции для обработки параметров командной строки.....	442
П2.4.3. Запуск внешних программ из программы на Turbo Pascal.....	442
П2.5. Дополнения к модулям.....	444
П2.5.1. Модуль <i>crtplus</i> .....	444
П2.5.2. Модуль для подключения мыши к программе на Turbo Pascal.....	447
П2.5.3. Дополнение к модулю <i>graph</i> — вывод рисунков в формате BMP.....	449

### **Приложение 3. Основы практической работы в интегрированной среде Turbo Pascal .....455**

П3.1. Работа в окне интегрированной среды, текстовый редактор Turbo Pascal .....	455
П3.1.1. Общие сведения.....	455
П3.1.2. Начало работы .....	456
П3.1.3. Создание новой программы .....	456
П3.1.4. Набор и редактирование текста.....	457
П3.1.5. Работа с окнами.....	458
П3.2. Справочная система.....	459
П3.3. Компиляция программы, поиск и устранение ошибок.....	460
П3.4. Запуск программы на выполнение, просмотр результатов, отладка.....	460
П3.4.1. Пример работы с отладчиком .....	461
П3.5. Перечень ошибок.....	463

### **Список литературы .....465**

### **Предметный указатель .....467**

# Введение

Книга, предлагаемая вниманию читателя, предназначена для использования в качестве *базового учебного пособия по дисциплине "Программирование на языке высокого уровня"*. Она может быть полезна как для *студентов*, обучающихся по специальностям направления "Информатика и вычислительная техника", так и для студентов других родственных направлений, а также для *школьников старших классов*.

Программирование — интересная, живая, быстро развивающаяся наука. Однако первые шаги при обучении программированию для многих оказываются очень нелегкими. Главное качество программиста — хорошее логическое мышление — развивается только в упорной и кропотливой работе.

С другой стороны, нелегко приходится и преподавателю начального курса программирования. Он понимает, что первые шаги начинающих программистов — основа их будущих успехов, но прекрасно видит и трудности начального этапа обучения. Авторы книги, преподаватели с многолетним стажем, стремились к тому, чтобы оказать максимальную поддержку и ученику, и преподавателю в нелегком деле освоения алгоритмов и программ.

Несмотря на обилие литературы по программированию, трудно найти учебник, который в первую очередь способствовал бы развитию навыков алгоритмизации, оставляя на втором плане особенности конкретного языка программирования. Данная книга представляет собой именно *учебник по программированию*, а не справочник по конструкциям языка программирования.

Нельзя, однако, научиться программировать на всех языках сразу. Поэтому в качестве базового языка был принят строгий и лаконичный Pascal (Паскаль), хорошо зарекомендовавший себя как *первый язык* при обучении. Авторы в дальнейшем будут пользоваться англоязычным названием, поскольку многие диалекты языка не имеют устойчивого перевода (Free Pascal, GNU Pascal и т. д.). Разумеется, все конструкции языка Pascal изложены в полном объеме и пояснены на примерах, так что никакого дополнительного справочника по языку не потребуется.

*Тематику книги* можно определить как "Конспект начинающего программиста" или "Популярный учебник". Ее объективную оценку могут дать читательская аудитория и специалисты. С точки же зрения авторов их труд отличается от множества других изданий удачное сочетание нескольких факторов:

- *краткости*, свойственной конспекту лекций;
- *строгой систематизации* материала;

- *доступности для усвоения*, присущей отработанному курсу, вобравшему в себя собственный опыт авторов и опыт, почерпнутый ими из многочисленных публикаций;
- *привлечения* внимания читателя к наиболее важным материалам в каждой теме;
- *наличия большого количества примеров с комментариями*, облегчающих обучение и стимулирующих интерес к материалу;
- *наличия контрольных вопросов в конце каждой главы*, которые способствуют проверке и закреплению пройденного теоретического материала;
- *наличия контрольных заданий*, которые могут быть использованы для практической работы со студентами и школьниками и существенно облегчают труд преподавателя.

Представленный учебный материал прошел успешную проверку на занятиях со студентами и школьниками.

## Как работать с книгой

*Пособие состоит из четырех частей.*

В первой части рассматриваются ключевые вопросы алгоритмизации и программирования, вводится система основных понятий и определений. Эта часть не привязана ни к какому конкретному языку программирования, в ней нет примеров программ. Однако для успешного освоения практической части курса необходимо перед началом работы за компьютером внимательно прочитать материал первой части (она невелика по объему) и ответить на контрольные вопросы. Возможно, при первом чтении отдельные моменты покажутся непонятными или неинтересными — не фиксируйте на них внимание и переходите к работе над второй частью.

Главы второй части посвящены разбору конструкций языка высокого уровня и проиллюстрированы большим количеством примеров программ на языке Pascal.

Программы отлажены в среде Borland Pascal, однако они работоспособны и в среде Delphi, нужно только правильно выбрать тип приложения — консольное (Console Application). Все же рекомендуем начинать обучение с простой и испытанной среды Borland (Turbo) Pascal. Если она плохо работает под управлением вашей операционной системы, можно воспользоваться свободно распространяемой средой Free Pascal, которая имеется в сети Internet. Для получения дистрибутива можно, например, использовать адрес <http://www.ru.freepascal.org>. Среда Free Pascal имеет интерфейс, очень похожий на Borland Pascal, однако предназначена для современных операционных систем семейств Windows или UNIX. Входной язык очень похож на

Turbo Pascal, во всяком случае, разработчики системы говорят о совместимости с Turbo Pascal 7.0 (но не с Delphi).

