ИНФОРМАТИКА

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

10

ИНФОРМАТИКА

Учебное пособие для 10 класса
учреждений общего среднего образования
с русским языком обучения
(с электронными приложениями)

Допущено
Министерством образования
Республики Беларусь

Минск «Народная асвета» 2020

УДК 004(075.3=161.1)
ББК 32.81я721
И74

Авторы:
В. М. Котов, А. И. Лапо, Ю. А. Быкадоров, Е. Н. Войтехович

Рецензенты:
кафедра информационных технологий и моделирования экономических процессов учреждения образования «Белорусский государственный аграрный технический университет» (кандидат педагогических наук, доцент, заведующий кафедрой *О. Л. Сапун*); учитель информатики высшей квалификационной категории государственного учреждения образования «Гимназия № 1 г. Бреста» *И. Ю. Горбацевич*

Электронное приложение «Информационные технологии» (базовый уровень), электронное приложение для повышенного уровня размещены на ресурсе profil.adu.by

ISBN 978-985-03-3168-7

СОДЕРЖАНИЕ

Глава 2

КОМПЬЮТЕР КАК УНИВЕРСАЛЬНОЕ УСТРОЙСТВО ОБРАБОТКИ ИНФОРМАЦИИ

От авторов

Дорогие старшеклассники! Мы живем во время стремительных перемен, когда информатика становится важнейшей составляющей всей системы научного познания, определяет пути формирования информационного общества, основанного на знаниях.

В информатике выделяют такие разделы, как теоретическая и прикладная информатика. В учебном курсе представлены оба эти раздела.

Теоретическая информатика — фундаментальная составляющая информатики как учебного предмета. Общие закономерности протекания информационных процессов, основные понятия логики, теория алгоритмов и языки программирования — разделы, которые позволяют развивать у учащихся логическое и алгоритмическое мышление, воспитывать информационную культуру, формировать научное мировоззрение.

Прикладная информатика направлена на применение понятий и результатов теоретической информатики к решению конкретных задач в конкретных областях. Этот раздел информатики включает в себя архитектуру компьютера и компьютерных сетей, компьютерную графику, компьютерное моделирование, базы данных, информационные технологии и др. Изучение прикладной информатики направлено на формирование компьютерной грамотности — владения необходимым набором знаний и навыков работы на компьютере для обработки, хранения, передачи, получения и использования данных. Самой динамичной составляющей информатики является именно прикладная информатика.

Авторы учебного пособия постарались сделать так, чтобы вы смогли получить необходимые современному человеку знания и навыки в условиях стремительного изменения информационных технологий.

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

— основные материалы, обязательные для изучения;

— примеры, иллюстрирующие основные материалы;

— определения основных понятий;

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

В учебном пособии используются следующие условные обозначения:

— вопросы и задания для проверки знаний;

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

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

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

* — упражнение или пример для любознательных.

В тексте некоторых упражнений вам будет предложено открыть файл. Это означает, что упражнение можно выполнить, используя файл, размещенный на Национальном образовательном портале http://e-vedy.adu.by («Электронное обучение» → «Электронные образовательные ресурсы» → «Информатика» → «Информатика. 10 класс»). Зайти на портал и скачать файлы к упражнениям можно также с помощью матричного QR-кода:

Имя файла для скачивания содержит номер параграфа и номер упражнения. Например, имя файла upr10_3 означает, что файл относится к третьему упражнению десятого параграфа. Также на портале размещены файлы с программами, рассмотренными в примерах. Такие файлы имеют имя pr3_1.pas (программа для примера 3.1).

В учебном пособии размещено много иллюстративного материала. Экранные копии предназначены для ознакомления с интерфейсами программ, для указания расположения отдельных элементов. Подробно рассмотреть все структурные элементы окна используемой программы можно на экране компьютера.

Учебное пособие для базового уровня состоит из двух частей. В первой части, изданной в виде книги, размещены материалы, отражающие фундаментальный характер информатики. В электронном приложении для базового уровня «Информационные технологии» содержатся материалы прикладного характера.

— приложение «Информационные технологии» размещено на электронном образовательном ресурсе (http://profil.adu.by).

— материал для повышенного уровня размещен на электронном образовательном ресурсе (http://profil.adu.by).

ВВЕДЕНИЕ

§ 1. Алгоритм и его свойства

Термин «алгоритм» произошел от фамилии математика IX в. Мухаммеда ибн Муса аль-Хорезми, который сформулировал правила четырех основных арифметических действий. Именно эти правила вначале и назывались алгоритмами, но позже алгоритмом стали называть любой способ вычислений, единый для некоторого класса исходных данных.

Пример 1.1. Алгоритм «Решето Эратосфена» позволяет получить простые числа, не превосходящие N.

1. Выпишем подряд все натуральные числа от 2 до N.

2. Возьмем первое число 2 и зачеркнем каждое второе число, начиная отсчет со следующего за двойкой числа.

3. Возьмем первое незачеркнутое число, которое больше 2 (число 3), и зачеркнем каждое третье число, начиная отсчет от числа, стоящего после 3 (учитывая и ранее зачеркнутые числа).

4. Продолжим действия до тех пор, пока первое незачеркнутое число не окажется больше N.

5. В результате незачеркнутыми окажутся все простые числа, не превосходящие N, и только они.

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	~~49~~	~~50~~

В информатику понятие алгоритма пришло из математики.

Алгоритм — точно определенная система понятных исполнителю предписаний, формальное выполнение которых позволяет получить решение задачи для любого допустимого набора исходных данных за конечное число шагов.

Приведенное определение не является определением в математическом смысле слова, поскольку в нем использованы другие неопределенные понятия — «система предписаний», «формальное выполнение» и др. Это описание понятия «алгоритм», раскрывающее его сущность. (Рассмотрите пример 1.1.) Описание можно уточнить, указав общие свойства, которые характерны для алгоритмов. К ним относятся: дискретность, детерминированность (определенность), понятность, результативность, конечность, массовость.

Дискретность. Алгоритм разбивается на отдельные действия (шаги). Выполнение очередного действия возможно только после завершения предыдущего. При этом набор промежуточных данных конечен и получается по определенным правилам из данных предыдущего действия. **Команда** является специальным указанием для исполнителя, предписывающим ему произвести каждое отдельное дей-

ствие. Команды, которые относят к системе команд исполнителя, называют простыми; другие команды могут быть определены через простые.

Детерминированность. Если алгоритм неоднократно применить к одним и тем же исходным данным, то каждый раз должны получаться одни и те же промежуточные результаты и один и тот же выходной результат. Данное свойство означает, что результат выполнения алгоритма определяется только входными данными и командами самого алгоритма и не зависит от исполнителя алгоритма.

Понятность. Алгоритм не должен содержать команд, смысл которых исполнитель может воспринимать неоднозначно. Запись алгоритма должна быть четкой, полной и понятной. У исполнителя не должно возникать необходимости в принятии каких-либо самостоятельных решений.

Результативность. При точном выполнении команд алгоритма результатом должен быть ответ на вопрос задачи. Если способ получения последующих величин из каких-либо исходных не приводит к результату, то должно быть указано, что следует считать результатом исполнения алгоритма. В качестве одного из возможных ответов может быть установление того факта, что задача не имеет решения.

Конечность. Реализуемый по заданному алгоритму процесс должен остановиться через конечное число шагов и выдать искомый результат. Это свойство тесно связано со свойством результативности.

Необходимость построения формального определения алгоритма привела к появлению в 20—30-х гг. XX в. теории алгоритмов. Для определения различными математиками были предложены:

- машина Тьюринга;
- машина Поста;
- нормальный алгоритм Маркова.

Существовали и другие определения алгоритма. Впоследствии было доказано, что все они эквивалентны.

Алан Мэтисон Тьюринг (1912—1954) — английский логик и математик, оказавший существенное влияние на развитие информатики. Предложенная им в 1936 г. абстрактная вычислительная Машина Тьюринга позволила формализовать понятие алгоритма, которое используется в теоретических и практических исследованиях.

Эмиль Леон Пост (1897—1954) — американский математик и логик. Известен своими трудами по математической логике. Предложил абстрактную вычислительную машину — машину Поста (1936).

Пример 1.2. Часто рецепты приготовления каких-либо блюд называют алгоритмами. В данном случае нарушается свойство детерминированности, поскольку при приготовлении блюда разными людьми результат может быть разным (например, он может зависеть от того, на какой плите готовили, или от качества продуктов). Кроме того, в рецептах часто бывают фразы «посолить по вкусу», «добавить 2—3 ложки сахара» и т. д., которые нарушают свойство понятности.

Пример 1.3. Задача может иметь решение, но сформулировать алгоритм решения этой задачи не всегда удается. Если человеку предложить фотографии животных, то он достаточно быстро сможет разделить их на две группы: дикие и домашние. Однако сформулировать алгоритм, согласно которому он это сделал, на сегодняшний день не представляется возможным. Некоторые задачи классификации сегодня успешно решаются системами искусственного интеллекта с помощью методов машинного обучения. Однако характерной чертой таких методов является не прямое решение задачи, а процесс обучения в ходе анализа множества решений сходных задач.

Пример 1.4. Способы проверки алгоритма на правильность работы:

- математическое доказательство;
- использование специально подобранных тестов.

Массовость. Алгоритм пригоден для решения любой задачи из некоторого класса задач, т. е. начальная система величин может выбираться из некоторого множества исходных данных, которое называется **областью применимости алгоритма**.

Для практического решения задач на компьютере наиболее существенно свойство массовости. Как правило, ценность программы для пользователя будет тем выше, чем больший класс однотипных задач она позволит решать.

Если разработанная последовательность действий не обладает хотя бы одним из перечисленных выше свойств, то она не может считаться алгоритмом. Для понимания свойств алгоритма рассмотрите примеры 1.2 и 1.3.

Для одного и того же алгоритма могут существовать различные формы записи: текстовое описание, блок-схема, машина Тьюринга и др. Независимо от формы записи любой алгоритм может быть представлен с использованием базовых алгоритмических конструкций: **следование, цикл** и **ветвление**.

В рамках теории алгоритмов происходит анализ различных алгоритмов решения задачи для выбора наиболее эффективного (оптимального). Разработка инструментов для анализа эффективности алгоритмов — одна из задач теории алгоритмов.

Любой алгоритм нужно проверять на правильность работы (пример 1.4).

1. Что такое алгоритм?
2. Какие свойства характеризуют алгоритм?
3. Какие базовые алгоритмические конструкции используются при составлении алгоритмов?

Упражнения

1 Прокомментируйте основные свойства алгоритма для решета Эратосфена.

2 Аль-Хорезми составил алгоритмы арифметических действий еще в IX в. Составьте алгоритмы выполнения арифметических действий в столбик. Проверьте для составленных алгоритмов основные свойства.

3 Приведите примеры алгоритмов, обладающих указанными признаками.

1. Используется только алгоритмическая конструкция *следование*.
2. Присутствует алгоритмическая конструкция *ветвление*.
3. Присутствует алгоритмическая конструкция *цикл*.
4. Используются подпрограммы.

4 Опишите последовательность действий для решения задачи «Волк, коза и капуста».

Однажды крестьянину понадобилось перевезти через реку волка, козу и капусту. У крестьянина есть лодка, в которой может поместиться, кроме самого крестьянина, только один объект — или волк, или коза, или капуста. Если крестьянин оставит без присмотра волка с козой, то волк съест козу; если крестьянин оставит без присмотра козу с капустой, коза съест капусту. Как крестьянину перевезти на другой берег и волка, и козу, и капусту в целости и сохранности?

Будет ли полученная последовательность действий алгоритмом? Какое свойство алгоритма не выполняется? Возможно ли переформулировать задачу так, чтобы аналогичная последовательность действий стала алгоритмом?

5 Есть двое песочных часов на 3 мин и на 8 мин. Как с их помощью отмерить 7 мин? Определите систему команд исполнителя, который может решать эту задачу, и составьте для него последовательность действий, приводящую к ответу. Какие алгоритмические конструкции были использованы при реализации? Можно ли считать полученную последовательность действий алгоритмом? Какое свойство алгоритма не выполняется? Возможно ли переформулировать задачу так, чтобы аналогичная последовательность действий стала алгоритмом?

§ 2. Языки программирования

2.1. Высокоуровневые языки программирования

Любой алгоритм рассчитан на конкретного исполнителя, имеющего свою систему команд. Алгоритм для компьютера должен быть записан с помощью команд, которые компьютер может выполнять.

Первым высокоуровневым языком программирования, который был реализован практически, стал в 1949 г. Краткий код (Short Code). Операции и переменные в этом языке программирования кодировались двухсимвольными сочетаниями.

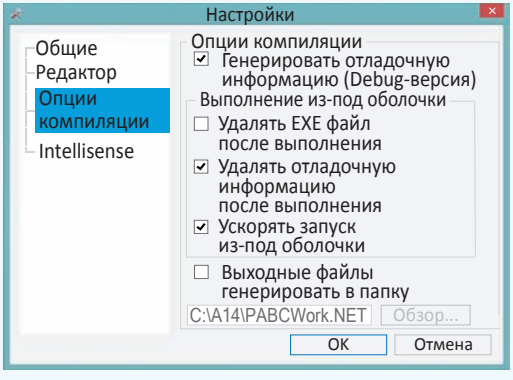
Пример 2.1. Некоторые языки программирования высокого уровня:

- C++;
- Python;
- Pascal (Delphi);
- C#;
- Java;
- JavaScript;
- Perl;
- Fortran;
- VisualBasic;
- Lisp.

Пример 2.2. При трансляции программы происходит преобразование текста с одного языка на другой. Различают следующие виды трансляции: компиляция и интерпретация.

Компилятор — транслятор, преобразующий исходный код с какого-либо языка программирования на машинный. В результате создается исполняемый файл, который может быть выполнен непосредственно в операционной системе.

Среда программирования PascalABC имеет встроенный компилятор, опции которого можно настроить, выполнив команду **Сервис → Настройки**:



Интерпретатор — транслятор, который может работать двумя способами:

- читать код и исполнять его сразу (чистая интерпретация);
- читать код, создавать в памяти промежуточное представление кода (байт-код или p-код), выполнять промежуточное представление кода.

Чистая интерпретация применяется для языков JavaScript, VBA, Lisp и др. Примеры интерпретаторов, создающих байт-код: Perl, PHP, Python и др.

Устройством, выполняющим команды в компьютере, является процессор. Систему команд процессора называют машинным языком или машинным кодом. Каждая команда процессора записывается в двоичном коде. Для записи этих команд в символьной форме используют язык Ассемблер. Ассемблер является языком низкого уровня, поскольку содержит команды, отражающие машинный код.

В языках программирования высокого уровня используются команды, которые объединяют последовательности машинных команд. Программы, написанные на таких языках, проще для понимания человеком, поскольку для обозначения команд используют слова естественного языка (чаще всего английского).

Языки программирования высокого уровня, или высокоуровневые языки программирования (пример 2.1), были разработаны для того, чтобы суть алгоритма могла не зависеть от аппаратной реализации компьютера. Для преобразования текста программы, написанной на языке высокого уровня, в элементарные машинные команды используются специальные программы — **трансляторы** (пример 2.2). Например, в тексте программы достаточно написать «while», а уже транслятор языка переведет эту команду в последовательность машинных кодов. Принципы работы трансляторов зависят от аппаратного и программного обеспечения компьютера.

Языки программирования высокого уровня являются формальными

искусственными языками. У каждого языка программирования можно выделить две составляющие: синтаксис и семантику. **Синтаксис** (грамматика языка) — совокупность правил для написания программы. **Семантика** — смысловая сторона языка (определяет смысловое содержание языковой конструкции).

Текст программы на языке высокого уровня представляет собой обычный текстовый файл. Для его «чтения» и превращения в последовательность машинных команд выполняется анализ текста программы — проверка на соответствие синтаксическим правилам и семантике данного языка программирования. В случае обнаружения несоответствия компилятор может выдавать сообщения об ошибках (пример 2.3).

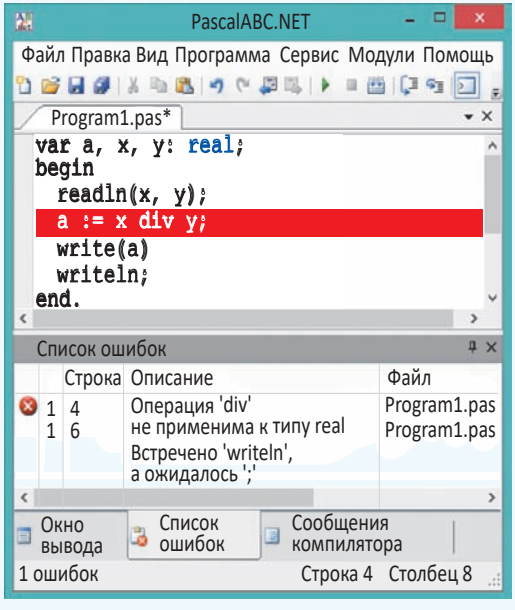
За время существования вычислительных машин было создано более 8 тыс. языков программирования, и ежегодно появляются новые. Некоторые из них являются узкоспециальными, другие используются миллионами программистов по всему миру.

Существуют различные подходы к классификации языков программирования. По степени отличия семантики языка от машинного кода языки делят на низкоуровневые и высокоуровневые. Часто языки программирования делят на компилируемые и интерпретируемые, однако такое деление условно, поскольку для любого традиционно компилируемого языка (такого, как Паскаль) можно написать интерпретатор.

В 1951 г. американский ученый Грейс Мюррей Хоппер (1906—1992) создала первый в мире компилятор и ввела сам этот термин.

Компилятор осуществлял функцию объединения команд, производил выделение памяти компьютера и преобразование команд высокого уровня (в то время псевдокодов) в машинные команды.

Пример 2.3. Семантическая (строка 4) и синтаксическая (строка 6) ошибки в среде PascalABC.



Роберт В. Флойд (1936—2001) — американский ученый в области теории вычислительных систем. Лауреат премии Тьюринга. Впервые термин «парадигма программирования» был применен Р. Флойдом в 1978 г. во время получения премии Тьюринга: «Если прогресс искусства программирования в целом требует постоянного изобретения и усовершенствования парадигм, то совершенствование искусства отдельного программиста требует, чтобы он расширял свой репертуар парадигм».

Флойд отмечал, что парадигмы программирования не являются взаимоисключающими, они могут сочетаться, обогащая инструментарий программиста.

Пример 2.4. Основная идея структурного программирования заключается в том, что программа должна иметь простую структуру, быть хорошо читаемой и легко модифицируемой. Структурированность кода поддерживается посредством подпрограмм, которые вызываются из других подпрограмм. Структурное программирование поддерживают такие языки, как Pascal, Go, C и многие другие языки программирования.

Пример 2.5. Особенность языков процедурного программирования заключается в том, что задачи разбиваются на шаги и решаются шаг за шагом. Решение для каждого отдельного шага оформляется в виде отдельной процедуры. К процедурным языкам относят: C, Pascal, Lua и др.

2.2. Парадигмы программирования

В истории развития языков программирования можно выделить различные парадигмы программирования — совокупность идей и понятий, определяющих стиль программирования. Между парадигмами и языками программирования нет жесткой связи. Парадигма показывает один из возможных способов использования средств языка программирования для написания кода программы. Часто язык программирования, созданный в рамках одной парадигмы, через некоторое время модернизируется, расширяется и начинает использоваться в рамках другой парадигмы.

Рассмотрим некоторые парадигмы программирования.

Структурное программирование — парадигма программирования, в основе которой лежит представление программы в виде блоков иерархической структуры. Разработана в конце 1960-х — начале 1970-х гг. В соответствии с данной парадигмой любая программа состоит из трех базовых управляющих структур: *ветвление, цикл* и *последовательность*; кроме того, используются подпрограммы (пример 2.4). Разработка программы ведется пошагово, методом «сверху вниз» (нисходящее программирование): сначала определяются цели решения задачи, а затем идет детализация каждого шага, который, став отдельной задачей, также может детализироваться.

Процедурное программирование — парадигма программирования, при которой последовательно выполняе-

мые команды можно собрать в подпрограммы с помощью механизмов самого языка (пример 2.5). Это концепция программирования «снизу вверх», или концепция восходящего программирования — разработка программ начинается с разработки подпрограмм (процедур, функций), в то время как проработка общей схемы не закончилась.

Парадигмы структурного и процедурного программирования основаны на подпрограммах. Разница заключается в порядке их разработки: «сверху вниз» или «снизу вверх».

Функциональное программирование — парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании. Основывается функциональное программирование на вычислении результатов функций от исходных данных и результатов других функций и не предполагает явного хранения состояния программы (пример 2.6).

Объектно-ориентированное программирование (ООП) — парадигма программирования, основанная на представлении программы в виде совокупности объектов и отражении их взаимодействия. Концепция ООП позволяет объединить данные и алгоритмы их обработки в единую структуру, называемую классом. Каждый объект является экземпляром какого-либо класса (пример 2.7). В ООП программа представляет собой набор объектов, имеющих состояние и поведение. Состояние и поведение объекта может измениться в результате события.

Пример 2.6. В основе функциональных языков лежит лямбда-исчисление (λ-исчисление) — математическая теория для формализации понятия вычислимости. К ним относят: Haskell, Lisp, F# и др.

Пример 2.7. В программе *текстовый редактор* объектами могут быть абзац текста, меню, кнопки и т. д. В классе, который описывает кнопку, содержатся данные о размере кнопки, ее внешнем виде, алгоритмы, позволяющие «нажать» кнопку или «навести на нее мышь» и др.

Конкретная кнопка, например Ч, является объектом. Такое событие, как «клик мышью по такой кнопке», изменит начертание текста. Событие «двойной клик мышью по абзацу текста» выделит его и т. д.

К объектно-ориентированным языкам относят:

- C++;
- Java;
- Delphi;
- Python;
- Ruby и др.

Развитие объектно-ориентированного программирования часто связывают с понятиями «событие» (событийно-ориентированное программирование) и «компонент» (компонентное программирование).

Среда PascalABC предоставляет возможность создавать оконные приложения. При разработке интерфейса программы используются визуальные компоненты, а программный код основан на событийном программировании. Создавать такие приложения вы научитесь в 11-м классе.

Пример 2.8. Мультипарадигменные языки программирования чаще всего поддерживают процедурную (структурную), объектно-ориентированную и функциональную парадигмы: C++, Python, JavaScript, Ruby, C#.

Существуют и другие классификации и способы сравнения различных языков программирования[1].

Пример 2.9. Учебный язык обеспечивает простоту, ясность и удобочитаемость конструкций языка. Как учебные языки программирования разрабатывались: Basic, Pascal, Logo, Scratch.

Пример 2.10. Некоторые эзотерические языки служат для проверки математических концепций (Thue, Unlambda), другие создаются для развлечения. Часто они пародируют «настоящие» языки программирования или являются абсурдным воплощением концепций программирования (Smetana, Var'aq, FiM++ и др.).

Пример 2.11. Псевдокод алгоритма нахождения суммы квадратов первых n натуральных чисел.

```
ввод n;
S = 0
нц для i от 1 до n
  S = S + i * i
кц
```

Служебные (ключевые) слова — зарезервированные слова, которые имеют специальные значения для компилятора. Их нельзя использовать как идентификаторы в программах.

Программа в целом — это объект. Для выполнения своих функций она обращается к входящим в нее объектам, которые, в свою очередь, могут обращаться к другим объектам, реализовывать свои методы или реагировать на события. Объектно-ориентированное программирование особенно важно при реализации крупных проектов.

Большинство современных языков программирования являются **мультипарадигменными** — поддерживают сразу несколько парадигм программирования (пример 2.8).

Отдельно рассматривают такие классы языков программирования, как **учебные** (пример 2.9) и **эзотерические** языки программирования (пример 2.10).

Часто для записи алгоритмов применяют **псевдокод** — язык описания алгоритмов, использующий ключевые слова языков программирования, но опускающий детали, несущественные для понимания алгоритма (например, описания переменных). Главная цель использования псевдокода — обеспечить понимание алгоритма человеком, сделать описание более воспринимаемым, чем исходный код на языке программирования. При написании псевдокода может использоваться лексика русского языка (пример 2.11).

2.3. Основные структурные элементы языка программирования

Для записи элементов языка программирования используется алфавит. **Алфавиты** большинства языков

[1] https://ru.wikipedia.org/wiki/Сравнение_языков_программирования (дата доступа: 10.02.2019).

программирования близки друг другу по синтаксису и, как правило, используют буквы латинского алфавита, арабские цифры и общепринятые спецсимволы (знаки препинания, знаки математических операций и сравнений, разделители, служебные слова). Большинство распространенных языков программирования содержат в своем алфавите следующие элементы:

- буквы — {AaBbCcDd...};
- цифры — {0 1 2 3 4 5 6 7 8 9};
- знаки арифметических операций — {× / + −...};
- знаки сравнения — {< > = ...};
- разделители — {. , ; : ( ) { } [ ]... };
- служебные слова — {**if while for** и т. д.};
- комментарии — любой набор символов и др.

Несмотря на значительные различия между языками, многие фундаментальные понятия в большинстве языков программирования схожи между собой (пример 2.12). Согласно известной формуле Н. Вирта «Алгоритмы + структуры данных = программы», рассматривая язык программирования, нужно говорить о способах записи команд алгоритма с помощью операторов и организации работы с данными (пример 2.13).

Операторы

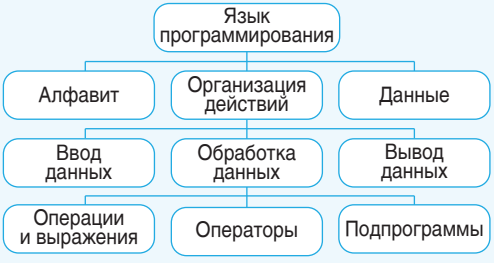
Одним из основных понятий всех языков программирования является **оператор**, который представляет собой законченную фразу языка и является предписанием на выполнение конкретных действий по обработке данных. Программа строится из операторов так же, как текст литературного

Пример 2.12. Некоторые служебные слова (в алфавитном порядке) в разных языках программирования.

Pascal	Python	C++

`const,`
`do,`
`else,`
`for,`
`if,`
`then,`
`var,`
`while`
	`def,`
	`elif,`
	`else,`
	`for,`
	`if,`
	`return,`
	`while`
		`const,`
		`do,`
		`else,`
		`for,`
		`if,`
		`return,`
		`while`

Пример 2.13. Структурная схема процедурного языка программирования.



Выражения строятся из величин (констант и переменных), функций, скобок, знаков операций и т. д. Тип выражения определяется результатом вычислений. Выражения могут принимать числовые, логические, символьные, строковые и другие значения.

Выражение 5 + 3 является числовым, а выражение $A + B$ может иметь самый разный смысл в зависимости от того, что стоит за идентификаторами A и B (если A и B — строки, то результат — строка, которая получилась при конкатенации исходных строк).

Пример 2.14. Запись оператора цикла (поиск суммы квадратов первых n натуральных чисел).

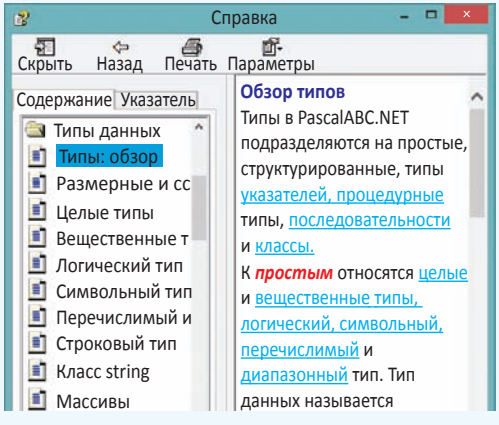
Pascal:
```
for var i := 1 to n do
  s := s + i * i;
```

Python:
```
for i in range(n + 1):
    s += i * i
```

C++:
```
for (int i = 1; i <= n; i++)
    s += i * i;
```

Пример 2.15. Классификация данных в языке программирования.

Подобная структура типов данных присуща многим языкам программирования. С другими типами данных, которые используются в языке PascalABC, можно познакомиться в справочной системе.



произведения формируется из предложений.

Выделяют следующие операторы: оператор присваивания, оператор условного перехода (ветвления), оператор цикла (пример 2.14), оператор выбора, составной оператор, иногда используют пустой оператор, оператор безусловного перехода и др.

Все операторы языка в тексте программы отделяются друг от друга с помощью явных или неявных разделителей (в Pascal таким разделителем является «;»). Операторы выполняются в том порядке, в котором они записаны в тексте программы. Порядок выполнения операторов может быть изменен только посредством управляющих операторов: ветвления, цикла и др.

Данные

Большая часть операторов предназначена для обработки величин. Величина характеризуется типом, именем и значением. Типы данных могут быть простыми и структурированными (пример 2.15). Величина простого типа в каждый момент имеет одно значение. Величина структурированного типа состоит из величин других типов. Например, строка состоит из символов, каждый отдельный символ строки имеет свое значение (код). Самым распространенным структурированным типом данных является массив, с которым вы познакомитесь в этом учебном году.

Всем объектам в языках программирования (переменным, функциям, процедурам и др.) даются имена. Имя объекта в программе называют **иден-**

тификатором (от слова «идентифицировать»). Идентификатором является любая конечная последовательность букв и цифр, начинающаяся с буквы (пример 2.16). Имя может содержать знак подчеркивания «_». Использовать служебные слова языка в качестве идентификатора запрещается в большинстве языков программирования.

Величины могут быть постоянными (константы) и переменными. **Переменная** может принимать некоторое значение в результате выполнения команды ввода или с помощью оператора присваивания. В ходе выполнения программы значения переменной могут неоднократно меняться. После описания переменная отождествляется с некоторым блоком памяти, содержимое которого является ее значением. Переменная хранит значение, соответствующее ее типу (например, переменная целого типа не может принимать значение вещественного числа). Это значение может извлекаться из памяти для выполнения с ним операций, соответствующих типу переменной.

Подпрограммы

Алгоритм, реализующий решение отдельной части основной задачи, называют **вспомогательным**, а его запись на языке программирования — **подпрограммой**. Подпрограммы могут быть реализованы в виде функций или процедур.

Пример 2.16. Имена переменных задает программист. Существуют рекомендации, как можно (нужно) именовать переменные в коде:

- имя переменной должно быть понятным, наглядным и отражать суть обозначаемого объекта (sum, number, count_of_positive);
- вводить переменным короткие имена (*s*, *i*, *n*) можно в том случае, когда они используются в небольшом фрагменте кода и их применение очевидно.

Некоторым идентификаторам заранее предписан определенный смысл, например sin, cos, sqrt, abs — имена математических функций.

Многие компании разрабатывают свои правила по оформлению кода, в которых прописаны также и правила именования переменных. В компании Microsoft используют так называемую «венгерскую нотацию»[1]. Известными являются также:

- «верблюжья нотация»;
- «змеиная нотация»;
- «шашлычная нотация»[2].

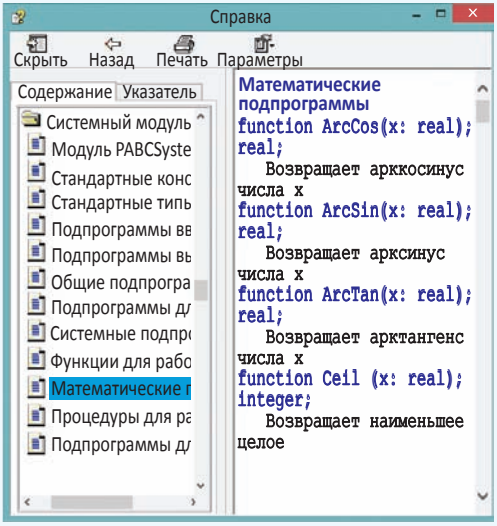
Правила оформления кода, разработанные в некоторых компаниях, являются открытыми и могут использоваться другими разработчиками. Например, в Google разработаны styleguide («гид по стилю») для разных языков программирования (C++, Java, Python, Lisp и др.)[3].

[1] https://ru.wikipedia.org/wiki/Венгерская_нотация#cite_note-hunganotat-1 (дата доступа: 25.02.2019).

[2] https://ru.hexlet.io/blog/posts/naming-in-programming (дата доступа: 25.02.2019).

[3] http://google.github.io/styleguide/ (дата доступа: 25.02.2019).

Пример 2.17. Список стандартных функций языка программирования PascalABC можно посмотреть в справочной системе:



Пример 2.18. Процедуры в языках программирования.

В языке VisualBasic процедуры объявляются как sub (сокращение от англ. *subroutine* — подпрограмма).

В языке C++ нет отдельных конструкций для описания процедур. Их роль выполняют функции, которые имеют тип *void* (англ. «пустота»).

На сайте Tiobe[1] ежемесячно публикуется рейтинг языков программирования.

Рейтинг составляется на основе подсчета результатов поисковых запросов, содержащих название языка, в Google, Blogger, Wikipedia, YouTube, Baidu, Yahoo!, Bing, Amazon.

Функция описывает процесс вычисления определенного значения, зависимого от некоторых аргументов, поэтому для функции всегда указывается тип возвращаемого значения. Для каждого языка высокого уровня разработана библиотека стандартных функций: арифметических, логических, символьных и т. д. (пример 2.17). Функции (как стандартные, так и задаваемые программистом) используются в программе в выражениях.

Часто используют подпрограммы, которые не возвращают конкретное значение, а представляют собой самостоятельный этап обработки данных. В языке Pascal их называют **процедурами**, в других языках они могут называться по-другому или не иметь собственного названия и описываться так же, как функции (пример 2.18).

Язык программирования — инструмент для решения конкретной задачи. Не существует единственного самого лучшего языка программирования. Для решения задач разного рода и уровня сложности требуется применять разные языки и технологии программирования. В простейших случаях достаточно освоить основы структурного написания программ, например на языке PascalABC. Для создания же сложных проектов требуется не только свободно владеть каким-то языком в полном объеме, но и иметь представление о других языках и их возможностях. Как правило, чем сложнее задача, тем больше времени требуется на освоение инструментов, необходимых для ее решения.

