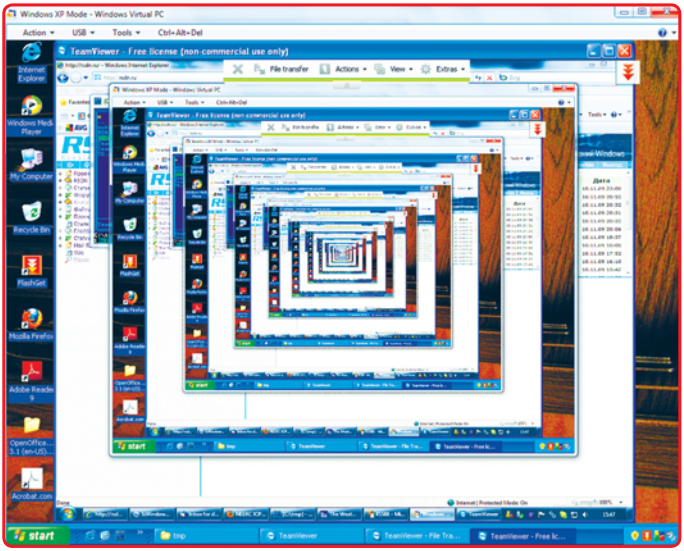
РАЗВИТИЕ ИНТЕЛЛЕКТА ШКОЛЬНИКОВ




С. М. Окулов

# ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАЗВИТИЕ ИНТЕЛЛЕКТА ШКОЛЬНИКОВ

**С. М. Окулов**

# ОСНОВЫ ПРОГРАММИРОВАНИЯ

9-е издание

Москва
Лаборатория знаний

УДК 519.85(023)
ББК 22.18
О-52

*Серия основана в 2008 г.*

**Окулов С. М.**

О-52 Основы программирования / С. М. Окулов. — 9-е изд. — М. : Лаборатория знаний, 2018. — 336 с. : ил. — (Развитие интеллекта школьников).

ISBN 978-5-00101-136-1

В книге рассмотрены фундаментальные положения программирования: конечная величина и конструируемые на ее основе различные типы данных; управляющие конструкции — элементарные составляющие любого алгоритма и основа управления вычислительным процессом; структуризация задач как основополагающий механизм их реализации на компьютере; упорядочение (сортировка) как основа эффективной работы с любыми данными и, наконец, перебор вариантов, как универсальная схема компьютерного решения задач.

Для учащихся старших классов, студентов и учителей информатики.

**УДК 519.85(023)**
**ББК 22.18**

---

*Учебное издание*

Серия: «Развитие интеллекта школьников»

**Окулов** Станислав Михайлович

**ОСНОВЫ ПРОГРАММИРОВАНИЯ**

Редактор *Е. В. Баклашова*
Художник *Н. А. Новак*
Технический редактор *Е. В. Денюкова*. Корректор *Е. Н. Клитина*
Компьютерная верстка: *Е. А. Голубова*

Подписано в печать 03.10.17. Формат 60×90/16.
Усл. печ. л. 21,00. Заказ

Издательство «Лаборатория знаний»
125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: info@pilotLZ.ru, http://www.pilotLZ.ru

---

**ISBN 978-5-00101-136-1**

# Содержание

# Предисловие

Данный учебник является переработанным вариантом книги автора «Основы программирования», которая выходит в издательстве начиная с 2002 года (пять изданий). Но даже не этот момент времени следует считать точкой отсчета. Приведем фрагмент предисловия к книге «Основы программирования».

«Из истории возникновения учебника. В 1988 году перед автором возникла проблема: «чему учить в информатике и как учить». Исходным «багажом» при принятии решения было образование в классическом университете по специальности «прикладная математика» и более чем 15-летний опыт разработки программного обеспечения специализированных вычислительных комплексов в промышленности. Попытки обучения по существующим в то время учебникам, а их было не так много, как в настоящее время, не принесли ни удовлетворения, ни результатов. Информатика воспринималась как обычный предмет. Дидактические возможности компьютера не использовались в полной мере. Может быть, причиной явилось отсутствие у автора соответствующего опыта и педагогического образования. Ограничимся констатацией факта, оставим критику этого результата, так же как и обсуждение достоинств и недостатков существующих учебников, доброжелателям.

