К. Ю. Поляков, Е. А. Еремин

ИНФОРМАТИКА

8 класс

Москва
БИНОМ. Лаборатория знаний
2017

УДК 004.9
ББК 32.97
П54

Поляков К. Ю.

П54 Информатика. 8 класс / К. Ю. Поляков, Е. А. Еремин. — М. : БИНОМ. Лаборатория знаний, 2017. — 256 с. : ил.

ISBN 978-5-9963-3096-6

Учебное издание предназначено для изучения предмета «Информатика» в 8 классе (базовое и углублённое изучение). Входит в состав УМК по информатике для 7–9 классов, включающего авторскую программу, учебные издания, рабочие тетради, практикум, электронные ресурсы и методическое пособие.

Содержание учебного издания является продолжением курса 7 класса. В нём рассматриваются вопросы, связанные с кодированием информации, программированием, робототехникой; работа с табличными процессорами и подготовка электронных документов.

Главная задача учебного издания — обеспечить освоение базовых понятий информатики и принципов работы цифровой техники, что позволяет использовать его независимо от конкретных типов компьютеров и версий программного обеспечения.

Значительное внимание уделяется систематической подготовке школьников к государственной итоговой аттестации по информатике в форме основного государственного экзамена (ОГЭ).

Предполагается широкое использование ресурсов федеральных образовательных порталов, в том числе Единой коллекции цифровых образовательных ресурсов (http://sc.edu.ru/).

Соответствует федеральному государственному образовательному стандарту основного общего образования и примерной основной образовательной программе основного общего образования.

УДК 004.9
ББК 32.97

Учебное издание

Поляков Константин Юрьевич
Еремин Евгений Александрович

ИНФОРМАТИКА

8 класс

Ведущий редактор *О. Полежаева*
Ведущие методисты *И. Сретенская, И. Хлобыстова*
Художник *Н. Новак*
Технический редактор *Е. Денюкова*
Корректор *Е. Клитина*
Компьютерная верстка: *В. Носенко*

Подписано в печать 27.02.17. Формат 70x100/16. Усл. печ. л. 20,80.
Тираж 3000 экз. Заказ № м4293.

ООО «БИНОМ. Лаборатория знаний»
127473, Москва, ул. Краснопролетарская, д. 16, стр. 1,
тел. (495) 181-5344, e-mail: binom@Lbz.ru
http://www.Lbz.ru, http://metodist.Lbz.ru

Отпечатано в филиале «Смоленский полиграфический комбинат»
ОАО «Издательство «Высшая школа». 214020, Смоленск, ул. Смольянинова, 1
Тел.: +7 (4812) 31-11-96. Факс: +7 (4812) 31-31-70
E-mail: spk@smolpk.ru http://www.smolpk.ru

ISBN 978-5-9963-3096-6

От авторов

В этом году вы продолжите изучать предмет «Информатика», с которым познакомились в 7 классе или даже раньше. Информатика изучает всё, что связано с компьютерами: как они устроены, как работают и как заставить их делать то, что нужно человеку.

Если вы изучаете предмет углублённо, в *первой главе* вас ждёт краткое введение в робототехнику. Роботы постепенно входят в нашу повседневную жизнь, заменяя человека при выполнении опасных, сложных и неинтересных работ.

Все виды информации кодируются в памяти компьютера как цепочки нулей и единиц (битов). Во *второй главе* учебника вы узнаете, как именно это происходит: каким образом удаётся записать числа, тексты, рисунки, звуки и видеофильмы в двоичном коде; научитесь вычислять информационный объём данных — определять, сколько места они займут в памяти компьютера.

Одно из самых интересных направлений в информатике — программирование. В 7 классе мы писали программы для управления исполнителями, а в этом году (изучая *третью главу* учебника) займёмся обработкой числовой информации, ведь для компьютера все данные в памяти — это числа. В учебнике рассмотрены два языка программирования — уже знакомый вам школьный алгоритмический язык системы КуМир (будем называть его просто алгоритмический язык) и язык программирования Паскаль, на котором можно писать программы профессионального уровня. Какой из них изучать, выберет ваш учитель, но в любом случае у вас будет возможность посмотреть, как те же самые алгоритмы записываются на других языках.

В *четвёртой главе* вы познакомитесь с электронными таблицами — незаменимым инструментом офисных работников всех профессий: менеджеров, экономистов, бухгалтеров, секретарей и др. С помощью таких таблиц можно не только хранить данные, но и обрабатывать их по заранее введённым формулам, строить графики, составлять прогнозы на будущее.

Пятая глава посвящена подготовке сложных текстовых документов. Вы узнаете, как вставлять в документ математические формулы и диаграммы, форматировать страницы, составлять рефераты и совместно работать над одним документом в Интернете.

В учебнике есть основной материал (обязательный для изучения) и дополнительный (для углублённого курса). Материал для углублённого курса обозначен чёрными горизонтальными линиями, значком ⊕ в начале материала и значком ● — в конце, выделен шрифтом. Даже если вы изучаете информатику на базовом уровне, всегда можно заглянуть в дополнительные разделы учебника — вдруг там окажется что-то интересное.

Учебник — это не просто книга для чтения. Для того чтобы изучить предмет, нужно действовать: решать задачи, выполнять практические работы. Вы должны научиться «добывать» знания, выполняя различные эксперименты, пробуя и ошибаясь (без этого тоже нельзя!), проверяя догадки, делая выводы. Именно так работают учёные, открывая новые законы природы.

При чтении учебника мы советуем сразу выполнять задания, выделенные в тексте шрифтом и отступом. Эти задания рекомендуют вам перед тем, как продолжить чтение, ответить на вопрос, выполнить небольшое упражнение в тетради или провести исследование с помощью компьютера. Задания специально подобраны так, чтобы легче было понять новый материал. Тип задания обозначается на полях навигационным знаком  (вопрос), или  (письменное задание), или  (компьютерный эксперимент).

Для полноценной работы желательно использовать рабочую тетрадь (значок ) — в ней вы будете выполнять письменные задания.

Значок  говорит о том, что при выполнении задания придётся использовать кроме учебника дополнительные источники, например сеть Интернет. Проектные и исследовательские работы, которые выполняются дома, отмечены значком  .

Значок  означает важное определение или утверждение.

Значок  служит для выделения дополнительного задания или разъяснения.

Значок  означает групповую работу.

Значок  выделяет межпредметные связи.

Задания повышенной сложности отмечены «звёздочкой» (*).

В конце каждой главы вам предлагается список электронных образовательных ресурсов из Единой коллекции цифровых образовательных ресурсов (ЕК ЦОР) **www.school-collection.edu.ru**.

Электронные материалы к учебнику (описания практических работ и файлы для их выполнения, презентации, тесты, материалы для подготовки к итоговой аттестации) можно загрузить с сайта поддержки учебника:

http://kpolyakov.spb.ru/school/osnbook.htm

В заключение нам хочется поблагодарить наших коллег, которые взяли на себя труд прочитать предварительные версии отдельных глав учебника и высказать множество полезных замечаний, позволивших сделать учебник более точным, ясным и понятным:

- А. П. Шестакова, кандидата педагогических наук, зав. кафедрой информатики и вычислительной техники Пермского государственного педагогического университета;
- М. А. Ройтберга, доктора физико-математических наук, зав. лабораторией прикладной математики Института математических проблем биологии РАН, г. Пущино;
- С. С. Михалковича, кандидата физико-математических наук, доцента кафедры алгебры и дискретной математики ЮФУ, г. Ростов-на-Дону;
- Н. Д. Шумилину, андидата педагогических наук, доцента кафедры математики с методикой начального обучения Тверского государственного университета, г. Тверь;
- А. В. Паньгина, инженера Центра информационных технологий, г. Сосновый Бор;
- А. С. Башлакова, учителя информатики МОУ СОШ № 3, г. Унеча Брянской области;
- Н. П. Радченко, учителя информатики ГБОУ Школа № 1095, г. Москва;
- Ю. М. Розенфарба, учителя информатики МОУ Межозёрная СОШ, Челябинская область;
- О. А. Тузову, учителя информатики школы № 550, г. Санкт-Петербург;
- В. Н. Разумова, учителя информатики МОУ «Большеелховская средняя общеобразовательная школа», с. Большая Елховка, Республика Мордовия;

- А. В. Атанову, учителя информатики МАОУ СОШ № 12 им. маршала Советского Союза К. К. Рокоссовского, г. Великие Луки;
- Г. В. Роньжину, учителя информатики ГБОУ «Гимназия № 1519», г. Москва;
- А. В. Паволоцкого, учителя информатики ГБОУ «Гимназия № 1514», г. Москва;
- Н. Г. Неуймину, учителя информатики МАОУ «Лицей № 110» им. Л.К. Гришиной, г. Екатеринбург;
- Н. Е. Леко, учителя информатики МОУ СОШ № 9, г. Тихвин;
- И. А. Волкову, учителя информатики МОУ СОШ № 170, г. Екатеринбург;
- Н. С. Семашко, учителя информатики МБОУ «Лицей № 6», г. Дубна;
- С. В. Гриневича, учителя информатики МАОУ СОШ № 146, г. Пермь;
- Г. М. Шульгину, учителя информатики МОУ СОШ № 9, г. Пермь;
- Т. В. Дедюлькину, учителя информатики МАОУ «Гимназия № 5», г. Ростов-на-Дону;
- С. В. Гайсину, методиста ЛОИРО, г. Санкт-Петербург.

С уважением, авторы:
Константин Юрьевич Поляков,
Евгений Александрович Еремин.

Глава 1
РОБОТОТЕХНИКА

§ 1
Введение

***Ключевые слова*:**

- робот
- автономный робот
- андроид
- робототехника
- исполнительное устройство
- микроконтроллер
- датчик

Роботы и робототехника

Конечно, вы слышали слово «робот» и знаете, что роботом называют техническое устройство, которое может заменить человека во время выполнения сложных, утомительных или опасных работ.

Используя дополнительные источники, выясните, как появилось слово «робот». Кто его придумал?

Большинство роботов не похожи на человека. На современных заводах используется огромное количество промышленных роботов — станков с числовым программным управлением (ЧПУ). Такие станки обрабатывают детали по заложенной в них программе. Для того чтобы перестроить станок на изготовление другого типа деталей, достаточно просто заменить программу.

Роботы используются на конвейерных линиях, изготавливающих микросхемы для компьютеров: процессоры, память и др.

Всё более популярными становятся технологии 3D-печати, позволяющие с помощью роботов специального типа (3D-принтеров) послойно изготавливать различные детали.

Нас окружают автоматизированные системы, которые тоже можно назвать роботами, например система управления движением поездов метро, система управления отоплением дома.

Роботы, в отличие от людей, не устают, работают 24 часа в сутки, не болеют, могут работать при повышенной и пониженной температуре, в опасных для человека условиях. Во многих странах используются роботизированные установки пожаротушения. Уже продаются бытовые

роботы — автоматические пылесосы, которые могут убрать пыль и самостоятельно вернуться на место для подзарядки. В XXI веке роботам стали доверять даже выполнение некоторых хирургических операций.

Некоторые роботы похожи на человека, их называют **андроидами**. Самый известный робот-андроид *Asimo* (рис. 1.1) выпущен компанией *Honda*.

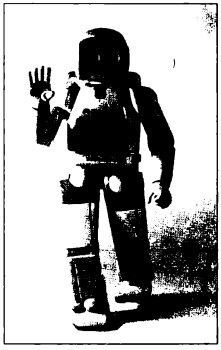


Рис. 1.1
(www.robotonline.net)

Существует особый класс роботов, которые работают «самостоятельно». Они называются **автономными роботами**. Это, например, беспилотные автомобили и летательные аппараты, роботы для исследования космоса и океана.

Полностью автономный робот может:

- перемещаться и работать длительное время без вмешательства человека;
- собирать информацию об окружающей среде;
- приспосабливаться к изменению обстановки, изменяя алгоритмы своей работы.

www Используя дополнительные источники, найдите ответы на вопросы.
— Зачем используются беспилотные автомобили Google?
— Что такое дрон?
— Какие автономные роботы применялись при исследованиях Луны и Марса?
— Какие риски появляются для человека и общества при использовании беспилотных автомобилей и летательных аппаратов?

Робототехника — это наука о разработке и использовании автоматизированных технических систем.

---

www Используя дополнительные источники, выясните, кто и когда впервые использовал слово «робототехника» и как оно записывается по-английски.

Робототехника применяет результаты таких наук, как механика, автоматика, кибернетика, информатика. Некоторые алгоритмы управления роботами используют элементы искусственного интеллекта.

www Известный писатель-фантаст, написавший множество рассказов о роботах и проблемах взаимомотношения с ними, сформулировал три основных закона робототехники. Используя дополнительные источники, определите имя и фамилию этого писателя. Запишите в тетрадь законы робототехники. Согласны ли вы с ними?

Из чего состоит робот?

Прежде всего робот — это механическое устройство. Поэтому его постройка — это инженерная, конструкторская работа. Нужно, чтобы все детали были хорошо закреплены, и робот не потерял равновесия и не развалился во время выполнения задания. В то же время все подвижные детали должны исправно двигаться и крутиться.

Роботом нужно как-то управлять, поэтому необходима **система управления**, которая в современных роботах строится на **микроконтроллерах**. Микроконтроллер — это миниатюрный компьютер, все части которого размещены на одном кристалле кремния. Он содержит процессор, разъёмы для управления внешними устройствами (порты), оперативную и постоянную память. В постоянную память микроконтроллера записана программа, которую он при включении сразу начинает выполнять.

Используя дополнительные источники, выясните, кто и когда впервые получил патент на микроконтроллер. www

Для того чтобы робот мог двигаться, нужны **исполнительные устройства** — моторы, которые преобразуют электрическую энергию в механическую энергию вращения. Автономные роботы чаще всего передвигаются с помощью колёс или гусениц, для неровных поверхностей иногда используют шагающие системы. Существуют также ползающие и плавающие роботы.

Передвижение роботов — это достаточно сложная самостоятельная задача. Например, современные роботы-спасатели должны уметь подниматься по лестнице здания.

Для того чтобы получать данные об окружающей обстановке, роботу нужны **датчики** (сенсоры, чувствительные элементы) — устройства, которые измеряют какую-то физическую величину и выдают информацию о ней в виде электрических сигналов. Поскольку эти сигналы затем будет обрабатывать цифровой компьютер, их нужно преобразовать в двоичный код, в числа. Эту работу выполняет аналого-цифровой преобразователь (АЦП), который, как правило, встроен в микроконтроллер.

Существуют различные типы датчиков (рис. 1.2): датчики температуры, давления, скорости, освещённости, датчики касания («кнопки»), датчики расстояния (они измеряют время, за которое ультразвук отражается от препятствия и возвращается обратно), инфракрасные датчики для поиска и обнаружения объектов и многие другие. Более сложные роботы имеют системы компьютерного зрения на основе видеокамер. Они могут «узнавать» простые предметы, определять их расположение в пространстве и достраивать невидимые части, используя информацию из своей базы данных.

Датчик освещённости | Датчик температуры | Датчик газа | Датчик движения | Датчик давления

Датчики из наборов *LEGO Mindstorms:*

Датчик касания | Датчик звука | Датчик расстояния | Датчик цвета

Рис. 1.2

Таким образом, встроенный микроконтроллер робота управляет исполнительными устройствами и обрабатывает данные, поступающие с датчиков (рис. 1.3).

Рис. 1.3

Используя дополнительные источники, выясните, какие микроконтроллеры применяются:

- в наборах *LEGO Mindstorms*;
- на платах *Arduino*, которые широко используются робототехниками-любителями.

На какой тактовой частоте они работают? Сравните её с тактовой частотой процессоров для настольных компьютеров.

Выводы

- Робот — это техническое устройство, которое может заменить человека во время выполнения сложных, утомительных и опасных работ.
- Автономные роботы способны собирать информацию об окружающей среде и работать длительное время без вмешательства человека.
- Робототехника — это наука о разработке и использовании автоматизированных технических систем.
- Системы управления роботов строятся на микроконтроллерах — миниатюрных компьютерах, все части которых размещены на одном кристалле кремния.
- Встроенный микропроцессор робота управляет исполнительными устройствами и обрабатывает данные, поступающие с датчиков.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. В каких ситуациях, на ваш взгляд, роботы не способны заменить человека?
2. Какие требования должны предъявляться к встроенным микроконтроллерам?
3. Используя язык высокого уровня, Петя написал программу для управления роботом, которая занимает 6 Мбайт. С какими проблемами он может столкнуться?
4. Если у робота откажут все датчики, к каким последствиям это может привести?
5. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Беспилотные транспортные средства»

б) «Боевые роботы»

в) «Роботы Big Dog»

г) «Роботы в медицине»

д) «Роботы-андроиды»

е) «Возможные проблемы использования роботов»

§ 2
Управление роботами

***Ключевые слова*:**

- контакты ввода и вывода
- порты
- команды управления
- команды обратной связи

Контакты ввода и вывода

Встроенный компьютер робота использует электрические сигналы для управления моторами, лампочками, устройствами вывода звука и т. п. Данные из внешнего мира (от датчиков) он тоже получает в виде электрических сигналов. Поэтому у него должны быть контакты ввода и вывода для связи с внешним миром. На рисунке 1.4 показаны контакты ввода и вывода — **пины** на плате *Arduino Uno*.

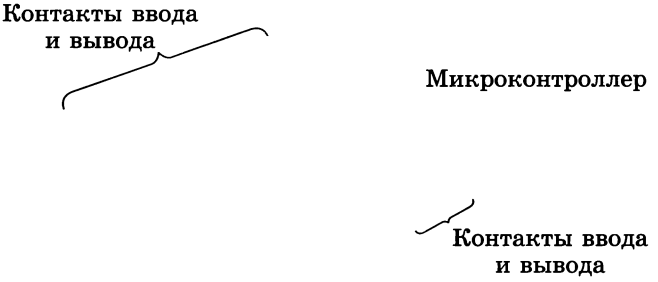


Рис. 1.4

Используя дополнительные источники, выясните, от какого слова образовано слово «пин».

Микроконтроллер может подать на любой контакт вывода электрический сигнал. Если мы подключим между этим контактом и контактом с сигналом «земля» («минусом») какую-нибудь лампочку, например светодиод, то при подаче сигнала она загорится, причём яркость лампочки будет зависеть от величины сигнала (точнее, от величины электрического напряжения между контактом и «землёй»). Светодиод имеет очень маленькое сопротивление, поэтому для того, чтобы ограничить силу тока, в цепь включают небольшой резистор (иначе светодиод может сгореть) — рис. 1.5.

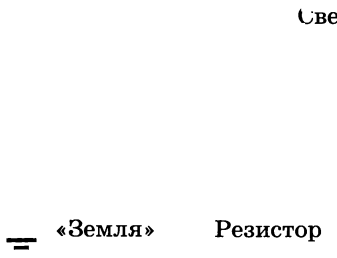


Рис. 1.5

Используя дополнительные источники, выясните, почему на электрических схемах и платах «земля» обозначается буквами *GND*.

Порты

Часто используют не отдельные контакты, а многоконтактные разъёмы — **порты**. На рисунке 1.6 показан управляющий блок (система управления роботом) из набора *LEGO Mindstorms EV3*, имеющий четыре порта вывода (для управления моторами) и четыре порта ввода для приёма данных с датчиков.

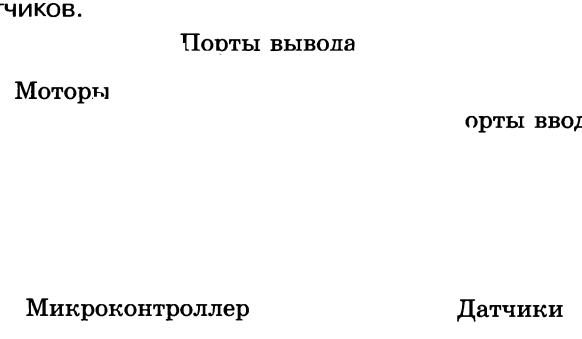


Рис. 1.6

Управление лампочками

При записи алгоритмов управления отдельными контактами (пинами) мы будем использовать алгоритмический язык, расширив его командой

```
записать(n, d)
```

Здесь `n` — это номер пина, а `d` — данные, которые определяют уровень сигнала на пине. Будем считать, что `d` — это целое число от 0 до 255, где 0 обозначает нулевой уровень сигнала (то же, что и «земля»), а 255 — максимальный уровень

Сколько бит нужно для хранения значения от 0 до 255? От 0 до 1023? от 0 до 65 535?

Рассмотрим схему на рис. 1.5, в которой лампочка подсоединена к пину 11. Если ввести и выполнить такую программу:

```
записать(11, 255)
ждать(1000)
записать(11, 0)
```

то лампочка загорится, будет гореть 1000 миллисекунд (пока работает команда `ждать`) и снова потухнет. Можно сделать, чтобы лампочка мигала бесконечно долго, включив эти команды в бесконечный цикл:

```
нц
  записать(11, 255)
  ждать(1000)
  записать(11, 0)
кц
```

В заголовке этого цикла нет ни условия, ни пределов изменений переменной. Это бесконечный цикл, который будет работать, пока мы не остановим программу. Такие циклы считаются плохим стилем в «обычном» программировании, но часто применяются при программировании роботов.

В только что рассмотренной программе есть ошибка. Исправьте её.

Система команд роботов

Управлять движущимся роботом можно вручную или с помощью программы, заранее записанной в память микроконтроллера. Список команд, которые может выполнять робот, обычно невелик. Как правило, можно:

- задать уровень мощности каждого мотора (например, в процентах от максимальной);
- выбрать направление вращения;
- включить мотор;
- выключить мотор;
- подождать некоторое время.

В более сложных системах есть возможность провернуть вал мотора на определённое число оборотов или градусов.

Для приёма данных с датчиков существуют **команды обратной связи**. Их действие сводится к тому, что встроенный компьютер читает числа, которые пришли с датчика на один из портов ввода. Что дальше делать с этими данными, определяет управляющая программа. Таким образом, алгоритмы работы роботов полностью определяются программным обеспечением и могут быть легко изменены.

Управление без обратной связи

В задачах с движущимися роботами мы будем управлять моторами, которые приводят в движение колёса или гусеницы. Расширим алгоритмический язык командой

```
мотор[n]:=d
```

Здесь `n` — это порт вывода, а `d` — мощность мотора в процентах, которая нам требуется (от 0 до 100%). Для вращения мотора в обратную сторону нужно задать отрицательную мощность — от −100 до 0%.

Будем обозначать порты для подключения моторов порядковыми номерами, начиная с нуля: 0, 1, 2 и т.д. Вот так можно включить мотор 0 на полную мощность на 2 секунды, а затем выключить:

```
мотор[0]:=100
ждать(2000)
мотор[0]:=0
```

Далее мы будем работать с роботом, у которого два мотора, связанные с двумя ведущими колёсами. Мотор 0 вращает левое колесо, а мотор 1 — правое.

Запишите в тетради программу, выполняя которую робот сначала проедет вперёд, потом — назад, потом — снова вперёд, и так будет продолжать бесконечно долго.

Что произойдёт, если включить один из моторов в одну сторону, а второй — в другую?

Сравните две программы и определите, чем отличаются действия Робота при их выполнении.

Программа 1:

```
мотор[0]:=100
мотор[1]:=-100
ждать(1000)
мотор[0]:=0
мотор[1]:=0
```

Программа 2:

```
мотор[0]:=100
мотор[1]:=0
ждать(1000)
мотор[0]:=0
мотор[1]:=0
```

Выводы

- Встроенный компьютер робота обменивается данными с внешним миром с помощью электрических сигналов.
- Для ввода и вывода данных используются контакты ввода и вывода — пины.
- На каждый из контактов вывода микроконтроллер может подавать сигналы разных уровней.

- Часто несколько контактов вывода объединяют в многоконтактный разъём — порт.
- При простейшем управлении моторами движущегося робота можно установить уровень мощности мотора и включить мотор на заданное время.
- Для приёма данных с датчиков существуют команды обратной связи. Их действие сводится к тому, что встроенный компьютер читает числа, которые пришли с датчика на один из портов ввода.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Что требуется для того, чтобы мотор можно было повернуть на заданное число оборотов?
2. Какие задачи управления движущимся роботом сложно решить, не используя сигналы с датчиков?
3. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Платы Arduino»
б) «Порты персональных компьютеров»

Практическая работа

Выполните практическую работу № 2 «Управление без обратной связи».

§ 3
Алгоритмы управления роботами

***Ключевые слова*:**

- порт ввода
- чтение из порта
- датчик касания
- датчик расстояния
- датчик освещённости
- ошибка измерения
- калибровка датчика
- движение по линии
- релейный регулятор
- П-регулятор

Управляющие кнопки

Простейший управляющий элемент — кнопка, с помощью которой включается или выключается какой-то режим работы робота. Например, с помощью кнопки можно управлять светодиодом: при нажатии

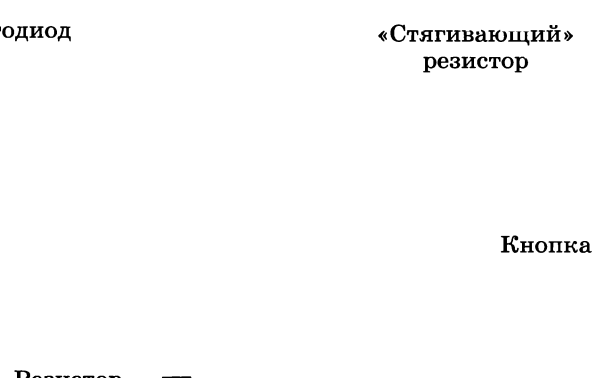


Рис. 1.7

кнопки лампочка загорается, после отпускания кнопки — гаснет. Схема, которую можно собрать на плате *Arduino*, показана на рис. 1.7.

На один контакт кнопки подаётся питание, второй подсоединён к входному пину, с которого микроконтроллер может прочитать значение сигнала. При нажатии кнопки цепь замыкается, так что на входной пин подаётся сигнал высокого уровня (величиной 5 вольт, на рис. 1.7 он обозначен как +5V). А какой сигнал будет на этом пине, если кнопка не нажата? Оказывается, в этом случае он может «плавать», т. е. меняться случайным образом из-за электромагнитных шумов. Чтобы этого не происходило, ставят специальный «стягивающий» резистор, который связывает входной пин с сигналом нулевого уровня (см. рис. 1.7).

Перейдём к программированию. Нам нужно постоянно читать значение напряжения на пине. Если подан высокий уровень сигнала (кнопка нажата), нужно включить лампочку, т. е. подать сигнал высокого уровня (255) на выходной пин, с которым она связана. Если кнопка не нажата, на выходной пин подаём сигнал низкого уровня (0), лампочка выключается.

Будем считать, что лампочка управляется пином 11, а сигнал от кнопки приходит на пин 2. Команда `прочитать` читает значение на входном пине, номер которого записан в скобках. Это функция, она возвращает значение, которое можно записать в переменную.

```
цел x
нц
  x:=прочитать(2)
  если x=255 то
    записать(11, 255)
  иначе
    записать(11, 0)
  все
кц
```

Запишите в тетради вариант программы, в котором нет оператора **если**.

Запишите в тетради программу для схемы, которая работает «наоборот» — когда кнопка нажата, лампочка не горит, а когда отпущена — лампочка включается. Постарайтесь найти два варианта решения задачи.

Движение в лабиринте

Теперь перейдём к программированию движущихся роботов в лабиринте. Роботу нужно как-то ориентироваться — определить, куда можно ехать, а куда нельзя. Если человеку завязать глаза, то он всё равно сможет обнаружить стену — на ощупь. У роботов для этого служат **датчики касания**. Они подают на входной порт значение 0, если касания нет, и значение 255, если касание произошло. Если перед роботом есть стена, то для того, чтобы доехать до стены, нужно использовать цикл «пока не сработал датчик касания».

Будем считать, что у платы управления роботом есть порты ввода с номерами 0, 1, 2 и т. д, и датчик касания подключён к порту 0. Программу движения до стены можно записать так:

```
мотор[0]:=100
мотор[1]:=100
нц пока датчик[0]<>255
  ждать(1)
кц
мотор[0]:=0
мотор[1]:=0
```

Мы дополнили язык программирования командой `датчик[0]`. Это функция, которая читает данные из порта 0, т. е. возвращает число, записанное датчиком в порт ввода 0. Команда `ждать` в теле цикла нужна для того, чтобы микроконтроллер смог при необходимости выполнить какие-то другие задачи, а не «зависал» во время выполнения цикла.

Как вы думаете, можно ли сразу разворачивать робота влево или вправо, если сработал датчик касания? Проверьте ваш вывод на тренажёре[1)] или при эксперименте с роботом.

---

1) Здесь и далее можно использовать тренажёры на сайте поддержки учебника:
http://kpolyakov.spb.ru/school/robotics/robotics.htm
http://kpolyakov.spb.ru/school/robotics/arduino.htm

Чаще всего не нужно допускать столкновения робота с препятствием — робота лучше остановить заранее. Для решения этой задачи датчик касания уже не подходит — нужно определять расстояние до ближайшей стены. Для этого используют датчики **расстояния**, чаще всего ультразвуковые, — их ещё называют **сонарами**.

Используя дополнительные источники, выясните, откуда произошло слово «сонар». Что оно означает?

Используя дополнительные источники, выясните, что такое ультразвук. Можно ли его услышать?

Ультразвуковые датчики расстояния состоят из излучателя и приёмника. Датчик излучает звуковую волну высокой частоты, она отражается от препятствия и возвращается обратно, где её «ловит» приёмник. Чем больше интервал между запуском волны и приёмом отражённого сигнала, тем больше расстояние.

Значение, поступающее на выходной порт с датчика расстояния[1)], — это целое число от 0 до 255. Как же нам определить само расстояние, например в сантиметрах? Для этого нужно прочитать описание этого датчика и выяснить, как вычислить расстояние по коду. Например, если датчик измеряет расстояния от 0 до 10 м, то расстояние в сантиметрах можно получить так:

$$S = d \cdot \frac{10 \cdot 100}{255} \text{ см},$$

где d — числовое показание датчика. Обычно максимальное кодовое значение (в нашем случае — 255) означает, что препятствий не обнаружено.

Определите, какому расстоянию соответствует показание датчика 100.

Определите, какое показание датчика соответствует расстоянию 20 см.

Пусть нам нужно остановить робота на расстоянии 0,3 м от стены. Вычислив соответствующее значение d, получаем 7,65. Но показание датчика должно быть целым числом, поэтому округляем это значение до 8.

У любого датчика есть неизбежная **ошибка измерения**, вызванная неидеальностью деталей и электромагнитными помехами от систем самого робота и других устройств. Обычно допустимая ошибка составляет одну единицу, т. е. показание датчика вместо 8 может быть равно 7 или 9. Для $d = 7$ получаем расстояние 27,45 см, а для $d = 9$ — расстояние 35,29 см. Это значит, что с помощью такого датчика мы определим

---

1) Некоторые реальные датчики выдают цифровое значение в диапазоне от 0 до 1023.

расстояние с ошибкой $\Delta S = \pm 5$ см. Относительная ошибка в процентах составит

$$\delta_S = \frac{5}{30} \cdot 100\% \approx 17\%.$$

Как вы думаете, при увеличении измеряемого расстояния, скажем, до 1 м, относительная ошибка будет уменьшаться или увеличиваться? Проверьте своё предположение вычислениями.

Новая модификация робота использует датчик расстояния, который выдаёт значение от 0 до 1023. Определите относительную ошибку при решении той же задачи (остановить робота на расстоянии 0,3 м от стены). Сделайте выводы.

Будем считать, что датчик расстояния «смотрит» вперёд (по направлению движения) и подключён к порту ввода 0. Программа, согласно которой робот едет вперёд и останавливается, очень похожа на предыдущую:

```
мотор[0]:=100
мотор[1]:=100
нц пока датчик[0]>8
  ждать(1)
кц
мотор[0]:=0
мотор[1]:=0
```

Объясните, почему не стоит записывать условие работы цикла в виде

```
нц пока датчик[0]<>8
  ...
кц
```

Движение по линии

В последние годы проводятся многочисленные соревнования роботов, в том числе Всемирные олимпиады. Классические задачи спортивной робототехники — движение робота по линии. Мы рассмотрим только одну задачу этого типа: робот должен ехать вдоль границы между чёрной и белой полосами. Движением робота управляет микроконтроллер, который выполняет программу, заранее записанную в его память. Такое управляющее устройство называется регулятором.

Регулятор — это автоматическое управляющее устройство, которое следит за состоянием робота и посылает ему управляющие сигналы.

Для того чтобы «увидеть» границу между чёрной и белой полосами, робот использует один или несколько датчиков освещённости (рис. 1.8).

Датчик освещённости
из набора LEGO NXT

Рис. 1.8

Датчик освещает поверхность светодиодом и измеряет количество отражённого света. Чёрные предметы отражают меньше всего света, белые — больше всего.

Датчик освещённости выдаёт код в интервале от 0 до 255, где 0 означает полное отсутствие отражённого света, а 255 — полное отражение. Однако не существует идеально чёрных и идеально белых предметов. Даже полоса, которая кажется нам совсем чёрной, отражает немного света, поэтому датчик покажет на ней не 0, а, например, 10. Также и на белой полосе вместо 255 мы получим, например, 230. Среднее арифметическое между этими значениями называют «средним значением серого». Теоретически такой сигнал выдаёт датчик, когда он находится точно над границей чёрной и белой полос (рис. 1.9). К этому состоянию мы и будем стремиться.

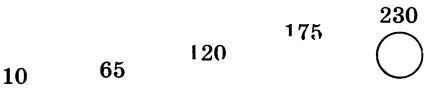


Рис. 1.9

Определение среднего значения серого в конкретных условиях называется **калибровкой датчика**.

Пусть требуется, чтобы слева от робота была белая полоса, а справа — чёрная (как на рис. 1.8). Для управления мы будем использовать очень простой принцип, который называют **релейным регулированием**. Показания датчика бóльшие, чем 120 (среднее значение серого), означают, что робот зашёл на белую полосу и нужно повернуть его вправо. Если датчик показывает меньше, чем 120, поворачиваем влево.

Используя дополнительные источники, узнайте, что такое реле. Откуда произошло это слово?

Запишите в тетради команды управления моторами для поворота влево и вправо. Мотор 0 находится с левой стороны робота, мотор 1 — с правой.

Получается такая программа, работающая в бесконечном цикле:

```
нц
  если датчик[0]>120 то
    мотор[0]:=100
    мотор[1]:=0
  иначе
    мотор[0]:=0
    мотор[1]:=100
  все
кц
```

Если запустить такую программу, мы увидим, что робот движется зигзагом, рывками. Дело в том, что мы задали ему такой алгоритм — резкое переключение между двумя состояниями. Это и есть релейное регулирование.

Для более плавного движения нужно создавать вращающий момент, который зависит от ошибки, т. е. от того, насколько робот отклонился от линии. Чем меньше ошибка, тем меньше вращающий момент. Если ошибки нет (датчик выдаёт значение, равное среднему значению серого), подаём одинаковую мощность на оба мотора, робот едет прямо.

Можно использовать, например, такой алгоритм управления:

```
вещ k=0.5
нц
  u:=k*(120-датчик[0])
  мотор[0]:=50-u
  мотор[1]:=50+u
кц
```

Изучите программу и определите, какой сигнал подаётся на моторы, когда датчик выдаёт значение: а) 100; б) 120; в) 140.

Выражение `120-датчик[0]` — это ошибка управления, т. е. отклонение значения, полученного с датчика, от заданного значения 120. Сигнал управления `u`, который изменяет мощность моторов, пропорционален ошибке, поэтому такой регулятор называется **пропорциональным**, или **П-регулятором**.

Коэффициент k — это **коэффициент усиления** П-регулятора. Правила выбора коэффициентов в законах управления изучает *теория автоматического управления*. Мы же будем выбирать коэффициенты экспериментально, не углубляясь в теорию.

Чем больше k, тем сильнее реакция робота на ошибку; чем меньше k, тем более плавно движется робот. Но при малых значениях k его реакция замедлена, и он может потерять линию на повороте.

Проведите эксперимент (на тренажёре или с реальным роботом) и выясните, при каком наименьшем коэффициенте k робот проходит всю трассу.

Проведите эксперимент и выясните, что происходит при выборе слишком большого значения k. При каком наибольшем коэффициенте k робот выполняет задание?

По результатам экспериментов выберите наилучшее, на ваш взгляд, значение k.

Выводы

- Для получения данных от внешних источников микропроцессор читает значения с портов, к которым подключены датчики.
- Для движения в помещении с препятствиями роботы могут использовать датчики касания и датчики расстояния (чаще всего ультразвуковые).
- Числовой код, полученный с датчика, нужно пересчитать в значение физической величины в соответствии с документацией на датчик.
- У датчика есть ошибка измерения, вызванная неидеальностью деталей и электромагнитными помехами от систем самого робота и других устройств.
- При движении робота по линии используется один или несколько датчиков освещённости.
- Для удержания робота на линии используют регуляторы — автоматические управляющие устройства.
- Релейный регулятор включает один из двух режимов движения в зависимости от сигналов с датчиков.
- Пропорциональный регулятор (П-регулятор) вырабатывает сигнал управления, пропорциональный ошибке управления. Для выбора коэффициента усиления П-регулятора используют методы теории автоматического управления или эксперимент.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Почему в программах управления роботами часто используется бесконечные циклы?
2. У робота вышел из строя датчик касания. Можно ли использовать вместо него датчик расстояния, который остался работоспособным?
3. Как нужно установить датчик расстояния, чтобы робот мог двигаться параллельно стенке?
4. У вас оказался датчик расстояния, на который нет документации. Как вы будете его использовать?
5. Как нужно установить два датчика освещённости, если нужно обеспечить движение робота вдоль тонкой чёрной линии на белом поле?
6. Почему на чёрной линии датчик освещённости выдаёт ненулевое значение?
7. Зависит ли результат, который выдаёт датчик освещённости, от внешнего освещения?
8. Какой регулятор обеспечивает более плавное движение: релейный или пропорциональный?
9. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Соревнования роботов»
б) «Кегельринг»

Практические работы

Выполните практические работы:
№ 3 «Использование датчиков»;
№ 4 «Движение робота по линии».

ЭОР к главе 1 из Единой коллекции цифровых образовательных ресурсов (school-collection.edu.ru)

Андроид-фокусник (видеофрагмент)
Робот адаптивный с техническим зрением (видеофрагмент)
Малые мобильные роботы (видеофрагмент)
Робот для починки трубопроводов
Шестиногое шагающее устройство

Глава 2
КОДИРОВАНИЕ ИНФОРМАЦИИ

§ 4
Язык — средство кодирования

Ключевые слова:

- язык
- алфавит
- мощность алфавита
- формальный язык

Кодирование — это один из видов обработки информации. При кодировании меняется форма представления информации, а её содержание сохраняется.

Язык и алфавит

Для того чтобы хранить и передавать информацию, её необходимо как-то закодировать, например записать с помощью знаков (символов) на каком-то *языке*.

Кодирование — это представление информации в форме, удобной для её хранения, передачи и автоматической обработки.

Код — это правило, по которому сообщение преобразуется в цепочку знаков.

Язык — это система знаков и правил, используемая для записи и передачи информации.

Естественные языки (русский, английский и др.) сформировались в результате развития человеческого общества и используются для общения людей.

Сначала древние люди овладели устной речью. Поскольку человек может издавать и различать на слух не так много звуков, он стал комбинировать их, составляя слова, каждому из которых приписывался некоторый смысл.

Затем люди стали записывать информацию, например, для передачи потомкам. В первое время жизненный опыт пытались зафиксировать в виде рисунков животных и предметов, затем пиктограмм (схематических изображений), иероглифов (рис. 2.1).

Египетское письмо	
	рука
	дом
	кобра
	лев
	вода
	рот
	мужчина
	женщина

Иероглифы (Китай)	
日	солнце
月	луна
雨	дождь
山	гора
马	лошадь
鱼	рыба
人	человек
女	женщина

Рис. 2.1

В большинстве современных языков используется *алфавитное письмо*, где каждый знак (или сочетание знаков) обозначает некоторый звук, так что с помощью небольшого набора знаков (*алфавита*) можно записать любые слова устной речи.

Алфавит — это набор знаков, который используется в языке.

Обычно знаки в алфавите расположены в определённом порядке.

Вспомните, сколько знаков входит в русский и английский алфавиты.

К алфавиту языка, вообще говоря, нужно отнести пробел (пропуск между словами), цифры (знаки для записи чисел), знаки препинания, скобки.

Мощность алфавита — это количество знаков в алфавите.

Определите мощность алфавита, состоящего из русских букв, цифр, пробела и знаков препинания (точка, запятая, точка с запятой, вопросительный и восклицательный знаки, тире, двоеточие, многоточие, кавычки, круглые скобки).

Естественные и формальные языки

В любом естественном языке есть исключения из правил и есть неоднозначности. Например, одно и то же слово может иметь различный смысл.

Что означает слово «рукав» в следующих предложениях на русском языке?

- Дельта Волги делится на множество рукавов.
- Рукав слишком длинный, но его легко укоротить.

Как вы это определили?

Смысл слова часто можно установить только из *контекста*, т. е. отрывка текста, в котором оно употребляется. Часто, например в научных публикациях, такая ситуация недопустима, потому что смысл текста должен быть понят однозначно. В таких случаях используют языки специального типа, в которых каждое слово и словосочетание имеют чётко определённое единственное значение и нет никаких исключений.

Формальный язык — это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.

Вот некоторые примеры формальных языков:

- математические формулы: $S = v \cdot t$;
- правила записи чисел: 12345;
- нотная запись: 
- язык записи шахматных партий: **1. e2-e4 e7-e5 ...**

- алгоритмический язык:

```
алг Программа
нач
  вывод 'Привет, Вася!'
кон
```

Все формальные языки — *искусственные*, они созданы людьми. В таблице 2.1 сравниваются естественные и формальные языки.

Таблица 2.1

Естественные языки	Формальные языки
Сформировались в результате развития общества	Созданы людьми специально
Используются для общения в быту	Используются в специальных областях знаний
Часто встречаются слова с неточным и неясным содержанием	Нет слов с неточным и неясным содержанием
Значения отдельных слов и предложений зависят не только от них самих, но и от их окружения (контекста)	Значения слов и предложений не зависят от контекста
Встречаются *синонимы* (разные слова имеют одинаковый смысл)	Как правило, синонимов нет
Встречаются *омонимы* (одно слово может иметь несколько значений)	Омонимов нет
Нет строгих правил образования предложений	Правила образования предложений строго определены
Для многих правил существуют исключения	Нет исключений из правил

Сообщения и их количество

С точки зрения теории информации, сообщение — это любой набор знаков некоторого алфавита. Пусть мы хотим отправлять различные сообщения одинаковой длины, используя какой-то алфавит. Конечно, чем короче будет длина сообщений, тем быстрее

можно будет их передать. Но вместе с тем если сообщения будут слишком короткими, то количество различных сообщений может оказаться недостаточным. Например, из двух двоичных цифр можно составить только четыре разных сообщения: 00, 01, 10 и 11 — больше, как ни комбинируй, не получится.

Рассмотрим алфавит из четырёх знаков: @#$%. Постройте все возможные сообщения из одного знака. Постройте все возможные сообщения из двух знаков, которые начинаются с буквы @ (вторая буква может быть любой).

Для алфавита @#$% в сообщении из двух знаков первый знак можно выбрать четырьмя способами, и для каждого из них есть 4 варианта выбора второго знака. Поэтому сообщений, состоящих из двух знаков, будет $4^2 = 16$ (рис. 2.2).

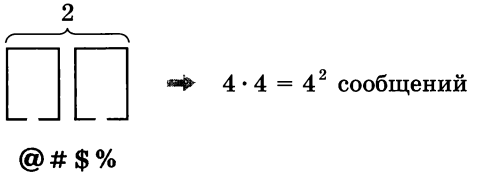


Рис. 2.2

Для алфавита из четырёх знаков определите:
а) количество сообщений из двух знаков;
б) количество сообщений из трёх знаков;
в) количество сообщений из четырёх знаков;
г) количество сообщений из L знаков.

Для алфавита из M знаков определите:
а) количество сообщений из двух знаков;
б) количество сообщений из трёх знаков;
в) количество сообщений из четырёх знаков;
г) количество сообщений из L знаков.

Если алфавит языка состоит из M знаков (имеет мощность M), количество различных сообщений длиной L знаков вычисляется как

$$N = M^L.$$

Для двоичного алфавита (его мощность равна $M = 2$), получается:

$$N = 2^L.$$

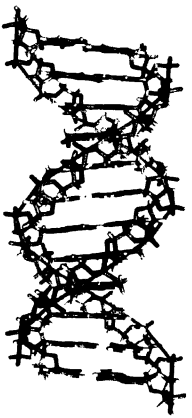
Алфавит языка содержит буквы «А» и «У». Определите, сколько сообщений из трёх знаков можно записать с помощью этого языка.

Алфавит языка содержит буквы «А», «О» и «У». Определите, сколько сообщений длиной не больше четырёх знаков можно записать с помощью этого языка.

Если длина сообщений может меняться (может быть равна L_1, L_2, ..., L_K), то для вычисления общего количества различных сообщений нужно сложить количества сообщений для каждой возможной длины:

$$N = N_1 + N_2 + ... + N_K.$$

Генетический код



Почему у кошек рождаются котята, а у собак — щенята? Дело в том, что родители передают детям наследственную информацию. Она определяет, из каких белков будет построен новый организм, каков будет его внешний вид и внутреннее устройство.

В середине XX века учёные выяснили, что эта наследственная информация хранится в молекулах ДНК (так сокращённо называют *дезоксирибонуклеиновую кислоту*). Эту молекулу обычно рисуют в виде очень длинной двойной спирали, т. е. спирали из двух цепей. Каждая цепь состоит из звеньев четырёх типов (*нуклеотидов*), вот их русские и английские названия:

аденин (Adenine, A);

цитозин (Cytosine, C);

гуанин (Guanine, G);

тимин (Thymine, T).

Для сокращения эти звенья обычно обозначают начальными буквами английских названий: A, C, G и T. Таким образом, молекула ДНК — это «сообщение», закодированное в четырёхбуквенном алфавите.

Часть молекулы ДНК (у человека — примерно 3% всей длины) составляют *гены* — участки, содержащие информацию о белках, которые будет строить организм. Белки — это основа жизни, они определяют все биологические свойства организма.

Каждый белок — это тоже цепь, состоящая из «кусочков», которые называются *аминокислотами*. Белки живых организмов строятся из 20 видов аминокислот.

Какую длину равномерного[1] кода, использующего алфавит {A, C, G, T}, нужно выбрать, чтобы можно было закодировать 20 различных видов аминокислот?

Сколько различных последовательностей длиной 2 и 3 можно составить с помощью алфавита {A, C, G, T}?

Гены в молекуле ДНК состоят из различных троек нуклеотидов, причём каждая тройка обозначает определённую аминокислоту. Кроме того, специальные тройки обозначают начало и конец гена.

Молекула ДНК человека состоит примерно из трёх миллиардов нуклеотидов. Исследование таких длинных последовательностей вручную невозможно, здесь необходима помощь компьютера. Поэтому на стыке биологии и информатики возникла наука биоинформатика, которая занимается анализом генов и белков. Сравнивая строение белков, учёные устанавливают родственные связи между видами животных и растений, выясняют причины болезней и учатся бороться с ними.

Выводы

- Кодирование — это представление информации в форме, удобной для её хранения, передачи и автоматической обработки.
- Код — это правило, по которому сообщение преобразуется в цепочку знаков.
- Язык — это система знаков и правил, используемая для записи и передачи информации.
- Алфавит — это набор знаков, который используется в языке. Обычно знаки в алфавите расположены в определённом порядке.
- Мощность алфавита — это количество знаков в алфавите.
- Сообщение — это любой набор знаков какого-то алфавита.
- Если алфавит языка состоит из M знаков (имеет мощность M), количество различных сообщений длиной L знаков вычисляется как $N = M^L$.
- Формальный язык — это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.

[1] Как вы знаете из курса информатики 7 класса, равномерным называется код, в котором все кодовые слова имеют одинаковую длину.

Интеллект-карта

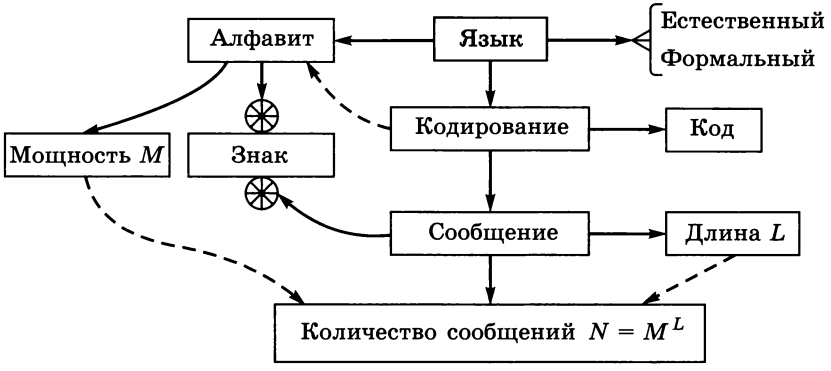


Рис. 2.3

Какие сведения из параграфа вы бы добавили в эту схему?

Вопросы и задания

1. Как вы думаете, почему алфавиты большинства современных языков содержат небольшое число знаков?
2. В чём, на ваш взгляд, достоинства и недостатки использования иероглифов?
3. В каких областях нужны формальные языки? Приведите примеры формальных языков, о которых не упоминалось в тексте учебника.
4. Почему меню, которое появляется при нажатии правой кнопки мыши над объектом, называют контекстным?
5. Почему любой язык программирования — это формальный язык?
6. Почему люди не отказываются от естественных языков и не переходят на формальные во всех областях?
7. Выберите из таблицы 2.1 те свойства естественных языков, которые затрудняют и не позволяют полностью автоматизировать перевод с одного языка на другой. Как вы рассуждали?
8. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Язык эсперанто»
б) «Правила сложения и умножения в комбинаторике»
в) «Генетический код»

§ 5
Дискретное кодирование

Ключевые слова:

- дискретизация
- равномерный код
- неравномерный код
- декодирование
- условие Фано
- код Морзе

Дискретизация

В чём принципиальное различие между картиной, нарисованной красками, и мозаикой?

Давайте подумаем, что на самом деле происходит, когда мы записываем информацию с помощью какого-либо алфавита. При этом информация, существовавшая ранее у нас в сознании в виде мыслей, записывается в виде отдельных «кусочков», знаков. Так же и линия, нарисованная на бумаге, при сканировании представляется в памяти компьютера в виде отдельных элементов — пикселей. Такая процедура называется дискретизацией.

Дискретизацию мы используем и в жизни. Например, когда измеряют температуру воздуха, обычно округляют её до целых градусов, хотя температура изменяется непрерывно, а не скачками: она может быть равной и 18,25 °C, и 18,251 °C, и 18,2513 °C и т. д. Математики говорят, что множество дробных чисел *непрерывно*, потому что между двумя любыми дробными числами находится бесконечно много других дробных чисел. В то же время множество целых чисел *дискретно*, потому что между двумя целыми числами находится конечное число других целых чисел, и его легко подсчитать. Таким образом, при округлении мы выполняем дискретизацию данных.

Дискретизация — это представление непрерывного объекта в виде множества отдельных элементов.

Картина художника — это *непрерывный* объект, а мозаика, сделанная на её основе, — дискретный. Переход от наскальных рисунков к алфавитному письму — это тоже переход от непрерывного способа представления информации к дискретному.

Все приборы, которые показывают результаты измерений в цифровом виде, выполняют дискретизацию. Например, стрелка в обычном спидометре автомобиля может принимать любое положение, это непрерывный (или, как говорят физики, *аналоговый*) прибор. А цифровой спидометр показывает дискретные данные — скорость с округлением до 1 км/ч (рис. 2.4).

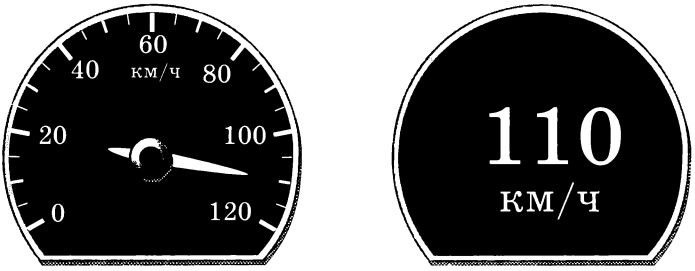


Рис. 2.4

Может ли цифровой спидометр показать скорость 110,231 км/ч? Почему?

Обратите внимание, что в результате дискретизации мы *теряем информацию*. Заменив картину художника мозаикой, мы сделали её более грубой, потеряли тонкие детали. Но часто потеря информации допустима. Например, при округлении температуры вместо 18,2513 °C мы получили 18 °C, но нам этого достаточно для решения бытовых задач.