При работе со второй частью книги необходимо добиться полного понимания и уверенных навыков программирования. Рекомендуется выполнить как можно больше заданий, которые приводятся в конце каждой главы. Они обычно расположены в порядке возрастания сложности.

Только после этого можно переходить к третьей части, посвященной вопросам структурного и модульного программирования. Главы этой части помогут выработать хороший стиль программирования, умение грамотно структурировать программы. Эти навыки окажутся полезными в дальнейшем при изучении других языков программирования. Главу 10, посвященную процедурам и функциям, возможно, придется перечитать несколько раз, пока не будут понятны все моменты, иначе неизбежны проблемы с усвоением дальнейшего материала.

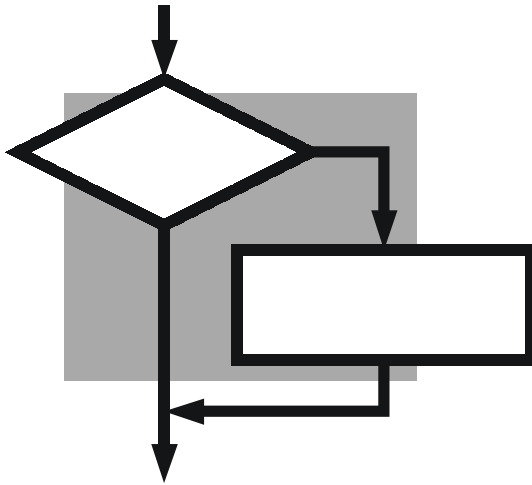
В четвертой части детально разбираются основные структуры данных и алгоритмы их обработки. Ее последние главы, посвященные динамическим структурам данных, абстрактным типам и объектно-ориентированному программированию, не просты для начинающих, т. к. требуют хорошо сформированного алгоритмического мышления. Авторы старались изложить данный материал настолько доходчиво, насколько это возможно для таких сложных абстрактных понятий. Опыт преподавания показывает, что все эти вопросы под силу не только студентам младших курсов, но и развитым ученикам старших классов.

Приложения, завершающие книгу, содержат краткую информацию о компиляторах и системах программирования, работающих на основе Pascal, перечень и описание стандартных модулей Turbo Pascal с дополнениями к ним, а также практическое руководство по работе в интегрированной среде Turbo Pascal.

Таким образом, двигаясь "шаг за шагом", читатель проходит путь от простейших программ до современных концепций и технологий программирования.

По мере накопления знаний и опыта рекомендуется снова возвратиться к более серьезному изучению материала первой части. При повторном чтении читатель посмотрит на этот материал другими глазами — глазами программиста, уже имеющего опыт практической работы, но нуждающегося в систематизации и обобщении знаний.

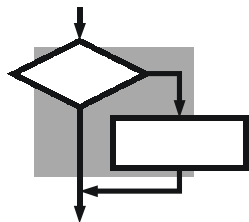
Если все теоретические вопросы и понятия программирования, изложенные в книге, покажутся понятными, интересными и подтвержденными практикой, значит цель авторов достигнута.



# **ЧАСТЬ I**

## **ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ. БАЗОВЫЕ ПОНЯТИЯ**

# Глава 1



## Основы алгоритмизации

### 1.1. Алгоритмизация и требования к алгоритму

Несмотря на развитие новых технологий разработки программ, наиболее важными и фундаментальными понятиями программирования по-прежнему являются *алгоритм* и *алгоритмизация*. По сути дела, решение любой задачи на компьютере можно представить как процесс разработки, описания и выполнения алгоритма.

*Алгоритм* — подробное описание последовательности действий, расположенных в строгом логическом порядке, которое позволяет решить конкретную задачу. Составление такого пошагового описания процесса решения задачи называется ее *алгоритмизацией*. Слово "алгоритм", по существу, является синонимом таких слов, как "способ", "рецепт" и т. п.

Алгоритм предназначен для конкретного *исполнителя*, это может быть не только компьютер, но и робот, станок с программным управлением, наконец, человек. Элементарные действия (шаги), понятные исполнителю, на которые разбивается алгоритм, называются *инструкциями* или *командами*. Другими словами, алгоритм — конечная последовательность инструкций, которая должна быть выполнена исполнителем для решения поставленной задачи.

Уточним данное определение перечислением требований, предъявляемых к алгоритму:

- однозначность* — недопустимы инструкции, которые имеют неопределенное или неоднозначное толкование;
- массовость* — пригодность алгоритма для решения не только данной задачи, а и множества родственных задач, относящихся к общему классу;
- детерминированность* — повтор результата при повторе исходных данных;
- корректность* — способность алгоритма давать правильные результаты решения задачи при различных исходных данных;



□ *конечность* — решение задачи должно быть получено за конечное число шагов алгоритма, "заикливание" недопустимо.

Дополнительным требованием является *эффективность* — для успешного решения задачи должны использоваться ограниченные ресурсы (время работы процессора, объем оперативной памяти, быстродействие жесткого диска и др.).