Тезисно обозначим исходные положения при принятии решения.

Первой посылкой было убеждение в том, что программирование является «стержнем информатики». В нем синтезировано все, что десятилетиями нарабатывалось в Computer Science. Это и результаты работы специалистов, работающих на «стыке» математики и информатики, это и достижения в вычислительной технике, это и, наконец, огромный опыт формализации и решения сложнейших проблем в самом программировании, связанный с созданием больших программных комплексов.

Второй исходной посылкой является убеждение в том, что занятия по информатике в корне должны отличаться от традиционных занятий по любому другому предмету. Во-первых, на занятиях по информатике должна поощряться ошибка, ибо только через ошибку можно прийти к результату, при изучении же любого другого предмета ошибка карается двойкой. Во-вторых, постоянная обратная связь с обучаемым через компьютер, объективная и лишенная эмоций, — это инструментарий индивидуального и развивающего обучения. В-третьих, стиль мышления у программистов свой, отличающийся от стиля мышления как математика, так и любого другого специалиста. Он настроен, если так можно выразиться, на борьбу с хаосом. Любая сложная программа — это миллионы и более составляющих, движущихся и взаимодействующих. И в результате этого взаимодействия должен получаться определенный результат. Представим себе техническую систему такого уровня сложности...

Итак, за основу обучения следует взять программирование, с максимальным использованием компьютера на занятиях, и при этом должен формироваться определенный стиль мышления. В таком ключе и шла вся последующая работа. Большое влияние на нее оказала подготовка школьников к олимпиадам по информатике. Синтезировались и обобщались все педагогические находки, которые затем находили применение в преподавании информатики и подтверждали обозначенные выше положения.

Основной методический принцип учебника — все познается через труд, через преодоление ошибок (собственных), через процесс решения задач. Этот принцип определяет структуру занятий. Вводная часть – обсуждение нового материала, эксперименты с заготовками решений задач, самостоятельное решение задач.

В идейном плане наиболее близкими к данному учебнику, несмотря на кажущиеся принципиальные отличия, являются учебники А. Г. Кушниренко и Г. В. Лебедева[1]. Эти учебники — наиболее цельные из существующих по идеям, заложенным в них, и, следовательно, по содержанию. Они имеют свое «лицо» и идут к методике преподавания информатики от методики обучения математике. Единственное отличие, на наш

[1] Ныне незаслуженно забытые.

взгляд, заключается в том, что при совпадении идейных установок мы в методике преподавания идем от компьютера, решая проблемы обучения через практическое программирование. При этом компьютер, система программирования являются не самоцелью обучения, а инструментом для реализации целей, хотя при этом познается и сам инструментарий».

И в 2010 году автор ничего не хочет изменять из написанного. Более того, в развитие данного подхода за прошедшие годы, автор пытался ответить на вопрос, что является фундаментальным в информатике, какие понятия являются основополагающими. Если, например, в математике они известны — число и форма, в физике, химии и других областях знания они также определены, то что с информатикой? Этот вопрос — далеко не праздный. Ответ на него определяет многое, в частности судьбу школьного предмета информатики[2]. Точка зрения автора в развернутом виде изложена в его монографии[3], здесь же тезисно обозначим основные положения, кратко изложив один из ее параграфов.

Мы говорим о содержании курса информатики. Фундаментальных понятий должно быть немного, они должны пронизывать все содержание и на каждом витке содержания, образно выражаясь, иметь свою «окраску», свой уровень сложности.

Перечислим эти понятия.

1. Величины, структуры данных (в первую очередь массив). Структуры данных и величины — это тот инструмент, с помощью которого данные о проблеме оформляются таким образом, чтобы она могла быть воспринята и обработана компьютером.

2. Управление вычислительным процессом (управляющие конструкции, рекурсия).

3. Структуризация проблем (процедуры, функции — инструмент реализации принципа «разделяй и властвуй», механизмов абстрагирования, декомпозиции и формализации).

---

[2] Понятие «информатика» — достаточно объемное, включающее и кибернетику, и моделирование, и т. д. В данной книге мы, скорее всего, говорим о Computer Science, об отрасли человеческого знания, охватывающей решения проблем с использованием компьютера, но называем ее, в силу сложившейся традиции отождествления понятий при переводе, информатикой.