[1] https://www.tiobe.com/tiobe-index/ (дата доступа: 26.06.2019).

1. Для чего предназначен транслятор?
2. Какие функции выполняет компилятор? Интерпретатор?
3. Что определяется парадигмой программирования?
4. Из каких элементов может состоять алфавит языка программирования?
5. Что представляет собой оператор языка программирования?
6. Какие типы данных вам известны?
7. Для чего используются функции и процедуры?

Упражнения

1 Напишите программы для решения следующих задач.

1. Определите последнюю цифру натурального числа N.
2. Два отрезка на плоскости задаются координатами своих концов. Определите, какой из них короче.
3. Найдите сумму $1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{N^2}$ для заданного N.
4. Вводится строка текста. Определите, является ли она палиндромом.
5. Вводятся два целых числа, являющихся числителем и знаменателем дроби. Сократите дробь, выведите полученные числитель и знаменатель.
Подсказка: можно воспользоваться алгоритмом Евклида.
6*. Дед Мазай и заяц играют в очень простую игру. Перед ними — гора из N одинаковых морковок. Каждый из игроков во время своего хода может взять из нее любое количество морковок, равное неотрицательной степени числа 2 (1, 2, 4, 8, ...). Игроки ходят по очереди. Кто возьмет последнюю морковку, тот и выигрывает. Составьте алгоритм, который при заданном значении N определяет победителя в этой игре. Учтите, что каждый из игроков хочет выиграть и не делает лишних ходов, т. е. играет оптимально.

2* Предложенные ниже алгоритмы записаны разными способами. Определите, что делает каждый из предложенных алгоритмов, и реализуйте их на языке Pascal.

1. Алгоритмический язык

```
ввод a
n := Длина(a)
m := 1
b := Извлечь(a, m)
нц для i от 7 до n
  c := Извлечь(a, i)
  b := Склеить(b, c)
кц
вывод b
```

Для слова «энергетика» программа выводит «этика».

2. Python

```
a = int(input())
k = 0
s = 1
while k < a:
  k = k + 1
  s = s + 1.0/k
print(s)
```

При $a = 5$ программа выводит 3.2833333333333337.

3. Basic

```
INPUT X
L = 0
M = 0
WHILE X > 0
   M = M + 1
    IF X MOD 3 <> 0 THEN
     L = L + 1
    END IF
   X = X \ 3
WEND
PRINT L
PRINT M
```

Для значения 5637 программа выводит 4 и 8.

4. C++

```
int F(int x)
{
   return x*x + 16*x + 15;
}
int main()
{
  int a, b;
  cin >> a >> b;
  int M = 0;
  for (int t = a; t <= b; t++)
    if (F(t) > 0)
      M = M + 1;
  cout << M;
  return 0;
}
```

При $a = -3$, $b = 5$ программа выводит 6.

3* Изобразите любой алгоритм из упражнения 2 в виде блок-схемы.

Глава 1
АЛГОРИТМЫ ОБРАБОТКИ МАССИВОВ

§ 3. Структурированный тип данных массив

Впервые тип данных *массив* появился в языке Фортран (создан в период с 1954 по 1957 г. в корпорации IBM). Уже первые версии языка поддерживали трехмерные массивы (в 1980 г. максимальная размерность массива была увеличена до 7). Массивы были необходимы для создания математических библиотек, в частности содержащих процедуры решения систем линейных уравнений.

Пример 3.1. В 10 А классе 25 учащихся. Известен рост каждого в сантиметрах. Для хранения значений роста можно использовать массив A, состоящий из 25 целых чисел. Индекс каждого элемента — порядковый номер учащегося из списка в классном журнале. Тогда запись A[5] — рост учащегося под пятым номером.

3.1. Понятие массива

В современном мире ежесекундно происходит обработка огромного числа данных с помощью компьютера. Если необходимо обрабатывать данные одного типа (числа, символы, строки и др.), то для их хранения можно воспользоваться типом данных, который называется **массив.**

Массив — упорядоченная последовательность данных, состоящая из конечного числа элементов, имеющих один и тот же тип, и обозначаемая одним именем.

Массив является структурированным (составным) типом данных. Это означает, что величина, описанная

как массив состоит из конечного числа других величин. Так, например, можно создать массивы из 10 целых или 100 вещественных чисел. Тип элементов массива называют **базовым типом**. Все элементы массива упорядочены по индексам, определяющим местоположение элемента в массиве.

Массиву присваивается имя, посредством указания которого можно ссылаться на данный массив как на единое целое. Элементы, образующие массив, упорядочены так, что каждому из них соответствует номер (индекс), определяющий место элемента в общей последовательности (примеры 3.1—3.3). Индексы представляют собой выражения любого простого типа, кроме вещественного. Доступ к каждому отдельному элементу осуществляется обращением к имени массива с указанием индекса нужного элемента, индекс элемента записывается после имени в квадратных скобках (пример 3.4).

Если обращение к элементам массива осуществляется при помощи только одного индекса, то такой массив называют **одномерным** или **линейным**. Элементы данного массива располагаются цепочкой друг за другом. Количество индексов, по которым обращаются к элементу в массиве, определяет **размерность массива**. Кроме одномерных, могут использоваться двумерные, трехмерные и другие массивы.

3.2. Описание массивов

Описание массива в языке Паскаль происходит следующим образом:

var <имя массива>: **array** [<тип индекса>] **of** <тип элементов>;

Пример 3.2. В 10 Б классе 27 учащихся. В классном журнале указаны фамилия и имя каждого из них. Для хранения списка учащихся можно использовать массив S, состоящий из 27 строк. Индекс каждого элемента — порядковый номер учащегося из списка в классном журнале. Тогда запись `S[5]` — фамилия и имя учащегося под номером 5.

Пример 3.3. Каждый день в декабре измеряли температуру воздуха. Для хранения значений температуры можно использовать массив `T`, состоящий из 31 вещественного числа. Индекс элемента — номер дня в декабре. Запись `T[15]` — температура воздуха 15 декабря.

Пример 3.4. Обращение к элементу массива: `a[3]`, `T[i]`, `S[n-1]`.

Двумерный массив — массив, элементами которого являются одномерные массивы. Его можно представить как таблицу с данными, в которой каждая строка — линейный массив. Обращение к элементу осуществляется по двум индексам: `a[3][5]` — элемент, находящийся в третьей строке и пятом столбце. Примером использования двумерного массива является лист электронной таблицы.

Массив p, описанный следующим образом:

```
var p: array [1..30] of array [1..30]
       of boolean;
```

,

можно использовать для определения свободных мест в зрительном зале. Тогда запись

```
p[2][13] := true;
```

будет означать, что во втором ряду место 13 свободно, а запись

```
p[5][7] := false;
```

— в пятом ряду место 7 занято.

Пример 3.5. Опишем массив, рассмотренный в примере 3.1. Размер описанного массива — 25 элементов:

```
var a: array[1..25] of integer;
```

Пример 3.6. Опишем массив, рассмотренный в примере 3.2:

```
var S: array[1..27] of string;
```

Пример 3.7. Опишем массив, рассмотренный в примере 3.3. Размер описанного массива — 31 элемент.

```
var T: array[1..31] of real;
```

Пример 3.8*. Описать массив для хранения следующих данных: имеется здание склада, в котором есть два подземных этажа, цокольный и три верхних. Необходимо хранить количество пустых отсеков склада на каждом этаже. Размер описанного массива — 6 элементов:

```
const verh = 3;
      niz = -2;
var Sklad: array[niz..verh] of
                          integer;
```

Пример 3.9. Команда `b := a;` допустима для массивов `a` и `b`, описанных следующим образом:

```
var a, b: array[1..10] of integer;
```

Но команда `b := a;` выдаст ошибку, если массивы будут описаны так:

```
var a: array[1..10] of integer;
    b: array[1..10] of integer;
```

или так:

```
var a: array[1..10] of integer;
    b: array[1..15] of integer;
```

или так:

```
var a: array[1..10] of integer;
    b: array[1..10] of real;
```

Имя массива является идентификатором и задается по тем же правилам, что и имена любых других переменных.

Тип индекса определяет, как будут нумероваться элементы в массиве. Для задания типа индекса указывают номер первого элемента в массиве, затем ставят две точки, после которых указывают номер последнего элемента. Данные, которые используются для задания индексов, должны быть константами. Диапазон индексов определяет максимально возможное количество элементов в массиве — **размер массива.**

Тип элементов задает значение базового типа для данного массива. Базовый тип может быть любым из известных вам типов (примеры 3.5—3.8).

3.3. Операции над массивами

Массивы, описанные одинаково (в одной команде описания), можно использовать в операциях присваивания. В результате выполнения этой команды все элементы одного массива будут переписаны во второй (пример 3.9). Если массивы описаны одинаково, но в разных строках или описаны поразному, то при попытке присваивания возникнет ошибка о невозможности преобразовать типы.

Никакие другие операции для массива как для типа данных не определены.

Операции, выполняемые с элементами массива, соответствуют операциям, выполняемым над базовым типом. Если, например, описан массив из чисел типа integer, то с элемента-

ми такого массива можно выполнять такие же операции, как и с целыми числами. Элементы массива называются **индексированными переменными.** Они могут использоваться так же, как и простые переменные (пример 3.10).

3.4. Ввод и вывод элементов массива

Чтобы работать с массивом, необходимо задать начальные значения элементов массива. Сделать это можно несколькими способами:

1) ввод элементов массива с клавиатуры;

2) использование случайных чисел для определения значений;

3) использование функций (стандартных или собственных) для определения значений;

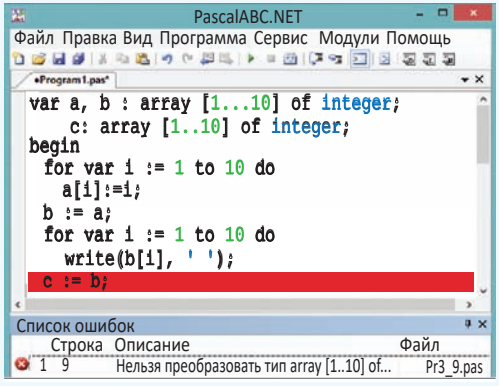
4) определение элементов массива как констант.

При вводе элементов массива с клавиатуры каждый элемент должен вводиться отдельно. Если количество вводимых элементов определено, то можно воспользоваться циклом **for** (пример 3.11).

При вводе элементов массива следует помнить, что количество вводимых элементов не может быть больше размера массива. В массив, описанный в примере 3.11, можно ввести любое количество чисел от 1 до 10, изменив значение 10 в заголовке цикла.

При описании массива размер определяет максимальное количество возможных элементов. При вводе можно определять количество элементов, которое необходимо для обработки в каждом конкретном случае (пример 3.12).

Пример 3.9. *Продолжение.*



Пример 3.10. Операции над индексированными переменными:

```
a[3] := 25 mod 7;
s := (t[1] + t[30])/2;
a[k] := b[k]*2;
Sum := Sum + a[i];
if a[i] < 0 then ...
```

Пример 3.11. Ввести 10 элементов массива a.

```
var a: array[1..10] of integer;
begin
 writeln('Введите 10 чисел
         через пробел');
 for var i := 1 to 10 do
  read(a[i]);
...
end.
```

Пример 3.12. Ввести заданное количество элементов массива a.

```
var a: array[1..100] of integer;
  n: integer;
begin
 writeln('Введите количество
         чисел в массиве');
 readln(n);
 writeln('Введите', n,
        'чисел через пробел');
 for var i := 1 to n do
  read(a[i]);
...
end.
```

Пример 3.13. Ввод массива строк.

```
var a: array[1..100] of string;
    n: integer;
begin
 writeln('Введите количество
        строк в массиве');
 readln(n);
 writeln('Введите', n,'строк,
       каждую с новой строки');
 for var i := 1 to n do
  readln(a[i]);
...
end.
```

Пример 3.14. Случайным образом задать n элементов массива a. Каждый элемент — число из отрезка [0; 100].

```
var a: array[1..100] of integer;
    n: integer;
begin
 writeln('Введите количество
        чисел в массиве');
 readln(n);
 for var i := 1 to n do
  a[i] := random(101);
...
end.
```

Пример 3.15. Описание массива, элементы которого являются числовыми константами.

```
const simple_numb: array[1..5]
      of integer = (2, 3, 5, 7, 11);
```

Пример 3.16. Описание массива, элементы которого являются строковыми константами.

```
const c_rgb: array of string =
    ('красный','синий','зеленый');
```

Пример 3.17. Вывод элементов массива в столбец (по одному в строке).

```
for var i := 1 to n do
  writeln(a[i]);
```

Пример 3.18. Вывод элементов массива в строку (через пробел).

```
for var i := 1 to n do
  write(a[i], ' ');
```

Если необходимо вводить массив из строк, нужно помнить, что каждый элемент вводится в отдельной строке с использованием команды readln (пример 3.13). Использовать пробел как разделитель не получится, поскольку пробел будет воспринят как очередной символ строки.

Иногда бывает удобно задавать элементы массива случайным образом. Для этого используется функция random(k), которая генерирует случайное целое число из промежутка [0; k) (пример 3.14).

Если элементы массива должны принадлежать отрезку [a; b], то можно использовать функцию random(a, b) или определить значение элемента массива так:

```
a[i]:= random(b-a+1)+ a;
```

Если элементы массива не будут изменяться при решении задачи, то массив может быть описан как константа (примеры 3.15, 3.16). При таком описании можно не указывать индексы элементов в массиве, тогда нумерация будет осуществляться от нуля до количества элементов в списке минус один.

Выводить элементы массива можно в столбец (пример 3.17) или в строку (пример 3.18). Если элементы массива выводятся в строку, то между ними нужно выводить символ-разделитель (чаще всего используют пробел), иначе все числа будут распечатаны подряд как одно число с большим количеством цифр. Выводить элементы массива можно не только

в прямом, но и в обратном порядке (пример 3.19).

3.5. Решение задач с использованием ввода-вывода массивов

Пример 3.20. Написать программу, которая введет элементы массива с клавиатуры и выведет сумму третьего и пятого элементов.

Этапы выполнения задания

I. Исходные данные: массив a и количество элементов n.

II. Результат: S — сумма третьего и пятого элементов.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Вычисление суммы.
3. Вывод результата.

IV. Описание переменных: a — **array**[1..10] **of** integer; n, S — integer.

Пример 3.21. Написать программу, которая сформирует массив из n чисел из отрезка [0; 100] случайным образом. Вывести массив на экран.

Этапы выполнения задания

I. Исходные данные: массив a и количество элементов n.

II. Результат: полученный массив.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Генерация массива.
3. Вывод результата.

IV. Описание переменных: a — **array**[1..100] **of** integer; n — integer.

Пример 3.19. Вывод элементов массива в строку (в обратном порядке).

```
for var i := n downto 1 do
  write(a[i], ' ');
```

Пример 3.20.
V. Программа:

```
var a: array[1..10] of integer;
   n, S: integer;
begin
 writeln('Введите количество
        чисел в массиве >=5');
 readln(n);
 writeln('Введите ', n,
        'чисел через пробел');
 for var i := 1 to n do
  read(a[i]);
 S := a[3] + a[5];
 write('Сумма чисел = ', S);
end.
```

VI. Тестирование:

Окно вывода

```
Введите количество чисел в массиве
7
Введите 7 чисел через пробел
12 3 4 2 1 19 7
Сумма чисел = 5
```

VII. Анализ результатов. Третий элемент массива равен 4, пятый элемент равен 1, сумма элементов равна 5.

Пример 3.21.
V. Программа:

```
var a: array[1..100] of integer;
    n: integer;
begin
writeln('Введите количество
        чисел в массиве');
 readln(n);
 for var i := 1 to n do
  a[i] := random(101);
 for var i := 1 to n do
  write(a[i],' ');
end.
```

VI. Тестирование:

Окно вывода

```
Введите количество чисел в массиве
8
66 44 80 3 100 66 3 78
```

Пример 3.22.

V. Программа:

```
var a: array[1..100] of integer;
      n, k: integer;
begin
 writeln('Введите количество
         чисел в массиве');
 readln(n);
 for var i:=1 to n do
 begin
  a[i]:=2*random(10, 35);
  write(a[i], ' ');
 end;
 writeln;
 writeln('Введите k');
 readln(k);
 write(a[k]);
end.
```

VI. Тестирование:

Окно вывода

```
Введите количество чисел в массиве
7
50 56 66 60 32 24 66
Введите k
5
32
```

Пример 3.23.

V. Программа:

```
var s: array [1..20] of string;
   n, k1, k2: integer;
begin
 writeln('Количество учащихся ');
 readln(n);
 writeln('Фамилии');
 for var i := 1 to n do
  readln(s[i]);
 writeln('k1 и k2');
 readln(k1, k2);
 for var i := k1 to k2 do
  writeln(s[i]);
end.
```

Пример 3.22. Написать программу, которая сформирует массив из n четных чисел из отрезка [20; 70] случайным образом. Вывести на экран k-й элемент массива.

Этапы выполнения задания

I. Исходные данные: массив a и количество элементов n.

II. Результат: искомый элемент.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Генерация массива.

1.1. Чтобы элементы массива были только четными, необходимо каждый полученный элемент умножать на 2.

1.2. Поскольку элементы умножаются на два, границы исходного отрезка нужно уменьшить в два раза.

1.3. Вывод массива по элементам.

3. Ввод значения k и вывод результата.

IV. Описание переменных: a — **array**[1..100] **of** integer; n, k — integer.

Пример 3.23. Написать программу, которая введет с клавиатуры список фамилий учащихся и выведет из него фамилии с номерами от k1 до k2.

Этапы выполнения задания

I. Исходные данные: массив s и количество учащихся n, номера фамилий — k1 и k2.

II. Результат: список заданных фамилий.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Вывод результата.

IV. Описание переменных: s — **array**[1..20] **of** string; n, k1, k2 — integer.

Пример 3.24. Задать случайным образом два массива X и Y, содержащих по n чисел из отрезка [100; 300], и массив R, содержащий n чисел из отрезка [5; 100]. Построить на экране окружности, координаты центров которых хранятся в массивах X и Y, а радиусы в массиве R.

Этапы выполнения задания

I. Исходные данные: массивы X, Y, R и количество элементов n.

II. Результат: рисунок n окружностей.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Генерация массивов.
3. Установка прозрачного стиля заливки для того, чтобы изображались окружности, а не круги.
4. Вывод результата.

IV. Описание переменных: X, Y, R — **array**[1..100] **of** integer; n — integer.

Все рассмотренные выше способы ввода и вывода массивов универсальны и могут использоваться для разных компиляторов языка Pascal. В среде PascalABC.Net дополнительно реализованы команды print и println, с помощью которых массив можно вывести без использования команды цикла. Команда print(b); выведет элементы массива b в квадратных скобках через запятую: [1,2,3,4,5,6].

Команда println после вывода массива дополнительно переводит курсор на новую строку. Элементы выводятся так же, как и при использовании команды print.

Пример 3.23. *Продолжение.*

VI. Тестирование:

Окно вывода

```
Количество учащихся
6
Фамилии
Белов
Иванов
Королев
Петров
Сидоров
Яшкин
k1 и k2
3
5
Королев
Петров
Сидоров
```

Пример 3.24.

V. Программа:

```
uses graphABC;
var X, Y, R: array[1..100]
      of integer;
  n: integer;
begin
 SetWindowSize(400,400);
 writeln('Введите количество
         чисел в массиве');
 readln(n);
 writeln(n);
 for var i := 1 to n do
 begin
  X[i]:= random(100,300);
  Y[i]:= random(100,300);
  R[i]:= random(5,100);
 end;
 SetBrushStyle(bsClear);
 for var i := 1 to n do
   circle(X[i],Y[i],R[i])
end.
```

VI. Тестирование.

1. Что такое массив?
2. Как описываются массивы?
3. Что такое размер массива?
4. Какие операции допустимы для массивов?
5. Какие способы задания значений элементам массива вы знаете?
6. Как можно вывести массив?

Упражнения

1 Используя примеры 3.14—3.18, выполните следующие задания.

1. Введите 5 чисел и выведите их в одной строке.
2. Введите 7 чисел и выведите их в одной строке в обратном порядке.
3. Задайте 10 случайных чисел и выведите их по одному в строке.
4. Выведите на экран элементы массива, заданного в примере 3.16.

2 Измените программу из примера 3.19 так, чтобы выводилось произведение первых трех элементов.

3 Используя программы из примера 3.19 или 3.20, задайте массив из n случайных чисел из отрезка [–10; 10]. Выведите: первый элемент; последний элемент; элемент, стоящий на среднем месте.

4 Введите массив из n строк с клавиатуры. Выведите элементы массива в обратном порядке.

5 Для массива, описанного в примере 3.2, введите данные с клавиатуры. Задайте номер учащегося. Выведите его фамилию.

6* Введите рост учащихся своего класса, организовав ввод следующим образом:

```
Введите количество учащихся в классе: 15
Вводите рост учащихся
учащийся номер 1: 165
учащийся номер 2: 170
учащийся номер 3: 156
```

7* Для массива, описанного в примере 3.3, задайте значения случайными вещественными числами из интервала (–20; 10). Выведите значения температур для указанного диапазона дат. Пример вывода для диапазона дат от 1 декабря до 8 декабря:

```
1 декабря температура была = 9.4
2 декабря температура была = -11.8
3 декабря температура была = -16.6
4 декабря температура была = 8
5 декабря температура была = 0.9
6 декабря температура была = -9.3
7 декабря температура была = -11.5
8 декабря температура была = 6.6
```

8 Измените программу из примера 3.24 так, чтобы окружности рисовались разными цветами.

§ 4. Выполнение арифметических действий над элементами массива

4.1. Вычисление сумм и произведений элементов массива

Операции, выполняемые с элементами массива, соответствуют операциям, которые выполняются над базовым типом элементов массива (пример 4.1).

Пример 4.2. Задан одномерный массив из целых чисел. Найти сумму и произведение элементов этого массива.

I. Исходные данные: массив a и количество элементов n.

II. Результат: S — сумма элементов и P — произведение элементов массива.

III. Алгоритм решения задачи.

1. Ввод исходных данных. Массив вводится поэлементно с клавиатуры.
2. Определение начального значения для суммы (S := 0) и для произведения (P := 1).
3. В цикле добавляем очередной элемент массива к сумме и к произведению.
4. Вывод результата.

IV. Описание переменных: a — **array**[1..10] **of** integer; n, S, P — integer.

Пример 4.3. Известны отметки по информатике всех учащихся 10 Б класса за первую четверть. Успеваемость в классе будем считать хорошей, если средний балл больше 7, плохой, если средний балл ниже 4, в остальных случаях — успеваемость средняя. Определить успеваемость класса по заданным отметкам.

I. Исходные данные: массив a для хранения отметок и количество учащихся n.

Пример 4.1.

Если базовым типом элементов массива является тип integer, то для элементов массива допустимы следующие операции: +, –, *, **div**, **mod**.

Если в массиве хранятся числа типа real, то допустимыми будут операции +, –, *, /.

Если в массиве хранятся строки, то для каждого его элемента допустимы строковые функции и процедуры.

Пример 4.2.

V. Программа:

```
var a: array[1..10] of integer;
   n, S, P: integer;
begin
 write('Введите n =');
 readln(n);
 writeln('Вводите элементы');
 for var i := 1 to n do
  read(a[i]);
 S := 0;
 P := 1;
 for var i := 1 to n do
 begin
  S := S + a[i];
  P := P * a[i];
 end;
 writeln('Сумма = ', S);
 writeln('Произведение = ', P);
end.
```

VI. Тестирование.

Окно вывода

```
Введите n = 5
Вводите элементы
3 2 44 -1 3
Сумма = 51
Произведение = -792
```

VII. Анализ результатов. Проверить правильность вычислений можно на калькуляторе.

Пример 4.3.

V. Программа:

```
var a: array[1..30] of integer;
 n, S: integer; Sr: real;
begin
 write('Количество учащихся ');
 readln(n);
 writeln('Вводите отметки');
 for var i := 1 to n do
  read(a[i]);
 S := 0;
 for var i := 1 to n do
  S := S + a[i];
 Sr := S / n;
 if Sr > 7 then
  writeln('Хорошая')
 else
  if Sr < 4 then
   writeln('Плохая')
  else
   writeln('Средняя');
end.
```

VI. Тестирование.

Окно вывода

```
Количество учащихся 5
Вводите отметки
10 5 6 8 9
Хорошая
```

Пример 4.4.

V. Программа:

```
var Kol, Cen: array[1..50] of
    integer;
  n, Sum: integer;
begin
 write('Введите количество
       видов товаров ');
 readln(n);
 for var i := 1 to n do
 begin
  writeln('Введите количество
      товара', i, ' и его цену ');
  read(Kol[i], Cen[i]);
 end;
 Sum := 0;
 for var i := 1 to n do
  Sum := Sum + Kol[i]*Cen[i];
 writeln('Суммарная стоимость
         товаров =', Sum);
end.
```

II. Результат: одно из слов — «хорошая», «средняя», «плохая» в зависимости от значения среднего балла.

III. Алгоритм решения задачи.

1. Ввод исходных данных. Сначала вводим количество учащихся в классе, затем массив отметок (поэлементно с клавиатуры).

2. Для определения успеваемости нужно вычислить средний балл (переменная Sr). Средний балл определяется как сумма (переменная S) всех отметок, деленная на количество учащихся в классе. Начальное значение для суммы — `S := 0`.

3. В цикле добавляем очередной элемент массива к сумме.

4. Делим полученную сумму на количество учащихся в классе.

5. Проверяем значение среднего балла и выводим результат.

IV. Описание переменных: `a` — `array[1..30] of integer`; `n`, `S` — `integer`; `Sr` — `real`.

4.2. Вычисление сумм и произведений при работе с двумя массивами

Пример 4.4. На складе хранятся товары. Для каждого вида товара известно количество единиц товара и цена за единицу товара. Определить суммарную стоимость всех товаров, хранящихся на складе.

I. Исходные данные: `Cen` — одномерный массив для хранения цены единицы товара каждого вида, `Kol` — массив для хранения количества товара каждого вида, `n` — количество видов товаров.

II. Результат: `Sum` — значение суммарной стоимости товаров на складе.

III. Алгоритм решения задачи.

1. Ввод исходных данных. Для каждого вида товара задается его цена и количество.

2. Стоимость всех товаров одного вида определяется как произведение количества на цену. Суммарная стоимость — сумма всех таких произведений. Начальное значение суммы `Sum := 0`; В цикле к сумме прибавляются произведения `Kol[i]*Cen[i]`.

3. Вывод результата.

IV. Описание переменных: `Cen, Kol — array[1..50] of integer; n, Sum — integer`.

4.3*. Использование массива, элементы которого являются константами

Пример 4.5. Задано натуральное число `n` (`n < 5000`). Определить, является ли это число простым.

I. Исходные данные: `n` — натуральное число.

II. Результат: вывод сообщения «простое» или «составное».

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Известно, что число `n` является простым, если оно не делится ни на одно простое число, не большее $\sqrt{n}$. Максимальное число по условию — 5000, $\sqrt{5000} \approx 70,7107$. Создадим массив констант `s_n` из простых чисел, не больших 71.

3. В цикле будем делить число `n` на каждое из чисел, не больших $\sqrt{n}$

Пример 4.4. *Продолжение.*

VI. Тестирование.

Окно вывода

```
Введите количество видов товаров 3
Введите количество товара 1 и его цену
5 2
Введите количество товара 2 и его цену
7 3
Введите количество товара 3 и его цену
4 5
Суммарная стоимость товаров =51
```

Пример 4.5.

Условие `s_n[i] <= sqrt(n)` проверяется долго за счет вызова функции `sqrt(n)`. Это условие обычно заменяют эквивалентным: `s_n[i]*s_n[i] <= n`.

V. Программа:

```
const s_n: array of integer =
(2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
1, 37, 41, 43, 47, 53, 59, 61, 67, 71);
var n,i: integer;
begin
 writeln('Введите число');
 read(n);
 i := 0;
 while (s_n[i] * s_n[i] <= n)
      and (n mod s_n[i] <> 0) do
  i := i + 1;
 if s_n[i]*s_n[i] > n then
  writeln('Простое')
 else
  writeln('Составное')
end.
```

VI. Тестирование.

Окно вывода

```
Введите число
2027
Простое
```

Окно вывода

```
Введите число
2021
Составное
```

VII. Анализ результатов. Проверить правильность вычислений можно на калькуляторе или посмотреть в таблице простых чисел[1].

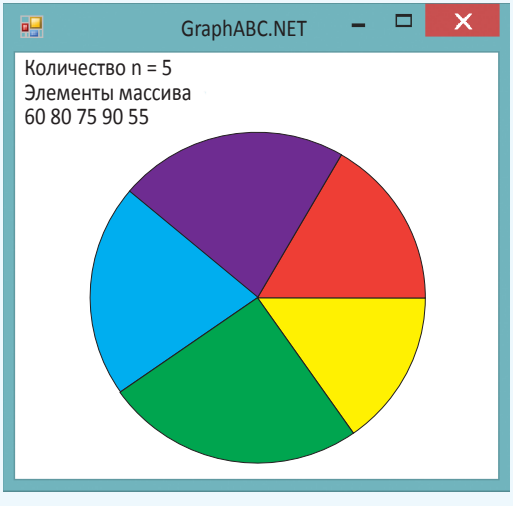
[1] Технические таблицы: http://tehtab.ru/guide/guidemathematics/guidemathematicsfigurestables/simplefigures/ (дата доступа: 10.02.2019).

Пример 4.6.

V. Программа:

```
uses graphABC;
var a: array[1..10] of integer;
    n, S, u0, u1: integer;
begin
 write('Количество n =');
 readln(n);
 writeln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 for var i := 1 to n do
  write(a[i], ' ');
 S := 0;
 for var i := 1 to n do
  S := S + a[i];
 u0 := 0;
 for var i := 1 to n do
 begin
  u1:= u0+trunc(a[i]*360/S);
  SetBrushColor(clRandom);
  Pie(150,150,100,u0,u1);
  u0 := u1;
 end;
end.
```

VI. Тестирование.



и хранящихся в массиве констант. Если число n не разделилось ни на одно из них, то оно — простое, иначе — составное.

4. Проверяем, с каким условием закончил работу цикл: число является простым, если последний просмотренный элемент массива больше $\sqrt{n}$ (число ни на что не разделилось).

5. Вывод результата.

IV. Описание переменных: s_n — const **array of** integer; n, i — integer.

4.4. Построение круговой диаграммы

Пример 4.6. Задан одномерный массив из целых чисел. Построить круговую диаграмму по числовым данным, хранящимся в массиве. Например, для 5 элементов массива — 60, 80, 75, 90, 55.

I. Исходные данные: массив a для хранения данных и n — количество данных.

II. Результат: круговая диаграмма.

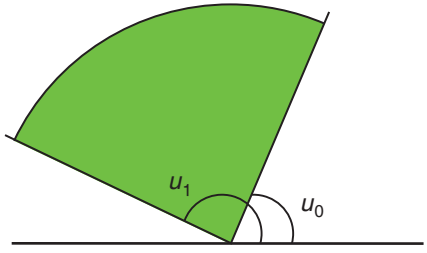
III. Алгоритм решения задачи.

1. Ввод исходных данных. Массив вводится поэлементно с клавиатуры.

2. Круговая диаграмма состоит из n секторов. Градусная мера сектора определяется числовым значением соответствующего элемента в массиве. Суммарное значение всех элементов массива (переменная S) соответствует величине в 360°. Тогда значению элемента массива A[i] будет соответствовать величина — $A[i] * \frac{360}{S}$.

3. Вычисляем сумму всех элементов массива.

4. В цикле строим секторы, градусная мера которых равна целой части величины A[i] * 360 / S.



Для построения сектора нужно знать величины двух углов: u_0 и u_1. Значение $u_1 = u_0 + A[i] * \frac{360}{S}$. Вначале $u_0 = 0$. Затем в цикле меняем значение u_0 на u_1. Цвет сектора будем задавать случайным образом. Для вычисления целой части можно использовать функции `trunc` и `round`.

IV. Описание переменных: a — **array**[1..10] **of** integer; n, S, u_0, u_1 — integer.

Пример 4.6. *Продолжение.*

VII. Постройте по этим данным диаграмму в Excel и сравните.

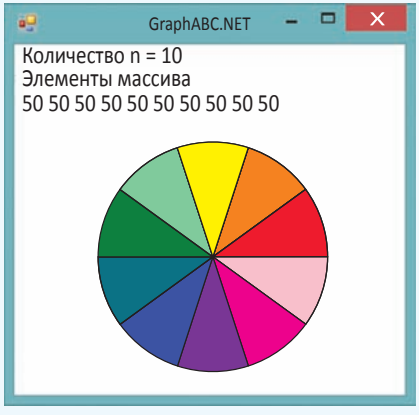
Для задания цвета сектора можно использовать массив, содержащий цветовые константы:

```
const d_color: array [1..10]
of Color = (clRed, clOrange,
clYellow, clLightGreen, clGreen,
clTeal, clBlue, clDarkViolet,
clMagenta, clPink);
```

Команду задания цвета сектора нужно будет заменить на:

```
SetBrushColor(d_color[i]);
```

Результат:



1. Какие операции допустимы для элементов массива целых чисел?
2. Какие операции допустимы для элементов массива вещественных чисел?
3. Как записать данные в массив констант?

Упражнения

1 Для задачи из примера 4.2 выполните перечисленные задания.

1. Заполните таблицу.

	n	a	S	P
1	3	−2 −3 −5		
2	5	1 2 3 4 5		
3	10	1 −3 −2 3 4 3 2 4 3 2		

2. Добавьте в таблицу свои значения n и a.

3. Попытайтесь подобрать такие значения элементов массива, чтобы S = P, для n = 2,5.

4. Для n = 10 ввели все элементы массива, равные 9. Какой результат получили? Почему? Что нужно исправить в программе для получения правильного результата?

2 Для задачи из примера 4.3 добавьте вывод среднего балла.

