

Федеральный государственный образовательный стандарт
Образовательная система «Школа 2100»

ИНФОРМАТИКА

УЧЕБНИК • 7 класс • Книга 2



БАЛЛАСС

Федеральный государственный образовательный стандарт
Образовательная система «Школа 2100»

А.В. Горячев, Л.А. Макарина, А.В. Павлоцкий,
Н.С. Платонова

ИНФОРМАТИКА

УЧЕБНИК • 7 класс • Книга 2



Рекомендовано Министерством образования и науки
Российской Федерации

Москва
БАЛАСС
2012

УДК 373.167.1:004+004(075.3)

ББК 32.81я721

Г67

**Федеральный государственный образовательный стандарт
Образовательная система «Школа 2100»**

Совет координаторов предметных линий Образовательной системы «Школа 2100» –
лауреат премии Правительства РФ 2008 года в области образования
за теоретическую разработку основ образовательной системы нового поколения
и её практическую реализацию в учебниках

На учебник получены положительные заключения Российской академии наук (от 14.10.2011)
№ 10106-5215/447 и Российской академии образования (от 24.10.2011) № 01-5/7д-125

Руководитель издательской программы –
доктор пед. наук, проф., чл.-корр. РАО Р.Н. Бунеев

Авторский коллектив:

А.В. Гиглавый – научный редактор, А.В. Горячев – автор концепции курса, научный руководитель, Л.А. Макарина (книга 1: модуль «Общение в сети Интернет»), А.В. Павлоцкий (книга 1: модуль «Укрощение компьютера»; книга 2: модуль «Алгоритмизация и программирование»), Н.С. Платонова (книга 1: модули «Создание документов и печатных изданий», «Создание мультимедийной продукции»; книга 2: модули «Основы издательских технологий», «Основы дизайна и печати изображений»).

Г67

Горячев, А.В.

Информатика. 7 кл. : учеб. для общеобразоват. учреждений : в 2-х кн. Кн. 2 /
А.В. Горячев, Л.А. Макарина, А.В. Павлоцкий, Н.С. Платонова. – М. : Баласс, 2012. – 144 с., ил. (Образовательная система «Школа 2100»).

ISBN 978-5-85939-938-3 (кн. 2)

ISBN 978-5-85939-981-9

Учебник предназначен для учащихся 7-го класса общеобразовательных учреждений. Соответствует Федеральному государственному образовательному стандарту основного общего образования, является продолжением непрерывного курса информатики и составной частью комплекта учебников развивающей Образовательной системы «Школа 2100».

Содержание учебника представлено в виде отдельных учебных модулей, из которых учитель может выбрать нужные в соответствии с требованиями основной образовательной программы школы. Учебный материал предлагается на необходимом и повышенном уровне.

УДК 373.167.1:004+004(075.3)

ББК 32.81я721

Данный учебник в целом и никакая его часть не могут быть
скопированы без разрешения владельца авторских прав

ISBN 978-5-85939-938-3 (кн. 2)

ISBN 978-5-85939-981-9

© А.В. Горячев, Л.А. Макарина,
А.В. Павлоцкий, Н.С. Платонова, 2012
© ООО «Баласс», 2012

Дорогие ребята!

На многих уроках, таких, как биология, физика или география, мы изучаем мир вокруг нас. Уроки информатики появились в школе после того, как возник ещё один мир, но уже созданный руками человека, – мир компьютерной техники. Поэтому на уроках в школе присутствуют как пользователи компьютеров, так и будущие разработчики – те, для кого создание и развитие мира компьютерной техники и компьютерных программ станет профессией.

Современный разработчик компьютерной техники или компьютерных программ – это человек образованный, хорошо ориентирующийся в теории и практике технологий создания и развития компьютерной техники и компьютерных программ. Он может создавать новые компьютерные инструменты для пользователей, помогать им в настройке существующих программ под их потребности.

В образование современного разработчика компьютерной техники и компьютерных программ входит и получение теоретических знаний науки о компьютерах. Изучение теории не стоит откладывать на этап обучения конкретной профессии, а лучше начать ещё в школе.

В книге 2 наших учебников для 7–9-го классов мы разместили учебные модули, с помощью которых вы сможете изучить теоретические основы информатики, сквозную линию модулей с 7-го по 9-й класс, нацеленную на обучение программированию, а также модули профессиональной ориентации, с помощью которых можно научиться основам профессий, опирающихся на применение компьютеров.

Так, в книге 2 учебника для 7-го класса, которую вы держите в руках, в модуле «Алгоритмизация и программирование» вы сможете освоить первые умения по составлению алгоритмов и написанию компьютерных программ. Кроме того, в учебнике есть два модуля профессиональной ориентации – «Основы дизайна и печати изображений» и «Основы издательских технологий».

Как работать с учебником

Пролистайте учебник. Вы заметите, что он разделён на модули. Вы будете изучать модули в том порядке, который предложит учитель.

Практически в каждом модуле мы предусмотрели пять основных параграфов. Изучив эти параграфы, вы напишете проверочную работу, по итогам которой узнаете, как вы освоили новый материал: не освоили на необходимом уровне либо освоили на необходимом или повышенном уровне. Далее вы будете работать самостоятельно, выполняя по указанию учителя задания того уровня, которого вы пока не достигли. Если проверочная работа покажет, что вы освоили и повышенный уровень, то вы будете выполнять задания самого высокого – максимального уровня. Учитель в любой момент может предложить вам перейти на выполнение заданий более высокого уровня. По окончании выполнения заданий учитель проведёт итоговую проверочную работу.

Далее в модуле расположены дополнительные параграфы, задания к ним и проверочные работы. Основные параграфы выделены в учебнике зелёной полосой вверху страницы, дополнительные – розовой полосой.

Дополнительные параграфы, которые вы не изучите на уроках, вы сможете использовать на факультативах и кружках.

На уроках информатики вы сможете освоить умения, которые помогут вам более эффективно использовать компьютеры и компьютерные сети для решения возникающих в вашей жизни задач. Кроме того, учитель может решить, что вам надо освоить умения, которые помогут вам заниматься разработкой новых компьютерных программ или заложат основы профессиональной деятельности, тесно связанной с применением компьютерной техники.

Кроме того, наш учебник, как и все учебники Образовательной системы «Школа 2100», поможет вам в развитии универсальных учебных умений (действий), будет учить вас учиться. В учебниках Образовательной системы «Школа 2100» вам могут встретиться задания, обозначенные кружками и фоном разного цвета – это условные знаки. Каждый цвет соответствует определённой группе умений:

- – организовывать свои действия: ставить цель, планировать работу, действовать по плану, оценивать результат;
- – работать с информацией: самостоятельно находить, осмысливать и использовать её;
- – общаться и взаимодействовать с другими людьми, владеть устной и письменной речью, понимать других, договариваться, сотрудничать;
- – развивать качества своей личности, оценивать свои и чужие слова и поступки;



так обозначены задания, где нужно применить разные группы умений, мы называем их жизненными задачами и проектами.

Для успешного изучения информатики и овладения универсальными умениями на уроках используется образовательная технология «проблемный диалог». Поэтому структура параграфа, где вводится новый материал, имеет в учебнике следующий вид.

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА

Это подведение к теме (вопросу, цели) урока: вы обсуждаете проблему в предложенном материале и формулируете главный вопрос урока (всем классом, в группе или в паре). Сравните свой вариант вопроса с авторским. Авторские вопросы к параграфам расположены в «Содержании» под названиями параграфов и выделены курсивом.

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Так обозначены вопросы и задания по изученному материалу, который вам необходим для открытия нового знания.

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Вы в группе, в паре или совместно с учителем, ведя диалог, осуществляете поиск решения проблемы. Для решения проблемы вы работаете с текстом.

ФОРМУЛИРУЕМ ВЫВОД

Здесь вы формулируете вывод и проверяете свои предположения, сравнивая их с авторским решением проблемы – научными формулировками правил или определений.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

Так обозначены задания на применение новых знаний.

Задания, отмеченные «звездочкой», имеют повышенную сложность.

ОПЕРАЦИИ

Раздел «Операции» позволит вам научиться выполнять действия с компьютерными программами, необходимые для решения ваших задач.

В конце модуля вы найдёте раздел «Решаем жизненные задачи и работаем над проектами». Задачи и проекты могут выполняться как на уроках, так и на факультативах и кружках.

Там же находится очень важный раздел «О профессиях». Прочитайте его и подумайте, какие профессии вам больше по душе.

Что такое жизненные задачи?

Это проблемы, с которыми вы можете столкнуться в жизни, и для решения которых вам понадобятся разные знания и умения. Они оформлены следующим образом:

Название задачи

Ваша роль: человек, в роли которого вы должны себя представить, решая проблему.

Описание. Условия, в которых возникла проблема.

Задание. То, что нужно сделать и получить в итоге.

Что такое проект?

Это любое самостоятельное дело, которое предполагает:

- 1) оригинальный замысел (цель);
- 2) выполнение работы за определённый отрезок времени;
- 3) конкретный результат, представленный в итоге.

Что можно считать результатом проекта?

- Предметы, сделанные своими руками: макеты, модели или вещи для практического использования.
- Мероприятия: спектакли, фотовыставки, викторины, конференции, праздники и тому подобное – при условии, что они подготовлены самими учениками.
- Информационные продукты: газеты, книжки, плакаты, карты, стихотворения, рассказы, доклады, отчёты об исследованиях и т. д.
- Решение конкретных проблем: изменение, улучшение конкретной ситуации, например уборка мусора на школьном дворе.

Правила проектной деятельности

1. Каждый может начать собственный проект.
2. Каждый может объединиться с другими в ходе работы над проектом.
3. Каждый может выйти из проекта при условии, что он не подводит других.
4. Каждый может не участвовать ни в одном проекте.

Как оценить свои учебные достижения?

Для этого надо освоить алгоритм самооценки:

1. Какова была цель задания (что нужно было получить в результате)?
2. Вы выполнили задание (получен ли результат)?
3. Вы выполнили задание верно или с ошибкой?
4. Вы выполнили задание самостоятельно или с чьей-то помощью?
5. Вспомните, как вы ставите оценки. Определите свою оценку.



Содержание

Модуль 1. Алгоритмизация и программирование

Этот модуль поможет вам:

- узнать, что такое алгоритм и как его записывать;
- научиться писать программы на языке программирования;
- овладеть средой разработки программ;
- освоить базовые алгоритмы.

Для этого вам надо научиться:

- понимать, что такое алгоритм;
- пользоваться интегрированной средой разработки;
- владеть конструкциями языка программирования.

Введение



Энциклопедия
Информатики
и программирования

Вы уже много работаете на компьютере, оперируете файлами, папками, создаёте текстовые документы, осуществляете поиск в Интернете, пользуетесь другими программными продуктами.

Все программы, которыми вы пользуетесь, создали люди. Сначала программы были продуманы и проработаны; затем были выбраны те языки программирования, на которых данные продукты эффективнее писать, подобраны команды программистов и назначены сроки выполнения.

Написание программ – очень непростое занятие. Оно требует от человека как профессиональных знаний в компьютерной области, так и усидчивости и терпения.

Приглашаем вас познакомиться с миром программирования!

§ 1. Алгоритм

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



В канцелярских магазинах продаются ежедневники. В этих книгах люди записывают свои планы на день, которые очень часто называют алгоритмами.

- Назовите тему урока. Сравните свой вариант с авторским (с. 142 учебника).

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

ПОРЯДОК ДЕЙСТВИЙ

Представьте, что мы хотим построить дом. Это очень ответственная и важная задача. Ведь мы не должны ничего забыть и построить дом точно к сроку. А для этого нам надо составить план наших действий. Давайте пофантазируем и создадим такой план.

План постройки дома:

1. Старт!
2. Найти и подготовить место для строительства.
3. Выполнить чертёж дома.
4. Выкопать котлован под фундамент.
5. Залить фундамент.
6. Построить стены.
7. Собрать и установить крышу.
8. Поставить окна и двери.
9. Произвести внутреннюю отделку.
10. Повесить на дверной проём красную ленточку.
11. Всё! Дом готов.

Теперь давайте посмотрим на наш план внимательно. Что мы написали?

Мы описали порядок действий, последовательно выполнив которые мы получим решение нашей задачи – задачи постройки дома. Сколько шагов нужно выполнить? Одиннадцать. Если бы мы составляли план более детально, то мы бы получили большее количество шагов. Но чтобы наш дом рано или поздно был построен, количество этих шагов должно быть конечным. Что ещё нам нужно, чтобы мы могли построить дом по плану? Мы, разумеется, должны понимать все пункты этого плана.

Рассмотрим другой пример. Вы видели когда-нибудь, как собирают мебель, например шкаф? Для этого читают прилагаемую к шкафу инструкцию и действуют по ней – раскладывают детали на полу, закручивают винты и саморезы и так далее. Прилагаемая к шкафу инструкция – это тоже порядок (последовательность) действий, предназначенных для решения задачи сборки шкафа.

Рассмотрим очень часто решаемую программистами задачу.

Задан числовой промежуток от a до b : $[a; b]$, где a и b – натуральные числа. Необходимо сформировать список (вывести на экран) всех простых чисел, которые находятся в этом промежутке (рис. 1.1). Давайте определим последовательность действий, которая позволила бы нам сделать это.



Рис. 1.1

1. Начало.
2. Положить $x = a$.
3. Проверить, является ли x простым числом. (Вспомните, какое число называется простым и как это можно проверить.)
4. Если x является простым, то вывести его, в противном случае проигнорировать.
5. Увеличить x на 1.
6. Если x стало больше, чем b , то перейти на шаг № 7 (алгоритм закончился), иначе перейти на шаг № 3 (работа продолжается).
7. Всё! Конец!

Давайте посмотрим на описанную последовательность.

Мы описали действия, направленные на решение нашей задачи? Да. Описание нам понятно.

Мы можем посчитать, какое количество проверок необходимо сделать, чтобы получить ответ? Да. Давайте посчитаем. Допустим, $a = 1$ и $b = 5$. Сколько мы сделаем проверок? 5. А если $a = 5$ и $b = 7$, то мы сделаем 3 проверки. (Попробуйте доказать это самостоятельно.)

Оказывается, во всех приведённых выше примерах мы имели дело с алгоритмами. Что же такое алгоритм?

АЛГОРИТМ

В информатике существуют разные формулировки определения понятия «алгоритм». Главное, что смысл понятия от этого не меняется.

В дальнейшем мы будем пользоваться следующим определением.

Алгоритм – это последовательность (порядок) действий (инструкций, команд) для некоторого исполнителя, направленных на решение поставленной задачи за конечное число шагов.

В этом определении есть незнакомое для нас слово: исполнитель. Что оно означает?

Исполнитель – это человек или устройство (машина), которые способны выполнить (исполнить) команды нашего алгоритма. Для исполнителя мы и создаём алгоритм.

Для кого мы писали инструкции, когда собирались строить дом? Для человека, для самих себя. А кто может искать простые числа на промежутке? Не только человек, но и машина, компьютер. И человек, и компьютер могут быть исполнителями алгоритма.

Важно понимать, что инструкции создаются для конкретного исполнителя, они не всегда универсальны. Если мы напишем инструкции для строителя, то их может не понять биолог, и наоборот. Каждый исполнитель может выполнять команды только из некоторого строго заданного набора – **системы команд исполнителя (СКИ)**.

Очень интересно происхождение термина «алгоритм». Оно происходит от имени среднеазиатского учёного Мухаммеда ибн Мусы Аль-Хорезми, который знаменит тем, что заложил основы того, что мы сейчас называем алгеброй. Около 825 года он написал книгу под названием «Об индийском счёте», в которой изложил основы придуманной в Индии десятичной системы счисления. Книга была переведена в XII веке на латинский язык, и имя автора стало звучать как *Algorizmi*, или *Algorizmus*. По его имени стали называть систему арифметических действий, основанных на десятичной системе счисления.



Аль-Хорезми
(ок. 787–ок. 850)

СВОЙСТВА АЛГОРИТМА

Любой алгоритм обладает следующими свойствами:

- 1. Дискретность** (пошаговость). Алгоритм состоит из последовательности отдельных шагов, причём на выполнение каждого шага исполнителю необходимо потратить конечное время.
- 2. Понятность.** Алгоритм должен содержать только те команды, которые входят в систему команд исполнителя, то есть которые в состоянии понять тот исполнитель, для которого алгоритм создаётся.

3. **Конечность** (результативность). Выполнение алгоритма должно заканчиваться за конечное число шагов. Если ваша последовательность действий бесконечна, то она алгоритмом не является.
4. **Определённость** (точность). Каждая команда алгоритма должна определять однозначное действие исполнителя. Также должен быть строго определён порядок выполнения команд. Алгоритм должен выдавать один и тот же результат для одних и тех же входных данных.
5. Алгоритм должен обеспечивать получение правильных результатов для любых допустимых исходных данных.

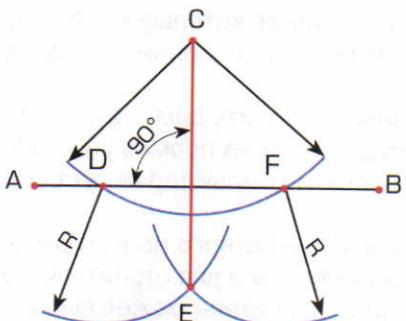


Рис. 1.2

Множество известных вам математических задач решаются по алгоритмам. Приведём примеры:

- 1) построение прямой, перпендикулярной данной и проходящей через указанную точку (рис. 1.2);
- 2) построение треугольника по трём сторонам (рис. 1.3);
- 3) решение различных уравнений, например $ax + b = 0$.

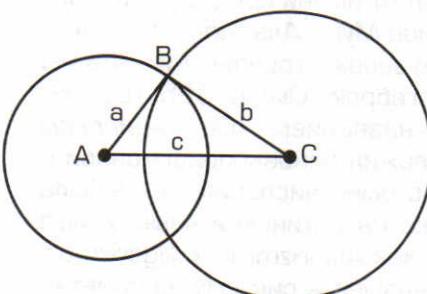


Рис. 1.3

Из свойств алгоритма следует возможность его **формального выполнения**, что очень важно для решения задач на компьютере. Компьютер не понимает смысл команд, содержание задачи, он работает с данными и может только строго выполнять заданные действия по командам (формально). Компьютер – формальный исполнитель.

ФОРМУЛИРУЕМ ВЫВОД

Алгоритм – одно из основных понятий информатики, окружающее и сопровождающее человека в его повседневной жизни.

Алгоритм – это последовательность (порядок) действий (инструкций, команд) для некоторого исполнителя, направленных на решение поставленной задачи за конечное число шагов.

Алгоритм обладает свойствами дискретности, понятности, конечности, определённости. Алгоритм обеспечивает получение правильных результатов для любых допустимых исходных данных.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Опишите понятие «алгоритм» своими словами. Сделайте упор на те фразы в определении, которые, по-вашему, наиболее важны. Объясните почему.

2. Составьте алгоритм приготовления вашего любимого пирога или сборки любой модели самолёта.

3. Проверьте созданный вами алгоритм на выполнение указанных в параграфе свойств.

4*. Есть некоторый автомат, который умеет только умножать число на 2 и прибавлять к числу 1. Составьте алгоритм получения из числа 1 числа 100.

§ 2. Способы записи алгоритма

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Теперь вы знаете, что такое алгоритмы, и умеете их составлять. Но как вы поступите, если необходимо передать алгоритм вашему партнёру в Австрию? И что делать, если вам нужно, чтобы ваш алгоритм был выполнен на компьютере?

- Какая проблема возникает? Сформулируйте основной вопрос урока. Сравните свою формулировку с авторской (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Что такое алгоритм? (§ 1)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

МНОГО РЕШЕНИЙ ОДНОЙ ЗАДАЧИ

Итак, теперь мы знаем, что такое алгоритм. Но давайте зададимся вопросом: всегда ли существует только один алгоритм для решения задачи?

Для того чтобы ответить на этот вопрос, представим себе следующую проблему. Мы хотим доехать из Москвы в Санкт-Петербург. Как вы думаете, сколькими способами мы можем это сделать? Правильно, таких способов много. Мы можем поехать на скором поезде, на пассажирском поезде, полететь на самолёте, добраться на автомобиле и найти ещё большое количество вариантов решения этой задачи.

Другой пример: необходимо вычислить сумму $1 + 2 + 3 + \dots + 100$. Мы можем начать суммировать числа от 1 до 100, а можем – от 100 до 1. От перемены мест слагаемых сумма не меняется. Кроме того, мы можем выписать формулу для этой суммы: $(100 + 1) \cdot 100 : 2$ (подумайте, откуда следует эта формула).

Вывод: для решения любой задачи существует множество алгоритмов. И задача программиста – выбрать тот, который в конкретной ситуации лучше.

Для того чтобы не запутаться в различных алгоритмах, их надо уметь записывать. Существует несколько способов записи алгоритмов. Давайте их рассмотрим. И поскольку мы изучаем программирование, то в дальнейшем

будем разбирать в основном алгоритмы, предназначенные для выполнения компьютером.

СЛОВЕСНЫЙ СПОСОБ

Словесный способ записи алгоритма мы с вами уже использовали. Он заключается в том, что шаги алгоритма записывают на родном языке автора этого алгоритма, при этом присваивая каждому шагу номер.

Давайте запишем алгоритм проверки числа M на простоту.

1. Взять число M .
2. Выписать все целые числа от 2 до $M - 1$.
3. Проверить делимость числа M на все выписанные числа.
4. Если M делится хотя бы на одно из них, то сделать вывод, что M – составное число, иначе сделать вывод, что M – простое число.
5. Конец.

БЛОК-СХЕМЫ

Но у словесного способа записи алгоритмов есть недостаток. Алгоритм записывается на человеческом языке: русском, английском, немецком или каком-то другом, значит, он может получиться многословным и не наглядным. Его могут не понять люди, говорящие на других языках. Такой алгоритм легко записать неправильно, потому что инструкции, записанные на разговорном языке, зачастую допускают неоднозначное толкование. Чтобы улучшить ситуацию, люди придумали **графический способ** записи, который называется языком **блок-схем**.

Создадим блок-схему рассмотренного выше алгоритма, проверяющего число на простоту (рис. 1.4).

Что обозначают геометрические фигуры в блок-схеме? Это стандартные обозначения блоков алгоритма:

Обозначение	Описание
	Начало или конец блок-схемы
	«Перейти к»
	Процесс или действие
	Ввод/вывод данных
	Условие

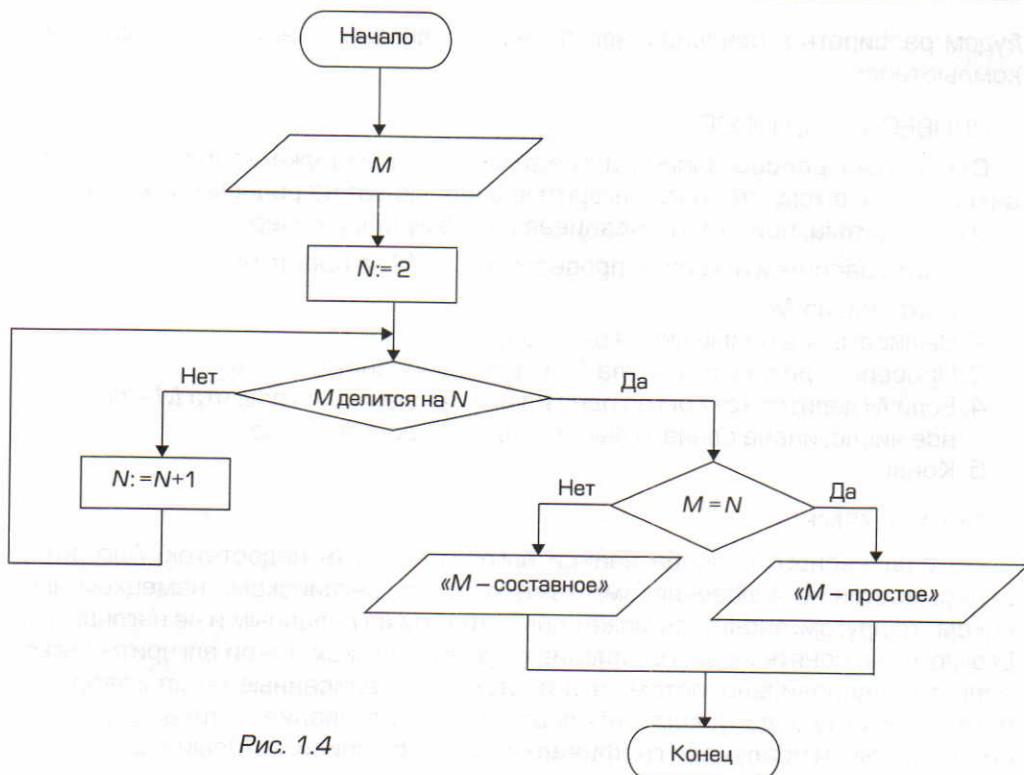


Рис. 1.4

С первого взгляда может показаться, что запись алгоритма усложнилась. Вы можете сказать, что в словесном способе было только 5 пунктов, а тут схема на полстраницы! Это, конечно, так. Но согласитесь, что во втором случае запись алгоритма приобрела более чёткий и наглядный вид. К тому же в блок-схемах применяются в основном символьные обозначения, записи на разговорном языке сводятся к минимуму или вовсе не используются.

Этот способ сродни чертежу, созданному инженером, по которому рабочий собирает изделие. Научимся разбирать этот «чертёж».

Блок «Начало» в овале обозначает начало алгоритма, вход в него.

Стрелки в блок-схеме обозначают **переход** к следующему блоку алгоритма.

Дальше идёт значок **параллелограмма**. В нём записываются операции **ввода и вывода** данных в компьютер. В нашем примере мы должны ввести число. Для этого нам понадобится переменная *M*. В программировании переменная обозначает область памяти компьютера, куда можно заносить и изменять разные значения. В нашем случае мы вводим в переменную *M* число, которое будем исследовать на простоту.

Затем идёт **прямоугольник**. В нём указываются **действия** (их может быть несколько), которые компьютер должен выполнить в данный момент. В нашем случае мы заносим в переменную *N* значение 2. Знак «*:*=» означает занесе-

ние в переменную значения (в программировании говорят: присваивание переменной значения).

Переходим к проверке **условия**, делится ли введённое нами число M на N . Условие в блок-схеме обозначается **ромбом**. В зависимости от выполнения (стрелка с надписью «Да») или невыполнения (стрелка с надписью «Нет») условия происходит переход от ромба к одному из двух блоков.

По правилам, в блок-схеме все стрелки «Да» идут в одну сторону от блок-схемы, а все стрелки «Нет» – в другую. Посмотрите внимательно на блок-схему и убедитесь в этом.

Если M не делится на N , то мы увеличиваем N на 1 и выполняем проверку снова. Если M делится на N , то переходим к проверке второго условия (второй ромб на блок-схеме): равно ли число M значению N . Если да, то выводим ответ «Число простое», а если нет, то ответ «Число составное».

Что означает тот факт, что N стало равным M ? Он означает, что мы перебрали все числа от 2 до $M - 1$ и ни на одно из них M не разделилось, то есть число M – простое.

И завершающий блок блок-схемы – **«Конец»**.

Поздравляем! Мы прочитали нашу блок-схему.

Теперь мы знаем, как составлять и читать алгоритмы, записанные в виде блок-схем.



Оказывается, для того чтобы определить, является ли число N простым, достаточно проверить в качестве делителей только числа, не превосходящие квадратный корень из N .

- Попробуйте объяснить, почему это так.

Но способ записи в виде блок-схем не годится для передачи этой записи компьютеру. Как мы говорили, компьютер – формальный исполнитель, для него нужен точный и однозначный язык, не допускающий неоднозначного толкования. Такие языки тоже называют формальными. Формальный компьютерный язык называется **языком программирования**.

Это и есть третий способ записи алгоритмов – запись на языке программирования.

ЯЗЫК ПРОГРАММИРОВАНИЯ

Язык программирования – это специальный язык, созданный исключительно для того, чтобы записывать на нём алгоритмы для компьютера. Алгоритм, записанный на языке программирования, называется **программой**.

Современные языки программирования для удобства работы с ними программистов используют слова естественного человеческого языка, цифры и другие знаки. Они обладают **алфавитом** (набором используемых в языке знаков), **синтаксисом** (правилами написания элементов программы) и **семантикой** (это означает, что любое слово и инструкция языка несут в себе определённый смысл).

Мы будем писать программы на языке **Паскаль**.

Алфавит языка Паскаль:

- 1) буквы латинского и русского алфавитов;
- 2) цифры;
- 3) специальные символы, такие, как «*», «+», «-», «.», «/», «'» и т. д.

В программировании символом «*» обозначают операцию умножения, символом «/» – операцию деления.

Предлагаем вам написать нашу первую программу на языке Паскаль. Пусть это будет программа, вычисляющая значение выражения $(\beta - \alpha) \cdot (\beta + \alpha)$.

```
program first;
var
    alfa, beta: integer;
begin
    alfa := 1;
    beta := 2;
    alfa := (beta - alfa) * (beta + alfa);
    writeln(alfa);
end.
```

Разберём эту полноценную программу по строкам.

Первая строка начинается ключевым словом **program**.

Ключевое слово – это важное понятие языка, означающее, что такие слова имеют специальные, раз и навсегда заданные разработчиками языка значения и употребляются всегда в одном и том же смысле. Говорят, что эти слова **зарезервированы** в языке.

Итак, ключевое слово **program** всегда означает начало программы.