[3] Окулов С. М. Информатика: развитие интеллекта школьника. 2-е изд. – М.: БИНОМ. Лаборатория знаний, 2008.

4. Отношение порядка (упорядоченности) на множестве объектов определенной структуры.

5. Перебор вариантов в пространстве состояний задачи.

Итак, только пять, и выбор их не случаен. Коротко обоснуем его.

1. **Структуризация данных.** Фундаментальная основа любого управления есть данные (информация) и действия, совершаемые над данными (с информацией). При их структуризации очевидна аналогия со структурой ЭВМ (с ее процессором и памятью, организованной по принципу массива).

2. **Управляющие конструкции.** Перечень управляющих конструкций полный: любое алгоритмическое действие, вероятно, в любой области деятельности, может быть представлено с использованием только следования, ветвления и повторения. В классической работе[4] показано, что этих конструкций достаточно для реализации любого алгоритма.

Структуры данных и управляющие конструкции позволяют в компактной форме записать большое количество действий и обозначить большие объемы информации.

3. **Структуризация задач.** Нельзя объять необъятное. Человек при решении сложной проблемы всегда пытается выделить главное, расчленить ее на более простые. Такова природа человеческого интеллекта. По данным когнитивных психологов, человек может следить не более чем за семью непрерывно меняющимися во времени величинами, эффективно работать не более чем с пятью–семью людьми. Инструментами этого разъятия (разделения и властвования) являются процедуры и функции. Причем этот процесс — нелинейный. Если на каком-то этапе дальнейшее продвижение невозможно (из разъятого не удается методами синтеза создать гармонию), то человек возвращается в своих действиях назад и повторяет на новом витке, в новом варианте последовательный анализ проблемы.

4. **Отношение порядка** (упорядоченности) на множестве объектов определенной структуры. Множество ячеек оперативного запоминающего устройства упорядочено, множество регистров процессора имеют свои имена, множество дорожек магнитного диска упорядочено, множество электронных адре-

[4] Bohm C., Jacopini G. Flow Diagrams Turing Machines, and Languages with Only Two Formulation Rules//Communicatins of the ASM. 1966. May.

сов в сети Интернет связано отношениями порядка и т. д. Это один аспект — связь с нижним уровнем компьютера. С другой стороны, упорядоченность воспринимается как что-то естественное и для человека, на что даже не стоит специально обращать внимание. Действительно, данное понятие формируется на стадии конкретных операций (Ж. Пиаже), а это возраст от 7 до 11 лет.

Проблема упорядоченности неразрывно связана с другой проблемой — поиска данных (информации). Упорядоченность упрощенно можно трактовать как необходимое условие быстрого поиска. Проблема эта как была, так и остается одной из ключевых в информатике. Например, поиск общей подпоследовательности максимальной длины в двух текстах. Несмотря на возросшую многократно производительность компьютера, исследованием методов нахождения данных за возможно меньшее время по-прежнему занимаются ведущие ученые отрасли. Достижение, исследование упорядоченности на множестве объектов — очень непростая задача, даже на простых структурах данных. Фундаментальность понятия «упорядоченность» следует из того, что всякое развитие есть, в определенной степени, увеличение упорядоченности в системе.

5. **Перебор вариантов** в пространстве состояний задачи. В принципе универсальная, хотя далеко не всегда реально достижимая схема компьютерного решения задач. Особенность компьютера заключается в том, что он действует только с упорядоченными структурами данных. Только там, где есть отношение порядка, функционируют циклические конструкции, и т. д. — решаются переборные задачи. Только перечисляемое подвластно компьютеру, а в этом и есть сущность перебора вариантов.

Именно эти фундаментальные для информатики понятия и рассматриваются в данной книге. Двадцатипятилетний опыт работы в образовательной информатике позволяет утверждать, что школьник, своевременно их освоивший, может стать (при его желании) успешным специалистом в Computer Science.

# Часть I

# Программирование в среде Паскаль

## 1.1. Основные управляющие конструкции

### Занятие № 1. Первая программа

***План занятия***

1. Краткое знакомство со средой программирования.
2. Экспериментальный раздел занятия.
3. Выполнение самостоятельной работы.