## 1.2. Способы записи алгоритмов

Разработка алгоритма решения задачи — сложный творческий процесс. Записать алгоритм в виде компьютерной программы без каких-либо предварительных рассуждений может только опытный программист при решении небольшой по объему, четко поставленной задачи. В реальной жизни такие задачи встречаются редко, поэтому обычно разработчик сначала продумывает алгоритм и записывает его в какой-либо удобной форме, а затем реализует алгоритм в виде программы.

При разработке сложных коммерческих программных продуктов часто алгоритмизацию выполняет один человек, а запись (кодирование) алгоритма на языке программирования — другой. Следовательно, необходимо иметь такие методы записи алгоритмов, которые легко воспринимаются человеком, но являются достаточно строгими, чтобы их было нетрудно реализовать на каком-либо языке программирования или, иначе говоря, перевести на язык, понятный компьютеру.

Существуют различные варианты записи алгоритмов. К основным вариантам относятся описательный и графический способы. *Описательным (словесным)* называется способ записи алгоритма на естественном языке (русском, английском и т. д.) или с использованием специального *псевдоязыка*. *Графический* способ отличает компактная и наглядная форма записи в виде специальных графических знаков с указанием связи между ними.

### 1.2.1. Описательный способ

#### Запись алгоритма на естественном языке

Это наименее формализованный способ записи, который, в основном, используется для пояснения сути алгоритма и предназначен для чтения или исполнения человеком. При этом степень детализации алгоритма произвольно выбирается разработчиком. В литературе по программированию такой способ применяется очень часто. В данной книге словесный способ, как наиболее удобный для восприятия, будет использоваться для пояснений текстов программ, которые также называются *листингами*.

*Обратите внимание*, что, например, книгу кулинарных рецептов можно рассматривать как своего рода сборник "гастрономических" алгоритмов, запи-

санных на естественном языке. Таким образом, слово "алгоритм" зачастую выступает как синоним слов "способ", "рецепт" и т. п.

## Использование псевдоязыков

Алгоритм, записанный на естественном языке, как правило, недостаточно формализован и может содержать какие-либо неясности или допускать неоднозначное толкование. Желание получить более формальный и вместе с тем наглядный способ записи алгоритма послужило толчком к разработке псевдоязыков.

*Псевдоязык* предполагает сочетание элементов естественного языка и языка программирования высокого уровня. Обычно предложения естественного языка встраиваются в конструкции формального языка. Получается достаточно строгое описание алгоритма, и вместе с тем оно обычно короче и легче для восприятия, чем текст программы, записанный на языке высокого уровня.

Хорошим примером псевдоязыка является школьный алгоритмический язык, разработанный А. П. Ершовым. К сожалению, не существует никаких стандартных псевдоязыков, поэтому авторы книг по программированию и алгоритмизации, использующие их для записи алгоритмов, вынуждены предварительно давать описание "своего" псевдоязыка. В настоящей книге авторы не будут применять указанный способ и ограничатся небольшим примером записи алгоритма на псевдоязыке, близком к школьному алгоритмическому (см. разд. 1.2.3).

### 1.2.2. Блок-схемы алгоритмов

*Блок-схема* — это графическое изображение алгоритма в виде плоских геометрических фигур (блоков), соединенных линиями. Внутри блока записывается действие, которое необходимо выполнить, или условие, которое необходимо проверить. Блок-схема — стандартный способ записи алгоритма. Существует государственный стандарт, содержащий перечень правил построения блок-схем — ГОСТ 19.701—90 (ИСО 5807—85). Дата введения стандарта: 01.01.1992 г.

Основные блоки, которые используются при графической записи алгоритма, изображены на рис. 1.1, где  $a$  — начало (конец) алгоритма;  $b$  — блок ввода/вывода;  $v$  — операционный (вычислительный) блок;  $z$  — логический (условный) блок;  $d$ — $ж$  — циклы.

Операционный блок служит для описания действий, логический — для проверки условий. Используя блоки  $a$ — $z$ , можно описать практически любой алгоритм. Однако для изображения повторяющихся действий (циклов) ГОСТом предлагаются дополнительные блоки.

