Рекомендовано Министерством образования и науки
Республики Казахстан

Г.И. Салгараева
Ж.Б. Базаева
А.С. Маханова

ИНФОРМАТИКА

Учебник для 10 класса естественно-математического направления
общеобразовательной школы

10

басы
арман-пв

УДК 373.167.1
ББК 32.973 я 72
С16

Научный консультант:
Ж.У. Кобдикова – *доктор педагогических наук.*

Салгараева Г.И. и др.
С16 **Информатика:** Учебник для 10 кл. естеств.-математ. направления общеобразоват. шк./Г.И. Салгараева, Ж.Б. Базаева, А.С. Маханова. – Нур-Султан: Издательство «Арман-ПВ», 2019. – 240 стр.

ISBN 978-601-318-245-2

Учебник «Информатика» разработан в соответствии с Типовой учебной программой по предмету «Информатика» для 10 класса уровня общего среднего образования по обновленному содержанию с учетом возрастных особенностей учащихся. Материал учебника изложен доступным языком, содержание включает дополнительные сведения.

УДК 373.167.1
ББК 32.973 я 72

ISBN 978-601-318-245-2

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

Задания для самостоятельного усвоения темы – задания для формирования функциональной грамотности

1 Отвечаем на вопросы

2 Думаем и обсуждаем

3 Анализируем и сравниваем

4 Выполняем в тетради

5 Выполняем на компьютере

6 Делимся мыслями

Вспомните!

Вопросы по пройденному материалу, направленные на изучение новой темы

Вы узнаете:

Ожидаемые результаты освоения материала; учебные цели

Термины

Научные термины

Это интересно!

Дополнительная информация, относящаяся к содержанию темы

Внимание

При необходимости вы всегда сможете найти CD с электронным приложением на сайте *arman-pv.kz* и загрузить его на свой компьютер для дальнейшей работы

Предисловие

Дорогие друзья!

В этом учебном году вы продолжите изучение курса информатики. Учебник состоит из 5 разделов: «Компьютерные сети и информационная безопасность», «Представление данных», «Алгоритмизация и программирование», «Web-программирование» и «Информационные системы».

В разделе «Компьютерные сети и информационная безопасность» описывается назначение компонентов сети, IP-адресов, системы доменных имен и частных виртуальных сетей.

В разделе «Представление данных» объясняется назначение основных логических элементов, дается сравнение таблиц кодировки Unicode и ASCII.

При изучении раздела «Алгоритмизация и программирование» вы будете писать код на языке программирования, используя функции и процедуры, использовать файлы для чтения и записи информации.

В разделе «Web-программирование» вы будете использовать HTML-теги и CSS при разработке web-страниц, применять HTML-теги для вставки объектов мультимедиа на web-страницы.

В разделе «Информационные системы» вы будете оценивать положительные и отрицательные стороны использования Big Data, определять типы данных и создавать базы данных SQL, устанавливать связь с web-страницами.

Рубрика «Это интересно» содержит дополнительные сведения для расширения и углубления знаний. В каждом параграфе предложены задания познавательных уровней «Отвечаем на вопросы», «Думаем и обсуждаем», «Анализируем и сравниваем», «Выполняем в тетради», «Выполняем на компьютере» и «Делимся мыслями». Выполнение заданий поможет вам повторить и закрепить изученный материал. Глоссарий поможет восстановить в памяти те или иные определения.

Учебник дополнен электронным приложением (CD-диском), в котором вы сможете самостоятельно выполнить интерактивные задания, направленные на повторение знаний.

Информатика является наукой, которая применяется во всех сферах жизнедеятельности человека. Желаем вам увлекательного изучения этого предмета и успешного практического применения ваших знаний!

Раздел I

КОМПЬЮТЕРНЫЕ СЕТИ И ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Цели обучения:

- описывать назначение компонентов сети (узлы, маршрутизаторы, коммутаторы);
- объяснять назначение и представление IP-адреса;
- объяснять назначение системы доменных имен (DNS), частной виртуальной сети;
- объяснять назначение частной виртуальной сети
- объяснять значение терминов «информационная безопасность», «конфиденциальность», «целостность» и «доступность»;
- оценивать необходимость шифрования данных;
- объяснять использование мер безопасности данных пользователя: пароли, учетные записи, аутентификация, биометрическая аутентификация.

§ 1. Принципы работы компьютерных сетей. Компоненты сети

Вспомните!

- *Что такое компьютерная сеть?*
- *Какие виды компьютерных сетей вы знаете?*

Вы узнаете:

- *о компьютерных сетях;*
- *о типах компьютерных сетей;*
- *об аппаратных компонентах.*

Термины:

- маршрутизатор;
- коммутатор;
- концентратор.

В настоящее время развитие телекоммуникаций в Казахстане основано на появлении новых высокотехнологичных услуг – передачи данных, сотовой связи и доступа в Интернет.

Сеть – это объединение нескольких устройств между собой для обмена данными. Главной целью объединения компьютеров в сети является предоставление пользователям возможности доступа к различным информационным ресурсам (документам, программам, базам данных и т.д.), распределенным по этим компьютерам, и их совместного использования. Например, чтобы одновременно пользоваться одним принтером, плоттером, факсом и т.д.

Компьютерная сеть – это совокупность компьютеров, объединенных каналами связи и обеспеченных коммуникационным оборудованием и программным обеспечением для совместного использования данных и оборудования. Важной характеристикой любой компьютерной сети является широта территории, которую она охватывает. Широта охвата определятся взаимной удаленностью компьютеров, составляющих сеть и, следовательно, влияет на технологические решения, выбираемые при построении сети.

Классически выделяются два типа сетей: *локальные* и *глобальные* (*рис. 1*).

К **локальным сетям** (Local Area Network, LAN) обычно относят сети, компьютеры которых сосредоточены на относительно небольших территориях (как правило, в радиусе до 1–2 км). Классическим примером локальной сети является сеть одного предприятия, расположенного в одном или нескольких рядом стоящих зданиях.

Глобальные сети (Wide Area Network, WAN) – это сети, предназначенные для объединения отдельных компьютеров и локальных сетей, расположенных на значительном удалении (сотни и тысячи километров) друг от друга.

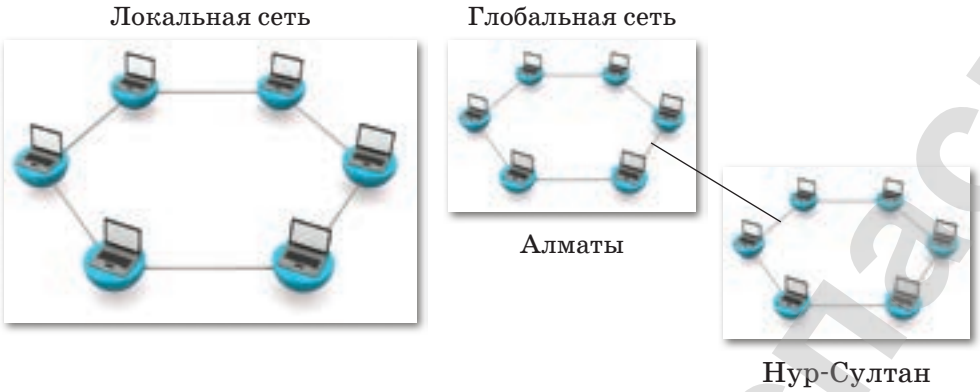


Рис. 1. *Локальная и глобальная сети*

Независимо от используемых на каждом компьютере приложений, все машины сети делятся на два класса – *серверы* и *рабочие станции*. **Сервером** называют компьютер, предоставляющий свои ресурсы (например, диски) другим компьютерам сети, т.е. серверы предоставляют свои ресурсы рабочим станциям. **Рабочая станция,** или **клиент,** использует ресурсы сервера. Рабочие станции имеют доступ к сетевым ресурсам, но своих ресурсов в общее пользование не предоставляют.

Компьютерная сеть состоит из основных аппаратных и программных компонентов, которые должны работать согласованно. Для корректной работы устройств в сети их нужно правильно инсталлировать и установить на них верные рабочие параметры (*рис. 2*).

Рис. 2. *Компоненты и рабочие станции сети*

Основные аппаратными компоненты:

- Абонентские системы (компьютеры (рабочие станции, или клиенты, и серверы), принтеры, сканеры и др.).
- Сетевое оборудование (сетевые адаптеры, концентраторы, мосты, маршрутизаторы и др.).
- Коммуникационные каналы (кабели, разъемы, устройства передачи и приема данных в беспроводных технологиях).

Основные программные компоненты:

- сетевые операционные системы – Windows NT, Windows NT Server, Windows for Workgroups, LANtastic, NetWare, Unix, Linux и др.
- сетевые программные обеспечения – клиент сети, сетевые адаптеры, протоколы, служба дистанционного доступа.

Сетевые компоненты компьютерных сетей являются основными составляющими сети. Каждый из них важен и выполняет разные функции для оптимизации связи между компьютерами сети. Снаружи эти устройства могут выглядеть одинаково: металлические коробочки со множеством соединителей, или портов, куда подсоединяются кабели Ethernet (*рис. 3*).

Рис. 3. *Маршрутизатор, коммутатор, концентратор*

В отличие от коммутаторов и маршрутизаторов, **концентраторы** (hub) – самые дешевые и простые устройства сети. С помощью концентраторов осуществляется обмен данными между компьютерами сети. Все данные, которые поступают в один порт концентратора, пересылаются на все другие порты. Следовательно, все компьютеры, подсоединенные к одному концентратору, «видят» друг друга в сети.

Работа **коммутатора** (switch) во многом схожа с функционированием концентратора, но он делает ее более эффективно. Каждый пакет данных (фрагмент Ethernet), передаваемый

в сети, имеет MAC-адреса источника и адресата. Коммутатор способен «запоминать» адрес каждого компьютера, подключенного к его портам, и действовать как регулировщик – передавать данные только на компьютер адресата и ни на какие другие. Это может оказать существенный положительный эффект на производительность всей сети, потому что не осуществляются ненужные передачи пакетов и освобождается сетевая пропускная способность.

Маршрутизатор (router) – интеллектуальное («умное») устройство, связывающее две или более сети для доставки пакетов. По сравнению с коммутаторами, маршрутизаторы медленны и относительно дорогостоящи. Они выполняют такие функции, как быстрое определение изменений в сети и проверка содержимого сообщений, причем правила доставки могут изменяться в зависимости от содержания сообщений. Эта особенность позволяет маршрутизаторам играть важную роль в сфере сетевой безопасности.

Чтобы объяснить различия между маршрутизатором и коммутатором, проведем аналогию с почтовым сервером корпорации. Когда служащий посылает письмо, оно может быть доставлено его конечному адресату через внутреннюю систему доставки почты компании или через локальное почтовое отделение (если получатель постоянно находится вне компании). Коммутатор здесь представлен почтовым сервером компании, а маршрутизатор – локальным почтовым отделением.

1 Отвечаем на вопросы

1. Как вы понимаете, что такое компьютерная сеть?
2. Назовите виды компьютерных сетей.
3. В каких случаях используются сетевые компоненты?
4. Какие сетевые компоненты используются для создания компьютерных сетей?
5. Чему вы можете научиться при создании компьютерных сетей?

2 Думаем и обсуждаем

1. Для чего необходима компьютерная сеть?
2. Почему возможности глобальных сетей более широки?
3. Определите принципы распределения компьютерных сетей по признакам.

3 Анализируем и сравниваем

1. Чем схожи сетевые компоненты? Назовите различия между ними.
2. Сравните функции основных аппаратных компонентов компьютерной сети и определите их сходства.
3. Чем отличаются локальные и глобальные сети? В какой из них больше возможностей?

4 Выполняем в тетради

1. Заполните таблицу.

Название аппаратного компонента	Функции
Концентратор	
Коммутатор	
Маршрутизатор	

2. Найдите в Интернете определение типов сетевых кабелей и информацию о них. Запишите в тетради.

5 Выполняем на компьютере

Определите, к какой сети относятся компьютеры в вашей школе. Выясните, в каком кабинете находится сервер.

6 Делимся мыслями

Что вы узнали на уроке? Чему научились? Поделитесь мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры.

Какую сеть вы используете часто? Почему?

§ 2. Принципы работы компьютерных сетей. IP–адрес

Вспомните!

- *Что такое компьютерная сеть?*
- *Что такое аппаратные и программные компоненты компьютерной сети?*

Вы узнаете:

- *о понятии «IP-адрес»;*
- *о необходимости IP-адресов.*

Термины:

- IP-адрес;
- провайдер;
- маска.

Для того чтобы отправить обычное письмо по почте, нужно указать точный адрес проживания получателя: область, город или село, улицу, номер дома и квартиры. Так же и IP-адрес – это не что иное, как адрес вашего компьютера в сети. Он действует так же, как и адрес проживания человека, без которого передача информации конкретному лицу будет невозможна.

IP-адрес (англ. Internet Protocol Address) – уникальный сетевой адрес, необходимый для нахождения, получения и передачи информации от одного узла к другому. Под **узлом** понимается любое устройство (мобильный телефон, компьютер, принтер, концентратор, коммутатор, маршрутизатор и т.п.), имеющее доступ к сети. IP-адрес присваивается устройству вне зависимости от величины сети, к которой он подключен – это может быть как глобальный доступ в Интернет, так и локальная сеть, состоящая из нескольких устройств (*рис. 4*).

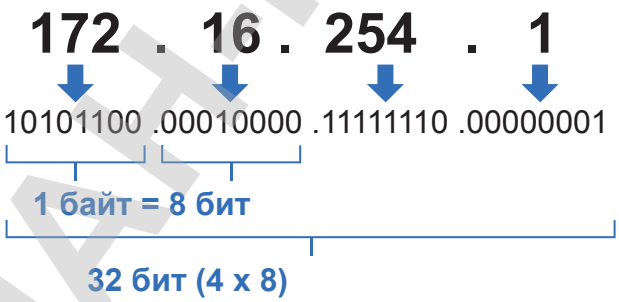


Рис. 4. *Пример IP-адреса*

IPv4-адрес

Октет – каждый разряд IP-адреса, состоящий из 1 байта. IP-адрес может быть представлен в формате IPv4 или IPv6.

IPv4 – интернет-протокол, использующий 32-битные адреса. Четвертая версия протокола была выпущена в 1981 году и стала

наиболее широко используемой. Пример такого IP-адреса: *123.45.67.89*.

Главная проблема этого протокола – ограниченность возможных адресов. Несмотря на то, что их более четырех миллиардов (4 294 967 296), этого не хватает для всех устройств, выходящих в сеть.

В 1996 году была представлена шестая версия IP-протокола, которая должна была решить проблемы предыдущей, четвертой, версии. Длина адреса, используемая в IPv6, составляет уже 128 бит. Пример такого IP-адреса: *21DA:00D3: 0000:2F3B:02AA: 00FF:FE28:9C5A*.

Существует два вида **IP-адресов**:

1. Внутренние IP-адреса (частный, локальный, «серый»).
2. Внешние IP-адреса (публичный, глобальный, «белый»).

Внутренние (частные) IP-адреса не используются в сети Интернет. К внутренним относятся адреса, используемые в локальных сетях. Доступ к внутреннему IP-адресу можно получить лишь в пределах локальной подсети. К частным относятся IP-адреса, значения которых лежат в следующих диапазонах:

- 10.0.0.0 – 10.255.255.255
- 172.16.0.0 – 172.31.255.255
- 192.168.0.0 – 192.168.255.255

IP-адреса присваиваются провайдерами.

Провайдер (от англ. *internet service provider*, сокр. *ISP* – поставщик интернет-услуг) – компания, которая предоставляет возможность доступа к сети Интернет и другие связанные с Интернетом услуги.

Внешние (публичные) IP-адреса используются в сети Интернет. Публичным IP-адресом называется IP-адрес, под которым вас видят устройства в Интернете, и он является уникальным во всей сети Интернет. Доступ к устройству с публичным IP-адресом можно получить из любой точки глобальной сети.

К примеру, IP-адреса компьютеров, присоединенных к сети класса, – внутренние, а IP-адрес сетевого компонента, который дает этим компьютерам доступ в Интернет – внешний, т.е. он присваивается для передачи запросов компьютеров серверу.

IP-адреса также бывают *статическими* и *динамическими*.

Статический (постоянный, неизменный) IP-адрес задается в настройках устройства либо назначается провайдером. Он

не может быть присвоен другому устройству и не меняется с течением времени.

Динамический (непостоянный, изменяемый) также назначается провайдером, однако только на ограниченный срок. Через некоторое время этот адрес заменяется на другой, потом на третий и т.д. Сервис whoer.net позволяет определить, к какому типу (статическому или динамическому) относится ваш IP-адрес. Попробуйте несколько раз зайти на сайт, предварительно сбросив интернет-соединение. Если сервис каждый раз показывает один и тот же IP-адрес, то вы обладатель статического адреса. Если адреса меняются – значит провайдер назначил динамический диапазон.

У компьютеров сети кроме IP-адресов также имеется **маска**, которая необходима для определения границ (диапазона) подсети, т.е. определяет, какая часть IP-адреса сетевого узла относится к адресу самой сети, а какая часть – к адресу узла в этой сети. Таким образом, маска подсети отделяет адрес подсети от адреса конечного устройства, которое находится в этой сети.

Маска подсети занимает 32 бита, но в отличие от IP-адреса, нули и единицы в ней не могут чередоваться. Единицы всегда идут сначала, потом нули.

Например, маска не может иметь вид:

120.22.123.12 = 01111000.00010110.01111011.00001100.

Но может выглядеть так:

255.255.248.0 = 11111111.11111111.11111000.00000000.

Чтобы определить границы подсети, компьютер производит побитовое умножение (логическое И) между IP-адресом и маской, а на выходе получает адрес с обнуленными битами в позициях нулей маски. К примеру, 192.168.11.10/21:

11000000.10101000.00001011.00001010
11111111.11111111.11111000.00000000

11000000.10101000.00001000.00000000 = 192.168.8.0

Адрес 192.168.8.0, полученный в результате логического умножения, называется **адресом подсети**.

По формуле Хартли можно вычислить общее количество информации, которое находится в сообщении общей длиной N.

Решение: Количество информации в интернет-адресе I = 32 бит, тогда N – общее количество интернет-адресов:

$N = 2^i = 2^{32} = 4\ 294\ 967\ 296$.

Вывод: 32-разрядный интернет-адрес позволит подключить более 4 миллиардов компьютеров к Интернету.

1 Отвечаем на вопросы

1. Как конструируется IP-адрес?
2. Какую роль выполняет IP-адрес в компьютерной сети?
3. Как происходит присвоение IP-адреса?
4. Что входит в обязанности провайдера?
5. Что такое маска?

2 Думаем и обсуждаем

1. Для чего нужен IP-адрес?
2. Для чего было необходимо разделить IP-адреса на внутренние и внешние?
3. Если IP-адрес вашего компьютера постоянно меняется при каждом соединении к Интернету, то к какому типу относится ваш IP-адрес?
4. Почему выполняется умножение (логическое И) между IP-адресом и маской для определения границ подсети?
5. Если IP-адрес длиной 32 бита позволяет подключить к глобальной сети более 4 млрд компьютеров, то сколько компьютеров можно будет подключить к сети с помощью IP-адреса длиной в 128 бит?

3 Анализируем и сравниваем

1. В чем разница между протоколами IPv4 и IPv6?
2. Проанализируйте особенности внутренних и внешних IP-адресов.
3. Чем отличается статический IP-адрес от динамического IP-адреса?
4. Чем определяется взаимосвязь между IP-адресом и маской подсети?

4 Выполняем в тетради

Заполните таблицу по типам IP-адресов.

Название	Функции
Внутренние IP-адреса	
Внешние IP-адреса	
Статические IP-адреса	
Динамические IP-адреса	

5 Выполняем на компьютере

1. С помощью сервиса *whoer.net* определите IP-адреса компьютеров, находящихся в классе, и к какому типу они относятся.
2. Воспользовавшись поисковыми системами, найдите другие сервисы определения IP-адресов компьютеров сети.

6 Делимся мыслями

С помощью какого IP-адреса обмениваются информацией устройства, которые вы используете? Какой провайдер вас обслуживает? Какой адрес более эффективен – статический или динамический?

§ 3. Принципы работы компьютерных сетей. Домен. Частная виртуальная сеть

Вспомните!

- *Что такое IP-адрес?*
- *Для чего нужен IP-адрес?*

Вы узнаете:

- *о понятии «домен»;*
- *о частных виртуальных сетях.*

Термины:

- домен;
- протокол;
- виртуальная сеть.

Это интересно!

Британский сайт Cable, который оценивает качество интернет-подключения, составил рейтинг стран по скорости Интернета, проанализировав более 163 миллионов тестов широкополосных подключений в 200 странах мира. Казахстан оказался на 95 месте рейтинга. Средняя скорость Интернета – 4,45 мегабит в секунду. А у трех стран (Сингапур, Швеция, Дания), этот показатель составил от 11 до 15 минут.

Людям нелегко запомнить цифровые адреса, поэтому для удобства пользователей Интернета была введена система доменных имен. В этой системе цифровому интернет-адресу компьютера назначается уникальное доменное имя. С помощью специальной серверной программы устанавливается связь между цифровыми и доменными адресами.

Домен – символьное имя, разделенное точкой. Система доменных имен является иерархической структурой: домены верхнего уровня – домены второго уровня – домены третьего уровня.

Домены верхнего уровня бывают двух типов: **географические** и **административные**. Каждой стране мира присвоен свой географический домен с двухбуквенным кодированием, например: kz – Казахстан, ru – Россия, uk – Великобритания, fr – Франция. Обычно домен определяет страну, в которой находится компьютер, который, следовательно, является частью национальной сети. Например, www.zakon.kz. Национальный домен верхнего уровня для Казахстана.*kz* впервые был зарегистрирован 19 сентября 1994 года.

Административные домены отмечаются тремя или более буквами. Каждая компания использует их для регистрации своих доменов второго уровня. Например, сайт компании Microsoft зарегистрирован в домене административного верхнего уровня, как .com, а Московский открытый образовательный институт

зарегистрировал домен второго уровня metodist в географическом домене верхнего уровня .ru.

Доменное имя интернет-сервера состоит (справа налево) из имени домена верхнего уровня, имени домена второго уровня и имени самого компьютера. Например, имя основного сервера компании Microsoft – www.microsoft.com, а имя сервера института – iit.metodist.ru. Каждый компьютер, подключенный к Интернету имеет интернет-адрес, но может не иметь своего доменного имени. Обычно его не имеют компьютеры, подключенные к Интернету через телефонную линию.

Протокол – стандарт для предоставления, модификации и передачи информации в компьютерной сети.

Другими словами, протокол – это определенный сетевой язык. Когда разные глобальные сети работали автономно, они «разговаривали на разных языках». Чтобы их объединить, было необходимо разработать общий сетевой язык. Таким языком стал протокол TCP/IP. Термин TCP/IP состоит из двух протоколов:

- TCP (Transmission Control Protocol) – транспортный протокол;
- IP (Internet Protocol) – протокол маршрутизации.

На основе протокола TCP/IP были реализованы другие прикладные интернет-протоколы, которые являются основой сервиса сетей.

Этот протокол поддерживает программные и аппаратные устройства сетей. Он стандартизирует следующие процессы:

- разделение данных на пакеты (части);
- адресацию пакетов и их доставку в пункт назначения по указанным маршрутам;
- сбор пакетов в исходный тип данных.

В то же время проверяется правильность приема-передачи пакетов, правильность всех пакетов, отправленных в нужное место.

Таким образом, если вы находитесь в одном учреждении, то у вас есть возможность подключиться к одной сети и выйти в глобальную сеть. Если нужно подключить компьютеры, расположенные на расстоянии друг от друга, к одной сети, управлять работой компьютера с другого компьютера, усилить безопасность или произвести какие-то другие действия, то возникает необходимость создания частной сети.

Частная виртуальная сеть (VPN – Virtual Private Network) – это технология, позволяющая создать защищенную (закрытую от внешнего доступа) связь логической сети поверх частной или публичной при наличии высокоскоростного Интернета (*рис. 5*).

Рис. 5. *Частная виртуальная сеть*

Рассмотрим наиболее часто используемые применения VPN.

- **Доступ в Интернет.** Чаще всего применяется провайдерами городских сетей, но этот способ также весьма распространен и в сетях предприятий. Основным достоинством использования VPN является более высокий уровень безопасности, так как доступ в локальную сеть и Интернет осуществляется через две разные сети, что позволяет задать для них разные уровни безопасности. При классическом решении – раздача Интернета в корпоративной сети – выдержать разные уровни безопасности для локального и интернет-трафика практически не представляется возможным.
- **Доступ в корпоративную сеть извне** (а также объединение сетей филиалов в единую сеть). Это то, для чего и задумывался VPN: организация безопасной работы в единой корпоративной сети для клиентов, находящихся вне предприятия. Широко используется для объединения территориально удаленных друг от друга подразделений, обеспечения доступа

в сеть для сотрудников, находящихся в командировке или на отдыхе, дает возможность работать из дома (*рис. 6*).

Рис. 6. *Удаленный доступ к корпоративной сети*

- **Объединение сегментов корпоративной сети.** Зачастую сеть предприятия состоит из нескольких сегментов с различными уровнями безопасности и доверия. В этом случае для взаимодействия между сегментами можно использовать VPN, ведь это гораздо более безопасное решение, нежели простое объединение сетей. Например, таким образом можно организовать доступ сети складов к отдельным ресурсам сети отдела продаж. Так как это отдельная логическая сеть, для нее можно задать все необходимые требования безопасности, не влияя на работу отдельных сетей.

1 Отвечаем на вопросы

1. Какую роль выполняет домен в компьютерной сети?
2. Для чего нужен протокол?
3. Каково практическое применение Virtual Private Network?
4. Возможности какой технологии можно применить, чтобы дистанционно просмотреть файлы компьютера, находящегося в школе?

2 Думаем и обсуждаем

1. Для чего нужен домен?
2. Почему домены делятся на географические и административные?
3. По какой причине растет актуальность применения частных виртуальных сетей?

4. Почему частные виртуальные сети используют для усиления защиты информации в компьютерных сетях?

3 Анализируем и сравниваем

В чем разница между географическими и административными доменами?

4 Выполняем в тетради

1. Запишите в тетради домены, используемые в Республике Казахстан.
2. Запишите в тетради цели использования частных виртуальных сетей.

5 Выполняем на компьютере

Посетите сайты, использующие домены *edu.kz*, *gov.kz*, *mil.kz*.

6 Делимся мыслями

Что вы узнали на уроке? Чему научились? Поделитесь мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры.

§ 4. Информационная безопасность

Вспомните!

- *Что такое домен и его виды?*
- *Каковы функции системы доменных имен?*
- *Какова цель виртуальной частной сети?*

Вы узнаете:

- *о мерах информационной безопасности;*
- *об организации мер конфиденциальности, целостности и доступности информации.*

Термины:

- информационная безопасность;
- конфиденциальность;
- доступность;
- целостность.

На сегодняшний день все наши действия тесно связаны с компьютерными технологиями, поэтому мы должны строго соблюдать правила информационной безопасности.

Прежде всего, самой важной задачей является защита государственных, военных, юридических и врачебных тайн. Также необходимо защищать личную информацию – данные документов, удостоверяющих личность, логины и пароли на сайтах и т.п.

Мы часто слышим о понятии информационной безопасности, но не до конца понимаем суть этого понятия.

Что такое информационная безопасность?

Информационная безопасность – это защищенность информации от любых действий, в результате которых информация может быть искажена или утеряна, а пользователям информации нанесен недопустимый ущерб.

Информационная безопасность – это процесс, обеспечивающий доступность, целостность и конфиденциальность информации.

В связи с целями и задачами, которые мы ставим перед собой в виртуальном пространстве, для трех названных аспектов информационной безопасности необходимы различные способы защиты информации.

Например, если мы будем использовать виртуальное пространство только для просмотра web-страниц, то для обеспечения защиты информации нужно будет применять антивирусные программы и соблюдать простые правила работы в сети Интернет.

Защита информации – комплекс мер, направленных на обеспечение информационной безопасности.

На практике защита информации объясняется как поддержка целостности, доступности и сохранности информации, которая используется для ввода, сохранения, редактирования и передачи данных.

В результате защиты информации должны быть обеспечены следующие условия (*схема 1*):

Схема 1. *Результат защиты информации*

Доступность информации нарушается, когда компьютер выходит из строя или web-сайт не отвечает на запросы пользователей в результате массовой атаки вредоносных программ через Интернет.

Нарушение целостности информации – это кража или искажение информации. Например, изменение содержания писем электронной почты и других цифровых документов.

Конфиденциальность информации нарушается, когда информация становится известной тем людям, которые не должны ее знать, что может повлечь за собой распространение секретной информации.

Основные угрозы доступности информации

К ним можно отнести внутренний отказ информационной системы и выход поддерживающей инфраструктуры из строя.

К внутренним отказам информационной системы относятся:

- нарушение правил передачи (случайно или намеренно);
- выход системы из строя (чрезмерное количество запросов, редактируемой информации и т.д.);
- вредоносное программное обеспечение;
- выход аппаратного и программного обеспечения из строя;
- повреждение информации.

К угрозам выхода из строя поддерживающей инфраструктуры относятся:

- нарушения в работе систем связи, электропроводки или циркуляции воздуха;
- прекращение работы системы.

Основные угрозы целостности информации

Есть два вида угроз целостности информации: **статические** и **динамические**. К угрозам статической целостности информации относятся введение неправильной информации и изменение информации.

К угрозам динамической целостности информации относятся повтор данных, введение дополнительной информации и кража информации.

Основные угрозы конфиденциальности информации

Конфиденциальность информации делится на предметную и служебную. Служебная информация не относится к определенной предметной сфере (например, пароли пользователей). Самая главная угроза, от которой сложно защититься, – это превышение должностных полномочий. Например, системный администратор может открыть и прочитать любой файл и получить доступ к любой информации, зайти на электронную почту любого пользователя и т.д.

Для обеспечения безопасности информации необходимо знать, что защищать, от кого защищать, как защищать, какие методы защиты использовать и какие меры предпринимать.

Безопасность информации в компьютерных сетях ниже, чем в самих компьютерах, потому что:

- в Сети работают очень много пользователей и их состав постоянно меняется;
- есть возможность незаконного подключения к Сети;
- существует уязвимое сетевое программное обеспечение;
- высока вероятность атаки вредоносных программ через Сеть.

В Казахстане вопросы о защите информации регулирует законодательство Республики Казахстан в сфере информационной безопасности.

Самой слабой стороной любой системы защиты является человек. Некоторые пользователи записывают свои пароли

в доступных местах и даже могут передать их другим. В этом случае увеличивается риск незаконного доступа к любой информации. Поэтому обучение пользователей правилам информационной безопасности является очень важным фактором ее обеспечения.

1 Отвечаем на вопросы

1. Что такое информационная безопасность?
2. Что делать при возникновении необходимости защиты информации?
3. В каких случаях применяются меры защиты информации?
4. Какие действия необходимо выполнить для обеспечения целостности информации?
5. Как эффективно обеспечить доступность информации?
6. Какие действия выполняются для сохранения конфиденциальности информации?

2 Думаем и обсуждаем

1. В чем важность сохранения конфиденциальности и целостности информации?
2. Почему безопасность информации в компьютерных сетях ниже, чем в самих компьютерах?

3 Анализируем и сравниваем

Выделите особенности целостности, конфиденциальности и доступности информации.

Целостность информации	Конфиденциальность информации	Доступность информации

4 Выполняем в тетради

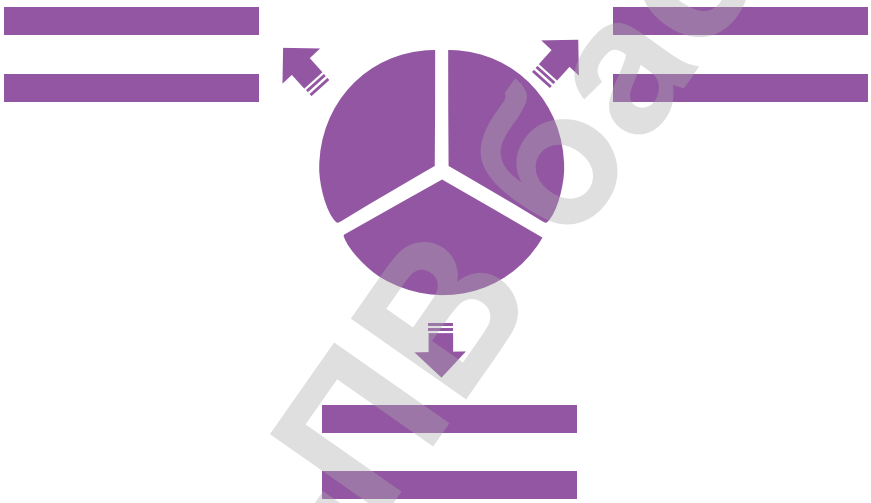
Запишите угрозы для целостности, конфиденциальности и доступности информации в таблице.

Угрозы	Виды	Пример
Угрозы конфиденциальности информации		

Угрозы	Виды	Пример
Угрозы целостности информации		
Угрозы доступности информации		

5 Выполняем на компьютере

Запишите основные понятия информационной безопасности и классификацию ее угроз, используя текстовый редактор (SmartArt). Пример схемы:



6 Делимся мыслями

Объясните основные понятия информационной безопасности и ответьте на следующие вопросы:

1. Как вы думаете, как можно заранее предотвратить угрозы информационной безопасности? Объясните свои ответы.
2. Как обеспечивается информационная безопасность на ваших компьютерах?

§ 5–6. Методы защиты информации

Вспомните!

- *Что такое защита информации?*
- *Что такое доступность информации?*
- *Что такое конфиденциальность информации?*
- *Что такое целостность информации?*

Вы узнаете:

- *о мерах безопасности;*
- *о резервном копировании данных;*
- *о шифровании данных.*

Обеспечение и поддержка информационной безопасности включают комплекс технических, программных и организационных мер.

К **техническим мерам** защиты информации можно отнести системы видеонаблюдения и сигнализации, а также другие средства предотвращения и блокировки всех возможных способов распространения информации.

Программные меры защиты информации обеспечивают возможность установки паролей на доступ к определенным данным, шифрование текста, временное удаление файлов и защиту от вредоносных программ.

К **организационным мерам** защиты информации относятся политика безопасности организаций и такое расположение каналов связи, которое затруднило бы доступ к ним.

Главная опасность, которую могут нанести вредоносные программы – уничтожение информации или уничтожение ключа доступа к секретной информации. Чтобы это предотвратить, необходимо создать **резервные копии** важной информации. Если не предпринять мер резервного копирования, то можно потерять важные файлы без возможности их восстановления.

Создать резервную копию несложно: для этого есть множество способов. Рассмотрим, резервные копии каких файлов необходимо создавать в первую очередь. Это, конечно же, личные файлы. Если возникнет проблема, операционную систему всегда можно переустановить, но личные файлы восстановить невозможно. Поэтому необходимо часто создавать резервные копии документов, фотографий, видеозаписей и другой личной информации, хранящейся на компьютере, с помощью специальных программ. Также можно сделать резервные копии операционной системы, установленных программ и системных настроек. Есть множество методов создания резервных копий информации, начиная с использования внешних накопителей и заканчивая удаленными серверами. Каждый из них имеет свои преимущества и недостатки.

Создание резервных копий на внешнем накопителе

С помощью внешнего накопителя USB можно создать резервную копию сразу на этом устройстве, используя комбинированные функции резервного копирования. Для Windows 8 и 10 применяется функция **История файлов** (File History). В ОС Windows 7 используется **Резервное копирование Windows**, а в Mac-устройствах – функция **Time Machine**. Для этого нужно активировать функцию резервного копирования и часто подключать внешний накопитель к компьютеру.

Преимущества: большая скорость.

Недостатки: если внешний накопитель потеряется либо повредится, то уничтожится и вся скопированная информация.

Создание резервных копий с помощью сервисов Интернета

Если вы хотите быть уверены в безопасности своих файлов, то можете сделать резервное копирование с помощью таких серверов, как Backblaze, Carbonite и Mozy. Эти программы автоматически создают копии файлов в фоновом режиме. Если ваши файлы исчезнут, их можно восстановить в любое время.

Преимущества: резервное копирование, осуществляемое в онлайн-режиме, защищает от любой опасности, которая может произойти с файлами.

Недостатки: услуги этих серверов платные. Если копируется большой объем данных, первичное резервное копирование занимает больше времени по сравнению с внешними накопителями.

Резервное копирование данных с помощью облачных сервисов

Большинство людей считают, что с технической точки зрения облачные технологии не могут выполнять функцию резервного копирования. Однако использование таких сервисов, как Dropbox, Google Диск, Microsoft OneDrive, эффективнее использования внешних накопителей. Если возникнет какая-либо проблема, копии файлов сохранятся в облачном хранилище.

Преимущества: это простая, быстрая и в большинстве случаев бесплатная услуга. Онлайн-хранение личных файлов защитит их от любой опасности.

Недостатки: многие облачные сервисы предлагают бесплатные хранилища объемом всего в несколько гигабайт, а увеличить их можно только за дополнительную плату.

Для того чтобы не потерять свои файлы, необходимо постоянно осуществлять резервное копирование в облачные сервисы или на внешние накопители.

Чтобы вредоносные программы не повредили информацию, нельзя открывать подозрительные письма, присланные по электронной почте, особенно с вложенными файлами. Также опасно переходить по внешним ссылкам, расположенным в тексте письма, так как они могут вести на зараженные вирусами web-страницы.

Еще одна мера защиты информации – **шифрование**, то есть специальное кодирование.

Шифрование применяют при перемещении секретной информации через незащищенные каналы связи. Шифровать можно тексты, фотографии, аудиофайлы, базы данных и любую другую информацию. Методами шифрования и расшифровки данных занимается наука с 4-тысячелетней историей – криптография. Она состоит из двух разделов: криптографии и криптоанализа.

Криптография – наука о методах шифрования информации.

Криптоанализ – наука о методах и способах расшифровки зашифрованной информации.

Обычно алгоритм шифрования бывает известен всем, а ключ для его расшифровки – нет. Это указывает на основное отличие шифрования от кодирования.

Ключ – параметр алгоритма шифрования. Зная ключ, можно скрыть и открыть сообщение. Все системы шифрования делятся на две группы: симметричные и асимметричные. То, что шифр симметричный, означает, что при зашифровке и расшифровке сообщения используется один и тот же ключ.

При асимметричном шифровании два ключа связаны между собой. Открытый ключ доступен для всех, кто хочет отправить вам сообщение. Второй ключ – закрытый – хранится в секрете и известен только вам.

Криптографическая стойкость шифра – способность криптографического алгоритма противостоять попыткам его расшифровать.

Стойкость алгоритма – стойким считается алгоритм, требующий проделать множество вычислений большого объема для

его расшифровки, при этом после расшифровки скрытая информация теряет свою актуальность.

В современных программах есть возможность шифрования информации при помощи кодового слова. Например, пакеты OpenOffice.org и Microsoft Office позволяют зашифровать все созданные документы, то есть для просмотра и изменения этих документов нужно будет ввести кодовое слово. При архивировании информации в архиваторах  WinRAR и  WinZip предлагается услуга установки кодового слова для разархивации файлов. Программа  GnuPG (gnupg.org) относится к открытому программному обеспечению, в ней используются симметричные и асимметричные шифры, а также различные алгоритмы электронной цифровой подписи.

1 Отвечаем на вопросы

1. Как применяются меры информационной безопасности?
2. Что такое резервное копирование?
3. Где и как используется шифрование?
4. Какова роль криптографии в различных повседневных ситуациях?
5. Как выполняется криптоанализ?
6. Насколько важно использовать ключ шифрования?

2 Думаем и обсуждаем

1. Для чего необходимо соблюдать меры информационной безопасности?
2. Почему важно сохранять резервные копии любой ценной информации?
3. Насколько важна стойкость алгоритма шифрования?

3 Анализируем и сравниваем

Проанализируйте методы резервного копирования.

Методы резервного копирования	Преимущества	Недостатки
Резервное копирование на внешних накопителях		
Резервное копирование с помощью Интернета		
Резервное копирование с применением облачных технологий		

4 Выполняем в тетради

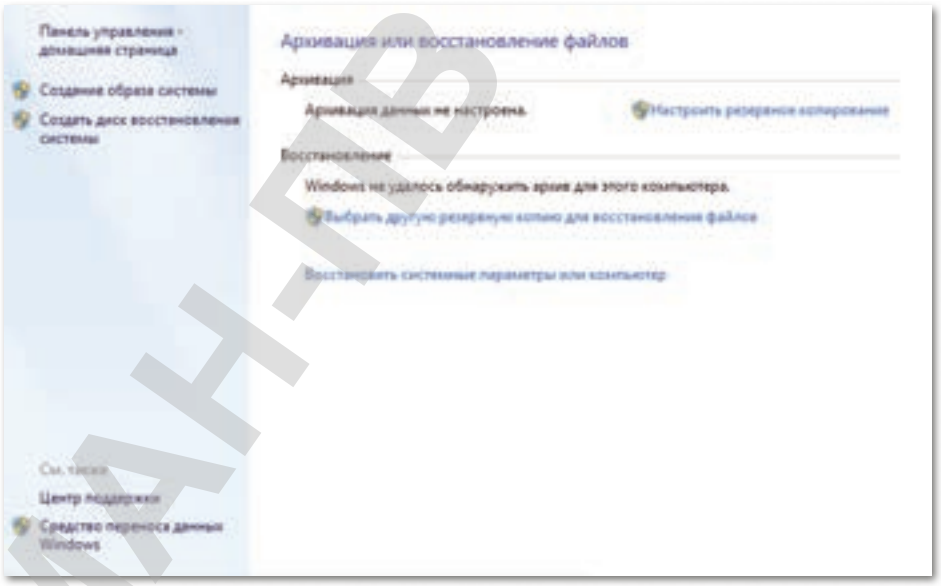
Заполните таблицу. Опишите характеристики мер безопасности для защиты информации.

Меры безопасности для защиты информации	Характеристики
Технические	
Программные	
Организационные	

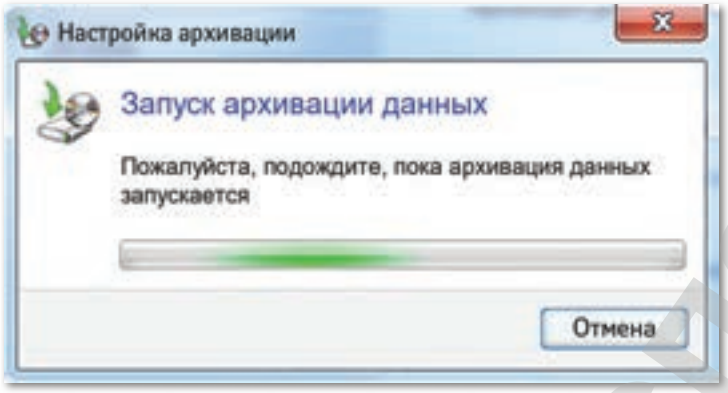
5 Выполняем на компьютере

Используя комбинированный инструмент **Архивирование и восстановление**, выполните задание.