#### Краткое знакомство со средой программирования

После загрузки системы программирования на экране появляются три окна (рис. 1.1):

| | |
|---|---|
| `File Edit ...` | – окно 1 – главное меню |
| `Line 1 Col 1 ...` | – окно 2 – основное или рабочее окно |
| `F1-Help ...` | – окно 3 – окно помощи, в нем указано назначение основных функциональных клавиш |

**Рис. 1.1.** Схематическое изображение структуры экрана

Переход из первого окна во второе и наоборот осуществляется нажатием клавиши F10.

В рабочем окне редактора среды программирования наберем текст первой программы — вычисления произведения двух целых чисел:

```
Program My1_1;
 Var a,b,rez: Integer;
 Begin
  WriteLn('Введите два числа через пробел');
  ReadLn(a,b);
```

```
  rez:=a*b;
  WriteLn('Их произведение равно ',rez);
  WriteLn('Нажмите Enter');
  ReadLn;
End.
```

Программа начинается с **заголовка**, имеющего следующий вид:

```
Program  <имя программы>;
```

Имя нашей программы — My1_1. Заметим, что в имени программы не должно быть пробелов, оно должно начинаться с буквы, состоять только из латинских букв, цифр и некоторых символов, не допускается использование символов точки и запятой.

За заголовком идет **раздел описаний**, в котором должны быть описаны все идентификаторы (константы, переменные, типы, процедуры, функции, метки), которые будут использованы в программе. В данном случае из разделов описаний имеется лишь один — раздел переменных. Он начинается со служебного слова Var, после которого идет последовательность объявления переменных, разделенных точкой с запятой. В каждом объявлении перечисляются через запятую имена переменных (идентификаторы) одного типа, после чего ставится двоеточие и указывается тип переменных. В нашем примере описаны три переменные: *a*, *b* и *rez*; все они имеют целый тип (Integer), т. е. значениями переменных этого типа являются целые числа.

Понятие переменной — центральное в любом языке программирования. Для описания переменной (величины, которая изменяется в процессе работы программы) следует указать имя переменной, ее тип и значение. Соблюдается следующий принцип: использовать переменную можно лишь тогда, когда ей присвоено некоторое значение. Использование в выражениях неинициализированных переменных, то есть переменных, значения которым не были присвоены явно, часто является причиной ошибок.

После раздела описаний идет **раздел операторов**, который начинается со служебного слова Begin и заканчивается служебным словом End. В этом разделе задаются действия над объектами программы, введенными в разделе описаний. Операторы в этом разделе отделяются друг от друга точкой с запятой. После

последнего слова End ставится точка. После слова Begin ни точка, ни точка с запятой не ставятся.

Первый встречающийся оператор — это **WriteLn**('текст'); — записать (вывести) на экран текст, заключенный между апострофами и взятый в скобки. Ln добавляется в конце имени оператора для того, чтобы после вывода на экран текстов или результатов выполнения программы курсор автоматически переходил на следующую строку.

Следующий оператор — это **ReadLn**(a,b); — читать данные с клавиатуры. В данном случае необходимо ввести два целых числа через пробел, тогда переменной *a* присваивается значение, равное первому введенному числу, а переменной *b* присваивается значение, равное второму введенному числу. Например, вы ввели числа 12 и 45, тогда *a*=12, а *b*=45. В конце этого оператора также можно ставить Ln.

После этих двух операторов стоит оператор присвоения: rez:=a*b; (:= — это знак оператора присвоения в языке Паскаль; * — это знак умножения). Значение выражения из правой части оператора присвоения заменяет текущее значение переменной из левой части. Тип значения выражения должен совпадать с типом переменной. При выполнении оператора пе-

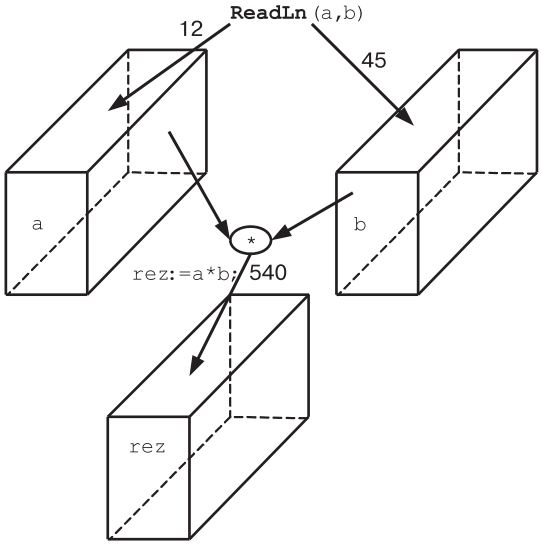



**Рис. 1.2.** Схема выполнения операторов **ReadLn**(a,b) и rez:=a*b

ременная *rez* получит значение, равное произведению числа *a* на число *b* (см. рис. 1.2). Так как в результате умножения двух целых чисел получается целое число, то переменная *rez* описана типом Integer (значениями которого могут быть лишь целые числа).

