Н. С. Кольева, Е. В. Шевчук

ИНФОРМАТИКА

10

Учебник для 10 классов
естественно-математического направления
общеобразовательных школ

Утверждено Министерством образования и науки Республики Казахстан

Алматы «Мектеп» 2019

УДК 373.167.1
ББК 32.973я72
К62

Кольева Н. С., Шевчук Е. В.
К62 **Информатика.** Учебник для 10 кл. естеств.-матем. направления общеобразоват. шк. — Алматы: Мектеп, 2019. — 184 с.

ISBN 978—601—07—1102—0

К $\frac{4306020500—068}{404(05)—19}$ 64(1)—19

УДК 373.167.1
ББК 32.973я72

ISBN 978—601—07—1102—0

ВВЕДЕНИЕ

Современное общество живет в период, характеризующийся небывалым ростом объема информации. Применение современной компьютерной техники дает возможность переложить трудоемкие, рутинные операции на автоматические или автоматизированные устройства, которые могут работать со скоростью, превышающей скорость обработки информации человеком в миллионы раз.

Современный специалист должен быть готов к работе в новых условиях высокоинформатизированного общества, а значит, должен быть компетентным в области информатики и владеть практическими навыками работы с компьютерной техникой и современными информационными технологиями, знать основные принципы работы с информационными системами, уметь оценивать точность и полноту информации, влияющей на принятие управленческих решений.

И учебник вам в этом поможет.

В учебнике используются условные обозначения, которые помогут вам ориентироваться в параграфе.

Желаем вам успехов!

Авторы

Условные обозначения

— сведения, которые надо знать

— проверьте себя и убедитесь, что теперь знаете

— определение

— контрольные вопросы и задания

— практическая работа за компьютером:

Уровень А — учебно-познавательные задания;

Уровень В — частично-исследовательские задания;

Уровень С — задания, требующие самостоятельной творческой работы

РАЗДЕЛ 1

Компьютерные сети и информационная безопасность

Из данного раздела вы узнаете:

- принципы работы компьютерных сетей;
- основы информационной безопасности;
- методы защиты информации;
- методы идентификации личности;
- перевод чисел из одной системы счисления в другую;
- логические операции (дизъюнкция, конъюнкция, инверсия);
- логические элементы компьютера;
- принципы кодирования текстовой информации.

Вы научитесь:

- описывать назначение компонентов сети (узлы, маршрутизаторы, коммутаторы);
- объяснять назначение и представление IP-адреса;
- объяснять назначение системы доменных имен (DNS);
- объяснять значения терминов «информационная безопасность», «конфиденциальность», «целостность» и «доступность».

§1 Принципы работы компьютерных сетей. Компоненты сети

Вы научитесь:

- описывать назначение компонентов сети (узлы, маршрутизаторы, коммутаторы).

Ключевые понятия:

- компьютерная сеть
- коммутаторы (свитчи)
- маршрутизатор
- узел

Зачем нужны компьютерные сети? Как можно перенести информацию с одного компьютера на другой? Назовите несколько способов.

Раньше для того, чтобы передать информацию с одного компьютера на другой, использовали устройства внешней памяти — сначала перфоленты и перфокарты, потом дискеты, магнитные ленты и лазерные диски. Сейчас обмен данными чаще идет через компьютерные сети.

Компьютерная сеть — это группа компьютеров, объединенных линиями связи.

С помощью сети можно быстро передавать файлы с одного компьютера на другой. Все учителя работают с электронным журналом «Күнделік», в котором хранится информация о школе, в том числе электронные журналы классов.

Для создания локальной сети нужно провести все кабели, настроить аппаратуру и сетевые программы. От качества настройки и обслуживания сети зависят скорость обмена информацией и безопасность данных. Решением этих задач занимается специалист — системный администратор (или сетевой администратор).

В современных кабельных локальных сетях используется технология пакетной передачи данных, которая называется *Ethernet* (от лат. *aether* — «эфир»). Для связи компьютеров могут применяться электрические кабели или оптоволокно.

При наличии сетевой карты (сетевой адаптер, англ. *network interface card*) компьютер можно подключить к сети. В современных материнских платах и в ноутбуках обычно уже есть сетевая карта.

Для передачи данных на большие расстояния применяют оптоволоконные кабели, в которых данные передаются с помощью светового

луча. Свет идет внутри кабеля, отражаясь от стенок стеклянного или пластикового цилиндра-световода.

Коммутаторы (свитчи) используются для объединения компьютеров в единую сеть по схеме «звезда», которая чаще всего применяется на практике (рис. 1.1).

Рис. 1.1. Коммутаторы

Рис. 1.2. Разъем RJ-45

Коммутаторы передают пакет только тому узлу, которому он предназначен, а не дублируют на все выходы, в отличие от концентраторов. Компьютер соединяется с коммутатором отрезком кабеля с двумя разъемами RJ-45 (рис. 1.2), который называется «патч-корд» (от англ. *patching cord* — «соединительный шнур»).

Все компьютеры, объединенные в локальную сеть, получают доступ в Интернет через один канал связи. Для связи локальной сети с Интернетом необходим ***маршрутизатор*** или ***роутер*** (англ. *router*). Задача маршрутизатора — определить дальнейший маршрут движения пакета и направить его на нужный выход (порт). Для этого используются *таблицы маршрутизации*, в которых записано, куда направлять пакеты в зависимости от адреса назначения. Роль маршрутизатора в локальной сети может выполнять обычный компьютер с несколькими сетевыми картами.

Беспроводной маршрутизатор используется для связи компьютеров в беспроводной сети и для обеспечения доступа в Интернет (рис. 1.3).

Рис. 1.3. Беспроводной маршрутизатор

Контрольные вопросы

1. Что такое *локальная сеть*?
2. Какое оборудование необходимо для создания беспроводной сети?
3. Какое оборудование необходимо для кабельных сетей?
4. Что такое *коммутатор*?
5. Что такое *маршрутизатор*? Какую роль он выполняет в локальной сети?
6. Каковы преимущества и недостатки компьютерных сетей?

Практикум

Уровень А

Подготовьте сообщение на темы:
— «Серверные операционные системы»;
— «Что такое *клиповое мышление*».

Уровень В

Напишите эссе на тему «Хотели бы вы стать блогером? Почему?»

Уровень С

Создайте сообщение и по локальной сети в классе отправьте друг другу.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§2 Принципы работы компьютерных сетей. IP-адрес

Вы научитесь:

- объяснять назначение и представление IP-адреса.

Ключевые понятия:

- IP-адрес
- маска

Все компьютеры, подключенные к Интернету, находят друг друга в автоматическом режиме. Люди вообще не участвуют в процессе пересылки сообщений, и это возможно благодаря тому, что каждый компьютер (хост или узел) имеет свой адрес, называемый *IP-адресом*.

IP-адрес — это запись, которая однозначно определяет местоположение компьютера в Интернете. IP-адрес представляет собой запись четырех чисел в диапазоне от 0 до 255, отделенных друг от друга точками, например 192.168.0.13.

В этих числах закодированы номер сети и номер компьютера в сети. Для того чтобы выделить эти две части из IP-адреса, используют маски-шаблоны. Маска — это тоже четыре числа в диапазоне [0; 255]. Маска записывается по принципу «n единиц, потом — нули» в двоичном коде. Например, маска 255.255.255.0 в двоичном виде запишется так:

11111111.11111111.11111111.0000000

В ней сначала идут 24 единицы, а потом — нули. Это значит, что первые 24 бита адреса — номер сети (192.168.0.0), а оставшиеся 8 битов — номер компьютера (узла) в нашей сети (13).

Также можно использовать другую запись, которая означает то же самое («/ 24» говорит о том, что в маске 24 единицы).

В такой сети может быть 254 узла, а не 256, как можно было бы ожидать. Дело в том, что младший адрес (192.168.0.0) используется для обозначения всей сети, а старший (192.168.0.13) — для так называемой *широковещательной рассылки* (сообщение отправляется всем компьютерам данной сети).

На адрес узла отводится три бита (в маске три нуля), в такой сети доступно только $2^3 = 8$ адресов. Учитывая, что два из них специальные (номер сети и широковещательный адрес), в сеть может входить не более 6 узлов.

Компьютерам маска подсети нужна для определения границ подсети, для того, чтобы каждый мог определить, кто находится с ним в одной [под]сети, а кто — за ее пределами. Дело в том, что внутри одной сети

компьютеры обмениваются пакетами «напрямую», а когда нужно послать пакет в другую сеть — шлют их шлюзу по умолчанию. Чтобы определить границы подсети, компьютер делает побитовое умножение (логическое И) между IP-адресом и маской, получая на выходе адрес с обнуленными битами в позициях нулей маски.

Определим номер сети для адреса 215.17.125.177 и маски 255.255.255.240.

IP-адрес: 215.17.125.177 (11010111.00010001.01111101.10110001).
Маска: 255.255.255.240 (11111111.11111111.11111111.11110000).
Применив побитовое умножение, получим:
11010111.00010001.01111101.10110000.

Переведем двоичное представление в десятичное, получим адрес сети: 215.17.125.176.

Для определения номера компьютера нужно перевести байты IP-адреса и маски подсети в двоичную систему счисления. Номер компьютера в IP-адресе записан под нулями маски: в нашем случае 0001 в двоичном представлении. Переведем в десятичное число и получим, что номер компьютера 1.

IP-адрес присваивается не компьютеру, а интерфейсу — каналу передачи данных (сетевой карте, модему). Поэтому один компьютер может иметь несколько IP-адресов, (например, если на нем установлены две сетевые карты).

В связи с бурным развитием Интернета адресов разработана новая (шестая) версия протокола IP, которая обозначается IPv6. В ней на каждый адрес отводится 128 битов, а не 32, как сейчас.

В настоящее время существует более 4400 сетей, где применяется IPv6. Данный протокол поддерживается всеми современными операционными системами и производителями оборудования.

Полный переход на IPv6 происходит постепенно, так как требует больших денежных затрат и замены всех устаревших устройств.

Контрольные вопросы

1. Что представляет собой *IP-адрес*?
2. Сколько места в памяти занимает IP-адрес.
3. Что такое *маска для IP-адреса*? Как она строится?
4. Может ли компьютер иметь несколько IP-адресов? В каких случаях?
5. Почему становится необходимым переход на протокол IPv6?

Практикум

Уровень А

По заданным IP-адресу сети и маске определите адрес сети:

IP-адрес	Маска
а) 12.16.196.10	255.255.224.0;
б) 145.92.137.88	255.255.240.0;
в) 217.16.246.2	255.255.252.0;
г) 146.212.200.55	255.255.240.0;
д) 148.8.238.3	255.255.248.0.