1. Выполнить команды **Пуск** ⇒ **Все программы** ⇒ **Сервис** ⇒ **Архивирование и восстановление.**

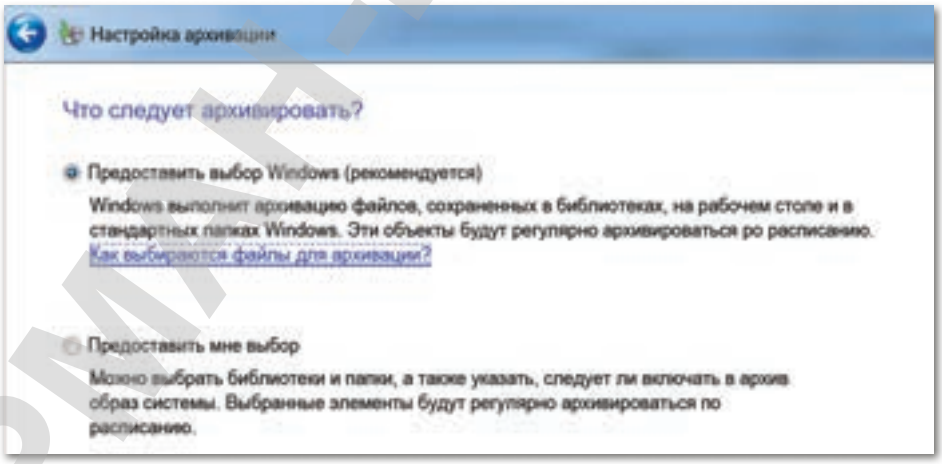


2. Настроить сервис резервирования. Для этого нажмите на ссылку  . Это действие включит мастер **backup.**

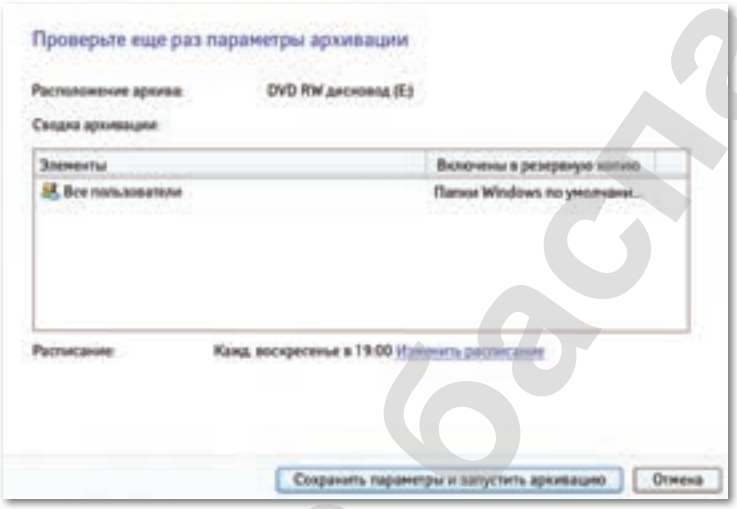


3. Выбрать месторасположение будущей копии:
   - доступные тома;
   - DVD диск;
   - сетевое расположение и т.д.

4. Предоставить операционной системе Windows выбрать, какие документы необходимо архивировать.



5. В следующем окне укажите, доверите вы ли системному инструменту архивирование объектов или выберите сами.
6. Еще раз проверьте параметры архивирования.



7. Для архивации потребуется некоторое время.

8. Задание выполнено. Резервное копирование документов завершено.

6 Делимся мыслями

Какие меры вы принимаете для защиты информации? Почему?

§ 7–8. Методы идентификации физического лица

Вспомните!

- *Какие меры безопасности вы знаете?*
- *Что такое резервное копирование данных?*
- *Что такое шифрование данных?*

Вы узнаете:

- *что такое идентификация и аутентификация;*
- *о применении методов идентификации.*

Пользователь средств информационных технологий вводит определенные сведения для того, чтобы получить доступ к какому-либо ресурсу или сервису. К примеру, при вводе логина и пароля на сайты и т.п.

Существует 2 ступени входа в систему и ввода личных данных: *идентификация* и *аутентификация*.

Идентификация – это ввод личных данных пользователя, известных только ему.

Аутентификация – прием и проверка сервером личных данных пользователя.

Иногда вместо указанного способа используется и простое регистрационное слово.

Процесс регистрации прост. Форму регистрации можно просмотреть в любой социальной сети.

- **Регистрация** – пользователь вводит адрес электронной почты, номер телефона и пароль. Эти данные не должны повторяться в системе, поэтому для одного лица регистрация более одного аккаунта не допускается.
- **Идентификация** – ввод данных, указанных при регистрации, в данном случае это электронная почта или номер телефона.
- **Аутентификация** – после нажатия на кнопку «Вход» страница связывается с сервером, идет проверка наличия и правильности введенных логина или пароля. Если все верно, откроется страница социальной сети.

Методы идентификации

Существует несколько видов идентификации пользователя, которые отличаются друг от друга уровнем защиты и областью применения.

Защита с помощью пароля. Пользователь знает ключ, или пароль, который известен только ему. Сюда можно отнести и идентификацию через смс-уведомления. При вводе имени и пароля пользователя сервер сравнивает введенные данные

с сохраненными данными. В случае полной идентичности введенных данных появляется возможность войти в систему.

Различают два вида паролей: *динамические* и *постоянные*. **Постоянные** пароли изменяются только по требованию пользователя, а **динамические** – по определенным параметрам. Например, если пользователь забудет пароль, сервер может предложить ему динамический пароль для входа в систему.

В работе некоторых фирм или организаций используется **метод проверки с помощью специальных предметов:** карточек, специальных браслетов, флеш-накопителей. При взаимодействии этих устройств с системой сервер проверяет их и либо пропускает, либо останавливает пользователя.

Биометрическая проверка включает в себя методы сканирования отпечатков пальцев, радужной оболочки глаза, формы лица и др. Современные средства могут различать даже мимику лица человека. Это одна из самых надежных, но дорогих систем безопасности.

Использование конфиденциальной информации. Этот способ чаще всего применяется для защиты программного обеспечения. При его использовании проверяется кэш браузера, установленного на персональном компьютере, места расположения и другие параметры.

Знание о понятиях регистрации, идентификации и аутентификации дает возможность правильно применять их по назначению. А это, в свою очередь, способствует сохранению безопасности всех интернет-пользователей.

1 Отвечаем на вопросы

1. Как осуществляется идентификация?
2. В каких случаях выполняется аутентификация?
3. Какими методами осуществляется идентификация?
4. Как выполняется идентификация с помощью пароля?
5. Приведите пример идентификации личности с использованием специальных предметов.

6. Насколько эффективна биометрическая проверка?
7. Как осуществляется метод использования секретной информации?

2 Думаем и обсуждаем

1. Для чего важна идентификация физического лица?
2. Насколько эффективно применение методов идентификации?

3 Анализируем и сравниваем

Рассмотрим простой пример.

Сотрудник, только что устроившийся на работу, говорит охраннику, что работает в этой организации менеджером.

По инструкции охранник не верит словесным убеждениям и требует от менеджера подтверждающий документ, чтобы впустить его в здание.

Новый сотрудник показывает соответствующий документ, двери организации открываются, и охранник пропускает сотрудника внутрь.

Проанализируйте, какая часть этого примера является идентификацией, какая – аутентификацией, а какая – регистрацией.

4 Выполняем в тетради

Укажите характеристики постоянных и динамических паролей в таблице.

Виды паролей	Характеристика
Динамический пароль	
Постоянный пароль	

5 Выполняем на компьютере

Создайте учетную запись пользователя.

1. Нажмите на кнопку **Пуск**, выберите **Панель управления**. В открывшемся окне из раздела **Учетные записи и семейная безопасность** выберите **Учетные записи пользователей**.

2. Выберите строку **Добавить или Удалить учетную запись пользователя**  .
3. Из открывшегося окна выберите раздел **Создать новую учетную запись.**
4. После введения имени и типа учетной записи пользователя нажмите на кнопку **Создать учетную запись**.
5. Для изменения учетной записи щелкните по ней дважды.
6. В открывшемся окне можно изменить имя учетной записи, установить пароль, изменить фото, удалить учетную запись. Выберите раздел **Установить пароль.**
7. Напишите пароль в поле **Введение нового пароля,** введите данный пароль **повторно** в следующей строке.
8. Для того чтобы вспомнить пароль в случае, если забудете его, введите нужное слово или число в поле ключевого слова. Нажмите на кнопку **Создание пароля.**
9. Перезагрузите компьютер и проверьте работу: введите пароль, чтобы войти в созданную учетную запись.

6 Делимся мыслями

Какие методы идентификации могут выйти из эксплуатации в будущем в связи с развитием информационных технологий?

Раздел II

ПРЕДСТАВЛЕНИЕ ДАННЫХ

Цели обучения:

- переводить целые числа из десятичной системы счисления в двоичную, восьмеричную, шестнадцатеричную и обратно;
- использовать логические операции (дизъюнкция, конъюнкция, инверсия);
- строить таблицы истинности для заданного логического выражения;
- объяснять назначение основных логических элементов: конъюнктор, дизъюнктор, инвертор;
- преобразовывать логические выражения в логические схемы и наоборот;
- описывать функции Устройства управления, Арифметико-логического устройства и регистров памяти как отдельных частей процессора;
- сравнивать таблицы кодировки символов Unicode и ASCII.

§ 9–10. Перевод чисел из одной системы счисления в другую

Вспомните!

- *Что такое идентификация?*
- *Для чего нужны пароль и регистрация?*
- *Что такое аутентификация и ее виды?*

Вы узнаете:

- *о системе счисления;*
- *о видах систем счисления;*
- *об особенностях двоичной системы счисления;*
- *о переводе чисел из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную.*

Термины:

- позиционные системы счисления;
- непозиционные системы счисления.

В 1666 году В. Лейбниц предложил идею записи любого числа в двоичной системе и возможность использовать двоичную систему в вычислительном устройстве.

Все виды информации, хранящиеся в памяти компьютера (слова, числа, рисунки, программы управления работой компьютера), записываются в виде последовательности двоичных чисел. В вычислительной технике двоичные числовые символы 0 и 1 называются специальным термином – **бит**, который является единицей измерения информации.

Система счисления – это совокупность правил записи чисел и арифметических операций над ними. Системы счисления делятся на *позиционные* и *непозиционные*.

В **непозиционной системе счисления** цифры не зависят от значения (мощности) числа, от положения (позиции). В качестве примера можно привести римские цифры, которые пишутся с использованием латинского алфавита: CCLXVII (100+100+50+10+7). Здесь C – число сто, где бы оно не стояло, L – пятьдесят и т.д.

В **позиционной системе счисления** значение каждой цифры зависит от ее положения, например, в числе 777,7 первая цифра 7 – сотни, вторая – десятки, третья – единицы, последняя цифра 7 показывает 7/10 часть числа.

Системы счисления бывают четырех видов:

1) десятичная система счисления;
2) двоичная система счисления;
3) восьмеричная система счисления;
4) шестнадцатеричная система счисления.

В **десятичной системе счисления** для выражения чисел используются арабские цифры от 0 до 9: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Например: 234 = 200 + 30 + 4, 2 из разряда сотен, 3 из разряда

десятков, 4 из разряда единиц. Десятичная система счисления является позиционной, так как в записи десятичного числа значение цифры зависит от его позиции или от его места в числе. Позицию, выделенную на цифру числа, называют *разрядом*.

Если записать число 234 в виде суммы, то получим следующее: $2*10^2 + 3*10^1 + 4*10^0$. Число 10 в этой записи является *основанием* системы счисления. Для каждой цифры числа 10 в зависимости от позиции цифры основания возводится в степень и умножается на эту цифру. Для единиц – 0; для десятичных знаков – 1, для сотен – 2 и так далее

Двоичная система счисления – позиционная система счисления с основанием числа 2. В этой системе счисления используются только два знака – 0 и 1. Каждая единица следующего разряда будет в 2 раза больше предыдущей, следовательно, эти единицы образуют последовательность чисел 2, 4, 8, 16, ..., 2n, ...

Восьмеричная система счисления – позиционная целочисленная система счисления с основанием 8. Для представления чисел в ней используются цифры от 0 до 7. Например, число 356 в восьмеричной системе записывается так: $356=3*8^2+5*8^1+6*8^0$.

Шестнадцатеричная система счисления – позиционная система счисления по целочисленному основанию 16. В шестнадцатеричной системе счисления используются цифры от 0 до 9 и шесть первых латинских букв – A (10), B (11), C (12), D (13), E (14), F (15).

При переводе двоичного числа в шестнадцатеричное первое разбивается на группы по четыре разряда, начиная с конца. В случае, если количество разрядов не делится нацело, то к первой четверке впереди дописываются нули. Каждой четверке соответствует цифра шестнадцатеричной системы счисления.

Для того, чтобы перевести число из десятичной системы счисления в любую другую, нужно выполнять целочисленное деление исходного числа на основание той системы счисления, в которую нужно перевести число. При этом важен остаток от деления и частное. Частное нужно делить на основание до тех пор, пока не останется 0. После этого все остатки нужно выписать в обратном порядке – это и будет число в новой системе счисления.

Например:

$37_{10} \to ?_2$

37	2
36	18
1	18	2
	0	8	2
		1	4	2
			4	2	2
			0	2	1
				0	

$37_{10} \to 100101_2$

$103_{10} \to ?_8$

103	8
96	12	8
7	8	1
	4	

$103_{10} \to 147_8$

$419_{10} \to ?_{16}$

419	16
416	26	16
3	16	1
	10	

$419_{10} \to 1A3_{16}$

Для перевода числа из любой системы счисления в десятичную необходимо записать его в виде многочлена, состоящего из произведений цифр числа и соответствующей его степени, и вычислить по правилам десятичной арифметики:

$10110_2 \to ?_{10}$

$10110_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 1 = 16 + 4 + 2 = 22$

$10110_2 \to 22_{10}$

$721_8 \to ?_{10}$

$721_8 = 7 \cdot 8^2 + 2 \cdot 8^1 + 1 \cdot 1 = 448 + 16 + 1 = 465$

$721_8 \to 465_{10}$

$3FA_{16} \to ?_{10}$

F ↓ A ↓

$3FA_{16} = 3 \cdot 16^2 + 15 \cdot 16^1 + 10 \cdot 1 = 768 + 240 + 10 = 1018$

$3FA_{16} \to 1018_{10}$

Числа также можно перевести с помощью таблицы соответствия систем счисления (*таблица 1*).

Таблица 1. *Таблица соответствия систем счисления*

Систем счисления			
Десятичная	**Двоичная**	**Восьмеричная**	**Шестнадцатеричная**
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4

Систем счисления			
Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

В таблицы видно, что число 1011_2 в двоичной системе счисления равно числу 11_{10} в десятичной системе счисления.

1 Отвечаем на вопросы

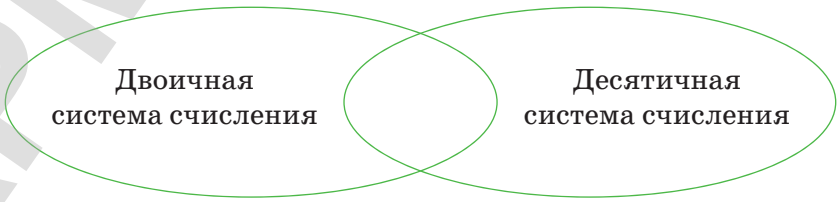
1. В какой сфере применяется систем счисления?
2. Как можно перевести чисела из десятичной системы счисления в любую другую?
3. Как можно перевести чисела из двоичной системы счисления в любую другую?

2 Думаем и обсуждаем

1. Почему персональный компьютер работает с кодом, написанным в двоичной системе счисления?
2. Почему в вычислительной технике используются только цифры 0 и 1?
3. Любое ли число можно перевести в двоичную систему?

3 Анализируем и сравниваем

Сравните двоичную и десятичную системы счисления и определите их различия.



4 Выполняем в тетради

Переведите из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную систему счисления.

$12_{10}=$
$57_{10}=$
$642_{10}=$
$841_{10}=$
$123_{10}=$
$456_{10}=$

5 Выполняем на компьютере

Выполните задания последовательно.

1. Откройте **Пуск** ⇒ **Программы** ⇒ **Стандартные** ⇒ **Калькулятор.**
2. Измените вид калькулятора. Для этого в стандартном меню **Вид** выберите калькулятор **Программист.**
3. Этот вид калькулятора работает в четырех системах счисления: **десятичной, восьмеричной, двоичной и шестнадцатеричной.** Выберите нужную, нажав на одну из четырех кнопок:
   Hexadecimal (шестнадцатеричная),
   Decimal (десятичная),
   Octa (восьмеричная),
   Binary (двоичная).
   Если ввести число в десятичной системе, затем нажать одну из оставшихся трех кнопок, то десятичное число в поле индикации будет автоматически преобразовано в новую систему.
4. Используя эти данные, проверьте правильность решенных ранее задач.

6 Делимся мыслями

Что вы изучили на уроке? Чему научились? Поделитесь своим мнением с друзьями.

§ 11–12. Логические операции (дизъюнкция, конъюнкция, инверсия). Построение таблиц истинности

Вспомните!

- *Что такое система счисления?*
- *Какие виды систем счисления вы знаете?*
- *Какие правила перевода из десятичной системы счисления в двоичную вы знаете?*

Вы узнаете:

- *что такое логика;*
- *о видах логических операций;*
- *что такое таблица истинности логических операций;*
- *о построении таблицы истинности.*

Термины:

- конъюнкция;
- дизъюнкция;
- инверсия.

Логика – это наука о видах и законах человеческого мышления, в том числе о закономерностях высказываний, которые можно доказать. Как научные дисциплины сформированы формальная, математическая, вероятностная и др. виды логики.

Формальная логика – это логика, связанная с анализом нашего содержательного мнения, которое выражается языком речи.

Вероятностная логика – логика, создаваемая случайными параметрами, основанная на использовании нескольких серий испытаний.

Математическая логика является частью формальной логики, имеет четко определенные формы и высказывания. Она изучают только те мысли, которые можно решить их истинность или ложь.

Высказывание – какое-либо предположение, которое может быть истиной или ложью. Например, высказывания «Нур-Султан – столица Республики Казахстан» и «2 * 3 = 6» – истинные, а такие, как «гора плоская», «2 * 2 = 5» – ложные.

В математике логические союзы являются логическими операциями, описывающими сложные утверждения.

Для работы с логическими утверждениями им присваивают имя.

Это интересно!

Алгебра логики впервые появилась в XIX веке в работах английского математика Джорджа Буля. Это был результат стремления к решению традиционных логических задач алгебраическими методами.

Утверждение «Летом Алуа отправится на море» обозначим через A, а через B обозначим утверждение «Летом Алуа отправится в горы».

Тогда составное утверждение A и B можно записать таким образом «Летом Алуа отправится и на море, и в горы».

Здесь «и» – логический союз, A, B – логические переменные, они принимают два значения: «истина» или «ложь», соответственно обозначаются через «1» или «0».

В математической логике есть такие логические операции И, ИЛИ, НЕ и они определяются таблицей истинности.

Таблица истинности – это представление логических операций в виде таблицы, в которой все возможные последовательности значений истинности встроенных операнд перечислены вместе с фактическим значением результата операции для каждой из этих последовательностей.

Логическое умножение (конъюнкция)

Соединение двух простых высказываний A и B в одно составное с помощью союза И называется **логическим умножением**, или **конъюнкцией**, а результат операции – **логическим произведением**. Операция И отмечается знаком «$\wedge$», «$\cdot$» или «&».

Таблица истинности логической операции И.

A	B	A&B
1	1	1
0	1	0
1	0	0
0	0	0

Здесь A и B – два высказывания, принимающие значения *да* или *нет*. Если оба высказывания истинны, то конъюнкция высказываний A и B истинна. Если одно из высказываний A и B ложно или оба высказывания ложны, то конъюнкция A и B ложна.

Логическое сложение (дизъюнкция)

Объединение двух простых утверждений A и B в одно составное утверждение с помощью союза ИЛИ называется **логическим сложением**, или **дизъюнкцией**, а результат

операции – **логической суммой**. Операция ИЛИ отмечается знаком «|», «v» или «+».

Таблица истинности логической операции ИЛИ.

A	B	AvB
1	1	1
0	1	1
1	0	1
0	0	0

Если одно из высказываний A и B истинно, то дизъюнкция A и B будет истинной. Если же оба высказывания A и B ложны, то дизъюнкция A и B ложна.

Логическое отрицание (инверсия)

Присвоение союза НЕ простому утверждению А называется **логическим отрицанием**, или **инверсией**, в результате выполнения этой операции появляется новое утверждение. Операция НЕ обозначается чертой над утверждением $\overline{A}$ или знаком «¬».

Таблица истинности логической операции НЕ.

A	$\overline{B}$
0	1
1	0

Если исходное высказывание ложно, тогда отрицание является истинным, и наоборот, если исходное высказывание истинно, то его отрицание ложно.

Пример 1. Построим таблицу истинности выражения для $A \cdot (\overline{B})$.

A	B	$\overline{B}$	$A \cdot (\overline{B})$
1	1	0	0
1	0	1	1
0	1	0	0
0	0	1	0

Пример 2. Построим таблицу истинности выражения для $(\overline{A}) \cdot (\overline{B}) \cdot (\overline{C})$.

A	B	C	$(\overline{A})$	$(\overline{B})$	$(\overline{C})$	$(\overline{A})\cdot(\overline{B})$	$(\overline{A})\cdot(\overline{B})\cdot(\overline{C})$
1	1	1	0	0	0	0	0
1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0
0	0	1	1	1	0	1	0
1	1	0	0	0	1	0	0
1	0	0	0	1	1	0	0
0	1	0	1	0	1	0	0
0	0	0	1	1	1	1	1

1 Отвечаем на вопросы

1. Что такое логика?
2. Какова роль логики в повседневной жизни?
3. В чем разница между вероятностной и формальной логикой?
4. Зачем используются логические операции?
5. Каков порядок выполнения логических операций?
6. Для чего используются высказывания?

2 Думаем и обсуждаем

1. От чего зависит формальная логика?
2. Для чего нужна вероятностная логика?
3. Какие существуют связи между математической и формальной логикой?
4. Для чего в информатике нужны логические операции И, ИЛИ, НЕ?

3 Анализируем и сравниваем

1. Сравните логические операции И, ИЛИ, НЕ, сделайте выводы.
2. Установите соответствие между терминами и их определениями.

Логика	всегда считает, что все составляющие высказываний истинны
Высказывание	используется для формулировки отрицания
И	наука о видах и законах человеческого мышления, в том числе, о законах доказанных высказываний
НЕ	основана на применении нескольких серий экспериментов со случайными параметрами
Вероятностная логика	какое-либо предположение, которое может быть истиной или ложью

4 Выполняем в тетради

Постройте таблицы истинности для следующих логических выражений.

1. $F(x1, x2, x3) = x3 \vee (\overline{x}2 \,\&\, x1 \,\&\, x3)$
2. $F(x1, x2, x3) = \overline{x}1 \,\&\, \overline{x}2 \vee x2 \vee x1 \,\&\, x3$
3. $F(x1, x2, x3) = \overline{x}1 \,\&\, x2 \,\&\, x3 \vee \overline{x}1 \vee x2 \vee x3$

5 Выполняем на компьютере

Изобразите таблицу истинности логической функции $F = (A \vee B) \,\&\, (\overline{A} \vee B)$ с помощью текстового или табличного редактора.

6 Делимся мыслями

1. Как вы думаете, используем ли мы логические операции в повседневной жизни? Проведите дискуссию.
2. Является ли данная тема одной из самых актуальных тем в науке информатики?

§ 13–14. Практикум. Логические операции

Цель практической работы:

1) закрепить знания об основных логических операциях и таблицах истинности логических выражений;
2) сформировать навыки построения таблиц истинности с использованием электронных таблиц MS Excel.

На прошлом уроке мы рассмотрели наиболее часто используемые логические операции. Однако существуют также такие логические операции, как импликация (следование) и эквивалентность (тождество).

Рассмотрим каждую из них подробно. Используем таблицы истинности для их описания.

Логическая операция/ соответствие в русском языке	Обозначение	Таблица истинности		
Импликация (следование)/ «если…,то…», «когда…, тогда…»	$\rightarrow$	A	B	A→B
		0	0	1
		0	1	1
		1	0	0
		1	1	1
Эквивалентность (тождество) «тогда и только тогда, когда»	$\leftrightarrow$, $\equiv$	A	B	A↔B
		0	0	1
		0	1	0
		1	0	0
		1	1	1

Например: $A \vee \overline{B} \wedge C \rightarrow D \leftrightarrow E$.

Порядок выполнения:

1. $\overline{B}$
2. $(\overline{B}) \wedge C$
3. $A \vee ((\overline{B}) \wedge C)$
4. $(A \vee ((\overline{B}) \wedge C)) \rightarrow D$
5. $((A \vee ((\overline{B}) \wedge C)) \rightarrow D) \leftrightarrow E$

Теперь выполним практические задания на заполнение таблицы истинности с помощью электронной таблицы Excel.

Порядок выполнения работы.

1. Найдите обозначения логических функций, которые имеются в Excel.
2. Используя Мастер функций, начните заполнять таблицу:

	A	B	C	D	E
1	A	B	$\overline{A}$	A&B	A или B
2	ложь	ложь			
3	ложь	истина			
4	истина	ложь			
5	истина	истина			

3. Используя Мастер функций, продолжите заполнение таблицы:
   А) В ячейку C2 занесите формулу: =НЕ(A2).
   В ячейку D2 занесите формулу: =И(A2;B2).
   В ячейку E2 занесите формулу: =ИЛИ(A2;B2).
   Б) Выделите ячейки C2:E2.
   В) Скопируйте выделенный блок в ячейки C3:E5.
4. Проверьте полученную таблицу.
5. Перейдите на лист 2.
6. Используя Мастер функций, постройте таблицу истинности функций $A \vee A \vee A \vee A$, $A \,\&\, A \,\&\, A \,\&\, A$:

A	B	A или A или A или A	A и A и A и A
ЛОЖЬ	ЛОЖЬ	= ИЛИ(A2;A2;A2;A2)	= И(A2;A2;A2;A2)
ЛОЖЬ	ИСТИНА		
ИСТИНА	ЛОЖЬ		
ИСТИНА	ИСТИНА		

7. Перейдите на лист 3.
8. Используя Мастер функций, постройте таблицу истинности для следующих сложных высказываний:
   1. $F = (A \vee B) \wedge (\overline{A} \vee \overline{B})$
   2. $F = X \vee Y \wedge \overline{Z}$
   3. $F = X \wedge Y \vee (X \vee Y) \vee X$
   4. $F = A \wedge (B \rightarrow C)$
   5. $F = (B \wedge \overline{B}) \leftrightarrow (A \vee D)$

§ 15. Логические элементы компьютера

Вспомните!

- *Что такое конъюнкция, дизъюнкция, инверсия?*
- *Как строятся таблицы истинности?*

Вы узнаете:

- *что такое логические элементы и логическая схема.*

Термины:

- конъюнктор;
- дизъюнктор;
- инвертор.

С развитием вычислительной техники элементы математической логики стали широко использоваться в вычислительной технике и программировании. Логика компьютера основана на комбинации электронных элементов, которые выполняют определенные логические операции. Эти электронные элементы называются *логическими элементами*.

Логический элемент – это электронное устройство, реализующее одну из логических функций. В зависимости от типа элемента выводится один или несколько входных сигналов (1 – сигнал, 0 – сигнала нет), а на выходе – один выходной сигнал. Названия и символы логических элементов являются стандартными и используются для создания и описания компьютерных логических схем. Процессор и оперативная память компьютера построены на основе базовых логических элементов.

Рассмотрим основные логические элементы.

ИНВЕРТОР реализует операцию отрицания (инверсию). В схемах он изображается так: у инвертора один вход и один выход. Сигнал на выходе появляется тогда, когда на входе его нет, и наоборот (*рис. 7, 8*).

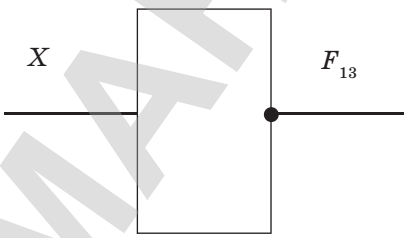


Рис. 7. *ГОСТ*

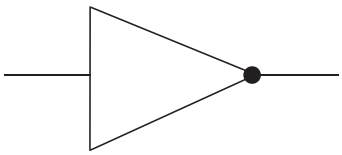


Рис. 8. *Стандарт ANSI*

Простая модель логического элемента «НЕ» может представлять собой электрическую схему электрических элементов (*рис. 9*).

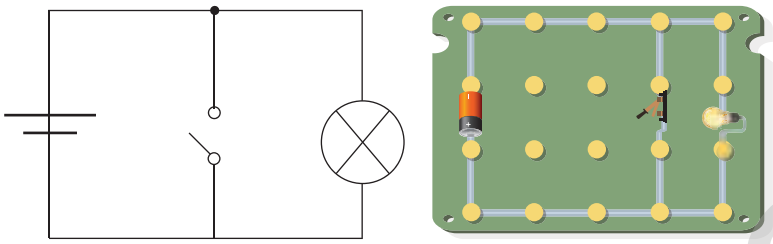


Рис. 9. *Электрическая схема логического элемента «НЕ»*

Это интересно!

Американский логик Чарльз Сандерс Пирс (в его честь названа одна из логических операций – «стрелка Пирса») работал над модификацией и расширением булевой алгебры с 1867 года. Пирс первым осознал, что бинарная логика имеет сходство с работой электрических переключательных схем. Электрический переключатель либо пропускает ток (что соответствует значению Истина), либо не пропускает (что соответствует значению Ложь). Позже Пирс придумал простую электрическую логическую схему, но не собрал ее.

Как видно из схемы, если переключатель выключен (на входе 0), то лампа загорается (на выходе 1) и наоборот.

КОНЪЮНКТОР реализует операцию конъюнкции (логическое умножение). В схемах изображается так: у конъюнктора не менее двух входов и один выход. Сигнал на выходе появляется тогда, когда на все входы поданы сигналы (*рис. 10, 11*).

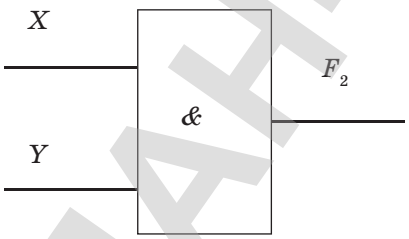


Рис. 10. *ГОСТ*

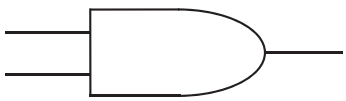


Рис. 11. *Стандарт ANSI*

В качестве простой модели логического элемента «И» можно рассмотреть электрическую схему источника тока, лампы и двух переключателей (*рис. 12*).

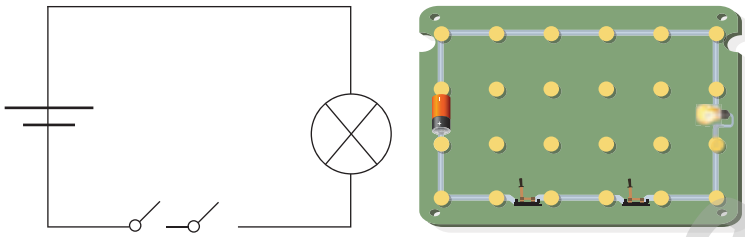


Рис. 12. *Электрическая схема логического элемента «И»*

Как видно из схемы, если оба переключателя замкнуты (на обоих входах 1), лампа загорается (на выходе 1).

ДИЗЪЮНКТОР реализует операцию дизъюнкции (логическое сложение). В схемах изображается так: у дизъюнктора не менее двух входов и один выход. Сигнал на выходе **не** появляется тогда, когда на все входы не поданы сигналы (*рис. 13, 14*).

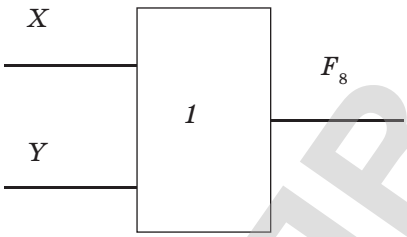


Рис. 13. *ГОСТ*

Рис. 14. *Стандарт ANSI*

С помощью этих логических элементов можно реализовать любую логическую функцию в виде логической схемы.

Простая модель логического элемента «НЕ» может представлять собой электрическую схему электрических элементов (*рис. 15*).

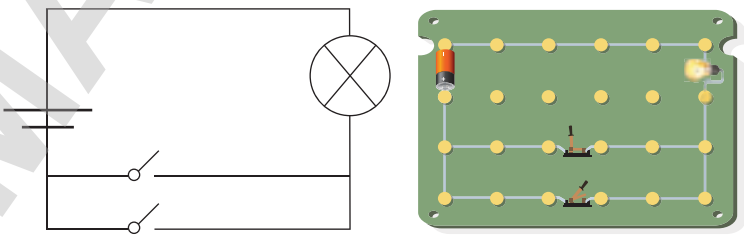


Рис. 15. *Электрическая схема логического элемента «ИЛИ»*

Как видно из схемы, если хотя бы один из выключателей выключен (на выходе 1), то лампа загорается (на выходе 1).

Например, для логической функции $F(A, B) = (\overline{A} \,\&\, B) \vee (A \,\&\, \overline{B})$ необходимо построить комбинационную схему (*рис. 16*). Построение схемы начнем с логической операции, которую нужно будет выполнить в самом конце. В нашем случае этой операцией является логическое умножение, значит, в конце логической схемы должен быть дизъюнктор. Он получает сигналы от двух соединений, один сигнал является нормальным для этих соединений, а один сигнал – инвертированным.

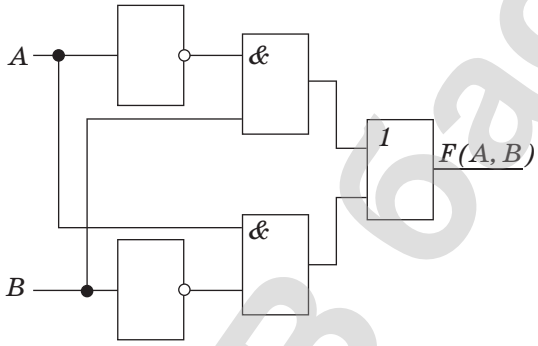


Рис. 16. *Комбинационная схема логической функции* $F(A, B) = (\overline{A} \,\&\, B) \vee (A \,\&\, \overline{B})$

1 Отвечаем на вопросы

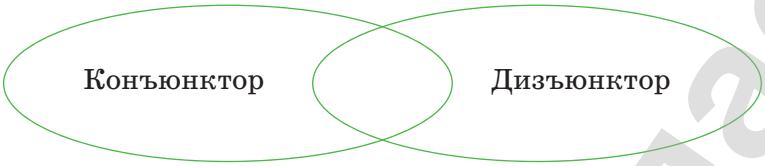
1. Зачем нужны логические элементы?
2. Какие устройства компьютера создаются на основе логических элементов?
3. Какие виды логических элементов вы знаете?
4. Где применяется конъюнктор?
5. Где применяется дизъюнктор?
6. Где применяется инвертор?

2 Думаем и обсуждаем

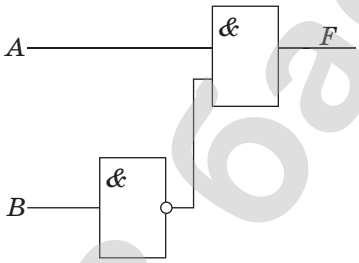
1. Почему вычислительную технику создают на основе логических элементов?
2. В чем важность логических элементов?
3. Почему модель инвертора можно объяснить простой электрической схемой?

3 Анализируем и сравниваем

1. В чем различия между конъюнктором и дизъюнктором? Сравните с помощью диаграммы Венна.

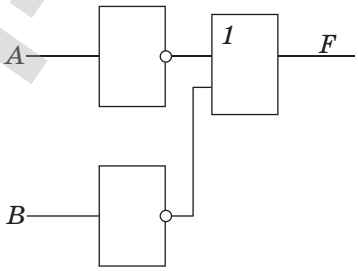


2. Проанализируйте результат выхода F данной электрической схемы.



4 Выполняем в тетради

1. Учитывая набор возможных сигналов каждого входа, определите сигнал на выходе электрической схемы и вид логического высказывания.



2. Постройте электрические схемы для следующих логических функций.
   1) $B \,\&\, (A \wedge B)$.

2) $A \,\&\, (B \vee \overline{B})$.
3) $A \,\&\, (A \vee B \vee C)$.
4) $\overline{A \vee B \vee \overline{C}}$.

5 Выполняем на компьютере

1. С помощью сети Интернет найдите и запишите в тетради основные законы булевой алгебры.
2. Изобразите электрическую схему логической функции $F = \overline{A} \vee B$ с помощью компьютера.

6 Делимся мыслями

1. Объясните действия электрических схем, выполняющих модели логических схем, с точки зрения законов постоянного тока.
2. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры.

§ 16. Логические основы компьютера

Вспомните!

- *Что относится к логическим элементам?*

Вы узнаете:

- *о структуре процессора;*
- *арифметико-логических устройствах;*
- *об устройствах управления;*
- *о видах регистров памяти и их функциях.*

Термины:

- АЛУ;
- УУ;
- регистр;
- сумматор;
- шина.

С тех пор, как в 1959 году впервые была создана микросхема, в мире производятся тысячи различных универсальных и специальных процессоров. Однако любой процессор по сей день состоит из арифметико-логического устройства (АЛУ), устройства управления (УУ) и памяти регистров (*схема 2*).

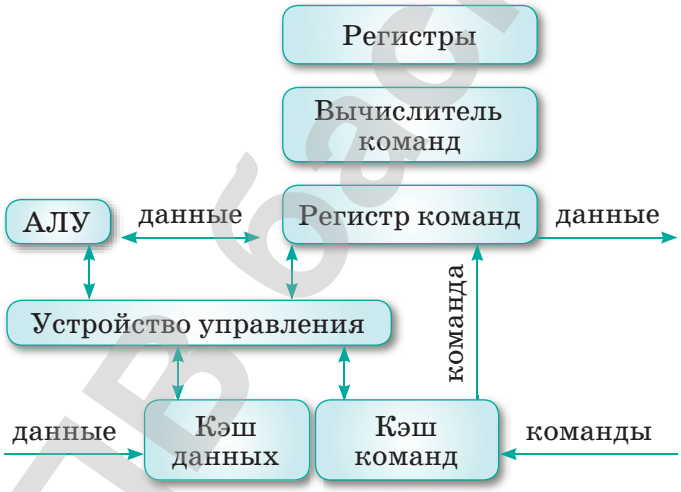


Схема 2. *Структура процессора*

Устройство управления руководит работой всех устройств компьютера. Это устройство берет следующую команду из регистра команд, определяет, что делать с данными, и дает последовательность выполнения поставленной задачи.

Арифметико-логическое устройство выполняет арифметические и логические операции преобразования информации.

Регистры представляют собой внутреннюю память процессора. Каждый регистр имеет свою функцию. Например, необходимо, чтобы процессор выполнил сложение двух чисел. Для этого нужно сначала прочитать первое слагаемое из памяти, затем прочитать второе слагаемое и отправить их в оперативную память.

Процессору требуется место для хранения первого и второго слагаемого, а также их суммы. Для этого в самом процессоре имеется внутренняя ячейка, называемая *сумматором*. Кроме этого, имеется *вычислитель команд*, который дает процессору

информацию о считываниях следующей команды оперативной памяти. Сама команда после оперативной памяти помещается в так называемый *регистр команд*. После выполнения всех команд результат копируется из регистра в ячейку оперативной памяти.

Как видно из примера, в зависимости от выполняемой операции существует несколько типов регистров:

- **сумматор** – регистр арифметико-логического устройства, участвующего в выполнении каждой операции;
- **вычислитель команд** – регистр устройства управления, который соответствует адресу следующей выполняемой команды;
- **регистр команд** – регистр блока управления, который хранит код команды в течение времени, необходимого для ее выполнения.

Современные процессоры включают в себя множество других устройств, но указанные выше компоненты и соединяющие их внутренние шины данных являются необходимым минимумом. Все устройства процессора связаны между собой через внутренние шины данных.

Шина – это группа проводников, используемая в качестве связующего передачи цифровой информации. В процессоре есть 3 основные шины: *шина данных*, *адресная шина* и *шина управления*.

Шина данных. Через эту шину распределяются данные между различными устройствами. Например, информация, считываемая из оперативной памяти, отправляется в процессор для обработки, а обработанные данные снова отправляются в оперативную память для хранения. Таким образом, данные могут передаваться с одного устройства на другое в любом направлении через шины данных.

Адресная шина предназначена для отправки адреса ячейки памяти или устройства, к которому обращается процессор.

Через **шины управления** передаются такие сигналы, как чтение, запись и подготовка, определяемые описанием обмена информации. Работа вышеуказанных устройств процессора влияет на производительность процессора. Кроме того, скорость процессора зависит от следующих характеристик:

- количество ядер в процессоре – позволяет нескольким приложениям работать одновременно;
- частота процессора – это скорость передачи данных между системной шиной и процессором;

- тепловая проводимость процессора измеряется в ваттах и показывает, с какой мощностью работает вентилятор (кулер).

1 Отвечаем на вопросы

1. Каковы функции АЛУ и УУ в логической структуре компьютера?
2. Назовите виды регистров.
3. Какие действия выполняет сумматор?
4. В чем разница между регистром команд и вычислителем команд?

2 Думаем и обсуждаем

1. Почему арифметико-логическое устройство относится к основной части процессора?
2. Как УУ контролирует работу всех устройств компьютера?

3 Анализируем и сравниваем

1. Проанализируйте основные части архитектуры Джона фон Неймана. Обменяйтесь мнениями.
2. Изучите типы регистров.

4 Выполняем в тетради

Устройство управления (УУ) и арифметико-логическое устройство (АЛУ), как правило, интегрированы в центральный процессор. Изобразите связь между памятью, внешней памятью и устройствами ввода и вывода в виде схемы.

5 Выполняем на компьютере

С помощью поисковых систем найдите в Интернете новейшую модель процессора и представьте информацию об этом устройстве.

6 Делимся мыслями

1. Что вы узнали на уроке? Чему научились? Поделитесь своим мнением с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры.
2. Какой процессор вы выбираете? Почему?

§ 17–18. Принципы кодирования текстовой информации

Вспомните!

- *Что такое логика?*
- *Какие виды логики вы знаете?*

Вы узнаете:

- *что такое кодирование;*
- *о видах кодирования текста;*
- *что такое ASCII;*
- *что такое Unicode.*

Термины:

- кодирование;
- ASCII;
- Unicode.

Текст в компьютере состоит из отдельных символов. Чтобы отобразить каждый символ на дисплее, вам понадобится соответствующий правилу машинный код. На клавиатуре написаны обычные буквы, цифры, знаки препинания и другие символы. В оперативную память они поступают как двоичный код. Это означает, что каждый символ является 8-битным двоичным кодом.

Кодирование – это соответствие символов машинному коду, а также перевод информации с компьютерного языка на удобный для пользователя язык и наоборот.

Особенность кодирования – каждому символу соответствует десятичный код от 0 до 255 или двоичный код от 00000000 до 11111111 соответственно. Таким образом, человек различает символы путем записи, а компьютер – по коду.

Байтовое кодирование символов очень удобно, т.к. байт является минимальной адресной частью памяти, и процессор, обрабатывая текст, дает отдельное значение каждому символу. При этом 256-ти символов вполне достаточно для введения различной символьной информации.

Все символы компьютера пронумерованы от 0 до 255. Каждому номеру соответствует 8-разрядный двоичный код от 00000000 до 11111111. Этот код представляет собой порядковый номер символов в обычной двоичной вычислительной системе. Таблица всех символов, соответствующих порядковому номеру в алфавите компьютера, называется **таблицей кодирования.**

Это интересно!

Самым значимым изменением в таблице кодировки Unicode стало введение в 1998 году символа валюты евро.

Поскольку создатели компьютера были англоязычными, они ввели 26 букв английского алфавита, 9 знаков препинания (. , : ! " ; ? ( )), пробел, 10 цифр, 5 знаков арифметических действий (+, –, *, /, ^) и специальные символы (№, % , _, #, $, ^, &, >, <, |, \).