Следующий оператор — это снова оператор **WriteLn**`('Их произведение равно',rez);` — он выведет на экран текст, заключенный между апострофами, а за ним значение переменной *rez*.

Затем следующий оператор WriteLn выведет на экран сообщение: `Нажмите Enter`, а оператор ReadLn будет ожидать этого нажатия в окне выполнения.

В конце раздела операторов стоит служебное слово End, после которого стоит точка.

**Запуск программы.** Для того чтобы запустить программу, выходим в главное меню (нажатием F10) — первое окно, выбираем режим Run и дважды нажимаем Enter. На экране появляется сообщение:

```
Введите два числа через пробел
```

Курсор мигает в следующей строке, вводим в нее два целых числа через пробел и нажимаем Enter, после этого появляется сообщение:

```
Их произведение равно ...
Нажмите Enter
```

Вместо точек будет написано значение переменной *rez*, то есть число, равное произведению первого введенного числа на второе. Это сообщение будет на экране до тех пор, пока не нажата клавиша Enter.

**Сохранение программы.** Для того чтобы сохранить программу на внешнем носителе, необходимо:

- выйти в главное меню и выбрать режим **File**;
- нажать Enter, из появившегося окна выбрать режим **Save as...** и нажать клавишу Enter;
- в появившемся окне набрать имя файла и нажать клавишу Enter. Например, f:\prim1_1.pas; здесь f:\ – это логическое имя диска, на котором будем сохранять файл, prim1_1 — имя файла (оно может содержать не более 8 символов), pas — расширение, сообщающее о том, что файл содержит программу, написанную на языке Паскаль.

*Примечания*

1. Список символов, которые нельзя употреблять в именах файлов: *, =, +, [, ], \, |, :, ., <, >, /, ?. Также запрещены запятая, пробел и буквы русского алфавита.
2. Для быстрого сохранения файла можно воспользоваться командами **Save** или **Save all** меню **File**.

**Выход из системы программирования.** Для того чтобы закончить работу, необходимо:

- выйти в главное меню и выбрать режим **File**;
- нажать Enter и из появившегося окна выбрать режим **Quit**, после чего нажать либо Enter, либо комбинацию клавиш Alt–X.

### Экспериментальный раздел занятия

**1.** Введите числа при работе программы, например 4567 и 789. Убедитесь, что у вас получается неправдоподобный результат — отрицательное число (–1117). Найдите экспериментальным путем такой интервал значений переменных *a* и *b*, при котором результат умножения правильный.

**2.** Введите вместо числа какой-нибудь символ. Убедитесь, что выполнения программы не произошло, компьютер выдал сообщение об ошибке `Error 106: Invalid numeric format`.

**3.** Добавьте лишний знак апострофа в операторе WriteLn. Убедитесь, что компиляция программы не произошла, а компьютер вас порадовал ошибкой `Error 8: String constant exceeds line`.

**4.** Измените в программе My1_1 оператор `rez:=a*b` на `rez:=a-(a Div b)*b`. Выясните действие операции Div для переменных целого типа. Измените текстовую часть следующего за оператором присвоения оператора WriteLn, отразив в ней результат вашего исследования.

**5.** Добавьте в программу предыдущего примера переменную с именем *ost*, оператор присвоения `ost:=a Mod b` и оператор вывода `WriteLn('???????? ',ost);`. Выясните, какое действие выполняется с помощью оператора Mod. Замените знаки вопроса вашими пояснениями его работы.