Уровень В

По заданным IP-адресу сети и маске определите номер компьютера в сети:

IP-адрес	Маска
а) 162.198.0.157	255.255.255.224;
б) 156.128.0.227	255.255.255.248;
в) 192.168.156.235	255.255.255.240;
г) 10.18.134.220	255.255.255.192;
д) 122.191.12.220	255.255.255.128;
е) 156.132.15.138	255.255.252.0;
ж) 112.154.133.208	255.255.248.0

Уровень С

Для каждого приведенного адреса определите номер сети, номер узла, наибольшее возможное количество компьютеров в сети:

а) 192.168.104.109/30;
б) 172.16.12.12/29;
в) 193.25.5.136/28;
г) 10.10.40.15/27.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§3 Принципы работы компьютерных сетей. DNS

Вы научитесь:

- объяснять назначение системы доменных имен (DNS).

Ключевые понятия:

- компьютерная сеть
- IP-адрес
- DNS

Доменные имена DNS

IP-адреса удобны для идентификации компьютеров в Интернете, однако неприемлемы для пользовательской работы (не наглядны, плохо запоминаются, при вводе можно допустить ошибки). Поэтому вместо числовых IP-адресов используется буквенная система доменных имен, которая называется **DNS** (*Domain Name Sever* — доменное имя сервера). Согласно этой системе имя каждого web-сервера состоит из последовательности слов, разделенных точками, и легко запоминается.

Доменное имя однозначно определяет сервер в Интернете и строится по иерархическому принципу:

— на самом верхнем уровне (домен верхнего уровня) обычно находится название страны, например, kz (Казахстан), ru (Россия), uk (Великобритания). Но часто вместо названия страны ставится сокращение, отвечающее типу организации, которой принадлежит домен: com (коммерческий), gov (правительственный), mil (военный), edu (образовательный), net (сетевой), org (других организаций);

— слева от домена верхнего уровня через точку дописывается обозначение города, штата или организации. Однако эта часть имени может отсутствовать;

— левее от обозначения города (организации) через точку следует обозначение сервера, которое таким образом занимает крайнюю левую позицию в доменном имени.

В итоге доменное имя сервера (другими словами, домен) может иметь следующий вид:

beeline.kz — сервер оператора мобильной связи.

Соответствие между IP-адресами и доменными именами устанавливается с помощью баз данных, которые размещены на специальных DNS-серверах. *Серверы DNS* — это электронные роботы, которые выполняют повседневную работу, необходимую для функционирования системы доменных имен.

Когда вы вводите адрес сайта (доменное имя) в адресной строке браузера, сначала отправляется запрос на DNS-сервер, цель которого — определить IP-адрес сервера. Если это удалось, направляется запрос

на получение веб-страницы, причем драйвер протокола IP использует полученный IP-адрес, а не доменное имя.

Заметим, что одному доменному имени может соответствовать несколько IP-адресов. Такой прием применяется для распределения нагрузки на сайты с большим количеством посетителей. Таким образом, соответствие между доменными именами и IP-адресами можно описать как «многие ко многим»: с одним IP-адресом может быть связано несколько доменных имен и наоборот.

Контрольные вопросы

1. Опишите принцип построения доменных имен.
2. Что такое *сервер DNS*?

Практикум

Уровень А

Выясните примерное географическое месторасположение корневых серверов имен (для этого воспользуйтесь сервисом **whois**).

Уровень В

Настройте DNS сервер (в качестве forward-сервера используйте DNS-сервер 192.168.128.1 или 192.168.128.5).

Уровень С

Проверьте работоспособность настроенного сервиса. При помощи сниффера **wireshark** исследуйте механизм работы утилиты **nslookup**.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§4 Принципы работы компьютерных сетей. Частная виртуальная сеть

Вы научитесь:

- объяснять назначение частной виртуальной сети.

Ключевые понятия:

- компьютерная сеть
- IP-адрес
- частная виртуальная сеть

Виртуальная частная сеть (VPN) — дает возможность организациям и компаниям расширить сети за счет уже существующей общей сети, например, Интернет. VPN позволяет управлять сетевым потоком данных и предоставляет такие важные функции, как идентификация и защита данных.

Виртуальная частная сеть (VPN) — это технологии, предоставляющие возможность обеспечивать одно или же сразу несколько сетевых соединений поверх другой сети.

Данное соединение представляет собой зашифрованный туннель, который связывает напрямую компьютер пользователя и удаленный сервер, что дает возможность не только скрыть реальный IP, но также зашифровать свой трафик (рис. 4.1). Другими словами, есть возможность скачивать что угодно и откуда угодно, и об этом никто не узнает.

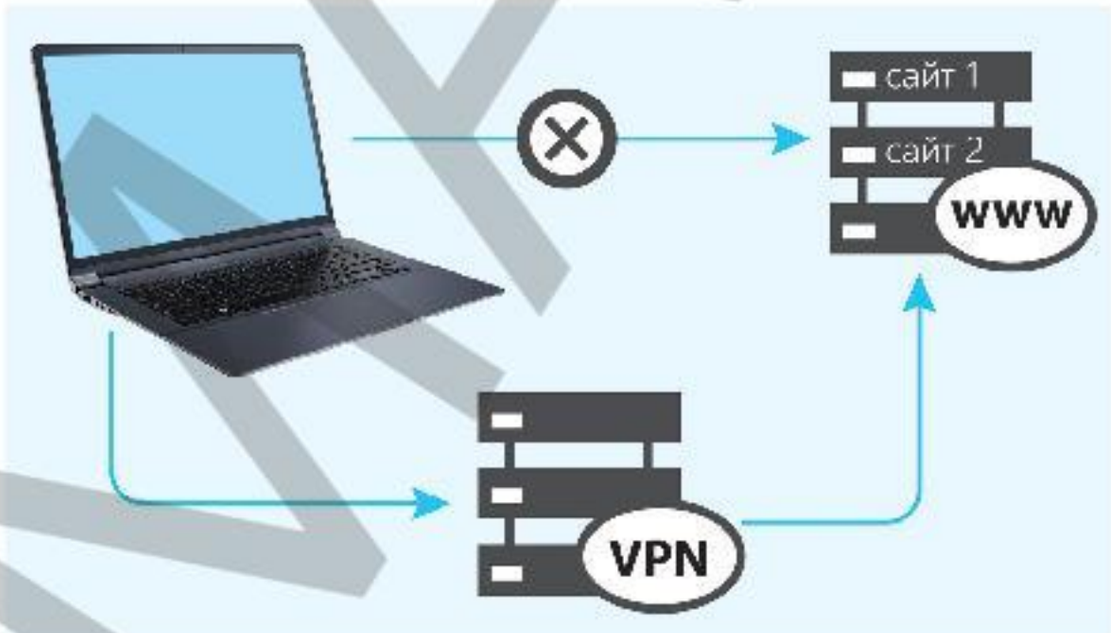


Рис. 4.1. Схема VPN

Но VPN могут себе позволить только очень крупные и богатые предприятия (нефтяные или газовые компании). Создание частной сети — привилегия тех, кто имеет производственные предпосылки для разработки собственной сетевой инфраструктуры. Частные сети были популярны в относительно далеком прошлом, когда общедоступные сети передачи данных (например, Интернет) были развиты очень слабо. Сегодня же Интернет вытеснил сети VPN, которые представляют собой компромисс между качеством услуг и их стоимостью.

Также к недостаткам относится сложность настройки. Использование VPN поддерживается оборудованием различных производителей, однако подключение к продукту, изготовленному не компанией NETGEAR усложняет процедуру настройки.

Среди преимуществ виртуальной частной сети можно выделить:

— возможность доступа к рабочему компьютеру из дома;

— почти полное отсутствие вероятности перехвата данных, передаваемых по VPN-туннелю.

Контрольные вопросы

1. Что такое *виртуальная частная сеть* (VPN)?
2. Перечислите достоинства и недостатки виртуальной частной сети.

Практикум

Уровень А

Проведите исследовательскую работу и заполните таблицу.

Название	Достоинства	Недостатки
Виртуальная частная сеть PPTP		
Виртуальная частная сеть OpenVPN		
Виртуальная частная сеть L2TP		

Уровень В

Используя Интернет и другие дополнительные источники, составьте классификацию виртуальных сетей. Результат в виде схемы, таблицы, блок-схемы представьте в текстовом или графическом редакторе.

Уровень С

Создайте частную виртуальную сеть (VPN) в облаке с помощью бесплатной учетной записи Azure.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§5 Информационная безопасность

Вы научитесь:

- объяснять значения терминов «информационная безопасность», «конфиденциальность», «целостность» и «доступность».

Ключевые понятия:

- информационная безопасность
- угрозы
- информационная сфера
- конфиденциальность
- целостность
- доступность

В Интернете можно найти информацию для реферата, послушать любимую мелодию, купить понравившуюся книгу, билеты на поезд или самолет, а также обсудить горячую тему на многочисленных форумах. Интернет может быть прекрасным и полезным средством для обучения, отдыха или общения с друзьями.

Что можно сказать о безопасности данных при объединении компьютеров в сеть? Увеличивается ли она или снижается? Почему?

Но сеть Интернет скрывает и угрозы. Перечислим часто встречающиеся угрозы:

— угроза заражения вредоносным программным обеспечением (ПО). Хакерами довольно часто используются скачанные из сети Интернет файлы, электронная почта, флэшки и другие для распространения троянских вирусов;

— контакты с незнакомыми людьми с помощью чатов или электронной почты (злоумышленники используют эту информацию, чтобы заставить детей выдать личную информацию);

— поиск развлечений (например, игр) в Интернете. Иногда при поиске нового игрового сайта можно попасть на карточный сервер и проиграть большую сумму денег.

Информационная безопасность — состояние сохранности информационных ресурсов и защищенности законных прав личности и общества в информационной сфере.

Информационная безопасность — это процесс обеспечения конфиденциальности, целостности и доступности информации.

Конфиденциальность: обеспечение доступа к информации только авторизованным пользователям.

Целостность: обеспечение достоверности и полноты информации и методов ее обработки.

Доступность: обеспечение доступа к информации авторизованных пользователей по мере необходимости.

В компьютерных сетях защищенность информации снижается в сравнении с отдельным компьютером, потому что:

— в сети работает много пользователей, их состав меняется;

— есть возможность незаконного подключения к сети;

— существуют уязвимости в сетевом программном обеспечении;

— возможны атаки взломщиков и вредоносных программ через сеть.

Вопросы, связанные с защитой информации, регулируют следующие законы: «Об информатизации», «О персональных данных и их защите», «Об электронном документе и электронной цифровой подписи» Республики Казахстан.