3 В ходе хоккейного матча удалялись игроки обеих команд. Для каждого удаленного игрока известно время его отсутствия на поле. Определите, какая из команд провела больше времени на скамейке штрафников.

4 Для задачи из примера 4.5 выполните следующее задание:

Введите число 5557. Почему появилась ошибка? Дополните массив констант простыми числами так, чтобы программа могла выдавать ответ для чисел, меньших 10 000. (Для этого можно воспользоваться самой программой или таблицей простых чисел.)

5 Для задачи из примера 4.6 выполните перечисленные задания.

1. Внесите в программу изменения так, чтобы цвет сектора выбирался из массива констант.

2*. Измените программу так, чтобы диаграмма всегда строилась в центре графического окна. Диаметр круга определяется меньшей из двух величин — шириной или высотой окна.

6* В массивах X и Y хранятся координаты точек. Постройте многоугольник, заданный этими координатами. Запросите у пользователя номера двух точек и постройте диагональ многоугольника, соединяющую эти точки.

§ 5. Поиск элементов с заданными свойствами

Человек постоянно сталкивается с задачами поиска требуемой информации. Типичным примером может служить работа со справочниками или библиотечной картотекой. В современном мире информацию ищут с использованием сети Интернет.

Чтобы поиск был результативным и быстрым, разрабатывают эффективные алгоритмы поиска. Важную роль в процессе поиска информации играет способ хранения данных. Одной из самых простых структур для этого является массив.

5.1. Линейный поиск

Рассмотрим, как осуществляется поиск для данных, хранящихся в массиве.

Среди разновидностей простейших задач поиска, встречающихся на практике, можно выделить следующие типы:

1. Найти хотя бы один элемент, равный заданному элементу X. В результате необходимо получить i —

индекс (номер) элемента массива, такой, что a[i] = X.

2. Найти все элементы, равные заданному X. В результате необходимо получить количество таких элементов и (или) их индексы.

Иногда поиск организуется не по совпадению с элементом X, а по выполнению некоторых условий. Примером может служить поиск элементов, удовлетворяющих условию: $X_1 \leqslant a[i] \leqslant X_2$, где X_1 и X_2 заданы.

Если нет никакой добавочной информации о разыскиваемых данных, то самый простой подход — последовательный просмотр элементов массива.

Алгоритм, при котором для поиска нужного элемента последовательно просматривают все элементы массива в порядке их записи, называется **линейным** или **последовательным поиском.**

5.2. Поиск одного элемента, удовлетворяющего условию поиска

Пример 5.1. Задан одномерный массив из n чисел. Определить, есть ли в нем хотя бы один элемент, равный x (значение x вводится).

I. Исходные данные: массив a, количество чисел n, искомое число x.

II. Результат: вывод сообщения «Элемент найден» или «Элемент не найден».

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Пусть p — переменная логического типа, которая имеет значение «истина», если элемент

Алгоритмы поиска можно разделить на алгоритмы, использующие неупорядоченные наборы данных, и на алгоритмы, работающие с предварительно упорядоченным набором данных.

Примером поиска в неупорядоченном наборе данных может служить поиск тетради конкретного учащегося в стопке тетрадей, сданных на проверку. Чтобы найти нужную тетрадь, возможно, придется пересмотреть все. Поиск в словаре — поиск в упорядоченном наборе данных, т. к. все слова расположены в алфавитном порядке.

Пример 5.1.

V. Программа:

```
var a: array[1..10] of integer;
 n, x: integer;
 p: boolean;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 write('Число x =');
 readln(x);
 //линейный поиск элемента
 p := false;
 for var i := 1 to n do
  if a[i] = x then
   p := true;
 if p then
  writeln('Элемент найден')
 else
  writeln('Элемент не найден');
end.
```

VI. Тестирование.

Окно вывода

```
Количество n = 5
Элементы массива
1 2 3 4 5
Число x = 4
Элемент найден
```

Окно вывода

```
Количество n = 5
Элементы массива
1 3 5 7 9
Число x = 6
Элемент не найден
```

Пример 5.2.

V. Программа:

```
var a: array[1..10] of integer;
 n, x, k: integer;
begin
 write('Количество n=');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 write('Число x =');
 readln(x);
 //линейный поиск элемента
 k := 0;
 for var i := 1 to n do
  if a[i] = x then
   k := i;
 if k = 0 then
  writeln('Элемент не найден')
 else
  writeln('Элемент найден
           на месте ', k);
end.
```

VI. Тестирование.

Окно вывода
Количество n = 5 Элементы массива 1 3 3 3 5 Число x = 3 Элемент найден на месте 4

Окно вывода
Количество n = 5 Элементы массива 1 3 5 7 9 Число x = 4 Элемент не найден

Пример 5.3.

Фрагмент программы:

```
//линейный поиск элемента
 k:=1;
 while (k<=n) and (a[k]<>x) do
  k:=k+1;
 if k = n+1 then
  writeln('Элемент не найден')
 else
  writeln('Элемент найден
           на месте ', k);.
```

в массиве найден, и «ложь» — в противном случае. До просмотра элементов массива p := false.

3. В цикле будем просматривать все числа в массиве и сравнивать их с числом x.

4. После окончания поиска возможна одна из двух ситуаций:

4.1. Искомый элемент найден (p := true), т. е. в массиве есть такой элемент a[i], что a[i] = x.

4.2. Весь массив просмотрен, и совпадений не обнаружено (p := false).

5. Вывод результата.

IV. Описание переменных: a — **array**[1..10] **of** integer; n, x — integer; p: boolean.

Часто требуется не только определить, есть ли в массиве искомый элемент, но и установить, на каком месте он находится.

Будем хранить индекс найденного элемента (пример 5.2) в переменной k. После выполнения данного алгоритма по значению переменной k можно определить, есть ли в массиве искомый элемент, и если есть, то где он стоит. Если в массиве несколько таких элементов, то в переменной k будет храниться номер последнего из них. Если такого элемента нет, то значение переменной k не изменится (k останется равным 0).

На практике операцию поиска приходится выполнять достаточно часто, и скорость работы программы находится в прямой зависимости от используемого алгоритма поиска.

В рассмотренных выше алгоритмах требуется просмотреть весь массив, даже в том случае, если искомый элемент находится в массиве на первом месте.

Для сокращения времени поиска можно останавливаться сразу после того, как элемент найден. В этом случае весь массив придется просмотреть только тогда, когда искомый элемент последний или его нет вообще (пример 5.3). Цикл заканчивает работу, когда будет найден искомый элемент либо когда k = n + 1, т. е. элемента, совпадающего с x, не существует.

*При такой реализации на каждой итерации цикла требуется увеличивать индекс k и вычислять логическое выражение. Ускорить поиск можно, упростив логическое выражение. Поместим в конец массива дополнительный элемент со значением x. Тогда совпадение с x обязательно произойдет, и можно не проверять условие a[k] <> x. Такой вспомогательный элемент часто называют **«барьером»** или **«часовым»**, так как он препятствует выходу за пределы массива. В исходном массиве теперь будет n + 1 элемент (пример 5.4).

5.3. Нахождение всех элементов, удовлетворяющих условию поиска

Если требуется определить количество элементов, удовлетворяющих какому-либо условию, то для этого определяют отдельную переменную, значение которой увеличивают на 1 каждый раз, когда найден нужный элемент. Такую переменную называют **счетчиком**. До начала просмотра

Пример 5.4*.
Фрагмент программы:

```
//линейный поиск с барьером
 a[n + 1] := x;
 k := 1;
 while a[k]<>x do
  k := k + 1;
 if k = n + 1 then
  writeln('Элемент не найден')
 else
  writeln('Элемент найден
          на месте ', k);
```

V. Тестирование.

Окно вывода
Количество n = 5 Элементы массива 1 5 6 7 1 Число x = 7 Элемент найден на месте 4

Окно вывода
Количество n = 5 Элементы массива 1 2 3 4 5 Число x = -2 Элемент не найден

В современных языках программирования используются библиотеки, содержащие функции для поиска элементов в массивах (и других структурах данных). В PascalABC.Net такие функции реализованы только для динамических массивов (размер массива может изменяться во время выполнения программы, элементы нумеруются с нуля). Описание функций можно найти в справочнике в разделе «Методы расширения одномерных динамических массивов». Пример использования функции поиска всех элементов массива, больших 5:

```
var c : array of integer;
begin
 setlength(c, 10);
 for var i := 0 to 9 do
  c[i] := random(1, 10);
 println(c);
 c.FindAll(p -> p > 5). Println;
end.
```

Пример 5.5.

V. Программа:

```
var a: array[1..10] of integer;
 n, x, k: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 write('Число x =');
 readln(x); k := 0;
 for var i := 1 to n do
  if a[i] mod x = 0 then
   k := k + 1;
 writeln('В  массиве ', k, '
   элемент(-а,-ов), кратный(-х)', x);
end.
```

VI. Тестирование.

Окно вывода

```
Количество n = 5
Элементы массива
1 2 3 4 5
Число x = 2
В массиве 2 элемент(-а,-ов),
кратный(-х) 2
```

Пример 5.6.

V. Программа:

```
var a, b: array[1..10] of integer;
  n, x, k: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 write('Число x =');
 readln(x); k := 0;
 for var i := 1 to n do
  if a[i] mod x = 0 then
  begin
   k := k + 1; b[k] := i;
  end;
 writeln('В массиве ', k, '
  элемент(-а,-ов), кратный(-х) ', x);
 writeln('Местоположение ');
 for var i := 1 to k do
  write(b[i],' ');
end.
```

элементов массива счетчику нужно задать начальное значение, или, другими словами, **инициализировать** значение переменной. В случае подсчета количества элементов, удовлетворяющих условию, счетчик инициализируется нулем. Для решения задачи нужно просматривать весь массив.

Пример 5.5. Задан одномерный массив из n чисел. Определить количество элементов, кратных x в линейном массиве.

I. Исходные данные: массив a, количество чисел n, искомое число x.

II. Результат: количество элементов, удовлетворяющих условию, — k.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Инициализация счетчика.
3. В цикле будем просматривать все числа в массиве и сравнивать с нулем их остатки от деления на число x. Если остаток равен нулю, то счетчик увеличиваем на 1.
4. Вывод результата.

IV. Описание переменных: a — **array**[1..10] **of** integer; n, x, k — integer.

Если необходимо не только посчитать, сколько элементов удовлетворяют условию, но и сохранить индексы таких элементов, то для этого можно воспользоваться дополнительным массивом. Создадим новый массив b. Как только будет найден необходимый элемент, его индекс будет заноситься в массив b. Переменная k будет хранить номер последнего занятого места в массиве b. Вначале k = 0 (пример 5.6).

После завершения работы первые k элементов массива b будут содержать индексы искомых элементов.

Если для решения задачи потребуются значения всех найденных элементов, то в программе возможно такое обращение к элементам массива: a[b[i]]. Адрес элемента в массиве a будет определяться значением элемента массива b по адресу i. Для вывода значений элементов в примере 5.6 последний цикл нужно заменить на

```
for var i := 1 to k do
  write(a[b[i]],' ');
```

5.4. Решение задач с использованием алгоритма линейного поиска

Пример 5.7. Известны результаты ЦТ по математике для n человек. Определить, есть ли среди них хотя бы один человек с баллом выше x. Значение x вводится с клавиатуры. Результаты экзамена получить случайным образом.

I. Исходные данные: массив a, количество чисел n, число x.

II. Результат: сообщение соответствует условию задачи.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Просмотр элементов с начала. Как только элемент найден, остановимся.
3. Если весь массив просмотрен, значит, в исходном массиве нет элемента, удовлетворяющего условию задачи, иначе выводим номер найденного элемента.

IV. Описание переменных: a — **array**[1..20] **of** integer; n, x, k — integer.

Пример 5.6. *Продолжение.*
VI. Тестирование.

Окно вывода

```
Количество n = 5
Элементы массива
6 3 2 4 5
Число x = 2
В массиве 3 элемент(-а,-ов),
кратный(-х) 2
Местоположение
1 3 4
```

Пример 5.7.
V. Программа:

```
var a: array[1..20] of integer;
 n, x, k: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
 begin
  a[i] := random(0,100);
  write(a[i],' ');
 end;
 writeln;
 write('Число x =');
 readln(x);
 //линейный поиск с барьером
 a[n+1] := x + 1;
 k := 1;
 while (a[k] <= x) do
  k := k + 1;
 if k = n+1 then
  writeln('Нет таких')
 else
  writeln('Это человек с № ',
          k, ', его балл - ', a[k]);
end.
```

VI. Тестирование.

Окно вывода

```
Количество n = 10
Элементы массива
48 12 96 48 9 95 5 71 77 24
Число x = 80
Это человек с №3, его балл - 96
```

Пример 5.8.
V. Программа:

```
uses graphABC;
var X,Y: array [1..1000] of
    integer;
  n, k1, k2, R: integer;
begin
 write('Количество точек n =');
 read(n);
 writeln(n);
 for var i := 1 to n do
 begin
  X[i]:= random(-200,200);
  Y[i]:= random(-200,200);
 end;
 writeln('Радиус окружности');
 read(R);
 writeln(R);
 {Построение окружности и
 осей координат}
 circle(200,200,R);
 line(0,200,400,200);
 line(200,0,200,400);
 k1 := 0; k2 := 0;
 for var i := 1 to n do
  if X[i]*X[i]+Y[i]*Y[i]<=R*R then
  begin
   k1 := k1 + 1;
   SetPixel(X[i]+200,200-Y[i],
            clred);
  end
  else
  begin
   k2 := k2 + 1;
   SetPixel(X[i]+200,200-Y[i],
            clblue);
  end;
  if k1 > k2 then
   writeln('Внутри больше')
  else
  if k1<k2 then
    writeln('Снаружи больше')
   else
    writeln('Поровну')
end.
```

Пример 5.8. В двух линейных массивах X и Y, заданных случайным образом, хранятся координаты точек плоскости ($-200 \leqslant X[i], Y[i] \leqslant 200$). Определить, каких точек больше — лежащих внутри или снаружи области, ограниченной окружностью радиуса R с центром в начале координат (будем считать, что точки, лежащие на окружности, лежат внутри области). Построить окружность и точки. Точки, принадлежащие внутренней области, нарисовать красным цветом, а внешней области — синим цветом.

I. Исходные данные: X, Y — массивы чисел, R — радиус окружности, n — количество точек.

II. Результат: рисунок, соответствующий условию задачи, и сообщение: «Внутри точек больше», «Снаружи точек больше» или «Точек поровну».

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Инициализация счетчиков: k1:= 0; k2:= 0.

3. Будем просматривать все точки и для каждой проверять принадлежность области. Если $X^2 + Y^2 \leqslant R^2$, то точка лежит внутри области, тогда увеличим значение счетчика k1 на 1, если нет, то увеличим на 1 значение счетчика k2.

4. Сравним значения k1 и k2 и выведем результат.

5. Поскольку координаты точек принадлежат отрезку [–200, 200], то оси координат можно нарисовать пересекающимися в точке с координатами (200, 200). Для преобразования координат точек в экранные нужно к значению аб-

сциссы прибавить 200, а значение ординаты нужно отнять от 200 (ось Y на экране направлена вниз, поэтому нужно поменять знак ординаты). Строить точки можно в том же цикле, в котором происходит проверка.

IV. Описание переменных: X, Y — **array**[1..1000] **of** integer; n, k1, k2, R — integer.

Пример 5.9. На складе хранятся пустые ящики для упаковки товара. Известно, что масса одного пакета с конфетами x кг. Какова суммарная масса пакетов с конфетами, которые можно упаковать в такие ящики, заполнив ящик целиком?

I. Исходные данные: массив a, количество чисел n, число x.

II. Результат: количество ящиков — k, суммарная масса — S.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Инициализация счетчика и значения суммы: k := 0; S := 0.

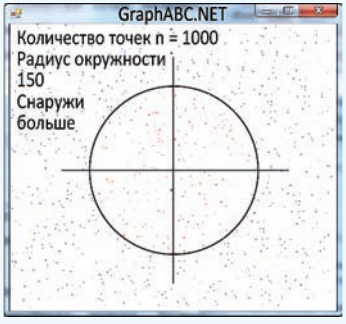
3. Просматривая массив, проверим, является ли текущий элемент числом, кратным x (в этом случае ящик будет заполнен целиком). Как только элемент найден, увеличим счетчик k на 1, а переменную S — на значение найденного элемента массива.

4. Вывод результата.

IV. Описание переменных: a — **array**[1..20] **of** integer; n, x, k, S — integer.

Пример 5.10. Имеется список мальчиков 10 В класса и результаты их бега на 100 м. Для сдачи норматива

Пример 5.8. *Продолжение.*
VI. Тестирование.



Пример 5.9.
V. Программа:

```
var a: array [1..20] of integer;
    n, x, k, S: integer;
begin
 write('Количество ящиков:');
 readln(n);
 writeln('Вместимость ящиков');
 for var i := 1 to n do
  read(a[i]);
 write('Масса конфет:'); read(x);
 k := 0; S := 0;
 for var i := 1 to n do
  if a[i] mod x = 0 then
  begin
      k := k + 1;
      S := S + a[i];
  end;
 writeln('На складе',k,'ящ.');
 writeln('Суммарная масса', S);
end.
```

VI. Тестирование.

Окно вывода

```
Количество ящиков: 8
Вместимость ящиков
15 23 64 27 35 10 48 13
Масса конфет: 5
На складе 3 ящ.
Суммарная масса 60
```

VII. Анализ результатов. Ящики, удовлетворяющие условию задачи, имеют вес — 15, 25 и 10.

Пример 5.10.

V. Программа:

```
var r: array [1..20] of real;
  fam: array [1..20] of string;
  n, k: integer;
begin
 writeln('Количество учащихся: ');
  readln(n);
 writeln('Фамилия и результат: ');
  for var i := 1 to n do
  begin
    readln(fam[i]);
    readln(r[i]);
  end;
  writeln('Фамилии не сдавших
           норматив:');
  k := 0;
  for var i := 1 to n do
    if r[i] > 16 then
    begin
      k := k + 1;
      writeln(fam[i]);
    end;
 writeln('Не сдали норматив:', k);
end.
```

VI. Тестирование.

Окно вывода

```
Количество учащихся:
5
Фамилия и результат:
Иванов
13.5
Петров
14.0
Сидоров
21
Королев
16.1
Веремей
15.9
Фамилии не сдавших норматив:
Сидоров
Королев
Не сдали норматив: 2
```

VII. Анализ результатов. Результат Сидорова 21, а Королева 16.1, что превышает норматив.

необходимо пробежать дистанцию не более чем за 16 с. Вывести фамилии учащихся, которые не выполнили норматив по бегу. Сколько таких учащихся в классе?

I. Исходные данные: массивы `fam` (фамилии учащихся) и `r` (результаты бега в секундах), количество учащихся `n`.

II. Результат: фамилии тех учащихся, которые не выполнили норматив по бегу.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Инициализация счетчика: `k := 0`.
3. Будем просматривать массив с результатами и проверять, является ли текущий элемент числом больше 16 (норматив не сдан). Если такое значение найдено, то выведем элемент массива `fam` с соответствующим номером и увеличим значение счетчика на 1.
4. Вывод значения счетчика.

IV. Описание переменных: `fam` — **array**[1..20] **of** string; `r` — **array**[1..20] **of** real, `n`, `k` — integer.

Пример 5.11*. Задан одномерный массив из `N` целых чисел. Определить количество элементов, которые являются числами Смита. (Число Смита — это такое составное число, сумма цифр которого равна сумме цифр всех его простых сомножителей.) Например, числом Смита является $202 = 2 \times 101$, поскольку $2+0+2=4$, и $2+1+0+1=4$.

I. Исходные данные: `a` — массив чисел, `n` — количество чисел в массиве.

II. Результат: числа Смита и их количество в массиве.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Инициализация счетчика: k := 0.

3. Будем просматривать каждый элемент массива и определять, является ли он числом Смита. Для проверки создадим функцию check, которая будет получать в качестве параметра элемент массива, а также возвращать значение true, если число является числом Смита, и false в противном случае.

3.1. Найдем сумму цифр числа.

3.2. Будем раскладывать число на простые множители и для каждого множителя находить сумму цифр.

3.3. Для разложения числа на простые множители будем делить его сначала на 2 (пока делится), затем на 3. На 4 число уже делиться не будет, будем делить его на 5 и т. д. Закончится разложение тогда, когда после всех делений число станет равным 1.

4. Также нам понадобится функция sum, которая для числа будет возвращать его сумму цифр.

IV. Описание переменных: a — **array**[1..100] **of** integer; n, k — integer.

Пример 5.12*. Задан одномерный массив из n строк. Каждая строка является предложением из слов, разделенных пробелами. Найти и вывести те предложения, в которых нечетное количество слов.

Пример 5.11*.

V. Программа:

```
var a: array [1..100] of
   integer;
   n, k: integer;

function sum(x: integer):
         integer;
var s: integer;
begin
 s := 0;
 while x > 0 do
 begin
  s := s + x mod 10; x := x div 10;
 end;
 sum := s;
end;

function check(x: integer):
         boolean;
var s1, s2, d: integer;
begin
 s1 := sum(x); s2 := 0; d := 2;
//разложение на простые множители
 while x <> 1 do
 begin
  while x mod d = 0 do
  begin
   s2 := s2 + sum(d);
   x := x div d;
  end;
  d := d + 1;
 end;
 check := s1 = s2;
end;
begin
 writeln('Количество');
 readln(n);
 writeln('Элементы');
 for var i := 1 to n do
  read(a[i]);
 k := 0;
 writeln('Числа Смита');
 for var i := 1 to n do
  if check(a[i]) then
  begin
   inc(k);
   write(a[i], ' ');
  end;
 writeln;
 writeln('Всего - ', k);
end.
```

Пример 5.11*. *Продолжение.*
VI. Тестирование.

Окно вывода	Окно вывода
Количество 5 Элементы 202 3 323 85 117 Числа Смита 202 3 85 Всего - 3	Количество 5 Элементы 8 8 8 8 8 Числа Смита Всего - 0

Пример 5.12.
V. Программа:

```
var a: array [1..100] of
   string;
   n, k: integer;

function check(x: string):
         integer;
var
 s, len: integer;
begin
 s := 0;
 x := ' ' + x;
 len := length(x);
 for var i := 1 to len - 1 do
  if (x[i] = ' ') and (x[i + 1] <> ' ')
  then
    inc(s);
 check := s;
end;

begin
 writeln('Количество');
 readln(n);
 writeln('Элементы');
 for var i := 1 to n do
  readln(a[i]);
 k := 0;
 writeln;
 writeln('Искомые строки:');
 for var i := 1 to n do
  if check(a[i]) mod 2 <> 0 then
  begin
   inc(k);
   writeln(a[i]);
  end;
 writeln('Всего - ', k);
end.
```

I. Исходные данные: a — массив строк, n — количество строк в массиве.

II. Результат: искомые строки и их количество.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Инициализация счетчика: k := 0.

3. Будем просматривать каждую строку и определять, сколько в ней слов. Для проверки создадим функцию check, которая будет получать в качестве параметра элемент массива и возвращать количество слов в строке. Если количество слов является нечетным числом, то выведем строку и увеличим значение счетчика.

3.1. Перед каждым словом предложения, кроме первого, стоит пробел, слово начинается с символа, который пробелом не является.

3.2. Добавим пробел перед первым словом, тогда количество слов будет определяться количеством сочетаний пар символов: пробел и не пробел.

IV. Описание переменных: a — **array**[1..100] **of** string; n, k — integer.

При вводе данных для тестирования программы нужно помнить, что после каждого предложения необходимо нажимать клавишу Enter.

В данном случае строка как тип данных может не соответствовать строке в окне вывода. В примере 5.12 вводятся 3 строки:

1. «Многие компании разрабатывают свои правила по оформлению кода».

2. «В них прописаны также правила именования переменных».

3. «Компания Microsoft использует так называемую «венгерскую нотацию».

В окне вывода количество строк может быть больше (на рисунке их 6).

То, как будут выглядеть вводимые строки в окне вывода, зависит от ширины окна приложения PascalABC.NET. На большом мониторе, если окно приложения развернуто на весь экран, строк может быть три.

Компилятор определяет, что ввод строки закончен, если была нажата клавиша Enter. Внешний вид строк в окне вывода для компилятора не имеет значения.

Пример 5.12. *Продолжение.*
VI. Тестирование.

Окно вывода

```
Количество
3
Элементы
Многие компании разрабатывают
свои правила по оформлению кода.
В них прописаны также правила
именования переменных.
Компания Microsoft использует так
называемую «венгерскую нотацию».

Искомые строки
В них прописаны также правила
именования переменных.
Компания Microsoft использует так
называемую «венгерскую нотацию».

Всего - 2
```

1. Что называют последовательным поиском?
2. Как определить, что в массиве был найден элемент с определенными свойствами?
3. Для чего используют переменные «счетчики»?
4. Что такое инициализация переменной?

Упражнения

1 Для примеров 5.1—5.3 выполните перечисленные задания.

1. Заполните таблицу.

№	n	Массив	x	Результат
1	5	2 14 7 20 16	20	
2	5	2 3 4 5 6	8	
3	7	2 4 6 8 10 11 12	8	
4	5	6 3 9 12 15	3	

2. Добавьте в таблицу свои данные, такие, чтобы искомый элемент был первым в массиве; последним в массиве.
3. Какой ответ выдаст каждая из программ, если в массиве несколько элементов, удовлетворяющих условию задачи? Почему?
4. Что нужно изменить в программе 5.2, чтобы выдавался не последний из найденных элементов, а первый?

5. Что нужно изменить в программе 5.1, чтобы выдавался не последний из найденных элементов, а первый?

6. Измените условие цикла **while** примера 5.3 так, чтобы использовалась логическая операция not.

2 Рост учащихся класса представлен в виде массива. Определите количество учащихся, рост которых больше среднего роста по классу.

3 Заданы фамилии и рост учащихся 10-го класса. Вывести фамилии тех учащихся, рост которых меньше среднего роста по классу.

4 Известны данные о площади n стран (в млн кв. км) и численности населения (в млн жителей). Выведите номера тех стран, плотность населения которых больше X.

5 Для упражнения 4 добавьте возможность вводить и выводить названия стран.

6 Определите, есть ли в линейном массиве хотя бы один элемент, который является нечетным числом, кратным 7. Если да, то следует вывести его номер.

7* В линейном массиве найдите и выведите все простые числа с нечетной суммой цифр. Укажите, сколько чисел вывели.

8* В линейном массиве найдите и выведите все числа Армстронга. (Числом Армстронга называется такое число, которое равно сумме своих цифр, возведенных в степень, равную количеству его цифр. Например, числом Армстронга является число 371 : $371 = 3^3 + 7^3 + 1^3 = 27 + 343 + 1$.) Укажите, сколько чисел вывели.

9 Задан одномерный массив из N строк. Каждая строка является предложением из слов, разделенных пробелами. Найдите и выведите те предложения, в которых есть слова, начинающиеся на гласную (строчную или прописную).

§ 6. Максимальный и минимальный элементы массива

Пример 6.1.
V. Программа:

```
var a: array[1..20] of integer;
 n, max: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 max := a[1];
 for var i := 2 to n do
  if a[i] > max then
   max := a[i];
 writeln('Максимум = ', max);
end.
```

6.1. Поиск максимального (минимального) элемента в массиве

Очень часто для решения задачи требуется находить не заданный элемент массива, а максимальный (наибольший) или минимальный (наименьший).

Рассмотрим задачу нахождения максимального элемента. Если в массиве один-единственный элемент, то он и есть максимальный. Если элементов больше одного, то максимальным в массиве из i элементов является максимум из a[i] и максимального

среди первых i-1 элементов. Находить максимум будем последовательно, сравнивая текущий элемент с максимумом, найденным на предыдущем шаге. Если текущий элемент больше, то значение максимума, найденное на предыдущем шаге, нужно обновить (пример 6.1).

Данный алгоритм находит значение максимального элемента, но не позволяет определить, на каком месте в массиве расположен этот максимальный элемент.

Будем использовать переменную n_max для хранения индекса максимального элемента. Значение переменной n_max будет изменяться тогда, когда изменяется значение максимального элемента (пример 6.2).

Если в массиве несколько элементов имеют максимальное значение, то значением переменной n_max будет индекс первого из них. Если использовать условие a[i] ⩾ max, то переменная n_max будет хранить индекс последнего из максимальных элементов.

В случае, когда известен индекс i элемента массива, значение элемента можно получить, обратившись к элементу по индексу: a[i]. Поэтому при поиске максимального элемента достаточно хранить только его индекс n_max. Значение максимального элемента — a[n_max] (пример 6.3).

Для поиска минимального элемента необходимо заменить знак > в условии оператора ветвления на знак < (пример 6.4).

Пример 6.1. *Продолжение.*
VI. Тестирование.

Окно вывода
```
Количество n = 5
Элементы массива
1 2 6 3 2
Максимум = 6
```

Пример 6.2.
V. Программа:

```
var a: array[1..20] of integer;
    n, max, n_max: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 max := a[1]; n_max := 1;
 for var i := 2 to n do
  if a[i] > max then
  begin
   max := a[i]; n_max :=i;
  end;
 writeln('Максимум = ', max);
 writeln('Его место ', n_max);
end.
```

VI. Тестирование.

Окно вывода
```
Количество n = 5
Элементы массива
2 5 3 5 1
Максимум = 5
Его место 2
```

Пример 6.3. Фрагмент программы:

```
n_max := 1;
for var i := 2 to n do
 if a[i] > a[n_max] then
  n_max := i;
writeln('Максимум =',a[n_max]);
writeln('Его место ', n_max);
```

Пример 6.4. Фрагмент программы:

```
n_min := 1;
for var i := 2 to n do
 if a[i] < a[n_min] then
  n_min := i;
```

Пример 6.5.
V. Программа:

```
var a: array [1..20] of real;
    n, n_min: integer;
begin
 writeln('Количество
       спортсменов');
 readln(n); writeln('Время');
 for var i := 1 to n do
  read(a[i]);
 //поиск минимального элемента
 n_min := 1;
 for var i := 2 to n do
  if a[i] < a[n_min] then
    n_min := i;
 writeln('Победитель – лыжник
        номер ', n_min);
 writeln('Его время - ', a[n_min]);
end.
```

VI. Тестирование.

Окно вывода

```
Количество спортсменов
5
Время
6.31 6.17 7.32 6.54 7.03
Победитель – лыжник номер 2
Его время - 6.17
```

Пример 6.6.
V. Программа:

```
var a: array [1..20] of integer;
    n, min, k : integer;
begin
 write('Количество n =');
 readln(n); writeln('Числа');
 for var i := 1 to n do
  read(a[i]);
 //поиск минимального элемента
 min := a[1];
 for var i := 2 to n do
  if a[i] < min then
     min := a[i];
 //подсчет количества
 k := 0;
 for var i := 1 to n do
  if a[i] = min then
     k := k + 1;
 writeln('Минимальный ', min);
 writeln('Встретился ',k,
        ' раз(-а)');
end.
```

6.2. Решение задач с использованием алгоритма поиска максимального (минимального) элемента

Пример 6.5. В массиве хранится информация о результатах спортсменов, участвующих в лыжной гонке. Определить результат победителя и его номер.

I. Исходные данные: массив `a` — числа, являющиеся временем прохождения трассы, количество спортсменов — `n`.

II. Результат: `a[n_min]` — минимальное время, `n_min` — номер победителя.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Для решения задачи воспользуемся алгоритмом поиска минимального элемента в массиве и его номера (пример 6.4).
3. Вывод результата.

IV. Описание переменных: `a` — **array**`[1..20]` **of** `real`; `n`, `n_min` — `integer`.

Пример 6.6. Определить, сколько раз в линейном массиве встречается элемент, равный минимальному.

I. Исходные данные: массив `a`, количество чисел `n`.

II. Результат: `min` — минимальный элемент, `k` — количество минимальных.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Поиск минимального элемента.
3. Линейный поиск элементов, равных минимальному.
4. Вывод результата.

IV. Описание переменных: a — **array**[1..20] **of** integer; n, min, k — integer.

Пример 6.7. Задан массив из слов различной длины. Найти в нем самое длинное и самое короткое слово.

I. Исходные данные: массив a, количество слов n.

II. Результат: min_s — самое короткое слово, max_s — самое длинное слово.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Поиск самого короткого слова. Самое короткое слово — слово, в котором минимальное количество символов. Для его поиска можно воспользоваться алгоритмом поиска минимального элемента в массиве. Однако, если сравнивать сами элементы массива, то сравнение будет происходить не по длине[1]. Для сравнения строк по длине нужно использовать функцию вычисления длины строки length.

3. Для поиска самого длинного слова можно использовать алгоритм поиска максимального элемента и сравнивать элементы с использованием функции вычисления длины строки length.

4. Вывод результата.

IV. Описание переменных: a — **array**[1..20] **of** string; n — integer; min_s, max_s: string;

Пример 6.6. *Продолжение.*
VI. Тестирование.

Окно вывода
```
Количество n = 5
Числа
3 1 2 1 1
Минимальный 1
Встретился 3 раза(-а)
```

Пример 6.7.
V. Программа:

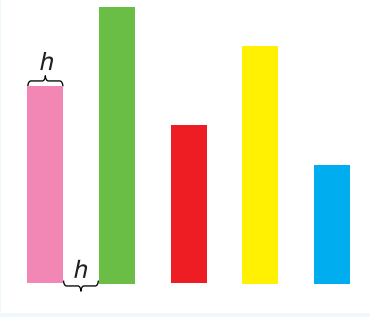
```
var a: array [1..20] of string;
    n: integer;
    min_s, max_s: string;
begin
 write('Количество n =');
 readln(n); writeln('Слова');
 for var i := 1 to n do
  readln(a[i]);
 //поиск короткого слова
 min_s := a[1];
 for var i := 2 to n do
  if length(a[i]) < length(min_s)
  then
    min_s := a[i];
 //поиск длинного слова
 max_s := a[1];
 for var i := 2 to n do
  if length(a[i]) > length(max_s)
  then
    max_s := a[i];
 writeln('Короткое - ',min_s);
 writeln('Длинное - ',max_s);
end.
```

VI. Тестирование.

Окно вывода
```
Количество n = 5
Слова
Все
дороги
ведут
в
Рим
Короткое - в
Длинное - дороги
```

[1] Сравнение строк осуществляется **лексикографически**: s1 < s2, если для первого несовпадающего символа с номером i верно неравенство s1[i] < s2[i] или все символы строк совпадают, но s1 короче s2.