Всего было согласовано закодировать двоичным кодом более 100 символов от 0 до 2^7 (128 мест).

Эта таблица кодирования была названа ASCII (American Standard Code for Information Interchange).

В *таблице 2* символы ASCII для удобства нумеруются шестнадцатеричной системой счисления (0–7F). В ней сначала расположены невводимые символы (от 0 до 7F), а затем вводимые символы (от 20 до 7F).

Таблица 2. *Таблица символов ASCII*

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

При перемещении русского текста из одного компьютера в другой и из одной системной программы в другую возникает множество трудностей. Поэтому для русских букв разработан тип кодирования KOI8 по хронологическому стандарту.

Этот тип кодирования применяется еще с 1970-х годов в серии компьютера ЕС ЭВМ, а с середины 80-х годов вступила в действие операционная система UNIX.

В начале 1990 года при операционной системе MS DOS применялся тип кодирования CP866 («CP» «Code Page», означает «страница кодирования»).

Компьютеры фирмы Apple под управлением операционной системы Mac OS пользуются своим собственным видом кодирования Mac.

Кроме того, Международная организация по стандартизации (International Standards Organization, ISO) предложила вид кодирования для русского языка под названием ISO 8859-5.

В январе 1991 года решилась проблема стандартизации символьного кодирования – было введено новое кодирование Unicode.

Это 16-разрядное кодирование, и один символ здесь занимает 2 байта.

Конечно, от этого объем памяти увеличивается в два раза. Эта таблица кодирования может включать в себя до 65536 символов.

В таблицу Unicode введен всемирный алфавит, а также многие математические, музыкальные, химические и другие символы. Таблица кодирования Unicode для кириллицы (опубликованная на сайте Unicode Consortium часть таблицы Unicode 4.0) приведена в *таблице 3*.

Таблица 3. *Часть таблицы UNICODE 4.0*

0400	Cyrillic															04
	040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04
0	È 0400	А 0410	Р 0420	а 0430	р 0440	è 0450	Ѡ 0460	Ψ 0470	Ҁ 0480	Ґ 0490	Ҡ 04A0	Ұ 04B0	І 04C0	Ă 04D0	З 04E0	Ў 040
1	Ё 0401	Б 0411	С 0421	б 0431	с 0441	ё 0451	ѡ 0461	ψ 0471	ҁ 0481	ґ 0491	ҡ 04A1	ұ 04B1	Ж 04C1	ă 04D1	з 04E1	ў 041
2	Ђ 0402	В 0412	Т 0422	в 0432	т 0442	ђ 0452	Ъ 0462	Θ 0472	≠ 0482	Ғ 0492	Ң 04A2	Ҳ 04B2	ж 04C2	Ä 04D2	Ӣ 04E2	Ӳ 042
3	Ѓ 0403	Г 0413	У 0423	г 0433	у 0443	ѓ 0453	ѣ 0463	ѳ 0473	◌҃ 0483	ғ 0493	ң 04A3	ҳ 04B3	Җ 04C3	ä 04D3	ӣ 04E3	ӳ 043
4	Є 0404	Д 0414	Ф 0424	д 0434	ф 0444	є 0454	Ю 0464	V 0474	◌҄ 0484	Ҕ 0494	Ҥ 04A4	Ҵ 04B4	җ 04C4	Æ 04D4	Ӥ 04E4	Ӵ 044
5	S 0405	Е 0415	Х 0425	е 0435	х 0445	s 0455	ю 0465	v 0475	◌҅ 0485	ҕ 0495	ҥ 04A5	ҵ 04B5	Л 04C5	æ 04D5	ӥ 04E5	ӵ 045
6	І 0406	Ж 0416	Ц 0426	ж 0436	ц 0446	і 0456	Ѧ 0466	V̋ 0476	◌҆ 0486	Ж 0496	Ҧ 04A6	Ҷ 04B6	л 04C6	Ё 04D6	Ӧ 04E6	

0400																04
										Cyrillic						
7	Ї 0407	З 0417	Ч 0427	з 0437	ч 0447	ї 0457	ѧ 0467	ѷ 0477		ж 0497	ҧ 04A7	ҷ 04B7	ӈ 04C7	ё 04D7	ӧ 04E7	
8	Ј 0408	И 0418	Ш 0428	и 0438	ш 0448	ј 0458	ѩ 0468	ѹ 0478	◌҈ 0488	ҙ 0498	ҩ 04A8	ҹ 04B8	ӊ 04C8	҈ 04D8	ө 04E8	Ы 048
9	Љ 0409	Й 0419	Щ 0429	й 0439	щ 0449	љ 0459	ѫ 0469	ѻ 0479	◌҉ 0489	ҙ 0499	ҩ 04A9	ҹ 04B9	ӊ 04C9	҉ 04D9	ө 04E9	Ы 049
A	Њ 040A	К 041A	Ъ 042A	к 043A	ъ 044A	њ 045A	ѫ 046A	ѽ 047A	ҋ 048A	ҟ 049A	ҫ 04AA	ҳ 04BA	ӌ 04CA	ҋ 04DA	ө 04EA	
B	Ћ 040B	Л 041B	Ы 042B	л 043B	ы 044B	ћ 045B	ѫ 046B	ѽ 047B	ҋ 048B	ҟ 049B	ҫ 04AB	ҳ 04BB	ӌ 04CB	ҋ 04DB	ө 04EB	
C	Ќ 040C	М 041C	Ь 042C	м 043C	ь 044C	ќ 045C	ѭ 046C	ѿ 047C	ҍ 048C	ҡ 049C	ҭ 04AC	ҽ 04BC	ӌ 04CC	ӝ 04DC	ӭ 04EC	
D	Ѝ 040D	Н 041D	Э 042D	н 043D	э 044D	ѝ 045D	ѭ 046D	ѿ 047D	ҍ 048D	ҡ 049D	ҭ 04AD	ҽ 04BD	Ӎ 04CD	ӝ 04DD	ӭ 04ED	
E	Ў 040E	О 041E	Ю 042E	о 043E	ю 044E	ў 045E	ѯ 046E	ѽ 047E	ҏ 048E	ҟ 049E	ү 04AE	ҿ 04BE	ӎ 04CE	ӟ 04DE	ӯ 04EE	
	Ц 040	П 041	Я 042	п 043	я 044	ц 045	ѯ 046	ѿ 047	ҏ 048	ҡ 049	ү 04A	ҿ 04B		ӟ 04D	ӯ 04E	

1 Отвечаем на вопросы

1. Какие виды кодирования вы знаете?
2. Как создаются таблицы кодирования?
3. Какова функция таблиц кодирования?
4. Какие примеры кодирования можно привести из повседневной жизни?

2 Думаем и обсуждаем

1. Чем удобно байтовое кодирование символов?
2. Назовите причины появления таблицы Unicode.

3 Анализируем и сравниваем

Сравните виды кодирования.

4 Выполняем в тетради

Заполните пробелы:

1. Все таблицы символов в соответствии с порядковым номером в компьютерном алфавите называются_____.
2. _____ – это символы, соответствующие коду машины. Переводит информацию с компьютерного языка на понятный пользователю язык и наоборот.
3. Для удобства в таблице ASCII символы _____ нумеруются системой счисления.

5 Выполняем на компьютере

С помощью табличного процессора Excel составьте таблицу кодирования ASCII.

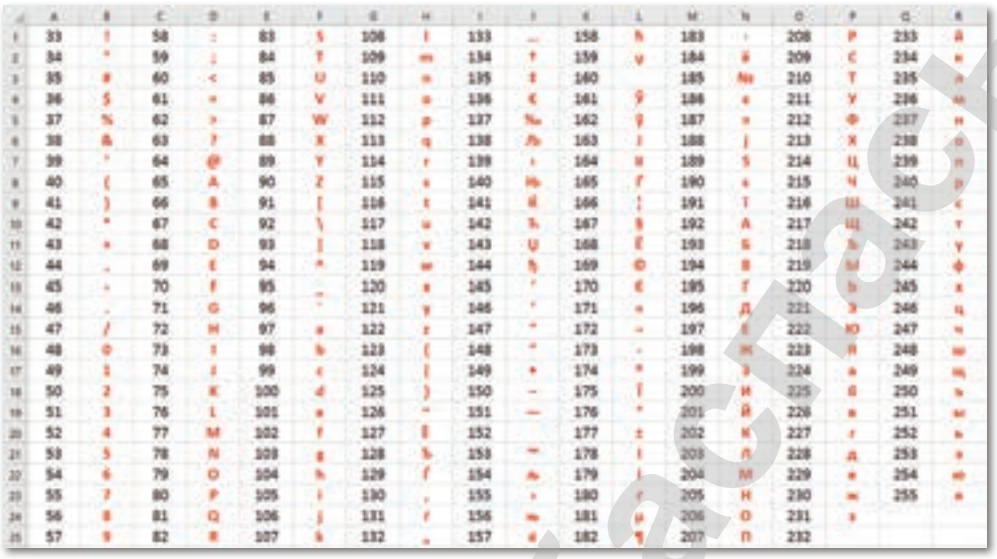
1. Запустите MS Excel.
2. Введите числа от 33 до 255, начиная с ячейки A1 (по 25 в каждой строке через столбец: A, C, E, ... , S).

	A	C	E	G	I	K	M	O	Q
1	33	58	83	108	133	158	183	208	233
2	34	59	84	109	134	159	184	209	234
3	35	60	85	110	135	160	185	210	235
4	36	61	86	111	136	161	186	211	236
5	37	62	87	112	137	162	187	212	237
6	38	63	88	113	138	163	188	213	238
7	39	64	89	114	139	164	189	214	239
8	40	65	90	115	140	165	190	215	240
9	41	66	91	116	141	166	191	216	241
10	42	67	92	117	142	167	192	217	242
11	43	68	93	118	143	168	193	218	243
12	44	69	94	119	144	169	194	219	244
13	45	70	95	120	145	170	195	220	245
14	46	71	96	121	146	171	196	221	246

3. В ячейку B1 введите формулу =СИМВОЛ(A1), нажмите Enter.

	A	B	C	D	E
1	33	=СИМВОЛ(A1)	58		83
2	34		59		84

4. Введите эту формулу в соответствующие ячейки для остальных столбцов: B, D, F, ..., T с помощью функции Автозаполнение.
5. В результате получим таблицу кодирования ASCII.



6 Делимся мыслями

1. Как вы думаете, используем ли мы кодирование символов в повседневной жизни? Проведите дискуссию между собой.
2. Будут ли изменения в таблице кодирования, когда казахский язык полностью перейдет на латинский алфавит?

Раздел III

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Цели обучения:

- писать код на языке программирования, используя функции и процедуры;
- использовать процедуры и функции для обработки строк;
- использовать файлы для чтения и записи информации;
- реализовывать алгоритмы сортировки для решения практических задач
- реализовывать алгоритмы поиска на графах для решения практических задач.

§ 19. Пользовательские функции и процедуры. Процедуры

Вспомните!

- *Что вы знаете о принципах кодирования текстовой информации?*
- *Как применяют таблицу кодирования?*
- *Расскажите о различиях кодов Unicode и ASCII.*

Вы узнаете:

- *что такое процедуры;*
- *как примененяют функции и процедуры.*

Термины:

- процедура;
- параметр;
- служебное слово def.

Разработка программ с машинным кодом сложна. Поэтому в настоящее время все программы создаются с помощью языков программирования. Язык программирования Python позволяет работать вместе с компилятором и интерпретатором.

Процедура – вспомогательный алгоритм, который выполняет несколько действий.

В языке Python процедура начинается со служебного слова `def`, состоит из пустых или непустых скобок и двоеточия.

Рассмотрим пример написания процедуры:

```
def Err(): #определение процедуры
  print ("Ошибка: некорректные данные")
n = int (input('введите положительное число'))
if n < 0:
  Err() #вызов процедуры
```

- Код процедуры записывается до тех пор, пока не будет вызван в основной программе.
- Программа может иметь несколько процедур.
- Чтобы процедура работала правильно, ее необходимо вызвать из основной программы или другой процедуры.
- Процедура должна быть определена до вызова. Определение процедуры начинается с помощью служебного слова `def`.
- Вызов процедуры осуществляется под именем перед двойными скобками. Например, `Err()`.
- Использование процедуры в программе позволяет сократить код и обеспечить быстрое чтение программы.

Параметр процедуры

Рассмотрим использования параметров языка программирования Python на примере.

Пример: написать программу процедуры, которая выводит введенный символ с новой строки.

```
def printChar(s):
   print (s)
sim = input('введите символ')
printChar(sim) #первый вызов, вывод вызванного символа
printChar('*') #второй вызов, * результат
```

Глобальная переменная – это значение, которое присваивается процедуре в основной программе. Невозможно получить **локальную (внутреннюю) переменную** из основной программы или другой процедуры, которая используется только на уровне этой процедуры. Параметры процедуры – локальные переменные.

1 Отвечаем на вопросы

1. Что такое процедура?
2. Когда мы можем использовать процедуру?
3. Какие программы вы знаете?
4. С чего начинается процедура на языке Python?
5. Что такое глобальная переменная?
6. Что относится к параметрам процедуры?

2 Думаем и обсуждаем

1. Для чего нужны процедуры?
2. Почему программы создаются с помощью языков программирования?
3. Почему необходимо использовать процедуры при написании программы?

3 Анализируем и сравниваем

1. Проанализируйте действия процедур.
2. Сравните действия процедур и определите их сходства.

4 Выполняем в тетради

1. Заполните таблицу в тетрадях.

Название	Функции
Процедура	
Подпрограмма	
Параметр	

2. Запишите в тетради определение и виды процедур.

5 Выполняем на компьютере

Составьте программу, которая выводит все делители введенного числа на экран (в одной строке), с помощью процедуры.

6 Делимся мыслями

Что вы узнали на уроке? Чему вы научились? Поделитесь своими мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры. Какие примеры можно привести на использование программы в повседневной жизни?

§ 20. Практикум. Написание кода на языке программирования с использованием процедур

Пример 1. Создание программы с использованием процедуры, которая будет выводить введенные положительные числа в столбец.

Решение:

```
n = int(input())
n = abs(n)
def printDigits (n):
    n = str (n)
    for i in range (0, len(n)):
        print (n[i])
printDigits(n)
```

Результат программы:

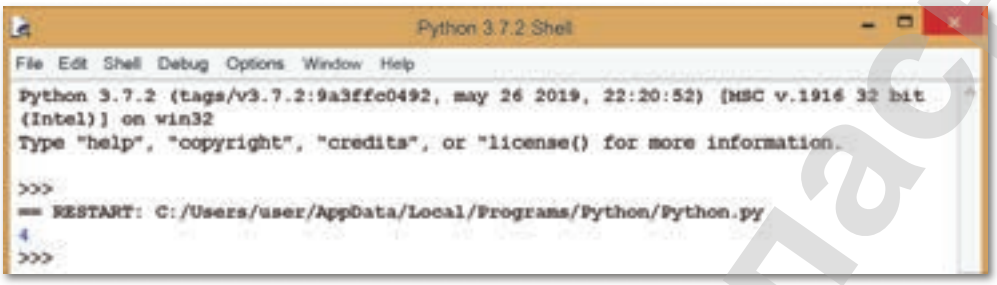
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, may 26 2019, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits", or "license() for more information.

=== RESTART: C:/Users/user/AppData/Local/Programs/Python/Python.py ===
5487231
5
4
8
7
2
3
1
>>>
```

Пример 2. Создание программы, которая выводит параметр глобальной переменной.

Решение:

```
x = 3 #глобальная переменная
def pr(a):#параметрическая процедура
    a = 4 #локальная переменная
    print (a) #4
        pr(x) #вывод параметра глобальной переменной (3)
```

Результат программы:



Задание 1. Напишите программу решения группы уравнений $p_i x^2 + q_i x + r_i = 0$, где p, q, r – массив фактических чисел, состоящий из элемента k. Решение одного уравнения можно организовать в виде процедуры.

Задание 2. Дано натуральное число n. Найдите все числа, которые могут показать сумму квадратов двух натуральных чисел 1, 2, 3 ..., n. (Необходимо определить процедуры для нахождения полных квадратов).

Задание 3. Найдите наибольший элемент в массивах a, b, c и его номер.

Задание 4. Напишите процедуру расчета объема и площади поверхности параллелепипеда.

Задание 5. Введите функцию, которая возвращает наибольшее из двух целых чисел, взятых в качестве аргумента.

Задание 6. Создайте процедуру, которая выводит на экран введенные цифры столбцом, начиная с последней.

§ 21. Пользовательские функции и процедуры. Функции

Вспомните!

- *Что такое процедуры?*
- *Для чего применяют процедуры?*
- *Почему в ходе программы используют процедуру?*

Вы узнаете:

- *о подпрограммах;*
- *что такое функция;*
- *о видах функции.*

Термины:

- функция;
- оператор;
- переменная.

При описании алгоритма решения конкретных задач возникает необходимость многократного повтора части циклических действий на разных этапах вычислительного процесса. Конечно, в программе можно записать группу операторов, выполняющих повторяющуюся часть, но это неэффективно. Такие действия являются эффективными для интеграции во внутреннюю программу, их можно описать один раз и при необходимости изменить входящие в нее исходные данные.

В. Ф. Очков дал очень поэтическое определение подпрограммы: «Подпрограмма – это припев песни, который поют несколько раз, а в тексте песен печатают только один раз».

С математической точки зрения любая внутренняя структура – это замкнутая часть целостности, которая может рассматриваться как самостоятельная структура: подгруппа – группа, внутренняя алгебра – алгебра, внутреннее пространство – пространство и т. д.

В языке программирования Python существует два типа подпрограмм: **процедуры** и **функции**. Для описания их общих признаков мы можем использовать термин «подпрограмма». Если в тексте встречается термин «процедура» или «функция», то эта информация характерна только для конкретного вида одной подпрограммы: только для процедуры или функции.

Функция – это часть программы, которую вызывают в начале программы. Функция должна быть определена до ее вызова.

- Функция, в отличие от процедуры, возвращает значение.
- Для возврата значения функции используется оператор `return`.
- Вызов функции заканчивается записью ее имени и выводом значения.

Синтаксис написания функции на языке Python:

`def` **Имя_функции (Список_Параметров):**
Выражения

В языке Python перед определением функции пишется **служебное слово** `def` (заголовок функции), с помощью которого мы вызываем функцию.

После заголовка функции за закрывающей скобкой следует двоеточие и выражения, относящиеся к вычислению (нажатием клавиши TAB), записываются в последующем ряду:

- при присвоении имени функции имя переменной должно быть записано латинскими буквами;
- список параметров содержит значения, передаваемые функции. Эти параметры записываются через запятую.

Часть функций в языке Python представляет собой **встроенные функции**, используемые в синтаксисе языка. К ним относятся следующие служебные слова: `int`, `input`, `randint`, `print`.

Встроенные функции делятся на две группы.

1. Функции, работающие с символами `ord()`, `chr()`, `len()`.
2. Математические функции `abs()`, `round()`, `divmod()`, `pow()`, `max()`, `min()`, `sum()`.

Рассмотрим пример создания функции пользователя.

Вычислить сумму цифр числа:

```
def sumD(n): #определение функции параметром
    sum=0
      while n!= 0:
        sum += n % 10
        n = n // 10
      return sum #возврат значения функции
print(sumD(1075)) #вызов функции с параметром
```

Обратимые значения

Основное отличие функций и процедур заключается в количестве возвращаемых ими значений.

Любая функция должна завершить свою работу и вернуть значение основной программе (или вызванной подпрограмме).

Для возврата результата используется специальная переменная с именем, соответствующим названию функции. Оператор присвоения значения переменной в обязательном порядке должен по крайней мере один раз использоваться в теле функции.

Каждая функция имеет следующий вид: функция всегда находится на правой стороне равенства, после имени в скобках записывается значение аргумента функции. На левой стороне от знака равенства в конце находится переменная, которой присваивается значение функции.

Глобальные переменные объявляются в начале, до объявления любых подпрограмм, программ типов данных, переменных и констант. Эти объекты отображаются во всех программах, а также во всех их подпрограммах. Глобальные переменные принимают участие во всей работе программы.

Локальные переменные объявляются внутри определенной подпрограммы и просматриваются только в этой подпрограмме. Локальные переменные не исполняются до тех пор, пока не будет вызвана подпрограмма.

Синтаксис функции (*рис. 17*):

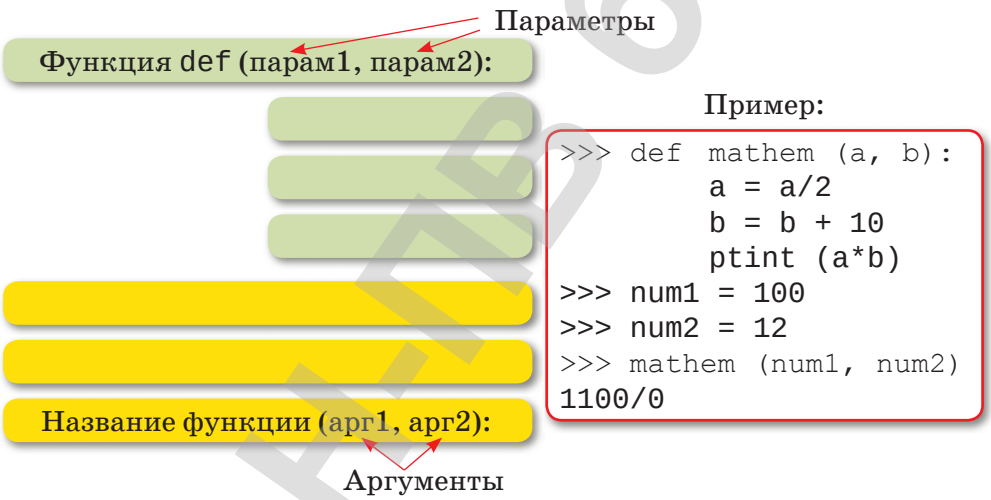


***Рис. 17.** Синтаксис написания функции*

Описание простой функции:

```
def add(x, y):
    return x + y
```

Руководство `return` указывает, что значение должно быть возвращено. В данном примере функция возвращает сумму `x` и `y`.

Теперь вызов этой функции во время вычисления будет выглядеть следующим образом:

```
>>>
>>> add(1, 10)
11
>>> add('abc', 'def')
'abcdef'
```

В ходе программирования функция не только возвращает данные, но и осуществляет применение настроек функции. Количество параметров может быть любым.

Параметры являются локальными переменными, значение которых присваивается при вызове функции. Точное значение, передаваемое при вызове функции, называется **аргументом**.

Аргументы функции могут не отображаться четко или могут принимать аргументы по умолчанию. Функция может быть вызвана любым количеством аргументов. В этом случае перед аргументом ставится знак * и чаще всего этот аргумент объявляется в виде цепочки.

Функция может быть вызвана самим телом функции, которую также называют **рекурсией.**

Одна из особенностей функции в языке Python – имя функции может быть построено без объявления. Такие функции называются **лямбда-функциями**.

Объявление лямбда-функции:

Аргументы lambda:результаты

Лямбда-функция может быть представлена в виде аргумента функции:

```
fun1(lambda x,y:x*y +pow(x,2)+ pow(y,2), 1,4)
```

Лямбда-функция может также использоваться в выражении:

```
z = 1 + (lambda x, y: x*y + pow(x, 2) + pow(y, 2))(1,3)**2
```

1 Отвечаем на вопросы

1. Что такое функция?
2. В каких ситуациях используется функция?
3. Какие виды функций вы знаете?
4. Что такое обратимые значения?

2 Думаем и обсуждаем

1. Почему глобальные переменные сообщаются в начале программы?
2. Почему локальные переменные не могут быть выполнены до вызова внутренней программы?

3 Анализируем и сравниваем

Сравните функции локальной переменной и глобальной переменной, определите их сходства.

4 Выполняем в тетради

1. Заполните таблицу.

Названия	Назначения
Функция	
Переменные	
`return`	
`def()`	
`print()`	

2. Выполните на компьютере задания для функций, которые приведены в этом параграфе, и запишите в тетради определение и виды функций.

5 Выполняем на компьютере

Вычислите сумму и произведение элементов массива *b*, *c*, *d* с использованием функции.

6 Делимся мыслями

1. Что вы узнали на уроке? Чему вы научились? Поделитесь своими мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры.
2. Составьте кроссворд или ребус с терминами: функция, аргумент, параметр.

§ 22. Практикум. Запись кода на языке программирования с использованием функций

Пример 1. Вычисление площади цилиндра, используя функции.

```
#описание функции
def cylinder():
#исходные переменные
    r = float(input())
    h = float(input())
#площадь боковой поверхности цилиндра:
    side = 2 * 3.14 * r * h
#площадь части цилиндра:
    circle = 3.14 * r * 2
#площадь цилиндра:
    full = side + 2 * circle
    return full
square = cylinder()
print(square)
```

Результат программы:

```
3
7
188.4
```

В этой программе значение локальной переменной `full` возвращается из функции в часть основной программы. В данном случае это значение, полученное при расчете площади цилиндра.

В основной части программы данное значение принадлежит глобальной переменной `square`.

Выражение `square = cylinder()` выполняется следующим образом:

`cylinder()` *вызов функции.*

Она возвращает значение.

Это значение присваивается переменной `square`.

Результат можно вывести на экран сразу, без присвоения переменной:

```
...
print(cylinder())
```

Здесь число, полученное от функции `cylinder()`, передается в функцию `print ()`. Если функция `cylinder()` записана в программе без присвоения переменной полученным значениям, то эти данные теряются. Но синтаксической ошибки не будет.

Пример 2. Создание программы с использованием вызова функции на примере расчета чисел Фибоначчи.

```
def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end = ' ')
        a, b = b, a + b
    print()
fib (400)
```

Сокращение `a, b = 0, 1` означает следующую запись:

```
a = 0
b = 1
ряд a, b = b, a + b:
a = b
b = a + b
```

Рассмотрим код по рядам:

1

`def fib(n)` – определение параметров функции `fib`, взятой в скобки. Параметру `n` придаем значение для расчета. Эта цифра передается в функцию как аргумент. Цифры, вводимые в интерпретатор Python, после двоеточия отображаются с помощью шага. Так мы указываем, что данные относятся к функциям.

2

Настроим переменные `a, b = 0, 1` в соответствии со значениями: `a = 0, b = 1`

3

```
while a < n:
```

Циклический оператор `while` будет исполняться пока не будет удовлетворено `a < n`. При этом после двоеточия открывается новый блок, который касается только цикла. Этот блок записывается через 8 пробелов.

4

```
print(a, end = ' ')
```

выводит данные переменной `a` к ответу и записывает пробел в результате каждого цикла.

5

```
a, b = b, a + b
```

Присваиваем переменным соответствующие значения: `a = b`

`b = a + b`

Для изменения исходных данных и расчета чисел Фибоначчи выполним действие 6.

6

```
print()
```

Обратите внимание, что `print()` должен быть выведен после цикла. Он относится к телу функции `fib`, а не к телу цикла `while`. Но для чего этот второй пустой `print()`? В данном случае выводится пустая строка. Эта функция предназначена для вывода новой пустой строки на экран. Для проверки программы необходимо вызвать функцию и задать параметр.

Вызовем функцию, зададим значение аргумента 40. В результате мы должны получить ряд чисел Фибоначчи до 40:

Пишем в интерпретаторе Python: `fib(40)`

Результат:

```
0 1 1 2 3 5 8 13 21 34
```

Функцию `fib()` можно вызвать повторно, присвоив другой параметр.

Например,

```
fib(400)
```

Результат:

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Таким образом функцию можно вызвать несколько раз.

Задание 1. Составьте программу для расчета степени чисел с использованием функции. Параметры входа: (число и степень).

Задание 2. Напишите функцию, которая вычисляет степень числа.

Задание 3. Составьте программу для расчета факториала натурального числа n.

Задание 4. Дано натуральное число n. Рассчитайте сумму:

$$1 - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!} + \frac{1}{5!} - \ldots (-1)^{n+1} \frac{1}{n!}$$

Задание 5. Найдите все трехзначные числа, равные сумме факториалов своих цифр.

Задание 6. Даны два треугольника, заданные координатами. Рассчитайте площадь треугольников по формуле Герона и определите, площадь какого треугольника больше.

§ 23. Работа со строками

Вспомните!

- *Что такое функция?*
- *Как применяют процедуры и функции?*

Вы узнаете:

- *что такое строки и символы;*
- *об операциях над строками.*

Термины:

- строка;
- символ.

В середине XX века первые компьютеры играли ключевую роль в выполнении сложных математических вычислений, а в настоящее время их основная работа – обработка текстовой (символьной) информации.

Символьные строки – ряды символов, расположенных последовательно.

Мы рассматривали строки вместе с целыми и точными числами как обычный тип данных, и знаем, что строка представляет собой последовательность символов, взятых в апостроф или кавычки.

В языке Python тип символических данных, объектом которого является один символ, отсутствует. Однако этот язык программирования позволяет рассматривать строки как объекты, длина которых состоит из одного или нескольких символов. При этом разница строк в списке не относится к структуре данных. Структуры данных могут состоять из простых типов данных, а в языке Python нет простого (символьного) типа для строк.

Во многих языках программирования существует тип символьных переменных для работы со строками: символы, строки и символьные массивы (в отличие от массивов, которые рассматриваются как один объект в целом). В языке программирования Python основной тип данных, используемый для обработки символьных измерений, – это символьные строки, тип **string**.

Для записи значения на строке используется оператор присвоения.

```
s = "Гульден учит уроки"
```

Строки заключаются в двойные кавычки или один апостроф. Если строка ограничена апострофом, то апострофа там может и не быть.

Для ввода строки с клавиатуры используется функция `input`:

```
s = input("Введите имя:")
print(s)
```

Длина строки (англ. *length* – длина) определяется с помощью функции `len`. В следующем примере переменная `n` определяет длину строки `s`:

```
n = len(s)
```

Для выделения отдельного символа из строки в квадратных скобках пишется номер символа, как для работы с элементом массива. Например, вывод на экран символа 5 индекса строки `s` выглядит следующим образом (в этом случае число строк должно быть не менее 6):

```
print(s [5])
```

Отрицательный индекс означает, что подсчет начинается с конца строки. Например, символ `s[-1]` означает `s[len(s)-1]`, последний символ строки.

В отличие от современных языков программирования, на языке программирования Python нельзя изменять символическую строку. Строка – неизменный объект.

С одной стороны, строки, как и списки, состоят из упорядоченных цепочек элементов. Соответственно, из них можно получить символы и отдельные части.

```
>>> s = "Hello, World!"
>>> s[0]
'H'
>>> s[7:]
'World!'
>>> s[::2]
'Hlo ol!'
```

В последнем случае видно, что шаг выделения равен 2, то есть выделяется каждый 2-й символ.

Примечание. Можно также разделить части из списка с шагами.

В языке программирования Python не изменяется существенное отличие строк. Не допускается повторная запись какого-либо отдельного символа или части:

```
>>> s[-1] = '.'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Как сообщает интерпретатор, этот тип объекта не поддерживает добавление элементов, не являющихся элементами типа `str`.

Если строку необходимо изменить, то нужно создать новую строку из старых частей:

```
>>> s = s[0:-1] + '.'
>>> s
'Hello, World.'
```

В этом примере часть снимается с начальной строки и добавляется в другую строку. Будет получена новая строка, которой будет присвоена переменная s. Ее прежнее значение будет удалено.

С введенной строки можно создать новую строку. Для этого нужно внести необходимые изменения. Напишем программу, которая заменяет букву "а" на букву "б" в строке, введенной с клавиатуры.

```
s = input("Введите строку:")
s1 = ""
for c in s:
   if c == "а":
      c = "б"
   s1 = s1 + c
print (s1)
```

Здесь в цикле for c in s располагаются все символы, входящие в строку s. Каждая поочередно записывается в переменную c. Затем мы проверяем значение переменной: если значение соответствует букве "а", мы заменяем его буквой "б", далее записываем его в новую строку s1 с помощью оператора соединения.

Операции, используемые в строках

Для двух строк есть способ суммирования (конкатенация) и способ умножения строк на число:

```
a = "спа"
b = "сибо"
print(a + b) #спасибо
a = "снег"
print (a * 4) #снегснегснегснег
```

Строки индексируются аналогично массивам: (индексация начинается с 0):

```
a = "привет"
print (a[2]) #и
```

Длина строки определяется с помощью функции `len()`:

```
a = "информатика"
print (len(a)) #11
```

Срезы

Способ разделения части строки [X:Y]

X – индекс начала части, а Y – конец

```
tday = 'morning, afternoon, night'
tday[0:7] #'morning'
```

Применение частей в Python:

```
s = 'spameggs' информатика
s[3:5] #'ор'
s[2:-2] #'форм'
s[-4:-2] #'рм'
s[:6] #'информ'
s[1:] #'нформатика'
s[:] #'информатика'
```

Unicode позволяет ввести все символы, используемые в текстах на разных языках. Ранее мы использовали 256 символов на определенной кодовой странице. Перед строкой следует поставить спецификатор u. На каждый символ отводится 2 байта.

Форматирование в Python – главный инструмент управления строками. Есть несколько способов использования шаблонов и стандартов. При форматировании строк в Python используется стандартный оператор – символ % . Слева от знака процента указываем строку, а справа записывается значение или список значений (*таблица 4*):

```
>>> s = 'Hello %s' % 'world'
>>> s
'Hello world'
>>> s = 'one %s %s' % ('two','three')
>>> s
'one two three'
```

Для преобразования числа в строку используется цифровой спецификатор – символы %d или %f:

```
>>> s = 'one %d %f' % (2, 3.5)
>>> s
'one 2 3.500000'
```

Таблица 4. Форматирование строк

Код	Значения
s	Строчный
c	Символьный
d	Десятичный
i	Целый
u	Десятичный (no longer unsigned)
o	Восьмеричный
x	Шестнадцатеричный
X	Шестнадцатеричный в большом регистре
e	Floating-point exponent, малый регистр
E	Floating-point exponent, большой регистр
f	Floating-point decimal
F	Floating-point decimal
g	Floating-point e или f
G	Floating-point E или F
%	Символьный %

1 Отвечаем на вопросы

1. Что такое символьные строки?
2. Как дается значение символьной строке? Рассмотрите различные способы.
3. Как установить соединение с элементом строки с присвоенным номером?
4. Какой принцип расчета длины строки вы знаете?
5. Какое значение имеет оператор ‘+’?
6. Какие основные типы форматирования можно использовать в строках?
7. Как перевести символическое число в цифровой вид?

2 Думаем и обсуждаем

1. Для чего нужны символьные строки?
2. Почему нельзя сразу записать новое значение в заданной позиции строки? Как можно решить эту проблему?
3. Почему не всегда можно преобразовать строку в цифровой вид?

3 Анализируем и сравниваем

1. Определите операции, используемые в символьных строках.
2. Сравните массивы, списки и строки, определите сходства.
3. Чем символьные строки отличаются от списков?
4. Какие существуют сходства строк с массивами?

4 Выполняем в тетради

1. Запишите операции, используемые в символьных строках, в тетради в виде таблицы.
2. Нарисуйте в тетрадях блок-схему и алгоритм решения приведенных в этом параграфе задач.

5 Выполняем на компьютере

1. Дана строка, длина которой *N*. Создайте программу, которая выводит символы строки в обратном порядке. (Не используйте цикл).
2. Составьте программу, которая заменяет букву `"а"` на букву `"б"` и наоборот в данной символьной строке, написанной всеми заглавными и строчными буквами. При вводе строки `"оглваоОГЛВАО"` результат должен быть в виде `"головаГОЛОВА"`.
3. Создайте программу, которая вводит на экран символьную строку и проверяет, является ли заданное число палиндромом (палиндром – слово или фраза, которые читаются одинаково слева направо и справа налево, например: *довод*, *626*).
4. Введите имя, фамилию и отчество с помощью сочетаний клавиш. В качестве ответа напишите фамилию и инициалы. Например, `"Орманов Мухтар Есенович"` должен быть выведен в виде `"Орманов М.Е."`.

6 Делимся мыслями

Что вы узнали на уроке? Чему научились? Поделитесь своими мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры.

§ 24. Процедуры и функции, используемые для обработки строк

Вспомните!

- *Что такое строки и символы?*
- *Какие операции используются для строк?*

Вы узнаете:

- *о функциях обработки строк;*
- *о методах обработки строк;*
- *о видах методов.*

Термины:

- строка;
- символ;
- функция.

Методы, применяемые для строк

Метод – функция, применяемая для объекта, т. е. предназначенная для строк.

В языке программирования Python существует множество способов работы со строками. Их можно увидеть, выполнив команду `dir(str)`. Для получения личной информации о методах нужно выполнить команду `help(str.название_метода)`. Давайте рассмотрим самые интересные из них.

Методы split () и join ()

Метод `split()` позволяет разделить строку пробелами. В результате появится список слов. Если пользователь вводит в одну строку несколько слов или цифр, то каждая из них должна обрабатываться в программе отдельно, а это невозможно без метода `split()`.

```
>>> s = input()
red blue orange white
>>> s
'red blue orange white'
>>> sl = s.split()
>>> sl
['red', 'blue', 'orange', 'white']
>>> s
'red blue orange white'
```

С помощью метода `split()` можно присвоить возвращенному списку переменную `s`, т. е. `s = s.split()`.

С помощью метода строки `join()` выполняется обратное действие. Хотя он и является методом, перед ним ставится знак «–». А список заключается в апострофы:

```
>>> '-'.join(sl)
'red-blue-orange-white'
```

метод `find()` и `replace()`

Эти методы строки работают с внутренними строками. Метод `find()` ищет внутреннюю строку в строке и возвращает индекс первого элемента, который был найден во внутренней строке. Если внутренняя строка не найдена, он возвращает –1.

```
>>> s
'red blue orange white'
>>> s.find('blue')
4
>>> s.find('green')
-1
```

Метод `replace()` заменяет внутреннюю строку на другую:

```
>>> letters.replace('DA', 'NET')
'ABCNETCFNET'
```

Метод format()

Метод строки `format()` рассматривается при выводе результата на экран с помощью функции `print`:

```
>>> print("This is a {0}. It's {1}.".format("ball", "red"))
This is a ball. It's red.
```

Строки вводятся с помощью стандартной функции ввода `input()`. Вспомните, что существует способ объединения (соединения) двух строк.

В Python любой объект можно заменить на соответствующие строки. Для этого в качестве параметра вам нужно вызвать функцию `str()` к объекту, который преобразуется в строку (*таблица 5*).

С точки зрения языка Python, каждая строка – это объект класса `str`. Для получения объекта другого класса необходимо использовать функцию вывода. Название этой функции должно совпадать с именем класса, к которому принадлежит возвращаемый объект (эта функция – конструктор объектов данного класса). Например: `int` – класс целых чисел. Замена строк на цифры осуществляется через функцию `int()`.

```
s = input()
print(len(s))
t = input()
```

```
number = int(t)
u = str(number)
print(s * 3)
print(s + ' ' + u)
```

***Таблица 5.** Функции и методы обработки строк*

Функция или метод	Описание
`S = 'str'; S = "str";` `S = '''str'''; S = """str"""`	Литерал строки
`S = "s\np\ta\nbbb"`	Экранированный список
`S = r"C:\temp\new"`	Форматированные строки
`S = b"byte"`	Байтовые строки
`S1 + S2`	Конкатенация (объединение строк)
`S1 * 3`	Умножение строк
`S[i]`	Поиск по индексу
`S[i:j:step]`	Вычитание из части
`len(S)`	Длина строки
`S.replace` (шаблон, замена)	Изменить шаблон
`S.split` (символ)	Разделить строку с помощью разделителя
`S.isdigit()`	Проверка наличия цифр в строке
`S.isalpha()`	Проверить, состоит ли строка из букв
`ord` (символ)	Его символ ASCII-кода
`chr` (число)	Код ASCII символа
`S.lstrip([chars])`	Удаление пробелов в начале строки
`S.rstrip([chars])`	Удаление пробелов в конце строки
`S.strip([chars])`	Удаление пробелов в конце и в начале строки
`S.format(*args, **kwargs)`	Форматирование строки
`str.isupper()` `str.islower()`	Проверяет, состоит ли строка только из символов в нижнем и верхнем регистре

Строки можно передавать с такой процедурой и функцией, как параметр, и возвращать как результат функции.

1 Отвечаем на вопросы

1. Что такое метод?
2. Какие методы, применяемые к символам, вы знаете?
3. Как установить соединение с элементом строки с присвоенным номером?
4. Какая функция характеризует длину строки?
5. Какие основные функции, используемые при работе со строками, вы знаете?

2 Думаем и обсуждаем

1. Для чего нужны строчные методы?
2. Для чего применяются функции и процедуры?

3 Анализируем и сравниваем

1. Что вы помните о методах обработки строк?
2. Сравните данные методы и функции, определите их сходства.
3. Чем функции отличаются от процедур?
4. Какие существуют сходства строк с массивами?

4 Выполняем в тетради

1. Запишите функции, используемые в строках, в тетради в виде таблицы.
2. Запишите алгоритмы выполнения и блок-схемы данных в параграфе задач в тетради.

5 Выполняем на компьютере

1. Метод строки `isdigit()` проверяет, состоит ли строка только из цифр. Введите два целых числа и напишите программу, рассчитывающую их сумму. В случае неправильного ввода программа должна не завершаться ошибкой, а выдавать запрос числа. Нельзя использовать особенность try-except.
2. Введите строку, содержащую строчные и заглавные буквы. Необходимо указать эту строку и заменить строчные буквы заглавными и заглавные строчными. Например, если исходная строка – `"aB!cDEf"`, то новая строка – `"Ab!CdeF"`. В коде используется цикл `for` для проверки регистра строки или символа, методы `upper()` и `lower()` (преобразование в верхний и нижний регистр), а также методы `isupper()` и `islower ()`.

6 Делимся мыслями

Что вы узнали на уроке? Поделитесь своими мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите пример.

§ 25. Практикум. Использование процедур и функций для обработки строк

Пример 1. Создание программы, которая выводит символы, индексы которых в строке кратны 3.

Решение: Решаем с использованием расчетного цикла (сложный вид расчета):

```
s = 'процедура'
x=3
l=len(s)//3
for i in range(l):
print(s[x:x+1:3]) #ц у
x+=3
```

Простейший способ – вы можете решить эту задачу с помощью метода разделения на части:

```
s = 'процедура'
print(s[1::3]) #ц у
```

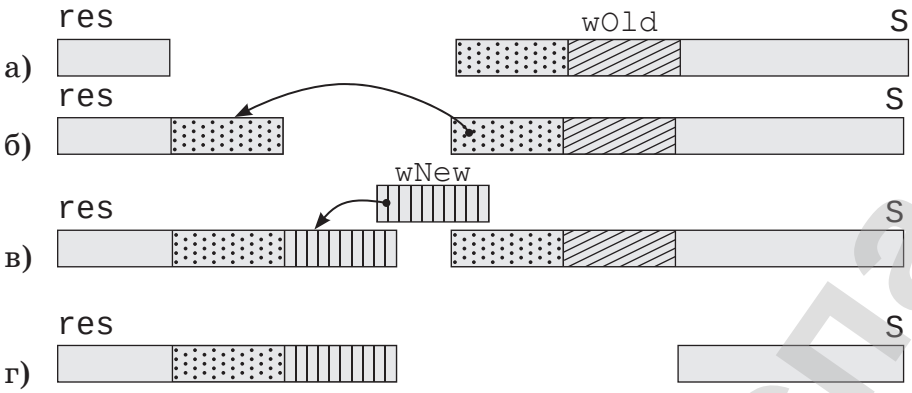
Пример 2. Создание программы процедуры, которая заменяет слова строки `S`, слово `wOld` на слово `wNew` (здесь `wOld` и `wNew` – имена переменных).

