В. М. Котов А. И. Лапо Е. Н. Войтехович

ИНФОРМАТИКА

Учебное пособие для 7 класса
учреждений общего среднего образования
с русским языком обучения

Допущено
Министерством образования
Республики Беларусь

Минск «Народная асвета» 2017

УДК 004(075.3=161.1)
ББК 32.81я721
К73

Рецензенты:
кафедра информационных технологий в культуре факультета культурологии и социокультурной деятельности учреждения образования «Белорусский государственный университет культуры и искусств» (кандидат физико-математических наук, доцент, заведующий кафедрой *П. В. Гляков*); учитель информатики высшей квалификационной категории государственного учреждения образования «Гимназия № 25 г. Минска» *М. Ю. Симакова*

ISBN 978-985-03-2824-3

СОДЕРЖАНИЕ

От авторов 6

Глава 1. Информация и информационные процессы

§ 1. Информация в жизни человека 8
1.1. Виды информации . . . . . —
1.2. Носители информации 10
1.3. Информационные процессы 11
§ 2. Представление информации в компьютере 14
2.1. Кодирование информации . . . . . —
2.2. Единицы измерения объема информации 16

Глава 2. Представление о логике высказываний. Множества и операции над ними

§ 3. Логика высказываний 19
3.1. Понятие высказывания 20
3.2. Логическая операция **НЕ** 21
§ 4. Логические операции **И** и **ИЛИ** 26
4.1. Логическая операция **И** . . . . . —
4.2. Логическая операция **ИЛИ** 27
§ 5. Множества 31
5.1. Понятие множества . . . . . —
5.2. Понятие подмножества 32
§ 6. Операции над множествами 36
§ 7. Использование логических операций для построения поисковых запросов в Интернете 39
7.1. Поиск информации . . . . . —
7.2. Сокращение области поиска 40
7.3. Использование операторов в поисковых запросах 41

Глава 3. Основные алгоритмические конструкции

§ 8. Алгоритмы и исполнители 44
8.1. Понятие алгоритма . . . . . —
8.2. Исполнитель Чертежник 45
8.3. Алгоритмическая конструкция *следование* 47
8.4. Вспомогательные алгоритмы 48
§ 9. Исполнитель Робот 50
9.1. Роботы в жизни человека . . . . . —
9.2. Среда обитания и система команд исполнителя Робот 52

9.3. Использование алгоритмической конструкции *следование* для исполнителя Робот 54
9.4. Вспомогательные алгоритмы 56
§ 10. Алгоритмическая конструкция *повторение* 61
10.1. Алгоритмы с циклами . . . . . —
10.2. Использование команды цикла с параметром для исполнителя Робот 64
§ 11. Использование условий 68
11.1. Понятие условия . . . . . —
11.2. Цикл с предусловием 70
§ 12. Алгоритмическая конструкция *ветвление* 76
12.1. Команда ветвления . . . . . —
12.2. Составные условия 80
§ 13. Использование основных алгоритмических конструкций для исполнителя Робот 83
§ 14. Язык программирования Паскаль 88
14.1. Команда вывода 89
14.2. Понятие типа данных 91
14.3. Оператор присваивания 93
14.4. Ввод данных 94
14.5. Структура программы 95
§ 15. Организация вычислений 97
15.1. Вычисление значения арифметического выражения 98
15.2. Использование языка программирования для решения задач 99
§ 16. Реализация алгоритмов работы с целочисленными данными 102
16.1. Целочисленный тип данных . . . . . —
16.2. Использование целочисленных данных для решения задач . . 104

Глава 4. Аппаратное и программное обеспечение компьютера

§ 17. Современные компьютерные устройства 108
17.1. Виды компьютеров . . . . . —
17.2. Назначение устройств персонального компьютера 110
§ 18. Операционная система 114
18.1. Основные виды операционных систем . . . . . —
18.2. Элементы графического пользовательского интерфейса 116
18.3. Основные элементы файловой системы 119
18.4. Типовые операции с файлами и папками 121

§ 19. Локальная компьютерная сеть 125
§ 20. Архивация 128
20.1. Программы-архиваторы . . . . . —
20.2. Создание архивов и извлечение файлов из архива 129
§ 21. Программное обеспечение 132
21.1. Классификация программного обеспечения . . . . . —
21.2. Вредоносные программы и способы защиты от них 133

Глава 5. Работа с векторной графикой

§ 22. Понятие векторной графики 137
§ 23. Интерфейс векторного графического редактора Inkscape 142
§ 24. Создание и редактирование векторного изображения 145
24.1. Создание фигур . . . . . —
24.2. Редактирование фигур 149
24.3. Обводка и заливка 150
24.4. Работа с цветом 151
§ 25. Операции над объектами векторного изображения 158
25.1. Копирование, выравнивание и взаимное расположение объектов . . . . . —
25.2. Группировка. Объединение и пересечение объектов 160
§ 26. Работа с текстом 166
Приложение 170

От авторов

Дорогие семиклассники! Вы держите в руках учебное пособие по информатике — предмету, изучение которого позволит вам получить необходимые знания и умения в области информационных технологий, ставших неотъемлемой частью нашей жизни.

В настоящее время исследования по информатике особенно востребованны и актуальны. Роль этой дисциплины в условиях современного мира все более возрастает. Мы, авторы учебного пособия, постарались сделать так, чтобы изучение информатики было для вас интересным и увлекательным. Надеемся, что полученные знания вы сможете применить для решения практических задач из различных предметных областей.

Материал учебного пособия разделен на две колонки. Цвет фона поможет вам разобраться в назначении размещенной на этом фоне информации:

— основные материалы, обязательные для изучения;

— примеры, иллюстрирующие основные материалы;

— определения основных понятий;

— исторические сведения, информация об ученых, внесших вклад в развитие информатики, и другие интересные факты.

В учебном пособии используются следующие условные обозначения:

 — вопросы и задания для проверки знаний;

 — раздел «Упражнения» содержит задания, при выполнении которых используется компьютер;

 — раздел «Упражнения» содержит задания для выполнения в тетради;

 — раздел «Упражнения» содержит задания, при выполнении которых может быть использована информация, размещенная на Национальном образовательном портале;

* — задание или пример для любознательных.

В тексте некоторых заданий вам будет предложено открыть файл. Это означает, что задание можно выполнить, используя файл, размещенный на Национальном образовательном портале («Электронное обучение» → «Электронные образовательные ресурсы» → «Информатика» → «Информатика. 7 класс»). Зайти на портал и скачать файлы к упражнениям можно по ссылке http://e-vedy.adu.by или с помощью матричного QR-кода:

Имя файла для скачивания содержит номер параграфа и номер задания из упражнения после этого параграфа. Например, имя файла upr4_3 означает, что файл относится к третьему заданию из упражнения после четвертого параграфа. Также на портале размещены файлы с программами, рассмотренными в примерах. Такие файлы имеют имя pr8_3.pas (программа для примера 8.3).

Данное учебное пособие имеет электронное приложение (режим доступа: http://informatika7.adu.by).

Глава 1
ИНФОРМАЦИЯ И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

§ 1. Информация в жизни человека

Понятие «информация» имеет множество определений.

Советский ученый в области общей механики и прикладной математики академик Н. Н. Моисеев (1917—2000) считал, что «в силу широты этого понятия нет и не может быть строгого и достаточно универсального определения информации».

Норберт Винер (1894—1964), американский математик и философ, основоположник кибернетики и теории искусственного интеллекта, говорил: «Информация — это не материя и не энергия, информация — это информация».

1.1. Виды информации

Каждый из нас не раз слышал слово «информация». Информацию мы получаем из книг и газет, из Интернета, от людей, с которыми общаемся. А что же означает данное понятие?

Информация — сведения о предметах, событиях, явлениях и процессах окружающего мира.

Большая часть сведений, на основании которых формируется представление человека о мире, поступает к нему благодаря органам чувств. Наличие зрения, слуха, осязания, вкуса и обоняния позволяет нам получать знания об окружающей действительности. В зависимости от того, с помощью каких органов чувств информация поступила к человеку, ее классифицируют по способу восприятия (пример 1.1).

• **Визуальная информация** воспринимается органами зрения (глазами), различающими фор-

му, объем, цвет, перемещение и изменение объектов.

• Органы слуха (уши) воспринимают **звуковую информацию.** С их помощью можно распознавать звуки, различать их тембр, высоту, ритм.

• **Тактильная информация** связана с органами осязания, позволяющими на ощупь определить характер поверхности, ее температуру, почувствовать прикосновение.

• С помощью органов вкуса человек получает **вкусовую информацию** о пище: горькая, сладкая, кислая, соленая.

• Орган обоняния (нос) воспринимает и распознает **информацию о запахах.**

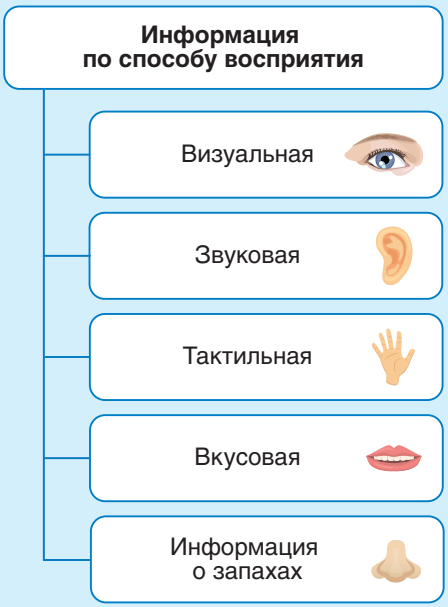
Существует также классификация информации по форме представления (пример 1.2).

• **Графическая информация** — сведения в виде рисунков, схем, фотографий.

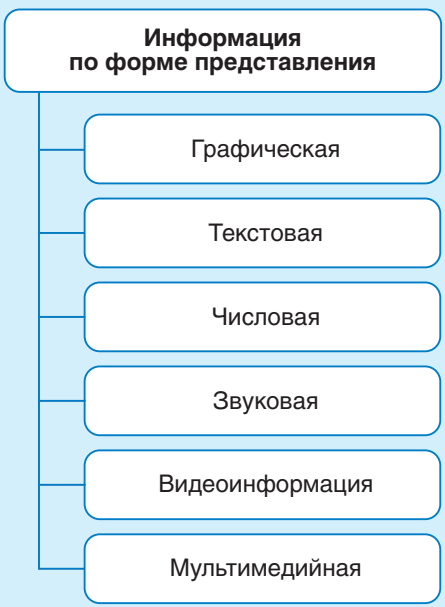
• **Текстовая информация** — сведения в виде специальных символов (букв различных алфавитов; иероглифов, с помощью которых записывают отдельные слоги или слова).

• **Числовая (цифровая) информация** — сведения, отражающие

Пример 1.1.



Пример 1.2.



Помимо классификации информации по способу восприятия и по форме представления, существуют также и другие классификации. К примеру, по сфере возникновения информацию можно разделить на следующие группы:

- механическая;
- биологическая;
- социальная.

Механическая информация отражает процессы и явления неодушевленной природы.

Сфера возникновения биологической информации связана с процессами животного и растительного мира.

Социальная информация отражает процессы человеческого общества.

Пример 1.3. Примеры древних носителей информации.

Восковые таблички

Пергамент

Глиняные таблички

количественную меру объектов и их свойств с помощью чисел и цифр.

- **Звуковая информация** — сведения в виде звуков.

Существуют и комбинированные виды информации — **видеоинформация** и **мультимедийная информация.**

Один и тот же вид информации может поступать и храниться в различной форме. Например, музыкальное произведение может храниться в виде аудио- или нотной записи.

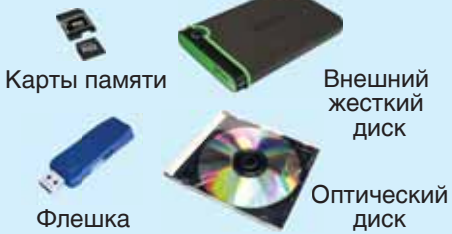
Для преобразования информации из одного формата в другой используются различные алгоритмы и устройства.

1.2. Носители информации

Для записи, хранения и считывания информации используются носители информации. В древности человек сохранял важные сведения лишь в собственной памяти, т. е. память человека является естественным носителем информации. Потребность запоминать и хранить постоянно возрастающие объемы информации привела к использованию и созданию различных материалов и устройств.

К носителям информации относятся бумага, книга, фотография, оптические диски, флеш-память и др. (примеры 1.3 и 1.4). Для обработки, хранения и распространения информации строят специальные здания — дата-центры.

Пример 1.4. Компьютерные носители информации.

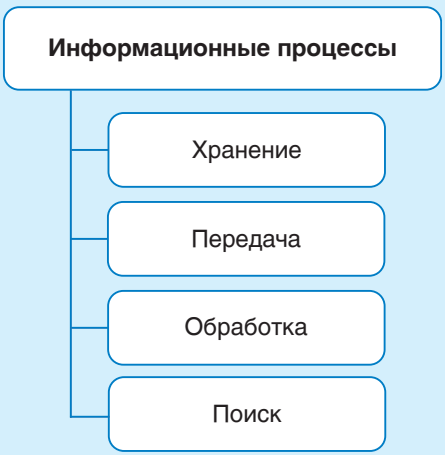


1.3. Информационные процессы

В повседневной жизни мы записываем, запоминаем и считываем полученную информацию. Человек может поделиться известной ему информацией с другими людьми. Кроме того, на основе обработки уже имеющейся информации можно создавать новую.

Любая деятельность человека, связанная с информацией, является **информационным процессом.** Различают следующие информационные процессы: **хранение, передача, обработка, поиск** информации (пример 1.5).

Пример 1.5.



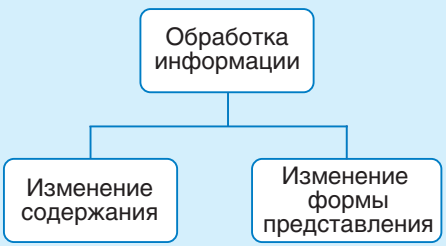
Пример. 1.6.

Люди хранят информацию либо в собственной памяти, либо на каких-либо внешних носителях (пример 1.6). Иногда человек забывает информацию, а информация на внешних носителях

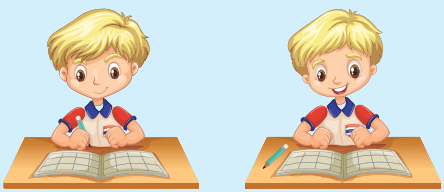
Пример. 1.7.

Передача информации

Пример. 1.8.



Пример. 1.9.



$2x + 3 = 7$

$2x = 7 - 3$

$2x = 4$

$x = \frac{4}{2}$

$x = 2$

может храниться долго и быть доступна разным людям.

Передача информации происходит, к примеру, при разговоре двух людей, при чтении книги или журнала, при просмотре страниц в сети Интернет и др.

В процессе передачи всегда участвуют две стороны: **источник** и **приемник**. Передача информации происходит через **канал связи**: звуковые волны при непосредственном разговоре, услуги почтового сервиса при переписке, сотовая связь при разговоре по мобильному телефону (пример 1.7).

В результате изменения содержания или формы представления информации происходит ее обработка (пример 1.8).

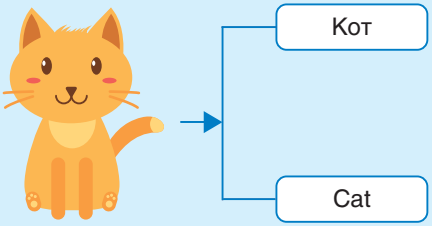
Содержание информации может измениться в результате вычислительных действий, допустим, при решении любой математической задачи, уравнения (пример 1.9). Процесс рассуждений человека также может приводить к появлению новой информации. Новая информация возникает при исследовании явлений природы, физических процессов и др.

Форма представления информации изменяется при рисовании картин по текстовому описанию

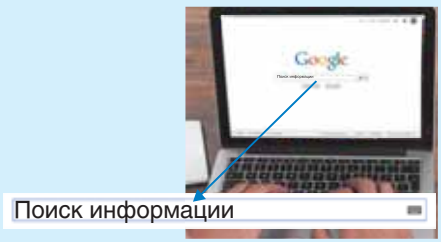
или при описании сюжета видеофильма в виде статьи в журнале, при переводе текста с одного языка на другой (пример 1.10) и т. д.

Если человеку необходимо найти интересующие его сведения, то он осуществляет поиск информации. Для поиска нужной информации используются разнообразные способы и методы: чтение энциклопедий, словарей, книг и журналов, просмотр видеофильмов и телевизионных передач, поиск в сети Интернет (пример 1.11) и т. д.

Пример. 1.10.



Пример. 1.11.



1. Что понимается под информацией?
2. Как можно классифицировать информацию по способу ее восприятия человеком?
3. На какие виды разделяют информацию по форме представления?
4. Что такое носитель информации?
5. Какие информационные процессы выполняет человек?

Упражнения

1 Какие информационные процессы выполняет семиклассник при следующих видах деятельности?

1. Запись в конспект материалов урока.
2. Устный ответ у доски.
3. Перевод текста с русского языка на английский.
4. Подбор материалов для реферата по истории.

2 Приведите примеры профессий, в которых основная деятельность специалиста связана с информацией.

3 Опишите ситуации, в которых вы можете играть роль источника информации; приемника информации. Каким каналом связи вы пользуетесь при этом?

4 Подготовьте сообщение на одну из следующих тем:

1. «Древние носители информации».
2. «Электронные носители информации».
3*. «Дополненная реальность как форма представления информации».

§ 2. Представление информации в компьютере

2.1. Кодирование информации

Для общения люди используют естественный язык, например белорусский или русский. В основе естественного языка лежит алфавит — система графических знаков для передачи звуков устной речи. Алфавит естественного языка является универсальным кодом любой письменной культуры.

Кроме естественных, человек использует искусственно созданные языки со своими особыми кодами: язык математических или химических формул, ноты и др.

Код — совокупность условных знаков, каждому из которых присваивается определенное значение (примеры 2.1 и 2.2).

Процесс записи или преобразования информации в соответствии с правилами, заданными некоторым кодом, называют **кодированием**. Процесс, обратный кодированию, называют **декодированием**.

Кодирование информации использовалось с древности. Широко известен шифр Юлия Цезаря, применявшийся для записи секретных сообщений. Каждый символ в тексте заменялся символом, находящимся на некотором постоянном расстоянии левее или правее его в алфавите.

Например, при кодировании информации с помощью букв русского алфавита путем сдвига вправо на 3 буква «А» была бы заменена на «Г», буква «Б» станет «Д» и т. д.

Пример 2.1. Сегодня широко применяются штрих-коды на различных товарах. Перед вами штрих-код сгущенного молока:

Кодировать и передавать информацию можно различными способами: устно, письменно, жестами и др. Компьютер может обрабатывать числовую, текстовую, графическую и звуковую информацию только в цифровом формате, который в компьютере представлен в виде двоичного кода.

Двоичный код — способ кодирования, в котором каждый **разряд** принимает одно из двух возможных значений, обычно обозначаемых цифрами 0 и 1. **Разряд** в этом случае называется **двоичным разрядом.**

Такой способ кодирования связан с тем, что проще всего реализуются технические устройства, обладающие двумя устойчивыми состояниями: включено/выключено, соединено/разъединено и др.

Для кодирования числовой информации в компьютере вместо десятичной системы счисления используется двоичная, основанная на двоичном коде.

Кодирование текстовой информации в компьютере выполняется при помощи специальных кодовых таблиц, где каждому символу ставится в соответствие определенная последовательность из нулей и единиц (пример 2.3).

Пример 2.2. С появлением смартфонов начали распространяться QR-коды. Они позволяют быстро заносить в телефон текстовую информацию, добавлять контакты в адресную книгу, переходить по web-ссылкам, отправлять SMS-сообщения и т. д. Вот, например, QR-код со ссылкой на статью в Wikipedia о QR-кодах:

Пример 2.3. Кодирование некоторых букв английского алфавита на компьютере.

Буква	Двоичный код
A	01000001
B	01000010
C	01000011
D	01000100
E	01000101
F	01000110
G	01000111
H	01001000
I	01001001
J	01001010
K	01001011
L	01001100
M	01001101
N	01001110

Пример 2.4.

2.2. Единицы измерения объема информации

Человек применяет различные единицы измерения. Так, для измерения времени используются секунды, минуты, часы, для измерения расстояния — метры, километры и др. Измерения проводят с помощью измерительных приборов (пример 2.4).

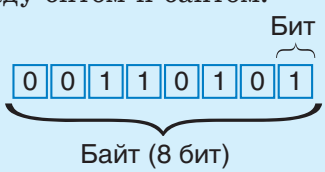
Для определения количества информации есть свои единицы измерения. Минимальное количество информации, для кодирования которой достаточно одного двоичного разряда, называют **битом** (bit).

Слово «бит» произошло от английских слов *binary* (двоичный) и *digit* (знак). Бит — минимальная единица, выражающая количество информации. Он может принимать одно из двух значений — 0 или 1. Для удобства введена более крупная единица измерения информации — байт.

Байт — единица измерения количества информации, состоящая из восьми последовательных и взаимосвязанных битов.

1 байт = 2^3 бит = 8 бит (пример 2.5).

Пример 2.5. Соотношение между битом и байтом.



Для обозначения большего объема информации используются другие единицы измерения:

1 Кбайт (килобайт) = = 1024 байта;

1 Мбайт (мегабайт) = = 1 048 576 байт;

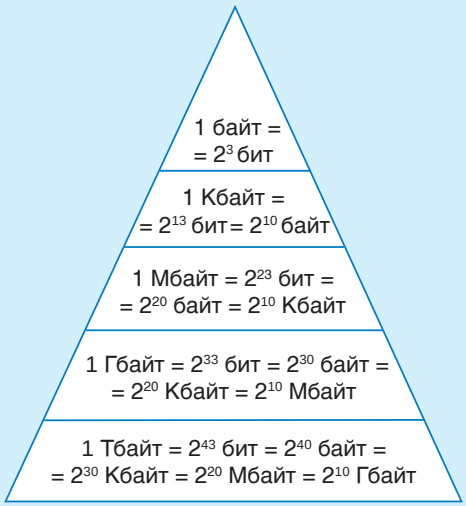
1 Гбайт (гигабайт) = = 1 073 741 824 байта;

1 Тбайт (терабайт) = = 1 099 511 627 776 байт.

Значения данных единиц измерения информации для удобства кодирования связаны со степенью числа 2 (пример 2.6).

В этих единицах измеряются количество (объем) оперативной или внешней памяти компьютера, размеры файлов. В примере 2.7 показано, каким образом выполняется перевод одних единиц измерения информации в другие.

Пример 2.6. Соотношение единиц измерения информации.



Пример 2.7.

Переведем 2364 Мбайт в килобайты и гигабайты:

2368 Мбайт = = (2368 · 2^{10}) Кбайт = = 2 424 832 Кбайт;

2368 Мбайт = = (2368 / 2^{10}) Гбайт ≈ ≈ 2,3 Гбайт.

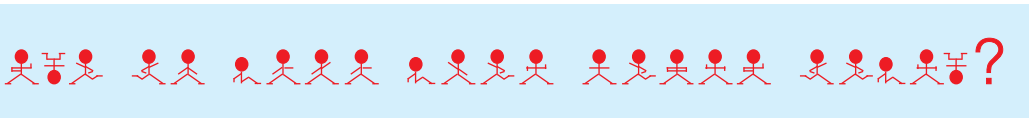
1. Что такое код?
2. Какой процесс называют кодированием информации?
3. Какой код используют для кодирования информации в компьютере?
4. Какие единицы измерения информации вы знаете?

Упражнения

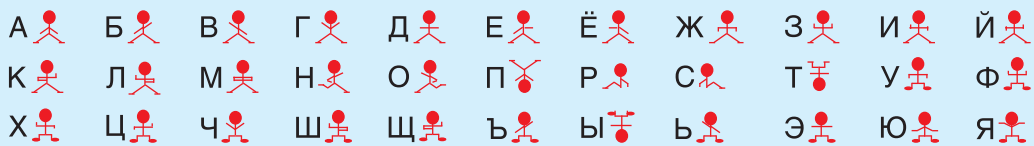
1 Используя шифр Юлия Цезаря со сдвигом вправо на 3, закодируйте фразу «Кто владеет информацией, тот владеет миром».

2 Используя шифр Юлия Цезаря со сдвигом влево на 2, закодируйте фразу Стива Джобса «Компьютер — это как велосипед для нашего мозга».

3 В одном из рассказов А. Конан Дойля великий сыщик Шерлок Холмс разгадывает шифр пляшущих человечков. Расшифруйте фразу, используя алфавит, применявшийся при ее кодировании.



Алфавит для кодирования информации



4 В азбуке Морзе буквы и цифры заменяются последовательностями из коротких и длинных сигналов — точек и тире:

А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О
.-	-...	.--	--.	-..	.	...-	--..	..	-.-	.-..	--	-.	---
П	**Р**	**С**	**Т**	**У**	**Ф**	**Х**	**Ц**	**Ч**	**Ш**	**Щ**	**Ъ, Ь**	**Ы**	**Э**
.--.	.-.	...	-	..-	..-.	....	-.-.	---.	----	--.-	-..-	-.--	..-..
Ю	**Я**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**0**		
..--	.-.-	.----	..---	...--	....-	.....	-....	--...	---..	----.	-----		

1. С помощью азбуки Морзе запишите:
«Запас беды не чинит».
2. Расшифруйте информацию, записанную на азбуке Морзе.

-.	.-		---	----	..	-...	-.-	.-	....		..-	---.	.-	-	...	.-.-

5 Выполните перевод единиц измерения информации:

1. 174 байта в биты.
2. 342,3 Кбайт в байты.
3. 45 638 Мбайт в гигабайты.

Глава 2
ПРЕДСТАВЛЕНИЕ О ЛОГИКЕ ВЫСКАЗЫВАНИЙ. МНОЖЕСТВА И ОПЕРАЦИИ НАД НИМИ

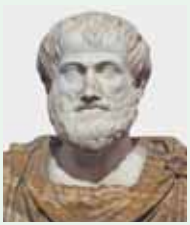
§ 3. Логика высказываний

Возможности компьютера велики. Он может помочь врачу поставить правильный диагноз пациенту, пассажиру — выбрать билет на нужный поезд; компьютер может управлять автомобилем, составлять прогнозы погоды и многое другое.

Для того чтобы выяснить, может ли компьютер «думать», сначала нужно понять, как думает человек. Ведь именно человек создал компьютер, и компьютер выполняет только те действия, которым его научил человек.

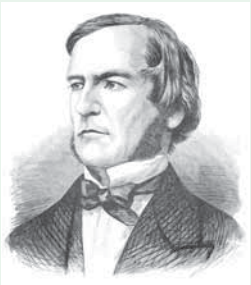
Наши знания об окружающем мире мы выражаем в повествовательных предложениях. Такие предложения могут отражать действительность верно или неверно. Думая, человек строит свои рассуждения, основываясь на собственных знаниях.

Еще Аристотель заметил, что правильность рассуждений не зависит от содержания, а определяется формой.



Древнегреческий философ Аристотель (384—322 гг. до н. э.) первым систематизировал формы и правила мышления, разработал теорию умозаключений и доказательств, описал логические операции. Аристотелю принадлежат формулировки основных законов мышления.

У истоков современной логики стоит немецкий математик Готфрид Вильгельм Лейбниц (1646—1716). Ученый предложил идею представить логические рассуждения как вычисления, подобные вычислениям в математике.



Английский математик и логик Джордж Буль (1815—1864) перенес на логику законы и правила математических (алгебраических) действий, создав тем самым алгебру логики.

На логических элементах строятся логические схемы электронных устройств. Законы булевой алгебры применяются и в программировании.

Наука, изучающая формы рассуждений, называется **формальной логикой.**

Математическая логика использует математические методы для исследования способов построения рассуждений, доказательств, выводов.

Одним из разделов современной математической логики является **логика высказываний.**

На правилах математической логики построены процессы «рассуждений» компьютера. Изучение логики высказываний поможет понять, как можно научить компьютер «думать».

3.1. Понятие высказывания

Высказывание — повествовательное предложение (утверждение), о котором в настоящее время можно сказать, истинно оно или ложно (пример 3.1).

Пример 3.1. Следующие предложения являются высказываниями:

1. Атом водорода самый легкий (истинно).

2. Клетка — центральная часть атома (ложно).

3. Кирилл Туровский — известный английский писатель и оратор, живший во второй половине XII в. (ложно).

4. При делении любого числа (кроме нуля) на само себя получается число 1 (истинно).

Об истинности высказывания можно говорить только в настоящем времени: высказывание «Идет дождь» может быть истинным сейчас и ложным через час.

Как правило, высказывания обозначают заглавными латинскими буквами. Если высказыва-

ние A истинно, пишут $A = 1$, если ложно — $A = 0$ (пример 3.2). Часто используют такие обозначения: A = true (истина) и A = false (ложь).

3.2. Логическая операция НЕ

С высказываниями можно производить различные операции, подобно тому как в математике — с числами (сложение, умножение, вычитание и др.).

Логическая операция **НЕ** (отрицание) меняет значение высказывания на противоположное: истинно на ложно, а ложно на истинно.

Логическое отрицание получается из высказывания путем добавления частицы «не» к сказуемому или с использованием оборота «неверно, что...» (пример 3.3). Иногда при построении отрицаний некоторые слова заменяют их антонимами, если это возможно.

Если высказывание содержит слова «все», «всякий», «любой», то его отрицание строится с использованием слов «некоторые», «хотя бы один». И наоборот, для высказываний со словами «некоторые», «хотя бы один» отрицание будет содержать слова «все», «всякий», «любой» (пример 3.4).

Пример 3.2.

A = «a^0 равно 1, если $a \neq 0$»;

B = «Массу измеряют в литрах».

Для приведенного примера $A = 1$, $B = 0$.

Пример 3.3. Построим отрицание высказываний.

Высказывания:

1. У цветковых растений развивается плод.
2. Фреска — это живопись водяными красками по свежей штукатурке.

Отрицание высказываний:

1. У цветковых растений **не** развивается плод.
2. **Неверно, что** фреска — это живопись водяными красками по свежей штукатурке.

Пример 3.4. Построим отрицание высказываний.

Высказывания:

1. Все учащиеся занимаются спортом.
2. Некоторые птицы умеют плавать.
3. Любой цветок имеет запах.
4. Иногда у мамы бывает плохое настроение.

Отрицание высказываний:

1. **Некоторые** учащиеся **не** занимаются спортом.
2. **Все** птицы **не** умеют плавать.
3. **Хотя бы один** цветок **не** имеет запаха.
4. У мамы **всегда** бывает **хорошее** настроение.

Пример 3.5. Определение истинности высказываний с отрицаниями.

1. Ель — это дерево (истинное высказывание). Ель — это не дерево (ложное высказывание).

$A = 1$, **НЕ** $A = 0$.

2. Число -7 является положительным (ложное высказывание). Число -7 не является положительным (истинное высказывание).

$A = 0$, **НЕ** $A = 1$.

3. Все вещества — металлы (ложное высказывание). Некоторые вещества не металлы (истинное высказывание).

$A = 0$, **НЕ** $A = 1$.

4. Все составляющие воздуха являются газами (истинное высказывание). Некоторые составляющие воздуха не являются газами (ложное высказывание).

$A = 1$, **НЕ** $A = 0$.

5. Длительность суток не зависит от скорости вращения планеты (ложное высказывание). Длительность суток зависит от скорости вращения планеты (истинное высказывание).

$A = 0$, **НЕ** $A = 1$.

6. Дома на левой стороне улицы имеют четные номера (ложное высказывание). Неверно, что дома на левой стороне улицы имеют четные номера (истинное высказывание).

$A = 0$, **НЕ** $A = 1$.

Любую операцию над числами в математике обозначают каким-либо знаком: «+», «−», «·», «:». Для логических операций тоже определены свои обозначения. Если операцию отрицания применяют к высказыванию A, то это можно записать так: **НЕ** A. Можно встретить и другие обозначения для логической операции отрицания: **Not** A, $\neg A$, $\overline{A}$, $\sim A$.

Если нас интересует истинность высказывания **НЕ** A, то ее (вне зависимости от содержания) можно определить по таблице истинности:

A	**НЕ** A
1	0
0	1

Из таблицы истинности следует, что отрицанием истинного высказывания будет ложное, а отрицанием ложного — истинное (пример 3.5). Высказывание и его отрицание никогда не могут быть истинными или ложными одновременно.

Например, отрицанием высказывания «У меня есть компьютер» будет высказывание «У меня нет компьютера» (или высказывание «Неверно, что у меня есть

компьютер»). Истинность этих высказываний зависит от конкретного человека. Для одних будет истинным первое высказывание, а для других — второе. Но оба высказывания не могут быть истинными или ложными одновременно для одного и того же человека.

Часто трудно установить истинность высказывания. Высказывание «Площадь озера Нарочь 79,6 км^2» в одной ситуации можно посчитать ложным, а в другой — истинным. Ложным — так как указанное значение не является постоянным. Истинным — если рассматривать его как некоторое приближение, приемлемое на практике.

1. Что такое высказывание?
2. Какие значения могут иметь высказывания?
3. Что делает логическая операция *отрицание*?
4. Как построить отрицание высказывания?

Упражнения

1 Какие из предложений являются высказываниями, а какие — нет?

1. Включи монитор.
2. Кислород — это газ.
3. Шишка — это цветок.
4. Сколько воды утекло?
5. Все дети — учащиеся.
6. Хотя бы один пароль будет верным.

2 Определите истинность высказываний.

1. 123 — это цифра.
2. Слово «стол» — это существительное.
3. Число 46 является степенью числа 2.
4. Значение выражения $a = \frac{x + y}{3}$ равно 0,75.
5. Железо легче воды.

3 Постройте отрицания высказываний.

1. Миша не может пойти в кино.
2. Соня любит рисовать.
3. Все планеты не имеют атмосферы.

4. В сентябре не бывает дождей.
5. Солнце светит ярко.
6. Некоторые птицы улетают на юг.

4 Откройте файл с данными ниже предложениями и отредактируйте их, удалив или вставив частицу «не» так, чтобы все предложения стали истинными высказываниями.

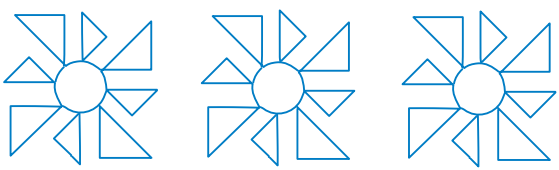
Озеро Нарочь не является крупнейшим озером Беларуси.
Все горы являются вулканами.
Дуб — хвойное дерево.
Число 27 является простым числом.
Термометр не позволяет определить температуру тела.
Число 2016 не делится на 3.
Треугольник не является геометрической фигурой.

5 Какие утверждения о животных, представленных на рисунках, истинные, а какие — ложные?



1. Некоторые из этих животных умеют лазать по деревьям.
2. Все животные обитают в лесах.
3. Ни одно из животных не является домашним.
4. Каждое животное можно погладить.
5. Все люди любят мышей.
6. Ни одно из животных не умеет плавать.

6 Откройте файл с рисунком трех цветков. Раскрасьте их так, чтобы каждое из высказываний было истинным.



1. Все цветки имеют желтый круг в середине.
2. На рисунке есть цветок с синими лепестками.
3. На рисунке нет цветка с красными лепестками.
4. Неверно, что цвет круга в середине цветка совпадает с цветом лепестков.
5. Хотя бы у одного цветка лепестки разного цвета.

7 Создайте 4 копии рисунков, полученных в задании 6. Дополните каждую копию изображениями ваз (выберите из файла) так, чтобы соответствующее из нижеприведенных высказываний было ложным.

1. Все изображения ваз — четырехугольники.
2. На вазах есть орнамент в виде кругов.
3. Все круги в орнаменте разного размера.
4. Хотя бы один круг в орнаменте белого цвета.

8 Задумано некоторое число x. Среди высказываний $x > 1$, $x > 2$, $x > 3$, $x > 4$, $x > 5$ есть два верных и три неверных. Какие высказывания неверные?

9* Решите задачу-стихотворение.