Пример 6.8.



V. Программа:

```
uses graphABC;
var a: array[1..20] of integer;
 n, max, h, x, y1, y2: integer;
 m: real;
begin
 write('Количество n =');
 readln(n);
 writeln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
 begin
  read(a[i]);
  write(a[i],' ');
 end;
 max := a[1];
 for var i := 2 to n do
  if a[i] > max then
   max := a[i];
 h := trunc(WindowWidth/(2*n+1));
 m := WindowHeight/max;
 x := h;
 for var i := 1 to n do
 begin
  SetBrushColor(clrandom);
  y1 := WindowHeight;
  y2 := y1 - trunc(a[i]*m);
  Rectangle(x, y1, x+h, y2);
  x := x + 2*h;
 end;
end.
```

6.3. Построение гистограммы (столбчатой диаграммы)

Пример 6.8. Дан одномерный массив из целых чисел. Построить гистограмму по числовым данным, хранящимся в массиве.

I. Исходные данные: массив a, количество чисел n.

II. Результат: построенная диаграмма.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Гистограмма состоит из n прямоугольников одинаковой ширины. Элементы массива определяют высоту соответствующего прямоугольника. Максимальное значение элементов массива (переменная max) должно по высоте поместиться в окне (WindowHeight). Обозначим $m = \frac{\text{WindowHeigth}}{\text{max}}$ (масштабный коэффициент). Тогда значению элемента массива a[i] будет соответствовать целая часть от величины a[i] * m.

3. Находим максимальный элемент массива.

4. В цикле строим прямоугольники. Все прямоугольники имеют одинаковую ширину (h), расстояние между ними можно определить равным ширине прямоугольника. Тогда ширина прямоугольника — целая часть от деления ширины окна на $(2n + 1)$:

$$h = \frac{\text{WindowMidth}}{2n + 1}.$$

5. При вычислении высоты прямоугольника нужно учесть то, что ось Y направлена сверху вниз.

6. Цвет прямоугольника будем задавать случайным образом.

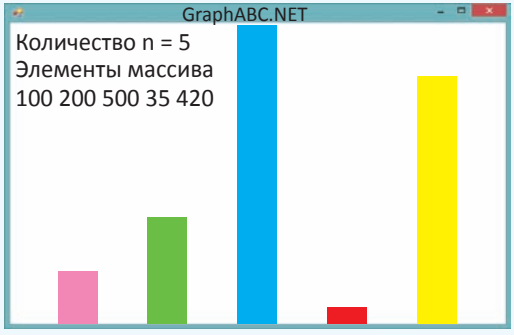
7. Местоположение прямоугольника определяется переменной x. Начальное значение x = h. Новое значение получается из предыдущего увеличением на 2*h.

8. Вывод результата.

IV. Описание переменных: `a — array[1..20] of integer; n, max, h, x, y1, y2 — integer; m: real.`

Пример 6.8. *Продолжение.*

VI. Тестирование.



VII. Постройте по этим данным диаграмму в Excel и сравните.

1. Какой элемент массива является максимальным? Какой — минимальным?
2. Как найти максимальный элемент в массиве?
3. Как найти минимальный элемент?
4. Каким образом определить номер первого элемента, равного максимальному?
5. Как определить номер последнего элемента, равного минимальному?

Упражнения

1 Измените программы примеров 6.1 и 6.2 так, чтобы находился минимальный элемент в массиве.

2 Для примера 6.5 выполните перечисленные задания.

1. Найдите номер спортсмена, пришедшего на финиш последним.
2. Определите, был ли победитель единственным или есть еще лыжник, прошедший трассу с таким же результатом (см. пример 6.6).
3. Добавьте еще один массив. Введите в него фамилии спортсменов. Реализуйте пункты 1—3 так, чтобы выводилась фамилия, а не номер (см. пример 5.10).

3 В массиве хранится информация о стоимости автомобилей. Определите стоимость самого дорогого автомобиля и его номер в массиве. Если есть несколько таких автомобилей, то выведите все номера.

4 В массиве хранится информация о среднедневной температуре декабря. Определите, сколько в декабре было дней с самой низкой и с самой высокой температурой.

5 Задан массив из слов. Найдите в нем самое длинное слово, заканчивающееся буквой *а*.

6* Задан массив из слов. Найдите в нем самое короткое слово, начинающееся с прописной буквы.

7 Для примера 6.8 выполните перечисленные задания.

1. Измените программу так, чтобы при построении диаграммы использовалась не вся высота окна, а оставались поля сверху и снизу.
2. Измените программу так, чтобы столбики строились без промежутков между ними.
3. Создайте массив цветовых констант и используйте эти цвета для закрашивания столбиков.
4. Постройте линейчатую диаграмму.

8 По данным массива постройте диаграмму в виде ломаной линии. Соответствуют ли ординаты вершин ломаной значениям массива?

§ 7. Преобразование элементов массива

Пример 7.1.
V. Программа:

```
var a: array[1..20] of integer;
    n: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 for var i := 1 to n do
 begin
  if a[i] > 0 then
     a[i] := a[i] * 2;
  if a[i] < 0 then
     a[i] := a[i] + 5;
 end;
 writeln('Преобразованный массив');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тестирование.

Окно вывода

```
Количество n = 5
Элементы массива
3 -2 0 -1 5
Преобразованный массив
6 3 0 4 10
```

VII. Анализ результатов. Элементы 3 и 5 увеличены в 2 раза, элементы −2 и −1 увеличены на 5, элемент 0 остался неизмененным.

7.1. Основные задачи

Среди задач преобразования элементов массива можно выделить задачи следующих типов:

1. Изменение элементов массива в зависимости от условий.
2. Обмен местами элементов массива.
3. Удаление элемента из массива.
4. Вставка элемента в массив.

Рассмотрим каждую из задач.

7.2. Изменение элементов массива в зависимости от выполнения некоторых условий

Пример 7.1. Задан одномерный массив целых чисел. Преобразовать его элементы по следующему правилу: положительные элементы увеличить в 2 раза, а отрицательные — увеличить на 5.

I. Исходные данные: одномерный массив a, количество элементов n.

II. Результат: преобразованный массив a.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. В цикле проверяем текущий элемент. Если он положительный,

умножаем его на 2. Если элемент отрицательный, то прибавляем к нему 5. Помним, что отрицание условия «элемент положительный» — условие «элемент не положительный», что подразумевает возможность равенства элемента нулю. Поэтому нужны два оператора ветвления для проверки условия задачи.

3. Вывод результата.

IV. Описание переменных: a — **array**[1..20] **of** integer; n — integer.

7.3. Обмен местами элементов в массиве

Для обмена местами двух элементов массива можно использовать дополнительную переменную, которую называют буфером. Буферу присваивают значение одного из элементов массива, этому элементу присваивают значение другого элемента массива, затем второму элементу присваивают значение буфера:

```
buf := a[i];
a[i] := a[k];
a[k] := buf;
```

Пример 7.2. Задан одномерный массив целых чисел. Поменять местами максимальный и минимальный элементы массива (минимальный и максимальный элементы встречаются в массиве только один раз).

I. Исходные данные: одномерный массив a, количество элементов n.

II. Результат: преобразованный массив a.

III. Алгоритм решения задачи.

1. Введем исходные данные.

2. Найдем максимальный элемент массива и его индекс (n_max).

Если обмен элементов осуществлять следующим образом:

```
a[i] := a[k]; a[k] := a[i];
```

то мы потеряем значение элемента, стоящего изначально на месте a[i], и получим два элемента со значением, равным a[k]. Для обмена элементов можно использовать встроенную функцию swap: swap(a[i], a[k]).

Пример 7.2.

V. Программа:

```
var a: array[1..20] of integer;
  n, n_min, n_max, buf: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 n_min := 1;
 n_max := 1;
 for var i := 1 to n do
 begin
  if a[i] > a[n_max] then
   n_max := i;
  if a[i] < a[n_min] then
   n_min := i;
 end;
 buf := a[n_min];
 a[n_min] := a[n_max];
 a[n_max] := buf;
 writeln('Преобразованный
        массив');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тестирование.

Окно вывода

```
Количество n = 5
Элементы массива
2 1 3 5 4
Преобразованный массив
2 5 3 1 4
```

Пример 7.3.
V. Программа:

```
var a: array[1..20] of integer;
    n, d, j: integer;
procedure del_mas(k: integer);
begin
 for var i := k + 1 to n do
  a[i - 1] := a[i];
 n := n - 1;
end;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 d := 0;
 j := 1;
 while j <= n do
 begin
  if a[j] mod 5 = 0 then
  begin
   del_mas(j);
   d := d + 1;
   j := j - 1;
  end;
  j := j + 1;
 end;
 writeln('Удалили ', d,
   'элемент(-а, -ов');
 writeln('Преобразованный
    массив');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тестирование.

Окно вывода

```
Количество n = 7
Элементы массива
5 3 15 35 10 4 30
Удалили 5 элемент(-а,-ов)
Преобразованный массив
3 4
```

VII. Анализ результатов. Элементы 5, 15, 35, 10 и 30 кратны 5, поэтому их удалили из массива. Элементы 3 и 4 не кратны 5, поэтому они остались в массиве и сдвинулись, соответственно, на 1-е и 2-е места.

3. Найдем минимальный элемент массива и его индекс (n_min).

4. Поменяем местами элементы, стоящие на местах n_max и n_min.

5. Выведем результат.

IV. Описание переменных: a — **array**[1..20] **of** integer; n, n_min, n_max, buf — integer.

7.4*. Удаление элемента из массива

Для удаления элемента массива на месте k нужно сдвинуть на одну позицию влево все элементы, стоящие после него. Количество элементов при этом уменьшаем на 1.

```
for var i := k + 1 to n do
        a[i-1] := a[i];
n := n - 1;
```

Если в массиве нужно удалить не один, а несколько элементов, удовлетворяющих условию, то можно использовать вспомогательный алгоритм в виде соответствующей процедуры **procedure** del_mas(k: integer); Параметр k — номер удаляемого элемента.

Пример 7.3. Задан одномерный массив целых чисел. Удалить из линейного массива все числа, кратные 5. Сколько чисел удалили?

I. Исходные данные: одномерный массив a, количество элементов n.

II. Результат: преобразованный массив a и количество удаленных чисел d.

III. Алгоритм решения задачи.

1. Введем исходные данные.

2. Будем последовательно просматривать элементы массива. Если найдем число, кратное 5, то удалим его из массива, используя процедуру del_mas. Так как количество удаляемых элементов

заранее не известно, то применим цикл **while**.

3. При удалении элемента счетчик d будем увеличивать на 1.

4. Выведем результат.

IV. Описание переменных: a — **array**[1..20] **of** integer; n, d, j — integer.

7.5*. Вставка элемента в массив

Для вставки элемента на место k нужно освободить данное место в массиве. Для этого сдвинем на одну позицию вправо все элементы массива, стоящие после k – 1. Сдвиг начинаем с последнего элемента. Количество элементов в массиве увеличится на 1.

Пример 7.4. Задан массив целых чисел. Вставить число x на k-е место.

I. Исходные данные: одномерный массив a, количество элементов n, число, которое нужно вставить в массив x, номер позиции в массиве, на которую нужно вставить число k.

II. Результат: преобразованный массив a.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Сдвигаем все элементы массива, стоящие после k – 1 на одну позицию вправо.

3. Увеличим n — количество элементов.

4. Вставляем число x на место k.

5. Выводим результат.

IV. Описание переменных: a — **array**[1..20] **of** integer; n, k, x — integer.

Пример 7.4.
V. Программа:

```
var a: array[1..20] of integer;
    n, k, x: integer;
begin
 write('Количество n =');
 readln(n);
 writeln('Элементы массива');
 for var i := 1 to n do
  read(a[i]);
 write('Число x =');
 readln(x);
 write('Номер позиции k =');
 readln(k);
 //сдвиг элементов вправо на 1
 for var i:= n downto k do
  a[i+1] := a[i];
 //вставка x на место k
 a[k] := x;
 n := n + 1;
 writeln('Преобразованный
    массив');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тестирование.

Окно вывода

```
Количество n = 5
Элементы массива
3 2 5 -3 7
Число x = 6
Номер позиции k = 2
Преобразованный массив
3 6 2 5 -3 7
```

Методы расширения одномерных динамических массивов позволяют преобразовывать массивы с помощью встроенных функций ConvertAll, Replace, Transform и др. (см. справочную систему PascalABC.Net).

1. Какие типы задач преобразования массивов вы можете назвать?
2. Как можно поменять местами два элемента в массиве?
3. Как удалить элемент из массива?
4. Как вставить элемент в массив?

Упражнения

1 Для задачи из примера 7.1 выполните перечисленные задания.

1. Заполните таблицу.

№	n	a	Преобразованный массив
1	3	−2 −3 −5	
2	5	1 2 3 4 5	
3	10	1 −3 −2 0 4 0 2 −4 0 2	

2. Добавьте в таблицу свои значения n и a.

3. Можно ли заменить команды из п. 3.1. командами из п. 3.2?

3.1.
```
if a[i] > 0 then
  a[i] := a[i] * 2;
if a[i] < 0 then
  a[i] := a[i] + 5;
```

3.2.
```
if a[i] < 0 then
  a[i] := a[i] + 5;
if a[i] > 0 then
  a[i] := a[i] * 2;
```

4. В каких случаях программа будет давать неверный результат?

2 Задан одномерный массив. Преобразуйте его элементы по следующему правилу: из всех положительных элементов вычесть элемент с номером k, ко всем отрицательным добавить введенное число x. Нулевые элементы оставьте без изменения.

3 Задан одномерный массив из четного количества элементов. Поменяйте местами его «половинки».

4 В массиве записаны фамилии и имена учащихся класса. Из класса выбыли два учащихся. Известны их номера. Исключите данные этих учащихся из массива.

5 Для задачи из примера 7.4 выполните перечисленные задания.

1. Заполните таблицу:

№	n	a	x	k	Преобразованный массив
1	3	−2 −3 −5	0	2	
2	5	1 2 3 4 5	0	1	
3	10	1 −3 −2 0 4 0 2 −4 0 2	10	11	

2. Добавьте в таблицу свои значения n, a, x, k.

3. Какой результат выдаст программа, если ввести n = 5, a k = 120? Вставьте в программу проверку для числа k ($1 < k \leqslant n + 1$).

4. Какой результат получим, если заменить цикл из п. 4.1 циклом из п. 4.2?

4.1.
```
for var i := n downto k do
  a[i + 1] := a[i];
```

4.2.
```
for var i := k to n do
  a[i + 1] := a[i];
```

6 Переставьте первый элемент массива на последнее место, второй — на первое, третий — на второе и т. д.

Глава 2
КОМПЬЮТЕР КАК УНИВЕРСАЛЬНОЕ УСТРОЙСТВО ОБРАБОТКИ ИНФОРМАЦИИ

§ 8. Аппаратные средства компьютера

8.1. Структурная схема компьютера

Прообразом первого компьютера принято считать аналитическую машину (пример 8.1), разработанную Чарльзом Бэббиджем в 1834 г. Бэббидж определил структурные элементы современного компьютера: память, устройство для обработки данных (названное им «мельница») и устройства для ввода и вывода данных.

По существу, аналитическая машина является моделью умственной деятельности человека. Работая с информацией, человек выполняет следующие функции:

- прием (т. е. ввод) информации;
- запоминание (т. е. хранение) информации;
- мышление (т. е. обработка) информации;
- передача (т. е. вывод) информации.

Компьютер является универсальным устройством для работы с данными, поэтому он должен уметь выполнять аналогичные функции: ввод, обработку, хранение и вывод данных.

Под **структурой компьютера** понимают модель, которая определяет состав, порядок и принципы взаимодействия элементов компьютера.

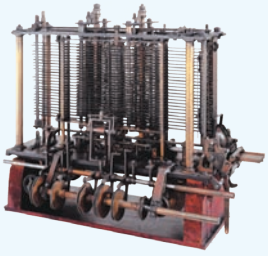
Архитектура компьютера — общее описание его структуры и функций.

Вместо того чтобы все помнить, человек начал делать пометки: сначала это были наскальные рисунки, а с появлением письменности — книги. Таким образом, данные стали хранить на внешних носителях информации.

Для передачи и получения информации использовались различные сигналы: зажженные костры, звуки гонга и др. В дальнейшем, с развитием науки, сигналы научились передавать с помощью радиоволн.

Обработка информации требовала вычислительных действий, что привело к появлению различных устройств для облегчения счета: от простейшего абака до современных компьютеров.

Пример 8.1. Аналитическая машина Бэббиджа.



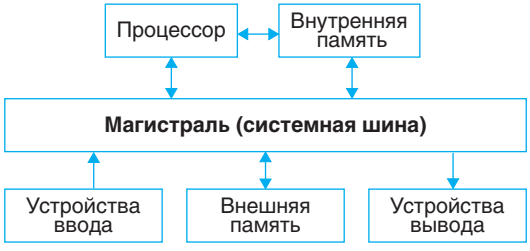
Архитектура компьютера не включает в себя подробных описаний электронных схем. Эти сведения нужны конструкторам, специалистам по наладке и ремонту компьютеров.

Современные компьютеры собраны в соответствии с принципом открытой архитектуры. Этот принцип позволяет собирать компьютеры, подбирая комплектующие в зависимости от заявленных критериев. Спецификации на создание устройств разрабатываются заинтересованными производителями совместно. Сборка или модернизация компьютера происходит из совместимых блоков, произведенных различными изготовителями.

Джон фон Нейман (1903—1957) — математик, сделавший важный вклад в информатику и другие науки. Наиболее известен как праотец современной архитектуры компьютеров (архитектура фон Неймана).

Архитектура фон Неймана предусматривает одно устройство, через которое проходит поток данных, и одно устройство управления, через которое проходит поток команд: SISD (Single Instruction Single Data) — «один поток команд, один поток данных».

В понятие «архитектура компьютера» входят: устройство компьютера, физические, арифметические и логические принципы работы его блоков, состав и функции программного обеспечения. В основу архитектуры современных компьютеров положен магистрально-модульный принцип. Структурная схема компьютера имеет следующий вид:



(Более подробная структура компьютера представлена в *Приложении к главе 2*, с. 114.)

В соответствии с магистрально-модульным принципом компьютер представляет собой набор блоков, взаимодействующих с общим каналом для обмена данными — системной шиной (магистралью). Каждый блок выполняет специализированные операции.

Общность архитектуры разных компьютеров обеспечивает их совместимость с точки зрения пользователя.

Основные принципы архитектуры компьютеров разработаны Д. фон Нейманом в 1945 г. Приведем их перечень:

1. Использование двоичного кода.
2. Программное управление.
3. Хранение данных программ в памяти и одинаковое кодирование их в двоичном коде.
4. Наличие у ячеек памяти компьютера последовательно пронумерованных адресов.

5. Возможность условного перехода при выполнении программы. Команды выполняются последовательно, но при необходимости можно реализовать переход к любой части кода.

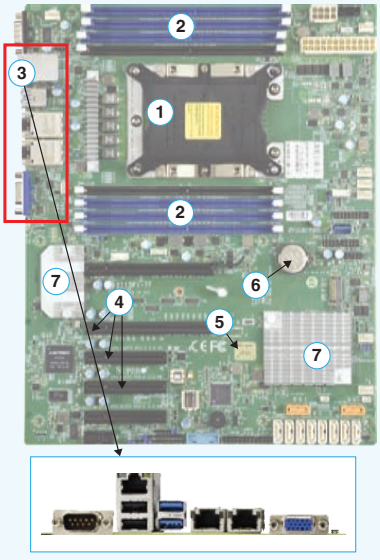
8.2. Системная плата, системная шина, процессор

Системная (материнская) плата является главной платой в системном блоке компьютера (пример 8.2). На ней располагаются основные компоненты компьютерной системы (процессор, оперативная память, системная шина и др.). Материнская плата обеспечивает связь важнейших компонентов персонального компьютера между собой.

На материнской плате имеются специальные разъемы для установки внутренних устройств компьютера. Для подключения каждого устройства к материнской плате разработаны различные разъемы. В примере 8.2 номером «1» отмечен **сокет** — разъем для размещения процессора. Номер «2» указывает на разъемы для установки оперативной памяти. Разъемы для подключения внешних устройств отмечены номером «3».

Номер «4» указывает на **слоты** — разъемы для вставки карт расширения. **Карта расширения** — специальная плата, которую устанавливают в слот расширения материнской платы с целью добавления компьютеру дополнительных функций. К платам расширения относятся: видеокарта, звуковая карта, сетевая карта и др. При такой конструкции замена одних

Пример 8.2. Материнская плата компьютера.



Номер «5» указывает на микросхему, хранящую BIOS — программное обеспечение для начальной загрузки компьютера. Номером «6» отмечена батарейка, необходимая для поддержания сохраненных настроек материнской платы. Также на материнской плате находятся чипсеты (номер «7») — микросхемы, которые позволяют процессору обмениваться информацией с памятью и периферийными устройствами.

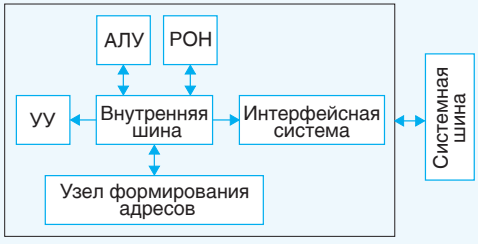
Современные системы включают два типа шин (архитектура DIB — Dual independent bus, двойная независимая шина):

- первичная шина (FSB, frontside bus), связывающая процессор с оперативной памятью и оперативную память с периферийными устройствами;
- вторичная шина (BSB, backside bus) для связи с кэш-памятью.

Использование двойной независимой шины повышает производительность процессора, поскольку в этом случае он может параллельно обращаться к различным уровням памяти.

Пример 8.3. Процессоры.

Пример 8.4. Структурная схема процессора (упрощенная).



Внутренняя шина микропроцессора осуществляет взаимосвязь между его составляющими. Узел формирования адреса — блок, отвечающий за формирование списка адресов для выбора следующих команд или данных.

Конструкция микропроцессора обеспечивает передачу адресов, данных, команд и управляющих сигналов. По системной шине в устройство управления вводится код команды. Затем сигнал дешифрируется и создается последовательность микрокоманд, которую исполняют блоки компьютера или процессор. При необходимости одновременно с этим формируется адрес для загрузки следующей команды или данных.

Частота работы всех современных процессоров в несколько раз превышает частоту системной шины, поэтому процессор работает так быстро, как ему это позволяет системная шина. Величину, на которую частота процессора превышает частоту системной шины, называют множителем.

внешних устройств на другие сопровождается простой заменой карты расширения.

Системная шина выполняет роль информационной магистрали, связывающей все устройства компьютера друг с другом. Упрощенно системную шину можно представить как группу проводников и электрических (токопроводящих) линий на системной плате. К системной шине подсоединены все основные блоки компьютера. Главной функцией системной шины является обеспечение взаимодействия между процессором и остальными компонентами компьютера. По системной шине осуществляется передача данных, адресов памяти и управляющих команд. Частота шины характеризует пропускную способность канала передачи данных.

Центральным устройством компьютера является **процессор** (пример 8.3). Он непосредственно выполняет операции по обработке данных (арифметические и логические) и управлению вычислительным процессом. Процессор осуществляет выборку машинных команд и данных из оперативной памяти, их выполнение и запись результатов обратно в оперативную память, управляет внешними устройствами.

Процессор (микропроцессор) представляет собой микросхему, которая содержит устройство управления (УУ), арифметико-логическое устройство (АЛУ) и регистры общего назначения (РОН). Структурная схема процессора показана в примере 8.4.

Устройство управления вырабатывает управляющие сигналы для выполнения заданной команды микропроцессором и компьютером в целом.

Арифметико-логическое устройство предназначено для выполнения арифметических и логических операций обработки данных.

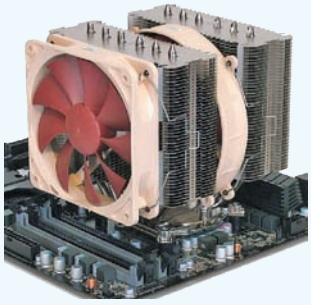
Регистры общего назначения — специальные ячейки сверхбыстрой памяти внутри процессора, к которым он может обращаться напрямую, используются при выполнении арифметических операций.

Процессоры являются энергоемкими устройствами и при работе сильно нагреваются, поэтому на них ставят специальные системы охлаждения (пример 8.5).

Основными характеристиками процессора являются:

- **тактовая частота** (показывает скорость работы процессора в герцах (ГГц), т. е. определяет количество рабочих операций в секунду);
- **разрядность** (максимальное количество двоичных разрядов, над которым одновременно может производиться операция передачи и обработки данных);
- **размер кэш-памяти процессора** (размер дополнительной высокоскоростной памяти, которая хранит копии наиболее часто используемых участков оперативной памяти);
- **количество вычислительных ядер** (каждое ядро представляет собой часть процессора, которая может обрабатывать отдельный поток данных).

Пример 8.5. Система охлаждения процессора.



Системы охлаждения также устанавливают на другие компоненты материнской платы: видеокарту, чипсет и др.

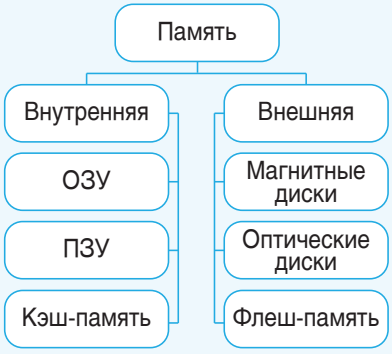
Первые многоядерные процессоры представляли собой самые простые схемы: два процессорных ядра, размещенных на одном кристалле без разделения каких-либо ресурсов, кроме шины памяти.

Сегодня существует два часто употребляемых термина для процессоров, имеющих несколько ядер: мультиядерный, или многоядерный (multi-core), и многопроцессорный (many-core). Многоядерный процессор содержит несколько вычислительных ядер на одной интегральной схеме (процессорном кристалле). Термином *многопроцессорный* обозначают компьютеры, имеющие несколько физически раздельных процессоров, управляемых одним экземпляром операционной системы (ОС).

Вычислительные ядра многоядерного процессора совместно используют кэш третьего или второго уровня.

В августе 2019 г. компания Cerebras представила самый большой в мире многоядерный суперпроцессор Cerebras Wafer Scale Engine. Он имеет более 1,2 трлн транзисторов и 400 000 ядер и занимает почти всю площадь полупроводниковой пластины диаметром 300 мм.

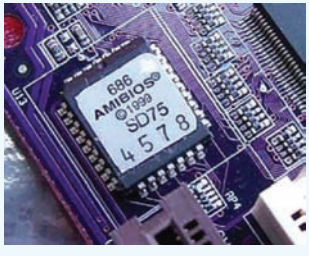
Пример 8.6. Виды компьютерной памяти.



Пример 8.7. Оперативная память.

Оперативная память является быстродействующей и позволяет обращаться к каждой ячейке памяти отдельно (прямой доступ к ячейке по адресу).

Пример 8.8. Постоянная память.



Важнейшая микросхема ПЗУ — модуль BIOS (от англ. basic input/output system — базовая система ввода/вывода).

На материнской плате также установлена CMOS (полупостоянная память) — память для хранения параметров конфигурации компьютера и текущего времени.

8.3. Виды и назначение памяти

Компьютерная память служит для хранения данных и бывает нескольких типов. Каждый тип памяти предназначен для выполнения различных задач.

Различают внутреннюю и внешнюю память (пример 8.6). К **внутренней памяти** относят:

- оперативную память (оперативное запоминающее устройство — ОЗУ, англ. *random access memory — RAM*);
- постоянную память (постоянное запоминающее устройство — ПЗУ, англ. *read only memory — ROM*);
- кэш-память.

Внешнюю память разделяют по физическим принципам записи данных: магнитные носители, оптические носители, флеш-память.

Оперативная память служит для хранения программ и данных, с которыми процессор работает в текущий момент (пример 8.7). Современные типы оперативной памяти не могут хранить данные после выключения питания компьютера — память энергозависимая. Объем оперативной памяти современных компьютеров составляет 4—64 Гбайт.

Постоянная память хранит программы автоматического тестирования устройств и загрузки ОС в оперативную память (пример 8.8). ПЗУ является энергонезависимой памятью, поскольку сохраняет информацию после отключения питания компьютера. В большинство микросхем ПЗУ невозможно внести изменения. Имеет небольшой объем — от 384Кб до 8 Мб.

Кэш-память – быстродействующая память, которая позволяет увеличить

скорость выполнения операций. Служит буфером между оперативной памятью и микропроцессором.

Внешняя память предназначена для длительного хранения информации, является энергонезависимой, имеет большие размеры (до нескольких Терабайт).

К магнитным носителям относится **винчестер** (накопитель на жестких магнитных дисках — НЖМД, англ. *hard disk drive — HDD*). Он представляет собой совокупность из нескольких дисков (пластин) с нанесенными магнитными слоями (пример 8.9). Диски располагаются на одной оси электродвигателя и находятся в специальном металлическом корпусе.

Большое распространение получили внешние винчестеры, использующие для подключения к компьютеру разъем USB. Многие из них объединяют традиционный жесткий диск с модулем флеш-памяти, что позволяет увеличить скорость его работы.

Данные на **оптические носители** записываются с помощью лазера. Наиболее известные типы оптических дисков — CD, DVD и Blu-ray (пример 8.10).

Флеш-память — полупроводниковая память, построенная на основе интегральных микросхем (пример 8.11). Флеш-память компактна и долговечна, имеет высокое быстродействие. Ее используют в цифровых фото- и видеокамерах, мобильных телефонах и т. д.

Во многих современных компьютерах устанавливают твердотельные накопители SSD (Solid State Drive), которые зарекомендовали себя как более надежные и быстрые альтернативы

Кэш-память делится на три уровня: L1, L2, L3. Каждый из уровней отличается по размеру памяти, скорости, выполняемым задачам. L1 (устанавливается на процессоре) — самый маленький (до 128 Кбайт) и быстрый, L2 (может размещаться на том же кристалле, что и процессор, или быть отдельной микросхемой) — средний (от 256 Кбайт до 12 Мбайт), L3 — самый большой (0—16 Мбайт) и медленный, устанавливается на серверах. К каждому уровню процессор обращается поочередно (от меньшего к большему), пока не обнаружит в одном из них нужные данные. Если ничего не найдено, то процессор обращается к оперативной памяти.

Пример 8.9. Винчестер:

Пример 8.10. Оптические диски:

Пример 8.11. Флеш-память:

Технологию флеш-памяти используют следующие виды устройств:

- compactFlash (применяется в цифровых фотоаппаратах);
- microSD/miniSD (флеш-карта, используемая в мобильных телефонах);
- внешние накопители (флешки, подключаются к компьютерной технике с помощью USB-разъема).

Пример 8.12. Твердотельный SSD.

Пример 8.13. Схема передачи данных от внешних устройств к процессору имеет сдедующий вид:

HDD. Внутреннее устройство SSD представляет из себя набор микросхем флеш-памяти, размещенных на одной плате (пример 8.12).

Внешняя память предназначена для длительного хранения программ и данных, и целостность ее содержимого не зависит от того, включен или выключен компьютер. В отличие от оперативной памяти она не имеет прямой связи с процессором. Данные от внешних устройств (ВУ) к процессору и обратно передаются через оперативную память (пример 8.13).

1. Что понимают под структурой и архитектурой компьютера?
2. Какие принципы архитектуры компьютера сформулировал Д. фон Нейман?
3. Какие устройства входят в состав процессора?
4. Каково назначение системной шины?
5. На какие виды делится компьютерная память?

Упражнения

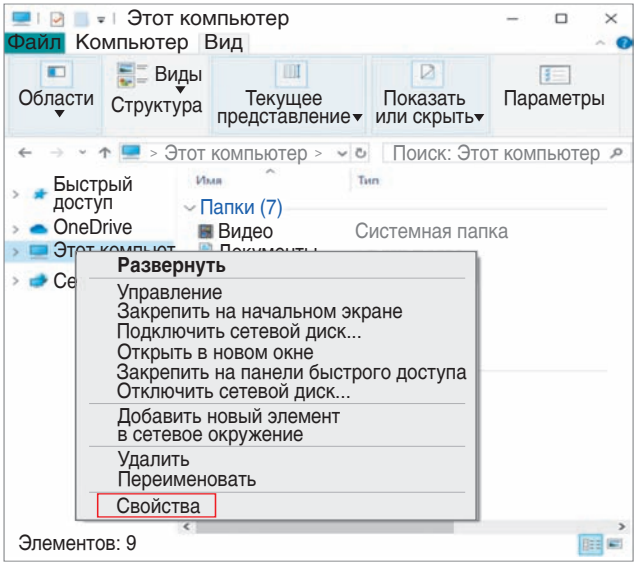
1 Подготовьте в режиме совместного доступа презентацию на одну из тем.

1. История носителей информации.
2. Виды компьютерной памяти.
3. Поколения ЭВМ.