Сначала давайте создадим алгоритм решения задачи. Прежде всего нам в голову приходит следующий псевдокод: `while` **слово** `wOld` **находится в строке** `S`:

удалить слово `wOld` из строки
вместо этого вставить слово `wNew`

Слово `wOld` входит в состав `wNew`, например, нужно заменить `"12"` на `"A12B"` (показываем, что это приводит к бесконечному циклу).

Для того, чтобы не попасть в такую ситуацию, стираем обработанную часть из строки `s`, результат записываем в другую символьную строку `res`. К примеру, в нескольких шагах остальной части строки будет встречаться слово `wOld` в строке `s` (*рисунок а*).



Теперь нужно выполнить следующие действия:

1) часть строки `s`, которая находится на левой стороне слова `wOld`, следует продолжить в конце строки `res` (*рисунок б*);
2) слово `wNew` необходимо вписать в конце строки `res` (*рисунок в*);
3) удалить исходную часть из строки `s` со словом `wOld` (*рисунок г*).

Все эти операции будут выполняться до тех пор, пока строка `s` не будет пуста. Если программа не найдет очередное слово, остальная часть строки `s` выходит на результат и цикл заканчивается.

В начале работы алгоритма в поле `res` записывается пустая строка `""`, которая не имеет никаких символов. В следующей таблице представлен алгоритм замены слова `"12"` для строки `"12.12.12"`, который должен быть заменен словом `"A12B"`:

Работа строки `s`	Результат `Res`
`"12.12.12"`	`""`
`".12.12"`	`"A12B"`
`".12"`	`"A12B.A12B"`
`""`	`"A12B.A12B.A12B"`

Теперь можно записать функцию на языке Python. Его параметр – начальная строка `s`, шаблон-строка `wOld` и путь–переключение `wNew`:

```
def replaceAll (s, wOld, wNew):
lenOld = len(wOld)
```

```
res = ""
while len(s) > 0:
p = s.find (wOld)
if p < 0:
return res + s
if p > 0:
res = res + s[:p]
res = res + wNew
if p+lenOld >= len(s):
s = ""
else:
s = s[p+lenOld:]
return res
```

Переменная p – номер первого символа, найденного первым в слове wOld, в переменную lenOld записывается длина этого слова. Если после поиска слова значение p меньше 0, то происходит выход из цикла:

```
if p < 0: res = res + s; return
```

Если p > 0, то на левой стороне слово-шаблон будет иметь определенные символы, которые необходимо вписать в поле res: if p > 0: res = res + s[:p].

Условие p + lenOld > = len(s) означает «модель стоит в конце слова». Остаток строки s – пустая строка. В конце программы результат записывается в первую строку s. Приведем пример использования этой функции:

```
s = "12.12.12"
s = replaceAll (s, "12", "A12B")
print (s )
```

Очень часто используется способ перевода внутренней строки на другую строку. В Python есть встроенная функция, выполняющая эту задачу. Она была опубликована в качестве метода для строчного типа переменных (str) и называется точечной записью:

```
s = "12.12.12"
s = s.replace("12", "A12B")
print (s )
```

Задание 1.

Возьмите следующие части из строки:

- первые 8 символов;
- 4 символа из центра строки;
- 5 символов из конца строки.

Задание 2.

Дана строка, длина которой равна *N*. Выведите символы на экран в обратном порядке (не использовать цикл).

Задание 3.

Дана строка, длина которой равна *N* (*N* – четное число). Выведите индекс четных символов. `a2, a4, a6, … an` на экран в порядке возрастания (не использовать условный оператор).

Задание 4.

Дана строка, длина которой равна N. Сначала выведите на экран четные числа (в порядке возрастания номера), а затем – нечетные: `a2, a4, a6, …, a1, a3, a5, …` (не использовать условный оператор).

Задание 5.

Дана строка. Составьте программу, которая выводит нечетные символы на экран в порядке убывания (см. 0-й символ).

§ 26–27. Работа с файлами

Вспомните!

- *Функции обработки строк.*
- *Методы обработки строк.*
- *Виды методов.*

Вы узнаете:

- *о видах файлов;*
- *о текстовых файлах;*
- *о работе с файлами.*

Термины:

- файл.

В программировании словом «файл» обозначается несколько несовместимых понятий. Во-первых, это операционная система – «часть диска, которому присвоено имя», во-вторых, это абстрактная структура систематического ввода данных, в-третьих, это переменные файлового типа, реализующие структуру данных на конкретном языке программирования.

Таким образом, **файл** – это цепочка символов, записанная в постоянной памяти компьютера. В английском языке слово «file» имеет значение «цепочка», хорошо показывающее внутреннюю структуру любого файла. Файл – это последовательность символов, связанных в определенной последовательности: символы файлов не могут самостоятельно перемещаться с одного места на другое.

«Самостоятельность» файлов не зависит от работы какой-либо программы. Даже если компьютер выключен, файлы сохраняются на жестком диске.

Файлы могут сохранять все, что кодируется:

- тексты программ или входные данные;
- машинные коды выполняемых программ (игры, вирусы, обучающие и сервисные программы и т.д.);
- информацию о каких-либо действиях;
- различные документы и интернет-страницы;
- изображения (фотографии, видео);
- музыку.

В области программирования:

- необходимые файлы, если количество вводимых данных можно ввести вручную;
- если необходимо ввести одну и ту же информацию несколько раз с небольшими изменениями или без каких-либо изменений (например, при восстановлении программы);
- файлы необходимы для того, чтобы сохранить данные о результатах работы программы, полученных при вводе различных данных (при поиске ошибок в программе).

Например, если в нашей программе необходимо получить два или три числа (пять будет много) или строку, состоящую из десяти символов, то можно ввести эти данные вручную с клавиатуры. Допустим, если нам нужно ввести массив 10x10, то количество ошибок при ручном вводе может увеличиться в несколько раз. Теперь нужно устранить эти ошибки: при необходимости записать в файл данные, которые очень легко обрабатывать. Кроме того, созданный файл можно использовать несколько раз (т.к. могут возникнуть несущественные изменения).

Существует два типа общих файлов (различается и работа с ними):

- текстовые файлы неизвестной длины;
- двоичные (бинарные) файлы (сохраняют коды таких данных, как изображения, звуки, видеофильмы).

Этапы работы с файлами:

1) открытие файла;
2) работа с файлом;
3) закрытие файла.

Открытие файла. Метод `open()`

Перед тем, как прочесть и записать что-либо в файл, его следует открыть. В Python для этого используется встроенная функция `open()`. При вызове эта функция создает объект файлового типа, с которым можно будет работать в дальнейшем.

Открыть файл с двумя параметрами в Python также можно с помощью функции `open()`:

- *имя файла* (путь к файлу);
- *режим открытия файла*:
  - *«r»* – открыть для чтения,
  - *«w»* – открыть для записи (если файл существует, его содержимое удаляется),
  - *«a»* – открыть для добавления.

Синтаксис написания функции `open()`:

```
Fin = open ("input.txt")
Fout = open("output.txt", "w")
#работа с файлами
Fout.close()
Fin.close()
```

Работа с текстовыми файлами

Метод `read ()` читает файл с открытой строки.
Синтаксис метода `read()`:

```
my_file.read([count])
```

Дополнительный параметр `count` – это количество байт, которые следует прочитать из открытого файла. Этот метод читает информацию с начала файла и, если параметр `count` не указан, до конца файла.

Например, чтение файла `some.txt`:

```
my_file = open("some.txt")
my_string = my_file.read()
print("Прочитано:")
print(my_string)
my_file.close()
```

Чтение из файла осуществляется двумя способами:
Чтение с помощью метода `readline()`:
Файл input.txt:

```
str1 = Fin.readline() #str1 = 1
str2 = Fin.readline() #str2 = 2
```

Метод `read ()` читает данные до конца файла:
файл input.txt:

```
str = Fin.read()
""
str = 1
2
3
""
```

Метод `write()` предназначен для записи строк в файл:

```
Fout = open("D:/out.txt","w")
Fout.write("hello")
```

Запись в файл можно осуществлять, используя определенный **шаблон вывода**. Например:

```
Fout.write("{:d} + {:d} = {:d}\n".format(x, y,
x + y))
```

В этом случае вместо шаблонов {:d} последовательно подставляются значения параметров метода `format` (сначала `x`, затем `y`, затем `x + y`).

Закрытие файла. Метод `close()`

Метод файлового объекта `close()` автоматически закрывает файл, любая несохраненная информация при этом теряется. Работать с файлом (читать, записывать) после этого нельзя.

Python автоматически закрывает файл, если файловый объект, к которому он привязан, присваивается другому файлу. Однако хорошей практикой будет закрывать файл вручную с помощью команды `close()` (*таблица 6*).

```
my_file = open("some.txt")
print("Имя файла:", my_file.name)
print("Файл закрыт:", my_file.closed)
my_file.close()
print("А теперь закрыт:", my_file.closed)
```

Таблица 6. *Метод закрытия файла*

Название	Функция
`file.closed`	Если файл закрыт, то возвращает значение True
`file.mode`	Возвращает режим доступа к открытому файлу
`file.name`	Возвращает имя файла
`file.softspace`	Возвращает значение False, если вы хотите добавить пробел отдельно при отображении содержимого файла

Список режимов доступа к файлу в программе Python (*таблица 7*).

Таблица 7. Список режимов доступа к файлу

Режим	Функция
r	Открывает файл для чтения. Показатель располагается в начале
rb	Открывает файл для чтения в двоичном формате
r+	Открывает файл для чтения и записи
rb+	Открывает файл для чтения и записи в двоичном формате
w	Открывает файл только для записи
wb	Открывает файл для записи в двоичном формате
w+	Открывает файл для чтения и записи. Показатель располагается в начале. Создает файл с именем «Имя файла»

1 Отвечаем на вопросы

1. Что такое файл?
2. В каких случаях одна переменная файла может быть использована для работы с несколькими файлами, а в каких – нет?
3. Что такое «последовательный доступ к данным»?
4. Что такое файловая переменная?

2 Думаем и обсуждаем

1. Как начать чтение данных с начала файла?
2. Как определить, закончились ли данные в файле?
3. В каких случаях нужно одновременно открыть несколько файлов?
4. Почему другие программы не имеют доступа к открытому в программе файлу, когда он заблокирован в соответствии с правилами?
5. Почему рекомендуется закрывать файлы вручную, если при закрытии программы файлы закрываются автоматически? В каких случаях это имеет значение?
6. Почему, когда нужно работать с файлами, используется переменная файла, а не его имя?

3 Анализируем и сравниваем

1. Определите отличие файлов от функций.
2. Какая разница между текстом и двоичными файлами с внутренним содержанием? Можно ли считать текстовой файл уникальной разновидностью двоичного файла?

4 Выполняем в тетради

Заполните таблицу в тетрадях.

Название	Функция
Метод `close()`	
Метод `readline()`	
Метод `read()`	
Метод `open()`	

5 Выполняем на компьютере

1. Создайте программу, которая будет находить максимальное и минимальное четное положительное число в файле и выводить результат в другой файл. Обратите внимание, что таких чисел в файле может и не быть.
2. В файле столбцом записаны целые числа, начиная с последнего числа. Создайте программу, которая будет записывать числа в порядке возрастания и выводить результат в другой файл.

6 Делимся мыслями

1. Чему вы научились? Поделитесь своими мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите пример.
2. Как можно использовать «принцип сэндвича» при работе с файлами?

§ 28. Практикум. Использование файлов для чтения и записи информации

Пример 1. Открытие нового текстового файла some. Для этого необходимо создать программу, которая будет проверять режим работы и открытость файла.

Программа:

```
my_file = open("some.txt", "w")
print("Имя файла: ", my_file.name)
print("Файл закрыт: ", my_file.closed)
print("Открытый файловый режим: ", my_file.mode)
print("Пробелы: ", my_file.softspace)
```

Результат:

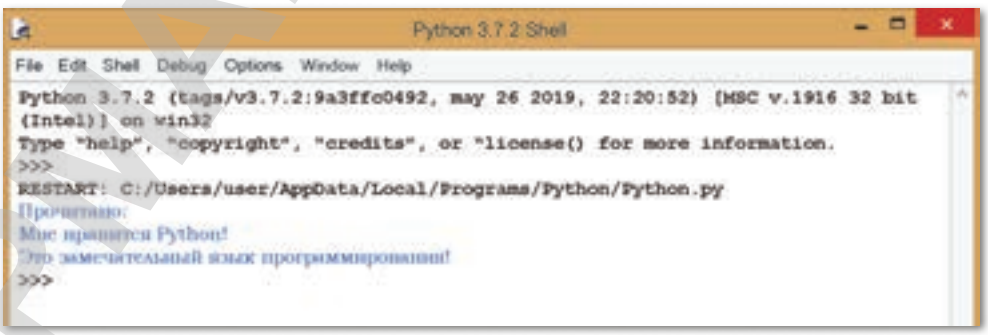
Имя файла: some.txt
Файл закрыт: False
Открытый файловый режим: w

Пример 2. Запись в файл, чтение и вывод на экран.

Программа:

```
my_file = open("some.txt", "w")
my_file.write("Мне нравится Python!\Это замечательный
язык программирования!")
my_file.close()
my_file = open("some.txt")
my_string = my_file.read()
print("Прочитано:")
print(my_string)
my_file.close()
```

Результат:



Задание 1.

В файле записаны целые числа. Найдите максимальное и минимальное число и запишите результат в другой файл.

Задание 2.

В файле в столбец записаны целые числа. Выполните сортировку чисел по возрастанию и запишите результат в другой файл.

Задание 3.

В файле записаны следующие сведения о сотрудниках определенной фирмы:

```
Есжанов 45 бухгалтер
```

Информация о сотрудниках младше 40 лет должна быть записана в текстовый файл.

Задание 4.

В файле записаны сведения о детях в детском саду:

```
Абай Касымович 5 лет
```

Сведения о возрасте самых старших и самых младших детей необходимо записать в текстовый файл.

```
age = int ( s.split()[1] )
if age < 5:
Fout.write (s )
```

Задание 5.

Создайте программу, которая найдет среднее арифметическое введенных в файл чисел, записанных в столбец, и выведет результат в другой файл.

Задание 6.

Создайте программу, которая прочет текст из файла и подсчитает количество слов.

§ 29–30. Методы сортировок

Вспомните!

- *Что такое файл?*
- *Виды файлов.*
- *Текстовой файл.*
- *О работе с файлами.*

Вы узнаете:

- *о понятии «сортировка»;*
- *о видах быстрой сортировки;*
- *что такое пузырьковая сортировка.*

Термины:

- сортировка;
- пузырьковая сортировка.

Пузырьковая сортировка – это метод последовательного сравнения массивов и списков и их сортировки, который меняет местами соседние элементы, если предыдущий элемент больше последующего элемента. При выполнении этого алгоритма элементы с большим значением располагаются в конце списка, а элементы с меньшим значением постепенно перемещаются к началу списка. Образно говоря, это похоже на то, как тяжелые элементы падают на дно, а легкие медленно поднимаются вверх в виде пузырьков.

При пузырьковой сортировке число итераций внешнего цикла определяется –1, т.к. при замене второго элемента первый находится на минимуме и располагается на своем месте.

Количество итераций во внутреннем цикле зависит от иерархического номера внешнего списка, так как конец списка уже отсортирован, и эти элементы не нужно сортировать заново.

Например, дан список [6, 12, 4, 3, 8].

В первой итерации внешнего цикла число 12 движется к концу. Для этого требуется 4 сравнения во внутреннем цикле:

- 6 > 12? Нет
- 12 > 4? Да. Меняем местами
- 12 > 3? Да. Меняем местами
- 12 > 8? Да. Меняем местами

В результате: [6, 4, 3, 8, 12].

Во второй итерации внешнего цикла число 8 перемещается на предыдущее место. Для этого требуется 3 сравнения:

- 6 > 4? Да. Меняем местами
- 6 > 3? Да. Меняем местами
- 6 > 8? Нет

В результате: [4, 3, 6, 8, 12].

При третьей итерации внешнего цикла последние два элемента удаляются. Внутреннее число итераций равно двум:

- 4 > 3? Да. Меняем местами
- 4 > 6? Нет

В результате: [3, 4, 6, 8, 12].

При четвертой итерации внешнего цикла для сопоставления остаются только первые два элемента, поэтому число итераций внутреннего цикла будет равным 1:

- 3 > 4? Нет

В результате: [3, 4, 6, 8, 12].

Этот метод основан на сопоставлении двух последовательных элементов, расположенных в простом виде. Если элементы массива расположены в вертикальном направлении, то их можно представить в виде пузырьков в стеклянной емкости. В этом случае каждый пузырек поднимается до соответствующей своему весу высоты и располагается там. Название пузырьковой сортировки основано на этой аналогии.

Значение сортировки:

1. Сравниваются первые 2 элемента. Если 1-й элемент меньше 2-го, то они меняются местами.
2. Сравниваются 2-й и 3-й элементы, 3-й и 4-й элементы и т. д., в случае необходимости они меняются местами. В результате наименьший элемент перемещается в начало.

При полном выборочном размещении элементов массива данное действие выполняется $(n - 1)$ раз. Где n – количество элементов массива.

В каждом повторении можно ввести переменное значение, характеризующее выполнение замены, и следить за окончанием процесса.

Иногда этот метод называют **методом сортировки путем замены**. Количество сравнений в этом методе равно $n(n - 1)/2$.

Примечание:

1. Если на каком-либо шаге не выполняется замена, то работа алгоритма должна быть прекращена.
2. На текущем этапе должно запомниться минимальное значение индекса массива, на котором была произведена замена. Так как исходные элементы массива до данного индекса отсортированы, сравнение данного индекса и соседнего элемента массива не требуется.
3. В случае, если малое значение помещается в необходимое место после одной замены, то элементы с большим значением могут быть помещены в необходимое место только после полного выполнения алгоритма.

Встроенные функции

mas.reverse() – стандартный метод сортировки элементов массива в обратном порядке; ***mas2 = sorted (mas1)*** – утвержденная функция для сортировки массивов (списков).

Вид использования цикла `while` для поиска массива:

```
import random #добавить библиотеку
from random import randint
n=10; x=5
mas = [randint(1,10) for i in range(n)] #инициализация массива
i = 0
while i < n and mas[i] != x: #если элемент не равен
   i + = 1
if i < n:
   print ("mas[", i, "]=", x, sep = "")
else:
print ("Не найден!" )
```

Вид использования цикла `for` для поиска массива:

```
import random
from random import randint
n = 10; x = 5
mas = [randint(1,10) for i in range(n)]
for i in range (n):
   if mas[i] = = x:
      nomer = i
      break
if nomer > = 0:
   print ("mas[", nomer, "]=", x, sep = "")
else:
   print ("Не найден!")
```

В данном случае сохраняется число и значение элемента массива, найденного в переменной `nomer`.

Однако уникальное свойство цикла `for` на языке Python – это блок `else`, который выполняется, когда оператор `break` не используется в цикле.

Рассмотрим второй способ поиска, т.к. он гораздо проще:

```
import random
from random import randint
n = 10;x = 5
mas = [randint(1,10) for i in range(n)]
```

```
nomer = -1
for i in range (n):
   if mas[i] = = x:
      print ("mas[", i, "]=", x, sep = "")
      break
else:
   print ("Не найден!")
```

1 Отвечаем на вопросы

1. В чем смысл сортировки?
2. В каких случаях используется пузырьковая сортировка?
3. В чем основная идея метода быстрой сортировки?
4. Какие встроенные функции для сортировки массивов в Python вы знаете?

2 Думаем и обсуждаем

1. Для чего нужна пузырьковая сортировка?
2. Почему при пузырьковой сортировке легкие элементы располагаются выше?
3. Для чего нужен цикл в данных методах сортировки?

3 Анализируем и сравниваем

1. Чем быстрая сортировка отличается от пузырьковой?
2. Сравните метод выбора и метод пузырьковой сортировки.

4 Выполняем в тетради

1. Перечислите все виды сортировок и запишите в тетради.
2. Напишите функции, используемые при пузырьковой сортировке.

5 Выполняем на компьютере

Напишите программу, сортирующую множество чисел, а затем найдите наибольшее число, которое возникло в массиве несколько раз. Не используйте встроенные функции.

6 Делимся мыслями

Что нового вы узнали на уроке? Чему вы научились? Поделитесь своими мыслями с друзьями. В каких повседневных ситуациях можно применить знания, полученные на уроке? Приведите примеры.

§ 31–32. Практикум. Реализация алгоритмов сортировки для решения практических задач

Пример 1. Создание программы пузырьковой сортировки с помощью цикла `for`.

Решение:

```
from random import randint
N = 10
a = [ ]
for i in range(N):
    a.append(randint(1, 99))
print(a)
for i in range(N-1):
    for j in range(N-i-1):
        if a[j] > a[j + 1]:
            a[j], a[j + 1] = a[j + 1], a[j]
print(a)
```

Результат выполнения кода:

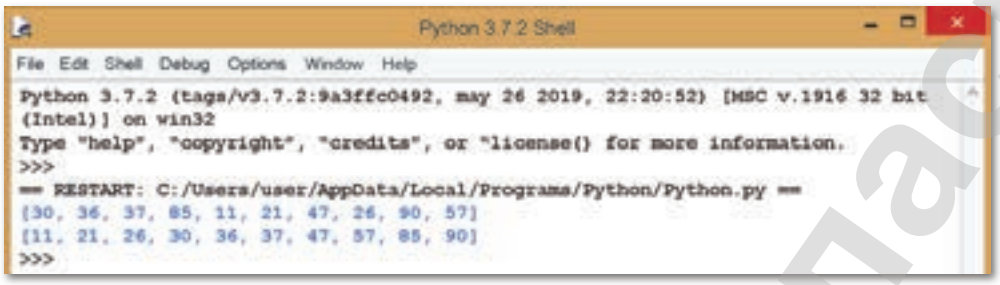
```
[63, 80, 62, 69, 71, 37, 12, 90, 19, 67]
[12, 19, 37, 62, 63, 67, 69, 71, 80, 90]
```

Пример 2. Создание программы пузырьковой сортировки с помощью цикла `while`.

Решение:

```
from random import randint
N = 10
a = []
for i in range(N):
    a.append(randint(1, 99))
print(a)
i = 0
  while i < N - 1:
j = 0
  while j < N - 1 - i:
    if a[j] > a[j+1]:
      a[j], a[j + 1] = a[j + 1], a[j]
    j + = 1
  i + = 1
print(a)
```

Результат:

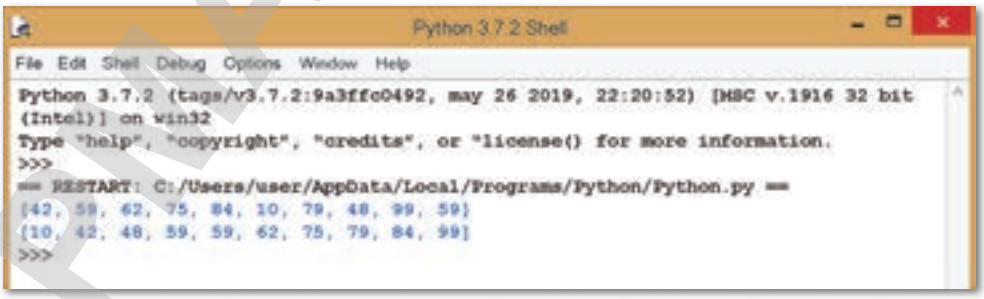


Пример 3. Создание программы сортировки с использованием функции пузырьковой сортировки в Python.

Решение:

```
from random import randint
def bubble(array):
   for i in range(N-1):
      for j in range(N-i-1):
        if array[j] > array[j+1]:
             buff = array[j]
             array[j] = array[j + 1]
             array[j + 1] = buff
N = 10
a = []
for i in range(N):
   a.append(randint(1, 99))
print(a)
bubble(a)
print(a)
```

Результат:

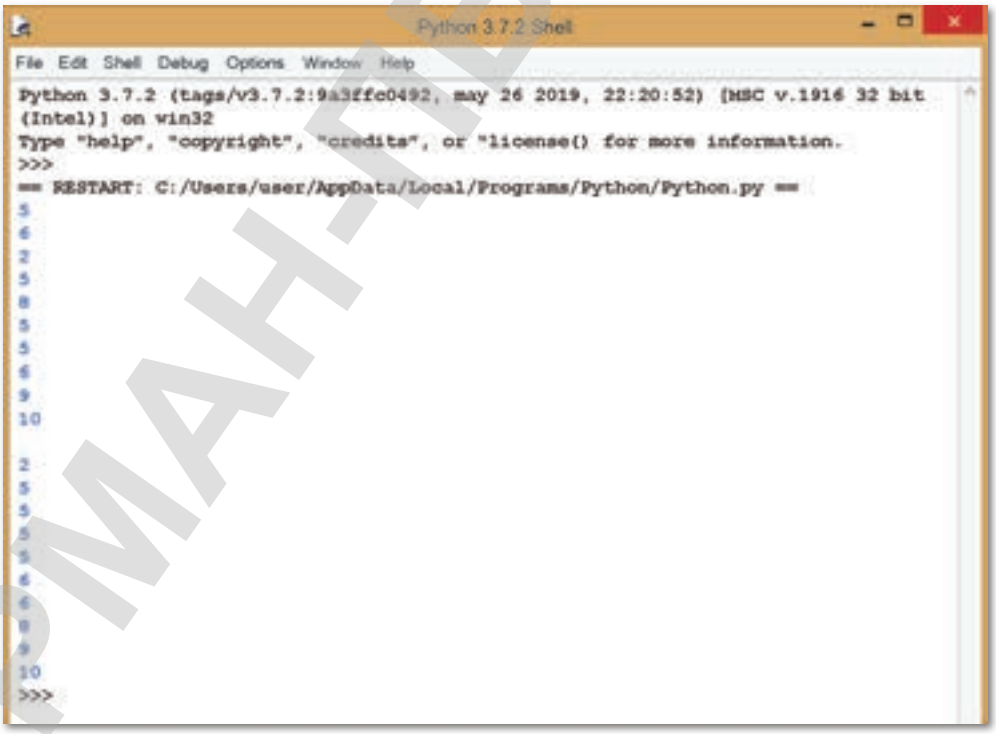


Пример 4. Создание программы в Python, осуществляющую сортировку массива с помощью пузырькового метода.

Решение:

```
import random
from random import randint
n = 10
mas = [randint(1,10) for i in range(n)]
for i in range(n):
      print(mas[i],sep = "")
print(" ")
for i in range(n-1):
        for j in range(n-2, i-1 ,-1):
        if mas[j+1] < mas[j]:
        mas[j], mas[j + 1] = mas[j + 1], mas[j]
for i in range(n):

   print(mas[i],sep = "")
```

Результат:

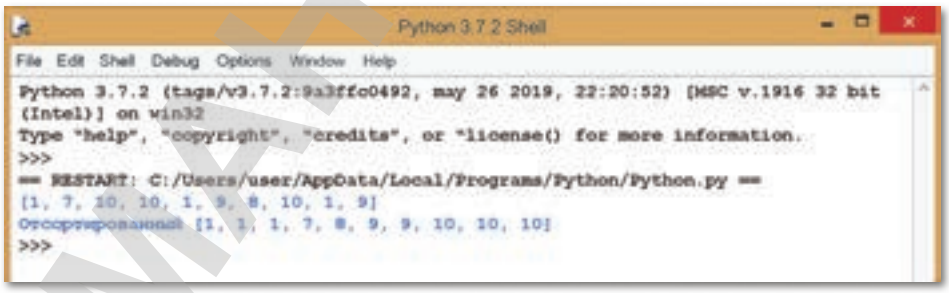


Пример 5. Создание программы быстрой сортировки массива.

Решение:

```
import random
from random import randint
#процедура
def qSort (A, nStart, nEnd):
   if nStart >= nEnd: return
        L = nStart; R = nEnd
        X = A[(L+R)//2]
        while L <= R:
             while A[L] < X: L + = 1 #деление
   while A[R] > X: R -= 1
   if L < = R:
        A[L], A[R] = A[R], A[L]
        L + = 1; R -= 1
      qSort (A, nStart, R) #рекурсивный вызов
   qSort (A, L, nEnd)
N = 10
A = [randint(1,10) for i in range(N)]
print(A)
#вызов процедуры
qSort (A, 0, N-1)
print('отсортированный', A)
```

Результат:



Задание 1.

Необходимо написать программу, которая будет сортировать массив с первого номера массива в порядке возрастания (с помощью быстрой сортировки).

Задание 2.

Напишите программу «каменного метода» сортировки, в которой самый «тяжелый элемент» будет выпадать в конец массива.

Задание 3.

Дан массив. С помощью сортировки найдите трехзначные числа.

Задание 4.

Заполните массив случайными числами с интервалом 0...4 и выведите на экран номер всех элементов, равных x (вводится с клавиатуры).

Задание 5.

Напишите программу, которая выводит на экран числа заданного массива элементов в порядке возрастания, не изменяя его. Воспользуйтесь запасным массивом чисел.

Задание 6.

Напишите программу, которая сортирует список и находит сумму различных чисел, не используя встроенные функции.

Задание 7.

Дан массив. Ряд последовательных одинаковых элементов серии и длина серии – это количество элементов. Составьте два новых массива, напишите длину всех серий в один из них, и напишите значения элементов, которые преобразуют эти серии, во второй.

Задание 8.

Напишите вариант пузырькового метода, который прекратит работу, если на следующем этапе внешнего цикла нет повторной постановки. Не используйте встроенные функции.

§ 33–34. Алгоритмы на графах

Вспомните!

- *Что такое сортировка?*
- *О видах быстрой сортировки.*
- *О пузырьковой сортировке.*

Вы узнаете:

- *о понятии «граф»;*
- *о видах графов;*
- *об алгоритмах поиска на графах;*
- *что такое поиск в глубину;*
- *что такое поиск в ширину.*

В последнее время теория графов широко используется в различных отраслях науки и техники. Данная теория получила быстрое развитие с созданием электронно-вычислительной техники, которая позволяла решить многие задачи алгоритмизации.

Граф – это совокупность двух конечных множеств: множества точек и множества линий, попарно соединяющих некоторые из этих точек. Точки называются *вершинами (узлами) графа*. Множество линий, соединяющих вершины графа, *называются ребрами (дугами) графа*.

Виды графов даны в *схеме 3*.

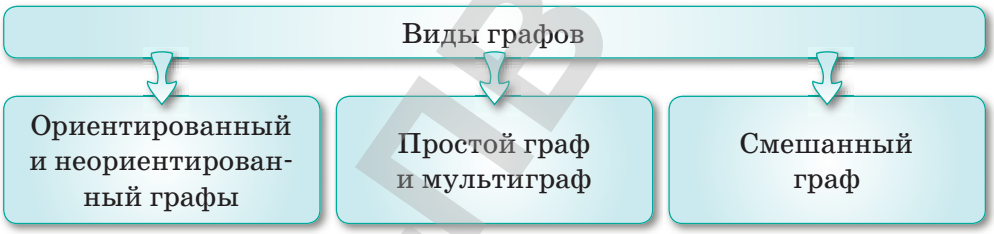


Схема 3. *Виды графов*

Ориентированный граф – граф, у которого все ребра ориентированы, т.е. ребрам которого присвоено направление.

Неориентированный граф – граф, у которого все ребра не ориентированы, т.е. ребрам которого не задано направление.

Смешанный граф – граф, содержащий как ориентированные, так и неориентированные ребра.

Петлей называется ребро, соединяющее вершину саму с собой. Две вершины называются *смежными*, если существует соединяющее их ребро. Ребра, соединяющие одну и ту же пару вершин, называются *кратными*.

Простой граф – граф, в котором нет ни петель, ни кратных ребер.

Мультиграф – граф, у которого две любые вершины соединены более чем одним ребром.

Маршрутом графа называется конечная чередующаяся последовательность смежных вершин и ребер, соединяющих эти вершины.

Маршрут называется **открытым**, если его начальная и конечная вершины различны, в противном случае он называется **замкнутым**. Маршрут называется **цепью**, если все его ребра различны. Открытая цепь называется **путем**, если все ее вершины различны. Замкнутая цепь называется **циклом**, если различны все ее вершины, за исключением концевых. Граф называется **связным**, если для любой пары вершин существует соединяющий их путь.

Вес вершины – число (действительное, целое и рациональное), соответствующее данной вершине (интерпретируется как стоимость, пропускная способность и т. д.).

Вес (длина) ребра – число или несколько чисел, которые интерпретируются по отношению к ребру как длина, пропускная способность и т. д.

Взвешенный граф – граф, каждому ребру которого соответствует некое значение (вес ребра).

Выбор структуры для хранения графа в памяти компьютера имеет принципиальное значение при разработке **эффективных алгоритмов**. Рассмотрим несколько **способов представления** графа.

Пусть задан граф (*рис. 18*), количество вершин которого равно n, а количество ребер – m. Каждое ребро и каждая вершина имеют вес – целое положительное число. Если граф не является помеченным, то считается, что вес равен единице.

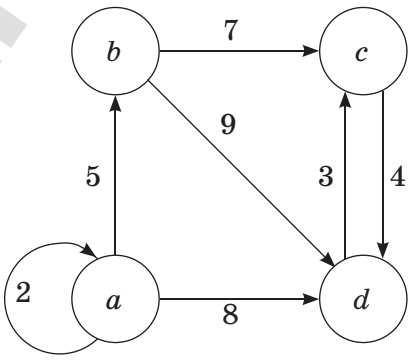


***Рис. 18**. Пример графа*

Список ребер – множество, образованное парами смежных вершин. Для его хранения обычно используют одномерный массив размером m, содержащий список пар вершин, смежных с одним ребром графа. По сравнению с другими способами список ребер более удобен для реализации различных алгоритмов на графах (*таблица 8*).

Таблица 8. *Список ребер графа*

a	*a*	*a*	*b*	*b*	*c*	*d*
a	*b*	*d*	*c*	*d*	*d*	*c*
2	5	8	7	9	4	3

Матрица смежности – двумерный массив размерности *n* x *n*, значения элементов которого характеризуются смежностью вершин графа. При этом значению элемента матрицы присваивается количество ребер, которые соединяют соответствующие вершины. Данный способ используется, когда нужно проверить смежность или найти вес ребра по двум заданным вершинам (*таблица 9*).

Таблица 9. *Матрица смежности графа*

	a	*b*	*c*	*d*
a	2	5	0	8
b	0	0	7	9
c	0	0	0	4
d	0	0	3	0

Матрица инцидентности – двумерный массив размерности *n* x *m*, в котором указываются связи между инцидентными элементами графа (ребро и вершина). Столбцы матрицы соответствуют ребрам, строки – вершинам. Ненулевое значение в ячейке матрицы указывает на связь между вершиной и ребром. Данный способ является самым емким для хранения и облегчает нахождение циклов в графе (*таблица 10*).

Таблица 10. *Матрица инцидентности графа*

	a	*b*	*c*	*d*
(a, a)	2	0	0	0
(a, b)	0	5	0	0
(a, d)	0	0	0	8
(b, c)	0	0	7	0
(b, d)	0	0	0	0
(c, d)	0	0	0	4
(d, c)	0	0	3	0

В графах существует множество алгоритмов, в основе которых лежит такой систематический перебор вершин графа, при котором каждая вершина просматривается (посещается) ровно один раз. Поэтому важной задачей является нахождение лучших методов поиска в графе.

Под **обходом графов (поиском на графах)** понимается процесс систематического просмотра всех ребер или вершин графа с целью нахождения ребер или вершин, удовлетворяющих некоторому условию.

При решении многих задач, использующих графы, необходимы эффективные методы регулярного обхода вершин и ребер графов. Стандартные и наиболее распространенные методы приведены в *схеме 4*.

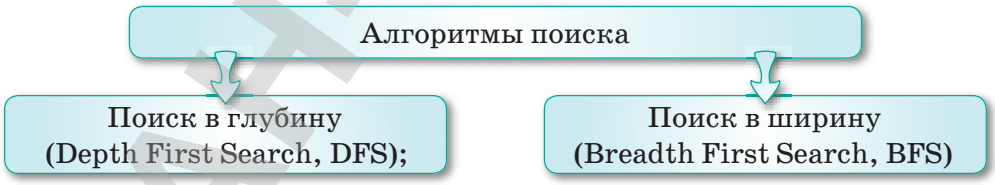


Схема 4. *Алгоритмы поиска*

Эти методы чаще всего рассматриваются на ориентированных графах, но они применимы и для неориентированных, ребра которых считаются двунаправленными. Алгоритмы поиска в глубину и в ширину лежат в основе решения различных задач обработки графов, например, проверки связности, ацикличности, вычисления расстояний между вершинами и других.

Поиск в глубину

При поиске в глубину посещается первая вершина, затем необходимо идти вдоль ребер графа до попадания в тупик. Вершина графа является тупиком, если все смежные с ней вершины уже посещены. После попадания в тупик нужно вернуться назад вдоль пройденного пути, пока не будет обнаружена вершина, у которой есть непосещенная смежная вершина, и двигаться в этом новом направлении. Процесс считается завершенным при возвращении в начальную вершину, когда все смежные с ней вершины уже посещены.

Таким образом, основная идея поиска в глубину заключается в том, что, когда возможные пути по ребрам, выходящим из вершин, разветвляются, нужно сначала полностью исследовать одну ветку и только потом переходить к другим веткам (если они останутся нерассмотренными).

Алгоритм поиска в глубину

Шаг 1. Всем вершинам графа присваиваются значения. Выбирается первая вершина и помечается, как «посещенная».

Шаг 2. Для последней вершины, помеченной как «посещенная», выбирается смежная вершина, являющаяся первой, помеченной, как «непосещенная», и ей присваивается значение «посещенная». Если таких вершин нет, то берется предыдущая помеченная вершина.

Шаг 3. Повторить шаг 2 до тех пор, пока все вершины не будут помечены, как посещенные (*рис. 19*).

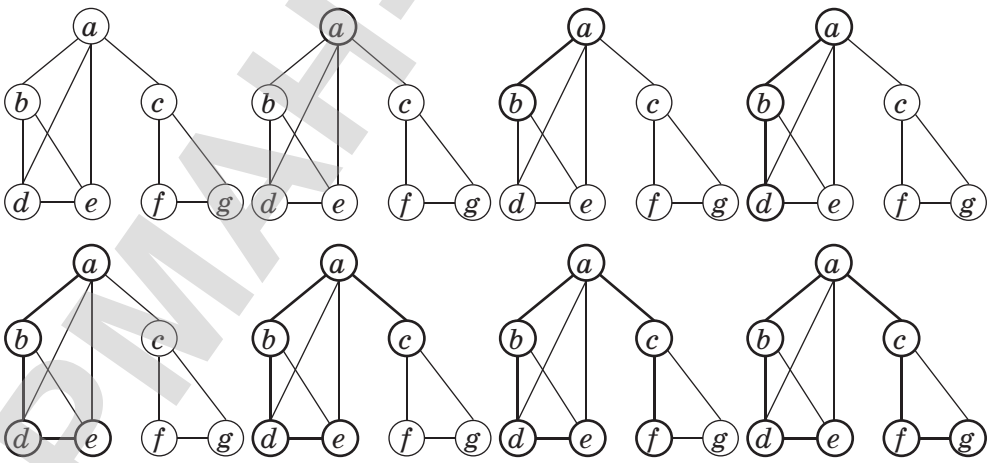


Рис. 19. *Демонстрация алгоритма поиска в глубину*

Описание функции алгоритма поиска в глубину:

```
#
# 2--0--6--7 1--9 5
# | | |
# 3--4 8
#
n = 10 #число вершин
adj_list = [[2, 4, 6],
            [9],
            [0, 3],
            [2, 4],
            [0, 3],
            [],
            [0, 7, 8],
            [6],
            [6],
            [1]]
s = 0
visited = [False] * n #массив "просмотра вершин"

def dfs(v):
visited[v] = True
for w in adj_list[v]:
if visited[w] == False: #просмотрена ли текущая
соседняя вершина?
dfs(w)

dfs(s)
print (visited. count(True))
```

Также часто используется нерекурсивный алгоритм поиска в глубину. В этом случае рекурсия заменяется на стек. Как только вершина просмотрена, она помещается в стек, а использованной она становится, когда больше нет новых вершин, смежных с ней.

Поиск в ширину

При поиске в ширину после посещения первой вершины посещаются все смежные с ней вершины. После этого

посещаются все вершины, находящиеся на расстоянии двух ребер от начальной. При каждом новом шаге посещаются вершины, расстояние от которых до начальной на единицу больше предыдущего. Чтобы предотвратить повторное посещение вершин, необходимо вести список посещенных вершин. Для хранения временных данных, необходимых для работы алгоритма, используется *очередь* – упорядоченная последовательность элементов, в которой новые элементы добавляются в конец, а старые удаляются из начала.

Таким образом, основная идея поиска в ширину заключается в том, что сначала исследуются все вершины, смежные с начальной вершиной (вершина, с которой начинается обход). Эти вершины находятся на расстоянии 1 от начальной. Затем исследуются все вершины на расстоянии 2 от начальной, затем все на расстоянии 3 и т.д. Обратите внимание, что при этом для каждой вершины сразу находится длина кратчайшего маршрута от начальной вершины.

Алгоритм поиска в ширину

Шаг 1. Всем вершинам графа присваивается значение «непосещенная». Выбирается первая вершина, которая помечается, как «посещенная», и заносится в очередь.

Шаг 2. Посещается первая вершина из очереди (если она не помечена, как «посещенная»). Все ее соседние вершины заносятся в очередь. После этого она удаляется из очереди.

Шаг 3. Шаг 2 повторяется до тех пор, пока очередь не опустеет (*рис. 20*).

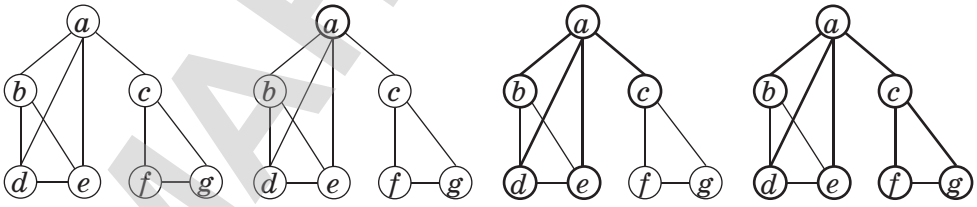


Рис. 20. *Демонстрация алгоритма поиска в ширину*

Описание функции алгоритма поиска в ширину:

```
adj = [
#соседние вершины
     [1,3], #0
```

```
        [0,3,4,5], #1
        [4,5], #2
        [0,1,5], #3
        [1,2], #4
        [1,2,3] #5
]

level = [-1] * len(adj)
#список вершин по уровням
def bfs(s):
        global level
        level[s] = 0
#начальный уровень вершины
        stack = [s]
    #поставить вершины в очередь
        while stack:
          v = stack.pop(0)
#удаление вершин
        for w in adj[v]:
#просмотр вершин v
        if level[w] is -1:
#если вершина не предусмотрена, тогда
        stack.append(w)
#поставить вершину в очередь
        level[w] = level[v] + 1
#считать уровень вершина
for i in range(len(adj)):
    if level[i] is -1:
        bfs(i)
    #в случае наличия нескольких взаимосвязанных ком-
понентов
    print(level[2])
```

1 **Отвечаем на вопросы**

1. Что такое граф?
2. Какие виды графов вы знаете?
3. Где можно использовать маршрут графа в повседневной жизни?
4. Как создается список ребер?
5. Как строятся матрицы инцидентности и смежности графа?

6. Какие типы алгоритма поиска в графах вы знаете?
7. Как работает алгоритм поиска в глубину?
8. Как работает алгоритм поиска в ширину?

2 Думаем и обсуждаем

1. Почему необходимо использовать алгоритмы поиска в графах при выполнении различных расчетов?
2. Для чего используется очередь?
3. Для чего нужны цикл, цепь и путь?