Как вы знаете, все виды информации в компьютере представлены в двоичном коде, как цепочки нулей и единиц. Это не случайно, потому что для хранения каждого бита в компьютере используется электронный блок с двумя состояниями. Поэтому компьютер — это дискретное устройство.

Для того чтобы ввести данные в компьютер, нужно выполнить их дискретизацию, например представить текст как набор букв, а рисунок — как набор пикселей. Затем каждому элементу (букве, пикселю) нужно присвоить двоичный код — битовую цепочку. Как это делается и какие бывают коды, вы узнаете далее.

Равномерные коды

Если нам нужно записать в память компьютера какой-то текст на русском языке, его нужно представить в виде двоичного кода, т. е. *перекодировать*.

Например, перекодируем слово ГАГАРА в двоичный алфавит, считая, что в тексте есть только буквы «А», «Г» и «Р», т. е. алфавит состоит из трёх знаков. Присвоим каждой из этих букв двоичные коды — **кодовые слова** (рис. 2.5).

А	Г	Р
000	010	100

Рис. 2.5

Закодируйте с помощью этого кода слово ГАГАРА.

Такой код называется равномерным, потому что длина всех кодовых слов одинакова.

Равномерный код — это код, в котором все кодовые слова имеют одинаковую длину.

Теперь предположим, что по компьютерной сети передана цепочка

000010000100000010000100

Известно, что для кодирования использовалась таблица, показанная на рис. 2.5, и нам нужно узнать, какое сообщение было закодировано. Эта операция называется декодированием.

Декодирование — это восстановление исходного сообщения из кода.

Сообщение 000010000100000010000100 закодировано с помощью равномерного кода, приведённого на рис. 2.5. Определите, сколько знаков было в исходном сообщении. Как вы рассуждали? Декодируйте это сообщение.

Равномерный 5-битный двоичный код, разработанный в конце XIX века Жаном Морисом Бодо, использовался в телеграфных аппаратах. В современных компьютерных системах при передаче текстовых сообщений также часто применяют равномерный (8-битный или 16-битный) код.

Можно ли было для кодирования букв «А», «Г», «Р» использовать более короткий равномерный код? Определите наименьшую возможную длину кодовых слов.

Если для кодирования используется алфавит мощностью M, то с помощью кодовых слов длиной L можно закодировать M^L различных знаков. Это число должно быть не меньше, чем мощность алфавита исходного сообщения M_0, потому что иначе какие-то буквы обязательно получат одинаковые коды.

Длину кодовых слов L выбирают из условия $M^L \geq M_0$, где M_0 — мощность алфавита исходного сообщения и M — мощность нового алфавита.

Как выбрать наименьшую возможную длину кодовых слов при равномерном кодировании?

В сообщении используются 33 русские прописные буквы и пробел. Определите наименьшую длину кодовых слов для равномерного кодирования этого сообщения в трёхбуквенном и четырёхбуквенном алфавитах.

Неравномерные коды

Недостаток равномерного кода в том, что закодированные сообщения получаются довольно длинными, и на их передачу компьютерная система тратит много времени. Попробуем сократить длину сообщения, используя кодовые слова разной длины, например так (рис. 2.6).

А	Г	Р
0	1	10

Рис. 2.6

Такой код называется неравномерным.

Неравномерный код — это код, в котором кодовые слова имеют различную длину.

Закодируйте с помощью этого кода слово ГАГАРА. При использовании равномерного или неравномерного кода закодированное сообщение получилось короче?

Декодируйте сообщение 010010, закодированное с помощью кода на рис. 2.6. Попробуйте построить разные варианты декодированного сообщения.

Сообщения, закодированные с помощью неравномерного кода, не всегда можно декодировать однозначно.

Есть ли такие неравномерные коды, сообщения в которых однозначно декодируются? Оказывается, есть. Например, такой код (рис. 2.7).

А	Г	Р
0	10	11

Рис. 2.7

Закодируйте с помощью этого кода слово ГАГАРА. Сравните длины полученного закодированного сообщения и сообщений, которые вы построили ранее. Какое из них самое короткое? Самое длинное?

Декодируйте сообщение 01001011, закодированное с помощью кода на рис. 2.7. Попробуйте построить различные варианты декодирования.

Неравномерный код декодируется однозначно, если выполняется **условие Фано**: ни одно кодовое слово не совпадает с началом другого кодового слова.

Для кода на рис. 2.7 условие Фано выполняется: код буквы «А» (0) не совпадает с началом остальных кодовых слов, то же выполняется и для остальных букв.

Код Морзе

Долгое время для передачи сообщений по телеграфу и радио применялся **код Морзе (азбука Морзе)**, предложенный американским художником и изобретателем Самюэлем Морзе. В этом коде все буквы и цифры кодируются в виде различных последовательностей точек и тире.

Код Морзе для русских букв и цифр

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

С. Морзе (1791–1872)

Определите, к каким кодам относится код Морзе — к равномерным или неравномерным. Как вы рассуждали?

Как вы думаете, какие буквы должны иметь более короткие коды: те, которые в текстах встречаются чаще, или те, которые встречаются реже?

Чтобы узнать, как часто встречается каждая буква в текстах, Морзе посетил типографию и подсчитал количество используемых литер с изображениями разных букв. Поэтому английская буква «E», которая встречается в текстах чаще всего, получила код •. Коды Морзе для русских букв совпадают с кодами похожих по звучанию английских букв, например коды букв «Л» и «L» одинаковы[1)].

Проверьте, выполняется ли для кода Морзе условие Фано.

Чтобы отделить последовательности (коды букв) друг от друга, вводят ещё один знак — пробел (пауза). Если бы не было разбивки на буквы, текст было бы невозможно декодировать однозначно.

---

1) Поэтому код Морзе для русских букв менее эффективен.

Работа в парах. Закодируйте какое-нибудь слово из 4–5 букв с помощью кода Морзе и предложите своему напарнику декодировать его.

Получено сообщение •––•–, закодированное с помощью кода Морзе, но все паузы пропущены. Декодируйте это сообщение всеми возможными способами. Сколько различных вариантов вы нашли?

Измерение количества информации

Давайте вспомним, как измеряется количество информации (данных) в компьютерных системах. Сообщение кодируется с помощью двоичного кода, и количество информации в битах определяется как длина полученной битовой цепочки.

Сколько бит информации содержится в сообщении 10100011?

Сколько знаков содержит алфавит двоичного кода?

Определите, сколько различных сообщений можно закодировать с помощью i бит.

На практике используются следующие единицы количества информации:

1 байт = 8 бит.
1 Кбайт = 1024 байта = 2^{10} байта = 2^{13} бит.
1 Мбайт = 1024 Кбайт = 2^{20} байта = 2^{23} бит.
1 Гбайт = 1024 Мбайт.
1 Тбайт = 1024 Гбайт.

Выводы

- Дискретизация — это представление непрерывного объекта в виде множества отдельных элементов.
- Компьютеры обрабатывают информацию в двоичном коде, в котором используется два знака (обычно 0 и 1).
- Данные, с которыми работает компьютер, — дискретные. Для того чтобы информацию можно было обрабатывать с помощью компьютеров, её нужно дискретизировать — перевести в дискретный вид. Как правило, дискретизация приводит к потере части информации.
- Равномерный код — это код, в котором все кодовые слова имеют одинаковую длину.

- При равномерном кодировании длину кодовых слов L выбирают из условия $M^L \geq M_0$, где M_0 — мощность алфавита исходного сообщения и M — мощность нового алфавита.
- Неравномерный код — это код, в котором кодовые слова имеют различную длину.
- Декодирование — это восстановление исходного сообщения из кода.
- Сообщения, закодированные с помощью неравномерного кода, не всегда можно декодировать однозначно.
- Неравномерный код декодируется однозначно, если выполняется условие Фано: ни одно кодовое слово не совпадает с началом другого кодового слова.
- С помощью i бит можно закодировать 2^i различных сообщений.

Интеллект-карта

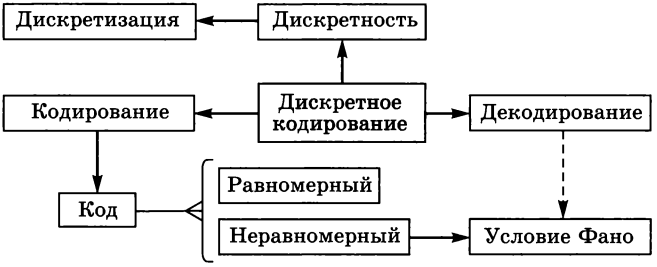


Рис. 2.8

Какие сведения из параграфа вы бы добавили в эту схему?

Вопросы и задания

1. Расскажите о том, какие преимущества нам даёт дискретизация. Какие у неё недостатки?
2. Как вы думаете, какие приборы лучше: цифровые или аналоговые? Какие достоинства и недостатки имеет каждый из этих типов? Обсудите этот вопрос в классе.
3. Почему при дискретизации, как правило, происходит потеря информации? В каких случаях потери информации не будет?
4. Как можно уменьшить потери информации при дискретизации?
5. Приведите примеры, когда одна и та же информация может быть представлена в аналоговой и дискретной форме.

6. Сравните равномерные и неравномерные коды: какие достоинства и недостатки имеет каждый тип?
7. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Телеграфные коды»
б) «QR-коды»
в) «Обратное условие Фано»

Интересные сайты

telegraphist.ru/book/morse-translator — онлайн-переводчик азбуки Морзе

Проекты

а) Выясните, какие музыкальные инструменты позволяют извлекать только дискретные звуки (заранее определённые ноты), а какие — звуки любой частоты.
б) Найдите в Интернете часто́ты встречаемости русских букв. Предложите неравномерный код, который при кодировании русского текста даёт в среднем более короткое сообщение, чем код Морзе.

§ 6
Кодирование с обнаружением ошибок

***Ключевые слова*:**

- коды с обнаружением ошибок
- избыточность
- бит чётности
- контрольная сумма
- помехоустойчивые коды

В реальных каналах связи всегда присутствуют помехи, искажающие полезный сигнал. В некоторых случаях ошибки допустимы, например при прослушивании радиопередачи через Интернет небольшое искажение звука не мешает понимать речь. Однако чаще всего требуется обеспечить точную передачу данных. Например, ошибка в передаче одной команды компьютерной программы может привести к катастрофе космического корабля. В таких случаях в первую очередь нужно обнаружить ошибку и, если это произошло, передать блок данных ещё раз. Как мы увидим, в некоторых случаях даже удаётся исправить небольшое число ошибок без повторной передачи данных.

Коды с обнаружением ошибок

Представьте себе, что получена цепочка нулей и единиц 1010101110, причём биты независимы: каждый бит цепочки может быть нулём или единицей независимо от других. В этом случае нет абсолютно никакой возможности определить, верно ли передана последовательность. Поэтому для того, чтобы обнаружить ошибку, в передаваемое сообщение нужно добавлять какую-то дополнительную информацию («лишние» биты), т. е. вводить *избыточность*.

Простейший вариант — добавить в конец блока данных дополнительный бит, который будет равен 1, если в основном сообщении нечётное число единиц, и равен 0 для сообщения с чётным числом единиц. В результате в новом блоке всегда будет чётное число единиц. Этот дополнительный бит называется *битом чётности*. Бит чётности используется при передаче данных в компьютерных сетях.

К каждому из этих двухбитных сообщений добавьте бит чётности (так, чтобы общее число единичных битов в каждом сообщении стало чётным):

00 01 10 11

Сообщения передаются с битом чётности. Какие из этих сообщений были переданы с ошибкой?

00101 0011010101 1001001010011

Как вы рассуждали?

Как вы думаете, можно ли вместо бита чётности добавлять «бит нечётности», который делает общее число единичных битов в блоке данных нечётным?

К каждому из этих сообщений добавьте бит чётности:

1000 0110 1011101011 11111111

При передаче сообщения с битом чётности произошли две ошибки. Сможет ли обнаружить ошибку принимающая сторона? А если ошибок будет три? Четыре?

Получено сообщение с битом чётности, в котором оказалось нечётное число единиц. Можно ли сказать, в каком именно бите произошла ошибка?

Если при передаче изменились два бита, чётность не меняется, и такая ошибка не обнаруживается. Однако на практике две ошибки в одном небольшом (например, 8-битном) блоке данных могут появиться очень редко.

Применение бита чётности позволяет обнаруживать нечётное число ошибок (1, 3, 5, ...), а ошибки в чётном количестве разрядов остаются незамеченными.

При передаче больших файлов может сразу возникнуть множество ошибок, поэтому используют другой метод — вычисляют *контрольную сумму* файла (по достаточно сложным алгоритмам), которая передаётся вместе с данными. Изменение даже одного бита данных сильно изменяет контрольную сумму. Если контрольная сумма файла, вычисленная приёмником, не совпадает с контрольной суммой, записанной передающей стороной, то произошла ошибка.

Коды с исправлением ошибок

Значительно сложнее исправить ошибку сразу (без повторной передачи данных), однако в некоторых случаях и эту задачу удается решить. Для этого ещё больше увеличивают избыточность кода (добавляют «лишние» биты).

Когда вы говорите по телефону, иногда приходится повторять какие-то фразы, если собеседник не понял вас из-за помех. Эту идею можно использовать и для компьютеров, например применить код, в котором каждый бит повторяется трижды: вместо каждого нуля будем передавать кодовое слово 000, а вместо каждой единицы — кодовое слово 111.

При передаче сообщения каждый передаваемый бит повторяется три раза подряд. Могли ли быть переданы без ошибок такие сообщения?

1) 000111111111;
2) 000101000111000;
3) 111000111010111.

Как вы рассуждали?

При передаче каждой тройки одинаковых битов произошло не более одной ошибки. Какие биты пытались передать, если получены тройки битов:

001 010 011 100 101 110?

Как вы рассуждали?

Передавались сообщения, в которых каждый бит был утроен. При передаче произошли ошибки (не более одной в каждой тройке), и были получены следующие битовые цепочки:

1) 000011110101010;
2) 000101100011100;
3) 110000101010011.

Восстановите битовые цепочки (без утроения), которые передавались.

Код с утроением битов обнаруживает одну или две ошибки. Кроме того, если сделана одна ошибка, он позволяет исправить её, т. е. является помехоустойчивым.

Помехоустойчивый код — это код, который позволяет исправлять ошибки, если их количество не превышает некоторого уровня.

Попробуйте придумать другой трёхбитовый код (содержащий два кодовых слова), который позволяет исправлять одну ошибку и обнаруживать одну или две ошибки. Что общего в коде с утроением битов и в вашем коде?

Кроме простого дублирования битов есть и другие, более сложные помехоустойчивые коды. Например, предположим, что передаются сообщения, содержащие только четыре буквы — «П», «О», «Р», «Т». Для кодирования букв используются 5-битные кодовые слова (рис. 2.9).

П	О	Р	Т
11111	11000	00100	00011

Рис. 2.9

Для этого набора кодовых слов выполнено такое свойство: любые два слова из набора различаются не менее чем в трёх битах. В этом случае говорят, что *расстояние Хэмминга* между кодовыми словами больше или равно трём. Например, слова «П» — 11111 и «О» — 11000 различаются в трёх последних битах, а слова «П» — 11111 и «Р» — 00100 — в четырёх битах (они подчёркнуты). Это позволяет обнаруживать и даже исправлять ошибки.

Для передачи данных использовался код, заданный на рис. 2.9. Принята цепочка 00110. Определите букву, код которой отличается от этой цепочки меньше всего.

Для передачи данных использовался код, заданный на рис. 2.9. Принята цепочка 10101. Определите знаки, коды которых отличаются от этой цепочки меньше всего.

Для передачи данных использовался код, заданный на рис. 2.9. При передаче каждого кодового слова произошло не более двух ошибок. Декодируйте сообщение, исправив ошибки:
00111 11100 11110 11000 00000 01110 11011 11100 00011 11000
Если ошибку исправить нельзя, поставьте символ «*».

В каком случае при использовании кода, заданного на рис. 2.9:
а) можно обнаружить ошибки, а исправить нельзя;
б) нельзя даже обнаружить ошибки?

Если все кодовые слова отличаются друг от друга не менее чем в трёх битах, такой код позволяет обнаружить одну или две ошибки. Если сделана только одна ошибка, код позволяет исправить её. Если же произошли три ошибки и более, в результате мог получиться другой правильный код, и эти ошибки обнаружить нельзя.

Выводы

- Данные при передаче могут искажаться из-за помех в канале связи.
- Для обнаружения и исправления ошибок код должен быть избыточным, т. е. содержать лишние биты по сравнению с кодом минимальной длины.
- Бит чётности — это бит, который добавляется в конец блока данных так, чтобы количество единиц в расширенном блоке было чётным.
- Использование бита чётности позволяет обнаружить нечётное число ошибок при передаче, но не позволяет их исправить.
- Помехоустойчивый код — это код, который позволяет исправлять ошибки, если их количество не превышает некоторого уровня.
- Код, в котором все кодовые слова отличаются друг от друга не менее чем в трёх битах, — помехоустойчивый. Он позволяет обнаружить одну или две ошибки и исправить одну ошибку в каждом блоке данных.

Интеллект-карта

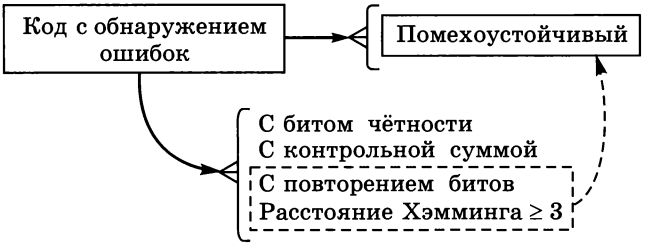


Рис. 2.10

Вопросы и задания

1. Почему код с обнаружением и/или исправлением ошибок должен быть избыточным?
2. Сравните коды, использующие бит чётности и коды, использующие контрольную сумму. Какие достоинства и недостатки имеет каждый метод?
3. Зачем нужны коды, которые позволяют обнаружить ошибки, но не позволяют их исправить?
4. Сравните код, в котором каждый бит повторяется три раза, и код на рис. 2.9. Какой из них более экономичный, т. е. требует меньше «лишних» битов?
5. Как вы думаете, почему бы не использовать везде только помехоустойчивые коды?
6. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Контрольные суммы»
б) «Коды Хэмминга»

§ 7
Системы счисления

Ключевые слова:

- система счисления
- непозиционная система
- позиционная система
- алфавит
- основание
- разряд

В этом и следующих параграфах вы узнаете, как различные виды информации представляются в двоичном коде, т. е. кодируются на том языке, который «понимает» процессор. Мы начнём с чисел, поскольку, в конечном счёте, все данные хранятся в памяти как числа. Нам предстоит разобраться, как вообще записывают числа и как представить их в двоичном коде.

Что такое система счисления?

Система счисления — это правила записи чисел с помощью специальных знаков — цифр, а также правила выполнения операций с этими числами.

Первоначально люди считали на пальцах — это самый простой способ, который используется и сейчас. Один загнутый (или отогнутый) палец обозначал единицу (один день, одного человека, одного барана и т. п.). Такая система счисления называется **унарной** (от латинского слова *unus* — один). В качестве цифры можно использовать камешки, узелки, счётные палочки (как в начальной школе), зарубки на дереве (как Робинзон Крузо) или на кости, чёрточки и точки на бумаге, другие одинаковые знаки или предметы.

С помощью унарной системы можно записывать только натуральные числа, причём запись больших чисел получается очень длинной (представьте себе, как записать число «миллион»).

Непозиционные системы счисления

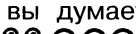
Непозиционная система счисления — это такая система, в которой значение цифры не зависит от её места (позиции) в записи числа.

Например, цифра в любой позиции числа, записанного в унарной системе, всегда обозначает единицу, поэтому унарная система — одна из *непозиционных* систем счисления.

Непозиционную систему счисления использовали в Египте. Египтяне придумали 7 знаков-иероглифов, которые обозначали степени числа 10 (чёрточка, хомут, верёвка, лотос, палец, лягушка, человек) — рис. 2.11.

1 10 10^2 10^3 10^4 10^5 10^6

Рис. 2.11

Как вы думаете, какое число записывается в египетской системе как  ?

Запишите в египетской системе число 647.

В **римской системе** (она также считается непозиционной) в качестве цифр используются латинские буквы: I обозначает 1, V — 5, X — 10, L — 50, C — 100, D — 500, M — 1000. Единицы, десятки, сотни и тысячи кодировались отдельными группами цифр, например:

$$\begin{aligned} 2368 &= 2000 + 300 + 60 + 8 = \\ &= (1000 + 1000) + (100 + 100 + 100) + \\ &+ (50 + 10) + (5 + 1 + 1 + 1) = \text{MMCCCLXVIII} \end{aligned}$$

Больше трёх одинаковых цифр подряд не ставили, поэтому число 4 записывали как IV. Здесь меньшая цифра (I) стоит перед большей (V); такая запись означает вычитание меньшего числа из большего, т. е.

IV = 5 — 1 = 4.

Запишите в римской системе счисления числа 9, 40, 90, 400 и 900.

Запишите в римской системе счисления числа: 2323, 1786, 1944 и 3499.

Обсудите в классе, можно ли назвать римскую систему счисления непозиционной. Приведите доводы в защиту разных точек зрения.

У римской системы есть несколько серьёзных недостатков:

- можно записывать только натуральные числа (что делать с дробными и отрицательными?);
- чтобы записывать большие числа, необходимо вводить всё новые и новые цифры (иногда использовались цифры с подчёркиванием или чертой сверху, что обозначало увеличение в 1000 раз: $\underline{V}$ — 5000, $\underline{X}$ — 10000 и т. д.);
- сложно выполнять арифметические действия.

Запишите в римской системе числа 8494 и 12849.

Сейчас римская система применяется для нумерации веков (XXI век), глав в книгах, на циферблатах часов (например, на Спасской башне Московского Кремля).

В **славянской системе счисления** в качестве цифр использовались буквы алфавита, над которыми ставился знак ҃ («титло») — рис.2.12.

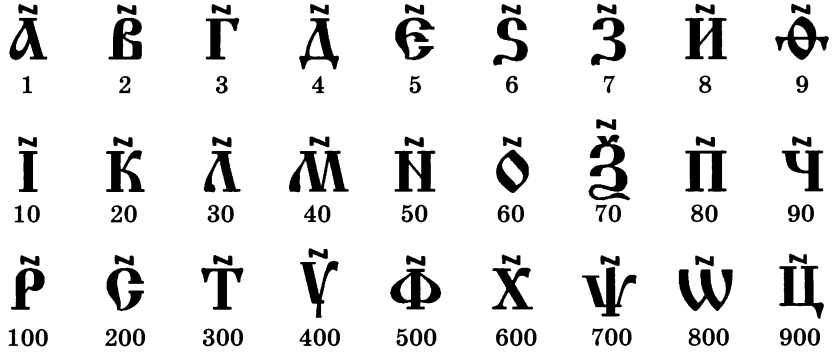


Рис. 2.12

Если в ряд стояло несколько цифр, знак «титло» ставился только у второй с конца (предпоследней). Старшие цифры записывались *справа* от младших, например число 11 записывалось как  .

Используя Интернет, выясните, где находятся часы, на циферблате которых нанесены числа в славянской системе счисления.

Запишите числа 27, 58, 137 и 879 в славянской системе счисления.

Позиционные системы счисления

Позиционная система счисления — это такая система, в которой значение цифры полностью определяется её местом (позицией) в записи числа.

Пример позиционной системы счисления — привычная нам десятичная система. В числе 6375 цифра 6 обозначает тысячи (6000), цифра 3 — сотни (300), цифра 7 — десятки (70), а цифра 5 — единицы:

$$6375 = 6 \cdot 1000 + 3 \cdot 100 + 7 \cdot 10 + 5 \cdot 1$$

Алфавит системы счисления — это используемый в ней набор цифр.
Основание системы счисления — это количество цифр в алфавите (мощность алфавита).

В десятичной системе основание — 10, *алфавит* состоит из 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9. Число 10, вероятно, было выбрано потому, что люди сначала использовали для счета свои 10 пальцев на руках.

Разряд — это позиция цифры в записи числа. Разряды в записи целых чисел нумеруются с нуля справа налево.

В числе 6375 цифра 6 стоит в третьем разряде (тысячи, 10^3), 3 — во втором разряде (сотни, 10^2), 7 — в первом (десятки, 10^1), а 5 — в нулевом (единицы, 10^0). Поэтому

разряды → 3 2 1 0

$$6375 = 6 \cdot 1000 + 3 \cdot 100 + 7 \cdot 10 + 5 \cdot 1$$

Это **развёрнутая форма** записи числа. Не забывайте, что любое число (кроме нуля!) в нулевой степени равно 1.

Запишите числа 158 и 1879 в развёрнутой форме.

Используя развёрнутую форму записи числа в десятичной системе счисления, определите, чему равен:
а) остаток от деления некоторого числа на 10;
б) остаток от деления некоторого числа на 100.
Как определить, что число без остатка делится на 10? На 100?

Все другие позиционные системы счисления, которые мы будем изучать, устроены так же, как и десятичная система, изменяется только основание. В первую очередь нас будет интересовать двоичная система (система с основанием 2). Она позволяет записать любое число в двоичном коде, который используется в компьютерах для хранения всех данных.

Мы познакомимся также с восьмеричной и шестнадцатеричной системами, которые применяют для сжатой записи двоичных кодов.

Как вы думаете, какие основания имеют восьмеричная и шестнадцатеричная системы счисления?

Какой алфавит может быть у двоичной системы счисления? У восьмеричной системы? У шестнадцатеричной системы?

Если число записано в позиционной системе с основанием, не равным 10, это основание записывают справа от числа как нижний индекс. Например, число 10110_2 записано в двоичной системе, 123_5 — в пятеричной, а 745_8 — в восьмеричной, а 296_{16} — в шестнадцатеричной.

Запишите числа 10110_2, 123_5, 745_8 и 296_{16} в развёрнутой форме и представьте их в десятичной системе счисления.

Найдите числа, которые записаны неправильно.
456_8 102_2 365_{12} 578_8 172_9 521_4
Как вы рассуждали?

Представьте число 23 в развёрнутой форме через степени числа 2. Как теперь можно записать это число в двоичной системе счисления?

Составьте таблицу степеней числа 2, от 2^1 до 2^{13}.

Для чисел 12, 75, 150 и 513 определите старшую степень числа в развёрнутой форме через степени числа 2. Как вы рассуждали?

Сформулируйте правило перевода числа из любой позиционной системы в десятичную.

Если основание системы счисления неизвестно, всё равно можно записать число в развёрнутой форме, обозначив основание как неизвестную величину x:

$$325_x = 3 \cdot x^2 + 2 \cdot x^1 + 5 \cdot x^0 = 3 \cdot x^2 + 2 \cdot x + 5.$$

В последнем равенстве учтено, что $x^1 = x$ и $x^0 = 1$.

Задача. В некоторой системе счисления число 58 записывается как 46_x. Определите основание x этой системы счисления.

Решение. Поскольку в записи числа 46_x есть цифра 6, можно сразу сказать, что $x > 6$ (в алфавитах систем счисления с меньшим основанием цифры 6 нет). Представим число 46_x в развёрнутой форме: $46_x = 4 \cdot x + 6$, и приравняем к 58:

$4 \cdot x + 6 = 58$.

Решив это уравнение, получаем: $x = 13$.

В некоторой системе счисления число 45 записывается как 63_x. Определите основание x этой системы счисления.

Найдите основание x системы счисления, в которой выполняется равенство $16_x + 33_x = 52_x$.

Найдите все основания x систем счисления, в которых верно неравенство $2_x + 32_x > 102_x$.

Как записывается наименьшее трёхзначное число в системе счисления с основанием x? Чему оно равно в десятичной системе?

Как записывается наибольшее трёхзначное число в системе счисления с основанием x? Чему оно равно в десятичной системе?

Найдите наименьшее основание системы счисления, в которой запись числа 30 имеет 3 значащих разряда.

Вася составил задачу: «В какой системе счисления число 15 записывается как 25_x?». Есть ли у неё решение? Обоснуйте ответ.

Выводы

- Система счисления — это правила записи чисел с помощью специальных знаков — цифр, а также правила выполнения операций с этими числами.
- В непозиционных системах счисления значение цифры не зависит от её места в записи числа. К непозиционным системам относятся унарная, римская, славянская, египетская и другие системы счисления.
- В позиционной системе счисления значение цифры полностью определяется её местом (позицией) в записи числа.
- Алфавит системы счисления — это используемый в ней набор цифр.
- Основание системы счисления — это количество цифр в алфавите (мощность алфавита).
- Разряд — это позиция цифры в записи числа. Разряды в записи целых чисел нумеруются с нуля справа налево.

Интеллект-карта

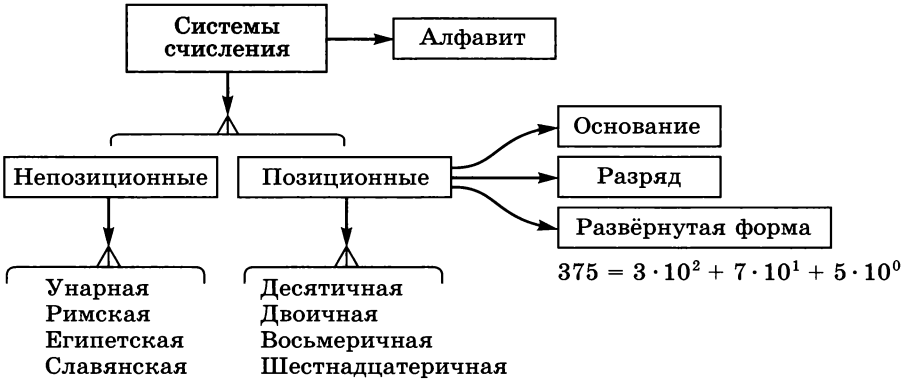


Рис. 2.13

Вопросы и задания

1. Как можно выполнить умножение в унарной системе счисления?
2. Как вы думаете, какие системы счисления появились первыми: позиционные или непозиционные?
3. Какое наибольшее число можно записать в римской системе счисления? Используя дополнительные источники, узнайте, как римляне записывали большие числа.
4. Почему римскую систему счисления не используют в компьютерах?
5. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Римская система счисления»
б) «Славянская система счисления»
в) «Египетская система счисления»
г) «Система счисления майя»
д) «Система остаточных классов»
е) «Фибоначчиева система счисления»

Проект

Найдите в Интернете фотографии часов Московского Кремля, часов Суздальского кремля, часов на здании МГУ им. М. В. Ломоносова в Москве, «Часов тысячелетия» в Варшаве, часов на башне Св. Стефана Вестминстерского дворца, часов Королевской обсерватории в Гринвиче. Какие системы счисления использованы на циферблатах этих часов?

§ 8
Двоичная система счисления

***Ключевые слова*:**

- двоичная система
- перевод чисел
- сложение
- вычитание
- перенос
- заём

В двоичной системе основание — 2, а алфавит состоит из двух цифр 0 и 1. Двоичная система счисления — самая важная для компьютеров, потому что все данные хранятся, обрабатываются и передаются именно в двоичном коде, как цепочки нулей и единиц. Двоичная система обладает важными *достоинствами*:

- для того чтобы построить компьютер, работающий с двоичными данными, достаточно иметь *устройства с двумя состояниями* (включено/выключено); оказалось, что сделать такие электронные устройства проще, чем какие-то другие, в том числе десятичные;
- действия над двоичными числами выполняются по очень простым правилам, поэтому все устройства компьютера тоже получаются достаточно простыми.

Перевод чисел

В предыдущем параграфе, решая задачи, вы научились переводить числа из любой позиционной системы в десятичную и обратно. Теперь применим эти знания для работы с двоичными числами.

Предположим, что нам дано двоичное число 1101_2. Нижний индекс «2» в этой записи обозначает основание системы счисления. Определим, какое это число (как оно записывается в десятичной системе). Запишем его в развёрнутой форме, так же как мы делали в предыдущем параграфе, но учтём, что основание здесь равно 2:

разряды → 3 2 1 0

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 1 = 13.$$

Так с помощью развёрнутой формы записи числа мы перевели его в десятичную систему.

Переведите в десятичную систему счисления числа 1011_2, 1001_2, 1111_2.

Василий перевёл некоторое число в двоичную систему счисления, эта запись содержит 5 цифр. Какое это могло быть число (назовите наименьшее и наибольшее возможные значения)?

Теперь осталось научиться решать обратную задачу: переводить числа из десятичной системы в двоичную. Это значит, что нужно получить 1101_2 из 13.

Заметим, что для этого нам достаточно записать число как сумму степеней основания числа 2, т. е. в развёрнутой форме. Полученные коэффициенты при степенях двойки будут цифрами двоичной записи числа.

Покажем, как работает этот способ для числа 13. Выделим из 13 старшую степень двойки — это $8 = 2^3$ (потому что $2^4 = 16$ будет уже больше 13):

$$13 = 8 + 5 = 2^3 + 5.$$

Дальше из «остатка» 5 снова выделяем старшую степень двойки — это $4 = 2^2$:

$$13 = 2^3 + 4 + 1 = 2^3 + 2^2 + 1.$$

Оставшаяся единица — это тоже степень двойки: $2^0 = 1$, поэтому получаем:

$$13 = 2^3 + 2^2 + 2^0.$$

Здесь не хватает ещё одного разряда, 2^1, можно добавить его к сумме, умножив на 0:

$$13 = \textcircled{1} \cdot 2^3 + \textcircled{1} \cdot 2^2 + \textcircled{0} \cdot 2^1 + \textcircled{1} \cdot 2^0 = 1101_2.$$

Таким образом, мы фактически получили развёрнутую запись числа 13 в двоичной системе счисления. Числа, обведённые кружками, — это цифры записи числа в двоичной системе счисления. Поэтому $13 = 1101_2$.

Переведите в двоичную систему счисления все натуральные числа от 1 до 16.

Переведите в двоичную систему счисления числа 19, 25, 31, 56.

Существует и другой, «более математический» способ. Запишем число в развёрнутой форме:

$$1101_2 = \boxed{1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1} + 1.$$

На какое число делится нацело сумма, обведённая рамкой? Какая часть полной суммы не делится на это число? Как она называется?

С помощью какой математической операции можно найти последнюю цифру двоичной записи числа?

Вынесем за скобку 2 (основание системы счисления):

$$1101_2 = (1 \cdot 2^2 + 1 \cdot 2^1 + 0) \cdot 2^1 + 1.$$

Какое число записано в скобках в развёрнутой форме? Как связано это число с исходным числом 1101_2?

Какие числа получатся, если эти числа разделить на 2 и отбросить остаток?

1110_2 1011_2 11011_2 10001_2

Запишите результаты в двоичной системе счисления.

Для перевода числа из десятичной системы в систему счисления с основанием 2 нужно делить это число на 2, отбрасывая остаток на каждом шаге, пока не получится 0. Затем выписать найденные остатки в обратном порядке.

Переведём в двоичную систему число 19 с помощью этого алгоритма. В результате многократного деления на 2 (пока на очередном шаге в частном не получится 0) находим $19 = 10011_2$ (рис. 2.14).

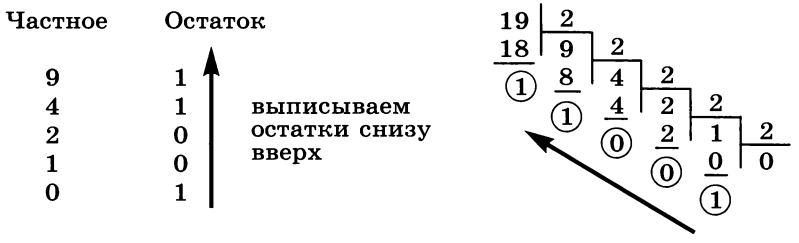


Рис. 2.14

Вспомним, что остатки — это цифры двоичной записи числа, начиная с последней. Поэтому их нужно выписать в обратном порядке (по стрелкам на рис. 2.14):

$$19 = 10011_2.$$

Как по двоичной записи числа определить, чётное оно или нет?

Переведите в двоичную систему счисления числа 16, 20, 24, 28. Как по двоичной записи числа сразу определить, делится ли оно на 4? На 8? На 16?

Арифметические действия

Компьютер выполняет все вычисления в двоичной системе. Разберём, как он это делает, на примере сложения и вычитания.

Вспомните, как выполняется сложение в столбик в десятичной системе счисления. С какого разряда начинается сложение? Когда происходит перенос в следующий разряд?

Как вы думаете, когда будет происходить перенос в следующий разряд при сложении в двоичной системе счисления?

Двоичные числа, как и десятичные, можно складывать в столбик, начиная с младшего разряда. При этом используют следующие правила (таблицу сложения):

$$0 + 0 = 0,\quad 1 + 0 = 1,\quad 1 + 1 = 2 = 10_2,\quad 1 + 1 + 1 = 3 = 11_2.$$

В двух последних случаях, когда сумма $2 = 10_2$ или $3 = 11_2$ не может быть записана с помощью одного двоичного разряда, происходит перенос в следующий разряд (как в десятичной системе, когда сумма получается больше, чем 9).

Например, сложим в столбик 10110_2 и 111011_2. Единицы сверху обозначают перенос из предыдущего разряда:

$$\begin{array}{r} {\scriptstyle 11111} \\ 10110_2 \\ +\ 111011_2 \\ \hline 1010001_2 \end{array}$$

Сложите числа, записанные в двоичной системе счисления:

$$1011101_2 + 110111_2$$

Проверьте решение, переведя слагаемые и результат в десятичную систему счисления.

Вычитание выполняется почти так же, как и в десятичной системе.

Вспомните, как выполняется в десятичной системе вычитание в столбик с заёмом из старшего разряда: 300000 – 1 = ?. Сформулируйте правило, по которому выполняется заём.

Вот основные правила вычитания в двоичной системе:

$$0 - 0 = 0,\quad 1 - 0 = 1,\quad 1 - 1 = 0,\quad 10_2 - 1 = 1.$$

В последнем случае приходится брать заём из предыдущего разряда.

Когда берётся заём в двоичной системе счисления, в «рабочий» разряд добавляется уже не 10, а $10_2 = 2$ (основание системы

счисления), а все «промежуточные» разряды (между «рабочим» и тем, откуда берется заём) заполняются единицами — старшей цифрой двоичной системы счисления. Например:

$$\begin{array}{r} \dot{0}\,1\,1\,2 \\ 11000_2 \\ -\quad 1_2 \\ \hline 10111_2 \end{array} \qquad \begin{array}{r} \dot{0}\,1\,1\,2\,\dot{0}\,2 \\ 1000101_2 \\ -\quad 11011_2 \\ \hline 101010_2 \end{array}$$

Выполните вычитание в двоичной системе счисления:

$$1011101_2 - 110111_2$$

Проверьте решение, переведя исходные числа и результат в десятичную систему счисления.

Вспомните, как вычитают из меньшего числа большее в десятичной системе счисления. Используя тот же принцип, выполните вычитание в двоичной системе счисления:

$$10111_2 - 110101_2$$

Проверьте решение, переведя исходные числа и результат в десятичную систему счисления.

Недостатки двоичной системы счисления

Вы уже знаете, что двоичная система удобна для компьютеров. Однако с точки зрения человека у неё есть существенные *недостатки*:

- двоичная запись больших чисел получается *длинная*: например, число 1024 записывается в виде 10000000000_2 — здесь легко перепутать количество идущих подряд нулей;
- запись *однородна*, т. е. содержит только нули и единицы; поэтому при работе с двоичными числами легко ошибиться или запутаться.

Для того чтобы облегчить человеку работу с данными в компьютерном формате, нужно придумать, как их записать в кратком виде. Для этого используют восьмеричную и шестнадцатеричную системы счисления, о которых пойдёт речь в следующих параграфах.

Выводы

- Данные всех видов в компьютерах хранятся, обрабатываются и передаются в виде чисел, представленных в двоичной системе счисления.
- Двоичная система счисления наиболее удобна для обработки данных в компьютерах.

- Чтобы перевести число в двоичную систему, нужно представить его как сумму степеней числа 2.
- Чтобы перевести число из двоичной системы в десятичную, нужно записать его в развёрнутой форме как сумму степеней числа 2 и выполнить сложение.
- Последняя цифра двоичной записи числа — это остаток от деления этого числа на 2. Двоичная запись чётного числа заканчивается на 0, а нечётного — на 1. Если число делится на 4, то его двоичная запись заканчивается на два нуля.
- При сложении двоичных чисел перенос в следующий разряд выполняется тогда, когда сумма в текущем разряде получается больше, чем 1.
- При вычитании двоичных чисел заём равен $2 = 10_2$; разряд, из которого берётся заём, уменьшается на 1; все промежуточные разряды вычитаемого (которые были равны нулю) изменяются на 1.
- Человеку неудобно работать с данными, записанными в двоичной системе счисления, потому что запись больших чисел получается длинной и однородной (содержит только 0 и 1).

Интеллект-карта

Рис. 2.15

Вопросы и задания

1. Сравните правила сложения и вычитания в десятичной и в двоичной системах счисления.
2. Сравните преимущества и недостатки использования двоичной системы счисления с точки зрения человека и с точки зрения компьютера.
3. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Запись десятичных дробей в двоичной системе счисления»
б) «Двоично-десятичная система счисления»
в) «Троичная уравновешенная система счисления»
г) «Троичные компьютеры»

Интересные сайты

trinary.ru — троичная логика и троичная цифровая техника

§ 9
Восьмеричная система счисления

Ключевые слова:

- восьмеричная система
- перевод чисел
- связь с двоичной системой
- сложение
- вычитание

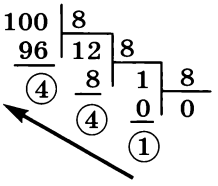
Восьмеричная система счисления (система с основанием 8) использует 8 цифр: от 0 до 7.

Как бы вы записали числа 8, 9, 18, 19, 89, 98, если бы цифр 8 и 9 не существовало?

Какие числа записаны в восьмеричной системе неверно: 345_8, 576_8, 961_8, 1823_8? Как вы рассуждали?

Перевод чисел

Для перевода десятичного числа в восьмеричную систему проще всего использовать стандартный алгоритм для позиционных систем (деление на 8, выписывание остатков в обратном порядке). Например:



$100 = 144_8$

Переведите в восьмеричную систему счисления числа 16, 25, 55, 75, 98, 127.

Запись некоторого числа в восьмеричной системе счисления заканчивается на 0. Что можно сказать о свойствах этого числа?

Запись некоторого числа в восьмеричной системе счисления заканчивается на 3. Что можно сказать о свойствах этого числа?

Платон перевёл некоторое число из десятичной системы в восьмеричную, полученная запись содержит 2 цифры. Какое это могло быть число (назовите наименьшее и наибольшее возможные значения)?

Для перевода числа из восьмеричной системы в десятичную значение каждой цифры умножают на 8 в степени, равной разряду этой цифры, и полученные произведения складывают:

разряды → 2 1 0

$$144_8 = 1 \cdot 8^2 + 4 \cdot 8^1 + 4 \cdot 8^0 = 64 + 4 \cdot 8 + 4 = 100.$$

Фактически число записывается в развёрнутой форме.

Переведите числа 16_8, 35_8, 57_8, 103_8, 177_8, 234_8 в десятичную систему счисления.

Связь с двоичной системой счисления

Более интересен перевод из восьмеричной системы в двоичную и обратно. Конечно, можно перевести число сначала в десятичную систему, а потом — в двоичную, но для больших чисел этот способ требует достаточно сложных вычислений и может приводить к ошибкам.

Переведите число 15_8 в десятичную систему счисления, а потом — из десятичной в двоичную.

Оказывается, можно сделать перевод из восьмеричной системы в двоичную напрямую, используя тесную связь между этими системами: их основания связаны равенством $2^3 = 8$. Покажем это на примере восьмеричного числа 753_8. Запишем его в развёрнутой форме:

$$753_8 = 7 \cdot 8^2 + 5 \cdot 8^1 + 3 \cdot 8^0 = 7 \cdot 2^6 + 5 \cdot 2^3 + 3 \cdot 2^0.$$

Теперь переведём отдельно каждую цифру в двоичную систему и также запишем в развёрнутой форме:

$$7 = 111_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0,$$
$$5 = 101_2 = 1 \cdot 2^2 + 1 \cdot 2^0,$$
$$3 = 11_2 = 1 \cdot 2^1 + 1 \cdot {}^0.$$

Подставим эти выражения в предыдущее равенство:

$$753_8 = (1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 2^6 + (1 \cdot 2^2 + 1 \cdot 2^0) \cdot 2^3 + (1 \cdot 2^1 + 1 \cdot 2^0) \cdot 2^0.$$

Раскрывая скобки, мы получим разложение исходного числа по степеням двойки, т. е. его развёрнутую запись в двоичной системе счисления (здесь для отсутствующих степеней числа 2 добавлены нулевые слагаемые):

$$753_8 = \underbrace{\textcircled{1} \cdot 2^8 + \textcircled{1} \cdot 2^7 + \textcircled{1} \cdot 2^6}_{7 \cdot 8^2} + \underbrace{\textcircled{1} \cdot 2^5 + \textcircled{0} \cdot 2^4 + \textcircled{1} \cdot 2^3}_{5 \cdot 8^1} + \underbrace{\textcircled{0} \cdot 2^2 + \textcircled{1} \cdot 2^1 + \textcircled{1}}_{3 \cdot 8^0}$$

Таким образом, $753_8 = 111\ 101\ 011_2$. Двоичная запись разбита на *триады* (группы из трех цифр), каждая триада — это двоичная запись *одной цифры* исходного восьмеричного числа. Фактически мы каждую восьмеричную цифру отдельно переводим в двоичную систему (табл. 2.2).

Таблица 2.2

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Алгоритм перевода восьмеричного числа в двоичную систему счисления

1. Перевести каждую цифру (отдельно) в двоичную систему. Записать результаты в виде триад, добавляя, если нужно, нули в начале триады.
2. Соединить триады в одно «длинное» двоичное число.

Лидирующие нули в самой первой триаде писать не нужно, потому что они никак не изменяют число. Но, если «потерять» нули в середине числа, получится неверный результат.

Переведите число 375_8 в двоичную систему счисления двумя способами — через десятичную систему счисления и напрямую. Какой способ проще?

Алгоритм перевода двоичного числа в восьмеричную систему счисления

1. Разбить двоичное число на триады, *начиная справа*. В начало самой первой триады добавить лидирующие нули, если это необходимо.
2. Перевести каждую триаду (отдельно) в восьмеричную систему счисления.
3. Соединить полученные цифры в одно «длинное» число.

Переведите число 1010011_2 в восьмеричную систему счисления двумя способами — через десятичную систему счисления и напрямую. Какой способ проще?

Так как переход между двоичной и восьмеричной записью чисел выполняется несложно, восьмеричную систему счисления можно использовать для сокращённой записи двоичных кодов.

Арифметические действия

Какая самая старшая цифра в восьмеричной системе счисления? Как вы думаете, в каком случае при сложении будет перенос в следующий разряд?

Разберитесь в примере:

$$\begin{array}{rl} {\scriptstyle 111} & 6 + 2 = 1 \cdot 8 + \textcircled{0} \\ + \ 356_8 & 5 + 6 + 1 = 1 \cdot 8 + \textcircled{4} \\ \underline{4662_8} & 3 + 6 + 1 = 1 \cdot 8 + \textcircled{2} \\ 5240_8 & 0 + 4 + 1 = \textcircled{5} \end{array}$$

и выполните по аналогии сложение чисел $1567_8 + 453_8$.

В примере на сложение запись $1 \cdot 8 + 2$ означает, что получилась сумма, бóльшая 7. Она не может быть записана с помощью одной восьмеричной цифры. Поэтому здесь единица идёт в перенос (в следующий разряд), а двойка остаётся в этом разряде.

Как вы думаете, как выполняется заём при вычитании в восьмеричной системе счисления?

Разберитесь в примере:

$$\begin{array}{rl} {\scriptstyle \cdot\,\cdot} & (6 + 8) - 7 = \textcircled{7} \\ - \ 456_8 & (5 - 1 + 8) - 7 = \textcircled{5} \\ \underline{277_8} & (4 - 1) - 2 = \textcircled{1} \\ 157_8 & \end{array}$$

и выполните по аналогии вычитание чисел $1432_8 - 567_8$.

В записи операций при выполнении вычитания «–1» означает, что из этого разряда раньше был заём (его значение уменьшилось на 1), а «+8» — заём из старшего разряда.

Применение восьмеричной системы

С помощью восьмеричной системы удобно кратко записывать содержимое областей памяти, содержащих количество битов, кратное трём. Например, 6-битные данные «упаковываются» в две восьмеричные цифры. Некоторые компьютеры 1960-х годов использовали 24-битные и 36-битные данные, они записывались соответственно с помощью 8 и 12 восьмеричных цифр. Восьмеричная система использовалась даже для компьютеров с 8-битной ячейкой памяти (PDP-11, ДВК), но позднее была почти полностью вытеснена шестнадцатеричной системой.

Выводы

- В восьмеричной системе (системе с основанием 8) используются цифры от 0 до 7.
- Восьмеричная система используется для сжатой записи двоичных кодов. Каждая тройка (триада) битов записывается как одна восьмеричная цифра.

- Для перевода числа из десятичной системы счисления в восьмеричную надо записать в обратном порядке цепочку остатков, полученных при делении числа на 8. Если число делится на 8, его восьмеричная запись заканчивается на 0.
- Для перевода числа из восьмеричной системы в десятичную нужно записать его в развёрнутой форме в восьмеричной системе и вычислить полученную сумму.
- Для перевода числа из восьмеричной системы в двоичную нужно записать каждую восьмеричную цифру как три двоичных (триаду) и соединить полученные триады.
- Для перевода числа из двоичной системы в восьмеричную нужно разбить число на триады, начиная **справа**, и записать каждую триаду отдельно как одну восьмеричную цифру.

Интеллект-карта

Рис 2.16

Вопросы и задания

1. Сравните восьмеричную и двоичную системы счисления с точки зрения человека и с точки зрения компьютера.
2. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

«Применение восьмеричной системы счисления»

§ 10
Шестнадцатеричная система счисления

Ключевые слова:

- шестнадцатеричная система
- перевод чисел
- связь с двоичной системой
- сложение
- вычитание

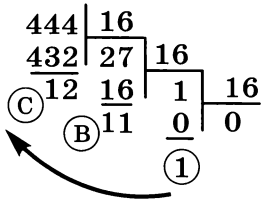
Шестнадцатеричная система (позиционная система с основанием 16) широко используется для записи адресов и содержимого ячеек памяти компьютера. Её алфавит содержит 16 цифр. Вместе с 10 арабскими цифрами от 0 до 9 используются первые буквы латинского алфавита:

A = 10, B = 11, C = 12, D = 13, E = 14 и F = 15.

Какие числа записаны в шестнадцатеричной системе неверно: $34AF5_{16}$, $5BG6_{16}$, $9FF61_{16}$, $ADH23_{16}$? Как вы рассуждали?

Перевод чисел

Для перевода чисел из десятичной системы в шестнадцатеричную используют алгоритм деления на 16 и выписывания остатков. Важно не забыть, что все остатки, бóльшие 9, заменяются на буквы:



$444 = 1BC_{16}$

Переведите в шестнадцатеричную систему счисления числа 31, 91, 126, 172.

Запись некоторого числа в шестнадцатеричной системе счисления заканчивается на 0. Что можно сказать о свойствах этого числа?

Запись некоторого числа в шестнадцатеричной системе счисления заканчивается на 3. Что можно сказать о свойствах этого числа?

Платон перевёл некоторое число из десятичной системы в шестнадцатеричную, полученная запись содержит 2 цифры. Какое это могло быть число (назовите наименьшее и наибольшее возможные значения)?

Для перевода числа из шестнадцатеричной системы в десятичную значение каждой цифры умножают на 16 в степени, равной её разряду, и полученные значения складывают:

разряды → 2 1 0

$$1BC_{16} = 1 \cdot 16^2 + 11 \cdot 16^1 + 12 \cdot 16^0 =$$
$$= 256 + 176 + 12 = 444.$$

Переведите числа 12_{16}, 57_{16}, 79_{16}, AB_{16}, DD_{16}, EF_{16} в десятичную систему счисления.

Не выполняя перевода чисел в другие системы счисления, выясните, может ли число 225 быть записано в шестнадцатеричной системе как $11F_{16}$. Как вы рассуждали?

Связь с двоичной системой счисления

Основания двоичной и шестнадцатеричной систем связаны соотношением $2^4 = 16$, поэтому можно переводить числа из шестнадцатеричной системы в двоичную напрямую, каждую цифру отдельно. Алгоритмы перевода чисел из шестнадцатеричной системы в двоичную и обратно полностью аналогичны соответствующим алгоритмам для восьмеричной системы.

Каждая шестнадцатеричная цифра представляется в виде *тетрады* (группы из четырёх двоичных цифр) — табл. 2.3.