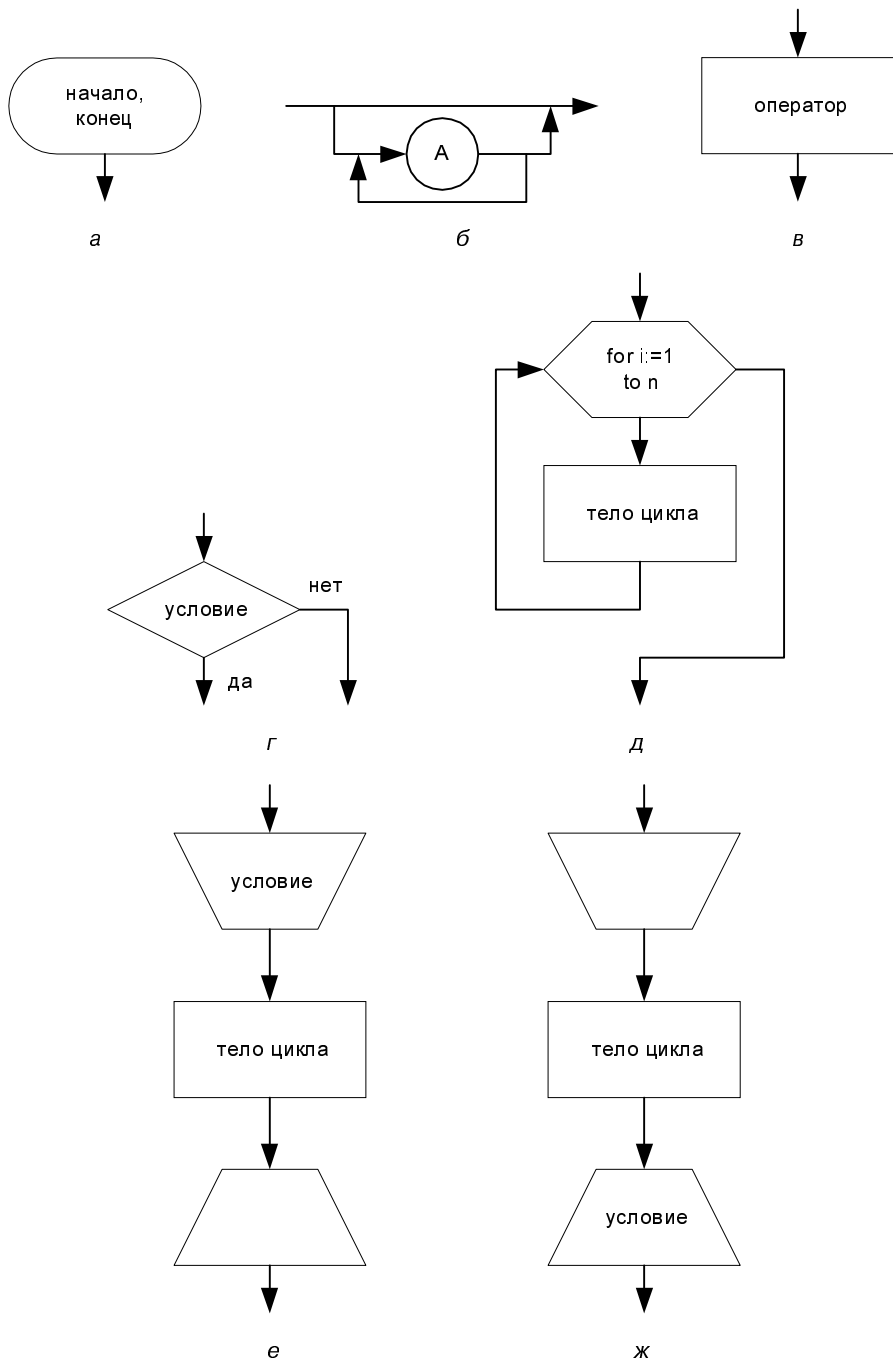


Рис. 1.1. Условные обозначения на схемах алгоритмов

Конструкция  $\delta$  специально не регламентируется ГОСТом для использования при изображении циклов, однако приведенное обозначение (шестиугольник) разрешено использовать для различных целей, поэтому такое изображение цикла на блок-схеме часто встречается в литературе по программированию. И наоборот, конструкции  $\epsilon$  и  $\zeta$  в литературе практически не встречаются. Авторам они также представляются неудачными. Чаще всего при изображении циклов используются логические и операционные блоки (см. рис. 1.2).

### 1.2.3. Пример блок-схемы алгоритма

В качестве примера рассмотрим различные способы записи алгоритма игры "Угадай число". Условие игры: игрок должен угадать число, "задуманное" компьютером, — случайное число в диапазоне от 0 до 1000.

#### Запись алгоритма на естественном языке

Компьютер задумывает случайное число  $A$ . В свою очередь, игрок вводит число  $B$ , пытаясь угадать значение  $A$ .

Выполняется проверка: число  $B$  строго больше числа  $A$ ?

Если это так, т. е. "да", то компьютер выводит на экран сообщение "много" и просит игрока ввести следующее число, еще раз попытав удачи. Если же результат — "нет", то компьютер выполняет следующую проверку:

$B$  строго меньше  $A$ ? Если ее результат — "да", то выводится сообщение "мало", и компьютер ожидает ввода игроком следующего числа.

Если же результат второй проверки — "нет", это означает, что игрок угадал число, задуманное компьютером. В таком случае выводится сообщение "вы угадали", и игра завершается.

#### Запись на псевдоязыке

Алгоритм Угадай Число

Начало

$A :=$  случайное число

Повторяй

Ввод  $B$

Если  $B > A$  то вывод "много"

Иначе

Если  $A > B$  то вывод "мало"

Иначе

вывод "вы угадали"

Все

До  $A=B$

Конец

## Блок-схема алгоритма

Алгоритм игры представлен на рис. 1.2.