Собаки с рыжими хвостами
Себе овсянку варят сами.
Тем, чьи хвосты стального цвета,
Не позволяют делать это.
Кто варит сам себе овсянку,
Гулять выходит спозаранку.
Все, кто гулять выходят рано,
Не терпят фальши и обмана.
Вид добродушный у Барбоса,
Но на сорок он смотрит косо.
Он видит: норовят сороки
У воробьев списать уроки!
Скажите — проще нет вопроса! —
Какого цвета хвост Барбоса?[1]

[1] Разговоров, Н. «Собаки с рыжими хвостами...» [Электронный ресурс] / Н. Разговоров. — Режим доступа: http://po.m-necropol.ru/razgovorov-nikita.html. — Дата доступа: 26.06.2017.

§ 4. Логические операции И и ИЛИ

В 1936—1938 гг. американский инженер и математик Клод Шеннон (1916 — 2001) нашел применение булевой логике при конструировании схем из реле и переключателей. В дальнейшем это открытие послужило основанием для построения логических элементов, на которых работает компьютерная техника. Состояние элементов компьютера соответствует логическим значениям:

- если сигнал присутствует, получаем логическую 1;
- если сигнал отсутствует, получаем логический 0.

Логика высказываний позволяет строить **составные высказывания.** Они создаются из нескольких простых высказываний путем соединения их друг с другом с помощью логических операций **НЕ, И, ИЛИ** и др.

4.1. Логическая операция И

Определение истинности или ложности составного высказывания зависит от того, являются ли истинными или ложными простые высказывания, входящие в его состав, а также от той логической операции, которая их связывает.

Составное высказывание A **И** B, образованное в результате объединения двух простых высказываний A и B логической операцией **И**, истинно тогда и только тогда, когда A и B одновременно истинны.

Пример 4.1. Проанализируем высказывание «Число 456 трехзначное и четное».

Данное высказывание является составным, поскольку содержит два простых высказывания:

«Число 456 трехзначное» (высказывание A);

«Число 456 четное» (высказывание B).

Высказывания A и B соединены вместе логической операцией **И**, в результате получено составное высказывание A **И** B. Высказывание A истинно, высказывание B истинно. Поэтому высказывание A **И** B истинно: (A **И** B) = 1.

Если хотя бы одно из простых высказываний, связанных операцией **И**, будет ложным, то и составное высказывание будет ложным (примеры 4.1 и 4.2).

Операцию **И** называют **логическим умножением.** Равенства

$1 \cdot 1 = 1, 1 \cdot 0 = 0, 0 \cdot 1 = 0, 0 \cdot 0 = 0$, верные для обычного умножения, верны и для логического умножения.

Представим таблицу истинности для логической операции **И**:

A	*B*	*A* **И** *B*
1	1	1
0	1	0
1	0	0
0	0	0

Для записи логической операции **И** используют следующие обозначения: *A* **И** *B*, *A* **AND** *B*, $A \cdot B$, $A * B$, $A \wedge B$, $A \,\&\, B$.

4.2. Логическая операция ИЛИ

Составное высказывание *A* **ИЛИ** *B*, образованное в результате объединения двух простых высказываний *A* и *B* логической операцией **ИЛИ**, ложно тогда и только тогда, когда *A* и *B* одновременно ложны.

Другими словами, составное высказывание *A* **ИЛИ** *B* будет истинным только в том случае, если истинно хотя бы одно из двух составляющих его простых

Пример 4.2. Высказывание *A*: «Геракл — герой древнерусской мифологии». **Ложно**, $A = 0$.

Высказывание *B*: «Геракл — сын бога Зевса». **Истинно**, $B = 1$.

Высказывание *A* **И** *B*: «Геракл — герой древнерусской мифологии **И** сын бога Зевса». **Ложно**, (*A* **И** *B*) = 0.

Пример 4.3. Проанализируем высказывание «Семиклассники изучают философию или астрономию».

Данное составное высказывание образовано из двух простых:

«Семиклассники изучают философию» (высказывание *A*);

«Семиклассники изучают астрономию» (высказывание *B*).

Высказывания связаны логической операцией **ИЛИ**. В результате получилось составное высказывание *A* **ИЛИ** *B*. Высказывание *A* ложно, высказывание *B* ложно. Поэтому высказывание *A* **ИЛИ** *B* ложно: (*A* **ИЛИ** *B*) = 0.

Пример 4.4. Высказывание *A*: «Франциск Скорина — белорусский первопечатник». **Истинно,** $A = 1$.

Высказывание *B*: «Стефан Баторий — турецкий султан». **Ложно,** $B = 0$.

Франциск Скорина

Стефан Баторий

Высказывание «Франциск Скорина — белорусский первопечатник, **ИЛИ** Стефан Баторий — турецкий султан» будет **истинным,** (*A* **ИЛИ** *B*) = 1.

Пример 4.5*. Рассмотрим выражение: *A* **ИЛИ** *B* **И НЕ** *C*. Распишем по действиям вычисление его значения:

1) *D* = **НЕ** *C*;
2) *E* = *B* **И** *D*;
3) *F* = *A* **ИЛИ** *E*.

Значение высказывания *F*, полученное в 3-м действии, определит значение исходного логического выражения.

Пример 4.6*. Пусть высказывание $A = 1$, $B = 0$, $C = 0$. Найдем значение логического выражения *A* **ИЛИ** *B* **И НЕ** *C*.

1) *D* = **НЕ** *C* = 1;
2) *E* = *B* **И** *D* = 0 **И** 1 = 0;
3) *F* = *A* **ИЛИ** *E* = 1 **ИЛИ** 0 = 1.

Значит, при начальных значениях $A = 1$, $B = 0$, $C = 0$ значение логического выражения *A* **ИЛИ** *B* **И НЕ** *C* истинно.

высказываний (см. пример 4.3 на с. 27 и пример 4.4).

Таблица истинности для логической операции **ИЛИ**:

A	*B*	*A* **ИЛИ** *B*
1	1	1
0	1	1
1	0	1
0	0	0

Операцию **ИЛИ** называют **логическим сложением.** Равенства 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0, верные для обычного сложения, верны и для логического сложения.

Для записи логической операции **ИЛИ** можно использовать следующие выражения: *A* **ИЛИ** *B*, *A* **OR** *B*, *A* + *B*, $A \vee B$, $A \mid B$.

Если в логическом выражении присутствует несколько логических операций, то важно определить порядок их выполнения. Наивысшим приоритетом обладает операция **НЕ.** Логическая операция **И,** т. е. логическое умножение, выполняется раньше операции **ИЛИ** — логического сложения (примеры 4.5* и 4.6*).

Для изменения порядка выполнения логических операций используют скобки: в этом случае сначала выполняются операции в скобках, а затем — все остальные.

Логические операции **И** и **ИЛИ** подчиняются переместительному закону:

A **И** *B* = *B* **И** *A*;

A **ИЛИ** *B* = *B* **ИЛИ** *A*.

Чтобы определить значение составного логического выражения, иногда достаточно знать значение только одного простого высказывания. Так, если в составном высказывании с операцией **И** хотя бы одно простое высказывание является ложным, то значение составного высказывания будет ложным. Если в составном высказывании с операцией **ИЛИ** хотя бы одно простое высказывание истинно, то значение составного высказывания будет истинным (пример 4.7).

Пример 4.7. Высказывание *A*: «Прогноз погоды обещает дожди». Высказывание *B*: «Сейчас на улице идет дождь».

Высказывание *A* **И** *B* будет ложным, если мы увидели, что на улице нет дождя (независимо от того, что обещал прогноз погоды).

Высказывание *A* **ИЛИ** *B* будет истинным, если прогноз погоды обещал дождь (независимо от того, какую погоду мы наблюдаем сейчас).

1. В каких случаях составное высказывание *A* **И** *B* может быть истинным?
2. В каких случаях составное высказывание *A* **ИЛИ** *B* может быть ложным?

Упражнения

1 Определите, истинными или ложными являются нижеприведенные составные высказывания.

1. Мяч круглый, **ИЛИ** Земля плоская.
2. Кролики — домашние животные, **И** баобаб растет в Беловежской пуще.
3. Клавиатура — устройство ввода информации, **ИЛИ** мышь — устройство вывода информации.

4. И. А. Крылов написал басню «Квартет», **И** М. Ю. Лермонтов написал стихотворение «Парус».
5. Сосна — хвойное дерево, **И** кедр — не хвойное дерево.
6. Монитор — устройство ввода информации, **ИЛИ** сканер — **НЕ** устройство вывода информации.
7*. Континенты и острова — это большие участки суши.

2 О том, как прошли летние каникулы, Кира рассказала своим друзьям следующее:

1. Я была у бабушки в деревне, и рядом с деревней было озеро.
2. По озеру плавала лодка или утка.
3. Мы с бабушкой насобирали малины и смородины.
4. Я составила букет из цветов. В нем были ромашки или гвоздики.

Подготовьте к каждому из высказываний Киры рисунки, учитывая, что все высказывания истинны.

3 Откройте файл с рисунком и разложите грибы по корзинкам так, чтобы было истинным следующее высказывание: «В большой корзине все грибы съедобные, и в маленькой корзине все грибы несъедобные».

4 Откройте файл с рисунком и поставьте все цветы в вазы так, чтобы было истинным высказывание: «В синей вазе все цветы розы, или в красной вазе все цветы не красного цвета».

5* Найдите значения логических выражений, если $A = 1$, $B = 1$, $C = 0$, $D = 0$.

1. A **ИЛИ** B **И НЕ** C.
2. A **И НЕ** B **ИЛИ** C.
3. A **ИЛИ** B **И НЕ** (C **И** D).
4. (A **И** B) **ИЛИ НЕ** C **И** (A **ИЛИ** B) **ИЛИ НЕ** D.

§ 5. Множества

5.1. Понятие множества

Рассмотрим высказывание «Все учащиеся нашего класса имеют дома компьютер». Истинно оно или ложно? Для ответа на этот вопрос вам нужно у каждого из одноклассников уточнить: «У тебя дома есть компьютер?» Если все учащиеся класса ответят утвердительно, то высказывание истинно, если хотя бы один из учащихся ответит «нет», то и высказывание будет ложным. Для разных классов это высказывание будет иметь различные значения, потому что различными будут множества учащихся класса.

Множество — совокупность каких-либо объектов, обладающих общим свойством. Эти объекты называют **элементами множества**.

Можно говорить о множестве учащихся 7 А класса, множестве отметок в классном журнале, множестве городов Беларуси, множестве букв русского алфавита и т. д. Понятие множества является одним из основных в математике.

Множества, как правило, обозначают прописными латинскими

Множества, в том числе и бесконечные, в неявной форме использовались в математике со времен Древней Греции.

До XIX в. считалось, что точного определения множества нет. Множеством называли любое скопление, объединение предметов.

В конце XIX в. немецкий математик Георг Кантор (1845—1918) определил множество как «единое имя для совокупности всех объектов, обладающих данным свойством».

По теории Г. Кантора некоторые множества конечны (например, целые числа от 1 до 7), а некоторые — бесконечны (например, целые числа). В некоторых случаях элементы одного множества строго соответствуют элементам другого множества, например множество цветов радуги и множество целых чисел от 1 до 7.

Пример 5.1. Пусть M — множество любимых учебных предметов семиклассника Игоря, состоящее из элементов: математика, информатика, английский язык. Тогда можно записать:

M = {математика, информатика, английский язык};

информатика $\in M$;

литература $\notin M$.

Пример 5.2. Пусть в множество M входят все учебные предметы, изучаемые в 7-м классе. Перечислить все его элементы можно, например, глядя на страницу школьного дневника. Тогда можно записать:

информатика $\in M$;

астрономия $\notin M$.

Леонард Эйлер (1707—1783) — ученый, внесший значительный вклад в развитие математики и механики, а также физики, астрономии и ряда прикладных наук.

Разработал удобный метод для графического изображения отношений между множествами.

буквами, а элементы множества — строчными. Напомним, что для обозначения принадлежности элемента множеству используют специальные знаки:

$a \in M$ (элемент a принадлежит множеству M), $a \notin M$ (элемент a не принадлежит множеству M). Если множество M состоит из элементов a, b, c, то это записывают так: $M = \{a, b, c\}$.

Чтобы задать множество, необходимо перечислить его элементы (пример 5.1) или назвать их общее свойство (пример 5.2).

5.2. Понятие подмножества

Рассмотрим множество учащихся какого-либо класса. В этом множестве можно выделить не только отдельного учащегося, но и некоторые группы учащихся: отличники, учащиеся, умеющие играть в теннис и т. д. Каждая из таких групп образует подмножество — часть множества учащихся.

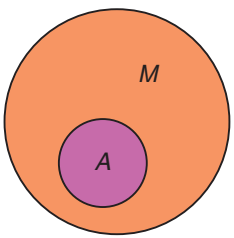
Если множество A является подмножеством множества M, то это записывают так: $A \subset M$. Запись $A \not\subset M$ обозначает, что множество A не является подмножеством множества M.

Подмножество может содержать все элементы множества,

а может не содержать ни одного (пустое множество; обозначается знаком ∅).

Некоторые элементы множества могут принадлежать одновременно разным подмножествам (пример 5.3).

Для наглядной геометрической иллюстрации множеств и отношений между ними используют круги Эйлера. Каждое множество изображается кругом. Если какое-либо множество является подмножеством другого множества, то один круг изображается внутри другого. Например, если M — множество всех хищников, A — множество всех львов ($A \subset M$), то это обозначается так:



Пример 5.3. Пусть M = {Вера, Сергей, Вася, Вика, Лиза, Костя, Надя} — множество учащихся 7 Б класса, занимающихся в драмкружке. Из этого множества можно выделить следующие подмножества:

A = {Вера, Вика, Лиза, Надя} — множество девочек (красная граница).

B = {Сергей, Вася, Костя} — множество мальчиков (синяя граница).

C = {Вера, Вася, Вика} — множество детей, чьи имена начинаются на букву «В» (зеленая граница).

D = {Вера, Сергей, Вася, Вика, Лиза, Костя, Надя} — множество детей, в именах которых по 2 гласных звука (желтая граница).

$E = \varnothing$ — множество трехлетних детей.

1. Что понимают под множеством?
2. Приведите примеры множеств.
3. Что понимают под подмножеством?
4. Что используется для геометрической иллюстрации множеств?
5. Что понимают под пустым множеством? Как оно обозначается?
6. Может ли элемент множества одновременно принадлежать различным подмножествам?

Упражнения

1 Дополните каждое из множеств 1—2 элементами.

1. A = {математика, информатика, история, литература}.
2. B = {яблоко, груша, апельсин, банан}.
3. C = {клавиатура, монитор, мышь}.
4. D = {карандаш, ручка, ластик, фломастер}.

2 Какие элементы могут входить в следующие множества?

1. Средства передвижения.
2. Цвета радуги.
3. Домашние животные.
4. Четные числа.

3 Откройте файл с группами слов. Разделите слова каждой группы на два множества. Слова первого множества выделите красным цветом, а второго — синим. По каким признакам вы разделили слова?

Образец:

1. Текст в файле: гусь, лебедь, заяц, волк, павлин, курица, кабан, лось.

Результат: A = {гусь, лебедь, павлин, курица};
B = {заяц, волк, кабан, лось}.

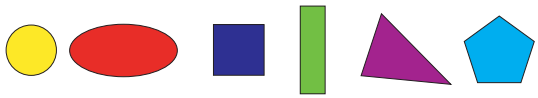
Признаки: A — множество птиц, B — множество зверей.

2. Мяч, стол, стул, коньки, шкаф, клюшка, шайба, комод.
3. Сом, уж, карась, окунь, щука, гадюка, кобра, питон.

4 Из множества геометрических фигур A = {круг, овал, квадрат, прямоугольник, треугольник, пятиугольник} выделите подмножества:

1. Фигур, не имеющих углов.
2. Фигур, являющихся четырехугольниками.
3. Фигур, количество углов у которых больше трех.

5 Откройте файл с изображениями геометрических фигур. С помощью операции копирования создайте подмножества *а*, *б*, *в*, обладающие признаками, указанными в задании 4. Все элементы каждого подмножества разместите внутри соответствующего прямоугольника.



а *б* *в*

6 Откройте файл с изображениями бабочек. Используя операцию копирования, создайте нижеперечисленные подмножества и разместите их в прямоугольниках.

1. Бабочки, в раскраске которых есть синий цвет.
2. Бабочки, в раскраске которых есть красный цвет.
3. Бабочки, в раскраске которых есть зеленый цвет.
4. Бабочки, в раскраске которых есть желтый цвет.

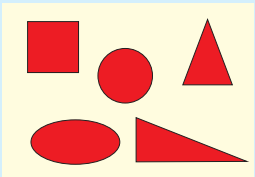
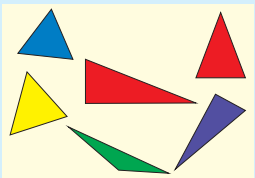
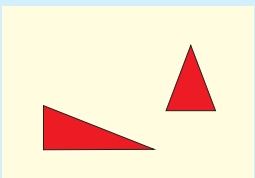
7 Заданы два множества: *K* — множество книг в школьной библиотеке; *U* — множество учебных пособий в этой же библиотеке. Какое из множеств является подмножеством другого? Изобразите их с помощью кругов Эйлера.

8 Составьте цепочку включений так, чтобы каждое следующее множество являлось подмножеством предыдущего: *A* — множество всех прямоугольников; *B* — множество всех четырехугольников; *C* — множество всех квадратов; *D* — множество всех многоугольников.

9* Придумайте примеры цепочек, состоящих из множеств и их подмножеств и содержащих не менее трех включений.

§ 6. Операции над множествами

Пример 6.1. Найдем пересечение множеств A и B.

Множество A	Фигуры красного цвета 
Множество B	Треугольники 
Множество $A \cap B$	Треугольники красного цвета 

Пример 6.2. Найдем пересечение множеств A и B.

Множество A — животные, умеющие летать: пчела, журавль, майский жук, воробей, аист, стрекоза.

Множество B — птицы: страус, журавль, пингвин, аист, курица, воробей.

Пересечение $A \cap B$ = {журавль, аист, воробей} — птицы, которые умеют летать.

Для множеств, как и для высказываний, определены свои операции. Такими операциями являются операции пересечения и объединения множеств.

Рассмотрим множество учащихся 7-го класса. Выделим в нем два подмножества: множество любителей игры в настольный теннис и множество учащихся, у которых дома есть компьютер. Некоторые из учащихся могут и иметь компьютер, и увлекаться теннисом. Значит, они будут входить в оба множества.

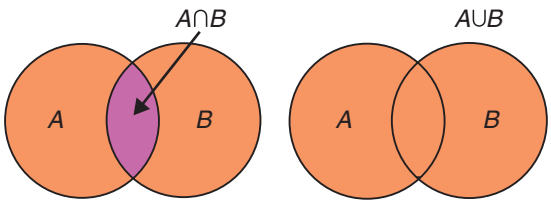
Пересечением множеств A и B называется множество, в которое входят только те элементы, которые принадлежат как множеству A, так и множеству B. Для обозначения операции пересечения используется знак $\cap$. Образцы выполнения заданий на нахождение пересечения множеств представлены в примерах 6.1 и 6.2.

Выделим среди учащихся 7-го класса два подмножества: множество любителей игры в настольный теннис и множество любителей игры в большой теннис. Тогда множество любителей тенниса будет включать в себя и тех, кто играет в настольный

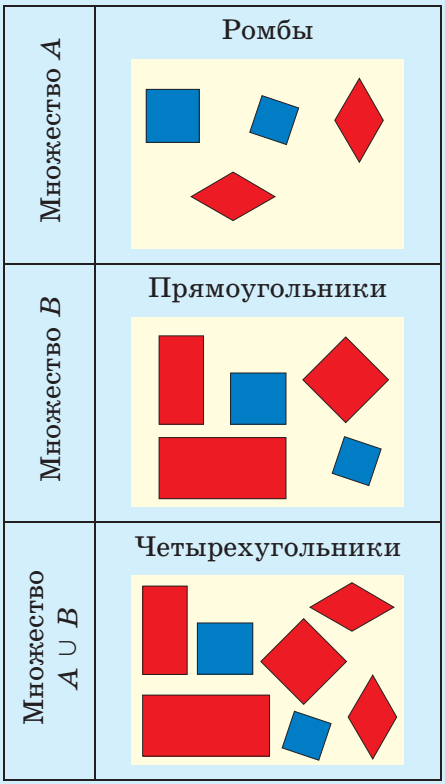
теннис, и тех, кто играет в большой теннис. Если кто-то играет и в большой, и в настольный теннис, то он тоже будет входить в множество любителей тенниса.

Объединением множеств A и B называется множество, в которое входят элементы, принадлежащие хотя бы одному из множеств A или B. Для обозначения операции объединения множеств используется знак $\cup$. Образец выполнения задания на объединение множеств показан в примере 6.3.

Пересечение и объединение двух множеств можно изобразить с помощью кругов Эйлера.



Пример 6.3. Найдем объединение множеств A и B.



1. Что называют пересечением множеств?
2. Что называют объединением множеств?
3. Как обозначаются операции пересечения и объединения множеств?

Упражнения

1 Найдите пересечение и объединение множеств A и B.

1. A = {математика, информатика, история, литература};
   B = {английский язык, математика, химия, история}.
2. A = {яблоко, апельсин, мандарин, лимон, киви};
   B = {апельсин, персик, мандарин, груша, лимон}.

2 Заданы два множества. Найдите их пересечение и объединение.

1. Множество задач, решаемых с помощью программы *графический редактор* = {открыть, сохранить, создать, заливка цветом, печать}.

2. Множество задач, решаемых с помощью программы *текстовый редактор* = {открыть, сохранить, создать, увеличить размер шрифта, печать}.

3 Решите задачи с использованием кругов Эйлера (нарисуйте их в графическом редакторе).

1. Об учащихся школы, участвовавших в физико-математическом конкурсе, известно, что 7 из них решили задачи и по математике, и по физике, 11 — задачи по математике, 9 — задачи по физике. Сколько учащихся принимали участие в конкурсе?

2. В киоске около школы продается мороженое двух видов: «Эскимо» и «Пломбир». После уроков 24 семиклассника купили мороженое. При этом 15 из них выбрали «Эскимо», а 17 — «Пломбир». Сколько семиклассников купили мороженое двух видов?

3*. Из 100 туристов, отправляющихся в путешествие, немецким языком владеют 30 человек, английским — 28, французским — 42. Английским и немецким одновременно владеют 8 человек, английским и французским — 10, немецким и французским — 5, всеми тремя языками — 3. Сколько туристов не владеют ни одним языком?

4* Используя рисунок, выполните задания.

1. Создайте два подмножества множества девочек. Для всех девочек, входящих в первое подмножество, истинно высказывание: «Девочка носит брюки синего цвета, **И** на ее майке есть красный цвет». Для всех девочек, входящих во второе подмножество, истинно высказывание: «Девочка одета не в брюки **ИЛИ** имеет волосы желтого цвета».

2. Найдите пересечение и объединение этих множеств.
3. Сколько девочек не попало ни в одно подмножество?
4. Выполните упражнение в графическом редакторе. Вокруг девочек из первого множества нарисуйте границу красным цветом, а вокруг девочек из второго множества — синим. Область пересечения обозначьте желтым цветом.

§ 7. Использование логических операций для построения поисковых запросов в Интернете

7.1. Поиск информации

В современный век информационных технологий люди используют для поиска информации сервисы Интернета. Поисковые системы постоянно собирают, систематизируют и сохраняют информацию по всему миру. Поиск информации в поисковой системе осуществляется по запросу.

Под **запросом** в поисковой системе понимают набор слов, фраз, символов, которые пользователь вводит в строку поиска, чтобы найти интересующую его информацию.

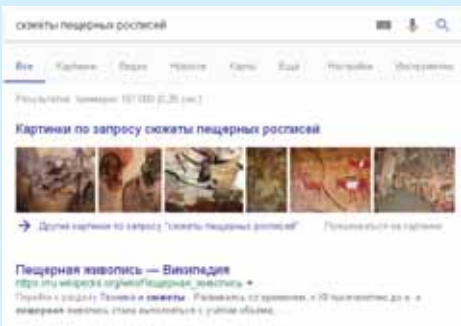
Современные поисковые системы позволяют осуществлять поиск по голосовым запросам или использовать в качестве запроса изображение.

Результатом поиска является перечень сайтов (пример 7.1

При поиске информации в Интернете важны полнота, точность и актуальность полученных результатов. Пользователь может повлиять на качество результатов поиска, если будет:

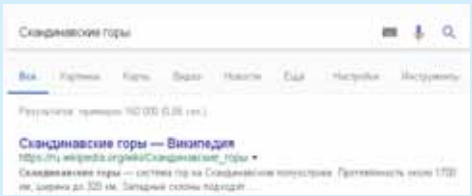
- продуманно выбирать поисковую службу;
- учитывать особенности поисковой системы;
- грамотно формулировать запросы на поиск информации.

Пример 7.1. Найдем информацию о сюжетах пещерных росписей.



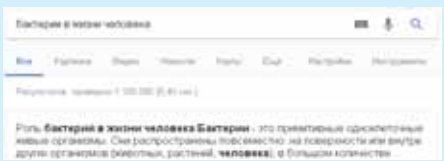
В результате поиска найдено более 160 тыс. сайтов, содержащих искомую информацию.

Пример 7.2. Найдем информацию о Скандинавских горах.

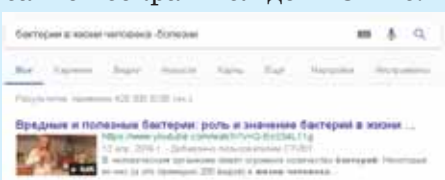


В результате поиска найдено 160 тыс. сайтов. Можно перейти в раздел «Картинки»:

Пример 7.3. Найдем информацию о бактериях в жизни человека.



Найдено более 1 млн сайтов. Если нас не интересуют бактерии, вызывающие болезни, то поисковый запрос можно изменить, добавив в конце «-болезни». Количество найденных сайтов сократится до 428 тыс.



и пример 7.2). Количество найденных сайтов может быть очень большим, и просмотреть их все часто не представляется возможным. На практике пользователи Интернета обычно просматривают 5—15 сайтов, найденных первыми.

Результативность поиска в значительной степени зависит от умения пользователя корректно сформулировать поисковый запрос. Формулировка фразы или выбор слов для поиска позволит получить более точный результат.

7.2. Сокращение области поиска

Для каждого из сайтов, найденных в результате поискового запроса, будет истинным следующее высказывание: «На странице сайта присутствует информация, соответствующая поисковому запросу». Все такие сайты образуют множество сайтов, удовлетворяющих поисковому запросу.

При построении поискового запроса некоторые сайты можно исключить из рассмотрения. Для этого к основному запросу добавляется слово со знаком минус («-») перед ним. Сайты, содержащие слова, отмеченные этим знаком

«-», не будут включены в список найденных (примеры 7.3 и 7.4).

Полученный перечень сайтов образует подмножество множества сайтов, удовлетворяющих основному запросу. Для всех таких сайтов высказывание «На странице сайта присутствует информация, соответствующая слову, отмеченному знаком “-”» будет восприниматься поисковой системой как ложное.

7.3. Использование операторов в поисковых запросах

Операторы поиска — слова или символы, добавляемые к поисковым запросам для уточнения результатов.

Оператор «+» позволяет осуществлять поиск документов, в которых обязательно присутствует слово, стоящее за символом. Допустимо использовать несколько операторов «+» в одном запросе (пример 7.5).

Оператор «-» мы рассмотрели в предыдущем пункте параграфа.

Оператор «*» заменяет любое неизвестное слово в запросе (пример 7.6).

Если поместить слова или фразу в кавычки, то в результатах поиска будут показаны только те

Пример 7.4. Найдем значение понятия «щит».

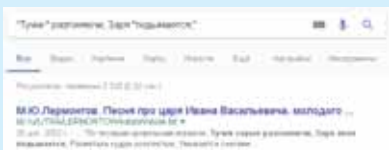
В результате поиска будет найдено более 20 млн сайтов, причем на нескольких первых страницах находится информация о сериалах, фильмах, магазинах.

Для уточнения информации введем запрос «щит -магазин -сериал -оружие». В таком случае количество ссылок сократится до 7 млн 270 тыс.

Пример 7.5. Найдем сайты, где есть информация о каждом из писателей: М. де Сервантесе, У. Шекспире и Ф. Рабле.

Пример 7.6. Составим запрос для поиска полной цитаты «Тучки ... разгоняючи, Заря ... подымается;». Из какого она произведения? Кто его автор?

Многоточие в поисковом запросе заменим знаком *.

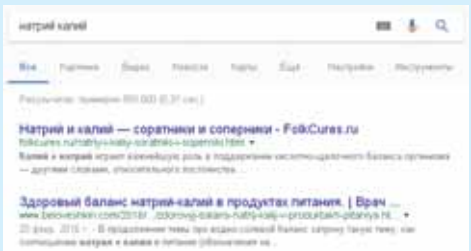


Как видим, это цитата из произведения М. Ю. Лермонтова «Песня про царя Ивана Васильевича, молодого опричника и удалого купца Калашникова».

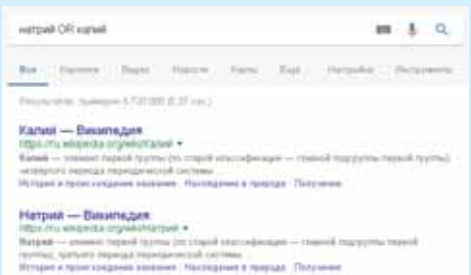
Пример 7.7. Найдем информацию о натрии или калии.

Сравним результаты запросов «натрий калий» и «натрий OR калий».

По первому запросу сначала размещены ссылки на сайты, содержащие информацию о двух химических элементах, а затем — о каждом из них:



По второму запросу сначала размещены ссылки на сайты об отдельных элементах, а потом — общая информация:



страницы, на которых эти слова (фразы) расположены в том же порядке, что и в запросе в кавычках. Кавычки используются тогда, когда необходимо найти точное слово или фразу, цитату.

Операторы, рассматриваемые далее, имеют различные обозначения для разных поисковых систем (например, для Google и Яндекс).

Оператор **OR** (поисковая система Google) позволяет найти страницы, содержащие хотя бы одно из нескольких слов, и соответствует логической операции **ИЛИ** (пример 7.7). Для поисковой системы Яндекс аналогичный оператор обозначается |.

Некоторые из операторов могут не иметь аналогов в других поисковых системах. Оператор & поисковой системы Яндекс осуществляет поиск документов, в которых слова запроса, объединенные данным оператором, встречаются в одном предложении.

1. Что называют запросом в поисковой системе?
2. Как исключить некоторые записи из области поиска?
3. Какие операторы можно использовать в поисковых запросах?

Упражнения

1 Найдите с помощью различных поисковых систем информацию о беговых видах легкой атлетики. Запишите результаты в таблицу (в тетрадь или в электронном виде). Сравните полученные результаты.

Поисковая система	Количество результатов поиска

2 Найдите с помощью поисковой системы изображения монет Великого Княжества Литовского. Выпишите в тетрадь 5—6 названий монет.

3 Сформулируйте запрос по поиску сюжетов пещерных росписей, исключающий роспись храмов.

4 Составьте запрос для поиска полной цитаты «Старость боится… Жизнь я… куплю». Кто автор этой фразы? В каком произведении она встречается?

5 С помощью соответствующих поисковых запросов получите ответ на вопрос: какое из событий произошло раньше — открытие Менделеевым периодического закона или изобретение Эдисоном фонографа?

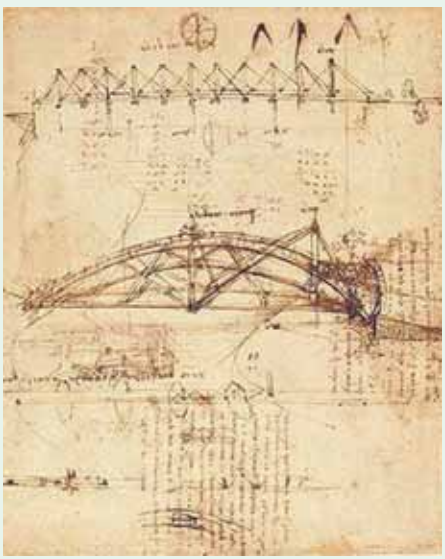
6 Составьте запросы на поиск названий белорусских озер, используя информацию из таблицы. Запишите названия найденных озер в тетрадь.

Березинский биосферный заповедник	П						
Мядельский район Минской области				О			
Браславский район Витебской области			И				
Гродненская область, деревня Валевка	С						
Крупнейшее среди Голубых озер						К	

Глава 3
ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ

§ 8. Алгоритмы и исполнители

Алгоритмы построения чертежей человек разрабатывает с глубокой древности. Появление чертежей связано с практической деятельностью человека — возведением укреплений и городских построек. Первые сведения о чертежах, напоминающих современные, связаны с именем Леонардо да Винчи (1452—1519) — итальянского ученого и художника, который в технических рисунках и эскизах раскрывал свои идеи в области техники и строительства.



8.1. Понятие алгоритма

Вспомним некоторые понятия, изученные в 6-м классе.

Алгоритм — понятная и конечная последовательность точных действий (команд), формальное выполнение которых позволяет получить решение поставленной задачи.

Исполнитель алгоритма — человек, группа людей или техническое устройство, которые понимают команды алгоритма и умеют правильно их выполнять.

Система команд исполнителя — команды, которые понимает и может выполнить исполнитель.

Любой исполнитель имеет ограниченную систему команд. Все они разделяются на группы:

1. Команды, которые непосредственно выполняет исполнитель.
2. Команды, меняющие порядок выполнения других команд исполнителя.

Компьютер является универсальным исполнителем.

Запись алгоритма в виде последовательности команд, которую может выполнить компьютер, называют **программой.**

Существуют следующие способы представления алгоритмов:

- **словесный** (описание алгоритма средствами естественного языка с точной и конкретной формулировкой фраз);
- **графический (блок-схема)** (графическое изображение команд алгоритма с использованием геометрических фигур, или блоков, и стрелок, соединяющих эти блоки и указывающих на порядок выполнения команд);
- **программный** (запись алгоритма в виде программы).

(Схематически данные способы представлены в примере 8.1.)

8.2. Исполнитель Чертежник

В 6-м классе вы познакомились с исполнителем Чертежник, предназначенным для построения рисунков и чертежей на координатной плоскости (пример 8.2).

Чертежник имеет перо, которым он может рисовать отрезки на плоскости. В исходном положении перо поднято и находится над точкой (0, 0) — началом координат. После завершения рисования перо также должно быть поднято.

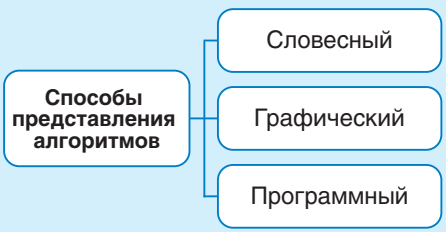
В настоящее время чертежи широко применяются в различных отраслях строительства, сельского хозяйства, промышленности и т. д. Сегодня для построения чертежей используются специальные программы, позволяющие автоматизировать процесс черчения. Вот логотипы подобных программ:

AutoCAD

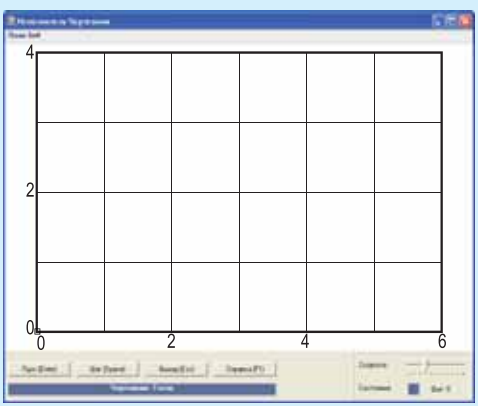

Компас-3D


NanoCAD

Пример 8.1.



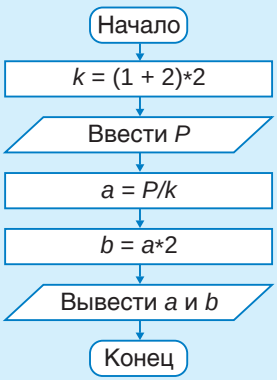
Пример 8.2. Поле исполнителя Чертежник.



Пример 8.3. Запись алгоритма по действиям:

1) 1 + 2 = 3 (части);
2) 3 · 2 = 6 (частей);
3) 120 : 6 = 20 (м);
4) 20 · 2 = 40 (м).