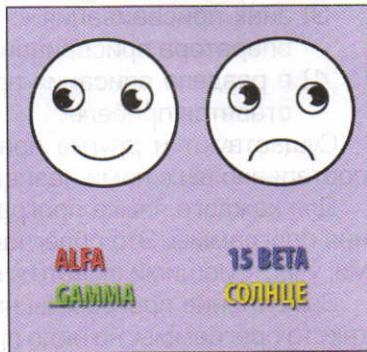
Слово, следующее за **program**, – это **имя программы**. Таким образом, по смыслу мы написали следующее: «Начинается программа с именем **first**». Имя программы задаётся программистом (в отличие от ключевых слов). Имя программы должно начинаться или с латинской буквы, или с символа «_».

Вторая строка в нашем примере содержит ключевое слово **var** (сокращение от английского слова **variable** – переменная). Это слово означает, что мы начинаем раздел описания **переменных** (вспомните, что такое переменная).

В языке Паскаль каждая переменная имеет определённый **тип**, он определяется **типовом данных**, которые могут храниться в области памяти, обозначенной этой переменной. Примеры типов данных: целые числа, символы, строки и т. д.

В приведённой программе мы описали две переменные с именами **alfa** и **beta** и говорим, что в этих переменных можно хранить только целые числа (тип **integer**). Тип переменных указывается после знака «:». Если переменных одного типа несколько, то их можно перечислять через знак «,». В языке Паскаль в именах переменных можно использовать только латинские буквы, цифры и символ «_», причём эти имена не могут начинаться с цифры.

Например, у переменных `alfa` и `_gamma` правильные имена, а у переменных `15beta` и `солнце` – неправильные. Кроме того, в качестве имён переменных нельзя использовать ключевые слова (вспомните, что мы говорили о ключевых словах). Нельзя создавать две переменные с одним именем в одной программе. Важно знать, что имена переменных и ключевые слова в языке Паскаль не зависят от регистра. Это означает, что имена `alfa`, `Alfa`, `ALFA` и `alFa` – это одно и то же имя.



После этого со слова `begin` начинается **основная часть** программы. Здесь записаны три оператора присваивания и один оператор вывода. Рассмотренные нами ранее инструкции (команды) алгоритма в языках программирования называются **операторами**. Постепенно мы рассмотрим разные операторы языка Паскаль. Операторы в Паскале разделяются между собой точкой с запятой.

Первый **оператор присваивания** заносит [присваивает] в переменную `alfa` значение 1 (обозначение присваивания «:=» нам уже знакомо из блок-схем), второй оператор присваивает переменной `beta` значение 2.

Разберём работу оператора присваивания на примере третьей строки основной части программы: `alfa := (beta - alfa) * (beta + alfa);`

Сначала вычисляется выражение, стоящее в правой части оператора присваивания, причём вместо переменных подставляются их значения. Получаем: $(2 - 1) \cdot (2 + 1) = 3$. Теперь занесём вычисленное значение в переменную, находящуюся в левой части оператора, то есть в `alfa`. Итак, `alfa` примет значение 3.

Оператор вывода `writeln(alfa)` выводит значения переменных, которые находятся у него в скобках. В нашем случае он выведет на экран значение переменной `alfa`, то есть 3.

Завершается любая программа ключевым словом `end`. [с точкой].

ХОРОШИЙ СТИЛЬ

Программы надо оформлять [форматировать] аккуратно и наглядно и в едином стиле. Скажите, вам было бы приятно читать книгу, в которой каждый абзац начался бы с разного отступа от края страницы и буквы постоянно меняли бы свой размер? К тому же в ряде случаев это затрудняло бы понимание смысла текста. Так же дело обстоит и с программами. Обратите внимание, как написана наша первая программа:

- 1) ключевые слова `program`, `var`, `begin`, `end` находятся строго друг под другом;
- 2) операторы имеют отступы от левого края программы;

- 3) знак присваивания «:=» отделён пробелами от левой и правой частей оператора присваивания;
- 4) в разделе описания переменных после запятой и двоеточия («,» и «::») ставятся пробелы.

Существуют и другие правила форматирования программы на Паскале, постепенно вы с ними познакомитесь.

Для каждого языка программисты договариваются о стандарте оформления программы. Это облегчает понимание программы вами и другими людьми, участвующими вместе с вами в создании программ.

Выполнение программы на языке Паскаль не зависит от форматирования текста программы, но надо проявлять уважение к тому человеку, кто будет читать ваши программы.

ФОРМУЛИРУЕМ ВЫВОД

Для решения поставленной задачи можно придумать множество алгоритмов.

Существуют разные способы записи алгоритмов. Мы изучили словесный способ, графический (блок-схемы) и запись на языке программирования (программы).

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Составьте блок-схему для приведённой в параграфе программы `first`.
2. Какие из имён переменных неправильные и почему: `a`, `папа`, `_p2`, `5b`, `zyablik?`
3. Напишите программу на языке Паскаль, в которой описаны три переменные: `a`, `b` и `c`. В переменную `a` заносится значение 10, в переменную `b` – 8, а в переменную `c` – 3. Необходимо вычислить выражение $2*a + b - c$ и результат занести в переменную `b`, значение которой потом вывести на экран.

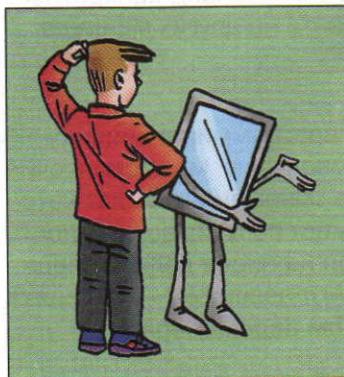
4*. Автомат умеет из любой дроби $\frac{a}{b}$ получить дроби $\frac{a+b}{b}$, $\frac{a-b}{b}$ и $\frac{b}{a}$. Как из $\frac{1}{2}$ получить $\frac{1}{4}$?

5*. Запишите словесно и составьте блок-схему алгоритма для вычисления суммы $S = 1 + 2 + 3 + \dots + N$ без использования соответствующей формулы.

6*. Составьте блок-схему алгоритма, вычисляющего факториал натурального числа. Факториалом числа n (обозначается $n!$) называется произведение всех натуральных чисел от 1 до n , то есть $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$.

§ 3. История языков программирования

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Мы уже немного умеем программировать для компьютера. Но стоп! Мы же знаем, что компьютер работает с данными, представленными только двумя цифрами – 0 и 1. Как же он понимает при этом язык Паскаль?

И всегда ли существовал язык Паскаль? Интересно, что было до него – больше полувека назад?

- Сформулируйте основной вопрос урока. Сравните свою формулировку с авторской (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Какие способы записи алгоритмов вы знаете? (§ 2)

Вспомните, что вы знаете про функции устройств компьютера, представление и обработку данных, программное обеспечение компьютера. (Книга 1 учебника, модуль 1.)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

МАШИННЫЙ КОД

Человек работает с информацией, а компьютер – с данными. Для того чтобы компьютер мог помочь нам обработать информацию, её нужно закодировать. Информация любого вида представлена в компьютере одинаково: в виде **данных** – последовательностей нулей и единиц (**в цифровой форме**). Из цифр составляются числа. Процессор компьютера обрабатывает числовые коды. Вспомним, что каждый исполнитель может выполнять команды только из своей системы команд. Система команд исполнителя компьютер состоит из числовых кодов, причём каждая модель процессора имеет свой собственный набор команд.

Как же «общаться» с компьютером? В какой форме передавать ему программы? На языке числовых кодов? Сначала люди так и поступали. Когда был изобретён первый компьютер, программисты придумали самый первый язык для компьютера и назвали его **машинным кодом**. Это был очень сложный для

человека язык. В нём не было слов, а все команды записывались числами. Могла быть создана, например, вот такая программа: 04 08 12 22 77 32 01.

Программирование в машинных кодах имело и ещё один недостаток – программы были ориентированы только на конкретный тип компьютеров, они были **машинно-зависимыми** – привязанными к набору команд конкретного процессора. Эти программы нельзя было использовать на других машинах.

ЯЗЫК АССЕМБЛЕРА

Со временем, чтобы упростить для человека программирование, было решено заменить числовые коды команд на символьные мнемонические обозначения [посмотрите в словаре значение слова «мнемонический»], которые человек воспринимает лучше. Теперь программист получил возможность не задумываться о конкретных числах, замену слов на числа делала специальная программа, называемая **ассемблером** (от англ. assembler – сборщик). Сам язык программирования стали называть **языком ассемблера**.

Небольшой фрагмент программы на ассемблере выглядит, например, так:

```
mov ah, 9
```

```
mov dx, OFFSET Msg
```

```
int 21h
```

Как и язык машинных кодов, язык ассемблера является машинно- зависимым.

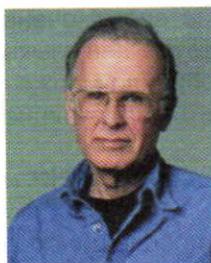
Машинный код и язык ассемблера являются языками программирования **низкого уровня**, то есть приближёнными к устройству машины (ближе к машине, чем к человеку). На языке ассемблера пишут программы, требующие большой скорости вычислений, на нём часто создают драйверы (программы, управляющие устройствами, подключёнными к компьютеру) и программы ядра операционной системы.

ЯЗЫКИ ВЫСОКОГО УРОВНЯ

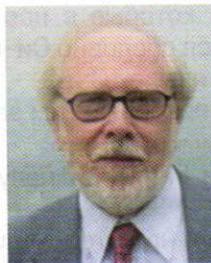
Технологии программирования развивались, и были придуманы языки программирования **высокого уровня** (близкие к человеку). В таких языках применяются конструкции, очень похожие на человеческий (естественный) язык. Они позволяют удобно описывать действия, в то время как на языке ассемблера это занимало много времени и усилий.

Для преобразования программы в машинный код применяются программы, называемые **трансляторами** (от англ. translator – переводчик).

Программы, написанные на языках высокого уровня, работают медленнее, чем их низкоуровневые аналоги, однако получаемые преимущества существенны.



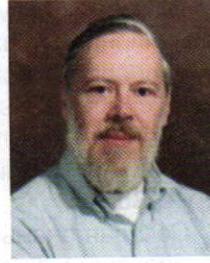
Джон Бэкус
(1924–2007)



Никлаус Вирт
(род. в 1934 г.)



Кен Томпсон
(род. в 1943 г.)



Деннис Ритчи
(1941–2011)

Одним из первых значимых языков высокого уровня был язык **Фортран**. Он был разработан в период с 1954 по 1957 год в корпорации IBM под руководством американского учёного в области информатики Джона Бэкуса. Название языка происходит от двух английских слов: **Formula Translator** (переводчик формул). Язык был очень распространён среди инженеров и учёных.

Параллельно с языком Фортран развивался язык **Алгол** (от английского **Algorithmic Language** – алгоритмический язык).

В 1970 году швейцарский учёный Никлаус Вирт придумал язык **Паскаль**, названный так в честь французского математика и философа Блёза Паскаля, создавшего механическую вычислительную машину Паскалину, которая умела выполнять простые арифметические действия над числами. Этот язык широко используется для обучения программированию. Мы с вами тоже учимся программированию на его основе.

В начале 70-х годов XX века был создан, наверное, самый популярный и мощный язык программирования, который применяется повсеместно по сей день. Это язык **C** (произносится как **Си**). На этом языке написаны практически все современные операционные системы. Язык С был создан для использования в операционной системе *Unix*.

Язык С заложил синтаксические основы всех последующих языков программирования. Это означает, что программист, знакомый с синтаксисом языка С, легко начинает работать и на другом языке.

Создали язык С в лаборатории Bell Labs Деннис Ритчи и Кен Томпсон.

В 80-х годах XX века появился язык, являющийся сегодня одним из популярных языков среди программистов. Это язык **C++** (++ означает «следующий»). Это название можно перевести как «язык, следующий за С». Придумал язык C++ также сотрудник Bell Labs Бьёрн Страуструп.



Бьёрн Страуструп
(род. в 1950 г.)

Мы хотим отметить ещё два языка, которые в настоящее время очень популярны. Это языки **Java** и **C#** (читается дословно **Си-шарп**). Эти два языка унаследовали принципы, заложенные в языках С и С++, но добавили множество современных технологий, которые позволяют им занимать лидирующие позиции по популярности среди программистов.

Для написаний приложений для сети Интернет популярны языки **PHP** и **Java Script**.

Сегодня очень популярен язык **Python**, на котором пишется большинство приложений во многих ведущих ИТ-компаниях, таких, как Google, Yandex и многих других.

Отметим ещё два языка, относящиеся к важному современному направлению – созданию *искусственного интеллекта*. Это язык *логического программирования Prolog* и язык *функционального программирования Lisp*.

Язык Prolog был создан в то время, когда человечество мечтала об искусственном интеллекте. Люди хотели создать объект, который смог бы мыслить подобно человеку, и развивающимся компьютерным технологиям необходим был язык, на котором можно было описать поведение такого объекта. Таким языком стал Prolog. Начало истории языка относится к 1970-м годам. В наше время этот язык используется в научных исследованиях и разработках.



Джон Маккарти
(1927–2011)

Язык *Lisp* был придуман известным американским информатиком, создателем термина «искусственный интеллект» Джоном Маккарти в 1958 году. Язык *Lisp* предлагает совсем иной способ создания программ. Вместо того чтобы описывать последовательность действий, как мы с вами делали, программист на языке *Lisp* формулирует описание задачи, а уже специальная *Lisp-машина* решает данную задачу. Программирование на языке *Lisp* очень захватывающее и интересное.

Согласно статистике, приводимой на различных интернет-ресурсах, в мире существует несколько тысяч языков программирования. Не удивляйтесь, пожалуйста. Ряд этих языков не нашли применения и были забыты, некоторая часть сыграла свою блестящую роль и отправилась на заслуженный отдых. Какие-то из них (С, С++, Java, C#, PHP и пр.) постоянно применяются, другие используются реже.

Хороший программист всегда может овладеть необходимым ему языком, но для этого надо знать основные технологии программирования.

ФОРМУЛИРУЕМ ВЫВОД

Языки программирования делятся на две группы: языки программирования низкого уровня и языки программирования высокого уровня.

Компьютер понимает только машинный код, который уникален для каждого типа компьютеров.

Язык ассемблера использует мнемонические символьные обозначения вместо чисел. Программы на ассемблере работают быстро.

Программы на языках высокого уровня позволяют писать понятные и переносимые с компьютера на компьютер программы.

К современным языкам высокого уровня относятся: Паскаль, С, С++, Java, С#, Python, PHP и др.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Что представляет собой система команд исполнителя компьютер?
2. Что такое машинный код?
3. Верно ли утверждение: язык ассемблера – это язык низкого уровня?

Объясните свой ответ.

4. Назовите известные вам языки высокого уровня. В каких областях они используются?

5*. Есть такой язык, символом которого является черепаха. Найдите в сети Интернет информацию об этом языке программирования и подготовьте о нём небольшой рассказ.

§ 4. Работа в среде программирования

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



После урока информатики семиклассник Вася пришёл к своему папе, который работает программистом. Вася был очень рад и горд, поскольку только что написал на листочке бумаги программу, которая умеет складывать два любых целых числа. «Папа! Запусти эту программу», – попросил Вася и протянул отцу листок. «Я не могу её запустить, – сказал пapa. – Для того чтобы это сделать, программу нужно ввести в компьютер».

- Какое решение проблемы предлагает пapa? Сформулируйте тему урока. Сравните свой вариант с авторским (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

- Что такое алгоритм? (§ 1)
- Как записывают алгоритмы? (§ 2)
- Вспомните структуру программы на языке Паскаль. (§ 2)
- Вспомните, как вы работали в операционной системе. (Книга 1 учебника, модуль 1.)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

КАК МОЖНО ВВОДИТЬ ПРОГРАММЫ В КОМПЬЮТЕР

Когда-то программы создавались в текстовом редакторе: автор программы набирал её текст и сохранял его в файле.

- Вспомните, какие типы файлов вы знаете.

После этого запускалась специальная программа – компилятор (разновидность транслятора), которая анализировала текст программы (он часто называется **исходным кодом**) и переводила его в машинные коды: создавала новый файл – **исполнимый**, который уже мог быть выполнен компьютером. Как вы думаете, есть ли исполняемые файлы на вашем компьютере? Конечно, есть! Все программы, которыми вы пользуетесь, – текстовые и графические редакторы, браузеры, игры, – всё это исполняемые файлы.

Такой способ имел следующий недостаток. Представьте, что мы допустили в тексте **синтаксическую ошибку** (ошибку написания языковой конструкции – ключевого слова, переменной, оператора и т. д.), например, написали слово `integer` как `iteger`. Компилятор, проверяя текст программы, ошибку непременно обнаружит и укажет нам её место в тексте, а мы должны будем заново открыть текстовый редактор, найти это место и исправить опечатку. А если таких ошибок много? Ведь программу написать сразу без ошибок очень трудно, и работа программиста предполагает поиск и исправление своих ошибок. В таких случаях приходилось много раз переключаться из текстового редактора в программу–компилятор и обратно.

Это было неудобно, и программисты придумали объединить текстовый редактор и транслайтер. При этом получилась система, которую называют **интегрированной средой разработки**. Слово «интеграция» произошло от латинского слова `integrum` – «целое» и означает объединение, взаимопроникновение. В такой среде объединены текстовый редактор, который умеет выделять ключевые слова программы и автоматически форматировать её текст, компилятор, который создаёт из текста программы исполнимый файл, программа, помогающая нам искать ошибки, – отладчик и много различных удобных вещей.

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ PascalABC.NET

Интегрированная среда разработки *PascalABC.NET*, в которой мы с вами будем создавать наши программы, доступна всем пользователям сети Интернет на сайте (является бесплатным программным обеспечением). Вы можете скачать эту среду и установить её самостоятельно на своём компьютере.

Давайте запустим *PascalABC.NET*. Мы увидим основное окно программы (рис. 1.5).

Окно программы делится на две области: в верхней программист пишет свой код, а в нижней осуществляется его диалог с программой.

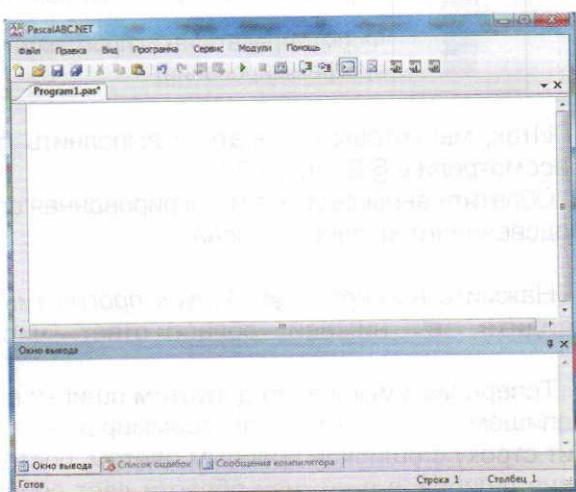


Рис. 1.5

Нижняя область имеет вкладки: *Окно вывода*, *Список ошибок* и *Сообщения компилятора*. В процессе работы количество вкладок может меняться.

Над верхней областью располагается панель кнопок, которая очень напоминает панель инструментов текстового редактора. Если вы будете наводить курсор мыши на кнопки, то система будет выдавать вам подсказки о том, какая кнопка за что отвечает.

Первые 11 кнопок отвечают за работу текстового редактора, они позволяют: создать программу, открыть программу, сохранить программу, сохранить программу под другим именем, вырезать, копировать и вставить фрагменты текста и т. д. Остальные кнопки мы рассмотрим подробнее.

Кнопка	Описание
	Запустить программу
	Остановить программу
	Компилировать программу
	Варианты отладки (поиска ошибок) программы
	Убрать или показать нижнюю область
	Форматировать текст программы (вспомним про хороший стиль программирования)

Итак, мы готовы написать и выполнить первую программу, которую мы рассмотрели в § 2 (рис. 1.6).

Обратите внимание, что интегрированная среда разработки помогает нам – подсвечивает ключевые слова.

Нажмите на кнопку Запуск программы. Программа выполнится, и вы увидите в нижней области ответ – число 3 (рис. 1.7).

Теперь мы умышленно допустим ошибку в тексте: вместо слова `integer` напишем `iteger`. Среда программирования тут же среагирует на это, выделит строку с ошибкой красным цветом, поставит курсор в то место, где найдена ошибка, а в нижней области даст описание. В нашем случае система сообщит: «Неизвестное имя 'iteger'» (рис. 1.8).

Мы можем не запускать программу сразу, а только *откомпилировать* её, то есть проверить на наличие синтаксических ошибок и создать исполнимый файл. Для того чтобы это сделать, надо нажать кнопку Компилировать.

```

program first;
var
  alfa, beta: integer;
begin
  alfa := 1;
  beta := 2;
  alfa := (beta - alfa) * (beta + alfa);
  writeln(alfa);
end.

```

Рис. 1.6

Okno вывода

3

Рис. 1.7

Список ошибок		
Строка	Описание	
1	3	Неизвестное имя 'iteger'

Рис. 1.8

ЧИСЛОВЫЕ ТИПЫ ЯЗЫКА ПАСКАЛЬ

Паскаль – язык со строгой типизацией. Это означает, что любая переменная, которую мы используем, должна быть описана в разделе **var**. Тем самым мы определяем принадлежность переменной какому-либо типу языка.

```

program EX1;
begin
  Alfa := 1;
  writeln(Alfa);
end.

```

ОШИБКА НА СТАДИИ КОМПИЛЯЦИИ

```

program EX2;
var
  Alfa: integer;
begin
  Alfa := 1;
  writeln(Alfa);
end.

```

ПРАВИЛЬНО

Рассмотрим два основных типа языка, используемых для работы с числами.

1. Для описания **целых чисел** используется тип *integer*. Этот тип описывает целое число в следующем диапазоне значений: [-2 147 483 648..2 147 483 647]. Примеры: 123; 0; -5; +8; -587.

Над целыми переменными можно производить следующие операции:

Название	Обозначение	Пример
Отрицание	-a	a := - a
Сложение	a + b	a := b + c
Вычитание	a - b	a := b - c
Умножение	a * b	a := b * c
Взятие целой части от деления	a div b	a := b div c
Взятие остатка от деления	a mod b	a := b mod c

Операции, находящиеся в таблице над жирной чертой, – это стандартные арифметические операции, нам знакомые. Но почему среди них нет операции деления? Дело в том, что если мы будем складывать, умножать или вычитать целые числа, то мы всегда будем получать также целые числа. А вот с делением ситуация иная. Попробуйте разделить 5 на 2. Получится 2,5. А это уже не целое число.

Поэтому для целых чисел существует операция **деление с остатком**. А при таком делении всегда получаются целая часть и остаток. Если мы 5 разделим с остатком на 2, то получим целую часть 2 и 1 в остатке.

Итак, $5 \text{ div } 2 = 2$ (получение целой части), $5 \text{ mod } 2 = 1$ (получение остатка).

- Как вы думаете, как можно проверить число на чётность?

2. Для описания **вещественных чисел** (в математике они называются действительными) применяется тип *real*. С его помощью описываются, например, числа: -12.50; 0; 12; 0.15. Обратите внимание на то, что разделителем целой и дробной частей является точка. Дробная часть может и отсутствовать. Операциями над вещественными числами могут быть: $-a$; $a + b$, $a - b$,

$a * b$, a / b . В этом случае операция деления присутствует, но только с одним ограничением: на 0 делить нельзя!

Рассмотрим несколько задач.

ЗАДАЧИ

Задача 1. Вводятся два числа. Вывести их сумму.

```
program summa;
var
  a, b: integer;
begin
  readln(a);
  readln(b);
  writeln('Сумма a+b=', a + b);
end.
```

Проанализируем решение.

Для того чтобы ввести значение с клавиатуры, необходимо воспользоваться оператором `readln`, при этом в скобках указывается переменная, значение которой мы вводим.

Если мы хотим внести в программу какие-то комментарии, то мы начинаем их с символов `«//»` и можем продолжать до конца строки.

Теперь посмотрим на вывод. Язык Паскаль позволяет выводить сразу несколько значений. При этом они разделяются символом `«,»`. В нашем примере мы сначала выведем строку `«Сумма a+b=»`, а затем значение суммы `a` и `b`.

- Введите эту программу в компьютер, выполните её и посмотрите на результат. Значения переменных надо вводить каждое в новой строке. Чтобы вводить значения в одной строке через пробел, в программе следует заменить, по крайней мере, первый оператор `readln` на `read`.

Задача 2. Вводится трёхзначное число. Вывести его в обратном порядке.

```
program rotate;
var
  n: integer;
begin
  write('Введите число N: ');
  readln(n);
  write('Обратный порядок: ');
  write(n mod 10); // Вывели последнюю цифру
  n := n div 10;
  write(n mod 10); // Вывели среднюю цифру
```

```
    writeln(n div 10); // Вывели первую цифру
end.
```

Заметим, что здесь мы использовали оператор `write` вместо известного нам `writeln`. Отличие `write` от `writeln` заключается в том, что `writeln` после вывода очередного значения переводит курсор на новую строку, а `write` – нет.

Разберём эту программу на примере. Допустим, мы ввели число 123.

Программа сначала выведет строку «Обратный порядок:», а затем остаток от деления 123 на 10, то есть 3. При этом курсор на новую строку не переводится.

После этого у нас идёт оператор `n := n div 10`. Сначала будет вычислена целая часть, получившаяся при делении 123 на 10, – это 12, а потом значение 12 будет занесено в переменную `n`. Итак, теперь в `n` находится значение 12.

Затем мы снова выводим остаток от деления `n` на 10, то есть 2.

В итоге мы выводим `n div 10`, то есть 1, и переводим строку.

Результат работы программы:

Обратный порядок: 321

- Предлагаем вам ввести эту программу в компьютер и проверить её работу.

Задача 3. Написать программу определения максимума из двух чисел.

Для решения этой задачи нам понадобится **условный оператор**. На языке блок-схем этот оператор обозначается ромбом.

Условный оператор имеет следующий вид:

```
if <условие> then оператор1; //без альтернативы
```

или

```
if <условие> then оператор1           //с альтернативой
else оператор2;
```

Обратите внимание, что точка с запятой перед `else` не ставится.

Для решения нашей задачи надо записать следующий условный оператор:

```
if a > b then writeln(a)
else writeln(b);
```

Эта запись читается так: если `a` больше `b`, то вывести `a`, иначе вывести `b`.

Вот полное решение нашей задачи:

```
program max;
var
  a, b: integer;
begin
  write('Введите число a: ');
  readln(a);
  write('Введите число b: ');
  readln(b);
  if a > b then writeln(a)
  else writeln(b);
end.
```

ФОРМУЛИРУЕМ ВЫВОД

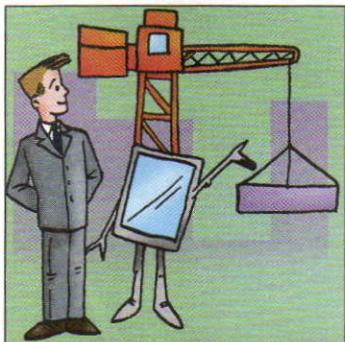
1. Для создания программ существуют интегрированные среды разработки. Они соединяют в себе удобный текстовый редактор и компилятор – программу, которая преобразует исходный код в исполимый файл. Мы используем среду PascalABC.NET.
2. Все переменные, используемые в программе, должны принадлежать к какому-либо типу.
3. Для чисел определены два типа: *integer* – для целых и *real* – для вещественных, причём для целых чисел нет операции деления.
4. Для ввода данных применяется оператор *read* или *readln*, а для вывода – оператор *write* или *writeln*.
5. Для реализации условий применяется условный оператор *if... then... else*.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. С клавиатуры вводится число – длина стороны квадрата. Вычислите и выведите площадь и периметр квадрата.
2. Вводится расстояние *L* в сантиметрах. Выясните, сколько в нём полных метров. Выведите это количество.
3. С клавиатуры вводится число *n*. Если оно положительное, то выведите n^2 , а если отрицательное, то выведите 0.
- 4*. С клавиатуры вводятся три числа. Определите максимальное из них.
- 5*. С клавиатуры вводится четырёхзначное число. Вычислите сумму его цифр.

§ 5–6. Циклы

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Представьте себе, что вы – строитель и что у вас есть инструкция, следуя которой, вы должны построить кирпичный дом из 1000 кирпичей. Вы открываете инструкцию, а там написано:

1. Взять кирпич № 1, положить его на место № 1.
2. Взять кирпич № 2, положить его на место № 2.

И так 1000 раз.

Удобно пользоваться такой инструкцией?

- Какая проблема возникает? Сформулируйте её. Сравните свою формулировку с авторской (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Вспомните, как записывают алгоритмы. (§ 2)

Вспомните структуру программы на языке Паскаль. (§ 2)

Какие вы знаете числовые типы данных? (§ 4)

Вспомните, что вы знаете о работе в интегрированной среде разработки. (§ 4)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

ПОВТОРЯЮЩИЕСЯ ДЕЙСТВИЯ

Программисты решают разные задачи. Очень редко бывает так, чтобы задачи не имели повторяющихся действий. Например, представьте, что нам надо вычислить сумму натуральных чисел от 1 до 100, но без использования формулы. Решение:

1. Начало.
 2. Положить $s = 0$, а $i = 0$ (напомним, что на языке Паскаль это записывается так: $s := 0$; $i := 0$).
 3. $i := i + 1$;
 4. $s := s + i$;
 5. Если $i = 10$, то вывести s и перейти на пункт № 6, иначе перейти на пункт № 2.
 6. Конец.
- Каким способом мы сейчас записали алгоритм решения поставленной задачи?

В данном алгоритме шаги с 3-го по 5-й будут повторяющимися. Конечно, мы могли записать наш алгоритм и без конструкции, обозначающей повторение, но тогда он вряд ли уместился бы на листе бумаги. Указание повторов позволяет упростить написание программ.