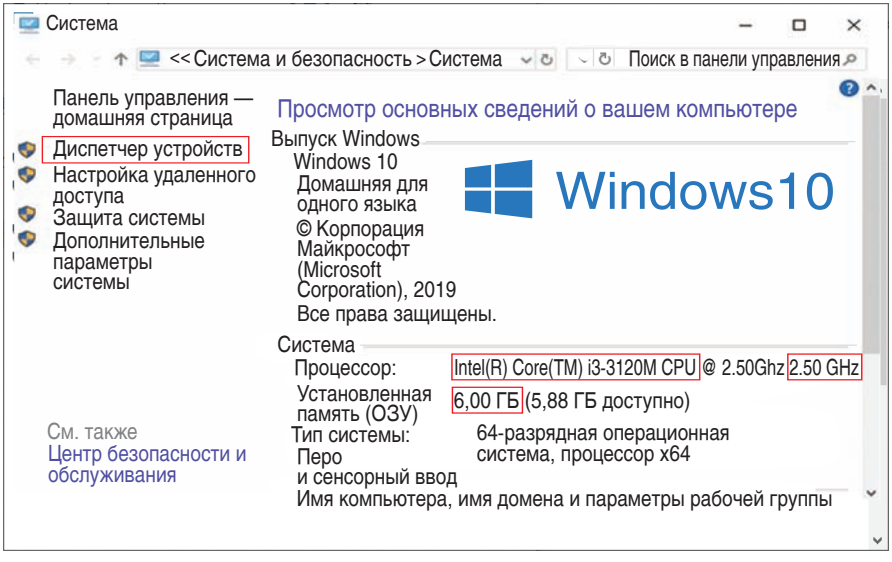
2 Заполните таблицу. (Работать с таблицей рекомендуется, используя облачные технологии.)

	Школьный компьютер	Домашний компьютер
Процессор		
Частота процессора		
Объем оперативной памяти		
Емкость диска C:		
Свободно на диске C:		
Другие устройства внешней памяти		

Данные для таблицы можно получить, открыв свойства компьютера в программе **Проводник**:

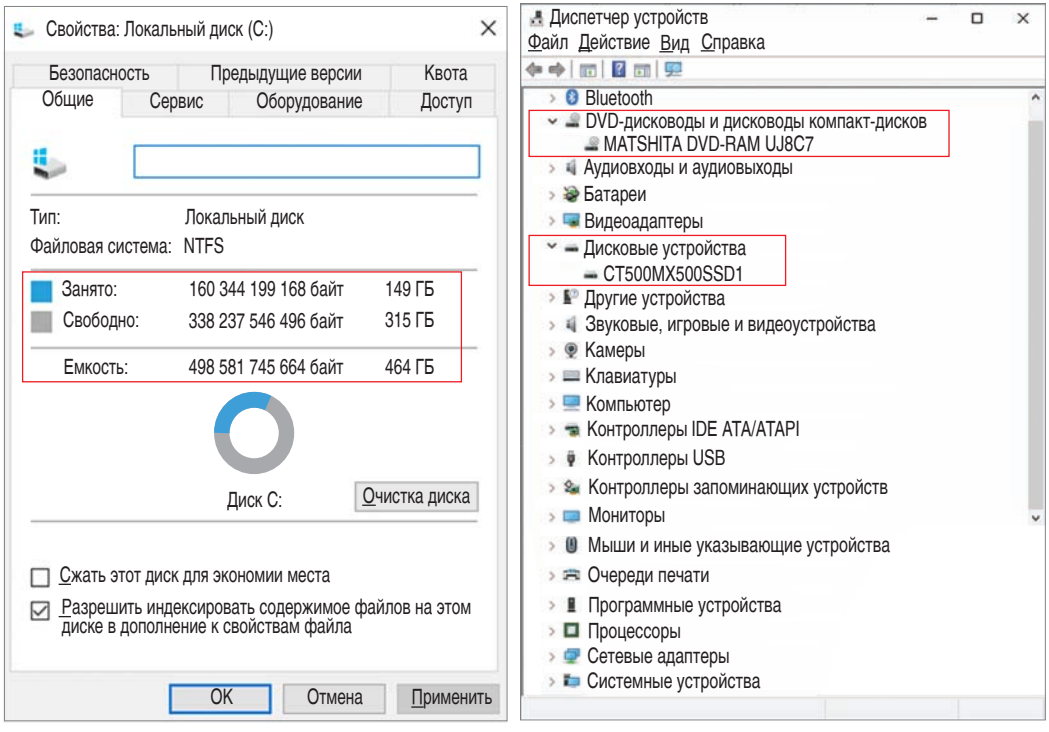


Определите тип и характеристики процессора, а также размер оперативной памяти, установленной на школьном и вашем домашнем (при его наличии) компьютерах.



Определите объем диска C: и количество свободной памяти на нем, используя свойства диска (команда **Свойства** в контекстном меню диска C:).

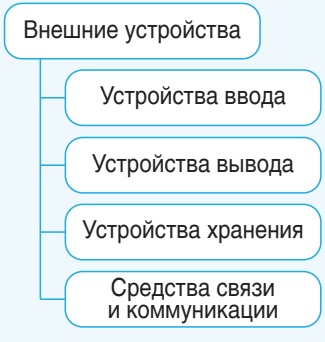
*Откройте **Диспетчер устройств** и определите, какие еще устройства внешней памяти подключены к компьютеру.



3* Откройте сайт (по указанию учителя) с виртуальным тренажером по сборке ПК и выполните задания.

§ 9. Внешние устройства

Пример 9.1. Внешние устройства:



9.1. Классификация внешних устройств

Внешние (периферийные) устройства обеспечивают взаимодействие компьютера с пользователем, другими компьютерами или техническими устройствами. Они подключаются к компьютеру через специальные разъемы — порты ввода-вывода. Большинство современных внешних устройств подключается к порту USB (Universal

Serial Bus — универсальная последовательная шина).

Внешние устройства по своему назначению можно разделить на: устройства ввода, устройства вывода, устройства хранения информации, а также средства связи и коммуникации (пример 9.1).

К **устройствам ввода** информации относятся:

- клавиатура;
- устройства указания, или графические манипуляторы: джойстик (приспособление в виде рычага — рукоятки, штурвала, — позволяющее управлять виртуальным объектом в двух- или трехмерном пространстве); световое перо (манипулятор, который позволяет вводить информацию путем прикосновения утройства к экрану); мышь; трекбол («мышь наоборот»: для работы необходимо вращать шар, закрепленный в неподвижном корпусе);
- графический планшет, или дигитайзер (состоит из пера и планшета, чувствительного к нажатию или близости пера; предназначен для ввода в компьютер информации, созданной «от руки»);
- сканер;
- микрофон.

(Рассмотрите пример 9.2.)

К **устройствам вывода** информации относятся:

- монитор;
- принтер (лазерные и струйные принтеры позволяют выводить информацию на бумагу, 3D-принтеры позволяют создавать объемные объекты);

Пример 9.2. Устройства ввода:

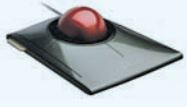
Клавиатура

Устройства указания:

Джойстик

Световое перо

Мышь



Трекбол

Графический планшет

Сканер

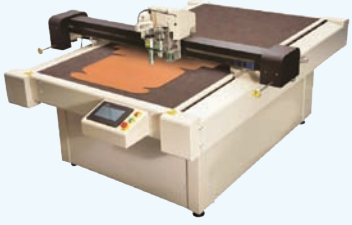
Микрофон

Пример 9.3. Устройства вывода:

Монитор

Принтер

3D-принтер

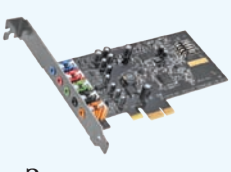


Плоттер

Колонки

Наушники

Пример 9.4. Платы расширения.



Звуковая карта

Видеокарта

- плоттер (устройство для автоматического вычерчивания рисунков, схем, чертежей, карт; режущий плоттер позволяет осуществлять вырезку лекал из картона, кожи, пластика и др.);
- колонки, наушники.

(Рассмотрите пример 9.3.)

Многие из устройств комбинируют в себе устройства ввода и вывода: многофункциональное устройство (МФУ) содержит в себе сканер и принтер; гарнитура объединяет микрофон и наушники. Сенсорные мониторы не только отображают информацию, но и позволяют ее вводить.

Устройства хранения — устройства внешней памяти, которые были рассмотрены в предыдущем параграфе.

Для работы многих устройств необходимы карты (платы) расширения (пример 9.4), которые содержат адаптеры. **Адаптер** необходим для преобразования сигнала, поступающего от устройства, в двоичный код и обратно. Некоторые адаптеры встроены непосредственно на материнскую плату.

К **средствам связи и коммуникации** относят устройства, которые позволяют организовать передачу данных по компьютерной сети:

- сетевой адаптер, или сетевая карта (предназначается для соединения компьютеров в локальную сеть по технологии Ethernet — проводное соединение);
- модем (устройство передачи данных между компьютерами по телефонной и другим линиям связи);
- маршрутизатор, или роутер (устройство, необходимое для перенаправления пакетов данных в одной

или нескольких подсетях; маршрутизаторы позволят обеспечить как проводной, так и беспроводной доступ в Интернет).

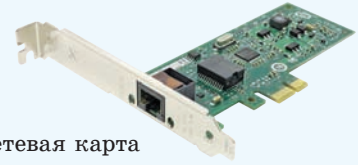
(Рассмотрите пример 9.5.)

9.2. Аппаратное обеспечение для подключения к сети Интернет

Количество пользователей Интернета в современном мире стремительно растет. Сегодня практически каждый человек может подключиться к Интернету.

Интернет — сложная система компьютерных сетей. Передача данных в компьютерных сетях требует согласованной работы большого количества разнообразных устройств. Для слаженного взаимодействия работы сетевых устройств Международной организацией стандартов (International Standard Organization — ISO) была разработана **сетевая модель OSI** (Open System Interconnection). Эта модель описывает правила и способы передачи данных в различных сетевых средах при организации сеанса связи. Основными элементами модели являются уровни, прикладные процессы и физические средства соединения (пример 9.6). Модель включает в себя семь уровней. Каждому из них отводится конкретная роль, а общая задача передачи данных разбивается на отдельные подзадачи. Основным с точки зрения пользователя является прикладной уровень. Этот уровень обеспечивает выполнение прикладных задач пользователей. На нем реализуются такие сервисы, как удаленная передача данных, электронная почта и работа веб-браузеров.

Пример 9.5. Устройства коммуникации.



Сетевая карта

4G-модем для подключения по каналам сотовой связи

Маршрутизатор со встроенным модемом

Современный мобильный телефон может выступать в качестве роутера и обеспечить доступ в Интернет для других устройств.

Пример 9.6. Уровни сетевой модели OSI.

Уровень протокола	Единица измерения данных (pdu — protocol data units)
Физический	Биты
Канальный	Фреймы
Сетевой	Пакеты
Транспортный	Блоки
Сеансовый	Данные
Представительский	Данные
Прикладной	Данные

Уровни сетевой модели OSI

1. **Физический уровень** связан с работой аппаратных средств и определяет физические аспекты передачи информации по линиям связи (уровень напряжения, частоты, природу передающей среды, способ передачи двоичных данных по физическому носителю). Для поддержки физического уровня в компьютеры устанавливают сетевой адаптер.

2. **Канальный уровень** отвечает за передачу данных по физическому уровню с проверкой возможности передачи данных, реализует механизм обнаружения и коррекции ошибок между узлами сети. За протоколы канального уровня отвечают сетевые адаптеры и их драйверы.

3. **Сетевой уровень** отвечает за доставку информации от узла-отправителя к узлу-получателю, обеспечивает маршрутизацию пакетов данных.

4. **Транспортный уровень** обеспечивает доставку информации вышележащим уровням с необходимой степенью надежности. Может обнаруживать и исправлять ошибки передачи, такие как искажение, потеря или доставка пакетов в неправильном порядке.

5. **Сеансовый уровень** выполняет задачу организации сеансов: установление, поддержание и завершение соединения между приложениями участников соединения.

6. **Представительский уровень** отвечает за форму представления данных, их шифровку/дешифровку, обеспечивает сжатие/распаковку и перекодировку данных из одного формата в другой.

7. **Прикладной уровень** обеспечивает взаимодействие пользовательских приложений с сетью.

Сегодня существуют разнообразные способы подключения к Интернету. Основные отличия: принцип работы, скорость передачи данных, надежность, сложность настройки оборудования, цена.

По количеству трафика использование Интернета можно разделить на две группы: просмотр страниц (малое количество трафика) и скачивание больших файлов — фильмов, музыки и т. д. (требует большого количества трафика). В первом случае достаточно скорости обычного модемного соединения. Однако такая скорость затрудняет скачивание файлов, поэтому для полноценного использования возможностей Интернета требуется высокоскоростной доступ.

Существует два вида технологий выхода в Интернет:

1) проводная.

2) беспроводная.

От технологии подключения к Интернету зависит тип используемого модема (пример 9.7).

Передача данных при проводной технологии осуществляется по специальному кабелю (оптоволокно или витая пара), который с одной стороны подключен к оборудованию провайдера, а с другой — в сетевую карту компьютера.

В зависимости от типа оборудования провайдера используются следующие способы проводного подключения к Интернету:

1. Модемные соединения (ADSL). Данная технология превращает аналоговые сигналы, передаваемые по стандартной телефонной линии, в

цифровые сигналы (пакеты данных). При этом во время работы можно совершать звонки.

2. Соединение по выделенной линии. При этом пользователь получает постоянный выход в Интернет через отдельную свободную телефонную линию, которая гарантирует высокое качество соединения и передачу данных на высокой скорости.

3. Подключение через телевизионный кабель. Этот способ возможен только в случае наличия кабельного телевидения.

Беспроводные технологии служат для передачи данных между двумя и более точками на расстоянии, не требуя проводной связи. Для передачи информации могут использоваться радиоволны, а также инфракрасное, оптическое или лазерное излучение. Наиболее существенными характеристиками беспроводных технологий передачи данных являются максимальная скорость передачи и максимальное расстояние, на которое можно передавать информацию (пример 9.8).

Беспроводные маршрутизаторы позволяют использовать Интернет, находясь в любом месте в пределах зоны доступа.

Еще недавно подключение к Интернету через спутниковую связь было практически недоступно для обычных пользователей из-за высокой цены.

Современная спутниковая связь по скорости, надежности и безопасности не уступает традиционной проводной. Существенным ее преимуществом является возможность применения в отдаленных и труднодоступных местах,

Пример 9.7. Классификация модемов по виду соединения:

- для цифровых коммутируемых телефонных линий;
- для организации выделенной линии в телефонной сети;
- кабельные;
- радиомодемы;
- спутниковые.

Пример 9.8. Классификация беспроводных сетей по дальности действия:

- Беспроводные персональные сети (Wireless Personal Area Networks — WPAN) основаны на технологии Bluetooth, которая позволяет устройствам сообщаться со скоростью до 24 Мбит/с, если они находятся в радиусе до 10 м друг от друга.
- Беспроводные локальные сети (Wireless Local Area Networks — WLAN) применяют технологию Wi-Fi. Технология быстро развивается. В конце 2018 г. был представлен стандарт поколения Wi-Fi 6, позволяющий передавать данные со скоростью до 11 Гбит/с в радиусе 300 м.
- Беспроводные сети масштаба города (Wireless Metropolitan Area Networks — WMAN) используют технологию WiMAX с охватом территории до 150 км и скоростью передачи данных до 1 Гбит/с.
- Беспроводные глобальные сети (Wireless Wide Area Network — WWAN) основываются на различных технологиях, таких как GPRS, EDGE, HSPA, LTE и др., которые являются надстройками над технологией мобильной связи. Они позволяют пользователю сети сотовой связи производить обмен данными с другими устройствами в сети, а также с внешними сетями, в том числе Интернет. В качестве модема для подключения к Интернету может использоваться мобильный телефон. Компьютер может подключаться к телефону посредством USB-кабеля или беспроводным способом.

Пример 9.9. Схема подключения к Интернету с использованием спутниковой связи:

Двоичный код используется для кодирования данных еще и потому, что устройствам гораздо проще и быстрее выполнять арифметико-логические операции в двоичной системе счисления, чем в десятичной.

В 1959 г. ученые из Московского государственного университета под руководством Николая Брусенцева разработали первую и единственную ЭВМ на основе троичной логики. Называлась она «Сетунь». Других компьютеров на основе троичного кода нет и не было.

Идею использовать для вычислений троичную систему высказал еще в XIII в. итальянский математик Фибоначчи. Он сформулировал и решил задачу о гирях: если можно класть гири только на одну чашу весов, то удобнее, быстрее и экономичнее делать подсчеты в двоичной системе, а если можно класть гири на обе чаши, то целесообразнее прибегнуть к троичной системе.

где невозможно или слишком дорого прокладывать интернет-кабель. Спутниковой связью широко пользуются государственные службы, геологоразведочные и нефтяные компании, телекоммуникационные фирмы и другие крупные организации. В ряде случаев она является единственным вариантом интернет-коммуникаций (пример 9.9).

9.3. Принципы работы аппаратных средств компьютера

Современный компьютер использует электрические сигналы, т. е. ток или напряжение, значения которых меняются по закону, отображающему передаваемое сообщение. С помощью сигналов кодируются передаваемые и обрабатываемые данные. Электрические сигналы можно использовать для кодирования как двоичный код: «1» — есть ток (ток больше пороговой величины); «0» — нет тока (ток меньше пороговой величины).

Чем меньше значений существует в системе, тем проще изготовить отдельные конструктивные элементы, оперирующие этими значениями. Наиболее надежным и дешевым является устройство, каждый разряд которого может принимать два состояния: есть ток/нет тока, высокое напряжение/низкое напряжение, намагничено/не намагничено и т. д.

1. Какие устройства компьютера относят к внешним?
2. На какие группы можно разделить внешние устройства в зависимости от их назначения?
3. Какие существуют способы подключения к Интернету?
4. Какое оборудование может использоваться для проводного подключения к Интернету?
5. Какие технологии используются для беспроводного подключения к Интернету?

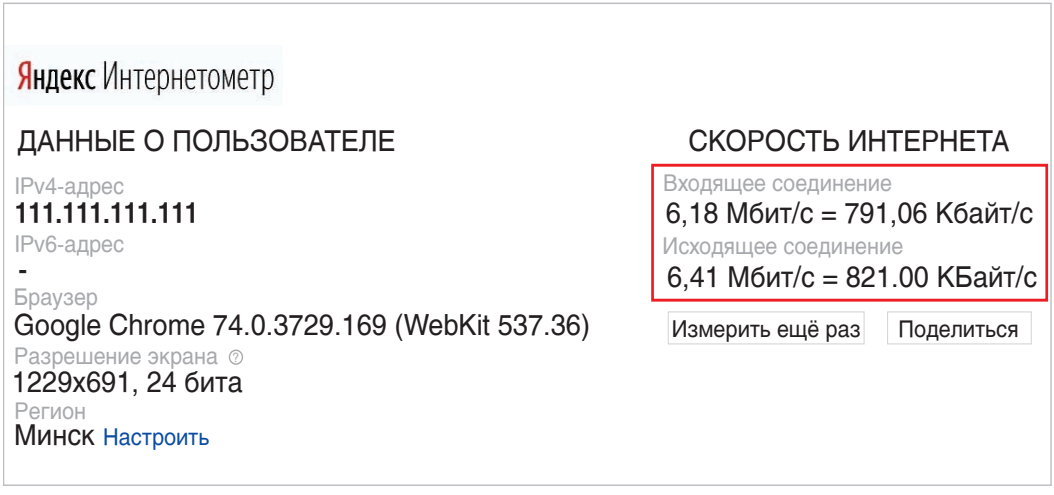
Упражнения

1 Подготовьте презентацию на одну из перечисленных тем в режиме совместного доступа.

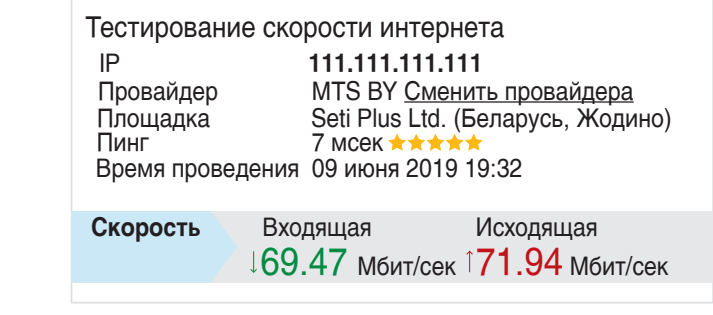
1. Принтеры.
2. Способы подключения к Интернету.
3. Преимущества и недостатки беспроводного подключения к Интернету.
4. Интернет через спутник.
5. Принцип работы ЭВМ «Сетунь».

2 Определите скорость подключения вашего устройства к Интернету, используя возможности перечисленных сайтов (или других аналогичных).

1. https://yandex.by/internet/



2. https://2ip.ru/speed/



§ 10. Программное обеспечение компьютера

Впервые идея о раздельном рассмотрении команд и данных была высказана Чарльзом Бэббиджем в XIX в. Позже, в XX в., она была развита в принципах Джона фон Неймана. Эти принципы учитываются и при разработке архитектур современных компьютеров, и при разработке компьютерных программ.

Совместное использование шины для памяти программ и памяти данных приводит к «узкому месту архитектуры фон Неймана». Из-за того что память программ и память данных не могут быть доступны в одно и то же время, пропускная способность канала «процессор-память» существенно ограничивает скорость работы компьютера.

Ученые из США и Италии в 2015 г. заявили о создании прототипа мем-процессора (mem — от англ. *memory*) с архитектурой, отличной от архитектуры фон Неймана. Мем-процессор реализует одновременное вычисление и хранение полученных данных в одном месте путем взаимодействия ячеек памяти.

10.1. Программный принцип работы компьютера

Основным принципом построения всех современных компьютеров является программное управление, в соответствии с которым команды программы и данные хранятся в оперативной памяти в закодированном виде. Информация, с которой работает компьютер, представлена в двоичном коде и делится на два типа: программа (набор команд по обработке данных); данные, обрабатываемые программой. Процессор может выполнять арифметические и логические операции, предусмотренные его системой команд. Команды и данные считываются по очереди из памяти и поступают в процессор, где они расшифровываются, а затем выполняются. Результаты выполнения различных команд могут быть записаны в память или переданы на различные устройства.

Работа компьютера по принципу программного управления описана в примере 10.1. Создание ЭВМ с хранимой в памяти программой положило начало программированию, а возможность обращения к любой ячейке памяти по ее адресу позволила использовать переменные в программировании.

Пример 10.1. Работа компьютера управляется программой, которая состоит из набора команд. Команды записываются в память компьютера и выполняются последовательно, одна за другой. Последовательность нарушается только в том случае, если выполняется команда условного или безусловного перехода. В команде перехода непосредственно указывается адрес следующей команды. Процесс вычислений продолжается до тех пор, пока не будет выполнена команда, предписывающая окончание вычислений.

10.2. Различные подходы к классификации программного обеспечения

Компьютер представляет собой единство аппаратных (hardware) и программных (software) средств. По-

явление персонального компьютера и развитие программирования привело к возникновению огромного количества различных программ. Совокупность всех программных средств называют **программным обеспечением (ПО)** компьютера.

Рассмотрим некоторые способы классификации ПО.

Классификация по назначению

В зависимости от назначения выделяют системное, прикладное и инструментальное ПО (пример 10.2). Каждый класс в свою очередь делится на подклассы. Подробную схему деления ПО можно посмотреть в *Приложении к главе 2* (с. 116).

Классификация по способу распространения и использования

Тип распространения и использования программы зависит от лицензии. Лицензия на программное обеспечение — правовой инструмент, определяющий использование и распространение программного обеспечения, защищенного авторским правом. Лицензия выступает гарантией того, что издатель ПО, которому принадлежат исключительные права на программу, не подаст в суд на пользователя. Обычно лицензия на программное обеспечение разрешает получателю использовать одну или несколько копий программы, причем без лицензии такое использование рассматривается как нарушение авторских прав издателя.

Способы распространения программных продуктов: коммерческий, условно-бесплатный, бесплатный и

Пример 10.2. Классы ПО в зависимости от назначения.

Системное ПО — совокупность программ для обеспечения работы компьютера и компьютерных сетей. Программы, входящие в состав системного ПО, позволяют пользователю осуществлять руководство и контроль над работой компьютера и компьютерной сети, а также обеспечивают возможность выполнения других программ.

Прикладное ПО — комплекс программ для решения задач определенного класса предметной области. Данный класс ПО является самым многочисленным, сюда входят редакторы, электронные системы обучения, компьютерные игры и т. д.

Инструментальное ПО предназначено для создания другого программного обеспечения. Сюда относят системы программирования, которые обеспечивают разработку программ.

Классификация по способу выполнения программы

В большей мере необходима программисту, чем обычному пользователю. По этому критерию программы делятся на компилируемые и интерпретируемые.

Исходный код у компилируемых программ преобразуется компилятором в машинный код и записывается в файл с особым заголовком и/или расширением. Операционная система идентифицирует такой файл как исполняемый.

У интерпретируемых исходный код программы последовательно исполняется с помощью специальной программы-интерпретатора.

Пример 10.3. Классы ПО в зависимости от способа распространения и использования.

Коммерческие программы (Commercial software) создаются с целью получения прибыли от их использования, например путем продажи.

Условно-бесплатные программы (shareware) распространяются по принципу «попробуй, прежде чем купить». Использовать программу можно в течение небольшого срока (2 недели или месяц). По истечении указанного срока пользователь обязан купить ее или прекратить использование программы и удалить ее.

Бесплатные программы (Freeware) — программное обеспечение, лицензионное соглашение которого не требует каких-либо выплат правообладателю. Лицензия не дает пользователю право на модификацию программы.

Пробные программы (Betaware) — обычно предварительные (тестовые) бета-версии коммерческого или некоммерческого ПО. Можно использовать бесплатно, но часто применение ограничивается периодом тестирования или функциональностью программы.

пробный (пример 10.3). Кроме того, различают свободное и проприетарное ПО. Свободное ПО распространяется с исходными кодами и может быть изменено пользователем. У проприетарного ПО все права (использование, распространение, модификация) принадлежат создателю.

Классификация по степени переносимости

Позволяет выделить кроссплатформенные и платформозависимые программы. Кроссплатформенные программы работают более чем на одной аппаратной платформе и/или операционной системе. Типичным примером является программное обеспечение, предназначенное для работы в операционных системах Linux и Windows одновременно. Платформозависимые программы работают только в той среде, для которой созданы.

1. В чем суть принципа программного управления?
2. По каким критериям можно классифицировать программное обеспечение?
3. Назовите основные классы ПО по назначению.

Упражнения

1 Определите, к какому классу программного обеспечения (по назначению) относятся перечисленные программы.

1. Архиватор.
2. Текстовый редактор.
3. PascalABC.
4. Windows.
5. Браузер.
6. Бухгалтерская программа.

2 Откройте сайт http://pascalabc.net/. Перейдите в раздел лицензионное соглашение. К какому классу относится лицензия PascalABC?

3 Найдите информацию о лицензии программы Inkscape.

4 Найдите информацию о кроссплатформенных программах: текстовый редактор, графический редактор, редактор электронных таблиц.

§ 11. Представление данных

11.1. Информация и данные

Из курса физики вам известно, что физические объекты в нашем мире находятся в состоянии непрерывного движения и взаимодействия, которое сопровождается появлением сигналов. Взаимодействие сигналов с физическими телами может изменять свойства тел. Изменения, которые можно измерять или регистрировать, образуют данные. **Данные** — зарегистрированные сигналы.

Данные несут в себе информацию о событиях, произошедших в материальном мире, поскольку они отражают зарегистрированные сигналы, возникшие в результате этих событий. Однако данные не тождественны информации (примеры 11.1—11.3). Для человека информация — содержание получаемых им сообщений. При получении информации уменьшается неопределенность знания. Знания определяют поведение человека, позволяют ему принимать решения, строить отношения с другими людьми.

Любая информация нематериальна, она не имеет формы, размеров, массы. Следовательно, для существования и распространения в нашем материальном мире она должна быть обязательно связана с каким-либо материальным объектом — **носителем информации**.

Материальным носителем информации может быть бумага, металл, пластмасса, воздух, электромагнитное поле и др. Сигналы также являются материальными носителями информации.

Пример 11.1. Открыв книгу с текстом на иностранном языке, человек получит данные, но не получит информацию, поскольку ему не известен способ преобразования данных, записанных с помощью неизвестных символов в известные ему понятия.

Пример 11.2. У вас есть файл с данными, но вы не знаете, в какой программе он был создан. В этом случае вы имеете данные, но не сможете извлечь информацию до тех пор, пока не установите соответствующую программу.

Пример 11.3. За тысячелетия эволюции запах не поддавался известным способам фиксации и передачи информации. Понять или представить незнакомый запах очень трудно. Однако человек получает информацию, почувствовав запах. Работы по получению данных о запахе ведутся, но о конечном результате пока говорить рано. На сегодняшний день запах — информация, но не данные.

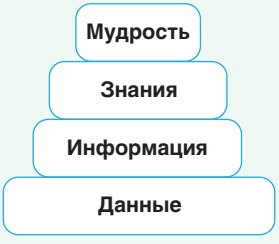
В 2013 г. ученые из Токийского аграрно-технического университета изобрели «пахнущий экран». Запах исходит из области на экране, соответствующей источнику аромата. Например, когда появляется изображение персика, соответствующий угол экрана пахнет фруктом.

На данный момент система единовременно может производить только один запах.

Пример 11.4. Характеристикой носителя, не изменяющейся с течением времени, может быть, например, намагниченность области поверхности диска или буква на бумаге. Характеристика носителя, которая изменяется с течением времени, — это, например, амплитуда колебаний звуковой волны или напряжение в проводах.

Пример 11.5. Примеры сигналов: электромагнитные волны, изменение электрического напряжения, звуковая волна, колебания земной коры, передача данных по каналу связи и др.

В 1989 г. американский ученый в области исследования операции и теории систем Рассел Акофф (1919—2009) предложил иерархическую модель DIKW (англ. *data, information, knowledge, wisdom* — данные, информация, знания, мудрость).



Каждый уровень добавляет определенные свойства к предыдущему:

- в основании находится уровень данных — знаки и сигналы;
- информация добавляет контекст — данные представляются в виде фактов, идей теорий;
- знания добавляют механизм использования информации, определяют, как человек будет ее применять;
- мудрость добавляет условия использования знаний, направленные на достижение поставленных целей.

Хранение информации связано с фиксацией состояния носителя, а распространение — с процессом, который протекает в носителе (пример 11.4).

Информация не существует сама по себе. Всегда имеется **источник**, который передает информацию, и **приемник**, который ее воспринимает. В роли источника или приемника может быть любой объект материального мира: человек, устройство, животное, растение. То есть информация всегда предназначена конкретному объекту.

Информация становится данными тогда, когда находится способ зафиксировать информацию на материальном носителе с помощью какого-либо формального языка.

Данные превращаются в информацию только тогда, когда ими заинтересуется человек. Человек извлекает информацию из данных, оценивает, анализирует ее.

Действия, выполняемые с информацией, называют **информационными процессами**. К ним относят процессы получения, создания, сбора, поиска, обработки, накопления, хранения, распространения и использования информации.

В информатике понятие «информация» часто отождествляется с понятием «данные», поскольку основным инструментом для изучения и осуществления информационных процессов являются компьютерные технологии. В качестве формального языка для представления данных в информатике является двоичный код. С помощью двоичного кода сегодня можно представлять числа, тексты, изображения, звук, видео.

11.2. Аналоговое и цифровое представление данных

Сигналы несут в себе информацию, представленную в виде данных. Получая значения сигнала, человек получает данные, из которых путем обработки извлекается информация. Большинство сигналов представляют собой физические величины, изменяющиеся во времени (пример 11.5).

Сигнал может быть представлен в **аналоговой** (непрерывной) или **дискретной**[1] форме.

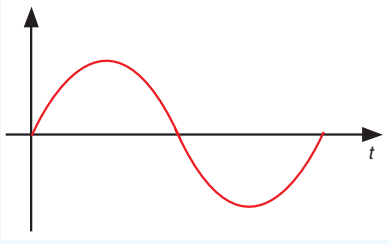
Аналоговый сигнал описывается функцией времени и непрерывным множеством возможных значений. **Дискретный сигнал** является прерывистым (примеры 11.6 и 11.7).

Благодаря своим органам чувств человек привык иметь дело с аналоговой информацией. Наши зрение и слух, а также все остальные органы чувств воспринимают поступающую информацию в аналоговой форме, т. е. непрерывно во времени.

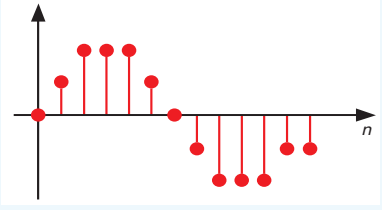
В компьютере информация представлена в цифровом виде. **Цифровой сигнал** — сигнал, который можно представить в виде последовательности числовых значений, записанных с помощью цифр. В настоящее время наиболее распространены двоичные цифровые сигналы. Это связано с их использованием в компьютерных устройствах и простотой кодирования.

Для получения цифрового представления какого-либо объекта его подвергают дискретизации: получают набор числовых значений, которые

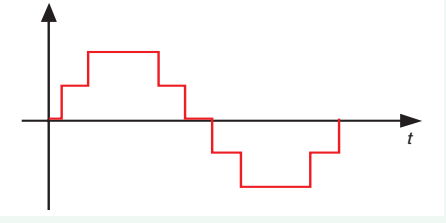
Пример 11.6. Аналоговый сигнал.



Пример 11.7. Дискретный сигнал.



Чтобы представить аналоговый сигнал последовательностью чисел, его следует сначала превратить в дискретный сигнал, а затем подвергнуть квантованию (сигнал, значения которого дискретны, а время непрерывно).



В результате сигнал будет представлен так, что на каждом промежутке времени окажется известно приближенное (квантованное) значение сигнала, которое можно записать числом. Если записать эти целые числа в двоичной системе, получится последовательность нулей и единиц, которая и будет являться цифровым сигналом.

[1] Дискретность — свойство, противопоставляемое непрерывности; прерывность.

Исходной величиной АЦП может быть любая физическая величина — напряжение, ток, сопротивление, емкость, частота следования импульсов, угол поворота вала и др.

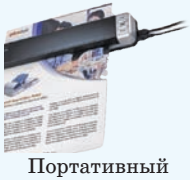
Частота дискретизации (или частота семплирования, англ. *sample rate*) — частота взятия отсчетов непрерывного по времени сигнала при его дискретизации (определяет, сколько раз в секунду будет измерен исходный сигнал). Измеряется в герцах.