Таблица 2.3

0	0000	8	1000
1	0001	9	1001
2	0010	A (10)	1010
3	0011	B (11)	1011
4	0100	C (12)	1100
5	0101	D (13)	1101
6	0110	E (14)	1110
7	0111	F (15)	1111

Алгоритм перевода шестнадцатеричного числа в двоичную систему счисления

1. Перевести каждую цифру отдельно в двоичную систему. Записать результаты в виде тетрад, добавляя, если нужно, нули в начале тетрады.
2. Соединить тетрады в одно «длинное» двоичное число.

Переведите число $EA123_8$ в двоичную систему счисления.

Алгоритм перевода двоичного числа в шестнадцатеричную систему счисления

1. Разбить двоичное число на тетрады, *начиная справа*. В начало самой первой тетрады добавить нули, если это необходимо.
2. Перевести каждую тетраду отдельно в шестнадцатеричную систему счисления.
3. Соединить полученные цифры в одно «длинное» число.

Переведите число 11111010011_2 в шестнадцатеричную систему.

Перевод из шестнадцатеричной системы в восьмеричную (и обратно) удобнее выполнять через двоичную систему. Можно, конечно, использовать и десятичную систему, но в этом случае объём вычислений будет значительно больше.

Переведите число $2FA_{16}$ в восьмеричную систему счисления двумя способами: через десятичную систему и через двоичную. Сравните объём работы для этих способов. Какой из них проще?

Переведите число 165_8 в шестнадцатеричную систему счисления двумя способами: через десятичную систему и через двоичную. Сравните объём работы для этих способов. Какой из них проще?

Арифметические действия

Какая самая старшая цифра в шестнадцатеричной системе счисления? В каком случае при сложении будет перенос в следующий разряд?

При выполнении арифметических операций в шестнадцатеричной системе удобно сначала переписать исходные числа, заменив все буквы на их численные значения.

Разберитесь в примере:

$$\begin{array}{r} A5B_{16} \\ +\ C7E_{16} \\ \hline 16D9_{16} \end{array} \qquad \begin{array}{r} {\scriptstyle 1\quad\quad 1} \\ 10\ \ 5\ \ 11 \\ +\ 12\ \ 7\ \ 14 \\ \hline 1\ \ 6\ \ 13\ \ 9 \end{array}$$

$$11 + 14 = 1 \cdot 16 + \textcircled{9}$$
$$5 + 7 + 1 = 13 = \textcircled{D}$$
$$10 + 12 = 1 \cdot 16 + \textcircled{6}$$
$$0 + 0 + 1 = \textcircled{1}$$

и выполните по аналогии сложение чисел $5AC_{16} + CD7_{16}$.

При вычитании заём из старшего разряда равен $10_{16} = 16$, а все «промежуточные» разряды заполняются цифрой F — старшей цифрой системы счисления.

Разберитесь в примере:

$$\begin{array}{r} C5B_{16} \\ -\ A7E_{16} \\ \hline 1DD_{16} \end{array} \qquad \begin{array}{r} {\scriptstyle \cdot\quad\ \cdot} \\ 12\ \ 5\ \ 11 \\ -\ 10\ \ 7\ \ 14 \\ \hline 1\ \ 13\ \ 13 \end{array}$$

$$(11 + 16) - 14 = 13 = \textcircled{D}$$
$$(5 - 1 + 16) - 7 = 13 = \textcircled{D}$$
$$(12 - 1) - 10 = \textcircled{1}$$

и выполните по аналогии вычитание чисел $A7C_{16} - 6DE_{16}$.

Если нужно работать с числами, записанными в разных системах счисления, их сначала приводят к какой-нибудь одной системе.

Вычислите $53_8 + 56_{16}$ и запишите результат в двоичной системе счисления.

Применение шестнадцатеричной системы

Шестнадцатеричная система оказалась очень удобной для записи значений ячеек памяти. Байт в современных компьютерах представляет собой 8 соседних бит, т. е. ровно две тетрады. Таким образом, значение байтовой ячейки можно записать как две шестнадцатеричные цифры:

0	1	0	1	1	1	1	0
		5				E	

Каждый полубайт (4 бита) «упаковывается» в одну шестнадцатеричную цифру. Благодаря этому замечательному свойству шестнадцатеричная система практически полностью вытеснила восьмеричную.

Запишите содержимое ячеек памяти в виде шестнадцатеричного числа:

1	1	0	0	1	0	1	0	1	1	0	1	0	0	1	1

Выводы

- В шестнадцатеричной системе (системе с основанием 16) используется 16 цифр: арабские цифры от 0 до 9 и латинские буквы от A до F.
- Шестнадцатеричная система используется для сжатой записи двоичных кодов. Каждая четвёрка (тетрада) битов записывается как одна шестнадцатеричная цифра.
- Для перевода числа из десятичной системы счисления в шестнадцатеричную надо записать в обратном порядке цепочку остатков, полученных при делении числа на 16. Если число делится на 16, его шестнадцатеричная запись заканчивается на 0.
- Для перевода числа из шестнадцатеричной системы в десятичную нужно записать его в развёрнутой форме в шестнадцатеричной системе и вычислить полученную сумму.
- Для перевода числа из шестнадцатеричной системы в двоичную нужно записать каждую шестнадцатеричную цифру как четыре двоичных (тетраду) и соединить полученные тетрады.
- Для перевода числа из двоичной системы в шестнадцатеричную нужно разбить число на тетрады, начиная справа, и записать каждую тетраду отдельно как одну шестнадцатеричную цифру.
- Перевод чисел из шестнадцатеричной системы в восьмеричную и обратно удобнее выполнять через двоичную систему.

Интеллект-карта

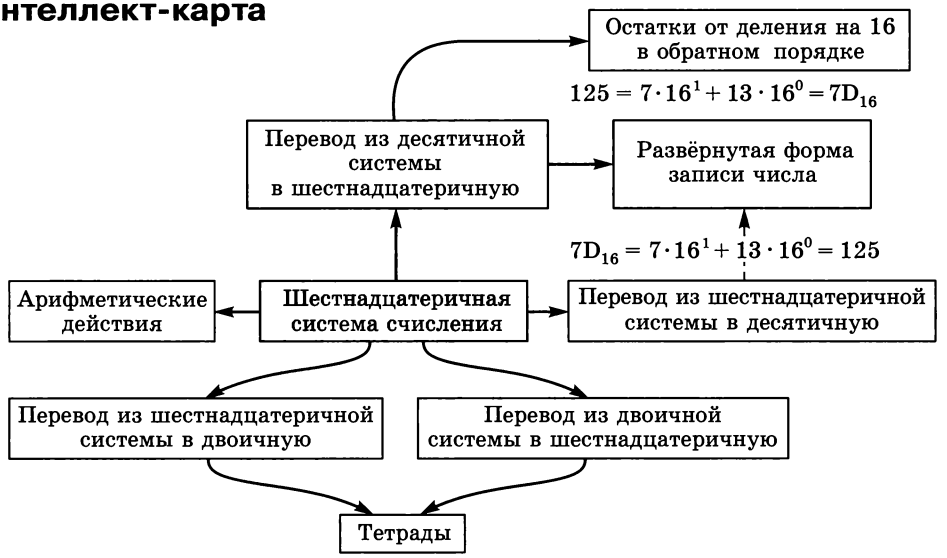


Рис. 2.17

Вопросы и задания

1. Почему в шестнадцатеричной системе счисления появилась необходимость использовать латинские буквы?
2. Как вы думаете, почему сейчас для записи данных в памяти и машинных команд чаще всего используют шестнадцатеричную систему, а не восьмеричную?
3. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

«Применение шестнадцатеричной системы счисления»

§ 11
Кодирование текстов

Ключевые слова:

- текст
- символ
- кодировка ASCII
- кодовая страница
- стандарт UNICODE

Разбейтесь на группы. Каждая группа готовит к уроку небольшое сообщение по одному из пунктов этого параграфа. Используя дополнительные источники, найдите ответы на вопросы в тексте каждого пункта.

С точки зрения компьютера, текст — это последовательность символов: букв (прописных или строчных, русских и латинских), цифр, знаков препинания, скобок, математических знаков, пробелов между словами.

Как кодируют символы?

Поскольку в современных компьютерах все виды информации представлены в двоичном коде, все символы нужно закодировать в виде цепочек нулей и единиц.

Часто для символов используют равномерное кодирование, при котором каждый символ кодируется одним и тем же количество битов.

Пусть кодовое слово для символа занимает 5 бит. Сколько различных символов можно закодировать с помощью такого кода?

Пусть требуется закодировать 35 различных символов. Сколько бит нужно выделить на каждый символ?

Обычно поступают следующим образом:

1) определяют, сколько символов нужно использовать (обозначим это число — *мощность алфавита* языка, на котором создан текст, через M);
2) определяют нужное количество двоичных разрядов i так, чтобы с их помощью можно было закодировать не менее M разных символов (т. е. $2^i \geq M$);
3) составляют **кодовую таблицу**, в которой каждому символу сопоставляют **код** — целое число в интервале от 0 до $2^i - 1$;
4) коды символов переводят в двоичную систему счисления.

Шрифты

В текстовых файлах (которые не содержат оформления, например, в файлах с расширением *txt*) хранятся не изображения символов, а их коды. Откуда же компьютер берет изображения символов, когда выводит текст на экран? Оказывается, при этом с диска загружается специальный шрифтовой файл. В нём хранятся изображения символов, соответствующие каждому из кодов. Именно эти изображения и выводятся на экран. Это значит, что при изменении шрифта текст, показанный на экране, может выглядеть совсем по-другому. Например, многие шрифты не содержат изображений русских букв. Поэтому, когда вы передаёте (или пересылаете) кому-то текстовый файл, нужно убедиться, что у адресата есть использованный вами шрифт.

В шрифтовом файле для каждого кода хранится изображение соответствующего символа. Поэтому очень важно, чтобы код, например буквы А, был одинаковым на всех компьютерах. Если два компьютера будут использовать разные кодовые таблицы, то, даже используя один и тот же шрифт, мы увидим два совершенно разных текста.

Современные текстовые процессоры умеют *внедрять* шрифты в файл. Хотя файл увеличивается в объёме, адресат увидит его в таком же виде, что и вы.

Используя дополнительные источники, найдите ответы на вопросы.

— Какие расширения имеют шрифтовые файлы?

— Как внедрить шрифт в документ в текстовом процессоре, который вы используете?

Кодировка ASCII

Теперь подумаем, как декодировать полученное двоичное сообщение, в котором закодирован текст. Для этого нам нужно знать, какой символ (буква, цифра и т. п.) соответствует каждому из принятых двоичных кодов, т. е. нужна кодовая таблица.

Для передачи текстов разработаны стандарты, которые закрепляют определённые коды за символами. Основной международный стандарт — 7-битная кодировка **ASCII**, в которую входят $2^7 = 128$ символов с кодами от 0 до 127:

- цифры от «0» до «9» с кодами от 48 до 57;
- латинские буквы: прописные, от «A» до «Z» (с кодами от 65 до 90) и строчные от «a» до «z» (с кодами от 97 до 122);
- знаки препинания: . , : ; ! ?
- скобки: [ ] { } ( )
- математические символы: + - * / = < >
- некоторые другие знаки: « ' # $ % & ^ | @ \ _ ~

Используя дополнительные источники, найдите ответы на вопросы.
— Как расшифровать сокращение ASCII?
— Какие символы имеют коды от 0 до 32?
— Как, зная код прописной латинской буквы, определить код соответствующей строчной буквы?

Однобайтные кодировки

В современных компьютерах память состоит из 8-битных ячеек — байтов. Поэтому к кодам ASCII можно добавить ещё один (старший) бит, таким образом, получается 8-битная кодировка. Этот дополнительный бит позволяет добавить в таблицу ещё 128 символов с кодами от 128 до 255. Такое расширение ASCII часто называют *кодовой страницей*. Первую половину кодовой страницы (коды от 0 до 127) занимает стандартная таблица ASCII, а вторую — символы национальных алфавитов (например, русские буквы):

Для русского языка существуют несколько кодовых страниц, которые были разработаны для разных операционных систем. Наиболее известны:

- **Windows-1251 (CP-1251)** — в системе *Windows*;
- **KOI8-R** — в *Unix*-совместимых операционных системах и электронной почте;

- альтернативная кодировка **CP-866**;
- **MacCyrillic** — на компьютерах компании *Apple*.

Проблема состоит в том, что, если набрать русский текст в одной кодировке (например, в Windows-1251), а просматривать в другой (например, в KOI8-R), текст будет очень сложно прочитать:

Windows-1251	KOI8-R
Здравствуй, мир!	гДПЮБЯРБСИ, ЛХП!
ъДТБЧУФЧХК, НЙТ!	Здравствуй, мир!

Для веб-страниц в Интернете часто используют кодировки Windows-1251 и KOI8-R. Браузер после загрузки страницы пытается автоматически определить её кодировку. Если ему это не удаётся, вы увидите странный набор букв вместо понятного русского текста. В этом случае нужно сменить кодировку вручную с помощью меню *Вид*.

Используя дополнительные источники, найдите ответы на вопросы.
— Что такое псевдографика и зачем она использовалась?
— В каких операционных системах для русских текстов используется кодировка CP-866?
— Найдите коды русских букв «А», «В», «П», «Р», «Я» в кодировках CP-866, CP-1251, KOI-8R, MacCyrillic;
— В каких кодировках русские буквы расположены по алфавиту, а в каких — нет?

Кодировки UNICODE

Любая 8-битная кодовая страница имеет серьезное ограничение — она может включать только 256 символов. Поэтому не получится набрать в одном документе часть текста на русском языке, а часть — на китайском.

Для решения этих проблем в 1991 году был принят новый стандарт **UNICODE**, который позволяет хранить в одной таблице коды символов любых существующих (и даже некоторых мёртвых) языков, математические и музыкальные символы и др.

Если мы хотим хранить в кодовой таблице больше разных символов, нужно увеличивать место, которое отводится под код каждого символа. Например, если на каждый символ выделить два байта, то можно закодировать $2^{16} = 65\,536$ символов. В современной версии UNICODE можно использовать до 1 112 064 различных символов. Символы из таблицы ASCII имеют в UNICODE те же самые коды, т. е. эти стандарты совместимы между собой.

В системе *Windows* используется кодировка UNICODE, называемая UTF-16 (от англ. *UNICODE Transformation Format* — формат преобразования UNICODE). В ней все наиболее важные символы кодируются с помощью 16 бит (2 байт), а редко используемые — с помощью 4 байт.

В *Unix*-подобных системах, например в *Linux*, чаще применяют кодировку UTF-8. В ней все символы, входящие в таблицу ASCII, кодируются с помощью 1 байта, а другие символы могут занимать от 2 до 6 байт. Текст, состоящий только из символов таблицы ASCII, кодируется точно так же, как и в кодировке ASCII, и его размер получается в два раза меньше, чем при использовании UTF-16. По данным поисковой системы *Google*, на конец 2014 года более 80% сайтов в Интернете использовали кодировку UTF-8.

Главное достоинство кодировок UNICODE в том, что они позволяют использовать символы разных языков в одном документе. За это приходится расплачиваться увеличением объёма файлов.

Используя дополнительные источники, найдите ответы на вопросы.
— Сколько символов сейчас добавлено в таблицы UNICODE?
— Сколько байт отводится на каждую русскую букву в кодировке UTF-8?
— Как будет выглядеть фраза «Здравствуй, мир!», если набрать её в кодировке UTF-8, а просматривать — в кодировке Windows-1251?

Информационный объём текста

Информационный объём текста — это количество информации, заключённое в этом тексте. Текст, как и все виды данных, хранится в памяти компьютера в двоичном коде. Поэтому за информационный объём текста принимают длину цепочки битов, в которой закодирован текст (вспомните материал 7 класса).

Часто тексты кодируются с помощью равномерного кода. Это значит, что на каждый символ выделяется одинаковое количество бит.

При равномерном кодировании информационный объём сообщения вычисляется по формуле

$$I = L \cdot i,$$

где L — длина сообщения (количество символов), а i — длина кодового слова каждого символа.

Определите информационный объём сообщения

ПУСТЬ ВСЕГДА БУДЕТ СОЛНЦЕ!

при использовании 16-битной кодировки.

Выпишите в тетрадь все различные символы, которые используются в сообщении

МАША ЛЮБИТ КАШУ.

Сколько бит нужно выделить на каждый символ, если никаких других символов в сообщении нет?

Определите информационный объём (в килобайтах) брошюры, в которой 10 страниц текста, на каждой странице 32 строки по 64 символа в каждой, используется 8-битная кодировка.

Во многих задачах на определение количества информации можно значительно упростить вычисления, если записывать все величины как степени числа 2.

Например, в последней задаче $32 = 2^5$, $64 = 2^6$. Поэтому количество символов на странице равно $32 \cdot 64 = 2^5 \cdot 2^6 = 2^{11}$, а общее количество символов в брошюре: $L = 10 \cdot 2^{11}$.

Каждый символ занимает 8 бит (1 байт), поэтому информационный объём текста равен $10 \cdot 2^{11}$ байт. Вспомним, что 1 Кбайт = 1024 байта = 2^{10} байт, поэтому для того, чтобы перевести количество информации из байтов в килобайты, нужно разделить число на 2^{10}. Получаем:

$$I = \frac{10 \cdot 2^{11}}{1024} = \frac{10 \cdot 2^{11}}{2^{10}} = 20 \text{ Кбайт}.$$

Текст романа А. С. Пушкина «Евгений Онегин» занимает 187 страниц, на каждой странице напечатано в среднем 885 символов. Оцените информационный объём этого текта (в килобайтах), если он записан в 16-битной кодировке.

Выводы

- Если кодовое слово для каждого символа занимает i бит, с помощью такого кода можно закодировать 2^i различных символов.
- Если нужно закодировать M различных символов, то количество битов i для каждого символа нужно выбирать из условия $2^i \geq M$.
- В текстовых файлах содержатся только коды символов, а их изображения хранятся в памяти компьютера. Современные операционные системы загружают изображения букв и других знаков из специальных шрифтовых файлов. Документы текстовых процессоров могут содержать внедрённые шрифты.
- ASCII — это 7-битная кодировка, которая является международным стандартом. Она содержит цифры, латинские буквы, скобки, знаки препинания, знаки арифметических операций и другие символы.

- Однобайтные (8-битные) кодировки (кодовые страницы) включают таблицу ASCII (символами с кодами 0..127) и дополнительную часть (символы с кодами 128...255), в которую входят символы национальных алфавитов.
- Стандарт UNICODE позволяет записывать в одной таблице коды символов любых существующих (и даже некоторых мёртвых) языков, математические и музыкальные символы и др. (всего до 1 112 064 знаков).
- В операционной системе *Linux* и в Интернете часто используется кодировка UTF-8. В ней все символы, входящие в таблицу ASCII, кодируются с помощью 1 байта, а другие символы могут занимать от 2 до 6 байт.
- Для кодирования веб-страниц на русском языке используют кодировки UTF-8, Windows-1251 и KOI8-R.
- При равномерном кодировании информационный объём сообщения вычисляется по формуле $I = L \cdot i$, где L — длина сообщения (количество символов), а i — длина кодового слова каждого символа.
- Во многих задачах на определение количества информации можно значительно упростить вычисления, если записывать все величины как степени числа 2.

Интеллект-карта

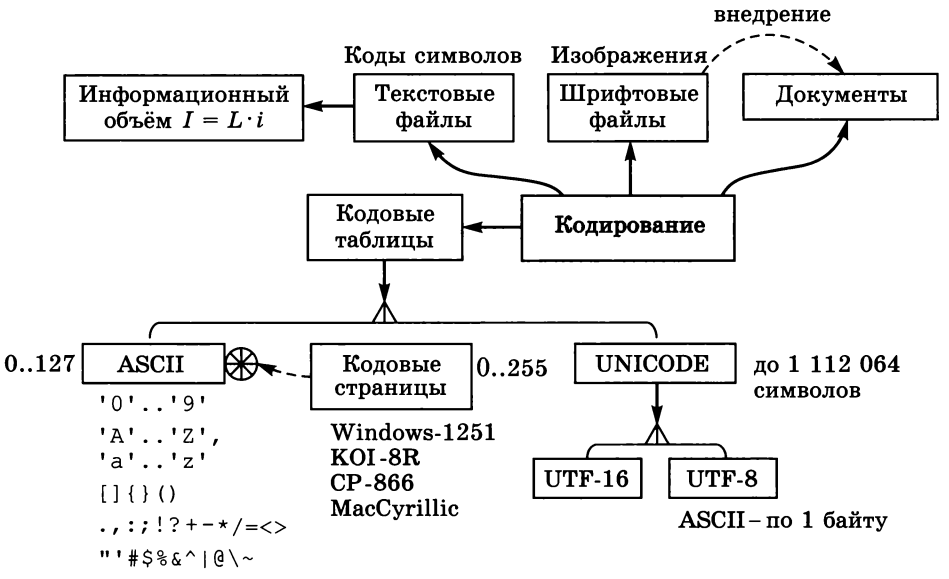


Рис. 2.18

Вопросы и задания

1. Как вы думаете, почему изображения символов хранятся в отдельных файлах?
2. Вы хотите использовать в тексте придуманный собственный символ, которого нет ни в одном шрифте. Какими путями это можно сделать?
3. Вы сами разработали шрифт и хотите переслать другу документ, в котором этот шрифт используется. Какими способами это можно сделать?
4. В чём достоинства и недостатки внедрения шрифтов в документ?
5. Почему в современных компьютерах используются кодировки, в которых каждый символ занимает целое число байтов?
6. Почему использование кодовых страниц для кодирования текста может привести к проблемам?
7. Что такое UNICODE? В чём достоинства и недостатки использования этого семейства кодировок?
8. Вооружившись таблицами различных однобайтных кодировок, *разбейтесь на пары* и напишите друг другу короткое сообщение, содержащее шестнадцатеричные коды символов. Используйте любую кодировку, но не сообщайте её напарнику! Постарайтесь раскодировать адресованное вам сообщение.
9. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Стандарт UNICODE»
б) «Кодировка UTF-16»
в) «Кодировка UTF-8»
г) «Внедрение шрифтов в документы»

Интересные сайты

unicode-table.com/ru/ — таблица символов UNICODE

§ 12
Кодирование рисунков: растровый метод

Ключевые слова:

- растр
- пиксель
- разрешение
- цветовая модель RGB
- цветовая модель CMYK
- цветовая модель HSB
- глубина цвета
- цветовая палитра

Рисунками мы будем называть все изображения, которых хранятся и обрабатываются компьютером, в том числе фотографии, карты, чертежи и др.

Разбейтесь на группы. Каждая группа готовит к уроку небольшое сообщение по одному из пунктов этого параграфа. Используя дополнительные источники, найдите ответы на вопросы в тексте каждого пункта.

Что такое растровое кодирование?

Рисунок состоит из линий и закрашенных областей. В идеале нам нужно закодировать все особенности этого изображения так, чтобы его можно было в точности восстановить из кода (например, распечатать на принтере).

И линия, и область состоят из бесконечного числа точек. Цвет каждой из этих точек нам нужно как-то закодировать. Так как точек бесконечно много, для этого нужно бесконечно много памяти, поэтому таким способом изображение закодировать не удастся. Однако «поточечную» идею всё-таки можно использовать.

Начнём с чёрно-белого рисунка. Представим себе, что на изображение ромба наложена сетка, которая разбивает его на квадратики. Такая сетка называется **растром**. Теперь каждый квадратик внутри ромба зальём чёрным цветом, а каждый квадратик вне ромба — белым. Для тех квадратиков, в которых часть оказалась закрашена чёрным цветом, а часть — белым, выберем цвет в зависимости от того, какая часть (чёрная или белая) больше (рис. 2.19).

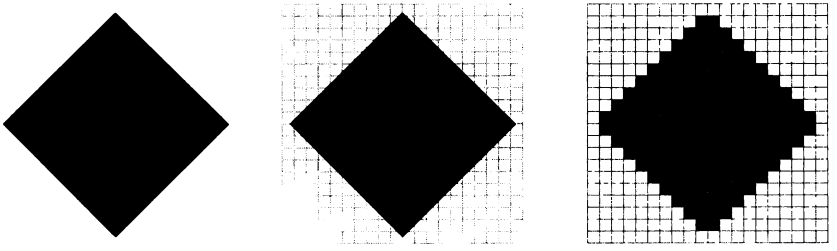


Рис. 2.19

У нас получился **растровый рисунок**, состоящий из квадратиков-пикселей.

Пиксель (англ. **pixel**: *picture element* — элемент рисунка) — это наименьший элемент рисунка, для которого можно задать свой цвет.

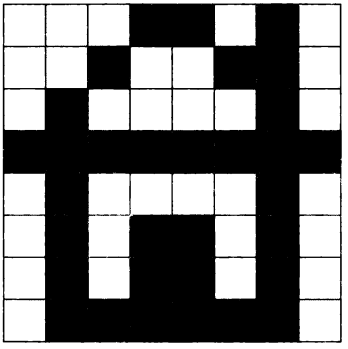
Разбив рисунок на квадратики, мы выполнили его **дискретизацию**. Действительно, у нас был непрерывный рисунок — изображение ромба. В результате мы получили **дискретный** объект — набор пикселей.

Двоичный код для чёрно-белого рисунка, полученного в результате дискретизации, можно построить следующим образом:

1) кодируем белые пиксели нулями, а чёрные — единицами[1];

2) выписываем строки полученной таблицы одну за другой.

Покажем это на простом примере (рис. 2.20).



0	0	0	1	1	0	1	0
0	0	1	0	0	1	1	0
0	1	0	0	0	0	1	0
1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	0
0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0

Рис. 2.20

Ширина этого рисунка — 8 пикселей, поэтому каждая строка таблицы состоит из 8 двоичных разрядов — битов. Чтобы не писать очень длинную цепочку нулей и единиц, удобно использовать шестнадцатеричную систему счисления, закодировав 4 соседних бита (тетраду) одной шестнадцатеричной цифрой. Например, для первой строки получаем код $1A_{16}$:

0	0	0	1	1	0	1	0
1				A			

а для всего рисунка: $1A2642FF425A5A7E_{16}$.

Используя полученный шестнадцатеричный код картинки, подсчитайте её информационный объём в битах и байтах.

Очень важно понять, что мы приобрели и что потеряли в результате дискретизации. Самое главное — мы смогли закодировать изображение в двоичном коде. Однако при этом рисунок исказился — вместо ромба мы получили набор квадратиков. При-

[1] Можно сделать и наоборот, чёрные пиксели обозначить нулями, а белые — единицами.

чина искажения в том, что в некоторых квадратиках части исходного рисунка были закрашены разными цветами, а в закодированном изображении каждый пиксель обязательно имеет один цвет. Таким образом, часть исходной информации при кодировании была потеряна. Это проявится, например, при увеличении рисунка — квадратики увеличиваются и рисунок ещё больше искажается. Чтобы уменьшить потери информации, нужно уменьшать размер пикселя, т. е. увеличивать разрешение.

Разрешение — это количество пикселей, приходящихся на единицу линейного размера изображения (чаще всего — на 1 дюйм).

Разрешение обычно измеряется в пикселях на дюйм (используется английское обозначение **ppi**: — *pixels per inch*). Например, разрешение 254 ppi означает, что на дюйм приходится 254 пикселя.

Чем больше разрешение, тем точнее кодируется рисунок (меньше информации теряется), однако одновременно растёт и объём файла.

Одна и та же картинка была отсканирована дважды: в первый раз с разрешением 300 ppi, а второй раз — с разрешением 600 ppi. Что можно сказать о размерах полученных файлов?

Существуют два основных способа получения растровых изображений:

1) ввод с помощью какого-либо устройства, например сканера, цифрового фотоаппарата или веб-камеры; напомним, что при сканировании происходит преобразование информации в компьютерные данные (оцифровка);
2) создание рисунка с помощью какой-либо программы.

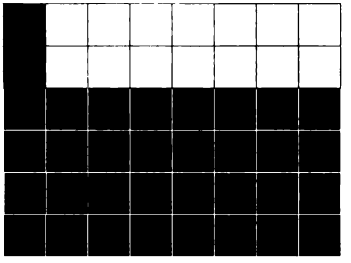
Используя дополнительные источники, найдите ответы на вопросы.
— Чему равен один дюйм в миллиметрах?
— Если отсканировать рисунок с разрешением 254 ppi, какой размер будет иметь изображение одного пикселя?
— Какие размеры в пикселях будет иметь изображение рисунка размером 10 × 15 см, если отсканировать его с разрешением 254 ppi?

Как кодируется цвет?

Что делать, если рисунок цветной? В этом случае для кодирования цвета пикселя уже не обойтись одним битом. Например, в изображении российского флага на древке (рис. 2.21, *а* и цвет-

ной рисунок на форзаце) четыре цвета: чёрный, синий, красный и белый. Для кодирования одного из четырёх вариантов нужно 2 бита, поэтому код каждого цвета (и код каждого пикселя) будет состоять из двух бит. Пусть 00 обозначает чёрный цвет, 01 — красный, 10 — синий и 11 — белый. Тогда получаем такую таблицу (рис. 2.21, *б* и цветной рисунок на форзаце).

а)



б)

00	11	11	11	11	11	11	11
00	11	11	11	11	11	11	11
00	10	10	10	10	10	10	10
00	10	10	10	10	10	10	10
00	01	01	01	01	01	01	01
00	01	01	01	01	01	01	01

Рис. 2.21

Определите информационный объём картинки на рис. 2.21 в битах и в байтах.

Проблема только в том, что при выводе на экран нужно как-то определить, какой цвет соответствует тому или иному коду. То есть информацию о цвете нужно выразить в виде числа (или набора чисел).

По современной теории цветного зрения, глаз человека содержит чувствительные элементы трёх типов. Элементы первого типа лучше всего чувствуют красный цвет, элементы второго типа — зелёный, а элементы третьего типа — синий. Цвет, который мы видим, получается как результат сложения сигналов от всех чувствительных элементов. Поэтому считается, что любой цвет (т. е. ощущения человека) можно имитировать, используя только три световых луча (красный, зелёный и синий) разной яркости. Эта модель цвета называется **моделью RGB** по начальным буквам английских слов *Red* (красный), *Green* (зелёный) и *Blue* (синий).

Можно представить себе, что мы в тёмной комнате светим тремя фонариками — красным, зелёным и синим. Там, куда попадает свет от красного и зелёного фонариков, будет область жёлтого цвета; при наложении красного и синего получается пурпурный цвет; синий и зелёный вместе дают голубой. Область, куда попадают лучи всех трёх цветов, будет белой (рис. 2.22 и цветной рисунок на форзаце).

Рис. 2.22

Яркость каждой составляющей (или, как говорят, каждого канала) обычно кодируется целым числом[1)] от 0 до 255. При этом код цвета — это тройка чисел (R,G,B), т. е. яркости отдельных каналов. Цвет (0,0,0) — это чёрный цвет, а (255,255,255) — белый (табл. 2.4). Если все составляющие имеют равную яркость, получаются оттенки серого цвета — от чёрного до белого.

Чтобы сделать светло-красный (розовый) цвет, нужно в красном цвете (255,0,0) одинаково увеличить яркость зелёного и синего каналов. Например, цвет (255,150,150) — это розовый. Равномерное уменьшение яркости всех каналов делает цвет темнее, например цвет с кодом (100,0,0) — тёмно-красный.

Таблица 2.4

Цвет	Код (R,G,B)
Красный	(255,0,0)
Зелёный	(0,255,0)
Синий	(0,0,255)
Белый	(255,255,255)
Чёрный	(0,0,0)
Серый	(128,128,128)
Пурпурный	(255,0,255)
Голубой	(0,255,255)
Жёлтый	(255,255,0)
Тёмно-пурпурный	(128,0,128)
Светло-жёлтый	(255,255,128)

1) Или десятичной дробью от 0 до 1.

Назовите цвета по их кодам в модели RGB:
а) (100, 100, 255);
б) (0, 80, 0);
в) (50, 50, 50);
г) (255, 100, 255).

Цвет на веб-страницах часто задаётся с помощью шестнадцатеричных кодов в модели RGB. Запись начинается знаком #, первые две шестнадцатеричные цифры обозначают яркость красного канала, следующие две — яркость зелёного канала и последние две — яркость синего канала. Таким образом, цвет кодируется в виде «длинного» числа, составленного из трёх чисел, каждое из которых занимает в памяти 1 байт.

Назовите наименьшее и наибольшее значения, которые можно закодировать с помощью двух шестнадцатеричных цифр.

Назовите цвета, заданные шестнадцатеричными кодами:
а) #FFFFFF;
б) #000000;
в) #A1A1A1;
г) #FF0000;
д) #FF6666;
е) #800000.

Запишите шестнадцатеричные коды следующих цветов:
а) зелёный;
б) синий;
в) жёлтый;
г) светло-зелёный;
д) светло-синий;
е) светло-жёлтый;
ж) тёмно-зелёный;
з) тёмно-синий;
и) тёмно-жёлтый.

Так как яркость каждого из трёх основных цветов может принимать 256 различных значений, всего можно закодировать $256^3 = 16\ 777\ 216$ оттенков, что достаточно для человека.

Сколько бит нужно, чтобы закодировать 256 различных вариантов? 32 варианта? 4 варианта?

Так как $256 = 2^8$, каждая из трёх составляющих занимает в памяти 8 бит (1 байт), а вся информация о каком-то цвете — 24 бита (3 байта). Эта величина называется глубиной цвета.

Глубина цвета — это количество бит, используемых для кодирования цвета пикселя.

24-битное кодирование цвета часто называют **режимом истинного цвета**. Для вычисления объёма рисунка в байтах при таком кодировании нужно определить общее количество пикселей K (умножить ширину на высоту) и умножить результат на 3, так как цвет каждого пикселя кодируется тремя байтами. Конечно, здесь не учитывается *сжатие* (уменьшение объёма файла с помощью специальных алгоритмов), которое применяется во

всех современных форматах графических файлов. Кроме того, в реальных файлах есть *заголовок*, в котором записана служебная информация (например, размеры рисунка).

Рисунок размером 20 × 30 пикселей закодирован в режиме истинного цвета. Определите информационный объём рисунка.

Как правило, чем меньше цветов используется, тем больше будет искажаться цветное изображение. Таким образом, при кодировании цвета тоже есть неизбежная потеря информации, которая добавляется к потерям, вызванным дискретизацией. Однако при увеличении количества используемых цветов одновременно растёт объём файла.

Используя дополнительные источники, найдите ответы на вопросы.

— Какие учёные считаются авторами современной теории цветового зрения?
— Какое выражение используется в английском языке для обозначений истинного цвета?
— Какая информация записана в заголовке файла формата BMP?

Цветовые модели

Мы только что познакомились с одним способом кодирования цвета — цветовой моделью RGB. Такая модель лучше всего описывает цвет, который **излучается** некоторым устройством, например экраном ноутбука или смартфона. Когда же мы смотрим на изображение, отпечатанное на бумаге, ситуация совершенно иная. Мы видим не прямые лучи источника, попадающие в глаз, а лучи, **отражённые** от поверхности. «Белый свет» от какого-то источника (солнца, лампочки) попадает на бумагу, на которую нанесена краска. Краска поглощает часть лучей (их энергия уходит на нагрев), а оставшиеся попадают в глаз, это и есть тот цвет, который мы видим (рис. 2.23 и цветной рисунок на форзаце).

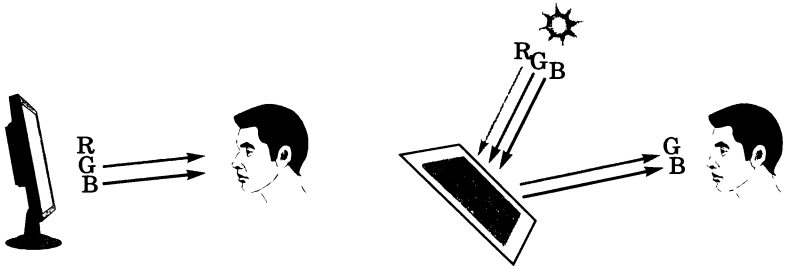


Рис. 2.23

Например, если краска поглощает красные лучи, остаются только синие и зелёные — мы видим голубой цвет. В этом смысле красный и голубой цвета дополняют друг друга, так же как и пары «зелёный — пурпурный» и «синий — жёлтый». Если из белого цвета (его RGB-код (255,255,255)) «вычесть» зелёный, то получится цвет RGB (255,0,255) (пурпурный), а если «вычесть» синий, то получится цвет с кодом RGB (255,255,0) — жёлтый.

На трёх дополнительных цветах — голубом, пурпурном и жёлтом — строится цветовая модель **CMY** (англ. ***Cyan*** — голубой, ***Magenta*** — пурпурный, ***Yellow*** — жёлтый), которая применяется для вывода рисунков на печать. Значения C = M = Y = 0 говорят о том, что на белую бумагу не наносится никакая краска, поэтому все лучи отражаются, это белый цвет. Если нанести на бумагу краску голубого цвета, красные лучи будут поглощаться, останутся только синие и зелёные. Если сверху нанести ещё жёлтую краску, которая поглощает синие лучи, останется только зелёный (рис. 2.24 и цветной рисунок на форзаце).

Рис. 2.24

Назовите цвета по их кодам в модели CMY:

а) (0,0,0);
б) (255,255,255);
в) (50,50,50);
г) (0,255,255);
д) (255,100,255);
е) (100,100,0).

Запишите коды следующих цветов в модели CMY:

а) зелёный;
б) синий;
в) жёлтый;
г) светло-зелёный;
д) светло-синий;
е) светло-жёлтый;
ж) тёмно-зелёный;
з) тёмно-синий;
и) тёмно-жёлтый.

При наложении голубой, пурпурной и жёлтой красок теоретически должен получиться чёрный цвет, все лучи поглощаются. Однако на практике всё не так просто. Краски не идеальны, поэтому вместо чёрного цвета получается грязно-коричневый. Кроме того, при печати чёрных областей приходится «выливать» тройную порцию краски в одно место. Нужно также учитывать, что на принтерах часто распечатывают чёрный текст, а цветные чернила значительно дороже чёрных.

Чтобы решить эту проблему, в набор красок добавляют чёрную, это так называемый *ключевой цвет* (англ. ***Key color***), поэтому получившуюся модель обозначают **CMYK**. Изображение, которое печатает большинство принтеров, состоит из точек этих четырёх цветов, которые расположены в виде узора очень близко друг к другу. Это создает иллюзию того, что в рисунке много цветов.

Кроме цветовых моделей RGB и CMY (CMYK) существуют и другие. Наиболее интересная из них — модель **HSB**[1] (от англ. ***Hue*** — тон, оттенок; ***Saturation*** — насыщенность, ***Brightness*** — яркость), которая ближе всего к естественному восприятию человека. Тон — это, например, синий, зелёный, жёлтый. Насыщенность — это чистота тона, при уменьшении насыщенности до нуля получается серый цвет. Яркость определяет, насколько цвет яркий или тёмный. Любой цвет при снижении яркости до нуля превращается в чёрный.

Используя дополнительные источники, найдите ответы на вопросы.
— Какая картинка используется для выбора цвета в модели HSB?
— Как перевести код цвета из модели RGB в модель CMY?
— Зачем используют цветовую модель Lab?

Кодирование с палитрой

Очень часто (например, в схемах, диаграммах и чертежах) количество цветов в изображении невелико (не более 256). Поэтому глубину цвета можно брать не 24 бита на пиксель, а меньше, таким образом, уменьшится размер файла. Для этого применяют **кодирование с палитрой.**

Если использовать кодирование с глубиной цвета 4 бита на пиксель вместо режима истинного цвета, как изменится объём файла?

Цветовая палитра — это таблица, в которой каждому цвету, заданному в виде составляющих в модели RGB, сопоставляется числовой код.

Кодирование с палитрой выполняется следующим образом:

- выбираем количество цветов N (не более 256);
- из палитры истинного цвета (16 777 216 цветов) выбираем любые N цветов, и для каждого из них определяем код в модели RGB;
- каждому выбранному цвету присваиваем номер (код) от 0 до $N - 1$;

[1] Или **HSV** (от английских слов ***Hue*** — тон, оттенок; ***Saturation*** — насыщенность, ***Value*** — величина).

- составляем палитру, записывая сначала RGB-составляющие цвета, имеющего код 0, затем — составляющие цвета с кодом 1 и т. д.;
- для каждого пикселя рисунка храним в памяти не яркости трёх цветовых каналов, а номер цвета в палитре.

Например, пусть при кодировании изображения российского флага (см. рис. 2.21) были выбраны 4 цвета:

- чёрный: RGB(0,0,0); двоичный код 00_2;
- красный: RGB(255,0,0); двоичный код 01_2;
- синий: RGB(0,0,255); двоичный код 10_2;
- белый: RGB(255,255,255); двоичный код 11_2.

Тогда палитра, которая обычно хранится в *заголовке файла*, представляет собой четыре трехбайтных блока:

0	0	0	255	0	0	0	0	255	255	255	255
цвет 0 = 00_2			цвет 1 = 01_2			цвет 2 = 10_2			цвет 3 = 11_2		

Код каждого пикселя занимает всего два бита.

Чтобы примерно оценить объём рисунка с палитрой, включающей N цветов, нужно:

- определить размер палитры, $3 \cdot N$ байт $= 24 \cdot N$ бит;
- определить глубину цвета (количество битов на пиксель), т. е. найти наименьшее натуральное число i, такое что $2^i \geq N$;
- вычислить общее количество пикселей K, перемножив размеры рисунка;
- определить информационный объём рисунка (без учёта палитры): $K \cdot i$ бит.

В таблице 2.5 приведены данные по некоторым вариантам кодирования с палитрой.

Таблица 2.5

Количество цветов	Размер палитры (байт)	Глубина цвета (бит на пиксель)
2	6	1
4	12	2
16	48	4
256	768	8

Палитры с количеством цветом более 256 на практике не используются.

Используя дополнительные источники, найдите ответы на вопросы.
— В файлах каких форматов можно сохранить изображение с палитрой?
— Определите размер палитры и глубину цвета при кодировании с палитрой 128 цветов.
— Как можно преобразовать рисунок, записанный в формате истинного цвета, в формат с палитрой?

Форматы файлов

Существует много разных форматов хранения растровых рисунков. В большинстве из них используют **сжатие**, т. е. уменьшают размер файла с помощью специальных алгоритмов. В некоторых форматах применяют **сжатие без потерь**, при котором исходный рисунок можно в точности восстановить из сжатого состояния. Ещё бóльшую степень сжатия можно обеспечить, используя **сжатие с потерями**, при котором незначительная часть данных (почти не влияющая на восприятие рисунка человеком) теряется.

Чаще всего встречаются следующие форматы графических файлов:

- **BMP** (англ. *bitmap* — битовая карта, файлы с расширением *bmp*) — стандартный формат растровых изображений в операционной системе Windows; поддерживает кодирование с палитрой и режим истинного цвета;
- **JPEG** (англ. *Joint Photographic Experts Group* — объединённая группа фотографов-экспертов, файлы с расширением *jpg* или *jpeg*) — формат, разработанный специально для кодирования фотографий; поддерживает только режим истинного цвета; для уменьшения объёма файла используется сильное сжатие, при котором изображение немного искажается (размывается), поэтому не рекомендуется использовать этот формат для хранения рисунков с чёткими границами;
- **GIF** (англ. *Graphics Interchange Format* — формат для обмена изображениями, файлы с расширением *gif*) — формат, поддерживающий только кодирование с палитрой (от 2 до 256 цветов); в отличие от предыдущих форматов части рисунка могут быть прозрачными, т. е. на веб-странице через них будет «просвечивать» фон; используется сжатие без потерь, т. е. при сжатии изображение не искажается;
- **PNG** (англ. *Portable Network Graphics* — переносимые сетевые изображения, файлы с расширением *png*) — формат, поддерживающий как режим истинного цвета, так и кодирование с палитрой; части изображения могут быть прозрачными и даже полупрозрачными (32-битное кодирование RGBA, где четвёртый байт задаёт прозрачность); изображение сжимается без искажения; анимация не поддерживается.

Свойства этих форматов сведены в таблицу 2.6.

Таблица 2.6

Формат	Истинный цвет	С палитрой	Прозрачность
BMP	да	да	—
JPEG	да	—	—
GIF	—	да	да
PNG	да	да	да

Давайте разберёмся, как компьютер различает форматы. Все виды данных хранятся в памяти компьютера в виде двоичных кодов, т. е. цепочек из нулей и единиц. Получив такую цепочку, абсолютно невозможно сказать, что это — текст, рисунок, звук или видео. Например, код 11001000_2 может обозначать число 200, букву «И», одну из составляющих цвета пикселя в режиме истинного цвета, номер цвета в палитре для рисунка с палитрой 256 цветов, цвета 8 пикселей чёрно-белого рисунка и т. п. Как же компьютер разбирается в двоичных данных? В первую очередь нужно ориентироваться на расширение имени файла. Например, чаще всего файлы с расширением *txt* содержат текст, а файлы с расширениями *bmp*, *gif*, *jpg*, *png* — рисунки.

Однако расширение файла можно менять как угодно. Например, можно сделать так, что текстовый файл будет иметь расширение *bmp*, а рисунок в формате JPEG — расширение *txt*. Поэтому в начало всех файлов специальных форматов (кроме простого текста — *txt*) записывается **заголовок**, по которому можно узнать тип файла и его характеристики. Например, файлы в формате BMP начинаются с символов «BM», а файлы в формате GIF — с символов «GIF». Кроме того, в заголовке указывается размер рисунка и, например, количество цветов в палитре, способ сжатия и т. п. Используя эту информацию, программа расшифровывает основную часть файла и выводит данные на экран.

Используя дополнительные источники, найдите ответы на вопросы.

— Сколько цветов может быть в палитре файлов форматов BMP, GIF, PNG?

— Какие форматы файлов могут хранить анимированные изображения?

— С каких символов начинаются файлы в формате JPEG?

Растровое кодирование: итоги

Итак, при растровом кодировании рисунок разбивается на пиксели (дискретизируется). Для каждого пикселя определяется единый цвет, который чаще всего кодируется с помощью RGB-кода. На практике эти операции выполняет *сканер* (устройство для ввода изображений) или цифровой фотоаппарат.

Растровое кодирование имеет *достоинства*:

- это универсальный метод (можно закодировать любое изображение);
- это единственный метод для кодирования и обработки размытых изображений, не имеющих четких границ, например фотографий;

и *недостатки*:

- при дискретизации всегда есть потеря информации;
- при изменении размеров изображения искажается цвет и форма объектов на рисунке, поскольку при увеличении размеров надо как-то восстановить недостающие пиксели, а при уменьшении — заменить несколько пикселей одним;
- размер файла не зависит от сложности изображения, а определяется только разрешением и глубиной цвета; как правило, растровые рисунки имеют большой объём.

Программа воспринимает растровый рисунок не как набор объектов, а как множество точек разного цвета. Человек же, глядя на рисунок, представляет нарисованные объекты у себя в сознании.

Растровая графика неприменима там, где нужно масштабировать рисунки, т. е. изменять их размеры без потери качества. Например, плакат должен хорошо выглядеть как на листе формата А4, так и на баннере, размеры которого измеряются в метрах.

Выводы

- При растровом кодировании изображение разбивается на пиксели — элементы, для каждого из которых можно задать свой цвет независимо от других элементов.
- Разрешение — это количество пикселей, приходящихся на единицу линейного размера изображения (чаще всего — на 1 дюйм).
- Глубина цвета — это количество битов, используемых для кодирования цвета пикселя.

- Качество растрового кодирования зависит от разрешения и глубины цвета.
- Цвет пикселя при выводе на экран монитора кодируется в модели RGB — раскладывается на красную, зелёную и синюю составляющие. При выводе на печать используется цветовая модель CMYK (голубой, пурпурный, жёлтый, чёрный).
- Информационный объём растрового рисунка вычисляется по формуле $I = K \cdot i$, где K — количество пикселей, а i — глубина цвета.
- При растровом кодировании, как правило, происходит потеря информации из-за дискретизации. Растровый рисунок искажается, когда его размеры изменяются.

Интеллект-карта

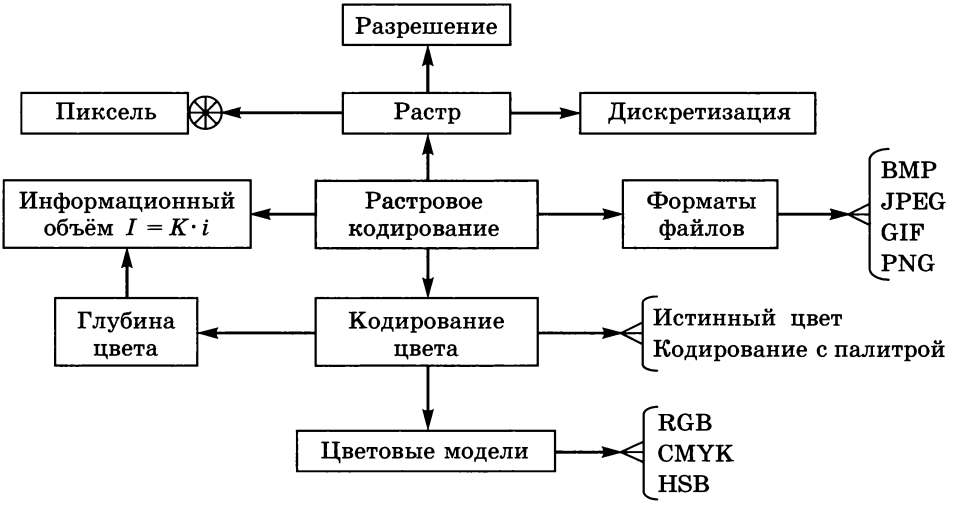


Рис. 2.25

Вопросы и задания

1. Как уменьшить потерю информации при дискретизации рисунков? Что при этом ухудшается?
2. Как можно уменьшить объём файла, в котором хранится рисунок? Чем при этом придется пожертвовать?
3. Как связаны глубина цвета и объём файла?
4. В каком случае при растровом кодировании нет потери информации?

5. Как компьютер определяет, что находится в файле — текст, рисунок, звук или видео?
6. Почему при увеличении растрового рисунка появляются «ступеньки»?
7. Сравните кодирование с палитрой и режим истинного цвета. Укажите достоинства, недостатки и области применения обоих подходов. В каких случаях вы рекомендуете использовать каждый из них?
8. Чем различаются основные современные форматы кодирования растровых рисунков?
9. В каких форматах целесообразно сохранять фотографии? рисунки с чёткими границами?
10. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Цветовая модель Lab»
б) «Цветовая модель HSB»
в) «Преобразования между цветовыми моделями»
г) «Формат BMP»
д) «Формат GIF»
е) «Формат JPEG»

Интересные сайты

colorscheme.ru/color-converter.html — кодирование цвета в разных цветовых моделях

§ 13
Кодирование рисунков: другие методы

***Ключевые слова*:**

- векторный рисунок
- 3D-графика
- фрактальная графика

Разбейтесь на группы. Каждая группа готовит к уроку небольшое сообщение по одному из пунктов этого параграфа. Используя дополнительные источники, найдите ответы на вопросы в тексте каждого пункта.

Векторное кодирование

Для того чтобы избавиться от недостатков растрового кодирования, была предложена такая идея: хранить в памяти компьютера не отдельные пиксели, а информацию о *геометрических фигурах*, из которых составлен рисунок:

- толщину, цвет и стиль контура;
- стиль заливки (один цвет, переход между несколькими цветами, узор);
- координаты фигуры, угол поворота, угол наклона.

Такая графика называется **векторной**.

Векторный рисунок хранится в памяти как математическое описание множества геометрических фигур с заданными свойствами контура и заливки внутренней области.

Векторный рисунок можно разобрать на части, растащив мышью его элементы, а потом снова собрать полное изображение (рис. 2.26).

Рис. 2.26

Для создания и редактирования векторных рисунков используют **векторные редакторы**. Самые известные профессиональные векторные редакторы — *Adobe Illustrator* и *Corel Draw*. Вместо них можно использовать бесплатную программу *Inkscape* (**inkscape.org**), которая относится к свободному программному обеспечению.

Векторные рисунки могут быть сохранены в различных форматах, в том числе:

- **WMF**, **EMF** (файлы с расширениями *wmf*, *emf*) — стандартные форматы векторных рисунков в Windows; в таких форматах хранятся рисунки в коллекции *Microsoft Office*;
- **ODG** (файлы с расширениями *odg*) — формат векторных рисунков пакета *OpenOffice*;
- **EPS** (файлы с расширением *eps*) — формат для хранения как растровых, так и векторных изображений и их комбинаций, используется при подготовке печатных изданий;
- **CDR** (файлы с расширением *cdr*) — формат векторных рисунков программы *CorelDRAW*;

- **AI** (файлы с расширением *ai*) — формат векторных рисунков программы *Adobe Illustrator*;
- **SVG** (англ. Scalable Vector Graphics — масштабируемые векторные изображения, файлы с расширением *svg*) — векторная графика для веб-страниц в Интернете.