Технические средства защиты информации — это замки, системы сигнализации и видеонаблюдения, другие устройства, которые блокируют возможные каналы утечки информации или позволяют их обнаружить.

Программные средства обеспечивают доступ к данным по паролю, шифрование информации, удаление временных файлов, защиту от вредоносных программ и др.

Организационные средства включают:

— распределение помещений и прокладку линий связи таким образом, чтобы нарушителю было сложно до них добраться;

— политику безопасности организации.

Серверы в основном находятся в отдельном (охраняемом) помещении и доступны только администраторам сети. Важная информация периодически копируется на резервные носители для сохранения ее в случае сбоев.

Самое слабое звено любой защиты — это человек. В основном утечка информации связана с инсайдерами (от англ. *inside* — «внутри — не-

добросовестные сотрудники, работающие в фирме»). В большинстве случаев утечка секретной информации осуществляется через вспомогательный персонал (секретари, уборщики и др.). Поэтому ни один человек не должен иметь возможности причинить непоправимый вред (в одиночку уничтожить, украсть или изменить данные, вывести из строя оборудование).

Контрольные вопросы

1. Какие меры безопасности существуют в сети Интернет?
2. Что вы будете делать, если вашу учетную запись в социальной сети взломали?

Практикум

Уровень А

Ответьте на вопросы:

1. К аспектам информационной безопасности относятся:
   а) дискретность;
   б) целостность;
   в) конечность;
   г) актуальность;
   д) понятность.
2. Свойство данных быть доступными для санкционированного пользования в произвольный момент времени, когда в обращении к ним возникает необходимость:
   а) конфиденциальность;
   б) целостность;
   в) доступность;
   г) аутентичность;
   д) апеллируемость.
3. Что такое *несанкционированный доступ*:
   а) доступ субъекта к объекту в нарушение установленных в системе правил разграничения доступа;
   б) создание резервных копий в организации;
   в) правила и положения, выработанные в организации для обхода парольной защиты;
   г) вход в систему без согласования с руководителем организации;
   д) удаление ненужной информации?
4. Что такое *целостность информации*:
   а) свойство информации, заключающееся в возможности ее изменения любым субъектом;

б) свойство информации, заключающееся в возможности изменения только единственным пользователем;
в) свойство информации, заключающееся в ее существовании в виде единого набора файлов;
г) свойство информации, заключающееся в ее существовании в неискаженном виде (неизменном по отношению к некоторому фиксированному ее состоянию);
д) свойство информации, заключающееся в ее существовании в зашифрованном виде?

5. Потенциально возможное событие, действие, процесс или явление, которое может причинить ущерб чьим-нибудь данным, называется:

а) угрозой;
б) опасностью;
в) намерением;
г) предостережением;
д) взломом.

Уровень B

Заполните таблицу, перечислив все возможные опасные воздействия, угрожающие информационной безопасности.

ОПАСНЫЕ ВОЗДЕЙСТВИЯ	
Случайные воздействия	Преднамеренные воздействия

Уровень C

Задание. Разработайте концепцию информационной безопасности компании (поликлиника, офис страховой компании, офис адвоката, гостиница, интернет-магазин) по следующему примерному плану:

- Цель системы информационной безопасности.
- Задачи системы информационной безопасности.
- Объекты информационной безопасности.
- Вероятные нарушители.
- Основные виды угроз информационной безопасности.
- Мероприятия по обеспечению информационной безопасности.
- Предложите ПО для антивирусной защиты (проведя сравнительный анализ цен, возможностей и пр.).

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§6 Методы защиты информации

Вы научитесь:

- оценивать необходимость шифрования данных.

Ключевые понятия:

- информационная безопасность
- защита информации
- конфиденциальность информации
- целостность информации
- аутентичность информации

Информация играет существенную роль в развитии науки, техники и экономики как во всем мире, так и в Казахстане. Очевидно, что информация характеризует экономический потенциал страны и отдельного предприятия. В последнее время сообщения об атаках на информацию заполонили все средства массовой информации. Несанкционированный доступ к банковским счетам, атаки с помощью вирусных программ — это только несколько примеров информационных атак. Государство несет большие затраты для борьбы с подобными атаками. Конечно, средства, затрачиваемые на защиту информации, должны быть соизмеримы с ценностью защищаемой информации.

Информационная безопасность — это защищенность информации от любых действий, в результате которых владельцам или пользователям информации может быть нанесен недопустимый ущерб.

Ущерб может заключаться в потере информации, ее искажении, а также неправомерном доступе к ней. В первую очередь необходимо защищать государственную и военную тайны. В защите также нуждаются и коммерческая, юридическая, банковская и личная информация (паспортные данные, пароли на сайте и др.).

В соответствии с этим выделим несколько групп источников атак на информацию (рис. 6.1):

— вандализм, который, как правило, осуществляют хакеры-любители, в основном бескорыстно, с целью самоутверждения;

— криминальные атаки с целью хищения денежных средств частных лиц или коммерческих структур;

— коммерческий шпионаж;

— фальсификация информации;

— шпионаж (разведка) по заданию правительственных органов.

Рис. 6.1. Действия злоумышленников

Под **защитой** понимается общий термин, который описывает механизмы защиты информации. Хорошим механизмам свойственно:

— не допускать потерю и искажение информации;

— предоставлять пользователю средства защиты для его программ и данных.

Шифрование — это преобразование открытой информации в зашифрованную, недоступную для понимания посторонним.

В результате необходимо обеспечить:

— *конфиденциальность информации* — означает, что доступ к информации могут получить только легальные пользователи;

— *целостность* — означает, что информация существует в исходном виде и при ее передаче либо хранении не было несанкционированных изменений;

— *аутентичность* — означает, что источником информации является именно то лицо, которое заявлено как ее автор.

Направления защиты информации: *антивирусология* и *системная защита*.

Антивирусология — наука о способах борьбы с компьютерными вирусами и прочими самораспространяющимися программами, направленная на обеспечение и поддержание целостности хранимых данных.

Системная защита — комплекс аппаратных и программных средств, направленных на обеспечение целостности и недоступности данных в случаях отказа техники, ошибочных действий и прочих причин стихийного характера.

Слабым звеном в системе защиты является человек. Нередко это недобросовестные сотрудники. Известны случаи утечки закрытой информации через секретарей, уборщиц и другой вспомогательный персонал.

С древних времен практиковалась охрана документа (носителя информации) физическими лицами, передача его специальным курьером (человеком (дипломатом) или животным (голубиная почта)) и т.д. Но документ можно выкрасть, курьера можно перехватить, подкупить, в конце концов, расправиться с ним. В настоящий момент для реализации этого механизма защиты используются современные телекоммуникационные каналы связи. Однако следует заметить, что данный подход требует значительных капитальных вложений. При современном уровне развития науки и техники сделать такой канал связи между удаленными абонентами для многократной передачи больших объемов информации практически нереально.

Контрольные вопросы

1. Что такое *информационная безопасность*?
2. Почему информацию необходимо защищать?
3. Что входит в понятие «защита информации»?
4. Перечислите способы тайной передачи информации на расстоянии.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§7 Методы идентификации личности

Вы научитесь:

- объяснять использование мер безопасности для защиты данных пользователя.

Ключевые понятия:

- пароль
- ключ
- учетная запись
- идентификация
- аутентификация
- биометрическая аутентификация

Для безопасности данных пользователей используют пароли, учетные записи. Операционная система Windows предоставляет возможность для совместной работы за одним компьютером нескольких пользователей, это делается через учетные записи.

Учетная запись представляет набор настроек компьютера, которые определяют права доступа, доступные программные средства, внешний вид и др. Имя учетной записи (логин) используется для идентификации ее в операционной системе. Конечно, на домашнем компьютере в основном используется одна учетная запись. Но желательно, чтобы у каждого пользователя была своя учетная запись. Как вы думаете, зачем это необходимо?

Во-первых, это защитит компьютер от неопытных пользователей (младших братьев и сестер или бабушки, которая только осваивает компьютер). У опытного пользователя, например у вас — права администратора, у остальных пользователей — обычные. Теперь никто не сможет нанести вред компьютеру.

Во-вторых, это обеспечит конфиденциальность данных и действий в компьютере. Каждый пользователь имеет доступ только к своим собственным файлам и полкам, а для доступа к другим папкам необходимо ввести пароль.

Как вы думаете, в чем разница между понятиями авторизация, идентификация и аутентификация.

Сначала рассмотрим, что такое *идентификация*.

Идентификация — это процедура распознавания субъекта по его идентификатору. Другими словами, это определение имени, логина или номера и т. д.

Например, при входе в операционную систему или электронную почту выполняется идентификация.

Давайте на примере рассмотрим, что такое *идентификатор*. Если нам позвонили с неизвестного номера, мы спрашиваем «Кто это», то есть узнаем имя. Имя в данном случае и есть идентификатор, а ответ нашего собеседника — и есть идентификация.

В качестве идентификатора может выступать: номер телефона, номер паспорта, e-mail, номер страницы в социальной сети и др. (рис. 7.1).

Рис. 7.1. Пример идентификатора в социальной сети «В Контакте»

После идентификации производится аутентификация.

Аутентификация — это процедура проверки подлинности. Другими словами, пользователя проверяют с помощью пароля.

Для определения подлинности используют несколько факторов:

— пароль — то, что мы знаем (слово, PIN-код, графический ключ, код для сейфа и т. д.);

— устройство — то, что мы имеем (пластиковая карточка, ключ от замка, USB-ключ);

— биометрика — то, что является частью нас (отпечаток пальца, портрет, сетчатка глаза).

Таким образом, когда вы вставляете ключ в замок, вводите пароль или прикладываете палец к сенсору отпечатков пальцев, вы проходите аутентификацию (рис. 7.2).

После того, как проверили подлинность, можно предоставить и доступ, то есть выполнить авторизацию.

Рис. 7.2. Отпечаток пальца может быть использован в качестве пароля при аутентификации

Авторизация — это предоставление доступа к какому-либо ресурсу (например, к электронной почте после ввода пароля, разблокировка смартфона после сканирования отпечатка пальца и др.).

Можно ли сказать, что все три понятия взаимосвязаны (рис. 7.3)?

Рис. 7.3. Идентификация → аутентификация → авторизация

1. Сначала идентификация — определяют имя (логин или номер);
2. Затем аутентификация — проверяют пароль (ключ или отпечаток пальца);
3. И последнее, авторизация — предоставляют доступ.

Контрольные вопросы

1. Для чего создаются учетные записи?
2. Что такое идентификация, аутентификация.
3. Перечислите области применения биометрической аутентификации.