Такие повторяющиеся конструкции называются **циклами** или **циклическими конструкциями**.

Любая циклическая конструкция состоит из трёх составляющих (этапов):

- Инициализация.** Это слово происходит от английского термина initialization, что означает «начальная установка». В этот момент мы должны задать параметры, необходимые для запуска цикла. В приведённом примере это шаг № 2.
- Тело цикла.** На этом этапе мы должны описать повторяющиеся действия. В нашем примере это шаги № 2 и № 3.
- Условие продолжения или окончания цикла.** В этом месте мы должны описать то правило, следуя которому цикл должен или продолжаться, или прекратиться.

Инициализация всегда располагается самой первой, а вот тело цикла и условие могут меняться местами. Давайте рассмотрим, какие циклы бывают и как они записываются на языке Паскаль.

ЦИКЛЫ С ПРЕДУСЛОВИЕМ

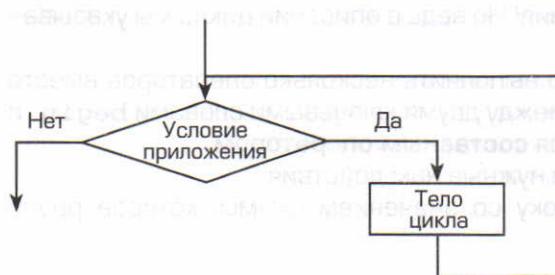


Рис. 1.9

Как вы думаете, что означает слово «предусловие»? Это означает, что в данном виде циклов условие находится перед телом цикла и является условием продолжения. То есть если это условие выполняется (истинно), то цикл будет продолжен, а если не выполняется (ложно) – закончен. Блок-схема такого цикла приведена на рис. 1.9.

На языке Паскаль такая конструкция описывается следующим образом:

while <условие_продолжения> do <оператор>;

Здесь **while** и **do** – ключевые слова, описывающие цикл. Дословно эту конструкцию можно перевести на русский язык так: пока <условие_продолжения> истинно, выполнять <оператор>.

Запишем и разберём решение нашей задачи по вычислению суммы чисел от 1 до 100 с помощью конструкции **while ... do**.

```

program p1;
var
    i, s: integer;
begin
    // инициализация
    s := 0;
    i := 0;
    // условие продолжения
    while (i < 100) do
        // тело цикла
        begin
            i := i + 1;
            s := s + i;
        end;
    // вывод результата
    writeln('Сумма равна: ', s);
end.

```

Мы начали программу *p1* и описали две переменные: *s* и *i*. На стадии инициализации мы занесли в переменные значение 0. Условием продолжения у нас будет *i* < 100.

Таким образом, цикл у нас будет читаться так: пока *i* меньше 100, выполнять... Что же надо выполнять? Нам надо к *i* прибавить единицу, и *s* увеличить, то есть совершить два действия! Но ведь в описании цикла мы указывали один <оператор>?

Не волнуйтесь. Когда нам нужно выполнить несколько операторов вместо одного, мы просто размещаем их между двумя ключевыми словами **begin** и **end**. Такая конструкция называется **составным оператором**.

Итак, в теле цикла мы выполним нужные нам действия.

В результате мы выведем строку со значением суммы, которое равно 5050.

ЦИКЛЫ С ПОСТУСЛОВИЕМ

В отличие от предусловия, **постусловие** – это условие, которое расположается после тела цикла. В языке Паскаль оно является условием окончания. Это означает, что цикл будет закончен в тот момент, когда это условие станет истинным, а пока оно ложное, цикл продолжается. Блок-схема такого цикла приведена на рис. 1.10.

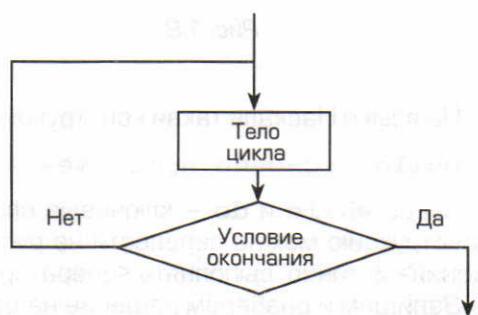


Рис. 1.10

Для описания подобной конструкции на языке Паскаль необходимо написать:

```
repeat <оператор1>;  
    <оператор2>;  
    ...  
    <операторN>  
until <условие_окончания>;
```

Здесь **repeat** и **until** – ключевые слова, описывающие цикл. На русский язык эту конструкцию можно перевести так: повторять <оператор1>, <оператор2>,... <операторN> до тех пор, пока <условие_окончания> не станет истинным.

Настало время записать и разобрать решение нашей задачи по поиску суммы чисел от 1 до 100 с помощью конструкции **repeat ... until**.

```
program p2;  
var  
    i, s: integer;  
begin  
    // инициализация  
    s := 0;  
    i := 0;  
    repeat  
        // тело цикла  
        i := i + 1;  
        s := s + i;  
    // условие окончания  
    until (i = 100);  
    // вывод результата  
    writeln('Сумма равна: ', s);  
end.
```

Так же, как и в предыдущем случае, в начале программы **p2** мы описали две переменные: *i* и *s*.

На стадии инициализации мы точно так же занесли в переменные значение 0.

Условием окончания у нас будет *i* = 100.

Прочтём наш цикл: повторять действия до того момента, как *i* станет равным 100.

Результатом работы нашей программы снова будет число 5050.

ПРИМЕРЫ ПРОГРАММ

Задача 1. Написать программы, вычисляющие факториал числа n , которое вводится с клавиатуры, используя известные нам циклические конструкции. Напомним, что факториал $n! = 1 \cdot 2 \cdot \dots \cdot (n - 1) \cdot n$. Кроме того, факториал нуля считается равным 1.

Цикл с предусловием:

```
program fact1;
var
  n, f, i: integer;
begin
  write('Введите число n: ');
  readln(n);
  write('Факториал ', n, ' равен ');
  i := 0;
  f := 1;
  while (i < n) do
    begin
      i := i + 1;
      f := f * i;
    end;
  writeln(f);
end.
```

Цикл с постусловием:

```
program fact2;
var
  n, f, i: integer;
begin
  write('Введите число n: ');
  readln(n);
  write('Факториал ', n, ' равен ');
  i := 0;
  f := 1;
  repeat
    inc(i);
    f := f * i;
  until (i = n);
  writeln(f);
end.
```

Во второй программе мы операцию увеличения значения i на 1 записали как `inc(i)`.

Задача 2. С клавиатуры вводятся числа до того момента, пока не будет введён 0. Требуется вычислить сумму введённых чисел.

Эту задачу мы решим с помощью цикла с постусловием.

```
program posled;
var
  s, n: integer;
begin
  writeln('Введите целые числа. Окончание ввода - 0');
  s := 0;
  repeat
    write('Введите число: ');
    readln(n);
    s := s + n;
  until (n = 0);
  writeln('Сумма равна: ', s);
end.
```

Задача 3. С клавиатуры вводится натуральное число. Посчитать количество цифр в нём.

```
program count;
var
  n, i: integer;
begin
  write('Введите натуральное число: ');
  readln(n);
  i := 0;
  while (n > 0) do
    begin
      i := i + 1;
      n := n div 10;
    end;
  writeln('Во введённом числе ', i, ' цифр');
end.
```

ФОРМУЛИРУЕМ ВЫВОД

Практически в любой задаче существуют повторяющиеся действия, которые реализуются при помощи циклических конструкций.

Циклические конструкции (циклы) состоят из трёх блоков: инициализации, тела цикла и условия. По расположению последних двух блоков циклы делятся на циклы с постусловием и с предусловием. В языке Паскаль они реализуются конструкциями `while ... do` и `repeat ... until`.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Расскажите, какие циклические конструкции вы теперь знаете. В чём их суть?
2. Введите приведённые в параграфе программы по вычислению факториала в систему PascalABC.NET и проверьте правильность их работы. Что выводят программы, если мы вводим в них число 0? Объясните.

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

ЦИКЛЫ С ПАРАМЕТРОМ

В принципе, двух описанных типов циклических конструкций достаточно для решения любых задач. В чём их сходство? Оба этих цикла выполняются **по условию**. Однако очень часто существуют алгоритмы, в которых нужно просто выполнить одни и те же действия определённое количество раз. Наша задача

с суммированием чисел от 1 до 100 является, кстати, одной из них. В подобных случаях применяются **циклы с параметром**.

На языке Паскаль подобный цикл описывается следующим образом:

```
for i := <начало> to <конец> do <оператор>;
```

или

```
for i := <начало> downto <конец> do <оператор>;
```

Здесь **for**, **to**, **downto** и **do** – ключевые слова, описывающие цикл, а переменная *i* – **параметр цикла**.

Приведённые варианты различаются тем, что в первом из них параметр изменяется *от меньшего значения к большему*, а во втором, наоборот, *от большего к меньшему*. На русский язык данный цикл переводится так: для *i*, изменяющегося от <начала> до <конца>, выполнить <оператор>.

И снова напишем нашу программу для вычисления суммы чисел от одного до ста, но уже с использованием цикла **for**. Приведём два варианта: с **to** и с **downto**.

```
program p3;
var
  i, s: integer;
begin
  // инициализация
  s := 0;
  for i := 1 to 100 do
    s := s + i;
  // вывод результата
  writeln('Сумма равна: ', s);
end.
```

```
program p4;
var
  i, s: integer;
begin
  // инициализация
  s := 0;
  for i := 100 downto 1 do
    s := s + i;
  // вывод результата
  writeln('Сумма равна: ', s);
end.
```

Посмотрите, в циклах с параметрами инициализация параметра цикла и условие как отдельные части отсутствуют, но они никуда не пропали, просто мы их «спрятали» в конструкцию **for**, и они там незримо присутствуют.

- Попробуйте найти в приведённых примерах инициализацию и условие.

Обе наши программы в качестве результата выведут число 5050.

СКОЛЬКО РАЗ РАБОТАЮТ ЦИКЛЫ

Это интересный вопрос.

Мы с вами уже рассмотрели все виды циклов, так давайте взглянем на них пристально по порядку.

Что будет, если в цикле с предусловием мы поставим заведомо ложное условие? Например, напишем вот такую конструкцию:

```
while (3 < 2) do writeln('Привет');
```

В такой цикл мы *никогда не войдём*. Аналогичную картину мы получим, если в цикле с параметром и ключевым словом **to** зададим стартовое значение, превышающее конечное:

```
for i := 10 to 1 do writeln('Привет');
```

Таким образом, мы можем сказать, что существуют циклы, которые **не выполняются ни разу**.

Но это не относится к циклу с постусловием. Отличительной особенностью этого цикла является то, что он выполняется **всегда**, как минимум, **один раз**.

Посмотрите на блок-схему цикла с постусловием (см. рис. 1.10). Мы сначала выполняем действия, а уже потом производим проверку. Даже если условие выхода сразу станет истинным и мы выйдем из цикла, то один шаг мы всё равно сделаем.

Посмотрим теперь на такой цикл:

```
while (1 < 2) do writeln('Привет');
```

Что вы можете про него сказать?

Это пример цикла, который *никогда не закончится*. Такие циклы называются **бесконечными**. Они приводят к тому, что программа **зацикливается**. Подобных ситуаций нужно всегда избегать.

- Подумайте, является ли программа, в которой есть бесконечные циклы, алгоритмом.

ПРИМЕРЫ ПРОГРАММ

Задача 4. Написать программы, вычисляющие факториал числа n , которое вводится с клавиатуры, используя циклические конструкции с параметром. Напомним, что факториал $n! = 1 \cdot 2 \cdot \dots \cdot (n - 1) \cdot n$.

Цикл с параметром и `to`:

```
program fact4;
var i, n, f: integer;
begin
  write('Введите число n: ');
  readln(n);
  f := 1;
  for i := 1 to n do f := f*i;
  writeln('Факториал ', n, ' равен ', f);
end.
```

Цикл с параметром и `downto`:

```
program fact5;
var i, n, f: integer;
begin
  write('Введите число n: ');
  readln(n);
  f := 1;
  for i := n downto 1 do
    f := f*i;
  writeln('Факториал ', n, ' равен ', f);
end.
```

ФОРМУЛИРУЕМ ВЫВОД

В дополнение к циклам с предусловиями и с постусловиями в языке Паскаль есть циклы с параметром, которые реализуются с помощью конструкции `for`.

Циклы могут не выполниться ни разу и быть бесконечными.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Приведите примеры бесконечных циклов.

2. Вычислите сумму $S(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} = \sum_{i=1}^n \frac{1}{i}$. При этом следует

учесть, что данная сумма – это вещественное число.

3. Дано натуральное чётное число N . Получите $S = 2 + 4 + 6 + \dots + N$. Подумайте, с помощью какого цикла – с условием или `for`, удобнее решать эту задачу.

§ 7. Отладка программ

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Вы собираете модель самолёта. «Ну, наконец, готово!» — думаете вы и вдруг замечаете, что у вас остались неиспользованные детали. «Вот досада! — восклицаете вы в сердцах. — Где же я сделал ошибку?»

- С какой проблемой вы столкнулись? Сформулируйте её. Сравните свою формулировку с авторской (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Вспомните свой опыт работы в интегрированной среде разработки программ. (§ 4)

Вспомните, как записываются циклические конструкции. (§ 5–6)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

ОШИБКИ В ПРОГРАММЕ

В любой программе могут появиться ошибки.

Ошибки делятся на две группы: **синтаксические и логические**. О синтаксических ошибках мы уже с вами говорили. У нас есть «умный» помощник – **интегрированная среда разработки**, которая отлично их находит и сообщает нам об этом. А вот с логическими ошибками дело обстоит гораздо сложнее.

Логической называется ошибка, при наличии которой программа является синтаксически правильной и даже работает, но выполняется некорректно, то есть не решает поставленной перед ней задачи: зацикливается или выдаёт неверный результат.

- Подумайте, является ли в таком случае наша программа алгоритмом.

В такой ситуации среда разработки не обратит наше внимание на ошибку. Что же делать?

Самый простой способ найти логическую ошибку заключается в следующем: нужно добавить в программу дополнительные строки, которые будут выводить на экран значения некоторых переменных. По выводимым значениям мы сможем проверить, где же ошибка. Такой способ называется **ручной отладкой**.

Добавим в задачу З подсчёта цифр во вводимом с клавиатуры числе из § 5–6 дополнительные строки (в учебнике выделены цветным фоном) для, как говорят программисты, вывода **отладочной информации**.

```

program count;
var
    n, i: integer;
begin
    write('Введите натуральное число: ');
    readln(n);
    i := 0;
    while (n > 0) do
        begin
            i := i + 1;
            n := n div 10;
            // вывод отладочной информации
            writeln(' n = ', n, 'i = ', i);
        end;
    writeln('Во введённом числе ', i, ' цифр');
end.

```

Выполним программу, введя число 12345. На экран будут выведены следующие строки:

Введите натуральное число: 12345

n = 1234 i = 1

n = 123 i = 2

n = 12 i = 3

n = 1 i = 4

n = 0 i = 5

Во введённом числе 5 цифр

То, что выделено цветным фоном, и есть отладочная информация. Мы добились своей цели. Теперь нам видно, как программа работала. Но что здесь явно бросается в глаза? Результаты выводились *неровно*. Попробуем решить эту проблему.

ФОРМАТИРОВАНИЕ ВЫВОДА

С понятием «форматирование» вы уже знакомы. Учителя математики постоянно просят вас писать цифры точно в клеточках, а учителя русского языка — писать аккуратно по линейкам. Работая в текстовом редакторе, вы тоже учились форматировать текст. Наш учебник тоже отформатирован в едином стиле.

В программировании **форматирование вывода** — это такой вывод данных, при котором выводимый текст располагается строго определённым образом. Программист имеет возможность управлять форматированием вывода.

Форматирование вывода осуществляется в операторе `write` или `writeln`.

При выводе целых чисел мы можем написать `writeln(a:10)`. Это означает, что для вывода переменной `a` определяется поле в 10 позиций, а число будет прижато к правой границе поля вывода.

Давайте посмотрим, как изменится вывод нашей программы с отладочной информацией, если мы будем использовать форматированный вывод.

Программа	Вывод
<pre>program count; var n, i: integer; begin write('Введите натуральное число: '); readln(n); i := 0; while (n > 0) do begin i := i + 1; n := n div 10; // вывод отладочной // информации writeln('n = ', n:6, ' i = ', i); end; writeln('Во введённом числе ', i, ' цифр'); end.</pre>	<pre>Синтаксическая ошибка Ведите натуральное число: 12345 n = 12345 i = 1 n = 123 i = 2 n = 12 i = 3 n = 1 i = 4 n = 0 i = 5 Во введённом числе 5 цифр</pre>

Вот теперь всё чётко!

ФОРМАТИРОВАНИЕ ВЫВОДА ВЕЩЕСТВЕННЫХ ЧИСЕЛ

Рассмотрим пример.

Программа	Вывод
<pre>program format_1; var f, g: real; begin f := 1.5; g := 2/3; writeln(f); writeln(g); end.</pre>	<pre>1.5 0.6666666666666667</pre>

Мы видим, что число 1.5 выводится так, как нужно, а вот для 2/3 система сама решает, сколько знаков после точки будет показано. Можем ли мы управлять и таким выводом? Конечно, можем.

Форматированный вывод вещественных чисел очень похож на форматированный вывод целых чисел. Вывод вещественных чисел мы записываем в такой форме: `writeln(a:8:3)`. Это означает, что для вывода переменной *a* определяется 8 позиций, под дробную часть числа отводится 3 позиции, число будет прижато к правой границе поля вывода и корректно округлено.

Посмотрим, что получится, если в наш пример добавить форматирование.

Программа	Вывод
<pre>program format_2; var f, g: real; begin f := 1.5; g := 2/3; writeln(f:8:3); writeln(g:8:3); end.</pre>	<pre>1.500 0.667</pre>

Символ «.», который является разделителем целой и дробной частей, а также знак отрицательного числа учитываются в общем количестве выводимых символов. Если программист ошибся и указал невозможные параметры форматирования, то система попытается сама поправить ситуацию.

ИСПОЛЬЗОВАНИЕ ОТЛАДЧИКА

Использование ручной отладки не очень удобно и применяется всё реже и реже. Практически любая интегрированная среда программирования имеет в своём составе специальный компонент — **отладчик**, задача которого — предоставлять программисту удобный способ поиска логических ошибок в программе.

Суть работы любого отладчика заключается в следующем.

- 1) Программист может поставить так называемую **точку остановки** — указать место в программе, где она должна остановить свою работу.
- 2) Программист имеет возможность производить выполнение программы **по шагам** и просматривать значения каждой используемой в программе переменной.
- 3) Программист может остановить выполнение программы.

Посмотрим, как работает отладчик в среде *PascalABC.NET*.

На рис. 1.11 показана рабочая среда, в которой уже введена наша программа, вычисляющая количество цифр в числе. Для того чтобы поставить **точку остановки**, необходимо щёлкнуть мышью напротив нужной строки в

серой области слева от текста программы. При этом нужная строка выделится красным, а в левой области появится шарик (рис. 1.12).

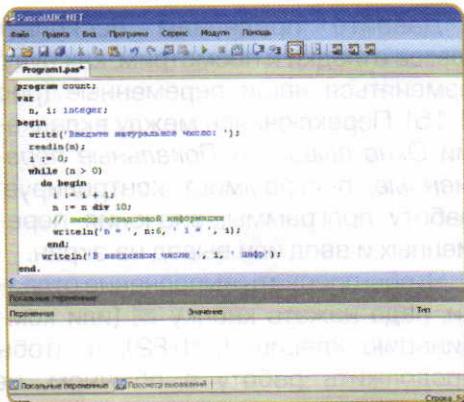


Рис. 1.11

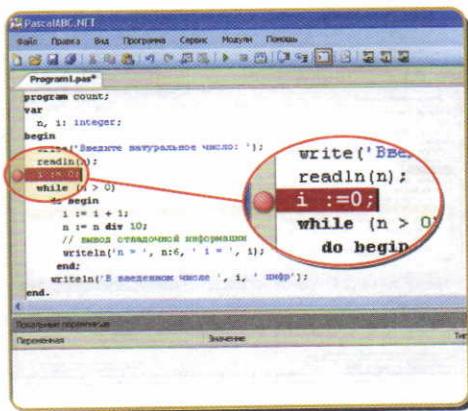


Рис. 1.12

Теперь если мы запустим программу, нажав кнопку , то до оператора `i := 0` она будет выполняться автоматически, а достигнув этого оператора, остановится. Страна выделится жёлтым, а в серой зоне появится стрелка (рис. 1.13).

Посмотрите на вкладки внизу окна среды — *Окно вывода* и *Локальные переменные*. Если вы их не видите, то необходимо их включить, используя меню *Вид* (рис. 1.14).

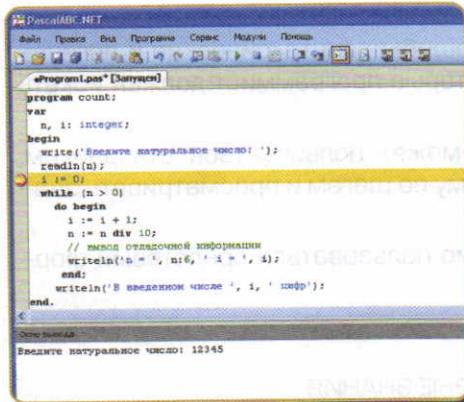


Рис. 1.13

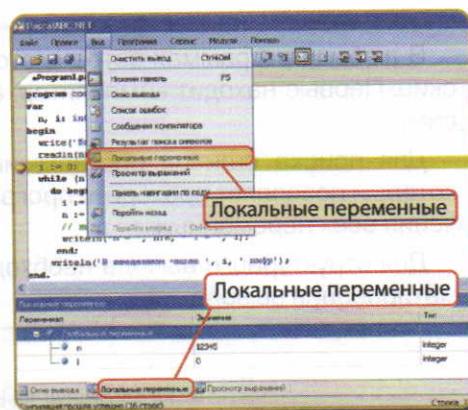


Рис. 1.14

Выбрав вкладку *Локальные переменные*, вы увидите значения всех переменных, которые вы используете в своей программе (см. рис. 1.14).

В нашем примере мы ввели число 12345 и занесли его в переменную `n`.

Нажимая клавишу F8 или кнопку , мы можем идти по программе по шагам.

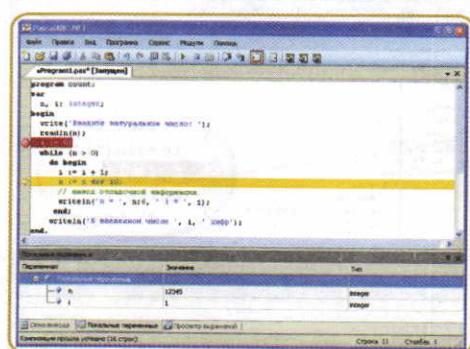


Рис. 1.15

Давайте проделаем несколько шагов отладки и посмотрим, как будут изменяться наши переменные (рис. 1.15). Переключаясь между вкладками **Окно вывода** и **Локальные переменные**, программист контролирует работу программы, значения переменных и ввод или вывод на экран.

Чтобы прервать выполнение отладки, надо нажать кнопку (или комбинацию клавиш Ctrl+F2), а чтобы продолжить работу в обычном, не отладочном, режиме — кнопку (или клавишу F9).

- Проведите отладку программы до конца.

Согласитесь, что с помощью отладки искать ошибки в программах очень удобно.

ФОРМУЛИРУЕМ ВЫВОД

В любых программах встречаются ошибки: синтаксические и логические. Первые находит компилятор, а вторые программист должен искать сам.

Для поиска ошибок программист может пользоваться отладчиком, который позволяет проходить программу по шагам и просматривать значения всех переменных.

Для структурного вывода необходимо пользоваться средствами форматирования вывода.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Объясните, что такое логическая ошибка и как можно её исправить.
2. Подумайте и скажите, что будет выведено на экран в результате работы следующей программы.

```

program test;
var
  a: real;
begin
  a := 2.48;
  writeln(a:5:2);
  writeln(a:5:3);
  writeln(a:3:3);
  writeln(a:5:1);
end.

```

3. Программисту поставили задачу написать программу, которая по введённому натуральному числу n должна выводить через пробел все нечётные числа из диапазона от 1 до n . Программист торопился и написал следующую программу:

```

program error;
var
  n, i: integer;
begin
  write('Введите натуральное число n: ');
  readln(n);
  for i := 1 to n do
    if (i mod 2 = 0) then write(i);
  writeln;
end.

```

Ведите эту программу в среду *PascalABC.NET*, с помощью отладки найдите и исправьте допущенные ошибки.

4. Дано натуральное число N ($N > 99$). Напишите программу, определяющую, сколько в этом числе сотен.

§ 8–9. Массивы

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Шерлок Холмс разбирался в своих бумагах. Среди множества писем он искал то, которое ему было столь необходимо сегодня. Он последовательно брал письмо из папки, читал его, и если текст письма его не устраивал, то он выкидывал письмо в камин. Наконец он нашел нужное письмо. Но тут Холмс опомнился: «Что же я наделал! Письмо–то я нашёл, а остальные документы не сохранил!»

Что надо было делать известному сыщику, чтобы не потерять временно не нужные ему письма?

- Сформулируйте основной вопрос урока. Сравните свой вариант с авторским (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Вспомните свой опыт работы в интегрированной среде разработки программ. (§ 4)

Как в программах используются циклы? (§ 5–6)

Что вы знаете об отладке программ. (§ 7)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Рассмотрим задачу: с клавиатуры вводятся целые числа до нуля. Найти среди них максимальное и минимальное числа.

Алгоритм решения этой задачи заключается в следующем [блок–схема — на рис. 1.16]:

1. Начало.
2. Ввести первое число. Считать его одновременно и максимумом, и минимумом (оно действительно является и максимумом, и минимумом для последовательности из одного числа).
3. Ввести следующее число. Если оно равно нулю, то вывести максимум и минимум и завершить работу алгоритма — перейти на шаг № 7.
4. В противном случае сначала сравнить нововведённое число с имеющимся у нас максимумом. Если оно больше максимума, то изменить значение максимума.
5. В противном случае, если оно меньше минимума, то изменить значение минимума.

6. Вернуться на шаг № 3.

7. Конец.

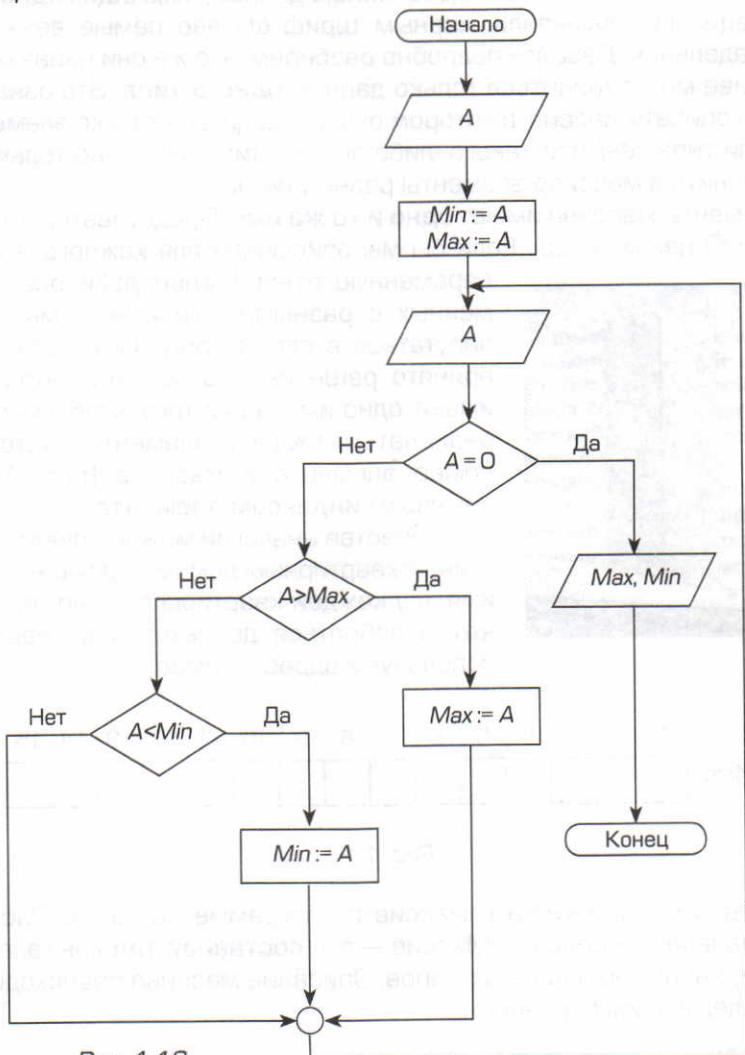


Рис. 1.16

В результате работы этого алгоритма в переменных Max и Min будут находиться искомые значения. Но теперь мы заинтересовались: среди каких чисел мы искали максимум и минимум? На этот вопрос мы ответить не сможем. Мы не сохранили все введённые числа, кроме последнего.

Для того чтобы иметь возможность выполнять различные действия с введёнными однотипными данными, нужно их сохранять. Для этого нам понадобятся массивы.

ЧТО ТАКОЕ МАССИВЫ

Массив — это совокупность **однотипных** данных, имеющих **одно имя**.

Мы специально выделили жирным шрифтом две самые важные части этого определения. Давайте подробно разберём, что же они означают.

В массиве могут храниться только данные **одного типа**. Это означает, что мы можем описать массив, в котором будут находиться только элементы типа *integer*, или типа *real*, или какого-либо другого типа языка, но только одного. Нельзя хранить в массиве элементы разных типов.