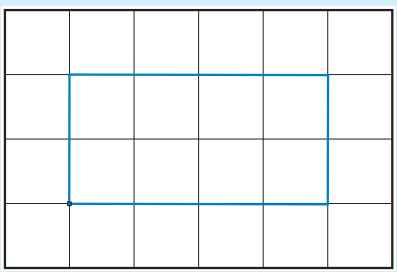
Блок-схема алгоритма:



Программа для исполнителя:

```
uses Drawman;
begin
  Field(6, 4);
  ToPoint(1, 1);
  PenDown;
  OnVector(4, 0);
  OnVector(0, 2);
  OnVector(-4, 0);
  OnVector(0, -2);
  PenUp;
end.
```

Результат работы программы:



Система команд Чертежника:

Команда	Действие
`ToPoint (x,y)`	Переместить перо в точку `(x,y)`
`PenUp`	Поднять перо
`PenDown`	Опустить перо
`Field (n,m)`	Создать поле размером $n \times m$
`OnVector (a,b)`	Сместить перо на a единиц по горизонтали и b единиц по вертикали

Пример 8.3. Составим алгоритм решения задачи.

Прямоугольный участок, длина которого в 2 раза больше ширины, огородили забором длиной 120 м. Определите длину и ширину участка. Напишите программу, выполнив которую исполнитель Чертежник построит чертеж забора этого участка. Масштаб: 1 клетка равна 10 м.

Словесное описание алгоритма:

1. Длина участка в 2 раза больше ширины, поэтому в сумме длина и ширина составят 3 одинаковые части. Забор огораживает участок по периметру, равному удвоенной сумме длины и ширины, т. е. периметр равен 6 одинаковым частям.

2. Ширина: 120 : 6 = 20 м.

3. Длина в 2 раза больше ширины: 20 · 2 = 40 м.

8.3. Алгоритмическая конструкция *следование*

Существует большое количество алгоритмов, в которых все команды выполняются последовательно одна за другой в том порядке, в котором они записаны. В подобных алгоритмах отсутствуют команды, меняющие порядок выполнения других команд. Такие программы вы составляли в прошлом году для исполнителя Чертежник.

Алгоритмическая конструкция *следование* — последовательность команд алгоритма, которые выполняются в том порядке, в котором они записаны.

Алгоритмическая конструкция *следование* отображает естественный, последовательный порядок выполнения действий в алгоритме.

Следование использовалось в примере 8.3, в котором описывались алгоритмы вычисления длины и ширины участка и построения прямоугольника исполнителем Чертежник.

Алгоритмическая конструкция *следование* представлена в примерах 8.4 и 8.5.

Пример 8.4. Алгоритм изготовления бутерброда:

1. Отрезать ломтик батона.
2. Положить на батон лист салата.
3. Отрезать кусочек ветчины.
4. Положить ветчину на лист салата.
5. Отрезать кусочек помидора.
6. Положить помидор на ветчину.

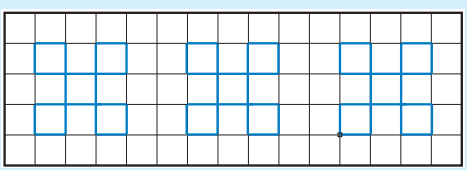
Пример 8.5. Алгоритм выполнения лабораторной работы по биологии «Строение инфузории туфельки»:

1. Рассмотреть внешний вид и внутреннее строение инфузории туфельки.
2. Зарисовать инфузорию туфельку и обозначить названия ее органов.
3. Подвести итог работе.

Пример 8.6. Программа для исполнителя Чертежник будет следующей:

```
uses Drawman;
procedure figura;
begin
  PenDown;
  OnVector(1,  0);
  OnVector(0,  3);
  OnVector(-1,  0);
  OnVector(0,  -1);
  OnVector(3,  0);
  OnVector(0,  1);
  OnVector(-1,  0);
  OnVector(0,  -3);
  OnVector(1,  0);
  OnVector(0,  1);
  OnVector(-3,  0);
  OnVector(0,  -1);
  PenUp;
end;
begin
  Field(15,  5);
  ToPoint(1,  1);
  Figura;
  ToPoint(6,  1);
  Figura;
  ToPoint(11,  1);
  Figura;
end.
```

Результат выполнения программы:



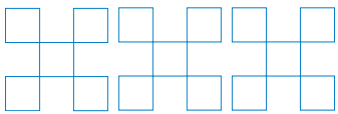
8.4. Вспомогательные алгоритмы

Часто в одной программе нужно рисовать одно и то же изображение несколько раз. Получение этого изображения удобно оформить в виде вспомогательного алгоритма, который можно использовать нужное число раз.

Вспомогательный алгоритм — алгоритм, целиком используемый в составе другого алгоритма.

Вспомогательный алгоритм решает некоторую подзадачу основной задачи. Вызов вспомогательного алгоритма в программе заменяет несколько команд одной.

Пример 8.6. Напишем программу, выполнив которую Чертежник нарисует изображение:



Данный рисунок состоит из одинаковых фигур. Для рисования одной из них можно оформить вспомогательный алгоритм figura.

Описание основного алгоритма:

перемещение в начальную точку;

рисование фигуры;

перемещение ко второй фигуре;

рисование фигуры;

перемещение к третьей фигуре;
рисование фигуры.

При решении задач над проектом могут работать несколько человек. Каждый из членов коллектива делает часть своей работы и оформляет ее как отдельный вспомогательный алгоритм.

Построение алгоритмов часто выполняют **методом пошаговой детализации.** При этом сложная задача разбивается на ряд более простых. Для каждой подзадачи составляется свой вспомогательный алгоритм. Подзадачи могут разбиваться на еще более простые подзадачи.

1. Что такое алгоритм?
2. Какие способы записи алгоритмов вам известны?
3. Что называют алгоритмической конструкцией *следование*?
4. Какой алгоритм называется вспомогательным?
5. Для чего нужны вспомогательные алгоритмы?

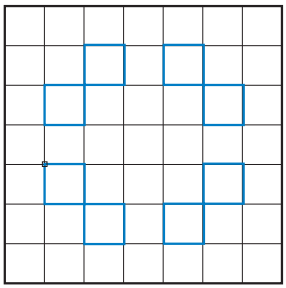
Упражнения

1 Какой рисунок получится после выполнения Чертежником следующей программы? Изобразите рисунок и проверьте правильность своих действий, выполнив программу на компьютере.

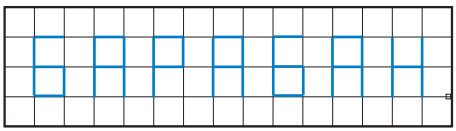
```
uses Drawman;
begin
  Field(8, 8);
  ToPoint(2, 1);
  PenDown;
  OnVector(4, 0);
  OnVector(0, 1);
  OnVector(1, 0);
  OnVector(0, 4);
  OnVector(-1, 0);
  OnVector(0, 1);
  OnVector(-4, 0);
  OnVector(0, -1);
  OnVector(-1, 0);
  OnVector(0, -4);
  OnVector(1, 0);
  OnVector(0, -1);
  PenUp;
end.
```

2 Напишите для исполнителя Чертежник программы получения следующих изображений:

а



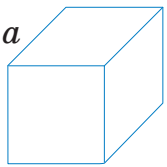
б



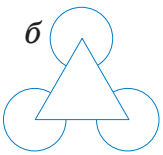
3 Придумайте свои рисунки и составьте программы для их рисования с помощью исполнителя Чертежник.

4* Проанализируйте рисунки. Какие из них мог выполнить исполнитель Чертежник? Почему? Какие команды вы можете предложить добавить исполнителю для выполнения остальных рисунков?

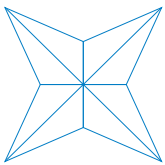
а



б



в



г

§ 9. Исполнитель Робот

9.1. Роботы в жизни человека

Роботы развозят заказы в ресторане в г. Харбин (Китай)[1].

Человек с глубокой древности мечтал об искусственном создании, которое могло бы выполнять его приказы. Сегодня эта мечта стала реальностью — в жизни людей появились роботы. Они способны выполнять практически любую работу, доступ-

[1]Материалы о роботах взяты с сайтов http://www.robogeek.ru и http://fishki.net/1211999-roboty-v-nashej-zhizni.html (дата доступа: 07.02.2017).

ную человеку, а также делать многие вещи, которые людям выполнить сложно или вообще невозможно. Роботы используются на производстве и в быту, могут работать в сфере услуг и развлечений. Есть роботы, похожие на человека, а есть совсем непохожие.

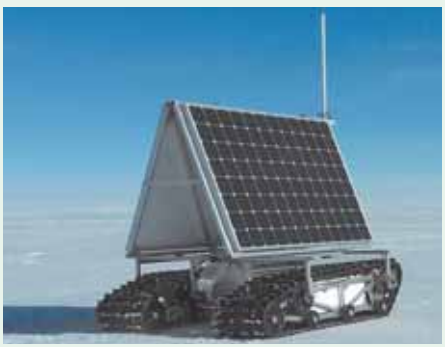
Робот — автоматическое устройство, которое действует по заранее составленной программе.

Робот получает информацию о внешнем мире от датчиков — аналогов органов чувств живых организмов — и предназначен для осуществления различных операций.

Мир роботов очень разнообразен. В быту современного человека используются автоматические стиральные и посудомоечные машины, роботы-пылесосы и др. С помощью роботов можно выращивать растения или управлять домом.

Робот может быть материальным или виртуальным. Виртуальный робот — специальная программа, выполняющая определенные действия.

Робот LS3, созданный для транспортировки грузов по пересеченной местности.



Автономный робот GROVER, который изучает слои льда на ледниковом щите Гренландии.

Роботизированная система, предназначенная для выращивания овощей. Управление данной системой осуществляется через Wi-Fi. Есть возможность удаленного контроля через Интернет.

Робот-пылесос с помощью системы камер и сенсоров может ориентироваться в комнате и строить маршрут уборки.

Роботы являются исполнителями. Для исполнителей обычно определяют среду обитания и систему команд.

Общим для всех роботов является то, что человек может ими управлять. Робот получает команды от оператора и выполняет их по одной или действует автономно по предварительно составленной программе.

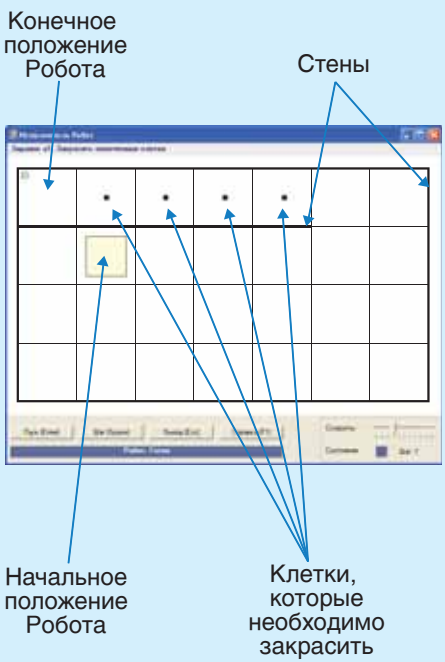
9.2. Среда обитания и система команд исполнителя Робот

В среде программирования PascalABC, кроме исполнителя Чертежник, можно выбрать исполнителя Робот.

Средой обитания исполнителя Робот является прямоугольное клетчатое поле. Размеры поля, как и для исполнителя Чертежник, задаются командой `Field(n, m)`. Первоначально Робот находится в центральной клетке поля.

Между некоторыми клетками, а также по периметру поля находятся стены. Робот может передвигаться по полю и закрашивать указанные клетки. Большой желтый квадрат внутри клетки означает начальное положение Робота, маленький — конечное (пример 9.1).

Пример 9.1. Поле исполнителя Робот с начальной обстановкой имеет следующий вид:



Поле Робота, на котором определено положение стен, начальное и конечное положение исполнителя, называют **обстановкой.**

Для подключения исполнителя Робот в программе прописывается команда **uses** Robot. Готовые задания с обстановками для Робота хранятся в задачнике, встроенном в систему программирования, и вызываются командой task. Эта же команда использовалась для Чертежника.

Система команд исполнителя:

Команда	Действие
Right	Перемещает Робота вправо
Left	Перемещает Робота влево
Up	Перемещает Робота вверх
Down	Перемещает Робота вниз
Paint	Закрашивает текущую ячейку

Робот может становиться на обычную и на закрашенную клетку, но не может переместиться с клетки на клетку, если между ними стена. Робот не может переместиться за границы поля. Эти действия вызывают ошибку (пример 9.2). Робот может закрасить уже закрашенную клетку. Такое действие ошибку не вызывает.

Пример 9.2. Вызов задачи a1 из встроенного задачника:

•Program1.pas

```
uses Robot;
begin
  Task('a1');
end.
```

Запись команд исполнителя:

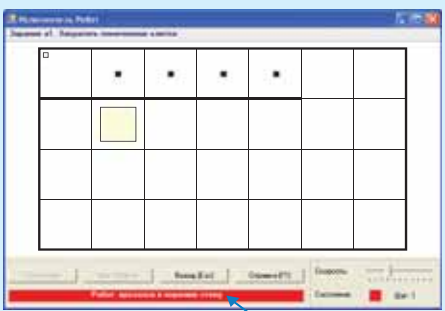
•Program1.pas*

```
uses Robot;
begin
  Task('a1');
  right;
  right;
end.
```

Запишем в программе команду up.

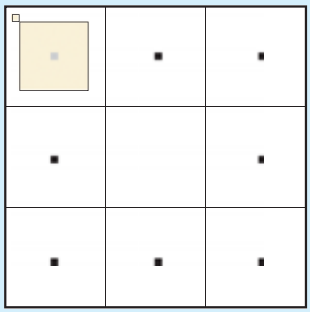
```
uses Robot;
begin
  Task('a1');
  up;
end.
```

Сверху находится стена, поэтому перемещение Робота вверх невозможно:



Сообщение об ошибке

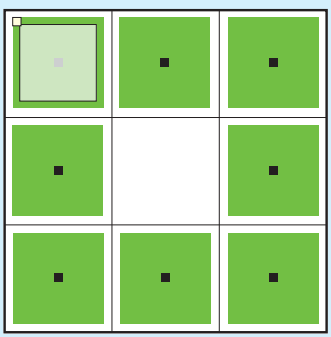
Пример 9.3. Начальная обстановка:



Программа для исполнителя Робот:

```
uses Robot;
begin
  Task('a2');
  paint; right;
  paint; right;
  paint; down;
  paint; down;
  paint; left;
  paint; left;
  paint; up;
  paint; up;
end.
```

Результат работы программы имеет следующий вид:



9.3. Использование алгоритмической конструкции *следование* для исполнителя Робот

Рассмотрим примеры решения задач для исполнителя Робот.

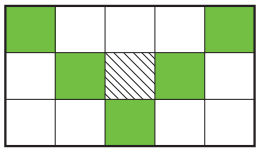
Пример 9.3. Робот находится на поле размером 3 × 3 клетки. Нужно закрасить все клетки, кроме средней (задача а2).

Для решения задачи Робот должен выполнить следующий алгоритм:

```
закрасить;
вправо;
закрасить;
вправо;
закрасить;
вниз;
закрасить;
вниз;
закрасить;
влево;
закрасить;
влево;
закрасить;
вверх;
закрасить;
вверх.
```

В данном алгоритме Робот обходит клетки, двигаясь по часовой стрелке. Тот же результат можно получить, если Робот будет обходить поле против часовой стрелки, изначально двигаясь вниз.

Пример 9.4. Составим программу для закрашивания клеток поля Робота по образцу:



Такой обстановки нет в задачнике, поэтому вначале нужно создать поле Робота размером 5 × 3. Начальное положение Робота на таком поле отмечено заштрихованной клеткой.

Для решения задачи Робот должен выполнить алгоритм:

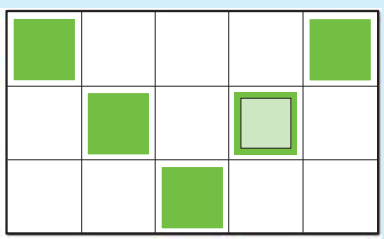
```
создать поле;
вниз;
закрасить;
влево;
вверх;
закрасить;
влево;
вверх;
закрасить;
вправо;
вправо;
вправо;
вправо;
закрасить;
влево;
вниз;
закрасить.
```

**Какими еще способами можно решить данную задачу?*

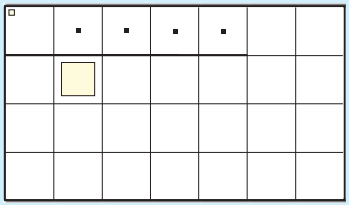
Пример 9.4. В данном случае программа для учебного компьютерного исполнителя Робот может быть составлена таким образом:

```
uses Robot;
begin
  Field(5, 3);
  down;
  paint;
  left;
  up;
  paint;
  left;
  up;
  paint;
  right;
  right;
  right;
  right;
  paint;
  left;
  down;
  paint;
end.
```

Результат работы записанной выше программы будет иметь следующий вид:



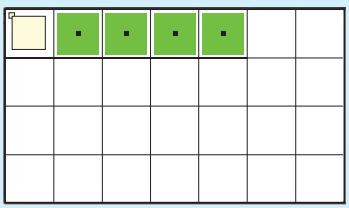
Пример 9.5. Начальная обстановка:



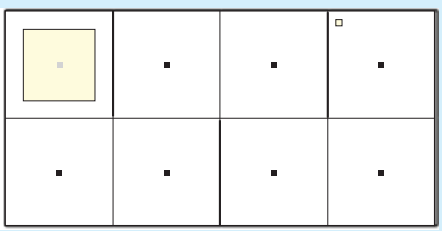
Программа для Робота:

```
uses Robot;
begin
  Task('a1');
  right;  right;
  right;  right;
  up;
  left;  paint;
  left;  paint;
  left;  paint;
  left;  paint;
  left;
end.
```

Результат работы программы:



Пример 9.6. Начальная обстановка:



Пример 9.5. Решим задачу a1 из встроенного задачника.

Для решения данной задачи Робот должен обойти линию (стену), закрасить указанные клетки и переместиться в клетку, определяющую конечное положение исполнителя.

Алгоритм решения задачи:

```
сдвинуться вправо на 4 клетки;
вверх;
сдвинуться влево на 4 клетки, закрашивая их по пути;
влево.
```

В примерах 9.3 — 9.5 команды исполнителя Робот выполнялись последовательно, одна за другой, в том порядке, в котором они были записаны. Поэтому все приведенные алгоритмы реализованы с использованием алгоритмической конструкции *следование*.

9.4. Вспомогательные алгоритмы

Пример 9.6. Решим задачу a3 из встроенного задачника.

Робот должен закрасить все клетки поля. Но двигаться по прямой ему мешают стены, которые исполнитель должен обходить.

Алгоритм решения задачи:

```
закрасить;  вниз;
закрасить;  вправо;
```

```
закрасить;  вверх;
закрасить;
вправо;
закрасить;  вниз;
закрасить;  вправо;
закрасить;  вверх;
закрасить.
```

Если проанализировать данный алгоритм, то можно заметить, что дважды повторяется последовательность команд, которая закрашивает квадрат из четырех клеток:

```
закрасить;  вниз;
закрасить;  вправо;
закрасить;  вверх;
закрасить.
```

Оформим эти команды как вспомогательный алгоритм, который назовем **квадрат**. Тогда алгоритм решения исходной задачи может быть записан так:

```
квадрат;
вправо;
квадрат.
```

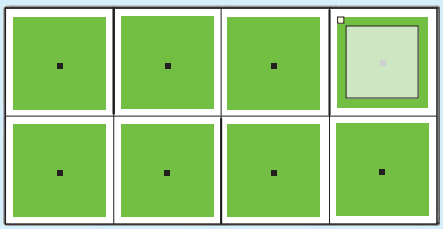
При решении данной задачи использование вспомогательного алгоритма позволило не записывать дважды одну и ту же последовательность команд.

Вспомогательные алгоритмы можно использовать и в том случае, когда исходная задача разбивается на несколько независимых друг от друга задач. Тогда

Программа 1 для исполнителя Робот:

```
uses Robot;
begin
  Task('a3');
  paint;  down;
  paint;  right;
  paint;  up;
  paint;
  right;
  paint;  down;
  paint;  right;
  paint;  up;
  paint;
end.
```

Результат работы программы:



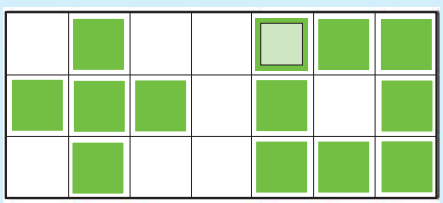
Программа 2 (с использованием вспомогательного алгоритма) для исполнителя Робот:

```
uses Robot;
procedure kvadrat;
begin
  paint;  down;
  paint;  right;
  paint;  up;
  paint;
end;
begin
  Task('a3');
  kvadrat;
  right;
  kvadrat;
end.
```

Пример 9.7. Программа для учебного компьютерного исполнителя Робот:

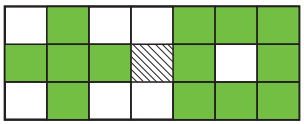
```
uses Robot;
procedure krest;
begin
  left; paint;
  down; left; paint;
  up; left; paint;
  right; paint;
  up; paint;
end;
procedure kvadrat;
begin
  paint; right;
  paint; right;
  paint; down;
  paint; down;
  paint; left;
  paint; left;
  paint; up;
  paint; up;
end;
begin
  field(7,3);
  krest;
  right; right; right;
  kvadrat;
end.
```

Результат работы записанной выше программы будет иметь следующий вид:



каждую из них можно оформить как вспомогательный алгоритм.

Пример 9.7. Напишем программу для закрашивания клеток поля Робота по образцу:



Такой обстановки нет в задачнике, поэтому создадим поле 7×3. Начальное положение Робота отмечено заштрихованной клеткой.

В данной задаче Робот должен нарисовать две отдельные фигуры: крест и квадрат. Составим два вспомогательных алгоритма.

Вспомогательный алгоритм **крест:**

```
влево; закрасить;
вниз; влево; закрасить;
вверх; влево; закрасить;
вправо; закрасить;
вверх; закрасить.
```

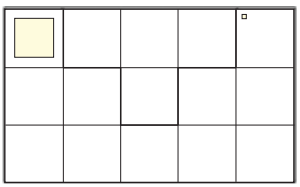
В качестве вспомогательного алгоритма для рисования квадрата можно использовать алгоритм решения задачи а2 (пример 9.3). Для перехода от одной фигуры к другой Робот должен сдвинуться на 3 клетки вправо:

```
крест;
вправо; вправо; вправо;
квадрат.
```

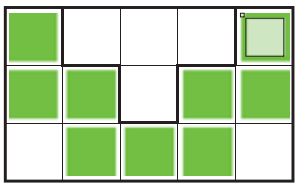
1. Что такое робот?
2. Какие команды входят в систему команд учебного компьютерного исполнителя Робот?
3. Опишите среду обитания учебного исполнителя Робот.

Упражнения

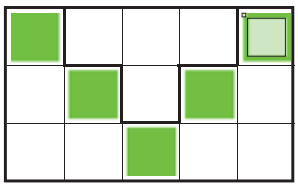
1 Начальная обстановка на поле Робота изображена на рисунке справа. Трое учащихся составили и выполнили алгоритм, по которому Робот закрасил все клетки пути от начальной к конечной. На каком из рисунков — *а*, *б* или *в* — изображено решение данной задачи? Почему?



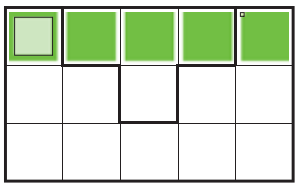
а



б



в



2 Какой из приведенных алгоритмов решает задачу, сформулированную в предыдущем задании? Объясните, почему другие программы не могут быть алгоритмами решения данной задачи.

а)
```
paint;  down;
right;
paint;  down;
right;
paint;  right;
up;
paint;  right;
up;
paint;
```

б)
```
paint;  down;
paint;  right;
paint;  down;
paint;  right;
paint;  right;
paint;  up;
paint;  right;
paint;  up;
paint;
```

в)
```
ToPoint(0,3);
PenDown;
OnVector(1,0);
OnVector(0,-1);
OnVector(1,0);
OnVector(0,-1);
OnVector(1,0);
OnVector(0,1);
OnVector(1,0);
OnVector(0,1);
OnVector(1,0);
```

3 Для какого исполнителя приведен алгоритм в задании 2, *в*? Сформулируйте для этого исполнителя задачу, решением которой будет приведенный алгоритм.

4 Для исполнителя Робот была составлена следующая программа:

```
paint;
right; up;
paint;
right; down;
```

Изобразите в тетради «узор», который нарисует Робот. При каких минимальных размерах поля Робот сможет выполнить данную программу?

5 Все команды в программе из задания 4 учащийся скопировал три раза. Как изменится «узор» после выполнения программы? Как можно по-другому записать этот алгоритм? Какого размера поле нужно создать? П о д с к а з к а. Воспользуйтесь вспомогательным алгоритмом.

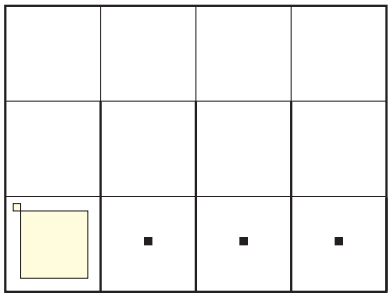
6 Программа решения задачи была записана на доске. Два учащихся, переписывая этот алгоритм для исполнителя Робот, пропустили из-за невнимательности по одной команде. Какую команду пропустил каждый из учащихся? Что будет результатом работы каждой программы?

Программа, записанная первым учащимся	Программа, записанная вторым учащимся
uses Robot; **begin** Field(15,15); paint; right; paint; right; paint; down; paint; down; paint; down; left; down; left; paint; down; paint; down; paint; right; paint; right; paint; up; paint; up; paint; up; left; up; left; paint; up; paint; up; **end.**	**uses** Robot; **begin** Field(15,15); paint; right; paint; right; paint; down; paint; down; paint; down; left; paint; down; left; paint; down; paint; down; paint; right; paint; right; paint; up; paint; up; paint; up; left; up; paint; up; paint; up; **end.**

7 Составьте программу для решения задачи а4 из встроенного задачника (см. рис. справа). Предложите два алгоритма:

1. С использованием алгоритмической конструкции *следование*.
2. С использованием вспомогательного алгоритма.

Сравните полученные решения.



8* Робот-огородник может разбить грядку на посадочные зоны-клетки. На рисунке справа изображена схема посадки овощей (красная клетка — томаты, зеленая — огурцы). Предложите систему команд для робота-огородника и разработайте алгоритм посадки овощей (робот сажает одно растение в одну клетку).

§ 10. Алгоритмическая конструкция *повторение*

10.1. Алгоритмы с циклами

В окружающем мире можно наблюдать ситуации, при которых различные действия и процессы повторяются. Некоторые повторяются несколько раз и завершаются. Другие могут повторяться очень долго (например, круговорот воды в природе, движение планет в космическом пространстве, смена времен года и т. д.). Человеку тоже регулярно приходится выполнять повторяющиеся действия: умываться, одеваться, посещать парикмахерскую, завтракать, ходить на работу и др.

Понятие цикла используется в различных сферах человеческой деятельности.

Под циклом понимают совокупность явлений, процессов, составляющих кругооборот в течение определенного промежутка времени. С этой точки зрения можно говорить о годовом цикле вращения Земли вокруг Солнца или о производственном цикле.

Циклом является законченный ряд каких-либо произведений, чего-либо излагаемого, исполняемого: цикл лекций, цикл стихотворений.

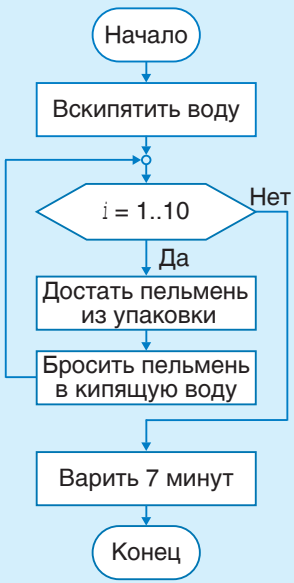
Пример 10.1. Приготовление пельменей.

Алгоритм:

1. Вскипятить воду.
2. Для i = 1..10 повторять:
   2.1. Достать пельмень из упаковки.
   2.2. Бросить пельмень в кипящую воду.
3. Варить 7 минут.

В данном примере параметр цикла i изменяется от 1 до 10. Действия «достать пельмень из упаковки» и «бросить пельмень в кипящую воду» выполняются 10 раз и составляют тело цикла.

Блок-схема данного алгоритма выглядит таким образом:



Как правило, человек составляет программы, в которых каждая команда в отдельности и весь алгоритм в целом выполняются за конечное число повторений.

Алгоритмическая конструкция *повторение* (цикл) определяет последовательность действий, выполняемых многократно. Эту последовательность действий называют **телом цикла.**

Существует несколько возможностей управлять тем, сколько раз будет повторяться тело цикла.

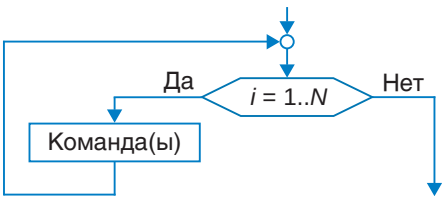
Алгоритмическая конструкция *цикл с параметром (цикл со счетчиком)* — способ организации цикла, при котором количество повторов зависит от начального и конечного значений параметра цикла.

Таким образом, цикл с параметром организует выполнение команд тела цикла заранее известное число раз (пример 10.1).

Параметр цикла определяет нумерацию действий в цикле. Параметр цикла может принимать только целые значения. Часто нумерацию начинают с 1 и закан-

чивают числом N (пример 10.2). В этом случае цикл выполнится N раз. Если нумерация установлена двумя произвольными числами $N1$ (начальное значение) и $N2$ (конечное значение), то цикл выполнится $(N2 - N1 + 1)$ раз.

Алгоритмическая конструкция цикла с параметром может изображаться на блок-схеме следующим образом (значение параметра изменяется от 1 до N):



В данной конструкции в прямоугольнике(-ах) записываются повторяющиеся команды алгоритма (тело цикла), которые выполняются N раз (Да). При этом после каждого выполнения команд тела цикла происходит проверка, который раз выполняется цикл. На блок-схеме переход на проверку условия изображается в виде стрелки, выходящей из тела цикла и возвращающейся к проверке. Как только команды тела цикла выполнятся N раз (Нет), цикл завершится (пример 10.3). Если $N \leqslant 0$, то команда тела цикла не выполнится ни разу.

Пример 10.2. Вычислим a^n (например, $3^5 = 243$).

Алгоритм возведения числа в степень:

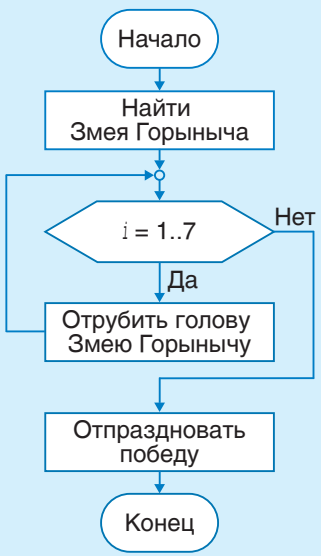
1. Ввести значения a и n.
2. Определить начальное значение результата $r = 1$.
3. Для $i = 1..n$ повторять:
   3.1. Умножить результат на a.
4. Записать результат.

Пример 10.3. В фольклорных произведениях часто встречается многоголовый Змей Горыныч (количество голов может быть, например, 7).

Алгоритм победы над Змеем Горынычем:

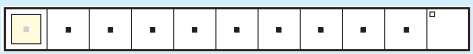
1. Найти Змея Горыныча.
2. Для $i = 1..7$ повторять:
   2.1. Отрубить голову Змею Горынычу.
3. Отпраздновать победу.

Блок-схема данного алгоритма:



Многие роботы, которые используются в быту и на производстве, могут выполнять циклические алгоритмы. Примером такого робота является суши-робот, который может производить от 450 до 4000 заготовок для суши за 1 час.

Пример 10.4. Начальная обстановка:



Программа для исполнителя Робот:

```
uses Robot;
begin
  Task('c2');
  for var i:= 1 to 10 do
  begin
    paint;
    right;
  end;
end.
```

Результат работы программы:

10.2. Использование команды цикла с параметром для исполнителя Робот

Чтобы составлять алгоритмы с циклами для компьютерного исполнителя Робот, нужно знать, как записывается команда цикла.

Для записи цикла с параметром используется команда **for**. Формат записи команды:

```
for var i:= N1 to N2 do[1]
begin
  тело цикла;
end;
```

Строка **for var** i:= N1 **to** N2 **do** является заголовком цикла. Его читают так: «Для переменной *i* от *N*1 до *N*2 делай». Если $N2 \geqslant N1$, то команды тела цикла выполнятся $(N2 - N1 + 1)$ раз, иначе цикл не выполнится ни разу.

Слова **begin** и **end;** являются операторными скобками в языке Pascal. Если тело цикла состоит из одной команды, операторные скобки можно опустить.

Операторные скобки — пара слов, определяющих в языке программирования блок команд, воспринимаемый как единое целое, как одна команда.

Пример 10.4. Решим задачу с2 из встроенного задачника.

[1] Команда в таком формате записывается только в среде PascalABC.Net.

Робот должен закрасить все клетки поля (кроме последней), перемещаясь вправо. Для этого в цикле нужно 10 раз выполнить команды:

закрасить;
вправо.

Данные команды образуют тело цикла.

Командами, образующими тело цикла, могут быть любые команды из системы команд исполнителя. Кроме того, в теле цикла может вызываться вспомогательный алгоритм. Использование вспомогательного алгоритма позволит сократить запись тела цикла и сделает программу более понятной.

Пример 10.5. Решим задачу с7 из встроенного задачника.

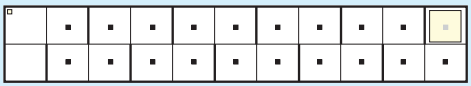
На поле исполнителя Робот есть стены. При обходе стен Робот выполняет следующие команды:

закрасить; вниз;
закрасить; влево;
закрасить; вверх;
закрасить; влево.

Чтобы решить задачу, Робот должен повторить эти команды 5 раз. Оформим данные команды как вспомогательный алгоритм kvadrat и вызовем его в цикле.

В данном примере тело цикла состоит из одной команды kvadrat, поэтому операторные скобки можно не использовать.

Пример 10.5. Начальная обстановка для учебного компьютерного исполнителя Робот:



Программа для исполнителя Робот составляется следующим образом:

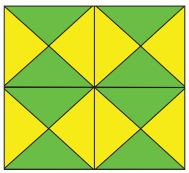
```
uses Robot;
procedure kvadrat;
begin
  paint;
  down;
  paint;
  left;
  paint;
  up;
  paint;
  left;
end;
begin
  Task('c7');
  for var i:= 1 to 5 do
    kvadrat;
end.
```

Результат работы записанной выше программы будет иметь следующий вид:

1. Что понимают под алгоритмической конструкцией *повторение*?
2. Что такое цикл с параметром?
3. Что такое операторные скобки?
4. Приведите примеры использования цикла.

Упражнения

1 Опишите словесно или изобразите с помощью блок-схемы следующие алгоритмы:

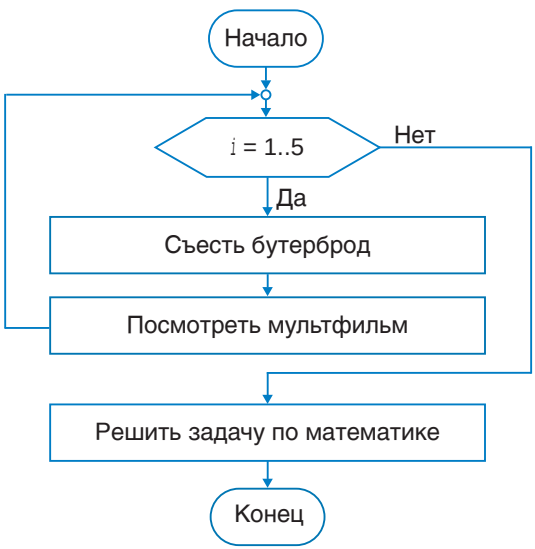


1. Рисование в графическом редакторе изображения из 4 квадратов с диагоналями и закрашенными областями (см. рис. справа).
2. Каждую минуту бактерия делится на две. Изначально есть одна бактерия. За бактериями наблюдали 10 минут. Определите количество бактерий в конце наблюдения. Заполните таблицу.