Практикум

Проектная работа

1. Вредоносные программы и закон.
2. Бесплатное антивирусное программное обеспечение.
3. Шифрование и закон.
4. Криптостойкость шифров.
5. Частотный анализ.

Критерии оценивания работы над проектом

Актуальность — обоснованность проекта в настоящее время, которая предполагает разрешение имеющихся по данной тематике противоречий.

Самостоятельность (планирование и выполнение всех этапов проектной деятельности самими).

Проблемность (умение формулировать проблему, проблемную ситуацию).

Содержательность (уровень информативности, смысловой емкости проекта).

Научность (использование конкретных научных терминов и возможность оперирования ими).

Работа с информацией (уровень работы с информацией, способа поиска новой информации).

Системность (умение выделять обобщенный способ действия и применять его при решении задач в работе).

Интегративность (связь различных областей знаний).

Коммуникативность — способность автора проекта четко, стилистически грамотно изложить результаты своей деятельности.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

РАЗДЕЛ 2

Представление данных

Из данного раздела вы узнаете:

- правила перевода чисел из одной системы счисления в другую;
- понятие логики;
- правила записи логических функций;
- что такое таблицы истинности.

Вы научитесь:

- переводить числа из одной системы счисления в другую;
- использовать логические операции (дизъюнкция, конъюнкция, инверсия);
- составлять таблицы истинности;
- применять логические функции при решении задач;
- сравнивать таблицы кодировки символов Unicode и ASII.

§8 Системы счисления. Перевод чисел из одной системы счисления в другую

Вы научитесь:

- переводить целые числа десятичной системы счисления в двоичную и обратно.

Ключевые понятия:

- развернутая и свернутая формы числа
- основание системы счисления
- двоичные, восьмеричные, десятичные, шестнадцатеричные, позиционные; непозиционные системы счисления

Мы привыкли считать десятками. Числа десять (10^1), сто (10^2), тысяча (10^3), миллион (10^6) мы воспринимаем как степени десяти, удобные для счета. Такой способ счета и такое восприятие круглых чисел сложились исторически и связаны с анатомическими особенностями человека. У человека 10 пальцев, а раз пальцы были первым естественным приспособлением для счета, то число 10 утвердилось в качестве основы для счета, поэтому мы называем систему счисления «десятичной».

В десятичной системе любое число выражается упорядоченной последовательностью десяти разных цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Системой счисления называется способ представления числа символами некоторого алфавита, которые называются **цифрами.**

Все системы счисления делятся на позиционные и непозиционные.

В позиционных системах счисления величина, обозначаемая цифрой, зависит от позиции цифры в числе.

Наиболее распространенными позиционными системами счисления являются десятичная, двоичная, восьмеричная и шестнадцатеричная (табл. 8.1).

Таблица 8.1

Алфавит основных систем счисления

Система счисления	Основание	Алфавит цифр
1	2	3
Позиционные		
Десятичная	10	0, 1, 2, 3, 4, 5 , 6, 7, 8, 9
Двоичная	2	0, 1

Продолжение

1	2	3
Восьмеричная	8	0, 1, 2, 3, 4, 5, 6, 7
Шестнадцатеричная	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
Непозиционные		
Римская		I(1), V(5), X(10), L(50), C(100), D(500), M(1000)

Рассмотрим правила записи чисел в десятичной системе более подробно. Записывая число, мы представляем его в виде суммы степеней числа 10 — основания системы счисления.

Например, запись числа 777 означает сумму семи сотен, семи десятков и семи единиц, что можно представить в *развернутой* форме записи числа:

$$777 = 7 \cdot 10^2 + 7 \cdot 10^1 + 7 \cdot 10^0,$$

т. е. одна и та же цифра в зависимости от позиции в записи числа обозначает разные величины. А привычная для нас запись числа 777 означает, что число находится в *свернутой* форме. Мы так привыкли к такой форме записи, что не замечаем, как в уме цифры числа раскладываются на различные степени числа 10.

Для записи десятичных дробей используются отрицательные значения степеней основания. Например, число 777,77 в развернутой форме записывается следующим образом:

$$777{,}77 = 7 \cdot 10^2 + 7 \cdot 10^1 + 7 \cdot 10^0 + 7 \cdot 10^{-1} + 7 \cdot 10^{-2}.$$

В двоичной системе счисления основание равно 2, а алфавит состоит из двух цифр (0 и 1).

Числа в двоичной системе в развернутой форме записываются в виде суммы степеней основания 2 с коэффициентами 0 или 1.

Вот пример многозначного двоичного числа 101101_2.

Двойка внизу (нижний индекс) указывает на основание системы счисления. Если число записывается в привычной нам десятичной системе счисления, то нижний индекс 10 обычно не пишется. Ведь есть же такое число 101101 в десятичной системе.

Развернутая форма записи данного двоичного числа выглядит так:

$$101101_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^3 + 0 \cdot 2^1 + 1 \cdot 2^0 = 45_{10}.$$

Итак, двоичное число представляет собой цепочку из нулей и единиц. При этом оно имеет достаточно большое число разрядов. Недостатком двоичной системы счисления является громоздкость записи чисел.

Например, число 227_{10} в двоичной системе выглядит так: 11100011_2. Удобнее пользоваться восьмеричной или шестнадцатеричной системой счисления.

ЭТО ИНТЕРЕСНО!

В древности широкое распространение имела и двенадцатеричная система, происхождение которой, вероятно, связано, как и десятичной системы, со счетом на пальцах: за единицу счета принимались фаланги (отдельные суставы) четырех пальцев одной руки, которые при счете перебирались большим пальцем той же руки. Остатки этой системы счисления сохранились и до наших дней и в устной речи, и в обычаях. Хорошо известно, например, название единицы второго разряда - числа 12 — «дюжина». Сохранился обычай считать многие предметы не десятками, а дюжинами, например, столовые приборы в сервизе или стулья в мебельном гарнитуре.

У ряда африканских племен и в Древнем Китае была употребительна пятеричная система счисления.

В Центральной Америке (у древних ацтеков и майя) и среди населявших Западную Европу древних кельтов была распространена двадцатеричная система. Все они также связаны со счетом на пальцах.

В Вавилоне, за две тысячи лет до н.э., люди пользовались шестидесятеричной системой (т.е. в ней использовалось шестьдесят цифр). Вавилонская шестидесятеричная система считается первой известной системой счисления, основанной на позиционном принципе. Система вавилонян сыграла большую роль в развитии математики и астрономии, ее следы сохранились до наших дней. Так, мы до сих пор делим час на 60 минут, а минуту на 60 секунд. Точно так же, следуя примеру вавилонян, окружность мы делим на 360 частей (градусов).

Перевод чисел в десятичную систему счисления

Как переводить целые числа десятичной системы счисления в двоичную и обратно?

Преобразовать числа, представленные в двоичной системе счисления, в систему десятичную — легко. Для этого достаточно записать число в развернутой форме и вычислить его значение.

Перевод числа из двоичной системы в десятичную. Возьмем любое двоичное число, например $11{,}01_2$. Запишем его в развернутой форме и произведем вычисления:

$$11{,}01_2 = 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 2 + 1 + 0 + 0{,}25 = 3{,}25_{10}.$$

Перевод числа из восьмеричной системы в десятичную. Возьмем любое восьмеричное число, например 17,48. Запишем его в развернутой форме и произведем вычисления:

$$17{,}4_8 = 1 \cdot 8^1 + 7 \cdot 8^0 + 4 \cdot 8^{-1} = 8 + 7 + 0{,}5 = 15{,}5_{10}.$$

Перевод числа из шестнадцатеричной системы в десятичную. Возьмем любое шестнадцатеричное число, например $51C_{16}$. Запишем его в развернутой форме и произведем вычисления:

$$51C_{16} = 5 \cdot 16^2 + 1 \cdot 16_1 + 12 \cdot 16^0 = 1280 + 16 + 12 = 1308_{10}.$$

Перевод чисел из десятичной системы в двоичную, восьмеричную и шестнадцатеричную.

Перевод чисел из десятичной системы в двоичную более сложен и может осуществляться различными способами. Мы рассмотрим один из алгоритмов перевода чисел.

Алгоритм перевода целого числа

1. Десятичное число делится с остатком на основание системы, а полученное частное снова делится с остатком. Так продолжается до тех пор, пока частное не станет меньше делителя.

2. Полученные остатки записываются в обратной последовательности.

Рассмотрим пример перевода десятичного числа 125 в двоичную, восьмеричную и шестнадцатеричную системы (рис. 8.1):

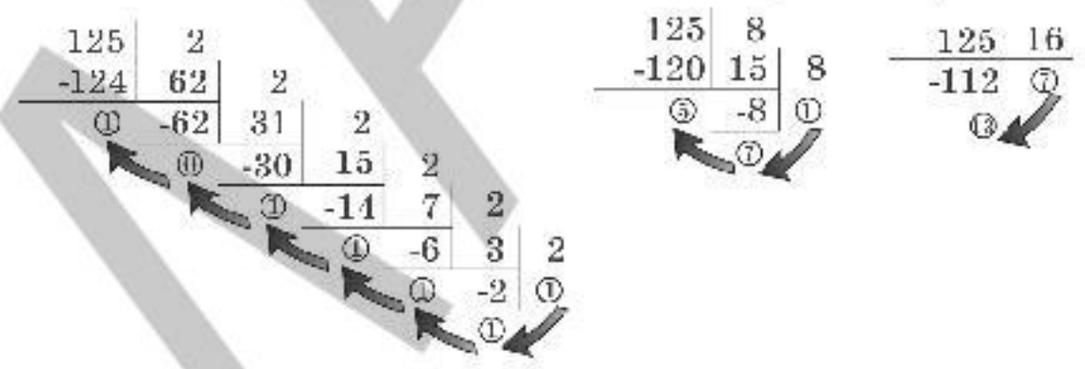


Рис. 8.1. Пример перевода целого числа

$125_{10} = 1111101_2$; $125_{10} = 175_8$; $125_{10} = 7D_{16}$. Отсюда следует: $125_{10} = 1111101_2 = 175_8 = 7D_{16}$. Не забывайте, что $13_{10} = D_{16}$.

Теперь рассмотрим алгоритм перевода десятичной дроби в другие системы счисления.

Алгоритм перевода десятичной дроби

1. Десятичная дробь последовательно умножается на основание системы, а получаемая дробная часть снова умножается на основание системы. Так продолжается до тех пор, пока не получится нулевая дробная часть или не будет достигнута требуемая точность вычислений.

2. Полученные целые части произведения записываются в прямой последовательности.