Пример 11.8. Сканеры.

Планшетный сканер

Книжный сканер

Сканер штрих-кода



Портативный сканер документов

3D-сканер

Сканер кинопленки

можно сохранить на электронном носителе. Эти данные являются цифровой моделью объекта.

Процесс перевода аналогового представления объекта в цифровое называют **оцифровкой (или аналого-цифровым преобразованием, АЦП).**

Оцифровка данных производится на специальном оборудовании, позволяющем преобразовать аналоговый сигнал в цифровой. Такое устройство называют аналого-цифровым преобразователем.

В дальнейшем оцифрованные данные могут использоваться для обработки на компьютере, передачи по компьютерным сетям. Оцифровать можно текст, фотографии, рисунки, звук, видео, кино- и фотопленки.

При сканировании изображения с физических объектов (текст, фотографии, рисунки) дискретизация характеризуется **разрешением** (количеством пикселей на единицу длины по каждому из измерений) и **глубиной цвета.**

Для оцифровки текста или графических изображений применяются различные сканеры (пример 11.8). Сегодня существуют 3D-сканеры — устройства, которые анализируют форму предмета и создают на основе полученных данных его 3D-модель. Сканеры кинопленки позволяют преобразовать изображения на кинопленке в цифровые видеофайлы. Программное обеспечение для работы со сканерами дает возможность настраивать параметры сканирования.

При выводе цифрового изображения на принтер или 3D-принтер про-

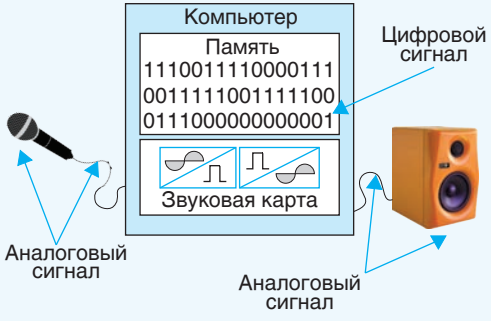
исходит обратное преобразование — из цифровой формы в аналоговую. В результате мы получаем аналоговое представление объекта: рисунок на бумаге или материальный объект.

При оцифровке сигнала, привязанного ко времени (звук, видео), основными параметрами являются **частота дискретизации** (частота измерения) и **разрядность** количества бит, выделяемых для записи результатов измерения.

Звук в компьютер можно ввести с микрофона или с любого аудиоустройства, подключенного к компьютеру. Аналого-цифровой преобразователь встроен в звуковую карту. Оцифровка производится специальным программным обеспечением (например, Audacity). При выводе звука происходит обратное преобразование сигнала из цифрового в аналоговый (пример 11.9). Для этого на звуковой карте имеется цифро-аналоговый преобразователь.

В современные смартфоны встроен цифровой фотоаппарат. Изображения, полученные с его помощью, сохраняются в цифровой форме. Затем они могут быть загружены в компьютер для обработки, передачи по компьютерным сетям или для хранения. Цифровые изображения можно просмотреть на экране монитора или распечатать на принтере.

Пример 11.9. Преобразование звука:



1. В чем отличие информации и данных? Приведите примеры.
2. Что такое носитель информации?
3. Что понимают под информационными процессами?
4. В чем отличие аналогового сигнала от цифрового?
5. Что понимают под оцифровкой?
6. Какие устройства применяют при оцифровке?

Упражнения

1 Костя учится в художественной школе и пишет картины акварелью. Никите понравилась последняя Костина картина, и он сфотографировал ее с помощью смартфона. Костя тоже решил сохранить картину в цифровом формате и отсканировал ее. Будут ли одинаковыми файлы у Кости и Никиты? Проведите свое исследование по оцифровке изображений с помощью сканера и смартфона (цифрового фотоаппарата). Сделайте выводы.

2 Подготовьте сообщения на одну из перечисленных тем.

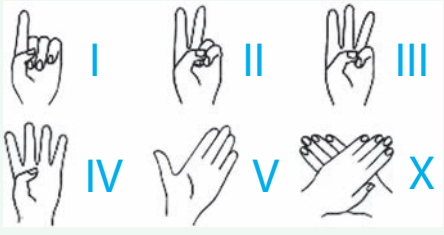
1. Цифровой и аналоговый звук. Преимущества и недостатки.
2. Правовые аспекты оцифровки книг.
3. Технологии оцифровки видео.

§ 12. Кодирование числовых данных

Пример 12.1. Способы записи чисел:

Современные	1	2	3	4	10	20	30
Египетские	I	II	III	IIII	∩	∩∩	∩∩∩
Греческие (старый стиль)	I	II	III	IIII	Δ	ΔΔ	ΔΔΔ
Греческие (новый стиль)	α	β	γ	δ	τ	κ	λ
Римские	I	II	III	IV	X	XX	XXX

У первобытного человека орудием счета были преимущественно пальцы. С их помощью можно было считать до 5, а если взять две руки, то и до 10. В древние времена люди ходили босиком. Поэтому они могли пользоваться для счета пальцами как рук, так и ног.



Известны народы, у которых единицами счета были не пальцы, а их фаланги.

Непозиционной системой счисления является славянская, в которой вместо цифр использовались буквы алфавита. Чтобы отличать буквы от цифр, над буквами с числовым значением писался специальный знак — титло.

12.1. Понятие системы счисления

Первые компьютеры называли ЭВМ — электронно-вычислительная машина. Их основным назначением было производство расчетов, для которых необходимы числовые данные. Существует большое количество способов представления числовых данных (пример 12.1). С древних времен люди использовали специальные значки для обозначения чисел. Такие значки называют цифрами.

Система счисления — способ записи числа с помощью набора условных знаков, называемых цифрами.

Системы счисления бывают позиционными и непозиционными. В позиционной системе счисления числовое значение цифры зависит от той позиции, которую цифра занимает в записи числа. В непозиционной системе счисления цифра всегда имеет одно и то же значение.

В наше время человечество использует в основном десятичную систему счисления. В ней для записи чисел используется 10 цифр: 0, 1, ... 9. Число 10 является основанием десятичной системы счисления.

Любое число в десятичной системе счисления можно записать как сумму разрядных слагаемых (пример 12.2). Числа 1, 10, 100... являются разрядными единицами. Каждая разрядная единица может быть записана в виде 10^n.

Аналогичную запись числа можно получить, если вместо 10 как основа-

ния системы счисления взять произвольное число p ($p > 1$). Разрядными единицами становятся степени основания системы счисления. Для записи числа в системе счисления с основанием p понадобится p цифр. Обычно используют первые p цифр десятичной системы счисления: 0, 1, ..., (p – 1). Например, для четверичной системы счисления это будут цифры 0, 1, 2, 3 (пример 12.3).

В общем виде число Z можно записать следующим образом:
$Zp = a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p^1 + a_0 p^0$,
где число p — основание системы счисления, коэффициенты a_n, a_{n-1}, ... a_1, a_0 — цифры числа, значения p^n, p^{n-1}, ..., p^1, p^0 — разрядные единицы.

Основание системы счисления принято указывать как нижний индекс в десятичной системе. Например, десятичное число 1443 можно записать как 1443_{10} или как $5A3_{16}$, 2643_8, 10110100011_2 (пример 12.4). Для десятичного числа индекс 10 можно не указывать.

Десятичная система счисления является примером позиционной системы счисления (пример 12.5). Примером непозиционной системы счисления является римская.

В настоящее время используются позиционные системы счисления с основаниями 2, 3, 8, 10, 16. При работе с компьютерами чаще всего используются шестнадцатеричная, восьмеричная, двоичная системы счисления.

Двоичная система счисления позволяет записывать числа с помощью двух цифр — 0 и 1. Запись числа в

Пример 12.2. Запись числа 5973 в виде суммы разрядных слагаемых в десятичной системе счисления:

$$5973 = 5 \cdot 1000 + 9 \cdot 100 + 7 \cdot 10 + 3 = 5 \cdot 10^3 + 9 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0.$$

Пример 12.3. Запись числа 12302_4 в виде суммы разрядных слагаемых в четверичной системе счисления:

$$12302 = 1 \cdot 4^4 + 2 \cdot 4^3 + 3 \cdot 4^2 + 0 \cdot 4^1 + 2 \cdot 4^0.$$

Пример 12.4. Запись числа 1443_{10} в разных системах счисления:

$$\mathbf{5A3}_{16} = 5 \cdot 16^2 + A \cdot 16^1 + 3 \cdot 16^0 = 5 \cdot 256 + 10 \cdot 16 + 3 = 1280 + 160 + 3 = 1443$$

$$2643_8 = 2 \cdot 8^3 + 6 \cdot 8^2 + 4 \cdot 8^1 + 3 \cdot 8^0 = 2 \cdot 512 + 6 \cdot 64 + 4 \cdot 8 + 3 = 1024 + 384 + 32 + 3 = 1443$$

$$\mathbf{10110100011}_2 = 1 \cdot 2^{10} + 0 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 * 1024 + 0 + 1 * 256 + 1 \cdot 128 + 0 + 1 \cdot 32 + 0 + 0 + 0 + 1 * 2 + 1 = 1024 + 256 + 128 + 32 + 2 + 1 = 1443$$

Пример 12.5. Запись числа.

В записи числа 111 первая единица обозначает сотни, вторая — десятки, а третья — единицы (числовое значение — сто одиннадцать).

В записи числа III каждая цифра I имеет значение единицы (числовое значение — три).

Из истории известно, что человек применял системы счисления с разными основаниями. В Китае долго пользовались пятеричной системой счисления. Племена майя считали в двадцатеричной системе счисления. Шестидесятеричную систему счисления использовали в Вавилоне. Напоминанием об этой системе счисления сегодня является деление минуты на 60 секунд, часа — на 60 минут, а угла — на 360 градусов.

В основе счета дюжинами лежит двенадцатеричная система счисления, которая используется до сих пор: в году 12 месяцев, на циферблате 12 часов. Для обозначения цифр в двенадцатеричной системе, кроме 10 цифр десятичной системы счисления, использовались еще два значка для обозначения чисел 10 и 11. В разные времена и в разных странах использовали: для 10 — T (англ. ten), D (лат. decem), X (римское 10); для 11 — E (англ. eleven) или O (фр. onze). Можно использовать буквы латинского алфавита — A(10) и B(11). Кроме того, иногда для обозначения 10 используют перевернутую двойку (ƻ), для 11 — перевернутую тройку (ε).

Пример 12.6. Числа в шестнадцатеричной системе счисления:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11 ... 19, 1A, 1B ... 1F, 20 ... 29, 2A ... 2F, 20 ... 99, 9A, 9B, 9C, 9D, 9E, 9F, A0, A1 ... FE, FF, 100...

Пример 12.7. Запись чисел в разных системах счисления.

Десятичная	Шестнадцатеричная	Двоичная
1	1	1
2	2	10
7	7	111
24	18	11 000
127	7F	1 111 111
256	100	100 000 000
1025	401	10 000 000 001

двоичной системе счисления является двоичным кодом числа.

В восьмеричной системе счисления используются цифры от 0 до 7. Ее применение для компьютеров обусловлено тем фактом, что в одном байте 8 бит. С помощью восьмеричных чисел записывают коды чисел и машинных команд. Сейчас восьмеричную систему счисления практически полностью вытеснила шестнадцатеричная.

В шестнадцатеричной системе счисления используются 16 цифр: 10 цифр из десятичной системы счисления — 0...9 — и 6 букв латинского алфавита — A, B, C, D, E, F (пример 12.6). Система счисления с основанием 16 широко используется в компьютерной документации и при написании программ непосредственно в машинном коде. Например, для записи адресов команд, цветовых констант.

12.2. Перевод чисел из одной позиционной системы счисления в другую

Любое число имеет значение и форму представления. Значение числа задает количественную меру и определяется его отношением к значениям других чисел («больше», «меньше», «равно»). Форма представления числа определяет способ записи числа с помощью предназначенных для этого цифр. Значение числа не зависит от способа его представления: число с одним и тем же значением может быть записано по-разному (пример 12.7). Системы счисления определяют форму представления чисел, а поскольку их много, то возникает вопрос о возможности

и способах перехода от одной формы представления числа к другой. В дальнейшем будем рассматривать только позиционные системы счисления.

Перевод числа из системы счисления с основанием p в систему счисления с основанием q обозначают как $Z_p \to Z_q$. Непосредственный перевод выполнять непросто, поэтому чаще всего рассматривают переводы $Z_p \to Z_r \to Z_q$, где обычно $r = 10$. То есть для выполнения переводов нужно уметь переводить числа в десятичную систему счисления и из десятичной в систему счисления с другим основанием.

Для получения алгоритма перевода числа из десятичной системы счисления в систему счисления с другим основанием рассмотрим запись числа в виде суммы разрядных слагаемых:

$$Z_p = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p^1 + a_0 p^0$$

В этой сумме каждое слагаемое за исключением последнего обязательно делится на p. Тогда получаем, что последнее слагаемое a_0 является остатком от деления исходного числа на p. Разделим число на p, получим сумму разрядных слагаемых со старшей степенью на 1 меньше. Найдя остаток его деления на p, получим значение $a1$. Продолжая таким образом, получим все значения a_i.

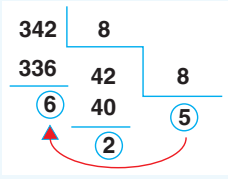
В примере 12.8 рассмотрены переводы чисел из десятичной системы счисления.

Алгоритм перевода $Z_{10} \to Z_p$:

1. Разделить нацело исходное число на основание новой системы счисления p и найти остаток от деления. Это будет цифра a_0.

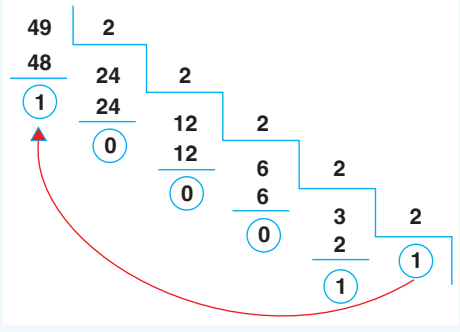
Пример 12.8. Перевод чисел из десятичной системы счисления.

Действия по алгоритму перевода чисел из десятичной системы счисления обычно представляют «лесенкой», т. е. следующим образом:



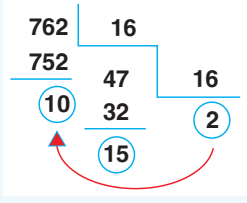
В примере реализован перевод числа 342 в восьмеричную систему счисления. Результат — 526_8.

1. Перевести число 49 в двоичную систему счисления:



$49 = 110001_2$

2. Перевести число 762 в шестнадцатеричную систему счисления:



Для записи результата необходимо полученные остатки представить шестнадцатеричными цифрами: $10 = A_{16}$, $15 = F_{16}$. Результат $762 = 2FA_{16}$.

Пример 12.9. Перевод чисел в десятичную систему счисления:

$11001_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 0 + 0 + 1 = 25_{10}$;

$3045_8 = 3 \cdot 8^3 + 0 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 3 \cdot 512 + 0 + 32 + 5 = 1573_{10}$;

$A3D_{16} = A \cdot 16^2 + 3 \cdot 16^1 + D \cdot 16^0 = 10 \cdot 256 + 3 \cdot 16 + 13 = 2621_{10}$

Дробные числа можно переводить аналогично:

$12{,}3_5 = 1 \cdot 5^1 + 2 \cdot 5^0 + 3 \cdot 5^{-1} = 5 + 2 + \frac{3}{5} = 7 + 0{,}6 = 7{,}6_{10}$.

Пример 12.10. Таблица тетрад, триад и двоичных пар.

Цифра	Тетрада	Триада	Пара
0	0000	000	00
1	0001	001	01
2	0010	010	10
3	0011	011	11
4	0100	100	
5	0101	101	
6	0110	110	
7	0111	111	
8	1000		
9	1001		
A	1010		
B	1011		
C	1100		
D	1101		
E	1110		
F	1111		

Пример 12.11. Перевод числа из шестнадцатеричной системы счисления в двоичную:

$B27_{16} = \underbrace{B}_{1011}\ \underbrace{2}_{0010}\ \underbrace{7}_{0111} = 101100100111_2$

2. Частное от деления снова разделить нацело на p с выделением остатка. Продолжать до тех пор, пока частное от деления не окажется меньше p.

3. Полученные остатки от деления, записанные в порядке, обратном порядку их получения, являются записью числа в системе счисления с основанием p.

В примере 12.9 приведены переводы чисел в десятичную систему счисления.

Алгоритм перевода $Z_p \rightarrow Z_{10}$ вытекает из способа представления числа в системе счисления с основанием p:

1. Представить число в виде суммы разрядных слагаемых по степеням p.

2. Выполнить арифметические операции в десятичной системе счисления.

Отдельно рассматривается ситуация переводов $Z_p \rightarrow Z_q$, если p и q являются степенями двойки. В этом случае в качестве промежуточной системы счисления удобно выбирать двоичную. Перевод из системы счисления с основанием степени двойки в двоичную основан на том, что каждой цифре в этой системе счисления соответствует группа двоичных цифр:

- шестнадцатеричной цифре соответствует группа из четырех двоичных цифр ($16 = 2^4$), называемая тетрадой;
- восьмеричной цифре соответствует группа из трех двоичных цифр ($8 = 2^3$), называемая триадой;
- четверичной цифре соответствует пара двоичных цифр ($4 = 2^2$).

Таблицы тетрад, триад и двоичных пар приведены в примере 12.10. Для перевода числа из системы счисления

с основанием 16 в двоичную систему счисления каждую цифру числа заменяют соответствующей тетрадой (пример 12.11). При переводе из восьмеричной системы счисления цифры заменяются триадами (пример 12.12), а из четверичной — парами (пример 12.13).

При переводе из двоичной системы счисления число разбивается соответственно на группы по 4, 3 или 2 цифры справа налево. При необходимости слева к числу можно приписать нули. Затем производится замена тетрады (триады или пары) на соответствующую цифру (пример 12.14). Перевод $Z_{16} \rightarrow Z_8$ и $Z_4 \rightarrow Z_{16}$ показан в примерах 12.15 и 12.16.

Калькулятор в ОС Windows позволяет выполнить переводы чисел из одной системы счисления в другую. Работает калькулятор с системами счислений, основаниями которых являются 2, 8, 10 и 16. Для осуществления переводов калькулятор должен быть в режиме **Программист** (пример 12.17). Обозначения для систем счисления: Hex — шестнадцатеричная, Dec — десятичная, Oct — восьмеричная, Bin — двоичная. Для перевода числа с помощью калькулятора нужно:

1. Выбрать основание системы счисления исходного числа.
2. Набрать число.
3. Результат отобразится сразу для всех систем счисления.

В режиме **Программист** калькулятор может работать только с целыми числами.

Пример 12.12. Перевод числа 362_8 в двоичную систему счисления:

$$362_8 = \underbrace{3}_{011}\ \underbrace{6}_{110}\ \underbrace{2}_{010} = 011110010_2$$

Нуль в начале записи числа можно опустить. Ответ: 11110010_2.

Пример 12.13. Перевод числа 3202_4 в двоичную систему счисления:

$$3202_4 = \underbrace{3}_{11}\ \underbrace{2}_{10}\ \underbrace{0}_{00}\ \underbrace{2}_{10} = 11100010_2$$

Пример 12.14. Перевод чисел из двоичной системы счисления (знак ′ отделяет тетрады, триады или пары).

В шестнадцатеричную:

$$110011010_2 = 0001\,'\,1001\,'\,1010 = 19A_{16}$$

В восьмеричную:

$$11011010111_2 = 011\,'\,011\,'\,010\,'\,111 = 3327_8$$

В четверичную:

$$10110111_2 = 10\,'\,11\,'\,01\,'\,11 = 2313_4$$

Пример 12.15. Перевод числа $C36_{16}$ в восьмеричную систему счисления.

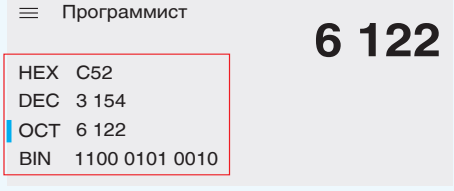
Сначала переведем число в двоичную систему счисления, затем разобьем его на триады и получим восьмеричную запись: $C56_{16} = 1100\ 0101\ 0110_2 = 110\,'\,001\,'\,010\,'\,110 = 6126_8$.

Пример 12.16. Перевод числа 23103_4 в шестнадцатеричную систему счисления.

Переведем число в двоичную систему счисления, затем разобьем его на тетрады и получим шестнадцатеричную запись:

$$23103_4 = 10\ 11\ 01\ 00\ 11_2 = 0010\,'\,1101\,'\,0011 = 2D3_{16}.$$

Пример 12.17. Перевод числа 6122_8 с помощью калькулятора.



1. Что такое система счисления?
2. Какими бывают системы счисления?
3. Как перевести число в десятичную систему счисления?
4. Как перевести число из десятичной системы счисления?
5. Для чего используются триады и тетрады при переводе чисел из одной системы счисления в другую?

Упражнения

1 Переведите числа в десятичную систему счисления.

1. 1001_2, 1110101_2, 100001_2.
2. 2121_3, 2001_3, 2213_4, 2332_4.
3. 456_8, 302_8, 165_8.
4. $A54_{16}$, 679_{16}, FDC_{16}.

2 Переведите числа из десятичной системы счисления в указанную:

1. 345, 219, 50270 → Z_{16}.
2. 234, 672, 1021 → Z_8.
3. 92, 131 → Z_4.
4. 85, 201 → Z_3.
5. 85, 129, 311 → Z_2.

3 Выполнение перевода «лесенкой» можно осуществить в Excel. Откройте файл с примером 12.8. Необходимые формулы можно посмотреть в режиме показа формул (Ctrl + ~). Используйте пример для перевода чисел из упражнения 2.

	A	B	C
1	342	8	
2	=B2*B1	=ЦЕЛОЕ(A1/B1)	8
3	=A1-A2	=C3*C2	=ЦЕЛОЕ(B2/C2)
4		=B2-B3	

4* Найдите в Excel справку по функциям ДЕС, ДВ.В.ДЕС, ВОСЬМ.В.ДЕС и др. Используйте эти функции для проверки правильности выполнения перевода чисел в упражнениях 1 и 2.

5 Осуществите перевод чисел между указанными системами счисления:

1. $3201_4 \rightarrow Z_2 \rightarrow Z_8 \rightarrow Z_{16}$;
2. $5612_8 \rightarrow Z_2 \rightarrow Z_4 \rightarrow Z_{16}$;
3. $F1A_{16} \rightarrow Z_2 \rightarrow Z_4 \rightarrow Z_8$;
4. $4567_8 \rightarrow Z_{16} \rightarrow Z_4$;
5. $D91_{16} \rightarrow Z_8 \rightarrow Z_4$.

6 Определите, в каком порядке следует осуществлять перевод числа в следующие системы счисления: $Z_{10} \rightarrow Z_2 \rightarrow Z_8 \rightarrow Z_{16}$ и $Z_4 \rightarrow Z_{16} \rightarrow Z_{10}$. Ответ обоснуйте.

7 Запишите минимальное и максимальное.

1. Пятизначные числа в четверичной системе счисления.
2. Четырехзначные числа в восьмеричной системе счисления.
3*. Трехзначные числа в двенадцатеричной системе счисления.

Получите десятичные представления записанных чисел. Сделайте выводы.

8 Запишите число, следующее за 34_8, 22_3, 34_5, 1111_2, CF_{16}.

9 Запишите число, предшествующее 54_7, 30_4, 100_{12}.

10 Решите задачи.

1. Найдите наименьшее из чисел A, B, C и D, записанных в различных системах счисления, если $A = 1023_4$, $B = 471_6$, $C = 69_{10}$, $D = 1001010_2$.
2. В системе счисления с некоторым основанием p число 58_{10} записывается как 213_p. Найдите это основание.
3. Укажите все основания систем счисления, в которых запись числа 17 оканчивается на 2.
4. К записи натурального числа в восьмеричной системе счисления справа приписали два нуля. Во сколько раз увеличилось число? Ответ запишите в десятичной системе счисления.

5*. В саду 100_p фруктовых деревьев. Из них 34_p яблони, 25_p груш и 5_p вишен. Какая система счисления используется при подсчете количества деревьев?

11 Подготовьте сообщения на одну из перечисленных тем.

1. История счета.
2. Механические счетные приспособления.
3. Использование троичной системы счисления.

§ 13. Кодирование текстовых данных

13.1. Представление текста

Естественной для органов чувств человека является аналоговая форма представления информации, однако дискретная форма представления с помощью некоторого набора знаков наиболее универсальная. Для записи текста используют символы алфавита (пример 13.1).

Представление информации в алфавитной (текстовой) форме — самый распространенный способ со времен изобретения письменности. Информация передается в виде текста, записанного на каком-либо языке: русском, белорусском и т. д. Для записи текста на разных языках можно использовать один алфавит. Например, для записи текста на русском или белорусском языках используют кириллицу, а для записи текста на английском или немецком языках — латиницу.

Пример 13.1. Различные алфавиты.

Аа	Бб	Вв	Гг	Дд	Ее	Ёё	Жж	Зз
Ии	Йй	Кк	Лл	Мм	Нн	Оо	Пп	Рр
Сс	Тт	Уу	Фф	Хх	Цц	Чч	Шш	Щщ
Ъъ	Ыы	Ьь	Ээ	Юю	Яя			

Кириллический алфавит

Aa	Bb	Cc	Dd	Ee	Ff	Gg	Hh	Ii
Jj	Kk	Ll	Mm	Nn	Oo	Pp	Qq	Rr
Ss	Tt	Uu	Vv	Ww	Xx	Yy	Zz	

Латинский алфавит

Αα	Ββ	Γγ	Δδ	Εε	Ζζ	Ηη	Θθ
Ιι	Κκ	Λλ	Μμ	Νν	Ξξ	Οο	Ππ
Ρρ	Σσ	Ττ	Υυ	Φφ	Χχ	Ψψ	Ωω

Греческий алфавит

Пример 13.2. Кодирование символов:

A	0100 0001
S	0101 0011
C	0100 0011
/	0010 1111
8	0011 1000
t	0111 0100
e	0110 0101

Пример 13.3. Фрагменты кодовых таблиц: 1 — шестнадцатеричные коды символов; 2 — десятичные коды символов.

①

sp 20	! 21	« 22	# 23	$ 24	% 25	& 26	‘ 27
(28	) 29	* 2A	+ 2B	, 2C	- 2D	. 2E	/ 2F
0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
8 38	9 39	: 3A	; 3B	< 3C	= 3D	> 3E	? 3F

②

sp 32	! 33	« 34	# 35	$ 36	% 37	& 38	‘ 39	(40	) 41
* 42	+ 43	, 44	- 45	. 46	/ 47	0 48	1 49	2 50	3 51
4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61

Пример 13.4. Кодирование буквы «Л» (русской) в разных 8-разрядных кодовых таблицах.

Название таблицы	Код	Десятичное значение
Windows-1251	11001011	203
КОИ-8	11101100	236
CP855	11010001	209
CP866	10001011	139

Пример 13.5. Текст в разных кодировках.

Windows 1251

Пример текста в разных кодировках

КОИ-8

оПХЛЕП РЕЙЯРЮ Б ПЮГМШУ ЙНДХПНБЙЮУ

CP866

╧ЕшьхЕ Єхъеєр т Ерчэ√i ъюфшЕютърї

Для записи текста в память компьютера используют двоичный код — алфавит из двух символов: 0 и 1.

13.2. Понятие кодовой таблицы

Текстовая информация состоит из символов: букв, цифр, знаков препинания и др. Множество этих символов образуют компьютерный алфавит. Текст, состоящий из данных символов, человек видит на экране монитора.

Компьютер может обрабатывать информацию только в числовой форме, представленной в виде двоичного кода. Поэтому для кодирования текста каждому символу алфавита ставят в соответствие двоичный код (пример 13.2). Часто всем знакам алфавита ставятся в соответствие коды, содержащие одинаковое число двоичных разрядов.

Совокупность всех символов компьютерного алфавита и соответствующих им двоичных кодов записывают в виде таблицы. Такую таблицу называют **кодовой (кодировочной) таблицей символов.**

С помощью кодовых таблиц выполняют кодирование и декодирование текста. Часто для удобства пользователя в кодовых таблицах вместо двоичного кода записывается его десятичный или шестнадцатеричный аналог (пример 13.3). Для получения двоичного кода (из десятичного или шестнадцатеричного) нужно осуществить перевод числа в двоичную систему счисления.

Для разных компьютерных систем могут использоваться различные кодовые таблицы символов. В разных кодовых таблицах одним и тем же

символам ставится в соответствие разный двоичный код (пример 13.4). В этом случае текст, созданный на одном компьютере, нельзя будет прочитать на другом компьютере без дополнительного перекодирования — символы будут отображаться некорректно (пример 13.5). Международным стандартом стала таблица кодировки ASCII (*American Standard Code for Information Interchange* — американский стандартный код для обмена информацией). Данная таблица поддерживает 8-разрядный двоичный код. Это значит, что каждый символ будет закодирован последовательностью из 8 нулей и единиц. Такая последовательность и будет кодом символа. Всего в таблице $2^8 = 256$ символов.

Так как каждый символ кодируют последовательностью из 8 нулей и единиц, он занимает в памяти компьютера 8 бит (1 байт). В примерах 13.6—13.9 показано, как с помощью таблицы символов ASCII (см. *Приложение к главе 2*, с. 118—119) кодировать и декодировать символы.

В таблице ASCII символы латинского и русского алфавитов (прописные и строчные) идут по алфавиту. Десятичные цифры расположены в порядке возрастания их числовых значений. Это правило обычно соблюдается и в других кодовых таблицах. Такой способ кодирования текста позволяет сортировать текстовые данные по алфавиту, а числовые — по возрастанию их значений.

В кодовой таблице ASCII хранится 256 символов. Если нужно работать с текстами сразу на нескольких языках, то этих символов недостаточно.

Таблица ASCII кодов состоит из двух частей. Первая часть **(стандартная)** содержит латинские буквы, цифры, пробел, знаки препинания и специальные символы: +, /, *, %, # и др. Символы этой части имеют коды от 00000000 до 01111111 (десятичные аналоги 0—127). Вторую часть кодовой таблицы называют **альтернативной.** Символы в ней кодируются значениями от 10000000 до 11111111 (десятичные аналоги 128—255). Эта часть таблицы используется для кодирования символов национальных алфавитов и символов псевдографики, которые используются для рисования рамок и линий (символы с кодами 176—223). В стандартной части кодовой таблицы коды всех символов начинаются с 0, в альтернативной — с 1.

Пример 13.6. Определение кода символа «@».

Номер символа «@» в кодовой таблице 64. Переведем число 64 из десятичной системы счисления в двоичную. Получим: 1000000. Для получения 8-битного кода добавим 0 перед числом. 01000000 — двоичный код символа «@».

Пример 13.7. Определение кода символа «Ф».

Слева от буквы «Ф» число 148 — ее десятичный код. Переведем число 148 в двоичную систему счисления. Получим 10010100 — двоичный код буквы «Ф».

Пример 13.8. Определение для символа «Z» шестнадцатеричного кода.

Найдем в таблице букву «Z». Рядом с ней число 90. Переведем десятичное число 90 в шестнадцатеричную систему счисления. Получим: 5A.

Пример 13.9. Декодирование последовательности кодов символов.

10101000 10101101 11100100 10101110 11100000 10101100 10100000 11100010 10101000 10101010 10100000

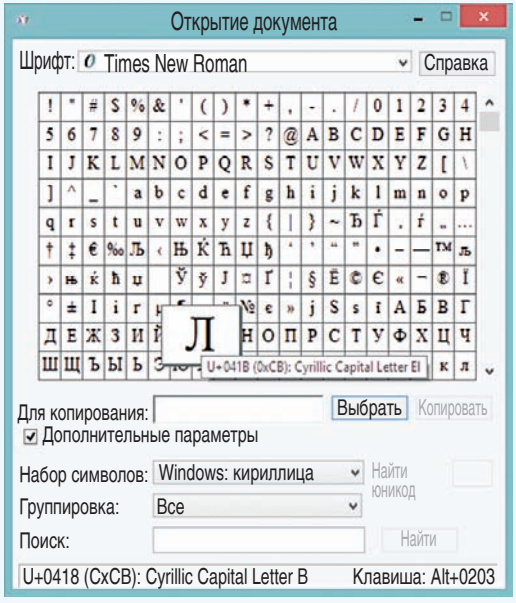
Заменим каждый двоичный код его десятичным аналогом и получим:

168 173 228 174 224 172 160 226 168 170 160

Теперь в кодовой таблице найдем соответствующие символы и получим:

информатика

Пример 13.10. Таблица символов (фрагмент).



Пример 13.11. Кодирование слова «диск».

В кодировке ASCII слову «диск» соответствует последовательность шестнадцатеричных кодов: A4 A8 E1 AA (в десятичном представлении: 164 189 225 170). Этой последовательности кодов в кодировке Unicode будут соответствовать символы: ¤¨á ª. В кодировке Unicode слово «диск» будет закодировано последовательностью шестнадцатеричных кодов: 0434 0438 0441 043A.

Формат UTF-8 был разработан 2 сентября 1992 г. Кеном Томпсоном и Робом Пайком и реализован в ОС Plan 9 (Операционная система, разработанная Bell Labs — сейчас подразделение Nokia — в конце 1980-х гг. с расчетом на сети и рабочие станции). Эта кодировка нашла широкое применение в UNIX-подобных ОС.

Сейчас широко используют кодировку Unicode (Юникод). В ней компьютерный алфавит состоит не из 256, а из 65 536 символов. Для кодирования одного символа используется последовательность из 0 и 1, имеющая длину 16 символов. При такой кодировке каждый символ будет занимать в памяти компьютера 2 байта. Просмотреть кодовую таблицу на вашем компьютере можно запустив программу **Таблица символов** (находится в разделе **Стандартные** → → **Служебные** → **Таблица символов**).