Все элементы массива имеют **одно и то же имя**. Представьте, что нам надо хранить 100 целых чисел. Если бы мы описывали для каждого числа свою

переменную, то нам бы понадобилось 100 переменных с разными именами, и мы бы могли запутаться в своей программе. Поэтому было принято решение, что все элементы массива имеют одно имя, а для того, чтобы мы могли их различать, каждому элементу даётся ещё и номер элемента в массиве (рис. 1.17). Его называют **индексом элемента**.

В качестве аналогии можно привести пример с многоквартирным домом. У дома есть адрес — имя, а у каждой квартиры есть номер. Поэтому чтобы добраться до некоторой квартиры, мы используем адрес + номер.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Имя																

Рис. 1.17

Чтобы начать использовать массив в программе на языке Паскаль, мы должны сначала его описать. Массив — это **составной тип** языка, в отличие от простых, например числовых, типов. Описание массива происходит в разделе **var** следующим образом:

M: array [1..100] of integer;

Здесь мы описали переменную *M*, представляющую собой массив [об этом говорит ключевое слово *array*], содержащий 100 элементов типа *integer*, индексы которых 1, 2, 3 и т. д. до 100 (это указано в квадратных скобках).

Вспомните, что мы говорили о переменных числовых типов — они обозначают области памяти компьютера, куда можно заносить значения. То же самое верно и для переменной типа массив: в элементы массива заносятся **значения элементов**.

Чтобы получить доступ, например, к пятому элементу массива, мы должны использовать конструкцию:

```
a := M[5];
M[5] := 50;
```

Первый оператор присваивания означает, что в переменную *a* заносится значение 5-го элемента массива *M*.

Во втором операторе присваивания в 5-й элемент массива *M* записывается значение 50.

Если мы описали массив для хранения 100 элементов, то мы не можем обратиться к 101-му элементу. Это вызовет ошибку. Обращайте на это внимание.

ВВОД И ВЫВОД МАССИВОВ

Ввод по признаку

В качестве первой задачи по работе с массивами рассмотрим пример того, как вводить элементы массива до нуля.

```
program input1;
var
  M: array [1..100] of integer;
  a, i: integer;
begin
  writeln('Введите целые числа до 0');
  // заполним массив. 0 в массив не записываем
  readln(a);
  i := 1;
  while (a <> 0) do
    begin
      M[i] := a;
      readln(a);
      i := i + 1;
    end;
  writeln('Введено ', i - 1, ' элементов');
end.
```

Нам понадобились две целые переменные *a* и *i*. Переменная *a* используется для ввода значений и проверки введённого числа на равенство нулю, а переменная *i* служит для индексации (адресации) элементов массива, а заодно и как их счётчик.

Индексация — это определение индекса (номера) элемента массива.

Итак, в нашей программе мы сначала выводим информационную строку, предупреждающую пользователя о том, что число 0 — это признак окончания ввода. После этого мы вводим значение с клавиатуры и заносим его в переменную *a*. Переменная *i* принимает значение 1. Далее начинается цикл, условием продолжения которого является не равенство *a* нулю. Соответственно, если первое введённое число — это 0, то в цикл мы не попадём.

Далее в цикле мы сохраняем введённое значение в *M[i]* — в *i*-том элементе массива, вводим новое число *a* и увеличиваем *i* на 1.

Цикл закончится, как только мы введём число 0.

Обратите внимание, что после окончания цикла в переменной *i* находится значение, на единицу большее количества введённых элементов.

- Подумайте, почему так происходит.

В результате мы выведем значение *i* – 1, как количество введённых в массив значений.

Ввод по количеству

В этом примере мы вначале введём количество элементов массива, которые надо заполнять (не больше 100), а после этого заполним массив.

```
program input2;
var
  M: array [1..100] of integer;
  n, i: integer;
begin
  write('Введите количество элементов: ');
  readln(n);
  for i := 1 to n do readln(M[i]);
  writeln('Введено ', i, ' элементов');
end.
```

В этом примере мы сначала вводим переменную *n* — количество заполняемых элементов, а затем в цикле с параметром вводим значения сразу в элементы массива *M[i]* (без использования вспомогательной переменной).

Вывод массивов

Давайте рассмотрим способы вывода элементов массива. В примере на стр. 57 слева каждый элемент выводится в отдельной строке, в примере справа элементы выводятся в одной строке через пробел (перевод строки происходит после окончания вывода).

```

program output1;
var
  M: array [1..100] of integer;
  n, i: integer;
begin
  write('Введите количество
        элементов массива: ');
  readln(n);
  for i := 1 to n do readln(M[i]);
  writeln('Введённый массив');
  for i := 1 to n do writeln(M[i]);
end.

```

```

program output2;
var
  M: array [1..100] of integer;
  n, i: integer;
begin
  write('Введите количество
        элементов массива: ');
  readln(n);
  for i := 1 to n do readln(M[i]);
  write('Введённый массив: ');
  for i := 1 to n do write(M[i], ' ');
  writeln;
end.

```

ФОРМУЛИРУЕМ ВЫВОД

Если для решения задачи требуется хранение множества однотипных данных, то для этого применяются массивы.

Массив — это совокупность однотипных данных, имеющих одно имя.

Для описания массива применяется следующая конструкция

M: array [1..100] of integer;

Для доступа к *i*-му элементу массива указывается имя массива и в квадратных скобках индекс элемента — *M[i]*.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Что такое массив? Объясните это понятие.
2. Какие самые важные свойства массива вы знаете?
3. Какие способы ввода и вывода значений элементов массива вы знаете?
4. Проверьте в среде разработки программ *PascalABC.NET* работу приведённых в параграфе примеров. Измените примеры так, чтобы массив содержал вещественные числа.

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

РАБОТА С МАССИВАМИ

Давайте разберём несколько задач на работу с массивами.

Примечание: в записи программ цветным фоном выделены моменты, на которые надо обратить внимание.

Задача 1. С клавиатуры вводится число n . Заполнить массив первыми n чётными числами 2, 4, 6, 8, ... и вывести его. В массиве может находиться не более 100 элементов.

```
program m1;
var
  i, n: integer;
  M: array [1..100] of integer;
begin
  write('Введите количество элементов массива: ');
  readln(n);
  // заполнение массива
  for i := 1 to n do M[i] := 2 * i;
  // вывод массива
  for i := 1 to n do write(M[i], ' ');
  writeln;
end.
```

Задача 2. С клавиатуры вводится число n , а за ним n элементов массива. Вычислить среднее арифметическое элементов массива. В массиве может находиться не более 100 элементов.

```
program m2;
var
  i, n, sum: integer;
  M: array [1..100] of integer;
begin
  write('Введите количество элементов массива: ');
  readln(n);
  // заполнение массива
  writeln('Введите элементы');
  for i := 1 to n do
    begin
      write('M[', i, '] = ');
      readln(M[i]);
    end;
  // расчёт суммы элементов массива
  sum := 0;
  for i := 1 to n do sum := sum + M[i];
  writeln('Среднее арифметическое элементов = ', sum / n);
end.
```

Обратите внимание, что программу можно упростить, убрав из неё второй цикл (в котором считается сумма). Мы можем считать сумму во время ввода. Тогда программа будет выглядеть следующим образом.

```
program m2_1;
var
    i, n, sum: integer;
    M: array [1..100] of integer;
begin
    write('Введите количество элементов массива: ');
    readln(n);
    // заполнение массива и расчёт суммы элементов массива
    sum := 0;
    writeln('Введите элементы');
    for i := 1 to n do
        begin
            write('M[', i, '] = ');
            readln(M[i]);
            sum := sum + M[i];
        end;
    writeln('Среднее арифметическое элементов = ', sum / n);
end.
```

Задача 3. Массив заполняется до нуля, при этом нуль в массив не записывается. Необходимо посчитать и вывести сумму элементов данного массива с нечётными значениями.

```
program m3;
var
    a, i, n, sum: integer;
    M: array [1..100] of integer;
begin
    writeln('Введите элементы массива до нуля: ');
    // заполнение массива
    readln(a);
    i := 1;
    while (a <> 0) do
        begin
            M[i] := a;
            i := i + 1;
            readln(a);
        end;
    n := i - 1;
```

```

program sum;
var n, i: integer;
    M: array [1..100] of integer;
begin
    sum := 0;
    if (n > 0) then
        begin
            for i := 1 to n do
                if (M[i] mod 2 <> 0) then sum := sum + M[i];
            writeln('Сумма нечётных значений: ', sum);
        end
    else
        writeln('В массиве значений нет');
end.

```

Задача 4. С клавиатуры вводится число n , а за ним n элементов массива. Переставить все значения элементов массива в обратном порядке.

Эта задача решается попарной перестановкой значений элементов массива: первого с последним, второго с предпоследним и т. д. Сколько всего будет таких пар? Если в массиве находится n элементов, то в нём будет ровно $n \text{ div } 2$ пар.

- Проверьте это утверждение.

Давайте подумаем, если у нас есть две переменные a и b , и нам нужно поменять между собой их значения, то как это можно сделать? Для этого нам понадобится **вспомогательная переменная** $temp$ (дадим переменной такое имя от английского слова temporary — временная). Задача решается в три шага:

```

temp := a;
a := b;
b := temp;

```

Сначала мы сохраняем значение переменной a в $temp$. Затем заносим в a значение b . Выполняя эту операцию, мы уничтожаем старое значение a . Но не волнуйтесь! Мы же сохранили его в $temp$! Поэтому теперь мы спокойно можем занести значение $temp$ в b .

Итак, текст нашей программы перестановки элементов будет следующим.

```

program m4;
var
    i, n, temp: integer;
    M: array [1..100] of integer;
begin
    write('Введите количество элементов массива: ');
    readln(n);
    // заполнение массива
    writeln('Введите элементы');

```

```

for i := 1 to n do
begin
  write('M[', i, '] = ');
  readln(M[i]);
end;
for i := 1 to n div 2 do
begin
  temp := M[i];
  M[i] := M[n - i + 1];
  M[n - i + 1] := temp;
end;
write('Перевёрнутый массив: ');
for i := 1 to n do write(M[i], ' ');
writeln;
end.

```

Обратите внимание на то, как определяются парные элементы: $M[i]$ и $M[n - i + 1]$.

Теперь мы готовы самостоятельно решать задачи, связанные с массивами.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Проверьте в среде разработки программ *PascalABC.NET* работу приведённых в параграфе примеров.
2. С клавиатуры вводится число n . Заполнить массив с клавиатуры n значениями и вывести их в обратном порядке.
3. С клавиатуры вводится число n . Заполнить n элементов массива числами 2, 4, 8, 16... и вывести массив. В массиве может находиться не более 100 элементов.
4. С клавиатуры вводится число n . Заполнить n элементов массива числами 1, 2, 4, 7, 11... и вывести его. В массиве может находиться не более 100 элементов.
5. Значения элементов массива вводятся до нуля. Посчитайте произведение значений его элементов.

Проверь себя

Задание 1

С клавиатуры вводятся целые числа a , b и c . Если $a \cdot b$ больше, чем $c^2 - b$, тогда выведите $a + b$, в противном случае выведите $b + c$.

Задание 2

С клавиатуры вводится натуральное число N . Выясните, является ли оно квадратом какого-то натурального числа. Если является, то выведите это число, если нет, то вывести -1 . Например, 4 является квадратом 2, а 17 не является квадратом никакого числа.

Задание 3

С клавиатуры вводится число N , за ним массив A из N элементов. Найдите номер первого по счёту элемента массива с нечётным значением. Известно, что в массиве точно будет хотя бы один такой элемент.

ПРИМЕНЯЕМ ЗНАНИЯ (необходимый уровень)

1. С клавиатуры вводятся два вещественных числа — длины сторон прямоугольника. Вычислите площадь прямоугольника и выведите её на экран. Вывод должен быть отформатирован следующим образом: под число отводится 10 знаков, а под дробную часть — 3 знака.
2. С клавиатуры вводятся 3 целых числа. Определите количество положительных чисел в этом наборе.
3. С клавиатуры вводится четырёхзначное число. Получите число, равное произведению его цифр.
4. С клавиатуры вводится натуральное число n . Определите количество его цифр.
5. С клавиатуры вводится натуральное число n . Вычислите сумму $S = 1 + 2 + \dots + n$.
6. Даны целые положительные числа N и K . Используя только операции сложения и вычитания, найдите частное от деления нацело N на K , а также остаток от этого деления.
7. С клавиатуры вводится натуральное число N . За ним последовательно вводятся данные о росте N учащихся класса. Определите средний рост учащихся всего класса. Попробуйте решить эту задачу без использования массивов.
8. Заполните элементы массива последовательностью чисел: 2, 5, 10, 17, 26, ...
9. Вводится число N , а за ним N целых чисел — значений элементов массива. Найдите количество элементов с отрицательными значениями в этом массиве.
10. С клавиатуры вводится число N . Выведите следующую фигуру (N строк, в последней строке N звёздочек). Подсказка: для решения этой задачи примените конструкцию «цикл в цикле».

*

* * *

* * * *

* * * * *

...

* * * * * * *
11. Массив заполняется целыми числами до нуля. Замените все положительные числа на противоположные, а отрицательные возведите в квадрат. Выведите изменившийся массив.

12. На вход программе поступает целое число N . Выведите количество делителей N , включая 1 и само число N . Внимание: для решения этой задачи заведите счётчик и не забудьте сначала занести в него значение 0.

ПРИМЕНЯЕМ ЗНАНИЯ (повышенный уровень)

- С клавиатуры вводятся два вещественных числа. Выведите через запятую в одной строке сначала меньшее из них, а потом большее.
- Помогите летописцу быстро вычислять век (столетие) по году. С клавиатуры вводится Y — некоторый год. Определите по нему число C — номер его столетия (необходимо учесть, что, например, началом XX столетия был 1901 год, а не 1900).
- Дано трёхзначное число. В нём удалили самую правую цифру и приписали её слева. Получите итоговое число и выведите его. В этой задаче нужно получить именно новое число, а не вывести на экран последнюю цифру числа, а затем первые две.
- Последовательность целых чисел вводится с клавиатуры до нуля. Найдите среди них максимум и минимум и выведите на экран их сумму.
- Обозначим дни недели числами от 1 (понедельник) до 7 (воскресенье) соответственно. Известно, что первое число текущего месяца попадает на понедельник. На вход программе подаётся целое число $1 < n < 31$. Выведите номер дня недели числа n . Подсказка: это задача на деление.
- Начальный вклад в банке равен 1000 руб. Через каждый месяц размер вклада увеличивается на P процентов от имеющейся суммы (P — вещественное число, $0 < P < 25$). По данному P определите, через сколько месяцев размер вклада превысит 1500 руб., и выведите найденное количество месяцев K (целое число) и итоговый размер вклада S (вещественное число). Для вывода вещественных чисел применяйте формат: 10 знаков под число, 4 знака под дробную часть.
- Дано натуральное число N . Переверните его. $12345 \rightarrow 54321$. Подсказка: вспомните, как «вытаскивать» цифры из числа.
- Вводится число N , а за ним N целых чисел — значений элементов массива. Обнулите все отрицательные значения элементов и посчитайте количество остальных. Выведите обновлённый массив и количество неотрицательных элементов на экран.
- Значения элементов целочисленного массива вводятся до нуля. Найдите среднее арифметическое всех значений элементов с нечётными значениями.

10. С клавиатуры вводится число N . Вычислите элементы последовательности Фибоначчи и выведите её «звёздочками»:

```

    *
    *
  * *
  * * *
  * * * *
  * * * * *

```

Последовательность Фибоначчи имеет вид: 1, 1, 2, 3, 5, 8, 13, ..., то есть первые два элемента равны 1, а каждый следующий, начиная с третьего, вычисляется как сумма двух предыдущих.

11. С клавиатуры вводится натуральное число $M < 27$. Найдите и выведите на экран все трёхзначные натуральные числа, сумма цифр которых равна M .

12. С клавиатуры вводится число N . Заполните массив S вещественных

чисел из N элементов следующим образом: $S[1] = \frac{1^2}{1!}$; $S[2] = \frac{1^2}{1!} + \frac{2^2}{2!}$;

$S[3] = \frac{1^2}{1!} + \frac{2^2}{2!} + \frac{3^2}{3!}$ и т. д. до $S[n] = \frac{1^2}{1!} + \frac{2^2}{2!} + \frac{3^2}{3!} + \dots + \frac{n^2}{n!}$. Выведите массив

на экран. Для вывода элементов выделите 8 позиций, а для дробной части — 3 позиции.

ПРИМЕНЯЕМ ЗНАНИЯ (максимальный уровень)

- С клавиатуры вводится натуральное четырёхзначное число N . Выведите количество чётных цифр этого числа.
- Винни-Пух узнал, что со времени последнего завтрака прошло K секунд. А сколько часов и минут прошло с этого момента?
- Единица товара стоит a рублей и b копеек. Было куплено n штук этого товара. Сколько рублей и копеек пришлось заплатить за всю покупку? На вход программе подаются три целых числа:
 $0 < a < 30\ 000$, $0 < b < 100$ и $0 < n < 30\ 000$.
 Выведите два искомых числа.
- Дано натуральное число N . Если сложить все цифры числа, затем все цифры найденной суммы и повторять этот процесс до тех пор, пока не будет получено однозначное число (цифра), то эта цифра будет цифровым корнем исходного натурального числа. Найдите цифровой корень числа N .

5. Обозначим дни недели числами от 1 (понедельник) до 7 (воскресенье) соответственно. По известному m — номеру дня недели первого числа текущего месяца определите день недели числа n . На вход программе подаются 2 целых числа $1 < n < 31$, $1 < m < 7$. Подсказка: это задача на деление.
6. Вкладчик положил на банковский счёт n рублей. Каждый год на сумму вклада начисляется k процентов годовых (будем считать, что процент всегда округляется до целого числа рублей по формуле $[x \cdot k / 100]$, где x — сумма вклада на начало года). Начисленные проценты добавляются к сумме вклада. Через сколько лет сумма вклада станет не менее m рублей? На вход программе подаются три натуральных числа: $n < 10^6$, $k < 100$, $m < 1000 \cdot n$. Выведите одно число — искомое количество лет.
7. Вычислите n -й элемент последовательности Фибоначчи. Последовательность Фибоначчи имеет вид: 1, 1, 2, 3, 5, 8, 13, ..., то есть первые два элемента равны 1, а каждый следующий, начиная с третьего, вычисляется как сумма двух предыдущих.
8. С клавиатуры вводится число n . Заполните массив первыми n элементами последовательности Фибоначчи {1, 1, 2, 3, 5, 8, ...} и выведите его. В решении этой задачи вам поможет предыдущая задача.
9. Вводится целое число N . Определите, является ли оно факториалом какого-либо числа. Например, для $N = 720$ программа должна вывести 6, для 15 — фразу «такого числа нет», а для 1 вывести два числа: 0 и 1.
10. С клавиатуры вводится число N , а за ним N элементов массива. Сдвиньте циклически элементы массива вправо на один элемент. Пример циклического сдвига: 1, 2, 3, 4, 5 → 5, 1, 2, 3, 4.
11. С клавиатуры вводится массив до нуля. Найдите в нём максимальный и минимальный элементы и поменяйте их местами. Выведите результат.
12. С клавиатуры вводится число n . Заполните массив первыми n простыми числами {2, 3, 5, 7, ...} и выведите его. Подсказка: 1 не является простым числом. Не забудьте про конструкции «цикл в цикле».

Итоговая проверочная работа

Уровень 1

- С клавиатуры вводится четырёхзначное число. Выведите сначала его 2-ю цифру, потом 4-ю, потом 1-ю и в конце 3-ю. Вывод осуществите в одной строке, так чтобы создалось впечатление, что вы вывели целиком число.
- С клавиатуры вводится число N , а за ним N целых значений элементов массива. Посчитайте количество чётных чисел, стоящих на нечётных местах в массиве.

Уровень 2

- С клавиатуры вводится число. Проверьте, делится ли оно на 3. (Вспомните признак делимости на 3.) Если число делится на 3, то выведите фразу «Да», в противном случае выведите «Нет».
- С клавиатуры вводится число N , а за ним в массив вводятся N целых чисел. Проверьте, все ли элементы массива положительные. Если да, то выведите фразу «Да», в противном случае выведите «Нет».

Уровень 3

- С клавиатуры вводится натуральное число N . Найдите и выведите самую большую цифру этого числа.
- Массив целых чисел вводится до нуля. Затем вводится число N . Определите, какое количество раз число N встречается в массиве.



БАЗЕШНГ

Модуль 2. Основы дизайна и печати изображений

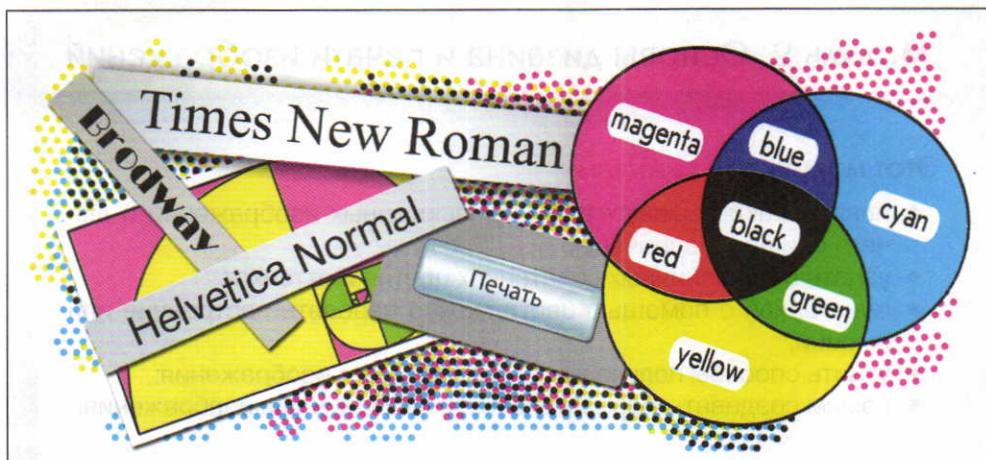
Этот модуль поможет вам:

- понять, какие бывают виды компьютерных изображений и для чего они предназначены;
- узнать о гармоничных сочетаниях цветов и оттенков;
- понять, как с помощью цвета можно передать настроение или эмоции;
- узнать способы получения выразительного изображения;
- самим создавать выразительные и гармоничные изображения.

Для этого вам надо научиться:

- создавать растровые изображения в соответствии с поставленными целями;
- сохранять полученные изображения в соответствии с поставленными целями;
- использовать основные правила композиции и цвета;
- соединять несколько изображений, получая фотоколлаж.

Введение



Человеку с рождения даны чувства гармонии, ритма, равновесия, дано чувство прекрасного. Человек быстро воспринимает яркие визуальные образы, ему необходимо постоянно пополнять свой запас образов. Образы помогают ему ориентироваться в окружающей действительности. Именно поэтому существуют такие профессии, как режиссёр, композитор, писатель, художник. Представители этих профессий – люди, создающие образы.

Но вовсе не обязательно быть художником, чтобы создавать простые иллюстрации, писать выразительные объявления или делать в подарок своим близким открытки. Всё это вы можете сделать сами. Но для создания интересных и выразительных изображений необходимо знать основы композиции, элементарные правила цветовых сочетаний. Взявшись за создание графической композиции, необходимо не только изучить команды и инструменты графического редактора, но и постараться понять основные принципы и закономерности построения изображений. Эти знания доступны всем, и их можно применять, даже не умея рисовать, «как художник».

Если посмотреть на историю развития искусства, то мы увидим непрерывный поток сменяющих друг друга направлений, стилей, течений. Компьютерный дизайн – это современная часть визуальной культуры, и мы можем говорить о течениях, направлениях, стилях и моде, которые возникают на наших глазах, но ещё не стали историей. Компьютерный дизайн использует основы и нормы современного искусства и искусства прошлых веков, развивается, вносит в искусство свои специфические особенности. В современном искусстве используются новые технологии, в том числе компьютерные.

§ 1. Основы композиции изображения

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА

Посмотрите на рис. 2.1.

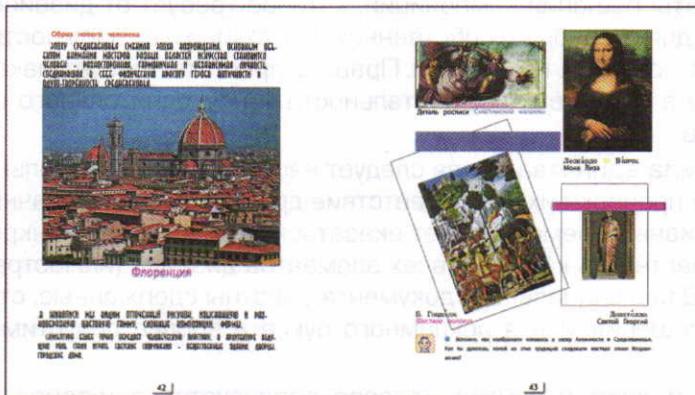


Рис. 2.1

Что хотели сказать дизайнеры этих страниц?

- Сформулируйте тему урока. Сравните свою формулировку с авторской (с. 142 учебника).

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Композиция в графическом дизайне – объединение различных элементов (изображений, текста и декоративных деталей) в единое целое. В композиции каждый элемент подчиняется общей художественной (образной) идеи.

Удачное дизайнерское решение невозможно без соблюдения принципов композиции. К принципам, которыми следует руководствоваться, относятся принцип единственного контраста, единство стиля, нюанс и ритм.

Контраст в композиции — это резкое различие однотипных элементов. Не считается контрастом круглое и красное, хотя между ними существует различие. Контраст может основываться на различии:

- форм (большой и маленькой, длинной и короткой, квадратной и круглой);
- направления (вертикального и горизонтального, прямого и наклонного);
- цвета (чёрного и белого, цветного и серого, светлого и тёмного, «тёплого» и «холодного») и пр.

Дизайн, как правило, должен определяться единственным контрастом. Например, если основной контраст строится на крупном и мелком шрифте, то не стоит добавлять контраст по цвету. Контраст эффективен только тогда, когда он ясно читается. А множество контрастов взаимно «гасят» друг друга.

Из **принципа единственного контраста** следует достаточно трудное **правило простоты** решения композиции, которое требует от дизайнера значительных сил для «борьбы» с собственной фантазией и возможностями современных компьютерных технологий. Правило простоты требует максимального ограничения в средствах выразительности и неукоснительного следования главной идеи.

Из **принципа единства стиля** следует необходимость тщательного отбора элементов и проверки их на соответствие друг другу. Очень удачный рисунок или декоративный элемент может оказаться неуместным в конкретной композиции. Уместность касается всех элементов дизайна (иллюстраций, цвета и шрифта). Для официального документа уместны сдержанные, строгие цветовые соотношения, а для рекламного буклета вполне допустимо цветовое «буйство».

Контраст требует развития, которое реализуется в **нюансе**. Например, если основной контраст строится на крупном [заголовок] и мелком [основной текст] шрифтах, то уместно использовать шрифт ещё одного размера, который слегка отличается от основного, для выделения важных абзацев. Он и будет служить нюансом, который подчеркнёт основной контраст и сохранит единство стиля.

Ритм задаёт темп и последовательность восприятия графического произведения, а также ожидаемые изменения. Максимальный ритмический рисунок присущ орнаментам, для которых характерно равномерное следование элементов друг за другом. Достаточно заметен также ритм строк и абзацев текста.

Дизайнеру следует проводить самый тщательный отбор деталей и **пристально следить за мелочами**. Коварство мелких деталей состоит в том, что незамеченная опечатка, неравномерные отступы и другие «мелочи» могут испортить в целом удачно решённую композицию.

Каждый элемент композиции имеет «визуальный вес», который создаётся сложным сочетанием размера, цвета и положения. Например, тёмный цвет «тяжелее» светлого, большой элемент «тяжелее» маленького элемента такого же цвета. Одним из способов **достижения равновесия** является симметричное расположение элементов. Но такая композиция считается не самой выразительной. Однаковым («равновесным»), как правило, делают соотношение элементов по горизонтальной линии.

Если композиция имеет визуальный центр, то, скорее всего, это должен быть **оптический центр листа** (на 1/8 выше физического центра).

Существуют ещё **композиционные точки** (рис. 2.2).

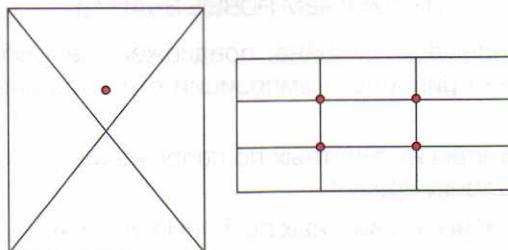


Рис. 2.2



Рис. 2.3

Посмотрите примеры обложек карманных календариков, сделанных учениками московского лицея № 1533 (информационных технологий), и определите, где находятся центры композиции (рис. 2.3).

Важно понимать, что следование принципам композиции не приводит автоматически к успеху. Создание удачной композиции требует большого практического опыта.

ФОРМУЛИРУЕМ ВЫВОД

Композиция – объединение различных элементов для образного решения какой-либо художественной идеи.

К основным принципам, которыми следует руководствоваться при создании композиции, относятся принцип единственного контраста, единство стиля, нюанс и ритм.

Основные принципы реализуются в виде множества правил, основными среди которых являются уместность элементов, используемых в композиции, простота общего решения, предельное внимание к деталям, достижение равновесия и пропорциональности.