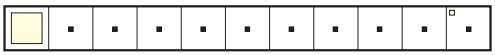
Время, мин	0	1	2	3	4	5	6	7	8	9	10
Количество бактерий	1	2									

3. Сверление 10 отверстий.
4. Сервировка стола к обеду на 6 персон.

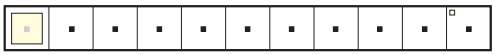
2 Семиклассник Андрей после школы пригласил своего друга Юру помочь ему в решении 5 задач по математике. В гостях Юра посоветовал Андрею провести остаток дня, воспользовавшись алгоритмом, записанным в виде блок-схемы (см. рис. справа). Объясните, почему за выполнения этого задания Андрей получил двойку по математике.



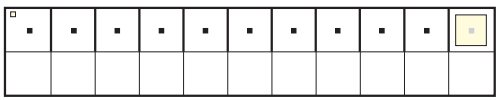
3 Составьте программу для решения задачи с3 из встроенного задачника. Сравните алгоритм решения этой задачи с примером 10.4. Что у них общего? Чем они отличаются?



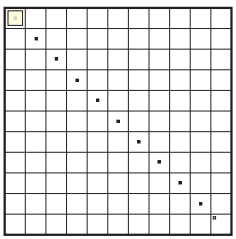
4 Составьте программу для решения задачи с4 из встроенного задачника. Сравните ее решение с предыдущим упражнением и с примером 10.4.



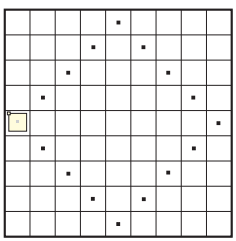
5 Составьте программу для решения задачи с8 из встроенного задачника. Используйте вспомогательный алгоритм.



6 Составьте программу для решения задачи с5 из встроенного задачника.



7 Для решения задачи с14 Петя составил алгоритм и записал программу. Петин младший брат Олег удалил несколько команд. Определите, сколько команд удалил Олег. Восстановите программу, которую написал Петя.



```
uses Robot;
begin
 Task('c14');
 paint;
 for var i:= 1 to 4 do
 begin
  paint;
  right;
  down;
 end;
 for var i:= 1 to 4 do
 begin
  right;
  up;
 end;
 for var i:= 1 to 4 do
 begin
  paint;
 end;
end.
```

8 Максим пытается представить, как можно было бы использовать роботов в различных ситуациях, описанных в литературных произведениях. Например, для Тома Сойера, которого тетушка Полли отправила красить забор, Максим придумал робота-маляра и решил, что такому роботу достаточно одной команды: покрась доску. Алгоритм покраски забора из 20 досок Максим записал так:

1. Установить робота у левого края забора.
2. Для i = 1..20 повторять:
   2.1. Покрась доску.

Сможет ли робот-маляр покрасить забор? В чем ошибка Максима? Исправьте алгоритм, добавив необходимую(-ые) команду(-ы).

§ 11. Использование условий

Условия используются в правилах дорожного движения. Так, если горит зеленый свет, то можно переходить улицу.

Условия также встречаются в фольклоре, например при выборе пути сказочными героями.

В. Васнецов. «Витязь на распутье» (фрагмент). 1882 г.

11.1. Понятие условия

Принятие решений зачастую зависит от различных условий. Если на улице дождь, то нужно взять зонт; если хорошо подготовился к уроку, то получишь высокую отметку, иначе низкую и др.

Человек способен понимать условия, сформулированные в произвольной форме. Но для того чтобы Робот или другой исполнитель мог принимать решения, нужно «научить» его «понимать» условия.

Условием для исполнителя является понятное ему высказывание, которое может быть истинным (соблюдаться) либо ложным (не соблюдаться).

Исполнитель может проверить истинность условий, входящих в его систему условий.

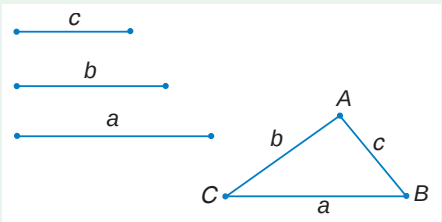
Рассмотрим систему условий для исполнителя Робот.

`WallFromLeft`	Истинно, если слева от Робота стена
`WallFromRight`	Истинно, если справа от Робота стена
`WallFromUp`	Истинно, если сверху от Робота стена
`WallFromDown`	Истинно, если снизу от Робота стена
`FreeFromLeft`	Истинно, если слева от Робота свободно
`FreeFromRight`	Истинно, если справа от Робота свободно
`FreeFromUp`	Истинно, если сверху от Робота свободно
`FreeFromDown`	Истинно, если снизу от Робота свободно
`CellIsPainted`	Истинно, если ячейка, в которой находится Робот, закрашена
`CellIsFree`	Истинно, если ячейка, в которой находится Робот, не закрашена

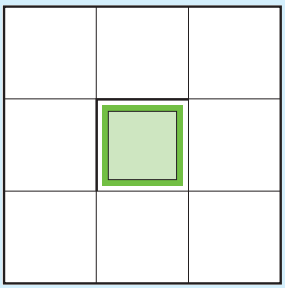
Образцы истинных и ложных условий для исполнителя Робот представлены в примере 11.1.

Применяются условия в математике, например:

Треугольник существует, если для большей стороны a выполняется неравенство $a < b + c$.



Пример 11.1. Рассмотрим начальную обстановку поля исполнителя Робот:



В данном случае для Робота будут истинны следующие условия:

```
WallFromLeft
WallFromUp
FreeFromRight
FreeFromDown
CellIsPainted
```

Ложными при такой начальной обстановке будут условия:

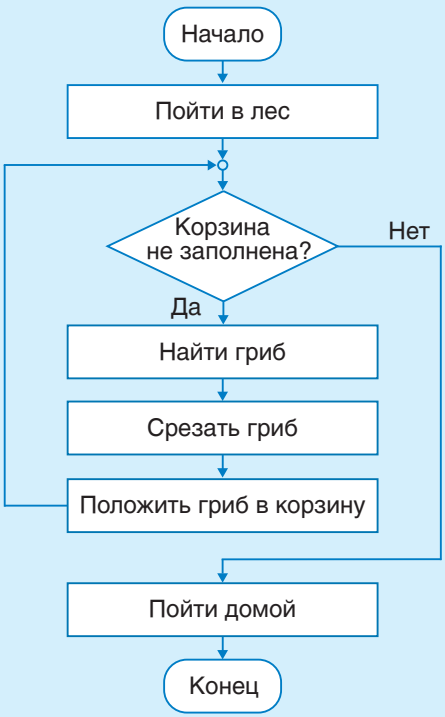
```
WallFromRight
WallFromDown
FreeFromLeft
FreeFromUp
CellIsFree
```

Пример 11.2. Сбор грибов.

Использование цикла с параметром при составлении алгоритма решения этой задачи может привести к разным результатам.

Корзина может быть или полупустой, или не все найденные грибы в нее поместятся.

Если использовать цикл с предусловием, то в результате домой можно унести полную корзину грибов.



11.2. Цикл с предусловием

Цикл с параметром используется при составлении алгоритма в том случае, когда заранее известно количество повторений. Однако часто до выполнения цикла количество повторений не известно.

Пример 11.2. Вы с родителями пошли в лес за грибами. Ваши действия можно описать командами: найти гриб, срезать гриб, положить гриб в корзину. Эти действия будут выполняться в цикле, но вы заранее не знаете, сколько грибов войдет в корзину. Поэтому следует говорить не о количестве повторений (количестве грибов), а об условии, при котором вы будете продолжать сбор грибов: пока корзина не заполнена.

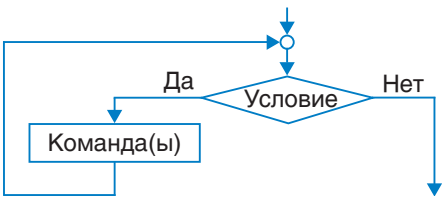
Алгоритмическая конструкция *цикл с предусловием (цикл «пока»)* — способ организации цикла, при котором количество выполнений команд тела цикла зависит от истинности или ложности условия цикла.

Цикл с предусловием используется, когда количество повторений тела цикла заранее не известно, но известно условие продолжения работы.

Условие цикла определяет, как долго будет выполняться цикл.

Пока условие истинно, выполняются команды, составляющие тело цикла. Цикл прекращает выполняться тогда, когда условие становится ложным. Цикл с предусловием имеет такое название, поскольку проверка условия предваряет выполнение команд тела цикла.

Алгоритмическая конструкция цикла с предусловием изображается на блок-схеме так:



В данной конструкции в прямоугольнике(-ах) записываются повторяющиеся команды алгоритма (тело цикла), которые совершаются, пока верно условие (Да). При этом после каждого выполнения команд тела цикла происходит проверка, истинно ли условие. Как только условие станет ложным (Нет), цикл завершается. Если условие сразу ложно, то цикл не выполнится ни разу.

Если условие в цикле будет всегда истинно (всегда Да), то такой цикл не сможет завершиться. Возникшую ситуацию называют **зацикливанием**.

Российский академик Андрей Андреевич Марков (младший) (1903—1979) в своих исследованиях в области теории алгоритмов показал, что в общем случае алгоритмы должны содержать предписания двух видов:

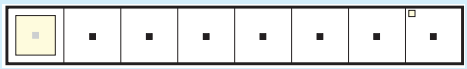
1) функциональные операторы, направленные непосредственно на преобразование информации;

2) логические операторы, определяющие дальнейшее направление действий.

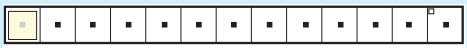
Оператор — элемент языка, задающий полное описание действия, которое необходимо выполнить. В английском языке данное понятие обозначается словом *statement*, означающим также ‘предложение’.

Если применить вышесказанное к компьютерным исполнителям, то предписания первого вида составляют систему команд исполнителя, а предписания второго вида — систему условий исполнителя.

Пример 11.3. Одна из возможных начальных обстановок:



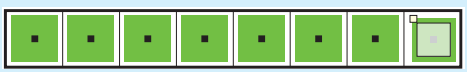
Другая возможная начальная обстановка:



Запишем программу для учебного компьютерного исполнителя Робот:

```
uses Robot;
begin
 Task('w2');
 while FreeFromRight do
 begin
  paint;
  right;
 end;
 paint;
end.
```

Результат работы указанной выше программы для первой начальной обстановки будет иметь следующий вид:



Результат работы программы для второй начальной обстановки:

Для записи цикла с предусловием используется команда **while**. Формат записи команды:

```
while <условие> do
begin
  тело цикла;
end;
```

Строка **while** <условие> **do** является заголовком цикла. Эту строку можно прочитать следующим образом: «Пока верно условие, делай». Команды **begin** и **end;** в данном случае играют роль операторных скобок.

Пример 11.3. Напишем программу для решения задачи w2 из встроенного задачника.

Робот должен закрасить коридор переменной длины.

В данной задаче нам точно не известна длина коридора, но известно, что Робот может двигаться, пока справа пусто, и закрашивать клетки:

```
Пока справа пусто, повторять
  закрасить;
  вправо.
```

После прохода всего коридора Робот должен закрасить последнюю клетку. Это происходит после выполнения цикла, так как для последней клетки условие «справа пусто» уже не выполняется.

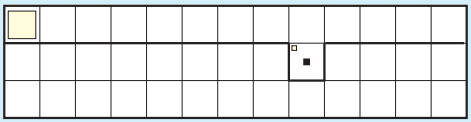
Пример 11.4. Напишем программу для решения следующей задачи. Робот находится в верхнем левом углу поля. Снизу от него вдоль всего поля расположена стена с проходом в одну клетку. Составить алгоритм, выполнив который Робот сможет пройти через проход и закрасить клетку. Расположение прохода заранее не известно.

Проход не ограничен стеной снизу. Робот может двигаться вправо, пока внизу есть стена:

Пока снизу стена, **повторять** вправо.

Робот остановится в той клетке, у которой снизу нет стены. После этого Робот должен сдвинуться вниз и закрасить клетку.

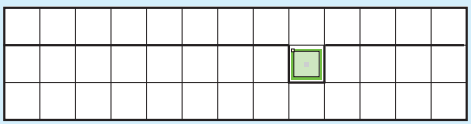
Пример 11.4. Одна из возможных начальных обстановок:



Программа для исполнителя Робот:

```
uses Robot, RobTasks[1];
begin
 Task('myrob3');
 while WallFromDown do
  right;
 down;
 paint;
end.
```

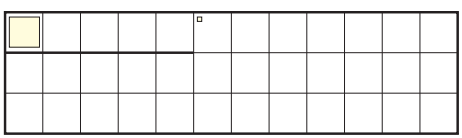
Результат работы программы будет следующим:

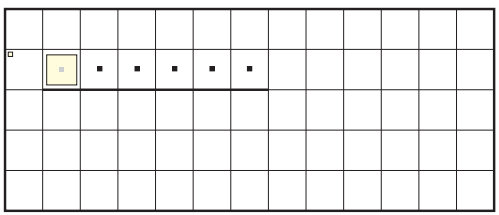


1. Что понимают под условием для исполнителя?
2. Когда используется цикл с предусловием?
3. В каком случае возникает ситуация зацикливания?

Упражнения

1 Напишите программу для решения задач w3 и w8 из встроенного задачника. Обращайте внимание на начальное и конечное положение Робота.





[1] Модуль RobTasks, содержащий данную обстановку и задачу, можно скачать по адресу: http://e-vedy.adu.by/course/view.php?id=423.

2 Для исполнителя Робот был написан следующий алгоритм:

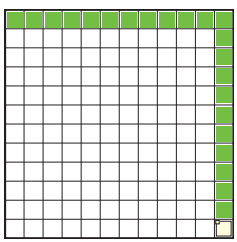
```
uses Robot;
begin
 Field( , );
 while FreeFromRight do
 begin
  paint;
  down;
  right;
  paint;
  up;
  right;
 end;
end.
```

Нарисуйте в тетради результат работы данного алгоритма. Какими должны быть размеры поля, чтобы Робот не врезался в стену? Определите начальное положение Робота.

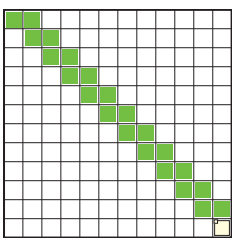
3 Составьте алгоритм, выполнив который Робот нарисует «узор» из задания 2 вдоль левого края поля исполнителя. Каким должен быть вертикальный размер поля исполнителя? (Задача myrob5 из модуля RobTasks.)

4 Робот находится на квадратном поле неизвестного размера. Начальное положение Робота — верхний левый угол. Составьте и выполните алгоритм, по которому Робот переместится из начального положения в нижний правый угол и закрасит все клетки своего пути. На каком (на каких) из рисунков изображено решение этой задачи? Почему?

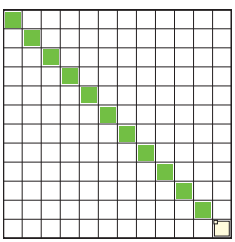
а



б

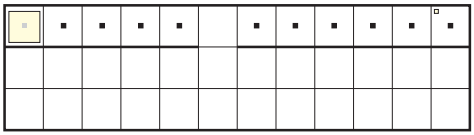


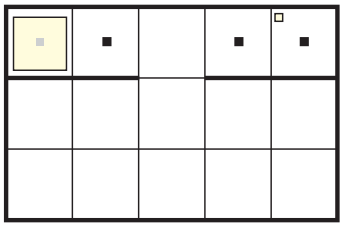
в



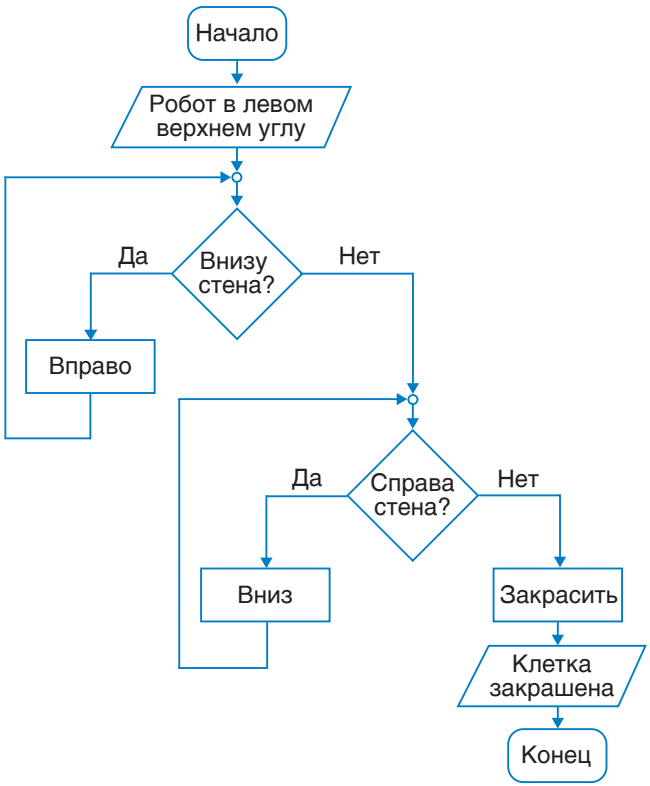
5 На поле Робота размещен «забор» — горизонтальная стена. Забор нужно «покрасить» — закрасить все клетки сверху стены. В «заборе» могут быть одни «ворота» — клетка без линий. Длина «забора» и

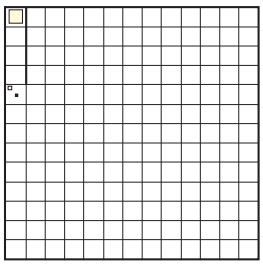
расположение «ворот» не известны. (Задача myrob7 из модуля RobTasks.)

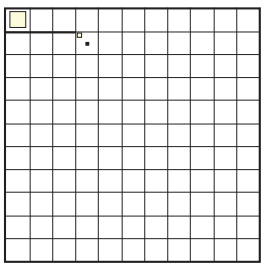


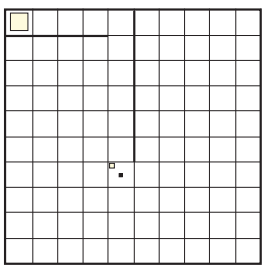


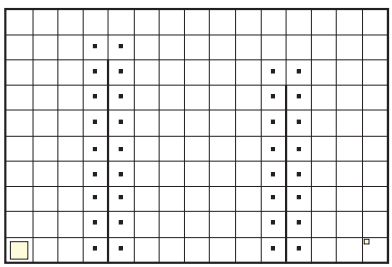
6 По блок-схеме запишите программу для исполнителя Робот. Каким будет результат для каждой из предложенных начальных обстановок? (Задача myrob8 из модуля RobTasks.)









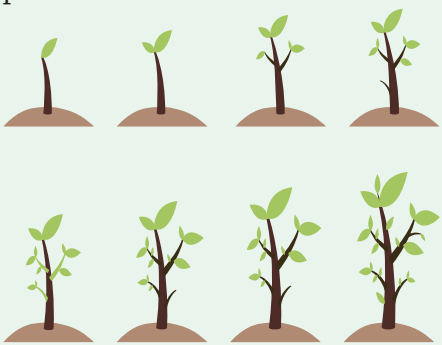


7* Решите задачу w10 из встроенного задачника. Напишите вспомогательный алгоритм для обхода одной стены.

§ 12. Алгоритмическая конструкция *ветвление*

Понятие ветвления используется в различных сферах человеческой деятельности.

В ботанике под ветвлением побегов понимают процесс образования боковых побегов у растений.



При употреблении термина в переносном смысле под ветвлением понимают наличие нескольких путей, направлений, сюжетных линий и т. д.

Ветвления используются в дорожной разметке и картографии.

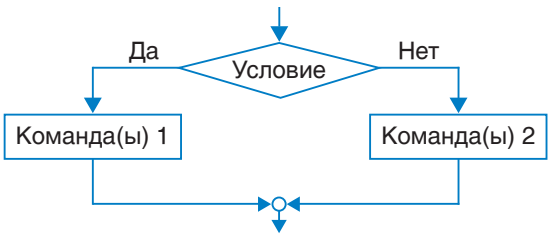
12.1. Команда ветвления

Довольно часто на поставленный вопрос человек получает ответ «Да» или «Нет». В зависимости от ответа он определяет свои действия и выполняет одну или другую команду (группу команд).

Роботы и другие технические устройства тоже могут выполнять различные действия в зависимости от условия. Если условие истинно (на вопрос получен ответ «Да»), то выполняются одни действия, если ложно, то другие.

Алгоритмическая конструкция *ветвление* обеспечивает выполнение одной или другой последовательности команд в зависимости от истинности или ложности некоторого условия.

Ветвление может изображаться на блок-схеме таким образом:



В данной конструкции в прямоугольнике(-ах) записываются команды алгоритма. При

такой организации алгоритма может выполниться **только** одна из двух команд (последовательностей команд). Другая последовательность будет проигнорирована (пример 12.1).

Для записи конструкции ветвления в языке программирования Pascal используется команда **if**. Формат записи команды:

```
if <условие> then
begin
 команды 1;
end
else
begin
 команды 2;
end;
```

Строка **if** <условие> **then** является заголовком ветвления. Эту строку можно прочитать так: «Если условие верно, то». После слова **then** записывается последовательность команд 1, которая выполнится, если условие истинно. После слова **else** записывается последовательность команд 2, которая выполнится, если условие ложно. Слова **begin** и **end;** в данном случае играют роль операторных скобок. Обратите внимание, что перед словом **else** точка с запятой не ставится.

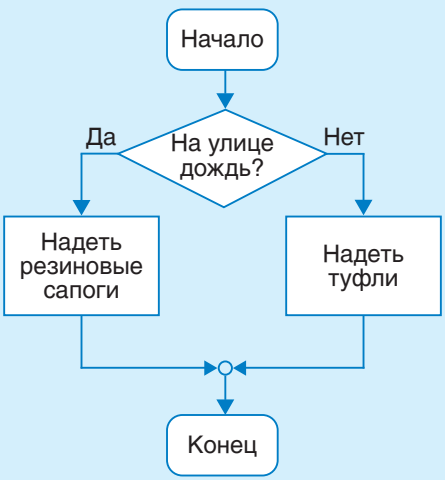
Ветвление может быть записано в **полной** или **сокращенной** форме.

Пример 12.1. Выбор обуви весной в зависимости от погоды:

```
Если на улице дождь, то
  надеть резиновые сапоги;
Иначе
  надеть туфли.
```

В данном примере в текущий момент времени может быть выполнена только одна команда из двух: или надеть сапоги, или надеть туфли.

Блок-схема данного алгоритма будет выглядеть следующим образом:

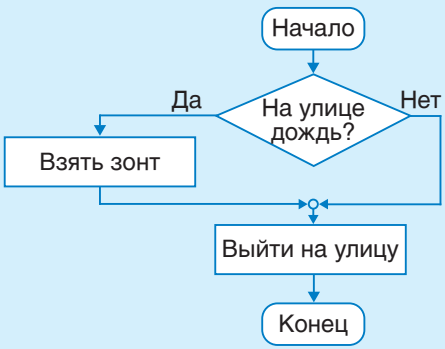


Пример 12.2. Выход на улицу осенью.

```
Если на улице дождь, то
  взять зонт;
выйти на улицу.
```

Здесь используется сокращенная форма команды ветвления. Если условие истинно, выполняется команда «взять зонт». Если условие ложно, никаких действий не происходит. Команда «выйти на улицу» выполняется всегда.

Блок-схема алгоритма:

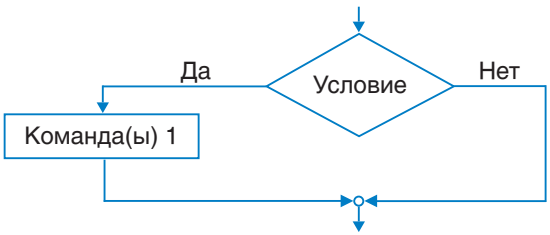


Пример 12.3. Имеется три монеты, среди которых одна фальшивая. Фальшивая монета легче настоящих. Найдем фальшивую монету за минимальное число взвешиваний на чашечных весах без гирь:

```
Положить на каждую чашу весов монеты 1 и 2;
Если весы в равновесии, то
  фальшивая монета 3;
Иначе
  Если монета 1 тяжелее, то
    фальшивая монета 2;
  Иначе
    фальшивая монета 1.
```

Полная форма ветвления предусматривает организацию выполнения двух разных наборов команд, из которых выполняется только один. В сокращенной форме один из наборов команд (чаще по ответу «Нет») опускается. В этом случае, если условие ложное, то никакие действия не выполняются.

Блок-схема сокращенной формы ветвления:



(Рассмотрите пример 12.2.)

На языке программирования Pascal команда запишется так:

```
if <условие> then
begin
 команды 1;
end;
```

Алгоритм может содержать более одной конструкции ветвления (пример 12.3).

Пример 12.4. Решим задачу if1 из встроенного задачника.

Робот должен закрасить клетку, которая находится за стеной. В зависимости от обстановки обход стены может осуществляться по-разному.

Вначале Робот должен сдвинуться вправо. Если стена снизу, то сверху свободно и можно обойти стену сверху, в противном случае Робот обходит стену снизу.

После обхода стены Робот закрашивает клетку. Алгоритм можно записать так:

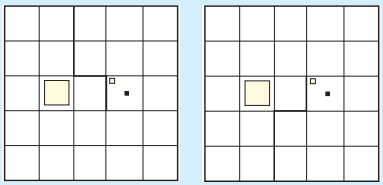
```
вправо;
Если сверху свободно, то
  вверх; вправо; вниз;
Иначе
  вниз; вправо; вверх;
закрасить.
```

Пример 12.5. Робот находится на неизвестной клетке поля без линий. Он должен закрасить клетку слева от себя.

Для того чтобы закрасить клетку слева от себя, Робот должен переместиться влево, а затем закрасить клетку. Однако сделать это Робот сможет только тогда, когда не находится в клетках, являющихся левой границей поля. Поэтому, прежде чем сдвинуться влево, Робот должен проверить, свободно ли слева.

Результат работы данной программы зависит от начального положения Робота. Поэтому для проверки правильности работы программы необходимо подготовить начальные обстановки, которые дают разные ответы на вопрос: слева пусто?

Пример 12.4. Возможные начальные обстановки:



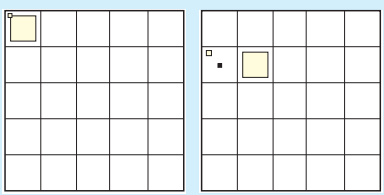
Программа для Робота:

```
uses Robot;
begin
 Task('if1');
 right;
 if FreeFromUp then
 begin
  up; right; down;
 end
 else
 begin
  down; right; up;
 end;
 paint;
end.
```

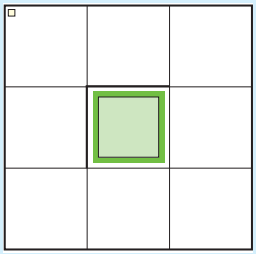
Пример 12.5. Программа для исполнителя Робот:

```
uses Robot, RobTasks;
begin
 Task('myrob9');
 if FreeFromLeft then
 begin
  left; paint;
 end;
end.
```

Возможные начальные обстановки:



Пример 12.6. Рассмотрим следующую начальную обстановку поля для исполнителя Робот:



Проверим для Робота следующие составные условия:

1. `WallFromLeft and CellIsPainted`.
2. `WallFromUp or WallFromDown`.
3. `Not (WallFromRight or FreeFromUp)`.

Первое условие состоит из двух простых: `WallFromLeft` (условие *A*) и `CellIsPainted` (условие *B*). Условие может быть записано как «*A* **И** *B*». Это условие верно только тогда, когда верны и *A*, и *B*. Условие *A* — `WallFromLeft` — истинно, условие *B* — `CellIsPainted` — истинно, условие *A* **И** *B* — истинно.

Второе условие может быть записано как «*A* **ИЛИ** *B*», где *A* — `WallFromUp`, *B* — `WallFromDown`. Условие *A* — истинно, условие *B* — ложно. Значит, условие «*A* **ИЛИ** *B*» — истинно.

12.2. Составные условия

В качестве условия в алгоритмах с циклами и ветвлениями используется любое понятное исполнителю этого алгоритма высказывание, которое может быть либо истинным, либо ложным.

Все условия, с которыми нам приходилось до сих пор встречаться при составлении алгоритмов для Робота, были простыми высказываниями. Однако можно строить и составные условия.

Составное условие — условие, которое образуется из нескольких простых условий, соединенных друг с другом логическими операциями.

С логическими операциями над высказываниями вы уже знакомы. В PascalABC используются следующие логические операции:

Логическая операция	Запись в PascalABC
Не	`Not`
И	`And`
Или	`Or`

(Рассмотрите пример 12.6.)

Система условий для Робота построена так, что можно обойтись без использования логической операции отрицания.

Для условия FreeFromLeft отрицанием будет условие not

FreeFromLeft. Но условие «слева не свободно» означает, что там стена. Поэтому вместо условия not FreeFromLeft можно писать WallFromLeft. Отрицания других условий показаны в таблице:

Условие	Отрицание
WallFromLeft	FreeFromLeft
WallFromRight	FreeFromRight
WallFromUp	FreeFromUp
WallFromDown	FreeFromDown
CellIsPainted	CellIsFree

В третьем условии частица Not отрицает составное условие WallFromRight or FreeFromUp. Условие может быть записано как **НЕ** («*A* **ИЛИ** *B*»). Для того чтобы определить, истинно или ложно это условие, нужно сначала определить истинность условия «*A* **ИЛИ** *B*». Условие *A* — ложно, условие *B* тоже ложно. Поэтому ложным будет и условие «*A* **ИЛИ** *B*», но тогда условие **НЕ** «*A* **ИЛИ** *B*» будет истинным.

1. Что такое алгоритмическая конструкция *ветвление*?
2. Чем отличается полная конструкция ветвления от сокращенной?
3. Что такое составное условие?
4. Какие логические операции можно использовать для записи составных условий?
5. Какими способами можно построить отрицание условия для компьютерного исполнителя Робот?

Упражнения

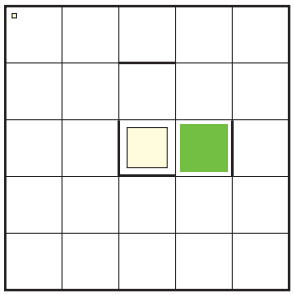
1 Выделите конструкцию ветвления в отрывке из поэмы А. С. Пушкина «Руслан и Людмила» и изобразите ее с помощью блок-схемы.

У лукоморья дуб зеленый;
Златая цепь на дубе том:
И днем и ночью кот ученый
Все ходит по цепи кругом;
Идет направо — песнь заводит,
Налево — сказку говорит.
Там чудеса: там леший бродит,
Русалка на ветвях сидит...[1]

[1] Пушкин, А. С. Руслан и Людмила : поэма. — М. : Изд. Дом «Прибой». — 1996. — С. 5.

2 Для заданной обстановки поля Робота определите, какие из составных условий истинны, а какие ложны.

Начальная обстановка	Условия

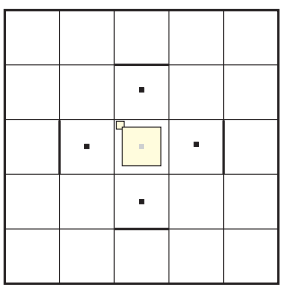
	`WallFromLeft or CellIsPainted;`
	`WallFromUp and WallFromDown;`
	`Not CellIsPainted and FreeFromRight;`
	`Not (WallFromUp or FreeFromRight);`
	`WallFromDown and CellIsFree;`
	`(WallFromUp or WallFromDown) and FreeFromRight.`

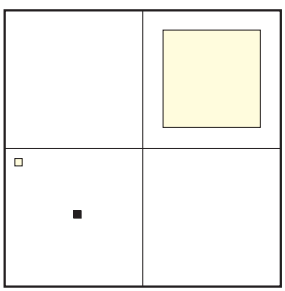
3* В задании 2 замените условия, содержащие not, соответствующими условиями без использования отрицания.

4 Для каждого из ложных условий задания 2 придумайте обстановку, в которой данное условие будет верным, а для каждого истинного — обстановку, в которой условие будет ложным.

5 Измените программу из примера 12.5 так, чтобы Робот закрашивал клетку справа (снизу, сверху) от себя. Нарисуйте в тетради различные начальные обстановки для проверки данного условия.

6 Решите задачи if2 и if3 из встроенного задачника.





§ 13. Использование основных алгоритмических конструкций для исполнителя Робот

Последовательное выполнение команд в программе определяется структурой ***следование***. Для организации повторяющихся действий в алгоритме используется команда **цикла**. Команда **ветвления** позволяет выполнять одну или другую последовательность команд в зависимости от истинности условия.

Следование, цикл и ветвление — **базовые алгоритмические конструкции**. Используя эти конструкции как элементы некоего «конструктора», можно составлять и разрабатывать любые алгоритмы.

Команды цикла и ветвления управляют порядком выполнения других команд в программе и относятся к командам управления. Использование алгоритмической конструкции *следование* предполагает отсутствие управляющих конструкций.

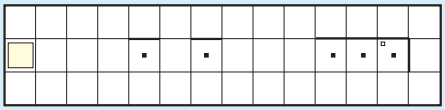
Рассмотрим подробнее примеры алгоритмов, содержащих несколько алгоритмических конструкций.

Перед человеком постоянно возникают разнообразные задачи, для которых существуют различные алгоритмы решения. При всем многообразии алгоритмов для их записи достаточно трех алгоритмических конструкций (структур): следование, цикл, ветвление.

Это положение было выдвинуто в середине 70-х гг. XX в. нидерландским ученым Эдсгером Вибе Дейкстрой (1930—2002).

Его труды оказали влияние на развитие информатики и информационных технологий. Э. Дейкстра является одним из разработчиков концепции структурного программирования, участвовал в создании языка программирования Алгол. Известен своими достижениями в области математической логики и теории графов.

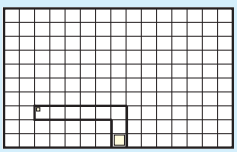
Пример 13.1. Одна из возможных начальных обстановок:

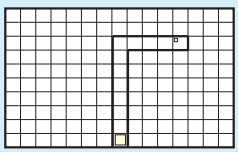


Программа для Робота:

```
uses Robot;
begin
 Task('cif1');
 while FreeFromRight do
 begin
  if WallFromUp then
   paint;
  right;
 end;
 if WallFromUp then
   paint;
end.
```

Пример 13.2. Возможные начальные обстановки:





Программа для Робота:

```
uses Robot;
begin
 Task('cif17');
 while FreeFromUp do
  up;
 if FreeFromLeft then
  while FreeFromLeft do
   left
 else
  while FreeFromRight do
   right;
end.
```

Пример 13.1. Решим задачу cif1 из встроенного задачника.

Робот передвигается вправо до тех пор, пока не встретит стену. По пути он должен закрасить клетки, над которыми есть стена.

Для решения задачи Робот должен проверять каждую клетку на своем пути. Если условие «сверху стена» выполняется, Робот закрашивает эту клетку. После проверки клетки Робот сдвигается вправо. Такие действия выполняются в цикле, пока справа пусто.

После цикла нужна команда ветвления, так как для крайней клетки поля команда «справа пусто» не выполняется и клетка в цикле не закрашивается. В этой задаче внутри структуры цикла используется структура ветвления.

Пример 13.2. Решим задачу cif17 из встроенного задачника.

Робот должен дойти до конца «коридора» переменного размера. «Коридор» может сворачивать влево или вправо.

Для решения задачи Робот сначала перемещается вверх до тех пор, пока вверху пусто. Стена, появившаяся сверху, означает, что начался поворот «коридора». «Коридор» поворачивает влево, если слева пусто, иначе «коридор» поворачивает вправо. Дальше Ро-

бот двигается в том направлении, где пусто, пока не встретит стену.

В данной задаче используется сначала структура цикла, а затем структура ветвления. Каждая последовательность команд в структуре ветвления, в свою очередь, является циклом.

Операторные скобки опущены, поскольку последовательность состоит из одной команды цикла.