3 Анализируем и сравниваем

1. Используя дополнительные источники, напишите пошаговые алгоритмы поиска в глубину и поиска в ширину и сравните их между собой.

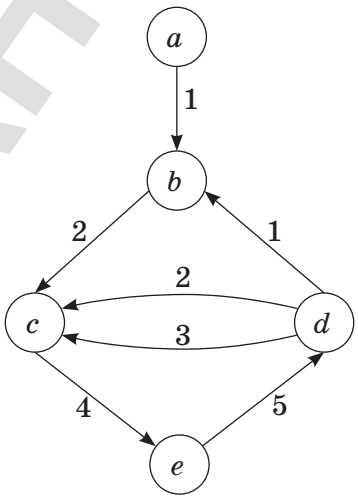
Алгоритм поиска в глубину	Алгоритм поиска в ширину

2. Какие различия смешанных и взвешенных графов вы знаете?

4 Выполняем в тетради

По заданному графу создайте:

1) список ребер;
2) матрицу смежности;
3) матрицу инцидентности.



5 Выполняем на компьютере

Задание 1. На основании приведенного в § 33–34 материала, реализуйте программу, в которой выполняется алгоритм обхода графа на основе поиска в глубину.

Задание 2. На основании приведенного в § 33–34 материала, реализуйте программу, в которой выполняется алгоритм обхода графа на основе поиска в ширину.

Задание 3. Используйте обход графа в ширину для определения всех вершин графа, находящихся на фиксированном расстоянии d от заданной вершины.

Инструкция по выполнению заданий:

Каждое задание необходимо решить в соответствии с изученными алгоритмами обхода графа, реализовав программный код на языке Python. Рекомендуется воспользоваться теоретическими материалами, где подробно рассматриваются описания алгоритмов обхода графа, примеры разработки функций, реализующих алгоритмы обхода графа на языке Python. Программу для решения каждого задания необходимо разработать методом процедурной абстракции, используя функции. Этапы решения сопроводить комментариями в коде. В конце задания следует отразить разработку и обоснование математической модели решения задачи, представить результаты тестирования программ.

Каждое задание следует реализовать по этапам в соответствии с приведенной последовательностью:

- изучить словесную постановку задачи, выделив при этом все виды данных;
- сформулировать математическую постановку задачи;
- выбрать метод решения задачи, если это необходимо;
- разработать графическую схему алгоритма;
- записать разработанный алгоритм на языке Python;
- разработать контрольный тест программы;
- проверить программу;
- представить отчет по работе.

6 Делимся мыслями

Как вы думаете, распространяются ли понятия «поиск в глубину» и «поиск в ширину» на несвязный граф?

Раздел IV

WEB-ПРОЕКТИРОВАНИЕ

Цели обучения:

- использовать HTML-теги при разработке web-страниц;
- использовать CSS при разработке web-страниц;
- применять HTML-теги для вставки мультимедиа объектов на web-страницу;
- использовать готовые скрипты при разработке web-страниц.

§ 35–36. Способы разработки web-сайтов. HTML

Вспомните!

- *Что такое граф?*
- *Как выполняется алгоритм на графах?*

Вы узнаете:

- *что такое HTML;*
- *что такое тег и его разновидности, атрибуты тегов;*
- *что такое web-сайт;*
- *из чего состоит web-сайт;*
- *этапы планирования web-сайта.*

Тексты, читаемые на компьютере или на других электронных устройствах, называются **гипертекстами**. Впервые термин «гипертекст» был введен в оборот в 1963 году американским социологом и философом Тедом Нельсоном. А принятие в 1986 году международного стандарта ISO-8879 «Standart Generalized Markup Language» послужило основой для появления языка HTML. Язык гипертекстовой разметки (Hyper Text Markup Language) – это основной язык разметки, используемый в web-браузерах для вывода web-страниц, а также любой другой информации, то есть с помощью тегов, которые помещаются в документ, описывается логическое строение документа, осуществляется форматирование документов и вставка объектов. Одна из общих особенностей всех данных Интернета, то есть всех web-документов, заключается в том, что большинство из них написаны на языке HTML. Хотя создание web-документов в HTML и является схожим с программированием, это не простой язык программирования.

HTML – язык гипертекстовой разметки. Он определяет набор правил для отображения обычных текстов в виде web-страниц.

При работе с языком HTML можно использовать обычное приложение Блокнот для создания web-страницы или сайта. Код пишется в Блокноте, далее указывается место сохранения: **Файл** ⇒ **Сохранить как ...** ⇒ название файла **name*.html**, для типа файла выбираем **Все файлы**, в разделе кодировки выбираем UTF-8 и нажимаем на кнопку **Сохранить**. Результат смотрим, открыв документ в браузере (*рис. 21*).

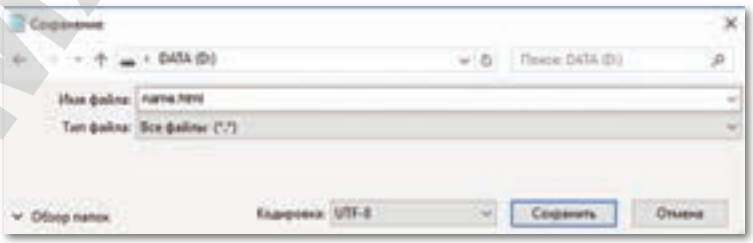


Рис. 21. *Сохранение html-документа*

Что такое тег?

Команды, используемые на языке HTML, называются **тегами**. Они делятся на две группы: парные и непарные.

Теги считаются **парными**, если открывается один тег, а другой его закрывает. Например, <html> открывает работу тега, а следующий тег </html> закрывает ее.

При использовании **непарных** тегов тег будет открываться и использоваться без необходимости его закрытия (*таблица 11*).

Таблица 11. *Применение тегов*

Название тега	Использование	Пример
<HTML> ... </HTML>	Эти теги обязательно должны быть на странице. Они сообщают браузерам и поисковым системам, что это страница html	<html> <head> ... Тематические теги ... </head> <body> ... Основное тело страницы ... </body> </html>
<body> </body>	Между этими тегами отображается все содержимое страницы	
<head> </head>	Внутри этих тегов должны быть размещены все теги заголовков	
<title> </title>	Между этими тегами записывается заголовок страницы, который отображается в верхней части браузера	<html> <head> <h2 align="center"> Заголовок</h2> <title> Заголовок 1.1. </title> </head> <body> <p align="center"> текст 1
 	С помощью этих тегов текст выравнивается по центру	
 	Эти теги предназначены для изменения шрифта, фона и т. д. Все, что касается форматирования текста, может быть настроено в одном теге	
	Текст, написанный между тегами и , выделяется жирным начертанием	

Название тега	Использование	Пример
	Меняет шрифт на курсив	<font size="6" текст 2 </h1> </p> </body> </html>
<u></u>	Подчеркивает текст	
<s></s>	Перечеркивает текст	
	Теги <h1>...<h6> являются одним из классов тегов заголовка. Обычно это может быть название страницы	<h1>html образец создания страницы </h1> <h2> Заголовок 1</ h2> ... <h2> Заголовок 1.1</h2> <h3> Заголовок 2</ h3> ... и т.д.
 	Единственный тег, который не требует закрытия. Он перемещает текст на следующую строку	<html> <body> ... Заголовок текст текст </body> </html>
<img alt= "Ссылка" src="URL_ Картинка">	Единственный тег, который показывает изображение. Параметр src отображает адрес изображения (вместо URL_Image необходимо прикрепить адрес, по которому хранится изображение)	<html> <body> ... <img src="https:// www.pinterest. com/pin/4763970 8537347914/_orig. jpg"> ... </body> </html>
<hr>	Непарный тег, который чертит горизонтальную линию	

Название тега	Использование	Пример
текст _ссылка 	Тег для создания ссылок	<a href="stranica_50.html"> stranica_50.html </ a> Можно записать полный адрес страницы
background= "URL"	Определяет фоновое изображение. Вместо URL будет записан адрес фонового изображения	<html> <body> <table align="center" width="100%" border="1"> <tr> <td colspan="2"> Пример таблицы </td> </tr> <tr> <td> Столбец 1 </td> <td> Столбец 2 </td> </tr> </table> </body> </html>
bgcolor= "цвет"	Определяет цвет таблицы. В качестве цвета вы можете выбрать любой цвет	
border= "цифра"	Определяет толщину рамки вокруг изображения	
<table></ table>	Тег создания таблицы. Между этими тегами располагается таблица	
<tr>	Создает новую строку	
<td>	Создает новый столбец	

Выполните примеры на основе тегов HTML, приведенных в таблице, и просмотрите результаты в окне браузера.

Web-документ – это текстовой файл, дополненный тегами языка HTML, который, соединяя друг с другом тексты, позволяет вам их обозначить.

Web-страница состоит из трех основных составляющих:

<HTML>
<HEAD>
</HEAD>
<BODY>

Между этими тегами содержание страницы будет доступно пользователю сайта.

</BODY>
</HTML>

Web-страницы могут и будут отличаться друг от друга, но все они состоят из стандартных компонентов (*схема 5*).

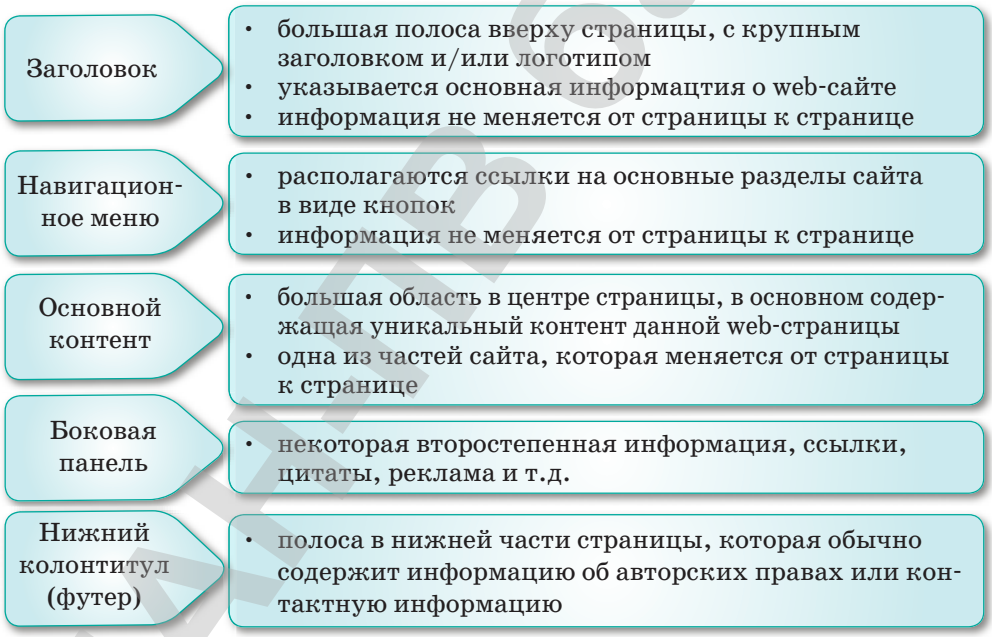


Схема 5. Структра web-документа

Web-сайт – это совокупность web-страниц, объединенных одной темой и взаимосвязанных между собой гиперссылками. Web-страницы обычно размещаются на сервере в виде папки с именем и адресом web-сайта. Добавлять web-страницы к разработанным сайтам несложно. Сайты могут быть большими, сложными, иерархическими. Поскольку вся информация, вносимая в них,

часто изменяется, время от времени необходимо обрабатывать сайт и вводить новые данные. Для быстрого поиска таких данных на главной странице вводятся ключевые слова – гиперссылки.

В сущности, web-сайт – это информационная система, состоящая из двух основных компонентов:

1) Компонент отображения (front end), который включает в себя представление содержимого (размер страниц, графика, аудио, текст).
2) Компонент реализации (back end), неотображаемые сценарии, являющиеся основой для компонента отображения, зависящие от эффективности связи текущих кодов с серверными компонентами.

Этапы проектирования web-сайта – определение объема, функциональности сайта и т. д. (*схема 6*).

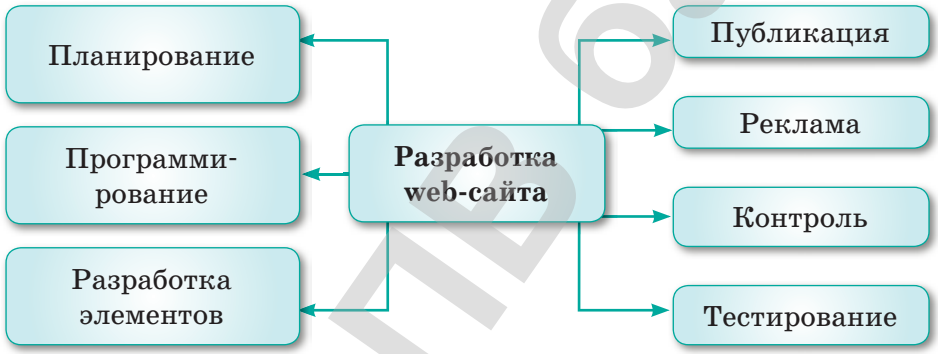


Схема 6. Этапы проектирования web-сайта

Этапы планирования сайта:

1. Определить основную задачу сайта.
2. Определить, какая информация должна быть размещена на сайте.
3. Собрать необходимую информацию.
4. Разработать дизайн сайта.
5. Продумать логическую структуру сайта.
6. Организовать физическую структуру сайта, т. е. выделить отдельные файлы, составляющие сайт, в папки.
7. Проверить, все ли правильно.

На этапе планирования необходимо решить следующие вопросы:

1) Где будет располагаться сайт.
2) Кто будет входить в аудиторию сайта.

3) Какая информация будет публиковаться.
4) В каком виде организуется общение с пользователями.

На этапе разработки элементов предусматривается реализация сайта в виде программного продукта:

1) Создание навигационной структуры.
2) Разработка дизайна страницы.
3) Подбор текстовой и визуальной информации для заполнения страницы.

Программирование. Суть этого этапа заключается в форматировании сайта.

Тестирование. Одним из основных этапов создания сайта является тестирование. На этом этапе проверяется правильность функционирования сайта, в том числе:

1) функционирование ссылок;
2) наличие ошибок в тексте;
3) эффективность навигации;
4) правильность почтовых и других форм;
5) открытие графических файлов;
6) работа сайта в различных браузерах.

Публикация. По окончании тестирования web-сайт будет опубликован на сервере и проверен повторно.

Реклама. Для того, чтобы web-сообществу стало известно о новом опубликованном сайте, необходимо распространить адрес сайта и аннотацию материала.

Для достижения этой цели можно использовать следующие возможности.

1) Опубликовать адрес web-сайта в различных изданиях.
2) Зарегистрировать web-сайт на различных серверах.
3) Опубликовать ссылку на web-сайт на других web-сайтах.
4) Использовать рекламные баннеры.

Контроль. После публикации и рекламирования web-сайта уровень его посещаемости определяется потребностью, новизной и актуальностью информации, размещенной на его страницах. Для сохранения востребованности web-сайта необходимо постоянно обновлять информацию на нем.

Разработка web-сайта состоит из следующих этапов:

- дизайн основных и типовых страниц сайта (выполняется в графическом редакторе);
- HTML-кодирование – с помощью браузера создается код, который можно просматривать;

- программирование сайта;
- размещение сайта в web;
- оптимизация web-сайта с целью повышения видимости в web-сетях;
- передача сайта заказчику.

После создания сайта его необходимо **опубликовать** на web-сервере. Существует три способа публикации сайта.

1. Все файлы сайта записываются на диск или другой носитель информации. Он должен быть доставлен администратору сервера. Администратор записывает папки с вашего диска в нужную папку сервера и настраивает программное обеспечение. **Это самый простой способ.**
2. Некоторые бесплатные web-серверы позволяют пользователю загружать файлы своего сайта через web-браузер. Успех этого подхода – простота загрузки. Пользователь должен записать в поле ввода названия файлов и нажать на кнопку Submit (отправить). **Этот способ используется реже.**
3. Администратор web-сервера устанавливает программу FTP-сервера. Затем по запросу автора создается папка для этого сайта, и автору дается разрешение войти только в эту папку. С помощью программы FTP-клиент автор сайта подключается к FTP-серверу и копирует файлы сайта в созданную конечную папку. После этого администратор объявляет о наличии нового сайта на сервере. Если автору необходимо обновить коренные файлы, он снова подключается к FTP-серверу, удаляет старые файлы и загружает новые. **Это самый распространенный способ.**

1 Отвечаем на вопросы

1. Что такое гипертекст?
2. Как появился HTML?
3. С помощью каких программ можно создать web-страницы?
4. Где можно создать web-страницу на языке HTML?
5. Какие виды браузеров вы знаете?

2 Думаем и обсуждаем

1. В чем заключается эффективность создания web-страниц на языке HTML?
2. Почему теги на языке HTML разделяются на парные и непарные?

3. Почему нужно соблюдать порядок при использовании ключевых тегов?
4. Почему в применении тегов необходимо соблюдать основные дисциплины?

3 Анализируем и сравниваем

Проанализировав важность создания web-сайтов с использованием тегов HTML, сравните три способа их публикации на сервере.

4 Выполняем в тетради

Заполните таблицу в тетради.

Теги и атрибуты	Функции

5 Выполняем на компьютере

1. Откройте программу Блокнот и сохраните пустой документ, как *job1.html*.
2. `<HTML> <HEAD> <TITLE>` мой первый web-сайт `</TITLE> </HEAD>` (вставить заголовок окна).
3. `<BODY BGCOLOR=yellow TEXT=blue>` (тело документа) `BGCOLOR=yellow` указывает, что фон документа желтый, а `TEXT=blue`, что цвет текста синий.
4. `<H1>` Добро пожаловать на мою первую работу `</H1>`.
5. `<H2>` Я покажу вам способы создания web-сайта `</H2>`.
6. Эти строки определяют заголовки разных уровней.
7. Поскольку информация, отображаемая в HTML документе, закончена, необходимо закрыть тег `<BODY>`, для этого нужно ввести `</BODY>`. Запись в HTML закончена, поэтому тег `<HTML>` тоже нужно закрыть, для этого введите `</HTML>`.

   Закройте Блокнот и откройте свою работу в браузере.

6 Делимся мыслями

1. Какой метод вы выбрали для разработки web-сайта HTML?
2. Приведите примеры использования тегов?

§ 37. Практикум. Разработка web-сайтов в HTML

Прежде чем приступать к выполнению заданий, ознакомьтесь с информацией о цветах, используемых на web-страницах.

Цвет, безусловно, является важной частью web-сайта. Цвет может вызывать у пользователей разные эмоции, а также стимулировать необходимые действия, так как цвета являются чрезвычайно мощным фактором воздействия на человека.

При выборе цветовой гаммы для сайта важно сделать это правильно, руководствуясь основными принципами теории цвета.

Цвет любого элемента сайта обычно формируется с помощью RGB-модели. RGB-модель – цветовая модель, основанная на принципе аддитивности (от англ. *add* – «добавлять»), которая с помощью трех цветов Red, Green, Blue образует 16,7 миллионов дополнительных цветов.

RGB-цвет получается в результате смешения красного, синего и зеленого в разных пропорциях: каждый оттенок можно описать тремя числами, обозначающими яркость трех основных цветов (*рис. 22*).

Рис. 22. *RGB-модель*

В HTML используется #RrGgBb-запись, называемая также шестнадцатеричной: каждая координата записывается в виде двух шестнадцатеричных цифр без пробелов. Например, #RrGgBb-запись белого цвета – #FFFFFF. Яркость цвета определяется в диапазоне от 0 до 255 (например, синий цвет – 0,0,255, красный – 255,0,0, черный – 0,0,0 и белый – 255,255,255).

#RrGgBb-запись основных цветов

Black	#000000	0, 0, 0
Gray	#808080	128, 128, 128
Silver	#C0C0C0	192, 192, 192
White	#FFFFFF	255, 255, 255
Fuchsia	#FF00FF	255, 0, 255
Purple	#800080	128, 0, 128
Red	#FF0000	255, 0, 0
Maroon	#800000	128, 0, 0

#RrGgBb-запись основных цветов

Yellow	#FFFF00	255, 255, 0
Olive	#808000	128, 128, 0
Lime	#00FF00	0, 255, 0
Green	#008000	0, 128, 0
Aqua	#00FFFF	0, 255, 255
Teal	#008080	0, 128, 128
Blue	#0000FF	0, 0, 255
Navy	#000080	0, 0, 128

Инструкция по выполнению заданий:

1. Создайте папку «Мой первый web-сайт» в папке «Ученик».
2. Откройте программу **Блокнот.**
3. Напишите простой текст HTML-файла в редакторе **Блокнот.**
4. Сохраните HTML-файл: **Файл** ⇒ **Сохранить как** ⇒ **Рабочий стол** ⇒ папка **Мой первый web-сайт** ⇒ имя файла **Первая страница.html** ⇒ **Сохранить.**
5. Для просмотра web-страницы необходимо открыть сохраненный файл в браузере.

Задание 1. Создайте web-страницу, при открытии которой в браузере будут отображаться ваши имя и фамилия.

Задание 2. **«Письмо другу».**

Нужно отобразить письмо вашему другу на странице браузера. Напишите письмо о своем друге с использованием пословиц.

Задание 3. Измените цвет заголовка документа.

Задание 4. Измените цвет фона подготовленного документа.

Задание 5. **«Мое хобби»**

Создайте web-страницу «Мое хобби». Web-страница должна состоять из выравненного по центру заголовка «Мое хобби», краткого рассказа о себе и о своих увлечениях (спорт, музыка, танцы и т. д.).

§ 38–39. Форматирование текста (шрифт, абзац, списки)

Вспомните!

- *Что такое гипертекст, кто впервые ввел этот термин?*
- *С помощью каких программ можно создавать web-страницы?*
- *Где можно создать web-страницу на языке HTML?*
- *Что такое тег?*

Вы узнаете:

- *о тегах форматирования текста;*
- *о тегах вставки абзаца;*
- *о видах тегов записи списков;*
- *о вводе бегущей строки на сайт.*

Теги форматирования текста

Заголовки в документе составляются тегами <**H**>, </h>. Например,

```
<H1> Меня зовут1 </H1>
<H2> Меня зовут2 </H2>
<H3> Меня зовут3 </H3>
<H1> Меня зовут4 </H1>
<H2> Меня зовут5 </H2>
<H3> Меня зовут6 </H3>
```

Знак <Hi> (где i – целое число от 1 до 6) позволяет выбрать из шести уровней заголовков разного размера. Заголовок первого уровня – самый крупный, а шестого – самый маленький.

Меня зовут1

Меня зовут2

Меня зовут3

Меня зовут4

Меня зовут5

Меня зовут6

Для **вставки абзаца** используются теги <P>, </p>, а для выравнивания текста по левому и правому краю или по центру используется атрибут *align*. Например, <P align=center> Моя первая страница </p>. Теперь предложение «Моя первая страница» располагается по центру страницы. Атрибут *align* может также принимать значения left (слева), right (справа).

С помощью тега <FONT> устанавливается шрифт, размер и цвет текста. Для этого используются атрибуты face, size, color. Например, <P align = center> Моя первая страница . Фраза «Моя первая страница» будет написана шрифтом Arial, 5 кеглем, синим цветом.

Для разделения строки можно использовать тег <**BR**>. Например,

```
<P> Говори по делу,
<BR> Живи по совести </p>
```

В данном коде тег
 переносит вторую часть предложения на следующую строку.

Чтобы выделить текст жирным шрифтом, нужно поместить его между тегами <B>..., для выделения курсивом используются теги <I>...</i>. Видимый текст «10 «А» класс предмет Информатика», написанный в HTML, выглядит так:

```
<HTML <HEAD><H3> 10 "A" класс предмет Информатика
</H3> </HEAD>
     <BODY> <P> <B> 10 "A" класс предмет Информатика </B>
          <P> <I> 10 "A" класс предмет Информатика </I>
          <P> <U> 10 "A" класс предмет Информатика </U>
     <P> <S> 10 "A" класс предмет Информатика </S>
     <P> <TT> 10 "A" класс предмет Информатика </TT>
     </BODY> </HTML>
```

10 «А» класс предмет Информатика

10 «А» класс предмет Информатика

<I> **10 «А» класс предмет Информатика**

10 «А» класс предмет Информатика

~~10 «А» класс предмет Информатика~~

10 «А» класс предмет Информатика

Нумерованные списки

Нумерованные списки выводятся как обозначенные списки, которые ограждаются только тегами <O<UL> и </O<UL> (ordered list – упорядоченный список), в результате чего в качестве номера списка записываются целые числа. Давайте немного преобразим один пример и пронумеруем список:

```
<HTML> <HEAD> <TITLE> пример </TITLE> </HEAD>
     <BODY text = green>
          <H2 ALIGN = CENTER> Строки пронумеро-
ванного списка </H2> <HR>
     <OL>
<LI> Айдар;
```

```
   <LI> Марат;
   <LI> Айдос;
   <LI> Жанна </LI>
</OL>
<HR>
</BODY>
   </HTML>
```

В результате работы этих тегов HTML выводится следующий список:

Строки пронумерованного списка

1. Айдар;
2. Марат;
3. Айдос;
4. Жанна

Если номер списка необходимо начать с определенного номера, то используется атрибут `start`, например:

```
<OL start = 5> Список использованной литературы
```

Для изменения вида списка используется атрибут `type`, например, чтобы записать номера латинскими цифрами, пишем следующим образом:

```
<OL type = I> Список использованной литературы
```

При записи маркированного списка используется тег `<UL>`, а для изменения вида маркера – атрибут `type`.

`<LI type = disk>` – маркер в виде точки;
`<LI type = circle>` – маркер в виде кружка;
`<LI type = square>` – маркер в виде квадрата.

Ненумерованные списки

Текст, расположенный между тегами `<UL>` и `</U<UL>` (`unordered list` – список без последовательности), рассматривается как список, написанный без нумерации, но с пометкой. Каждый новый элемент списка записывается, начиная со знака `<LI>` (`list` – список). Например, чтобы создать следующий список, набранный на экране буквами зеленого цвета:

- Айдар;
- Марат;

- Айдос;
- Жанна

Текст HTML необходимо набрать в Блокноте в следующем виде и просмотреть в любом браузере:

```
<HTML> <HEAD> <TITLE> пример </TITLE> </HEAD>
   <BODY text = green>
      <H2 ALIGN = CENTER> Строки маркированного
списка</H2> <HR>
      <UL>
      <LI> Айдар;
      <LI> Марат;
      <LI> Айдос;
      <LI> Жанна
   </U<UL>
   <HR>
   </BODY>
</HTML>
```

Строки маркированного списка

- Айдар;
- Марат;
- Айдос;
- Жанна

Наверное, вы заметили, что для знака <LI> не требуется тег закрытия.

Атрибуты тега <UL> type = disc | circle | square меняют внешнюю форму маркера на круг, окружность и квадрат соответственно.

Многоуровневые списки

В элемент любого списка могут быть включены и другие виды списков, таким образом составляются многоуровневые списки. Но если часто использовать многоуровневые списки, то многоступенчатый текст, который выводится на экран, увеличивает длину документа. Поэтому считается целесообразным использовать их только при необходимости.

Наиболее эффективным использованием многоуровневых списков является применение их при разработке содержания текста и различных планов.

Давайте кратко рассмотрим организацию таких списков на следующем примере:

```
<html> <head> <title> пример </title> </head>
<body>
<H1 ALIGN = center> работать в HTML интересно
</H1>
<DL>
<DT> Ненумерованные списки
<DD> Строки ненумерованного списка помечены специальным знаком слева, текст слегка перемещается вправо:
<UL>
<LI> 1 элемент
<LI> 2 элемент
<LI> 3 элемент </UL>
<DT> Строки нумерованного списка
<DD> Строки нумерованного списка пронумерованы с левой стороны:
<OL>
<LI> 1 элемент
<LI> 2 элемент
<LI> 3 элемент </LI>
</ol>
<DT> Списки определения
<DD> Такие списки сложнее двух предыдущих, но удобнее для чтения
<P> Списки можно записать внутри друг друга и создать многоуровневый список, но следует помнить, что не стоит сильно увлекаться этим методом
<P> В списке внутри одного элемента могут располагаться несколько абзацев. Такие абзацы располагаются с левой стороны, на одинаковых расстояниях </P>
</DL>
</body>
</html>
```

«Бегущие» строки

Теги <MARQUEE> и </MARQUEE> создают «бегущую» строку в окне браузера, которая перемещается от одного края строки до другого, и имеют следующие параметры:

```
<MARQUEE [ALIGN = "align'] [BEHAVIOR = 'behavior"]
[BGCOLOR = #rrggbb]
[DIRECTION = "direction"] [HEIGHT = "integer"]
[HSPACE = "integer"]
[LOOP = "integer"] [SCROLLAMOUNT = "integer"]
[SCROLLDELLAY = "integer"]
[VSPACE = "integer"] [WIDTH = "integer"] > Любой
текст </MARQUEE>
```

Значения и синтаксис написания некоторых из них:

ALIGN – позволяет задать «бегущий» текст по верхнему краю, центру или нижнему краю строки, принимает одно из следующих значений (слов): TOP, MIDDLE, BOTTOM.

BGCOLOR – определяет цвет фона «бегущей» строки в шестнадцатеричном формате RGB или с помощью определенного цветового имени на английском языке.

DIRECTION – перемещение по строке, то есть тег определяет направление скольжения, его допустимые значения left (влево) и right (вправо). В случае, если значение не указано, значение left включается по умолчанию. Для чередования движение строки в обе стороны используется атрибут BEHAVIOR=ALTERNATE.

HEIGHT – высота «бегущей» строки, указывается целым числом, определяемым количеством пикселей (точек) или в процентах (%).

LOOP – целое число, определяющее число повторений «бегущей» строки, может принять значение INFINITE (бесконечность).

SCROLLAMOUNT – целое число, которое определяет, на сколько пикселей перемещается текст за один шаг перемещения.

SCROLLDELAY – целое число, определяющее интервал между двумя скольжениями в миллисекундах.

WIDTH – ширина «бегущей» строки, указывается целым числом, которое означает количество пикселей (точек), или определяется в процентах (%).

Рассмотрим пример создания «бегущей» строки. Наберите код в Блокноте, сохраните под именем *пример.html* и просмотрите результат в любом браузере:

```
<HTML> <HEAD> <TITLE> пример 3-1 </TITLE> </HEAD>
      <BODY text = red>
            <CENTER>
            <H2> Бегущие строки </H2> <HR>
            <H3> <MARQUEE BGCOLOR = "yellow"
            DIRECTION = "RIGHT"
            SCROLLAMOUNT = "10" SCROLLDELAY = "200"
      WIDTH = "90%">
            Это первая бегущая строка
                  </MARQUEE>
      <P> <MARQUEE BGCOLOR = "green" DIRECTION = "LEFT"
      HEIGHT=30 SCROLLAMOUNT = "10"
      SCROLLDELAY = "100" WIDTH = "90%">
            Вторая бегущая строка </MARQUEE>
            </H3> <HR>
      <H3> <MARQUEE BGCOLOR = "blue"
      BEHAVIOR = alternate
            SCROLLAMOUNT = "10" SCROLLDELAY =
            "200" WIDTH = "90%">
            Третья бегущая строка
                  </MARQUEE>
      </CENTER> </BODY>
</HTML>
```

1 Отвечаем на вопросы

1. Что относится к форматированию текста?
2. Что такое «бегущие» строки?
3. С помощью каких тегов создаются «бегущие» строки?

2 Думаем и обсуждаем

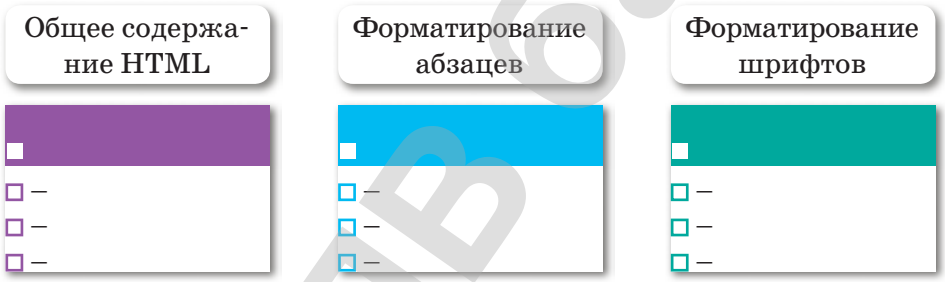
1. Почему текст форматируется с помощью тегов?
2. Для чего мы используем «бегущие» строки?

3 Анализируем и сравниваем

1. В чем разница между нумерованным и ненумерованным списком?
2. Сравните и проанализируйте «бегущие» строки.

4 Выполняем в тетради

1. Приведите примеры использования тегов для составления списков.
2. Заполните таблицу тегов, разделенных на группы в зависимости от их использования.



5 Выполняем на компьютере

Откройте текстовый редактор Блокнот, создайте и сохраните пустой документ под именем *Задание2.html*.

1. <HTML> должен быть первым тегом в документе. Для ввода заголовка окна введите следующие строки:
```
<HEAD> <TITLE> Моя первая страница </TITLE>
</HEAD>
```
2. Теперь введем тело документа, то есть, то, что будет показано в HTML-документе. Для этого введите следующие строки:

```
<BODY BGCOLOR = yellow TEXT = blue>
```

Здесь `BGCOLOR = yellow` определяет, что фон документа желтый, а `TEXT = blue` – что цвет текста синий.

3. Добавим на страницу нумерованные и ненумерованные списки. Для этого введите следующие теги:

```
<O<UL> Обозначает первый нумерованный список
<LI> ученик
<LI> учитель </LI>
</O<UL>
<UL TYPE = DISC> Обозначает ненумерованный список
<LI> Первый ученик
<LI> Второй ученик
<LI> Третий ученик
</U<UL>
```

4. На следующей строке наберите теги

. Это означает, что дважды были созданы пустые строки (воспроизводится в текстовых редакторах с помощью клавиши «Enter»).
5. Введите следующую строку: <MARQUEE>БЕГУЩАЯ СТРОКА </MARQUEE>. В результате появляется «бегущая» строка.
6. Информация, отображаемая в HTML-документе, закончилась, поэтому необходимо закрыть тег <BODY>, для этого наберите </BODY>. Запись в HTML-документе закончена, необходимо закрыть тег <HTML>, для этого необходимо набрать </HTML>.

   Закройте Блокнот, просмотрите отредактированные документы в браузере.

6 Делимся мыслями

1. В чем заключается эффективность использования тегов форматирования текста?
2. Где используются бегущие строки?

§ 40. Практикум. Форматирование текста

Инструкция по выполнению заданий:

1. Создайте папку «Мой первый web-сайт» в папке «Ученик».
2. Откройте программу Блокнот.
3. Напишите простой текст HTML-файла в редакторе Блокнот.
4. Сохраните HTML-файл, выполнив команды **Файл** ⇒ **Сохранить как** ⇒ **Рабочий стол** ⇒ папка **Мой первый web-сайт** ⇒ имя файла **Первая страница.html** ⇒ **Сохранить.**
5. Для просмотра web-страницы необходимо открыть ее в браузере.

Задание 1. Введение стихотворной строки с использованием специального тега для перехода на новую строку (абзац).

Введите на страницу web-сайта любую строку стихотворения.

Инструкция по выполнению задания:

1. Создайте отдельную папку, в которой будут храниться все файлы вашего сайта.
2. Запустите программу **Блокнот.**

```
<html> <head> <title>пример
   </title> </head>
<body>
<H1> Стихотворение </H1>
<H2> Абай </H2>
<P> Шлю поклон свой, каламкас.
   <BR>
Взор туманит влага глаз,<BR>
Лишь тобою очарован – <BR>
Сердца пламень не угас. </P>
<P>Совершенна без прикрас,<BR>
Словно жемчуг и алмаз. <BR>
Кто сравнится красотою <BR>
С чернобровой каламкас?! </P>
</body>
</html>
```

Стихотворение

Абай

Шлю поклон свой, каламкас.
Взор туманит влага глаз,
Лишь тобою очарован –
Сердце пламень не угас.

Совершенна без прикрас,
Словно жемчуг и алмаз.
Кто сравнится красотою
С чернобровой каламкас?!

3. Сохраните файл (обязательно укажите тип html-файла) под именем *Задание2.html*. Для просмотра web-страницы используйте кнопку Сверните в программе Блокнот, откройте свою

личную папку, затем с помощью двойного щелчка на файл *Задание2.html* откройте окно браузера.

Задание 2. Контроль расположения текста на экране.

Напишите поздравление народу Казахстана с праздником Наурыз. Выведите результат на экран, внесите изменения по своему усмотрению и сохраните файл повторно. Проверьте в браузере, сохранились ли изменения.

Задание 3. Создайте список своих одноклассников.

Напишите список учащихся вашего класса. После просмотра результата внесите изменения по своему желанию и сохраните их.

Инструкция по выполнению заданий:

1. Выберите текст из введенного списка, по необходимости установите цвет, размер, шрифт.
2. Выведите на страницу документа нумерованные и ненумерованные списки.

Задание 4. Создайте web-страницу, на которой будет отображаться текст, приведенный ниже.

Площадь поверхности круга $S = 4\pi r^2$
Формула воды H_2O
~~Текст~~
Редактор HTML

Задание 5. Запишите приведенный ниже список, используя теги HTML.

1. **Рабочие дни:**
   a. понедельник
   b. вторник
   c. среда
   d. четверг
   e. пятница
2. **Выходные дни:**
   a. суббота
   b. *воскресенье*

§ 41–42. Таблицы

Вспомните!

- *Какие теги для форматирования текста, вывода списка, вывода бегущих строк вы знаете?*

Вы узнаете:

- *о работе с таблицами в HTML;*
- *о тегах вывода таблиц.*

Таблицы – один из основных разделов web-сайта. Они состоят из следующих частей:

- заголовок таблицы;
- заголовки столбцов;
- ячейки.

Таблица заполняется последовательно по порядку строк (слева направо до конца строки, затем происходит переход на следующую строку). В каждую ячейку вводятся данные. Вставка таблицы выполняется с помощью тегов <TABLE> и </TABLE>, начало каждой строки определяется через теги <TR> и </TR>, а столбцы в этих строках – с помощью тегов <TD> и </TD> или <TH> и </TH>. Работа тегов <TD> и <TH> похожа, но тег <TH> выделяет графу жирным начертанием, а за тегами <TD> следуют обычные столбцы.

Заголовок таблицы заключен в теги <CAPTION> и </CAPTION>.

Правила полного построения общей таблицы выполняются по следующему образцу:

```
<TABLE ALIGN = "center" BGCOLOR = "#rrggbb"
BORDER = "integer"
   BORDERCOLOR = "#rrggbb" WIDTH = "integer">
......
</TABLE>
```

При построении таблицы некоторые из данных тегов могут не использоваться. Теперь рассмотрим работу атрибутов таблицы.

Атрибут **ALIGN** определяет выравнивание таблицы по краям (если не указано, по умолчанию слева). Значение ALIGN – слово в двойных кавычках – должно соответствовать одному из следующих: LEFT (по левому краю), CENTER (по центру), RIGHT (по правому краю).

Атрибут **BGCOLOR** устанавливает внутренний фон сетки таблицы (число в шестнадцатеричном формате RGB или название цвета на английском языке).

Атрибут **BORDER** – целое число, определяющее толщину линий рамок таблицы в пикселях. Если атрибут BORDER не предоставляется, линии рамок не будут отображаться.

Атрибут **BORDERCOLOR** устанавливает цвет линии рамок (число в шестнадцатеричном формате RGB или название цвета на английском языке), используется вместе с атрибутом BORDER.

Атрибут **WIDTH** – целое число, определяющее ширину таблицы, значение которому присваивается в виде количества пикселей или в процентах (%).

Заголовок таблицы задается с помощью тега <CAPTION>, который записывается следующим образом:

```
<CAPTION ALIGN = "top"> ..... </CAPTION>
```

При этом функции, выполняемые атрибутами, будут выглядеть следующим образом.

Атрибут **ALIGN**. При выравнивании заголовка таблицы по сторонам его значение должно соответствовать одному из слов LEFT, CENTER (если не указано, данное значение принимается по умолчанию) или RIGHT. Если заголовок необходимо разместить по верхнюю или нижнюю сторону таблицы, атрибут может принять в качестве значения одно из следующих: TOP – выше (данное значение принимается по умолчанию) и BOTTOM – ниже.

```
<TR ALIGN = "center" BGCOLOR = "#rrggbb"
BORDERCOLOR = "#rrggbb"> Строка таблицы … </TR>
```

Рассмотрим атрибуты тега <TR>.

ALIGN выравнивает края строки. Его возможные значения: LEFT (по умолчанию), CENTER, RIGHT.

BGCOLOR определяет цвет внутреннего фона строки (число в шестнадцатеричном формате RGB или название цвета на английском языке).

BORDERCOLOR – цвет боковых рамок (число в шестнадцатеричном формате RGB или название цвета на английском языке). Этот атрибут используется только в том случае, если значение атрибута BORDER тега <TABLE> не равно нулю.

ROWSPAN – количество ячеек, получаемых при объединении нескольких строк по вертикали в одну строку.

Столбцы (ячейки) строки таблицы определяются с помощью тегов <TD>...</TD> и <TH>... </TH>, как показано на примере ниже:

```
<TD или TH ALIGN = "right" BACKGROUND = "url"
[BGCOLOR = "#rrggbb" BORDERCOLOR = "#rrggbb"]>
Столбец </TD или /TH>
```

Функции атрибутов тегов <TD> и <TH>:

ALIGN – атрибут выравнивания текста в горизонтальном направлении. Его возможные значения: LEFT, CENTER (по умолчанию) и RIGHT.

BGCOLOR – атрибут установки цвета фона (число в шестнадцатеричном формате RGB или название цвета на английском языке).

BORDERCOLOR – атрибут цвета сторон ячейки (число в шестнадцатеричном формате RGB или название цвета на английском языке). Этот атрибут используется только в том случае, если значение атрибута BORDER тега <TABLE> не равно нулю.

COLSPAN – количество столбцов, размещаемых в объединенной строке (ячейке) для заголовка столбцов.

В качестве примера составим таблицу с информацией о группе учащихся. Откройте страницу в Блокноте, далее разместите на странице Блокнота код создаваемой таблицы.

```
<html> <body text = blue> <table border = 10
bordercolor = green
   width = 100% align = center bgcolor = "yellow">
   <caption align = bottom> <H2> Информация
   об учениках </H2>
   </caption>
   <tr> <th colspan = 4> Список учеников 6 класса
   </th> </tr>
   <tr> <th>  </th> <th> ФИО </th> <th>
   Адрес </th>
     </tr>
   <tr> <th rowspan = 3> Мальчики </th> <td>
   Жанабаев Жандос</td> <td>
      Улица Сатпаева, дом 10, квартира
   126 </td> </tr>
   <tr> <td> Мукан Куаныш </td> <td> проспект
```

```
      Аль-Фараби 15, квартира 225 </td>
      </tr>
      <tr> <td> Торегали Данияр </td> <td>
      улица С.Сейфуллина, дом 101, квартира 6 </td>
      </tr>
      <tr> <th rowspan = 3> Девочки</th>
      <td> Сакен Сандугаш </td> <td> улица
      С. Сейфуллина, дом 101, квартира 25
      </td>
   <tr> <td> Бакытжанова Айман </td> <td> проспект
Абая, дом 101, квартира 121
      </td> </tr>
   <tr> <td> Толебай Жанар </td> <td> улица Т.Озала,
дом 101, квартира 20
          </td> </tr>
        </table>
      </body>
   </html>
```

Список учеников 6 класса		
	ФИО	**Адрес**
Мальчики	Жанабаев Жандос	улица Сатпаева, дом 10, квартира 126
	Мукан Куаныш	проспект Аль-Фараби 15, квартира 225
	Торегали Данияр	улица Сейфуллина, дом 101, квартира 6
Девочки	Сакен Сандугаш	улица Сейфуллина, дом 101, квартира 25
	Бакытжанова Айман	проспект Абая, дом 115, квартира 121
	Толебай Жанар	улица Т.Озала, дом 36, квартира 95

Информация об учениках

1 Отвечаем на вопросы

1. Что такое таблица на web-странице?
2. Как заполняется таблица?