Указанные принципы и правила не имеют единиц измерения и не могут быть объективно истолкованы. Создание удачной композиции требует большого практического опыта.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Рассмотрите примеры дизайна, предложенные учителем. Попробуйте определить, на каких принципах композиции строится дизайн каждого изображения.
2. Приведите примеры неуместных по содержанию элементов в композиции. Докажите свою точку зрения.
3. Приведите примеры неуместных по форме элементов в композиции. Докажите свою точку зрения.
4. Откройте новый документ в текстовом редакторе *Microsoft Word*. С помощью инструментов рисования фигур создайте простую композицию с контрастом по форме или по размеру. Обсудите с одноклассниками правильность вашего варианта.
5. Найдите фирменные знаки – логотипы – вокруг себя. Рассмотрите их с точки зрения принципов композиции.

§ 2. Цвет в графическом дизайне

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА

Рассмотрите три варианта обработки на компьютере картины И. И. Левитана «Золотая осень» (рис. 2.4).



Рис. 2.4

Чем они различаются, какие чувства и эмоции вызывает каждая из них?

- Сформулируйте основной вопрос урока. Сравните свой вариант с авторским (с. 142 учебника).

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Что означает цвет? Человек на протяжении всей своей истории пытался постичь его смысл. Он наделял цвет магической силой, придавал ему философский смысл, пытался понять его значение. Цвет становился символом. Красный – цвет борьбы, белый у одних народов означает смерть, а у других это символ чистоты.

Некоторое восприятие цвета у нас от рождения. Например, светлые цвета кажутся «лёгкими», а тёмные – «тяжёлыми». Это ощущает человек даже в раннем детстве: красный всем кажется «горячим» и «громким», а голубой – «холодным» и «тихим».

Цвет может вызывать эмоции: радость и уныние, печаль и восторг. Это качество цвета активно используется в проектировании интерьеров, моделировании одежды.

Цвет можно использовать для общего настроя композиции, чтобы читатели или зрители его ощутили ещё до внимательного рассмотрения или чтения.

Если цвет используется в композиции, то на первый план выходит проблема сочетания цветов. Художники называют это гармонией колорита.

Использование цвета в любом графическом документе вносит выразительность и привлекательность и способствует большему эффекту. Однако цвет должен применяться не сам по себе, а только как часть определённого цветового решения (обоснование выбора цвета тем, что он, например, «красивый красный», – очень распространённая ошибка).

Например, один (и не очень яркий) цвет, введённый в чёрно–белый документ, действует на внимание гораздо сильнее, чем пестрота из многих и ярких цветов.

Для помощи в подборе цветовых сочетаний существуют цветовые системы. Разберём лишь некоторые из них, наиболее часто встречающиеся.

Нет необходимости приводить примеры употребления **чёрно–белой гаммы** (рис. 2.5) в различных областях произведений современной культуры – они слишком многочисленны и очевидны. Эта цветовая система прочна и устойчива во времени и пространстве.

На рис. 2.6 показана первая в истории **трёхцветная система** – «первичная триада», которая в наши дни так же актуальна, как в любой момент истории.

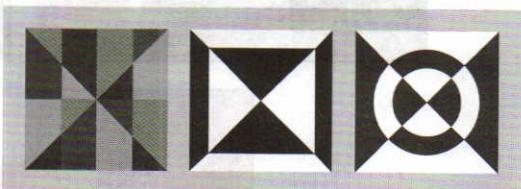


Рис. 2.5



Рис. 2.6

В **монохромной системе** доминирует какой–либо один хроматический цвет или его оттенки по цветовому тону, яркости или насыщенности (рис. 2.7). Хроматические цвета – это те цвета и их оттенки, которые мы различаем в спектре: от красного до фиолетового. Композицию могут дополнять оттенки серого, чёрный или белый цвет (рис. 2.8).



Рис. 2.7

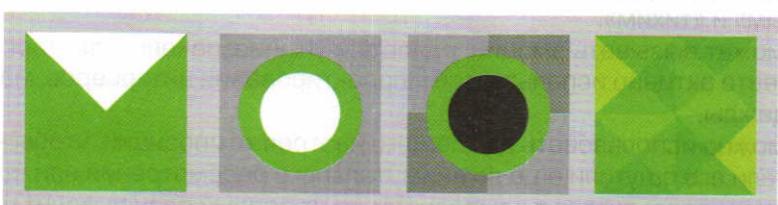


Рис. 2.8

Монохромия дает возможность сосредоточить внимание зрителя на какой-либо одной мысли, эмоции, чувстве, ассоциации. Наконец, если главным средством художника является форма, то ему нет необходимости в широкой палитре – ведь цвет вступает в конфликт с формой и может даже разрушить её.

- Рассмотрите картины известных художников (рис. 2.9). Как с помощью монохромного изображения они передают нам информацию?



Клод Моне. Снег в Аржантёе



Франиско де Сурбаран. Натюрморт

Рис. 2.9

Для нахождения других гармоничных сочетаний цвета воспользуемся **цветовым кругом** (рис. 2.10).

Цвета, расположенные на круге друг напротив друга, называются **дополнительными**. Дополнительные цвета при соблюдении правильного баланса позволяют сделать композицию более захватывающей и привлекательной (рис. 2.11).



Рис. 2.10

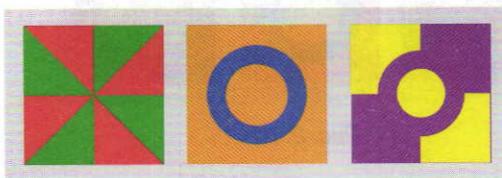
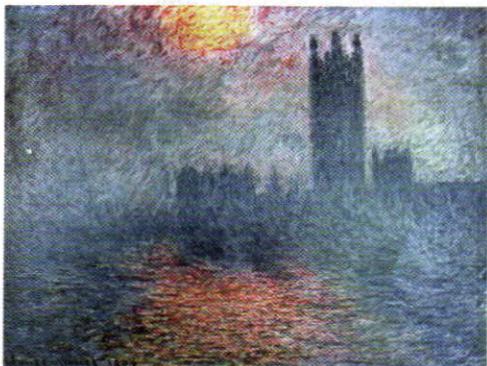


Рис. 2.11

В природе и искусстве существует множество полярных сочетаний хроматических цветов (рис. 2.12).



Клод Моне. Здание парламента
в Лондоне



Поль Гоген. Натюрморт для моего друга
Жакоба

Рис. 2.12

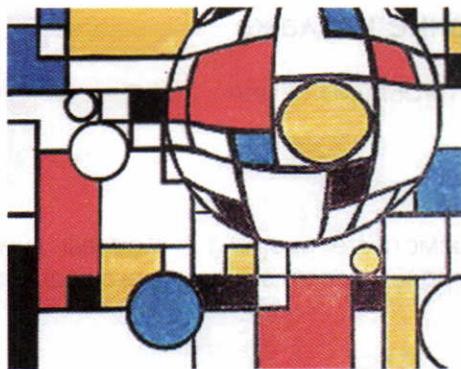
Хорошо сочетаются между собой три цвета, расположенные на равном расстоянии на цветовом круге [рис. 2.13 – 2.16].



Рис. 2.13

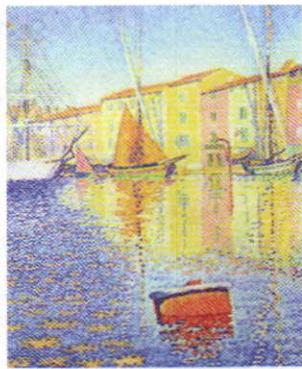


Рис. 2.14



Пит Мондриан

Рис. 2.15



Поль Синьяк. Красный буй

Рис. 2.16

ФОРМУЛИРУЕМ ВЫВОД

В повседневной жизни мы постоянно имеем дело с цветом, испытываем его воздействие.

Цвет может вызывать эмоции: радость и уныние, печаль и восторг.

Существует множество цветовых сочетаний, которые называются гармоничными. И, конечно, существуют цвета, которые не сочетаются друг с другом.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Составьте таблицу удачных цветовых сочетаний.
2. Попытайтесь почувствовать взаимоотношения между различными цветами, потренируйтесь в этом.
3. Рассмотрите картины известных художников, определите, какими оттенками цвета пользовались мастера. Какую информацию, какое настроение, какие чувства вызывает у вас каждая картина?
4. Откройте изображение pic2_1. Раскрасьте знаки, подбирая гармоничные сочетания цветов. Сохраните файл в папке и под именем, которые укажет учитель.

§ 3. Создание коллажа

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Рис. 2.17

Рассмотрите рис. 2.17. Как вы думаете, какой приём использован для его создания?

- Сформулируйте тему урока. Сравните свой вариант с авторским (с. 142 учебника).

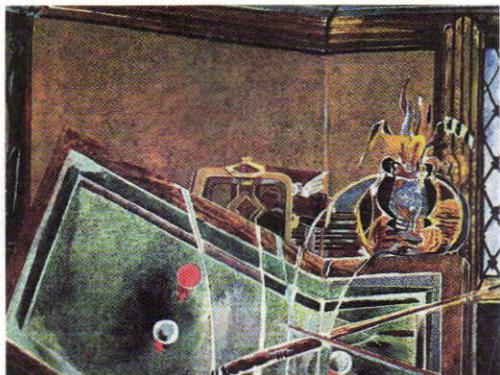
ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Вспомните, как создаются изображения на компьютере. (Книга 1 учебника, модуль 3, § 1–5.)

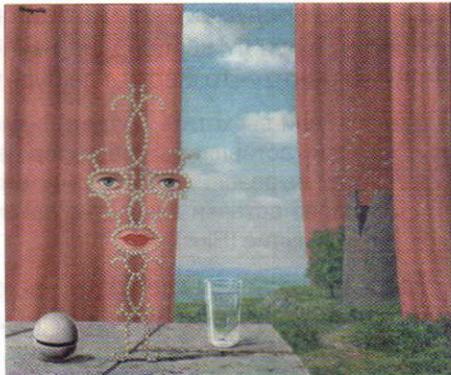
РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Специальные эффекты могут существенно украсить изображение. Вы можете соединить вместе части двух или нескольких фотографий и создать новый уникальный образ.

Коллажем (от французского слова *collage* — «приkleивание») называется технический приём создания графического произведения путём наклеивания на какую-либо основу предметов и материалов различных формы и цвета. Одними из первых коллаж в своих произведениях использовали испанский художник, скульптор и дизайнер Пабло Пикассо и французский художник, график, сценограф, скульптор и декоратор Жорж Брак. Сегодня под коллажем понимается приём создания целого изображения из ряда других изображений или их отдельных фрагментов, в основном с помощью компьютерных программ, а также само это изображение. Сегодня использование коллажа в компьютерном, книжном, журнальном дизайне, web-дизайне — это обычная практика. Коллаж — не просто наложенные и соединённые между собой разнородные фрагменты, это композиция, образ (рис. 2.18).



Жорж Брак. Бильярдный стол



Рене Магритт. Художник

Рис. 2.18

Умение со вкусом соединить разномасштабные изображения и пространства может пригодиться при создании зрелищных или интригующих изображений (рис. 2.19).



Рис. 2.19

Зачастую дизайнеры при создании монтажей используют слишком большое количество разных предметов, видимо, чтобы сделать свою работу более впечатляющей. Но объектов должно быть ровно столько, сколько нужно, чтобы передать идею изображения. Любой объект должен подвергаться сомнению: а зачем он здесь нужен? Если ответа на этот вопрос нет — смело убирайте объект из рисунка.

Сначала продумайте сюжет вашего изображения, а потом творите, но не наоборот. Прежде чем создавать подобную работу, определите её композиционный центр — то место, где будет находиться главный объект.



Если вы хотите создать достаточно правдоподобное изображение, учтите, что у вставляемых объектов не должно быть жёстких краёв (контуров). Края при вырезании можно смягчить, используя растушёвку (Select/Feather) или после вставки аккуратно пройтись по краям тонкой кисточкой инструмента Размытие (Blur).

Многие дизайнеры не задумываются о том, какую ответственность они берут на себя, работая с фотографиями людей. Особенно когда вносятся существенные изменения в содержание снимка. Манипулирование изображениями связано с определёнными моральными ограничениями. Публикуя чужую работу, не забудьте спросить разрешение у автора фотографии. При обработке и редактировании снимков, когда на них присутствуют изображения людей, опирайтесь на здравый смысл. Никогда не помещайте портреты людей в обидное для них окружение.

Создание коллажа в растровом редакторе технически сводится к нескольким операциям — это:

- 1) выделение области;
- 2) перенос выделенной области или копирование;
- 3) трансформация (поворот, изменение масштаба, искажение);
- 4) настройка тона и цвета фрагмента, для реалистичного или гармоничного сочетания с другими фрагментами.

Выделенную область можно скопировать и вставить (например, в другое изображение), можно просто перетащить мышью в другое место. Выделение можно трансформировать (масштабировать, повернуть, наклонить, исказить).

При трансформации часто особенно важно не нарушить пропорции изображения, особенно это касается лиц людей. Вам будет очень неприятно, если вы увидите свой портрет с вытянутым или расплывшимся в ширину лицом.

Одним из важнейших инструментов редактирования и рисования считаются **слои**. Разнообразие слоёв, с которыми приходится иметь дело в процессе обработки растровых изображений, достаточно велико. Мы же познакомимся с двумя видами.

- **Фоновый слой**, или слой заднего плана (Background). Это основной слой любого изображения. Любой отсканированный рисунок или цифровая фотография первоначально состоит из одного слоя заднего плана. Фон — это особый слой. Он не имеет режимов наложения, не допускает изменения прозрачности и в многослойном изображении может занимать только самую нижнюю позицию.
- **Изобразительные слои**. Первоначально при создании такого слоя он абсолютно прозрачен. Это слои, которые предназначены для хранения

фрагментов изображения. Для них разрешается менять прозрачность и менять положение «в стопке», задавать различные режимы наложения.

При работе с многослойным изображением очень важно понимать, на каком слое происходит работа. Все рисующие инструменты, команды выделения действуют только на активный слой. Для выбора активного слоя достаточно выделить его имя мышью в палитре слоёв.

Невозможно только создавать слои. Объединение слоёв является финальной операцией технологического процесса. Поэтому если вы не уверены, что закончили редактирование, то не объединяйте слои.



Прочтите советы по работе с выделениями и слоями.

- Все скопированные объекты располагаются на разных слоях, поэтому их можно расположить выше, ниже, удалить объект, удалив слой, на котором он располагался.
- Не забывайте, что созданные слои прозрачны; если на слое объект только один, можно сразу выполнять трансформацию слоя. Если же вам всё-таки надо выделить изображение на слое, щёлкните мышью на пиктограмме слоя в палитре слоёв с нажатой клавишей Ctrl.
- Сохраняйте вашу работу в процессе и по её окончании во внутреннем формате PSD графического редактора *Adobe Photoshop*, этот формат позволяет сохранить изображение со слоями. Это удобно для дальнейшего редактирования работы.
- Свойство инструментов выделения — *растушёвка* (Feather) — изменяет выделение таким образом, что по его краям появляется область частично выделенных пикселей (плавный переход изображения). На частично выделенный пиксель все инструменты воздействуют также частично, в зависимости от степени выделенности.

ФОРМУЛИРУЕМ ВЫВОД

Специальные эффекты могут существенно украсить изображение. Вы можете соединить вместе части двух или нескольких фотографий и создать новый уникальный образ — коллаж.

Создание коллажа в растровом редакторе технически сводится к нескольким операциям — это:

- выделение области;
- перенос выделенной области или копирование;
- трансформация (поворот, изменение масштаба, искажение);
- настройка тона и цвета фрагмента, для реалистичного или гармоничного сочетания с другими фрагментами.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Рассмотрите картины художников-сюрреалистов. Какие чувства и эмоции вызывают эти картины? Какую информацию мы получаем?

2. Откройте файл pic5_1.psd. Ответьте на вопросы:

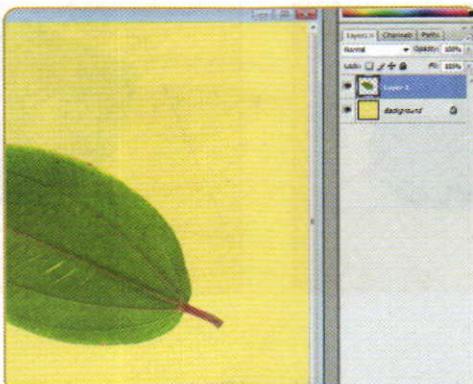
- Сколько слоёв использовано в изображении?
- Что изображено на фоновом слое?
- Почему изображение фонового слоя не видно при просмотре?
- Удалите часть изображения так, чтобы стал виден фоновый слой.

3. Откройте файл pic5_2.jpg. Выделите объект и трансформируйте его: измените масштаб, угол поворота, наклон. Сохраните файл в папке и под именем, которые укажет учитель.

ОПЕРАЦИИ

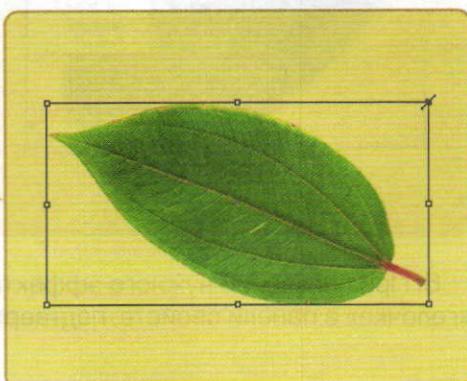
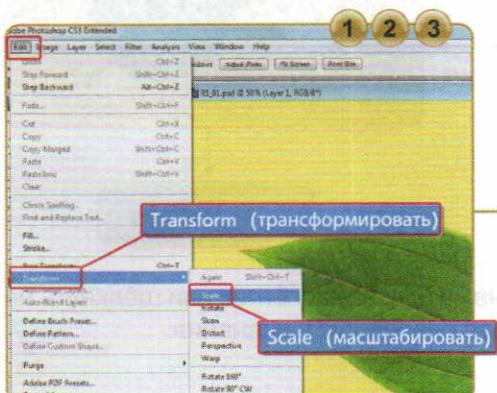
Выделение изображения на отдельном слое

Для выделения изображения на отдельном слое можно щёлкнуть левой кнопкой мыши на пиктограмме слоя с нажатой кнопкой Ctrl.

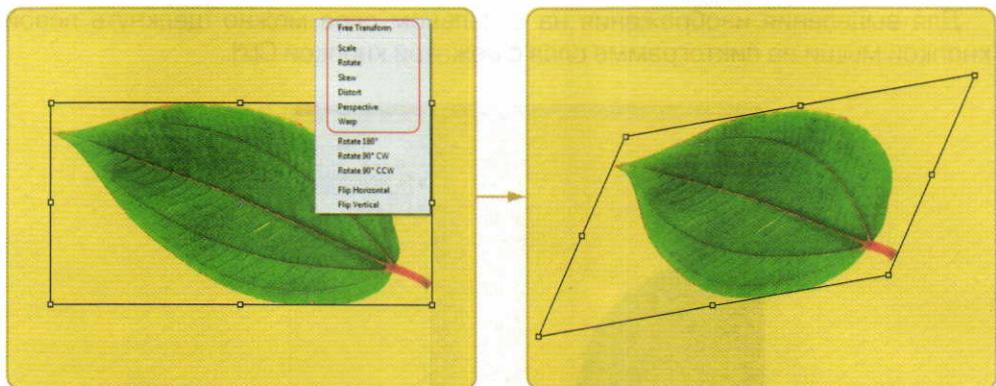


Трансформация выделенной области

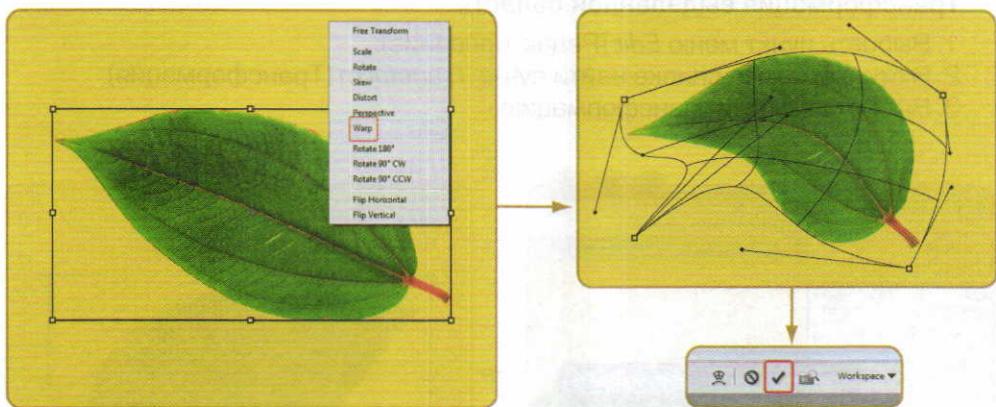
1. Выбрать пункт меню *Edit* (Редактирование).
2. В открывшемся списке найти пункт *Transform* (Трансформация).
3. Выбрать нужную трансформацию.



4. После применения любой трансформации можно выполнять последующие, нажав правую кнопку мыши и выбирая команды из контекстного меню.

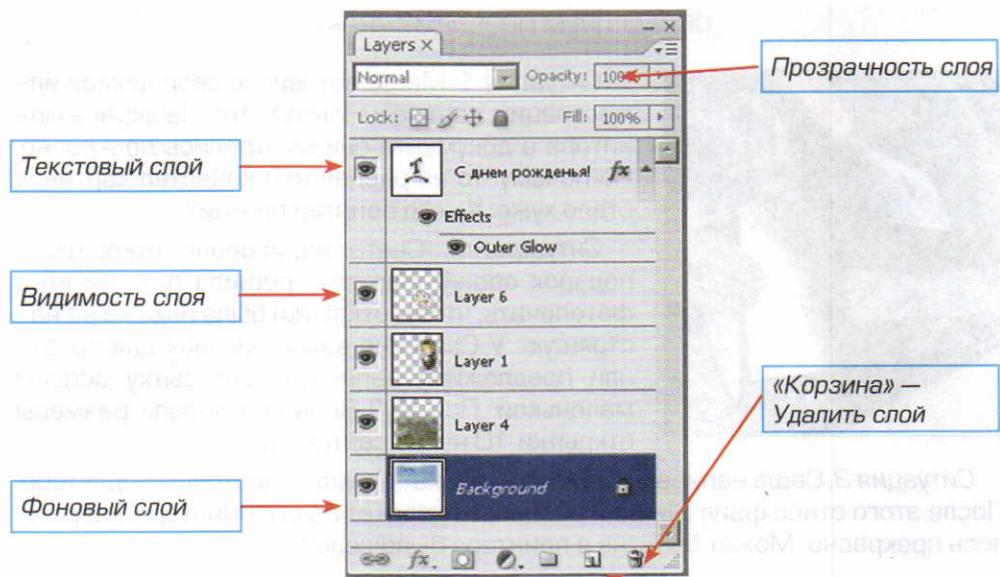


5. При выборе трансформации *Wrap* (Деформация) область трансформирования можно искажать за узелки или по готовым схемам в панели свойств.



6. При получении нужного эффекта нажать клавишу Enter или щёлкнуть на «галочке» в панели свойств, подтверждая сделанные изменения.

Панель работы со слоями



«Чистый лист» — Создать новый слой.
Новый изобразительный слой состоит из прозрачных точек и становится активным

§ 4. Сканирование изображений. Подготовка изображений к печати

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Ситуация 1. Миша вставил в свой доклад иллюстрации, взятые из Интернета. На экране монитора в документе они смотрелись прекрасно. Но почему-то на распечатке качество картинок стало хуже. У него принтер плохой?

Ситуация 2. Света нарисовала открытку в подарок своей подруге и решила отнести её в фотопечать, чтобы открытка была похожа на настоящую. У Светы не взяли рисунок для печати или предложили напечатать открытку совсем маленькой. Почему? Ведь она задала размеры открытки 10 на 15 сантиметров.

Ситуация 3. Саша написал–нарисовал плакат прямо в текстовом редакторе. После этого отнёс файл Мише и распечатал плакат на его принтере. Получилось прекрасно. Может быть, не в принтере было дело?

- Есть ли общая проблема в этих ситуациях? Если да, то сформулируйте её. Сравните свою формулировку с авторской (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Что такое пиксель? (Книга 1 учебника, модуль 3, § 1.)

Что такое разрешение изображения и в чём оно измеряется? (Книга 1 учебника, модуль 3, § 1.)

Какое разрешение изображения нужно для вывода изображения на экран монитора? (Книга 1 учебника, модуль 3, § 1.)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Одним из способов получения растрового изображения является оцифровка его с помощью сканера (**сканирование**).

Чаще всего при сканировании изображений не затрудняются выбором параметров, а нажимают кнопку *Сканировать*, полагаясь на универсальность настроек сканера.

После рассмотрения всех особенностей растровой графики очевидно, что универсальных значений для пиксельных изображений не существует. Чтобы

получить при сканировании качественное оцифрованное изображение, надо ответить на ряд вопросов при задании параметров:

- **Цель сканирования (разрешение)**

Разрешение для вывода на экран монитора (страница в Интернете, мультимедийная презентация и др.) обычно ограничивается значением 72 или 96 ppi. Однако если изображение предполагается подвергать предварительному редактированию, следует увеличить разрешение вдвое — 144–150 ppi. После редактирования разрешение понижается до 72 ppi.

- **Нужная палитра**

Если оригинал для сканирования — чёрно–белые изображения, предназначенные только для вывода на экран монитора (страница в Интернете, мультимедийная презентация и др.), или чёрно–белая фотография, то сканировать нужно в цветовом режиме *Grayscale* (полутона серого).

Цветное изображение будем сканировать в цветовом режиме *RGB* (только если оно нам не нужно в чёрно–белом виде).

- **Область сканирования (рамка сканирования)**

Нецелесообразно загромождать и память компьютера, и изображение ненужными деталями — установим рамку сканирования в окошке просмотра так, чтобы максимально отсечь всё лишнее. Но обычно рамку располагают немного шире необходимой области, а потом средствами редактирования в редакторе подрезают края.

- **Размеры изображения (масштаб)**

Изображение можно увеличить или уменьшить при оцифровке. Практически все сканеры позволяют изменить масштаб при сканировании.

Распространённой причиной появления зернистости и «цифровой грязи» является сканирование с неверно выбранными параметрами или загрязнённость стекла сканера. Загрязнённость и физические повреждения могут стать причиной появления обширных площадей серого тона и множества чёрных точек, хаотично покрывающих поверхность отсканированной картинки. Если у вас появились такие дефекты в большом количестве, измените параметры сканирования и протрите стекло сканера!



Можно получить цифровое изображение методом сканографии — сканированием предметов, размещённых непосредственно на стекле сканера. Естественно, существуют ограничения общего размера, веса и толщины предметов. Если объекты сканирования могут испачкать стекло, надо подложить под них тонкую пищевую плёнку. Интересные композиции получаются, если сканировать: мелки, цветы, траву, мох, пуговицы, перья, макаронные изделия, ножницы, карандаши, линейки, ластики, ручки и т. д.

- Вспомните, какое разрешение изображения нужно для экрана монитора. На что влияет разрешение изображения?

Какое же разрешение изображения будет оптимальным для печати?

Программы растровой графики и браузеры отображают пиксели изображения при помощи пикселей экрана. Если пиксели изображения на экране будут отображаться пикселями экрана один к одному, то это создаст наилучшие условия для просмотра.

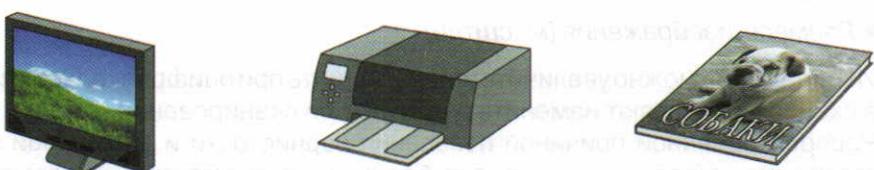
Если вы собираетесь распечатывать изображение, то наилучшими условиями для печати будут отображения одной точкой на бумаге одного пикселя изображения. У любого принтера напечатанная точка гораздо меньше, чем пиксель экрана. Поэтому для печати пиксель должен быть меньше, а разрешение изображения соответственно больше. Считается, что разрешение изображения в 200–300 пикселей на дюйм достаточно для качественной печати практически на любом принтере.



В подавляющем большинстве случаев для качественной печати в типографии потребуется изображение с разрешением 300 ррі.

В любом случае необходима консультация специалистов именно той полиграфической студии или типографии, которые будут заняты подготовкой к печати или печатью изображения.

Таким образом, можно разделить все компьютерные изображения по целям использования и выделить значения разрешения изображения в зависимости от этих целей (рис. 2.20).



<p><i>Изображения для монитора – 72 ррі, 96 ррі</i></p>	<p><i>Изображения для печати на принтере – 200–300 ррі</i></p>	<p><i>Полиграфическая печать – 300, 600 ррі</i></p>
---	--	---

Рис. 2.20



Одним из самых распространённых растровых форматов, используемых при подготовке изображений к печати, является **TIFF** (Target Image File Format). Формат незаменим при подготовке документов для печати.

TIFF поддерживает несколько алгоритмов сжатия, специальные функции управления изображением. LZW-сжатие является сжатием без потерь — когда файл сжимается, никакие данные не уничтожаются и не должно произойти ни малейшего ухудшения качества.

Следует учесть, что для размещения в Интернете не подойдёт формат TIFF, а в типографии у вас никогда не попросят файл в формате JPEG.

ФОРМУЛИРУЕМ ВЫВОД

Программы растровой графики и браузеры отображают пиксели изображения при помощи пикселей экрана. Если пиксели изображения на экране будут отображаться пикселями экрана один к одному, то это создаст наилучшие условия для просмотра.