Пример 13.3*. Решим задачу cc5 из встроенного задачника.

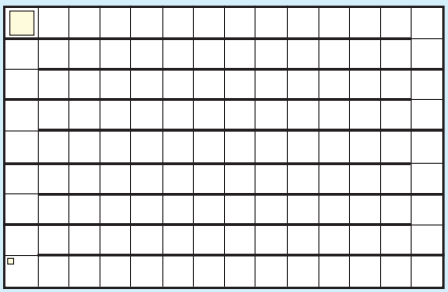
Робот находится в верхнем левом углу поля и должен переместиться в нижний левый угол. На поле присутствуют стены, которые Робот должен обойти. При этом он должен сначала двигаться до правой границы поля, затем спуститься вниз, а потом двигаться до левой границы поля и спуститься вниз. Эти действия Робот должен повторить 4 раза.

В данной задаче внутри цикла с параметром используются два других цикла с предусловием.

Структуру, когда внутри одного цикла выполняется другой, называют **вложенными** циклами.

Таким образом, базовые алгоритмические структуры можно комбинировать друг с другом.

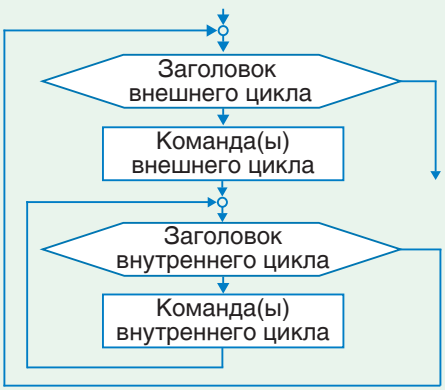
Пример 13.3*. Начальная обстановка:



Программа для Робота:

```
uses Robot;
begin
 Task('cc5');
 for var i:= 1 to 4 do
 begin
  while FreeFromRight do
   right;
  down;
  while FreeFromLeft do
   left;
  down;
 end;
end.
```

Блок-схема вложенных циклов:



1. Назовите базовые алгоритмические конструкции.
2. Приведите примеры использования базовых алгоритмических конструкций.
3*. Что такое вложенный цикл?

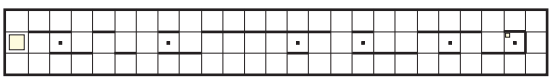
Упражнения

1 Какие алгоритмические конструкции используются в приведенных программах? Нарисуйте блок-схемы данных алгоритмов. Предложите пример начальной обстановки, в которой алгоритм выполнится корректно.

а)
```
uses Robot;
begin
 while WallFromLeft do
 begin
  down;
  paint;
 end;
end.
```

б)
```
uses Robot;
begin
 while CellIsPainted do
  if FreeFromleft then
   left;
end.
```

2 Для решения задачи cif3 из встроенного задачника Миша написал программу, но она работает неправильно. Какие ошибки допустил Миша?



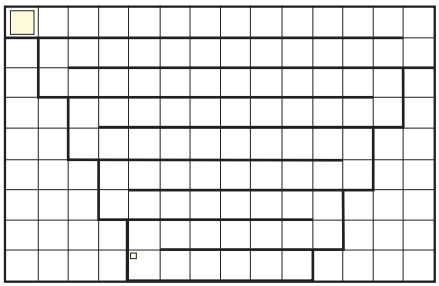
```
uses Robot;
begin
 Task('cif3');
 while WallFromRight do
 begin
  if WallFromDown or
  WallFromUp then
   paint;
  right;
 end;
 if WallFromUp and
 WallFromDown then
  paint;
end.
```

3 Используя базовые алгоритмические конструкции, запишите алгоритмы, соответствующие описаниям. Постройте для них блок-схемы.

1. Тело цикла, выполняющегося при условии `WallFromUp`, состоит из двух команд: `right` и `paint`.
2. Если условие `FreeFromRight` не выполняется, то, если клетка не закрашена, ее нужно закрасить, а если закрашена, то сдвинуться влево.

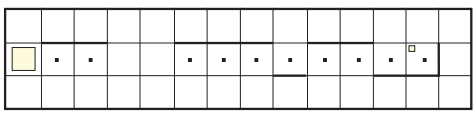
3. Проверку условия `CellIsPainted` нужно производить до тех пор, пока снизу нет стен. При выполнении условия сдвинуться вниз, при невыполнении условия закрасить клетку.

4 Заполните пропуски в программе решения задачи cc14 из встроенного задачника так, чтобы она работала верно.

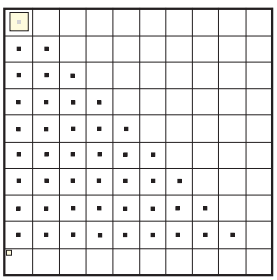


```
uses Robot;
begin
 Task('cc14');
 for var i: =1 to 4 do
 begin
  while … do
   right;
  down;
  while … do
   left;
  down;
 end;
end.
```

5 Решите задачу cif2 из встроенного задачника, используя внутри цикла команду ветвления.



6 Решите задачу cc7 из встроенного задачника, использовав внутри одного цикла два вложенных цикла.



7* Придумайте задачу для исполнителя Робот, в которой будут использоваться различные алгоритмические конструкции.

§ 14. Язык программирования Паскаль

Компьютер (от англ. *computer* — вычислитель) — устройство или система, способные выполнять заданную четко определенную изменяемую последовательность операций (чаще всего численных расчетов).

Электронно-вычислительная машина (ЭВМ) — комплекс технических средств, где основные функциональные элементы (логические, запоминающие, индикационные и др.) выполнены на электронных приборах, предназначенных для автоматической обработки информации в процессе решения вычислительных задач.

Никлаус Вирт (родился в 1934 г.) — швейцарский ученый, специалист по информатике, один из известнейших теоретиков в области разработки языков программирования, профессор компьютерных наук. Создатель и ведущий проектировщик языков программирования Паскаль, Модула-2, Оберон.

Желание упростить и ускорить всевозможные расчеты присуще человеку с древних времен. Сегодня компьютер способен выполнять сотни миллионов операций в секунду. Для решения вычислительных задач требуется сначала составить алгоритм их решения, а затем записать его в виде программы, используя какой-либо язык программирования.

Язык программирования устанавливает набор правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под ее управлением.

Язык программирования Паскаль (Pascal) используется для обучения программированию и является базой для ряда профессиональных языков программирования.

Существует множество сред программирования, поддерживающих язык Паскаль: PascalABC, FreePascal, Delphi, GNU Pascal, Dev-Pascal, Rad Studio и др. В учебном курсе используется среда PascalABC (с ней вы работали, знакомясь с учебными компьютерными исполнителями).

14.1. Команда вывода

Демонстрировать работу любой программы имеет смысл только тогда, когда она выводит какую-либо информацию.

Программа на языке Pascal (тело программы) должна начинаться со слова **begin**, а заканчиваться словом **end** и точкой. Программа, состоящая из этих команд, разделенных пробелом или переводом строки, может быть запущена на выполнение, но она ничего не делает. Добавим в нее команду вывода приветствия:

```
begin
 write('Привет!');
end.
```

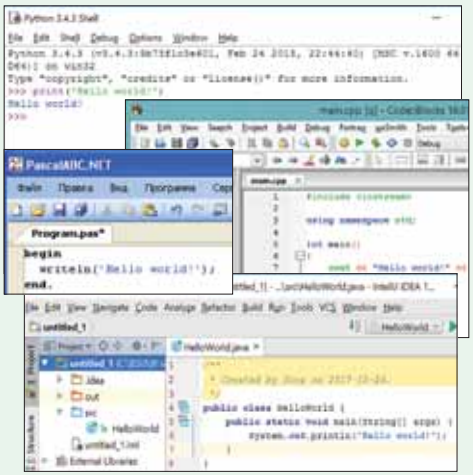
Результат работы программы отражается в нижней части окна программы PascalABC в окне вывода (пример 14.1).

Команда `write( );` предназначена для **вывода данных.**

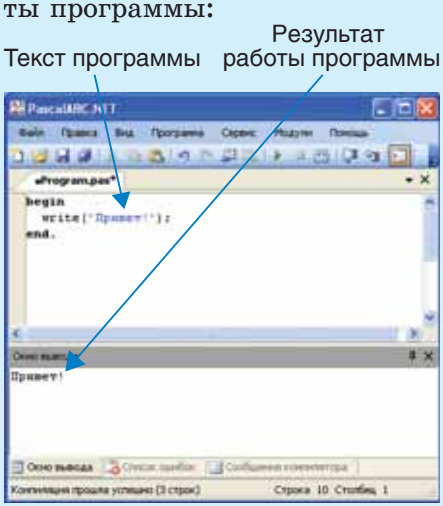
Текст, который нужно вывести на экран, заключают в апострофы (одинарные кавычки). Этот текст не анализируется и выводится в том виде, в котором он записан. Текст можно записать на любом языке. Текстом может быть произвольный набор символов.

В одной программе может быть несколько команд вывода. Для

По традиции, начавшейся в 1978 г. с примера из книги Б. Кернигана и Д. Ритчи «Язык программирования Си», первая программа на любом языке программирования должна выводить на экран приветствие миру:



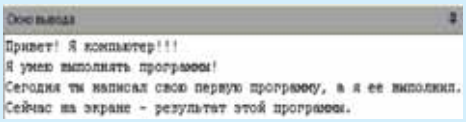
Пример 14.1. Окно среды PascalABC с результатом работы программы:



Пример 14.2. Текст программы:

```
begin
  write('Привет!  ');
  writeln('Я  компьютер!!!');
  write('Я умею выполнять ');
  writeln('программы!');
  write('Сегодня  ты  ');
  write('написал  свою  ');
  write('первую программу,');
  writeln(' а я ее выполнил.');
  write('Сейчас на экране -');
  writeln('  результат  этой
 программы.');
end.
```

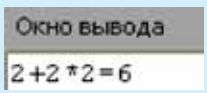
Результат работы программы:



Пример 14.3. Текст программы:

```
begin
  write('2+2*2=');
  write(2+2*2);
end.
```

Результат работы программы:



Две команды `write` в программе можно объединить в одну, отделив текст от выражения запятой:

```
begin
  write('2+2*2=',  2+2*2);
end.
```

вывода текста, записанного в несколько строк, используют команду `writeln( )`. Сочетание «`ln`» (сокр. от англ. *line* — линия, строка), записанное в конце команды, означает, что после вывода нужно перевести курсор в новую строку.

Пример 14.2. Выведем на экран компьютера следующий текст: «`Привет! Я компьютер!!! Я умею выполнять программы! Сегодня ты написал свою первую программу, а я ее выполнил. Сейчас на экране – результат этой программы`».

Используя сочетание команд `write` и `writeln`, текст можно расположить по-разному.

Как вы уже знаете, текст в команде `write( )`, записанный в кавычках, не анализируется. Если кавычки опустить, то производится анализ данных, записанных в скобках. Так, если в скобках написать арифметическое выражение, то сначала вычисляется его значение, а затем выводится результат.

Пример 14.3. Посчитаем значение выражения 2 + 2 * 2.

Если записать выражение в кавычках, то будет выведено само выражение. При отсутствии кавычек на экран будет выведено значение данного выражения.

14.2. Понятие типа данных

На практике редко приходится писать программы, которые решают только одну задачу. Обычно программы пишутся для решения целого класса задач, которые можно сформулировать в общем виде.

С такими задачами вы уже сталкивались в курсе математики. Например, решение задачи «Найдите площадь прямоугольника» можно записать так: $S = a \cdot b$, где переменные a и b обозначают соответственно длину и ширину прямоугольника, а S — площадь. Зная эту формулу, можно найти площадь любого прямоугольника.

В программировании для решения задач в общем виде также используют переменные. Поскольку с такими переменными будет работать компьютер, то они должны храниться в его памяти.

Информацию, представленную в пригодном для обработки на компьютере виде, называют **данными.**

Переменная в программировании — это именованная ячейка памяти, хранящая значение переменной.

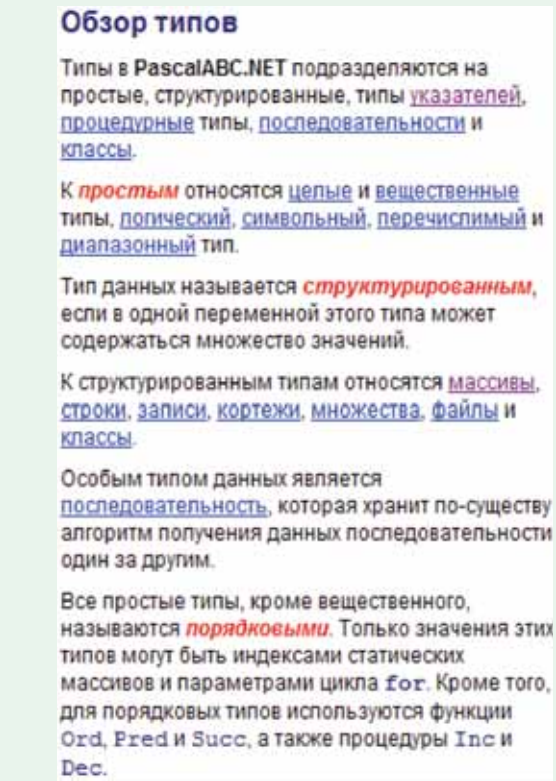
До начала 1950-х гг. XX в. программисты ЭВМ при создании программ пользовались машинным кодом. Запись программы на машинном коде состояла из единиц и нулей. Машинный код принято считать языком программирования первого поколения. Типы данных не использовались.

Первым языком программирования, в котором появилась возможность создавать переменные, считается Ассемблер. В этом языке вместо машинных кодов стали использовать команды, записанные текстом. Ассемблер относится к языкам программирования второго поколения.

В 1957 г. появился язык Фортан, открывший эру языков программирования третьего поколения. Он позволил использовать разные числовые типы данных, необходимые для сложных расчетов: целые, вещественные (действительные) и комплексные.

Дальнейшее развитие языков программирования позволило добавить возможность работы с другими типами данных. Современные языки программирования позволяют работать с большим количеством типов данных.

В среде программирования PascalABC реализовано более 30 различных типов данных.



Обзор типов

Типы в **PascalABC.NET** подразделяются на простые, структурированные, типы указателей, процедурные типы, последовательности и классы.

К *простым* относятся целые и вещественные типы, логический, символьный, перечислимый и диапазонный тип.

Тип данных называется *структурированным*, если в одной переменной этого типа может содержаться множество значений.

К структурированным типам относятся массивы, строки, записи, кортежи, множества, файлы и классы.

Особым типом данных является последовательность, которая хранит по-существу алгоритм получения данных последовательности один за другим.

Все простые типы, кроме вещественного, называются *порядковыми*. Только значения этих типов могут быть индексами статических массивов и параметрами цикла `for`. Кроме того, для порядковых типов используются функции `Ord`, `Pred` и `Succ`, а также процедуры `Inc` и `Dec`.

Пример 14.4. Примеры описания переменных:

```
var x: real;
var x1, y1: real;
var a_1, a_2, a_3: real;
```

Диапазон возможных значений типа `real` задается числами в стандартном представлении от $-1.8 \cdot 10^{308}$ до $1.8 \cdot 10^{308}$. Наименьшее положительное число типа `real` приблизительно равно $5.0 \cdot 10^{-324}$. При вычислениях в числе хранится до 16 цифр.

Компьютер может обрабатывать данные разных типов: целые и действительные числа, символы, тексты и др.

Тип данных определяет способ хранения данных в памяти компьютера, диапазон возможных значений данных и операции, которые с этим типом данных можно выполнять.

Чтобы использовать какую-либо переменную, ее нужно описать. Описание переменных выполняется до начала программы (команды **begin**) (пример 14.4). При описании переменной выделяется память для хранения ее значения. В процессе выполнения программы значение переменной может изменяться.

Для описания переменных используется команда **var** (сокр. от англ. *variable* — переменная).

Формат записи команды:

var <имя переменной>: <тип>;

Для обозначения имени переменной используют буквы латинского алфавита, цифры и знак «_». Первым символом должна быть буква или знак подчеркивания.

Тип данных `real` в языке Pascal позволяет работать с числами и выполнять над ними арифметические действия.

14.3. Оператор присваивания

Одной из основных команд для обработки данных в программе является оператор присваивания.

Оператор присваивания предназначен для того, чтобы:

- задавать значения переменным;
- вычислять значения арифметического выражения (результат вычисления будет записан как значение переменной).

Формат записи оператора:

<имя переменной>:= <выражение>;

(Рассмотрите пример 14.5.)

В записи арифметического выражения используются знаки математических действий.

Математические операции	Запись в Pascal
+ (сложение)	+
− (вычитание)	-
· (умножение)	*
: (деление)	/

Приоритет выполнения операций соответствует принятому в математике: сначала выполняются умножение и деление, а затем сложение и вычитание. Для изменения порядка действий в выражениях используют скобки.

Пример 14.5. Примеры записи оператора присваивания:

```
x:= 7;
x1:= 3.5;
a_1: = 20 * (x + x1) - 32;
y:= y + 7;
```

Пример 14.6. Запишем оператор присваивания на Pascal для математических выражений:

Выражение	Запись на Pascal
$S = 2(a + b)$	`S:=2*(a+b);`
$S = a^2$	`S:=a*a;`
$a = \frac{x + y}{3}$	`a:=(x+y)/3;`

Пример 14.7. Запишем оператор присваивания, после выполнения которого значение переменной a увеличится в 2 раза, а переменной b уменьшится на 3.

В Pascal допустимы команды присваивания следующего вида:

```
a:= a * 2;
```

Смысл такой команды следующий: из ячейки памяти извлекается значение переменной a, затем оно умножается на 2, результат записывается в ту же ячейку памяти. Старое значение переменной a будет потеряно.

Запись оператора присваивания для изменения значения переменной b следующая:

```
b:= b - 3;
```

В PascalABC.NET определены операторы присваивания со значками +=, −=, *=, /=. Они позволяют изменить значение переменной. Например:

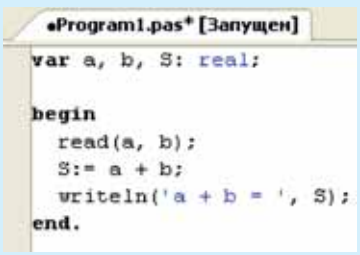
```
a *= 2;  // увеличить a в 2 раза;
b -= 3;  // уменьшить b на 3.
```

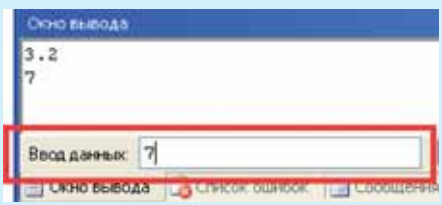
Пример 14.8. Ввести два числа, найти и вывести их сумму.

Текст программы:

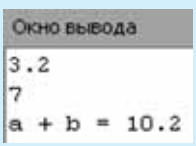
```
var a, b, S: real;
begin
 read(a, b);
 S:= a + b;
 writeln('a + b =', S);
end.
```

Ввод данных:





Результат:



Для записи обыкновенной дроби используется знак деления. Знак умножения опускать нельзя. Целая часть дробного числа отделяется от дробной части точкой. (Рассмотрите примеры 14.6 и 14.7 на с. 93.)

14.4. Ввод данных

Начальные значения переменным можно задавать не только с помощью оператора присваивания, но и путем ввода с клавиатуры. В этом случае, если необходимы вычисления с новым набором значений исходных данных, текст программы не нужно изменять.

Команда `read( )` предназначена для **ввода данных**. В скобках через запятую перечисляются имена переменных, значения которых необходимо ввести.

Ввод данных происходит в нижней части окна программы PascalABC. Для этого используется окно «Ввод данных». После нажатия кнопки «Ввести» или клавиши «Enter» введенные значения переносятся в окно вывода. После завершения работы программы в этом же окне будет выведен результат (пример 14.8).

14.5. Структура программы

Все программы на языке программирования Pascal имеют общую структуру. В программе можно выделить следующие разделы:

- заголовок (не обязателен);
- подключаемые библиотеки (модули) (если подключать дополнительные библиотеки не нужно, раздел отсутствует; известные библиотеки: `Drawman`, `Robot`, `RobTasks`);
- описание переменных с указанием их типа;
- описание вспомогательных алгоритмов (если использовать вспомогательные алгоритмы не нужно, раздел отсутствует);
- **begin** ... **end.** — служебные слова, обрамляющие тело основной программы, в которой находятся исполняемые команды; **begin** начинает исполняемую часть программы, а **end.** (точка в конце обязательна) ее завершает.

В минимально возможном наборе программа состоит из пустого тела программы: **begin end.** Программа, содержащая все разделы, представлена в примере 14.9.

Для каждого раздела определено ключевое служебное слово, которым он начинается. При написании программы ключевые слова выделяются полужирным шрифтом.

Пример 14.9. Программа, содержащая все разделы (подсчитывается количество закрашенных клеток в поле Робота размером 10 × 10):

```
//заголовок программы
//(необязательно)
program Primer;
//описание библиотек
uses Robot, RobTasks;
//описание переменных
var k: integer;
//описание вспомогательных
//алгоритмов
procedure line;
begin
 for var i:= 1 to 9 do
 begin
  if CellisPainted then
   k:= k + 1;
  right;
 end;
 if CellisPainted then
  k:= k + 1;
end;
procedure back;
begin
 for var i:= 1 to 9 do
  left;
end;
//основная программа
begin
//тело программы
 Task('myrob11');
 k:= 0;
 for var i:= 1 to 9 do
 begin
  line; back; down;
 end;
 line;
 writeln(k);
end.
```

1. Какая команда языка программирования Pascal предназначена для вывода данных?
2. Что определяет тип данных?
3. Для чего используется команда присваивания?
4. Какая команда языка программирования Pascal предназначена для ввода данных?
5. Из каких разделов состоит программа на языке программирования Pascal?

Упражнения

1 Для программы из примера 14.2 выполните следующие задания (файл с программой можно скачать):

1. Замените все команды `writeln` на команды `write` и выполните программу. Что произошло? Объясните почему.
2. Как изменится результат работы программы, если в исходном тексте заменить все команды `write` на `writeln`?
3. Измените программу так, чтобы текст на экране выглядел следующим образом:

```
Привет! Я компьютер!!! Я умею выполнять программы!
Ты сегодня написал свою первую программу!!!
Я выполнил твою программу. Посмотри на экране результат!
```

2 Внесите необходимые изменения в программу из примера 14.3, чтобы действия выполнялись в том порядке, в котором записаны, т. е. сначала сложение, а потом умножение.

3 Вводится возраст пользователя в годах. Определите возраст пользователя через 5 лет.

4 Напишите программу, в которой вводятся два числа a и b. Затем первое число уменьшается в 2 раза, а второе увеличивается на 30. Выведите измененные значения переменных.

5 Напишите программу для вычисления значения числовых выражений:

1. $23 + 45 \cdot 11 - 15$.
2. $\dfrac{37 + 2 \cdot 27}{41}$.
3. $\dfrac{5638 - 2347}{49} + \dfrac{123 \cdot 756}{4455}$.

§ 15. Организация вычислений

При решении любой задачи человеку приходится выполнять следующие действия:

- определение исходных данных (что дано в задаче);
- определение результатов (что нужно получить);
- обработка исходных данных в соответствии с известными правилами так, чтобы получить результат.

Применяя указанные правила к решению задачи по программированию, получим следующие этапы решения задачи:

I. Определение исходных данных.

II. Определение результатов.

II. Составление алгоритма решения задачи.

IV. Определение типов данных для переменных, используемых при реализации алгоритма.

V. Написание программы.

VI. Тестирование программы.

VII. Анализ результатов.

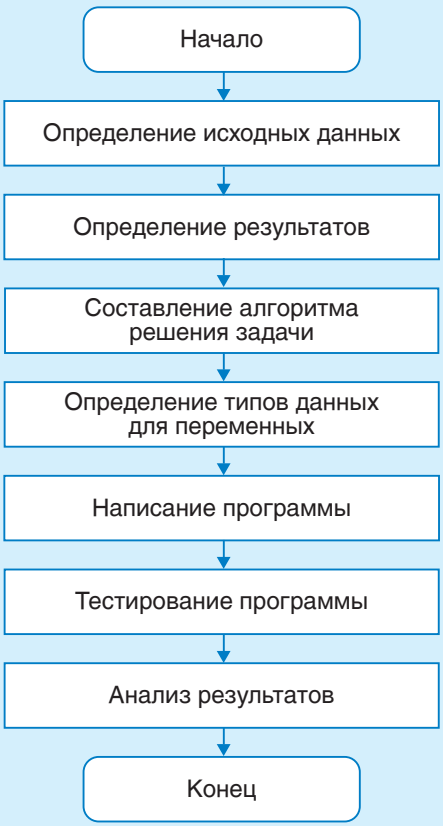
(Рассмотрите пример 15.1.)

Тестирование программы — проверка правильности работы программы при разных наборах исходных данных.

Пример 15.1. Решение задач по физике принято оформлять определенным образом.

Слева записывается то, что дано и что нужно получить, справа — последовательность действий, приводящая к решению задачи. Аналогично оформляются решения задач по химии, геометрии.

Этапы решения задачи по программированию можно представить следующим образом:



Пример 15.2.

V. Программа:

```
var  x, y, z, a:  real;
begin
 write('введите  x  =  ');
 read(x);
 write('введите  y  =  ');
 read(y);
 write('введите  z  =  ');
 read(z);
 a:=(2*x+3*y-z)/(3+x*x);
 writeln('a  =  ',a);
end.
```

VI. Тестирование программы.

Запустите программу и введите значения: $x = 2$, $y = 3$, $z = 1$.

Результат работы программы должен быть следующим:

```
Окно вывода
введите x = 2
введите y = 3
введите z = 1
a = 1.71428571428571
```

А для значений $x = 2$, $y = 4$, $z = 2$ получим:

```
Окно вывода
введите x = 2
введите y = 4
введите z = 2
a = 2
```

VII. Проверка правильности вычислений может быть выполнена на калькуляторе.

15.1. Вычисление значения арифметического выражения

Пример 15.2. Даны переменные x, y, z. Напишем программу для вычисления значения выражения $a = \dfrac{2x + 3y - z}{3 + x^2}$.

Этапы выполнения задания:

I. Определение исходных данных: переменные x, y, z.

II. Определение результатов: переменная a.

III. Алгоритм решения задачи:

1. Ввод исходных данных.
2. Вычисление значения выражения.
3. Вывод результата.

IV. Описание переменных.

Все переменные, определенные для решения задачи, имеют тип `real`.

В приведенном примере перед каждой командой ввода записана команда вывода с пояснениями о том, значение какой переменной нужно вводить.

При написании программ для вычисления значения арифметического выражения часто допускают следующие ошибки:

`2x+3;` `(x-4)(x+2);`	Пропущен знак *
`(2+y)/(x*x;`	Пропущена скобка

Будьте внимательны!

15.2. Использование языка программирования для решения задач

Пример 15.3. Напишем программу для решения геометрической задачи. Задан квадрат с длиной стороны a. Требуется найти его площадь и периметр.

Этапы выполнения задания:

I. Определение исходных данных: переменная a (длина стороны).

II. Определение результатов: переменные S (площадь) и P (периметр).

III. Алгоритм решения задачи:

1. Ввод исходных данных.

2. Вычисление значений площади в математике производится по формуле $S = a^2$, а периметра — по формуле $P = 4a$. В программе этим формулам будут соответствовать команды присваивания: S:= a * a; P:= 4 * a.

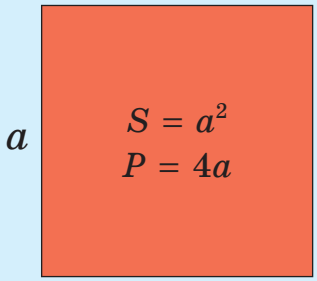
3. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют тип `real`.

Обратите внимание: запись формул в операторе присваивания может отличаться от записи математических формул.

Пример 15.3.



V. Программа:

```
var a,S,P: real;
begin
 write('введите a = ');
 read(a);
 s:= a*a;
 p:= 4*a;
 writeln('площадь = ',S);
 writeln('периметр = ',P);
end.
```

VI. Тестирование программы.

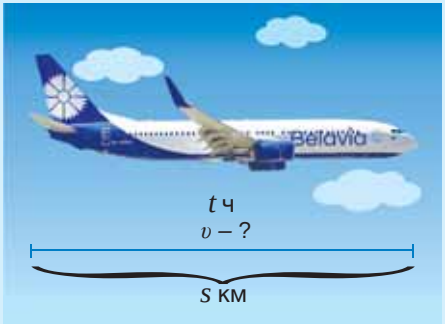
Запустите программу и введите значение a = 5.2.

Результат работы программы должен быть следующим:

```
Окно вывода
введите a = 5.2
площадь= 27.04
периметр= 20.8
```

VII. Проверка правильности вычислений может быть выполнена на калькуляторе.

Пример 15.4.



V. Программа:

```
var s, t, v: real;
begin
 write('введите s = ');
 read(s);
 write('введите t = ');
 read(t);
 v:= s / t;
 writeln('скорость = ', v);
end.
```

VI. Тестирование программы.

Запустите программу и введите значения $s = 3550$ и $t = 4$.

Результат работы программы должен быть следующим:

Окно вывода

```
введите s = 3550
введите t = 4
скорость = 887.5
```

VII. Проверка правильности вычислений может быть выполнена на калькуляторе.

Пример 15.4. Напишем программу для решения физической задачи. Расстояние между двумя городами составляет s км. Самолет пролетает это расстояние за t ч. Определите скорость самолета.

Этапы выполнения задания:

I. Определение исходных данных: переменные s (расстояние) и t (время).

II. Определение результатов: переменная v (скорость).

III. Алгоритм решения задачи:

1. Ввод исходных данных.

2. Согласно формуле расстояния: $s = vt$. Отсюда выразим v: $v = \frac{s}{t}$.

3. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют тип `real`.

При написании программ обращайте внимание на форматирование их текста:

- в первой позиции на экране пишут только слова **var**, **begin** **end**, а остальные со сдвигом на 2—4 позиции вправо;
- если в программе несколько частей, то их можно отделить друг от друга пустой строкой.

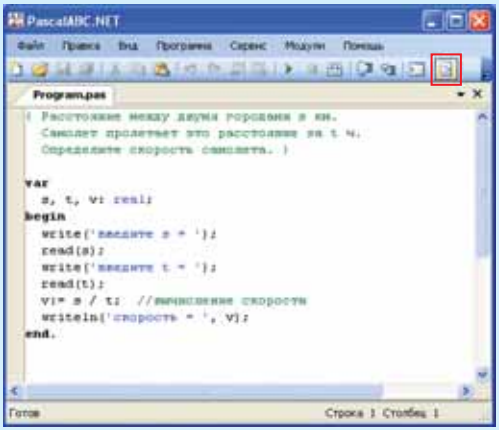
Выполнение этих правил повышает читаемость программы.

В программе можно использовать **комментарии** — текст, который не анализируется при запуске программы на выполнение.

Текст после символов // считается комментарием и выделяется зеленым цветом (пример 15.5).

В комментариях удобно записывать условие задачи. Пояснения к командам, записанные как комментарии, нужны для понимания действия, выполняемого командой. В языке программирования Pascal комментарии можно записать в несколько строк. Тогда текст, являющийся комментарием, заключают в фигурные скобки.

Пример 15.5. Программа с комментариями и отформатированным кодом:



Значок для форматирования кода на панели инструментов имеет следующий вид:

1. Перечислите этапы решения задачи по программированию.
2. Что понимают под тестированием программы?
3. Для чего можно использовать комментарии?

Упражнения

1 Даны x, y, z. Напишите программу для вычисления значения арифметических выражений.

1. $a = \dfrac{x + y - z}{x^2 + 2}$. 2. $a = 5\dfrac{2x - z}{3 + y^2}$. 3. $a = (1 + z)\dfrac{x + \dfrac{y}{x^2 + 4}}{2 + \dfrac{1}{x^2 + 4}}$.

2 Напишите программу для решения геометрической задачи.

1. Найдите длину окружности и площадь круга заданного радиуса.

2*. Найдите угол при основании равнобедренного треугольника, если известен угол при вершине.

3 Напишите программу для решения физической задачи.

1. Велосипедист едет с постоянной скоростью v км/ч. За сколько минут он проедет расстояние в s км?

2*. Автомобиль проходит первую часть пути длиной s_1 км за t_1 мин, участок пути длиной s_2 км за t_2 мин и участок длиной s_3 км за t_3 мин. Найдите среднюю скорость автомобиля в км/ч.

4 Напишите программу для решения химической задачи.

1. В организме человека на долю атомов кислорода приходится 65 % от массы тела. Найдите массу атомов кислорода для своей массы тела.

2*. Масса атома кислорода равна $26.56 \cdot 10^{-27}$ (это число на языке Pascal записывается так: 26.56E-27, буква E — английская). Сколько атомов кислорода содержится в вашем теле?

§ 16. Реализация алгоритмов работы с целочисленными данными

В PascalABC определены различные типы данных для работы с целыми числами, позволяющие выполнять действия над данными из разных числовых диапазонов. Чем больше диапазон, тем больше места в памяти компьютера отводится для хранения переменных.

Некоторые целочисленные типы данных:

Тип	Диапазон значений
`shortint`	–128..127
`smallint`	–32768..32767
`integer`, `longint`	–2147483648..2147483647
`byte`	0..255
`word`	0..65535

16.1. Целочисленный тип данных

Часто при решении задач нужно работать с целыми числами. Для этого в Pascal используется тип данных `integer`. С помощью переменных этого типа можно задавать целые числа из диапазона от –2147483648 до 2147483647. Для типа данных `integer` определены следующие операции:

Математические операции	Запись в Pascal
+ (сложение)	+
– (вычитание)	–
· (умножение)	*
целочисленное деление	`div`
нахождение остатка	`mod`

Для целочисленных данных не определена операция деления, как для действительных чисел. При попытке использовать операцию деления будет выдана ошибка (пример 16.1).

Для организации вычислений с целыми числами определены операции `div` и `mod`. Эти операции имеют такой же приоритет, как и операции деления и умножения.

Пример 16.2. Даны два целых числа a и b. Напишем программу, которая находит целую часть от деления a на b и остаток.

Этапы выполнения задания:

I. Определение исходных данных: переменные a и b.

II. Определение результатов: переменные c (целочисленное частное) и d (остаток).

III. Алгоритм решения задачи:

1. Ввод исходных данных.
2. Целочисленное частное находим как результат операции `a div b`, остаток — `a mod b`.
3. Вывод результата.

IV. Описание переменных.

Все переменные, определенные для решения задачи, имеют тип `integer`.

Значение, выдаваемое как результат операции `mod`, может

Пример 16.1. Ошибка использования операции деления для целочисленных типов данных:

Program.pas

```
var a, b, c: integer;
begin
  read(a, b);
  c:= a / b;
  write(c);
end.
```

Список ошибок

	Строка	Описание
1	4	Нельзя преобразовать тип real к integer

Пример 16.2.

V. Программа:

```
var a, b, c, d: integer;
begin
 read(a, b);
 c:= a div b;
 d:= a mod b;
 writeln('целая часть = ',
 c);
 writeln('остаток = ', d);
end.
```

VI. Тестирование программы.

Запустите программу и введите значения $a = 11$ и $b = 4$.

Результат работы программы должен быть таким, как показано ниже:

Окно вывода

```
11
4
целая часть = 2
остаток = 3
```

Результат операций `div` и `mod` для разных чисел:

a	b	a div b	a mod b
17	3	5	2
−17	3	−5	−2
17	−3	−5	2
−17	−3	5	−2

Так, a **mod** b = a − (a **div** b) * b.

Пример 16.3.

V. Программа:

```
var c, m, s: integer;
begin
 write('введите c = ');
 readln(c);
 {Минуты}
 m:= c div 60;
 {Секунды}
 s:= c mod 60;
 write(m, ':', s);
end.
```

VI. Тестирование программы.

Запустите программу и введите значение $c = 137$.

Результат работы программы:

```
Окно вывода
введите c = 137
2:17
```

Для значения $c = 24$ получим:

```
Окно вывода
введите c = 24
0:24
```

отличаться от математического определения остатка (в математике под остатком понимают неотрицательное число). Если остаток не равен нулю, то знак числа, являющегося результатом операции `mod`, определяет знак делимого.