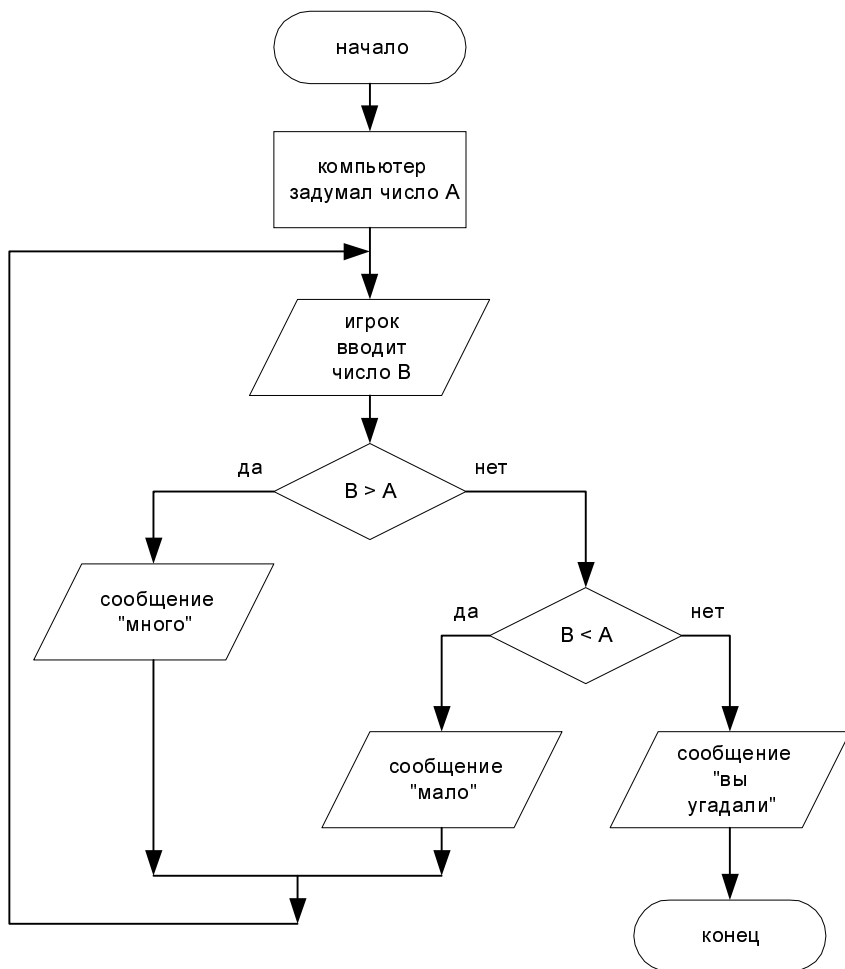


Рис. 1.2. Блок-схема алгоритма игры "Угадай число"

## 1.2.4. Следование, ветвление, цикл

Алгоритмические структуры (рис. 1.1, *а, б, в*) образуют линейную последовательность операций, которые выполняются по очереди в порядке записи — *следование*. Программную реализацию такой алгоритмической структуры называют *линейной программой*. Линейные программы обычно предназначены для решения простейших задач, в которых не предусмотрены выбор из нескольких возможных направлений хода программы или циклическое повторение операций.

Возможность альтернативного выбора при выполнении программы представляют *ветвления* (рис. 1.1, *г*), при выполнении которых алгоритм может пойти по одной из двух возможных ветвей в зависимости от справедливости проверяемого условия. Иногда выделяют также *обход*, который представляет собой пропуск нескольких шагов алгоритма при выполнении или невыполнении какого-либо условия.

*Цикл* (рис. 1.1, *д, е, ж*) представляет собой многократно повторяющуюся последовательность шагов алгоритма. Очень часто цикл также изображают, используя логический блок для проверки условия его завершения (такой способ был использован и в алгоритме игры "Угадай число").

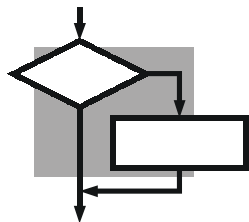
## 1.3. Контрольные вопросы и задания

1. Дайте определение алгоритма. Каким требованиям он должен удовлетворять?
2. Что такое алгоритмизация?
3. Для кого предназначен готовый алгоритм?
4. Какие варианты записи алгоритмов вы знаете? Дайте их сравнительную характеристику.
5. Выполните следующее задание.

□ Составьте и запишите блок-схемы алгоритмов:

- вычисления суммы двух целых чисел;
- деления двух чисел столбиком (нахождение частного и остатка);
- деления отрезка пополам с помощью циркуля и линейки;
- перехода улицы через переход, оборудованный светофором;
- приготовления десяти одинаковых бутербродов с сыром.

## Глава 2



# Язык программирования. Программа

## 2.1. Язык программирования

Алгоритм, представленный в виде блок-схемы или в словесной форме (даже с использованием псевдоязыка), не пригоден для исполнения *процессором* какого-либо вычислительного устройства (компьютера, робота и др.). В этом случае нужен абсолютно формальный способ записи, не допускающий даже малейшей неточности. Следовательно, нужен особый язык, который позволяет записывать алгоритмы с использованием ограниченного набора конструкций, не допускающих неоднозначного толкования. Такие языки называются языками программирования.

Таким образом, *язык программирования* — совокупность средств и правил представления алгоритма в виде, пригодном для выполнения вычислительной машиной.

Рассмотрим кратко историю языков программирования.

### 2.1.1. Исторический обзор

#### Машинный язык

На раннем этапе развития вычислительной техники программы писались на машинном языке — в *машинных кодах*, т. е. так, как их воспринимает процессор компьютера или другого цифрового устройства. Для каждой элементарной операции в этом случае требуется указать ее код и адреса ячеек памяти с данными. Запись выполняется в цифровом виде с использованием двоичной системы счисления. Основными неудобствами такого способа программирования являются следующие:

- исторически сложилось так, что имеется много типов процессоров, отличающихся друг от друга *архитектурой* (устройством) и *системой команд* (набором допустимых инструкций). В результате программа на *машинном языке* годится только для исполнения тем процессором, для которого она написана;

□ программу на машинном языке трудно читать даже профессионалу. В такой программе тяжело находить ошибки. Если объем программы превышает критический, программу практически невозможно полностью отладить. Даже если программа доведена до уровня, при котором она отвечает поставленной задаче, малейшие изменения могут вызвать непреодолимые трудности.