Если мы собираемся распечатывать изображение, то наилучшими условиями для печати будут отображения одной точкой на бумаге одного пикселя изображения. У любого принтера напечатанная точка гораздо меньше, чем пиксель экрана. Поэтому для печати пиксель должен быть меньше, а разрешение изображения соответственно больше. Считается, что разрешение изображения в 200–300 пикселей на дюйм достаточно для качественной печати практически на любом принтере.

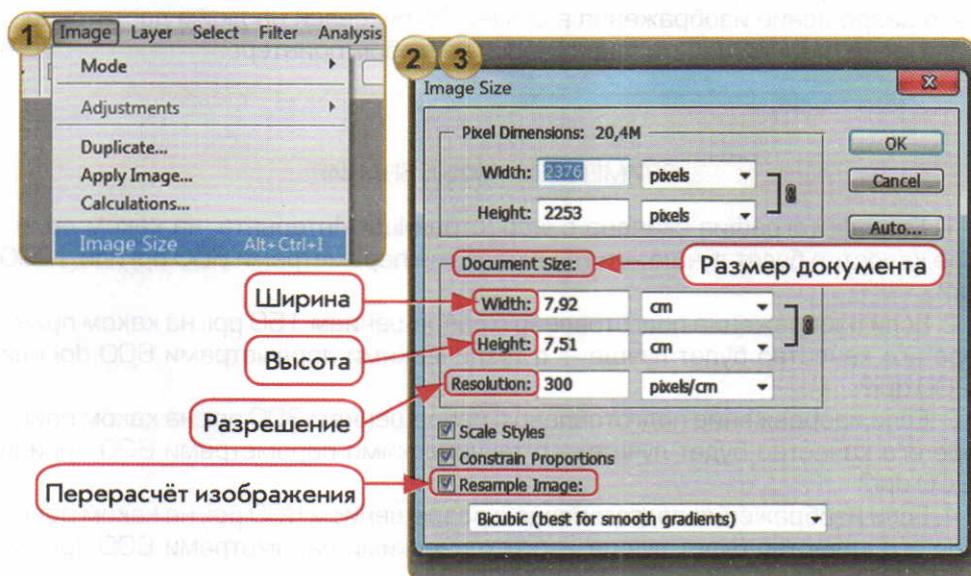
ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Если фотография скачана с web-страницы Интернета, на каком принтере качество будет лучше — с техническими параметрами 600 dpi или 1200 dpi?
2. Если изображение подготовлено с разрешением 150 ppí, на каком принтере его качество будет лучше — с техническими параметрами 600 dpi или 1200 dpi?
3. Если изображение подготовлено с разрешением 300 ppí, на каком принтере его качество будет лучше — с техническими параметрами 600 dpi или 1200 dpi?
4. Если изображение подготовлено с разрешением 600 ppí, на каком принтере его качество будет лучше — с техническими параметрами 600 dpi или 1200 dpi?
5. Откройте файл pic4_1. Проверьте, можно ли качественно распечатать это изображение на принтере. Какие размеры при печати получатся у фотографии?

ОПЕРАЦИИ

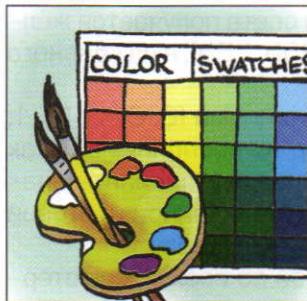
Изменение размеров изображения и его разрешения для печати

- Выбрать в меню *Image* (Изображение) команду *ImageSize* (Размер изображения). Здесь можно поменять размеры и разрешение изображения.
- Чтобы изменить размер изображения (не меняя разрешение), надо поменять размер по ширине и высоте в верхнем поле.
- Чтобы изменить размер изображения, не теряя и не добавляя новые пиксели, надо отключить флагок («галочку») *Resample Image* (Пересчёт изображения).



§ 5. Цветовые модели и палитры в компьютере

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Задание: нужно «объяснить» компьютеру, какой цвет мы называем синим.

- Как вы считаете, можно ли выполнить это задание? Сформулируйте основной вопрос урока. Сравните свой вариант с авторским (с. 142 учебника).

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Цвет всегда составлял значительные трудности для описания. При развитии компьютерных технологий (цифровой графики, цифровой фотографии, устройств сканирования и цветной печати) стало необходимым, чтобы компьютерные программы «понимали», какой цвет пользователь применяет или собирается выбрать.

Художники часто объединяют понятия цвета и краски, поскольку имеют дело только с красками, а пользователи компьютерных технологий должны различать эти понятия, поскольку им приходится рассматривать цвет также в качестве световых волн (например, в процессе сканирования или отображения рисунка или фотографии на экране).

Цветовая модель — это способ описания цвета с помощью количественных характеристик. Это позволяет не только сравнивать отдельные цвета и их оттенки между собой, но и использовать их в цифровых технологиях.

Существует множество моделей цвета, которые описывают различные его характеристики. Мы познакомимся лишь с самыми распространёнными.

Рассмотрим цвет, который *создают световые волны*. Для описания таких цветов используются три основных цвета — красный, зелёный и синий.

Цветовая модель RGB названа по первым буквам слов Red (красный), Green (зелёный) и Blue (синий).

Цвет в этой модели описывается тремя значениями: количеством красного, количеством зелёного и количеством синего цвета — от 0 до 255 (рис. 2.21).

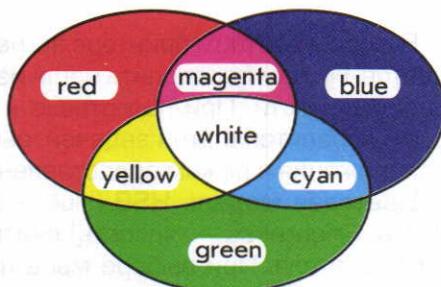


Рис. 2.21. Основные цвета модели RGB

Поскольку эта модель описывает световые потоки, то максимум света — белый цвет будет описываться так: (255, 255, 255). Отсутствие света — чёрный цвет: (0, 0, 0).

Значения (255, 0, 0), (0, 255, 0), (0, 0, 255) соответствуют чистым красному, зёлению и синему цветам. Из красного и зелёного света получается жёлтый: (255, 255, 0), из зелёного и синего — голубой: (0, 255, 255), а из красного и синего — пурпурный: (255, 0, 255).

Для нашего глаза оттенки цвета, описываемые значениями [0, 0, 1], [0, 1, 0] и даже [3, 6, 4], будут неразличимы; но компьютер их воспринимает как различные — числа-то разные! Если перебрать все различные варианты значений, то получится число 16 777 216. Это означает, что в данной цветовой модели компьютер различает более 16 миллионов оттенков цвета.

Модель цвета RGB применяется в качестве основной во всех компьютерных системах. Поэтому в этой модели доступны максимальные возможности редактирования изображения.

Если бы изображения «жили» только на экране монитора, нам бы хватило одной модели для описания цвета, но достаточно часто изображения приходится распечатывать на принтере или в полиграфической фирме.

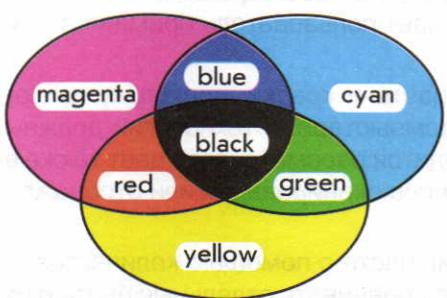


Рис. 2.22. Триада основных цветов модели CMYK

Цветовая модель CMYK (Cyan — голубой, Magenta — пурпурный, Yellow — жёлтый) описывает реальные полиграфические краски (рис. 2.22). В практических печатных процессах к этим цветам добавляется еще чёрный цвет (black).

Цвет описывается процентным содержанием краски при печати. Значения (0%, 0%, 0%, 0%) описывают белый цвет — красок нет, чистая белая бумага. А, например, значения (0%, 0%, 100%, 0%) будут описывать чистую жёлтую краску.

В любом цветном принтере вы найдёте картриджи с красками этих четырёх цветов. Но нам переводить наши изобразительные творения в эту модель нет необходимости. При подготовке к печати на цветном принтере программы сами справляются с этой задачей, выполняя внутреннее преобразование RGB-изображения в режим представления CMYK-цветов.

Цветовая модель HSB (Hue — цветовой оттенок, Saturation — насыщенность, Brightness — яркость) считается самой удобной в подборе цвета для пользователя. При выборе мы в любой программе фактически пользуемся этой моделью, а не вводим цифровые характеристики цвета (рис. 2.23).



Рис. 2.23. Модель HSB даёт пользователю удобство выбора цветового тона



С моделями цвета связаны ещё два сложных и похожих понятия: глубина цвета и палитра.

Глубина цвета измеряется числом двоичных разрядов (битов), отведённых для хранения каждого пикселя. Единица глубины цвета — b/p (бит на пиксель).

Установка глубины цвета необходима в начале работы с изображением и определяет его тип и количество возможных оттенков тона (цвета).

Различают следующие основные типы изображений по глубине цвета: чёрно-белые изображения, изображения в полутонах серого и полноцветные изображения (см. таблицу).

Цветовая глубина	Компьютерная палитра	Цветовой режим в <i>Adobe Photoshop</i> , пояснения	Пример
1 b/p	Чёрно-белая двухцветная (2 цвета)	Штриховой рисунок (BITMAP)	
8 b/p	Чёрно-белый полутон (256 цветов)	Тип такого изображения называют «grayscale» («серая шкала»)	

Цветовая глубина	Компьютерная палитра	Цветовой режим в <i>Adobe Photoshop</i> , пояснения	Пример
24 b/p	RGB TrueColor (Истинный цвет) (более 16 млн оттенков цвета)	Тип такого изображения называется по названию цветовой модели — RGB	

Цвет может быть представлен в природе, на экране монитора, на бумаге. В каждом случае возможный диапазон цветов, или цветовой охват, будет разным. Самый широкий охват представлен в природе, он ограничивается только возможностями человеческого зрения. Часть из того, что существует в природе, может передать монитор. Часть из того, что передаёт монитор, можно напечатать.

ФОРМУЛИРУЕМ ВЫВОД

В цифровых технологиях для описания цвета используются различные цветовые модели — RGB, CMYK, HSB и другие.

Модель RGB — цветовая модель, которая используется в устройствах, работающих со световыми потоками: сканерах, мониторах. Для описания световых волн используются основные цвета — красный, зелёный и синий.

Модель CMYK — цветовая модель, которая описывает реальные красители, используемые в полиграфическом производстве.

Модель HSB — пользовательская цветовая модель, которая позволяет выбирать цвет традиционным способом.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Рассмотрите чёрно-белые фотографии. Подумайте (или найдите ответ на вопрос в тексте параграфа), сколько оттенков цвета необходимо, чтобы отобразить их на экране монитора. Как называются такие изображения?

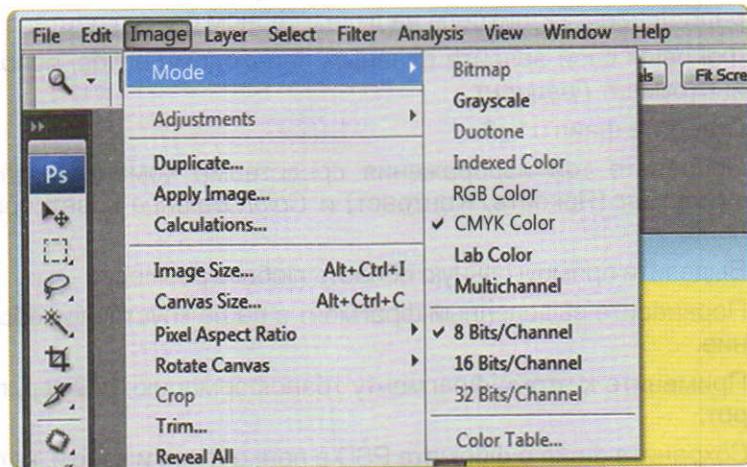
2. Приведите примеры из жизни, когда потребуется палитра всего из двух цветов: чёрного и белого.

3. Найдите примеры, когда из чёрно-белого двухцветного изображения можно получить больше информации, чем из цветного.

ОПЕРАЦИИ

Задание цветовой модели при создании изображения в Adobe Photoshop

1. В меню *Image* (Изображение) выбрать команду *Mode* (Режим).
2. В открывшемся списке выбрать нужный режим.



Проверь себя

Задание 1

- Создайте новое изображение с размерами 500 x700 пикселей, разрешением 72 пикселя на дюйм, фоном белого цвета.
- Фоновый слой залейте плавным переходом цветов, используя инструмент *Градиент*.
- Откройте файл d1_1.
- Поправьте тон изображения средствами команд *Brightness /Contrast* (Яркость/Контраст) и *Color Balance* (Цветовой баланс).
- Выделите прямоугольную область любого размера.
- Перенесите выделенный фрагмент в ваше «пустое» изображение.
- Примените к этому фрагменту трансформацию: размер, поворот.
- Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 2

- Откройте файлы d1_2, d1_3, d1_4.
- Создайте новое изображение с размерами и разрешением для печати.
- В пустом изображении соберите коллаж из фрагментов первых трёх изображений.
- Исправьте цветовой тон вставленных фрагментов так, чтобы они подходили друг другу по цвету.
- Добавьте, если это нужно, текст и декоративные элементы.
- Сохраните файл в формате и цветовой модели для печати в папке и под именем, которые укажет учитель.

ПРИМЕНЯЕМ ЗНАНИЯ (необходимый уровень)

Задание 1

- Создайте новое изображение с размерами 800x600 пикселей, разрешением 72 пикселя на дюйм, фоном белого цвета.
- Фоновый слой залейте неярким цветом, используя инструмент *Заливка*.
- Откройте файл ex1_1.
- Поправьте тон изображения средствами команд *Brightness/Contrast* (Яркость/Контраст) и *Color Balance* (Цветовой баланс).
- Выделите прямоугольную область любого размера.
- Перенесите выделенный фрагмент в ваше «пустое» изображение.
- Примените к этому фрагменту трансформацию: размер, поворот.
- Подпишите ваше изображение на пустом месте фона.
- Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 2

- Создайте новое изображение с размерами 10x15 сантиметров, разрешением 200 пикселей на дюйм, фоном белого цвета.
- Фоновый слой залейте плавным переходом цветов, используя инструмент *Градиент*.
- Откройте файлы ex2_1, ex2_2.
- Выделите овальную область любого размера в каждом изображении.
- Перенесите выделенные фрагменты в ваше «пустое» изображение. Разместите их в композиционных точках листа.
- Примените к фрагментам трансформацию: размер, поворот.
- Поправьте тон изображения и фрагментов средствами команд *Brightness/Contrast* (Яркость/Контраст) и *Color Balance* (Цветовой баланс).
- Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 3

- Откройте файлы ex3_1, ex3_2, ex3_3.
- Соберите из элементов изображение. Примените трансформацию при необходимости.
- Перекрасьте элементы в два контрастных цвета.
- Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 4

1. Откройте файлы ex4_1.
2. Раскрасьте нарисованные объекты в три подходящих по гамме цвета.
3. Сохраните файл в формате JPG в папке и под именем, которые укажет учитель.

Задание 5

1. Откройте файлы ex5_1, ex5_2, ex5_3.
2. Соберите из элементов изображение. Изображение из файла ex5_1 поместите в композиционный центр листа.
3. Изображение из файла ex5_2 положите на фон.
4. Изображение из файла ex5_3 расположите на нижних композиционных точках.
5. Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

ПРИМЕНЯЕМ ЗНАНИЯ (повышенный уровень)

Задание 1

1. Откройте файлы ex6_1, ex6_2, ex6_3.
2. Вставьте изображения из второго и третьего файлов в первый, применив трансформацию.
3. Внимательно посмотрите на получившееся сборное изображение. Исправьте яркость, контраст, оттенок у фрагментов так, чтобы они подходили друг другу.
4. Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 2

1. Создайте новое изображение с размерами 500x700 пикселей, разрешением 72 пикселя на дюйм, фоном белого цвета.
2. Откройте файл ex7_1.
3. По очереди выделите объекты инструментами выделения и перетащите мышью в новое изображение.
4. Распределите объекты на листе, сделайте дубликаты некоторых из них и их трансформацию.
5. Поправьте тон и цвет всех фрагментов средствами команд *Brightness/Contrast* (Яркость/Контраст) и *Color Balance* (Цветовой баланс) так, чтобы они подходили друг другу.
6. Фоновый слой залейте неярким цветом.
7. Сохраните файл в формате PSD со слоями в папке и под именем, которые укажет учитель.

Задание 3

- Нарисуйте логотип (знак), используя два контрастных цвета. Цель создания — печать на принтере.
- Сохраните в файле в формате PSD в папке и под именем, которые укажет учитель.

Задание 4

- Откройте файл ex8_1.
- Измените размер и разрешение файла для печати на принтере изображения размером 8x12 см.
- Сохраните в файле в формате JPG в папке и под именем, которые укажет учитель.

Задание 5

- Создайте изображение для печати на принтере размером 10x15 см. Копию этого изображения оптимизируйте для web-страницы под размер по ширине 300 пикселей.
- Сохраните файлы в формате JPG в папке и под именем, которые укажет учитель.

ПРИМЕНЯЕМ ЗНАНИЯ (максимальный уровень)**Задание 1**

Создайте по описанию эскиз для открытки к празднику (а может быть, это будет приглашение на спектакль, концерт или школьный бал?)

- Создайте новое изображение с подходящими размерами для печати на принтере.
- Разработайте композицию открытки. Подумайте, для кого будет предназначена открытка.
- Создайте композицию, не забудьте про надпись.
- Проверьте все элементы и сохраните открытку в файле в формате PSD со слоями в папке и под именем, которые укажет учитель.

Задание 2

Создайте по описанию коллаж из двух–трёх фотографий, одна из которых — портрет.

- Откройте файлы ex9_1, ex9_2, ex9_3 или свои подготовленные фотографии.
- Наложите фоновые изображения друг на друга так, чтобы одно плавно перешло в другое.
- Перенесите портретное изображение в созданный фон.

4. Проверьте все элементы на соответствие тонового и цветового диапазона.
5. Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 3

Конструируйте и фантазируйте — создайте по описанию чудо-героя для новой сказки.

1. Подберите несколько фотографий различных объектов, из которых будем переносить изображения (для этого урока не стоит выбирать изображения людей и животных. Лучше остановиться на фотографиях предметов, овощей, игрушек и т. п.), или откройте файлы ex10_, ex10_2, ex10_3.
2. Создайте героя сказки из частей или целых объектов.
3. Сохраните файл со слоями в формате PSD в папке и под именем, которые укажет учитель.

Итоговая проверочная работа

Задание 1

1. Откройте файлы с именами d11_1, d11_2, d11_3.
2. Вставьте изображения из второго и третьего файла в первый, примените трансформацию.
3. Контрастным цветом подпишите название получившейся композиции.
4. Проверьте вашу композицию и цветовую гармонию.
5. Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 2

1. Откройте файлы с именами d12_1, d12_2, d12_3, d12_4 или свои подготовленные фотографии.
2. Создайте коллаж из двух–трёх фотографий с подходящими размерами для печати на принтере.
3. Придумайте и добавьте текст в коллаж.
4. Проверьте все элементы на соответствие тонового и цветового диапазона.
5. Сохраните файл в формате PSD в папке и под именем, которые укажет учитель.

Задание 3

1. Создайте плакат–объявление к новогоднему празднику с подходящими размерами для печати на принтере. Для плаката можно воспользоваться изображениями из файлов d13_1, d13_2, d13_3, d13_4, d13_5.
2. Сохраните файл со слоями в формате PSD в папке и под именем, которые укажет учитель.



Решаем жизненные задачи и работаем над проектами

Жизненная задача 1. Создание эскиза сувениира

Ваша роль: заботливый сын или дочь.

Описание. Вы решили подарить на праздник своим родным неожиданный подарок. В городах в фотостудиях существует услуга нанесения рисунка на белую кружку.

Задание. Для рисунка на кружку вам потребуется изображение размером 20x9 см с разрешением 150ppi. Этот «листок» обернёт всю кружку от ручки и до ручки. Представьте мысленно кружку, придумайте композицию, нарисуйте её или скомпонуйте из фотографий. Если рядом с вашим местом жительства есть фотостудия, можете заказать там готовый сувениир.

Жизненная задача 2. Создание обложки карманного календаря

Ваша роль: дизайнер.

Описание. В школе решили напечатать тираж карманных календариков на следующий год для сувениров ученикам, учителям и гостям школы.

Задание. В крупных городах фотостудии и полиграфические салоны предлагают услуги печати карманных календарей. Календарный блок у них уже есть готовый. Ваша задача — придумать и создать лицевую сторону календарика. Стандартный размер календаря — 7x10 см. Не забудьте на обложке указать год, на который создаётся календарь.

Жизненная задача 3. Создание рекламной листовки

Ваша роль: работник рекламной фирмы.

Описание. Глядя на листовку, плакат или рекламу в журнале, вы когда-нибудь задавали себе вопрос — как это сделано? Часто смотришь на такие работы с благоговением и удивлением. Однако сейчас для получения многих отличных художественных эффектов вовсе не обязательно иметь большой талант. Достаточно просто понимать, какой эффект будет хорошо смотреться.

Задание. Создайте рекламную листовку важного события для школы. Подберите подходящие материалы. Разработайте подходящий стиль для листовки: она должна получиться яркой, броской, все элементы должны быть хорошо видны и «читаться». Подготовьте макет для печати на принтере или в типографии.

Жизненная задача 4. Восстановление старой фотографии

Ваша роль: работник фотостудии.

Описание. У вас дома лежит несколько старинных фотографий. Родители собираются восстановить и распечатать их.

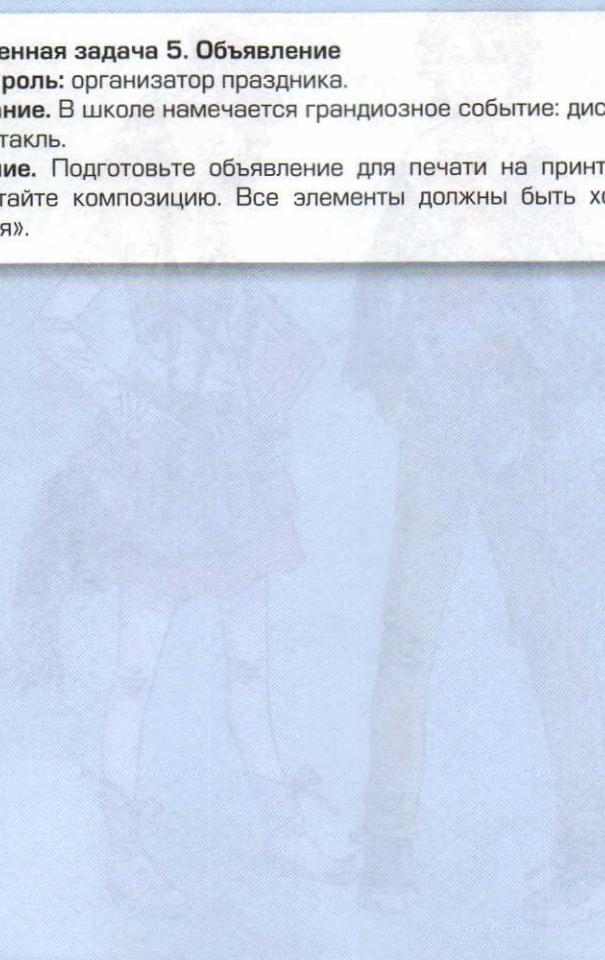
Задание. Подготовьте фотографии для печати на струйном принтере или в фотостудии. Отсканируйте или аккуратно сфотографируйте бумажные фотографии на цифровой фотоаппарат. Выполните ретушь снимков — уберите дефекты, поправьте тон и цветовой оттенок, обрежьте немного края.

Жизненная задача 5. Объявление

Ваша роль: организатор праздника.

Описание. В школе намечается грандиозное событие: дискотека, концерт или спектакль.

Задание. Подготовьте объявление для печати на принтере о событии. Разработайте композицию. Все элементы должны быть хорошо видны и «читаться».





Модуль 3. Основы издательских технологий

Этот модуль поможет вам:

- понять, какие бывают виды печатных изданий и для чего они предназначены;
- узнать, какие выразительные средства используются в печатных изданиях и для чего;
- создавать свои макеты печатных изданий и пользоваться готовыми макетами.

Для этого вам надо научиться:

- создавать публикации;
- оформлять текст;
- правильно располагать и оформлять различные элементы публикации.

Введение



При помощи компьютерной полиграфии изготавливают самые разнообразные документы: от визитных карточек до многоцветных журналов и книг.

Одни из самых важных вопросов, которые задаются при создании любой публикации: почему читатель нуждается в информации? кто будет читателем? Читателя надо привлечь содержанием текста, иллюстрациями или общим видом издания.

Различные виды публикаций требуют различных текстовых и изобразительных элементов.

Например, книжка для дошкольников будет содержать яркие и большие по размеру иллюстрации, крупный легкочитаемый текст небольшими кусочками на странице; доклад по математике будет содержать в себе хорошо организованный текст, возможно, на несколько страниц, математические формулы, графики или схемы.

Ещё один важный вопрос: какое впечатление вы хотите произвести на читателя? Ваша информация серьёзна? Игрива? Дружеская? Официальная? Предназначена убедить читателя? Насмешить? Обучить?

Компьютерные технологии расширили возможности размножения документов. В зависимости от числа копий (тиража) вы можете распечатать публикацию на принтере, воспользоваться светокопировальным устройством, отнести публикацию в офис цифровой печати или отдать в типографию.

§ 1. Макеты для публикаций

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Рис. 3.1

Рассмотрите страницу (рис. 3.1). Где начало текста, куда двигаться дальше и где — конец?

- Сформулируйте проблему урока. Сравните свою формулировку с авторской (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Вспомните, что вы знаете об организации материала на странице. (Книга 1 учебника, модуль 2, § 4.)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Основу организации материала на странице составляют приёмы компоновки текста, иллюстраций, декоративных элементов. Общая структура **полосы** (страницы) задаётся с помощью **модульной сетки**. Дизайн страниц по модульной сетке удобен как для автора, так и для читателя. Многим людям свойственно стремление к порядку. Сетки — простое и эффективное средство внесения порядка, сберегающее ваше время. Страница должна быть предсказуемой для читателя: где начало текста, куда двигаться дальше, а где — конец.

Линии модульной сетки отражаются на экране при подготовке публикации, но не выводятся на печать. Создаются линии сетки с помощью направляющих линий. (Простейший способ создать направляющие — вытащить их мышью с линеек, предварительно отобразив линейки на экране.)

С помощью модульной сетки задаются ширина полей, количество колонок, отбивки заголовков и подзаголовков и другие элементы оформления страницы.

Сетки помогают выдерживать единый стиль оформления страниц одного документа, придавать общий облик серии документов. Вы можете изготовить один макет и с небольшими вариациями использовать его для различных работ.

Колонки представляют собой важнейший элемент модульной структуры и широко используются для организации текста и иллюстраций на странице. Количество колонок на странице может колебаться от одной до шести—семи. Чем их больше, тем шире простор для творчества. Не все колонки текста должны иметь одинаковую ширину.

Например, сетка всего из трёх колонок допускает различные варианты оформления страницы (рис. 3.2).



Рис. 3.2

В разных издательских системах используются различные способы создания модульных сеток. Некоторые программы содержат наборы готовых шаблонов, которые пользователь может использовать и редактировать по своему усмотрению.

Иногда для задания границ полей и элементов страницы используются направляющие линии. Часто в качестве основного средства расположения материала выступают окна или рамки.

Стили обеспечивают мгновенный доступ к целому набору параметров оформления текста, это мощнейшее средство автоматизации труда дизайнера и верстальщика. Применение стилей совершенно необходимо, если вы создаёте многостраничный документ или разрабатываете форму документа, которая будет часто повторяться в других публикациях. Определение стилей — один из самых важных начальных этапов создания документа после разработки внешнего облика страницы.

Когда предварительные работы сделаны, говорят, что подготовлен **макет публикации**.

Первоначальную подготовку текста удобно выполнить в текстовом редакторе, а окончательную вёрстку и подготовку к печати в типографии — в настольных издательских системах.

Настольные издательские системы, в отличие от графических и текстовых редакторов, предназначены не для создания изображений или текста, а для объединения текста и рисунков в соответствии с макетом. Это действие называют **вёрсткой**.

Мы будем создавать несложные публикации в программе *Microsoft Publisher*, которая максимально облегчает работу новичку в области макетирования и вёрстки.

ФОРМУЛИРУЕМ ВЫВОД

Страница должна быть предсказуемой для читателя: где начало текста, куда двигаться дальше, а где — конец. Модульные сетки — простое и эффективное средство внесения порядка. Часто в качестве основного средства расположения материала выступают окна или рамки.

Предварительным этапом подготовки публикации является макет.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Рассмотрите различные печатные издания (журналы, листовки, книги). Попробуйте представить себе, как выглядели макеты и модульные сетки для этих изданий.

2. Откройте файл pic1. Рассмотрите макет страницы. Определите, где на макете находится иллюстрация, где — текст, где — элементы оформления.

3. Возьмите лист бумаги. Разделите его на 3, 4 или 6 (!) колонок. Нарисуйте макет рекламного объявления. Отметьте линиями или прямоугольниками, где будут располагаться заголовок, текст, иллюстрации, фирменный знак, как разместить рекламный лозунг или девиз.