16.2. Использование целочисленных данных для решения задач

Пример 16.3. Пусть таймер показывает время только в секундах. Напишем программу, переводящую время в минуты и секунды.

Этапы выполнения задания:

I. Определение исходных данных: переменная c (время в секундах).

II. Определение результатов: переменные m (полное количество минут) и s (остаток секунд).

III. Алгоритм решения задачи:

1. Ввод исходных данных.
2. Для нахождения полного числа минут нужно найти целую часть от деления исходного числа секунд на 60.
3. Оставшиеся секунды находим как остаток от деления исходного числа секунд на 60.
4. Вывод результата.

IV. Описание переменных.

Переменные, определенные для решения задачи, имеют тип `integer`.

Пример 16.4. Задано двузначное число. Нужно поменять местами первую и вторую цифры числа.

Этапы выполнения задания:

I. Определение исходных данных: переменная *a* (исходное число).

II. Определение результатов: переменная *b* (преобразованное число).

III. Алгоритм решения задачи:

1. Ввод исходных данных.

2. Для преобразования числа необходимо выполнить следующие действия:

а) в переменной *a*1 сохраним вторую цифру числа. Для выделения цифры из числа нужно найти остаток от деления исходного числа на 10 (`a mod 10`);

б) для выделения первой цифры (переменная *a*2) нужно найти целую часть от деления числа на 10;

в) искомое число *b* получим, если умножим *a*1 на десять и к полученному произведению прибавим значение переменной *a*2.

3. Вывод результата.

IV. Описание переменных.

Все переменные, определенные для решения задачи, имеют тип `integer`.

Пример 16.4.

V. Программа:

```
var
  a, b, a1, a2: integer;
 begin
  write('введите a = ');
  readln(a);
  {Выделение последней
цифры}
  a1:= a mod 10;
  {Выделение первой цифры}
  a2:= a div 10;
  b:= a1 * 10 + a2;
  write('результат = ', b);
end.
```

VI. Тестирование программы.

Запустите программу и введите значение *a* = 25.

Результат работы программы должен быть следующим:

Окно вывода

```
введите a = 25
результат = 52
```

Издавна на Руси применялась система мер, отличная от современной Международной системы единиц (СИ). Например:

1 локоть = 45 см;
1 аршин = 16 вершков;
1 вершок = 4 ногтя;
1 ноготь ≈ 11 мм.

Пример 16.5.

V. Программа:

```
var l, m, s, x: integer;
begin
 write('введите l = ');
 readln(l);
 x:= l * 45;
 {метры}
 m:= x div 100;
 {сантиметры}
 s:= x mod 60;
 write(l, 'локтей = ');
 write(m, ' м ', s, ' см');
end.
```

VI. Тестирование программы.

Запустите программу и введите значение l = 7.

Результат работы программы должен быть следующим:

Окно вывода

```
введите l = 7
7 локтей = 3 м 15 см
```

Пример 16.5. В исторической книге длина отреза ткани измерялась в локтях. Напишем программу, которая переведет локти в метры и сантиметры.

Этапы выполнения задания:

I. Определение исходных данных: переменная l (локти).

II. Определение результатов: переменные m (метры) и s (сантиметры).

III. Алгоритм решения задачи:

1. Ввод исходных данных.
2. Сначала переведем локти в сантиметры. Для этого количество локтей нужно умножить на 45 и сохранить значение в переменной x.
3. Для определения числа метров найдем целую часть от деления x на 100.
4. Оставшиеся сантиметры можно найти как остаток от деления x на 100.
5. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют тип `integer`.

? Какой тип данных можно использовать в Pascal для работы с целочисленными данными?

2. Какое максимальное значение можно задать переменной типа `integer`?

3. Какие операции определены для целочисленных данных?

Упражнения

1 Вася написал программу, которая переводит длину из метров в километры и метры. Но он не может решить, где нужно использовать `div`, а где `mod`. Помогите ему. Откройте файл и исправьте программу.

```
var d, m, k: integer;
begin
  write('введите d = ');
  readln(d);
  k:= d … 1000;
  m:= d … 1000;
  write(d, ' м = ');
  write(k, ' км ', m, ' м');
end.
```

2 Ответьте на вопросы для примера 16.4.

1. При каких значениях переменной a значение переменной b будет таким же?
2. Всегда ли в результате выполнения программы мы будем получать двузначное число? Почему?
3. Попробуйте ввести трехзначное число (например, 125). Объясните получившийся результат.

3 Напишите программы для решения задач. Используйте операции `div` и `mod`.

1. Задано двузначное число. Найдите среднее арифметическое цифр числа.
2. Задано двузначное число. Найдите разность между количеством десятков и единиц.
3. Дана масса в граммах. Переведите ее в килограммы и граммы.
4. Площадь участка измеряется в арах. Найдите количество полных км^2.

4* Для старорусской системы весов известны следующие соотношения:

1 берковец = 10 пудов = 400 фунтов = 38 400 золотников.

Напишите программу, которая переводит массу, заданную в золотниках, в фунты, пуды и берковцы.

Глава 4
АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

§ 17. Современные компьютерные устройства

17.1. Виды компьютеров

Развитие вычислительной техники привело к появлению большого разнообразия устройств. Современные компьютеры имеют различную конструкцию и внешний вид.

Совокупность всех устройств компьютера называют его **аппаратным обеспечением.**

Пример 17.1.

Настольный компьютер

Ноутбук

Планшет

Настольный компьютер состоит из системного блока и подключенных к нему внешних устройств. Пользователь сам определяет качественный и количественный состав подключаемых к системному блоку устройств.

В **мобильных компьютерах** все необходимые устройства находятся в одном корпусе. Переносные компьютеры имеют возможность беспроводного подключения к внешним устройствам и сетям.

Основные разновидности мобильных компьютеров:

1. **Ноутбуки** — полноценные компьютеры с клавиатурой, экра-

ном, жестким диском и возможностью использования широкого спектра программ.

2. **Планшетные компьютеры (планшеты)** имеют ограниченные возможности, виртуальную клавиатуру и операционную систему с набором команд.

3. **Смартфоны** — телефоны с некоторыми возможностями компьютера. Современные смартфоны прекрасно справляются со многими задачами, не свойственными телефонам. Это работа с электронной почтой, создание и редактирование текстовых документов, просмотр фильмов, прослушивание музыки и многое другое.

Для решения наиболее сложных задач применяют **суперкомпьютеры**. Они обладают огромной вычислительной мощностью и превосходят по своим характеристикам большинство существующих в мире компьютеров. Среди областей их применения можно отметить математическое моделирование, метеорологию, авиационную промышленность, сейсмологию и др. Изображения различных видов компьютеров представлены в примере 17.1.

Компьютер-трансформер (ноутбук-планшет)

Смартфон

Суперкомпьютер

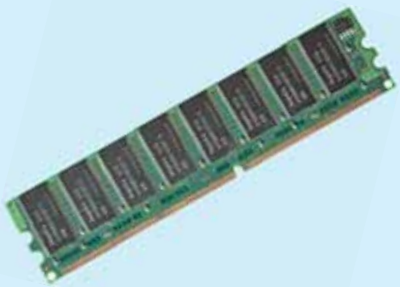
Один из возможных компьютеров будущего

Пример 17.2. Внутренние устройства компьютера.

Материнская плата

Процессор

Оперативная память конструктивно представляет собой набор микросхем, размещенных на одной небольшой плате. Модули оперативной памяти вставляются в соответствующие разъемы материнской платы.



Модуль оперативной памяти

17.2. Назначение устройств персонального компьютера

Состав устройств (конфигурация) компьютера может изменяться в зависимости от решаемых задач.

Базовая конфигурация настольного компьютера содержит следующие функциональные блоки: системный блок, монитор, клавиатуру, мышь. В мобильных компьютерах эти устройства интегрированы в единое целое.

В **системном блоке** размещаются: материнская плата, блок питания, устройства памяти, карты расширений (видеокарта, звуковая карта, сетевая карта).

Все компоненты компьютера связаны между собой самой большой печатной платой. Эту плату называют **материнской платой**. На ней установлен процессор.

Процессор — важнейшее устройство компьютера, его мозг. Он обрабатывает информацию, выполняя вычисления.

Устройства памяти предназначены для хранения информации. Память компьютера бывает внутренняя и внешняя.

Внутренняя память находится внутри компьютера и предназначена для хранения программ

и их данных в процессе работы компьютера.

Внешняя память предназначена для долговременного и энергонезависимого хранения программ и данных. К одному компьютеру можно подключить несколько устройств внешней памяти.

Внутренняя память подразделяется на оперативную и постоянную.

Оперативная память (RAM) служит для хранения программ и данных, с которыми компьютер работает в данный момент.

Обмен данными между процессором и оперативной памятью выполняется за очень короткие промежутки времени. При выключении электропитания вся информация исчезает из оперативной памяти.

Постоянная память (ROM) — энергонезависимая память для хранения программ управления работой и тестирования устройств компьютера.

Кроме программы первоначального тестирования компьютера, в постоянной памяти хранится BIOS (базовая система

Пример 17.3. Устройства внешней памяти.

Винчестер, который размещается внутри системного блока (на рисунке крышка винчестера снята).

Внешний винчестер, который подключается к портам системного блока.

Для переноса данных используют:

Оптические диски	Флеш-память
	

Пример 17.4. Периферийные устройства.

Видеопроектор

Документ-камера

Веб-камера

Микрофон

ввода-вывода). Данные в постоянную память заносятся при изготовлении компьютера.

Основным устройством долговременного хранения информации является **винчестер (жесткий диск).**

Винчестер находится внутри системного блока, но относится к внешним устройствам памяти. Существуют винчестеры, которые могут подключаться к системному блоку.

Винчестер можно условно разделить на несколько **логических дисков (разделов).** Обслуживание одного логического раздела не затрагивает другие разделы.

Кроме винчестера, к устройствам внешней памяти относятся оптические диски и флеш-память.

Не входящие в системный блок устройства называют **периферийными.**

Периферийные устройства ввода-вывода подключаются к **портам (разъемам)** материнской платы или карт расширений. Обычно они выводятся на заднюю панель компьютера.

С назначением клавиатуры, мыши, монитора, принтера и сканера вы познакомились в 6-м классе.

Рассмотрим назначение других периферийных устройств.

Видеопроектор предназначен для проецирования изображения на большой экран.

Документ-камера позволяет получить цифровое изображение любых предметов.

Веб-камера — малоразмерная цифровая видео- или фотокамера, способная в реальном времени фиксировать изображения, предназначенные для дальнейшей передачи по сети Интернет.

Для ввода звуковой информации используют **микрофон**, а для воспроизведения — акустические системы (**звуковые колонки** и **наушники**). Иногда микрофон и наушники объединяются в одно устройство — **гарнитуру**.

Рассмотрите примеры 17.2—17.4 на с. 110—113.

Колонки

Наушники

Гарнитура

? 1. Что такое аппаратное обеспечение компьютера?
2. Какие конструкции компьютеров вам известны?
3. Назовите мобильные компьютеры.
4. Для чего предназначен процессор?
5. Какая бывает память компьютера?
6. Для чего предназначена оперативная память?
7. Для чего предназначена постоянная память?
8. Назовите устройства внешней памяти.
9. Перечислите устройства ввода информации.
10. Назовите устройства вывода информации.

Упражнения

1 Определите объем памяти жесткого диска компьютера, к которому вы имеете доступ в вашем учебном заведении или дома. Какой объем занимает информация, хранящаяся на нем? Запишите данные в тетрадь.

2 Каким, на ваш взгляд, может быть компьютер будущего? Напишите эссе на эту тему.

3 С помощью графического редактора Paint создайте изображение компьютера будущего.

§ 18. Операционная система

18.1. Виды операционных систем

Операционная система (ОС) — комплекс программ, позволяющий пользователю общаться с компьютером, управляющий устройствами компьютера, программами и информацией, хранящейся в памяти компьютера.

Компьютеры работают под управлением различных операционных систем. Иногда на одном компьютере устанавливают несколько ОС. Наиболее распространенные семейства ОС для настольных компьютеров и ноутбуков: Windows, Mac OS, Linux. Многие смартфоны и планшеты работают под управлением ОС Android (примеры 18.1 и 18.2).

При включении компьютера операционная система начина-

Пример 18.1.

Значки наиболее распространенных ОС	
	Windows 10
	Mac OS
	Linux
	Android

ет автоматически загружаться с диска в оперативную память компьютера и остается там до выключения компьютера. ОС все время находится в рабочем состоянии. Загрузка ОС из долговременной памяти компьютера в оперативную называется **загрузкой компьютера.**

ОС выводит на экран монитора приглашение к работе в какой-либо форме. В ответ пользователь дает команду на выполнение конкретного действия. Если такая команда знакома ОС и в данный момент времени может быть выполнена, то ОС ее выполняет, если нет — пользователю выдается соответствующее сообщение. После этого ОС ожидает следующую команду пользователя. Такой режим работы называется **диалоговым режимом.**

У современных ОС диалоговый режим графический.

В **графическом режиме** пользователь может задавать команды ОС, выбирая их из различных меню. При такой организации диалога командам ОС соответствуют определенные значки (небольшие картинки).

Выбор команд часто осуществляют с помощью мыши. Так,

Пример 18.2. Графические режимы ОС.

Windows 7

Windows 10

Mac OS

Linux

ОС Android

Первая операционная система UNIX была разработана в 1969 г. в подразделении Bell Labs компании AT & T. С тех пор было создано большое количество различных UNIX-систем.

В 1981 г. корпорация IBM разместила запрос на создание операционной системы, которая должна была использоваться в новом семействе компьютеров IBM PC. Microsoft выкупила права на операционную систему 86-DOS у Seattle Computer Products и начала работу по ее модификации под требования IBM.

Пример 18.3.

Значки Рабочего стола	
	Компьютер
	Корзина
	Сетевое окружение
	Мои документы
	Проводник

например, организован диалог пользователя с операционными системами семейства Windows.

ОС позволяет пользователю работать с другими программами. Она находит программу в долговременной памяти (на диске), загружает ее в оперативную память и заставляет процессор выполнять команды данной программы. В оперативную память может быть загружено несколько программ. Такой режим работы называется **многозадачным.** ОС также организует и контролирует работу всех устройств компьютера.

На жесткий диск ОС устанавливается с загрузочного диска.

18.2. Элементы графического пользовательского интерфейса

Пользовательский интерфейс — совокупность средств и способов взаимодействия человека и компьютера.

Современные ОС имеют графический пользовательский интерфейс. Основными элементами графического интерфейса являются **окна** и **меню.**

Рассмотрим пользовательский интерфейс ОС Windows 7. После загрузки ОС Windows 7 на экра-

не компьютера появляется Рабочий стол с ярлыками и значками (пример 18.3).

Рабочий стол — основное окно графического интерфейса ОС, занимающее все пространство экрана.

Внизу Рабочего стола находится **Панель задач**. На ней расположена кнопка **Пуск**, кнопки выполняемых программ и открытых окон документов, индикатор языка и времени (даты и времени), индикатор сетевых подключений, регулятор громкости звука (пример 18.4). Кнопка **Пуск** вызывает **Главное меню**. С помощью Главного меню пользователь может получить доступ ко всем программам, установленным на компьютере.

Ярлык представляет собой ссылку на объект (файл, каталог) и всегда содержит стрелку . Двойной щелчок левой клавишей мыши запускает программу, на которую ссылается ярлык. **Значок** без стрелки используется для обозначения самого объекта — папки или файла (вид значков некоторых документов представлен в примере 18.5).

Пример 18.4.

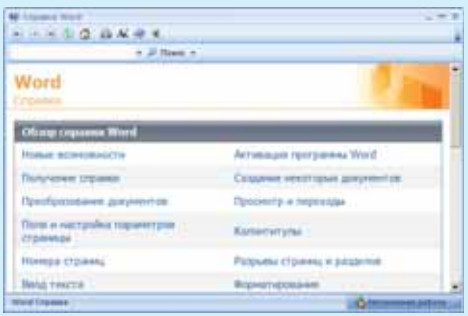
Элементы Панели задач	
	Кнопка **Пуск**
	Значки загруженных программ и открытых папок
	Индикатор даты и времени
	Индикатор сетевых подключений
	Регулятор громкости звука
	Индикатор языка

Пример 18.5.

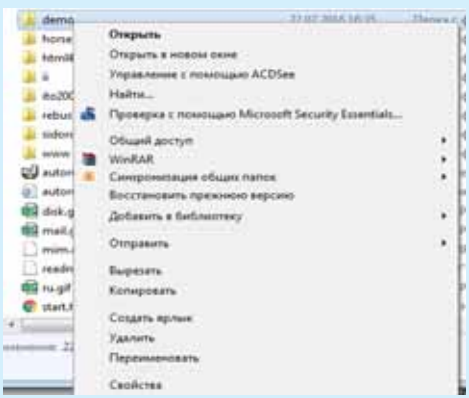
Значки программ и соответствующих документов	
Значок программы	Значок документа
Текстовый редактор Блокнот	
	
Power Point	
	
PascalABC.NET	
	

Пример 18.6. Диалоговое окно «Открытие документа».

Пример 18.7. Окно справочной системы MS Word.



Пример 18.8. Контекстное меню папки.



В процессе работы на Рабочем столе могут раскрываться диалоговые окна, окна папок, окна программ и окна справочной системы.

Диалоговые окна предназначены для организации диалога пользователя с компьютером (пример 18.6).

Окно программы появляется после запуска программы.

В **окне справочной системы** можно получить справку о работе программы (пример 18.7). Открываются такие окна после нажатия клавиши F1 в процессе работы с программой.

В 6-м классе вы узнали, что управлять работой программ позволяют различные меню.

Меню — список для выбора команд.

Доступ ко всем командам, возможным для данного объекта, можно получить с помощью **контекстного меню** (пример 18.8).

Контекстное меню вызывается щелчком правой клавиши мыши. Но есть и другие способы:

- специальной клавишей на клавиатуре:

- сочетанием клавиш Shift + F10.

18.3. Основные элементы файловой системы

Нужная нам информация хранится в компьютере в виде файлов. Работа с ними производится с помощью **файловой системы.**

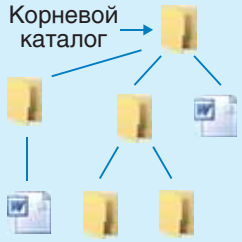
Файловая система предназначена для организации выполнения операций над файлами и папками (каталогами).

Каждая ОС поддерживает определенные файловые системы. Объектами любой файловой системы являются файлы, папки и диски.

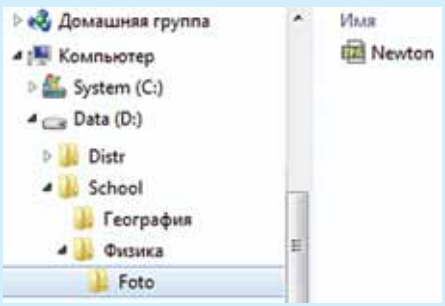
Структура файловой системы Windows представляет собой систему вложенных папок (пример 18.9). В каждой папке могут храниться другие папки и файлы. О папках или файлах, находящихся в другой папке, говорят, что они вложены в эту папку. Структуры, построенные на принципах вложенности (подчинения), называются **иерархическими.** Файловая система ОС Windows является иерархической.

Файловая система позволяет создавать, переименовывать и удалять файлы, переносить и копировать файлы с одного носителя на другой, искать файлы, хранящиеся на разных носителях, запускать программы на выполнение.

Пример 18.9. Структуру файловой системы можно представить как дерево с перевернутой кроной:



Пример 18.10. Путь к файлу `Newton.jpg`: `D:\School\Физика\Foto\`.



При переходе из папки `Foto` в папку `Физика` происходит подъем на один уровень иерархии. При обратном перемещении — спуск.

Полное имя файла `D:\School\Физика\Foto\Newton.jpg` означает, что: 1) файл `Newton.jpg` вложен в папку `Foto`; 2) папка `Foto` вложена в папку `Физика`; 3) папка `Физика` вложена в папку `School`, находится на диске `D:`.

Отображение пути к файлу `Newton.jpg` в адресной строке:

Пример 18.11. Семиклассник Вадим работал с каталогом E:\Mat\Form\1_ch\. Сначала он поднялся на один уровень вверх, затем еще раз вверх, потом опустился в каталог Prog, в котором находился файл z1.pas. Каков путь к этому файлу?

Решение

Вадим работал с каталогом E:\Mat\Form\1_ch\. Поднявшись на один уровень вверх, он оказался в каталоге E:\Mat\Form\. Поднявшись еще раз вверх, он оказался в E:\Mat\. После, опустившись в каталог Prog, Вадим оказался в E:\Mat\Prog\. Это и есть путь к файлу z1.pas.

Пример 18.12.

Значки файловых менеджеров для ОС Windows	
	Проводник
	Total Commander
	Far
	WinSCP
	Q-Dir

Одни папки создает пользователь, другие, такие как Мой компьютер или Корзина, создаются автоматически при установке операционной системы. Чтобы найти файл в файловой структуре, нужно указать путь к файлу.

Путь к файлу — последовательность папок, начиная от самой верхней и заканчивая той, в которой непосредственно хранится файл. Путь к файлу вместе с именем файла называют **полным именем файла.**

Путь начинается с **корневой папки** (имени диска) и содержит последовательность имен папок, в которые вложен файл. Диски именуются большими буквами английского алфавита с двоеточием после буквы. Имена дисков начинаются с C:. После имени каждой папки ставится обратный слэш (см. пример 18.10 на с. 119 и пример 18.11).

Для работы с файлами и папками используют программы, которые называют **файловыми менеджерами.**

Для каждой ОС созданы разные файловые менеджеры. В ОС Windows популярны Проводник, Total Commander, Far и др. (пример 18.12).

18.4. Типовые операции с файлами и папками

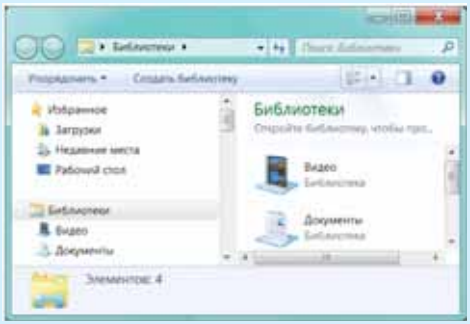
Программа **Проводник** позволяет пользователю создавать и удалять файлы и каталоги (папки), копировать и переносить их с одного носителя на другой, а также переименовывать файлы и папки (способы запуска Проводника представлены в примере 18.13; изменить отображение информации в Проводнике можно с помощью способов, указанных в примере 18.14). Действия по копированию, переносу и удалению файлов аналогичны действиям по копированию, переносу и удалению текстовых или графических фрагментов.

После **копирования** получают два одинаковых файла. В папке-источнике нужно выбрать объект для копирования и в контекстном меню объекта выполнить команды **Правка** → **Копировать.** После этого следует выбрать папку-приемник и в ее контекстном меню выполнить команды **Правка** → **Вставить.** В качестве источника и приемника может быть выбрана одна и та же папка.

Действия по **перемещению** файла (папки) аналогичны действиям по копированию. Сначала

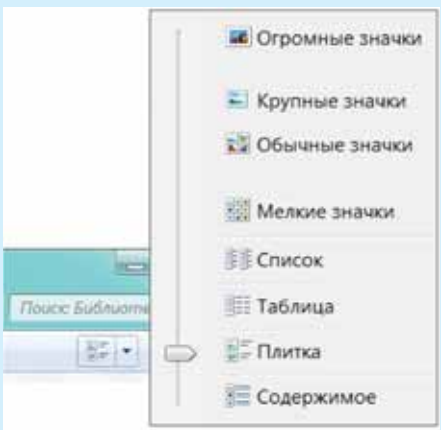
Пример 18.13. Разные способы запуска Проводника:

- ярлык программы Проводник на Рабочем столе;
- кнопка **Пуск (Все программы** → **Стандартные** → **Проводник);**
- контекстное меню кнопки **Пуск.**



Окно файлового менеджера Проводник

Пример 18.14. Способ отображения информации в окне Проводника выбирается с помощью меню Вид или кнопкой Изменить представление ( ) на Панели инструментов.



Пример 18.15. «Горячие» клавиши, используемые в Проводнике:

Вырезать	Ctrl + X
Копировать	Ctrl + C
Вставить	Ctrl + V
Выделить все	Ctrl + A

Пример 18.16.

Дополнительные операции с файлами, которые поддерживаются файловыми менеджерами:

Файловый менеджер / Операция			
Архивация файлов	Да	Да	Да
Цветовое выделение имен файлов	Нет	Да	Да
Вставка содержимого буфера обмена в виде файла	Частично	Да	Да

в папке-источнике нужно выбрать объект для перемещения и в контекстном меню объекта выполнить команды **Правка** → **Вырезать.** После этого выбирается папка-приемник и в контекстном меню папки-приемника выполняется команда **Правка** → **Вставить.**

Ненужные файлы и папки могут быть удалены. Для этого их выделяют, а затем нажимают клавишу Delete на клавиатуре или выполняют команду **Файл** → → **Удалить.** После удаления объекты обычно помещаются в **Корзину.** Корзина предназначена для временного хранения удаленных объектов. Удаленные из Корзины объекты восстановить с помощью операционной системы невозможно.

Для выполнения операций сразу с несколькими объектами их нужно выделить. Если объекты расположены рядом, то их выделяют так: щелкают мышью на первом объекте и, удерживая клавишу Shift, щелкают на последнем. Если выделяемые объекты не расположены рядом, то их выделяют, удерживая клавишу Ctrl.

Для работы в Проводнике можно использовать «горячие»

клавиши (пример 18.15), что позволяет ускорить выполнение некоторых действий.

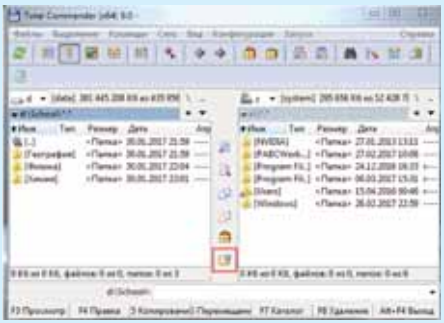
Внутри открытой папки можно **создать** новую папку. Для этого нужно нажать кнопку **Новая папка** в меню окна Проводника. Появится папка с именем «Новая папка» (пример 18.16). Это имя можно поменять на иное.

Чтобы **переименовать** файл или папку, можно воспользоваться соответствующим пунктом контекстного меню файла или папки.

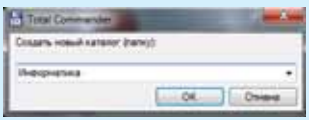
С процессом создания новой папки в файловом менеджере Total Commander познакомьтесь самостоятельно, рассмотрев пример 18.17.

Пример 18.17. Создание папки в Total Commander:

- открыть папку, внутри которой будет находиться новая папка;
- нажать кнопку **Создать каталог** или клавишу F7:



- в открывшемся окне **Создать новый каталог (папку)** ввести имя новой папки:



1. Что такое операционная система?
2. Какие основные функции выполняет ОС?
3. Что такое значок?
4. Что такое ярлык?
5. Для чего используется кнопка **Пуск**?
6. Что такое файловая система?
7. Какие операции позволяет выполнять файловая система?
8. Какие программы называют файловыми менеджерами?
9. Как запустить файловый менеджер?

Упражнения

1 Учитель работал в папке `E:\Для уроков\7 класс\Задания\`. Потом перешел на уровень выше, вошел в папку `Тема 4` и удалил из нее файл `Обобщение.docx`. Каково полное имя файла, который удалил учитель?

2 Измените фон Рабочего стола. Для этого выполните следующие действия:

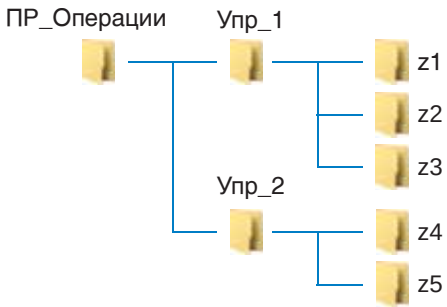
1. Щелкните правой клавишей мыши по Рабочему столу и в контекстном меню выберите **Персонализация.**
2. В открывшемся окне выберите одну из тем.

Покажите учителю результат своей работы.

3 Используя кнопку **Пуск**, загрузите на Рабочий стол следующие программы из подменю Стандартные: Блокнот, Калькулятор, Paint, WordPad. Расположите окна с программами Каскадом, Стопкой, Рядом, используя контекстное меню Панели задач. Покажите учителю свою работу, после этого закройте все окна.

4 Разнесите значки по разным углам Рабочего стола. Выстройте значки. Используя контекстное меню Рабочего стола, отсортируйте значки по типу, дате, размеру. Покажите учителю результат своей работы.

5 Создайте дерево папок следующей структуры:



1. В папке `z1` создайте файл — документ MS Word с именем `text1.docx`, в котором ответьте на вопрос 8 после этого параграфа (рубрика «вопросы и задания для проверки знаний»).
2. В папке `z2` создайте изображение смайла в графическом редакторе Paint, сохранив его под именем `ris1.bmp`.

3. Скопируйте файл `ris1.bmp` из папки `z2` в папку `z4`.
4. Скопируйте папку `Упр _ 1` в папку `z5`. Переименуйте папку `z5` в папку с именем `Copy`. Скопированные папки `z1`, `z2`, `z3` переименуйте в `Copy1`, `Copy2`, `Copy3` соответственно. Удалите папку `Copy3`.

§ 19. Локальная компьютерная сеть

Желание передавать информацию от одного компьютера к другому, обеспечить пользователям совместный доступ к техническим устройствам, программному обеспечению и информационным ресурсам компьютеров вызвало необходимость объединения компьютеров в единую сеть.

Компьютерная сеть — объединение компьютеров, обеспечивающее совместное использование сетевых ресурсов.

Компьютеры, расположенные на небольших расстояниях друг от друга, могут быть объединены в **локальную сеть.** Это, как правило, сеть одной организации, учебного заведения и др. (пример 19.1).

По способу организации локальные компьютерные сети делятся на **одноранговые** и **сети с выделенным сервером.**

В одноранговых сетях все компьютеры равноправны. Сеть с выделенным сервером имеет один высокопроизводительный компьютер, управляющий работой всей сети. Этот компьютер называется **сервером.** Он предоставляет свои ресурсы для совместного использования остальным

Пример 19.1.

Схема локальной сети

Примером локальной компьютерной сети является сеть в кабинете информатики. Она существует для того, чтобы учащиеся могли работать с одними и теми же информационными ресурсами и использовать общий принтер.

Единых правил поведения пользователей в локальной сети не существует. Тем не менее отметим некоторые общие требования:

- не передавайте другим пользователям ваше имя и пароль для входа в сеть;
- по возможности сохраняйте информацию на диске вашего компьютера, а не на дисках общего пользования.

Пример 19.2. К аппаратному обеспечению работы локальной сети относятся сетевые платы (карты) и специальный кабель. Сетевыми платами должны быть оснащены все компьютеры сети. Они предназначены для приема и передачи информации в сети.

Сетевая карта

Сетевой кабель

Сетевой порт

В беспроводных локальных сетях используется точка доступа, а на каждом компьютере должна быть установлена специальная беспроводная сетевая плата типа Wi-Fi.

компьютерам сети, называемым **клиентами**, и может управлять их работой.

По способу подключения компьютерные сети могут быть **проводными** и **беспроводными**.

Для организации работы компьютеров, объединенных в локальную сеть, необходимо соответствующее аппаратное (пример 19.2) и программное обеспечение.

Программную поддержку работы компьютеров в локальной сети выполняет операционная система.

Компьютеры объединяют в сети для совместного использования сетевых ресурсов. **Сетевыми ресурсами (ресурсами сети)** компьютеров могут являться:

1) технические устройства (модемы, принтеры, дисководы и др.);

2) программное обеспечение (операционные системы, различные редакторы и др.);

3) информационные ресурсы (файлы с информацией).

Для доступа к сетевым ресурсам часто бывает нужно указать имя пользователя и его пароль.

Пользователь, на компьютере которого находится ресурс (файл, диск, папка, устройство), является его владельцем и имеет полный доступ к этому ресурсу. Владелец ресурса может разрешить дру-

гим пользователям сети доступ к своему диску, папке, файлу.

Просмотр доступных сетевых ресурсов осуществляется в папке **Сеть**. В ней отображаются общие ресурсы сети, к которой подключен компьютер (компьютеры, папки, файлы, принтеры).

Важнейшей характеристикой работы локальной сети является скорость передачи информации в ней — количество информации, передаваемое за единицу времени. Скорость передачи информации по сети обычно измеряется в бит/с. (Рассмотрите решение задачи в примере 19.3.)

Пример 19.3. Определите объем файла компьютерной презентации, если передача его по сети происходит за 5 с при скорости 1 024 000 бит/с. Запишите полученный результат в килобайтах.

Решение

$1\,024\,000$ бит/с $\cdot$ 5 с =
$= (2^{10} \cdot 10^3 \cdot 5)$ бит =
$= 2^{10} \cdot (2^3 \cdot 5^4)$ бит $= 2^{13} \cdot 5^4$ бит.

Переведем биты в килобайты:

1 байт = 8 бит, или 2^3 бит;

1 Кбайт = 1024 байта, или 2^{10} байт;

1 Кбайт $= 2^3 \cdot 2^{10} = 2^{13}$ бит;

$(2^{13} \cdot 5^4) / 2^{13} = 5^4 = 625$ Кбайт.

1. Для чего компьютеры объединяют в сети?
2. Какая сеть называется локальной?
3. С помощью чего компьютеры объединяются в локальную сеть?
4. Какие бывают локальные сети?
5. Назовите сетевые ресурсы.

Упражнения

1 Определите объем видеофайла, если передача его по сети длилась 1 мин 20 с при скорости $80 \cdot 10^{20}$ бит/с. Запишите полученный результат в мегабайтах.

2 Определите объем звукового файла, если передача его по сети длилась 0,5 с при скорости 155 мегабит/с. Запишите полученный результат в байтах.

3 Определите скорость передачи информации по сети, если архивный файл объемом 1,5 Гбайт передавался 2 мин.

4 Определите скорость передачи информации по сети, если файл с компьютерной игрой объемом 3,2 Гбайт передавался 3 мин 45 с.

§ 20. Архивация

В 40-х гг. XX в. ученые, работавшие в области информационных технологий, пришли к выводу, что можно разработать такой способ хранения данных, при котором пространство для хранения будет расходоваться более экономно. Одними из первых алгоритмов по сжатию данных являются алгоритмы Шеннона — Фано и Хаффмана.

Клод Элвуд Шеннон (1916—2001) — американский инженер и математик, основатель теории информации. Большинство базовых понятий теории сжатия информации было разработано Клодом Шенноном.

Роберт Марио Фано (1917—2016) — итальяно-американский ученый, известный работами в области теории информации. Он независимо от Клода Шеннона изобрел алгоритм сжатия информации.

20.1. Программы-архиваторы

Для рационального хранения информации на компьютерных носителях используются программы-архиваторы, позволяющие представить информацию в упакованном виде. Архивы создают в следующих случаях:

- необходимо создать резервные копии наиболее важных файлов;
- требуется освободить место на диске;
- следует передать файлы по электронной почте;
- планируется перенести большое количество файлов на другой носитель;
- нужно защитить информацию от несанкционированного доступа — запаролить ее.

Упаковывать файлы и размещать их в специальных архивах позволяют **программы-архиваторы. Архивный файл (архив)** хранит в упакованном виде другие файлы (один или несколько), которые при необходимости могут быть извлечены из архива в первоначальной форме.

Программы-архиваторы могут выполнять следующие функции:

- помещение исходных файлов в архив;

- извлечение файлов из архива;
- удаление файлов из архива;
- просмотр оглавления архива;
- проверка архива.

Размер архива обычно меньше размера всех входящих в него файлов. Для преобразования информации программы-архиваторы используют разные алгоритмы, поэтому размеры архивов, содержащих одни и те же файлы, но созданных с помощью различных архиваторов, могут отличаться.

Информация в архиве хранится в закодированном виде, поэтому для просмотра содержимого архивного файла нужно воспользоваться программой-архиватором. Чтобы работать с файлом, его следует извлечь из архива. Делают это, используя ту же программу, с помощью которой создавался архив, или с помощью другой программы, распознающей данный тип архива.