Рассмотрим пример перевода десятичной дроби 0,125 в двоичную, восьмеричную и шестнадцатеричную системы (рис. 5.2).

0	125		0	125		0	125	
	x 2			x 8			x 16	
(0)	250		(1)	000		(2)	000	
	x 2							
(0)	500							
	x 2							
(1)	000							

Рис. 8.2. Пример перевода десятичной дроби

Вертикальная черта отделяет целые части от дробных частей.

$0{,}125_{10} = 0{,}001_2$, $0{,}125_{10} = 0{,}1_8$, $0{,}125_{10} = 0{,}2_{16}$.

Отсюда: $0{,}125_{10} = 0{,}001_2 = 0{,}1_8 = 0{,}2_{16}$.

Перевод целых чисел из двоичной системы счисления в восьмеричную и шестнадцатеричную.

Чтобы перевести целое двоичное число в восьмеричное необходимо его разбить по три цифры, справа налево, а затем преобразовать каждую группу в восьмеричную цифру. Если в последней, левой группе окажется меньше трех цифр, то необходимо ее дополнить слева нулями.

Рассмотрим пример перевода двоичного числа 101111_2 в восьмеричное:

$$101111_2 \rightarrow \underbrace{1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0}_{5}\underbrace{1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0}_{7} \rightarrow 57_8.$$

Для быстрого перевода можно воспользоваться таблицей преобразования двоичных групп по три цифры в восьмеричные цифры (табл. 8.2).

Таблица 8.2

Таблица соответствия чисел в двоичной системе счисления восьмеричной системе счисления

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

Чтобы перевести целое двоичное число в шестнадцатеричное, необходимо его разбить по четыре цифры, справа налево, а затем преобразовать каждую группу в шестнадцатеричную цифру. Если в последней, левой группе окажется меньше трех цифр, то необходимо ее дополнить слева нулями. Рассмотрим пример перевода двоичного числа 101111_2 в шестнадцатеричное:

$$00101111_2 \rightarrow \underbrace{0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0}_{2}\ \underbrace{1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0}_{F} \rightarrow 2F_{16}.$$

Для быстрого перевода можно воспользоваться таблицей преобразования двоичных групп по четыре цифры в восьмеричные цифры (табл. 8.2).

Таблица 8.2

Таблица соответствия десятичной, двоичной и шестнадцатеричной систем счисления

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

Перевод целых чисел из восьмеричной и шестнадцатеричной систем счисления в двоичную

Чтобы перевести целые числа из восьмеричной и шестнадцатеричной систем счисления в двоичную, необходимо цифры числа преобразовать в группы двоичных цифр. Для перевода из восьмеричной системы в двоичную каждую цифру числа надо преобразовать в группу из трех двоичных цифр, а при преобразовании шестнадцатеричного числа — в группу из четырех цифр.

Рассмотрим пример перевода числа 127 из восьмеричной и шестнадцатеричной системы в двоичную:

$$127_8 = \underbrace{001}_{1}\underbrace{010}_{2}\underbrace{111}_{7} = 1010111_2;$$

$$127_{16} = \underbrace{0001}_{1}\underbrace{0010}_{2}\underbrace{0111}_{7} = 100100111_2.$$

Контрольные вопросы

1. Из каких знаков состоит алфавит двоичной системы счисления?
2. Почему в вычислительной технике взята за основу двоичная система счисления?
3. Как вы думаете, можно ли использовать систему счисления с основанием 1000000? В чем могут быть проблемы?

Практикум

Задание

Перевод десятичных и смешанных чисел в другие системы счисления

Уровень А

Переведите целые числа из десятичной системы счисления в двоичную: 12; 523; 76; 121.

Уровень В

Переведите из десятичной системы счисления следующие числа по схеме: $0{,}025 \to A_2$; $0{,}0625 \to A_{16}$; $0{,}0625 \to A_8$.

Уровень С

Переведите из десятичной системы счисления следующие числа по схеме: $1{,}25 \to A_2$; $2{,}25 \to A_8$; $3{,}75 \to A_{16}$.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§9 Практикум. Перевод чисел из одной системы счисления в другую

Вы научитесь:

- переводить целые числа десятичной системы счисления в двоичную и обратно.

Ключевые понятия:

- развернутая и свернутая формы
- основные системы счисления
- алгоритм

Практикум «Системы счисления»

Вариант 1

Почему и для чего перевели электронно-вычислительную технику с десятичной системы счисления в двоичную?
Какие цифры используются в восьмеричной системе счисления?
Какая цифра следует за цифрой 9_{16} в шестнадцатеричной системе счисления?
Переведите числа из одной системы счисления в другую:
Разложите по позициям: $2154_{10} = ...$
$190_{10} = ..._2$
$101110_2 = ..._{10}$
$217_{10} = ..._8$
$563_8 = ..._{10}$
$436546_8 = ..._2$
$11100101010010100_2 = ..._8$
$432_{10} = ..._{16}$
$F12B_{16} = ..._{10}$
$7FD56E_{16} = ..._2$
$10101010101000101011110_2 = ..._{16}$

Практикум «Системы счисления»

Вариант 2

Информация — это...
Шестнадцатеричная система счисления — это...
Какая цифра самая «старшая» в двоичной системе счисления?
Переведите числа из одной системы счисления в другую:
Разложите по позициям: $1256_{10} = ...$
$309_{10} = ..._2$
$10110_2 = ..._{10}$
$156_{10} = ..._8$
$363_8 = ..._{10}$
$1666_8 = ..._2$
$1010011011010100_2 = ..._8$
$436_{10} = ..._{16}$
$12CA_{16} = ..._{10}$
$E15C6E_{16} = ..._2$
$10010011001011010_2 = ..._{16}$

Контрольные вопросы

1. Что называется *основанием в позиционной системе счисления*?
2. Какое преимущество отличает шестнадцатеричную систему счисления от других?
3. Как удобнее всего произвести перевод числа из двоичной системы в восьмеричную (шестнадцатеричную) и обратно?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§10-11 Логические операции. Построение таблиц истинности

Вы научитесь:

- использовать логические операции (дизъюнкция, конъюнкция, инверсия);
- строить таблицы истинности для заданного логического выражения.

Ключевые понятия:

- логика
- дизъюнкция
- конъюнкция
- инверсия

С незапамятных времен существовала такая наука, как логика.

Логика — наука о высказываниях и их связях.

Первые учения о способах рассуждения возникли в странах Древнего Востока, но в основе лежат учения, созданные древнегреческим мыслителем Аристотелем. Немецкий ученый Г. В. Лейбниц первым (в 1666 г.) попытался перевести законы мышления из словесного царства неопределенностей в царство математики, в которой связь между высказываниями определяется в виде математических соотношений.

Г. В. Лейбниц

Через столетие английский математик Дж. Буль развил идею Лейбница о создании логического универсального языка, который подчиняется математическим законам. Он изобрел своеобразную алгебру — систему обозначений и правил, которую можно применить ко всем объектам, от чисел и букв до предложений. Именно Дж. Буль считается отцом алгебры логики.

До XX в. логика практически не продвигалась вперед и только с появлением теории ЭВМ начала свое бурное развитие.

Джордж Буль

Логика, как и любая наука, включает в себя несколько дисциплин. Например, такие дисциплины, как: формальная логика, математическая логика, вероятностная логика, диалектическая логика и т. д.

Формальная логика связана с анализом рассуждений, выраженных разговорным языком.

Вероятностная логика — основана на применении больших серий испытаний со случайными параметрами. Точность полученных результатов зависит от количества проведенных опытов. Например, вероятность выпадения «орла» или «решки» при бросании монеты равна S, и чем больше произведено бросаний монеты, тем эта вероятность точнее.

Математическая логика — изучает технику доказательств. Программисты, как и математики, требуют точности и строгости в определениях и доказательствах, чем они и отличаются от обычных людей. Областью математической логики является алгебра высказываний.

Алгебра высказываний

Алгебра логики изучает идеализированное высказывание, относительно которого можно утверждать, истинно высказывание или ложно. Алгебра логики не вникает в содержание высказываний. Таким образом, алгебра логики является двухзначной, т. е. любое из высказываний может дать характеристику, передающую одно из двух значений — «истина» или «ложь».

Простые высказывания в алгебре логики обозначаются заглавными латинскими буквами:

A = {Абай — великий поэт казахского народа}.

B = {Пушкин А.С. — великий математик}.

Истинному высказыванию ставится в соответствие «1», ложному — «0». Таким образом: $A = 1$, $B = 0$.

Составные высказывания на естественном языке образуются с помощью союзов «и», «или», «не», которые в алгебре высказываний заменяются на логические операции. Логические операции задаются таблицами истинности и могут быть графически проиллюстрированы с помощью диаграмм Эйлера-Венна.

Логическое умножение (конъюнкция)

Объединение двух или более высказываний в одно с помощью союза «и» называется операцией **логического умножения** или **конъюнкцией**.

Конъюнкция:

— в естественном языке соответствует союзу **и**;

— в алгебре высказываний обозначается $\wedge$ или &.

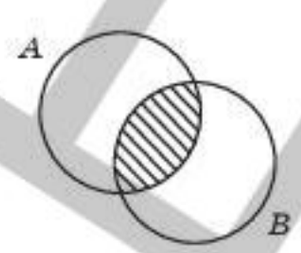
Конъюнкция — это логическая операция, ставящая в соответствие каждым двум простым высказываниям составное высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания истинны.

В алгебре множеств конъюнкции соответствует операция *пересечения множеств*, т. е. множеству, получившемуся в результате логического умножения множеств A и B, соответствует множество, состоящее из элементов, принадлежащих одновременно двум множествам.

Таблица истинности и графическая иллюстрация функции логического умножения представлены в таблице 10.1.

Таблица 10.1

Таблица истинности функции логического умножения

Таблица истинности			Диаграмма Эйлера-Венна
A	B	$A \wedge B$	
0	0	0	
0	1	0	
1	0	0	
1	1	1	

Рассмотрим пример. Даны составные высказывания:

1) «6 : 2 = 2 и 6 : 3 = 3»;
2) «6 : 2 = 2 и 6 : 3 = 2»;
3) «6 : 2 = 3 и 6 : 3 = 3»;
4) «6 : 2 = 3 и 6 : 3 = 2».

Из данных высказываний, образованных с помощью операции логического умножения (что соответствует союзу «и»), истинно только четвертое, так как в нем каждое простое высказывание истинно, а в первых трех составных высказываниях хотя бы одно из простых высказываний ложно.

Логическое сложение (дизъюнкция)

Объединение двух или более высказываний в одно с помощью союза «или» называется операцией **логического сложения или дизъюнкцией.**

Дизъюнкция:

— в естественном языке соответствует союзу **или**;

— в алгебре высказываний обозначается $\vee$.