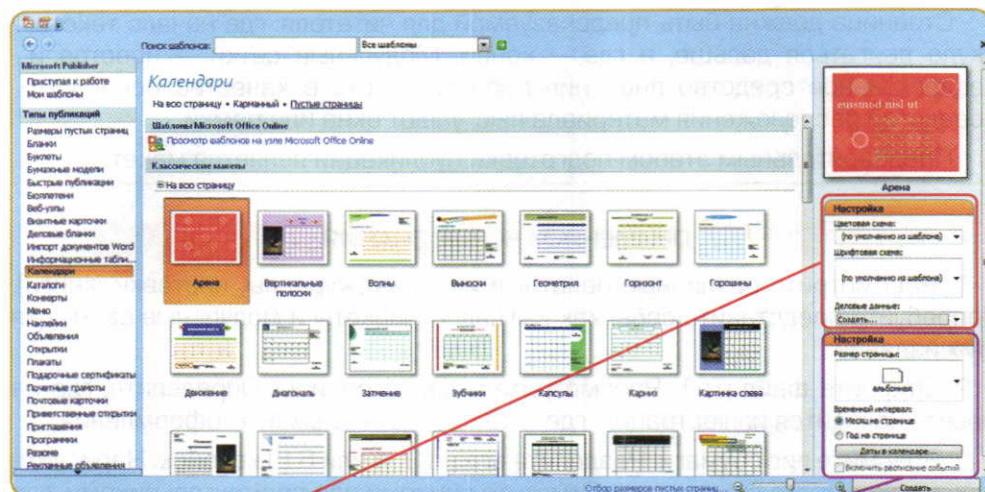
ОПЕРАЦИИ

Создание публикации

1. В меню Пуск найти программу *Microsoft Office Publisher*.
2. Щёлкнуть мышью на названии программы. Откроется окно *Приступая к работе с Microsoft Office Publisher*.
3. В окне выбрать тип публикации и макет.
4. Нажать кнопку *Создать*.



Выбор макета из списка готовых



Настройка

Цветовая схема:
(по умолчанию из шаблона)

Шрифтовая схема:
(по умолчанию из шаблона)

Деловые данные:
Создать...

Размер страницы:

альбомная

Временной интервал:

Месяц на странице

Год на странице

Даты в календаре...

Включить расписание событий

§ 2. Календари

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



- Какое сегодня число?
- Посмотри в календаре.
- Когда у нас будет праздник?
- Посмотри в календаре.
- Сформулируйте тему урока. Сравните свой вариант с авторским (с. 142 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Какие существуют правила оформления текста? (Книга 1 учебника, модуль 2, § 2.)

Вспомните, что вы знаете об иллюстрировании документов. (Книга 1 учебника, модуль 2, § 3.)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Календарь – один из самых распространённых и популярных видов фирменной рекламной полиграфии. Широкое распространение календарей в первую очередь определяется функциональностью – календарь содержит важную для большинства информацию. Очень часто различные организации печатают календари в качестве сувенира или корпоративного подарка. Существует множество конструкций календарей: от карманных до широкоформатных календарей–плакатов, от стандартных до уникальных.

Календари карманного формата предназначены для широкого круга. Их раздают посетителям выставок, распространяют в рамках различных рекламных кампаний. Функциональная направленность такого календарика схожа с визитной карточкой, только не человека, а организации.

Оформление такого календаря может быть достаточно лаконичным, а может привлекать внимание всеми красками. Иногда дизайнеры превращают маленький календарик в своеобразный рекламный листок.

Календарь является одним из элементов корпоративного стиля. На обложке такого календаря размещают фирменную символику. Поскольку этот сувенир функционален, на обложке обязательно размещают год, на который создаётся календарь.

Фоном для надписей и знаков чаще всего становится производственный пейзаж. Но иногда на обложке размещают видовую фотографию, тематически не связанную с деятельностью организации. На обложках карманных календарей можно увидеть цветы, фотографии животных, пейзажи и натюрморты. О принадлежности к фирме напоминает логотип. Знак должен вписываться в цветовую гамму и быть достаточно крупным.

- Подумайте, что будет являться производственным пейзажем для вас – учеников и учителей школы. А производственным натюрмортом?

Настольный календарь является одним из самых распространённых типов фирменного календаря. Он предназначен для людей, работающих в офисе. Конструкция таких календарей может быть различной. Наиболее распространёнными моделями можно назвать календарь-«домик» и перекидной календарь с блоком на пружине. Технологии меняются, и появляются новые модели настольных календарей (рис. 3.3).

Настенные отрывные календари слегка варьируются по конструкции, размеру и композиции. Наиболее распространён на данный момент квартальный календарь на пружине (рис. 3.4).



Рис. 3.3



Рис. 3.4

Календарь-плакат украшает стены помещения там, где мы живём или работаем. Не лишаясь своей функциональности, такой календарь в первую очередь выполняет художественные функции. Изображения к таким календарям – видовые фотографии, портреты, репродукции произведений живописи. И, наконец, существуют календари-альбомы, с красивыми фотографиями, репродукциями или дизайнерскими композициями на каждый месяц.

Простая композиция календаря предполагает единственное изображение, определяющее основной сюжет и цветовую основу. В более сложной композиции добавляются второстепенные элементы – декоративные знаки, тексты.

Календарь, построенный по схеме «времена года», обычно показывает различные состояния природы, иногда к ним добавляются предметы, соответствующие времени года. Например, осенний месяц сентябрь можно оформить так: пейзаж из золотистых кленовых листьев, зонтик и портфель первоклассника. Иногда в календарях используют знаки зодиака.

Композиция календарного блока зависит от общего дизайнераского решения. Размер и пропорции, горизонтальная или вертикальная ориентация надписей, метод выделения праздничных дней определяются общим стилем и моделью календаря. Необходимо помнить главное правило – текст должен хорошо читаться.

Программа *Microsoft Publisher* предлагает на выбор большое количество уже готовых макетов. У вас есть возможность выбрать, будет ли ваш календарь на один месяц и какой конкретно, будет ли в календаре 12 листов по числу месяцев в году или это будет календарный блок на год на одном листе.

ФОРМУЛИРУЕМ ВЫВОД

Широкое распространение календарей в первую очередь определяется функциональностью – календарь содержит важную для большинства информацию. Фоном для надписей и знаков чаще всего становится производственный пейзаж. Но иногда на обложке размещают видовую фотографию, тематически не связанную с деятельностью организации. На обложках карманных календарей можно увидеть цветы, фотографии животных, пейзажи и натюрморты.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Рассмотрите различные календари. Определите:

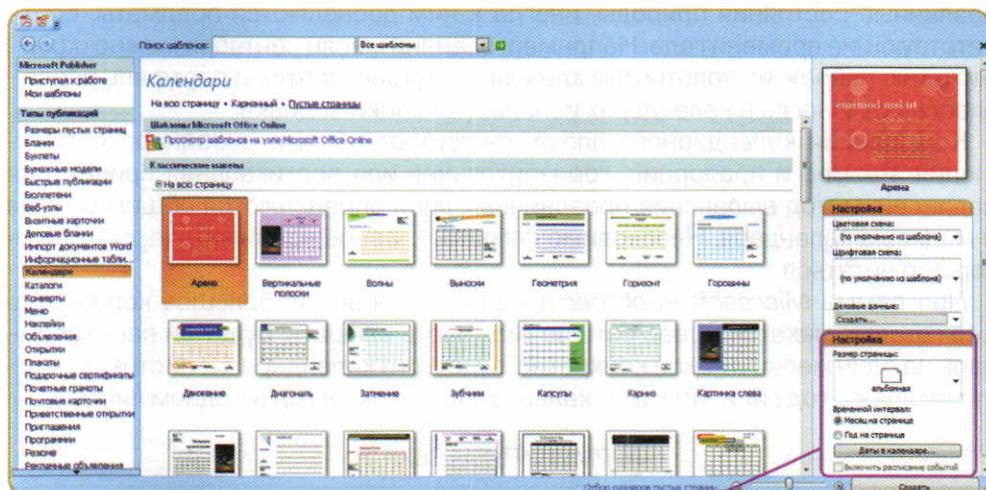
- к какому типу относится календарь;
- какое изображение является изобразительной основой;
- где и как расположен календарный блок;
- какими средствами он оформлен.

2. Создайте в программе *Microsoft Publisher* макет календаря на текущий месяц.

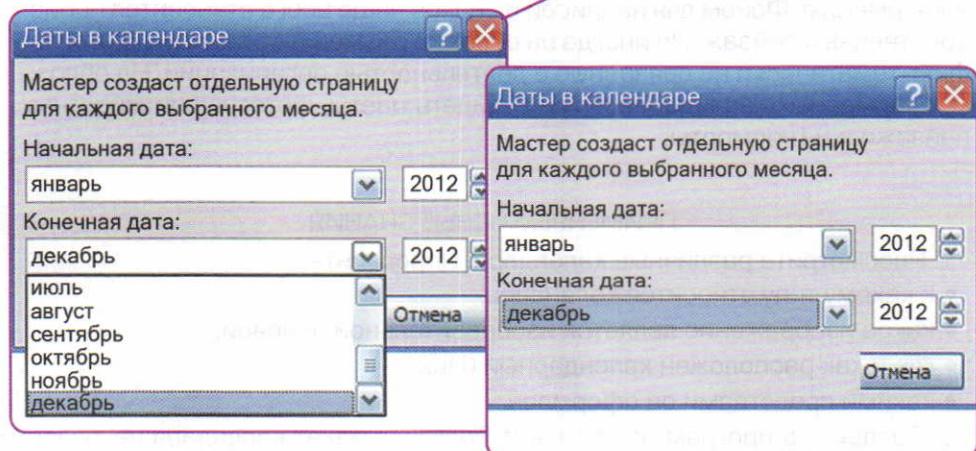
ОПЕРАЦИИ

Выбор макета календаря из готовых макетов

- При создании нового макета в списке **Тип публикаций** выбрать пункт **Календари**.
- Выбрать понравившийся макет календаря.
- На панели справа определить, на месяц или на год будет календарь.
- Там же определить, будут ли календарные блоки по месяцам располагаться на отдельных листах или на одном.
- Нажать кнопку **Создать**.

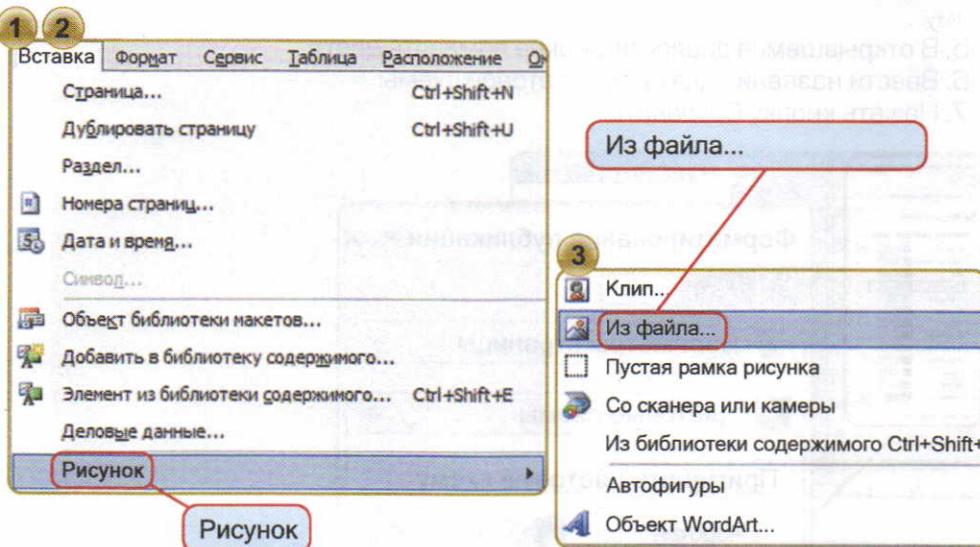


Выбор диапазона месяцев



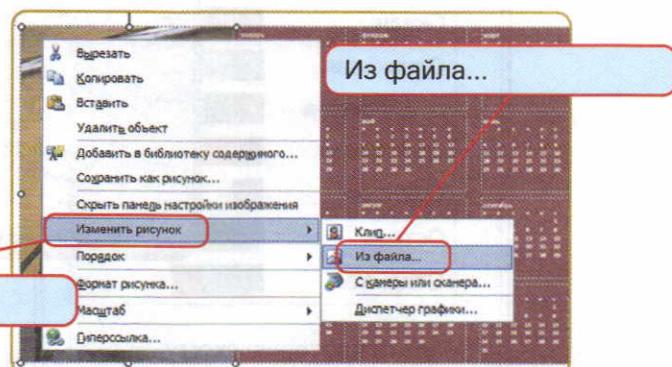
Вставка нового изображения в макет

1. Выбрать меню *Вставка*.
2. Выбрать пункт *Рисунок*.
3. Выбрать команду *Из файла*, если нужно вставить своё изображение, или *Клип*, если требуется выбрать изображение из библиотеки картинок.
4. При выборе из файла в диалоговом окне найти нужный файл.



Изменение изображения в макете

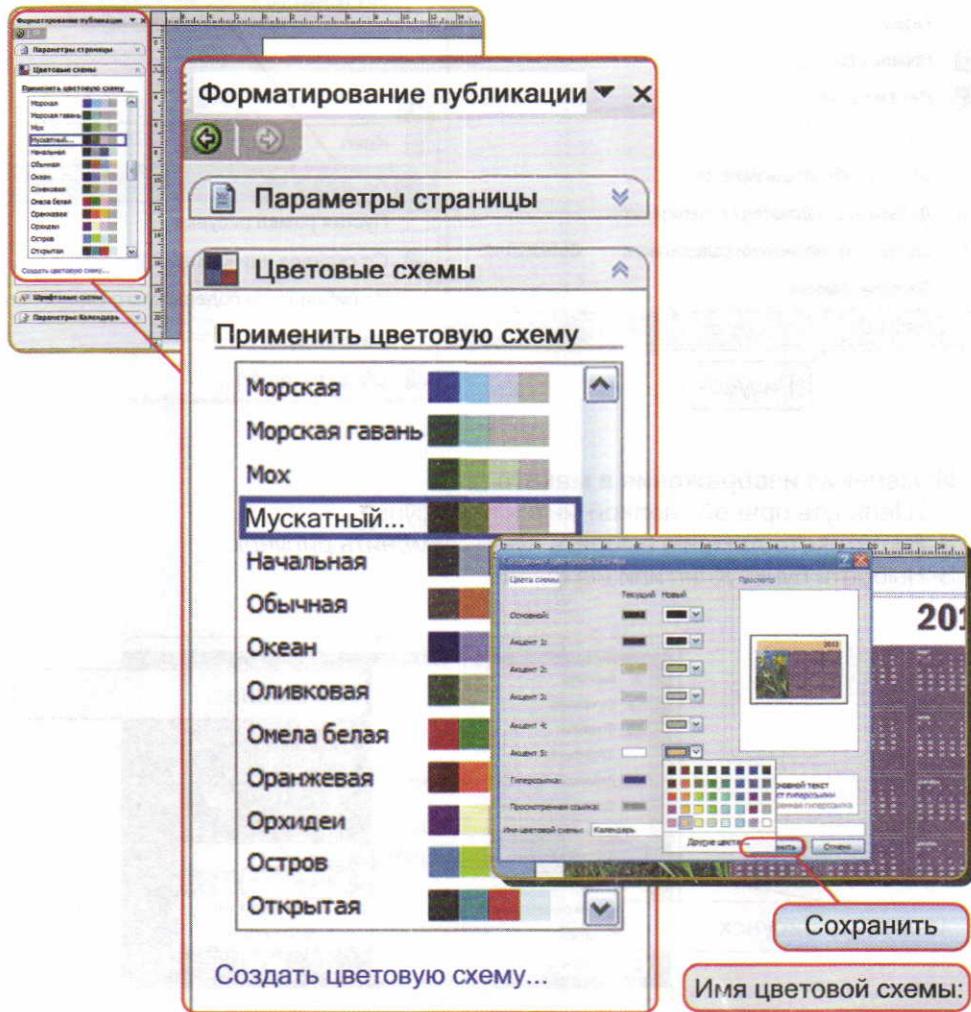
1. Щёлкнуть правой кнопкой мыши на рисунке.
2. В контекстном меню выбрать пункт *Изменить рисунок*.
3. Выбрать пункт *Клип* или *Из файла*.



Изменение цветовой схемы

(используемых в макете цветов по умолчанию)

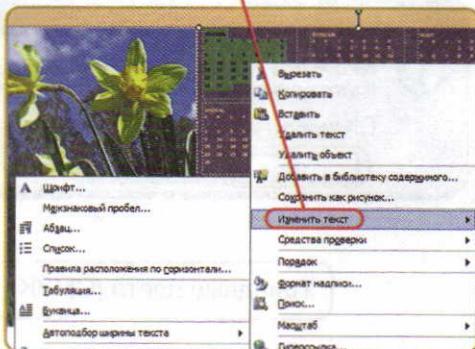
1. На панели слева щёлкнуть на вкладке **Форматирование публикации**.
2. Выбрать окно **Цветовые схемы**.
3. Выбрать одну из предложенных схем.
4. Для создания своей цветовой схемы выбрать команду **Создать цветовую схему**.
5. В открывшемся диалоговом окне поменять цвета.
6. Ввести название для своей цветовой схемы.
7. Нажать кнопку **Сохранить**.



Изменение параметров текста

Чтобы изменить форматирование символов и абзацев, можно:

1. Выделить редактируемый текст.
2. Щёлкнуть на нём правой кнопкой мыши.
3. В контекстном меню выбрать команду *Изменить текст*.



Подготовка к печати

Для печати публикации в цвете следует выполнить проверку макета на ошибки. Для этого:

1. Выбрать вкладку *Форматирование публикации*.
2. Выбрать пункт *Проверка макета*.
3. Для печати в цвете можно все цвета перевести в цветовую модель CMYK.

2

Форматирование публикации

Справочные материалы

Проверка макета

Задачи Publisher

Исправить: преобразовать
к другому цветовому режиму

3

Проверка макета

Выполнить общие проверки макета

Выполнить проверки профессиональной печати

Выполнить проверки веб-узлов

Выполнить проверки электронной почты (только текущая страница)

Выберите элемент для исправления

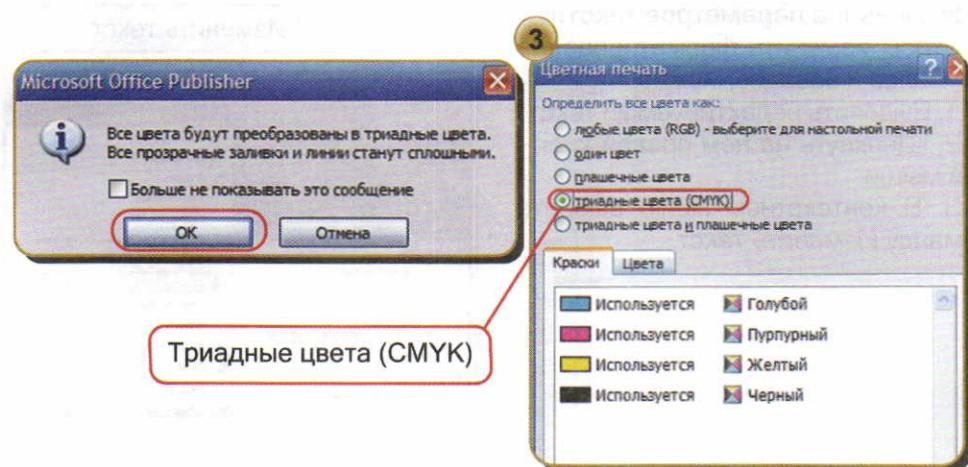
Публикация в режиме RGB
(Публикация)

Перейти к этому элементу

Исправить: преобразовать к другому цветовому режиму...

Больше не выполнять эту проверку

Объяснить...



§ 3. Открытки

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Подслушанный разговор:

- Давайте всем учителям подарим на День учителя открытки!
- Они их будут открывать?
- Сформулируйте тему урока. Сравните свою формулировку с авторской (с. 143 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Вспомните, что вы знаете о создании и редактировании изображений. (Книга 1 учебника, модуль 2.)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Большое распространение в нашей жизни получили открытки. Открытку можно отнести к малым формам графического дизайна.

Жанр открыток может быть различным. Поздравительная открытка создаётся к определённой дате или событию. Видовая открытка изображает какое-то конкретное место или просто привлекательный пейзаж. Художественная открытка воспроизводитrepidурукции изобразительного искусства, портреты, натюрморты. Развлекательная открытка строится на основе юмористического сюжета.

Работая над дизайном открытки, в первую очередь выберите стиль, про-думайте композицию листа. Желательно, чтобы, ещё не открыв открытку, читатель угадал, по какому поводу она подарена. Именно поэтому на открытку добавляется короткая фраза «Поздравляем!», «С юбилеем!», «Приглашение» и другие.

Какой, например, должна быть открытка к Новому году (рис. 3.5)? Новый год – это праздник, с которым принято поздравлять всех знакомых. От Нового года положено ждать чудес и детям, и взрослым. Поэтому стиль оформления открытки можно сделать «детским»: в композицию включить простые геометрические фигуры, сказочные персонажи, для фона использовать яркие цвета. Часто на открытках размещают еловую ветвь или саму ёлку как символ Нового года.

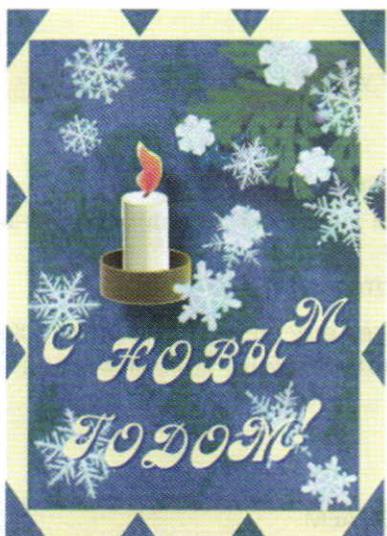


Рис. 3.5

В открытке к семейной дате часто используются различные коллажи из семейных фотографий. Иногда героев знаменательной даты помещают в комическое или фантастическое окружение.

Если вы создаёте открытку-приглашение, стоит разместить на ней какое-то изображение, связанное с поводом приглашения. Например, на приглашении на вечеринку – танцующие пары или весёлые лица, на приглашении на просмотр фильма – плёнку фильма, или стопку дисков, или кадры из фильма.

Программа *Microsoft Publisher* предлагает на выбор большое количество уже готовых макетов открыток. Но ваши великолепные произведения с вашими личными поздравлениями и фотографиями или рисунками будут намного оригинальнее, чем стандартные коммерческие открытки.

Можно изменить размер и разрешение добавляемых изображений и получить хорошие результаты. Но иногда рисунок нельзя достаточно уменьшить или увеличить. Поэтому желательно до начала работы определить, что именно требуется, и затем выбрать наиболее подходящий рисунок.

Если разрешение рисунка слишком мало, он будет печататься с явно выраженнымами точками. Если разрешение рисунка слишком велико, размер файла публикации будет слишком большим и потребуется больше времени на его открытие, редактирование и печать.

Каждый раз при вставке в публикацию рисунка её размер увеличивается. Используя связанные рисунки, можно избежать увеличения размера файла, вызванного внедрёнными рисунками.

При использовании связанных рисунков любые дальнейшие изменения файлов изображений будут отражены в рисунках публикации.

Если необходимо перенести публикацию на другой компьютер, следует переносить также копии связанных рисунков. При использовании мастера упаковки это выполняется автоматически.

При наличии хороших идей, способностей и средств можно создавать собственные рисунки. Но можно также найти рисунки во многих интерактивных источниках.

При копировании рисунка из Интернета, во избежание нарушения авторских прав, убедитесь до публикации, что у вас есть право использовать изображение. Например, можно без ограничений использовать любое изображение из библиотеки клипов и мультимедиа *Microsoft Office*.

ФОРМУЛИРУЕМ ВЫВОД

По своему типу открытки условно делятся на две большие группы: открытка как художественный объект и открытка как элемент фирменного стиля. Эти группы не имеют резких границ. Жанр открыток может быть различным. Поздравительная открытка создаётся к определенной дате или событию. Видовая открытка изображает какое-то конкретное место или просто привлекательный пейзаж. Художественная открытка воспроизводит репродукции изобразительного искусства, портреты, натюрморты. Развлекательная открытка строится на основе юмористического сюжета.

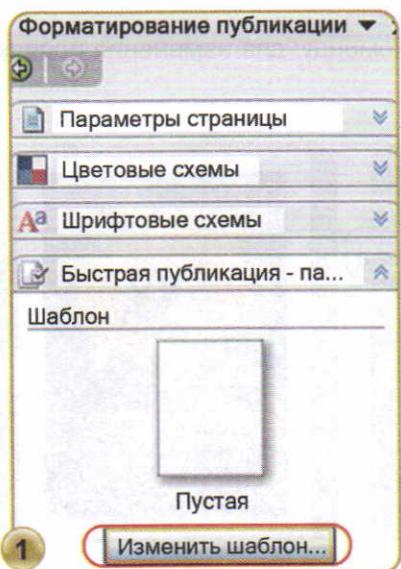
ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Рассмотрите различные открытки, определите жанр каждой.
2. Создайте простую открытку на основе шаблона. Измените надпись. Результат сохраните в виде проекта.

ОПЕРАЦИИ

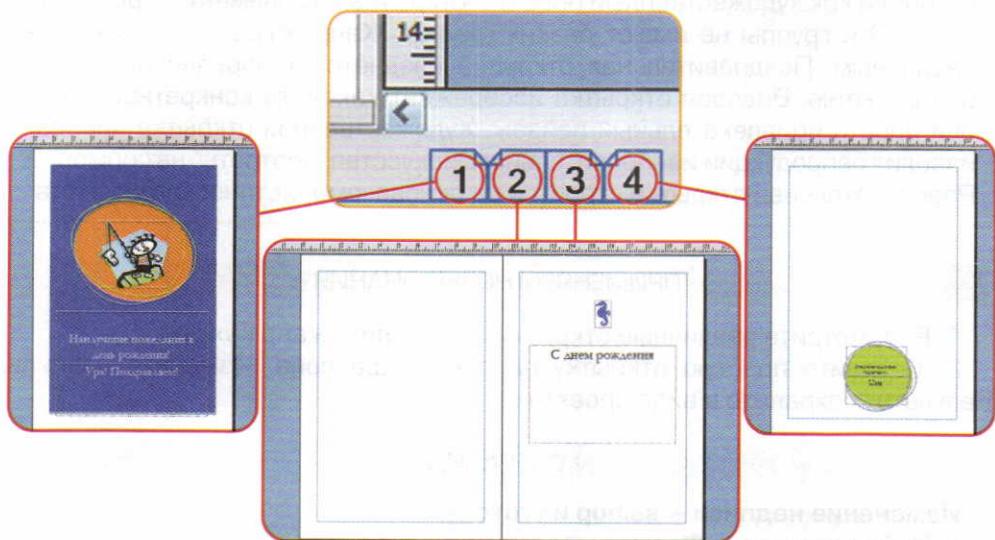
Изменение надписи – выбор из готовых

1. Выбрать вкладку **Форматирование публикации**.
2. Выбрать пункт **Открытки**.
3. Выбрать вариант надписи.
4. Нажать кнопку **OK**.



Переход по страницам

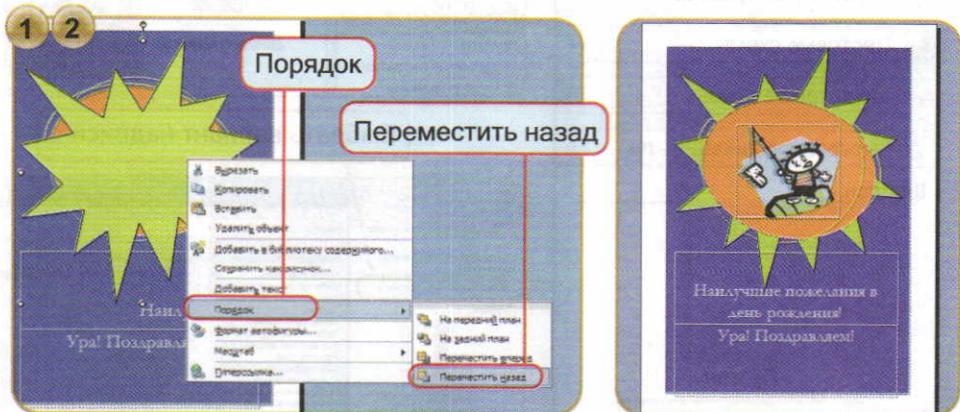
Щёлкнуть на пиктограмме страницы в нижней части рабочего окна. Отобразится страница с выбранным номером.



Изменение порядка наложения объектов

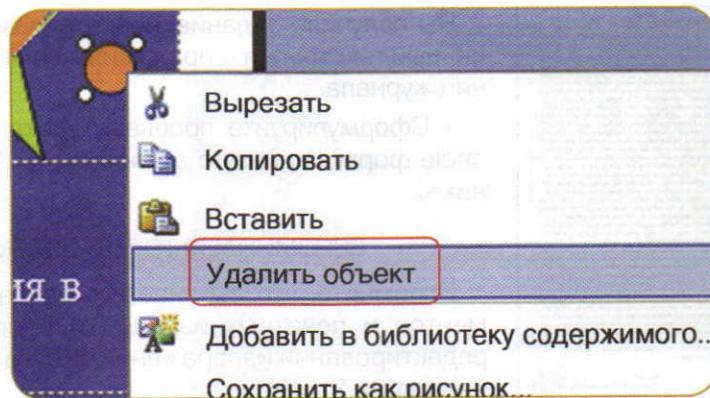
Для редактирования объектов на странице:

1. Щёлкнуть на объекте правой кнопкой мыши.
2. В контекстном меню выбрать нужную команду: для изменения порядка наложения объектов – команду *Порядок*.



Удаление объектов

- Щёлкнуть на объекте правой кнопкой мыши.
- В контекстном меню выбрать команду Удалить объект.