Векторный способ кодирования рисунков обладает значительными преимуществами по сравнению с растровым, когда изображение (например, чертёж, схема, карта, диаграмма) может быть полностью разложено на простейшие геометрические фигуры. В этом случае при кодировании нет потери информации.

Объём файлов напрямую зависит от сложности рисунка — чем меньше элементов, тем меньше места занимает файл. Как правило, векторные рисунки значительно меньше по объёму, чем растровые.

При увеличении и уменьшении векторного рисунка не происходит никакого искажения формы элементов, не появляются «ступеньки», как при растровом кодировании (рис. 2.27).

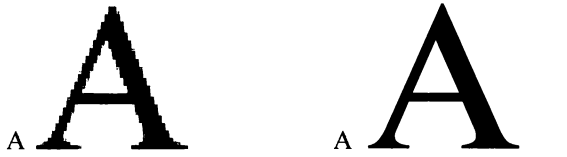


Рис. 2.27

Самый главный *недостаток* этого метода — он практически непригоден для кодирования изображений, в которых объекты не имеют чётких границ, например для фотографий.

www Используя дополнительные источники, найдите ответы на вопросы.
— В каких областях используется векторная графика?
— В каких форматах обычно сохраняют векторные рисунки, созданные в редакторах *Adobe Illustrator* и *CorelDraw*?
— Какой формат векторных рисунков можно использовать для веб-страниц?

Трёхмерная графика

Сейчас инженеры разрабатывают новые автомобили, самолёты, приборы в системах автоматизированного проектирования. В них строится трёхмерная (объёмная) модель объекта, которую затем нужно просматривать с разных сторон и рассчитывать на прочность. Объёмные модели объектов создаются с помощью **трёхмерной графики** (**3D-графики**).

Трёхмерные модели хранятся в памяти компьютера как математическое описание элементарных фигур (отрезков, треугольников, четырёхугольников и др.) и поверхностей. С этой точки зрения, трёхмерная графика — это векторная графика.

Каркас объекта обычно строится из многоугольников (*полигонов*), потом его углы сглаживаются. На поверхности наносят рисунок, имитирующий реальные материалы, настраивают их свойства: цвет, прозрачность, блики и др. Затем устанавливают источники света, задают свойства атмосферы, в которой находится объект.

Для того чтобы построить двумерную картинку (проекцию трёхмерной модели на плоскость), нужно выбрать точку наблюдения («установить камеру») и просчитать, как выглядит модель с этой точки. Быстрая смена таких картинок позволяет строить анимацию — создавать иллюзию движения и изменения.

Используя дополнительные источники, найдите ответы на вопросы.
— Как образовалось сокращение «3D»?
— Что такое рендеринг?
— Какие программы используют для работы с 3D-моделями?

Фрактальная графика

Существует ещё один интересный вид графики — фрактальная графика. **Фрактал** — это фигура, обладающая *самоподобием*. Это значит, что основная фигура состоит из нескольких таких же, только меньшего размера. Такие рисунки хранятся в памяти не как отдельные элементы (пиксели или описание геометрических фигур), а в виде математической формулы и алгоритма построения. Примеры фракталов показаны на рис. 2.28 (и см. цветной рисунок на форзаце).

а) *б*)

Рис. 2.28

Фрактальная графика применяется для построения изображений растений, облаков, гор, водных поверхностей, а также для оформления рекламных листовок и веб-сайтов.

www Используя дополнительные источники, найдите ответы на вопросы.
— От какого иностранного слова произошло слово «фрактал»?
— Какие фракталы изображены на рис. 2.28?
— Приведите ещё один пример фрактала.

Выводы

- Векторный рисунок хранится в памяти как математическое описание множества геометрических фигур с заданными свойствами контура и заливки внутренней области.
- При векторном кодировании чертежей, схем, карт нет потери информации.
- При изменении размеров векторного рисунка он не искажается.
- Объём файла с векторным рисунком определяется количеством элементарных фигур.
- Векторное кодирование неприменимо для хранения размытых изображений, например фотографий.
- Трёхмерные модели (3D-модели) хранятся в памяти компьютера как математическое описание элементарных фигур и поверхностей.
- Фрактал — это фигура, обладающая самоподобием, т. е. основная фигура состоит из нескольких таких же, только меньшего размера. Фракталы хранятся в памяти в виде математической формулы и алгоритма построения.

Интеллект-карта

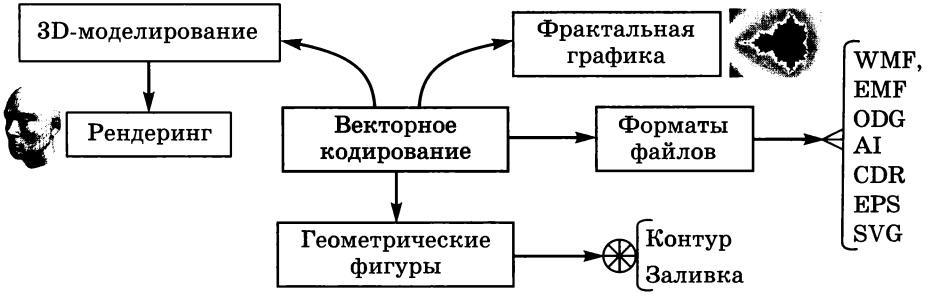


Рис. 2.29

Вопросы и задания

1. Почему не удаётся придумать единый метод кодирования рисунков, пригодный во всех ситуациях?
2. Выделите самое важное отличие векторной графики от растровой.
3. От чего зависит размер файла, в котором хранится векторный рисунок?
4. Почему векторные рисунки не искажаются при изменении размеров?
5. Как вы думаете, можно ли сохранить фотографию в векторном формате?
6. В каких задачах лучше использовать растровую графику, а в каких — векторную:
   а) создание коллажей;
   б) проектирование зданий;
   в) составление плана помещений;
   г) подготовка обложки журнала мод;
   д) подготовка чертежей и схем;
   е) спутниковая съёмка поверхности Земли;
   ж) разработка географических карт?
7. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Кривые Безье»
б) «Векторная графика вокруг нас»
в) «Векторные графические редакторы»
г) «Форматы векторных рисунков»
д) «Векторная графика на веб-страницах»
е) «Трёхмерная графика и её применения»
ж) «Фрактальная графика»

Интересные сайты

jasondavies.com/animated-bezier/ — анимация векторных кривых
inkscape.org — редактор *Inkscape*
inkscape.paint-net.ru — уроки по редактору *Inkscape*
wiki.linuxformat.ru/wiki/LXF74-75:Inkscape — уроки по редактору *Inkscape*
github.com/SVG-Edit/svgedit — онлайн-редактор векторной графики
corel.demiart.ru — уроки по редактору *CorelDraw*
coreldrawgromov.ru — уроки по редактору *CorelDraw*
dejurka.ru/inspiration/silvia_cordedda/ — рисунки, выполненные с помощью фрактальной графики

§ 14
Кодирование звука и видео

Ключевые слова:

- аналоговый сигнал
- оцифровка
- отсчёт
- частота дискретизации
- разрядность кодирования
- инструментальное кодирование

Разбейтесь на группы. Каждая группа готовит к уроку небольшое сообщение по одному из пунктов этого параграфа. Используя дополнительные источники, найдите ответы на вопросы в тексте каждого пункта.

Оцифровка звука

Звук — это колебания среды (воздуха, воды), которые воспринимает человеческое ухо. С помощью микрофона звук преобразуется в непрерывный **(аналоговый)** электрический сигнал, который в любой момент времени может принимать любое значение в некотором интервале (рис. 2.30).

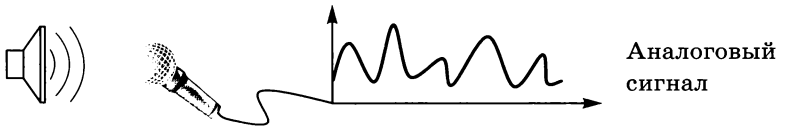


Рис. 2.30

Как вы знаете, современные компьютеры обрабатывают только дискретные сигналы (двоичные коды). Поэтому для работы со звуком необходима **звуковая карта** — специальное устройство, которое полученный с микрофона аналоговый сигнал превращает в двоичный код. Это называется оцифровкой звукового сигнала.

Оцифровка — это преобразование аналогового сигнала в цифровой код.

Ситуация напоминает ту, с которой мы столкнулись при кодировании рисунка — любая линия состоит из бесконечного числа точек, поэтому, чтобы её закодировать, нужна бесконечная память. Здесь тоже придётся использовать *дискретизацию* — представить аналоговый сигнал в виде набора чисел, т. е. записать в память только значения сигнала в отдельных точках, взятых с некоторым шагом T по времени (рис. 2.31).

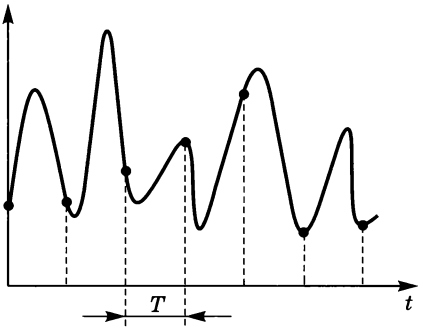


Рис. 2.31

Число T называется *интервалом дискретизации*, а обратная ему величина $1/T$ — **частотой дискретизации**. Частота дискретизации обозначается буквой f и измеряется в герцах (Гц) и килогерцах (кГц). Это число показывает, сколько раз в секунду выполняются измерения.

Чем больше частота дискретизации, тем точнее мы записываем сигнал, тем меньше информации теряем и тем лучше будет качество звучания. Однако при этом возрастает количество данных и увеличивается объём файла, в котором хранится закодированный звук.

Как же выбрать оптимальную частоту при кодировании? Учёные установили, что частоту дискретизации нужно брать в два раза больше, чем максимальная частота колебаний сигнала, который мы хотим записывать.

Известно, что человек в среднем слышит только звуки (колебания воздуха или другой среды) с частотами от 16 Гц до 20 кГц, поэтому все частоты выше 20 кГц можно «потерять» практически без ухудшения качества звука (человек не почувствует разницу!). Поэтому достаточно использовать частоту дискретизации около 40 кГц, повышать её дальше нет смысла. Более низкие частоты дискретизации применяют тогда, когда важно всячески уменьшать объём звуковых данных (например, для трансляции радиопередач через Интернет), даже ценой ухудшения качества.

С помощью оцифровки можно закодировать любой звук, который принимает микрофон. Это единственный способ кодирования человеческого голоса и различных природных звуков (шума прибоя, шелеста листвы и т. п.).

Однако у этого метода есть и *недостатки*:

- при оцифровке звука всегда есть потеря информации (из-за дискретизации);
- звуковые файлы, полученные с помощью оцифровки, имеют, как правило, большой размер.

Используя дополнительные источники, выясните, какая частота дискретизации используется:
а) для того чтобы можно было распознать речь человека;
б) на звуковых компакт-дисках;
в) в фильмах формата DVD;
г) для высококачественного кодирования звука в формате DVD-audio.

Глубина (разрядность) кодирования

Кроме того, что при кодировании звука выполняется дискретизация с потерей информации, нужно учитывать, что для хранения одного результата измерения в памяти отводится ограниченное место. При этом вносятся дополнительные ошибки.

Представим себе, например, что на одно измерение выделяется 3 бита. При этом код каждого сохранённого значения — это целое число от 0 до 7. Весь диапазон возможных значений сигнала делится на 8 полос, каждой из которых присваивается свой код. Все значения, попавшие в одну полосу, имеют одинаковый код (рис. 2.32).

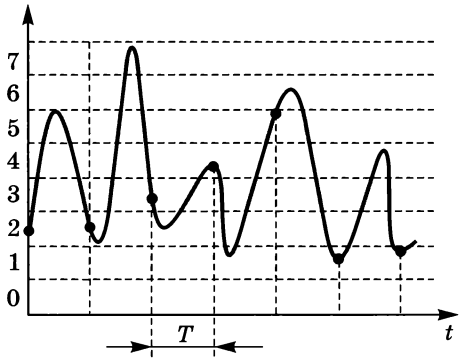


Рис. 2.32

Преобразование измеренного значения сигнала в целое число называется *дискретизацией по уровню*, или **квантованием**. Эту операцию выполняет АЦП — специальный блок звуковой карты.

Глубина кодирования (или разрядность кодирования) звука — это число бит, используемое для хранения одного отсчёта.

Используя дополнительные источники, найдите ответы на вопросы.
— Как расшифровывается сокращение АЦП?
— Сколько различных значений можно закодировать, если разрядность кодирования равна 8 бит; 16 бит; 24 бита?
— Какую разрядность имеют современные звуковые карты?

Вывод звука

Устройства вывода звука — наушники и звуковые колонки — это аналоговые (не цифровые) устройства, поэтому при проигрывании звука звуковая карта должна как-то восстановить аналоговый сигнал. Вспомним, что у нас есть только значения, измеренные с интервалом T. В простейшем случае по ним можно восстановить ступенчатый сигнал, который будет существенно отличаться от исходного (до кодирования). В современных звуковых картах для повышения качества звука этот ступенчатый сигнал сглаживается с помощью специальных фильтров, однако восстановить точно исходный сигнал всё равно не удаётся, так как информация о значениях сигнала между моментами квантования была потеряна при оцифровке (рис. 2.33).

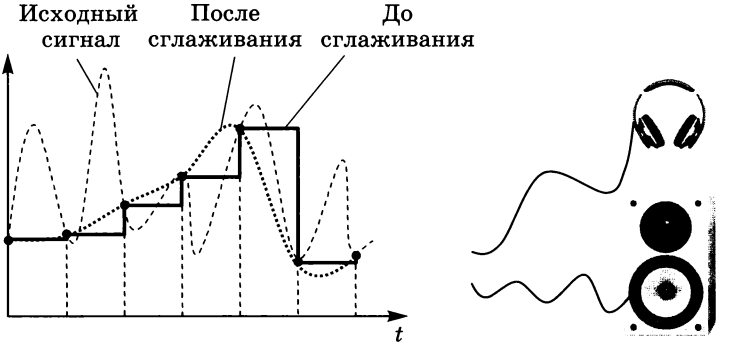


Рис. 2.33

Для повышения качества звука, т. е. для большего соответствия между сигналом, принятым микрофоном, и сигналом, выведенным из компьютера на колонки, нужно увеличивать частоту дискретизации, однако при этом, как вы уже знаете, увеличивается и объём файла.

Используя дополнительные источники, найдите ответы на вопросы.
— Что такое ЦАП?
— Зачем нужен ЦАП в звуковой карте?
— Что такое звуковой фильтр?

www

Объём звуковых данных

Объём данных, полученный после оцифровки звука, зависит от разрядности кодирования и частоты дискретизации. Например, пусть частота оцифровки равна 44 кГц (за 1 с выполняется 44 000 измерений сигнала), и используется 16-разрядное кодирование

(каждое из измеренных значений занимает 16 бит = 2 байта). Тогда за 1 секунду накапливается $44\,000 \cdot 2 = 88\,000$ байт информации, а за 1 минуту:

$$88\,000 \cdot 60 = 5\,280\,000 \text{ байт} \approx 5 \text{ Мбайт}.$$

Если записывается стереозвук (левый и правый каналы), это число нужно удвоить.

www Используя дополнительные источники, найдите ответы на вопросы.
— Как называется одноканальный звук?
— Что такое квадрофонический звук (квадрозвук)?
— Что такое битрейт?

Форматы файлов

Среди форматов оцифрованных звуковых файлов наиболее известны:

- **WAV** (англ. *Waveform Audio File Format*, файлы с расширением *wav*) — стандартный формат звуковых файлов в операционной системе *Windows*; сжатие данных возможно, но используется редко;
- **MP3** (файлы с расширением *mp3*) — самый популярный формат звуковых файлов, использующий сжатие с потерями: для значительного уменьшения объёма файла снижается качество кодирования для тех частот, которые практически неразличимы для человеческого слуха;
- **Ogg Vorbis** (файлы с расширением *ogg*) — свободный (не требующий коммерческих лицензий) формат сжатия звука с потерями.

Все эти форматы являются **потоковыми**, т. е. можно начинать прослушивание до того момента, как весь файл будет получен (например, из Интернета).

www Используя дополнительные источники, найдите ответы на вопросы.
— Как называется программа, которая выполняет сжатие и распаковку звуковых данных?
— Какая компания разработала формат WMA? Используется ли в нём сжатие?
— Где используется формат файлов AAC?

Инструментальное кодирование звука

Существует ещё один способ кодирования звука, который можно применить только для записи инструментальных мелодий. Он основан на стандарте **MIDI** (англ. *Musical Instrument Digital Interface* — цифровой интерфейс музыкальных инструментов).

В отличие от оцифрованного звука в таком формате хранятся последовательность нот, коды инструментов, громкость, тембр, время затухания каждой ноты и т. д.

Фактически такая запись звука — это программа для звуковой карты, в памяти которой хранятся образцы звуков реальных инструментов (**волновые таблицы**, англ. *wave tables*).

Современные звуковые карты поддерживают многоканальный звук, т. е. в звуковом файле может храниться несколько дорожек, которые проигрываются одновременно.

Существуют специальные MIDI-клавиатуры, которые позволяют вводить звук и сразу сохранять его в стандарте MIDI (рис. 2.34).

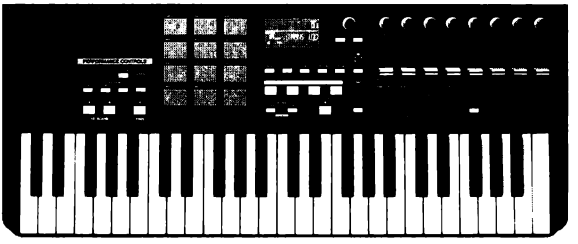


Рис. 2.34

Для проигрывания MIDI-файла используют **синтезаторы** — электронные устройства, имитирующие звук реальных инструментов. Простейший синтезатор — звуковая карта компьютера.

Главные *достоинства* инструментального кодирования:

- кодирование мелодии (нотной записи) происходит без потери информации;
- файлы имеют значительно меньший объём в сравнении с оцифрованным звуком той же длительности.

Однако произвольный звук (например, человеческий голос) в таком формате закодировать невозможно.

Используя дополнительные источники, найдите ответы на вопросы.

— Сколько различных музыкальных инструментов включено в стандарт MIDI?

— Что такое полифония?

— Какое расширение имеют файлы, содержащие звук, закодированный в стандарте MIDI?

Кодирование видеоинформации

Для того чтобы сохранить видео в памяти компьютера, необходимо закодировать звук и изменяющееся изображение. При этом нужно обеспечить их *синхронность* (одновременность), т. е. звук не должен ни «отставать» от изображения, ни опережать его.

Для кодирования звука чаще всего используют оцифровку с частотой 48 кГц.

Изображение состоит из отдельных растровых рисунков, которые меняются с частотой не менее 25 кадров в секунду, так что глаз человека воспринимает смену кадров как непрерывное движение. Это значит, что для каждой секунды видео нужно хранить в памяти 25 изображений.

Подсчитаем, сколько памяти потребуется для хранения видео. Если размер картинки 768 × 576 точек (стандарты PAL/SECAM) и глубина цвета 24 бита на пиксель, то закодированная 1 секунда видео (без звука) будет занимать примерно 32 Мбайт, а 1 минута — около 1,85 Гбайт. Это недопустимо много, поэтому в большинстве форматов видеоизображений используется сжатие. Упаковку и распаковку видеоданных выполняют специальные программы — **кодеки**.

Основная идея сжатия видео заключается в том, что за короткое время изображение изменяется очень мало, поэтому можно запомнить «базовый» кадр, а затем сохранять только изменения. Через 10–15 секунд изображение изменяется настолько, что необходим новый базовый кадр. Для того чтобы ещё больше уменьшить объём файла, применяют сжатие с потерями, при котором теряются некоторые детали, несущественные для восприятия человеком.

Наиболее известны следующие видеоформаты:

- **AVI** (англ. *Audio Video Interleave* — чередующиеся звук и видео, файлы с расширением *avi*) — формат, разработанный компанией *Microsoft* для системы *Windows*; может использовать разные алгоритмы сжатия;
- **WMV** (англ. *Windows Media Video*, файлы с расширением *wmv*) — формат, разработанный компанией *Microsoft*; может использовать разные алгоритмы сжатия;
- **MPEG** (файлы с расширениями *mpg*, *mpeg*) — формат, использующий один из лучших алгоритмов сжатия, который разработала экспертная группа по вопросам движущегося изображения (англ. **MPEG** — *Motion Picture Experts Group*);
- **MP4** (файлы с расширением *mp4*) — формат, позволяющий хранить несколько потоков видео, а также *субтитры* (текстовые надписи на экране);
- **WebM** — открытый (не требующий оплаты лицензии) видеоформат, который поддерживается в современных браузерах без установки дополнительных модулей.

www

Используя дополнительные источники, найдите ответы на вопросы.

— Что такое HD?

— Что такое артефакты сжатия видео? как они проявляются?

— Где используется видеоформат MOV?

Выводы

- Для кодирования звука применяют оцифровку и инструментальное кодирование.
- Оцифровка — это дискретизация сигнала, полученного с микрофона: в памяти сохраняются значения этого сигнала, измеряемые с некоторой частотой, которая называется частотой дискретизации.
- Глубина кодирования (разрядность кодирования) — это количество битов, выделяемых для хранения одного измерения.
- Качество оцифровки звука определяется частотой дискретизации и глубиной кодирования.
- Инструментальное кодирование звука позволяет без искажений кодировать нотную запись, но его нельзя использовать для кодирования нестандартных звуков, например речи человека.
- Кодирование видеофильмов сводится к кодированию изменяющегося изображения и звука. Изображение и звук должны проигрываться одновременно (синхронно).
- Для уменьшения объёма файлов при кодировании звука и видео, как правило, используют сжатие. Программа, которая выполняет упаковку и распаковку звуковых и видеоданных, называется кодеком.

Интеллект-карты

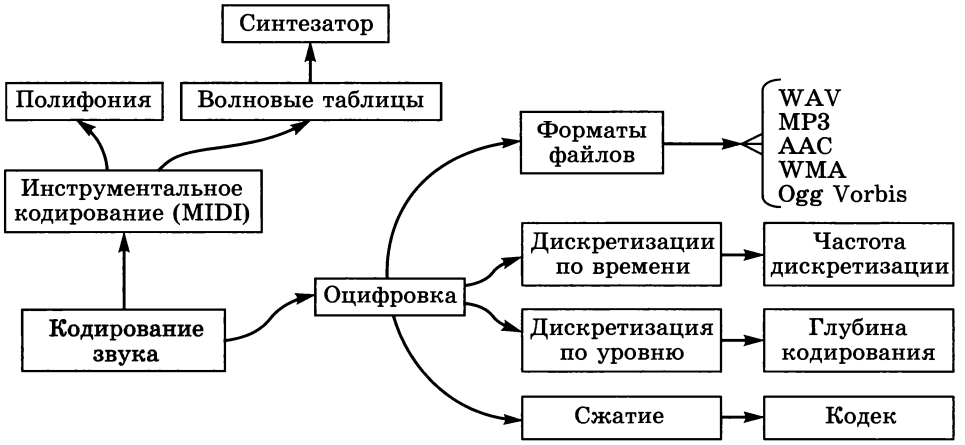


Рис. 2.35

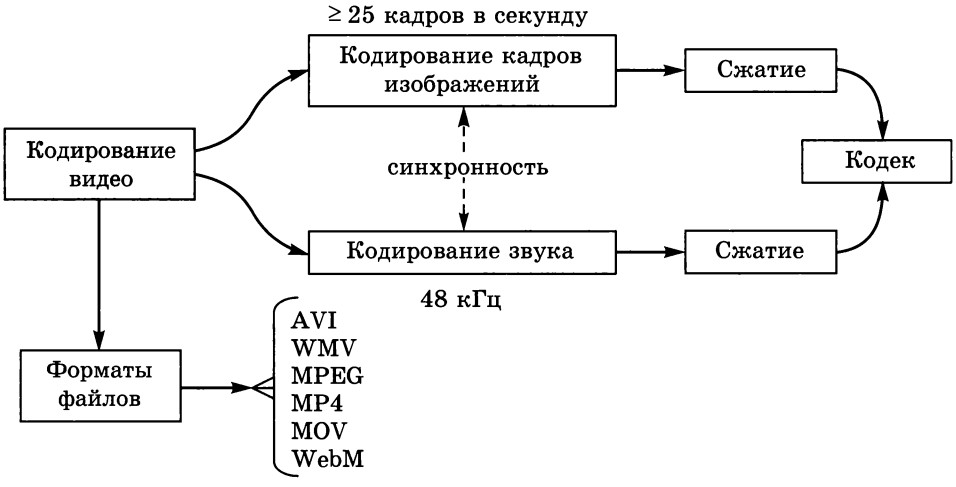


Рис. 2.36

Вопросы и задания

1. Объясните, почему при оцифровке звука происходит потеря информации.
2. Как связана частота дискретизации с потерей информации и объемом файла?
3. От чего зависит выбор частоты дискретизации?
4. Почему частоты дискретизации более 48 кГц применяются очень редко?
5. На что влияет разрядность кодирования звука?
6. У Иннокентия есть оцифрованная запись любимой песни вокальной группы «Рёв бизонов». Он хочет удалить из записи голос, оставив только мелодию. Как вы думаете, просто ли это сделать? Почему?
7. Сравните оцифровку и инструментальное кодирование.
8. Почему MIDI-файлы могут звучать по-разному на разной аппаратуре?
9. Что такое синхронность?
10. Почему при кодировании видео используется частота не менее 25 кадров в секунду?
11. Почему компьютерные фильмы чаще всего хранятся в сжатом виде?
12. Что означает «сжатие с потерями»? В чём состоит его основная идея при кодировании видео?
13. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Как устроена звуковая карта?»
б) «Стандарт MIDI»
в) «Что такое кодек?»
г) «Что такое медиаконтейнер?»
д) «Формат MP3»
е) «Свободные звуковые и видеоформаты»

Интересные сайты

audacity.sourceforge.net — бесплатный аудиоредактор *Audacity*
virtualdub.org — бесплатный видеоредактор *VirtualDub*
videosoftdev.com — бесплатный видеоредактор *VSDC*
kdenlive.org — бесплатный видеоредактор *Kdenlive* для *Linux*
avidemux.sourceforge.net — бесплатный кроссплатформенный видеоредактор *Avidemux*

§ 15
Передача информации

Ключевые слова:

- источник
- приёмник
- канал связи
- помехи
- скорость передачи данных
- биты в секунду
- пропускная способность

После того как информация закодирована, с ней можно что-то делать, например передавать. В этом параграфе мы разберёмся, как это происходит.

Как происходит передача информации?

Информация передаётся от источника к приёмнику через **канал связи.**

Канал связи — это среда и технические устройства, с помощью которых передаётся информация.

Например, при отправке писем и посылок канал связи — это почта, а при разговоре людей — воздух, в котором распространяется звук. Для обмена данными между частями компьютера и в компьютерных сетях используются электрические (кабельные) и оптоволоконные (световые) каналы связи, а в беспроводных сетях — радиоканалы.

Информация идёт по каналу связи не «сама по себе», а должна быть связана с каким-то носителем (рис. 2.37).

Рис. 2.37

Носитель — это объект, который может некоторое время сохранять информацию.

В быту носителем информации может быть бумага, звуковые волны (при разговоре), свет (при получении информации с помощью зрения), химическое вещество (человек ощущает запахи и вкусы). Носители информации в компьютерных системах — это электрический ток или радиоволны. Компьютерные данные можно передавать, записав их на специальные носители: магнитные или оптические диски или устройства флэш-памяти, которые используются не только для передачи, но и для хранения информации.

К сожалению, в реальных каналах связи всегда действуют **помехи**. Они могут искажать сообщение, вплоть до полной потери информации (вспомните телефонные разговоры при слабом сигнале сотовой сети).

Для передачи информации свойства носителя должны изменяться со временем.

Сигнал — это изменение свойств носителя, которое используется для передачи информации.

С помощью одного сигнала (одного изменения) не получится передать много информации. Поэтому чаще всего используется не одиночный сигнал, а последовательность сигналов, которая называется сообщением.

Сообщение — это последовательность сигналов.

В информатике обычно рассматривают передачу информации (данных) именно как передачу сообщений между компьютерными системами, отвлекаясь от смысла этих сообщений.

Вы знаете, что в результате кодирования информации любого вида всегда получается некоторая цепочка нулей и единиц. С точки зрения передачи данных, всё равно, что она обозначает — текст, рисунок, звук или видео, важна только её длина. Как почтальону всё равно, что написано в письмах, так же и при передаче данных их содержание не имеет значения.

Скорость передачи данных

Вспомним, в каких единицах измеряется скорость в уже знакомых нам ситуациях. Для автомобиля скорость — это расстояние, пройденное за единицу времени; скорость измеряется в километрах в час или метрах в секунду. В задачах перекачки жидкости скорость измеряется в литрах в минуту (или в секунду, в час).

Неудивительно, что скоростью передачи данных называют количество данных, переданное по сети за единицу времени (чаще всего — за секунду).

По сети за 10 секунд было передано 200 бит данных. Найдите среднюю скорость передачи данных. В каких единицах она измеряется?

Как вы думаете, почему в предыдущей задаче речь идёт именно о средней скорости?

Количество данных, которые передаются за 1 секунду, можно измерить в любых единицах количества информации: битах, байтах, килобайтах и др. Поскольку данные обычно передаются в виде потока битов (без разделения на байты), на практике скорость передачи данных чаще всего измеряют в битах в секунду (бит/с).

Скорость передачи данных в битах в секунду — это количество битов, переданных за одну секунду.

Средняя скорость передачи данных v вычисляется по формуле

$$v = I / t,$$

где I — количество переданных данных, а t — время передачи.

Эта формула связывает между собой три величины: количество данных, время и среднюю скорость передачи. Зная любые две из них, с помощью этой формулы можно всегда выразить и вычислить третью.

В скоростных сетях скорость обмена данными может составлять миллионы и миллиарды битов в секунду, поэтому используются кратные единицы: 1 кбит/с (килобит в секунду), 1 Мбит/с (мегабит в секунду) и 1 Гбит/с (гигабит в секунду).

---

1 кбит/с = 1 000 бит/с.
1 Мбит/с = 1 000 000 бит/с.
1 Гбит/с = 1 000 000 000 бит/с.

Обратите внимание, что здесь приставки «кило-», «мега-» и «гига-» обозначают (как и в международной системе единиц СИ) увеличение ровно в тысячу, миллион и миллиард раз. Напомним, что в традиционных единицах измерения количества информации «кило-» означает увеличение в 1024 раза, «мега-» — в 1024^2 и «гига-» — в 1024^3.

Файл размером 2000 бит был передан по каналу связи за 40 с. Найдите среднюю скорость передачи данных в битах в секунду.

Файл размером 200 байт был передан по каналу связи за 16 с. Найдите среднюю скорость передачи данных в битах в секунду.

Файл размером 15 Мбайт был передан по каналу связи за 30 с. Найдите среднюю скорость передачи данных в битах в секунду (число в ответе запишите как степень числа 2).

Для каждого канала связи существует некоторая предельная скорость передачи данных, которая зависит от оборудования. Она называется **пропускной способностью** канала связи.

Пусть скорость передачи данных по некоторой сети равна v бит/с. Это значит, что за одну секунду передаётся v битов, а за t секунд — $v \cdot t$ бит.

Скорость передачи данных по линии связи равна 600 бит/с. Сколько бит будет передано за 10 секунд?

Скорость передачи данных по линии связи равна 800 бит/с. Сколько байт будет передано за 100 секунд?

Скорость передачи данных по линии связи равна 80 бит/с. Сколько байт будет передано за 5 минут?

Скорость передачи данных по линии связи равна 2^{22} бит/с. Сколько мегабайт будет передано за 4 минуты?

Если известно количество переданных данных I и скорость передачи v, то время передачи t вычисляется как их отношение: $t = I/v$.

Скорость передачи данных по линии связи равна 1000 бит/с. Сколько секунд потребуется на передачу файла размером 4000 бит?

Скорость передачи данных по линии связи равна 100 бит/с. Сколько секунд потребуется на передачу файла размером 125 байт?

Скорость передачи данных по линии связи равна 4096 бит/с. Сколько секунд потребуется на передачу файла размером 2 Кбайт?

Скорость передачи данных по линии связи равна 2^{25} бит/с. Сколько секунд потребуется на передачу файла размером 12 Мбайт?

Некоторое количество данных передаётся по линии связи за 10 секунд. Оцените, за какое время может быть передано в 10 раз больше данных. Почему на практике это время может отличаться от расчётного?

Некоторое количество данных передаётся по первой линии связи за 10 секунд, а по второй — за 50 секунд. Что можно сказать о скоростях передачи по первой и второй линии? Можно ли определить объём данных, который передавался?

Выводы

- Информация передаётся от источника к приёмнику через канал связи.
- Носитель — это объект, который может сохранять информацию.
- Сигнал — это изменение свойств носителя, которое используется для передачи информации.
- Сообщение — это последовательность сигналов.
- Скорость передачи данных в битах в секунду — это количество битов, переданных за одну секунду.
- Средняя скорость передачи данных вычисляется по формуле $v = I/t$, где I — количество переданных данных, а t — время передачи.
- Для описания скоростных каналов связи используют более крупные единицы:
  1 кбит/с = 1 000 бит/с;
  1 Мбит/с = 1 000 000 бит/с;
  1 Гбит/с = 1 000 000 000 бит/с.
- Пропускная способность канала связи — это наибольшая возможная скорость передачи данных, которая определяется оборудованием.

Интеллект-карта

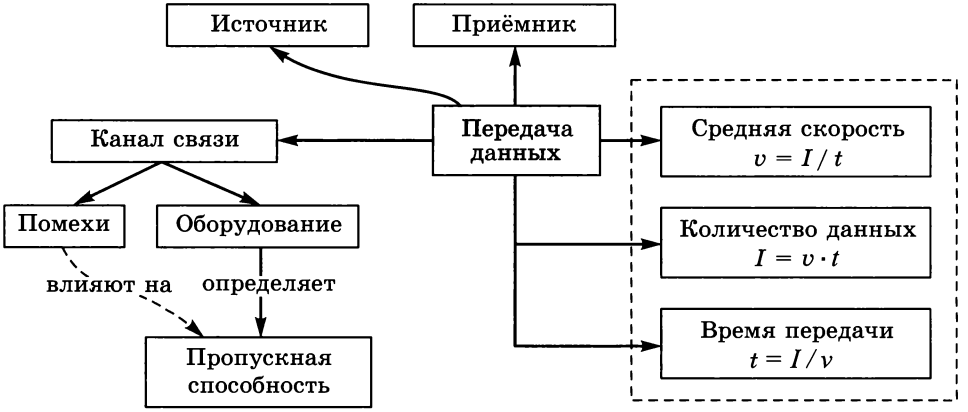


Рис. 2.38

Вопросы и задания

1. Что означают приставки «кило-», «мега-» и «гига-» в единицах измерения скорости передачи данных?
2. Поясните разницу между скоростью передачи данных и пропускной способностью канала связи. Как вы думаете, почему эти величины могут различаться?
3. Выполните по указанию учителя задания в рабочей тетради.

§ 16
Сжатие данных

Ключевые слова:

- сжатие данных
- коэффициент сжатия
- сжатие без потерь
- сжатие с потерями
- архивация
- программа-архиватор
- контрольная сумма
- самораспаковывающийся архив

Зачем и как сжимать данные?

Для того чтобы сэкономить место на внешних носителях (жёстких дисках, «флэшках») или ускорить передачу данных по компьютерным сетям, можно сжать данные — уменьшить их информационный объём, сократить размер файла.

Как вы уже знаете, рисунки часто хранятся в сжатом виде. Кроме того, сжатие почти всегда используется при хранении и передаче звука и видео — упаковку и распаковку этих данных выполняют специальные программы-кодеки.

Покажем, как можно сжать данные, на простом примере. Есть файл, в котором в 8-битной кодировке записаны сначала 100 русских букв А, а потом — 100 букв Б (рис. 2.39).

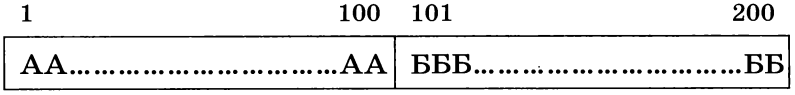


Рис. 2.39

Каждая буква на рис. 2.39 занимает 8 бит. Определите информационный объём файла в байтах.

Теперь запишем те же самые данные иначе: сначала количество повторений первого символа, а затем — сам первый символ, потом так же для второго символа (рис. 2.40).

100	А	100	Б

Рис. 2.40

Каждая ячейка на рис. 2.40 занимает 8 бит. Определите информационный объём файла в байтах.

Объём файла уменьшился, это значит, что мы сжали данные.

Коэффициент сжатия — это отношение размера исходного файла I_0 к размеру сжатого файла $I_{сж}$:

$$k_{сж} = I_0 / I_{сж}.$$

Определите коэффициент сжатия файла в рассмотренном выше примере.

Почему же этот файл удалось так удачно сжать? Всё дело в том, что в нём были длинные цепочки повторяющихся символов, и мы применили алгоритм, который очень удачно их сжимает. Этот алгоритм называется *кодированием цепочек одинаковых символов* (по-английски — **RLE**[1]: *Run Length Encoding*).

---

[1] Алгоритм RLE можно успешно использовать для сжатия рисунков, в которых большие области закрашены одним цветом.

В файле записаны 100 различных символов. Определите коэффициент сжатия файла с помощью алгоритма RLE. Что означает полученное число?

Данные можно сжать, если в них есть какие-то закономерности (избыточность), например одинаковые символы, стоящие рядом, или одинаковые цепочки символов («слова»). Поэтому хорошо сжимаются данные, в которых таких закономерностей много, например тексты и рисунки. Хуже всего сжимаются случайные данные, в которых нет ничего закономерного.

Программы для сжатия данных выявляют избыточность данных и устраняют её, поэтому сжимать второй раз уже сжатые данные чаще всего бесполезно.

Сжатие без потерь

Давайте вернёмся к примеру сжатия, приведённому в начале параграфа. Можем ли мы из сжатого файла (рис. 2.40) восстановить исходный, несжатый (рис. 2.39)? В данном случае — да: читаем из файла число повторений первого символа, затем сам символ, повторяем его нужное количество раз и т. д.

Такое сжатие называют сжатием без потерь. Оно используется для упаковки текстов, программ и других данных, которые ни в коем случае нельзя искажать.

Сжатие без потерь — это такое уменьшение объёма данных, при котором можно восстановить их исходный вид без искажений.

Для каких данных, на ваш взгляд, можно использовать только сжатие без потерь (ни один байт не должен быть искажён):

а) текст;
б) рисунки;
в) программы;
г) звук;
д) видео?

Сжатие с потерями

В некоторых случаях небольшие неточности в передаче данных допустимы. Например, небольшие помехи в телефонном разговоре или в радиопередаче, которая транслируется через Интернет, не мешают пониманию речи. Некоторое дополнительное размытие уже и без этого размытого изображения на фотографии непрофессионал даже не заметит. Ещё в большей степени это относится к видеофильмам, где каждый кадр виден очень короткое время.

Поэтому иногда можно пожертвовать высоким качеством изображения или звука ради того, чтобы значительно сократить объём данных. В этих случаях используется сжатие с потерями.

Сжатие с потерями — это такое уменьшение объёма закодированных данных, при которых распакованный файл может отличаться от оригинала.

Самые известные примеры сжатия с потерями — это алгоритмы JPEG (для изображений), MP3 (для упаковки звука) и все алгоритмы упаковки видеофильмов. Эти алгоритмы используют особенности человеческого зрения и слуха, удаляя данные, которые меньше всего влияют на наше восприятие.

Программы-архиваторы

Архивация — это создание файла-архива, который объединяет группу файлов. Архивацию используют для того, чтобы создать резервную копию данных (на случай выхода из строя основного устройства внешней памяти) или для передачи файлов по сети.

Простейшие программы-архиваторы просто объединяют множество файлов в один архив. Современные архиваторы выполняют ещё и сжатие данных для уменьшения объёма архива. В операционной системе *Windows* наиболее популярен архиватор **WinRAR**, в *Linux* — **Ark** и **File Roller**; в *OS X* применяют **Stuffit Expander** и **BetterZip**.

Свободно распространяемый архиватор **7-Zip** можно скачать бесплатно вместе с исходным текстом программы.

Архиваторы могут упаковать данные в архив с паролем (зашифровать). Тогда для того, чтобы восстановить файлы из архива, потребуется ввести тот же самый пароль (ключ для расшифровки).

Пароли в архивах не хранятся. Перед упаковкой архиватор вычисляет для каждого файла *контрольную сумму* — так называется число, которое зависит от всех данных и вычисляется по достаточно сложному алгоритму, и эти контрольные суммы записываются в архив. При распаковке программа использует пароль, который ввёл пользователь, и сравнивает контрольную сумму полученного файла с той, которая записана в архиве: если они различаются, то пароль был введён неверно.

Для того чтобы пароль было сложно подобрать, он должен быть достаточно длинный (не менее 6 знаков) и содержать разные символы (а не только цифры). Пример хорошего пароля — rH3cyJfl98H.

Придумайте примеры простых паролей, которые можно быстро набрать на клавиатуре, но и поэтому легко подобрать.

Придумайте примеры паролей, которые сложно подобрать.

Многие программы-архиваторы могут создавать **самораспаковывающиеся архивы** (**SFX**-архивы). Так называют файлы, которые содержат не только данные, но и программу для их распаковки. Чтобы распаковать такой архив, не нужно устанавливать архиватор — достаточно просто запустить архив-программу на выполнение.

Самораспаковывающиеся архивы получаются немного больше обычных, потому что содержат программу распаковки (её размер — около 15 Кбайт). Кроме того, их могут заражать компьютерные вирусы. Поэтому для обмена данными лучше использовать обычные архивы.

Международным стандартом архивов считается формат **ZIP** (файлы с расширением *zip*). В нашей стране широко используются также **RAR**-архивы (файлы с расширением *rar*) и архивы в формате **7-Zip** (файлы с расширением *7z*).

Выводы

- Сжатие (уменьшение объёма) данных нужно для того, чтобы сэкономить место во внешней памяти и ускорить передачу данных по сети.
- Сжатие возможно только тогда, когда в данных есть какие-то закономерности (избыточность).
- Коэффициент сжатия — это отношение размеров исходного и сжатого файлов.
- Сжатие без потерь — это такое уменьшение объёма данных, при котором можно восстановить их исходный вид без искажений.
- Сжатие с потерями — это такое уменьшение объёма данных, при которых распакованный файл может отличаться от оригинала. Сжатие с потерями используют при кодировании фотографий, звука и видео.
- Архивация — это создание файла-архива, который объединяет группу файлов.
- Программы-архиваторы могут создавать архивы с паролем.
- Самораспаковывающиеся архивы — это файлы, которые содержат не только данные, но и программу для их распаковки.

Интеллект-карта

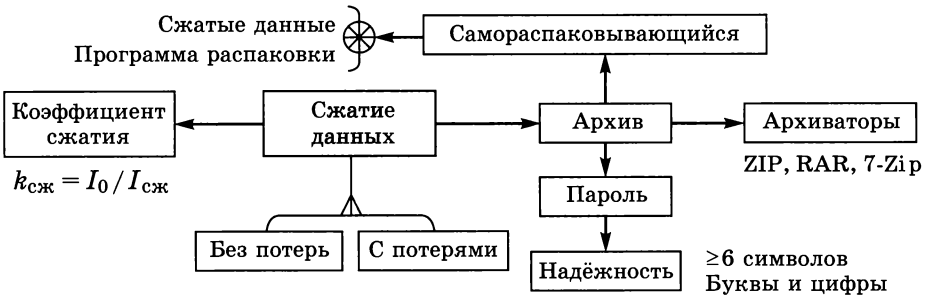


Рис. 2.41

Вопросы и задания

1. Почему некоторые данные сжимаются хорошо, а некоторые — плохо?
2. При сжатии получилось, что коэффициент сжатия равен 0,9. Что это значит?
3. Почему чаще всего нет смысла сжимать уже сжатые данные?
4. Когда допустимо сжатие с потерями? Когда недопустимо?
5. Почему архивы с паролем сложно «взломать», не зная пароля?
6. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Архиватор WinRAR»
б) «Архиватор 7-Zip»

Интересные сайты

rarlab.com — архиватор *WinRAR*.
7-zip.org — свободно распространяемый архиватор *7Zip*.

Практическая работа

Выполните практическую работу № 5 «Использование архиватора».

ЭОР к главе 2 из Единой коллекции цифровых образовательных ресурсов (school-collection.edu.ru)

Декодирование сообщения, записанного азбукой Морзе
Телеграф пишущий Морзе (видеофрагмент)
Дискретизация аналогового сигнала
Вычисление количества информации: алфавитный подход
Вычисление количества информации: смысловой подход
Измерение количества информации: информация как мера уменьшения неопределённости
Вычисление количества информации
Измерение количества информации. Бит, байт, производные единицы
Интерактивный задачник. Раздел «Измерение информации»
История развития систем счисления
Непозиционные системы счисления
Цифровые весы
Перевод десятичных чисел в другие системы счисления
Преобразование десятичного числа в другую систему счисления
Перевод недесятичных чисел в десятичную систему счисления
Схема Горнера
Преобразование чисел между системами счисления 2, 8, 16
Калькулятор систем счисления
Интерактивный задачник, раздел «Системы счисления»
Сложение и вычитание многоразрядных двоичных чисел
Умножение и деление двоичных чисел
Что такое звук? Характеристики звука Звуки одинаковой мощности, но разных частот
Эффект движения

Глава 3
ПРОГРАММИРОВАНИЕ

§ 17
Введение

Ключевые слова:

- программирование
- программист
- программа
- оператор
- комментарий
- оператор вывода

В курсе информатики 7 класса вы уже познакомились с понятием алгоритма и составляли программы для исполнителей Робот и Рисователь. В этом году мы продолжим заниматься программированием, используя уже два языка — алгоритмический язык системы КуМир и язык программирования Паскаль. Вместе с учителем вы выберете для работы один из них, но у вас всегда будет возможность «подсмотреть» в учебнике, как то же самое можно сделать на другом языке.

Алгоритмы

Сначала вспомним основные сведения из курса 7 класса, которые нам понадобятся.

Алгоритм — это точное описание последовательности действий для исполнителя.

Исполнителем называют человека, животное или машину, способных понимать и выполнять некоторые команды.

Формальный исполнитель любую команду всегда выполняет одинаково, не обдумывая её.

Любой алгоритм можно составить с помощью трёх базовых конструкций: **следования** (последовательного выполнения команд), **ветвления** (выбора одного из двух вариантов действий) и **цикла** (повторения одинаковой группы действий).

Алгоритмы можно записывать на **естественном языке** (например, русском), в виде **блок-схем** или на **языке программирования**. Запись алгоритма на языке программирования называется **программой**.

Программирование и программисты

Программирование — это создание программ для компьютеров. Людей, которые этим занимаются, называют **программистами**.

Программист должен уметь:

- анализировать поставленную задачу: определять входные данные и результаты, устанавливать связь между ними, выделять этапы решения задачи и т. д.;
- разрабатывать алгоритм решения;
- писать тексты программ на различных языках программирования;
- отлаживать и тестировать программы;
- готовить описания программ и инструкции для пользователей (документацию);
- дорабатывать и сопровождать программы после сдачи заказчику.

В небольших фирмах все эти задачи часто решает один человек.

В крупных компаниях есть разделение труда: анализом задачи занимаются **системные аналитики**, разработкой алгоритма — **алгоритмисты** (специалисты в предметной области, математики), написанием и отладкой программ — **кодировщики**, тестированием — **тестировщики**, а составлением документации — **технические писатели**.

У каждого программиста есть своя специализация — область, в которой он работает:

- **системный программист** разрабатывает операционные системы, драйверы устройств, утилиты; эта работа требует самых глубоких знаний и способностей к самообразованию, она высоко ценится и оплачивается;
- **прикладной программист** создаёт прикладные программы, с которыми работают пользователи, в том числе программы для мобильных устройств;
- **веб-программисты** занимаются программированием веб-сайтов;
- **программисты баз данных** разрабатывают программы, которые управляют базами данных.

Первая программа

Сначала давайте посмотрим, что представляет собой пустая программа. Это такая программа, которая не содержит никаких команд, но удовлетворяет всем требованиям языка программирования. Компьютер может выполнить её, но делать она, разумеется, ничего не будет. Далее мы будем слева приводить программу на алгоритмическом языке, а справа — такую же программу на языке Паскаль.

```
алг Куку                    program qq;
нач                         begin
  | основная программа        { основная программа }
кон                         end.
```

Как вы знаете из курса 7 класса, программа на алгоритмическом языке начинается служебным словом **алг** (сокращение от слова «алгоритм»), за которым записывают название алгоритма. Название может содержать русские и латинские буквы, цифры, знак подчёркивания (_) и даже пробелы, но не может начинаться с цифры.

Между **служебными словами нач** и **кон** размещают операторы (команды) программы, они составляют **тело программы.**

Оператор — это команда языка программирования.

После вертикальной черты в алгоритмическом языке записывают комментарии — пояснения для человека, которые не обрабатываются транслятором[1].

Комментарии — это пояснения для человека внутри текста программы.

Программа на языке Паскаль выглядит очень похоже, но в ней есть слова, записанные на английском языке. Заголовок программы начинается словом **program** (по-английски — программа), в имени программы нельзя использовать русские буквы и пробелы, после него ставится точка с запятой.

Все служебные слова тоже записываются на английском языке. Тело программы ограничивают служебные слова **begin** и **end**, после слова **end** ставится точка. Комментариями считается всё, что записано в фигурных скобках.

Переведите слова *begin* и *end* с английского языка на русский.

[1] Напомним, что транслятор — это программа, которая переводит текст программы, написанной на языке программирования, в двоичные коды процессора (см. учебник для 7 класса).

Если вы наберёте рассмотренную первую программу без ошибок в среде программирования и запустите её, она отработает, но никаких результатов её работы видно не будет, ведь в теле программы нет ни одной команды.

Вывод текста

Теперь научим программу делать что-то полезное, например выводить текст на экран. Пусть она при запуске приветствует вас:

```
Привет!
```

Вот как выглядит такая программа:

```
алг Привет
нач
  вывод 'Привет!'
кон
```

```
program Hello;
begin
  write('Привет!')
end.
```

Чтобы в алгоритмическом языке вывести символы на экран, нужно записать их в апострофах после слова вывод. Такая команда называется **оператором вывода**. В языке Паскаль для вывода данных используют стандартную (встроенную) процедуру write.

Обратите внимание, что в программе на языке Паскаль каждая команда (оператор) заканчивается точкой с запятой, но перед словом **end** её можно не ставить.

Вася решил дополнить программу так, чтобы она выводила сообщение с его именем. Он записал оператор вывода так:

```
вывод 'Привет', Вася!
```

```
write('Привет', Вася!);
```

Эта программа не работает. Исправьте её.

Теперь попробуем вывести второе приветствие:

```
Привет,  Вася!
Привет,  Петя!
```

Если запустить программу с такими операторами:

```
вывод 'Привет, Вася!'
вывод 'Привет, Петя!'
```

```
write('Привет, Вася!');
write('Привет, Петя!');
```

то мы увидим не совсем то, что хотели:

```
Привет,  Вася!Привет,  Петя!
```

Действительно, мы ведь не сказали компьютеру, что после вывода первого сообщения нужно перейти на другую строку. Сделаем это так:

```
вывод 'Привет, Вася!', нс
вывод 'Привет, Петя!'
```

```
writeln ('Привет, Вася!');
write('Привет, Петя!');
```

Выводы

- Программирование — это создание программ для компьютеров. Людей, которые этим занимаются, называют программистами.
- Оператор — это команда языка программирования.
- Комментарии — это пояснения для человека внутри текста программы.
- Система программирования — это программные средства для создания новых программ.
- Транслятор — это программа, которая переводит в машинные коды (команды процессора) тексты программ, написанных на языке высокого уровня.
- Отладчик — программа для поиска ошибок в разрабатываемых программах.
- Среда программирования обычно включает редактор текста программ, транслятор и отладчик.