Для определения кода символа нужно выбрать этот символ в таблице (пример 13.10). На всплывающей подсказке и в нижней части окна будет указан код символа в шестнадцатеричной системе счисления и название данного символа (на английском языке).

Для поиска символа, соответствующего какому-либо коду, нужно ввести шестнадцатеричный код символа в поле **Поиск**. В выпадающем списке **Набор символов** можно выбрать определенный алфавит.

Стандартная часть кодовой таблицы ASCII совпадает с началом таблицы кодировки Unicode. Поэтому тексты, содержащие символы, которые расположены в стандартной части кодовой таблицы ASCII (цифры, буквы английского алфавита), будут без труда читаться и в кодировке Unicode.

Русские символы в таблице ASCII имеют коды, начиная с числа $80_{16} = 128_{10}$, а в таблице Unicode — с шестнадцатеричного числа 0410 (код прописной буквы А). Тексты на русском языке, набранные в кодировке ASCII, будут неверно отображаться при просмотре в Unicode (пример 13.11). Для

правильного просмотра текст необходимо преобразовать.

Распространена кодировка UTF-8 (англ. *Unicode Transformation Format, 8-bit* — формат преобразования Юникода, 8 бит). Она позволяет более компактно хранить и передавать символы, используя переменное количество байт (от 1 до 4) для кодирования. Стандарт UTF-8 сейчас является самым распространенным в Интернете. Латинские буквы, цифры и наиболее распространенные знаки препинания кодируются в UTF-8 одним байтом, и коды этих символов соответствуют их кодам в ASCII. Кириллические символы кодируются двумя байтами (пример 13.12). Структуру двоичного кода символа в кодировке UTF-8 можно посмотреть в *Приложении к главе 2* (с. 117).

Тексты вводятся в память компьютера в основном с помощью клавиатуры. На клавишах написаны знакомые нам буквы, цифры, знаки препинания и другие символы. Нажатие на определенную клавишу кодирует символ, и в памяти компьютера он хранится в форме двоичного кода. При выводе символа на экран монитора внешний вид символа восстанавливается по его двоичному коду. Кодирование и декодирование текстовых данных происходит также при записи текста в файл на компьютерный носитель и при считывании текста из файла (пример 13.13).

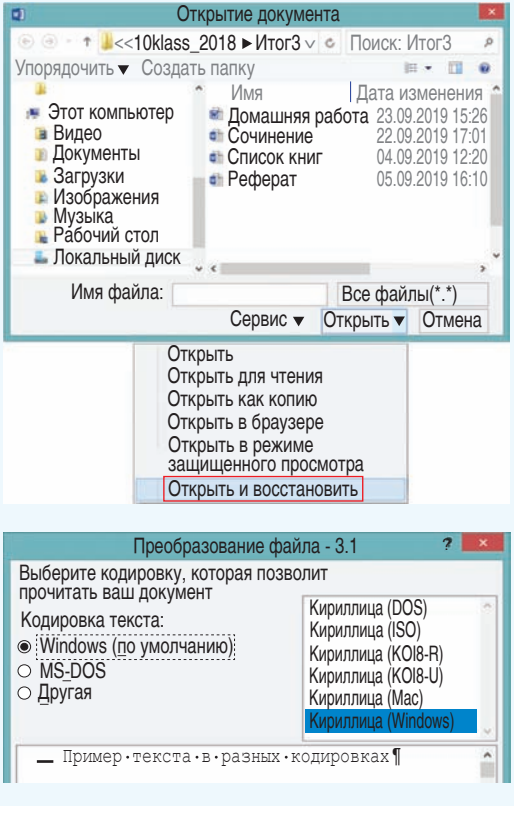
Информационный объем текста зависит от количества символов в нем и способа кодирования символов. Информационный вес одного символа равен 1 байту при использовании однобайтных (8-битных) кодовых таблиц и 2 байтам

Многие браузеры предоставляют пользователю возможность выбора кодировки страницы сайта. Однако распространение кодировки UTF-8 приводит к тому, что эта функция становится маловостребованной. По этой причине разработчики Google Chrome убрали данную функцию из последних версий браузера.

Пример 13.12. Кодирование символов в UTF-8 и Unicode.

Символ	UTF-8	Unicode
L	4C (1 байт)	00 4C
Л	D0 9B (2 байта)	04 1B

Пример 13.13. Преобразование кодировки текстового документа при его открытии в Word:



Пример 13.14.

В кодировке Unicode каждый символ кодируется 2 байтами. Нужно подсчитать количество символов в предложении и умножить на 2. В предложении 38 символов. Информационный объем — $38 \cdot 2 = 76$ байт.

Пример 13.15.

В кодировке Unicode каждый символ кодируется 2 байтами, в кодировке ASCII — 1 байтом. При перекодировке информационный объем уменьшился в 2 раза. Значит, исходный размер сообщения — $12 \cdot 2 = 24$ байта, $24 \cdot 8 = 192$ бит.

Пример 13.16.

Информационный объем сообщения 8,5 Кбайт = $8{,}5 \cdot 1024 = 8704$ байта. Одному символу сообщения соответствует 1 байт = 8 бит. С помощью 8 бит можно закодировать $2^8 = 256$ символов.

Пример 13.17*.

В Unicode каждый символ кодируется 2 байтами, поэтому в сообщении $(21 \cdot 1024)/2 = 10752$ символов. Если бы все символы в сообщении были латинскими, то информационный объем сообщения в кодировке UTF-8 составил бы 10752 байта. Информационный объем сообщения до перекодирования — 15 Кбайт = $15 \cdot 1024 = 15360$ байт. $15360 - 10752 = 4608$ — количество символов русского алфавита.

Пример 13.18.

На одной странице $30 \cdot 45 = 1350$ символов. В кодовой таблице Unicode один символ кодируется 2 байтами. Информационный объем одной страницы — $1350 \cdot 2 = 2700$ байт. Весь рассказ занимает $80 \cdot 1024 = 81\,920$ байт. Тогда количество страниц составляет $81920/2700 \approx 30{,}34 = 31$ страница.

при использовании таблицы Unicode. При использовании таблицы UTF-8 информационный вес одного символа может составлять от 1 до 4 байт.

13.3. Решение задач на кодирование текста

Пример 13.14. Определить информационный объем следующего предложения, если его закодировали с помощью кодовой таблицы Unicode:

Программирование — вторая грамотность.

Пример 13.15. Автоматическое устройство осуществило перекодировку сообщения из кодировки Unicode в кодировку ASCII. При этом информационный объем сообщения уменьшился на 12 байт. Сколько бит было в первоначальном сообщении?

Пример 13.16. Информационный объем сообщения 8,5 Кбайт. Данное сообщение содержит 8704 символа. Какое максимально возможное количество символов содержится в алфавите?

Пример 13.17*. Автоматическое устройство осуществило перекодировку сообщения, содержащего символы русского и латинского алфавитов из кодировки UTF-8 в 16-битный Unicode. (Символы латинского алфавита кодируются одним байтом, а русского — двумя байтами.) В результате преобразования сообщение стало занимать 21 Кбайт вместо первоначальных 15 Кбайт. Сколько в сообщении символов русского алфавита?

Пример 13.18. Текст рассказа занимает 80 Кбайт. На одной странице 30 строк по 45 символов. Каждый символ кодируется 16 битами в формате Unicode. Сколько страниц в рассказе?

1. Что такое алфавит?
2. Как кодируются символы?
3. Чем отличается кодирование текста при использовании разных кодовых таблиц?
4. Как определить код символа в приложении **Таблица символов**?

Упражнения

1 Используя кодовую таблицу ASCII или Unicode (программа **Таблица символов**), закодируйте следующие текстовые данные:

Файл	Байт
Кодирование	Disk
Printer	Bit
Система счисления	

2 Декодируйте двоичный код, используя кодовую таблицу ASCII.

11101000 10101010 10101110 10101011 10100000

3 Декодируйте информацию, используя таблицу ASCII.

172 174 164 165 172 (десятичные числа).
E1 AA A0 AD E0 (шестнадцатеричные числа).

4 Используя программу **Таблица символов**, определите коды символов $^1/_2$, ±, $, Щ, Ў.

5 Определите информационный объем сообщения «Участник олимпиады может писать программы на языках программирования Pascal, Python или C++».

1. В кодировке ASCII.
2. В кодировке Unicode.
3*. В кодировке UTF-8.

6 Сообщение, информационной объем которого в 16-битной кодировке равен 480 байт, перекодировали в 8-битную кодировку. После этого к сообщению дописали несколько символов, и его информационный объем стал равен 520 байт. Сколько символов дописали в сообщение?

7 Алфавит племени Тумба-Юмба состоит из 8 букв. Каков информационный объем одной буквы?

8 Сообщение, записанное буквами из 16-буквенного алфавита, содержит 21 символ. Каков информационный объем сообщения?

9 Статья, набранная на компьютере, содержит 6 страниц. На каждой странице одинаковое количество строк по 56 символов в строке. Информационный объем статьи 504 Кбит. Определите количество строк на каждой странице текста, считая, что каждый символ закодирован с использованием Unicode.

10 Скорость чтения учащегося 10-го класса составляет в среднем 1024 символа в минуту. Какой информационный объем получит учащийся, если будет непрерывно читать в течение 30 мин текст, набранный на компьютере в кодировке Unicode?

11 Для получения годовой отметки по географии учащемуся требовалось написать реферат на 15 страниц. Он выполнил это задание на компьютере, набирая текст в кодировке Unicode. Какой объем памяти (в Кбайтах) займет реферат, если в каждой строке по 72 символа, а на каждой странице помещается 28 строк? Каждый символ занимает 2 байта памяти.

12 Оцените информационный объем страницы текста из учебного пособия по информатике. Для этого посчитайте количество строк на странице и количество символов в строке. Для текста на белом и голубом фоне расчеты нужно проводить раздельно, а затем суммировать результаты. Текст набран с использованием кодировки Unicode.

13 Петя и Вася пишут друг другу письма, кодируя информацию следующим образом: каждый символ письма кодируется двоичным кодом по таблице ASCII. Затем 0 заменяется на 1, а 1 на 0. По полученным кодам в таблице отыскиваются символы, из которых складывается текст письма. Получивший письмо производит те же действия для того, чтобы письмо прочитать. Например, для кодирования слова «Привет» нужно поступить так:

П – 143 – 10001111 – 01110000 – 112 – p
р – 224 – 11100000 –00011111 – 31 – ▼
и – 168 –10101000 – 01010111 – 87 – W
в – 162 – 10100010 – 01011101 – 93 –]
е – 165 – 10100101 – 01011010 – 90 – Z
т – 226 – 11100010 – 00011101 – 29 – ↔

Получим: p ▼ W] Z ↔

С помощью программы **Калькулятор** можно не только переводить числа в двоичную систему счисления, но и производить замену 0 на 1, а 1 на 0. Чтобы заменить 0 на 1, а 1 на 0 на **Калькуляторе** (в режиме **Программист**), нужно выполнить действие Xor над двумя двоичными числами: исходным числом и числом 11111111 (например, 10001111 Xor 11111111 = 1110000).

Закодируйте этим способом: Привет, Вася! Как дела?

Декодируйте: p ▼ W] Z ↔ ▌■ } ▲ Z ■ ▒ Ф ▌

14 Для секретной переписки Оля и Света придумали свою кодовую таблицу. Декодируйте сообщение от Оли к Свете, используя часть таблицы. Придумайте коды для других букв русского алфавита.

Таблица					Сообщение
	○	□	◺	☆	
	А	Г	И	М	
	О	П	Р	С	
	Т	У	Х	Ч	
	Ь				

15 Подтвердите или опровергните утверждение «СМС-сообщение, набранное транслитом, будет стоить дешевле, чем аналогичное сообщение, набранное русскими буквами».

§ 14. Кодирование графики, звука и видео

14.1. Кодирование графики

В настоящее время при создании и хранении графических объектов в компьютере используются **растровое** и **векторное** изображения (примеры 14.1, 14.2).

Растровое изображение — совокупность отдельных точек (пикселей), каждая из которых имеет свой цвет.

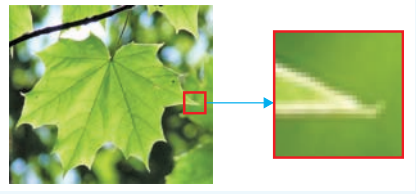
В векторном графическом изображении каждый нарисованный элемент является объектом: линия, овал, прямоугольник и др. Все объекты имеют определенный перечень значений свойств, которые описывают эти объекты (пример 14.3).

Векторное изображение — совокупность графических примитивов (объектов изображения), которые описаны с помощью числовых значений или математических формул.

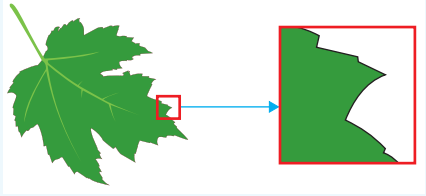
Различие в представлении растровых и векторных изображений существует лишь для графических файлов. При выводе на экран монитора изображения растрового или векторного типов, в видеопамяти компьютера *формируется информация растрового типа*. Эта информация состоит из двоичных кодов каждого пикселя. Код пикселя — информация о его цвете.

Если на черно-белое изображение наложить сетку и каждой ячейке белого цвета поставить в соответствие 1, а черного цвета — 0 (или наоборот: 1 — черный, 0 — белый), то можно создать

Пример 14.1. Растровое изображение.



Пример 14.2. Векторное изображение.



Пример 14.3. Свойства отрезка и круга в векторном изображении.

Отрезок:

- начало и конец отрезка — две пары чисел, определяющих координаты точек на координатной плоскости;
- значения, определяющие цвет, толщину и тип линии (сплошная, пунктирная и др.).

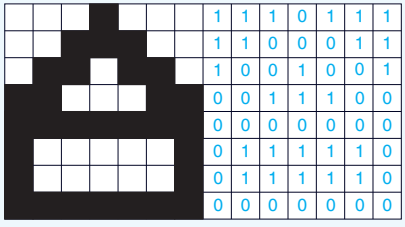
Таким образом, для описания отрезка необходимо 7 числовых значений, описывающих его свойства. Этих значений достаточно для описания отрезка любого размера, цвета и толщины.

Круг:

- координаты центра круга и его радиус;
- значения ширины контурной линии, цвета контура, типа линии контура окружности;
- цвет заливки внутренней области, ограниченной окружностью.

Для описания свойств круга может использоваться 3—7 числовых значений. Координаты центра и радиус являются обязательными параметрами, остальные параметры могут отсутствовать.

Пример 14.4. Кодирование черно-белого изображения:



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

Размер изображения 7×8 пикселей, информационный объем изображения равен $8 \cdot 7 \cdot 1$ бит = 56 бит = 7 байт.

Пример 14.5. Кодирование одного пикселя изображения.

Количество цветов	Количество бит для кодирования
$2^2 = 4$	2 бита
$2^3 = 8$	3 бита
$2^4 = 16$	4 бита
$2^8 = 256$	8 бит (1 байт)
$2^{16} = 65536$	16 бит (2 байта)
$2^{24} = 16777216$	24 бит (3 байта)

Пример 14.6. Кодирование цвета при использовании палитры из 16 цветов.

Яркость	Red	Green	Blue	Цвет
0	0	0	0	черный
0	0	0	1	синий
0	0	1	0	зеленый
0	0	1	1	голубой
0	1	0	0	красный
0	1	0	1	фиолетовый
0	1	1	0	коричневый
0	1	1	1	серый
1	0	0	0	темно-серый
1	0	0	1	ярко-синий
1	0	1	0	ярко-зеленый
1	0	1	1	ярко-голубой
1	1	0	0	ярко-красный
1	1	0	1	ярко-розовый
1	1	1	0	ярко-желтый
1	1	1	1	белый

матрицу изображения из нулей и единиц (пример 14.4).

Для черно-белого изображения информационный объем пикселя равен одному биту. Соответственно, информационный объем изображения в битах будет равен количеству пикселей в изображении — произведению ширины на длину изображения.

Чем больше цветов в изображении, тем больше битов понадобится для кодирования одной точки (пример 14.5).

На экране монитора цвет пикселя изображения формируется смешением трех цветовых лучей: красного (англ. **R**ed), зеленого (англ. **G**reen) и синего (англ. **B**lue). Поэтому при кодировании цветных изображений используется цветовая модель RGB. В современной версии модели RGB на каждый пиксель отводится 24 бита, по 8 бит на каждый из трех основных цветов, что дает возможность закодировать 16,7 млн оттенков.

Если для каждого из основных цветов использовать меньшее количество бит, то, соответственно, можно закодировать и меньшее количество цветовых оттенков. Кодирование цветов при использовании 16-цветной палитры приведено в примере 14.6. В этом случае информационный объем каждого пикселя составляет 4 бита.

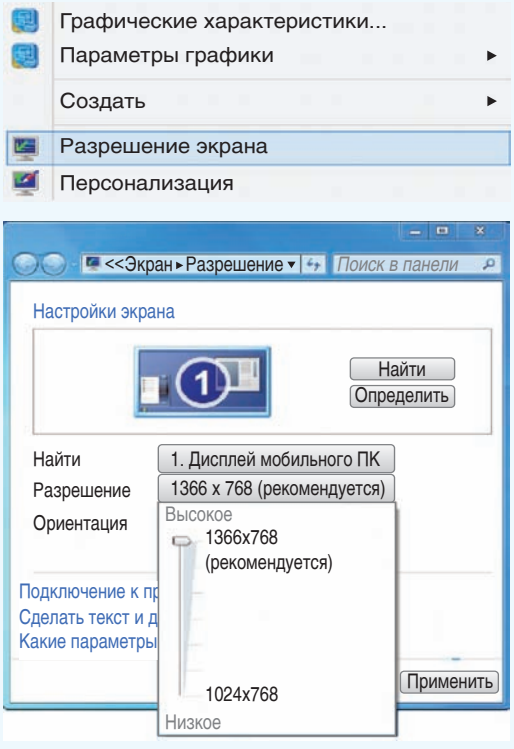
Качество изображения на экране зависит от разрешающей способности монитора и глубины цвета. Любое графическое изображение на экране монитора состоит из строк, которые содержат определенное количество пикселей. Мониторы могут иметь раз-

личные разрешающие способности: 1024×768, 1280×1024, 1366×768, 1920×1080 и др. Разрешение экрана может быть изменено. Для этого в контекстном меню **Рабочего стола** нужно выбрать команду **Разрешение экрана** (пример 14.7).

Глубина цвета определяется количеством бит, используемых для кодирования цвета пикселя. Современные мониторы поддерживают глубину цвета 32 бита: 24 бита хранят код цвета в RGB-палитре, еще 8 бит отводятся на хранение значений прозрачности цвета (альфа-канал).

В файле с графическим растровым изображением хранится информация о цвете каждого пикселя изображения. В таком виде сохраняются изображения в формате BMP. Другие растровые форматы (JPEG, GIF, PNG) хранят изображение в сжатом виде: при сохранении к изображению, которое на экране представлено матрицей пикселей, применяют алгоритмы архивации. При сохранении в формате GIF количество цветов уменьшается до 256. При сохранении в формате JPEG сохраняется информация не о каждом пикселе, а о группе пикселей, при этом часть информации теряется. Такое сжатие необратимо, восстановить изображение в исходном виде невозможно. Однако человеческий глаз не всегда способен заметить изменения, поэтому формат JPEG является одним из самых распространенных для компактного хранения фотографий. При сохранении изображения в формате PNG используется алгоритм сжатия без потерь.

Пример 14.7. Изменение разрешения экрана.



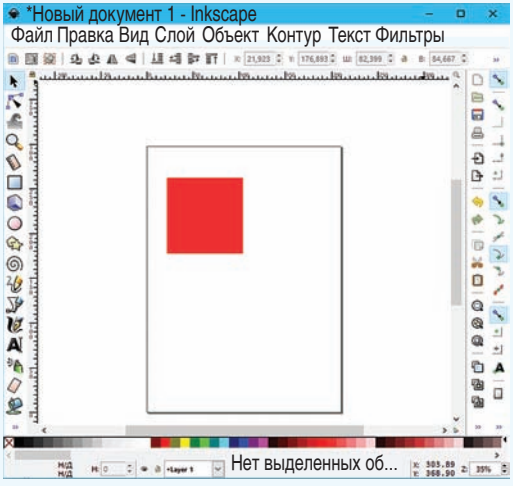
Научная дисциплина, изучающая вопросы измерения цветовых характеристик, называется *колориметрией* (или *метрологией цвета*).

Научную основу колориметрии как сочетание нескольких основных цветов положил Исаак Ньютон. Он в 1676 г. с помощью трехгранной призмы разложил белый солнечный свет на цветовой спектр и выделил семь основных цветов: красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый.

В 1756 г. М. В. Ломоносов сформулировал трехкомпонентную теорию цвета. До этого считалось, что цвет состоит из семи составляющих.

Спустя столетие Герман Грассман ввел для нее математический аппарат.

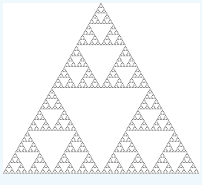
Пример 14.8. Рисунок в Inkscape:



Просмотр файла рисунка в NotePad:

```
<g
   inkscape:label="Layer 1"
   inkscape:groupmode="Layer"
   id="layer1">
   <rect
     style="opacity:1;fill:#ff0000;
     id="rect4485"
     width="50.648811"
     height="50.648811"
     x="70.303574"
     y="109.5238"
     rx="1.984375"
     ry="0.26458332"  />
  </g>
</svg>
```

Пример 14.9. Фрактальная графика.



Слово *фрактал* образовано от латинского *fractus* и в переводе означает «состоящий из фрагментов». Оно было предложено математиком Бенуа Мандель-Бротом в 1975 г. для обозначения самоподобных структур.

Формат PNG предназначен прежде всего для использования в Интернете.

В файле с векторным изображением сохраняются математические значения свойств объектов изображения, которые необходимы для его построения. Файлы формата SVG можно просматривать и редактировать в текстовом виде — например, в редакторе NotePad (пример 14.8).

Фрактальная графика, как и векторная, основывается на математических вычислениях. Базовым элементом фрактальной графики является математическая **формула**. Это приводит к тому, что в памяти компьютера не хранится никаких объектов, а изображение строится по уравнениям. При помощи этого способа можно строить как простейшие изображения, так и сложные иллюстрации, имитирующие ландшафты (пример 14.9).

14.2. Кодирование звука

Современные компьютерные устройства оснащены устройствами для ввода и вывода звуковой информации. Понятие *звук* тесно связано с понятием *волна*. Как и любая волна, звук имеет амплитуду и частоту. Амплитуда характеризует громкость звука. Частота определяет тон, высоту. Обычный человек способен слышать звуковые колебания в диапазоне частот от 16—20 Гц до 15—20 кГц.

При оцифровке звук подвергается дискретизации. Аналого-цифровой преобразователь, встроенный в звуковую карту, производит замеры амплитуды звуковой волны через равные

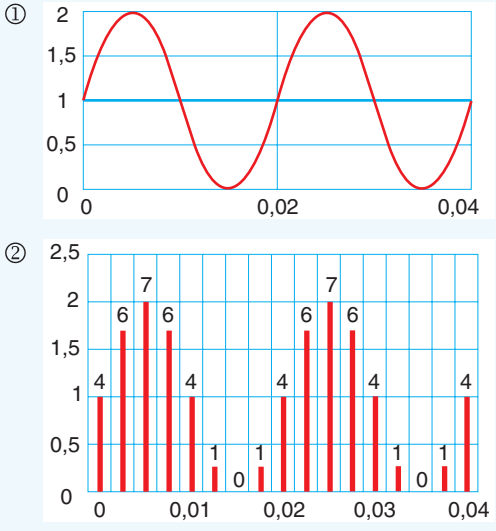
промежутки времени. Полученные числовые значения преобразуются в двоичный код и сохраняются (пример 14.10).

Количество измерений за одну секунду определяет **частоту дискретизации** звука. Точность преобразования зависит от разрядности АЦП. **Разрядность** АЦП характеризует количество дискретных значений, которые преобразователь может выдать на выходе. Например, двоичный 8-разрядный АЦП способен выдать 256 дискретных значений (0...255), $2^8 = 256$. С разрядностью связано **разрешение АЦП** — минимальное изменение величины аналогового сигнала, которое может быть преобразовано. Разрешение равно разности значений, соответствующих максимальному и минимальному выходному коду, деленной на количество выходных дискретных значений.

Звуковой файл длительностью в 1 с при частоте дискретизации 8 КГц и разрядности 8 бит будет занимать объем в 7,8 Кбайт. При увеличении частоты дискретизации до 44,1 КГц и разрядности 24 бита объем файла увеличится до 129,2 Кбайт.

Чтобы записать стереозвук, следует одновременно кодировать два независимых канала звука. Количество каналов может быть большим: 4 (квадро), 6 (Dolby Digital). Сегодня существуют технологии, поддерживающие до 16 звуковых каналов. Воспроизведение многоканальных фонограмм через систему громкоговорителей, расположенных по окружности от слушателя, называют **объемным звуком.**

Пример 14.10. Временная диаграмма кодирования звука: 1 — аналоговый сигнал; 2 — дискретный сигнал.



Для кодирования будем использовать трехразрядный двоичный код. С помощью такого кода можно закодировать 8 различных значений. Разобьем диапазон изменения амплитуды сигнала на 8 уровней. Каждому отсчету сигнала присвоим ближайший к нему номер от 0 до 7. Далее выполним кодирование полученных значений сигнала трехразрядным двоичным кодом (в таблице приведены первые 6 значений).

Время	Значение	Код
0	4	100
0,0025	6	110
0,005	7	111
0,0075	6	110
0,01	4	100
0,0125	1	001
0,015	0	000
0,0175	1	001

Максимальное и минимальное значения амплитуды сигнала равны 2 и 0 соответственно. Разрешение АЦП в данном случае определяется как $2 / 8 = 0{,}25$.

В многоканальном звуке один канал используют для низкочастотных эффектов (выводится на сабвуфер). Поскольку диапазон частот этого канала очень ограничен (по сравнению с другими каналами), то часто его обозначают «.1». Тогда обозначение 5.1 говорит о том, что это 5 каналов с полным диапазоном частот и 1 канал для низкочастотных эффектов. Общее количество каналов 6.

Пример 14.11. Используемые частоты дискретизации звука.

Частота	Устройство
8000 Гц	Телефон, достаточно для речи
22 050 Гц	Радио
48 000 Гц	DAT
5 644 800 Гц	Профессиональные устройства записи

Пример 14.12. Звуковые форматы.

Формат	Описание
WAV	Waveform audio format — формат без сжатия данных
FLAC	Free Lossless Audio Codec — сжатие без потерь (до 55 % от исходного размера)
MP3	MPEG-1 Audio Layer 3, сжатие с учетом восприятия человеком (до 10 % от исходного размера).
WMA	Windows Media Audio был представлен как замена MP3. В последних версиях формата, начиная с Windows Media Audio 9.1, предусмотрено кодирование без потери качества, многоканальное кодирование объемного звука и кодирование голоса.

Чем выше частота дискретизации и разрядность, тем качественнее получается звук (пример 14.11). Однако с увеличением частоты возрастает объем памяти, необходимый для хранения цифрового сигнала, а с увеличением разрядности — и вычислительная нагрузка на цифровые преобразователи.

Чтобы уменьшить объем, занимаемый цифровыми аудиоданными, применяют различные методы сжатия (пример 14.12). При сжатии звука без потерь к исходному звуку применяют алгоритмы архивации. Возможно удаление избыточных данных — связей между соседними отсчетами цифрового звукового сигнала. Сжатие звука с потерями основано на несовершенстве человеческого слуха (человек не воспринимает сверхнизкие и сверхвысокие частоты, более слабый сигнал становится неслышимым на фоне более сильного и др.).

14.3. Кодирование видео

Видео хранится на диске в виде файлов, содержащих видео-, аудио- и другие потоки, а также метаданные. Видеофайл часто называют медиаконтейнером. В любой момент из контейнера можно вынуть, например, видео- или аудиодорожки, перекодировать их и поместить в другой контейнер, т. е. изменить формат видеофайла. Существует несколько форматов видеоконтейнеров (пример 14.13).

Кодирование звукового сопровождения видеоинформации ничем не отличается от кодирования звука.

Изображение в видео состоит из отдельных кадров, которые меняются с

определенной частотой. Кадр кодируется так же, как обычное растровое изображение.

Видеоданные характеризуются частотой кадров и экранным разрешением. Если частота смены кадров равна 25, то для каждой секунды видео необходимо хранить в памяти 25 кадров. Разрешение для видео обычно составляет 768×484 (для стандарта NTSC) или 768×576 (для стандартов PAL и SECAM).

В основе кодирования цветного видео лежит стандартная модель RGB.

Если представить каждый кадр изображения как отдельный рисунок, то видеоизображение будет занимать очень большой объем. Например, одна секунда записи в системе PAL будет занимать 25 Мбайт. Поэтому на практике используются различные алгоритмы сжатия для уменьшения объема видеоданных (пример 14.14). Для просмотра такого видео нужен кодек.

Кодек (CoDec) — это сокращение слов *компрессор* и *декомпрессор*. Кодек — набор файлов, драйверов и библиотек, необходимых для упаковки видео или звукового файла в сжатый формат и воспроизведения сжатого файла. Кодек может отслеживать массивы точек изображения с одинаковыми значениями (например, синий цвет моря) и вместо того, чтобы запоминать информацию о каждой точке (яркость и цвет), записать лишь первую (ключевую) точку и количество повторений этой точки до момента изменения ее цвета.

Пример 14.13. Форматы видеофайлов.

Формат	Описание
AVI (Audio-Video Interleaved)	Представляет собой контейнер, который может содержать потоки четырех типов: Video, Audio, MIDI, Text
MP4	Современный формат файлов для хранения цифровых видео- и аудиопотоков, являющийся частью стандарта MPEG-4 (см. пример 14.14). Файлы в этом формате можно проиграть практически на любых устройствах, начиная от смартфонов и заканчивая игровыми приставками
FLV	Flash Video — медиаконтейнер, использующийся в сети Интернет
MOV	Формат файла, разработанный компанией Apple для хранения видео, графики, анимации и 3D

Пример 14.14. Стандарты сжатия видео.

MPEG
(Moving Pictures Expert Group)

Один из основных стандартов сжатия. Имеет разновидности:

MPEG-1 — формат сжатия для компакт-дисков (CD-ROM);

MPEG-2 — формат сжатия для DVD-дисков, цифрового телевидения;

MPEG-4 — формат, который уменьшает видеопоток сильнее, чем MPEG-2, но сохраняет хорошее качество.

HD

High Definition — формат высокого разрешения и особой четкости. Может использовать разрешение 1920×1080.

Windows Media

Разработан компанией Microsoft и предназначен для хранения сжатого видео и звука.

Пример 14.15. Изображение состоит из $128 \cdot 128 = 2^7 \cdot 2^7 = 2^{14}$ пикселей. Для хранения изображения выделено 4 Кбайт $= 4 \cdot 2^{10}$ байт $= 2^{12}$ байт $= 2^{15}$ бит. Значит, информационный объем одного пикселя равен $2^{15} / 2^{14} = 2$ бит. С помощью 2 бит можно закодировать $2^2 = 4$ цвета.

Пример 14.16. Информационный объем одного пикселя равен $3 \cdot 4 = 12$. Количество цветов $2^{12} = 4096$.

Пример 14.17. Размер фотографии в дюймах 4×4, т. к. $10 / 2{,}5 = 4$.

Количество пикселей

$$4 \cdot 4 \cdot 400 \cdot 400 = 2^8 \cdot 10000.$$

Информационный объем

$$2^8 \cdot 10000 \cdot 24 \text{ бит} =$$
$$= 2^8 \cdot 2^4 \cdot 625 \cdot 2^3 \cdot 3 \text{ бит} =$$
$$= 2^{15} \cdot 3 \cdot 5^4 \text{ бит} = 2^{12} \cdot 3 \cdot 5^4 \text{ байт} =$$
$$= 4 \cdot 3 \cdot 5^4 \text{ Кбайт} = 7500 \text{ Кбайт}.$$

Пример 14.18. При частоте 8 кГц за 1 с производится 8000 измерений. Чтобы сохранить одно измерение, нужно 8 бит. Тогда для всех измерений:

$$8 \cdot 8 \cdot 8000 = 2^9 \cdot 1000 \text{ бит} =$$
$$= 2^6 \cdot 1000 \text{ байт} = 62{,}5 \text{ Кбайт}.$$

Пример 14.19. Информационный объем стереоаудиофайла вычисляют по формуле: $V = 2 \cdot R \cdot t \cdot N$, где V — объем аудиофайла, R — разрядность аудиоадаптера, N — частота дискретизации, t — время звучания, умножение на 2 показывает, что кодируются два канала. Тогда $t = \frac{V}{2RN}$. Преобразуем исходные данные:

$$V = 70 \text{ Мбайт} = 70 \cdot 220 \cdot 8 = 70 \cdot 223 \text{ бит};$$
$$N = 32 \text{ кГц} = 32000 \text{ Гц} = 25 \cdot 1000.$$

Тогда $t = \frac{70 \cdot 2^{23}}{2 \cdot 16 \cdot 2^5 \cdot 1000} = \frac{70 \cdot 2^{23}}{2^{10} \cdot (2 \cdot 5)^3} =$

$$= \frac{70 \cdot 2^{10}}{5^3} = 573{,}44 \text{ с} = 9{,}56 \text{ мин}.$$

Пример 14.20. Найдем объем графики: $800 \cdot 600 \cdot 24$ бит $= 11520000$ бит $\approx$ $\approx 1{,}38$ Мбайт. Размер видео: $1{,}38 \cdot 24 \times$ $\times (5 \cdot 60) = 9936$ Мбайт. Разрядность при кодировании звука равна 8, т. к. $256 = 2^8$. Размер звука: $11250 \cdot 8 \cdot 2 \cdot (5 \cdot 60) =$ $= 54000000$ бит $\approx 6{,}4$ Мбайт. Объем видеофайла: $9936 + 6{,}4 = 9942{,}4$ Мбайт $\approx$ $\approx 9{,}7$ Гигабайт.