В настоящее время машинный язык используется только в исключительных случаях для программирования специальных задач с использованием специализированных вычислительных устройств.

## Ассемблер

Первоначальный прогресс в технологии программирования был связан с идеей использования символьных имен (названий) вместо цифровых кодов операций и адресов данных. Язык записи команд, основанный на этой идее, получил название языка Ассемблера. Использование осмысленных названий вместо кодов операций и адресов памяти существенно упрощает процесс программирования и внесения изменений в программу. К сожалению, основной недостаток машинного языка — зависимость программы от конкретного типа процессора — при переходе к Ассемблеру не был устранен, но *это был только первый шаг в правильном направлении.*

Программа, записанная на Ассемблере, не может восприниматься процессором непосредственно (ему нужны только двоичные коды операций и адреса). Следовательно, необходимо предварительное преобразование (перевод) программы с языка Ассемблера на машинный язык. Это делается с помощью специальной программы, называемой *транслятором*, а процесс преобразования программы в машинные коды называется *трансляцией* (см. разд. 3.1). Попутно на транслятор были возложены функции выявления некоторых ошибок в тексте программы, нарушающих правила записи. Такие ошибки называются *синтаксическими*.

## Языки программирования высокого уровня

Следующая ступень развития — это *языки программирования высокого уровня*. Их использование позволяет отвлечься от системы команд конкретного типа процессора. Такой язык содержит правила записи программ, которые, с одной стороны, достаточны и удобны для описания алгоритмов решения задач, а с другой стороны, толкуются однозначно и могут быть преобразованы в программы в машинных кодах.

Задача программиста заключается в том, чтобы подготовить правильный текст на языке программирования, а остальное возьмет на себя транслятор, который прочтет этот текст, проверит его на соответствие правилам языка и сформирует программу на машинном языке. Естественно, транслятор должен "знать" систему команд своего процессора и особенности работы уста-



новленного на компьютер программного обеспечения, его *операционной системы*. Это приводит к необходимости разработки нескольких различных трансляторов для одного и того же языка программирования.

Также понятно, что разработка транслятора с языка высокого уровня — задача более сложная и дорогостоящая, чем разработка транслятора с Ассемблера, а программа в машинных кодах, сформированная самым лучшим транслятором, все равно будет уступать по качеству и быстродействию программе, разработанной программистом-профессионалом на Ассемблере. За удобство программирования приходится платить некоторым снижением эффективности разработанной программы.

Говорят, что языки программирования высокого уровня являются *машинно-независимыми*. В противоположность этому, машинный язык и язык Ассемблера — *машинно-зависимые языки*. Иногда их называют *языками программирования низкого уровня*. Конечно, слова "низкий уровень" вовсе не означают низкое качество этих языков. Так подчеркивается их неразрывная связь с аппаратурой компьютера. Во многих задачах, например, при разработке ядра (основной части) операционных систем, где качество машинного кода ставится выше удобства разработки, без языка Ассемблера вообще не обойтись. Тем не менее, везде, где возможен компромисс, язык высокого уровня предпочитают Ассемблеру.

Возможность создания независимых от системы команд компьютера языков программирования привела к их бурному развитию. В настоящее время насчитывается более сотни самых разнообразных языков программирования высокого уровня. Большинство из них предназначены для применения в конкретных областях (например, языки программирования баз данных, языки программирования для Internet и т. д.). Эта группа языков развивается очень быстро, т. к. сфера применения компьютеров постоянно расширяется и возникает потребность как в новых языках программирования, так и в развитии уже имеющихся.

Вместе с тем имеется довольно устойчивая группа хорошо отработанных, испытанных языков универсального назначения, очень близких по принципам, заложенным в их основу, и по используемым конструкциям. На протяжении десятилетий эти языки постепенно совершенствовались, а заодно сближались, перенимая друг у друга наиболее удачные моменты. Однако и сейчас каждый язык имеет массу неповторимых, только ему присущих особенностей, у каждого есть свои поклонники, предпочитающие именно его. Очевидно, в обозримом будущем языки будут существовать и развиваться параллельно, хотя *тенденция к сближению языков универсального назначения* *налицо*.

Перечислим наиболее популярные языки универсального назначения, соблюдая исторический порядок их появления и указывая их отличительные особенности. Более полный обзор современного состояния языков про-

граммирования можно найти в [20], где рассматриваются и редко используемые языки функционального и логического программирования (Lisp, Prolog и др.).

**Fortran.** Является первым из языков высокого уровня (1954—1958). В то время основной сферой применения вычислительных машин (термин "компьютер" вошел в обиход гораздо позже) были научно-технические расчеты. Для этой цели и был разработан простой и эффективный язык программирования. Базовые принципы, заложенные в язык Fortran, впоследствии легли в основу других языков программирования высокого уровня. Особенно сильно его влияние на Basic. В настоящее время популярность языка Fortran невелика, но его последние версии — Fortran 77 и Fortran 90 — продолжают использоваться в сфере научно-технических и инженерных расчетов.