**6.** Наберите следующую программу.

```
Program My1_2;
 Var a: Integer;
 Begin
  WriteLn('Введите целое число');
```

```
  ReadLn(a);
  WriteLn('???????? ', Abs(a));
  WriteLn('Нажмите Enter');
  ReadLn;
End.
```

Вашей задачей является замена знаков вопроса в операторе WriteLn на текст, поясняющий работу операции Abs. Выполните аналогичное исследование для операций Sqr(a), Ord(a), Succ(a), Pred(a).

*Примечание*

Рекомендуется следующий порядок работы. Текст программы My1_1 сохраняется под новым именем, например Prim1_2.pas, изменяется в соответствии с заданием и затем вновь сохраняется. Это позволит сократить время на набор программы.

### Задания для самостоятельной работы

1. Измените программу для нахождения суммы двух чисел.
2. Измените программу для нахождения суммы четырех чисел.
3. Найдите значение выражения $(a+(d-12) \cdot 3) \cdot (c-5 \cdot k)$, где значения переменных $a$, $d$, $c$ и $k$ вводятся с клавиатуры.
4. Взяв за образец рис. 1.2, нарисуйте схему, иллюстрирующую выполнение следующего фрагмента программы:

```
x:=13;
y:=25;
t:=x;
x:=y;
y:=t;
```

5. Измените рисунок из предыдущего примера, заменив три последних оператора на:

```
x:=x-y;
y:=x+y;
x:=y-x;
```

6. Напишите программу вывода на экран нескольких чисел в виде

```
13
   14
      15
         16
```

или

```
101
    102
        103
```

### Справочные сведения

**1.** В системе программирования есть встроенный редактор текста — режим работы **Edit** (главное меню). Мы не будем приводить команды управления курсором, вставки, удаления элементов текста и т. д., считая их традиционными и совпадающими с аналогичными командами текстовых процессоров.

**2.** Данные — общее понятие для всего того, с чем работает компьютер. В аппаратуре все данные представляются как последовательности двоичных цифр (разрядов).

В языках высокого уровня, а к ним относится Паскаль, **абстрагируются** от деталей представления данных в памяти компьютера. Любой тип данных определяет множество значений, которые может принимать величина этого типа, и те операции, которые можно применять к величинам этого типа. В Паскале работают с пятью типами данных: простыми, строковыми, составными, ссылочными и процедурными. К простым типам данных относятся целый, вещественный, логический, символьный, перечислимый и ограниченный (два последних определяются пользователем). На простых типах данных, кроме вещественного, определено *отношение порядка*. Что это такое? Все множество значений типа рассматривается как упорядоченное множество, и каждое значение связано с некоторым целым числом, которое есть его порядковый номер. В любом порядковом типе для каждого значения, кроме первого, существует предшествующее значение, и для каждого значения, за исключением последнего, существует последующее значение. Определены следующие стандартные функции для работы с порядковыми типами:

- Ord — возвращает порядковый номер для заданного значения любого порядкового типа.
- Pred — возвращает предшествующее значение для заданного значения порядкового типа (а если заданное значение первое?).

- Succ — возвращает следующее значение для заданного значения порядкового типа (а если заданное значение последнее?).

В Паскале есть возможность **конструировать** (создавать) свои типы данных из имеющихся типов, причем весьма сложные.

Итак, абстрагирование и конструирование суть *концепции типа данных*.

**3.** Каждая программа взаимодействует с окружающей средой с помощью операторов ввода и вывода. Если трактовать термин «программа» достаточно вольно, то можно считать, что любая программа что-то откуда-то берет, что-то делает с введенными данными (преобразует) и затем выводит куда-то полученные результаты. В Паскале связь программы с внешними устройствами осуществляется через имена файлов. Самый простой случай, когда эти имена связаны с клавиатурой (для ввода данных) и с экраном дисплея (для вывода).

Для ввода с клавиатуры используется оператор Read или ReadLn.

---

Вызов: Read($r_1$,$r_2$,...,$r_n$).
Параметры: $r_1$,$r_2$,...,$r_n$ имеют тип Integer, или Real, или Char, или String.
Действие, если $r_i$ имеет тип:

- Integer или Real, то считывается одно число и значение его присваивается переменной $r_i$. При этом знаки пробела и перевода строки перед числом игнорируются;
- Char, то считывается один символ и присваивается переменной $r_i$;
- String, то считывается максимум $q$ символов при длине $q$ строковой переменной $r_i$.

Оператор ReadLn действует так же, как и Read. Отличие в том, что после ввода осуществляется переход на начало следующей строки.