Интеллект-карта

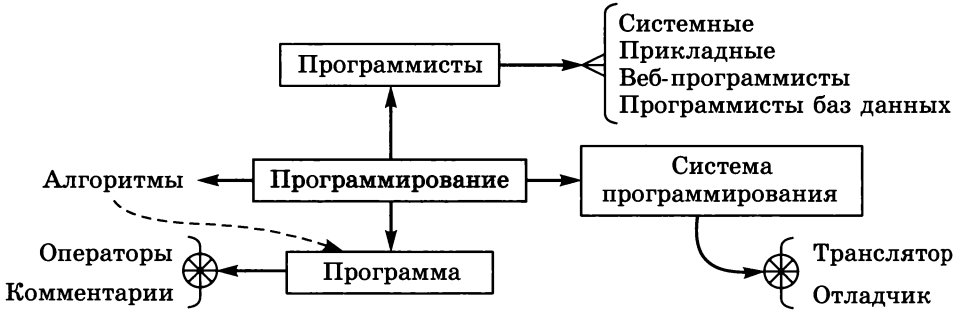


Рис. 3.1

Вопросы и задания

1. Как вы думаете, почему сейчас очень редко алгоритмы записывают в виде блок-схем?
2. Какими качествами, по вашему мнению, должен обладать программист? Обсудите этот вопрос в классе.
3. Зачем пишут комментарии в программах? Подумайте, как комментирование можно использовать при поиске ошибок в программе.
4. Вспомните, что такое служебные (зарезервированные) слова языка программирования.
5. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Профессии в сфере информационных технологий»
б) «Какие бывают языки программирования?»

Интересные сайты

petriv.ho.com.ua/algo/rus/ — среда *АЛГО* для программирования на языке Паскаль
pascalabc.net — среда программирования *PascalABC.NET*

Практическая работа

Выполните практическую работу № 6 «Оператор вывода».

§ 18
Линейные программы

Ключевые слова:

- линейная программа
- переменная
- идентификатор
- объявление переменной
- оператор ввода
- оператор присваивания
- список вывода
- арифметическое выражение
- приоритет операций
- форматный вывод
- случайные числа
- псевдослучайные числа

В этом параграфе мы научимся писать простые программы, которые выполняют вычисления. Команды в программе будут выполняться последовательно, одна за другой. Как вы знаете, такие алгоритмы (и программы) называются **линейными**.

Переменные

Начнём с того, что научим компьютер складывать два целых числа. Программа должна:

1) запросить у пользователя два целых числа;
2) сложить их;
3) вывести результат сложения.

Определите входные данные и результат этого алгоритма.

Запишем программу, в которую вместо команд пока вставим комментарии:

```
алг Сумма                    Program Sum
нач                          begin
  | ввести два числа           {ввести два числа }
  | сложить их                 {сложить их       }
  | вывести результат          {вывести результат}
кон                          end
```

Попробуйте запустить эту программу. Что получилось? Почему?

Компьютер не может выполнить эту программу, потому что команд «ввести два числа» и ей подобных, которые записаны в комментариях, нет в его системе команд. Будем постепенно расшифровывать комментарии — записывать вместо них операторы языка программирования.

Исходные данные (числа), которые будет вводить человек, нужно сохранить в памяти компьютера. Для этого используют переменные.

Переменная — это величина, которая имеет имя, тип и значение. Значение переменной может изменяться во время выполнения программы.

Переменная обозначает ячейку памяти. Она может хранить только одно значение. При записи в неё нового значения «старое» стирается и его уже никак не восстановить.

Переменные в программе необходимо объявлять. При объявлении указывается тип переменной и её **имя** (*идентификатор*, от слова «идентифицировать» — отличать от других). С идентификаторами мы уже встречались: имя программы — это тоже идентификатор.

Идентификатор — это имя программы или переменной.

Вот так объявляются целочисленные переменные (в которых могут храниться только целые значения) с именами *a*, *b* и *c*:

```
цел a, b, c                  var a, b, c: integer;
```

В языке Паскаль описание переменных начинается со служебного слова **var**, после него записывают список переменных и в конце через двоеточие — их тип.

Выясните, что означает английское слово *integer* и откуда произошло служебное слово **var** в языке Паскаль.

Имена переменных строятся по тем же правилам, что и имена программ. В языке Паскаль можно использовать в именах латинские буквы (строчные и прописные буквы не различаются), цифры и знак подчёркивания «_». Имя не может начинаться с цифры, иначе транслятору будет сложнее различить, где начинается имя, а где — число.

В алгоритмическом языке в именах разрешены также пробелы и русские буквы, причём строчные и прописные буквы различаются (поэтому x и X — это разные имена переменных).

Выберите правильные имена переменных:

1	Vasya	СУ-27	@mail_ru
m11	Петя	СУ_27	lenta.ru
1m	Митин брат	_27	"Pes barbos"
m 1	Quo vadis	СУ(27)	<Ладья>

Желательно давать переменным «говорящие» имена, чтобы можно было сразу понять, зачем нужна та или иная переменная. Например, переменная с именем *summa*, скорее всего, служит для хранения какой-то суммы.

Тип переменной нужен для того, чтобы определить:

- какие значения может принимать переменная;
- какие операции можно выполнять с этой переменной;
- сколько памяти нужно выделить для хранения значения.

Значение переменной сразу после объявления не определено: переменной выделяется некоторая область памяти, там мог быть до этого записан любой двоичный код.

Работа с переменными

Присвоить значение переменной можно двумя способами. Во-первых, можно записать нужное значение прямо в программе:

```
a:=5                                a:=5;
```

Оператор, содержащий символы «:=», — это **оператор присваивания**, с его помощью присваивают новое значение переменной. Он выполняется так: вычисляется выражение справа от символов «:=», а затем результат записывается в переменную, имя которой указано слева.

Для вывода значения переменной на экран используют тот же оператор `вывод` (в Паскале — `write`), который раньше применяли для вывода текста:

```
вывод a                             write(a);
```

Что появится на экране после выполнения программы?

```
c:=5                    c:=5;
вывод c                 write(c);
вывод 'c'               write('c');
```

Чем различаются эти два оператора вывода?

Что выведет на экран программма?

```
a:=1                    a:=1;
вывод a                 write(a);
a:=5                    a:=5;
вывод a                 write(a);
```

Оператор

```
i:=i+1                  i:=i+1;
```

заменяет значение `i` на `i+1`, т. е. увеличивает значение переменной `i` на **1**.

Что получится, если рассмотреть запись `i:=i+1` как равенство – уравнение относительно переменной `i`?

Чему будет равно значение переменной `i` после выполнения оператора `i:=i+1`, если до этого оно было равно 17?

Чему будут равны значения переменных *a* и *b* после выполнения программмы

```
a:=a+1                  a:=a+1;
b:=b+1                  b:=b+1;
a:=a+b                  a:=a+b;
b:=b+a                  b:=b+a;
a:=a+1                  a:=a+1;
```

если вначале они имели значения `a` = 4 и `b` = 7?

Очень часто программа хранится как **исполняемый файл** — готовые к выполнению машинные коды. В этом случае текст программы **(исходный код)** нам недоступен и в нём ничего нельзя исправить. Для того чтобы пользователь смог как-то поменять исходные данные, программист может предусмотреть их **ввод с клавиатуры**[1)].

Для ввода данных с клавиатуры используется оператор `ввод` (в Паскале — `read`). Например, ввести значение переменной `a` можно так:

```
ввод a                  read(a);
```

---

1) Можно также вводить данные из файла или принимать через компьютерную сеть, но пока мы не будем обсуждать эти довольно сложные способы.

Выполняя эту команду, компьютер ожидает, пока пользователь наберёт значение и введёт его, нажав клавишу *Enter*. Это значение будет присвоено переменной a, которая указана в операторе ввода.

Если в операторе ввода записаны две переменные:

```
ввод a, b                    read(a,b);
```

то программа ожидает ввода двух чисел (через пробел). Первое из них будет записано в первую переменную (в нашем примере — в переменную a), а второе — во вторую (b).

Приведём полную программу сложения двух чисел:

```
алг Сумма                    program Sum;
нач                          var a, b, c: integer;
  цел a, b, c                begin
  ввод a, b                    read(a, b);
  c:=a+b                       c:=a+b;
  вывод c                      write(c)
кон                          end.
```

У этой программы сложения чисел есть два недостатка:

1) перед вводом данных пользователь не знает, что от него требуется (сколько чисел нужно вводить и каких);
2) результат выдается в виде числа, которое означает неизвестно что.

Хотелось бы, чтобы диалог программы с пользователем выглядел так:

```
Введите два числа: 2 3
2+3=5
```

С помощью какого оператора можно сделать подсказку для ввода — вывести на экран фразу «Введите два числа:»?

При выводе результата ситуация несколько усложняется, потому что нужно вывести значения трёх переменных и два символа: «+» и «=». Можно выводить их по очереди:

```
вывод a                      write(a);
вывод '+'                    write('+');
вывод b                      write(b);
вывод '='                    write('=');
вывод c                      write(c);
```

но удобнее объединить все выводимые данные в один **список вывода**, элементы в котором разделены запятыми:

```
вывод a, '+', b, '=', c      write(a, '+', b, '=', c);
```

Обратите внимание, что имена переменных записаны без апострофов, а все выводимые символы — в апострофах. Если в списке вывода указано имя переменной a, программа выведет не букву «*a*», а значение, которое хранится в переменной a.

Что выведет эта программа при a = 4, b = 5 и c = 9?

```
вывод 'a', '+b', =, c          write('a', '+b', =, c);
```

Исправьте ошибки в операторе вывода:

```
вывод  'c', '-b', =, a          write('c', '-b', =, a);
```

так чтобы при *a* = 4, *b* = 5 и *c* = 9 программа вывела: 9-5=4.

В результате мы получаем такую программу:

```
алг Сумма                       program Sum;
нач                             var a, b, c: integer;
  цел a, b, c                   begin
  вывод 'Введите два числа: '     write('Введите два числа: ');
  ввод a, b                       read(a, b);
  c:=a+b                          c:=a+b;
  вывод a, '+', b, '=', c         write(a, '+', b, '=', c)
кон                             end.
```

Здесь можно было бы обойтись и без переменной c, потому что элементом списка вывода может быть арифметическое выражение, которое сразу вычисляется, и на экран выводится его результат:

```
вывод a, '+', b, '=', a+b      write(a, '+', b, '=', a+b);
```

Арифметические выражения

Арифметические выражения обычно записываются в одну строчку. Они могут содержать константы (постоянные значения), имена переменных, знаки арифметических операций, круглые скобки (для изменения порядка действий). Например, присваивание

$$a \leftarrow \frac{c + b - 1}{2} \cdot d$$

в программе запишется как

```
a:=(c+b-1)/2*d                  a:=(c+b-1)/2*d;
```

Операция умножения обозначается знаком «*», а операция деления — знаком «/».

Какое же действие будет выполняться первым, какое — вторым и т. д.? Это определяется **приоритетом** (старшинством) **операций**. Операции выполняются в следующем порядке:

- действия в скобках;
- умножение и деление, слева направо;
- сложение и вычитание, слева направо.

Таким образом, умножение и деление имеют одинаковый приоритет, более высокий, чем сложение и вычитание.

Определите порядок действий компьютера при вычислении выражения:

```
a:=c+b-1/2*5                    a:=c+b-1/2*5;
```

Запишите присваивание $z \leftarrow a + \frac{b-5}{c+8}$ на языке программирования.

Результат деления (операции «/») может быть нецелым числом, такие числа называются **вещественными**. Вещественное значение нельзя записать в целочисленную переменную. Вещественную переменную x объявляют так:

```
вещ x                           var x: real;
```

При записи вещественных чисел в программе целую и дробную части разделяют не запятой (как принято в России), а точкой. Например:

```
вещ x                           var x: real;
x:=123.456                      x:=123.456;
```

В алгоритмическом языке есть операция возведения в степень (для целых и вещественных чисел), которая обозначается двумя звездочками: «**». Например, присваивание $y \leftarrow 2x^2 + z^3$ запишется так:

```
y:=2*x**2+z**3
```

Возведение в степень имеет более высокий приоритет, чем умножение и деление. В языке Паскаль операции возведения в степень нет.

Как можно записать возведение в квадрат, куб, 10-ю степень на языке Паскаль?

Операции с целыми числами

Часто нужно получить целый результат деления целых чисел и остаток от деления. Например, известен интервал времени в секундах (скажем, 175 секунд) и нужно определить, сколько в нём целых минут и оставшихся секунд (175 с = 2 мин 55 с). Здесь число минут — это целая часть от деления 175 на 60, а 55 секунд — это остаток от этого деления.

В таких случаях в алгоритмическом языке используют команды div и mod, а в Паскале — операции с теми же именами (они имеют такой же приоритет, как умножение и деление):

```
t:=175                          t:=175;
m:=div(t, 60)    | = 2          m:=t div 60    {= 2}
s:=mod(t, 60)    | = 55         s:=t mod 60    {= 55}
```

С помощью этих операций удобно работать с отдельными цифрами числа. Как мы увидели в главе 2, остаток от деления числа на 10 — это последняя цифра его десятичной записи[1].

```
N:=123                          N:=123;
d1:=mod(N, 10)    | =3          d1:=N mod 10;    { =3 }
```

Чему равен остаток от деления числа N на 100?

```
d12:=mod(N, 100)                d12:=N mod 100;
```

Если разделить число на 10 и взять только целую часть, мы отбросим последнюю цифру числа: значение div(123, 10) равно 12.

```
N:=123                          N:=123;
d:=div(N, 10)          | =12    d:=N div 10;          { =12 }
```

Как с помощью операций div и mod выделить вторую с конца цифру числа?

Вывод данных на экран

Давайте произведём эксперимент. Посмотрим, что выведет на экран такая программа:

```
цел a, b, c                     var a, b, c: integer;
вещ x                               x: real;
a:=1; b:=2; c:=3                ...
x:=12.34567891234               a:=1; b:=2; c:=3;
вывод a, b, c, нс               x:=12.34567891234;
вывод x                         writeln(a, b, c);
                                writeln(x);
```

В среде КуМир вы увидите такую запись:

```
123
12.345679
```

[1] А остаток от деления на N — последняя цифра записи числа в системе счисления с основанием N.

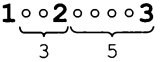
Все три значения, записанные в первом операторе вывода, сливаются друг с другом. Для того чтобы разделить их, нужно добавить в список вывода пробелы (в апострофах!) между каждой парой значений переменных:

```
вывод a, ' ', b, ' ', c, нс      writeln(a, ' ', b, ' ', c);
```

Есть и другой вариант: указать через двоеточие, сколько позиций на экране нужно отвести на вывод значения:

```
вывод a, b: 3 , c: 5 , нс      writeln(a, b: 3 , c: 5 );
```

Это так называемый **форматный вывод**: мы сказали, что значение переменной b должно занимать три позиции на экране, а значение c — пять:



Так как значение b (равное 2) состоит всего из одного знака, слева от него выводится два пробела, которые обозначены на рисунке знаком «∘». Значение c (оно равно 3) тоже занимает одну позицию, слева от него выводится четыре пробела, чтобы общее число позиций было равно пяти. Для значения a формат не указан, поэтому компьютер использует минимальное необходимое число знаков.

Что будет выведено в результате работы следующей программы?

```
цел a=1,b=2,c=3
цел d=4,e=5
вывод a:4, нс
вывод b:3,b:2, нс
вывод c:2,c:4, нс
вывод d, d, d, d, d, d, d, нс
вывод e:4
```

```
var a, b, c, d, e: integer;
...
a:=1; b:=2; c:=3;
d:=4; e:=5;
writeln(a:4);
writeln(b:3,b:2);
writeln(c:2,c:4);
writeln(d, d, d, d, d, d, d);
write(e:4);
```

Теперь разберёмся с выводом вещественного числа 12,34567891234. Компьютер округлил его до 6 знаков в дробной части (это **формат вывода по умолчанию**). Но мы можем сами задать формат вывода, например, такими способами:

		Результат вывода:
`вывод 'x=', x:10:3`	`write('x=', x:10:3);`	x=∘∘∘∘12.346
`вывод 'x=', x:8:2`	`write('x=', x:8:2);`	x=∘∘∘12.35
`вывод 'x=', x:2:2`	`write('x=', x:2:2);`	x=12.35
`вывод 'x=', x:0:2`	`write('x=', x:0:2);`	x=12.35
`вывод 'x=', x:0:1`	`write('x=', x:0:1);`	x=12.3

Исследуйте эту таблицу и выясните, что означают два числа, которые записываются после двоеточий.

Если не нужно выводить лишние пробелы, а требуется только ограничить количество знаков в дробной части числа, можно задать нулевое общее количество позиций:

```
вывод x: 0 :3                    writeln(x: 0 :3);
```

Тогда программа расширит поле вывода до необходимого числа знаков.

Выполните команду

```
вывод x                          write(x);
```

при различных значениях переменной *x*:

0,0001	0,00001	0,000001	0,0000001
10000	100000	1000000	10000000

и запишите в тетрадь результат вывода.

При выводе очень больших или очень маленьких вещественных чисел (а в некоторых версиях Паскаля по умолчанию для всех чисел) можно увидеть такую запись:

```
1.234568E+001
```

Это **научный формат** (стандартный вид числа). Число слева от буквы E — это значащая часть числа (она всегда больше или равна 1 и меньше 10), а число справа от неё — показатель степени числа 10. В нашем случае выведено число $1{,}234568 \cdot 10^1 = 12{,}34568$.

Что будет выведено в результате работы следующей программы?

```
вещ x                            var x: real;
x:=172.3658                      ...
вывод x, нс                      x:=172.3658;
вывод x:10:2, нс                 writeln(x);
вывод x:0:8                      writeln(x:10:2);
                                 writeln(x:0:8);
```

Программа вывела числа в научном формате:

а) 1.2345E+001
б) 2.345E+003
в) 5.6E+005
г) 8.74E+00
д) 1.8752E-01
е) 3.462752E-03

Запишите их в обычном виде.

Операции с вещественными числами

При работе с вещественными числами часто приходится округлять их до ближайших целых чисел. Для этого в языке Паскаль есть две функции:

- `trunc(x)` — отбрасывание дробной части вещественного числа *x*;
- `round(x)` — округление вещественного числа *x* до ближайшего целого.

В алгоритмическом языке есть функция выделения целой части числа: `int(x)`. Нужно иметь в виду, что функция `trunc` в Паскале отбрасывает дробную часть, а функция `int` в алгоритмическом языке находит целую часть по правилам математики — как наибольшее целое число, не превосходящее данное. Поэтому для отрицательных чисел они дают разные результаты:

```
вывод int(-1.5) | = -2    write(trunc(-1.5)); { = -1 }
```

Функция `frac(x)` в языке Паскаль выделяет дробную часть вещественного числа `x`.

Как можно выделить дробную часть положительного вещественного числа в алгоритмическом языке?

Что будет выведено на экран в результате работы следующей программы?

```
вещ a=1,b=2,c=3,d=7, x
вывод a/b:0:2, нс
x:=b/c
вывод x:0:2, int(x):2, нс
вывод x-int(x):0:2, нс
x:=d/c
вывод x:0:2, int(x):2, нс
вывод x-int(x):0:2
```

```
var a, b, c, d, x: real;
...
a:=1; b:=2; c:=3; d:=7;
writeln(a/b:0:2);
x:=b/c;
writeln(trunc(x):0:2,
        round(x):2);
writeln(x:0:2,frac(x):2);
x:=d/c;
writeln(x:0:2,frac(x):2);
writeln(trunc(x):0:2,
        round(x):2);
```

Чтобы извлечь квадратный корень из числа, т. е. найти такое число, квадрат которого равен заданному, применяют функцию `sqrt`. Вот так можно найти квадратный корень из числа 5:

```
вывод sqrt(5):0:3 | = 2.236  write(sqrt(5):0:3); { = 2.236}
```

Операции `div` и `mod` к вещественным числам неприменимы.

Напишите программу, которая вычисляет квадратный корень введённого числа. Вычислите с её помощью квадратные корни из чисел 221841; 32005,21 и 15239,9025.

Вычисления с вещественными числами могут давать неточные результаты. Вспомните, что число 1/3 не может быть точно записано в десятичной системе — его дробная часть содержит бесконечное число цифр. В компьютерах происходит то же самое: на каждое число выделяется конечное число разрядов, поэтому большинство вещественных чисел хранится в памяти компьютера неточно. Следовательно, вычисления тоже будут неточными.

Вычислите сумму $X = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} = \frac{n}{d}$ в виде простой дроби.

Проверьте, что выведет эта программа (вместо многоточий добавьте полученные значения n и d):

```
вещ x, y
цел n=..., d=...
x:=1/2+1/3+1/4+1/5
y:=n/d
вывод x, нс
вывод y, нс
вывод x-y, нс
```

```
var x, y: real;
    n, d: integer;
...
n:=...; d:=...;
x:=1/2+1/3+1/4+1/5;
y:=n/d;
writeln(x);
writeln(y);
writeln(x-y);
```

Сделайте выводы.

Случайные и псевдослучайные числа

В некоторых задачах, например в компьютерных играх, необходимо моделировать случайные явления, например результат бросания игрального кубика (на нём может выпасть число от 1 до 6). Как сделать это на компьютере, который «неслучаен», т. е. строго выполняет заданную ему программу?

Случайные числа — это последовательность чисел, в которой невозможно предсказать следующее число, даже зная все предыдущие.

Чтобы получить истинно случайные числа, можно, например, бросать игральный кубик или измерять какой-то шумовой сигнал (например, шум радиоэфира или сигнал, принятый из космоса). Так раньше составлялись таблицы случайных чисел, которые публиковались в книгах.

Но вернёмся к компьютерам. Ставить сложные приборы на каждый компьютер очень дорого, и повторить эксперимент будет невозможно — завтра все значения будут уже другими. Существующие таблицы слишком малы, когда, скажем, нужно получать 100 случайных чисел каждую секунду. Для хранения больших таблиц требуется много памяти.

Чтобы выйти из положения, математики придумали алгоритмы получения **псевдослучайных чисел** («как бы случайных»). Для стороннего наблюдателя псевдослучайные числа практически неотличимы от случайных, но они вычисляются по некоторой математической формуле: зная первое число («зерно»), можно по формуле вычислить второе, затем третье и т. п.

В алгоритмическом языке существуют две функции для получения случайных (точнее, псевдослучайных) чисел:

- `rand(a, b)` — случайное вещественное число в полуинтервале [a; b) (не включая b);
- `irand(a, b)` — случайное целое число на отрезке [a; b].

В Паскале есть аналогичные функции:

- `random` — случайное вещественное число в полуинтервале [0; 1);
- `random(N)` — случайное целое число на отрезке [0; N–1].

Для того чтобы записать в целую переменную *n* случайное целое число на отрезке [1; 6] (результат бросания кубика), можно использовать такие операторы:

```
n:=irand(1, 6)                    n:=random(6)+1;
```

Разберём подробно оператор на языке Паскаль. Вызов `random(6)` даёт нам случайное целое число на отрезке [0; 5]. Если мы добавляем к этому значению единицу, то весь отрезок сдвигается на единицу вправо, получается отрезок [1; 6], что нам и требовалось. В общем случае для получения случайного числа на отрезке [a; b] в Паскале нужно использовать вызов

```
n:=random(b-a+1)+a;
```

Докажите, что по этой формуле действительно получаются случайные числа на отрезке [a; b].

Вещественное случайное число в полуинтервале от 5 до 12 (не включая 12) получается так:

```
x:=rand(5, 12)                    x:=7*random+5;
```

В общем виде, для полуинтервала [a; b):

```
x:=rand(a, b)                     x:=(b-a)*random+a;
```

Докажите, что по этой формуле действительно получаются случайные числа в полуинтервале [a; b).

Выводы

- **Переменная — это величина, которая имеет имя, тип и значение. Значение переменной может изменяться во время выполнения программы.**
- **Идентификатор — это имя программы или переменной.**
- **В большинстве языков программирования переменные нужно объявлять.**

- Основные операции с переменными — ввод, вывод, присваивание нового значения.
- Если переменной присваивается новое значение, предыдущее уничтожается.
- Арифметические выражения обычно записываются в одну строку. Операция умножения обозначается знаком «*», а операция деления — знаком «/».
- Арифметические операции выполняются в следующем порядке: действия в скобках; умножение и деление слева направо; сложение и вычитание слева направо.
- Для целочисленного деления используется операция `div`, для вычисления остатка от деления — операция `mod`.
- Случайные числа — это последовательность чисел, в которой невозможно предсказать следующее число, даже зная все предыдущие. На компьютере обычно используют псевдослучайные числа, которые получают по некоторой формуле.

Интеллект-карта

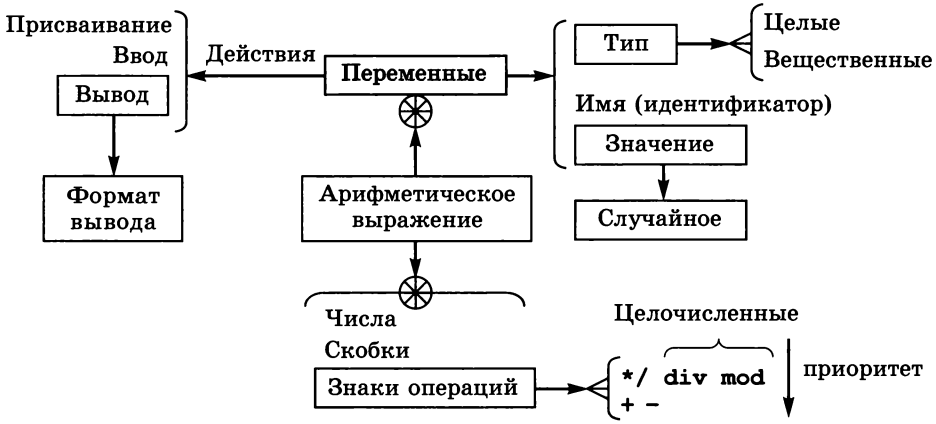


Рис. 3.2

Вопросы и задания

1. В некоторых языках программирования (например, в языке *Python*) переменные не нужно объявлять. Обсудите в классе, какие достоинства и недостатки имеет такой подход.
2. Какое значение записано в переменной сразу после объявления? Можно ли его использовать?
3. Почему желательно выводить на экран подсказку перед вводом данных?

4. Чем отличаются два оператора вывода:

```
вывод a                    write(a);
```

и

```
вывод 'a'                  write('a');
```

Какой из них может привести к ошибке? В каком случае?

5. Когда можно вычислять результат прямо в операторе вывода, а когда нужно заводить отдельную переменную?
6. В каком порядке выполняются операции, если они имеют одинаковый приоритет?
7. Зачем используются скобки в арифметических выражениях?
8. Объясните, чем отличаются случайные числа от псевдослучайных. Почему в компьютерах используются именно псевдослучайные числа?
9. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Научный формат вывода чисел»
б) «Как получают псевдослучайные числа?»

Практические работы

Выполните практические работы:
№ 7 «Линейные программы»;
№ 8 «Операции с целыми числами»;
№ 9 «Операции с вещественными числами»;
№ 10 «Случайные числа».

§ 19
Ветвления

Ключевые слова:

- условный оператор
- полная форма условного оператора
- неполная форма условного оператора
- составной оператор
- вложенный условный оператор
- сложное условие
- операция «И»
- операция «ИЛИ»
- операция «НЕ»
- логические переменные
- экспертная система

Условный оператор

Сейчас мы умеем писать *линейные программы*, в которых операторы выполняются последовательно друг за другом и порядок их выполнения не зависит от входных данных.

В большинстве реальных задач порядок действий может несколько изменяться, в зависимости от того, какие данные поступили. Например, программа для системы пожарной сигнализации должна выдавать сигнал тревоги, если датчики показывают повышение температуры или задымленность.

Требуется записать в переменную *M* наибольшее из значений переменных *a* и *b*. Сформулируйте алгоритм решения задачи в словесной форме.

Для этой цели в языках программирования предусмотрены **условные операторы (ветвления)**. В 7 классе мы изучали разветвляющиеся алгоритмы для исполнителя Робот, а сейчас будем работать с числами. Например, для того чтобы записать в переменную M максимальное (наибольшее) из значений переменных a и b, можно использовать оператор:

```
если a>b то
  M:=a
иначе
  M:=b
все
```

```
if a>b then
  M:=a
else
  M:=b;
```

Видно, что запись на Паскале получена в результате перевода служебных слов алгоритмического языка на английский язык. Обратите внимание, что в языке Паскаль перед служебным словом **else** точка с запятой **не** ставится.

Кроме знаков < и > в условиях можно использовать и другие знаки отношений: <= (меньше или равно), >= (больше или равно), = (равно) и <> (не равно, два знака, < и > без пробела).

В этом примере условный оператор записан в **полной форме**: в обоих случаях (истинно условие или ложно) нужно выполнить некоторые действия.

Найдите и запишите в тетрадь перевод английских слов *if*, *then*, *else*.

Программа выбора максимального значения может быть написана иначе:

```
M:=a
если b>a то
  M:=b
все
```

```
M:=a;
if b>a then
  M:=b;
```

Здесь использован условный оператор в **неполной форме**: в случае, когда условие ложно, ничего делать не требуется (нет слова **иначе** и операторов после него).

Программист написал программу для выбора наименьшего из двух чисел так:

```
если a<b то
  M:=a
все
если b<a то
  M:=b
все
```

```
if a<b then
  M:=a;
if b<a then
  M:=b;
```

В каких случаях эта программа будет работать неправильно? Запишите программу правильно, используя один условный оператор в полной форме.

Можно ли в этой программе два условных оператора в неполной форме заменить на один оператор в полной форме? Почему?

```
если a<5 то
  a:=5
все
если a>10 то
  a:=10
все
```

```
if a<5 then
  a:=5;
if a>10 then
  a:=10;
```

Что делает эта программа?

Для того чтобы сделать текст программы более понятным, всё тело условного оператора сдвигается вправо. Вообще говоря, это не обязательно: в Паскале вся программа может быть записана в одну строку, и транслятор её поймёт. Однако если программа записана с отступами, в ней значительно проще разбираться. Поэтому мы будем всегда записывать программы с отступами. Система КуМир делает отступы автоматически.

Напишите последовательность команд, с помощью которой можно поменять значения двух переменных.

Составной оператор

Часто при выполнения некоторого условия нужно выполнить сразу несколько действий. Например, в задаче сортировки значений

переменных a и b по возрастанию нужно поменять местами значения этих переменных, если a > b:

```
если a>b то
  c:=a
  a:=b
  b:=c
все
```

```
if a>b then begin
  c:=a;
  a:=b;
  b:=c
end;
```

В алгоритмическом языке форма записи не меняется, а в Паскале после служебного слова **then** нужно записать составной оператор, зону действия которого ограничивают слова **begin** и **end** (между ними может быть сколько угодно операторов).

Ветвления в других языках программирования

Знание хотя бы одного языка программирования позволяет понимать запись программы на многих других языках. Вот фрагменты программы, которая меняет местами значения двух переменных, на языках **Python** (слева) и **C** (справа):

```
if a>b:
  c=a
  a=b
  b=c
```

```
if (a>b) {
  c=a;
  a=b;
  b=c;
  }
```

В этих языках оператор присваивания записывается с помощью одного знака «=». В языке *Python* операторы, входящие в составной оператор, записываются с одинаковым отступом вправо, а в языке *C* — ограничены фигурными скобками. Условие в языке *C* обязательно заключается в круглые скобки.

Вложенные условные операторы

В теле условного оператора могут находиться любые операторы, в том числе и другие условные операторы. Например, пусть возраст Андрея записан в переменной a, а возраст Бориса — в переменной b. Нужно определить, кто из них старше. Одним условным оператором тут не обойтись, потому что есть три возможных результата: старше Андрей, старше Борис и оба одного возраста. Решение задачи можно записать так:

```
если a>b то
  вывод 'Андрей старше'
иначе
  если a=b то
    вывод 'Одного возраста'
  иначе
    вывод 'Борис старше'
все
```

```
if a>b then
  writeln('Андрей старше')
else
  if a=b then
   writeln('Одного возраста')
  else
   writeln('Борис старше')
```

Условный оператор, проверяющий равенство, находится внутри блока **иначе (else)**, поэтому он называется **вложенным** условным оператором. Использование вложенных условных операторов позволяет выбрать один из нескольких (а не только из двух) вариантов.

Напишите другой вариант решения последней задачи. Сколько всего вариантов можно придумать?

При работе с вложенными условными операторами в языке Паскаль нужно помнить правило: блок **else** относится к ближайшему предыдущему оператору **if**, у которого такого блока ещё не было. Например, оператор

```
if a>b then write('A') else if a=b then write('=')
else write('Б');
```

может быть записан с отступами так:

```
if a>b then
  write('A')
else
  if a=b then
    write('=')
  else
    write('Б');
```

Здесь второй блок **else** относится к ближайшему (второму, вложенному) условному оператору, поэтому буква «Б» будет выведена только тогда, когда оба условия окажутся ложными.

Запишите с отступами программу на Паскале:

```
if x>=0 then if x>0 then write(1) else write(0)
else write(-1);
```

Что выведет эта программа при $x = -3$? $x = 0$? $x = -123$?

Сложные условия

Предположим, что ООО «Слонопотам» набирает сотрудников, возраст которых от 25 до 40 лет включительно. Нужно написать программу, которая запрашивает возраст претендента и выдаёт ответ: подходит он или не подходит по этому признаку.

Какое же условие должно быть истинно для того, чтобы человека приняли на работу? Одного условия «возраст >= 25» недостаточно, это условие соблюдается и для людей старше 40 лет. Вместе с тем условия «возраст <= 40» тоже недостаточно, так как оно выполняется и для школьников. В этой задаче нужно, чтобы два **условия выполнялись одновременно**: «возраст >= 25» и «возраст <= 40».

Пусть в переменной *v* записан возраст сотрудника. Запишите решение этой задачи с помощью вложенных условных операторов.

Эту задачу можно решить с помощью вложенного условного оператора, но решение получается некрасивое: оно запутанное, кроме того, один и тот же ответ «не подходит» приходится выводить в двух местах программы.

Почти во всех языках программирования можно использовать так называемые сложные условия. В нашем случае условный оператор будет выглядеть так:

```
если v>=25 и v<=40 то
  вывод 'подходит'
иначе
  вывод 'не подходит'
все
```

```
if (v>=25) and (v<=40) then
  writeln('подходит')
else
  writeln('не подходит');
```

Это решение получилось значительно короче и понятнее.

Здесь в условном операторе мы записали **сложное условие**

```
v>=25 и v<=40
```

```
(v>=25) and (v<=40)
```

составленное из двух простых с помощью операции И. В алгоритмическом языке операция И записывается как строчная буква **и**, а в Паскале вместо «и» используется английское слово **and**.

Обратите внимание, что в Паскале каждое простое условие заключается в скобки. Это связано с тем, что в этом языке отношения выполняются позже (имеют более низкий приоритет), чем операция И.

Операция И означает одновременное выполнение двух или нескольких условий.

Что будет выведено на экран после выполнения программы?

```
если a=1 и a=2 то
  вывод 'Да!'
иначе
  вывод 'Нет.'
все
```

```
if (a=1) and (a=2) then
  write('Да!')
else
  write('Нет.')
```

Предположим, что нам надо убедиться, что значение целой переменной *a* — трёхзначное число, которое делится на 7. Для этого нужно, чтобы одновременно выполнились три условия:

1) число не меньше 100 (> 99);
2) число меньше 1000;
3) число делится на 7, т. е. остаток от его деления на 7 равен нулю.

В условном операторе тогда нужно использовать две операции И, которые связывают три простых условия:

```
если 99<a и a<1000 и mod(a, 7)=0 то
  вывод 'Да!'
иначе
  вывод 'Нет.'
все
```

Вот решение на Паскале:

```
if (99<a) and (a<1000) and (a mod 7=0) then
  writeln('Да!')
else
  writeln('Нет.');
```

Рассмотрим ещё одну задачу. В переменной *d* записан номер дня недели (1 — понедельник, ... 7 — воскресенье). Программа должна определить, выходной это день или рабочий (выходные дни у большинства людей — суббота и воскресенье).

Если мы напишем условие `d=6 и d=7`, то это будет неверно, потому что тогда мы потребуем, чтобы значение переменной *d* было *одновременно* равно и 6, и 7. Такого быть не может, поэтому условие всегда будет ложно. Значит, операция И не подходит. Вместо неё нужно применить другую операцию — ИЛИ, которая требует выполнения хотя бы одного из набора условий.

Операция ИЛИ означает выполнение хотя бы одного из двух или нескольких условий.

Решение нашей задачи выглядит так:

```
если d=6 или d=7 то
  вывод 'Выходной!'
иначе
  вывод 'Рабочий день.'
все
```

```
if (d=6) or (d=7) then
  writeln('Выходной!')
else
  writeln('Рабочий день.');
```

В языке Паскаль операция ИЛИ обозначается словом **or** (в переводе с английского — «или»).

Напишите другой вариант решения последней задачи, использующий операцию И.

В обоих языках существует ещё одна операция, которую можно использовать в сложных условиях, — НЕ, в Паскале она обозначается словом **not** (в переводе с английского — «не»).

Операция НЕ означает обратное условие (противоположное исходному).

Например, решение задачи определения выходных дней можно было записать так:

```
если не(d=6 или d=7) то
  вывод 'Рабочий день.'
иначе
  вывод 'Выходной!'
все
```

```
if not((d=6) or d=7))
then
  writeln('Рабочий день.')
else
  writeln('Выходной!');
```

Используя операцию НЕ, можно записывать условия по-разному, как нам удобнее в каждом случае. Например, условия `a=b` и **не**`(a<>b)` истинны для одних и тех же значений `a` и `b`, поэтому одно из них можно заменить на другое. Такие условия называются **равносильными**.

Запишите в тетради равносильные условия, не используя операцию НЕ:

а) **не**`(a<6)`	**not**`(a<6)`
б) **не**`(b=c+d)`	**not**`(b=c+d)`
в) **не**`(c<>15)`	**not**`(c<>15)`
г) **не**`(7<a` **и** `a<12)`	**not**`((7<a)` **and** `(a<12))`
д) **не**`(b<>c` **и** `d<5)`	**not**`((b<>c)` **or** `(d<5))`

Операции И, ИЛИ и НЕ — это **логические операции**, которые мы будем подробно изучать в 9 классе. Они работают с логическими значениями («да»/«нет», «истина»/«ложь»).

Если в сложном условии встречается несколько разных операций, они выполняются в следующем порядке (во всех случаях — слева направо):

1) операции в скобках;
2) операции НЕ;
3) операции И;
4) операции ИЛИ.

Изменить порядок действий можно с помощью круглых скобок.

Определите порядок операций при определении истинности условия.

Алгоритмический язык:

```
не(a>10) или не(a<20) и (a<b)
```

Паскаль:

```
not(a>10) or not(a<10) and (a<b)
```

Определите, истинно или ложно это выражение при a = 5, b = 10.

Для выражения в предыдущем задании запишите равносильное выражение без использования операции «НЕ». После этого расставьте одну пару скобок так, чтобы значение выражения при a = 5, b = 10 изменилось на обратное.

Логические переменные

И в алгоритмическом языке, и в Паскале можно использовать переменные, которые могут принимать только логические значения. В алгоритмическом языке они объявляются с помощью служебного слова **лог**, их возможные значения — да и нет. В Паскале логические переменные относятся к типу boolean[1] и принимают значения True («истина») или False («ложь»):

```
лог b                    var b: boolean;
...                      ...
b:=да                    b:=True;
b:=нет                   b:=False;
```

Обратите внимание, что логические значения записываются без апострофов.

В логической переменной можно хранить значение какого-то условия и затем использовать его в условном операторе:

[1] Их ещё называются *булевскими* в честь *Джорджа Буля* — создателя алгебры логики.

```
лог выходной
выходной:=(d=6 или d=7)
если не выходной то
  вывод 'Рабочий день.'
иначе
  вывод 'Выходной!'
все
```

```
var vyh: boolean;
...
vyh:=(d=6) or (d=7);
if not vyh then
  writeln('Рабочий день.')
else
  writeln('Выходной!');
```

С клавиатуры вводятся три числа и записываются в переменные a, b и c. Напишите программу, которая записывает в логическую переменную значение да (True), если среди чисел a, b, c найдётся пара чисел, сумма которых равна 25.

Экспертная система

Эксперт — это человек, который обладает глубокими теоретическими знаниями и практическим опытом работы в некоторой области. Например, врач-эксперт хорошо ставит диагноз и лечит потому, что имеет медицинское образование и большой опыт лечения пациентов. Он не только знает факты, но и понимает их взаимосвязь, может объяснить причины явлений, сделать прогноз, найти решение в конфликтной ситуации.

Экспертная система — это компьютерная программа, задача которой — заменить человека-эксперта при выработке рекомендаций для принятия решений в сложной ситуации.

Экспертная система содержит базу знаний, в которой хранятся факты и правила, по которым из этих фактов делаются выводы.

Мы построим простейшую экспертную систему, которая задаёт пользователю вопросы и по его ответам определяет класс животных.

Предположим, что в базу знаний внесены следующие правила:

- если у животного есть перья, то это птица;
- если животное кормит детенышей молоком, то это — млекопитающее;
- если животное — млекопитающее и ест мясо, то это — хищник.

Диалог пользователя с экспертной системой может быть, например, таким (ответы пользователя выделены курсивом):

Это животное кормит детей молоком? (*Нет.*)

Это животное имеет перья? (*Да.*)

Это птица.

Для того чтобы определить последовательность вопросов, эксперт строит **дерево решений**, например такое (рис. 3.3).

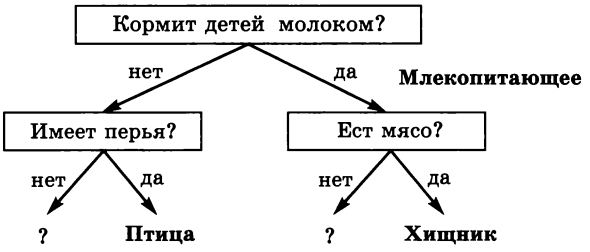


Рис. 3.3

В прямоугольниках записаны вопросы, которые задаёт система пользователю, у стрелок — его возможные ответы («да» или «нет»). Жирным шрифтом выделены **выводы** — результат работы экспертной системы.

Постройте трёхуровневое дерево решений для своей экспертной системы.

Конечно, эта система позволяет определить не все классы животных: в некоторых местах на схеме стоят знаки вопроса. В этих случаях наша программа будет выдавать ответ «Не знаю».

Человеку удобнее вводить ответ словами («да», «нет»). Для сохранения в памяти такого ответа нужно использовать переменную специального типа — строку символов. Такой тип данных в алгоритмическом языке называется **литерным** (от слова «литерный» — буквенный) и обозначается **лит**, а в Паскале называется строковым и обозначается словом `string` (в переводе с английского — строка):

```
лит ответ
```

```
var otvet: string;
```

Программа начинает диалог с вопроса «Кормит детей молоком?» и в зависимости от ответа выбирает одну из двух ветвей дерева решений (см. рис. 2.3).

```
вывод 'Кормит детей молоком? '
ввод ответ
если ответ = 'да' то
  ... | вариант 1
иначе
  ... | вариант 2
все
```

```
write('Кормит детей молоком? ');
read(otvet);
if otvet = 'да' then begin
  ... { вариант 1 }
end
else begin
  ... { вариант 2 }
end;
```

Конечно, вместо многоточий должны быть добавлены команды, которые нужно выполнить в том или другом случае.

Обратите внимание, что символьные строки можно сравнивать с помощью оператора «=» так же, как и числа.

Разберём дальнейшие действия системы при первом ответе «да». Во-первых, нужно вывести первый результат: «Млекопитающее». Во-вторых, в зависимости от ответа на второй вопрос надо сообщить второй результат:

```
  | вариант 1
вывод 'Млекопитающее.', нс
вывод 'Ест мясо? '
ввод ответ
если ответ='да' то
  вывод 'Хищник.', нс
иначе
  вывод 'Не знаю.', нс
все
```

```
  { вариант 1 }
writeln('Млекопитающее.');
write('Ест мясо? ');
read(otvet);
if otvet='да' then
  writeln('Хищник.')
else
  writeln('Не знаю.');
```

Вторую ветвь самого внешнего условного оператора (вариант 2) вы можете написать самостоятельно.

Обратите внимание, что условие `ответ='да'` сработает только тогда, когда пользователь введёт ответ именно так, всеми строчными буквами. Если он наберёт «Да» (с прописной буквы), программа примет это как ответ «нет». Чтобы решить эту проблему, нужно использовать сложное условие:

```
если ответ='да' или ответ='Да' то ...
```

или на Паскале:

```
if (otvet='да') or  (otvet='Да') then begin
```

Итак, теперь вы умеете использовать переменные ещё одного типа — **символьные строки**.

Выводы

- Условный оператор служит для организации ветвления — выбора одного из двух вариантов действий. Для выбора более чем из двух вариантов нужно использовать несколько условных операторов.
- Условный оператор в полной форме содержит: 1) условие, 2) список команд, которые нужно выполнить, если условие истинно, 3) список команд, которые нужно выполнить, если условие ложно.

- Если условный оператор записан в неполной форме, то при ложном условии никаких действий не выполняется.
- Условный оператор в полной форме можно заменить на два условных оператора в неполной форме; условия в них должны быть взаимно обратными: если одно из них истинно, то другое должно быть ложно.
- Внутри каждой ветви условного оператора можно использовать любые операторы языка программирования, в том числе и вложенные условные операторы.
- В сложных условиях для объединения нескольких условий используются логические операции «И», «ИЛИ» и «НЕ».
- Операция «И» означает одновременное выполнение двух или нескольких условий.
- Операция «ИЛИ» означает выполнение хотя бы одного из двух или нескольких условий.
- Операция «НЕ» означает обратное условие (противоположное исходному).
- Логическая переменная — это переменная, которая может принимать только логические значения: «истина» и «ложь».
- Экспертная система — это компьютерная программа, задача которой — заменить человека-эксперта при выработке рекомендаций для принятия решений в сложной ситуации.

Интеллект-карта

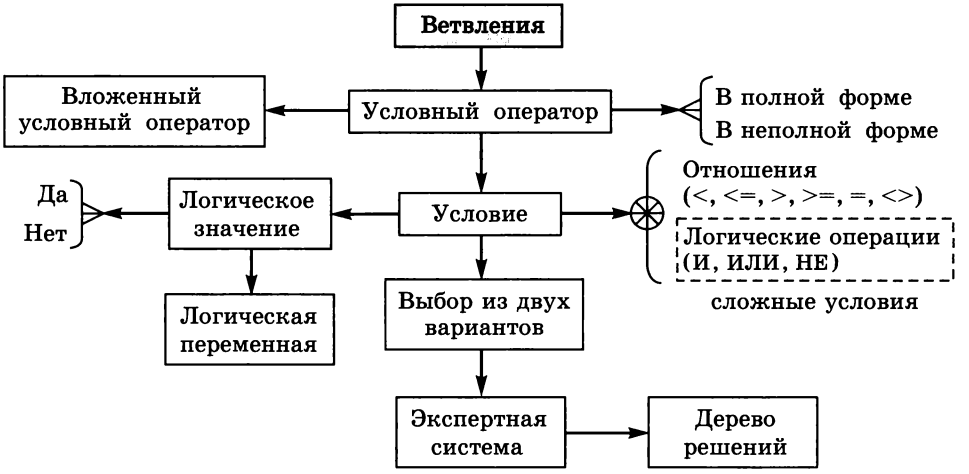


Рис. 3.4

Вопросы и задания

1. Какие задачи невозможно решить с помощью линейных алгоритмов?
2. Как вы думаете, хватит ли линейных алгоритмов и ветвлений для разработки любой программы?
3. Почему нельзя выполнить обмен значений двух переменных в два шага: `a:=b; b:=a`?
4. Как вы думаете, можно ли обойтись только неполной формой условных операторов?
5. Какие отношения вы знаете? Как обозначаются отношения «равно» и «не равно»?
6. Как определяется порядок вычислений в сложном условии? Как его изменить?
7. Как вы думаете, сколько места в памяти занимает каждая логическая переменная?
8. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Оператор выбора»
б) «Экспертные системы»

Практические работы

Выполните практические работы:
№ 11 «Ветвления»;
№ 12 «Сложные условия»;
№ 13 «Логические переменные»;
№ 14 «Экспертная система» (проект).

§ 20
Программирование циклических алгоритмов

Ключевые слова:

- цикл
- счётчик шагов цикла
- цикл с предусловием
- алгоритм Евклида
- цикл с постусловием
- цикл по переменной
- переменная цикла

Как организовать цикл?

Допустим, мы хотим вывести 5 раз на экран слово «привет». Можно, конечно, записать 5 одинаковых команд:

```
вывод 'привет', нс          writeln('привет');
вывод 'привет', нс          writeln('привет');
вывод 'привет', нс          writeln('привет');
вывод 'привет', нс          writeln('привет');
вывод 'привет', нс          writeln('привет');
```

Но что если нужно будет сделать какие-то действия 1000 или 1 000 000 раз?

Как вы знаете из курса 7 класса, для того чтобы выполнить одинаковые действия несколько раз, можно организовать цикл. Простейший цикл, нужный нам в этой задаче, в алгоритмическом языке записывается так:

```
нц 5 раз
  вывод 'привет', нс
кц
```

В Паскале и во многих других языках записать цикл в такой форме нельзя, однако можно легко запрограммировать те же действия немного по-другому. Давайте разберёмся, как можно организовать цикл в любом языке программирования.

Вы знаете, что программа выполняется автоматически. И при этом на каждом шаге нужно знать, сколько раз уже выполнен цикл и сколько ещё осталось выполнить. Для этого необходимо использовать ячейку памяти (переменную). В ней можно, например, запоминать количество завершённых шагов цикла. Такая переменная часто называется **счётчиком**.

Какого типа должна быть переменная-счётчик? Как её нужно объявить? ?

Сначала в переменную-счётчик записывают ноль (ни одного шага не сделано), а после каждого шага цикла увеличивают значение на единицу:

```
счётчик:=0                  count:=0;
нц пока счётчик<5           while count<5 do begin
  вывод 'привет', нс          writeln('привет');
  счётчик:=счётчик+1          count:=count+1
кц                          end;
```

В программе на алгоритмическом языке вам должно быть всё знакомо: **нц** означает начало цикла, а **кц** — конец цикла.

Найдите и запишите в тетрадь перевод английских слов *while*, *do*, *count*.

Возможен и другой вариант: сразу записать в счётчик нужное количество шагов, и после каждого шага цикла уменьшать счётчик на 1. Тогда цикл должен закончиться при нулевом значении счётчика.

Запишите в тетрадь цикл со счётчиком, который уменьшается от нужного значения до нуля.

Этот вариант несколько лучше, чем предыдущий, поскольку счётчик сравнивается с нулём, а такое сравнение выполняется в процессоре автоматически.

Циклы с предусловием

Цикл, в котором проверка условия выполняется при входе (перед выполнением очередного шага), называется **циклом с предусловием**, т. е. циклом с предварительной проверкой условия. Перед тем как начать выполнение цикла, мы проверяем, нужно ли это делать вообще. Это можно сравнить с такой ситуацией: перед тем как прыгнуть в бассейн, нужно проверить, есть ли в нём вода.

Все циклы, записанные в начале параграфа, — это циклы с предусловием. У них есть два важных свойства:

- условие проверяется при входе в цикл, поэтому цикл не выполнится ни разу, если условие в самом начале ложно;
- как только нарушается условие в заголовке цикла, работа цикла заканчивается.

Рассмотрим ещё одну задачу, которая решается с помощью цикла с условием. Требуется ввести с клавиатуры натуральное число и найти сумму цифр его десятичной записи. Например, если ввели число 123, программа должна вывести сумму $1 + 2 + 3 = 6$.

Сначала составим алгоритм решения это задачи. Предположим, что число записано в переменной `N`. Нам нужно как-то разбить число на отдельные цифры.

Запишите в тетрадь команды, с помощью которых можно:

1) записать в переменную `d` последнюю цифру числа, находящегося в переменной `N`;
2) отбросить последнюю цифру числа, находящегося в переменной `N`;
3) добавить значение переменной `d` к неизвестному значению, находящемуся в переменной `s`.

Остаток от деления числа на 10 равен последней цифре десятичной записи числа:

```
d:=mod(N, 10)                    d:=N mod 10;
```

Эту цифру числа нужно добавить к сумме всех цифр, которые мы уже обработали раньше. Сумма цифр — целое число, поэтому будем хранить её в целой переменной sum:

```
цел sum                          var sum: integer;
sum:=0                           sum:=0;
```

В самом начале (пока ни одну цифру мы не обработали) значение этой переменной равно нулю.

Для того чтобы добавить к предыдущей сумме новую цифру, нужно заменить значение переменной sum на sum + d, т. е. выполнить присваивание:

```
sum:=sum+d                       sum:=sum+d;
```

Для того чтобы затем удалить последнюю цифру числа N, разделим N на 10 (основание системы счисления):

```
N:=div(N, 10)                    N:=N div 10;
```

Эти три операции — выделение последней цифры числа, увеличение суммы и удаление последней цифры — нужно выполнять несколько раз, пока все цифры не будут обработаны (и удалены!) и в переменной N не останется ноль:

```
цел N, d, sum                    var N, d, sum: integer;
вывод 'Введите число: '          begin
ввод N                             write('Введите число: ');
sum:=0                             read(N);
                                   sum:=0;
нц пока N<>0                       while N<>0 do begin
  d:=mod(N, 10)                      d:=N mod 10;
  sum:=sum+d                         sum:=sum+d;
  N:=div(N, 10)                      N:=N div 10
кц                                 end;
вывод 'Сумма цифр ', sum           writeln('Сумма цифр ', sum)
                                 end.
```

Выполните ручную прокрутку программы при N = 123. Определите итоговое значение переменной sum.