Дизъюнкция — это логическая операция, которая каждым двум простым высказываниям ставит в соответствие составное высказывание, являющееся ложным тогда и только тогда, когда все исходные высказывания ложны, и являющееся истинным, когда хотя бы одно из исходных высказываний истинно.

В алгебре множеств дизъюнкции соответствует операция *объединения множеств*, т. е. множеству, получившемуся в результате логического сложения множеств A и B, соответствует множество, состоящее из элементов, принадлежащих либо множеству A, либо множеству B.

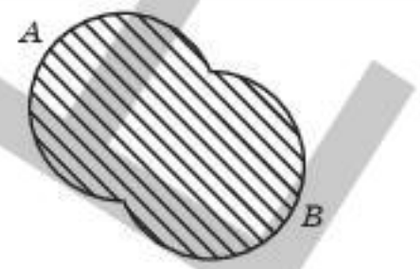
Таблица истинности и графическая иллюстрация функции логического сложения представлены в таблице 10.2.

Таблица 10.2

Таблица истинности функции логического сложения

Таблица истинности	Диаграмма Эйлера-Венна

A	B	$A \vee B$
0	0	0
0	1	0
1	0	0
1	1	1



Рассмотрим пример. Нам даны составные высказывания:

1) «6 : 2 = 2 или 6 : 3 = 3»;
2) «6 : 2 = 2 или 6 : 3 = 2»;
3) «6 : 2 = 3 или 6 : 3 = 3»;
4) «6 : 2 = 3 или 6 : 3 = 2».

В данных высказываниях, образованных с помощью операции логического сложения, ложно только первое, так как в нем оба простых высказывания ложны, а в последних трех составных высказываниях хотя бы одно из простых высказываний истинно.

Логическое отрицание (инверсия)

Присоединение частицы «не» к высказыванию называется **операцией логического отрицания** или **инверсией**.

Инверсия:

— в естественном языке соответствует словам **неверно, что...** и частице **не**;

— в алгебре высказываний обозначается $\overline{A}$.

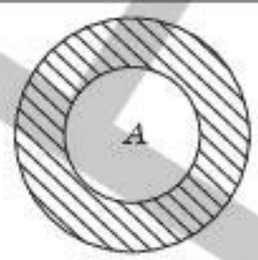
Отрицание — это логическая операция, которая каждому простому высказыванию ставит в соответствие составное высказывание, заключающееся в том, что исходное высказывание отрицается.

В алгебре множеств логическому отрицанию соответствует операция *дополнения до универсального множества*, т. е. множеству, получившемуся в результате отрицания множества A, соответствует множество $\overline{A}$, состоящее из элементов, не принадлежащих (не входящих) в множество A.

Таблица истинности и графическая иллюстрация функции логического отрицания представлены в таблице 10.3.

Таблица 10.3

Таблица истинности функции логического отрицания

Таблица истинности	Диаграмма Эйлера-Венна
A | $\overline{\overline{A}}$ 0 | 1 1 | 0	

В нашем примере пусть A = «Шесть разделить на два равно трем» — истинное высказывание, тогда его отрицание — высказывание $\overline{\overline{A}}$ = «*Шесть разделить на два не равно трем*», образованное с помощью логического отрицания, — ложно.

Таблица истинности — это таблица, в которой представлены значения логической функции при всех возможных значениях, входящих в нее логических переменных.

Порядок вычисления логических выражений:

1. Вычисляется выражение в скобках;
2. Вычисляется инверсия (отрицание);
3. Вычисляется конъюнкция (умножение);
4. Вычисляется дизъюнкция (сложение).

Рассмотрим пример $F = (K \vee C) \wedge \neg C$ и построим таблицу истинности для этого составного высказывания.

При построении таблиц истинности есть определенная последовательность действий:

— необходимо определить количество строк в таблице истинности. Количество строк = $2n$, где n — количество логических переменных;

— необходимо определить количество столбцов в таблице истинности. Количество столбцов = количеству логических переменных + количество логических операций;

— необходимо построить таблицу истинности с указанным количеством строк и столбцов, ввести названия столбцов таблицы в соот-

ветствии с последовательностью выполнения логических операций с учетом скобок и приоритетов (¬, &, V);

— заполните столбцы входных переменных наборами значений;

— проведите заполнение таблицы 10.4 истинности по столбцам, выполняя логические операции в соответствии с установленной последовательностью.

Таблица 10.4

Таблица истинности

K	C	$\neg C$	$K \vee C$	$(K \vee C) \wedge \neg C$
0	0	1	0	0
0	1	0	1	0
1	0	1	1	1
1	1	0	1	0

Контрольные вопросы

1. Что такое *конъюнкция*?
2. Что такое *дизъюнкция*?
3. Что понимается под *логическим умножением*?
4. Что понимается под *логическим сложением*?
5. Что понимается под *логическим отрицанием*?

Практикум

Уровень А

Из простых высказываний постройте сложные высказывания, используя логические связки «И», «ИЛИ», и определите их истинность.

Все ученики изучают информатику.
Все ученики изучают английский язык.
} Все ученики изучают информатику и английский язык.

Ербол старше Мадины. Салима старше Мадины.

Красный мяч больше зеленого. Красный мяч больше желтого.

Завтра пойдет снег. Завтра будет холодно.

Кайрат делает уроки. Кайрат смотрит футбол на DVD.

Айгуль обедает. Айгуль учит стихотворение.

Руслан друг Саши. Руслан друг Амана.

Уровень B

1. Запишите следующие суждения в виде логических выражений:
   а) Неверно, что $0 < X \leq 3$ и $Y < 5$;
   б) Каждое из чисел X, Y, Z равно 0.
2. Постройте таблицы истинности для следующих функций:
   а) $\neg(A \vee \neg B \,\&\, C)$;
   б) $A \,\&\, \neg\, (B \vee C)$.

Уровень C

1. Запишите математическое выражение по логическому выражению:
   а) $X > 0 \,\&\, X < 1 \vee X < 10 \,\&\, X < 5$;
   б) $X <> Y \,\&\, Y <> Z$.
2. Определите значение логического выражения $\neg X \,\&\, Y \vee X \,\&\, Z$, если логические переменные имеют следующие значения: X = ЛОЖЬ, Y = ИСТИНА, Z = ИСТИНА.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§12 Практикум. Логические операции. Построение таблиц истинности

Вы научитесь:

- использовать логические операции (дизъюнкция, конъюнкция, инверсия);
- строить таблицы истинности для заданного логического выражения.

Ключевые понятия:

- дизъюнкция
- конъюнкция
- инверсия

Вариант 1

1. Укажите, какое логическое выражение равносильно выражению $\neg(A \vee \neg B) \vee \neg C$:
   а) $A \wedge B \vee \neg C$;
   б) $A \wedge \neg B \wedge \neg C$;

в) $\neg A \wedge \neg B \vee \neg C$;

г) $A \wedge (\neg B \vee \neg C)$.

2. Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F:

X	Y	Z	F
0	1	1	0
1	0	0	1
0	0	1	1

Какое выражение соответствует F:

а) $X \wedge \neg Y \wedge Z$;

б) $X \wedge \neg Y \vee Z$;

в) $(X \wedge \neg Y) \vee \neg Z$;

г) $X \wedge \neg Y \vee \neg Z$?

3. Постройте таблицу истинности для выражения $A \wedge \neg B \wedge \neg C$.

4. Постройте выражения для логических функций, заданных таблицами истинности. Используйте разные методы и сравните их.

а)

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

б)

A	B	X
0	0	0
0	1	0
1	0	1
1	1	1

Вариант 2

1. Укажите, какое логическое выражение равносильно выражению $A \wedge (\overline{B} \vee C)$:

а) $A \wedge \overline{B} \wedge C$;

б) $A \wedge \overline{B} \vee C \wedge A$;

в) $A \wedge \overline{B} \vee C$;

г) $A \wedge B \vee A \wedge \overline{C}$).

2. Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F:

X	Y	Z	F
0	0	1	1
0	1	0	0
1	0	0	1

Какое выражение соответствует F:

а) $X \vee Y \vee Z$;

б) $X \wedge \overline{Y} \wedge Z$;

в) $X \vee \overline{Y} \vee Z$;

г) $\overline{X} \wedge Y \vee \overline{Z}$?

3. Постройте таблицу истинности для выражения $A \wedge \overline{B} \wedge C$.

4. Постройте выражения для логических функций, заданных таблицами истинности. Используйте разные методы и сравните их.

а)

A	B	X
0	0	0
0	1	1
1	0	0
1	1	0

б)

A	B	X
0	0	0
0	1	0
1	0	1
1	1	0

Контрольные вопросы

1. Что такое *таблица истинности*?
2. В каком порядке обычно записываются значения переменных в таблице истинности? Зачем это нужно?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§13 Логические элементы компьютера

Вы научитесь:

- объяснять назначение основных логических элементов: конъюнктор, дизъюнктор, инвертор.

Ключевые понятия:

- логический элемент компьютера
- триггер
- сумматор
- вентиль

Математический аппарат алгебры логики очень удобен для описания того, как функционируют аппаратные средства компьютера, поскольку основной системой счисления в компьютере является двоичная, в ко-

торой используются цифры 1 и 0, а значений логических переменных тоже два: 1 (true) и 0 (false).

Из этого следует два вывода:

— одни и те же устройства компьютера могут применяться для обработки и хранения как числовой информации, представленной в двоичной системе счисления, так и логических переменных;

— на этапе конструирования аппаратных средств алгебра логики позволяет значительно упростить логические функции, описывающие функционирование схем компьютера, и, следовательно, уменьшить число элементарных логических элементов, из которых состоят основные узлы компьютера.

Логический элемент компьютера — это часть электронной логической схемы, которая реализует элементарную логическую функцию.

Логическими элементами компьютеров являются электронные схемы **И, ИЛИ, НЕ, И–НЕ, ИЛИ–НЕ** и другие (их еще называют **вентилями**), а также триггер. С помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у вентилей бывает от двух до восьми входов и один или два выхода. Чтобы представить два логических состояния — «1» и «0» в вентилях, соответствующие им входные и выходные сигналы имеют один из двух установленных уровней напряжения. Например: 5 и 0 вольт (рис. 13.1).

Рис. 13.1. Логический элемент компьютера

Высокий уровень обычно соответствует значению «истина» (1), низкий — значению «ложь» (0).

Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая именно электронная схема в нем реализована. Это упрощает запись и понимание сложных логических схем.