---

Вывод на экран осуществляется с помощью операторов Write или WriteLn.

Вызов: Write($r_1$,$r_2$,...,$r_n$).
Параметры: $r_1$,$r_2$,...,$r_n$ имеют тип Integer, или Real, или Boolean, или Char, или String.
Действие: на экран выводится значение $r_i$ в стандартном формате.
Работа WriteLn отличается тем, что после вывода осуществляется переход на начало следующей строки.

## Занятие № 2. Целый тип данных

***План занятия***

1. Целый тип данных.
2. Разбор программы выделения цифр из десятичного числа.
3. Эксперименты с программами (перевод числа из десятичной системы счисления в двоичную систему счисления; выяснение сути арифметических операций с переменными целого типа; преобразование переменных целого типа).
4. Выполнение самостоятельной работы.

### Целый тип данных

Существует пять целых типов: ShortInt, Integer, LongInt, Byte, Word (табл. 1.1). Они отличаются диапазоном значений, а значит, и размером памяти, отводимой для их представления.

*Таблица 1.1*

| Тип | Диапазон значений | Объем памяти |
|---|---|---|
| ShortInt | –128 ... 127 | 1 байт, со знаком |
| Integer | –32768 ... 32767 | 2 байта, со знаком |
| LongInt | –2147483648 ... 2147483647 | 4 байта, со знаком |
| Byte | 0 ... 255 | 1 байт, без знака |
| Word | 0 ... 65535 | 2 байта, без знака |

Операции с величинами целого типа: сложение (+), вычитание (–), умножение (*), нахождение целой части от деления (Div), нахождение остатка от деления (Mod). Так как целый тип данных относится к типам, на которых определено отношение порядка, то работают стандартные функции Ord, Succ и Pred.

*Примечание*

Переменной целого типа нельзя присваивать значение результата обычной операции деления «/». Убедитесь в этом с помощью простой модификации программы первого занятия. Попробуйте найти объяснение этому факту.

Возникают по крайней мере два достаточно сложных (на этой стадии освоения языка) вопроса:

- Почему при представлении целых чисел со знаком диапазон отрицательных чисел на одно значение больше диапазона положительных чисел?
- Как выполняются операции, например, при вычислении выражений, если все величины имеют разные целые типы?

### Разбор программы выделения цифр из десятичного числа

Перед разбором программы следует рассмотреть выполнение операций Div и Mod (примеры приведены в табл. 1.2).

*Таблица 1.2*

| | |
|---|---|
| 19 Div 4=4 | 19 Mod 4=3 |
| 19 Div –4=–4 | 19 Mod –4=3 |
| –19 Div 4=–4 | –19 Mod 4=–3 |
| –19 Div –4=4 | –19 Mod –4=–3 |

В программе определяются цифры трехзначного числа. Можно ее использовать и для определения цифр двузначного числа, просто цифра сотен в этом случае равна нулю.

```
Program My2_1;
 Var a, one, dec, hun, rez:Integer;
 Begin
  WriteLn('Введите число');
  ReadLn(a);
  one:=a Mod 10;
  WriteLn('Цифра единиц числа - ',one);
  dec:=(a Div 10) Mod 10;
  WriteLn('Цифра десятков числа - ',dec);
  hun:=a Div 100;
  WriteLn('Цифра сотен числа - ',hun);
  rez:=hun*100+dec*10+one;
```

```
 WriteLn('А это тоже число - ',rez);
 Write('Enter ');
 ReadLn;
End.
```

Например, если вы введете число 137, то значение переменной *one* будет равно 7, *dec* – 3 и *hun* – 1. Вспомните деление чисел столбиком.

*Примечание*

Не забудьте сохранить программу под именем prim2_1.pas.