Для введённого числа 123 программа должна выдать ответ 6 (последнее значение переменной *sum*). Это правильный ответ.

В отличие от предыдущего примера здесь количество шагов цикла заранее неизвестно, оно определяется введённым числом.

Сколько раз выполнится цикл, если ввести однозначное число? Двузначное? *K*-значное?

Какова может быть сумма цифр двузначного числа? Трёхзначного? *K*-значного?

Докажем, что эта программа не зациклится, т. е. не будет работать бесконечно. Цикл завершается, когда переменная `N` становится равна нулю, поэтому нужно доказать, что это обязательно случится. По условию заданное число — натуральное, на каждом шаге цикла оно делится на 10 (остаток отбрасывается), поэтому значение переменной `N` каждый раз уменьшается. В результате после очередного уменьшения оно обязательно станет равно нулю.

Алгоритм Евклида

Мы уже знакомы с алгоритмом Евклида, который позволяет найти наибольший общий делитель (НОД) двух натуральных чисел.

Алгоритм Евклида для натуральных чисел: заменять большее из двух заданных чисел на их разность до тех пор, пока они не станут равны. Полученное число и есть их НОД.

Этот вариант алгоритма работает довольно медленно, если одно из чисел значительно меньше другого, например для пары чисел 2 и 2014. Значительно быстрее выполняется улучшенный (модифицированный) алгоритм Евклида.

Модифицированный алгоритм Евклида для натуральных чисел: заменять бóльшее из двух заданных чисел на остаток от деления бóльшего на меньшее, пока этот остаток не станет равен нулю. Тогда второе число и есть их НОД.

Мы видим, что здесь тоже нужно выполнять некоторые операции несколько раз, причём сколько раз — заранее неизвестно. Но нас выручит цикл с условием, ведь мы знаем, когда нужно остановиться — когда какое-нибудь из двух чисел станет равно нулю.

Запишите условие, которое означает «одно из значений переменных a или b равно нулю». После этого запишите обратное условие.

Теперь мы готовы записать цикл:

```
нц пока a<>0 и b<>0
  если a>b то
    a:=mod(a, b)
  иначе
    b:=mod(b, a)
  все
кц
```

```
while (a<>0) and
      (b<>0) do
  if a>b then
    a:=a mod b
  else
    b:=b mod a;
```

Обратите внимание, что цикл содержит только один условный оператор, поэтому слова **begin** и **end**, ограничивающие тело цикла в программе на Паскале, можно не писать.

Остаётся вывести результат — ненулевое значение переменной a или b.

Запишите условный оператор, который выводит результат, проверяя одну из переменных на равенство нулю.

Из двух переменных, a и b, одна равна нулю, а вторая — не ноль. Запишите арифметическое выражение, которое всегда равно второй (ненулевой) переменной.

Количество шагов такого цикла заранее неизвестно и зависит от исходных данных. Дополним программу так, чтобы она считала ещё и количество сделанных шагов цикла. Для этого нужно ввести переменную-счётчик целого типа. Перед началом цикла счётчик обнуляется (в него записывается ноль), и на каждом шаге цикла значение счётчика увеличивается на единицу:

```
счётчик:=0
нц пока a<>0 и b<>0
  ...
  счётчик:=счётчик+1
кц
вывод a+b, нс
вывод 'Шагов: ', счётчик
```

```
count:=0;
while (a<>0) and (b<>0) do
begin
  ...
  count:=count+1
end
writeln(a+b);
writeln('Шагов: ', count);
```

Вместо многоточия нужно вставить условный оператор, как и в предыдущем варианте программы. В цикле теперь находятся два оператора, поэтому в программе на Паскале нужно использовать составной оператор, ограниченный служебными словами **begin** и **end**.

Циклы в других языках программирования

Алгоритм Евклида на языках *Python* и *C* записывается очень похоже:

```
while a!=0 and b!=0:
  if a>b:
    a=a%b
  else:
    b=b%a
```

```
while (a!=0 && b!=0) {
  if (a>b)
    a=a%b;
  else
    b=b%a;
  }
```

В языке *Python* тело цикла (так же, как и тело каждой части условного оператора) обязательно записывается с отступом вправо, а в языке *C* оно ограничивается фигурными скобками. Если в теле цикла записан всего один оператор, фигурные скобки можно не ставить.

Исследуйте эти фрагменты программы и определите, как записываются в языках *Python* и *C* отношение «не равно», операция взятия остатка от деления, логическая операция И.

Обработка потока данных

На вход программы поступает **поток данных** — последовательность целых чисел, которая заканчивается нулём. Требуется найти сумму элементов этой последовательности.

В этой задаче не нужно сохранять все данные в памяти, мы можем добавлять их к сумме по одному. Объявим две целых переменных: в переменной `x` будем хранить последнее введённое число, а в переменной `sum` — накапливать сумму.

Запишите в тетради объявление двух целочисленных переменных — `x` и `sum`.

Какое начальное значение нужно присвоить переменной `sum`?

Как добавить к неизвестному значению переменной `sum` значение переменной `x`?

Сначала запишем основной цикл программы на алгоритмическом языке, «скрыв» шаги алгоритма в комментариях:

```
нц пока x<>0
  | добавить x к сумме
  | прочитать следующее число
кц
```

Однако перед таким циклом нужно прочитать первое число, иначе неясно, откуда возьмётся значение `x` при первой проверке условия. В итоге получается такая программа:

```
sum:=0
ввод x
нц пока x<>0
  sum:=sum+x
  ввод x
кц
вывод 'Сумма ', sum
```

```
sum:=0;
read(x);
while x<>0 do begin
  sum:=sum+x;
  read(x)
end;
writeln('Сумма ', sum);
```

Как нужно изменить программу для того, чтобы она вычисляла сумму только положительных чисел?

Циклы с постусловием

Во многих языках программирования существует **цикл с постусловием**, в котором условие проверяется не до, а после завершения очередного шага цикла. Это полезно в том случае, когда нужно обязательно выполнить цикл хотя бы один раз. Например, пользователь должен ввести с клавиатуры положительное число. Для того чтобы защитить программу от неверных входных данных, можно использовать цикл с постусловием:

```
нц
  вывод 'Введите N>0: '
  ввод N
кц при N > 0
```

```
repeat
  write('Введите N>0: ');
  read(N);
until N > 0;
```

Этот цикл закончится тогда, когда станет истинным условие `N > 0` в последней строке, т. е. тогда, когда пользователь введет допустимое значение. На блок-схеме этого алгоритма видно, что проверка условия выполняется после завершения очередного шага цикла (рис. 3.5).

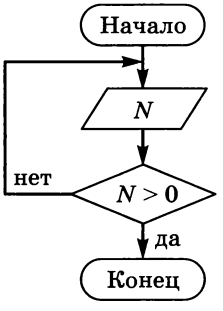


Рис. 3.5

Обратите внимание на особенности этого вида цикла:

- при входе в цикл условие не проверяется, поэтому цикл всегда выполняется хотя бы один раз;
- в последней строке указывают условие окончания цикла (а не условие его продолжения, как в цикле с предусловием).

Циклы по переменной

В информатике важную роль играют степени числа 2 (2, 4, 8, 16 и т. д.). Давайте выведем на экран все степени двойки от 2^1 до 2^{10}. Для решения этой задачи мы уже можем написать такую программу, использующую цикл с условием:

```
k:=1
N:=2
нц пока k<=10
  вывод N, нс
  N:=N*2
  k:=k+1;
кц
```

```
k:=1;
N:=2;
while k<=10 do begin
  writeln(N);
  N:=N*2;
  k:=k+1;
end;
```

Вы наверняка заметили, что переменная k используется трижды (см. выделенные блоки): в операторе присваивания начального значения, в условии цикла и в теле цикла (увеличение на 1).

Чтобы собрать все действия с ней в один оператор, во многие языки программирования введен особый вид цикла — **цикл по переменной** (или **цикл с параметром**). В заголовке этого цикла задаются начальное и конечное значения этой переменной (она называется **переменной цикла**), а шаг её изменения по умолчанию равен 1:

```
N:=2
нц для k от 1 до 10
  вывод N, нс
  N:=N*2
кц
```

```
N:=2;
for k:=1 to 10 do begin
  writeln(N)
  N:=N*2;
end;
```

Заголовок цикла в программе на алгоритмическом языке говорит о том, что тело цикла нужно выполнить для всех целых значений k от 1 до 10 включительно. Аналогичный заголовок на Паскале — это просто перевод тех же слов на английский язык.

Выясните, как переводятся на русский язык английские слова *for* и *to*.

Запишите циклы, с помощью которых можно вывести на экран:
а) целые числа от a до b ($a \le b$);
б) квадраты целых чисел от a до b ($a \le b$).

В отличие от цикла с условием переменная в таком цикле (в нашем случае — k) может быть только целочисленной[1)].

[1)] В Паскале переменная цикла может быть любого *перечислимого типа* (когда для каждого значения можно назвать предыдущее и следующее), например логического (boolean).

Рассмотрим ещё одну задачу — найдём сумму чисел от 1 до 1000. Для накопления суммы будем использовать переменную (назовём её sum). В цикле другая переменная (скажем, i) изменяется от 1 до 1000, и на каждом шаге этого цикла к сумме добавляется очередное число:

```
цел sum, i
sum:=0
нц для i от 1 до 1000
  sum:=sum+i
кц
```

```
var sum, i: integer;
...
sum:=0;
for i:=1 to 1000 do
  sum:=sum+i;
```

Запишите циклы, с помощью которых можно вычислить:
а) сумму целых чисел от a до b ($a \le b$);
б) сумму квадратов целых чисел от a до b ($a \le b$).

С каждым шагом цикла переменная цикла может не только увеличиваться, но и уменьшаться на 1. Для этого в алгоритмическом языке добавляется параметр **шаг**, а в Паскале служебное слово **to** заменяется на **downto** («вниз до...»). Следующая программа выводит квадраты натуральных чисел от 10 до 1 в порядке убывания:

```
нц для k от 10 до 1 шаг -1
  вывод k*k, нс
кц
```

```
for k:=10 downto 1 do
  writeln(k*k);
```

В алгоритмическом языке шаг изменения переменной цикла может быть любым целым числом, а в Паскале — только 1 или (–1).

Циклы по переменной в других языках программирования

Суммирование всех чисел от 1 до 1000 на языках *Python* и *C* выглядит так:

```
sum=0
for i in range(1, 1001):
  sum+=i
```

```
int sum, i;
sum=0;
for (i=1; i<=1000; i++)
  sum+=i;
```

В языке *Python* с помощью вызова стандартной функции `range` задаётся диапазон изменения переменной *i* от 1 до 1000, причём последнее указанное число (1001) в этот диапазон не входит. В языке *C* в заголовке определяются начальное и конечное значения переменной цикла, а запись `i++` означает «увеличить значение *i* на единицу».

Сравните эти фрагменты с программами на алгоритмическом языке и на Паскале и выясните, что значит запись `sum += i`.

Измените программы на языках *Python* и *C* так, чтобы они вычисляли сумму квадратов натуральных чисел от 5 до 25.

Выводы

- С помощью циклов в программе можно выполнять повторяющиеся действия.
- Различают два вида циклов: циклы с условием и циклы по переменной.
- Цикл с предусловием выполняется до тех пор, пока некоторое условие (условие продолжения работы цикла) не станет ложным. Если это условие никогда не станет ложным, программа зацикливается.
- В некоторых языках программирования есть циклы, запись которых содержит не условие продолжения, а условие выхода из цикла.
- Проверка условия может происходить перед выполнением очередного шага цикла (в циклах с предусловием) или после него (в циклах с постусловием). Цикл с предусловием может не выполняться ни разу, а цикл с постусловием всегда выполняется хотя бы один раз.
- Цикл по переменной применяют тогда, тогда количество шагов цикла заранее известно или может быть вычислено до начала цикла. В заголовке цикла по переменной указывают начальное значение, конечное значение и шаг изменения переменной цикла.
- Модифицированный алгоритм Евклида для вычисления НОД двух натуральных чисел: заменять большее из чисел на остаток от деления большего на меньшее, пока этот остаток не станет равен нулю. Тогда второе число и есть их НОД.

Интеллект-карта

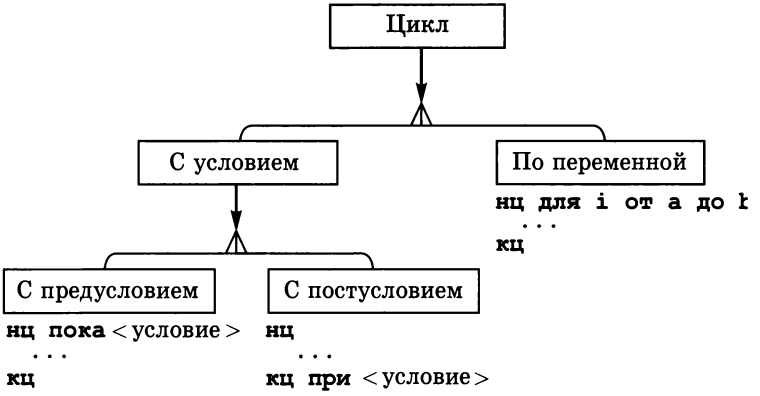


Рис. 3.6

Вопросы и задания

1. В каком случае программа, содержащая цикл с предусловием, может зациклиться?
2. В каком случае цикл с предусловием не выполняется ни разу?
3. В каких ситуациях, на ваш взгляд, лучше использовать цикл с постусловием?
4. В каком случае цикл по переменной не выполняется ни разу?
5. Может ли цикл по переменной работать бесконечно?
6. Сравните цикл по переменной и цикл с условием. Какие преимущества и недостатки есть у каждого из них?
7. Верно ли, что любой цикл по переменной можно заменить циклом с условием? Верно ли обратное утверждение?
8. В каком случае можно заменить цикл с условием на цикл по переменной?
9. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

«Три вида циклов: сравнение»

Практические работы

Выполните практические работы:

№ 15 «Циклы с условием»;
№ 16 «Алгоритм Евклида»;
№ 17 «Обработка потока данных»;
№ 18 «Циклы с постусловием»;
№ 19 «Циклы по переменной».

§ 21
Массивы

Ключевые слова:

- массив
- индекс элемента
- значение элемента
- константа
- заполнение массива
- вывод массива
- ввод массива

Что такое массив?

В программах, с которыми мы работали раньше, было всего несколько переменных. Каждой из них мы давали своё имя, и никаких сложностей при этом не возникало.

Объёмы данных, которые обрабатывают современные компьютеры, огромны: количество значений измеряется миллионами и миллиардами. Если каждую из этих переменных называть своим именем, очень легко запутаться, и работать с ними очень неудобно.

В программе есть переменные `a1`, `a2`, `a3`, `a4` и `a5`. Запишите оператор, который вычисляет их сумму в переменной `s`.

Как решить предыдущую задачу, если в одном операторе разрешается выполнять только одну операцию сложения?

Допустим, мы хотим сложить значения 1000 ячеек с именами $a1$, $a2$, ..., $a1000$. Для этого нужно будет написать очень длинный оператор присваивания:

```
sum:=a1+a2+...+a1000;
```

Учтите, что компьютер не понимает многоточий, поэтому нам придётся перечислить все 1000 имён переменных.

Какая проблема возникнет при решении этой задачи, если количество данных заранее неизвестно (например, передаётся по компьютерной сети)?

Для того чтобы было удобно работать с большим количеством данных, обычно дают общее имя группе переменных, которая называется массивом.

Массив — это группа переменных одного типа, расположенных в памяти друг за другом и имеющих общее имя.

Чтобы использовать массив, надо его объявить — присвоить ему имя, определить тип входящих в массив переменных (**элементов массива**) и их количество. По этим сведениям компьютер вычислит, сколько места требуется для хранения массива, и выделит в памяти нужное число ячеек.

Имена (идентификаторы) массивов строятся по тем же правилам, что и имена переменных.

В алгоритмическом языке массивы называются **таблицами**. При их объявлении к названию типа данных добавляются символы **таб**:

```
целтаб A[1:5]
вещтаб V[0:5]
```

В квадратных скобках через двоеточие записывают границы *индексов* — номеров элементов массива.

Индекс — это значение, которое указывает на конкретный элемент массива.

Массив `A` в нашем примере — это массив целых значений, элементы имеют индексы (номера) от 1 до 5. Массив вещественных значений `V` содержит 6 элементов с индексами от нуля[1] до 5. В алгоритмическом языке объявлять массивы (как и переменные) можно в любом месте программы.

В языке Паскаль массивы объявляются в блоке объявления переменных (выше служебного слова **begin**). При объявлении массива после имени записывают служебное слово **array**, затем в квадратных скобках — минимальный и максимальный индексы, разделяя их двумя точками:

```
var A: array[1..5] of integer;
    V: array[0..5] of real;
```

Переведите на русский язык английское слово *array*.

Для того чтобы обратиться к элементу массива (прочитать или изменить его значение), нужно записать имя массива и в квадратных скобках — индекс нужного элемента, например `A[3]` (рис. 3.7).

[1] Нумерация с нуля часто используется в языках программирования (например, в языках *C*, *Java*, *Javascript*, *Python* и др.).

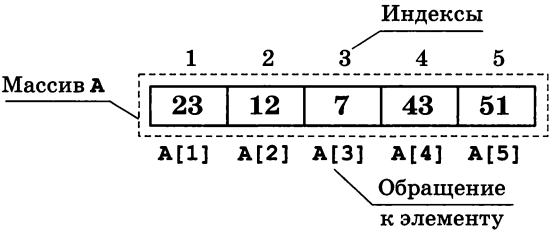


Рис. 3.7

Как вы думаете, что делают эти операторы?

а) вывод A[3]
```
write(A[3]);
```
б) A[3]:=5
```
A[3]:=5;
```
в) A[1]:=A[2]+2*A[3]
```
A[1]:=A[2]+2*A[3];
```

Индексом может быть, кроме целого числа, также значение целой переменной или арифметического выражения, результат которого — целое число.

Определите, что выведет этот фрагмент программы для массива на рис. 3.7:

```
i:=2;
A[3]:=A[i]+2*A[i-1]+A[2*i]
вывод A[3]+A[5]
```

```
i:=2;
A[3]:=A[i]+2*A[i-1]+A[2*i];
write(A[3]+A[5]);
```

Найдите ошибки в этом фрагменте программы:

```
целтаб A[1:5]
цел x
...
x:=2
вывод A[x-3]
A[x+4]:=A[x-1]+A[2*x]
```

```
var A: array[1..5]
       of integer;
    x: integer;
...
x:=2;
write(A[x-3]);
A[x+4]:=A[x-1]+A[2*x];
```

В чём заключаются ошибки?

Выход за границы массива — это обращение к элементу с индексом, который не существует в массиве.

При выходе за границы массива программа обычно завершается аварийно.

Индексом может быть даже значение элемента массива. Например, запись *A*[*A*[1]] означает, что нужно взять значение *A*[1] и использовать его как индекс нужного элемента.

Определите, что выведет этот фрагмент программы:

```
вывод A[1]                    write(A[1]);
вывод A[A[1]]                 write(A[A[1]]);
вывод A[A[A[1]]]              write(A[A[A[1]]]);
вывод A[A[A[A[1]]]]           write(A[A[A[A[1]]]]);
вывод A[A[A[A[A[1]]]]]        write(A[A[A[A[A[1]]]]]);
```

для массива

	1	2	3	4	5
A	5	4	1	3	2

Далее мы будем во всех программах использовать привычную для человека нумерацию с единицы, считая, что массив *A* объявлен так:

```
цел N=10                      const N=10;
целтаб A[1:N]                 var A: array[1..N] of integer;
```

Здесь размер массива (количество элементов) обозначен как N. В программе, как правило, размер массива встречается во многих командах, и при его изменении нужно исправить число только в одном месте программы. В нашем примере на алгоритмическом языке N — это переменная, значение которой задано до того, как объявлен массив. В программе на Паскале размер массива объявлен как **константа** (неизменяемая величина, имеющая имя) с помощью служебного слова **const**.

Иногда нужно вводить размер массива с клавиатуры. В этом случае в программе на Паскале нужно заранее выделить в памяти массив наибольшего размера, соответствующего условию задачи. В алгоритмическом языке можно объявить массив тогда, когда его размер уже стал известен.

Перебор элементов массива

Перебор элементов состоит в том, что мы в цикле просматриваем все элементы массива и, если нужно, выполняем с каждым из них некоторую операцию. Для этого удобнее всего использовать цикл по переменной, которая изменяется от минимального до максимального индекса. Для массива, элементы которого имеют индексы от 1 до N, цикл выглядит так:

```
нц для i от 1 до N
  ...
кц
```

```
for i:=1 to N do begin
  ...
end;
```

Здесь вместо многоточия можно добавлять операторы, работающие с элементом A[i].

Какие значения будет принимать переменная i при выполнении этого цикла?

Мы видим, что благодаря использованию массива нам достаточно описать, что делать с одним элементом, а затем поместить эти действия внутрь цикла, перебирающего значения индексов. Если бы мы применяли простые переменные, то нам пришлось бы описывать необходимые действия для каждого элемента (правда, при этом цикл бы не понадобился).

Выполните ручную прокрутку фрагмента программы:

```
нц для i от 1 до N
  A[i]:=i
кц
```

```
for i:=1 to N do
  A[i]:=i;
```

Какие значения будут записаны в массив?

Запишите фрагмент программы, который заполнит массив нулями.

Заполним массив первыми N натуральными числами в обратном порядке: в первый элемент массива должно быть записано число N, во второй — число N-1, а в последний — единица.

Сначала запишем цикл в развёрнутом виде: операторы, которые должны быть выполнены:

```
A[1]:=N
A[2]:=N-1
...
A[N]:=1
```

```
A[1]:=N;
A[2]:=N-1;
...
A[N]:=1;
```

Теперь запишем цикл, в котором значение, присваиваемое очередному элементу, пока обозначим через X:

```
нц для i от 1 до N
  A[i]:=X
кц
```

```
for i:=1 to N do
  A[i]:=X;
```

Однако не всё так просто: величина X должна изменяться при переходе к следующему элементу.

Определите, как меняется X: чему равно начальное значение этой переменной, как она изменяется с каждым шагом.

Можно записать цикл так:

```
X:=N
нц для i от 1 до N
  A[i]:=X
  X:=X-1
кц
```

```
X:=N;
for i:=1 to N do begin
  A[i]:=X;
  X:=X-1
end;
```

А можно его значительно упростить, заметив, что при увеличении номера элемента i на единицу значение X уменьшается, причём тоже на единицу. Поэтому сумма i + X остаётся постоянной! Её можно вычислить, заметив, что для первого элемента она равна 1 + N.

Выразите X из уравнения i + X = 1 + N.

В элемент с номером i записывается значение N+1−i, поэтому цикл можно записать так:

```
нц для i от 1 до N
  A[i]:=N+1-i
кц
```

```
for i:=1 to N do
  A[i]:=N+1-i;
```

Теперь предположим, что массив заполнен, и попробуем увеличить все его элементы на единицу. Этот значит, что нужно заменить значение элемента A[i] на A[i] + 1:

```
нц для i от 1 до N
  A[i]:=A[i]+1
кц
```

```
for i:=1 to N do
  A[i]:=A[i]+1;
```

Определите, какие значения окажутся в массиве

	1	2	3	4	5
A	5	4	3	2	1

после выполнения фрагмента программы:

```
нц для i от 1 до N
  A[i]:=A[i]+i
кц
```

```
for i:=1 to N do
  A[i]:=A[i]+i;
```

Запишите фрагмент программы, который умножает все элементы массива на 2.

Запишите фрагмент программы, который умножает первый элемент массива на 1, второй — на 2, третий — на 3 и т. д.

Вывод массива

Массив — это набор элементов, поэтому в большинстве языков программирования нельзя вывести массив одной командой. Для этого используют цикл, в котором на каждом шаге выводится один элемент:

```
нц для i от 1 до N
  вывод A[i], ' '
кц
```

```
for i:=1 to N do
  write(A[i], ' ');
```

Обратите внимание, что здесь элементы выводятся в строку, и чтобы они не слились в одно длинное число, после каждого выводится пробел. Можно также использовать и **форматный вывод**, указав, сколько знаков нужно отвести на каждое значение:

```
нц для i от 1 до N
  вывод A[i]:4
кц
```

```
for i:=1 to N do
  write(A[i]:4);
```

Приведите пример массива, для которого такой форматный вывод даст неправильный результат.

Ввод массива с клавиатуры

Иногда небольшие массивы вводятся с клавиатуры. В простейшем случае мы просто строим цикл, который выполняет оператор ввода для каждого элемента массива:

```
нц для i от 1 до N
  ввод A[i]
кц
```

```
for i:=1 to N do
  read(A[i]);
```

При этом пользователь вводит данные «вслепую», т. е. программа не подсказывает ему, значение какого элемента вводится в данный момент. Значительно удобнее, если перед вводом появляется сообщение с подсказкой:

```
нц для i от 1 до N
  вывод 'A[',i,']='
  ввод A[i]
кц
```

```
for i:=1 to N do begin
  write('A[',i,']=');
  read(A[i])
end;
```

В этом примере перед вводом очередного элемента массива на экран выводится подсказка. Например, после ввода второго элемента будет выведено `A[3]=` — программа будет ждать ввода третьего элемента.

Заполнение массива случайными числами

В учебных примерах массивы часто заполняют случайными числами, например так:

```
нц для i от 1 до N
  A[i]:=irand(20,100)
  вывод A[i], ' '
кц
```

```
for i:=1 to N do begin
  A[i]:=20+random(81);
  write(A[i], ' ')
end;
```

Элементы массива сразу же выводятся на экран (в одну строку через пробелы).

Определите, в каком отрезке содержатся все элементы этого массива.

Запишите цикл, который заполняет массив случайными числами: а) на отрезке [100; 150]; б) на отрезке [–10; 10].

Массивы в других языках программирования

Заполнение массива первыми N натуральными числами на языках *Python* и *C* выглядит так:

```
A=[0]*N
for i in range(N):
  A[i]=i+1
```

```
int A[N], i;
for (i=0; i<N; i++)
  A[i]=i+1;
```

В этих (и во многих других) языках нумерация элементов массива начинается с нуля: первый элемент массива имеет индекс 0, а N-й — индекс N – 1. Функция `range(N)` строит последовательность чисел от 0 до N – 1.

Почему в цикле в элемент с индексом `i` записывается значение `i + 1`, а не `i`?

Измените программы на языках *Python* и *C* так, чтобы массив заполнялся теми же числами в обратном порядке.

Выводы

- Массив — это группа переменных одного типа, расположенных в памяти друг за другом и имеющих общее имя. Массивы используют для того, чтобы было удобно работать с большим количеством данных.

- Индекс элемента массива — это значение, которое указывает на конкретный элемент массива.
- При обращении к элементу массива индекс указывают в квадратных скобках. Это может быть число, имя переменной целого типа или арифметическое выражение, результат которого — целое число.
- Перебор элементов массива — это выполнение какой-то операции с каждым элементом. Для этого удобно использовать цикл по переменной, которая изменяется от минимального до максимального значения индекса.
- Массив вводится и выводится поэлементно, как правило, с помощью цикла.

Интеллект-карта

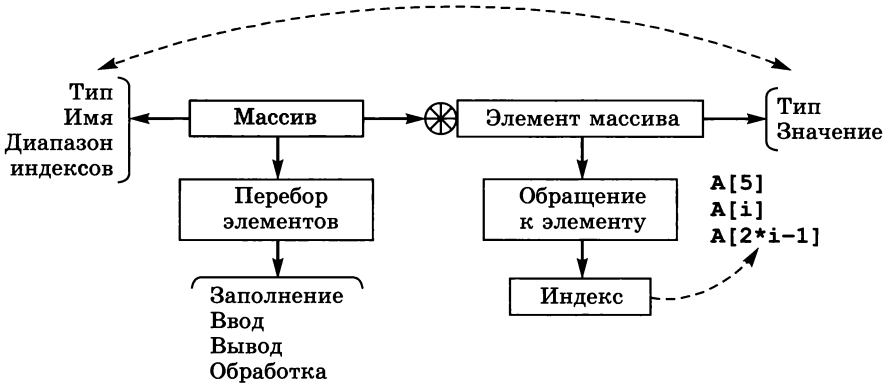


Рис. 3.8

Вопросы и задания

1. Как вы думаете, почему элементы массива размещают в памяти рядом?
2. Объясните разницу между терминами «индекс элемента массива» и «значение элемента массива».
3. Некоторые языки программирования разрешают обращаться к элементам за пределами массива (программа не завершается аварийно). Обсудите достоинства и недостатки такого решения.
4. Почему размер массива лучше вводить как константу, а не как число?
5. Массив из 22 элементов требуется заполнить случайными числами на отрезке [10; 30]. Будут ли в массиве одинаковые элементы? Почему?
6. Выполните по указанию учителя задания в рабочей тетради.

Практические работы

Выполните практические работы:
№ 20 «Заполнение массивов»;
№ 21 «Перебор элементов массива».

§ 22
Алгоритмы обработки массивов

Ключевые слова:

- сумма значений элементов массива
- подсчёт элементов по условию
- максимальный элемент

Сумма значений элементов массива

Представьте себе, что в массиве записаны зарплаты сотрудников фирмы и требуется найти общую сумму, которая будет им выплачена. Для этого нужно сложить все числа, которые находятся в массиве.

Для того чтобы накапливать сумму, нужно ввести переменную, назовём её sum.

Какое начальное значение нужно записать в переменную sum?

В программе есть переменные sum и x. Запишите оператор, с помощью которого можно добавить значение x к значению sum.

Для решения задачи нужно выполнить перебор элементов массива в цикле. На каждом шаге цикла к значению sum добавляется значение очередного элемента массива.

Будем считать, что массив уже заполнен. Тогда сумму его элементов можно найти так:

```
sum:=0
нц для i от 1 до N
  sum:=sum+A[i]
кц
вывод sum
```

```
sum:=0;
for i:=1 to N do
  sum:=sum+A[i];
write(sum);
```

Покажем, как работает этот алгоритм для массива A (рис. 3.9).

	1	2	3	4	5
A	5	2	8	3	1

Рис. 3.9

Выполним «ручную прокрутку» программы. Запишем в таблице выполняемые команды (операторы) и изменение всех переменных (сам массив A при этом не меняется):

	Оператор	i	sum
1	sum:=0		0
2	i:=1	1	
3	sum:=sum+A[1]		5
4	i:=i+1	2	
5	sum:=sum+A[2]		7
6	i:=i+1	3	
7	sum:=sum+A[3]		15
8	i:=i+1	4	
9	sum:=sum+A[4]		18
10	i:=i+1	5	
11	sum:=sum+A[5]		19

Фоном выделены команды, которые выполняются автоматически в цикле по переменной: в строке 2 переменной i присваивается начальное значение, а в строках 4, 6, 8 и 10 после выполнения очередного шага цикла значение этой переменной увеличивается на единицу.

Для массива на рис. 3.9 выполните ручную прокрутку программы и определите, какое значение будет выведено:

```
sum:=0
нц для i от 1 до N
  если mod(A[i], 2)=0 то
    sum:=sum+A[i]
  все
кц
вывод sum
```

```
sum:=0;
for i:=1 to N do
  if A[i] mod 2=0 then
    sum:=sum+A[i];
write(sum);
```

Измените условие отбора в программе из предыдущего задания так, чтобы при обработке массива на рис. 3.9 в переменной sum получилось число 13.

Напишите циклы, с помощью которых можно найти в переменной *p*:

а) произведение всех элементов массива;

б) произведение положительных элементов массива.

Подумайте, каким должно быть начальное значение переменной p и как оно должно изменяться на каждом шаге цикла.

Подсчёт элементов массива, удовлетворяющих условию

Во многих задачах нужно найти в массиве все элементы, удовлетворяющие заданному условию, и как-то их обработать, например подсчитать их количество. Для подсчёта элементов используется переменная-счётчик, назовём её *count*. Перед началом цикла в счётчик записывается ноль (ни одного нужного элемента не найдено). Если на очередном шаге цикла найден новый элемент, значение счётчика увеличивается на единицу.

Подсчитаем количество чётных элементов массива (элементов с чётными значениями).

Запишите условие, которое означает, что переменная *x* чётная. Предложите два равносильных варианта такого условия.

Условие «элемент A[i] чётный» можно сформулировать иначе: «остаток от деления A[i] на 2 равен нулю»:

```
если mod(A[i], 2)=0 то
  ... | увеличить счётчик
все
```

```
if A[i] mod 2=0 then
  ... { увеличить счётчик }
```

Теперь можно написать полный цикл:

```
count:=0
нц для i от 1 до N
  если mod(A[i], 2)=0 то
    count:=count+1
  все
кц
вывод count
```

```
count:=0;
for i:=1 to N do
  if A[i] mod 2=0 then
    count:=count+1;
write(count);
```

Для массива на рис. 3.9 выполните ручную прокрутку этой программы и определите, какое значение будет выведено.

Теперь усложним задачу. В массиве записан рост каждого члена баскетбольной команды в сантиметрах. Требуется найти средний рост игроков, которые выше 180 см (предполагаем, что хотя бы один такой игрок есть). Средний рост — это среднее арифметическое, т. е. «суммарный рост» интересующих нас игроков (тех, которые выше 180 см), поделённый на их количество.

Найдите ошибку в программе:

```
sum:=0
нц для i от 1 до N
  если A[i]>180 то
    sum:=sum+A[i]
  все
кц
вывод sum/N
```

```
sum:=0;
for i:=1 to N do
  if A[i]>180 then
    sum:=sum+A[i];
write(sum/N);
```

К какому типу ошибок относится эта ошибка?

Для решения задачи нам нужно считать и сумму, и количество элементов массива, значения которых больше 180:

```
count:=0
sum:=0
нц для i от 1 до N
  если A[i]>180 то
    count:=count+1
    sum:=sum+A[i]
  все
кц
вывод sum/count
```

```
count:=0;
sum:=0;
for i:=1 to N do
  if A[i]>180 then begin
    count:=count+1;
    sum:=sum+A[i]
  end;
write(sum/count);
```

Обратите внимание, что в теле условного оператора находятся две команды (увеличение счётчика и увеличение суммы), поэтому в программе на языке Паскаль они заключаются в **операторные скобки** — служебные слова **begin** и **end**.

Поиск максимального элемента в массиве

Представьте себе, что вы по очереди заходите в N комнат, в каждой из которых лежит арбуз. Вес арбузов такой, что вы можете унести только один арбуз. Возвращаться в ту комнату, где вы уже побывали, нельзя. Как выбрать самый большой арбуз?

Итак, вы вошли в первую комнату. По-видимому, нужно забрать лежащий в ней арбуз. Действительно, вдруг он самый большой? А вернуться сюда вы уже не сможете. С этим первым арбузом вы идёте во вторую комнату и сравниваете, какой арбуз больше — тот, который у вас в руках или новый. Если новый больше, берёте его, а старый оставляете во второй комнате. Теперь в любом случае у вас в руках оказывается самый большой арбуз из первых двух комнат. Действуя так же и в остальных комнатах, вы гарантированно выберете самый большой арбуз из всех.

На этой идее основан и поиск максимального элемента в массиве[1]. Для хранения значения максимального элемента выделим

[1] Этот пример хорошо объясняет принцип поиска максимального элемента, но нужно учитывать, что, в отличие от предметов (арбузов), данные не исчезают в том месте, откуда они были скопированы.

в памяти целочисленную переменную M. Будем в цикле просматривать все элементы массива один за другим. Если значение очередного элемента массива больше, чем максимальное из предыдущих (находящееся в переменной M), запомним новое значение максимального элемента в M.

Чего не хватает в этом фрагменте программы?

```
нц для i от 1 до N
  если A[i]>M то
    M:=A[i]
  все
кц
вывод M
```

```
for i:= 1 to N do
  if A[i]>M then
    M:=A[i];
write(M);
```

Остаётся решить, каково должно быть начальное значение M.

Предположим, что вначале переменной M мы будем присваивать значение 0. Всегда ли это будет правильно?

Во-первых, можно записать в переменную M значение, заведомо меньшее, чем значение любого из элементов массива. Например, если в массиве записаны натуральные числа, можно записать в M ноль.

Во-вторых, если содержимое массива неизвестно, можно сразу записать в M значение A[1] (сразу взять первый арбуз), а цикл перебора начать со второго элемента:

```
M:=A[1]
нц для i от 2 до N
  если A[i]>M то
    M:=A[i]
  все
кц
вывод M
```

```
M:=A[1];
for i:=2 to N do
  if A[i]>M then
    M:=A[i];
write(M);
```

Будет ли в этой задаче ошибкой цикл, начинающий перебор с первого элемента?

Викентий решил написать первый оператор в предыдущем фрагменте так:

```
M:=A[N]
```

```
M:=A[N];
```

Закончите программу Викентия.

Кирилл решил написать первый оператор в предыдущем фрагменте так:

```
M:=A[div(N,2)]
```

```
M:=A[N div 2];
```

Закончите программу Кирилла.

Теперь найдём номер максимального элемента. Казалось бы, нужно ввести еще одну переменную nMax для хранения номера, сначала записать в нее 1 (считаем первый элемент максимальным) и затем, когда найдём новый максимальный элемент, запоминать его номер в переменной nMax:

```
M:=A[1]; nMax:=1
нц для i от 2 до N
  если A[i]>M то
    M:=A[i]
    nMax:=i
  все
кц
вывод 'A[', nMax, ']=', M
```

```
M:=A[1]; nMax:=1;
for i:=2 to N do
  if A[i]>M then begin
    M:=A[i];
    nMax:=i
  end;
write('A[', nMax, ']=', M);
```

Однако это не самый лучший вариант. Одна из переменных в этой программе лишняя.

Можно ли, зная значение максимального элемента массива M, сразу (без перебора!) найти его номер? Если да, то как?

Можно ли, зная номер максимального элемента массива nMax, сразу (без перебора!) найти его значение? Если да, то как?

По номеру элемента i можно всегда определить его значение, оно равно A[i]. Поэтому достаточно хранить только номер максимального элемента nMax, тогда его значение равно A[nMax]:

```
nMax:=1
нц для i от 2 до N
  если A[i]>A[nMax] то
    nMax:=i
  все
кц
вывод 'A[', nMax, ']=', A[nMax]
```

```
nMax:=1;
for i:=2 to N do
  if A[i]>A[nMax] then
    nMax:=i;
write('A[', nMax, ']=',
      A[nMax])
```

Более сложная задача — найти максимальное значение не всех элементов массива, а только тех, которые удовлетворяют некоторому условию. Если вернуться к примеру с поиском арбуза: в некоторых комнатах лежат не арбузы, а дыни, но нужно найти именно самый большой арбуз. Это значит, что мы выбираем новый элемент, если он: а) подходит нам по условию отбора и б) больше, чем максимальный, найденный до этого.

Рассмотрим конкретную задачу: найти максимальный из отрицательных элементов массива.

Запишите в тетради для этой задачи условие обновления максимального значения в переменной M.

Самое сложное — определить, каким должно быть начальное значение M.

Предположим, что мы сначала записали в переменную M значение какого-либо элемента массива. Когда такой приём может дать неверный результат?

Выполните ручную прокрутку следующей программы:

```
M:=A[1]
нц для i от 2 до N
  если A[i]<0 и A[i]>M то
    M:=A[i]
  все
кц
вывод M
```

```
M:= A[1];
for i:=2 to N do
  if (A[i]<0) and
     (A[i]>M)
  then
    M:=A[i];
write(M);
```

для двух массивов (рис. 3.10).

а)

	1	2	3	4	5
A	5	−2	8	3	−1

б)

	1	2	3	4	5
A	−5	−2	8	3	−1

Рис. 3.10

Удалось ли найти максимальный отрицательный элемент в первом случае? Во втором?

Итак, если первый элемент массива положительный (нам не подходит!), он оказывается больше всех подходящих элементов, и программа выводит его как результат. Но этот результат неверный! Есть два способа исправить программу.

Первый вариант: можно сначала найти первый отрицательный элемент, записать его в переменную M, а потом перебирать в цикле все оставшиеся.

Второй вариант проще — мы будем заменять значение M в том случае, если очередной элемент A[i] — отрицательный, а значение M — неотрицательное. Например, так:

```
M:=A[1]
нц для i от 2 до N
  если A[i]<0 то
    если M>=0 или A[i]>M то
      M:=A[i]
    все
  все
кц
вывод M
```

```
M:=A[1];
for i:=2 to N do
  if A[i]<0 then
    if (M>=0) or (A[i]>M)
    then
      M:=A[i];
write(M);
```

При каких значениях A[i] и M условия M ≥ 0 и A[i] > M во вложенном условном операторе могут выполниться одновременно?

Выделите все случаи, при которых в этой программе будет изменяться значение M.

Выводы

- Для вычисления суммы элементов массива используется дополнительная переменная, в которой накапливается сумма. Начальное значение этой переменной равно нулю. Для добавления к сумме очередного элемента массива A[i] используют оператор вида

  sum:=sum+A[i]

- При вычислении произведения начальное значение дополнительной переменной должно быть равно 1.
- Для подсчёта количества элементов, удовлетворяющих условию, нужно использовать переменную-счётчик. Начальное значение счётчика должно быть равно нулю. При обнаружении очередного нужного элемента счётчик увеличивается на 1:

  count:=count+1

- При поиске максимального значения в массиве используют вспомогательную переменную, в которой хранится максимальное из всех просмотренных значений. Сначала в эту переменную записывают значение первого элемента массива, а затем просматривают все элементы, начиная со второго.

Интеллект-карта

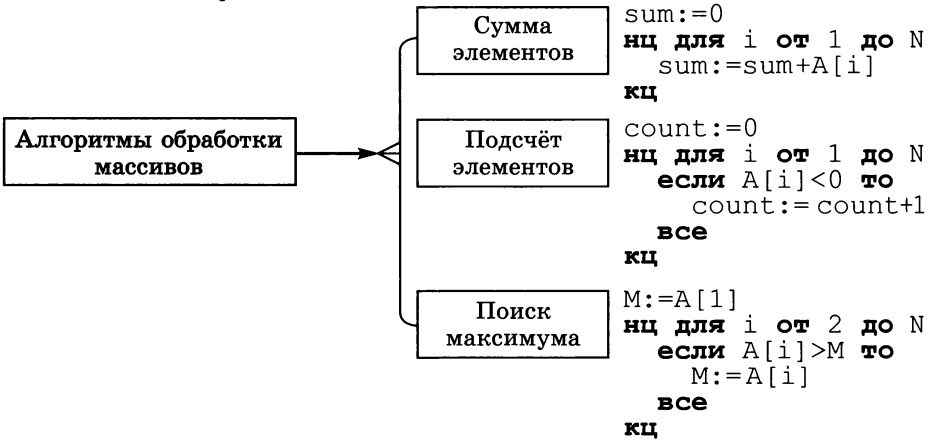


Рис. 3.11

Вопросы и задания

1. Как можно проверить, что число делится одновременно на 7 и на 5? Предложите два способа.
2. Объясните, почему при поиске максимального элемента и его номера не нужно запоминать само значение максимального элемента.
3. Выполните по указанию учителя задания в рабочей тетради.

Практические работы

Выполните практические работы:
№ 22 «Алгоритмы обработки массивов»;
№ 23 «Сумма значений элементов массива»;
№ 24 «Подсчёт элементов массива по условию»;
№ 25 «Поиск максимального элемента».

ЭОР к главе 3 из Единой коллекции цифровых образовательных ресурсов (school-collection.edu.ru)

Назначение и средства программирования
Структура программы на языке Паскаль
Команда присваивания
Числа в памяти компьютера
Полное и неполное ветвление
Реализация ветвления на Паскале
Программа с ветвлением на Паскале
Циклические операторы на Паскале

Глава 4
ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

§ 23
Введение

Ключевые слова:

- электронная таблица
- табличный процессор
- ячейка
- строка
- столбец
- адрес
- диапазон
- формула
- функция

Что такое электронная таблица?

Представьте себе, что нам нужно много раз решать одну и ту же вычислительную задачу при различных исходных данных. Конечно, хочется один раз «научить» компьютер выполнять все нужные действия, а потом подставлять новые исходные данные и сразу получать результат. Для этого придумали **электронные таблицы (табличные процессоры)**. Так называются программы, которые хранят все данные в виде таблиц. Но это не просто таблицы: в ячейках могут храниться формулы, вычисления по которым выполняются автоматически при любом изменении данных.

Электронная таблица (табличный процессор) — это программа, которая хранит данные в виде таблиц и автоматически пересчитывает результаты по введённым формулам при изменении этих данных.

Самый известный табличный процессор — **Microsoft Excel**, который входит в состав коммерческого пакета *Microsoft Office*. Существует ещё очень мощный бесплатный табличный процессор **OpenOffice Calc** (он работает в *Windows*, *Linux* и *macOS*).

Работать с электронными таблицами в режиме онлайн (через Интернет) можно на сайте **docs.google.com (Документы Google)**. При этом файлы хранятся на сервере и доступны везде, где есть доступ в Интернет. Это особенно удобно, если документ просматривают и редактируют несколько человек.

Основные элементы таблицы

Таблица состоит из отдельных **ячеек**, ячейки образуют **строки** и **столбцы**. Столбцы обозначаются латинскими буквами (A, B, C, ...), а строки — номерами, начиная с 1.

В латинском алфавите всего 26 букв. Как можно назвать следующие столбцы (27-й и далее), если использовать только буквы? Выясните, правильна ли ваша догадка.

Для того чтобы обратиться к ячейке (например, использовать её значение в вычислениях), нужно как-то задать её адрес. **Адрес ячейки** складывается из имени столбца и номера строки. Например, B3 — это ячейка, расположенная в 3-й строке столбца B. На рис. 4.1 эта ячейка выделена жирной рамкой, значит, это **активная ячейка**. Если начать набирать что-то на клавиатуре, символы будут введены именно в эту ячейку.

	A	B	C
1			
2			
3			
4			

Рис. 4.1

Имя столбца и номер строки, в которых находится активная ячейка, выделяются цветом.

Ввод данных

В каждую ячейку таблицы можно ввести какие-то данные, причём программа *по умолчанию* (т. е. если мы явно не дадим ей команду сделать иначе) сама определяет, к какому типу они относятся. Это может быть:

- текст;
- числа (целые или с дробной частью); в русских версиях программ дробная и целая части разделяются запятой;

- денежная сумма (вместе с числом на экран выводится обозначение денежной единицы, например «р.»);
- дата;
- время.

На рисунке 4.2 показаны различные типы данных в электронной таблице. Заметьте, что по умолчанию текст выравнивается по левой границе ячейки, а числовые значения — по правой.

	A	B	C
1	Текст	Привет!	
2	Число	100,45	
3		1,0045E+02	
4	Денежная сумма	100,45 р.	
5	Дата	20.10.2015	
6	Время	10:48:00	
7			

Рис. 4.2

Числа могут быть записаны как в обычной форме, так и в научной (с буквой «E»). Научный формат используется для записи очень больших или очень маленьких чисел. Например, 1,234E–06 означает $1{,}234 \cdot 10^{-6} = 0{,}000001234$.

Запишите в тетради числа в обычном формате:

а) 7,567E–09; б) 4,32E+06; в) 2,7E+00.

Число, месяц и год в записи даты разделяются точками, часы минуты и секунды — двоеточиями.

Для ввода данных в ячейку нужно сначала **выделить** её щелчком мышью. Другой вариант — клавишами-стрелками перевести курсор (жирную рамку) в нужное место.

Ввод любых данных заканчивается нажатием клавиши *Enter*. Если после этого снова начать вводить число или текст, предыдущее значение активной ячейки будет стёрто и вместо него запишется новое. Чтобы не вводить заново, а отредактировать содержимое ячейки, нужно нажать клавишу *F2*. Для этого можно также сделать двойной щелчок мышью в ячейке.

Кроме того, содержимое выделенной (активной) ячейки можно изменять в строке редактирования над таблицей (рис. 4.3).

Рис. 4.3

Использование формул

Самая важная возможность электронных таблиц — использование формул.

Запись формулы в ячейке электронной таблицы начинается знаком «=».

После знака «=» пишут выражение, которое нужно вычислить. Например, для того чтобы получить в ячейке A3 сумму значений, записанных в ячейках A1 и A2, нужно ввести в эту ячейку формулу

=A1+A2

Завершив ввод этой формулы нажатием клавиши *Enter*, мы увидим результат — число 3 (рис. 4.4). Здесь A1 и A2 — это ссылки на ячейки, т. е. адреса ячеек, значения которых используются в вычислениях.

Рис. 4.4

Ссылка — это адрес ячейки в записи формулы.

Теперь для того, чтобы подсчитать сумму двух чисел, нам достаточно ввести эти числа в ячейки A1 и A2, и табличный процессор будет сразу пересчитывать результат в A3.

Умножение обозначается знаком *, деление — знаком /, а возведение в степень — знаком ^ (рис. 4.5).

Рис. 4.5

Формулы всегда записываются в одну строку, даже если математическое выражение «многоэтажное». Такая же (*линейная*) запись используется во многих языках программирования. Например, математическая формула[1)]

$$C1 = \frac{A1 + A2}{B1 + B2}$$

в табличном процессоре должна быть записана (в ячейке C1) так:

=(A1+A2)/(B1+B2).

Пусть A1 = 1, A2 = 4, B1 = 2 и B2 = 3. Что подсчитает компьютер, если пропустить скобки и ввести формулу =A1+A2/B1+B2? Почему?

Какую формулу нужно записать в ячейку D1 электронной таблицы, чтобы вычислить значение выражений?

а) $D1 = \frac{A1 + B1}{A1} + \frac{B2}{1 + C2}$; б) $D1 = A1 + \frac{B1}{1 + \frac{C2}{1 + C3}}$.

Чтобы проверить или исправить формулу, можно войти в режим редактирования ячейки (щёлкнув по ней дважды или нажав клавишу *F2*). При этом все ячейки, на которые она ссылается, будут выделены цветными рамками. Эти рамки можно перетаскивать, изменяя ссылки в формуле.

Примеры решения задач

Задача 1. Автомобиль проехал 120 км за 2 часа. Найти среднюю скорость автомобиля.

---

1) Здесь A1, A2, B1 и B2 — адреса ячеек, значения которых нужно использовать, а C1 — адрес ячейки, в которой нужно получить результат.

Конечно, если вам нужно решить всего одну такую задачу, нет смысла строить электронную таблицу, можно просто подсчитать ответ с помощью калькулятора. Но если таких задач 50 или 100 и в каждой различные исходные данные? В этой ситуации электронная таблица поможет сэкономить много времени.

Как вычислить среднюю скорость в этой задаче?

Решение. Запишем расстояние в ячейку таблицы A1, а скорость — в ячейку A2. Тогда в A3 можно записать формулу для расчёта средней скорости: =A1/A2 (рис. 4.6).

Рис. 4.6

Теперь, если изменить значения в ячейках A1 и A2, программа автоматически пересчитает значение средней скорости.

Однако работа выполнена не до конца. Если вы вернётесь к этой таблице через несколько дней (месяцев, лет), будет непонятно, что она делает. Чтобы разобраться, придётся заново просматривать все формулы и вспоминать, какую задачу мы решали. Поэтому лучше сразу сделать поясняющие текстовые надписи в соседних ячейках таблицы (рис. 4.7).

Рис. 4.7

Но для этого нужно как-то освободить ячейки столбца A, сдвинув исходные данные и формулу вправо на один столбец.

К счастью, вводить всё заново не нужно. Мы выделим мышью **диапазон** — прямоугольную часть таблицы, которая включает ячейки A1, A2 и A3, и перетащим его за рамку на один столбец вправо (рис. 4.8).

Рис. 4.8

После этого можно добавить поясняющие надписи в освободившиеся ячейки A1, A2 и A3.

Диапазон — это прямоугольная часть таблицы.

Обратите внимание, что формула в B3 теперь другая. Программа определила, что исходные данные, необходимые для расчёта, перемещаются, и автоматически изменила обе ссылки в формуле: с A1 на B1 и с A2 на B2.

Задача 2. Автомобиль сначала проехал 120 км за 2 часа, а потом ещё 170 км за 3 часа. Найти среднюю скорость автомобиля.

Как вычислить среднюю скорость на всём маршруте в этой задаче? Получится ли верный ответ, если найти среднюю скорость отдельно для каждого перегона, а потом — среднее арифметическое из этих скоростей?

Решение. Сразу записываем в столбец A пояснения, а в столбцы B и C — данные о расстоянии и времени (в столбец B — для первого участка пути, в столбец C — для второго) — рис. 4.9.