Предварительный просмотр открытки перед печатью

- В меню Файл выбрать команду Предварительный просмотр. Если макет открытки состоит из четырёх страниц, вы увидите расположение этих страниц на листе.
- После просмотра можно отправить открытку на печать или закрыть окно предварительного просмотра и продолжить редактирование.

Задачи Publisher

Печать

Выберите способ печати публикации:

Печать на локальном принтере

Печать публикации на принтере подключеннем к вашему компьютеру или к сети.

Печать на другом принтере

Перенос публикации на другой компьютер для печати.

Печать в типографии

Отправка публикации на печать профессиональную типографию в бюро печати.

Печать на локальном принтере

Ссылки внизу позволяют подготовить публикацию для печати на локальном принтере.

- Проверка качества печати
- Выполнение проверки макета для поиска и исправления ошибок макета, влияющих на печать.
- Изменение размера или ориентации страницы.
- Просмотр

Просмотр

Предварительный просмотр публикации.

§ 4. Создание своей публикации

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



Вы получили задание: создать макет, заполненный текстом и изображениями для двух странниц журнала.

- Сформулируйте проблему урока. Сравните свою формулировку с авторской (с. 143 учебника).

ВСПОМИНАЕМ ТО, ЧТО ЗНАЕМ

Вспомните, что вы знаете о создании документов и печатных изданий и о создании и редактировании изображений. (Книга 1 учебника, модули 2 и 3.)

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Компьютерная полиграфия немыслимо ускорила темпы производства и предоставляет дизайнерам больше возможностей для управления материалом – размещением заголовков, элементов дизайна, размещением и оформлением текста, его перетеканием из блока в блок.

Заголовки являются важным средством организации текста. Действенной будет такая публикация, в которой заголовки отделены от основного текста. Чем больше заголовки отличаются (размером, цветом, шрифтом) от основного текста, тем легче читателям их находить. Не забываем, что эффективный заголовок должен легко читаться.

Подзаголовки объясняют читателю логическую структуру содержания статьи, помогают разбивать текст на удобные для чтения фрагменты.

Выноски, вертикальные и горизонтальные линейки разнообразят оформление и делают публикацию более привлекательной визуально. Выноски – небольшие отрезки текста длиной в одно–два предложения, которые выделяются шрифтом, возможно, рамкой и фоном, и располагаются в разрыве основной колонки или на поле страницы.

Если в вашей публикации предполагается большое количество длинных статей, поместите начало на первой странице и дайте ссылку на продолжение. Метод **связи текстовых блоков** с лёгкостью позволяет это сделать.

Характерный и заметный **фирменный или декоративный знак** подчёркивает индивидуальность публикации. Оформив заголовок публикации с помо-

щью запоминающегося знака, вы придадите ей неповторимый облик. А поместив его в уменьшенном виде на других страницах, вы подчеркнёте логическую цельность работы.

При выборе **шрифтов** для оформления документа следует руководствоваться **принципом контраста** – один наборный шрифт и один заголовочный, различающиеся по одному или нескольким параметрам, например, с засечками и без засечек, светлый и полужирный, мелкий и крупный; и **правилом простоты** – выбирать не более двух шрифтов.

Не стоит переполнять страницы материалом. **Эффект переполнения** возникает, когда текст, иллюстрации и другие элементы располагаются слишком близко друг к другу и к краям страницы. Страницы, в которых нет ни сантиметра свободного места, очень утомительны для чтения.

При взаимном расположении элементов на странице друг относительно друга необходимо **выравнивать** абсолютно все элементы. Выровненность всех элементов позволяет отличить профессиональную работу от работы неумелого ученика. Если величина промежутка, например между заголовком и основным текстом, постоянно меняется, работа выглядит небрежной, а, значит, содержащийся в ней текст воспринимается как малозначительный и недостойный внимания.

Один из способов придать публикации профессиональный вид – **добавить обтекание рисунков текстом**. Функция обтекания текстом позволяет поместить картинку внутри блока текста (рис. 3.6).

Основной текст, который чаще всего отличается значительным объемом, обычно оформляется колонками определенной ширины, в результате текст выглядит блоками, поэтому его можно назвать блочным.

Другой особенностью такого текста является автоматическая верстка по ширине колонки. При изменении ширины колонки, добавлении или удалении текст в пределах абзаца автоматически переворачивается. Важным элементом такого текста является абзац, поэтому его еще можно назвать абзацным.

Кроме того, можно объединить несколько блоков в единый текст, что позволит тексту при редакторских правках «перетекать» из одного блока в другой, сохранив .

Полученный текстовый прямоугольник не имеет цветовых или контурных



Уже в конце первого года обучения в 9 классе учащимся предлагается курсовая работа, которую они выполняют под руководством преподавателя. Итогом курсовой работы должна стать видео или анимированный ролик по собственному сценарию. Для выполнения этой работы отводится время на уроках, поэтому основная часть работы выполняется в утреннее время.



Организационная модель предполагает деятельность отработана и применяется в нашем лицее с 1993 г. В рамках этой модели преподаватели лицей, выступающие в качестве руководителей и консультантов по выпускным работам лицентов – дипломным проектам – создают временные творческие коллективы.

Во 2 четверти 10 класса учащиеся выбирают темы для своих выпускных – дипломных работ. Выбор тем ограничивается только дипломным смыслом и способностями конкретных учащихся. Каждая проектная группа выбирает себе руководителя из числа преподавателей, рассматривает необходимость консультанта.

Учащиеся ЛИП в своих выпускных работах анализируют положение в конкретных секторах рынка информационных технологий и услуг, формулируют стратегические базы данных, а также разрабатывают гипертекстовые и мультимедийные модели по курсам мировой и российской истории, географии и т.д. Работ отдаются научным содержанием, существенно выходящим за рамки действующих учебных планов школы.



Рис. 3.6



И ещё несколько советов для тех, кто хочет быть профессионалом вёрстке.

- Избегайте появления висячих строк в тексте. Висячей строкой называется строка, оторванная от своего абзаца – расположенная на другой странице (странице), чем все остальные строки абзаца. Для удаления висячих строк меняют расстановку переносов или меняют расстояние между символами или словами. Все текстовые редакторы и программы вёрстки позволяют контролировать висячие строки автоматически с помощью опций абзаца.
- «Белые пятна» в тексте – такое явление получается тогда, когда выравнивание абзацев по ширине даёт слишком большие пробелы между словами (рис. 3.7) и при этом используется слишком большой размер шрифта или узкая колонка.

Сегодня многие учителя пытаются использовать возможности компьютера для создания собственных учебных материалов. Использование компьютерных технологий как нового средства обучения открывает новые возможности для профессионального творчества учителя. Целесообразное и эффективное использование «готовых» цифровых образовательных ресурсов может успешно дополняться созданием и применением в обучении авторских цифровых образовательных ресурсов (ЦОР) учителя. Создание авторских ЦОР есть способ реализации индивидуального стиля профессиональной деятельности педагога. Такие ресурсы в значительно большей мере соответствуют индивидуальным особенностям учащихся, с которыми учитель работает.

Рис. 3.7

Чтобы исправить положение, надо или уменьшить размер букв, или увеличить ширину колонки. Можно поправить и то и другое. Можно расставить «мягкие» переносы в длинных словах – они отображаются только тогда, когда попадают на конец строки и слово переносится. (Ни в коем случае нельзя ставить вместо переноса просто знак дефиса! При дальнейшем редактировании эти знаки могут оказаться в тексте лишними и не на месте.)

ФОРМУЛИРУЕМ ВЫВОД

Существует определённый набор умений по оформлению и расположению материала на странице. Но кроме того, существуют определенные правила, которые задают способы расположения элементов на странице и их оформления.

Ваша публикация будет неинтересной, скучной или просто недостойной внимания читателя, если пренебречь правилами вёрстки и дизайна.

ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Создайте макет страницы книжки для дошкольников. Прежде чем взяться за работу, подумайте, чем будет отличаться оформление такой книги от, например, книги для подростков.

2. Создайте макет объявления о каком-либо событии.

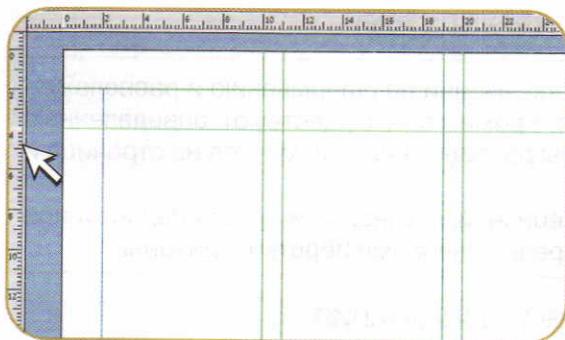
ОПЕРАЦИИ

Действия перед созданием макета

В окне *Параметры страницы*:

1. Указать размер и ориентацию листа.
2. Указать размеры полей публикации.
3. Нажать кнопку *OK*.



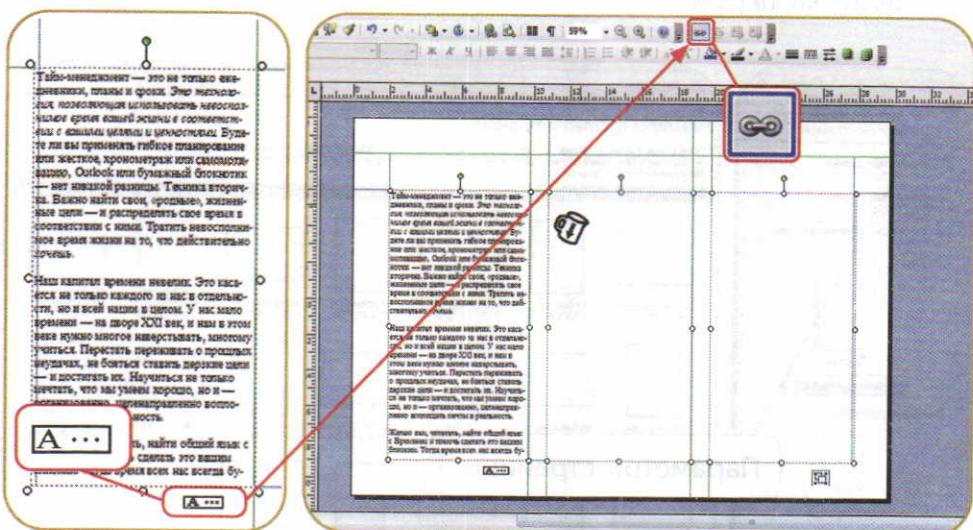


Создание сетки из направляющих

Вытянуть из линеек направляющие для размещения текста и иллюстраций. Для этого щёлкнуть мышью на горизонтальной или вертикальной линейке и, не отпуская кнопки мыши, протащить появившуюся линию на нужное место.

Перетекание текста

1. Разместить в публикации несколько пустых текстовых блоков.
2. Вставить подготовленный текст в один из блоков. Если текст не помещается в выбранном блоке, появится диалоговое окно с предупреждением.
3. Щёлкнуть мышью на пиктограмме под текстовым блоком.
4. Найти на верхней панели значок связи («цепочка») и щёлкнуть на нём мышью. Курсор примет вид чашки.
5. Щёлкнуть мышью в новом выбранном блоке. Текст продолжится в новом месте.



§ 5. Рекламные публикации

ОПРЕДЕЛЯЕМ ПРОБЛЕМУ УРОКА



«Вчера я зашёл в книжный магазин. Купил себе три постера – они мне понравились. Брошюры было много, но на нужную мне тему ни одной не нашлось. На выходе захватил буклет, а рассмотрел его уже дома». О чём идёт речь? «Переведите» текст.

- Как вы думаете, о каких публикациях пойдёт речь в параграфе? Сформулируйте тему урока. Сравните свой вариант с авторским [с. 143 учебника].

РЕШАЕМ ПРОБЛЕМУ, ОТКРЫВАЕМ НОВЫЕ ЗНАНИЯ

Ни для одного вида публикаций внешний облик не значит так много, как для рекламы, потому что информация заключена в самих средствах её выражения. Сильное первоначальное впечатление задаётся всем: общим решением, изображениями, цветом, размером и формой листа. Чтобы быть эффективной, реклама должна производить сильное зрительное впечатление хорошим оформлением и чёткой информацией.

К рекламным публикациям относят листовки, плакаты, постеры, буклеты, брошюры.

Слово **«постер»** пришло из полиграфии и обозначает плакат или афишу рекламного характера или произведения искусства, тем или иным образом оформленные для использования в интерьере.

Постеры – это качественные изображения, которые по силе их воздействия на человека практически не уступают традиционному искусству. Существуют растиражированные постеры, они продаются огромными тиражами по всему миру, ценность таких работ невелика. Их можете купить и вы, и ваш сосед. Они, несомненно, украсят интерьер, но не приадут ему уникальности. Более цennыми являются авторские работы. Такие работы сделают ваш интерьер оригинальным, будут яркими деталями обстановки.

Брошюра (рис. 3.8) – это непериодическое книжное издание небольшого объёма в мягкой обложке, страницы которого соединены между собой при помощи шитья скрепкой или ниткой.

Оформление брошюр редко бывает шаблонным – каждый дизайнер всегда старается отличиться, а брошюра даёт возможность «развернуться». Учиты-



Рис. 3.8

стка брошюры, особенно многополосной, – достаточно трудоёмкое занятие.

Самые популярные виды брошюр: рекламные брошюры, инструкции и памятки, справочники, агитационные брошюры, туристические брошюры и... обычные школьные тетради!

Термин «**буклеть**» пришёл к нам из Франции, и означал он «складывать» или «скручивать». Буклеть представляет собой лист, чаще всего с одним или двумя параллельными сгибами, и складывается как ширма (рис. 3.9). Это придаёт изделию компактность и структурно разделяет информационные блоки: они читаются по кольцу, отсюда и название. Поэтому информацию можно преподнести последовательно в заранее продуманном порядке.

Суть буклета состоит в том, чтобы на минимальной площади вместить максимум информации. Ещё одна важная особенность буклете, отличающая его от листовок, с одной стороны, и от брошюры – с другой, состоит в том, что буклеть не имеет сшитых страниц, как брошюра, но и не является однополосным листом, как листовка.

Интересно также, что буклеть не является предметом продажи. Он распространяется бесплатно, чаще – в общественных местах с высокой посещаемостью. Информация в буклете может быть как просветительского, так и рекламного характера. Ярость и красочность для буклете являются обязательным условием, ведь он должен привлечь к себе внимание и вызвать заинтересованность.

Самый распространённый вид буклете – так называемый трёхполосный буклеть. Лист бумаги, на котором напечатан текст, складывается в три раза – получаются три полосы (три страницы).

Компоновка буклете предполагает объединение всех элементов: иллюстраций, текста, информационных вставок (карт, таблиц, графиков), фирменной символики.

Работая над внешним видом – дизайном – буклете или брошюры, надо учесть восприятие развернутого печатного листа.

вая, что брошюра – многостраничная полиграфическая продукция, лучше всего она смотрится, если страницы выполнены в одном стиле, в едином дизайне, по чёткому макету.

Что значит создать брошюру? Это означает дизайнскую идею, подбор иллюстраций, написание и выверку текста, составление содержания и создание обложки. И, конечно, вёр-



Рис. 3.9

Обложку буклета или брошюры часто делают «единой». Цельность может создаваться одной фотографией, расположенной на лицевой и тыльной сторонах обложки, единым цветовым фоном, графическими элементами, единой рамкой.

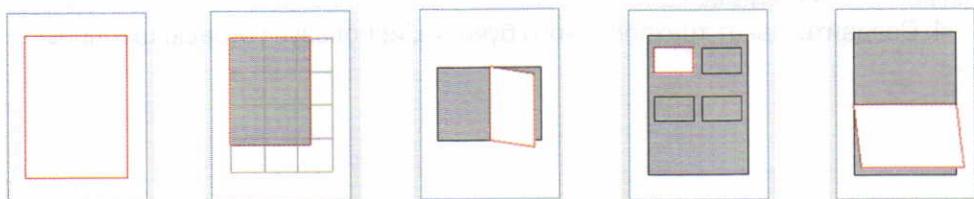
Дизайн обложки во многом определяет общее впечатление от публикации. Если мы вставляем на обложку единственную фотографию, она должна давать нам максимум информации о содержании буклета, быть выразительной и информативной. Иногда на обложку вставляют коллаж из фотографий, называемый «ковровой раскладкой» (рис. 3.10), или создают из изображений бордюр.



Рис. 3.10

Текст в буклете должен быть лаконичным, информативным и хорошо читаемым.

Программы вёрстки поддерживают возможность печати на принтере буклетов и брошюр. Варианты печати могут быть разнообразными (рис. 3.11).



На каждом листе печатается одна страница

Публикация размера афиши разбивается на несколько листов

На каждом листе печатается по две страницы

На каждом листе печатается несколько карточек

На каждом листе печатается по две страницы

Рис. 3.11

Брошюра будет печататься правильно только в случае, если общее количество страниц кратно четырём, но нет необходимости помещать содержимое на все четыре вставленные страницы. Некоторые из них можно оставлять пустыми.



Рис. 3.12

Microsoft Office Publisher печатает первую и последнюю страницы на одной стороне листа бумаги, вторую и предпоследнюю страницы – на другой стороне листа и т. д., как показано на рис. 3.12.

Если компьютер не поддерживает двухстороннюю печать, *Microsoft Office Publisher* печатает первую и последнюю страницы на одном листе бумаги, вторую и предпоследнюю – на другом листе и т. д. После фотокопирования страниц на обе стороны листа, складывания и последующего скрепления или прошивки они расположатся в правильном порядке.

ФОРМУЛИРУЕМ ВЫВОД

Постеры, брошюры и буклеты – непериодические публикации небольшого объёма. Программы макетирования и вёрстки позволяют сверстать такую публикацию и распечатать её на принтере или в полиграфическом салоне.

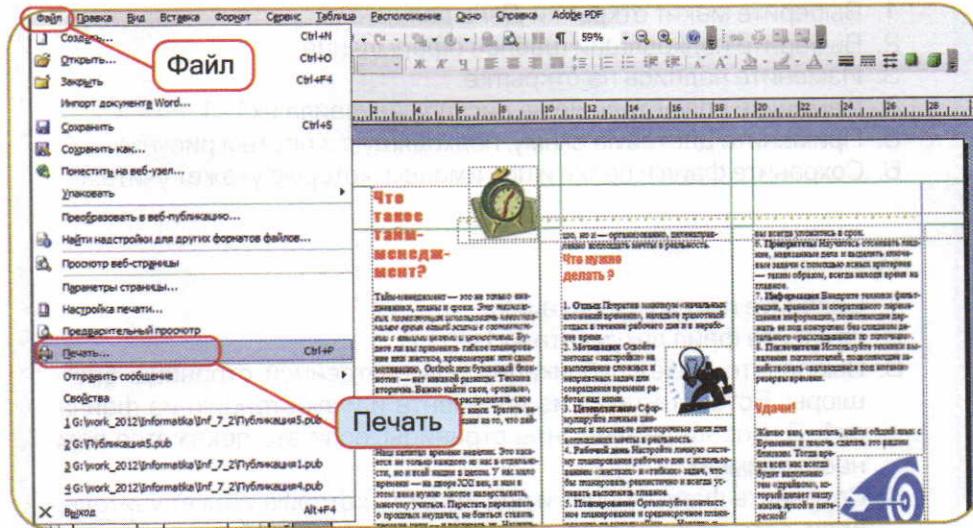
ПРИМЕНЯЕМ НОВЫЕ ЗНАНИЯ

1. Рассмотрите примеры рекламных плакатов. Какую информацию вы получили?
2. Рассмотрите примеры полиграфической продукции. Выберите буклеты и брошюры. Какую информацию передают выбранные вами публикации?
3. Создайте обложку брошюры из нескольких изображений «ковровым» покрытием фотографий.
4. Создайте макет трёхполосного буклета, используя готовые шаблоны.

ОПЕРАЦИИ

Печать буклета

1. В меню **Файл** выбрать команду *Печать*.
2. В открывшемся диалоговом окне выбрать нужный принтер, размер бумаги.



Проверь себя

Задание 1

1. Выберите макет открытки *День рождения*.
2. Выберите подходящий шаблон оформления.
3. Измените надпись на открытке.
4. Измените иллюстрацию на рисунок из файла ex1–1.
5. Примените цветовую схему, подходящую к цветам рисунка.
6. Сохраните файл в папке и под именем, которые укажет учитель.

Задание 2

1. Создайте пустую публикацию.
2. Откройте файл документа ex1–2 .
3. Выполните макет публикации для внутренней страницы брошюры. Вставьте текст из документа и иллюстрацию из файла ex1–3. Добавьте элементы страницы, если это покажется вам необходимым.
4. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

ПРИМЕНЯЕМ ЗНАНИЯ (необходимый уровень)

Задание 1

Создайте макет календаря по описанию.

1. Задайте временной интервал календаря – весь следующий год.
2. Выберите макет для календаря.
3. Разместите все месяцы на одной странице.
4. Измените иллюстрацию на рисунок из файла ex2–1.
5. Примените цветовую схему, подходящую к цветам рисунка.
6. Сохраните файл в папке и под именем, которые укажет учитель.

Задание 2

Создайте макет открытки по описанию:

1. Выберите макет открытки – *Домик*.
2. Выберите подходящий шаблон оформления.
3. Измените надпись на открытке.
4. Измените иллюстрацию на рисунок из файла ex2–2.
5. Примените цветовую схему, подходящую к цветам рисунка.
6. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

Задание 3

Создайте макет буклета по описанию.

1. Выберите макет для трёхполосного буклета *Событие*.
2. Разместите в буклете текст из файла ex2–3.
3. Измените иллюстрации на фотографии из папки 2–3.
4. Примените подходящую цветовую схему.
5. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

ПРИМЕНЯЕМ ЗНАНИЯ (повышенный уровень)

Задание 1

1. Создайте макет календаря на следующий год.

2. Сохраните файл в папке и под именем, которые укажет учитель.

Задание 2

1. Создайте макет открытки *Приглашение*.

2. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

Задание 3

Создайте макет листовки по описанию.

1. Создайте пустую публикацию.
2. Откройте файл документа ex3-1.
3. Поменяйте ориентацию страницы на альбомную.
4. Создайте текстовые блоки для заголовка и текста.
5. Скопируйте и вставьте из документа заголовок и текст.
6. Подберите подходящие к тексту иллюстрации из папки 3-1 и вставьте их на страницу.
7. Подберите шрифты и цвета для вашей публикации.
8. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

ПРИМЕНЯЕМ ЗНАНИЯ (максимальный уровень)

Задание 1

1. Создайте макет рекламного объявления: расположите на макете календарный блок на месяц, в котором будет происходить событие, выделите дату события, расположите текст и рисунки из файлов ex4-1, ex4-2.
2. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

Задание 2

1. Подумайте, какую информацию несёт почётная грамота. В каком стиле можно её создать?
2. Создайте макет почётной грамоты.
3. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

Задание 3

1. Создайте макет буклета. Продумайте его композицию, стиль, дизайн.
2. Вставьте текст из документа ex4-3 и иллюстрации из папки 4-3.
3. Добавьте элементы страницы, если это покажется вам необходимым.
4. Сохраните файл в папке и под именем, которые укажет учитель. Если есть техническая возможность, распечатайте публикацию.

Итоговая проверочная работа

Задание 1

Создайте макет буклета по описанию.

1. Откройте файл документа *ех5–1*.
2. Выберите подходящий макет буклета.
3. Скопируйте и вставьте из документа заголовок и текст.
4. Выберите шрифтовую схему для публикации.
5. Примените цветовую схему, подходящую к цветам рисунка.
6. Сохраните файл в папке и под именем, которые укажет учитель.

Задание 2

Создайте макет публикации по описанию.

1. Создайте пустую публикацию.
2. Откройте файл документа *ех5–2*.
3. Создайте текстовый блок вверху страницы для заголовка.
4. Скопируйте и вставьте из документа заголовок.
5. Создайте ещё два текстовых блока и поместите в них цитаты с подписями.
6. Создайте текстовый блок.
7. Скопируйте и вставьте из документа текст в этот текстовый блок.
8. Подберите подходящие к тексту иллюстрации из папки *5–2* и вставьте их на страницу.
9. Подберите шрифты и цвета для вашей публикации.
10. Сохраните файл в папке и под именем, которые укажет учитель.
Если есть техническая возможность, распечатайте публикацию.

Задание 3

1. Создайте черновой вариант рекламной брошюры вашей школы из 8 страниц. Обозначьте основные текстовые элементы, места для иллюстраций, подберите стили для оформления.
2. Сохраните файл в папке и под именем, которые укажет учитель.



Решаем жизненные задачи и работаем над проектами

Жизненная задача 1. Выпуск школьной газеты

Ваша роль: член редколлегии.

Описание. В школе появилась возможность издавать настоящую печатную газету. Вам доверили разработку макета страницы газеты.

Задание. Придумайте заголовок газеты, предложите оформление шапки и страницы (узнайте, где у газеты шапка). Подберите рисунки, которые постоянно будут находиться на первой странице обложки. Создайте макет газеты.

Жизненная задача 2. Создание календаря на год

Ваша роль: работник издательства.

Описание. Приближается следующий год. В издательство поступил заказ на создание красочного календаря с видами природы (возможно, с фотографиями школьных ситуаций).

Задание. Создайте календарь из 13 страниц [предположим, что потом он будет скреплён на пружине].

Жизненная задача 3. Сборник стихотворений (альманах)

Ваша роль: начинающий поэт.

Описание. Вы пишете стихотворения и хотите, чтобы их прочитало как можно больше людей.

Задание. Создайте небольшую брошюру – сборник своих стихотворений. Не забудьте добавить иллюстрации.

Жизненная задача 4. Рекламная кампания

Ваша роль: специалист по рекламе.

Описание: У вас есть небольшая партия слонов. Их надо продать.

Задание: Создайте рекламное объявление о продаже слонов.

Проект. Волшебный фонарь

Выясните, что такое «Волшебный фонарь» (*Laterna magica*). Проанализируйте примеры создания настроения в публикациях. Ищите идеи вокруг себя. Пример: скоро праздник День учителя; «Волшебный фонарь» посвящаем взаимопониманию людей (как жить без обид на старших).

Создайте пять страниц – «Волшебных фонарей» в таком сочетании текста и иллюстрации, с таким содержанием и оформлением, чтобы читатель захотел оставить эти страницы себе на память. Все пять страниц должны быть связанны. Связкой может служить тема, стиль рисунков, идея – что угодно.

Ваше решение – потребуется небольшой текстовый блок и рисунок (фото, коллаж). Можно идти от слов и к ним искать достойное оформление. А можно создать рисунок и подкрепить его текстом.

Подготовьте выступление и расскажите одноклассникам о результатах своих исследований. Если вам понравился проект, зайдите на образовательный портал «Школьная пресса» <http://portal.lgo.ru>.

Содержание

Дорогие ребята!	3
Модуль 1. Алгоритмизация и программирование	9
Введение	10
§ 1. Алгоритм	11
Что такое алгоритм?	
§ 2. Способы записи алгоритма	16
Каким образом люди записывают алгоритмы?	
§ 3. История языков программирования	23
Какие бывают языки программирования?	
§ 4. Работа в среде программирования	28
Как создавать и запускать программу на компьютере?	
§ 5–6. Циклы	36
Как записывать алгоритмы с повторами? Что такое циклы?	
§ 7. Отладка программ	45
Что делать с ошибками в программах?	
§ 8–9. Массивы	52
Как работать с однородными данными и не терять их?	
Модуль 2. Основы дизайна и печати изображений	69
Введение	70
§ 1. Основы композиции изображения	71
Как получить хорошее изображение, применяя правила композиции?	
§ 2. Цвет в графическом дизайне	75
Какие цвета сочетаются, какие цвета использовать в рисунке?	
§ 3. Создание коллажа	80
Как создать коллаж?	
§ 4. Сканирование изображений. Подготовка изображений к печати	88
Как подготовить изображение для печати?	
§ 5. Цветовые модели и палитры в компьютере	93
Как компьютер различает оттенок цвета?	
Модуль 3. Основы издательских технологий	107
Введение	108
§ 1. Макеты для публикаций	109
Как расположить все элементы на странице так, чтобы читатель мог быстро получить информацию?	
§ 2. Календари	113
Что такое календарь? Как его создать в печатном виде?	

§ 3. Открытки	121
Зачем нужны открытки и как их создать?	
§ 4. Создание своей публикации	126
Что надо знать и уметь, чтобы создать свой макет и расположить в нём тексты и иллюстрации?	
§ 5. Рекламные публикации	131
Что представляют собой виды полиграфической продукции постер, буклет, брошюра?	

**Горячев Александр Владимирович
Макарина Любовь Александровна
Павлоцкий Александр Владимирович
Платонова Наталья Сергеевна**

**Информатика
Учебник для 7-го класса
Книга 2**

Концепция оформления
и художественное редактирование – Е.Д. Ковалевская

Подписано в печать 28.05.12. Формат 70х90/16. Гарнитура Европа.
Печать офсетная. Бумага офсетная. Объём 9 п. л. Тираж 3000 экз. Заказ № 31843 (Sm-Sm).

Общероссийский классификатор продукции ОК-005-93, том 2; 953005 – литература учебная

Издательство «Баласс». 109147, Москва, Марксистская ул., д. 5, стр. 1
Почтовый адрес: 111123, Москва, а/я 2, «Баласс»
Телефоны для справок: (495) 368-70-54, 672-23-12, 672-23-34
<http://www.school2100.ru> E-mail: balass.izd@mtu-net.ru

Отпечатано в ОАО «Смоленский полиграфический комбинат»
214020, Смоленск, ул. Смольянинова, 1