2 Думаем и обсуждаем

1. Насколько важно создание таблиц на web-страницах?
2. Почему мы должны придерживаться правил правильного расположения тегов при составлении таблиц?

3 Анализируем и сравниваем

1. Проанализируйте приведенные ниже правила создания таблиц.

Правила создания таблицы	Причина
1. Следует отобразить полную информацию	
2. Следует обратить внимание на значение таблицы, показанной на экране	

2. В чем отличие тегов создания таблиц от тегов форматирования текста?

4 Выполняем в тетради

Составьте таблицу, заполнив ее информацией о ваших друзьях.

Информация о моих друзьях

- 1
- 2
- 3
- 4
- 5
- 6
- ...

5 Выполняем на компьютере

1. Запустите программу **Блокнот**.
2. В Блокноте напишите текст HTML-файла, представленный ниже.

```
<HTML> <HEAD>
<TITLE> Расписание уроков </TITLE> </HEAD>
<BODY BGCOLOR = "#FFFFFF">
<P ALIGN = CENTER>
<FONT COLOR = "RED" SIZE = "6" FACE = "KZ ARIAL">
<B> Расписание уроков </B> </FONT> </P> <BR>
```

```
<FONT COLOR = "BLUE" SIZE = "4" FACE = "Times New Roman">
<TABLE BORDER = "1" WIDTH = 100% BGCOLOR = "#99CCCC">
<TR BGCOLOR = "#CCCCFF" ALIGN = CENTER>
<TD> Время </TD>
<TD> 10 а класс </TD>
<TD> 10 б </TD>
<TD> 10 в </TD>
</TR> <TR> <TD> 8-30 – 9-50 </TD>
<TD> Русский язык </TD>
<TD> Информатика </TD>
<TD> История </TD>
</TR> <TR> <TD> 10-00 – 11-20 </TD>
<TD> Математика </TD>
<TD> История </TD>
<TD> Английский язык </TD>
<TR> <TD> 11-30 – 12-30 </TD>
<TD> История </TD>
<TD> Алгебра </TD>
<TD> Физика </TD> <TR>
</TABLE>
</BODY>
</HTML>
```

3. Для сохранения файла HTML выполните команды: **Файл** ⇒ **Сохранить как** ⇒ **Рабочий стол** ⇒ папка **Мой первый web-сайт** ⇒ имя файла **Расписание уроков. html** ⇒ **Сохранить.**
4. Для просмотра web-страницы необходимо подключить сетевой браузер.

6 **Делимся мыслями**

Запомнив теги для создания таблицы, поделитесь с одноклассниками своими знаниями о способах построения таблиц в HTML.

§ 43. Практикум. Построение таблиц

Вы уже знаете, что для построения таблиц используются теги <TABLE> и </TABLE>. Например:

```
<table> <tr> <td> Первая ячейка </td> <td> Вторая ячейка </td> </tr> </table>
```

Первая ячейка Вторая ячейка

Здесь нет границ таблицы. Атрибут border используется для отображения границ таблицы.

```
<table border = "1">
<tr>
  <td> Первая ячейка </td>
  <td> Вторая ячейка </td>
</tr>
</table>
```

На web-странице такая таблица выглядит так:

Первая ячейка	Вторая ячейка

Задание 1. Создайте новый документ. Вставьте в документ следующую таблицу, цвет границы сделайте красным, а для толщины границы установите размер 5.

Первая ячейка	Вторая ячейка

Задание 2. Вставьте в документ следующие теги, просмотрите результат в браузере, а затем объясните:

```
<table border = "3" cellpadding = "10" bgcolor = "#999999">
  <tr>
    <td> Первая ячейка </td> <td> Вторая ячейка </td> <td> Третья ячейка </td>
  </tr>
  <tr>
      <td> Четвертая ячейка </td>
      <td bgcolor = @#FF0000@> Пятая ячейка </td> <td> Шестая ячейка </td>
  </tr>
</table>
```

Задание 3. Создайте таблицу по образцу в документе.

Расписание уроков					
Понедельник	**Вторник**	**Среда**	**Четверг**	**Пятница**	**Суббота**
Математика	**Казахский язык**	Химия	Казахский язык	Математика	*Литература*
Информатика	**Химия**	Физкультура	Математика	Информатика	*Физика*
Английский язык	**Физика**	Физика	Химия	Английский язык	*Информатика*
Казахский язык	**Русский язык**	Русский язык	Физкультура	Казахский язык	*Химия*
Физкультура	**Литература**	Математика	Информатика	Физкультура	*Физкультура*

Задание 4. Установите фоновое изображение для составленной таблицы.

Расписание уроков					
Понедельник	**Вторник**	**Среда**	**Четверг**	**Пятница**	**Суббота**
Математика	**Казахский язык**	Химия	Казахский язык	Математика	*Литература*
Информатика	**Химия**	Физкультура	Математика	Информатика	*Физика*
Английский язык	**Физика**	Физика	Химия	Английский язык	*Информатика*
Казахский язык	**Русский язык**	Русский язык	Физкультура	Казахский язык	*Химия*
Физкультура	**Литература**	Математика	Информатика	Физкультура	*Физкультура*

§ 44–45. CSS

Вспомните!

- *Что такое web-документ?*
- *Что такое гипертекст?*
- *Что такое гипермедиа?*

Вы узнаете:

- *о понятии CSS;*
- *об использовании CSS в HTML;*
- *об отличиях CSS и HTML;*
- *о видах стилей.*

Термины:

- CSS;
- стиль;
- стиль пользователя;
- стиль браузера;
- встроенный стиль;
- связанный стиль;
- импортированный стиль.

CSS (Cascading Style Sheets – каскадные таблицы стилей) – это стандарт, определяющий отображение данных в браузере. Если HTML предоставляет информацию о структуре документа, то таблицы стилей CSS определяют, как его отображать. Обычно HTML создается с помощью тегов форматирования, которые в свою очередь задают цвет и размер текста. Однако если есть необходимость изменить параметры одинаковых элементов на сайте, то придется просматривать все страницы, чтобы найти и изменить соответствующие теги. Язык CSS расширяет возможности языка HTML и предоставляет дополнительные параметры для документов HTML.

HTML и XHTML наиболее часто используются в качестве инструментов проектирования web-страниц, для чего также можно использовать все типы документов в формате SVG, XUL и XML.

Это интересно!

Каскадные таблицы стилей (CSS) появились в 1977 году после HTML. Несмотря на то, что CSS работает совместно с HTML, он не является языком HTML. Кроме того, CSS представляет собой персональный код, который, определяя теги HTML, расширяет его возможности. Это означает, что WWW – способ обмена текстовыми документами, а HTML – язык, на котором создаются эти документы. Пользователей интересовало не оформление документа, а его составные части. Поэтому в первых проектах HTML отсутствовали методы добавления видео на web-страницы. Однако со временем в Интернете стали появляться и другие пользователи, которым требовалось внешнее оформление сайта. Таким образом, появились каскадные таблицы стилей, разработанные с целью облегчения работы web-дизайнеров.

CSS, в отличие от HTML, применяется для создания структуры контентов сайта. CSS используется для форматирования, оформления, создания дизайна и стиля контента.

Появление CSS стало революцией в мире web-дизайна.

Преимущества использования CSS:

- Четкий контроль значительно снижает объем кода и подходит для его чтения.
- С помощью языка CSS можно открыть параметры, которые нельзя описать на языке HTML. Например, вы можете удалить линию под ссылками.
- С помощью CSS можно легко изменить внешний вид web-страницы. Внешний вид многих документов можно отобразить через одну таблицу. К примеру, весь текст на 30 страницах зеленого цвета. Но спустя некоторое время у вас может появиться желание изменить цвет на синий или красный, для этого необходимо открыть все 30 страниц и изменить цвет в нужном атрибуте. С помощью же языка CSS можно изменить все 30 страниц в одной таблице стилей.
- Комбинированная и собранная техника дизайна. В языке CSS существует понятие «верстка сайта».

Если вам необходимо изменить параметры одного и того же элемента на сайте, вам придется просмотреть все страницы, чтобы найти и изменить теги. А каскадные таблицы стилей (CSS) позволяют вам сохранить цвет, размер текста и другие настройки стиля.

Стиль – набор правил форматирования, используемых для быстрого изменения внешнего вида документа.

Стили позволяют использовать все группы атрибутов форматирования одним действием. С их помощью можно изменить внешний вид всех заголовков. Например, если в HTML для форматирования заголовка выполняются три действия: сначала его размер, затем шрифт и в конце – выравнивание по центру, то в CSS эти действия можно производить одновременно, применив стиль с помощью тега `<H1>`.

Если вы хотите быстро изменить текстовое оформление, созданное одним из стилей, вы можете изменить настройки стиля во всех используемых документах и автоматически изменить текст (*таблица 12*).

Таблица 12. Эффективность использования CSS

Уменьшает размер документа. Это позволяет уменьшить нагрузку
Использование CSS позволяет управлять множеством документов, редактируя только один файл с описанием стиля
Предлагает гораздо больше вариантов форматирования стилей, чем обычный HTML
Производит кэширование документов CSS

Типы использования стилей таблиц (*таблица 13*):

Таблица 13. Типы использования стилей таблиц

Стиль браузера	Это стандартная таблица стилей. Если стили не указаны, то применяются эти стандарты
Стиль пользователя	С помощью изменения параметров браузера любой пользователь может создать и использовать свой стиль
Внедренный стиль	Определяется с использованием атрибута style
Встроенный стиль	Располагается внутри документа HTML
Связанный стиль	Связывается с документом с помощью элемента link
Импортированный стиль	Импортирует стили

По сравнению с HTML синтаксис CSS сложнее. Поскольку вы уже знаете несколько видов тегов HTML, то вы можете написать страницу без каких-либо затруднений. Хотя CSS и является определением стиля, у него больше нюансов, поэтому важно не только знать характеристики языка CSS, но и научиться им пользоваться.

Это язык стилевой разметки имеет две основные части:

1) селекторы-метки;

2) правила, применяемые к данным селекторам.

Существует три вида использования CSS вместе с языком HTML.

Запись кода стилей CSS в отдельный файл, который можно загрузить, используя тег `<link>` на этой странице.

Пример 1. Создание простой HTML-страницы. Для этого нужно ввести такой код:

```
body{
background: blue;
color: white;
}
h1{
color:red;
}
h2{
color:yellow;
}
```

Теперь в Блокноте откройте новую страницу и сохраните ее как ***style.css*** в папке, в которой расположена HTML-страница (*рис. 23*):

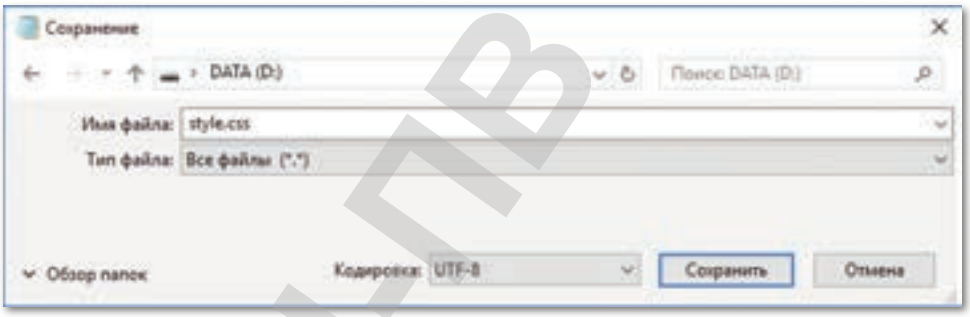


Рис. 23. *Сохранение файла* ***style.css***

Это будет страница наших стилей. Теперь необходимо привязать страницу ***style.css*** к странице HTML. Для этого в HTML имеется специальный тег `<link>`, который отвечает за привязку внешних файлов. Добавляем этот тег в нашу HTML-страницу:

```
<html>
<head>
<title> Привязка CSS HTML </title>
<link rel = "stylesheet" type = "text/css" href =
"style.css">
</head>
<body>
```

```
<h1> Заголовок первого уровня </h1>
Место для текста
<h2> Заголовок второго уровня </h2>
Место для текста
</body>
</html>
```

Пример 2. Для идентификации классов стилей с определенными тегами используется атрибут `class`:

```
<P class = def> текст </P>
Рассмотрим, как работает данный код.
<html>
<head>
<title> Селекторы по элементам </title>
<link rel = "stylesheet" type = "text/css" href =
"style.css">
</head>
<body>
<P class=def>
Класс &#151; Перед вставкой названия Класса
ставится точка
</body>
</html>
style.css
.def
{font-family: Helvetica; font-size:14pt;
border: solid 4pt red;padding: 6pt;
margin-left:5%; margin-right:5%}
```

Теперь наша HTML-страница выглядит следующим образом:

Класс – перед вставкой названия Класса ставится точка

Пример 3. Теги `<DIV>` и `<SPAN>`.

Эти теги важны для CSS. Они выделяют отдельные части документа и дают им специальные качества. Для этого необходимо поместить нужные элементы внутри тегов `<DIV> ... </DIV>` или `<SPAN>...</SPAN>`.

Они отличаются друг от друга следующим образом: после тега <DIV> браузер переходит на новую строку, а после тега <SPAN> – остается на прежней строке. Таким образом, использование этих тегов может назначить индивидуальные стилевые свойства для слов или символов в одной строке. Приведем примеры.

a) Использование тега <DIV>:

```
<html>
<head>
<title> Селекторы по элементам </title>
<link rel = "stylesheet" type = "text/css" href =
"style.css">
</head>
<BODY bgcolor = white text = black>
<DIV class = area1> Без труда </DIV>
<DIV class = area2> Не вытащишь и рыбку из пруда </DIV>
</DIV>
</BODY>
</html>
style.css
   .area1
    { color:red; font-weight:bolder;
    font-size:40pt; background:aqua}
   .area2
    { color:black; background:#CFB597}
   .area3
   { color:blue;background:#C0C0C0}
```

Наша HTML-страница выглядит следующим образом:

Без труда

Не вытащишь и рыбку из пруда

b)

```
<html>
<head>
<title> Селекторы по элементам </title>
<link rel = "stylesheet" type = "text/css" href =
"style.css">
```

```
   </head>
   <BODY bgcolor = white text = black>
   <SPAN class=area1> Без труда </SPAN>
   <SPAN class = area2> Не вытащишь и рыбку из пруда
</SPAN>
   </SPAN>
   </BODY>
   </html>
   style.css
      .area1
       { color:red; font-weight:bolder;
       font-size:40pt; background:aqua}
      .area2
       { color:maroon; background:#CFB597;
      padding:6pt}
```

Теперь наша HTML-страница выглядит так:

Не вытащишь и рыбку из пруда

Пример 4. Идентификаторы и классы можно передавать любым элементом HTML. Но часто бывает так: обычно мы хотим обозначить различные элементы одним и тем же стилем, например, зеленым цветом. В этом случае можно использовать унифицированный селектор. В таких параметрах имя элемента не указывается, в качестве класса или идентификатора и признака имени отображается точка или сетка.

Например:

```
   .red{
   color:red;
   }
   #yellow{
   color:yellow;
   }
```

Таким образом, к какому бы элементу (заголовок, абзац) не относился `class = "red"`, цвет текста все равно будет

красным. Можно задать только на один элемент id = "yellow", и цвет текста этого элемента будет желтым.

Размещение атрибута **style** внутри непосредственного тега HTML.

Пример 5.

```
<html>
<head>
<title> CSS id </title>
</head>
<body>
<p style = "background:#00ff00; color:red;">
Привет, по правилам CSS мои буквы закрасятся красным,
а фон – зеленым цветом
</p>
</body>
</html>
```

Теперь наша html-страница выглядит следующим образом:

Привет, по правилам CSS мои буквы закрасятся красным,
а фон – зеленым цветом

Размещение элемента **style** между тегами <head> ... </head>, то есть в шапке web-страницы.

Пример 6.

```
<html>
<head>
<title> CSS id </title>
<style type = "text/css"> p{background:#00ff00;
color:red;} </style>
</head>
<body>
<p> <b> <u> Если различные страницы оформляются
одинаковыми стилями, стили записываются в отдельный
текстовый файл с расширением CSS <u> </b> </p>
```

```
</body>
</html>
```

Теперь наша HTML-страница выглядит следующим образом:

Если различные страницы оформляются одинаковыми стилями, стили записываются в отдельный текстовый файл с расширением CSS

Пример 7. В HTML идентификатор элемента передается через параметр id, ему присваивается уникальное имя.

Например:

```
<p id = "pink"> Абзац с идентификатором текста (id) </p>
```

В качестве названия можно задать любое слово, кроме тега, параметра и названий элемента HTML и CSS. Например, идентификатор нельзя назвать *body*. Теперь добавляем к HTML-странице два абзаца и к одному из них – идентификатор:

```
<html>
<head>
<title> CSS id </title>
<link rel = "stylesheet" type = "text/css" href = "style.css">
</head>
<body>
<h1> Заголовок первого уровня </h1>
Место для текста
<h2> Заголовок второго уровня </h2>
Место для текста
<p> Обычный абзац </p>
<p id = "pink"> Абзац с идентификатором (id)
</p>
</body>
</html>
```

Теперь, если открыть страницы в браузере, то обе они будут белого цвета. Добавляем стили для абзацев в таблицу стилей (style.css):

```
body{
background: blue; color: white;
}
h1{ color:red;
}
h2{
color:yellow;
}
p{
color:black;
}
p#pink{
color:pink;
}
```

Сначала все абзацы в тексте были показаны в черном цвете, а текст абзаца с `id = "pink"` был розовым. В этом случае селектор состоит из элемента (p), элементов разделителя (#) и имени идентификатора (pink).

Следует отметить, что на одной странице может располагаться только один идентификатор (id). То есть, в нашем примере мы не можем сделать два абзаца `id "pink"`, абзац id должен быть только одним (id означает «уникальный, неповторимый»). Однако каждый абзац имеет свой идентификатор, создаем новый абзац `id = "green"` и задаем ему стиль в таблице стилей.

Пример 8. В приведенном выше примере мы сделали абзац с розовым текстом и указали, что такой id будет только один. А что будет, если мы захотим иметь розовый текст в двух и более абзацах? Для этого в HTML существует параметр `class`. В качестве значения класса отображается его название.

Добавляем к HTML-странице еще два абзаца и к ним `class = "pink"`:

```
<html>
<head>
<title> CSS class </title>
<link rel = "stylesheet" type = "text/css" href =
"style.css">
</head>
<body>
<h1> Заголовок первого уровня </h1>
Место для текста
<h2> Заголовок второго уровня </h2>
Место для текста
<p> Обычный абзац </p>
<p id = "pink"> Абзац с идентификатором </p>
<p class = "pink">(class) абзац с классом
pink </p>
<p class = "pink">(class) абзац с классом
pink </p>
</body>
</html>
```

Для этого класса добавляем правило в таблице стилей, чтобы показать стиль. В правилах в качестве селектора используется элемент и название pink. Но в данном случае, поскольку pink является названием класса, в качестве разделителя применяется знак точка (.):

```
body{
background:blue;
color:white;
}
h1{
color:red;
}
h2{
color:yellow;
}
p{
color:black;
```

```
}
p#pink{
color:pink;
}
p.pink{
color:pink;
}
```

С помощью этого класса можно создавать абзацы по необходимости.

Подводя итоги, следует отметить, что:

- если для всех аналогичных элементов (например, для всех заголовков h1) требуется один стиль, то селектор состоит только из следующих элементов: `p{color:black;}`;
- если элемент (любой – абзац, заголовок и т.д.) необходимо выделить, то к нему добавляется идентификатор (id), а в качестве разделителя в таблице стилей используется знак сетки (#). Например, `p#pink{color:pink;}`;
- если на странице имеется несколько элементов одного и того же стиля, то к ним добавляется класс (class) и разделительная точка (.) в таблице стилей. Например: `p.pink{color:pink;}`;
- в отличие от классов, идентификатор имеет несколько преимуществ. При этом, если для какого-либо элемента отображается как класс, так и идентификатор, используется стиль идентификатора.

Пример 9. У нас есть HTML-страница с данным кодом:

```
<html>
<head>
<title> Селекторы по элементам </title>
<link rel = "stylesheet" type = "text/css" href =
"style.css">
</head>
<body>
<p> Этот текст находится в абзаце. </p>
Это обычный текст.
```

```
<i> Этот текст выделен курсивом. </i>
<p> Этот текст находится в абзаце, но <i> эта часть
выделена курсивом. </i> </p>
</body>
</html>
```

Обозначим в тексте наклонные буквы (курсив), закрасив их желтым цветом. Пишем селектор по элементу в таблицу стилей:

```
i{
color:green;
}
```

Теперь наша HTML-страница выглядит следующим образом:

Этот текст находится в абзаце.
Этот обычный текст. *Этот текст выделен курсивом.*
Этот текст находится в абзаце, но *эта часть выделена курсивом.*

1 Отвечаем на вопросы

1. Что такое CSS?
2. Когда появился CSS?
3. Что такое стиль?
4. Каковы преимущества использования CSS?

2 Думаем и обсуждаем

1. В чем особенность использования CSS в HTML?
2. Почему CSS удобно использовать в мире web-дизайна?
3. Почему синтаксис CSS более сложен, чем HTML?

3 Анализируем и сравниваем

1. В чем отличие CSS от HTML?
2. Создайте ассоциативную карту.

4 Выполняем в тетради

Заполните таблицу:

Возможности HTML	Возможности CSS

5 Выполняем на компьютере

1. Создайте web-страницу «Мое хобби». Отредактируйте web-страницу так, чтобы по центру находился заголовок «Мое хобби», далее должен следовать небольшой рассказ о себе и своих увлечениях (спорт, музыка, танцы и т. д.).
2. Измените цифровую нумерацию на нумерацию буквами и римскими цифрами.
3. Используя различные типы маркеров, отредактируйте список различными видами маркеров.

6 Делимся мыслями

1. Как вы думаете, насколько важно использование CSS в создании web-страниц? Поделитесь мыслями.
2. С помощью сети Интернет сделайте постер, объясните преимущества и недостатки использования CSS.

§ 46–47. Внедрение мультимедиа

Вспомните!

- *Что такое CSS?*
- *Для чего используются таблицы CSS?*

Вы узнаете:

- *что такое мультимедиа;*
- *о видах мультимедийных тегов;*
- *как разместить аудио- и видеофайлы на web-странице.*

Термины:

- мультимедиа;
- фрейм;
- аудио и видео.

Мультимедиа – это совокупность технологий, позволяющих вводить, редактировать, сохранять, передавать и отображать такие типы данных, как текст, графика, анимация, цифровые изображения, видео и звук.

Одной из наиболее привлекательных особенностей Интернета является наличие ссылок на web-страницы внутри других web-страниц. Создавать гипертекстовые ссылки в документе HTML очень легко. Для этого используются легко описываемые теги <A. . .> и </A>. При составлении общих ссылок следует помнить следующие правила.

Фреймы. Структура фреймов. Язык HTML дает возможность разделить окно браузера на несколько частей и в каждом из них отобразить отдельные документы. Такие части называются **фреймами.**

1) Для создания фреймов используется специальный документ HTML, структура которого отличается от обычных документов. В таких документах не содержатся «тела» документов, которые на самом деле и не содержатся вообще в каком-либо тексте. Вместо этого эти документы содержат фреймы, расположенные между тегами **<FRAMESET>** и **</FRAMESET>.**
2) Атрибуты тега <FRAMESET>, определяющие методы разделения окон:
   - при использовании атрибута COLS = страница разделяется на части вертикальными линиями;
   - при использовании атрибута ROWS = страница разделяется на части горизонтальными линиями.

   Значения этих атрибутов определяют высоту (или ширину) частей окна. Параметры для каждой графы (строки) присваиваются с помощью указания количества пикселей запятыми по единице измерения или в процентах (знак %).
3) Между тегами <FRAMESET> и </FRAMESET> размещаются дополнительные теги, указывающие на необходимость

сформированных частей. Для этих целей можно использовать встроенный тег <FRAMESET>, позволяющий дополнительно разделить окно или индивидуальный тег <FRAME>, который определяет документы, вызываемые на отдельные части экрана.

Количество элементов, размещенных между тегами <FRAMESET> и </ FRAMESET>, должно соответствовать количеству сформированных частей.

4) В поле <FRAME> должны быть обязательные атрибуты SRC =, определяющие документы, вызываемые на определенную часть страницы. Это позволяет регулировать границы между дополнительными атрибутами и фреймами, а также некоторые другие свойства.

Пример 1.

```
<HTML>
<HEAD>
<TITLE> новости
     </TITLE>
          </HEAD>
               <FRAMESET>
<COLS = "25%">
<FRAME SRC = panel.htm>
<FRAME SRC = home1.htm>
</FRAMESET>
     </HTML>
```

Добавление графики на web-страницу. В настоящее время web-браузеры поддерживают не все форматы изображений, поэтому нельзя размещать изображение на каждой web-странице. Растровые и векторные изображения бывают разных форматов. Изображения Bitmap хранятся только в файлах с расширениями *jpg, gif, bmp, tiff, png, psd*. Для растровой графики JPEG, PNG, GIF и векторной графики используется SWF для обработки web-страниц.

Структура мультимедийной информации кардинально отличается от структуры текста, поэтому мультимедиа прямо не описывается в HTML-коде. Весь контент, необходимый для медиа, хранится в отдельных файлах. Ссылки на них записываются в HTML-код.

Для добавления изображений с помощью HTML используется непарный тег <img>, содержащий обязательный атрибут src.

Для тега <img> доступны следующие атрибуты:

- align – задает тип размещения изображения;
- alt – выводит текст на экран в случае, если изображение не загружено;
- border – определяет толщину рамок изображения;
- height – задает высоту изображения;
- hspace – задает расстояние отступления от изображения по горизонтали;
- ismap – определяет, представляет ли изображение собой карту;
- vspace – задает расстояние отступа от изображения по вертикали;
- width – задает ширину изображения;
- usemap – клиентская карта – определяет ссылку <map> с координатами изображений; • <img> – встроенный тег, т.е. его нельзя использовать вне блока.

Синтаксис: `<p> <img src = "sample.jpg"> </p>`

Основы работы с видео и аудио. Спецификация HTML содержит два тега для аудио- и видеосопровождения: <audio> и <video>.

Эти теги являются составной частью среды браузера. Во-первых, мультимедийная информация часто используется для повышения безопасности, а не для использования сторонних инструментов, во-вторых, сокращает аппаратные ресурсы для воспроизведения мультимедиа из-за тесной интеграции и в-третьих, устраняет многие проблемы с отображением информации. Кроме того, эти теги позволяют организовать управление web-сценариями с использованием <audio> и <video>. Но они также имеют определенные недостатки.

В качестве частичного решения проблемы кодирования используется элемент <source>, который позволит браузеру публиковать несколько источников мультимедиа.

Пример 2.

```
<audio>
<source src = "sound1.ogg">
<source src = "sound1.mp3">
</audio>
```

Вставка аудио- и видеозаписей. Для добавления аудиозаписей в HTML используется парный тег <audio>. Информация, которая находится между этими тегами, не будет отображаться в браузере, не поддерживающем <audio>.

Основной код вставки аудио:

```
<audio src = "sound1.mp3"> </audio>
```

Существует второй вариант обеспечения универсального воспроизведения в различных браузерах:

```
<audio>
<source src = "sound1.ogg">
<source src = "sound1.mp3">
</audio>
```

Тег <audio> состоит из следующих атрибутов:

- autoplay – файл начинает воспроизводиться сразу после загрузки страницы при ее включении;
- controls – добавляет панель управления к аудиофайлу;
- loop – воспроизводит медиа с самого начала после его завершения;
- preload – используется для загрузки файла вместе с загрузкой самой страницы, но если используется autoplay, то атрибут игнорируется;
- src – определяет путь к файлу для воспроизведения.

Пример 3.

```
<audio autoplay controls src = "1.mp3">
Тег <audio> не поддерживает
</audio>
```

Результат (*рис. 24*):

Рис. 24. *Результат программы*

Если браузер не поддерживает указанный тег, результат подключения тега <audio> (*рис. 25*).

Рис. 25. *Результат подключения тега <audio>*

Видео добавляется следующим образом:

```
<video src = "video1.avi"> </video>
```

Атрибуты тега *<video>:*

- autoplay – файл начинает воспроизводиться сразу после загрузки страницы при ее включении;
- controls – добавляет панель управления видео;
- height – задает высоту зоны для воспроизведения видео;
- loop – воспроизводит медиа с самого начала сразу после его завершения;
- poster – отображает путь к картинке, которая будет отображаться, когда видеозапись не воспроизводится или недоступна;
- src – определяет путь к файлу для воспроизведения;
- width – задает размер области воспроизведения видеозаписи.

1 **Отвечаем на вопросы**

1. Что такое мультимедиа?
2. Где расположен фрейм?
3. Каковы функции тега <audio>?
4. Каковы функции тега <video>?

2 **Думаем и обсуждаем**

1. Почему в HTML легко создавать гипертекстовые ссылки?
2. Почему нельзя размещать любые изображения на web-странице?
3. Почему материалы, необходимые для мультимедиа, хранятся в отдельном файле?
4. Для чего используется мультимедиа?

3 **Анализируем и сравниваем**

Запишите теги размещения мультимедиа на web-странице в таблице.

Аудио	Видео
1.	1.
2.	2.
…	…

4 **Выполняем в тетради**

Проанализируйте добавление изображений, аудио- и видеофайлов на web-страницу и заполните схему ниже.

5 **Выполняем на компьютере**

Создайте web-страницу, на которой будет написана ваша биография. Данные на web-странице должны соответствовать следующим требованиям:

1. Использовать теги обработки текста, выровнять заголовок биографии по центру документа, а текст – по краям.
2. Поместите на web-страницу свои фотографии и расположите изображение под текстом по центру.
3. Добавьте свою любимую музыку на web-страницу.
4. Добавьте на web-страницу интересные моменты из своей жизни.

6 **Делимся мыслями**

Обсудите web-страницы, созданные на компьютере. С какими трудностями вы столкнулись?

§ 48. Практикум. Внедрение мультимедиа

Попробуем вставить изображение в HTML-документ. Напишем тег в HTML-документе, в том месте, где будет размещено наше изображение.

<IMG SRC = "имя файла"> для определенного файла <IMG SRC = "dog1.gif">

В данном случае SRC является обязательным атрибутом для записи. В качестве его значения отображается путь к изображению и имя файла. Если файл находится в этой папке, строка не будет показана. Для изображения используются файлы формата GIF, JPEG или PNG.

В результате браузер помещает изображение в указанном месте, справа от предшествующего текста или другого объекта.

Обратите внимание на пример ниже, где показано одно изображение. В трех случаях изображение отобразится в строке, поэтому браузер помещает его по правую сторону от предшествующего текста.

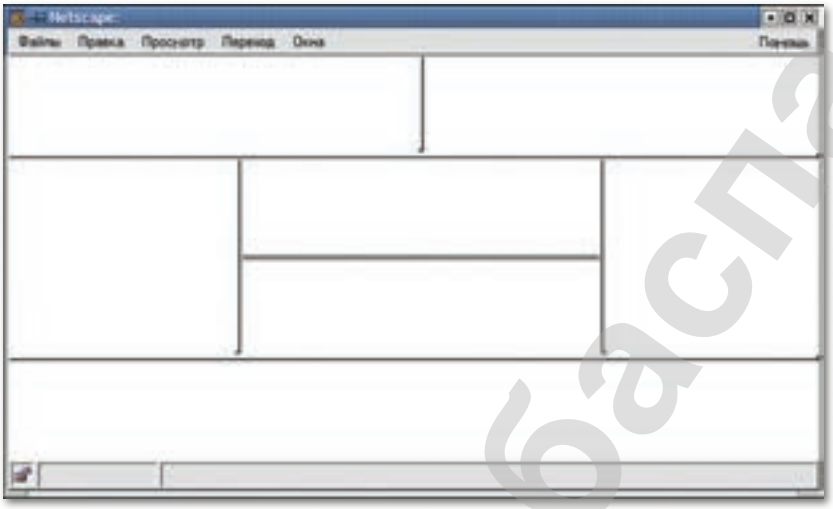
```
<HTML> <HEAD> <TITLE> Использование тега IMG
</TITLE> </HEAD>
<BODY>
<P> Здесь появится изображение после текста.
<IMG SRC = "kz.jpg"> </P>
</BODY>
</HTML>
```

Задание 1. Создайте новый документ. Вставьте изображение в документ, измените height – высоту изображения и width – ширину изображения.

```
<p> <img src = "kz.jpg" width = "…" height =
"..."> </p>
```

Задание 2. Добавьте атрибуты `align = left, align = right` во вставленное изображение. Просмотрите результаты в браузере и объясните.

Задание 3. С помощью тега FRAMESET разделите окно браузера на фреймы, как показано в рисунке ниже. Поместите в них файлы изображений.



Задание 4. Для вставки аудиозаписей в документ используйте атрибут autoplay, который начинает воспроизводить файл сразу после загрузки страницы. Используйте парный тег <audio>.

Задание 5. Вставьте в документ выбранный видеофайл и измените размер области воспроизведения видеозаписи с помощью атрибута width.

§ 49–50. Использование скриптов

Вспомните!

- *Что такое мультимедиа?*
- *Какие атрибуты включает в себя тег <audio>?*
- *Какие атрибуты включает в себя тег <video>?*

Вы узнаете:

- *о языке JavaScript;*
- *что такое объект.*

Термины:

- скрипты;
- JavaScript.

Термин **скрипт** представляет собой язык редактирования HTML, содержащий клиентские сценарии, которые могут быть выполнены при загрузке документа или позднее.

JavaScript–объектно-ориентированный, многопарадигмовый язык программирования, который может принимать императивный и функциональный стили.

Объект – это единая конструкция, состоящая из множества данных и функций, или набор свойств и способов в терминологии JavaScript.

JavaScript обычно используется как язык, который был введен для доступа к объектам программного обеспечения. Этот язык широко используется в качестве языка сценариев для интерактивного создания web-страниц в браузерах.

Для создания приложений JavaScript не требуется каких-либо дополнительных инструментов, требуется только текстовый редактор, который может работать с соответствующей версией JavaScript и позволяет создавать HTML-документы. Так как программа строится непосредственно в тексте HTML-документов JavaScript, есть возможность просмотреть результаты работы в браузере и при необходимости внести изменения.

Основные архитектурные особенности:

- динамический набор;
- слабый набор;
- автоматическое управление памяти;
- прототипное программирование, работает как объекты первой категории.

На JavaScript оказали влияние многие языки, так что даже неспециалисты программирования могут свободно им пользоваться.

JavaScript – объектно-ориентированный язык, однако прототипы, используемые в языке, различаются в работе

с объектами, что отличает его от традиционного объектно-ориентированного языка:

- функции первой категории;
- функции, похожие на список;
- анонимные функции;
- создатель ссылок.

Эти свойства придают языку дополнительную гибкость.

JavaScript позволяет выполнять математические расчеты, производимые ориентировочно. Кроме того, этот язык обладает развитыми рабочими инструментами, обладающими значениями дня и времени. JavaScript в основном был создан как альтернатива программам CGI и языку сценария Perl, а также как дополнительное приложение к языкам Java.

Расположение на web-страницах

Расположение внутри web-страниц

Для добавления кода JavaScript на страницу можно использовать теги <script> </script>, но их не обязательно помещать внутри контейнера <head>. В одном документе может содержаться множество контейнеров <script>, при этом не обязательно использовать атрибут "type = ‘text / javascript’" для каждого из них в отдельности, это значение используется по умолчанию.

Пример 1.

В браузере: сценарий, отображающий модульное окно с классической надписью «Hello, World!».

```
<html>
<head>
<title> Расположение внутри web-страницы </title>
</head>
<body>
<script type = "application/javascript">
alert('Hello, World!');
</script> </body>
</html>
```

Расположение внутри тега

Пример 2.

Спецификация HTML описывает набор атрибутов, используемых для установки редакторов событий. Например:

```
<a href = "delete.php" onclick = "return confirm('Вы уверены?');">
Удалить
</a>
```

Пример 3.

На странице редактирования в контексте использования кода JavaScript:

```
window.onload = function() {
var linkWithAlert = document.getElementById("alertLink");
linkWithAlert.onclick = function() {
return confirm('Вы уверены?');
};
};
```

Переход в личный файл

Пример 4.

Существует третий способ добавления JavaScript – запись сценария в отдельный файл и его включение с помощью устройства.

```
<head>
<script type = "application/javascript" src = "http://Путь к _файлу!!!">
</script>
</head>
```

1 **Отвечаем на вопросы**

1. Что такое скрипт?
2. Что понимается под «объектом»?
3. Каковы возможности JavaScript?

2 Думаем и обсуждаем

1. Почему добавление скрипта на web-страницу является важным?
2. В чем необходимость добавления файлов JavaScript на web-страницу?

3 Анализируем и сравниваем

Сравните возможности CSS и JavaScript.

Возможности CSS	Возможности JavaScript

4 Выполняем в тетради

Проанализируйте размещение на web-странице и заполните таблицу.

Web-страница	Тег	Файл

5 Выполняем на компьютере

Создайте объявление на любую тему. Данные на web-странице должны быть расположены согласно следующим требованиям:

1. Используя теги редактирования текста, выровняйте заголовок по центру документа, а текст – по ширине.
2. Поместите свои фотографии на web-страницу и расположите изображение под текстом по центру.
3. Добавьте JavaScript в свои сообщения.

6 Делимся мыслями

Что вы узнали на уроке? Чему вы научились? Поделитесь мыслями с друзьями. Какой метод добавления JavaScript на web-страницу использовали бы вы? Почему?

§ 51–52. Практикум. Использование скриптов

Для того, чтобы отредактировать программы на языке JavaScript, используется HTML-файл, теги <script> и </script>. Здесь вы можете увидеть результаты проделанной работы и при необходимости внести изменения.

```
<html>
<head>
<script language = "JavaScript">
document.write("Hello,world!<p>")
</script>
</head>
<body>
<H3> Мой первый созданный скрипт!!! <H3>
</body>
</html>
```

Открытие отдельного окна с изображением собаки dog.jpg.

```
<HTML>
<HEAD>
     <TITLE> Проверка браузера </TITLE>
</HEAD>
<BODY bgcolor = white text = black>
     <H2> Проверка браузера </H2> <HR>
     <SCRIPT language = JavaScript>
     <!--
      var win = open ("slon.jpg","",
"width = 320, height = 260" +
"resizable = 0, scrollbars =1" +
                "menubar = 0, location = 1" +
                "status = 0, toolbar = no");
     //-->
     </SCRIPT>
     <P>
```

Чтобы вернуться к основному тексту <EM> нажмите кнопку Назад </EM>, расположенную на панели инструментов браузера.

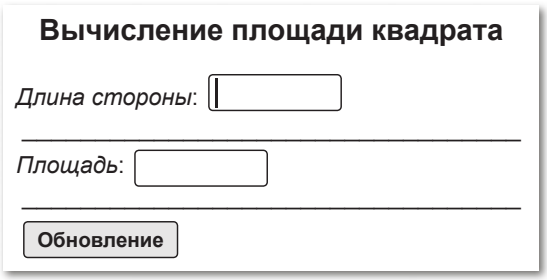
```
</BODY>
</HTML>
```

Создание кнопки, построенной браузером, с помощью HTML-кода.

```
<FORM>
<INPUT type=button
value = "Обычная кнопка">
</FORM>
```

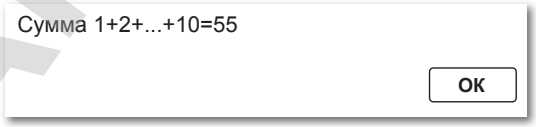
Вычисление площади квадрата.

```
<HTML>
<HEAD>
<title> Change – событие изменения значения элемента
   </title>
<script>
function srec(obj)
{ obj.res.value = obj.numl.value* obj.numl.value}
</script>
</HEAD>
<BODY>
<h2> Вычисление площади квадрата </h2>
<FORM name = "forml">
Длина стороны: <input type = "text" size = 7 name = "numl"
onChange = "srec(forml)">
<hr>
Площадь: <input type = "text" size = 7 name = "res"> <hr>
<input type = "reset" value = Обновить>
</FORM>
</BODY>
</HTML>
```



Запись скрипта, вычисляющего сумму чисел.

```
<HTML>
<HEAD>
<TITLE> Опыты с командой while </TITLE>
</HEAD>
<BODY bgcolor = white text = black>
      <H2> Опыты с командой while</H2>
      <HR>
<SCRIPT language = JavaScript>
      <!--
      var i = 1;
var sum = 0;
while (i <= 10)
{sum += i; i ++;}
alert ("Сумма 1 + 2 + … + 10 =" + sum);
      //-->
      </SCRIPT>
</BODY>
</HTML>
```



Задание 1. Создайте кнопку, построенную браузером с помощью HTML-кода.

Задание 2. Откройте отдельное окно с изображением крупного рогатого скота.

Задание 3. С помощью скриптов JavaScript создайте форму и рассчитайте периметр квадрата.

Задание 4. Выведите на экран окно alert с сообщением «Привет».

Задание 5. При нажатии на кнопку атрибут Onclick должен вывести на экран месяц, дату, год и текущее время.

Задание 6. Выведите на экран скрипт с сообщением «Ваш браузер поддерживает JavaScript»

Задание 7. Напишите программу, рассчитывающую сумму чисел в диапазоне от 5 до 13.

Задание 8. Напишите программу, рассчитывающую сумму чисел в диапазоне от 7 до 21.

Задание 9. Напишите программу, рассчитывающую сумму чисел в диапазоне от 10 до 100.

Задание 10. Напишите программу, рассчитывающую сумму чисел в диапазоне от 1 до 100.

Задание 11. Напишите данный программный код и рассчитайте результат.

```
var x = 5;
var s = 0;
while(x)
{s += x; x --;}
alert(s);
```

Задание 12. Напишите данный программный код и рассчитайте результат.

```
var x = 5;
var s = 0;
while (--x) s += x; s ++;
alert(s);
```

Задание 13. Напишите данный программный код и рассчитайте результат.

```
var x = 5;
var s = 0;
while(-- x || s < 10) s += x;
alert(s);
```

Задание 14. Напишите данный программный код и рассчитайте результат.

```
var x = 5;
var s = 0;
while(-- x && s) s += x;
alert(s);
```

Задание 15. Напишите данный программный код и рассчитайте результат.

```
var x = 5;
var s = 0;
while(-- x && s < 10) s += x;
alert(s);
```

Раздел V

ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Цели обучения:

- оценивать положительные и отрицательные стороны использования Big Data (бигдейта);
- объяснять понятие «реализация базы данных»;
- формулировать определения терминов «поле», «запись», «индекс»;
- определять первичный ключ в базе данных, типы данных в базе данных и создавать однотабличную, многотабличную базу данных (SQL);
- создавать форму для ввода данных, запросы, используя извлеченные данные (SQL);
- создавать отчеты, используя извлеченные данные (SQL);
- устанавливать связь web-страницы с базой данных.

§ 53–54. Big Data

Вспомните!

- *Что такое сайт?*
- *Назовите способы создания web-страниц.*
- *Для чего используются каскадные таблицы стилей?*
- *Можно ли внедрять мультимедиа на web-страницу?*
- *Где вы встречались с понятием Big Data?*

Вы узнаете:

- *о понятии «Big Data»;*
- *как оценивать положительные и отрицательные стороны использования Big Data.*

Термины:

- Big Data.