Рис. 4.9

Теперь в какую-нибудь ячейку 3-й строки, например в C3, можно ввести формулу для расчёта скорости.

В каких ячейках таблицы на рис. 4.9 записаны значения расстояния и времени движения на первом и втором участке?

Какую формулу нужно записать в ячейку C3?

Выводы

- Электронная таблица (табличный процессор) — это программа, которая хранит данные в виде таблиц и автоматически пересчитывает результаты по введённым формулам при изменении этих данных.
- Таблица состоит из ячеек, горизонтальный ряд ячеек называется строкой, а вертикальный — столбцом.
- В ячейках электронной таблицы можно хранить текст, числа, формулы, даты, отсчёты времени.
- Столбцы обозначаются латинскими буквами (одной или несколькими), а строки — порядковыми номерами.
- Адрес ячейки состоит из имени столбца и номера строки, на пересечении которых она находится.
- Ячейка, в которую выполняется ввод данных, называется активной. Она выделяется жирной рамкой.
- Диапазон — это прямоугольная часть таблицы.
- Запись формулы начинается знаком «=».
- Формулы записываются в одну строку. Для того чтобы операции выполнялись в правильном порядке, используются круглые скобки.
- В формулах можно использовать числа и адреса ячеек (ссылки на ячейки), в которых находятся данные для расчёта.
- Содержимое ячеек и диапазонов можно перетаскивать в другое место таблицы за рамку. При этом ссылки во всех формулах изменяются так, чтобы они относились к нужным данным.

Интеллект-карта

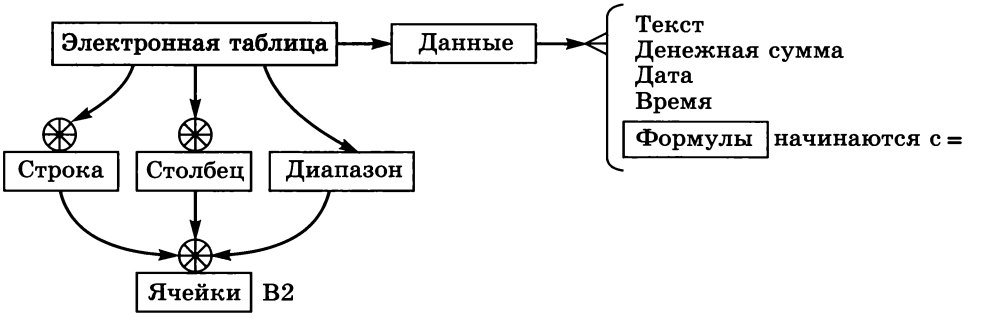


Рис. 4.10

Вопросы и задания

1. Никита предпочитает хранить все свои электронные таблицы в Интернете, в «облачных хранилищах». Оцените достоинства и недостатки этого решения.
2. Какими способами можно выделить нужную ячейку?
3. Сколько столбцов можно обозначить с помощью двух латинских букв?
4. Как редактировать содержимое ячейки, не вводя его заново? Назовите два способа.
5. Что отличает формулу от других типов данных?
6. В чём вы видите главное достоинство электронных таблиц?
7. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «История табличных процессоров»
б) «Табличные процессоры для компьютеров *Apple*»
в) «Табличные процессоры в режиме онлайн»
г) «Функции в электронных таблицах»

Практическая работа

Выполните практическую работу № 26 «Электронные таблицы».

§ 24
Редактирование и форматирование таблицы

Ключевые слова:

- выделение ячеек
- перемещение
- копирование
- удаление ячеек
- добавление ячеек
- форматирование

Все действия с электронными таблицами можно разделить на **редактирование** — изменение данных и структуры таблицы, и **форматирование** — изменение внешнего вида ячеек.

Выделение ячеек и диапазонов

Чтобы изменить какую-то часть таблицы, нужно сначала её выделить. Ячейка выделяется щелчком мышью на ней, после этого вокруг ячейки появляется жирная рамка.

Диапазон — прямоугольная часть таблицы — выделяется мышью так, как мы рисуем прямоугольник в графическом редакторе. Если нужно выделить сразу несколько диапазонов, второй и следующие выделяются при нажатой клавише *Ctrl*.

Выделить целую строку или несколько строк можно мышью в самом левом столбце таблицы, где показаны номера строк (рис. 4.11).

Рис. 4.11

Выясните экспериментально, как выделить:
а) одну строку таблицы;
б) несколько соседних строк;
в) несколько строк, расположенных в разных местах;
г) всю таблицу.

Аналогично выделяются и столбцы — в самой верхней строке, где написаны их имена (рис. 4.12).

Рис. 4.12

Кнопка в левом верхнем углу таблицы (на пересечении строки с именами столбцов и столбца с номерами строк) выделяет всю таблицу.

Запишите в тетради, как выделить:
а) один столбец таблицы;
б) несколько соседних столбцов;
в) несколько столбцов, расположенных в разных местах.

Перемещение и копирование и данных

Для перемещения и копирования данных, находящихся в ячейках, можно использовать буфер обмена:

- клавиши *Ctrl+C* (кнопка ) — скопировать выделенную часть в буфер обмена;
- клавиши *Ctrl+X* (кнопка ) — вырезать в буфер обмена;
- клавиши *Ctrl+V* (кнопка ) — вставить из буфера обмена.

Кроме того, можно «схватить» выделенную часть мышью[1] и перетащить в другое место. Если при этом удерживать клавишу *Ctrl*, данные будут *скопированы* в новое место.

Используя дополнительные источники, узнайте, как при перетаскивании вставить ячейки между существующими ячейками (рис. 4.13).

Рис. 4.13

Удаление ячеек

При нажатии клавиши *Delete* удаляется содержание выделенных ячеек. Если нужно совсем удалить ячейки (строки, столбцы), удобнее всего использовать команду *Удалить ячейки* из контекстного меню, которое появляется при щелчке правой кнопкой мыши на выделенной части. При этом можно сдвинуть соседние ячейки вверх или влево, а также удалить всю строку или весь столбец.

---

[1] В *Microsoft Excel* нужно перетаскивать диапазон за рамку.

Добавление ячеек

В контекстном меню есть команда *Вставить ячейки*, с помощью которой можно добавить ячейки в таблицу. Можно также вставить целые строки или столбцы.

Исследуйте, сколько ячеек вставляется по команде *Вставить ячейки* контекстного меню.

Форматирование ячеек

Для изменения оформления ячеек в *OpenOffice Calc* используется панель *Свойства*, а в *Microsoft Excel* — панель *Главная* на *Ленте* (рис. 4.14).

Рис. 4.14

С помощью кнопок из группы *Шрифт* можно выбрать гарнитуру и размер шрифта, цвет букв, установить стили: **полужирный**, *курсив*, подчёркнутый. Кнопка позволяет выбрать фоновый цвет (цвет заливки).

Если попробовать распечатать только что созданную таблицу, мы не увидим сетки — линий, разделяющих ячейки таблицы. Чтобы вывести их на печать, нужно добавить рамку к ячейкам таблицы. Такая возможность есть во всех табличных процессорах, например в программе *Calc* для этого существует кнопка *Обрамление* .

Кнопки группы *Выравнивание* определяют горизонтальное и вертикальное выравнивание данных в ячейке. Кнопка объединяет все выделенные ячейки в одну, это очень удобно, если нужно сделать заголовки, охватывающие несколько столбцов или строк таблицы (рис. 4.15).

	A	B	C	D	E	F
1	Весна			Лето		
2	март	апрель	май	июнь	июль	август
3	31	30	31	30	31	31

Рис. 4.15

С помощью кнопок группы *Число* задаётся формат вывода данных. Кнопки [←,0 / ,00] и [,00 / →,0] изменяют количество знаков в дробной части числа. Кнопка [денежный] задаёт денежный (финансовый) формат вывода: рядом с числом будет добавлено обозначение валюты (например, «р.»). С помощью кнопки [%] можно установить процентный формат — например, если в ячейке записано число «0,9», то вы увидите на экране «90%».

Выводы

- Все действия с электронными таблицами можно разделить на редактирование — изменение данных и структуры таблицы, и форматирование — изменение внешнего вида ячеек.
- Для того чтобы изменить свойства ячеек, их нужно выделить.
- Одна ячейка выделяется щелчком мышью на ней. Диапазон выделяется протаскиванием указателя мыши при нажатой левой кнопке из одного угла в другой (противоположный). Строки выделяются в левом столбце, где записаны их номера. Столбцы выделяются в верхней строке, где записаны их имена.
- Для перемещения и копирования данных можно использовать буфер обмена или перетаскивание диапазонов мышью.
- Для выделенных ячеек можно изменить формат вывода данных, шрифт, цвет фона, выравнивание, рамку.

Интеллект-карта

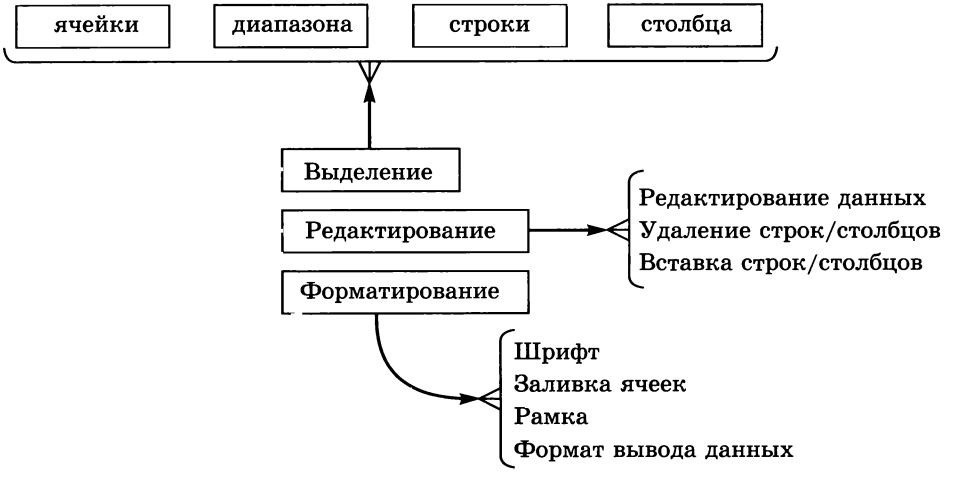


Рис. 4.16

Вопросы и задания

1. Как можно выделить одновременно несколько ячеек, расположенных в разных местах таблицы?
2. Что будет в ячейке электронной таблицы, если ввести в неё число 12 и установить процентный формат?
3. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «История электронных таблиц»
б) «Что такое VBA?»
в) «Условное форматирование»

Интересные сайты:

excelworld.ru — «Мир Excel»
planetaexcel.ru — «Планета Excel»
wiki.openoffice.org/wiki/RU/kb/module/calc — справка по *OpenOffice Calc*
help.libreoffice.org/Calc/Welcome_to_the_Calc_Help/ru — справка по *LibreOffice Calc*

Практическая работа

Выполните практическую работу № 27 «Оформление электронных таблиц».

§ 25
Стандартные функции

Ключевые слова:

- сумма
- диапазон
- минимум
- максимум
- среднее арифметическое

Мощь электронных таблиц связана ещё и с тем, что они содержат большое количество встроенных функций. Функции могут выполнять довольно сложные вычисления и обрабатывать данные сразу целого диапазона ячеек.

Суммирование

Пусть, например, нам нужно подсчитать сумму значений в ячейках A1, A2, A3, A4, A5.

Запишите формулу, с помощью которой можно сложить значения этих ячеек.

Формулу, в которой перечисляются все ячейки, очень тяжело использовать для диапазона, состоящего, скажем, из 1000 ячеек. Стандартная функция SUM (в русской версии *Excel* — СУММ) позволяет сделать то же самое более красиво:

=SUM(A1:A5)

В скобках записан адрес диапазона, включающего все нужные ячейки.

Найдите в дополнительных источниках перевод английского слова *sum* на русский язык.

Адрес диапазона составляется из двух адресов ячеек, расположенных в левом верхнем и правом нижнем углах диапазона, они разделяются двоеточием. На рис. 4.17 выделен диапазон B2:C5.

Рис. 4.17

Запишите адреса выделенных диапазонов (рис. 4.18)

а) б)

в) г)

Рис. 4.18

Выделите в таблице диапазоны A1:C1, A1:A2, A1:B2, B1:B3, A1:C3, B2:C3.

Сколько ячеек входит в диапазоны:
а) A1:A2; б) A1:C1; в) A15:A48; г) B16:G16; д) B3:G5; е) F12:G25?

Запишите формулу, с помощью которой можно найти сумму всех значений в столбце F, которые расположены в строках с 15-й по 37-ю.

Чтобы не ошибиться, при вводе формулы с функцией можно набрать название функции, открыть скобку, а затем мышью выделить нужный диапазон прямо в таблице и закрыть скобку.

Функция SUM умеет складывать данные сразу из несколько диапазонов. Например, нужно найти общую зарплату двух бригад рабочих (рис. 4.19).

	A	B	C	D
1	**Бригада 1**		**Бригада 2**	
2	Иванов	25 000 р.	Сидоров	30 000 р.
3	Петров	17 000 р.	Макеев	35 000 р.
4			Пименов	15 000 р.
5	Всего	**122 000 р.**		

Рис. 4.19

Запишите в ячейке B5 формулу, с помощью которой можно найти общую сумму зарплат рабочих обеих бригад.

Для решения этой задачи достаточно одного вызова функции:

=SUM(B2:B3;D2:D4)

Адреса диапазонов, которые нужно использовать, перечисляются через точку с запятой. В режиме редактирования формулы (клавиша *F2*) можно увидеть, какие диапазоны участвуют в вычислениях (они выделяются цветными рамками).

Вспомните, как можно выделить несколько диапазонов.

Функция SUM складывает данные только из ячеек с числами, на остальные «не обращает внимания». Поэтому в нашей задаче правильный результат можно было получить и с помощью формулы

=SUM(A1:D4)

Какие числа появятся в ячейках с формулами после ввода формул, показанных на рис. 4.20?

а)

	A	B
1	2	15
2	=B1–3*A1	
3		=SUM(A1:B2)

б)

	A	B
1	7	5
2	=SUM(A1:B1)	=SUM(A1:A2)
3		=SUM(A1:B2)

в)

	A	B
1	3	6
2	=B1^A1	=A2–B1
3		=SUM(B1:B2)

г)

	A	B
1	4	=B2–3*A1
2	=B1–3*A1	18
3	=SUM(B1:B3)	=SUM(A1:B2)

Рис. 4.20

В ячейках диапазона A1:A3 находятся однозначные натуральные числа. В ячейку B1 записали формулу =A1+A2, а в ячейку B2 — формулу =A2+A3. После ввода формул значение в B1 стало равно 16, а значение в B2 равно 15. Какие числа могли быть записаны в ячейки диапазона A1:A3? Найдите все варианты.

После ввода всех формул в ячейке B3 появилось число 16. Определите, какое число было записано в ячейке B1 и какие числа появятся в ячейках A2 и B2 после ввода формул (рис. 4.21).

	A	B
1	1	?
2	=SUM(A1:B1)	=SUM(A1:A2)
3		=SUM(A1:B2)

Рис. 4.21

Минимум, максимум, среднее арифметическое

Для вычисления минимального и максимального значений используются функции MIN и MAX (в русской версии — МИН и МАКС). При их вызове в скобках можно задать один или несколько диапазонов, например:

=MIN(B2:C4)

=MAX(A1:A20;C1:C20;D8:D34)

Определите значения всех ячеек после ввода формул (рис. 4.22).

	A	B	C	D
1	2	=A1^B2	=MIN(A1:B2)	=MAX(A1:B1)
2	=A1–B2	4	=MIN(A2:B2)	=MAX(A2:B2)
3	=MIN(A1:A2)	=MIN(B1:B2)	=MIN(A1:B2)	
4	=MAX(A1:A2)	=MAX(B1:B2)		=MAX(A1:B2)

Рис. 4.22

Запишите в тетради несколько различных формул, которые при вводе в ячейку D4 выведут тот же результат.

Среднее арифметическое (сумму чисел, разделённую на их количество) вычисляет функция AVERAGE (СРЗНАЧ), например,

=AVERAGE(B2:C4)
=AVERAGE(A1:A20;C1:C20;D8:D34)

Экспериментально выясните, какие ячейки учитывает функция AVERAGE. Проверьте её работу в трёх случаях, каждый раз вычисляя среднее арифметическое для диапазона A1:B2 (рис. 4.23).

а)

	A	B
1	1	2
2	3	4
3		

б)

	A	B
1	1	2
2	3	
3		

г)

	A	B
1	1	2
2	Сумма	
3		

Рис. 4.23

Определите значения всех ячеек после ввода формул (рис. 4.24).

	A	B	C
1	2	=B2–3*A1	=AVERAGE(A1:B1)
2	=3*A1+B2	6	=AVERAGE(A2:B2)
3	=AVERAGE(A1:A2)	=AVERAGE(B1:B2)	=AVERAGE(A1:B2)

Рис 4.24

Запишите в тетради несколько различных формул, которые при вводе в ячейку C3 выведут тот же результат.

Предположим, что нам нужно вычислить среднее арифметическое значений, записанных в два диапазона, A1:A2 и C1:D2 (рис. 4.25).

	A	B	C	D
1	2		6	2
2	4		3	9

Рис 4.25

Выясните, можно ли найти сначала среднее арифметическое для каждого диапазона, а затем — среднее арифметическое из двух полученных средних:

1) вычислите в ячейке A3 среднее арифметическое для диапазона A1:A2;
2) вычислите в ячейке C3 среднее арифметическое для диапазона C1:D2;
3) вычислите в ячейке B3 среднее арифметическое для диапазона A3:C3 («среднее из средних»);
4) вычислите в ячейке B4 среднее арифметическое для диапазона A1:D2 (правильное значение среднего арифметического).

Сравните значения в ячейках B3 и B4, сделайте выводы.

Другие функции

Мы изучим ещё одну функцию — **SUMPRODUCT (СУММПРОИЗВ)**, которая вычисляет сумму произведений ячеек двух диапазонов.

Предположим, что нам нужно построить электронную таблицу для расчёта стоимости покупки в магазине (рис. 4.26).

	A	B	C
1	Товар	Цена	Количество
2	Молоко	50 р.	3
3	Сметана	25 р.	1
4	Пряник	15 р.	4
5		Сумма	235 р.

Рис. 4.26

Запишите в тетради формулу, которую нужно ввести в ячейку C5.

Такие формулы очень тяжело вводить, когда в таблице несколько сотен (или даже тысяч!) строк. Обратите внимание, что в этой формуле участвуют ячейки двух диапазонов: B2:B4 и C2:C4. Сначала первая ячейка первого диапазона умножается на первую ячейку второго диапазона, вторая — на вторую и т. д., а затем

все эти произведения складываются. Функция SUMPRODUCT делает всё это сразу, поэтому в C5 можно записать формулу

=SUMPRODUCT(B2:B4;C2:C4)

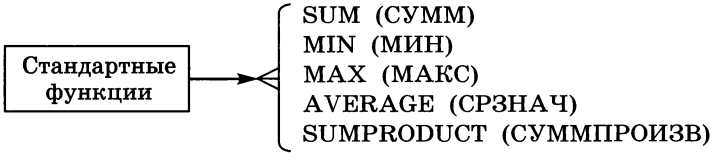
Эта функция принимает два аргумента — диапазоны одинаковой длины.

Электронные таблицы содержат очень много встроенных функций, которые разбиты на группы: математические, логические, текстовые и др. Более сложные функции, которые применяются при обработке больших массивов данных, мы будем изучать в следующем году.

Выводы

- Функция SUM (в русской версии — СУММ) используется для сложения всех чисел в указанном диапазоне, например SUM(A1:A100).
- Функции MIN (МИН), MAX (МАКС) и AVERAGE (СРЗНАЧ) применяют для вычисления минимального, максимального и среднего арифметического чисел указанного диапазона.
- Функции SUM, MIN, MAX, AVERAGE не учитывают (пропускают) все нечисловые ячейки, в том числе и пустые.
- Функции SUM, MIN, MAX, AVERAGE могут обрабатывать сразу несколько диапазонов; адреса нужных диапазонов перечисляются через точку с запятой, например SUM(A1:A100;B20:X50).
- Функция SUMPRODUCT (СУММПРОИЗВ) вычисляет сумму произведений элементов двух диапазонов, например SUMPRODUCT(A1:A100;B1:B100).

Интеллект-карта



Примеры:
=SUM(A1:A100)
=MIN(A1:A100;B20:X50)
=AVERAGE(B30:F100)
=SUMPRODUCT(A1:A100;B1:B100)

Рис. 4.27

Вопросы и задания

Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Стандартные математические функции»
б) «Стандартные функции для работы с временем и датой»
в) «Стандартные функции для работы с текстом»

Практическая работа

Выполните практическую работу № 28 «Стандартные функции».

§ 26
Сортировка данных

Ключевые слова:

- сортировка
- диапазон
- многоуровневая сортировка

Как вы знаете, **сортировка** — это расстановка элементов списка в заданном порядке. Для чисел обычно используют сортировку по возрастанию или убыванию, для текста — алфавитный (от А до Я) или обратный алфавитный порядок (от Я до А).

Простая сортировка

Для того чтобы отсортировать один столбец данных, нужно выделить эти данные и использовать кнопки сортировки  и  . В *Excel* эти кнопки находятся на вкладке *Данные*, а в *OpenOffice Calc* аналогичные кнопки  и  расположены на верхней панели инструментов.

Выясните экспериментально, как именно сортируют данные кнопки  и  в *Excel* (и аналогичные кнопки и в *OpenOffice Calc*).

Программа *Calc* по умолчанию предполагает, что первая строка выделенного диапазона — это заголовок, и он в сортировке не участвует. Изменить этот режим можно с помощью меню *Данные → Сортировать*.

Как правило, в электронной таблице хранятся связанные данные, например фамилия сотрудника и его зарплата. Предположим, что нам нужно отсортировать по алфавиту список сотрудников (рис. 4.28).

	A	B
1	**Сотрудник**	**Зарплата**
2	Иванов	12 000 р.
3	Петров	15 000 р.
4	Акимов	17 000 р.
5	Дубов	11 000 р.

	A	B
1	**Сотрудник**	**Зарплата**
4	Акимов	17 000 р.
5	Дубов	11 000 р.
2	Иванов	12 000 р.
3	Петров	15 000 р.

Рис. 4.28

Что произойдёт, если в таблице на рис. 4.28 отсортировать только значения из столбца A, не меняя данные в столбце B?

Перед сортировкой связанных данных нужно выделить все столбцы с данными, тогда строки будут переставляться целиком (см. рис. 4.28).

Выясните, что произойдёт, если в таблице на рис. 4.28 выделить только столбец с фамилиями и применить сортировку.

Сортировка по любому столбцу

Теперь попробуем отсортировать тот же список по убыванию зарплаты (рис. 4.29).

	A	B
1	**Сотрудник**	**Зарплата**
2	Иванов	12 000 р.
3	Петров	15 000 р.
4	Акимов	17 000 р.
5	Дубов	11 000 р.

	A	B
1	**Сотрудник**	**Зарплата**
4	Акимов	17 000 р.
5	Петров	15 000 р.
2	Иванов	12 000 р.
3	Дубов	11 000 р.

Рис. 4.29

Здесь уже кнопки  и не помогут, потому что они всегда сортируют по первому столбцу выделенного диапазона.

Сначала выделим весь диапазон A1:B5 (вместе с заголовками). После этого в программе *Excel* нужно щёлкнуть на кнопке  на вкладке *Данные*, затем выбрать нужный столбец и порядок

сортировки (в данном случае — столбец *Зарплата*, сортировка по убыванию). В программе *Calc* используется меню *Данные* → *Сортировать*.

Многоуровневая сортировка

Предположим, в таблице записаны данные об альбомах музыкальных групп, которые нужно отсортировать по названию группы, а альбомы каждой группы — по году выпуска (новые альбомы — в начале) — рис. 4.30. Такую сортировку называют **многоуровневой**.

	A	B	C
1	**Группа**	**Альбом**	**Год**
2	Любэ	Давай за...	2002
3	Город 312	Обернись	2007
4	Любэ	Свои	2009
5	Город 312	Вне зоны доступа	2006
6	Город 312	Новая музыка	2010
7	Любэ	Комбат	1996

	A	B	C
1	**Группа**	**Альбом**	**Год**
6	Город 312	Новая музыка	2010
3	Город 312	Обернись	2007
5	Город 312	Вне зоны доступа	2006
4	Любэ	Свои	2009
2	Любэ	Давай за...	2002
7	Любэ	Комбат	1996

Рис. 4.30

В программе *Excel* многоуровневая сортировка выполняется с помощью кнопки  на вкладке *Данные*, а в *Calc* — через меню *Данные* → *Сортировать*. Задание на сортировку в нашей задаче может быть задано так (рис. 4.31).

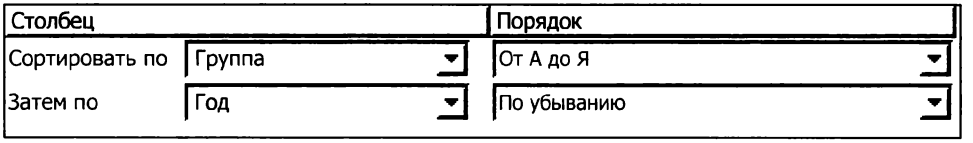


Рис. 4.31

Вопросы и задания

1. Объясните, почему в электронных таблицах, как правило, нельзя сортировать только один столбец с данными.
2. Как вы думаете, что произойдёт, если в столбце, по которому выполняется сортировка, есть пустые ячейки? Проверьте своё предположение с помощью программы.
3. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

«Умная сортировка списка файлов в *Windows*»

Интересные сайты

technet.microsoft.com/ru-ru/magazine/hh475812.aspx — сортировка файлов в *Windows*

Практическая работа

Выполните практическую работу № 29 «Сортировка».

§ 27
Относительные и абсолютные ссылки

Ключевые слова:

- относительная ссылка
- абсолютная ссылка
- смешанная ссылка

Что происходит при копировании?

Давайте проведём эксперимент: введём любые числа в ячейки диапазона A1:B2, а затем формулу =A1+B1 в ячейку C1 (рис. 4.32).

	A	B	C
1	1	3	=A1+B1
2	2	4	
3			

Рис. 4.32

Затем скопируем формулу из ячейки C1 в ячейку C2 (например, через буфер обмена или перетащив мышью при нажатой клавише *Ctrl*).

Выясните, как изменилась формула при копировании. Как вы думаете, почему она изменилась именно так?

В нашей формуле были две ссылки на другие ячейки, обе они изменились (рис. 4.33). Мы скопировали формулу на одну ячейку вниз, поэтому в каждой ссылке номера строк увеличились на

единицу (из A1 получилось A2, а из B1 — B2). Если скопировать формулу вправо, то увеличатся номера столбцов (см. рис. 4.33).

	A	B	C	D
1	1	3	=A1+B1	=B1+C1
2	2	4	=A2+B2	=B2+C2
3				

Рис. 4.33

В общем случае, если формула скопирована на n ячеек вправо и m ячеек вниз, во всех ссылках имена столбцов увеличиваются на n, а номера строк — на m. Такие ссылки называются *относительными*.

Адрес ячейки в **относительной ссылке** при копировании изменяется так же, как изменяется адрес ячейки, в которой записана формула.

Другими словами, относительная ссылка «запоминает» взаимное расположение ячеек. При копировании формулы сохраняется связь ячеек между собой.

Формула =C13+F4 записана в ячейку D8. Какая формула получится, если скопировать формулу из ячейки D8 в ячейки:
а) B8; б) F8; в) D6; г) D12; д) B6; е) B12; ж) F6; з) F12; и) A8; к) G3?

Такое изменение формул при копировании очень удобно при заполнении больших таблиц. Предположим, что в двух столбцах таблицы хранятся доходы и расходы компании за первые месяцы года, и нужно подсчитать прибыль (разность доходов и расходов) за каждый месяц (рис. 4.34).

	A	B	C	D
1	**Месяц**	**Доходы**	**Расходы**	**Прибыль**
2	январь	530 000 р.	120 000 р.	=B2–C2
3	февраль	532 000 р.	125 800 р.	
4	март	635 000 р.	224 000 р.	

Маркер заполнения

Рис. 4.34

Конечно, для нескольких строк можно и вручную вписать в ячейки D3 и D4 нужные формулы, но можно просто скопировать в них формулу из D2, при этом ссылки изменятся как раз так, как нужно.

Для быстрого копирования удобно использовать **маркер заполнения** — чёрный квадратик в правом нижнем углу выделенной ячейки: если перетащить его мышью вниз, то формула из D2 будет скопирована во все ячейки, через которые прошел указатель мыши. Если выполнить двойной щелчок на маркере заполнения, то формула будет скопирована вниз до конца данных в предыдущем столбце. Это удобно, если количество строк в таблице велико, например 1000 или 10000.

Проверьте экспериментально, можно ли с помощью маркера заполнения копировать формулу в других направлениях: вверх, вправо и влево.

Абсолютные ссылки

Часто бывает нужно, чтобы при копировании ссылка не изменялась. Например, пусть в столбце B записаны данные о зарплате работников компании. Из этой зарплаты вычитается подоходный налог, и остаток выдаётся на руки сотруднику. Размер налога записан в ячейку B1 и нужно, чтобы при изменении значения этой ячейки пересчитывались все данные в таблице (рис. 4.35).

Вспомните, как записать в ячейку электронной таблицы значение в процентах.

	A	B	C
1	**Размер налога**	13%	
2			
3	**Сотрудник**	**Зарплата**	**К выдаче**
4	Иванов И.И.	23 000 р.	
5	Петров П.П.	18 000 р.	
6	Сидоров С.С.	32 000 р.	

Рис. 4.35

Доля зарплаты, которая выдаётся сотруднику (за вычетом налога), равна 1–B1, это значение нужно умножить на величину полной зарплаты. Таким образом, в ячейке C4 должна быть записана формула =B4*(1–B1).

Что получится, если скопировать формулу =B4*(1–B1) из ячейки C4 в ячейки C5 и C6?

Как вы поняли, нам нужно как-то запретить изменение ссылки на B1 при копировании. Для этого перед именем столбца и номером строки вставляют знак «$», так что формула в C4 принимает вид =B4*(1–B1). Здесь ссылка на ячейку B1 — абсолютная, она не изменяется при копировании, потому что запоминается точное место ячейки в таблице.

Абсолютная ссылка при копировании не изменяется.

Для того чтобы быстро сделать ссылку в формуле абсолютной, нужно установить курсор внутрь ссылки и нажать клавишу *F4* (в *Excel*) или комбинацию клавиш *Shift+F4* (в *Calc*).

Смешанные ссылки

Теперь попробуем построить таблицу умножения 5×5 (рис. 4.36).

	A	B	C	D	E	F
1		1	2	3	4	5
2	1	1	2	3	4	5
3	2	2	4	6	8	10
4	3	3	6	9	12	15
5	4	4	8	12	16	20
6	5	5	10	15	20	25

Рис. 4.36

При изменении чисел в первой строке и в столбце A все значения в центральной части таблицы должны пересчитываться, т. е. в ячейках диапазона B2:F6 должны быть записаны формулы.

Конечно, можно вписать в каждую из 25 ячеек свою формулу, но это довольно утомительно. Попробуем записать одну формулу в ячейку B2, а затем скопировать её во все остальные ячейки.

Запишем нужные формулы для двух угловых ячеек, B2 и F6. В B2 должна быть записана формула =A2*B1, а в F6 — формула =A6*F1 (рис. 4.37).

	A	B	C	D	E	F
1		1	2	3	4	5
2	1	=A2*B1				
3	2					
4	3					
5	4					
6	5					=A6*F1
7						

Рис. 4.37

Запишите в тетради формулы, по которым должны вычисляться значения в ячейках C3, F2, B6 и E4.

Мы видим, что первый сомножитель в формулах — это значение ячейки из столбца A, а номер строки меняется. Поэтому имя столбца A в первой ссылке нужно заблокировать от изменений, а номер строки — нет. Для формулы в B2 получаем такую ссылку: $A2. Аналогично поступаем для второго сомножителя: он всегда берётся из первой строки, а столбец меняется. Поэтому нужно заблокировать номер строки, оставив свободным изменяющееся имя столбца. В итоге получаем для ячейки B2 такую формулу: =$A2*B$1. Эту формулу нужно записать в B2, затем скопировать вправо на диапазон B2:F2, а потом перетащить маркер заполнения (уже целого диапазона) вниз на остальную часть таблицы.

Ссылки вида $A2 и B$1 называются смешанными, у них одна часть (номер строки или имя столбца) защищены от изменений при копировании, а вторая может изменяться.

Смешанная ссылка — это ссылка, в которой только одна часть (номер строки или имя столбца) изменяется при копировании.

Для того чтобы быстро изменить тип ссылки в формуле, нужно установить курсор внутри ссылки и нажать клавишу *F4* (в *Excel*) или комбинацию клавиш *Shift+F4* (в *Calc*). Например, если начать со ссылки B1, после первого нажатия получится абсолютная ссылка B1, после второго — смешанная ссылка B$1, а после третьего — $B1.

Выводы

- Адрес ячейки в относительной ссылке (ссылке вида A1) при копировании изменяется так же, как изменяется адрес ячейки, в которой записана формула.
- Абсолютная ссылка (ссылка вида A1) при копировании не изменяется.
- Смешанная ссылка (ссылка вида $A1 или A$1) — это ссылка, в которой только одна часть (номер строки или имя столбца) изменяется при копировании.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопосы и задания

1. В каких случаях при копировании ссылка может стать недействительной?
2. В каких случаях при копировании не изменится ссылка $A12? ссылка A$12?
3. Сформулируйте правило составления смешанных ссылок.
4. В каких случаях нужна именно абсолютная ссылка (нельзя обойтись смешанной)?
5. Выполните по указанию учителя задания в рабочей тетради.

Практическая работа

Выполните практическую работу № 30 «Относительные и абсолютные ссылки».

§ 28
Диаграммы

Ключевые слова:

- диаграмма
- столбчатая диаграмма (гистограмма)
- ряд
- категория
- легенда
- график
- круговая диаграмма
- точечная диаграмма

Что такое диаграмма?

Когда человек видит и пытается понять числовые данные, он вынужден в уме анализировать эту информацию и делать выводы. Это требует значительных усилий, особенно если чисел много. Поэтому удобно представлять информацию в наглядном графическом виде.

Диаграмма — это графическое изображение данных.

Диаграммы позволяют быстро *сравнить* значения, увидеть изменения, сделать выводы.

С одним видом диаграмм — графиками функций — вы уже работали на уроках математики. В этом параграфе мы познакомимся с другими типами диаграмм и узнаем, когда они применяются.

Выясните, от какого греческого слова произошло слово «диаграмма».

Столбчатые диаграммы

Предположим, что в таблице записаны данные о высоте некоторых гор, и нужно сравнить их в наглядной форме. Как вы знаете из курса математики, для этого можно использовать **столбчатую диаграмму** (она также называется **гистограммой**) — рис. 4.38.

	A	B
1	**Гора**	**Высота, м**
2	Эверест	8848
3	Чогори (К2)	8614
4	Пик Сомони	7495
5	Эльбрус	5642

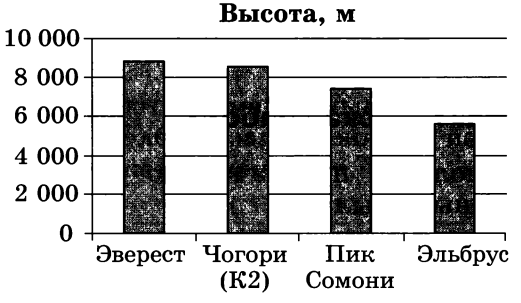


Рис. 4.38

В электронной таблице нужно выделить все данные (вместе с заголовками), т. е. диапазон A1:B5. В программе *Excel* гистограмма вставляется с помощью кнопки *Гистограмма* на вкладке *Вставка*, а в *Calc* — с помощью кнопки  или пункта меню *Вставка* → *Диаграмма*.

Настраивать свойства диаграммы удобнее всего через контекстное меню: при щелчке правой кнопкой мыши на диаграмме вы увидите список всех возможных операций. Например, так можно изменить тип диаграммы, размер шрифта надписей, цену делений на осях, цвет столбиков и др. В программе *Excel* можно использовать вкладки *Ленты Конструктор* и *Макет*, которые появляются, когда диаграмма выделена.

Рассмотрим таблицу, в которой записано количество разных домашних животных у трёх жителей деревни (рис. 4.39).

	A	B	C	D
1		овцы	кролики	куры
2	Аськин	1	2	5
3	Баськин	4	2	5
4	Сенькин	2	3	4

Рис. 4.39

Чтобы изобразить эти данные, можно использовать столбчатую диаграмму (рис. 4.40).

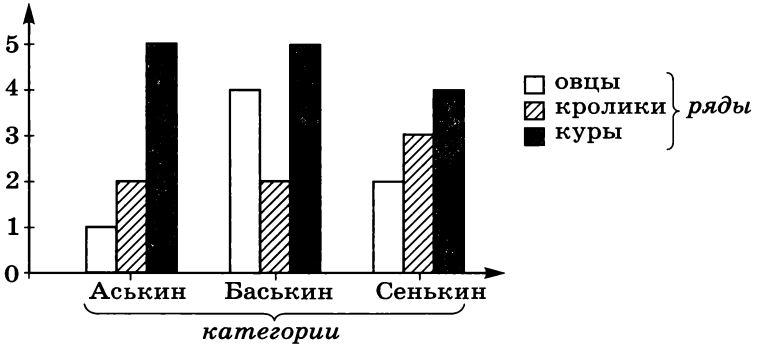


Рис. 4.40

Здесь на горизонтальной оси откладываются заголовки строк (или столбцов) таблицы, они называются **категориями**. Столбики одного цвета — это **ряд данных**, представляющий столбец таблицы. На диаграмме, приведённой на рис. 4.40, показаны три ряда данных — овцы, кролики и куры. Справа от диаграммы размещена **легенда** — список условных обозначений (цвет столбиков для каждого ряда).

По диаграмме на рис. 4.40 мы можем сразу найти ответы на вопросы типа «Каких животных больше всего у Аськина (Баськина, Сенькина)?». Заметим, что по тем же данным можно построить и другие диаграммы.

Как будет выглядеть диаграмма, если ряды и категории на рис. 4.40 поменять местами? На какие вопросы можно легко ответить с помощью этой диаграммы?

Тип диаграммы выбирается так, чтобы было лучше видно то, что хочет показать автор.

Биологи пересчитали лосей, белок и зайцев на трёх участках заповедника и построили диаграмму (рис. 4.41).

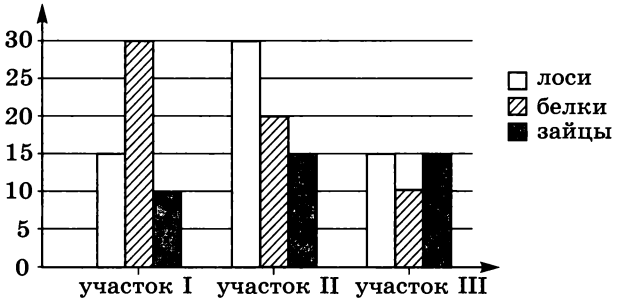


Рис. 4.41

Восстановите исходные данные в виде таблицы.

Круговые диаграммы

Вы не задумывались, из каких частей состоит бюджет вашей семьи? Допустим, что все расходы за месяц записаны в таблице и нужно наглядно представить эти данные. Для этого хорошо подходит круговая диаграмма (рис. 4.42).

Рис. 4.42

Круговая диаграмма показывает доли отдельных частей в общем количестве.

В нашем случае общее количество — это сумма расходов, а доли определяются расходами по отдельным пунктам бюджета.

Диаграмма изображается как круг, разрезанный на части (секторы), площадь каждого сектора пропорциональна доле, которую составляет соответствующая часть расходов. Как правило, на такой диаграмме только один ряд данных.

Рядом с секторами показаны **подписи данных** — дополнительная информация по каждой части. На рис. 4.42 в подписи добавлена величина доли в процентах (она вычисляется автоматически!) и название категории.

Определите доли составляющих частей по следующим диаграммам (рис. 4.43).

а) б) в) г)

д) е) ж) з)

Рис. 4.43

Нарисуйте в тетради круговую диаграмму, которая показывает соотношение общего количества различных видов животных у жителей деревни по данным рис. 4.39.

Нарисуйте в тетради круговую диаграмму, которая показывает соотношение общего количества различных видов животных в заповеднике по данным рис. 4.41.

Графики

Пусть в таблице записаны результаты измерений температуры воздуха в течение 15 дней (рис. 4.44).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	День	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	Температура, °C	15	12	8	6	9	14	12	11	15	17	18	14	16	18	12

Рис. 4.44

Если по этим данным построить гистограмму, то она будет состоять из большого числа узких столбцов, расположенных близко друг к другу. Если данных в ряду много, лучше использовать не гистограмму, а диаграмму типа *График* (в программе *Calc* этот тип диаграммы называется *Линии*) — рис. 4.45.

Рис. 4.45

На этой диаграмме добавлены названия осей. Ломаная линия строится по данным из таблицы, точки можно обозначить значками-маркерами. Можно оставить только маркеры, не соединяя точки линией. Линия может быть как ломаная, так и сглаженная.

При построении диаграмм типа *График* программа воспринимает данные по горизонтальной оси не как числа, а как текстовые надписи (категории, как в гистограммах). Все метки на горизонтальной оси для этого типа диаграмм расположены на одинаковых расстояниях друг от друга.

Построение графиков функций

Пусть вам дана какая-то функция и нужно определить, как выглядит её график. Для построения графиков используются диаграммы специального вида, которые называются *Точечные* (в программе *Excel*) или *Диаграммы XY* (в *Calc*).

Для построения графика функции нужно:

1) построить таблицу значений функции на заданном отрезке;
2) построить график по данным из таблицы.

Вспомните, как вы на уроках математики строили графики функций по точкам. Сначала необходимо задать отрезок, на котором строится график. Затем выбирается шаг — разница между соседними значениями независимой переменной. Если шаг будет слишком большой, график получится неточный, а если выбрать маленький шаг, потребуется много вычислений. Часто сначала выбирают шаг, равный 1/10 или 1/20 длины нужного отрезка, а затем, если нужно, изменяют его.

Например, построим график функции $y = x^3$ на отрезке [0; 2]. Выберем шаг, равный 0,2. Теперь нужно построить таблицу значений функции в электронной таблице. Используем для этого столбцы A и B. В ячейки A1 и B1 введём заголовки столбцов — *X* и *Y*. В ячейки A2 и A3 запишем *x*-координаты первых двух точек: 0 и 0,2. Это позволит программе автоматически определить нужный нам шаг заполнения столбца *X*. Теперь выделим эти две ячейки и протянем мышью маркер заполнения вниз, заполняя столбец *X* с заданным шагом (рис. 4.46, *а* и *б*).

а) *б*) *в*) *г*)

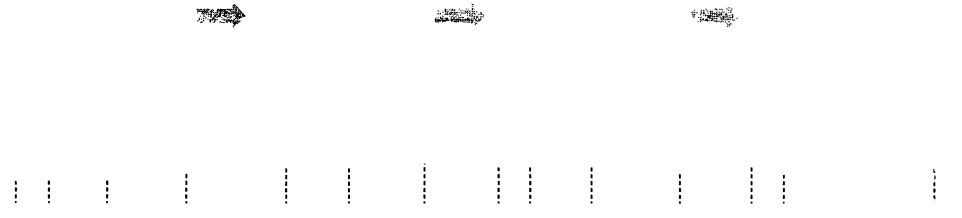


Рис. 4.46

Теперь заполним столбец *Y*. Введём в ячейку B2 (для первого значения *X*) формулу =A2^3 (вспомним, что знак «^» означает возведение в степень). Затем «протащим» мышью маркер заполнения этой ячейки вниз, копируя эту формулу во все остальные ячейки[1] (рис. 4.46, *в* и *г*).

Как при копировании из ячейки B2 в ячейку B3 изменится ссылка в формуле? Как называется такая ссылка?

Предложите ещё один вариант формулы, которую можно было бы записать в ячейку B2.

Для построения графика выделяем весь диапазон с данными (A1:B12) и вставляем диаграмму типа *Точечная* (*Диаграмма XY*). Окончательный результат показан на рис. 4.47. Таким способом можно строить графики функций и тогда, когда независимая величина *X* изменяется с переменным шагом.

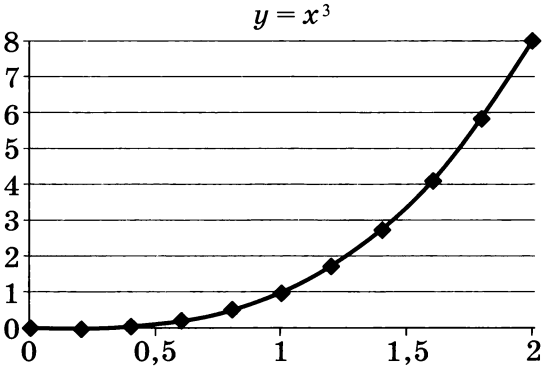


Рис. 4.47

Требуется построить график функции $y = k \cdot x^3 + b$ для значений k и b, записанных в ячейки таблицы (рис. 4.48).

	A	B	C	D	E
1	*k*	1		*X*	*Y*
2	*b*	2		0	
3				0,2	
4				0,4	
5				...	

Рис. 4.48

[1] Вместо этого можно просто сделать двойной щелчок на маркере заполнения.

Какую формулу нужно записать в ячейку E2, чтобы её можно было скопировать во все ячейки столбца E? Предложите разные варианты.

Выводы

- Диаграмма — это графическое изображение данных. Диаграммы позволяют быстро сравнить значения, увидеть изменения, сделать выводы.
- Столбчатые диаграммы (гистограммы) используются для рядов, в которых небольшое количество данных.
- Диаграммы типа *График* (*Линия*) используются для рядов данных с большим количество элементов.
- Круговые диаграммы применяют для того, чтобы показать доли частей в общем количестве.
- Для построения графиков функций используются точечные диаграммы (*диаграммы XY*).

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. В каких случаях для отображения одного ряда данных нужно использовать
   - гистограмму;
   - график;
   - круговую диаграмму?
2. Объясните разницу между двумя видами столбчатых диаграмм, которые можно построить по одним и тем же данным (возьмите для примера данные на рис. 4.39).
3. В каких случаях для построения графика функции можно использовать диаграмму типа *График* (*Линия*), а в каких — нельзя?
4. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Диаграммы с накоплением»
б) «Нормированные диаграммы»
в) «Лепестковые диаграммы»

Практическая работа

Выполните практическую работу № 31 «Диаграммы».

ЭОР к главе 4 из Единой коллекции цифровых образовательных ресурсов (school-collection.edu.ru)

Интерактивный задачник, раздел «Электронные таблицы. Запись формул»
Формулы в MS Excel
Диапазон (блок) электронной таблицы
Манипулирование фрагментами таблицы (очистка и удаление ячеек, добавление строк и столбцов, перемещение, копирование, автозаполнение) MS Excel
Сортировка данных в таблице MS Excel
Создание диаграмм MS Excel
Демонстрационная таблица с диаграммами

Глава 5
ПОДГОТОВКА ЭЛЕКТРОННЫХ ДОКУМЕНТОВ

§ 29
Работа с текстом

Ключевые слова:

- проверка орфографии
- проверка грамматики
- гиперссылки
- тезаурус
- распознавание текста

Проверка правописания

К сожалению, во время набора текста мы часто делаем ошибки и опечатки. В текстовых процессорах есть встроенные средства для проверки правильности текста. Если какое-то слово написано неверно, программа подчёркивает его красной волнистой линией[1] (рис. 5.1).

карова

Рис. 5.1

Как же программа определяет, что в слове допущена ошибка? Примерно так же, как и человек, — «заглядывает» в словарь. Только словарь хранится в памяти компьютера (в виде файла специального формата), и программа проверяет, есть ли набранное слово в этом словаре. Если слово подчёркнуто, это значит, что его нет в словаре, хотя может быть, что оно написано верно.

Щёлкните правой кнопкой мыши на слове, которое, по мнению текстового процессора, написано ошибочно. Как называется появившееся меню? Какие варианты решения проблемы вы можете выбрать?

Простая проверка по словарю — это самый простой, но не самый лучший способ. Дело в том, что в словарь приходится добавлять все формы слова — все падежи существительных и прилагательных, все формы глаголов. Более совершенные системы умеют выполнять анализ грамматики — «узнают» части речи,

[1] В учебнике — серой.

сами строят другие формы слов, проверяют, правильно ли построено предложения.

В других случаях слово или целое предложение подчёркивается не красной, а зелёной линией. Так программа сообщает о грамматической ошибке: фраза не соответствует правилам построения предложений в выбранном языке. Например, в предложении

Графический редактора изображение программы редактирования.

подчёркивается слово «графический», которое не согласовано ни с одним существительным. Такая проверка часто позволяет найти опечатки и пропущенные знаки препинания.

Даже если программа не обнаружила ошибку, это ещё не значит, что ошибок нет. Например, фраза

Саша ни хотел идти в магазин в места Коле.

содержит несколько ошибок, но, по мнению программы, в ней всё правильно. Поэтому нужно внимательно читать написанные вами тексты и проверять их вручную, используя ваши знания родного языка. Текстовый процессор может только немного помочь вам в поиске ошибок.

По умолчанию проверка орфографии и грамматики в текстовых процессорах выполняется автоматически, но можно запустить её и вручную, нажав клавишу *F7* или соответствующую кнопку на панели инструментов.

Используя всплывающие подсказки, найдите на панели инструментов кнопку для проверки орфографии и грамматики.

Компьютерные словари и переводчики

Компьютеры могут быстро искать нужную информацию в больших массивах данных, поэтому сейчас бумажные словари практически вытеснены компьютерными (электронными).

Вы уже знаете, что в текстовых процессорах есть встроенный словарь, по которому проверяется орфография — правильность написания слов. В текстовых процессорах есть также и словарь специального типа — тезаурус.

Тезаурус (в текстовом процессоре) — это словарь, который содержит синонимы, антонимы и родственные слова.

Вспомните, что такое синоним и антоним.

Найдите в дополнительных источниках другое значение слова «тезаурус».

Для вызова тезауруса в *Word* установите курсор на нужное слово и щёлкните на кнопке *Тезаурус* на панели *Рецензирование*. На рис. 5.2 показан результат вызова тезауруса для слова «справедливость».

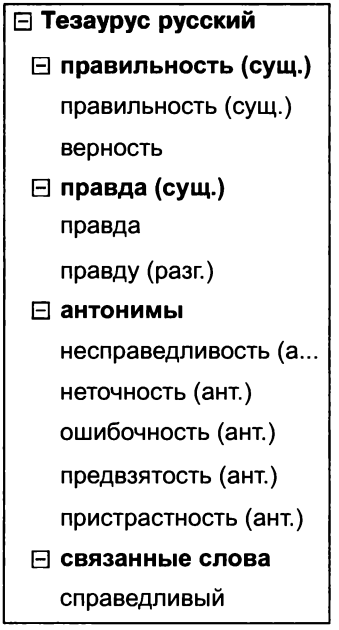


Рис. 5.2

В программе *OpenOffice Writer* тезаурус для выделенного слова вызывается через контекстное меню (пункт *Синонимы*) или пункт главного меню *Сервис* → *Язык* → *Тезаурус*.

Для тех, кому приходится переводить тексты на другие языки, очень важны двуязычные словари. Они могут устанавливаться как отдельные программы на компьютеры или смартфоны, но существуют и онлайн-версии, т. е. сайты в Интернете. Кроме всех возможных переводов слова такие словари показывают произношение (транскрипцию) и дают возможность прослушать, как произносится это слово носителями языка (для которых этот язык родной).

Переводить текст значительно сложнее, чем переводить отдельные слова. Прежде всего потому, что многие слова имеют несколько значений. Например, немецкое слово *Zug* имеет 21 значение. Но даже человеку, знающему значения всех слов, иногда бывает нелегко понять смысл всего предложения. Для компьютеров это тем более очень сложная задача, которая относится

к области искусственного интеллекта. Однако и её постепенно решают компьютерные программы-переводчики. В них заложены правила, которые позволяют выбрать нужный вариант перевода каждого слова. Для перевода текстов по специальной тематике (например, по математике или финансам) можно подключить дополнительные словари.

К сожалению, пока автоматический перевод можно использовать только для того, чтобы получить первое представление о тексте. Его нельзя применять в ответственных случаях, особенно при переводе художественной литературы. Например, предложение

Наша Таня громко плачет: уронила в речку мячик.

после автоматического перевода в одной из систем на английский язык и обратного перевода на русский стало выглядеть так:

Наша Таня громко кричит: я допустил ошибку к небольшой реке.