### Экспериментальный раздел занятия

**1.** Измените программу My2_1 для нахождения цифр двузначного числа. Сохраните ее под именем Prim2_2.pas.

**2.** Измените программу My2_1 для нахождения цифр четырехзначного числа. Сохраните ее под именем Prim2_3.pas.

**3.** Деление на 10 и нахождение остатков от деления нам известны. Рассмотрите пример деления столбиком на 2. Остатки от деления в данном случае — или 0, или 1.

```
137|2
  1 68|2
    0 34|2
      0 17|2
        1 8|2
          0 4|2
            0 2|2
              0 1
```

Наберите следующую программу, отладьте ее (найдите и исправьте ошибки) и попробуйте дать объяснение полученному результату. Измените программу так, чтобы она правильно работала, например, с числом 115.

```
Program My2_2;
 Var rez:Integer;
 Begin
  WriteLn('137');
  WriteLn('10001001');
  Rez:=1*128+0*64+0*32+0*16+1*8+0*4+0*2+1*1;
  WriteLn(rez);
  ReadLn;
 End.
```

4. Наберите следующую программу:

```
Program My2_3;
 Uses Crt;
 Var a:Integer; b:Word; r1:Integer; r2:LongInt;
 Begin
```

```
  ClrScr; {Очистка экрана, процедура модуля Crt}
  a:=32000;b:=64000;
  r1:=a+b;
  WriteLn(r1);
  r2:=a+b;
  WriteLn(r2);
  ReadLn;
End.
```

После запуска вы увидите, что значение переменной *r1* равно 30464, а значение переменной *r2* – 96000. Если изменить тип переменной *r1* на Word, то результат не изменится. Используя информацию из табл. 1.1, измените программу так, чтобы проделать аналогичные эксперименты с другими целыми типами. Попробуйте найти логику получения результата компьютером.

*Примечание*

В фигурных скобках записываются комментарии; они не отображаются на экране и не исполняются программой.

5. Добавьте в программу My2_3 перед оператором ReadLn следующие два оператора:

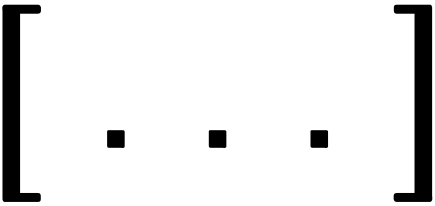
```
WriteLn(LongInt(100*a));
WriteLn(100*LongInt(a));
```

Функция LongInt преобразует переменную типа Integer в тип LongInt. В первом случае преобразование выполняется после умножения, во втором — перед умножением. В первом случае получен результат, далекий от истины: отрицательное число –11264, во втором правильный: 3200000. Приведем основные правила, по которым в Паскале выполняются операции с переменными целых типов.

---

Перед выполнением операций (бинарных) над двумя операндами оба операнда преобразуются к общему для них типу. Им является тип с наименьшим диапазоном, включающим все возможные значения обоих типов. Например, общим типом для Integer и Byte будет Integer, для Integer и Word — LongInt. Результат будет общего типа.

Выражение в правой части оператора присвоения вычисляется независимо от размера или типа переменной в левой части!



## РАЗВИТИЕ ИНТЕЛЛЕКТА ШКОЛЬНИКОВ

**Станислав Михайлович ОКУЛОВ**

Декан факультета информатики Вятского государственного гуманитарного университета, кандидат технических наук, доктор педагогических наук, профессор, почетный работник высшего профессионального образования РФ. Автор 9 изобретений по элементам ассоциативных вычислительных структур и автор (соавтор) 15 книг по информатике для школьников и студентов.

Книги автора, вышедшие в серии «Развитие интеллекта школьников»:

**«Программирование в алгоритмах»;**
**«Ханойские башни»;**
**«Абстрактные типы данных»;**
**«Алгоритмы обработки строк»;**
**«Динамическое программирование».**

Область интересов: развитие интеллектуальных способностей школьника при активном изучении информатики, методика преподавания информатики в школе и вузе.

С 1993 по 2003 год деятельность в вузе совмещал с работой учителя информатики. За это время его ученики отмечены 33 дипломами (1-й и 2-й степени) на российских олимпиадах школьников по информатике; трое из них представляли Россию на международных олимпиадах.

# ОСНОВЫ ПРОГРАММИРОВАНИЯ

В книге рассмотрены фундаментальные положения программирования: конечная величина и конструируемые на ее основе различные типы данных; управляющие конструкции – элементарные составляющие любого алгоритма и основа управления вычислительным процессом; структуризация задач как основополагающий механизм их реализации на компьютере; упорядочение (сортировка) как основа эффективной работы с любыми данными и, наконец, перебор вариантов как универсальная схема компьютерного решения задач.

Для учащихся старших классов,
студентов и учителей информатики.