**Algol.** Опубликован в 1960 г. От языка Fortran отличался значительно более строгими правилами синтаксиса, что позволило выявлять больше ошибок еще на этапе трансляции. Программы, написанные на этом языке, более понятны и выразительны, чем Fortran-программы. Благодаря этим качествам Algol нашел применение в научных кругах, в первую очередь среди специалистов по прикладной математике, теоретической и экспериментальной физике. Является непосредственным предшественником языка Pascal. Очевидно, язык Pascal оказался удачнее своего предшественника, поскольку практически его вытеснил.

**Basic.** Появился на свет в 1964 г. и был задуман как очень простой в изучении язык, который можно использовать как первый язык при обучении программированию. В настоящее время на его основе создан современный язык Visual Basic, используемый как простой и удобный язык широкого назначения, *но не в качестве языка для обучения программированию*. В этой роли он как-то не прижился.

**Pascal.** Язык разработан в 1967 г. швейцарским ученым Никлаусом Виртом *специально для целей обучения*. Автор несколько усовершенствовал правила языка Algol с целью удобства его освоения и применения, оставив неизменными достоинства — простоту и выразительность текста программы в сочетании с удобством отладки. В продвижении языка большая заслуга принадлежит специалистам фирмы Borland, которые разработали целую серию быстродействующих и эффективных трансляторов с Pascal и внесли ряд дополнений в сам язык.

Профессор Вирт не остановился на создании Pascal. На его основе им были разработаны два новых языка программирования — *Modula* (1970) и *Oberon* (1985), представляющие собой дальнейшее развитие Pascal. К сожалению, обстоятельства сложились так, что эти языки не приобрели той популярности, на которую можно было бы рассчитывать. Ученики Вирта сделали многое для популяризации и распространения языков Modula и особенно

Обегон. Ими был разработан новый диалект языка Обегон, названный *Component Pascal* (Компонентный Паскаль). Этим названием подчеркивается неразрывная связь всех языков на основе Pascal. Как в дальнейшем сложится судьба данного языка, пока сказать трудно.

Однако сам Pascal на протяжении многих лет был и остается не только *основным языком для обучения принципам программирования*, но и самым распространенным языком на многочисленных олимпиадах по программированию. На его основе фирмой Borland создана удобная среда разработки Delphi, пользующаяся большой популярностью, особенно в России.

Популярность языка Pascal в нашей стране столь велика, что его название обычно пишут по-русски — Паскаль. Однако авторы книги придерживаются англоязычного обозначения, поскольку названия многочисленных диалектов этого языка (см. разд. 2.1.4) не имеют устойчивых русскоязычных эквивалентов (Free Pascal, GNU Pascal, Object Pascal и т. д.).

C. Язык был создан в 1972 г. в компании Bell Laboratories. При разработке этого языка преследовались совсем другие цели, чем при создании языка Pascal. Язык C задумывался как *инструмент для разработки системного программного обеспечения*, т. е. как промежуточный между языками высокого и низкого уровня. От программ, написанных на этом языке, пытались добиться производительности, близкой к производительности программ на Ассемблере, но в то же время старались сохранить возможность переноса программ с одной *компьютерной платформы* на другую, что характерно для языков высокого уровня. При разработке синтаксиса основных конструкций языка преследовалась цель сделать текст программ как можно лаконичнее.

Разумеется, вопросы простоты изучения языка и удобства разработки программ на нем отошли на второй план, поскольку невозможно сразу удовлетворить всем требованиям. В связи с этим *при обучении программированию рекомендуется начинать с языка Pascal*. Хорошо усвоив основные принципы программирования на языке высокого уровня, можно затем по достоинству оценить лаконичность и эффективность программ, написанных на языке C, а само изучение этого языка уже не вызовет проблем.

C++. Объектно-ориентированная версия языка программирования C — C++ — была разработана в 1980 г. Бьорном Страуструпом в компании Bell Laboratories. Сегодня этот язык является одним из наиболее популярных среди профессионалов. Достаточно, например, сказать, что с его помощью была создана операционная система *Windows*.

Развитие языков C/C++ продолжается и в настоящее время. Появляются новые, молодые языки, авторы которых берут за основу наиболее удачные конструкции C/C++, убирая тяжеловесные или устаревшие элементы. Наиболее популярными современными языками программирования на основе C++ являются *Java* и *C#* (читается "Си шарп"). Оба эти языка не очень объемны, но достаточно стройны и удобны в освоении и использовании. К со-

жалению, программы на Java или C# пока уступают по скорости выполнения программам на C, что снижает возможности их использования в качестве языков системного программирования. Однако эти языки с успехом используются в сфере программирования для Internet.

**Ada.** Этот язык был разработан для создания программных систем с многолетним сроком службы и высокой степенью надежности. Ada создан по заказу и состоит на вооружении Министерства обороны США. На сегодняшний день этот язык считается одним из наиболее сложных, однако свои цели вполне оправдывает.

## 2.1.2. Состав языка программирования. Синтаксис и семантика

Обычный разговорный язык состоит из четырех основных элементов: символов (букв), слов, словосочетаний и предложений. Количество символов языка, образующих его алфавит, невелико. Количество слов неизмеримо больше, но все же конечно: все слова языка можно перечислить, например, сведя их в толковый словарь. Все словосочетания, а тем более предложения перечислить уже нельзя, но известны правила, по которым они составляются. Правила русского языка, например, изложены в соответствующих учебниках. *Аналогично устроены все языки программирования.*

### Состав языка программирования

Язык программирования содержит элементы, перечисленные ниже.