Дэвид Хаффман (1925—1999) разработал метод построения минимально-избыточных кодов. Ученый внес важный вклад в информатику и во множество других областей знания (по большей части в электронику). В 1952 г. создал алгоритм кодирования, известный как алгоритм, или код, Хаффмана.

Большие архивы данных хранят в специальных хранилищах информации — дата-центрах.

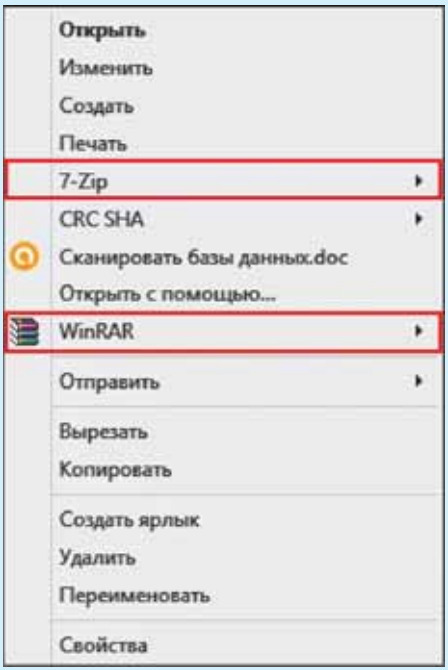
20.2. Создание архивов и извлечение файлов из архива

Сегодня существует большое количество программ-архиваторов: WinRar, 7-Zip, WinZip и др. (пример 20.1). Архивные файлы имеют расширения, соответствующие программам, с помощью которых их создавали: .rar, .7z, .zip.

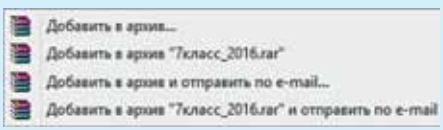
Пример 20.1.

Значки программ-архиваторов			
	WinRar		WinZip
	7-Zip		WinAce

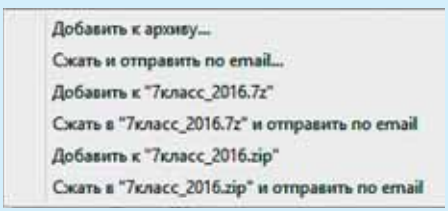
Пример 20.2. Контекстное меню с выбором архиватора:



Пример 20.3. Команды меню архиватора по добавлению файлов в архив:



Архиватор WinRar



Архиватор 7-Zip

При просмотре списка файлов в Проводнике архивные файлы помечаются значками (.rar), (.7z), (.zip).

При установке программ-архиваторов действия по созданию архивов и извлечению файлов из архива добавляются в контекстное меню любого объекта файловой системы.

Для создания архивного файла необходимо:

1. Открыть Проводник.
2. Выделить файлы.
3. Щелкнуть правой клавишей мыши.
4. Выбрать программу-архиватор (пример 20.2).
5. Выбрать одну из команд:

а) «Добавить в архив (к архиву)»;

б) «Добавить в архив (к архиву)» с предложенным именем (пример 20.3).

Архив с предложенным именем создается в текущей папке. Если выбрана команда «Добавить в архив (к архиву)», то пользователю нужно задать имя архива и указать папку, в которой он будет храниться.

Для извлечения файлов из архива нужно:

1. Открыть Проводник.
2. Выбрать архивный файл.

3. Щелкнуть правой клавишей мыши.

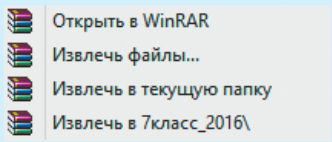
4. Выбрать одну из команд:

а) «Извлечь в текущую папку (Распаковать здесь)»;

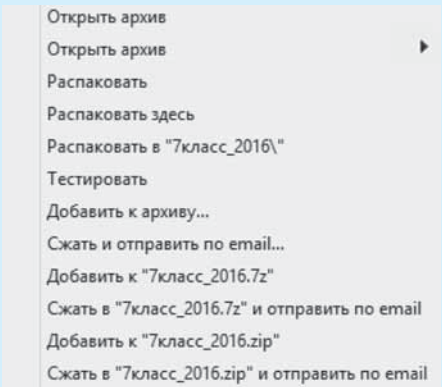
б) «Извлечь файлы... (Распаковать)» (пример 20.4).

При выборе команды «Извлечь в текущую папку (Распаковать здесь)» файлы из архива будут помещены в ту же папку, в которой находился архив. При выборе команды «Извлечь файлы... (Распаковать)» пользователь должен указать имя папки, в которую будут извлечены файлы.

Пример 20.4. Команды меню по извлечению файлов из архива:



Архиватор WinRar



Архиватор 7-Zip

1. Какой файл называют архивным?
2. Для чего предназначены программы-архиваторы?
3. Как заархивировать файл(-ы)?
4. Как извлечь файл(-ы) из архива?

Упражнения

1 Создайте рисунок в графическом редакторе Paint. Сохраните файл на диске. Заархивируйте этот файл. Сравните размеры исходного и архивного файлов.

2 Заархивируйте графический файл разными архиваторами. Сравните размеры полученных архивов.

3 Заархивируйте файлы разных типов: рисунки, тексты, программы.

4* Сравните размеры исходных файлов и их архивов разных типов. Какие файлы сжимаются лучше?

5 Извлеките файлы из архива, указанного учителем, в свою папку.

§ 21. Программное обеспечение

21.1. Классификация программного обеспечения

Программное обеспечение (ПО) — совокупность всех компьютерных программ.

Компьютер рассматривают как единую систему, состоящую из аппаратного обеспечения, программного обеспечения и информационных ресурсов. ПО компьютера постоянно изменяется, совершенствуется, дополняется.

Программное обеспечение компьютера по назначению бывает:

1) системное;
2) прикладное;
3) инструментальное.

Системное ПО — программы для обеспечения работы компьютера и компьютерных сетей. Системное ПО позволяет пользователю осуществлять руководство и контроль над работой компьютера и компьютерной сети, а также обеспечивает возможность выполнения других программ. К системному программному обеспечению относятся уже знакомые вам операционные системы, файловые менеджеры, архиваторы.

Прикладное ПО — программы для решения задач определенного

Пример 21.1. Прикладные программы.

Редакторы обрабатывают информацию, представленную в текстовой, графической, звуковой, числовой форме. Например: текстовый редактор MS Word (), графический редактор Paint (), графический редактор Inkscape ().

Мультимедийные программы сочетают возможность работы с видеофрагментами, звуком, анимацией, статическими картинками и гипертекстом. Например: программа для создания презентаций MS Power Point ().

Коммуникационные программы предназначены для поддержки пользовательского интерфейса при работе в сети. Например, браузеры: Internet Explorer (), Mozilla Firefox (), Google Chrome ().

Программы-переводчики переводят информацию с одного естественного языка на другой. Существуют программы-словари для перевода отдельных слов ABBYY Lingvo (), программы для перевода текстов Promt () и онлайновые сервисы на Google, Yandex и др.

класса предметной области. Прикладное ПО самое многочисленное (пример 21.1). Для наименования прикладных программ используют термин **приложения**. К прикладным программам относятся:

- программы общего назначения (требуются практически каждому пользователю);
- программы специального назначения (предназначены для профессионального использования в различных сферах деятельности);
- компьютерные игры.

Инструментальное ПО предназначено для создания другого ПО (пример 21.2). С инструментальным ПО работают программисты.

Программы по их правовому статусу можно разделить на группы:

- платные;
- свободно распространяемые;
- условно-бесплатные;
- пробные (оценочные);
- демонстрационные.

Примеры свободно распространяемого ПО: ОС Linux, графический редактор Inkscape, антивирусные программы AVAST и AVG, среда программирования PascalABC.NET.

Пример 21.2. К инструментальному программному обеспечению относится уже известная вам среда программирования PascalABC.NET ().

21.2. Вредоносные программы и способы защиты от них

Вредоносные программы — специально написанные программы, способные нанести ущерб информации, хранящейся на компьютере, или вывести компьютер из строя.

По способу распространения вредоносные программы делятся на компьютерные вирусы, сетевые черви и троянские программы.

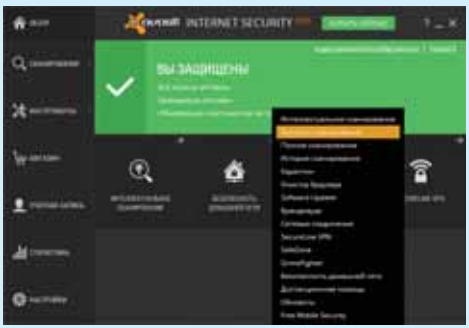
Компьютерные вирусы могут распространяться самостоятельно,

Прообраз современных вирусов — программа «Дарвин». В 1962 г. инженеры из американской компании создали игру с таким названием. Смысл ее состоял в удалении всех копий программы противника и захвате поля битвы. Программы-вирусы возникли более двадцати лет спустя.

Регулярное архивирование и резервное копирование файлов позволит минимизировать ущерб от вирусной атаки.

Пример 21.3. Интерфейс популярных антивирусных программ.

Антивирус Касперского



AVAST

Norton AntiVirus

добавляя свой код к другим файлам.

Сетевые черви не изменяют файлы на дисках, а распространяются в компьютерной сети — проникают в операционную систему компьютера, находят адреса других компьютеров или пользователей и рассылают по этим адресам свои копии.

Троянские программы — это вредоносные программы, которые сами не распространяются, а, маскируясь под популярную программу, побуждают пользователя переписать вредителя и установить его на свой компьютер самостоятельно.

Большинство вирусов разрабатывается для причинения вреда пользователям, работающим с операционными системами семейства Windows.

При заражении компьютера вирусом важно его обнаружить.

Признаки заражения:

1) медленная работа компьютера;

2) зависания и сбои в работе компьютера;

3) изменение размеров файлов;

4) уменьшение размера свободной оперативной памяти;

5) значительное увеличение количества файлов на диске;

6) исчезновение файлов и папок или искажение их содержимого.

Для борьбы с вредоносными программами используют программные средства антивирусной защиты: Антивирус Касперского, Norton AntiVirus, AVAST, Dr.Web, AVG и др. (примеры 21.3 и 21.4).

Сканирование компьютера в поисках вредоносных программ обычно выполняется автоматически при каждом включении. При сканировании антивирусная программа ищет вирус путем сравнения кода программ с кодами известных ей вирусов, хранящихся в базе данных.

Одним из основных способов борьбы с вредоносными программами является своевременная профилактика (предотвращение заражения).

Чтобы предотвратить заражение компьютера, нужно соблюдать следующие рекомендации:

1) не запускайте программы, полученные из Интернета, без проверки на наличие в них вируса;

2) проверяйте все внешние носители на наличие вирусов, прежде чем копировать или открывать содержащиеся на них файлы;

3) установите антивирусную программу и регулярно пользуйтесь ею для проверки компьютеров.

Пример 21.4.

Значки популярных антивирусных программ	
	Антивирус Касперского
	AVAST
	Norton AntiVirus
	Dr.Web
	AVG

Самым разрушительным вирусом за всю историю их существования считают вирус «ILOVEYOU». Он был разослан на почтовые ящики с Филиппин в 2000 г.; в теме письма содержалась строка «ILoveYou», а к письму был приложен скрипт «LOVE-LETTER-FOR-YOU.TXT.vbs». При открытии вложения вирус рассылал копию самого себя всем контактам в адресной книге Windows, а также на адрес, указанный как адрес отправителя. Он также совершал ряд вредоносных изменений в системе пользователя.

Вирус поразил более 3 млн компьютеров по всему миру. Ущерб, который червь нанес мировой экономике, был настолько велик, что вирус вошел в Книгу рекордов Гиннесса.

1. Что такое программное обеспечение?
2. На какие классы можно разделить программное обеспечение в зависимости от назначения?
3. Какие программы называются вредоносными?
4. Назовите виды вредоносных программ.
5. Какие признаки указывают на то, что компьютер заражен?
6. Что необходимо делать, чтобы предотвратить заражение компьютера?
7. Укажите программы антивирусной защиты.

Упражнения

1 Перечислите названия известных вам программ:

1. Платные.
2. Свободно распространяемые.

2 Запишите в тетради названия антивирусных программ, которые установлены у вас дома, в школьном компьютерном кабинете, у ваших друзей.

3 Определите, к какому классу программного обеспечения относятся программы, значки которых представлены на рисунках.

а 	*б* 	*в* 	*г* 
д 	*е* 	*ж* 	*з* 
и 	*к* 	*л* 	*м* 

Глава 5
РАБОТА С ВЕКТОРНОЙ ГРАФИКОЙ

§ 22. Понятие векторной графики

Одно из направлений использования компьютера — создание и обработка графических изображений. Например, схем, чертежей, рисунков, фотографий.

Компьютерная графика — область информатики, изучающая методы и средства создания и обработки изображений с помощью аппаратного и программного обеспечения компьютера.

В зависимости от способа представления в памяти компьютерные изображения можно разделить на два вида: растровую и векторную графику.

Растровая графика — изображение, представляющее собой совокупность пикселей, окрашенных в разные цвета.

Векторная графика — изображение в виде геометрических фигур (графических примитивов), описанных математическими формулами.

(Рассмотрите пример 22.1.)

Основные области применения компьютерной графики: научная, деловая, конструкторская, иллюстративная сферы.

Различия в представлении графической информации в растровом и векторном виде существует для графических файлов и способов их обработки. На экран монитора графическую информацию можно вывести только в растровом виде.

Пример 22.1.

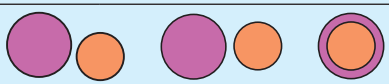
Векторное изображение

Векторное изображение можно сравнить с аппликацией, состоящей из кусочков цветной бумаги, наклеенных (наложенных) один на другой. Однако, в отличие от аппликации, в векторном изображении легко менять форму и цвет составных частей.

Пример 22.2.

Графические примитивы	
Линия	
Прямоугольник	
Эллипс	
Треугольник, многоугольник	
Звезда	

Пример 22.3. Интересной особенностью векторных редакторов является возможность изменения формы нарисованных от руки кривых. Также у векторных редакторов имеются средства расположения объектов относительно друг друга и команды спецэффектов.

Изменение формы кривых	
	
Взаимное расположение объектов	
	
Применение спецэффектов	
Тень	
Объем	

Графические примитивы — простые геометрические фигуры: прямоугольник, окружность, эллипс, линия и т. д. (пример 22.2). С помощью математических формул описываются форма, цвет и пространственное положение составляющих изображение графических примитивов.

Графический примитив — независимый объект, который можно редактировать.

Положение и форма графических примитивов задаются в системе графических координат, которая связана с экраном. Начало координат расположено в верхнем левом углу экрана. Ось *OX* направлена слева направо, ось *OY* — сверху вниз. Координатная сетка совпадает с сеткой пикселей.

Достоинства векторного изображения:

1) небольшой размер графического файла;

2) преобразования без искажений;

3) рисование осуществляется быстро и просто;

4) независимое редактирование частей изображения;

5) высокая точность прорисовки.

(Прочитайте пример 22.3.)

Однако в векторной графике практически невозможно достичь фотореалистичности.

Для описания цвета изображений используются различные цветовые модели.

Под **цветовой моделью** понимается способ описания цвета.

Цветовые модели описывают цветовые оттенки с помощью смешивания нескольких основных цветов. Любой цвет можно разложить на оттенки и сопоставить ему набор чисел — цветовых координат.

Основные цвета разбиваются на оттенки по яркости — от темного к светлому. Каждому оттенку присваивается числовое значение (например, самому темному — 0, самому светлому — 255).

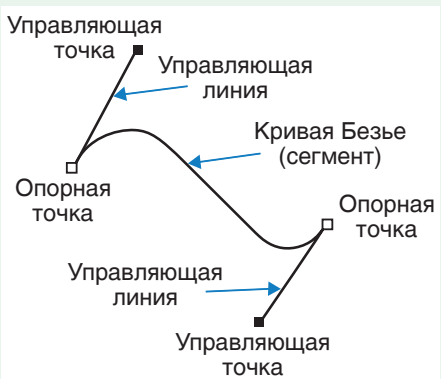
Одна из наиболее распространенных цветовых моделей называется RGB (пример 22.4). Любой цвет в этой модели представляет собой сложение трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). Именно на такой модели построено воспроизведение цвета современными мониторами и телевизорами.

В полиграфии используется цветовая модель, называемая

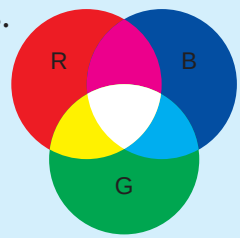
В прошлом инженеры, создавая чертежи больших деталей в натуральную величину, использовали тонкие планки, чтобы провести кривые по заданным точкам. Эти планки назывались сплайнами (гибкими лекалами).

Сейчас в векторных редакторах тоже используются сплайновые кривые — кривые Безье. Свое название они получили в честь французского математика Пьера Безье (1910—1999).

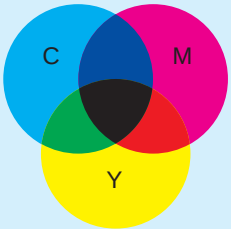
Ученый предложил описывать кривую, опираясь на вершины многоугольника, заключающего ее в себе:



Пример 22.4. Цветовая модель RGB.

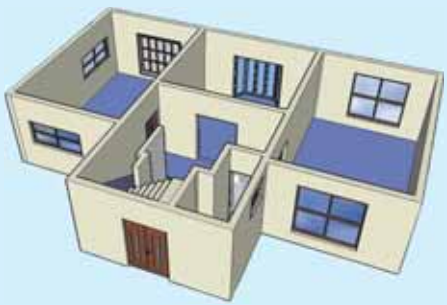


Пример 22.5. Цветовая модель CMYK.



Пример 22.6. Области применения векторной графики:

- промышленное проектирование;
- визуализация трехмерных объектов;
- архитектура и строительство;
- ландшафтный дизайн;
- построение графиков поверхностей;
- полиграфия, реклама.



CMYK (пример 22.5). Основные цвета в ней — голубой, пурпурный, желтый. Данную цветовую модель часто применяют для принтеров.

Уже знакомый вам графический редактор Paint предназначен для работы с растровой графикой. Растровые графические редакторы используют не столько для создания изображений, сколько для их обработки. Векторные редакторы ориентированы на создание изображений. Векторная графика может включать в себя и изображения растровой графики.

Векторные графические редакторы позволяют сохранять изображения в различных векторных форматах, среди которых можно выделить универсальные графические форматы и форматы отдельных векторных редакторов.

Одним из недостатков векторной графики является программная зависимость. Изображение, созданное в одном векторном редакторе, как правило, не преобразуется в формат другой программы без погрешностей.

Программы векторной графики нашли применение в области технического рисования, чертежно-графических и оформительских работ, графического и полиграфического дизайна (пример 22.6).

Известные векторные редакторы: CorelDraw, Adobe Illustrator, Inkscape (пример 22.7). Векторные графические редакторы позволяют выполнять разнообразные операции над графическими объектами.

Несмотря на разнообразие векторных графических редакторов, основные приемы работы с векторными изображениями остаются неизменными.

Пример 22.7. CorelDraw и Adobe Illustrator — платные программы. Редактором Inkscape можно пользоваться бесплатно (http://www.inkscape.org).

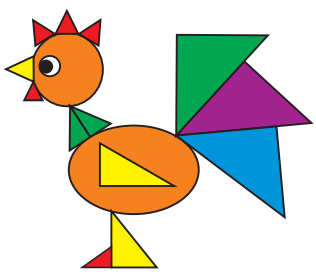
Значки векторных графических редакторов	
	CorelDraw
	Adobe Illustrator
	Inkscape

1. Какая графика называется векторной?
2. Что такое цветовая модель?
3. Как воспроизводится цвет в цветовой модели?
4. На какой модели построено воспроизведение цвета мониторами?
5. Что такое графический примитив?
6. Как называют программу, позволяющую работать с векторной графикой?

Упражнения

1 Приведите примеры графических примитивов.

2 Определите, из каких графических примитивов составлено изображение петуха.

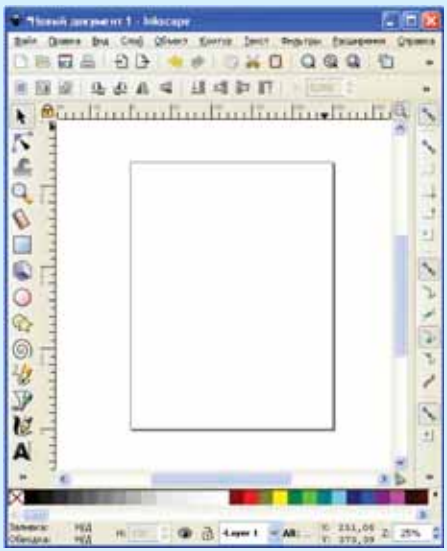


3 С помощью графических примитивов составьте в тетради изображения:

1. Домика.
2. Цветка.
3. Кошки.

§ 23. Интерфейс векторного графического редактора Inkscape

Пример 23.1. Окно векторного графического редактора Inkscape.



Пример 23.2. Векторный редактор Inkscape может работать с файлами различных форматов, например:

.svg — формат, который использует редактор Inkscape;

.eps — формат, обеспечивающий высокое качество рисунка;

.png — растровый формат изображений, поддерживающий прозрачность фона;

.bmp — несжатый растровый формат изображений;

.pdf — формат обмена документов от Adobe, который может содержать любые сочетания: текст, шрифты, растровую и векторную графику.

Рассмотрим технологию работы с векторной графикой на примере редактора Inkscape (пример 23.1). Редактор имеет встроенный учебник на русском языке и удобный интерфейс, позволяющий легко и быстро освоить основные приемы работы с векторной графикой.

Основную часть окна редактора Inkscape занимает **холст,** на котором пользователь создает и редактирует изображения. На холсте выделена **страница.**

Перемещаться по холсту можно при помощи полос прокрутки. Увеличение или уменьшение масштаба страницы осуществляется при помощи клавиш «+» или «−» на клавиатуре. Границы отображаемой на холсте страницы определяют границы изображения для печати или сохранения.

Многие действия в редакторе Inkscape можно выполнить несколькими способами:

- через пункты меню;
- при помощи кнопок на панелях;
- с помощью комбинаций клавиш.

Через меню **Правка** можно отменить последнее действие и повторить отмененное действие. Также доступна история действий.

Запуск программы осуществляется с помощью меню **Пуск: Все программы → Inkscape** — или двойным щелчком по ярлыку  на Рабочем столе. Для каждого документа редактор Inkscape открывает отдельное окно.

Чтобы сохранить изображение в редакторе Inkscape, в пункте меню **Файл** нужно выбрать **Сохранить как** и нажать кнопку Сохранить. Файл будет сохранен в собственном формате редактора Inkscape — .svg. Изображение, имеющее данный формат, следует открывать самим редактором Inkscape, а использование других программ может привести к некорректному результату. Основные форматы файлов, с которыми можно работать в Inkscape, представлены в примере 23.2. Как сохранить изображение в другом формате, описано в примере 23.3.

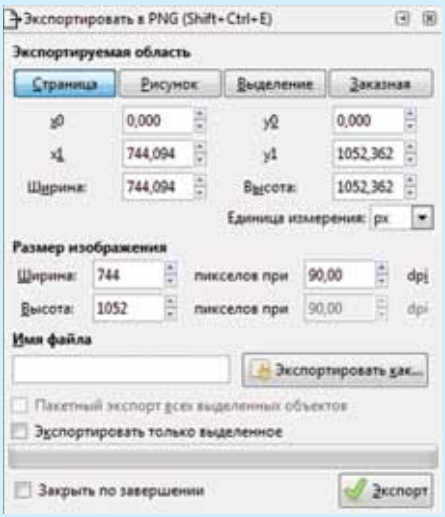
Для загрузки изображения в пункте меню **Файл** нужно выбрать **Открыть**.

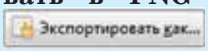
Подробнее об элементах интерфейса векторного редактора Inkscape можно узнать из материалов *Приложения* (см. с. 170).

Пример 23.3. Для сохранения какого-либо изображения в формате, отличном от собственного, необходимо при сохранении выбрать соответствующий тип файла.

Чтобы сохранить изображение в формате .png, нужно открыть окно «Экспортировать в PNG». Для этого необходимо:

1) в пункте меню **Файл** выбрать **Экспортировать в PNG**:



2) в окне **Экспортировать в PNG** нажать кнопку 

и в открывшемся окне выбрать папку для сохранения файла и указать имя;

3) в окне **Экспортировать в PNG** нажать кнопку 

.

Можно выделить часть изображения, и в файл PNG будет сохранено только выделение.

1. Как запустить векторный редактор Inkscape?
2. Какой формат файла является собственным форматом редактора Inkscape?
3. Как увеличить (уменьшить) масштаб страницы?
4. С файлами каких форматов может работать векторный редактор Inkscape?
5. Как сохранить часть изображения в файле формата PNG?

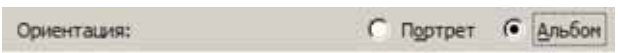
Упражнения

1 Загрузите векторный редактор Inkscape. Откройте встроенный в него учебник. Для этого в главном меню выполните команду **Справка** → **Учебник** → **Inkscape: Основы**. Руководствуясь материалом разделов **Перемещение по холсту** и **Изменение масштаба**, выполните:

1. Вертикальное перемещение по холсту; горизонтальное перемещение по холсту.
2. Увеличение и уменьшение масштаба холста.

2 Внесите изменения в интерфейс редактора.

1. Измените ориентацию холста на Альбом. Для этого выполните команду главного меню редактора **Файл** → **Свойства документа**. В диалоговом окне **Свойства документа** во вкладке **Страница** выберите:



2. Отобразите сетку. Для этого выполните команду главного меню редактора **Вид** → **Сетка страницы**.
3. Скройте линейки и палитру цветов, выполнив команду главного меню редактора **Вид** → **Показать или скрыть** и убрав «птички» в соответствующих пунктах. Повторив команду, возвратите отображение линеек и палитры цветов.
4. Просмотрите историю выполненных вами действий с помощью команды главного меню редактора **Правка** → → **История действий**.

§ 24. Создание и редактирование векторного изображения

Работу в векторных графических редакторах можно представить как создание фигур (объектов) нужной формы и задание им определенных заливок и обводок (абрисов) (пример 24.1).

Рассмотрим технологию создания фигур в редакторе Inkscape.

Начать работу над созданием векторного изображения необходимо с определения:

1) набора фигур, из которых будет состоять изображение;

2) стиля и цвета обводки;

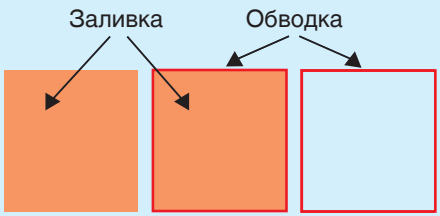
3) стиля заливки;

4) взаимного расположения фигур.

Пример 24.1. Основные инструменты создания фигур в векторном редакторе Inkscape:

	Прямоугольники и квадраты
	Круги, эллипсы и дуги
	Звезды и многоугольники

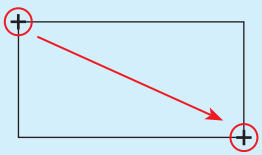
Образцы заливки и обводки:



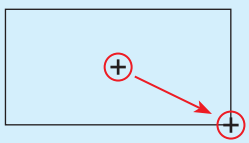
24.1. Создание фигур

Прямоугольники создают с помощью инструмента «Прямоугольник»:  . Прямоугольник строят, перемещая мышь по холсту и удерживая левую кнопку мыши. Если при этом удерживать клавишу Ctrl клавиатуры, то получится квадрат. Если при рисовании прямоугольника удерживать клавишу Shift, то начальная точка будет соответствовать не вершине, а центру прямоугольника (пример 24.2).

Пример 24.2. Рисование прямоугольника.

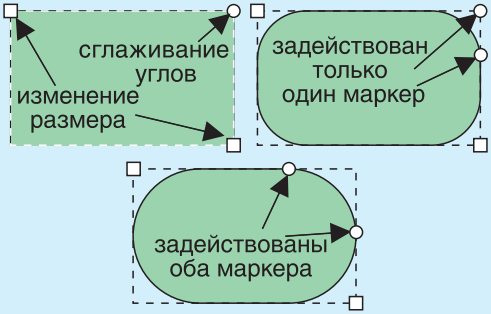


Начальная точка — конец диагонали

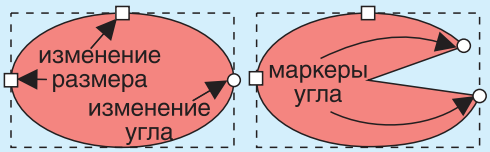


Начальная точка — центр прямоугольника (при нажатой клавише Shift)

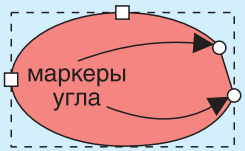
Пример 24.3. Использование инструмента «Прямоугольник».



Пример 24.4. Использование инструмента «Круги, эллипсы и дуги».



Если потянуть видимый круглый маркер вниз, то получится сектор и появится другой маркер, который можно перемещать вверх.



Если попробовать переместить круглый маркер внутрь эллипса, то режим рисования сегмента изменится на дугу. Для того чтобы снова переключиться на сегмент, нужно переместить один из круглых маркеров за пределы контура эллипса.

Квадратные маркеры в противоположных углах служат для изменения размера прямоугольника, а круглый маркер в верхнем правом углу управляет сглаживанием углов (пример 24.3).

Сглаживание (закругление) угла — замена угла на часть окружности или эллипса, вписанного в этот угол.

Заменить угол частью окружности можно, перетащив круглый маркер вниз. При этом появляется второй маркер. Чтобы заменить угол частью эллипса, нужно перемещать оба маркера.

Убрать сглаживание можно, нажав на кнопку  на панели свойств инструмента «Прямоугольник» (кнопка появляется, если инструмент активен). На этой же панели можно задать ширину и высоту прямоугольника с помощью числовых параметров.

Для рисования эллипсов и кругов выбирают инструмент «Круги, эллипсы и дуги»:  . Эллипсы и круги можно превратить в сегменты и дуги (пример 24.4). Рисуются эллипс и круг так же, как и прямоугольник. При удержании клавиши Shift начальная точка рисования будет соответствовать центру эллипса.

Для рисования звезд и многоугольников используют инструмент «Звезды и многоугольники»: (пример 24.5). Начальная точка соответствует центру звезды, конечная — одной из ее вершин. Определить тип фигуры (звезда или многоугольник) можно только на панели свойств инструмента, на которой есть два значка с соответствующим изображением многоугольника и звездочки:

Там же можно изменять и количество углов. По умолчанию рисуются пятиугольные звезды.

Для изменения звезды или многоугольника есть два маркера узлов. Верхний узел служит для управления положением конца звезды или углом многоугольника. Второй маркер (базовый узел) контролирует положение внутренней вершины звезды.

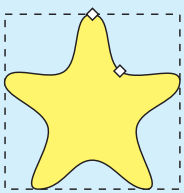
Для изменения формы звезды используются клавиши:

- Shift — управляет сглаживанием углов звезды (пример 24.6);
- Alt — все вершины звезды или многоугольника будут перемещаться в произвольном порядке (пример 24.7);
- Ctrl — перемещает узлы звезды или многоугольника строго по прямой линии.

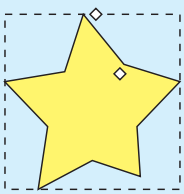
Звезда — вид плоских невыпуклых многоугольников, не имеющий однозначного математического определения. Правильной называется звезда, у которой все внутренние углы равны и все внешние углы равны. Слова, оканчивающиеся на *-грамма*, обозначают *n*-конечную звезду (звездчатый многоугольник). Октограмма — восьмиконечная звезда, гексаграмма — шестиконечная и т. д.

Пример 24.5. Использование инструмента «Звезды и многоугольники» для рисования.

Пример 24.6. Использование инструмента «Звезды и многоугольники» для сглаживания углов.

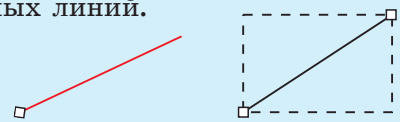


Пример 24.7. Использование инструмента «Звезды и многоугольники» для перемещения вершин.



Пример 24.8. Использование инструмента «Карандаш».

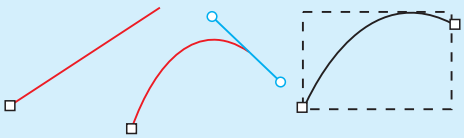
Пример 24.9. Рисование прямых линий.



Начало рисования

Завершение рисования

Пример 24.10. Рисование кривых линий.



1. В начале линии щелкнуть по ней левой кнопкой мыши. За курсором будет тянуться прямая. Нажать на левую кнопку мыши в точке, где должен находиться конец линии.

2. Не отпуская левую кнопку мыши, перенести курсор в сторону. Появится прямая линия, к которой как касательная будет примыкать кривая.

3. Отпустить левую кнопку мыши и щелкнуть правой. Появится кривая, а прямая пропадет. Можно продолжать проводить линию, не нажимая правую кнопку мыши.

Для рисования произвольных кривых используется инструмент «Карандаш»:  . Чтобы им воспользоваться, нужно щелкнуть в начальной точке на холсте и, не отпуская кнопку мыши, тащить указатель, проводя линию.

Чтобы получить замкнутый контур, достаточно вернуться к первоначальной точке. Когда указатель мыши приближается к ней, точка меняет цвет на красный. В этот момент кнопку мыши следует отпустить.

Для инструмента «Карандаш» устанавливается степень сглаживания:  . Чем выше значение сглаживания, тем более гладкой будет кривая (пример 24.8).

Для создания кривых и прямых линий предназначен инструмент «Кривая Безье и прямые линии»:  . Рисуя прямую, нужно щелкнуть левой кнопкой мыши по месту, где должна находиться начальная точка. Затем следует отвести курсор в другое место. При этом за курсором будет следовать прямая линия от ее начала. Завершается рисование линии щелчком правой кнопкой мыши или двойным щелчком левой кнопкой мыши в точке окон-

чания линии. Линия изменит красный цвет на черный, и на ее конце появится квадратный маркер (пример 24.9).

Создание кривой линии описано в примере 24.10.

П. Безье разработал метод вычерчивания кривых в 1960 г., работая в автомобилестроительной фирме «Рено». Одновременно эта же разработка возникла в ходе исследования П. де Кастельжо из компании «Ситроен».

24.2. Редактирование фигур

Инструментом «Выделить и трансформировать» ( ) выделяют фигуры (пример 24.11). Выделенные фигуры можно: удалять клавишей Delete; перемещать в рабочей области (с помощью Ctrl строго по горизонтали или вертикали); масштабировать (Ctrl позволяет сохранить пропорции); отражать горизонтально (клавиша H) и вертикально (клавиша V); поворачивать; наклонять.

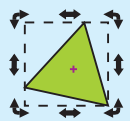
После выделения вокруг фигуры появляются восемь двунаправленных стрелок, и объект можно удалять, перемещать и масштабировать.

Группа фигур выделяется при нажатой клавише Shift. Щелчок по выделенному в группе объекту исключает его из выделения. Esc снимает всякое выделение. Ctrl + A выделяет все объекты.

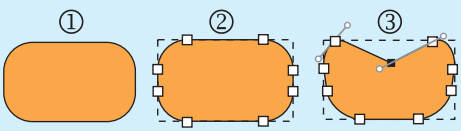
Форму объекта меняют с помощью инструмента «Редактировать узлы контура»: (пример 24.12).

Пример 24.11. Использование инструмента «Выделить и трансформировать».

Если выполнить щелчок по уже выделенной фигуре, вид стрелок изменится:



Пример 24.12. Использование инструмента «Редактировать узлы контура».

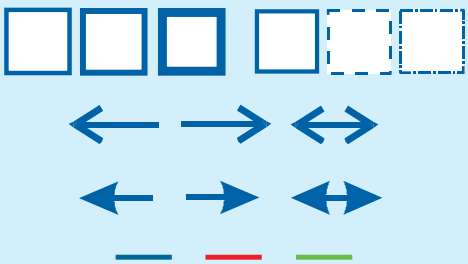


1. Нарисовать графический примитив.