По определению оксфордского словаря, **Big Data** (дословно – «большие данные»), **данные** – это величины, знаки или символы, которыми оперирует компьютер и которые могут храниться и передаваться в форме электрических сигналов, записываться на магнитные, оптические или механические носители.

Термин «Big Data» используется для описания большого и растущего экспоненциально со временем набора данных. Для обработки такого количества данных не обойтись без машинного обучения.

До недавнего времени данные были ограничены электронными таблицами или базами данных – и все было очень упорядоченно и аккуратно. Все то, что нельзя было легко организовать в строки и столбцы, расценивалось как слишком сложное для обработки и игнорировалось. Однако прогресс в области хранения аналитической информации означает, что мы можем фиксировать, хранить и обрабатывать большое количество данных различного типа. В результате «данные» на сегодняшний день могут означать что угодно, начиная с баз данных и заканчивая фотографиями, видео, звукозаписями, письменными текстами и данными датчиков. Этот постоянно увеличивающийся поток информации означает, что мы можем использовать данные теми способами, которые невозможно было представить еще несколько лет назад. Сегодня компании могут с невероятной точностью предсказать, какие конкретные категории клиентов захотят сделать покупку и когда. Big Data помогает компаниям выполнять свою деятельность намного эффективнее.

Термин используется в сферах, где актуальна работа с количественно большими объемами данных, где постоянно происходит увеличение скорости потока данных в организационный процесс: в экономике, банковской деятельности, производстве, маркетинге, телекоммуникациях, web-аналитике, медицине и др.

К примеру, **Нью-Йоркская Фондовая Биржа** ежедневно генерирует 1 терабайт данных о торгах за каждую сессию.

Социальные медиа: статистика показывает, что в базы данных Facebook ежедневно загружается 500 терабайт новых данных, которые генерируются в основном благодаря загрузкам фото и видео на серверы данной социальной сети, обмену сообщениями, комментариям под постами и так далее.

Во время полета **реактивный двигатель** генерирует 10 терабайт данных каждые 30 минут. Так как ежедневно совершаются тысячи перелетов, то объем данных достигает петабайтов.

Вместе со стремительным накоплением информации быстрыми темпами развиваются и технологии анализа данных. Если еще несколько лет назад было возможно, скажем, лишь сегментировать клиентов на группы со схожими предпочтениями, то теперь возможно строить модели для каждого клиента в режиме реального времени, анализируя, например, его перемещения по сети Интернет для поиска конкретного товара. Интересы потребителя могут быть проанализированы, и в соответствии с построенной моделью выведена подходящая реклама или конкретные предложения. Модель также может настраиваться и перестраиваться в режиме реального времени, что было немыслимо еще несколько лет назад.

Большие данные различаются по объему, скорости генерации, разнообразию и изменчивости. Рассмотрим эти характеристики подробнее.

1. **Объем.** Сам по себе термин Big Data связан с большим размером. Размер данных – важнейший показатель при определении возможной извлекаемой ценности. 6 миллионов людей ежедневно используют цифровые медиа, что, по предварительным оценкам, генерирует 2.5 квинтиллиона байт данных. Поэтому объем – первая характеристика для рассмотрения.
2. **Разнообразие.** Этот аспект характеризуют гетерогенные источники и природу данных, которые могут быть как структурированными, так и неструктурированными. Раньше электронные таблицы и базы данных были единственными источниками информации, рассматриваемыми в большинстве приложений. Сегодня же данные в форме электронных писем, фото, видео, PDF-файлов и аудио также рассматриваются в аналитических приложениях. Такое разнообразие

неструктурированных данных приводит к проблемам при сборе, хранении и анализе: 27% компаний не уверены, что работают с подходящими данными.

3. **Скорость генерации.** То, насколько быстро данные накапливаются и обрабатываются для удовлетворения требований, определяет их потенциал. Скорость определяет быстроту притока информации из источников – бизнес-процессов, логов приложений, сайтов социальных сетей и медиа, сенсоров, мобильных устройств. Поток данных огромен и непрерывен во времени.
4. **Изменчивость** описывает непостоянство данных в некоторые моменты времени, которые усложняют обработку и управление. Так, например, большая часть данных неструктурирована по своей природе.

Преимущества, которые предоставляет Big Data:

1. **Сбор данных** из разных источников.
2. Улучшение бизнес-процессов через **аналитику в реальном времени.**
3. **Хранение** огромного объема данных.
4. **Инсайты**, то есть Big Data более проницательна к скрытой информации при анализе структурированных и полуструктурированных данных.
5. Большие данные помогают уменьшать риски и принимать умные решения благодаря подходящей **риск-аналитике.**

Проблемы Big Data:

1. **Конфиденциальность данных.** Big Data, которую мы сегодня генерируем, содержит много информации о нашей личной жизни, на конфиденциальность которой мы имеем полное право.
2. **Защита данных.** Даже если мы решаем, что нас устраивает то, что у кого-то есть наши данные для определенной цели, можем ли мы доверить им сохранность и безопасность наших данных?

1 **Отвечаем на вопросы**

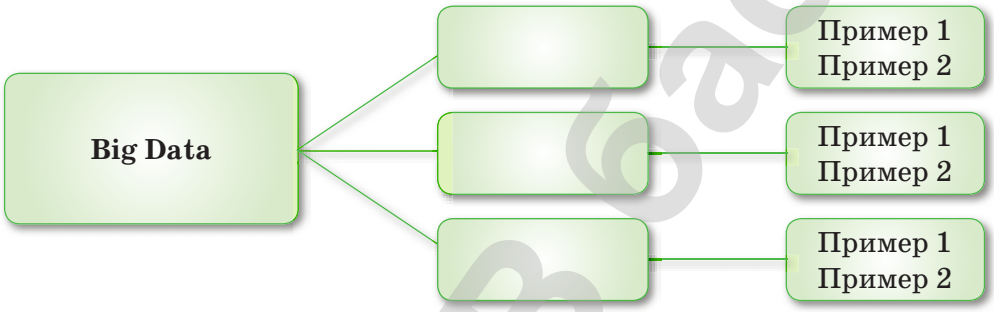
1. Что такое Big Data?
2. Какие основные характеристики Big Data вы знаете?
3. Где используется Big Data?
4. Каковы преимущества использования Big Data?
5. Каковы недостатки использования Big Data?

2 Думаем и обсуждаем

1. Насколько важно значение Big Data в нашей жизни?
2. Почему применение Big Data быстро развивается?
3. Почему мы не всегда уверены в безопасности наших данных?

3 Анализируем и сравниваем

Используя источники дополнительной информации, приведите примеры применения Big Data из жизни и проанализируйте знания и умения, которыми нужно обладать, чтобы стать специалистом в области больших данных.



4 Выполняем в тетради

Заполните таблицу характеристиками больших данных и традиционных данных.

	Традиционные базы данных	Большие данные
Область применения		
Характеристика данных		
Способ хранения данных		
Модель хранения и обработки данных		
Количество информации для обработки		

5 Выполняем на компьютере

Используя браузер, установленный на вашем компьютере, выполните задания.

1. Найдите СУБД:
   - SQL;
   - dBase;
   - Access;
   - Fox Pro;
   - Paradox.
2. Создайте базу данных, в которой будет отражаться: название СУБД, характер использования СУБД, функции, вид модели.
3. Сравните их и выясните, какая СУБД обладает наибольшим количеством функций.
4. Сохраните работу.

6 Делимся мыслями

Найдите в Интернете информацию о госпрограмме «Цифровой Казахстан» и концепции «Smart City» и поделитесь мнением о реализации этих проектов в будущем.

§ 55–56. Основные понятия баз данных

Вспомните!

- *Что такое Big Data?*
- *Как используется Big Data?*

Вы узнаете:

- *о понятии «реляционная база данных»;*
- *об определениях терминов: поле, запись, индекс;*
- *о реляционных связях между таблицами баз данных.*

Любая профессиональная деятельность так или иначе связана с информацией, с организацией ее сбора, хранения и обработки. Можно сказать, что базы данных стали неотъемлемой частью повседневной жизни, для их поддержки требуется некий организационный метод или механизм. Такой механизм называется системой управления базами данных (СУБД).

База данных (БД) – совместно используемый набор логически связанных данных (и их описание), предназначенный для удовлетворения информационных потребностей организации.

СУБД (система управления базами данных) – программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базы данных, а также получать контролируемый доступ к ним.

Системы управления базами данных существуют уже много лет, многие из них обязаны своим происхождением системам с неструктурированными файлами на больших вычислительных машинах. Наряду с общепринятыми современными технологиями, в области систем управления базами данных начинают появляться новые направления, что обусловлено требованиями растущего бизнеса, все увеличивающимися объемами корпоративных данных и, конечно же, влиянием технологий Интернета.

Реляционная база данных – база данных, основанная на реляционной (англ. *relation* – «отношение», «зависимость», «связь») модели данных. Реляционная база данных – это совокупность взаимосвязанных таблиц. Каждая таблица содержит информацию об объектах определенного типа. Строка таблицы содержит данные об одном объекте (например, о товаре, клиенте), а столбцы таблицы описывают различные характеристики этих объектов – атрибуты (например, наименование, код товара, сведения о клиенте). Строки таблицы (записи) имеют одинаковую структуру – они состоят из полей, хранящих

атрибуты объекта. Примеры реляционных СУБД: **MySql**, **PostgreSql**.

В реляционной модели объекты реального мира и взаимосвязи между ними представляются с помощью совокупности связанных между собой таблиц (отношений).

Даже в том случае, когда функции СУБД используются для выбора информации из одной или нескольких таблиц (т.е. выполняется запрос), результат также представляется в табличном виде. Более того, можно выполнить новый запрос с применением результатов предыдущего запроса.

Каждая таблица БД представляется как совокупность строк и столбцов, где строки (записи) соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы (поля) – атрибутам (признакам, характеристикам, параметрам) объекта, события или явления.

Одна из основных задач, возникающих при работе с базами данных, – это задача поиска. При этом, поскольку информации в базе данных, как правило, содержится много, перед программистами встает задача не просто поиска, а эффективного поиска, т.е. поиска за сравнительно небольшое время и с достаточной точностью. Для оптимизации эффективности запросов производят **индексирование** некоторых полей таблицы. Использовать **индексы** полезно для быстрого поиска строк с указанным значением одного столбца. Без индекса чтение таблицы осуществляется по всем столбцам, начиная с первой записи, пока не будут найдены соответствующие строки. Чем больше таблица, тем больше накладные расходы. Если же таблица содержит индекс по рассматриваемым столбцам, то база данных может быстро определить позицию для поиска в середине таблицы без просмотра всех данных.

Реляционные связи между таблицами баз данных

Связи между объектами реального мира могут находить свое отражение в структуре данных, а могут и подразумеваться, т.е. присутствовать на неформальном уровне.

Между двумя или более таблицами базы данных могут существовать отношения подчиненности, которые определяют, что для каждой записи главной таблицы (также называемой родительской) возможно наличие одной или нескольких записей в подчиненной таблице (называемой дочерней).

Выделяют три разновидности связи между таблицами базы данных: 1) «один-ко-многим»; 2) «один-к-одному»; 3) «многие-ко-многим».

Связь «один-ко-многим» предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы.

Связь «один-к-одному» предполагает, что одному атрибуту первой таблицы соответствует только один атрибут второй таблицы и наоборот.

Связь «многие-ко-многим» предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы и наоборот.

Всякую связь «многие-ко-многим» в реляционной базе данных необходимо заменить на связь «один-ко-многим» (одну или более) с помощью введения дополнительных таблиц.

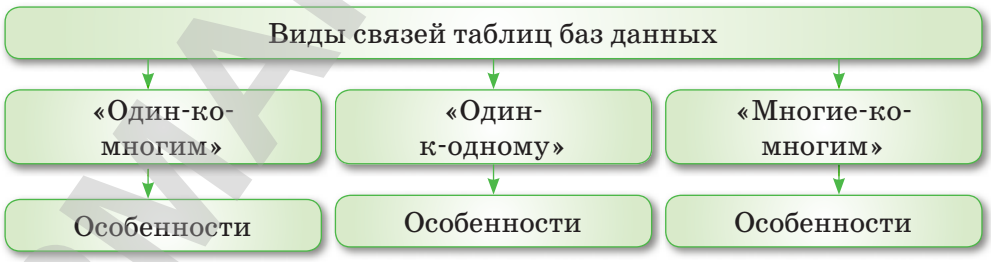
1 Отвечаем на вопросы

1. Что такое базы данных?
2. Необходимо ли управлять базами данных?
3. Что такое реляционная база данных?
4. Какие виды связи таблиц баз данных существуют?

2 Думаем и обсуждаем

1. Для чего нужны системы управления базами данных?
2. Почему реляционные базы данных тесно связаны с таблицами?

3 Анализируем и сравниваем



4 Выполняем в тетради

Дайте определения объектам реляционных баз данных и запишите в тетради.

5 **Выполняем на компьютере**

Установка сервера MySQL.

1. Скачайте MySQL Database Server с сайта *http://dev.mysql.com*.
2. Запустите процесс установки двойным щелчком по загруженному файлу.
3. Установите сервер, используя вариант **Обычная установка** (typical install). Установка должна пройти быстро.
4. По завершении установки, перед тем, как нажать кнопку завершения, убедитесь, что флажок **Конфигурировать сервер MySQL сейчас** (Configure the MySQL Server now) установлен. Это необходимо для запуска **Мастера конфигурации** (Configuration Wizard).
5. При запуске **Configuration Wizard** выберите переключатель **Стандартная конфигурация** (Standard Configuration), затем установите флажки **Установить, как службу** (Windows Install as Windows Service) и **Включить каталог Bin в путь поиска Windows** (Include Bin Directory in Windows Path).
6. Во время конфигурации вам будет предложено выбрать пароль для привилегированного пользователя **root**. Не забудьте записать пароль, он понадобится позже.
7. Откройте консоль (с помощью **Пуск** ⇒ **Выполнить** ⇒ **Command** (Start ⇒ Run ⇒ Command) и зарегистрируйтесь как привилегированный пользователь с помощью команды **mysql u root p**. Вам будет предложено ввести пароль, после этого появится подсказка **mysql**.
8. Создайте нового пользователя базы данных с помощью команды grant all privileges on *.* to 'lrng sql'@'localhost' identified by 'xxxxx'; (замените xxxxx паролем, который вы выбрали для этого пользователя).
9. Завершите сеанс с помощью команды **quit** (выйти) и зарегистрируйтесь в консоли как новый пользователь посредством команды **mysql**.

6 **Делимся мыслями**

С какими языками запросов вы встречались в повседневной жизни?

§ 57–58. Первичный ключ в базе данных

Вспомните!

- *Что такое реляционная база данных?*
- *Что такое поле, запись, индекс?*
- *Расскажите о реляционных связях между таблицами баз данных*

Вы узнаете:

- *как определять первичный ключ в базе данных.*

Для начала давайте подумаем над таким вопросом: какую информацию нужно дать о человеке, чтобы собеседник точно знал, что это именно тот человек, сомнений быть не может, второго такого нет? Сообщить фамилию, очевидно, недостаточно, поскольку существуют однофамильцы. Если собеседник человек, то мы можем приблизительно объяснить, о ком речь, например, вспомнить поступок, который совершил этот человек, или еще какую-либо информацию. Компьютер же такого объяснения не поймет, ему нужны четкие правила, как определить, о ком идет речь. В системах управления базами данных для решения такой задачи ввели понятие «первичный ключ».

В каждой таблице БД необходимо наличие первичного ключа – так именуют поле или набор полей, однозначно идентифицирующий каждый экземпляр объекта или запись. Значение первичного ключа в таблице БД должно быть уникальным, т.е. в таблице не допускается наличие двух и более записей с одинаковыми значениями первичного ключа. Он должен быть минимально достаточным, а значит, не содержать полей, удаление которых не отразится на его уникальности.

Первичный ключ (primary key, PK) – минимальный набор полей, уникально идентифицирующий запись в таблице. Значит, первичный ключ – это в первую очередь набор полей таблицы, во-вторых, каждый набор значений этих полей должен определять единственную запись (строку) в таблице и в-третьих, этот набор полей должен быть минимальным из всех обладающих таким же свойством. Поскольку первичный ключ определяет только одну уникальную запись, то никакие две записи таблицы не могут иметь одинаковых значений первичного ключа.

Таблица 14. *Телефонная книжка*

ФИО	Номер телефона	Адрес
Асанов Асан Асанович	233-44-55	ул. Шарипова, 12
Усенов Усен Усенович	233-66-77	пр. Абылайхана, 32

Например, в *таблице 14* ФИО и адрес позволяют однозначно выделить запись о человеке. Если же говорить в общем, без связи с решаемой задачей, то такие знания не позволяют точно указать на единственного человека, поскольку существуют однофамильцы, живущие в разных городах по одному адресу. Все дело в границах, которые мы сами себе задаем. Если считать, что знания ФИО, телефона и адреса без указания города для наших целей достаточно, то все замечательно, тогда поля ФИО и адрес могут образовывать первичный ключ. Проблема создания первичного ключа ложится на плечи того, кто проектирует базу данных, то есть разрабатывает структуру хранения данных. Решением этой проблемы может стать либо выделение характеристик, которые естественным образом определяют запись в таблице – задание так называемого *логического*, или *естественного*, первичного ключа, либо создание дополнительного поля, предназначенного именно для однозначной идентификации записей в таблице – задание так называемого *суррогатного*, или *искусственного*, первичного ключа.

Примером логического первичного ключа является номер паспорта в базе данных о паспортных данных жителей или ФИО и адрес в телефонной книге (таблица выше). Для задания суррогатного первичного ключа в нашу таблицу можно добавить поле `id` (идентификатор), значением которого будет целое число, уникальное для каждой строки таблицы. Использование таких суррогатных ключей имеет смысл, если естественный первичный ключ представляет собой большой набор полей или его выделение нетривиально.

Кроме однозначной идентификации записи, первичные ключи используются для организации связей с другими таблицами. Например, у нас есть три таблицы: 1) содержащая информацию об исторических личностях (`Persons`), 2) содержащая информацию об их изобретениях (`Artifacts`) и 3) содержащая изображения как личностей, так и артефактов (`Images`)- (*рис. 26*).

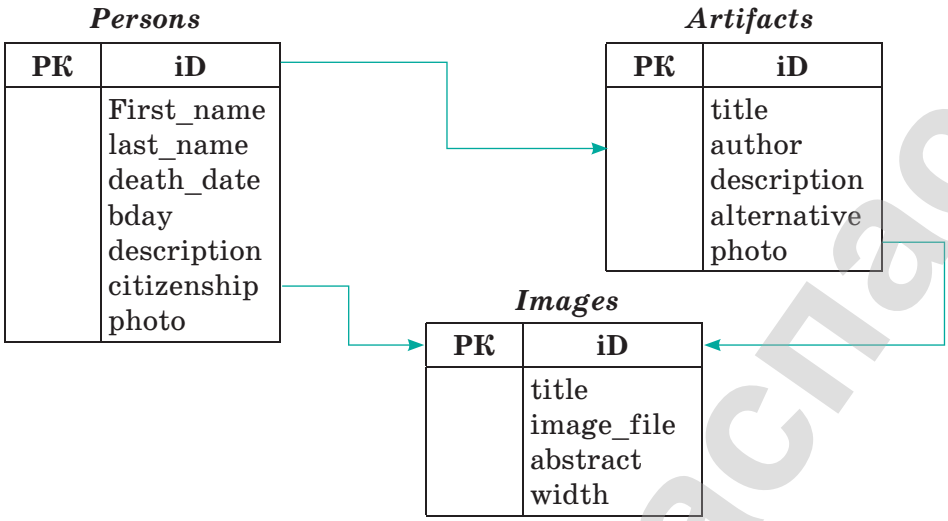


Рис. 26. *Пример использования первичных ключей для организации связей с другими таблицами*

Первичным ключом во всех этих таблицах является поле id (идентификатор). В таблице `Artifacts` есть поле author, в котором записан идентификатор, присвоенный автору изобретения в таблице `Persons`. Каждое значение этого поля является внешним ключом для первичного ключа таблицы `Persons`. Кроме того, в таблицах `Persons` и `Artifacts` есть поле photo, которое ссылается на изображение в таблице `Images`. Эти поля также являются *внешними ключами* для первичного ключа таблицы `Images` и устанавливают однозначную логическую связь `Persons-Images` и `Artifacts-Images`. То есть, если значение *внешнего ключа* photo в таблице личностей равно 10, то это значит, что фотография этой личности имеет `id=10` в таблице изображений. Таким образом, внешние ключи используются для организации связей между таблицами базы данных (родительскими и дочерними) и для поддержания ограничений ссылочной целостности данных.

1 **Отвечаем на вопросы**

1. Что такое первичный ключ?
2. Что можно использовать в качестве альтернативы первичному ключу в повседневной жизни?
3. Что такое логический ключ?

4. Как создать суррогатный ключ?
5. Что такое внешний ключ?

2 Думаем и обсуждаем

1. Для чего используются первичные ключи?
2. Для чего используются внешние ключи для первичных ключей?
3. Могут ли две записи таблицы иметь одинаковые значения первичных ключей?
4. Для чего нужен внешний ключ?

3 Анализируем и сравниваем

Определите правильные характеристики первичного ключа.

4 Выполняем в тетради

Создайте в тетрадях базу данных «Телефонная книжка» и заполните ее именами и номерами телефонов одноклассников.

База данных «Телефонная книжка»		
ФИО	Телефон	Адрес

5 Выполняем на компьютере

Создайте две таблицы данных: «Люди» и «Номера телефонов». Установите в таблицах первичный и внешний ключ.

Таблица. *Люди*

?	ФИО
1	Асанов Асан Асанович
2	Усенов Усен Усенович
3	Дуйсенов Дуйсен Дуйсенович

Таблица. *Номера телефонов*

?	Телефон	?
1	87011234567	1
2	87084561223	2
3	87001231234	3
4	87784564568	3

Ответьте на следующие вопросы:

1. Какое поле является уникальным в таблице «Номера телефонов»?
2. Является ли внешний ключ этой таблицы первичным ключом таблицы «Люди»?
3. С помощью чего обеспечивается связь между таблицами «Номера телефонов» и «Люди»?
4. Определите, по сколько номеров телефона есть у каждого человека из таблицы «Люди».

6 Делимся мыслями

Каким может быть первичный ключ базы данных учебников в школьной библиотеке?

§ 59–60. Разработка базы данных

Вспомните!

- *Как определяется первичный ключ в базе данных?*

Вы узнаете:

- *о типах данных в базе данных SQL;*
- *как создать однотабличную базу данных SQL;*
- *как создать многотабличную базу данных SQL.*

Все популярные базы данных, включая MySQL, имеют возможность хранить типы данных, такие как строки, даты и числа. Обычно их различия заключаются в возможности хранения специальных типов данных, например, XML-документов, очень больших текстов или двоичных документов.

MySQL (название «MySQL» состоит из слова «My» в честь дочери разработчика программы Майкла Видньюса (Michael Widenius) Мю и фразы «SQL, Structured Query Language», которая означает «язык структурированных запросов») – это одна из самых популярных и самых распространенных систем управления базами данных (СУБД) в мире, которая является открытой и бесплатной. В качестве серверной программы используется несколько баз данных для нескольких пользователей.

Данные – это совокупная информация, хранимая в базе данных в виде одного из нескольких различных типов данных. С учетом типов данных устанавливаются основные правила для информации, содержащейся в конкретном столбце таблицы, в том числе размер выделяемой для нее памяти.

Мы рассмотрим только *символьные*, *числовые* и *временные* типы данных.

Символьные данные

Символьные данные могут храниться как строки фиксированной или переменной длины. Разница заключается в том, что строки фиксированной длины справа дополняются пробелами, тогда как строки переменной длины – нет. При определении столбца символьного типа необходимо задать максимальный размер сохраняемой в нем строки. Например, если предполагается хранить строки длиной до 20 символов, можно использовать любое из этих описаний:

```
CHAR(20) /* строка фиксированной длины */
VARCHAR(20) /* строка переменной длины */
```

В настоящее время максимальная длина этого типа данных составляет 255 символов (хотя в будущих версиях будут

допустимы более длинные строки). Для сохранения более длинных строк (таких, как сообщения электронной почты, XML-документы и т. д.) используется один из текстовых типов – `tiny text` (крошечный текст), `text` (текст), `medium text` (средний текст), `long text` (длинный текст), которые будут рассмотрены в данном разделе позже. Тип `char` подходит для случаев, когда в столбце предполагается хранить только строки одинаковой длины, например, сокращенные названия государств, а тип `varchar` – для строк разной длины. Типы `char` и `varchar` одинаково применимы во всех основных серверах БД.

Наборы символов. В языках, использующих латинский алфавит, например, в английском, довольно мало символов, поэтому каждый символ хранится как один байт. В других языках, таких как японский, китайский и корейский, символов много, значит для хранения одного символа здесь требуется несколько байт. Такие наборы символов называют **многобайтовыми наборами символов** (`multibyte character sets`). MySQL может хранить данные, используя разные наборы символов, как одно-, так и многобайтовые. Просмотреть поддерживаемые сервером наборы символов можно с помощью команды **show** (показать):

```
mysql> SHOW CHARACTER SET;
```

Charset	Description	Default collation	Maxlen
big5	Big5 Traditional Chinese	big5_chinese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
cp850	DOS West European	cp850_general_ci	1
hp8	HP West European	hp8_english_ci	1
koi18r	KOI8-R Relcom Russian	koi18r_general_ci	1
latin1	ISO 8859-1 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Sentral European	latin2_general_ci	1
swe7	7bit Swedish	swe7_swedish_ci	1
ascii	US ASCII	ascii_chinese_ci	1

Текстовые данные

Если нужно хранить данные, для которых не хватит 255 символов столбца типа `char` или `varchar`, вам понадобится один из текстовых типов.

В *таблице 15* показаны доступные текстовые типы и их максимальные размеры.

Таблица 15. Текстовые типы данных MySQL

Тип	Максимальное количество символов
Tinytext	255
Text	65 535
Mediumtext	16 777 215
Longtext	4 294 967 295

Выбирая тот или иной текстовый тип, необходимо помнить:

- Если размер данных, загружаемых в текстовый столбец, превышает максимальный размер для этого типа, не поместившиеся данные отсекаются.
- В отличие от столбца типа `varchar`, при загрузке данных в такой столбец пробелы в конце строки не удаляются.
- При использовании столбцов типа `text` для сортировки или группировки используются только первые 1024 байта, хотя при необходимости это ограничивающее значение можно увеличить.
- Разные текстовые типы присущи исключительно MySQL. У SQL Server для больших символьных данных есть только один тип `text`, а в DB2 и Oracle применяется тип данных под названием `clob` (Character Large Object). При создании столбца для данных произвольного формата, например, столбца **примечания** (notes) для хранения информации о взаимодействиях клиента с отделом клиентского сервиса, которую вам не хотелось бы ограничивать 255 символами, следует выбрать тип `text` или `mediumtext`.

Числовые данные

Хотя и кажется, что хватило бы одного числового типа данных с названием **числовой** (numeric), все же есть разные числовые типы, отражающие разные способы использования чисел. Столбец, являющийся индикатором поставки заказа покупателю: столбец такого типа, называемого Boolean (булев), может содержать 0, что означает ложь (false) и 1, что означает истина (true).

Первичный ключ для таблицы транзакций, генерируемый системой, обычно начинается с 1 и увеличивается с шагом 1, возможно, до очень больших значений.

Номер позиции в клиентской электронной корзине для покупок значениями столбца данного типа являются положительные целые числа от 1 до 200 (максимум).

Данные компоновки буровой установки для печатных чертежей, высокоточные научные или технологические данные, как правило, требуют точности до восьмизначного числа.

MySQL располагает несколькими разными числовыми типами для работы с этими (и многими другими) видами информации. Наиболее часто числовые типы используют для хранения целых чисел. При задании одного из таких типов можно также указать, что данные беззнаковые, тогда сервер будет знать, что все хранящиеся в столбце данные не отрицательные. В *таблице 16* показано пять разных типов данных, предназначенных для хранения целых чисел.

Таблица 16. *Целые типы данных MySQL*

Тип	Диапазон значений со знаком	Диапазон значений без знака
Tinytext	от -128 до 127	от 0 до 255
Smallint	от -32 768 до 32 767	от 0 до 65 535
Mediumint	от -8 388 608 до 8 388 607	от 0 до 16 777 215
Int	от -2 147 483 648 до 2 147 483 647	от 0 до 4 294 967 295
Bigint	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807	от 0 до 18 446 744 073 709 551 615

При создании столбца одного из целых типов MySQL выделит для хранения данных соответствующее количество памяти – от 1 байта для типа `tinyint` до 8 байт для `bigint`. Поэтому попытайтесь подобрать тип достаточного размера для хранения самого большого из предполагаемых чисел без неоправданного расхода памяти. Для чисел с плавающей точкой (таких, как 3,1415927) можно выбрать один из типов, приведенных в *таблице 17*.

Таблица 17. *Типы данных MySQL для чисел с плавающей точкой*

Тип	Числовой диапазон
Float (p.s)	от -3,402823466E+38 до -1,175494351E-38 и от 1,175494351E-38 до 3,402823466E+38
Double (p.s)	от 1,7976931348623157E+38 до -2,225073858072014E-308 и от 2,2250738585072014E-38 до 1,7976931348623157E+308

Для типа с плавающей точкой можно задать точность (`precision`) – общее допустимое число разрядов, как справа, так и слева от десятичной точки – и масштаб (`scale`) – допустимое число разрядов справа от десятичной точки, но эти параметры не являются обязательными. В *таблице 17* они представлены как *p* и *s*. Задавая точность и масштаб для столбца, имеющего

тип **с плавающей точкой**, необходимо помнить, что сохраняемые в нем данные будут округляться, если число разрядов в них превысит заданный масштаб и/или точность.

Временные данные

Наряду со строками и числами довольно часто приходится работать с информацией о датах и/или времени. Этот тип данных называют **временным** (temporal). К примерам временных данных в базе данных относятся:

1) дата будущего события, например, доставки заказа покупателю;
2) фактическая дата доставки заказа покупателю;
3) дата и время изменения пользователем определенной строки таблицы;
4) дата рождения сотрудника;
5) год, соответствующий строке таблицы yearly_sales (продажи за год) в хранилище данных;
6) время, необходимое для монтажа электропроводки в автомобиле на сборочном конвейере.

В MySQL есть типы данных для обработки всех подобных ситуаций.

В *таблице 18* показаны временные типы данных, поддерживаемые MySQL.

***Таблица 18**. Временные типы данных MySQL*

Тип	Формат по умолчанию	Допустимые значения
Date	YYYY-MM-DD	от 1000-01-01 до 9999-12-31
Datetime	YYYY-MM-DD HH:MI:SS	от 1000-01-01 00:00 до 9999-12-31 23:59:59
Timestamp	YYYY-MM-DD HH:MI:SS	от 1970-01-01 00:00 до 2037-12-31 23:59:59
Year	YYYY	от 1901 до 2155
Time	HHH:MI:SSS	от -838:59:59 до 838:59:59

Создание таблиц

Теперь, имея четкое представление о том, какие типы данных могут храниться в базе данных MySQL, самое время взглянуть, как эти типы используются при описании таблиц. Начнем с описания таблицы для хранения информации о человеке.

Шаг 1: проектирование

Прежде всего, давайте определим, какие характеристики вам нужно сделать перед отправкой личных данных.

- имя, фамилия (name);
- пол (gender);
- дата рождения (birth date);
- адрес (address);
- любимое блюдо (favorite foods).

Разумеется, список не полный, но этого пока достаточно. Следующий шаг – дать столбцам имена и назначить типы данных. В *таблице 19* показан первый вариант.

***Таблица 19**. Таблица Person (человек)*

Столбец	Тип	Допустимые значения
Name	Varchar (40)	
Gender	Char (1)	*M, F*
Birth_date	Date	
Address	Varchar (100)	
Favorite_foods	Varchar (200)	

Столбцы name, address и favorite_foods типа varchar позволяют записывать информацию в свободной форме. В столбце gender (пол) допускается только один символ, M (M) или F (Ж). Столбцу birth_date (дата рождения) назначен тип date, поскольку точное время не требуется.

Шаг 2: уточнение

При повторном анализе столбцов таблицы можно сделать следующие выводы:

- Столбец name на самом деле является составным объектом, включающим имя и фамилию.
- Поскольку несколько человек могут иметь одинаковые имя, пол, дату рождения и т. д., в таблице person нет столбцов, гарантирующих уникальность.
- Столбец address – тоже составной объект, включающий улицу, город, штат/область, страну и почтовый индекс.
- Столбец favorite_foods – это список, содержащий 0,1 или более независимых элементов. Следует вынести эти данные в отдельную таблицу, включающую внешний ключ к таблице person, чтобы обозначить человека, к которому приписано конкретное блюдо.

В *таблице 20* можно увидеть оптимизированный вариант таблицы person после учета всех этих замечаний. Теперь, когда у таблицы person есть первичный ключ (person_id), гарантирующий уникальность, следующим шагом будет построение таблицы favorite_food, включающей внешний ключ к таблице person. Результат показан в *таблице 21*.

Столбцы person_id и food (блюдо) образуют первичный ключ таблицы favorite_food. Столбец person_id также является внешним ключом к таблице person.

Таблица 20. *Таблица Person (с внесенными изменениями)*

Столбец	Тип	Допустимые значения
Person_id	Smallint	
First_name	Varchar (20)	
Lastt_name	Varchar (20)	
Gender	Char (1)	*M, F*
Birth_date	Date	
Street	Varchar (30)	
City	Varchar (20)	
State	Varchar (20)	
Country	Varchar (20)	
Postal_code	Varchar (20)	

Таблица 21. *Таблица Favorite_food (любимое блюдо)*

Столбец	Тип
Person_id	Smallint (unsignet)
Food	Varchar (20)

Шаг 3: построение выражений SQL управления схемой данных

Теперь, по завершении проектирования двух таблиц для размещения персональной информации, следующим шагом является формирование выражений SQL для создания таблиц в БД. Выражение для создания таблицы person:

```
CREATE TABLE person (person_id SMALLINT UNSIGNED,
fname VARCHAR(20),
lname VARCHAR(20),
gender CHAR(1),
birth_date DATE,
address VARCHAR(30),
city VARCHAR(20),
state VARCHAR(20),
country VARCHAR(20),
postal_code VARCHAR(20),
CONSTRAINT pk_person PRIMARY KEY (person_id) ).
```

В этом выражении вам должно быть понятно все, кроме последнего элемента. При описании таблицы необходимо сообщить серверу

БД, какой столбец или столбцы будут играть роль первичного ключа таблицы. Это осуществляется путем создания ограничения (`constraint`) для таблицы. В описание таблицы можно добавить ограничение одного из нескольких типов. Данное ограничение является ограничением первичного ключа (primarykey constraint). Оно накладывается на столбец `person_id` и получает имя `pk_person`. Обычно имена ограничений первичного ключа начинаются с приставки `pk`, а затем указывается имя таблицы, чтобы при просмотре списка таких ограничений было ясно, чем является каждое из них.

1 Отвечаем на вопросы

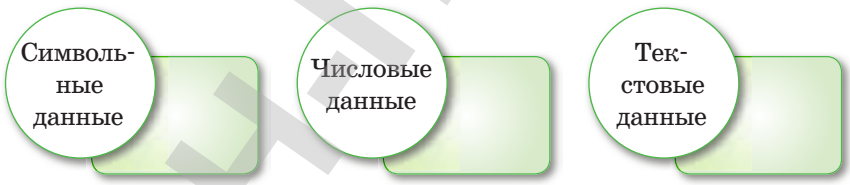
1. Какие типы данных существуют?
2. Какие шаги построения таблицы с использованием различных типов данных вы знаете?

2 Думаем и обсуждаем

1. Для чего нужны различные типы данных?
2. Как можно обобщить символьную, числовую и текстовую информацию в базе данных?

3 Анализируем и сравниваем

1. В чем отличие данных от информации?
2. Выделите признаки типов данных, дополните схему.

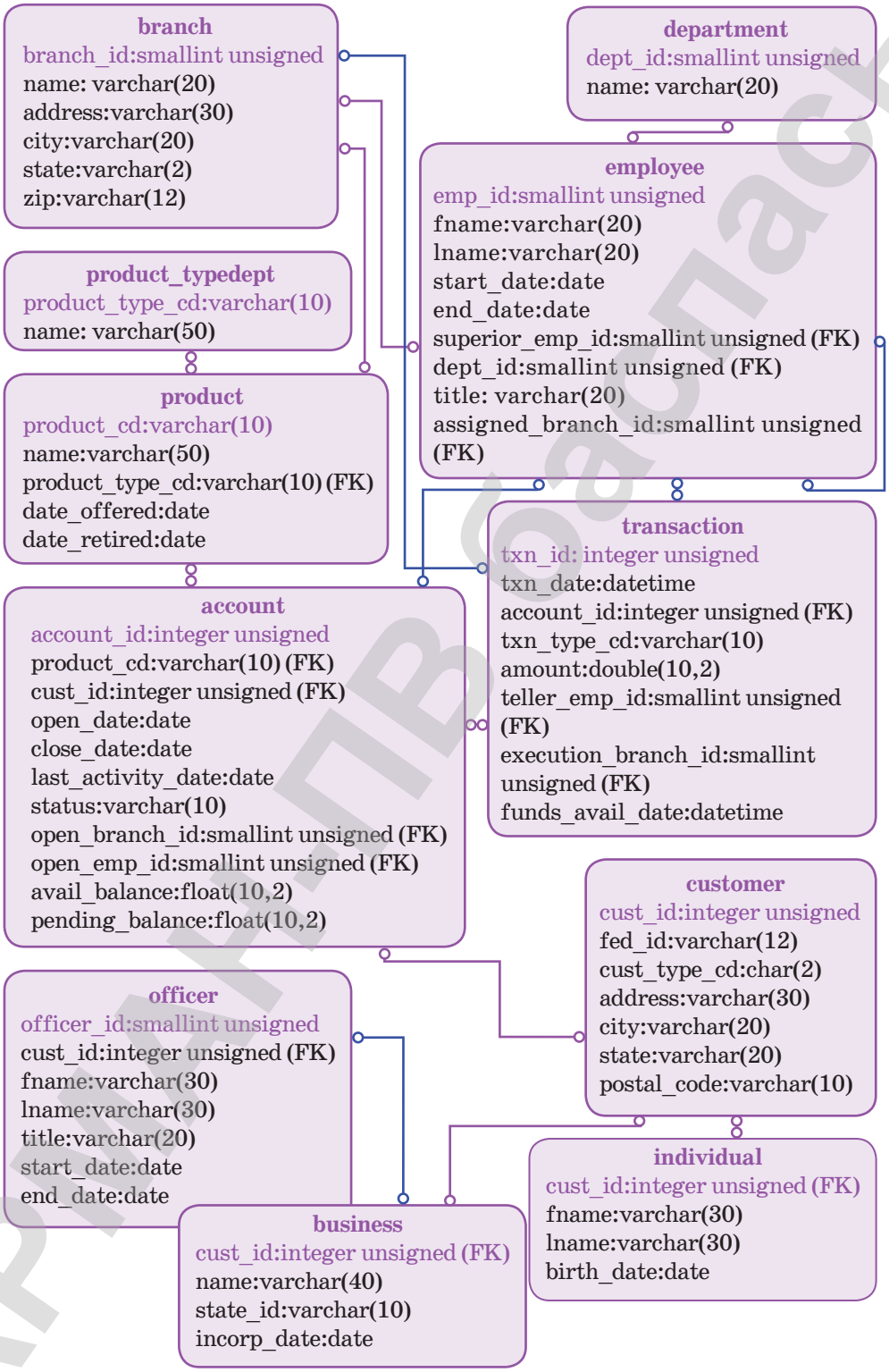


4 Выполняем в тетради

Запишите понятия данных и типов данных в тетрадях своими словами.

5 Выполняем на компьютере

Создайте таблицы, характеризующие банк, обслуживающий небольшой населенный пункт. Среди этих таблиц должны быть Employee (сотрудник), Branch (отделение), Account (счет), Customer (клиент), Product (услуга), Transaction (транзакция) и Loan (заем). Диаграмма с таблицами, их столбцами и связями выглядит следующим образом:



В таблице показаны все таблицы, используемые в банковской схеме, и даны их краткие описания.

Таблица	Описание
Account	Конкретный счет, открытый для конкретного клиента
Business	Клиент – юридическое лицо (подтип таблицы Customer)
Customer	Физическое или юридическое лицо, известное банку
Department	Группа сотрудников банка, реализующая определенную банковскую функцию
Employee	Человек, работающий в банке
Individual	Клиент – юридическое лицо (подтип таблицы Customer)
Officer	Человек, которому разрешено вести дела от лица клиента – юридического лица
Product	Услуга банка, предлагаемая клиентам
Product, type	Группа функционально схожих услуг
Transaction	Группа функционально схожих услуг

6 **Делимся мыслями**

На каком языке вы работаете в базе данных в повседневной жизни?

§ 61–62. Формы

Вспомните!

- *Что такое типы данных в базе данных?*
- *Как создать однотабличную базу данных?*
- *Как создать многотабличную базу данных?*

Вы узнаете:

- *как создавать форму для ввода данных.*

Форма – это объект, позволяющий организовать комфортный для пользователя интерфейс для ввода, отображения и редактирования данных таблиц и запросов. Формы для ввода данных, как правило, соответствуют разработанным на этапе проектирования шаблонам, эквивалентным принятым в организации формам документов. Можно сказать, что формы – это «пользовательский канал» передачи данных в таблицы или пользовательская среда доступа к данным. Таким образом, пользователь находится в среде привычных для себя документов и не тратит время на изучение незнакомых объектов, таких как таблицы и схемы данных.

При необходимости форма может иметь одно или несколько подключений к данным получателя. В зависимости от целей пользователя формы могут служить для запроса или отправки данных формы к внешнему источнику данных, например, базе данных Microsoft SQL Server или web-службе подключения к данным.

При разработке шаблона формы, основанного на базе Microsoft SQL Server, создается основной источник данных с группами, которые содержат поля запроса и поля данных, и подключение данных для запроса в качестве основного подключения данных для шаблона формы. Этим полям и группам соответствуют способы хранения данных в таблицах БД. Поля запроса с данными, введенными пользователем, чтобы ограничить результаты запроса к записям, соответствуют данным в полях запроса. Тогда формы на основе этого шаблона формы используют основное подключение к данным.

Создание запроса с использованием данных в полях запроса. Microsoft SQL Server отправляет запрос через подключение данных для запроса. База данных возвращает результаты запроса обратно в форму через подключение данных для запроса. Результаты запроса помещаются в поля данных, которые можно редактировать по элементам управления, привязанных к этим полям.

Так как структуры данных запроса и полей данных должны соответствовать тем данным, которые хранятся в базе данных, нельзя изменять эти поля или группы в основном источнике данных. Поля или группы можно добавлять только в корневую группу основного источника данных.

Формы могут быть отправлены в базу данных посредством базового соединения, для которого шаблон формы должен удовлетворять следующим требованиям:

- Microsoft SQL Server не создаст подключения для отправки данных в основное подключение к данным, если вы разрабатываете шаблон формы с поддержкой браузера. Для того, чтобы получить доступ к информации по форме, необходимо создать шаблон формы для отправки информации пользователям с помощью web-служб, работающих в базе данных.
- Первый столбец слева от таблицы является первичным ключом как минимум для одного соединения с парой связанной таблицы.
- Если в запросе хранятся большие данные двоичного типа, например, изображения, объекты OLE, встроенные файлы, Microsoft SQL Server удаляет передачу данных.