Работу логических элементов описывают с помощью таблиц истинности. Рассмотрим структурные схемы логических элементов компьютера и их таблицы истинности (табл. 13.1).

Таблица 13.1

Структурные схемы логических элементов компьютера и их таблицы истинности

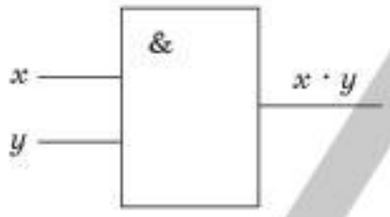
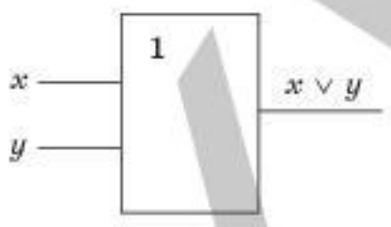
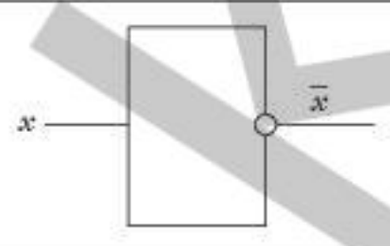
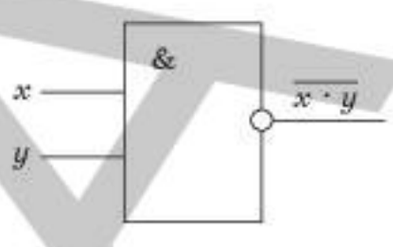
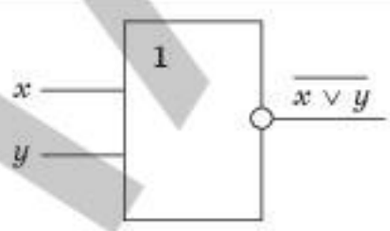
Условное обозначение	Структурная схема	Таблица истинности
И		x | y | $x \,\&\, y$ 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
ИЛИ		x | y | $x \vee y$ 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
НЕ		x | $\overline{y}$ 0 | 0 0 | 1
И–НЕ		x | y | $\overline{x \,\&\, y}$ 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
ИЛИ–НЕ		x | y | $\overline{x \vee y}$ 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 0

Схема И реализует конъюнкцию двух или более логических значений. На выходе схемы И будет 1 тогда и только тогда, когда на всех входах будут единицы. Когда хотя бы на одном входе будет 0, на выходе также будет 0.

Схема ИЛИ реализует дизъюнкцию двух или более логических значений. Когда хотя бы на одном входе схемы ИЛИ будет единица, на ее выходе также будет единица.

Схема НЕ (инвертор) реализует операцию отрицания. Связь между входом x этой схемы и выходом z можно записать соотношением $z = \bar{x}$, где $\bar{x}$ читается как «не x» или «инверсия x».

Схема И–НЕ состоит из элемента И и инвертора и осуществляет отрицание результата схемы И. Связь между выходом z и входами x и y схемы записывают следующим образом: $z = \overline{x \cdot y}$, где $\overline{x \cdot y}$ читается как «инверсия x и y».

Контрольные вопросы

1. Связано ли появление алгебры логики с разработкой персонального компьютера?
2. Назовите основные логические операции.
3. Приведите примеры предложений, которые не являются логическим высказыванием.
4. Покажите связь между алгеброй логики и двоичным кодированием информации.
5. Назовите приоритеты логических операций.

Практикум

Уровень А

Пусть x, y, z — логические величины, которые имеют следующие значения: x = истина, y = ложь, z = истина. Нарисуйте логические схемы для следующих логических выражений и вычислите их значения:

а) $\neg x \vee y$;

б) $x \vee y \,\&\, z$.

Уровень В

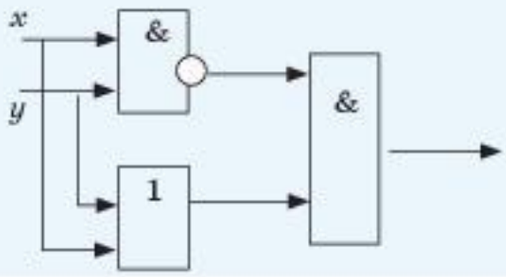
Постройте логические схемы по следующим формулам:

а) $x \,\&\, (\neg y \vee z)$;

б) $x \,\&\, y \vee \neg x \,\&\, z$.

Уровень С

Запишите логическое выражение по логической схеме и постройте таблицу истинности для этой функции:



Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§14 Логические основы компьютера

Вы научитесь:

- описывать функции устройства управления (УУ), арифметико-логического устройства (АЛУ) и регистров памяти как отдельных частей процессора.

Ключевые понятия:

- устройство управления (УУ)
- запоминающее устройство (ЗУ)
- арифметико-логическое устройство (АЛУ)

В настоящее время мы не можем представить свою жизнь без персонального компьютера. Компьютер мы можем встретить везде: в школе, в больнице, в банке, на любом предприятии, в аэропорту. В отличие от других изобретений человечества, история компьютеров не так велика. В прошлом столетии, в 40-е годы, было положено начало разработки вычислительной машины современной архитектуры и с современной логикой (принципы фон Неймана). С того времени в своем развитии компьютер проделал стремительный путь, с каким не сравнится ни одно другое изобретение человека, включая космическую технику. Благодаря развитию электроники, в частности массовому производству интегральных схем, компьютер превратился из огромного гиганта, который занимал в 50-е годы целые залы, в довольно компактное устройство, которое удобно размещается на письменном столе или даже его можно носить с собой.

С каждым новым поколением ЭВМ (а их шесть) увеличивались быстродействие и надежность работы при уменьшении стоимости и размеров, совершенствовались устройства ввода и вывода информации.

Обычно, описывая архитектуру ПК, особое внимание уделяют тем принципам ее организации, которые характерны для большинства машин, относящихся к описываемому семейству, а также оказывающим влияние на возможности программирования.

От архитектуры компьютера зависят возможности программирования на нем, поэтому при описании архитектуры ПК уделяют внимание описанию команд и памяти.

Рассмотрим архитектуру машины фон Неймана. Машина фон Неймана состоит из запоминающего устройства (памяти) — ЗУ, арифме-

тико-логического устройства — АЛУ, устройства управления — УУ, а также устройств ввода и вывода (рис. 14.1).

Рис. 14.1. Схема вычислительной машины фон Неймана

Программы и данные вводятся в память из устройства ввода через арифметико-логическое устройство. Все команды программы записываются в соседние ячейки памяти, а данные для обработки могут содержаться в произвольных ячейках. У любой программы последняя команда должна быть командой завершения работы.

Команда состоит из *указания*, какую операцию следует выполнить, и *адресов ячеек памяти*, где хранятся данные, над которыми следует выполнить указанную операцию, а также *адреса ячейки*, куда следует записать результат.

Арифметико-логическое устройство выполняет указанные командами операции над указанными данными. Из арифметико-логического устройства результаты выводятся в память или устройство вывода. Принципиальное различие между запоминающим устройством (ЗУ) и устройством вывода заключается в том, что в ЗУ данные хранятся в виде, удобном для обработки компьютером, а на устройство вывода (принтер, монитор и др.) поступают так, как удобно человеку.

Устройство управления (УУ) управляет всеми частями компьютера. От управляющего устройства на другие устройства поступают сигналы «что делать», а от других устройств УУ получает информацию об их состоянии.

Управляющее устройство содержит специальный регистр (ячейку), который называется «счетчик команд». После загрузки программы и данных в память в счетчик команд записывается адрес первой команды

программы. УУ считывает из памяти содержимое ячейки памяти, адрес которой находится в счетчике команд, и помещает его в специальное устройство — «Регистр команд». УУ определяет операцию команды, «отмечает» в памяти данные, адреса которых указаны в команде, и контролирует выполнение команды. Операцию выполняет АЛУ или аппаратные средства компьютера.

В результате выполнения любой команды счетчик команд изменяется на единицу и, следовательно, указывает на следующую команду программы.

В современных компьютерах функции УУ и АЛУ выполняет одно устройство, называемое *центральным процессором.*

Практикум

Уровень А

1. Создайте схему вычислительной машины фон Неймана в графическом редакторе Paint.
2. Подготовьте сообщение «Джон фон Нейман и его вклад в науку».

Уровень В

1. Создайте схему вычислительной машины фон Неймана в MS Word.
2. Подготовьте сообщение «Гарвардская архитектура».

Уровень С

1. Создайте схему вычислительной машины фон Неймана в любой программе.
2. Подготовьте сообщение, в котором приведены факты, подтверждающие, что Джон фон Нейман не был единоличным автором «фон-неймановской» архитектуры ЭВМ.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§15 Практикум. Решение логических задач

Вы научитесь:

- использовать логические операции (дизъюнкция, конъюнкция, инверсия);
- строить таблицы истинности для заданного логического выражения.

Ключевые понятия:

- дизъюнкция
- конъюнкция
- инверсия

Важно решать задачи, направленные на представление исходных данных задач и рассуждений в виде схем и таблиц, которые являются наглядным графическим представлением информации, ускоряют и облегчают процесс решения задач.

Предлагается такая последовательность решения задач с помощью схем: кратко записать условие, вопрос задачи и приступить к ее решению. Элементы условия задачи отображаются символьными переменными. Если по условию между двумя элементами есть соответствие, то они соединяются сплошной линией. Если же между элементами соответствия нет, то они соединяются пунктирной линией.

С помощью таблиц решаются задачи с четырьмя, пятью и более парами элементов, когда использование схем становится неудобным и ненаглядным из-за их чрезмерной громоздкости.

Задача 1. Школьные учителя.

В старших классах работают три учителя — Аубакиров, Галиев и Семенов. Каждый из них преподает по два предмета, так что в расписании у них всего шесть предметов — математика, физика, химия, история, литература и английский язык. Семенов — самый молодой из преподавателей. Учитель химии старше учителя истории. Все трое — учитель химии, учитель физики и Галиев — занимаются спортом. Когда между учителями литературы и английского языка возникает спор, то Семенов тоже принимает в нем участие. Галиев не преподает ни английский язык, ни математику.

Кто какие предметы преподает?

Решение.

Дано:
Учителя:
Аубакиров
Галиев
Семенов

Предметы:
математика
физика
химия
история
литература
английский язык

Надо:
Кто какие предметы преподает?

Рассуждения:

По условию задачи учитель химии, физики и Галиев занимаются спортом, значит, Галиев не учитель физики и химии. Ставим минусы в ячейки «Химия, Галиев» и «Физика, Галиев».

По условию задачи, когда между учителем литературы и английского языка возникает спор, Семенов тоже принимает в нем участие, значит, Семенов не учитель литературы и английского языка. Ставим минусы в ячейки «Литература, Семенов» и «Английский язык, Семенов».