14.4. Решение задач на кодирование графики, звука и видео

Пример 14.15. Для хранения изображения размером 128×128 точек выделено 4 Кбайт памяти. Определите, какое максимальное число цветов в палитре.

Пример 14.16. Цвет пикселя, формируемого принтером, определяется тремя составляющими: голубой, пурпурной и желтой красками. Под каждую составляющую одного пикселя отвели по 4 бита. В какое количество цветов можно раскрасить пиксель?

Пример 14.17. Фотография размером 10×10 см была отсканирована с разрешением 400 dpi при глубине цвета 24 бита. Определите информационный объем полученного растрового файла в килобайтах (принять 1 дюйм = = 2,5 см).

Пример 14.18. Определить информационный объем в Кбайтах моноаудиофайла длительностью звучания 8 с при глубине звука 8 бит и частоте 8 кГц.

Пример 14.19. Рассчитать время звучания стереоаудиофайла, который был закодирован с частотой дискретизации 32 кГц. Разрядность аудиоадаптера — 16 бит, информационный объем файла равен 70 Мбайт.

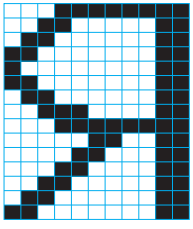
Пример 14.20. Какой объем будет иметь видео, передаваемое с разрешением кадра 800×600 пикселей с 24-битовой глубиной цвета, скоростью воспроизведения 24 кадра в секунду и длительностью 5 мин? Известно, что стереозвук, наложенный на видео, имеет 256 уровней громкости, частота дискретизации равна 11 250 Гц.

1. Какие два принципа представления графических изображений используются в компьютерной графике?
2. Из чего состоит растровое изображение?
3. Что представляет собой векторное изображение?
4. Чем отличается растровое изображение от векторного?
5. Графические изображения какого типа выводятся на экран монитора?
6. Что понимается под разрешающей способностью экрана монитора и глубиной цвета?
7. Как хранятся растровые и векторные изображения в файле?
8. Чем определяется частота дискретизации звука?
9. Что такое разрядность аналого-цифрового преобразователя?
10. Как кодируется видео?

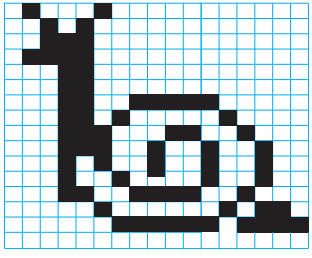
Упражнения

1 Создайте матрицу из нулей и единиц для кодирования следующих черно-белых изображений (можно использовать электронные таблицы; для проверки правильности посчитайте суммы по строкам и столбцам). Определите информационный объем изображений.

1.

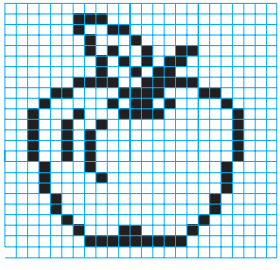


2.

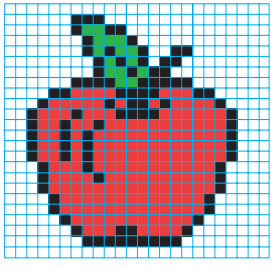


2 Определите информационный объем размещенных ниже растровых изображений (одна клетка — один пиксель).

1. Черно-белое изображение



2. Четырехцветное изображение

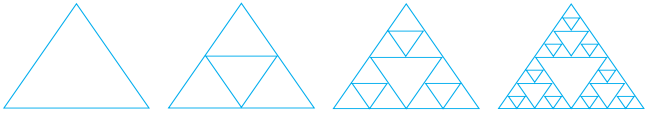


3 16-цветный рисунок содержит 500 байт информации. Из скольких точек он состоит?

4 Определите требуемый объем (в мегабайтах) видеопамяти для реализации графического режима монитора с разрешающей способностью 1024×768 пикселей при количестве отображаемых цветов 65 536.

5* На рисунке в примере 14.9 изображена фигура, которая называется «треугольник Серпинского». Для ее построения можно воспользоваться следующим алгоритмом:

1. Строим равносторонний треугольник (уровень 0).
2. Соединяем середины сторон построенного треугольника отрезками. Получается 4 новых треугольника. Из исходного треугольника удаляется внутренность срединного треугольника. Получаем 3 треугольника уровня 1.
3. Поступая точно так же с каждым из треугольников первого уровня, получим множество, состоящее из 9 равносторонних треугольников второго уровня.
4. Повторяем процесс до нужного уровня.



Реализуйте данный алгоритм в среде программирования.

6 Определите длительность звучания стереоаудиофайла, занимающего 468,75 Кбайт памяти при глубине звука 16 бит и частоте 48 кГц.

7 При переводе в дискретную форму аналогового сигнала длительностью 2 мин 8 с использовалась частота дискретизации 32 Гц и 16 уровней дискретизации. Найти в байтах размер полученного кода аналогового сигнала.

8 Экспериментально было установлено, что на временном отрезке [0; 20] амплитуда звукового сигнала изменялась в соответствии с законом $A(t) = 2\ \sin 1{,}7x \cdot \sin 0{,}2x$. Используя электронные таблицы, постройте временные диаграммы кодирования звука.

а) диаграмму аналогового сигнала (в Excel можно использовать тип диаграммы — точечная);

б) диаграмму дискретного сигнала (в Excel можно использовать тип диаграммы — гистограмма).

9 Какой объем будет иметь черно-белое видео, передаваемое с разрешением кадра 800×600, скоростью воспроизведения 24 кадра в секунду и длительностью 30 мин без звука?

10 Кадры видеозаписи закодированы в режиме истинного цвета (24 бита на пиксель) и сменяются с частотой 25 кадров в секунду. Кадр имеет размеры 720×480 пикселей. Частота дискретизации 22 кГц, глубина кодирования звука 16 бит. Оцените объем минуты видеозаписи в мегабайтах (с точностью до десятых), если файл записан с 10-кратной степенью сжатости.

11 Камера снимает видео без звука с частотой 60 кадров в секунду, при этом изображения используют палитру, содержащую 224 цвета. При записи файла на сервер полученное видео преобразуют так, что частота кадров уменьшается до 20, а изображения преобразуют в формат, использующий палитру из 256 цветов. Другие преобразования и иные методы сжатия не используются. 10 секунд преобразованного видео в среднем занимают 512 Кбайт. Сколько Мбайт в среднем занимает 1 минута исходного видео?

§ 15. Различные подходы к измерению информации

15.1. Содержательный подход

Сегодня информация является одним из основных ресурсов человечества. Поэтому так важны ответы на вопросы, как много информации мы получили, передали, обработали, создали.

При физических измерениях величину сравнивают с эталоном, а с чем сравнивать информацию?

Известно несколько подходов к измерению количества информации.

При **содержательном подходе** измерение информации происходит с точки зрения ее содержания, т. е. определяется, в какой мере пришедшая информация (знания) уменьшает незнание. Человек получает знания посредством сообщений. Чем больше пополняет наши знания сообщение, тем большее количество информации в нем заключено (пример 15.1).

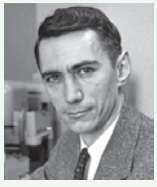
Основателем такого подхода к измерению информации является К. Шенон, который ввел приведенное ниже определение.

Сообщение, которое уменьшает неопределенность знания в два раза, несет **1 бит** информации.

Неопределенность знания о результате некоторого события — количество возможных результатов.

Если в некотором сообщении содержатся сведения о том, что произошло одно из N равновероятных событий, то количество информации i, содержащееся в сообщении, можно определить из формулы Хартли: $N = 2^i$ (пример 15.2).

Клод Элвуд Шеннон (1916—2001) — американский инженер, криптоаналитик и математик. Является основателем теории информации.



Пример 15.1. У вас сегодня контрольная по математике. Учитель обычно дает 2 варианта заданий. До контрольной вы не знаете свой вариант, поэтому неопределенность знания равна 2. Если вариантов на контрольной 4, то неопределенность знания равна 4.

Ральф Винтон Лайон Хартли (1888—1970) — американский ученый-электронщик. Предложил генератор Хартли, преобразование Хартли и сделал вклад в теорию информации, введя в 1928 г. логарифмическую меру информации: $i = \log_2 N$.

Пример 15.2. Количество информации, которую вы получите, узнав свой вариант контрольной работы, можно рассчитать по формуле Хартли.

Если вариантов два, то $2 = 2^i$, следовательно, $i = 1$. Вы получите 1 бит информации.

Если вариантов 4, то $4 = 2^i$, следовательно, $i = 2$. Вы получите 2 бита информации.

Если вариантов 6, то $6 = 2^i$, следовательно, $i \approx 2{,}58$. Вы получите 2,58 бита информации. Для получения значения i в этом случае нужно посчитать значение $i = \log_2 6$ (например, на калькуляторе или по таблице[1]).

Пример 15.3. При компьютерном наборе текста на русском языке обычно используется 32 буквы (буква «ё» применяется очень редко). Тогда, согласно формуле Хартли, $32 = 2^5$, одна буква русского алфавита несет 5 бит информации.

Пример 15.4. Единицы измерения объемов информации.

Килобайт (Кбайт):
2^{10} = 1024 байта

Мегабайт (Мбайт):
2^{20} = 1024 килобайта = 1 048 576 байт

Гигабайт (Гбайт):
2^{30} = 1024 мегабайта = 1 073 741 824 байта

Терабайт (Тбайт):
2^{40} = 1024 гигабайта = 1 099 511 627 776 байт

Петабайт (Пбайт):
2^{50} = 1024 терабайта =
= 1 125 899 906 842 624 байта

Эксабайт (Эбайт):
2^{60} = 1024 петабайта =
= 1 152 921 504 606 846 976 байт

Зеттабайт (Збайт):
2^{70} = 1024 эксабайта =
= 1 180 591 620 717 411 303 424 байта

Йоттабайт (Йбайт):
2^{80} = 1024 зеттабайта =
= 1 208 925 819 614 629 174 706 176 байт

С точки зрения теории измерений единицы измерения количества информации, в названии которых есть части «кило», «мега» и др., некорректны. Эти приставки используются в метрической системе мер, в которой в качестве множителей кратных единиц применяется коэффициент 10^n, где n = 3, 6, 9 и т. д.

15.2. Алфавитный подход

Если человек получает текстовое сообщение, то количество информации может быть измерено количеством символов в нем. Однако каждый символ алфавита тоже несет какое-то количество информации. Если предположить, что все символы алфавита встречаются в тексте с одинаковой частотой (равновероятно), то количество информации i, которое несет каждый символ, вычисляется по формуле Хартли: $N = 2^i$, где N — мощность алфавита (пример 15.3). Под мощностью алфавита понимают количество символов в нем.

Алфавитный (объемный) подход используется, если для преобразования, хранения и передачи информации применяют технические средства.

При использовании двоичного алфавита один символ несет 1 единицу информации — **1 бит.**

Для измерения объемов информации применяют производные единицы измерения (пример 15.4).

Для двоичного представления текстов в компьютере часто используется восьмиразрядный код. С его помощью можно закодировать алфавит из 256 символов. Один символ из алфавита мощностью $256 = 2^8$ несет в тексте 8 бит информации. Такое количество информации называется **байтом.**

Объем текста измеряется в байтах. При восьмиразрядном кодировании 1 символ = 1 байт, и информационный объем текста определяется количе-

[1] Таблица двоичных логарифмов: http://sokolova-aa.ru/cribs/tablitsa-dvoichnykh-logarifmov-tselykh-chisel-ot-1-do-64 (дата доступа: 28.07.2019).

ством символов в нем. Если весь текст состоит из K символов, то при алфавитном подходе объем V содержащейся в нем информации равен: $V = K \cdot i$, где i — информационный вес одного символа в используемом алфавите.

15.3. Вероятностный подход

В жизни различные события происходят с разной вероятностью. Событие «летом идет снег» маловероятно, а у события «осенью идет дождь» вероятность велика. Если в коробке 10 красных шаров и 40 зеленых, то вероятность достать не глядя зеленый шар больше, чем вероятность достать красный.

Для количественного измерения вероятности используют следующий подход: если общее количество возможных исходов какого-либо события равно N, а K из них — те, в которых мы заинтересованы, то вероятность интересующего нас события может быть посчитана по формуле $p = \frac{K}{N}$ (пример 15.5).

Чем меньше вероятность события, тем больше информации содержит сообщение о том, что это событие произошло.

Вероятностный подход применяется для измерения количества информации при наступлении событий, имеющих разную вероятность. Связь между вероятностью события и количеством информации в сообщении о нем выражается формулой $\frac{1}{p} = 2^i$, где p — вероятность события, а i — количество информации (пример 15.6).

В 1999 г. утвердили ряд новых приставок для единиц измерения количества информации: киби (kibi), меби (mebi), гиби (gibi), теби (tebi), пети (peti), эксби (exbi)[1]. В настоящее время они используются наравне с «кило», «мега» и др.

Пример 15.5. В коробке 16 красных шаров и 48 зеленых. Какова вероятность достать зеленый шар не глядя? Красный?

Всего в коробке $N = 16 + 48 = 64$ шара. Нас интересует зеленый шар. Благоприятный исход — достать любой из 48 зеленых шаров. Поэтому $p = \frac{48}{64} = 0{,}75$. Аналогично получим вероятность достать красный шар: $p = \frac{16}{64} = 0{,}25$. Значит, вероятность вытащить зеленый шар в 3 раза больше, чем вытащить красный шар.

* Если произошло несколько разновероятностных событий, то количество информации можно определять по формуле Шеннона, предложенной им в 1948 г: $I = -(p_1 \log_2 p_1 + p_2 \log_2 p_2 + \ldots\ p_N \log_2 p_N)$, где I — количество информации; N — количество возможных событий; p_i — вероятность i-го события.

Легко заметить, что если вероятности p_1, ..., p_N равны между собой, то каждая из них равна $1/N$ и формула Шеннона превращается в формулу Хартли.

Пример 15.6. Какое количество информации несет сообщение «Из коробки достали красный шар» для примера 15.5?

По формуле $\frac{1}{p} = 2^i$ получаем $\frac{1}{0{,}25} = 2^i \Rightarrow 4 = 2^i$. Тогда $i = 2$, т. е. мы получили 2 бита информации.

[1] https://ru.wikipedia.org/wiki/Двоичные_приставки (дата доступа: 28.07.2019).

Пример 15.7. Выпадение каждой грани кубика равновероятно. Поэтому количество информации от одного результата бросания находится из уравнения $2^i = 6$. Тогда $2^i = 6 < 8 = 2^3$, $i = 3$ бита.

Можно рассуждать и так:

$i = \log_2 6 = 2{,}585$ бита ≈ 3 бита.

Пример 15.8. 11 Кбайт = 11 * 1024 = = 11264 байт. Поскольку количество байт равно количеству символов, то использована восьмибитная кодовая таблица. Мощность алфавита: $2^8 = 256$.

Пример 15.9. Нужно закодировать 65 равновероятностных значений. По формуле Харли: $2^i = 65 < 128 = 2^7$, $i = 7$ бит.

Пример 15.10. Пусть в ящике x желтых мячей. Тогда вероятность достать желтый мяч равна $\frac{x}{32}$. Подставляем в формулу, связывающую вероятность с количеством информации:

$$\frac{1}{p} = 2^i \Rightarrow \frac{1}{x/32} = 2^4 \Rightarrow \frac{32}{x} = 16 \Rightarrow x = 2.$$

В ящике 2 желтых мяча.

Пример 15.11*. Всего в коробке $10 + 8 + 6 = 24$ кубика. Вероятности доставания кубиков: $p_{кр} = \frac{10}{24}$, $p_{зел} = \frac{8}{24}$, $p_{ж} = \frac{6}{24}$. Количество информации по формуле Шеннона:

$$I = -(p_{кр}\log_2 p_{кр} + p_{зел}\log_2 p_{зел} + p_{ж}\log_2 p_{ж}) =$$
$$= \left(\frac{10}{24}\log_2\frac{10}{24} + \frac{8}{24}\log_2\frac{8}{24} + \frac{6}{24}\log_2\frac{6}{24}\right) \approx$$
$$\approx -(0{,}42 \cdot (-1{,}26) + 0{,}33 \cdot (-1{,}58) +$$
$$+ 0{,}25 \cdot (-2)) = 1{,}5506$$

15.4. Решение задач на определение объема информации

Пример 15.7. При игре в кости используется кубик с шестью гранями. Сколько бит информации получает игрок при каждом бросании кубика? Ответ округлить в бóльшую сторону до ближайшего целого количества бит.

Пример 15.8. Объем сообщения равен 11 Кбайт. Сообщение содержит 11 264 символа. Какова мощность алфавита?

Пример 15.9. Измеряется температура воздуха, которая может быть целым числом от –30 до 34 градусов. Какое наименьшее целое количество бит необходимо, чтобы закодировать одно измеренное значение?

Пример 15.10. В ящике находится 32 теннисных мяча, среди которых есть мячи желтого цвета. Наудачу вынимается один мяч. Сообщение «Извлечен мяч желтого цвета» несет 4 бита информации. Сколько желтых мячей в ящике?

Пример 15.11*. В коробке лежат кубики. Известно, что среди них 10 красных, 8 зеленых, 6 желтых. Вычислите вероятность доставания кубика каждого цвета. Сколько информации несет сообщение, что достали кубик любого цвета? Для решения задачи воспользоваться формулой Шеннона.

1. В чем сущность содержательного подхода к измерению информации?
2. Что обозначает 1 бит информации при алфавитном подходе к измерению информации?
3. Когда применяют вероятностный подход к измерению информации?

Упражнения

1 Сколько различных звуковых сигналов можно закодировать с помощью 6 бит?

2 Какое количество информации несет сообщение о том, что человек живет в первом или втором подъезде, если в доме 16 подъездов?

3 Сообщение о том, что ваш друг живет на 10-м этаже, несет 4 бита информации. Сколько этажей в доме?

4 Азбука Морзе позволяет кодировать символы для радиосвязи, задавая комбинацию точек и тире. Сколько различных символов (цифр, букв, знаков пунктуации и т. д.) можно закодировать, используя код Морзе длиной не менее пяти и не более шести сигналов (точек и тире)?

5 В ящике находится 32 теннисных мяча, среди которых есть мячи черного цвета. Наудачу вынимается один мяч. Сообщение «Извлечен мяч НЕ черного цвета» несет 3 бита информации. Сколько черных мячей в ящике?

6 К празднику надували белые и синие шарики. Белых шариков 24. Сообщение о том, что лопнул синий шарик, несет 2 бита информации. Сколько всего надули шариков?

7 В школьной библиотеке 32 стеллажа с книгами, на каждом — по 8 полок. Пете сообщили, что нужный учебник находится на 2-й полке 4-го стеллажа. Какое количество информации получил Петя?

8 Для регистрации на некотором сайте пользователю нужно придумать пароль, состоящий из 10 символов. В качестве символов можно использовать десятичные цифры и шесть первых букв латинского алфавита, причем буквы используются только заглавные. Пароли кодируются посимвольно. Все символы кодируются одинаковым и минимально возможным количеством бит. Для хранения сведений о каждом пользователе в системе отведено одинаковое и минимально возможное целое число байт. Какой объем будет занимать информация о паролях 1000 пользователей?

9* В некоторой стране автомобильный номер длиной 6 символов составляют из заглавных букв (задействовано 30 различных букв) и десятичных цифр в любом порядке. Каждый такой номер в компьютерной программе записывается минимально возможным и одинаковым целым количеством байт (при этом используют посимвольное кодирование, и все символы кодируются одинаковым и минимально возможным количеством бит). Определите объем памяти в байтах, отводимый этой программой для записи 50 номеров.

10 В озере обитают 12 500 окуней, 25 000 пескарей, а карасей и щук по 6250. Какое количество информации несет сообщение о том, что поймали пескаря? Сколько информации мы получим, когда поймаем какую-нибудь рыбу?

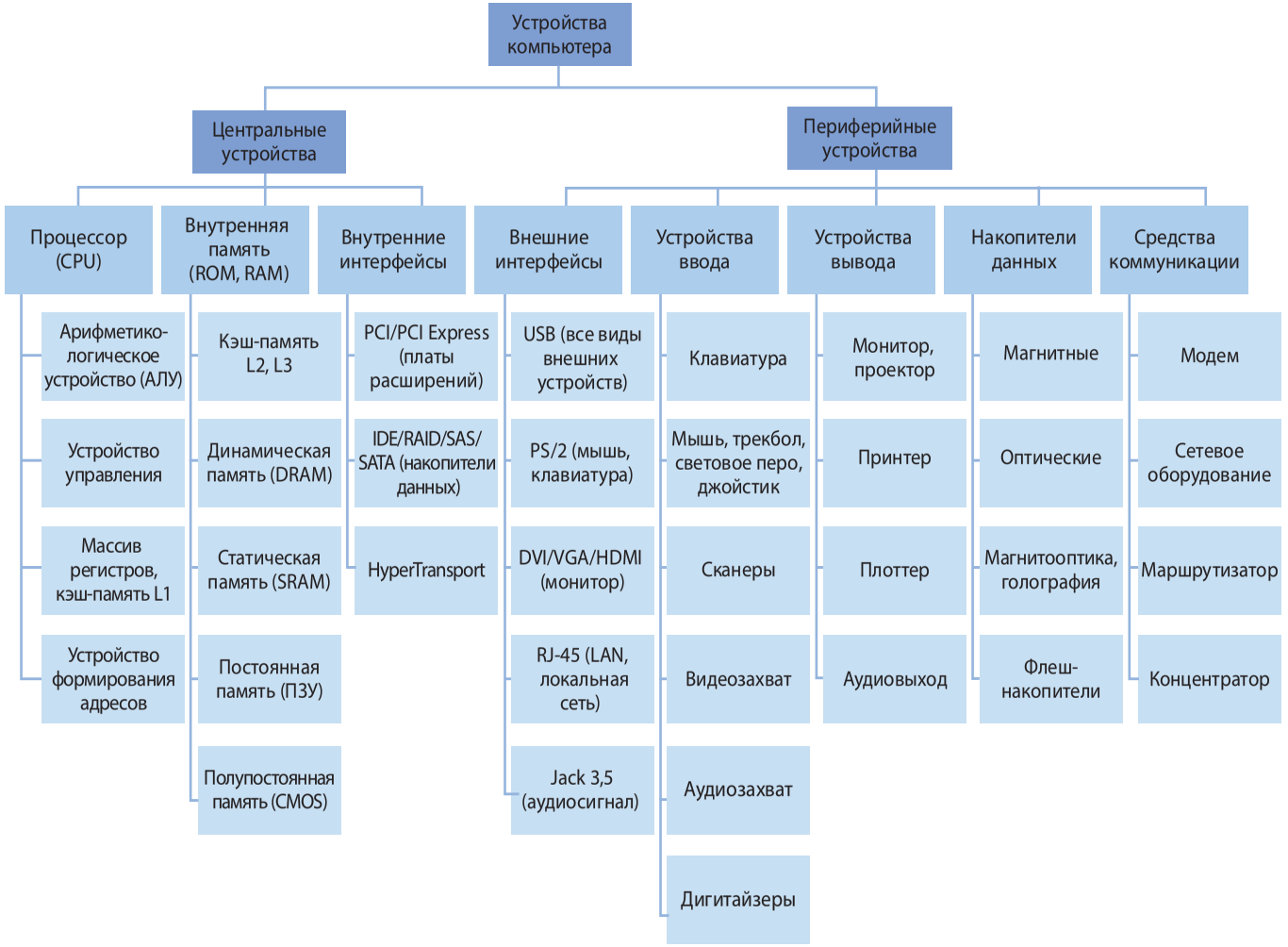
11* Какое сообщение содержит большее количество информации?

1. Бабушка испекла 16 пирожков. Лера съела один пирожок.
2. Бабушка испекла 12 пирожков с капустой, 12 пирожков с повидлом. Маша съела один пирожок.
3. Бабушка испекла 16 пирожков с капустой, 24 пирожка с повидлом. Миша съел один пирожок.

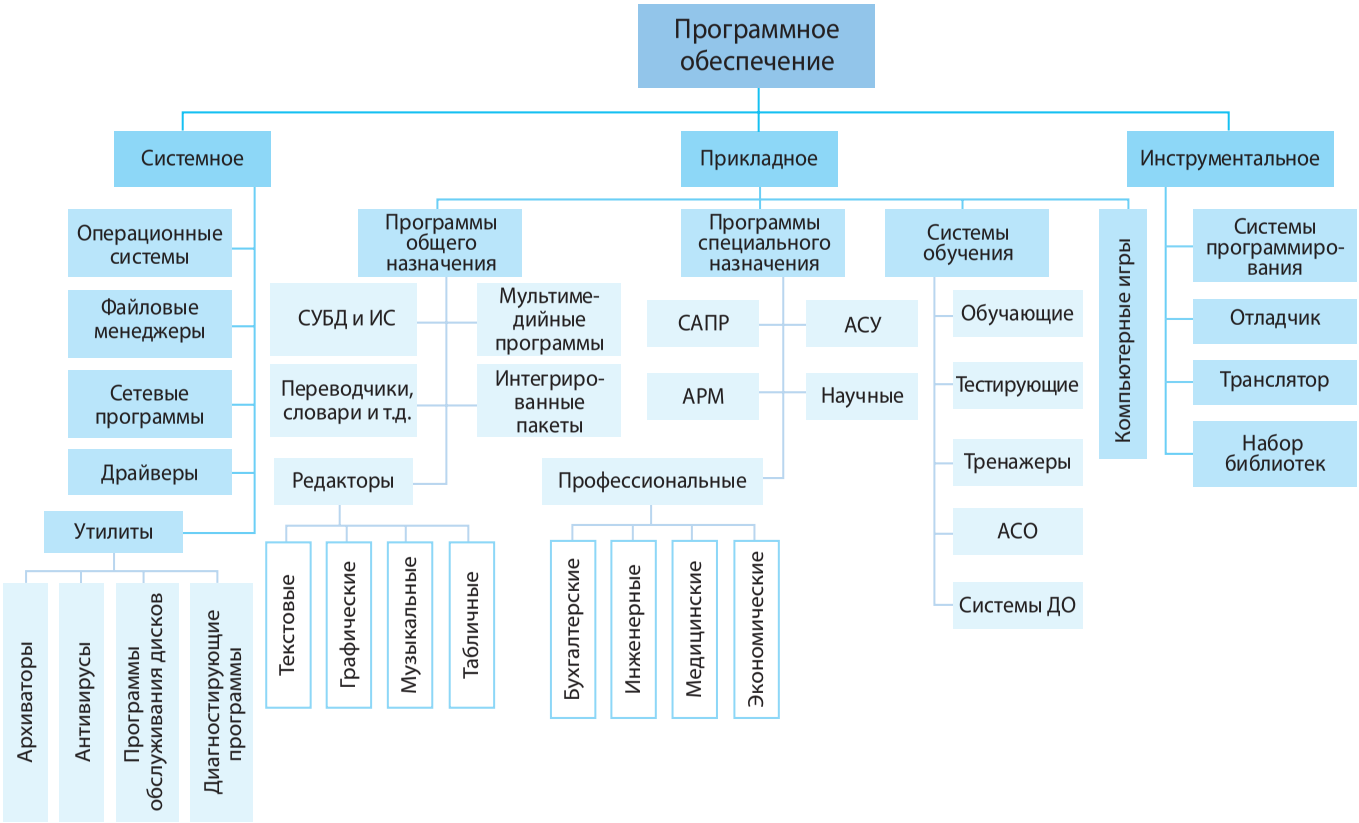
ПРИЛОЖЕНИЕ К ГЛАВЕ 2

Структурная схема компьютера

Основные устройства компьютера



Классификация программного обеспечения по назначению:



СУБД и ИС — системы управления базами данных и информационные системы;
САПР — системы автоматизированного проектирования;
АРМ — автоматизированные рабочие места;
АСУ — автоматизированные системы управления;
АСО — автоматизированные системы обучения;
ДО — дистанционное обучение.

Структура кода в кодировке UTF-8

Количество байт	Значащих бит	Первый байт	Шаблон полностью
1	7	0xxxxxxx	0xxxxxxx
2	11	110xxxxx	110xxxxx 10xxxxxx
3	16	1110xxxx	1110xxxx 10xxxxxx 10xxxxxx
4	21	11110xxx	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Таблица двоичных логарифмов

№	x	№	x	№	x	№	x
1	0,00000	17	4,08746	33	5,04439	49	5,61471
2	1,00000	18	4,16993	34	5,08746	50	5,64386
3	1,58496	19	4,24793	35	5,12928	51	5,67243
4	2,00000	20	4,32193	36	5,16993	52	5,70044
5	2,32193	21	4,39232	37	5,20945	53	5,72792
6	2,58496	22	4,45943	38	5,24793	54	5,75489
7	2,80735	23	4,52356	39	5,28540	55	5,78136
8	3,00000	24	4,58496	40	5,32193	56	5,80735
9	3,16993	25	4,64386	41	5,35755	57	5,83289
10	3,32193	26	4,70044	42	5,39232	58	5,85798
11	3,45943	27	4,75489	43	5,42626	59	5,88264
12	3,58496	28	4,80735	44	5,45943	60	5,90689
13	3,70044	29	4,85798	45	5,49185	61	5,93074
14	3,80735	30	4,90689	46	5,52356	62	5,95420
15	3,90689	31	4,95420	47	5,55459	63	5,97728
16	4,00000	32	5,00000	48	5,58496	64	6,00000

Кодовая таблица символов стандарта ASCII

№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ
0		16	▶	32		48	0	64	@	80	P	96	`	112	p
1		17	◀	33	!	49	1	65	A	81	Q	97	a	113	q
2		18	↕	34	"	50	2	66	B	82	R	98	b	114	r
3	©	19	‼	35	#	51	3	67	C	83	S	99	c	115	s
4	♥	20	¶	36	$	52	4	68	D	84	T	100	d	116	t
5	♦	21	§	37	%	53	5	69	E	85	U	101	e	117	u
6	♣	22	■	38	&	54	6	70	F	86	V	102	f	118	v
7	•	23		39	'	55	7	71	G	87	W	103	g	119	w
8		24	↑	40	(	56	8	72	H	88	X	104	h	120	x
9		25	↓	41	)	57	9	73	I	89	Y	105	i	121	y
10		26	→	42	*	58	:	74	J	90	Z	106	j	122	z
11	♂	27	←	43	+	59	;	75	K	91	[	107	k	123	{
12	♀	28	∟	44	,	60	<	76	L	92	\	108	l	124	|
13		29	↔	45	-	61	=	77	M	93	]	109	m	125	}
14	♫	30	▲	46	.	62	>	78	N	94	^	110	n	126	~
15		31	▼	47	/	63	?	79	O	95	_	111	o	127	⌂

№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ	№	Символ
128	А	144	Р	160	а	176	▒	192	└	208	╨	224	р	240	Ё						
129	Б	145	С	161	б	177	▒	193	┴	209	╤	225	с	241	ё						
130	В	146	Т	162	в	178	▓	194	┬	210	╥	226	т	242	Є						
131	Г	147	У	163	г	179	│	195	├	211	╙	227	у	243	є						
132	Д	148	Ф	164	д	180	┤	196	─	212	╘	228	ф	244	І						
133	Е	149	Х	165	е	181	╡	197	┼	213	╒	229	х	245	і						
134	Ж	150	Ц	166	ж	182	╢	198	╞	214	╓	230	ц	246	Ў						
135	З	151	Ч	167	з	183	╖	199	╟	215	╫	231	ч	247	ў						
135	И	152	Ш	168	и	184	╕	200	╚	216	╪	232	ш	248	°						
137	Й	153	Щ	169	й	185	╣	201	╔	217	┘	233	щ	249	∙						
138	К	154	Ъ	170	к	186	║	202	╩	218	┌	234	ъ	250	·						
139	Л	155	Ы	171	л	187	╗	203	╦	219	▉	235	ы	251	√						
140	М	156	Ь	172	м	188	╝	204	╠	220	▄	236	ь	252	№						
141	Н	157	Э	173	н	189	╜	205	═	221	▌	237	э	253	¤						
142	О	158	Ю	174	о	190	╛	206	╬	222	▐	238	ю	254	■						
143	П	159	Я	175	п	191	┐	207	╧	223	▀	239	я	255							

(Название учреждения образования)

Учебный год	Имя и фамилия учащегося	Состояние учебного пособия при получении	Оценка учащемуся за пользование учебным пособием
20 /			
20 /			
20 /			
20 /			
20 /			
20 /			

Учебное издание

Котов Владимир Михайлович
Лапо Анжелика Ивановна
Быкадоров Юрий Александрович
Войтехович Елена Николаевна

ИНФОРМАТИКА

Учебное пособие для 10 класса
учреждений общего среднего образования с русским языком обучения
(с электронными приложениями)

Зав. редакцией *Г. А. Бабаева*. Редактор *Е. И. Черникова*. Художественный редактор *О. Н. Карпович*. Художник *А. Н. Богушевич*. Техническое редактирование и компьютерная верстка *И. И. Дубровской*. Корректоры *О. С. Козицкая, Е. П. Тхир, А. В. Алешко*.

Подписано в печать 29.06.2020. Формат 70 × 90 $^{1}/_{16}$. Бумага офсетная. Офсетная печать. Усл. печ. л. 8,78. Уч.-изд. л. 8,0 + 20,0 эл. прил. Тираж 121 500 экз. Заказ .

Издательское республиканское унитарное предприятие «Народная асвета» Министерства информации Республики Беларусь. Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий 1/2 от 08.07.2013. Пр. Победителей, 11, 220004, Минск, Республика Беларусь.

Открытое акционерное общество «Полиграфкомбинат им. Я. Коласа». Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 2/3 от 10.09.2018. Ул. Корженевского, 20, 220024, Минск, Республика Беларусь.