Если Microsoft SQL Server разрешает подключение к передаче данных, можно настроить параметры передачи данных для форм.

Совместимость

При разработке шаблона формы, основанного на базе данных, имеется возможность разработки шаблона формы с поддержкой web-браузера. Microsoft SQL Server создаст подключение данных для запроса как основное подключение к данным в шаблоне формы с поддержкой браузера. Тем не менее, чтобы пользователи могли отправлять данные в базу данных, невозможно настроить шаблоны форм с поддержкой браузера. Таким образом, если вы разрабатываете базы данных SQL Server на основе шаблона формы и хотите отправлять пользователям свои данные в базе данных через основное подключение к данным, нельзя внести шаблон формы, совместимый с обозревателем.

Для создания базы данных SQL Server на основе шаблона формы администратору базы данных необходимы указанные ниже сведения:

1) имя сервера, содержащего базу данных, к которой будет подключен шаблон формы;
2) имя базы данных, которая будет использоваться с этим шаблоном формы;
3) проверка подлинности в базе данных. Чтобы определить способ доступа пользователей к базе данных, можно использовать либо проверку подлинности Microsoft Windows либо SQL Server;
4) имя таблицы, содержащей данные, которые вы хотите отправить, или формы, которая будет получать данные из формы. Это главная таблица. Если вы собираетесь использовать несколько таблиц в базе данных, необходимы имена других пользователей и дочерние таблицы. Вам также понадобятся имена полей в дочерней таблице, связанные с полями в главной таблице.

Разработка шаблона формы

Для разработки шаблона формы запроса с подключением к базе данных, нужно сделать следующее:

1. **Создание шаблона формы.** При создании базы данных на основе шаблона формы Microsoft SQL Server создает подключение для запроса данных в качестве основного подключения между шаблоном формы и базой данных. Этот процесс автоматически создает шаблон формы основного источника данных.
2. **Добавление одного или нескольких элементов управления для отображения результатов запроса.** Чтобы разрешить пользователям просматривать и изменять данные в полях основного источника данных при открытии формы, можно добавить элемент управления в шаблоне формы и затем привязать его к полю основного источника данных.

1 Отвечаем на вопросы

1. Что такое форма?
2. Как выполняется запрос через форму?
3. Какая информация необходима для создания базы данных?
4. Какое соединение необходимо для создания формы?
5. Что нужно сделать, чтобы создать шаблон формы?

2 Думаем и обсуждаем

1. В чем главная особенность создания формы?
2. В чем необходимость создания форм для ввода данных?
3. Чем важно соединение при создании шаблона формы?

3 Анализируем и сравниваем

Запишите особенности создания шаблона формы в таблицу. Проанализируйте и сравните их.

Выполняемое действие	Особенность
Создание шаблона формы	
Добавление одного или нескольких элементов управления для отображения результатов запроса	

4 Выполняем в тетради

Напишите требования, которым должен соответствовать шаблон формы.

1. _____
2. _____
3. _____
4. _____

5 Выполняем на компьютере

Создание шаблона формы.

1. В меню выберите пункт **Создание шаблона формы.**
2. В разделе **Разработка нового,** в диалоговом окне **Создание шаблона формы** щелкните **Шаблон формы.**
3. В списке **На основе** выберите **Базу данных.**
4. Если вы разрабатываете шаблон формы с поддержкой браузера, установите флажок **Включить только возможности, совместимые с обозревателем.**
5. Нажмите кнопку **ОК.**
6. В мастере подключения данных нажмите кнопку **Выбор базы данных.**
7. В диалоговом окне **Выбор источника данных** выберите **Создать источник.**

8. Щелкните **Microsoft SQL Server** в списке **Выберите тип источника данных, к которым нужно подключиться**, и нажмите кнопку **Далее**.
9. В поле **Имя сервера** введите имя сервера с базой данных SQL Server.
10. В разделе **Ввод учетных данных** выполните одно из указанных ниже действий.
    1) Если база данных определяет, кто имеет доступ, на основании учетных данных, используемых в сети Microsoft Windows, нажмите кнопку **Использовать проверку подлинности Windows**.
    2) Если база данных определяет, кто имеет доступ, на основе заданного имени пользователя и пароля, полученного от администратора базы данных, установите флажок **Использовать следующие имя пользователя и пароль**, а затем введите их в соответствующие поля.
11. Нажмите кнопку **Далее**.
12. В списке **Выберите базу данных, которая содержит нужные данные,** выберите имя базы данных, которую вы хотите использовать, установите флажок **Подключиться к определенной таблице**, щелкните имя главной таблицы и нажмите кнопку **Далее**.
13. На следующей странице мастера введите имя файла, в котором хранятся сведения о подключении данных в поле **Имя файла** и нажмите кнопку **Готово**, чтобы сохранить эти параметры.
14. Нажмите кнопку **Далее**.
15. На последней странице мастера введите имя для основного подключения данных. Это имя будет отображаться в списке **Источник данных**.

6 **Делимся мыслями**

Какие формы ввода данных применяются, когда правила формы не соответствуют шаблонам дизайна, используемым при организации формы? Поделитесь мыслями с одноклассниками.

§ 63–64. Запросы

Вспомните!

- *Что такое форма?*
- *Как создать форму для ввода данных?*

Вы узнаете:

- *как создавать запросы, используя извлеченные данные.*

Как сервер MySQL выполняет запросы?

Запрос – это требование на получение определенной информации из БД. Клиентская программа командной строки MySQL при входе в базу данных проверяет правильность имени пользователя и пароля, после этого создается соединение с БД. Это соединение удерживается запросившим его приложением до тех пор, пока приложение не сбросит соединение или пока соединение не будет закрыто сервером.

После того, как сервер открыл соединение, проверив достоверность имени пользователя и пароля, можно выполнять запросы и другие выражения SQL. При каждом запросе перед выполнением выражения сервер проверяет следующее:

- Есть ли у вас разрешение на выполнение выражения?
- Есть ли у вас разрешение на доступ к необходимым данным?
- Правилен ли синтаксис выражения?

Если выражение проходит все три теста, оно передается *оптимизатору запросов*, работа которого заключается в определении наиболее эффективного способа выполнения запроса. Оптимизатор рассмотрит порядок соединения таблиц, перечисленных в запросе, и доступные индексы, а затем определит план выполнения, используемый сервером при выполнении этого запроса.

По завершении выполнения запроса сервер возвращает результирующий набор (result set) в вызывающее приложение (инструмент MySQL).

Блоки запроса

Выражение `select` может образовывать несколько компонентов, или блоков (`clauses`). Хотя при работе с MySQL обязательным является только один из них – блок `select` – обычно в запрос включаются по крайней мере два-три из шести доступных блоков. В *таблице 22* показаны разные блоки и их назначения.

Таблица 22. Блоки запроса

Блок	Назначения
Select	Определяет столбцы, которые должны быть включены в результирующий набор запроса
From	Указывает таблицы, из которых должны быть извлечены данные, и то, как эти таблицы должны быть соединены
Where	Ограничивает число строк в окончательном результирующем наборе
Group by	Используется для группировки строк по одинаковым значениям столбцов
Having	Ограничивает число строк в окончательном результирующем наборе с помощью группировки данных
Order by	Сортирует строки окончательного результирующего набора по одному или более столбцам

Блок select

Несмотря на то, что блок `select` является первым в выражении `select`, сервер БД обрабатывает его одним из последних. Причина в том, что прежде чем можно будет определить, что включать в окончательный результирующий набор, необходимо знать все столбцы, которые могли бы быть включены в этот набор. Поэтому, чтобы полностью понять роль блока `select`, нужно разобраться с блоком `from`. Изначальный запрос выглядит следующим образом:

```
Mysql > SELECT *
   > FROM department;
+---------------+----------------+
| dept_id       | name           |
+---------------+----------------+
| 1             | Operations     |
| 2             | Loans          |
| 3             | Administration |
+---------------+----------------+
 3 rows in set (0.04 sec)
```

В данном запросе в блоке `from` указана всего одна таблица (`department`) и блок `select` показывает, что в результирующий

набор должны быть включены все столбцы (это обозначено символом «*») таблицы department. Этот запрос можно перевести на естественный язык следующим образом:

Покажи мне все столбцы таблицы department

Таким образом, задача блока select заключается в следующем: блок select определяет, какие из всех возможных столбцов должны быть включены в результирующий набор запроса.

Блок from

Блок from определяет таблицы, используемые запросом, а также средства связывания таблиц.

При встрече с термином table большинство людей представляют себе набор взаимосвязанных строк, хранящихся в базе данных. Этому свободному определению соответствуют три разных типа таблиц:

1) постоянные таблицы (т. е. созданные с помощью выражения create table);
2) временные таблицы (т. е. строки, возвращенные подзапросом);
3) виртуальные таблицы (представления) (т. е. созданные с помощью выражения create view).

Каждый из этих типов таблиц может быть включен в блок запроса from.

Блок where

Блок where – это механизм отсеивания нежелательных строк из результирующего набора.

Например, из таблицы employee требуется извлечь данные, но только для сотрудников, нанятых в качестве старших сотрудников (head tellers). В данном запросе блок where служит для извлечения только четырех старших сотрудников:

```
mysql> SELECT emp_id, fname, lname, start_date,
title
    > FROM employee
    > WHERE title = 'Head Teller';
```

```
+--------+---------+---------+------------+--------------+
| emp_id | fname   | lname   | start_date | title        |
+--------+---------+---------+------------+--------------+
| 6      | Helen   | Fleming | 03 10 2019 | Head Teller  |
| 10     | Paula   | Roberts | 03 10 2019 | Head Teller  |
| 13     | John    | Blake   | 03 10 2019 | Head Teller  |
| 16     | Theresa | Markham | 03 10 2019 | Head Teller  |
+--------+---------+---------+------------+--------------+
4 rows in set (0.00 sec)
```

Блоки group by и having

Все рассмотренные до сих пор запросы извлекали необработанные строки данных, не выполняя над ними никаких действий. Однако иногда возникает необходимость выявить в данных общие закономерности, прежде чем предоставить вам результирующий набор. Одним из средств выполнения таких действий является блок `group by`, предназначенный для группировки данных по значениям столбцов. Например, вместо списка сотрудников и отделов, в которых они числятся, нужен список отделов с числом сотрудников, работающих в каждом из них. С блоком `group by` также можно использовать блок `having`, позволяющий фильтровать данные групп аналогично блоку `where`, позволяющему фильтровать необработанные данные.

Блок order by

Обычно строки результирующего набора запроса возвращаются в произвольном порядке. Если требуется упорядочить результирующий набор определенным образом, необходимо предписать серверу сортировать результаты с помощью блока `order by`.

Блок `order by` – это механизм сортировки результирующего набора на основе данных столбцов или выражений, использующих данные столбцов.

```
mysql>SELECT open_emp_id, product_cd
    > FROM account;
```

```
mysql > SELECT open_emp_id, product_cd
    > FROM account
    > ORDER BY open_emp_id;
```

open_emp_id	product_cd
10	CHK
10	SAV
10	CD
10	CHK
10	SAV
13	CHK
13	MM
1	CHK
1	SAV
1	MM
16	CHK
1	CHK
1	CD
10	CD
16	CHK
16	SAV
1	CHK
1	MM
1	CD
16	CHK
16	BUS
10	BUS
16	CHK
13	SBL

24 rows in set (0.00 sec)

open_emp_id	product_cd
1	CHK
1	SAV
1	MM
1	CHK
1	CD
1	CHK
1	MM
1	CD
10	CHK
10	SAV
10	CD
10	CHK
10	SAV
10	CD
10	BUS
13	CHK
13	MM
13	SBL
16	CHK
16	CHK
16	SAV
16	CHK
16	BUS
16	CHK

24 rows in set (0.00 sec)

Сортировка по возрастанию и убыванию

При сортировке можно задать порядок по возрастанию (ascending) и по убыванию (descending) с помощью ключевых слов asc и desc.

Сортировка по возрастанию выполняется по умолчанию, поэтому добавлять нужно только ключевое слово desc – если требуется сортировка по убыванию. Например, по следующему запросу выводится список всех счетов, отсортированный по доступному остатку, начиная с самого большого:

```
mysql> SELECT account_id, product_cd, open_date,
avail_balance
    -> FROM account
    -> ORDER BY avail_balance DESC;
```

account_id	product_cd	open_date	avail_balance
24	SBL	03 10 2019	50000.00
23	CHK	03 10 2019	38552.05
20	CHK	03 10 2019	23575.12
13	CD	03 10 2019	10000.00
22	BUS	03 10 2019	9345.55
18	MM	03 10 2019	9345.55
10	MM	03 10 2019	5487.09

1 Отвечаем на вопросы

1. Что такое запросы?
2. Какие условия должны быть удовлетворены для выполнения запросов?
3. Из каких блоков состоит запрос?
4. Какую функцию выполняет блок `select`?
5. Для чего используется блок `from`?
6. Какую функцию выполняет блок `where`?
7. Каковы назначения блоков `group by` и `having`?
8. Какими методами блок `order by` сортирует данные?

2 Думаем и обсуждаем

1. Насколько важно пользоваться запросами при работе с базами данных?
2. В чем необходимость блоков запроса?

3 Анализируем и сравниваем

Проанализируйте функции, выполняемые следующими блоками, и сравните их между собой.

Блок	Функция
`Select`	
`From`	
`Where`	
`Group by`	
`Having`	
`Order by`	

4 Выполняем в тетради

Приведите примеры для ключевых слов `asc` и `desc`, которые используются при сортировке по возрастанию (`ascending`) или убыванию (`descending`).

5 Выполняем на компьютере

Задание 1. Извлеките ID, имя и фамилию всех банковских сотрудников. Выполните сортировку по фамилии, затем по имени.

Задание 2. Извлеките ID счета, ID клиента и доступный остаток всех счетов, имеющих статус `'ACTIVE'` (активный) и доступный остаток более 2500 тенге.

Задание 3. Напишите запрос к таблице account, возвращающий ID сотрудников, открывших счета (используйте столбец `account.open_emp_id`). Результирующий набор должен включать по одной строке для каждого сотрудника.

Задание 4. В этом запросе к нескольким наборам данных заполните пробелы (обозначенные, как `<число>`) так, чтобы получить результат, приведенный ниже:

```
mysql> SELECT p.product_cd, a.cust_id, a.avail_
   balance
      > FROM product p INNER JOIN account
      > ON p.product_cd = <2>
      > WHERE p. <3> = 'ACCOUNT';
```

6 Делимся мыслями

Насколько важна эффективность выполняемых запросов по базам данных? Поделитесь мыслями со своими одноклассниками.

§ 65–66. Отчеты

Вспомните!

- *Как создавать запросы, используя извлеченные данные?*
- *Какие блоки запросов вы знаете?*

Вы узнаете:

- *как создавать отчеты, используя извлеченные данные.*

Отчет представляет собой форматированное сочетание данных из одной или нескольких таблиц, выводимое на устройство отображения (экран, принтер) либо в файл. Обычно в основе отчета лежат записи таблиц БД или запросов. Также в отчет могут быть включены схемы и диаграммы, в том числе фотографии и иллюстрации, графические элементы управления, верхние и нижние колонтитулы, содержащие служебную информацию (*рис. 27*).

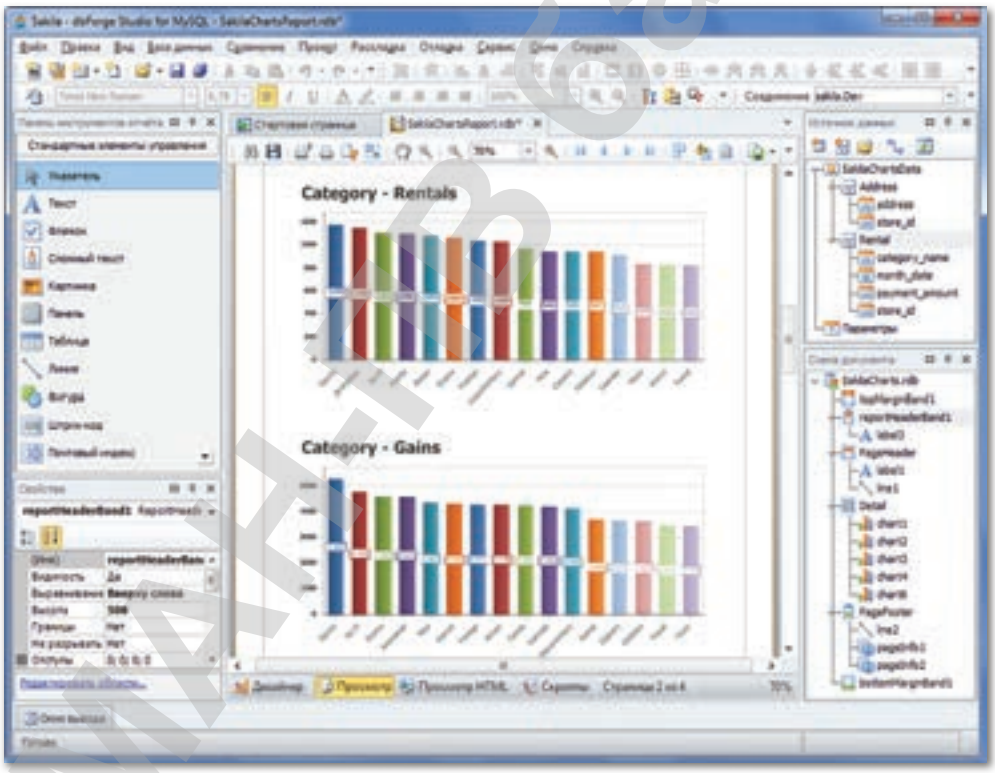


Рис. 27. *Тип отчета*

Системы управления базами данных разрешают обращаться к данным и изменять их не только одному пользователю, но и нескольким одновременно. Если каждый пользователь

выполняет запросы, как это происходит с хранилищем данных в течение обычных рабочих часов, для сервера БД это не создает больших проблем. Однако если некоторые пользователи добавляют и/или изменяют данные, серверу приходится сохранять довольно много промежуточных результатов.

Например, создается отчет о доступном остатке всех текущих счетов, открытых в отделении. Однако одновременно с выполнением отчета происходит следующее:

- Служащий отделения обрабатывает вклад для одного из клиентов.
- Клиент заканчивает снимать деньги с банкомата в операционном зале.
- Банковское приложение, выполняющееся в конце каждого месяца, начисляет процент по счетам.

Следовательно, пока создается отчет, несколько пользователей изменяют данные. Какие цифры должны появиться в отчете? Ответ отчасти зависит от того, как сервер реализует блокировку (`locking`) – механизм управления одновременным использованием ресурсов данных. Большинство серверов БД применяют одну из двух стратегий блокировки:

1. Пользователи, осуществляющие запись в БД, должны запросить и получить от сервера блокировку записи (write lock) для изменения данных. А пользователи, считывающие данные из БД, должны запросить и получить от сервера блокировку чтения (`read lock`) для осуществления запросов к данным. В то время как чтение может осуществляться одновременно несколькими пользователями, для каждой таблицы (или ее части) одновременно выдается только одна блокировка записи, и запросы на чтение блокируются до тех пор, пока не будет снята блокировка записи.
2. Пользователи, осуществляющие запись в БД, должны запросить и получить от сервера блокировку записи для изменения данных, но пользователи, считывающие данные, не нуждаются ни в каком типе блокировки. Вместо этого сервер гарантирует, что пользователь видит непротиворечивое представление данных (данные представляются неизменными, несмотря на то, что другие пользователи могут их модифицировать), начиная с момента начала запроса до его завершения. Этот подход известен как контроль версий (`versioning`).

У обеих стратегий есть свои достоинства и недостатки. При первом подходе время ожидания может оказаться длительным, если одновременно поступило множество запросов на чтение и запись. Второй подход может создать проблемы в случае продолжительных запросов, поскольку происходит изменение данных.

Также есть ряд различных стратегий блокировки ресурса (*схема 7*). Блокирование может выполняться на одном из трех разных уровней, или с одной из трех детализаций (granularities).

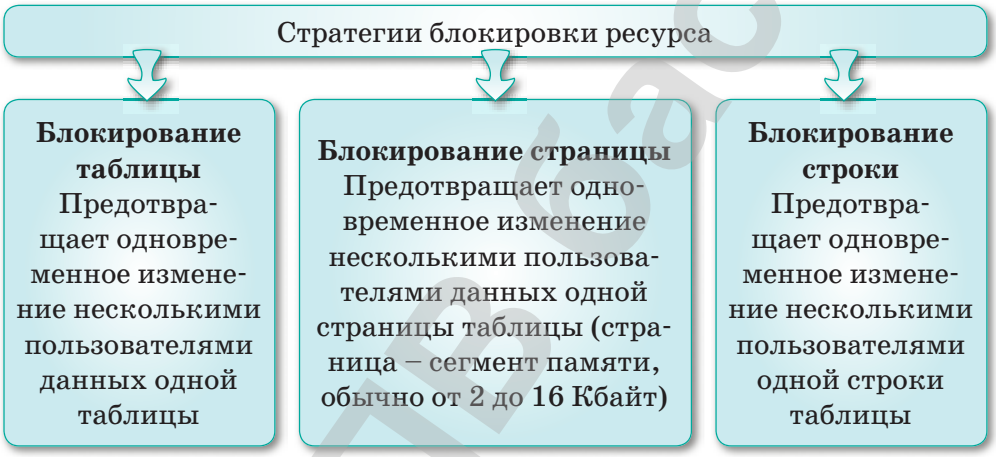


Схема 7. *Стратегия блокировки ресурса*

Данные, появляющиеся на страницах отчета, будут отражать состояние БД или на момент начала создания отчета (если сервер использует контроль версий), или на момент осуществления сервером блокировки чтения (если сервер использует блокировки и чтения, и записи).

В системах, используемых для создания отчетов или наборов данных, часто встречаются следующие запросы:

```
mysql    > SELECT p.name product, b.name branch,
    > CONCAT(e.fname, ' ', e.lname) name,
      > SUM(a.avail_balance) tot_deposits
      > FROM account a INNER JOIN employee e
      > ON a.open_emp_id = e.emp_id
      > INNER JOIN branch b
      > ON a.open_branch_id = b.branch_id
```

```
    > INNER JOIN product p > ON a.product_cd =
p.product_cd
    > WHERE p.product_type_cd = 'ACCOUNT'
    > GROUP BY p.name, b.name, e.fname, e.lname;
```

product	branch	name	tot_deposits
certificate of deposit	Headquarters	Michael Smith	11500.00
certificate of deposit	Woburn Branch	Paula Roberts	8000.00
checking account	Headquarters	Michael Smith	782.16
checking account	Quincy Branch	John Blake	1057.75
checking account	So. NH Branch	Theresa Markham	67852.33
checking account	Woburn Branch	Paula Roberts	3315.77
money market account	Headquarters	Michael Smith	14832.64
money market account	Quincy Branch	John Blake	2212.50
savings account	Headquarters	Michael Smith	767.77
savings account	So. NH Branch	Theresa Markham	387.99
savings account	Woburn Branch	Paula Roberts	700.00

11 rows in set (0.02 sec)

Формирование отчетов

Существует два наиболее важных навыка, необходимых для разработки и формирования отчетов для какой-либо организации:

- Знание возможностей механизма создания отчетов, используемых в организации.
- Совершенное владение реализацией SQL, используемой сервером БД.

Хотя большинство инструментов создания отчетов претендует на формирование SQL на базе визуального представления отчета, настоятельно рекомендуется игнорировать эту возможность и самостоятельно создавать выражения SQL для всех нетривиальных отчетов. В этом случае вы точно будете знать, что отправляется серверу БД, и впоследствии сможете лучше поддерживать и настраивать отчеты. Глубокое понимание SQL, особенно подзапросов, операций работы с множествами и условной логики, позволит создавать гораздо более сложные отчеты.

1 Отвечаем на вопросы

1. Что такое отчеты? В какой сфере вы наблюдали широкое использование отчетов?

2. Что входит в структуру отчета?
3. Почему используется несколько видов отчетности?
4. Какова роль функции блокировки при создании отчетов?

2 Думаем и обсуждаем

1. Насколько важно пользоваться услугами отчета в базе данных?
2. Зависит ли информация на страницах отчета от состояния базы данных?
3. Почему используется несколько видов отчетов?

3 Анализируем и сравниваем

Проанализируйте два метода создания отчетов, зависящие от того, как сервер реализует блокировку, и сравните преимущества и недостатки этих методов.

4 Выполняем в тетради

Приведите дополнительную информацию о системах, используемых для создания отчетов или наборов данных, которые часто встречаются в таких запросах, как:

```
mysql > SELECT p.name product, b.name branch,
      > CONCAT(e.fname, ' ', e.lname) name,
      > WHERE p.product_type_cd = 'ACCOUNT'
      > GROUP BY p.name, b.name, e.fname, e.lname;
```

5 Выполняем на компьютере

Создайте модель отчета для отчета.

1. Откройте MS SQL Server.
2. В меню **Файл** выберите **Создать** ⇒ **Проект**.
3. В списке **Шаблоны** выберите **Проект моделей отчетов**.
4. В поле **Имя** введите **Образец модели отчета**.
5. Чтобы создать проект модели отчета, нажмите кнопку **OK**.
6. В области **Обозревателя решений окна MS SQL Server** щелкните правой кнопкой мыши на **Источники данных**, а затем выберите **Добавить новый источник данных**. Нажмите кнопку **Далее**.
7. Убедитесь, что на странице **Выбор метода определения соединения** выбран пункт **Создать источник данных**

на основе существующего или нового подключения, а затем нажмите кнопку **Создать**.

8. В диалоговом окне **Диспетчер подключений** укажите следующие свойства подключения для источника данных:
   - **Имя сервера** – введите имя сервера базы данных MS SQL Server или выберите его в раскрывающемся списке.
   - Выберите **Использовать проверку подлинности Windows**.
   - В раскрывающемся списке **Выберите или введите имя базы данных** выберите имя базы данных MS SQL Server.
9. Чтобы проверить работоспособность подключения базы данных, нажмите **Проверить подключение**.
10. Если подключение установлено, нажмите кнопку **ОК**. Если не удалось установить подключение, убедитесь, что введенные сведения правильны, и снова нажмите **Проверить подключение**.
11. Убедитесь, что на странице **Выбор метода определения соединения** выбран пункт **Создать источник данных на основе существующего или нового подключения**, убедитесь, что в списке **Подключение данных** выбран указанный источник данных, а затем нажмите кнопку **Далее**.
12. В поле **Имя источника данных** введите **Пример модели отчета** и нажмите кнопку **Готово**.
13. В поле **Обозреватель решений** щелкните правой кнопкой мыши на **Представление источника данных** и выберите команду **Добавить новый источник данных**.
14. На странице **Приветствие мастера представления источника данных** нажмите кнопку **Далее**.
15. На странице **Выбор источника данных** убедитесь, что в окне **Реляционные источники данных** выбран источник данных **Пример модели отчета** и нажмите кнопку **Далее**.
16. На странице **Выбор таблиц и представлений** выберите таблицы и представления из базы данных MS SQL Server, которые будут использоваться в модели отчета. Нажмите кнопку **Далее**.

17. На странице **Завершение работы мастера** в поле **Имя** введите **Пример модели отчета** и нажмите кнопку **Готово**.
18. В обозревателе решений щелкните на **Модель отчетов** правой кнопкой мыши и выберите команду **Добавить новую модель отчета**.
19. На странице **Приветствие мастера моделей отчетов** нажмите кнопку **Далее**.
20. Убедитесь, что на странице **Выбор представлений источника данных** в списке **Доступные представления источника данных** выбран **Пример модели отчета.dsv**, а затем нажмите кнопку **Далее**.
21. На странице **Выбор правил формирования модели отчета** назначьте значения **По умолчанию** и нажмите кнопку **Далее**.
22. Убедитесь, что на странице **Завершение работы мастера** в поле **Имя** отображено **Пример модели отчета**.
23. Чтобы завершить работу мастера и создать модель отчета, нажмите **Выполнить**.

6 Делимся мыслями

Как вы думаете, как системы управления базами данных разрешают обращаться к данным и изменять их не только одному пользователю, но и нескольким одновременно? Поделитесь мыслями со своими одноклассниками.

§ 67–68. Связь web-страницы с базой данных

Вспомните!

- *Что такое отчеты в базе данных?*
- *Как создавать отчеты?*
- *О функциях блока* `Select`;
- *О функциях блока* `From`;
- *О блоках* `Group by` *и* `Having`;

Вы узнаете:

- *о базе данных MySQL;*
- *о преимуществах и недостатках MySQL;*
- *о языке PHP.*

Термины:

- MySQL;
- PHP.

PhpMyAdmin – это web-программа, написанная на языке PHP, а также web-интерфейс, предназначенный для управления системой MySQL. Через PhpMyAdmin вы можете управлять сервером MySQL с помощью браузера, выполнять команды SQL и редактировать записи в таблицах базы данных.

Одной из главных причин широкого применения PhpMyAdmin является легкое управление системой MySQL с помощью данного интерфейса без необходимости набора операторов SQL вручную.

PhpMyAdmin – это управление базами данных, не требующее мастерства и большого количества работы. MySQL – свободная система управления базами данных, предназначенная для малого и среднего бизнеса.

Сервер баз данных MySQL – это система быстрого и мощного управления распределенными базами данных. Он позволяет эффективно хранить, искать, сортировать и выбирать информацию, и работает на UNIX-сервисах с многопоточным соединением. MySQL является бесплатным для некоммерческого использования.

Возможности MySQL

MySQL – использует запросы SQL в стандартах ANSI 92. Он имеет следующие возможности:

1. Неограниченное количество пользователей могут одновременно работать с базами данных.
2. Количество строк в таблицах может достигать 50 млн.
3. Команды выполняются очень быстро. Среди существующих в настоящее время серверов MySQL является самым быстрым.
4. Простая и эффективная безопасная система.

Преимущества:

- По сравнению с файловыми версиями web-приложений количество кода уменьшается в 2–3 раза. Это экономит время его сборки и облегчает процесс обработки.
- Так как программа написана на языке С, процедуры запроса СУБД выполняются с высокой скоростью.

Пример 1. Создание таблиц и баз данных MySQL в сценариях PHP.

Переход к управлению данными с помощью запросов MySQL, исполненных в сценариях PHP.

Для это на локальном диске создаем файл: */WebServers/home/localhost /www/my_phone create.php:*

1. Набираем в Блокноте данный код.

```
<html> КГУ ВО №165 им Абая <body><br>
кафедра информатики <br> <br>
<hr>
<b>лабораторная работа </b><br>
<?php
$key1 =0; echo "<b>создание базы данных в сервере
mysql </b> "."<br>"."<hr>";
$sdb_name = "localhost";
$user_name = "root"; $user_pass = " ";
$db_name = "phone";
// связь с сервером
$link=mysql_connect($sdb_name,$user_name,$user_
pass); echo "report mysql-server"."<br>";
if (!$link)

{echo "связь с mysql-server потеряна <br>";exit();}
echo "установлена связь с mysql-server <br> <hr>";
// создание базы данных
$str_sql_query = "create database $db_name"; echo
"message:".$str_sql_query;
if (!mysql_query($str_sql_query, $link))
{echo "<br> база данных не создана!!! <br>";}
// выбрать базу данных
```

```
if (!mysql_select_db($db_name,$link))
{echo "Вы не выбрали базу данных."."<br>";
exit();} echo "<br> Вы выбрали базу данных <br>";
//Создать таблицу
mysql_query("create table tb1(nom text(20), fam
text(50))") or die ("<br> ошибка при создании таблицы
<br>". mysql_error()); mysql_close($link);
?> </body> </html>
```

2. Запустите **Start Denwer**.
3. Вызовите сценарий в адресной строке браузера (например: *localhost / MY_PHONE / create.php*), проверьте его работу.
4. Для создания таблиц и названий полей в этой базе данных зайдите в базу данных MySQL и в папке MY_PHONE1 запустите сценарий Create1.php для нового проекта.
5. При копировании текста сценария из Word помните, что текст длинной строки в Word автоматически переносится на следующую строку.
6. В сценарии же текст от начала до отметки [;], считается одной строкой.

Пример 2. Взаимодействие с записями базы данных.

```
   if (!mysql_select_db($db_name,$link))
{echo "Not find for use data base"."<br>"; exit();
}
echo "Open data base - Yes"."<br>";
// Спрос на таблицу
$str_sql_query = "SELECT * FROM Tb1";
// Спрос на таблицу
if (!$result = mysql_query($str_sql_query,$link))
{
echo "Not RUN query Tb1"."<br>"; exit();
} echo "Query Table data base - Yes"."<br>";
$newnom = '203040';
$newfam = 'Фамилия5';
// Ввод данных
//
```

```
$str_sql_query = "INSERT INTO Tb1 SET
Nom = 'Новый номер',
Fam='Новая Фамилия'";
//
$str_sql_query = "INSERT INTO Tb1 SET
     Nom = '$newnom', Fam = '$newfam'";
//
//$str_sql_query= "INSERT INTO Tb1(nom, fam)VALUES
('$newnom','$newfam')";
// Обновление данных – Изменение номера
// $str_sql_query = "UPDATE Tb1    SET fam = 'Изме-
нено' WHERE nom = '55555555'";
// Удаление данных
// $str_sql_query = "DELETE FROM Tb1 WHERE nom = '55555555' ";
(!mysql_query($str_sql_query,$link))
```

1 Отвечаем на вопросы

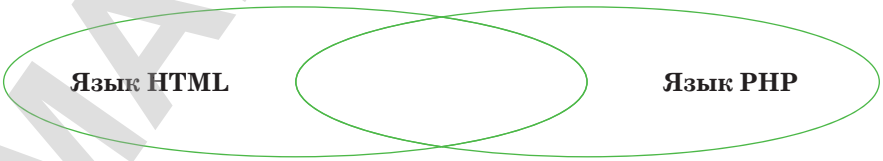
1. Что такое MySQL?
2. Где используется язык PHP?
3. Какие распространенные web-браузеры вы знаете? Какой из них используете вы?

2 Думаем и обсуждаем

1. Чем связаны базы данных MySQL и язык PHP?
2. Почему PhpMyAdmin используется чаще?

3 Анализируем и сравниваем

Сравните языки HTML и PHP.



4 Выполняем в тетради

Найдите информацию о базах данных Access, MySQL, проанализируйте и запишите их характеристики в следующей таблице.

Создание запроса в базах данных	
Access	MySQL

5 Выполняем на компьютере

Чтобы получить доступ к базе данных из Web, используя PHP, необходимо выполнить следующие действия:

1. Подключение к серверу MySQL.
2. Выбор базы данных.
3. Выполнение запроса к базе данных:
   - добавление;
   - удаление;
   - изменение;
   - поиск;
   - сортировка.
4. Получение результата запроса.
5. Отсоединение от базы данных.

Для подключения к серверу базы данных в PHP есть функция `mysql_connect( )`. Ее аргументы: имя компьютера, имя пользователя и пароль. Эти аргументы можно опустить. Имя компьютера по умолчанию = `localhost`, тогда имя пользователя и пароль не требуется. Если PHP используется в сочетании с сервером Apache, то можно воспользоваться функцией `mysql_pconnect()`. В этом случае соединение с сервером не исчезает после завершения работы программы или вызова функции `mysql_close()`. Функции `mysql_connect()` и `mysql_pconnect()` возвращают идентификатор подключения, если оно прошло успешно. Например:

```
$link = mysql_pconnect ();
if ( !$link ) die ("Невозможно подключение
   к MySQL");
```

После того, как соединение с сервером MySQL установлено, нужно выбрать базу данных. Для этого используется функция mysql_select_db(). Ее аргумент: имя базы данных. Функция возвращает true, если указанная база данных существует и доступ к ней возможен.

Например:

```
$db = "sample";
mysql_select_db ( $db ) or die ("Невозможно открыть
   $db");
```

Для добавления, удаления, изменения и выбора данных нужно сконструировать и выполнить запрос SQL. Для этого в языке PHP существует функция `mysql_query()`. Ее аргумент: строка с запросом. Функция возвращает идентификатор запроса.

Пример:

```
<html>
<head>
<title> Добавление записи в таблицу</title>
</head>
<body>
<?php
$db = "sample";
$link = mysql_pconnect ();
if ( !$link )
      die ("Невозможно подключение к MySQL");
      mysql_select_db ( $db ) or die ("Невозможно
   открыть $db");
$query = "INSERT INTO books
      VALUES ('966-7393-80-1', 'Аллен Вайк',
      'PHP. Справочник', '213', '4')";
mysql_query ( $query );
mysql_close ( $link );
?>
</body>
</html>
```

При каждом выполнении примера в таблицу будет добавляться новая запись, содержащая одни и те же данные.

Задания:

1. Создать HTML-форму для добавления новых книг в базу данных.
2. Создать HTML-форму для поиска определенных книг в базе данных.

6 Делимся мыслями

Что вы узнали на уроке? Чему вы научились? Поделитесь своими мыслями с друзьями. Как вы думаете, насколько важно использование My SQL в создании совместимых для работы с ним web-страниц? Приведите пример.

Глоссарий

Аутентификация – прием и проверка сервером личных данных пользователя.

Вес вершины – число (действительное, целое или рациональное), соответствующее данной вершине (интерпретируется как стоимость, пропускная способность и т. д.).

Взвешенный граф – граф, каждому ребру которого соответствует некое значение (вес ребра).

Глобальные переменные – объявляются в начале, до объявления любых подпрограмм, программ типов данных, переменных и констант.

Глобальные сети – это сети, предназначенные для объединения отдельных компьютеров и локальных сетей, расположенных на значительном удалении (сотни и тысячи километров) друг от друга.

Граф – это совокупность двух конечных множеств: множества точек и множества линий, попарно соединяющих некоторые из этих точек.

Домен – это символьное имя, разделенное точкой.

Защита информации – комплекс мер, направленных на обеспечение безопасности информации.

Идентификация – это ввод личных данных пользователя, известных только ему.

Информационная безопасность – это процесс, обеспечивающий доступность, целостность и конфиденциальность информации.

Конфиденциальность информации – ее нарушение приводит тому, что информация становится известной тем людям, которые не должны о ней знать, что может повлечь за собой распространение секретной информации.

Криптоанализ – наука о методах и способах расшифровки зашифрованной информации.

Криптография – наука о методах шифрования информации.

Логика – это наука о видах и законах человеческого мышления, включая закономерности высказываний, которые можно доказать.

Маршрутизатор – это интеллектуальное («умное») устройство, связывающее две или более сети для доставки пакетов.

Матрица инцидентности – это двумерный массив размерности n x m, в котором указываются связи между инцидентными элементами графа (ребро и вершина).

Мультиграф – это граф, у которого любые две вершины соединены более чем одним ребром.

Неориентированный граф – граф, у которого все ребра не ориентированы, т.е. ребрам которого не задано направление.

Объект – это единая конструкция, состоящая из множества данных и функций, или в терминологии JavaScript – это набор свойств и способов.

Простой граф – это граф, в котором нет ни петель, ни кратных ребер.

Процедура – вспомогательный алгоритм, который выполняет несколько действий.

Пузырьковая сортировка – это метод последовательного сравнения массивов и списков для их сортировки, который меняет местами соседние элементы, если предыдущий элемент больше следующего элемента.

Система счисления – это совокупность правил записи чисел и арифметических операций над ними.

Смешанный граф – граф, содержащий как ориентированные, так и неориентированные ребра.

Функция – это часть программы, которую вызывают в начале программы.

HTML – язык гипертекстовой разметки. Он определяет набор правил для отображения обычных текстов в виде web-страниц.

IP-адрес – уникальный сетевой адрес, необходимый для нахождения, передачи и получения информации от одного узла к другому.

JavaScript – объектно-ориентированный язык, однако прототипы, используемые в языке, различаются в работе с объектами, что отличает его от традиционных объектно-ориентированных языков.

PhpMyAdmin – это управление базами данных, не требующее мастерства и большого количества работы.

Список использованной и рекомендуемой литературы

1. Божко А.Н. Adobe FrameMaker. Сложная верстка: Учеб. пособие. – Аскери, М.: 2015. – 255 с.
2. Глушаков С.А., Кнабе Г.А., Компьютерная графика. Учебный курс – М.: Фолио, 2010.
3. Иванов В.П., Батраков А.С. Трехмерная компьютерная графика./ Под ред. Полищука. К.М. - М.: Радио и связь, 2015.
4. Компьютерная графика и анимация: А. Калбег – Санкт-Петербург, АСТ, Астрель, 2014 г. – 72 с.
5. Компьютерная графика. Учебник / М.Н. Петров, В.П. Молочков – СПб.: Питер, 2012.
6. Компьютерная графика: Практикум./ Л.А.Залогова – М.: ЛБЗ, 2009.
7. Корриган Дж. Компьютерная графика – М.: ЭНТРОП, 1995.
8. Культин Н.Б., C/C++ в задачах и примерах. БХВ – Петербург, 2012. – 288 с.
9. Маргулис Д. Препресс-ресурсы: Учеб. пособие. – М.: Попурри, 2010. – 256 с.
10. Матвеева Р.В., Трубникова Г.Г., Шифрина Д.А. Основы полиграфического производства: Учеб. пособие. – М.: Книга, 2014. – 312 с.
11. Мэйрин Д., Шэффер Д. Формат PDF в полиграфии: Учеб. Пособие. – М.: ПРИНТ-МЕДИА центр, 2014. – 234 с.
12. Немцова Т. И., Назарова Ю. В. Компьютерная графика и web-дизайн. Практикум: учебное пособие. ИД «ФОРУМ», ИНФРА-М, 2011.
13. Персональный компьютер: настройка и техническая поддержка: Учебное пособие. – Алматы, 2013. – 224 с.: ил.
14. Тутубалдин Д.К., Ушаков Д.А./ Компьютерная графика. Adobe Photoshop: Учеб. Пособие – изд. 2-е – Томск, 2015. – 131 с.
15. Шишкин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистичные изображения. – М.: Диалог-МИФИ, 2015.

Электронные ресурсы

www.web-systems.com.ua
http://nashdesign.org.ua
http://bibliofond.ru/
http://3d.demiart.ru
http://www.f1cd.ru/soft/
http://ru.wikibooks.org/wiki/Blender_3D
http://digital-fantasy.ru
http://blender-school.ru

Содержание

Учебное издание

Гульназ Ибрагимовна Салгараева
Жулдыз Болатхановна Базаева
Айгуль Сейсенбаевна Маханова

ИНФОРМАТИКА

Учебник для 10 класса естественно-математического направления общеобразовательной школы

Главный редактор К. Караева
Редакторы Н. Хасенова, Г. Маликова
Технический редактор В. Бондарев
Художественный редактор Е. Мельникова
Бильд-редактор Ш. Есенкулова
Художник-оформитель О. Подопригора
Дизайн обложки В. Бондарев, О. Подопригора
Дизайнер Е. Молчанова
Верстка Л. Костина

Внимание
При необходимости вы всегда сможете найти CD с электронным приложением на сайте *arman-pv.kz* и загрузить его на свой компьютер для дальнейшей работы

По вопросам приобретения обращайтесь по следующим адресам:
г. Нур-Султан, м-н 4, д.2, кв.55
Тел.: 8 (7172) 92-50-50, 92-50-54. E-mail: astana@arman-pv.kz
г. Алматы, м-н Аксай 1А, д.28Б
Тел./факс: 8 (727) 316-06-30, 316-06-31. E-mail: info@arman-pv.kz

Книжный магазин «Арман-ПВ»
г. Алматы, ул. Алтынсарина, д.87. Тел: 8 (727) 303-94-43.

Сдано в набор 02.05.18. Подписано в печать 02.06.19. Формат 70 x $100^{1}/_{16}$.
Бумага офсетная. Гарнитура «MM Mekteptik» Печать офсетная.
Объем 19,35 усл.печ.л. Тираж 25000 экз.

Артикул 810-006-002р-19