Распознавание текстов

Несколько сотен лет человечество накапливало информацию в бумажном виде: в форме книг, газет, журналов. Сейчас возникла необходимость ввести эти данные в компьютер. Это тяжёлая работа, отнимающая очень много времени. Представьте, сколько времени потребуется вам, чтобы набрать текст книги размером в 500 страниц.

Существует другой способ — вводить страницы с помощью сканера (выполнять их оцифровку). Но, как вы знаете, сканер вводит изображение как точечный рисунок, т. е. набор пикселей. Для того чтобы значительно уменьшить объём файлов и сделать возможным поиск по тексту, нужно закодировать документ как текст. Это позволит легко редактировать его, например заменять и удалять фрагменты.

Итак, сканер вводит рисунок, а нам нужен текст. Это значит, что необходима какая-то программа, которая умеет распознавать буквы и цифры в комбинациях пикселей разного цвета. Такая задача называется задачей **оптического распознавания символов.** Она тоже относится к области искусственного интеллекта, как и задача машинного перевода. Существуют программы, которые в некоторых случаях могут распознать даже рукописный текст, но они работают не очень надёжно (подумайте, почему).

Используя дополнительные источники, выясните, от каких английских слов произошло сокращение *OCR*. Что оно обозначает?

Вспомните, в каких единицах измеряется качество сканирования? Как связано качество сканирования и качество распознавания документа?

Самая известная программа для распознавания текста — *FineReader* компании *ABBYY*. Это коммерческая программа, но часто одна из её версий поставляется на диске при покупке сканера.

Бесплатная программа распознавания текста — *CuneiForm*, которая относится к свободному программному обеспечению. Она работает под управлением *Windows*, *Linux*, *macOS* и других операционных систем.

Голосовой ввод текста

Для того чтобы ввести текст, можно использовать не только клавиатуру, но и специальные программы, распознающие речь человека. Они анализируют оцифрованный звук, сравнивая фрагменты записи с образцами звуков речи заданного языка, распознают произнесённые слова и записывают их в виде текста.

Ввести текст таким образом можно в браузере *Google Chrome* на сайте **speechpad.ru**. Качество распознавания зависит от качества записи и разборчивости речи и может достигать 80–90%.

Гиперссылки

Любую часть текста в текстовых документах можно сделать гиперссылкой. **Гиперссылка** может указывать на другой документ, расположенный на том же компьютере или в Интернете.

Выделите какое-нибудь слово в тексте и создайте гиперссылку на сайт **www.yandex.ru**, выбрав пункт меню *Вставка* → *Гиперссылка* или нажав комбинацию клавиш *Ctrl+K*.

Часто при вводе адреса сайта программа сама превращает его в гиперссылку.

Попробуйте открыть гиперссылку, щёлкнув на ней при нажатой клавише *Ctrl*.

Точно так же можно ставить внутренние ссылки на какой-то заголовок того же самого документа или на **закладки** — любые точки документа, которые помечены специальным образом. Добавить новую закладку можно с помощью меню *Вставка* → *Закладка*.

Выводы

- Текстовые процессоры могут проверять орфографию и грамматику текста. Для этого используется встроенный словарь и правила построения предложений.
- Если программа не обнаруживает ошибку, это ещё не доказывает, что слово или предложение написано правильно.
- Тезаурус (в текстовом процессоре) — это словарь, который содержит синонимы, антонимы и родственные слова.
- Компьютерные двуязычные словари позволяют найти различные значения слова, позволяют услышать его произношение.
- Задача компьютерного перевода текста — это одна из задач искусственного интеллекта. Пока программам-переводчикам нельзя доверять в ответственных случаях.
- Оптическое распознавание текста — это преобразование отсканированного изображения в текстовый документ. В некоторых случаях программы могут распознавать даже рукописный текст.

Интеллект-карта

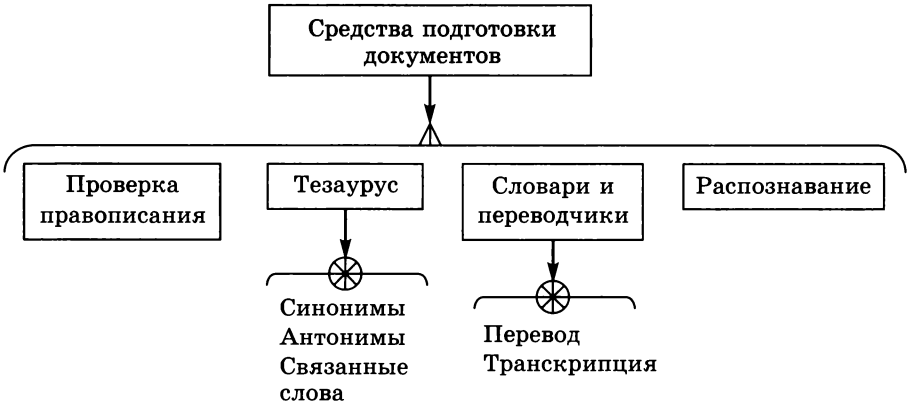


Рис. 5.3

Вопросы и задания

1. Обсудите в классе такую идею: «Не нужно знать грамматику, программа покажет все ошибки». Что вы думаете по этому поводу?
2. Чем отличается проверка грамматики от проверки орфографии? Какая из этих задач сложнее?
3. Почему задача распознавания текста относится к области искусственного интеллекта?

4. Почему сложно распознавать рукописный текст? Как вы думаете, будет ли когда-нибудь решена эта задача?
5. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Как проверяют орфографию?»
б) «Как проверяют грамматику?»
в) «Компьютерный перевод текстов – за и против»
г) «Распознавание символов»

Интересные сайты

abbyy.ru — сайт компании *ABBYY*
cognitiveforms.com — бесплатная программа для распознавания текста *CuneiForm*
lingvo-online.ru/ru — онлайн-словарь *Lingvo*
translate.ru — онлайн-переводчик *Prompt*
translate.yandex.ru — онлайн-переводчик *Яндекс*
translate.google.com — онлайн-переводчик *Google*
newocr.com — распознавание текста онлайн
free-ocr.com — распознавание текста онлайн
ocronline.com — распознавание текста онлайн
speechpad.ru — голосовой ввод текста в браузере *Google Chrome*

Практические работы

Выполните практические работы:
№ 32 «Работа с текстом»;
№ 33 «Распознавание текстов».

§ 30
Математические тексты

Ключевые слова:

- формула
- дробь
- верхний индекс
- нижний индекс

В электронных документах по математическим предметам (например, по математике или физике) встречаются формулы, которые желательно оформить красиво и в одинаковом стиле.

Простые формулы можно набирать прямо в тексте, используя стили оформления, принятые для формул. Обычно в формулах

используется шрифт с засечками (например, *Times New Roman*), имена переменных выделяются курсивом. Оформить выделенные символы как нижний или верхний индекс в *Microsoft Word* можно с помощью кнопок $\boxed{x_2}$ и $\boxed{x^2}$ на панели *Главная*. В *OpenOffice Writer* для этой цели используется меню *Формат* → *Символ*.

Греческие буквы (α, β, γ и др.) и математические знаки (≥, ≤, ≈, ≠, ∈) входят в состав шрифта *Symbol*, их можно вставить в документ с помощью меню *Вставка* → *Символ*.

Какие из этих формул невозможно набрать с помощью обычных приёмов форматирования текста?

а) $a^2 + b^2 = c^2$; б) $a = \frac{F}{m}$; в) $(\alpha^2 + \beta^2) = 1$; г) $\begin{cases} x - y = 1 \\ x + y = 5 \end{cases}$.

Как вы рассуждали?

Сложные (например, «многоэтажные») формулы ввести простым способом не удаётся, поэтому в современных текстовых процессорах есть специальные средства для набора формул.

Набор формул (*Microsoft Word*)

Наибольшими возможностями среди текстовых процессоров обладает программа *Microsoft Word*. Формула в *Word* редактируется в графическом режиме, так что сразу видно, как она будет выглядеть при печати документа.

Используя дополнительные источники, выясните, от каких слов произошло сокращение *WYSIWYG* и что оно обозначает.

На вкладке *Вставка* есть кнопка π *Уравнение*, после щелчка на ней в позиции курсора вставляется поле для объекта-формулы (рис. 5.4).

Рис. 5.4

Когда выделено это поле, появляется дополнительная вкладка *Конструктор* для ввода элементов формул (рис. 5.5).

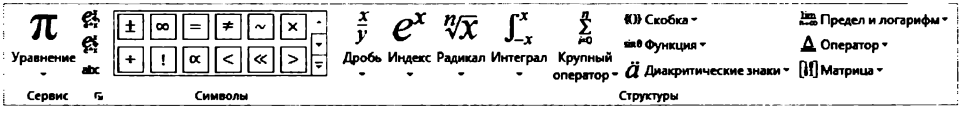


Рис. 5.5

Буквы и цифры в формулах набираются на клавиатуре обычным способом. С помощью панели *Символы* можно вставить математические символы, которых нет на клавиатуре, например знаки «±» и «≠». Меню *Дробь* содержит шаблоны для дробей разного стиля (числитель и знаменатель разделены чертой): $\frac{a}{b}$, a/b или a/b. Меню *Индекс* служит для добавления верхних и нижних индексов, например x^2, x_2 или x_1^2. С помощью меню *Скобка* можно добавить «растягивающиеся» скобки, которые автоматически увеличиваются так, чтобы вместить всё содержимое.

Для примера покажем, как набрать формулу для вычисления корней квадратного уравнения $ax^2 + bx + c = 0$:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Добавим формулу и сразу вставим поле с нижним индексом $\square_{\square}$ с помощью меню *Индекс*. В первую выделенную точками область заносим «x», а во вторую — индекс «1,2» (рис. 5.6).

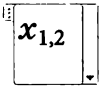


Рис. 5.6

Теперь нужно выйти из области нижнего индекса, нажав на клавишу → на клавиатуре. Набираем на клавиатуре знак «=», затем вставляем дробь $\frac{\square}{\square}$ (рис. 5.7).

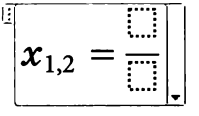


Рис. 5.7

Выделяем мышью «окно» числителя $\square$, сначала набираем «$-b$», затем вставляем знак «±» с помощью панели *Символы*, затем знак квадратного корня $\sqrt{\square}$ из панели *Радикал* (рис. 5.8).

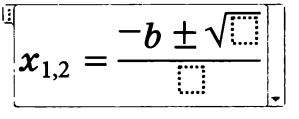


Рис. 5.8

Дальше заполняем оставшиеся две пустые области, используя те же приёмы.

Набор формул (*OpenOffice Math*)

В программе *OpenOffice Writer* можно вставить формулу с помощью редактора *Math*. После выбора пункта меню *Вставка* → *Объект* → *Формула* в нижней части окна редактора появляется область, в которой вводится текстовое описание формулы, и всплывающая панель *Элементы* для выбора элементов — дробей, степеней, корней и др. В основной части документа мы видим формулу так, как она будет напечатана (рис. 5.9).

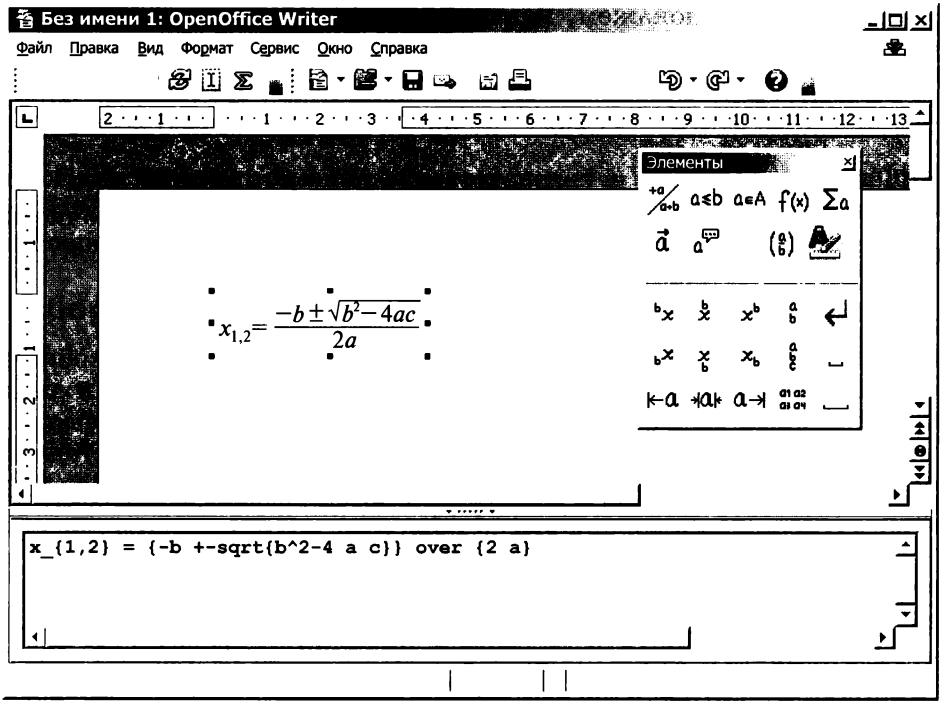


Рис. 5.9

Наберём ту же самую формулу, которую мы вводили в *Word*. Вставим объект-формулу, на панели *Элементы* в верхней части выберем группу *Формат*, а затем в нижней части панели из этой группы выберем элемент x_b (нижний индекс). После этого в окне редактирования формулы мы увидим шаблон

<?>_{<?>}

Вместо первого сочетания <?> нужно вписать x, а вместо второго (в фигурных скобках) — нижний индекс «1,2»:

```
x_{1,2}
```

Вы сразу увидите, что изображение формулы в окне редактора изменилось. Теперь вводим знак «=» и выбираем на панели *Элементы* дробь $\frac{a}{b}$ из группы *Операторы*:

```
x_{1,2} = {<?>} over {<?>}
```

Первая комбинация <?> — это место для числителя, после слова *over* (в переводе с англ. — «над») в фигурных скобках нужно записать знаменатель. Начинаем вводить числитель: сначала «$-b$», затем знак «±» из группы *Операторы*. Остальная часть числителя стоит под знаком квадратного корня, вставляем его из группы *Функции* и видим, что квадратный корень обозначается словом *sqrt*, а подкоренное выражение записывают в фигурных скобках:

```
x_{1,2} = {-b +-sqrt{<?>} } over {<?>}
```

Вводим выражение до конца, заполняя все пустые поля. Для ввода «b^2» используем элемент x^b из группы *Формат*:

```
x_{1,2} = {-b +-sqrt{b^2 - 4 a c} } over {2 a}
```

Обратите внимание, что между отдельными элементами выражения (например, между 4, a и c) нужно ставить пробелы.

Используя меню *Сервис* → *Каталог*, выясните, как вставляются в формулу буквы греческого алфавита.

Итак, мы познакомились ещё с одним подходом к набору формул: формула вводится как простой текст с помощью условных обозначений (например, over, sqrt). Фактически мы создали программу, по которой строится изображение формулы при выводе на экран.

Такой же метод используется в самой популярной и совершенной системе для набора математических текстов, которая называется $\mathrm{T_EX}$.

Запишите в тетради текстовое описание формулы для вычисления коэффициента полезного действия (КПД) теплового двигателя: $\eta = \frac{Q_1 - Q_2}{Q_1} \cdot 100\%$. Проверьте ваш вариант в программе.

Система TEX

Система TEX — это система компьютерной вёрстки статей, книг и презентаций. Её придумал и разработал американский математик Дональд Кнут.

Используя дополнительные источники, выясните, какую всемирно известную книгу по программированию написал Дональд Кнут.

Дональд Кнут (род. в 1938 г.)

Одно из главных достоинств TEX — детально разработанная система набора сложных математических формул. Поэтому TEX очень популярен среди математиков и физиков, а также в издательствах, которые выпускают математическую литературу.

Система TEX — кроссплатформенное свободное программное обеспечение. Она выдаёт одинаковый результат на всех компьютерах под управлением всех операционных систем.

Большинство систем компьютерной вёрстки при наборе документа сразу показывают результат, который получится при печати (используют режим *WYSIWYG*). В TEX используется другой принцип: автор задаёт только текст и его структуру (заголовки, параграфы, списки, таблицы), а затем программа самостоятельно форматирует документ.

Исходный документ TEX — это обычный текстовый файл с расширением *tex*. Программа TEX обрабатывает этот файл и в результате готовит документ с расширением *dvi* (англ. *device independent* — независимый от устройства), который можно вывести на экран или преобразовать в формат PDF.

Чаще всего используют TEX с пакетами расширений LaTEX или AMS-TEX, которые значительно упрощают набор документа. Мы кратко познакомимся с пакетом LaTEX.

Документ LaTEX состоит из заголовка (вступительной части — *преамбулы*) и основного текста. **Все команды TEX начинаются с символа «\».** Заголовок начинается с объявления класса документа:

```
\documentclass{article}
```

Команда `\documentclass` **задаёт класс документа, в фигурных скобках указан аргумент команды: слово** `article` **обозначает статью. Есть и другие классы документов, например** `report` **— отчёт,** `book` **— книга,** `slides` **— слайды.**

В заголовке нужно определить язык, на котором написан документ (`russian` — русский):

```
\usepackage[russian]{babel}
```

и кодировку, в которой закодирован текст (здесь — UTF-8):

```
\usepackage[utf8]{inputenc}
```

Основное содержание документа записывается между командами

```
\begin{document}
```

и

```
\end{document}
```

Главное достоинство TEX — это удобный набор формул. Формулы в тексте выделяются символами «$», например: `$a^2+b^2=c^2$`. Здесь (как и в *OpenOffice Math*) значок «^» обозначает верхний индекс. Формула, занимающая отдельную строку, ограничивается с двух сторон двумя знаками «$»:

```
$$ a^2 + b^2 = c^2. $$
```

Некоторые примеры формул и обозначений приведены в табл. 4.1.

Таблица 4.1

Элемент формулы	Как набрать?	Результат
Верхний индекс (степень)	`$x^2,  x^{y+1}$`	x^2, x^{y+1}
Нижний индекс	`$x_2,  x_{y+1}$`	x_2, x_{y+1}
Верхний нижний индексы	`$x_1^2$`	x_1^2
Квадратный корень	`$\sqrt{a+b}$`	$\sqrt{a+b}$
Дробь	`$\frac{a+b}{2}$`	$\frac{a+b}{2}$
Высокие скобки	`$\left(\frac{1}{x}\right)^n$`	$\left(\frac{1}{x}\right)^n$
Знак умножения	`$x_1  \cdot  x_2$`	$x_1 \cdot x_2$
Горизонтальный интервал	`$x  \quad  y$`	$x \quad y$

Информацию о других возможностях LaTEX вы можете найти в литературе или в Интернете.

Используя дополнительные источники, выясните, как оформляется тире в TEX.

Вот пример готового документа:

```
\documentclass{article}
\usepackage[russian]{babel}
\usepackage[utf8]{inputenc}
\begin{document}
\textbf{Теорема Пифагора.} Пусть $a$ и $b$ --
катеты прямоугольного треугольника, а $c$ --
его гипотенуза.
Тогда выполняется равенство:
$$
a^2 + b^2 = c^2.
$$
\end{document}
```

Здесь использована новая команда `\textbf`, которая выделяет жирным шрифтом текст, записанный в фигурных скобках. Для того чтобы выделить текст курсивом, применяют команду `\textit`.

Если этот документ обработать программой LaTEX (например, в онлайн-редакторе на сайте **www.overleaf.com**), получим следующий результат (рис. 5.10).

Теорема Пифагора. Пусть a и b — катеты прямоугольного треугольника, а c — его гипотенуза. Тогда выполняется равенство:

$$a^2 + b^2 = c^2$$

Рис. 5.10

Диаграммы

При подготовке математических (да и не только математических!) текстов часто нужно добавить в документ график какой-нибудь функции или диаграмму.

Вставить диаграмму в программе *Word* можно с помощью кнопки *Диаграмма* на вкладке *Главная*. Программа предлагает выбрать тип диаграммы и добавляет в документ новую диаграмму с некоторыми данными. Одновременно открывается окно для редактирования этих данных (запускается программа *Excel*) — рис. 5.11.

	A	B	C	D	E
1		Ряд 1	Ряд 2	Ряд 3	
2	Категория 1	4,3	2,4	2	
3	Категория 2	2,5	4,4	2	
4	Категория 3	3,5	1,8	3	
5	Категория 4	4,5	2,8	5	
6					

Рис. 5.11

Диапазон, по которому строится диаграмма, выделен цветной рамкой, её правый нижний угол можно перетаскивать мышью, таким образом добавляя или удаляя строки и столбцы. Все данные в таблице (числа и надписи) тоже можно изменять так, как вы хотите.

Вспомните, из каких элементов состоит диаграмма.

Когда выделена диаграмма, на *Ленте* появляются вкладки *Конструктор*, *Макет* и *Формат*. С их помощью можно настроить внешний вид диаграммы, например добавить название диаграммы и осей (вкладка *Макет*).

Кнопка *Строка/столбец* на вкладке *Конструктор* позволяет выбрать, как расположены ряды данных — в строках или в столбцах. Если окно с данными больше не нужно, его можно закрыть. Вернуться к редактированию данных можно с помощью кнопки *Изменить данные*.

Чтобы изменить оформление выделенного элемента, используется вкладка *Формат* или контекстное меню. Например, для изменения размера шрифта какой-нибудь оси нужно щёлкнуть на ней правой кнопкой мыши и выбрать команду *Формат оси*.

Для вставки диаграммы в *OpenOffice Writer* используют меню *Вставка* → *Объект* → *Диаграмма*.

Выясните экспериментально, какая диаграмма добавляется по умолчанию. Как она называется?

Диаграмму в *Writer* настраивают с помощью панели инструментов *Форматирование*, которая появляется тогда, когда диаграмма выделена (рис. 5.12).

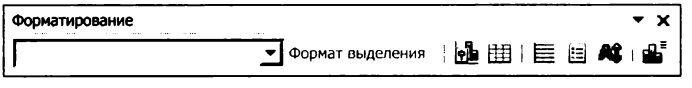


Рис. 5.12

Удобно также использовать контекстное меню и меню *Формат*.

Исследуйте и запишите в тетрадь назначение всех кнопок панели *Форматирование*.

Выводы

- Все формулы в документах должны быть набраны в одном стиле, принятом в математической литературе.
- Для набора формул можно использовать встроенные средства текстовых процессоров.
- Существует два способа ввода формул: 1) формула редактируется в том виде, в каком она выглядит на печати (режим *WYSIWYG*); 2) формула редактируется в виде текста, содержащего специальные команды.
- Для набора больших математических текстов чаще всего применяют систему TEX. Исходный файл TEX — это простой текст без оформления, формулы редактируются в текстовом виде.
- Диаграмма, внедрённая в текстовый документ, хранится вместе с данными. При изменении этих данных диаграмма перестраивается автоматически.

Интеллект-карта

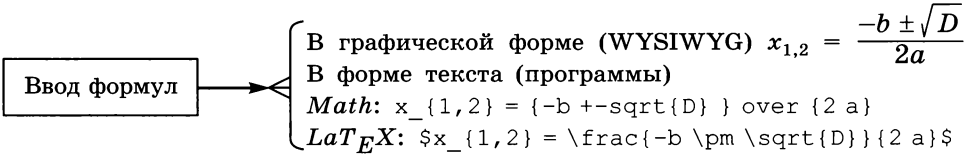


Рис. 5.13

Вопросы и задания

1. Сравните два способа набора формул. Обсудите достоинства и недостатки каждого из них.
2. Составьте алгоритм набора формулы $x = \sqrt{\frac{a^2 - b^2}{2c}}$ в текстовом процессоре, с которым вы работаете.
3. Обсудите в классе достоинства и недостатки системы TEX.
4. Оформите в текстовом процессоре небольшое сообщение по математике или физике, которое содержит формулы.
5. Выполните задание 4 с помощью системы LaTEX, используя онлайн-редакторы.

6. Узнайте, что означает математический знак Σ (греческая прописная буква «сигма») и как набирать формулы, содержащие этот знак.
7. Используя материалы Интернета, выясните, как оформляются списки и таблицы в LaTeX.
8. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Списки в LaTeX»
б) «Таблицы в LaTeX»

Интересные сайты

overleaf.com — онлайн-редактор LaTeX
ru.sharelatex.com — онлайн-редактор LaTeX с возможностью совместной работы

Практические работы

Выполните практические работы:
№ 34 «Математические тексты»;
№ 35 «Набор текстов в LaTeX».

§ 31
Многостраничные документы

Ключевые слова:

- формат страницы
- ориентация листа
- поля страницы
- колонтитулы
- оглавление

В этом параграфе мы научимся работать с большими документами (размером более 10–15 страниц):

- определять формат страницы (размер бумаги, поля);
- добавлять нумерацию страниц;
- сохранять единый стиль оформления всего документа;
- строить оглавление.

Грамотно используя возможности тестовых процессоров, можно значительно ускорить работу и подготовить профессионально оформленный документ.

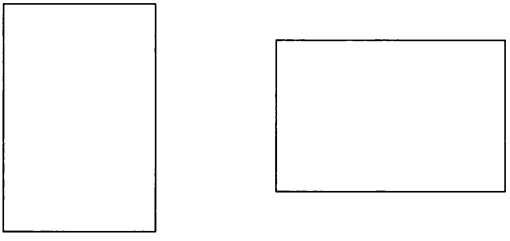
Форматирование страниц

Прежде всего нужно настроить **формат страницы**: выбрать размер бумаги, ориентацию листа, установить поля. В программе *Word* эти свойства настраиваются на вкладке *Разметка страницы*, а в *OpenOffice Writer* — в окне *Формат* → *Страница* (вкладка *Страница*).

Размер бумаги можно выбрать из стандартных форматов или задать свой собственный — определить длину и ширину страницы в сантиметрах.

Используя диалоговые окна текстового процессора, выясните, какие размеры имеют листы форматов A3, A4 и A5.

Ориентация листа выбирается из двух вариантов (рис. 5.14).



Книжная ориентация | Альбомная ориентация

Рис. 5.14

Поля — это свободные области по краям страницы. У многих принтеров область печати меньше, чем полный размер листа, поэтому в любом случае поля слева или справа будут оставлены, даже если вы установите для них нулевые значения. Кроме того, читать документ без полей очень неудобно. Часто рекомендуют оставлять поле слева не менее 3 см, справа — не менее 1 см, сверху и снизу — не менее 2 см.

Для документов, которые печатаются с двух сторон листа бумаги и потом сшиваются, обычно устанавливают *зеркальные поля*. Это значит, что для страниц с чётными номерами левое и правое поля меняются местами.

Колонтитулы

Колонтитулы — это информация, которая помещается над и под текстом каждой страницы.

Вы можете добавить верхние и нижние колонтитулы. В колонтитулах помещают названия глав и параграфов книг, фамилии авторов, номера страниц (рис. 5.15).

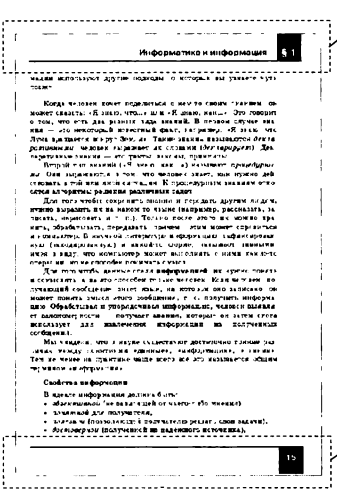


Рис. 5.15

Колонтитулы могут быть разные на чётных и нечётных страницах. Например, в колонтитуле чётных страниц может выводиться название главы, а в колонтитуле нечётных страниц — название параграфа книги. Колонтитул первой страницы также может отличаться от всех остальных.

Определите, какие данные находятся в колонтитулах в этом учебнике.

Для того чтобы добавить информацию в колонтитулы, в программе *Word* нужно выполнить двойной щелчок на верхнем или нижнем поле страницы (или использовать кнопки *Верхний колонтитул* и *Нижний колонтитул* на вкладке *Вставка*).

В *OpenOffice Writer* колонтитулы включаются с помощью окна *Формат* → *Страница*.

Чаще всего в колонтитулах помещают номер страницы. В программе *Word* для этого используется кнопка *Номер страницы* на вкладке *Вставка*, а в *OpenOffice Writer* — меню *Вставка* → *Поля* → *Номер страницы* (курсор нужно поставить в колонтитул).

На первой (титульной) странице больших документов, например рефератов или отчётов, номер обычно не ставят.

Оглавление

Для того чтобы облегчить чтение больших документов, в них выделяют разделы и подразделы. Они показывают **структуру документа** — его составные части. Чтобы нужный раздел было легче искать, строят **оглавление** — список всех заголовков разделов и подразделов с указанием страниц, где эти разделы начинаются (рис. 5.16). Оглавление обычно расположено в начале или в конце документа.

Рис. 5.16

С какими проблемами вы можете столкнуться, если будете строить оглавление вручную?

Давайте подумаем: как программа может обнаружить заголовок? Как она отличит заголовок раздела от заголовка подраздела? Для этого нужно выделять заголовки **стилями** (вспомните материал учебника для 7 класса). Стандартные стили, которые автоматически определяются при составлении оглавления, называются *Заголовок 1*, *Заголовок 2*, *Заголовок 3* и т. д. Поэтому все заголовки разделов лучше сразу оформлять стилем *Заголовок 1*, заголовки подразделов — стилем *Заголовок 2* и т. д.

Когда все заголовки оформлены нужными стилями, можно строить оглавление. Курсор нужно установить в то место, куда надо вставить оглавление. Затем в программе *Word* используем кнопку *Оглавление* на вкладке *Ссылки*, а в *OpenOffice Writer* — пункт меню *Вставка* → *Оглавление и указатели*.

Оглавление — это особая часть документа, которая использует данные из других частей и может обновляться. Такие обновляемые элементы документа называются *полями*. Для того чтобы обновить поле, можно применить команду *Обновить поле* из контекстного меню[1].

Выводы

- Большие документы обычно разбивают на разделы и подразделы.
- Все части документа должны быть оформлены в едином стиле.
- Настройка формата страницы включает выбор размера бумаги, ориентации листа, размеров полей.
- Заголовки разделов и подразделов оформляются стилями *Заголовок 1*, *Заголовок 2* и т. д. Из абзацев, оформленных этими стилями, автоматически собирается оглавление.
- Колонтитулы — это информация, которая помещается над и под текстом каждой страницы. В колонтитулах выводятся названия книг, глав и параграфов, фамилии авторов, номера страниц.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы рассуждаете, когда выбираете ориентацию листа (книжную или альбомную)?
2. Зачем нужны поля страницы?
3. Какую информацию, на ваш взгляд, можно помещать в колонтитулы (кроме той, которая приведена в тексте параграфа)?
4. Зачем создают разные колонтитулы и поля у страниц с чётными и нечётными номерами?
5. Обсудите, где лучше строить оглавление — в начале или в конце документа.
6. Оформите какой-нибудь достаточно большой документ, включающий несколько разделов и подразделов. Добавьте колонтитулы, номера страниц, заголовки, оглавление.

[1] В программе *Word* можно также использовать клавишу *F9*.

7. Выясните, как в текстовом процессоре форматировать часть текста в несколько колонок. Где это может понадобиться?
8. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Набор текста в несколько колонок»
б) «Сноски и ссылки в документе»
в) «Нумерация рисунков и таблиц»
г) «Нумерация формул»

Практическая работа

Выполните практическую работу № 36 «Многостраничный документ».

§ 32
Правила оформления рефератов

Ключевые слова:

- реферат
- аннотация
- титульный лист
- правила оформления
- список использованных источников

Реферат — это письменный доклад (сообщение) по определённой теме, в котором представлена информация из одного или нескольких источников.

Автор реферата должен исследовать выбранную тему, переработать найденную информацию и сделать выводы, а не просто списать текст с сайтов в Интернете или из книжек.

Реферат должен читаться как единый связный текст, написанный автором. Если вы используете цитаты из разных источников, их нужно состыковать. Возможно, вам придётся дописать связующие фразы и переписать какие-то абзацы своими словами.

Используя дополнительные источники, выясните, от какого слова произошло слово «реферат».

Структура реферата

Реферат обычно содержит:

- титульный лист;
- содержание (оглавление);
- аннотация;
- введение (1–2 страницы);
- основную часть (10–15 страниц);
- выводы или заключение (1–2 страницы);
- список использованных источников.

Каждая часть начинается с новой страницы. Страницы нумеруются, на втором листе реферата строится **содержание (оглавление)** с указанием номеров страниц.

Аннотация — это краткая характеристика реферата (обычно объёмом до 500 знаков). В аннотации нужно чётко определить проблему и цели работы, описать полученные результаты. В аннотации к научной статье обязательно указывают, что нового содержится в этой работе по сравнению с другими, какое значение могут иметь её результаты для науки, где их можно применить на практике.

В **введении** вы должны написать, почему выбрана такая тема, чем она важна для вас, чем актуальна для других, какую культурную или научную ценность она представляет. Здесь нужно поставить вопросы, на которые вы хотите ответить в конце реферата (в выводах). Часто бывает удобно писать введение уже после того, как реферат готов.

Основная часть реферата состоит из нескольких разделов, в которых постепенно раскрывается тема. Основные мысли подкрепляются доказательствами, взятыми из литературы. В конце каждого раздела основной части обязательно формулируется вывод. Обычно в рефератах используют два уровня заголовков — разделы и подразделы.

Выводы — это ваши мысли о том, что вы узнали при изучении темы. В выводах вы должны ответить на вопросы, поставленные во введении, и высказать своё мнение о проблеме, рассмотренной в реферате.

Список использованных источников содержит названия всех книг, статей и адреса веб-страниц, откуда вы брали информацию для реферата.

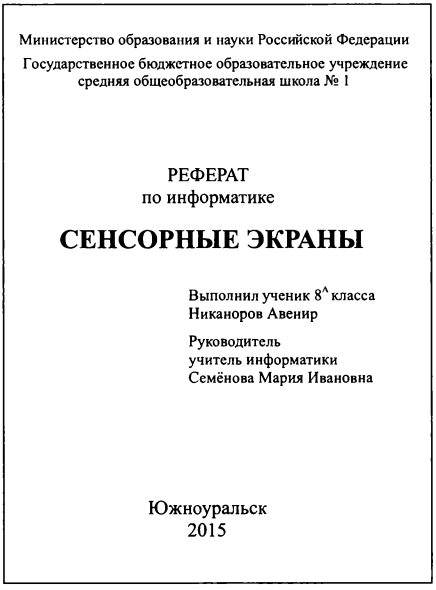
Титульный лист

Титульный лист — это первый лист реферата. На нём должны быть:

- название министерства (Министерство образования и науки Российской Федерации);

- название организации (школы, лицея);
- слово «реферат», название предмета;
- название реферата;
- фамилия и имя автора;
- фамилия, имя и отчество руководителя;
- в последних двух строчках — город и год.

Пример правильного оформления титульного листа показан на рис. 5.17.


Министерство образования и науки Российской Федерации
Государственное бюджетное образовательное учреждение
средняя общеобразовательная школа № 1

РЕФЕРАТ
по информатике

СЕНСОРНЫЕ ЭКРАНЫ

Выполнил ученик 8^ класса
Никаноров Авенир

Руководитель
учитель информатики
Семёнова Мария Ивановна

Южноуральск
2015

Рис. 5.17

Оформление текста

Для реферата используются два типа шрифтов: для основного текста — шрифт с засечками (Times New Roman, Cambria, Garamond), для заголовков — рубленый шрифт (Arial, Calibri, Pragmatica).

Заголовки разных уровней выделяются стилями *Заголовок 1*, *Заголовок 2* и т. д., из которых потом автоматически строится оглавление. Заголовки выделяются жирным шрифтом. Обычно для заголовков первого уровня используют шрифт размером 16 пунктов, для заголовков второго — шрифт размером 14 пунктов, для заголовков третьего уровня — шрифт размером 14 пунктов, курсив. Точку в конце заголовка не ставят.

Для шрифта основного текста, как правило, устанавливается размер 14 пунктов, полуторный интервал (для того, чтобы было легче читать), выравнивание по ширине.

Текст обязательно разбивается на абзацы, каждый абзац выражает самостоятельную законченную мысль. По правилам, принятым в России, каждый абзац начинается с красной строки.

Для рефератов используется формат А4, книжная ориентация страницы, поле слева — 3 см, поля сверху и снизу — по 2 см, поле справа — 1,5 см.

Список использованных источников

В списке использованных источников перечисляются все материалы, использованные при составлении реферата: книги, статьи, интернет-сайты, электронные ресурсы и др. Обычно они сортируются в алфавитном порядке по фамилии первого автора, а работы одного автора — по возрастанию года издания. Ссылки на интернет-ресурсы записывают в конце списка.

Приведём примеры правильного оформления элементов списка использованных источников:

Книга:

1. *Маслов Л. А.* Нашествия инопланетян на Европу [Текст] / Л.А. Маслов. — СПб.: НЛОиздат, 2001. — 344 с.

Статья в журнале:

2. *Васильев А. Н.* О глобальных физических проблемах современной астрономии [Текст] / А. Н. Васильев, А. Л. Петров, М. Д. Сидоренко // Вестн. Моск. ун-та. Сер. 3, Физика. Астрономия. — 2011. — № 6. — С. 43–45.

Электронный документ в Интернете:

3. *Артемьев К. С.* Развитие самосознания в эпоху раннего неолита [Электронный ресурс] // Вестн. НИУГУ. 2012. № 3. URL: http://www.niugu.ru/download/1238.pdf (дата обращения: 20.11.2015).

Статья на сайте в Интернете:

4. Археологи узнали о влиянии ячменя на судьбу Тибета // LENTA.RU: ежедн. интернет-изд. 2014. 21 ноября. URL: http://lenta.ru/news/2014/11/21/tibet/.

Сайт целиком:

5. Официальный сайт Государственного Эрмитажа // Санкт-Петербург, 2014. URL: http://www.hermitagemuseum.org (дата обращения: 20.11.2015)

Ссылки на использованные источники в тексте реферата заключают в квадратные скобки, например: «Как отмечал К. А. Мясоедов [5], проблема значительно шире».

Можно точно указать страницу, на которую вы ссылаетесь: «Как известно, лошади едят сено [8; с. 234].».

Ссылки необходимо делать всегда, когда вы приводите слова других людей, а также данные, результаты, диаграммы, графики, рисунки,

сделанные не вами. Это обеспечивает соблюдение закона по охране авторского права и повышает доверие к вашему тексту.

Используя дополнительные источники, выясните, что означают термины «конспект», «аннотация», «резюме».

Выводы

- Реферат — это письменный доклад (сообщение) по определённой теме, в котором представлена информация из одного или нескольких источников. Реферат должен читаться как единый связный текст, написанный его автором.
- Реферат обычно содержит титульный лист, содержание, аннотацию, введение, основную часть, выводы и список использованных источников.
- Рефераты оформляются в соответствии с правилами оформления научных и исследовательских работ.
- Если вы приводите слова других людей, а также данные, результаты, диаграммы, графики, рисунки, сделанные не вами, обязательно дать ссылку на источник информации.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Обсудите основные сложности, которые возникают у ваших одноклассников при оформлении рефератов.
2. Зачем нужен титульный лист реферата?
3. Как вы думаете, почему оглавление в реферате помещают в начало документа, а во многих книгах — в конец?
4. Оформите реферат по одному из предметов. Добавьте колонтитулы, номера страниц, заголовки, оглавление.
5. Узнайте с помощью литературы и ресурсов Интернета, что такое реферативный журнал.
6. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

«Что такое реферат статьи?»

Практическая работа

Выполните практическую работу № 37 «Оформление реферата».

§ 33
Коллективная работа над документами

Ключевые слова:

- рецензирование
- примечание
- исправления

Очень часто над документом работает несколько человек. Один пишет начальный вариант, другие комментируют его, высказывают замечания и предложения по улучшению. Это называется коллективной работой над документом. При этом важно, чтобы была возможность восстановить предыдущую версию в том случае, если изменения были внесены ошибочно или в результате изменений результат (текст) стал хуже. В этом параграфе вы узнаете, какие средства можно использовать для решения этой задачи.

Рецензирование

Начнём с простой задачи: у вас есть готовый текст и нужно высказать замечания по нему, предложить исправления. Такую процедуру называют **рецензированием**, а человека, выполняющего эту работу, — **рецензентом**.

Текст на бумаге очень легко исправить — вычеркнуть ненужный фрагмент текста и вписать сверху нужный, записать замечания и вопросы к автору на полях страницы. Важно, что при такой правке виден и исходный текст, и все исправления. Так раньше работали редакторы и корректоры в издательствах[1]. Но сейчас можно и нужно делать такие исправления в электронном виде.

Какие правки вы бы сделали в этом тексте?

Хорошо, если платье твоё твоё без прорех.
И хлебе насущном падумать не грех.
А всиго остального и даром надо —
Жизнь дороже богатства и почестей всех.

Омар Хайям.

[1] Редактор — это человек, который помогает автору улучшить текст путём изменения содержания документа (например, научный редактор). Корректор исправляет ошибки в тексте (грамматические, стилистические и др.).

Конечно, можно выделить (цветом или маркером) фрагменты текста, которые вызывают вопросы. Но это не очень удобно, потому что непонятно, что имел в виду тот, кто выделил текст. Комментарии можно вставлять прямо в текст, но тогда придётся выделять их цветом, чтобы они были заметны другим авторам. Лучше использовать для этой цели специальные средства текстовых процессоров.

На *Ленте* программы *Word* есть вкладка *Рецензирование*, с помощью которой можно вносить исправления и писать примечания (комментарии) — рис. 5.18.

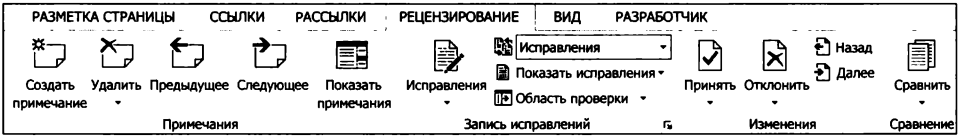


Рис. 5.18

Если выделить часть текста и щёлкнуть на кнопке *Создать примечание*, то будет создано примечание на полях страницы (рис. 5.19).

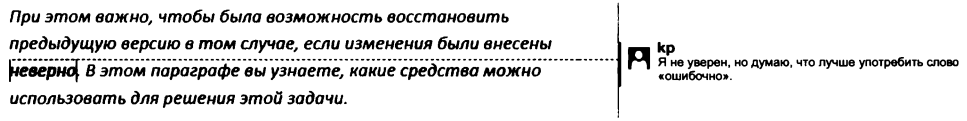


Рис. 5.19

Над текстом примечания показаны инициалы автора (из настроек программы).

Кнопка *Исправления* включает (а при повторном нажатии — отключает) режим исправлений. Это означает, что все ваши действия по исправлению документа будут записываться программой и на экране будут показаны как исходный, так и исправленный варианты текста (рис. 5.20).

При этом важно, чтобы была возмржность восстановить предыдущую версию в том случае, если изменения были внесены ~~неверно~~ошибочно. В этом параграфе вы узнаете, какие средства можно использовать для решения этой задачи.

Рис. 5.20

С помощью кнопок панели *Изменения* (см. рис. 5.18) можно принять или отклонить примечания. Для этой цели также можно использовать команды контекстного меню.

Рецензирование в программе *OpenOffice Writer* выполняется точно так же, как и в *Word*. Примечания вставляются с помощью меню *Вставка* → *Примечание*, а для работы в режиме исправлений нужно использовать меню *Правка* → *Изменения*.

Онлайн-офис

Бурное развитие Интернета привело к появлению **онлайн-офисов** — специальных сайтов (интернет-сервисов), которые предоставляют основные возможности офисных пакетов: текстовый редактор, электронные таблицы, средства для создания презентаций. Для использования такой службы необходим компьютер с доступом в Интернет, причем не имеет значения, какая операционная система на нём установлена. Документы пользователей хранятся на сервере, для доступа к ним нужно зайти на сайт под своей учётной записью, которая защищена паролем.

Онлайн-офисы используют технологию, известную под названием **облачные вычисления** (англ. *cloud computing*). Её суть в том, что пользователь размещает свои данные на серверах Интернета и не должен заботиться о способе их хранения, операционной системе и программном обеспечении. Словом «облако» в информатике называется сложная система, детали работы которой знать необязательно (вспомните облачные хранилища данных из курса 7 класса).

Одно из достоинств «онлайн-офисов» — возможность совместной работы над документами через Интернет. Другим пользователям можно открыть доступ к отдельным документам для просмотра и/или изменения. Так вы можете:

- познакомить группу людей с документом;
- организовать его обсуждение через Интернет;
- провести анкетирование;
- вместе редактировать документ.

Любой документ может быть экспортирован (сохранён) в файл на диске компьютера.

Самый известный онлайн-офис — *Google Docs*, или *Документы Google* (**docs.google.com**). Для работы с ним нужно бесплатно зарегистрироваться (получить учётную запись) на сайте

accounts.google.com. Все ваши документы будут храниться в Интернете, в удалённом хранилище, которое называется *Google-диск*. Здесь можно создавать и редактировать текстовые документы, электронные таблицы, презентации, рисунки, формы для проведения анкетирования. Вы будете работать с документами в привычных редакторах, напоминающих программы *Microsoft Office* и *OpenOffice*, но сами документы (и программы для работы с ними!) будут находиться не на вашем компьютере, а на серверах в Интернете.

Вспомните достоинства и недостатки хранения документов в облачных хранилищах.

По умолчанию к любому документу имеет доступ только его автор. Если вы хотите пригласить к совместной работе других пользователей, нужно изменить уровень доступа к документу. Вы можете открыть доступ:

- для всех пользователей Интернета;
- только для тех, у кого есть точная ссылка;
- только для тех, кому отправлено приглашение по электронной почте.

Для всех, кому вы высылали персональное приглашение, можно установить три уровня доступа:

- только чтение;
- чтение и комментирование;
- редактирование.

В любой момент вы видите, сколько пользователей работает с документом. С ними можно общаться с помощью чата, и даже можно увидеть, как другие редактируют текст.

Все изменения автоматически сохраняются на сервере. Вы можете просмотреть, какие изменения были внесены в текст, с помощью меню *Файл* → *Просмотреть историю изменений*. В правой части экрана открывается список изменений. Щёлкнув на какой-нибудь строке этого списка, вы увидите версию документа после этих изменений и при желании сможете вернуться к ней (рис. 5.21).

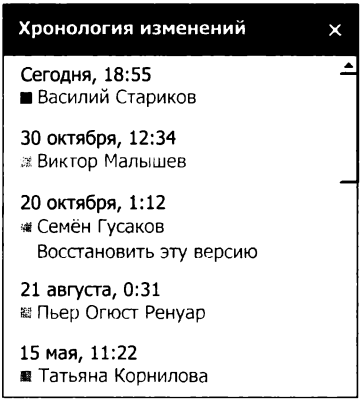


Рис. 5.21

Созданный документ можно сохранить на диске своего компьютера с помощью меню *Файл* → *Скачать как*. При этом программа предлагает выбрать формат, в котором сохраняется документ.

Выясните, в каких форматах можно сохранить на локальном диске:

- текстовый документ;
- электронную таблицу.

С помощью кнопки *Создать* можно загрузить новый документ для совместной работы с вашего компьютера. Если вы загружаете файл в формате *DOCX*, он появляется с иконкой программы *Word*, и редактировать его в таком виде нельзя. При попытке открыть документ система предлагает работать с ним в режиме чтения или перевести в формат *Google-документа* (при этом будет создан новый файл). Если необходимо редактировать текст, нужно выбрать второй вариант. Точно так же можно загружать и редактировать файлы с электронными таблицами и презентациями.

В разделе *Доступны мне* будут храниться те документы, которыми поделились с вами (рис. 5.22).

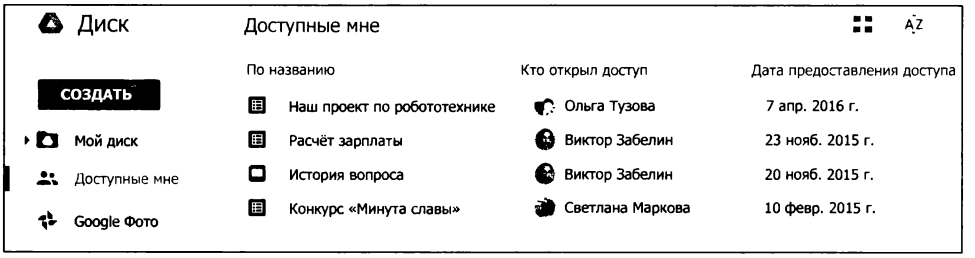


Рис. 5.22

Если автор документа предоставил вам соответствующие права, можно оставлять комментарии (примечания) к документу и редактировать его. Все изменения записываются в журнал и хранятся на сервере.

Правила коллективной работы

Когда вы работали над каким-то документом самостоятельно, вы могли делать с ним, что хотите, и полностью отвечали за результат. При совместной работе могут возникать проблемы, связанные с тем, что кто-то удалил или изменил фразу, написанную другим. Поэтому нужно установить правила, позволяющие избежать конфликтов.

1. Самое важное правило — это уважение к чужому тексту. Ни в коем случае нельзя удалять или изменять фразу, написанную другим, не согласовав это с автором. По всем спорным вопросам решение нужно принимать сообща. Для этого можно задавать вопросы своим соавторам, чтобы выяснить их позицию, использовать комментарии и чат (при работе в режиме «онлайн»).

2. Каждый имеет право на ошибку, и к ним нужно относиться терпимо. Нельзя использовать в комментариях грубые выражения и резкую критику.

3. Все, кто совместно создают документ, являются его авторами, никто из них не может присваивать себе единоличное авторство и представлять документ как результат только своей работы.

4. Коллективная работа требует активного сотрудничества всех участников. Никто не должен делать всё за других, и никто не должен оставаться в стороне, «отмалчиваться». Поэтому желательно, чтобы при совместной работе появился лидер, который будет руководить работой остальных участников.

Выводы

- Рецензирование — это комментирование документа. В текстовых процессорах есть специальные возможности для рецензирования: добавление примечаний и запись исправлений.
- Онлайн-офис — это специальный сайт, предоставляющий основные возможности офисных пакетов: текстовый редактор, электронные таблицы, средства для создания презентаций.

- Сайт *Google Docs* позволяет организовать совместную работу над документами. Автор документа приглашает к обсуждению других людей, которые могут читать, комментировать или редактировать документ (в зависимости от предоставленных им прав). Все изменения записываются, в любой момент можно восстановить любую из предыдущих версий.
- При коллективной работе над документом нужно соблюдать правила, позволяющие избежать конфликтов. Главное из них — уважение к чужому тексту.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Во всех научных журналах предусмотрено рецензирование поступающих статей. Как вы думаете, зачем это нужно?
2. Какие правила нужно соблюдать при коллективной работе с документами?
3. Какие преимущества есть у онлайн-офисов в сравнении с пересылкой документов по электронной почте?
4. В чём вы видите недостатки онлайн-офисов?
5. Чем отличаются режимы доступа «для всех, у кого есть ссылка» и «для всех, кому выслано приглашение»?
6. *Работа в парах.* Найдите в Интернете небольшой текст на любую интересную вам тему. Внесите в него некоторое количество ошибок, которые легко обнаружить. Перешлите полученный текст напарнику и попросите исправить (в режиме исправлений) и передать текст обратно. Кто из вас пропустил меньше ошибок?
7. *Работа в парах.* Узнайте (в литературе или в Интернете), что такое эссе. Напишите небольшое эссе (размером до одной страницы) на любую интересную вам тему. Перешлите его напарнику и попросите оставить комментарии и замечания.
8. *Работа в группах.* Зарегистрируйтесь на сайте **accounts.google.com** (создайте учётную запись — *аккаунт*). Выберите руководителя, который будет организовывать работу группы. Подготовьте совместно какой-либо документ (сообщение, презентацию, рисунок).
9. *Работа в группах.* Составьте анкету по выбранной теме и оформите её в виде формы на сайте документов *Google*. Проведите анкетирование одноклассников, обработайте результаты и представьте их в виде совместного сообщения.
10. Выполните по указанию учителя задания в рабочей тетради.

Подготовьте сообщение

а) «Онлайн-офисы — "за" и "против"»
б) «Облачные хранилища данных»
в) «Облачные сервисы»

Интересные сайты

docs.google.com — документы *Google*
cloud.mail.ru — облачное хранилище данных *Облако@Mail.Ru*

Практическая работа

Выполните практическую работу № 38 «Коллективная работа над документом (проект)».

ЭОР к главе 5 из Единой коллекции цифровых образовательных ресурсов (school-collection.edu.ru)

Диаграммы: столбчатая, контурная, линейная, цилиндрическая
Диаграммы: графическая и графическая накопительная
Орфографическая проверка текста в MS Word
Может ли машина переводить?
Что внутри электронного словаря?

ОГЛАВЛЕНИЕ