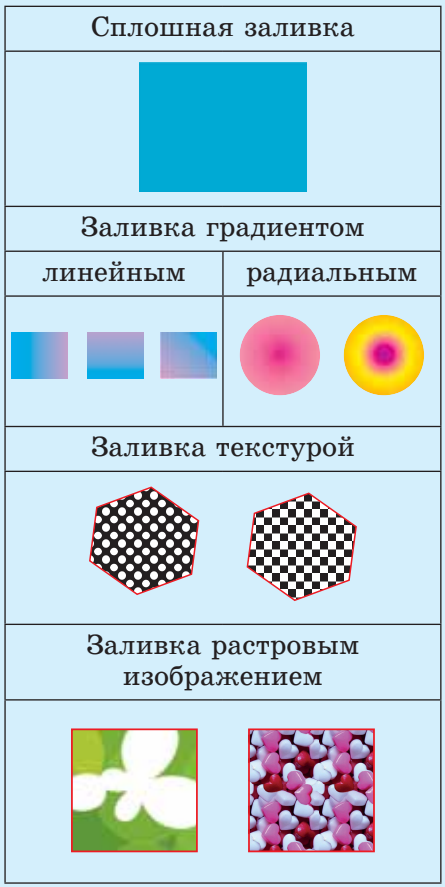
2. Выбрать инструмент и настройку (преобразовать в контур).

3. Перемещая узлы, изменить контур. При необходимости добавить ( ) или удалить ( ) узлы.

Пример 24.13. Стили обводки фигур и стрелки.



Пример 24.14. Способы заливки.



24.3. Обводка и заливка

Обводка — линия, ограничивающая фигуру.

Для обводки можно определять:

1) толщину;

2) стиль (сплошной, пунктирный, штрихпунктирный);

3) стрелки (маркеры);

4) цвет.

(Рассмотрите пример 24.13.)

Заливкой называется закраска внутренней области фигуры.

Заливка фигуры может быть сделана одним из нижеперечисленных способов:

1) сплошным цветом;

2) градиентом;

3) текстурой (выбирается из набора декоративных заливок, в которых используются заранее созданные авторами программы заготовки);

4) растровым изображением из файла.

(Рассмотрите пример 24.14.)

Цветовой градиент — плавный переход одного цвета в другой.

В редакторе Inkscape изменить обводку и заливку фигур мож-

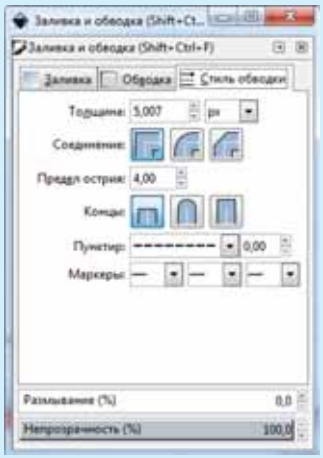
но в диалоговом окне **Заливка и обводка** или с помощью палитры. Окно открывается по команде главного меню редактора **Объект → Заливка и обводка** (пример 24.15). В этом окне можно выбрать цветовую модель и настроить цвет и его прозрачность.

В палитре цвет заливки изменяется при помощи щелчка мыши по нужному цвету. Для изменения цвета обводки щелчок следует производить при нажатой клавише Shift.

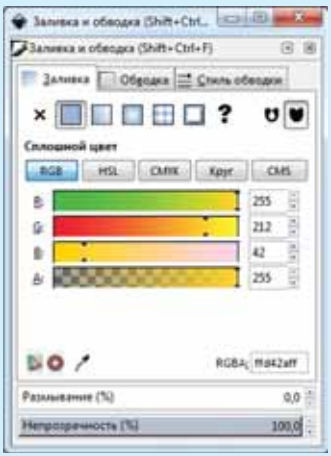
Объекту можно назначить отсутствие обводки и заливки с помощью красного крестика в левой части палитры. Градиентную заливку можно применять и к обводке.

При переходе от сплошной заливки к градиенту создается настройка градиента. Только что созданная настройка градиента использует предыдущий цвет сплошной заливки фигуры, который переходит из непрозрачного цвета в прозрачный.

Пример 24.15. Окно «Заливка и обводка».



Изменение стиля обводки



Изменение заливки

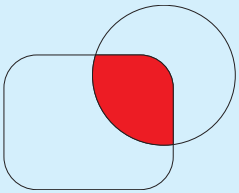
24.4. Работа с цветом

Задавать цвет в окне **Заливка и обводка** удобно, меняя цифровые значения основных цветов (пример 24.16).

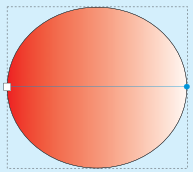
Пример 24.16. Настройка цвета в цветовой модели RGB:

Цифровые значения	Цвет
R = 115, G = 200, B = 155	

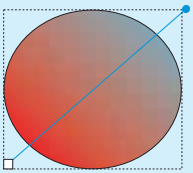
Пример 24.17. Использование инструмента «Заливка».



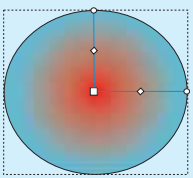
Пример 24.18. Использование инструмента «Градиент».



Для линейного градиента все опорные точки расположены на одной прямой.



Опорные точки могут располагаться и за пределами объекта.



Радиальный градиент имеет центральную опорную точку, которая является начальной. Из нее выходят две линии градиента.

Для заливки замкнутой области используется инструмент «Заливка»:  (пример 24.17).

При рисовании часто возникает необходимость выбора цвета на участках холста. Для этого используется инструмент **«Пипетка»**:  .

Для создания градиентных заливок предназначен инструмент **«Градиент»**:  . Цвета градиента настраиваются при помощи опорных точек (пример 24.18). Количество точек соответствует количеству цветов в градиенте. Опорные точки можно добавлять или удалять, предварительно выбрав  или  на панели инструмента. Добавляется опорная точка двойным щелчком мыши по линии градиента. Удаляется выделенная опорная точка с помощью клавиши Delete.

Состояние опорной точки при определенном цвете:

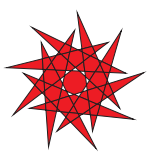
- белый — невыделенная (неактивная);
- **синий** — выделенная (активная, текущая, выбранная);
- **красный** — наведенная (под указателем мыши) или изменяемая (если производится действие).

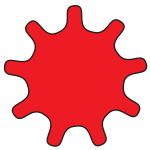
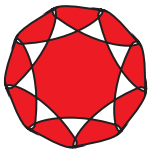
1. С помощью какой клавиши можно нарисовать правильную фигуру?
2. Как сгладить углы прямоугольника в редакторе Inkscape?
3. Какие фигуры можно нарисовать с помощью инструмента  ?
4. Для чего служит инструмент  ?
5. Какой инструмент позволяет редактировать узлы фигуры?
6. Что называется обводкой векторного изображения?
7. В чем состоит процесс заливки фигуры?
8. Что такое цветовой градиент?
9. Как настроить градиент в редакторе Inkscape?

Упражнения

1 Нарисуйте несколько разных фигур и примените к ним масштабирование, отражение, поворот, наклон.

2 Создайте изображения звезд.

№	Изображение	Приемы создания и редактирования
1		Инструмент  
2		Выполнить п. 1 и изменить форму, перемещая верхний узел звезды к центру
3		Выполнить п. 1 и изменить форму, перемещая базовый узел звезды вправо
4		Выполнить п. 1 и изменить форму, перемещая базовый узел звезды влево и вверх

5		Выполнить п. 1 и изменить форму, перемещая верхний узел звезды при нажатой клавише Alt
6		Выполнить п. 1 и изменить форму, перемещая верхний узел звезды вправо при нажатой клавише Shift
7		Выполнить п. 1 и изменить форму, перемещая верхний узел звезды влево при нажатой клавише Shift

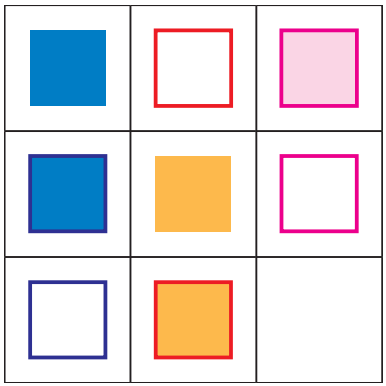
3 Нарисуйте пять фигур: прямоугольник, квадрат, овал, круг и правильный шестиугольник. Примените к каждой фигуре соответствующие заливки: сплошная (с прозрачностью и без), градиент (линейный и радиальный), с использованием текстуры.

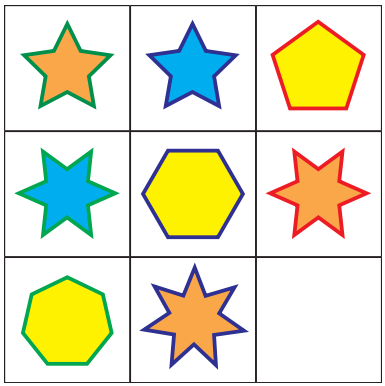
4 Выполните пример:

Нарисуйте малиновый квадрат без обводки. На вкладке **Заливка** установите R = 255, G = 0, B = 225, A = 255, размывание — 0, непрозрачность — 55. На вкладке **Обводка** — без обводки (×).

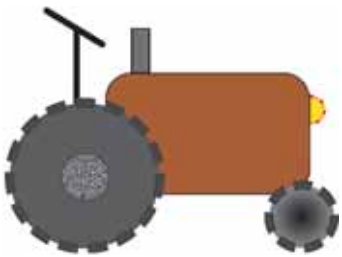
Нарисуйте фиолетовый прямоугольник с черной обводкой так, чтобы он перекрывал часть квадрата. На вкладке **Заливка** установите R = 100, G = 0, B = 105, A = 255, размывание — 0, непрозрачность — 100. На вкладке **Стиль обводки** — толщина 15 px.

5 Откройте файл и дополните изображения недостающими фигурами.





6 Создайте изображение:

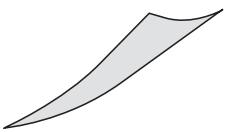


В этом вам поможет таблица.

Элемент изображения	Инструмент и настройки	Обводка	Заливка
		Сплошная, цвет черный	Однородная, цвет темно-серый
	 переключиться на 	Штрих-пунктирная, цвет красный, толщина 4 px	Однородная, цвет желтый (FFFF00)

Элемент изображения	Инструмент и настройки	Обводка	Заливка
		Сплошная, цвет черный	Однородная, цвет коричневый (A05A2C)
		Пунктирная, цвет черный, толщина 15 px	Градиент радиальный (переход от черного к серому)
	 (режим — прямые линии)	Сплошная, цвет черный	Нет
		Сплошная, цвет черный, концы круглые	
		Пунктирная, цвет черный, толщина 15 px	Однородная, цвет серый (4D4D4D)
		Сплошная, цвет черный	Текстура «песок»

7 Создайте изображение:

Элемент изображения	Инструмент и настройки	Обводка	Заливка
		Сплошная, цвет серый, толщина 3 px	Линейный градиент, два цвета
			Линейный градиент, три цвета
			Однородная, цвет черный
	Для треугольника:   Для линии:  (режим — прямые линии)	Для треугольника: сплошная, цвет серый, толщина 3 px	Для треугольника: однородная, цвет 918A6F
	  Изменить форму: 	Сплошная, цвет черный, толщина 3 px	Однородная, цвет серый
	  Изменить положение маркеров узлов	Сплошная, цвет темно-серый, толщина 7 px	Однородная, цвет серый

§ 25. Операции над объектами векторного изображения

25.1. Копирование, выравнивание и взаимное расположение объектов

Создать копию объекта в редакторе Inkscape можно, как и в других программах, воспользовавшись буфером обмена. В этом случае копия появится рядом с оригиналом. В редакторе Inkscape есть дополнительные операции копирования объектов — дублирование и клонирование (пример 25.1).

Чтобы создать дубль объекта, нужно выбрать пункт меню **Правка** → **Продублировать** или нажать  на панели инструментов. Созданная копия помещается поверх оригинала. Как видно из примера 25.1, **дубль** является самостоятельным объектом. При изменении оригинала он не меняется.

Клон создается по команде **Правка** → **Клоны** → **Создать клон** или с помощью кнопки  на панели инструментов. При создании клон, как и дубль, помещается над оригиналом. **Клоны** изменяются вместе с изменением оригинала.

Пример 25.1. Дублирование и клонирование фигур.

Оригинал Дубль Клон

После редактирования оригинала получим:

Оригинал Дубль Клон

Пример 25.2. Создание дубля и клона.

Для создания дублей и клонов можно использовать инструмент «Распылитель»:  . Этот инструмент рисует объектами из буфера обмена. Распыление выполняется в трех режимах:

 — создает дубли объектов;

 — создает клоны объектов;

 — все распыленные объекты объединяются в один объект.

Еще один способ создания дубля или клона, помимо вышеперечисленных, приведен в примере 25.2.

Для размещения определенным образом относительно друг друга объектов векторного изображения используются операции выравнивания.

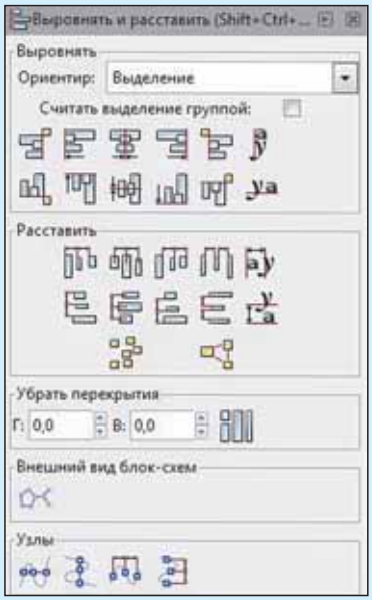
Выравниванием называется такое размещение всех выделенных объектов, при котором определенные точки объектов располагаются на одной прямой.

Выравнивание объектов можно осуществить с помощью окна **Выровнять и расставить.** Окно можно открыть, выполнив команду **Объект** → **Выровнять и расставить** или нажав на кнопку **Выровнять и расставить объекты** () на панели инструментов (пример 25.3).

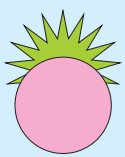
На панели **Выровнять** в первой строке выбирается способ выравнивания фигур по вертикали, а во второй — по горизонтали (пример 25.4).

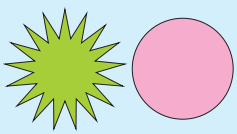
На панели **Расставить** определяется способ распределения объектов на некотором расстоянии друг от друга. В первой строке указывается распределение объектов по вертикали, а во

Пример 25.3. Окно **Выровнять и расставить.**



Пример 25.4. Выравнивание фигур.



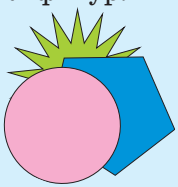


Пример 25.5. Распределение линий.

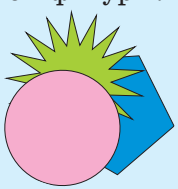
Расположение линий до расстановки

Расстояния между линиями выровнены по горизонтали

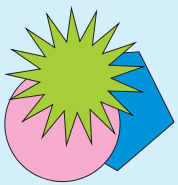
Пример 25.6. Порядок расположения фигур.



Изменяем положение салатовой фигуры:



Поднять



Поднять на передний план

Группировка предотвращает случайные изменения положения объекта относительно других объектов. При группировке можно создавать вложенные группы, группируя существующие группы.

Операции объединения и пересечения объектов позволяют создавать различные неправильные формы.

второй — по горизонтали (пример 25.5).

Для изменения **порядка расположения** объектов их необходимо выделить. Затем воспользоваться командами меню **Объект:**

Команды **Поднять на передний план** и **Опустить на задний план** поставят выделенные объекты на самую верхнюю или самую нижнюю позицию. Две других команды — **Поднять** и **Опустить** — опустят или поднимут выделенные объекты на один уровень относительно ближайшего невыделенного объекта (пример 25.6).

25.2. Группировка. Объединение и пересечение объектов

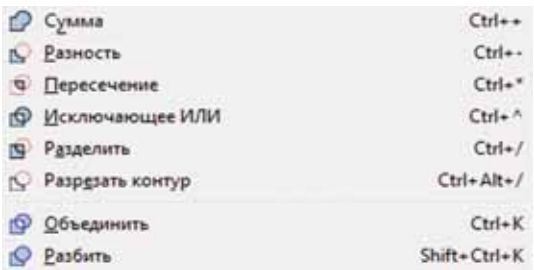
В Inkscape несколько объектов можно объединить в группу. При перемещении и трансформации группа ведет себя как один объект.

Чтобы сгруппировать несколько объектов, нужно выделить их все и выбрать в меню **Объект** → → **Сгруппировать** или нажать Ctrl + G. Чтобы разгруппировать

одну или несколько групп, нужно выбрать в меню **Объект** → → **Разгруппировать** или выполнить двойной щелчок мышью по группе. Сами группы также можно объединять в группы. Двойной щелчок отменяет только группирование верхнего уровня.

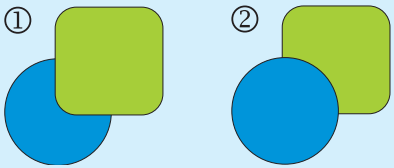
Не обязательно разбивать группу, если нужно отредактировать один объект из нее. Достаточно щелкнуть по объекту с нажатой Ctrl (или Shift + Ctrl, если нужно отобрать несколько объектов), и можно будет работать с объектом в группе отдельно.

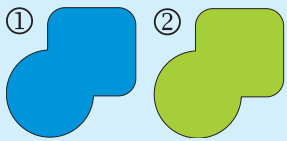
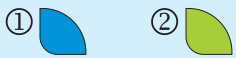
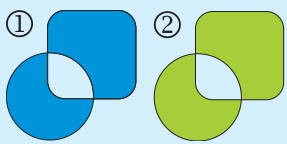
При создании изображений часто возникает необходимость выполнить над объектами логические операции объединения и пересечения. В Inkscape эти операции выполняются через меню **Контур**:



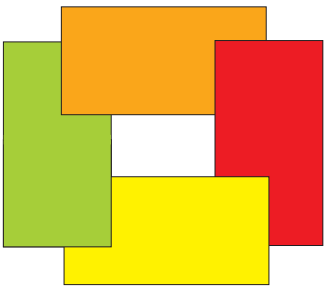
В примере 25.7 показан результат применения некоторых операций объединения для двух объектов, расположенных в разном порядке.

Пример 25.7. Применение операций объединения.



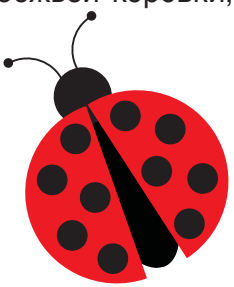
Операция	Результат операции
Сумма	
Разность	
Пересечение	
Исключающее ИЛИ	
Разделить	

1. Какие операции над объектами векторного изображения можно выполнить в редакторе Inkscape?
2. Что произойдет, если к объектам применить сначала выравнивание по вертикальной оси, а потом по горизонтальной оси?
3. Что происходит при группировке объектов?
4. Важно ли расположение объектов при выполнении операций объединения?
5*. Можно ли, меняя лишь взаимное расположение объектов, создать следующий рисунок?



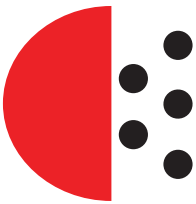
Упражнения

1 Создайте изображение божьей коровки, как показано на рисунке:



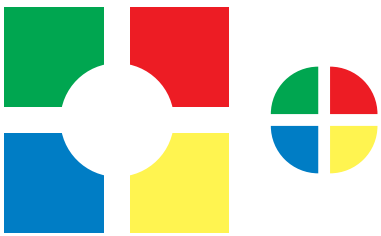
Для этого подготовьте элементы рисунка, используя инструменты **Эллипс** и **Кривая Безье**:

Примените к созданным элементам операции, указанные в таблице.

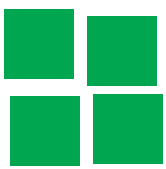
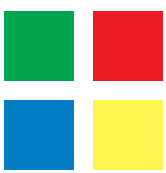
Исходные объекты	Операции	Результат
	Продублировать объект и уменьшить размер дубля. Создать 4 дубля измененного объекта	
	Разместить черные круги в красном полукруге и сгруппировать объекты	
	Продублировать объект и уменьшить размер дубля	
	Соединить черный круг с дугой и сгруппировать объекты	
	Продублировать объект и отразить дубль по горизонтали. Выполнить повороты	
		
	Продублировать объект и изменить размеры и пропорции дубля. Выполнить поворот	

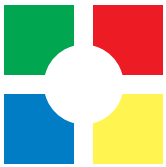
2 Используя рисунок из задания 1, создайте изображение:

3* Создайте изображение, применив операции над объектами:



Для этого создайте элемент рисунка, используя инструмент **Прямоугольник**:  . Примените к созданным элементам операции, указанные в таблице.

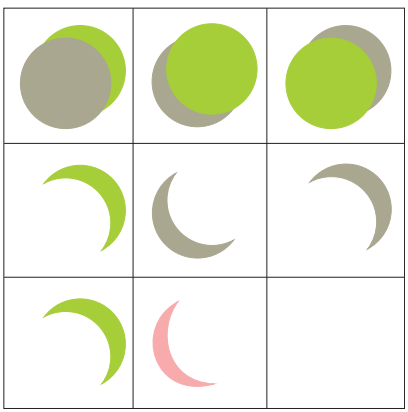
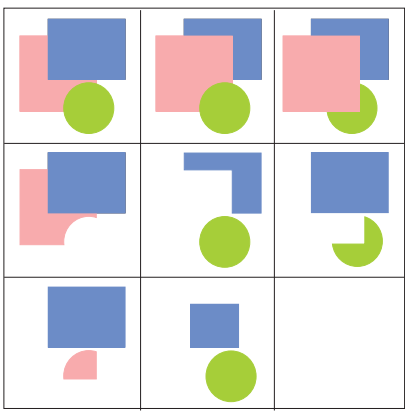
Операции	Результат
Дублирование + перемещение	
Выравнивание + изменение цвета	

Операции	Результат
Нарисовать белый круг, выровнять по центру и создать три его дубля	
Выделяя попарно один квадрат и круг, применить операцию пересечения	

4 Нарисуйте одно из изображений:



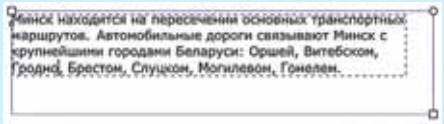

5* Откройте файл и дополните изображения недостающими фигурами:

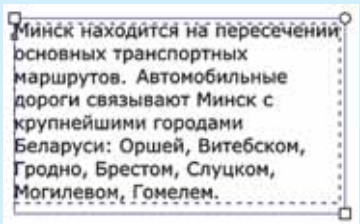



6 Используя изображения из задания 4, придумайте и нарисуйте сюжетный рисунок.

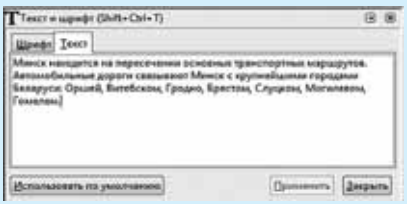
§ 26. Работа с текстом

Пример 26.1. Создание простого текста.





Вводить текст удобно в текстовом редакторе. Его вызов выполняется по команде **Текст** → → **Текст и шрифт.** Для ввода текста необходимо перейти на вкладку **Текст:**



Текст можно вставить через буфер обмена из другого текстового редактора (Блокнота и др.).

Простой текст можно преобразовать в художественный: **Текст → Преобразовать в текст.** Перед преобразованием текст должен быть выделен. Если после этого выделить текст, то текстовая рамка с узлами исчезнет. Это уже художественный текст.

Для создания текста используется инструмент «Текст»:  . В векторных редакторах существуют две возможности работы с текстом — простой текст и художественный текст.

Простой текст используется для отображения на рисунках больших текстовых фрагментов и может разделяться на абзацы. Простой текст можно форматировать точно так же, как и в обычном текстовом редакторе.

Для создания простого текста необходимо:

1) выбрать инструмент  ;

2) установить параметры текста на контекстной панели управления;

3) обрисовать с помощью мыши текстовую рамку и ввести в нее текст.

Текстовая рамка представляет собой сплошную прямоугольную рамку. Текст в ней ограничивается пунктирной рамкой. При помощи маркера в правом нижнем углу можно менять размеры текстовой рамки. При этом текст подгоняется под размеры рамки (пример 26.1).

Художественный текст используется для создания худо-

жественных надписей. Созданные надписи являются графическими объектами. К ним можно применять различные эффекты, например изменить форму букв, создать тень, расположить текст по нужной траектории. Для создания художественного текста необходимо щелкнуть мышью в любой точке холста. После этого в данной точке начнет мигать курсор, приглашая к вводу текста. Переход на новую строку происходит по нажатию клавиши Enter.

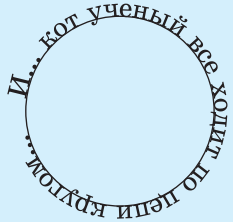
При любом способе создания текста создается текстовый объект, который можно перемещать, поворачивать и т. д. с помощью инструмента  .

Для редактирования текстового объекта необходимо выбрать инструмент  и щелкнуть по объекту.

Много интересных возможностей работы с художественным текстом предлагает меню **Текст.** Рассмотрим некоторые из них:

1. **Разместить по контуру** (пример 26.2). Перед применением этой команды нужно выделить текст и контур.

Пример 26.2. Расположение текста[1] по окружности.

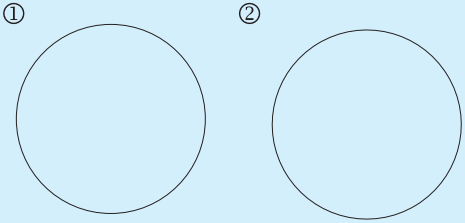


После связывания текста и контура можно редактировать каждый из этих объектов, не нарушая связи между ними:

1. Приподнять текст над контуром.

2. Повернуть контур (при этом текст поворачивается вместе с контуром).

① ②



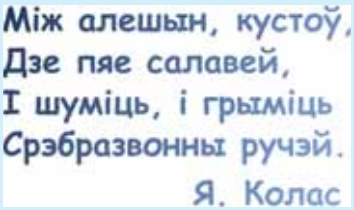
3. Убрать контур (сделать прозрачным).

③

[1] Текст в примере 26.2 цитируется по: Пушкин, А. С. Руслан и Людмила : поэма. — М. : Изд. Дом «Прибой». — 1996. — С. 5.

Пример 26.3. Расположение текста внутри контура.

Пример 26.4. Применение градиента к тексту[1].



Пример 26.5. Применение эффекта к тексту.

2. **Заверстать в блок**, т. е. поместить в контур (пример 26.3). Порядок работы с блоками такой же, как и с контурами.

Текст можно раскрасить в разные цвета (целиком или частично). Также к тексту можно применить градиент (пример 26.4). При этом в дальнейшем текст можно редактировать обычным образом. Кроме того, буквы текста можно превратить в контур, выполнив команду **Контур** → → **Оконтурить объект.** После этого текст уже нельзя редактировать, но к его буквам можно применять разнообразные эффекты. В примере 26.5 использована команда **Расширения** → **Изменения контура** → **Дрожание контуров.**

?

1. Какие возможности работы с текстом существуют в редакторе Inkscape?
2. Какие действия можно выполнять с текстовым объектом?
3. Можно ли редактировать художественный текст?
4. Каким образом можно превратить буквы текста в контуры?
5. Можно ли редактировать текст после превращения его букв в контуры?
6. Какие действия необходимо выполнить перед применением к тексту эффектов?

[1] Цитируется по: Колас, Я. Ручэй // Лазарук, М. А., Логінава, Т. У. Беларуская літаратура : вучэб. дапам. для 7 кл. агульнаадукацыйных устаноў з беларус. і рус. мовай навучання. — Мінск : «Нацыянальны інстытут адукацыі». — 2010. — С. 28.

Упражнения

1 Нарисуйте открытку. Оформите ее художественным текстом, размещенным по контуру.

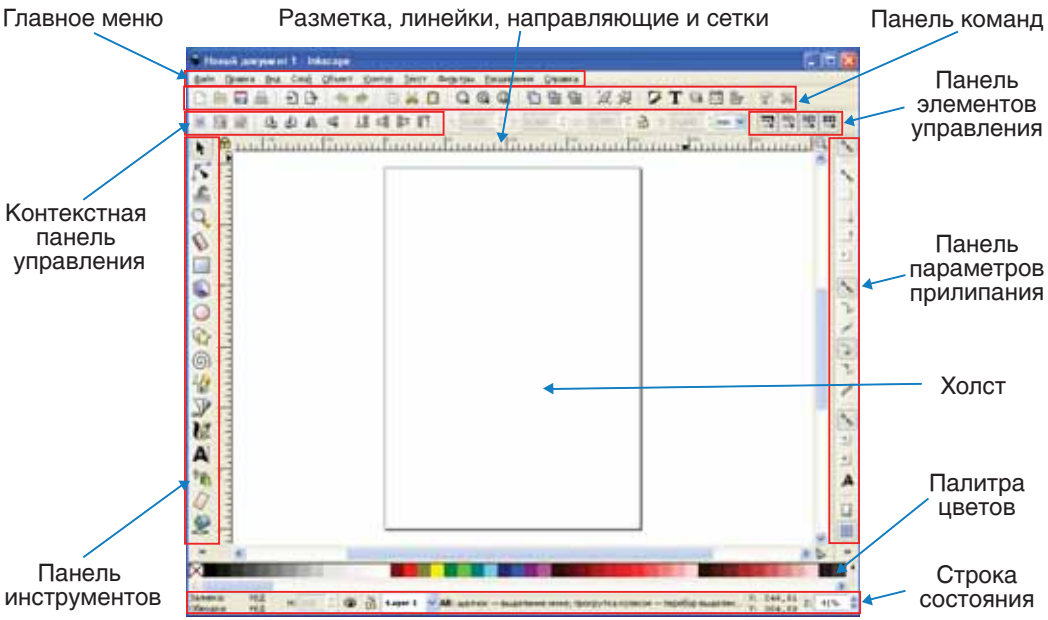
2 Нарисуйте плакат. Примените различные виды заливок и вставьте простой текст.

3 Придумайте и нарисуйте плакат на одну из тем:

1. «Мойте руки перед едой».
2. «Мое счастливое детство».
3. «Мы в ответе за тех, кого приручили».
4. «Берегите электроэнергию».
5. «Правила поведения в компьютерном классе».

ПРИЛОЖЕНИЕ

Окно интерфейса Inkscape можно разделить на десять основных областей:



Главное меню

Содержит основные функции работы с программой: работа с файлами; функции редактирования и просмотра; функции редактора работы с текстом, фильтрами, объектами и контурами. Включает дополнения и справочную информацию.

Панель команд

Содержит значки-иконки, которые вызывают определенные команды. Эти команды также доступны в главном меню или по комбинации клавиш. Панель инструментов предназначена для

легкого доступа к наиболее используемым функциям редактора. Из нее в один клик можно открыть новый или существующий документ, напечатать его, загрузить изображение, отменить предыдущие команды, масштабировать, открыть диалоговое окно для настройки свойств документа и т. д. Каждый значок при наведении курсора мыши отображает свою функцию с помощью всплывающих подсказок.

Если все значки панели инструментов не помещаются на экране, то доступ к ним может быть осуществлен через кнопку с двумя стрелочками внизу панели. В результате щелчка по этой кнопке в виде меню появятся все остальные команды панели, значки которых не помещаются на ней.

Панель инструментов

Состоит из вертикального ряда кнопок, расположенного в левой части окна редактора. Это основной элемент для работы в векторном редакторе Inkscape. Он содержит набор графических инструментов для создания и редактирования фигур.

В окне инструментов присутствуют:

- инструменты, предназначенные для работы с геометрическими фигурами;
- инструменты, предназначенные для свободной трансформации фигур и линий;
- инструменты, предназначенные для работы с текстом;
- инструменты, предназначенные для работы с цветом (заливка и градиенты).

Контекстная панель управления

В зависимости от того, какой инструмент выбран в окне инструментов, вид контекстной панели изменяется. В ней отображаются настройки и параметры активного инструмента. В за-

висимости от ситуации изменение этих параметров может сразу повлиять на выбранный объект. При использовании инструментов  ,  и  вернуться к исходным параметрам можно с помощью кнопки  контекстной панели этих инструментов.

Холст

Холст, или канва, является главной рабочей областью программы. Это основная часть интерфейса, поскольку именно здесь пользователь создает и редактирует рисунки. Расположен холст посередине окна программы и похож на изображение чистого белого листа бумаги.

Несмотря на то что границы отображаемой на холсте страницы определяют границы изображения для печати или сохранения, при рисовании размер страницы вовсе не ограничивает область для изображения. Вы можете сделать границы страницы и тени этих границ невидимыми. Настроить видимость границ страницы можно в свойствах документа.

Разметка, линейки, направляющие и сетки

Линейка разметки расположена вверху и слева от холста. Деления линейки разметки определяют расстояния в некоторых единицах (по умолчанию в пикселях). Изменить настройку единиц измерения можно в свойствах документа (**Файл → Свойства документа**).

Когда указатель мыши находится над холстом, на линейке появляются два черных треугольника, которые отображают координаты курсора относительно нижнего левого угла страницы. Эти координаты X и Y можно увидеть в строке состояния (в нижнем правом углу окна программы), рядом с параметром масштаба Z.

Комбинация клавиш Ctrl + R позволяет скрыть или отобразить линейки разметки. Также это можно сделать в главном меню **Вид → Показать или скрыть → Линейки.**

Направляющие создаются в Inkscape пользователем для облегчения рисования или построения фигур. Направляющие позволяют установить положение некоторых инструментов точно по ним. Применение направляющих облегчает пользователям выравнивание объектов, создаваемых с помощью мыши. Чтобы установить направляющие, щелкните указателем мыши на горизонтальной или вертикальной линейке и, удерживая кнопку мыши нажатой, перетащите появившуюся направляющую в ту точку холста, где она должна быть, после этого отпустите кнопку мыши. С помощью горизонтальной линейки создаются горизонтальные направляющие, с помощью вертикальной — вертикальные.

Сетка может оказаться полезной, чтобы не использовать большое количество направляющих. Отобразить сетку можно с помощью главного меню **Вид → Сетка.**

Панель параметров прилипания

Панель параметров прилипания позволяет легко настроить параметры прилипания объекта. Функции этой панели удобны для правильного и точного размещения объектов. Панель параметров прилипания расположена вертикально по правому краю рабочей области окна.

Палитра цветов

Обеспечивает быстрый доступ к цветам, она же позволяет назначить цвета к фигурам. Отображается в нижней части окна программы или может быть открыта в виде отдельного окна (**Вид →** → **Образцы цветов** или комбинация клавиш Shift + Ctrl + W).

Строка состояния

Находится в самом низу окна. Строка состояния отображает (слева направо):

- цвет заливки и обводки объекта;
- возможность быстрой работы со слоями и переключения между ними;
- область сообщений;
- индикатор координат указателя мыши;
- управление масштабом.

Учебное издание
Котов Владимир Михайлович
Лапо Анжелика Ивановна
Войтехович Елена Николаевна

ИНФОРМАТИКА

Учебное пособие для 7 класса
учреждений общего среднего образования
с русским языком обучения

Зав. редакцией *Г. А. Бабаева*. Редактор *Е. И. Даниленко*. Художественные редакторы *А. Н. Богушевич, А. И. Резанович*. Обложка *Н. В. Кузьменковой*. Техническое редактирование и компьютерная верстка *И. И. Дубровской*. Корректоры *В. С. Бабеня, О. С. Козицкая, Е. П. Тхир, А. В. Алешко*.

Подписано в печать 27.10.2017. Формат $70\times90\,{}^{1}/_{16}$. Бумага офсетная. Гарнитура школьная. Печать офсетная. Усл. печ. л. 12,87. Уч.-изд. л. 8,8. Тираж 114 500 экз. Заказ .

Издательское республиканское унитарное предприятие «Народная асвета»
Министерства информации Республики Беларусь.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий 1/2 от 08.07.2013.
Пр. Победителей, 11, 220004, Минск, Республика Беларусь.

ОАО «Полиграфкомбинат им. Я. Коласа». Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 2/3 от 04.10.2013. Ул. Корженевского, 20, 220024, Минск, Республика Беларусь.

(Название и номер учреждения образования)

Учебный год	Имя и фамилия учащегося	Состояние учебного пособия при получении	Оценка учащемуся за пользование учебным пособием
20 /			
20 /			
20 /			
20 /			
20 /			
20 /			