- **Символы** — это основные неделимые знаки, из которых составляются все тексты программ на данном языке. Совокупность всех символов образует *алфавит языка*. Алфавит языка программирования несколько шире, чем алфавит естественного языка, и включает обычно латинские буквы, знаки арифметических операций, символы-разделители и ряд других специальных символов.
- **Лексемы** — это неделимые последовательности символов алфавита (элементарные конструкции), имеющие самостоятельный смысл. Они образуются из основных символов языка, так же как слова обычного языка строятся из букв. Возможны лексемы, состоящие из одного символа, например, знаки операций. Нельзя, однако, провести прямую аналогию между лексемами и словами обычного языка. Дело в том, что любой язык программирования, конечно, имеет определенное количество *зарезервированных (ключевых) слов*, которые составляют *словарь* языка, но таких слов неизмеримо меньше, чем в естественном языке (всего несколько десятков). Однако программист может формировать по определенным правилам собственные слова — *идентификаторы*. Из-за этой особенности текст программы на языке высокого уровня воспринимается сложнее,

чем запись алгоритма на естественном языке. Совокупность лексем и правил их формирования образует *лексику языка*.

- *Выражения* строятся из лексем в строгом соответствии с правилами языка. Они задают порядок вычисления некоторого *значения*. Выражения играют в языке программирования ту же роль, что и словосочетания в обычном языке. Еще более близкий аналог выражений — математические формулы.
- *Операторы* (инструкции или команды языка) задают полное описание некоторого действия, которое необходимо выполнить. Это аналог предложения, выражающего законченную мысль, в обычном языке. Для описания сложного действия может потребоваться группа операторов. В этом случае операторы объединяются в *составной оператор* или *блок*.

Действия, заданные операторами, выполняются над *данными*. Предложения языка, в которых даются сведения о данных, называются *описаниями* или *неисполняемыми операторами*.

*Совокупность описаний и операторов языка программирования, реализующая алгоритм решения конкретной задачи, образует программу на данном языке.*

## **Синтаксис и семантика языка**

Система правил записи элементов языка программирования (лексем, выражений, операторов) образует его *синтаксис*.

Смысл конструкций языка — это его *семантика*.

Иными словами, синтаксис языка определяет внешний вид отдельных конструкций ("как они выглядят"), а семантика толкует действия, которые выполняет компьютер по этим конструкциям ("что они делают").

*Современные языки высокого уровня очень схожи по своей семантике, но имеют некоторые непринципиальные различия в синтаксисе.* Отсюда проистекает важный вывод: *начинающим изучать программирование следует обратить внимание именно на семантику языка.* Именно она представляет собой те *базовые знания*, которые не зависят от особенностей конкретного языка. Хорошее знание основ поможет в дальнейшем изучать новые языки программирования в короткие сроки и при этом писать программы хорошего качества независимо от синтаксиса языка.

Справедливости ради следует признать, что изучение правил синтаксиса языка обычно не вызывает затруднений. Понять до конца семантику каждой конструкции гораздо труднее. Еще труднее научиться использовать каждую конструкцию языка по назначению при записи своих собственных алгоритмов. К сожалению (или к счастью), тут можно предложить только один старый испытанный способ — упорную работу за компьютером, разбор многочисленных примеров, решение задач.

### 2.1.3. Описание языка

В отличие от естественных языков, для каждого языка программирования имеется его строгое описание. Зачем нужно такое описание, очевидно понятно, ведь речь идет о языке, предназначенном для общения с компьютером, где малейшая двусмысленность приводит к ошибкам.

Описание языка есть описание его основных элементов, причем описание каждого элемента языка задается его синтаксисом и семантикой. Синтаксические определения устанавливают правила построения элементов языка. Семантика определяет смысл и правила использования тех элементов языка, для которых были даны синтаксические определения.

#### Неформальное описание языка

Для описания языка программирования можно использовать обычный естественный язык. Но такое описание (назовем его неформальным) будет страдать теми же недостатками, что и описание какого-либо алгоритма в словесной форме: излишним многословием при опасности неоднозначного толкования. Правда, в отличие от описания алгоритма, описание языка рассчитано на изучение его человеком (программистом), поэтому в литературе по программированию такое описание применяется довольно часто.

Однако разработчики трансляторов (это тоже программисты, но не *прикладные*, а *системные*) предпочитают иметь более формальное описание языка, т. е. такое, которое не допускает никакого неоднозначного толкования и при этом является компактным и удобным в использовании для профессионалов. В соответствии с их пожеланиями уже для описания одного из первых языков (Algol) был придуман специальный сугубо формальный язык описания.

#### Способы формального описания языка

Сейчас имеется несколько языков описания. Для того чтобы отличать их от языков программирования, которые они описывают, их называют *метаязыками*.

Дадим представление о двух разных метаязыках, которые используются для одних и тех же целей и являются полностью взаимозаменяемыми.

- ❑ **Формы Бэкуса—Наура (БНФ).** Именно этот метаязык был предложен для описания языка Algol, а затем многократно использовался авторами различных языков программирования для формального описания их творений. В настоящее время в основном используется расширенный вариант, который называется РБНФ (расширенные БНФ).
- ❑ **Синтаксические диаграммы Вирта.** Графическое изображение основных конструкций языка.