По условию задачи Галиев не преподает ни английский язык, ни математику. Ставим минусы в ячейки «Английский язык, Галиев» и «Математика, Галиев».

По условию задачи Семенов — самый молодой из преподавателей, а учитель химии старше учителя истории, значит, Семенов не учитель химии. Ставим минус в ячейку «Химия, Семенов».

Итак, по условиям задачи получаем следующую таблицу 15.1.

Таблица 15.1

	Математика	Физика	Химия	История	Литература	Английский язык
Аубакиров						
Галиев	—	—	—			—
Семенов			—		—	—

Из условия задачи известно, что каждый учитель преподает два предмета. Из таблицы видно, что Галиев преподает историю и литературу, химию и английский язык преподает Аубакиров. Ставим плюсы в ячейки «История, Галиев», «Литература, Галиев», «Химия, Аубакиров», «Английский язык, Аубакиров» (табл. 15.2).

Таблица 15.2

	Математика	Физика	Химия	История	Литература	Английский язык
Аубакиров			+			+
Галиев	—	—	—	+	+	—
Семенов			—		—	—

Заполним свободные ячейки в столбцах (табл. 15.3) «История» и «Литература» и строке «Аубакиров» минусами, так как Галиев пре-

подает историю и литературу (по доказательству), значит, Аубакиров и Семенов не преподают историю и литературу, а так как Аубакиров преподает химию и английский язык (по доказательству), значит, он не преподает другие предметы.

Таблица 15.3

	Математика	Физика	Химия	История	Литература	Английский язык
Аубакиров	—	—	+	—	—	+
Галиев	—	—	—	+	+	—
Семенов			—	—	—	—

Из таблицы видно, что Семенов не преподает химию, историю, литературу, английский язык, т. е. Семенов — учитель физики и математики. Ставим плюсы в ячейки «Математика, Семенов» и «Физика, Семенов» (табл. 15.4).

Окончательно имеем:

Таблица 15.4

	Математика	Физика	Химия	История	Литература	Английский язык
Аубакиров	—	—	+	—	—	+
Галиев	—	—	—	+	+	—
Семенов	+	+	—	—	—	—

Ответ. Семенов преподает физику и математику, Галиев — историю и литературу, Аубакиров – химию и английский язык.

Задача 2. Школа собаководства.

Друзья усердно занимались в школе собаководства, тренируя своих питомцев — Джека, Лесси и Грифа, и вскоре приняли участие в соревнованиях.

Один из судей на вопрос друзей о результатах соревнований ответил:

«Джек занял второе место.

Лесси, по-моему, не второе.

Гриф не был первым».

После объявления результатов оказалось, что судья дважды ошибся, а один раз был прав.

Как распределились призовые места, если все участники заняли разные места?

Решение.

Дано:
Собаки:
Джек (Д.)
Лесси (Л.)
Гриф (Г.)

Места:
1, 2, 3

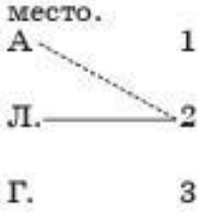
Надо:
Кто занял 1-е место, 2-е место, 3-е место?

Вариант 1.

Пусть истинно высказывание о том, что Джек занял 2-е место, тогда высказывание о том, что Лесси заняла не 2-е место, ложно, значит, Лесси заняла 2-е место. Получили противоречие с условием задачи, так как Джек и Лесси не могли поделить 2-е место. Значит, наше предположение о том, что первое высказывание истинно, неверно.

Графическая иллюстрация.

Пусть истинно высказывание о том, что Джек занял 2-е место.



противоречие.

Вариант 2.

Пусть истинно высказывание о том, что Лесси заняла не 2-е место, тогда высказывание о том, что Гриф не был первым, ложно, значит, Гриф занял 1-е место.

Так как (по доказательству) Гриф занял 1-е место, значит, Лесси и Джек не заняли 1-е место.

Так как Лесси заняла не 2-е место (по предположению) и не 1-е (по доказательству), значит, Лесси заняла 3-е место.

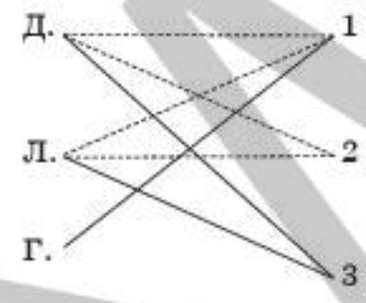
Так как высказывание о том, что Джек занял 2-е место, ложно (по предположению), значит, Джек занял не 2-е место, но он занял и не 1-е место (по доказательству), значит, Джек занял 3-е место.

Получили противоречие с условием задачи, так как Джек и Лесси не могли поделить 3-е место.

Значит, наше предположение о том, что второе высказывание истинно, неверно.

Графическая иллюстрация.

Пусть истинно высказывание о том, что Лесси заняла не 2-е место.



противоречие

Вариант 3.

Пусть истинно высказывание о том, что Гриф не был первым, тогда высказывание о том, что Джек занял 2-е место, ложно, значит, Джек не занял 2-е место.

Так как высказывание о том, что Лесси заняла не 2-е место, ложно (по предположению), значит, Лесси заняла 2-е место.

Так как Лесси заняла 2-е место (по доказательству), то Гриф не занял 2-е место.

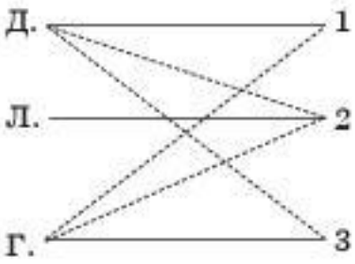
Так как Гриф занял не 1-е место (по предположению) и не 2-е место (по доказательству), значит, Гриф занял 3-е место.

Так как Гриф занял 3-е место (по доказательству), значит, Джек не занял 3-е место.

Так как Джек не занял 2-е и 3-е места (по доказательству), значит, Джек занял 1-е место.

Графическая иллюстрация.

Пусть истинно высказывание о том, что Гриф не был первым.



верное предположение.

Построение вентильной схемы по известному выражению

Из вентилей составляют более сложные схемы, которые позволяют выполнять арифметические операции и хранить информацию. Причем схему, выполняющую определенные функции, можно построить из различных по сочетанию и количеству вентилей.

Логические схемы необходимо строить из минимально возможного количества элементов, что, в свою очередь, обеспечивает большую скорость работы и увеличивает надежность устройства.

Правило построения логических схем:

1) Определить число логических переменных;

2) Определить количество базовых логических операций и их порядок;

3) Изобразить для каждой логической операции соответствующий ей вентиль и соединить вентили в порядке выполнения логических операций.

Задача 3. В двухэтажном доме лестница освещается одной лампой X. На первом этаже установлен один выключатель A, на втором этаже — выключатель B. Если включают A, то лампа загорается. При поднятии на второй этаж и включении B лампа гаснет. Если кто-то выходит и нажмет B, то лампа включается, при спуске на первый этаж и нажатии A лампа должна погаснуть.

Алгоритм решения:

- составить таблицу истинности;

- определить логическую функцию;
- построить логическую схему.

A	B	X
0	0	0
1	0	1
1	1	0
0	1	1
0	0	0

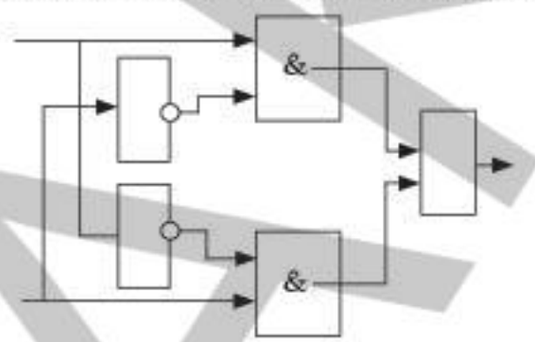
Чтобы создать логическую функцию по таблице истинности, надо записывать значения выходной переменной.

Для каждого набора переменных, на которых функция принимает значение логической 1, записываются конъюнкции, которые объединяются дизъюнкциями.

Переменные каждой строки, имеющие значение логического 0, в конъюнкцию входят с отрицанием, а переменные, имеющие значения логической 1 — без отрицания.

$(A \,\&\, \neg B) \vee (\neg A \,\&\, B)$.

Построим логическую схему по найденной функции:



Контрольные вопросы

1. Чем отличается формальная логика от «обычной», «бытовой»?
2. Что можно сделать для того, чтобы изменить естественный порядок действий?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§16 Принципы кодирования текстовой информации

Вы научитесь:

- сравнивать таблицы кодировки символов Unicode и ASII.

Ключевые понятия:

- кодирование
- код
- бит

Одну и ту же информацию можно представить и передать по-разному. Каждый народ имеет свой язык, состоящий из набора символов (букв): казахский, русский, английский и многие др. Например, древние люди, чтобы передать информацию на расстояние, кодировали сообщения звуками барабана (глухими и звонкими), также кодировали информацию в письменной форме. Эти предметы (глиняные таблички и папирус) были отнесены археологами к 5500 г. до н. э. А в старинном телеграфе информация кодировалась и передавалась с помощью азбуки Морзе — в виде последовательностей из точек и тире (рис. 16.1).

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

Рис. 16.1. Азбука Морзе

Представление информации с помощью какого-либо языка часто называют *кодированием*. Разные народы мира по-разному кодируют информацию, поэтому мы разговариваем на разных языках. Рисунки, буквы, знаки на различных носителях информации (бумага, камень, компьютер) — это *письменное кодирование*.

Кодирование — процесс представления информации в виде кода.

Код — набор символов (условных обозначений) для представления информации.

В повседневной жизни вы встречаетесь с различным кодированием информации: для водителей и пешеходов информация о правилах дорожного движения кодируется сигналами светофора или знаками дорожного движения, записывая диктант в тетрадь, вы кодируете в виде набора букв слова учителя.

Благодаря кодированию, предложенному немецким ученым Готфридом Вильгельмом Лейбницем в XVII в, компьютер может обрабатывать любую информацию: числовую, текстовую, графическую, звуковую, видео. Все эти виды информации после кодировки приводятся к одному виду — последовательности электрических импульсов (сигналов), в которой наличие импульса (сигнала) обозначается единицей, а его отсутствие — нулем (рис. 16.2).

Рис. 16.2. Кодирование информации

И сегодня используется такой способ представления информации, с помощью языка, содержащего два символа 0 и 1. Эти два символа 0 и 1 принято называть *битами* (от англ. *binary digit* — «двоичный знак»).

Бит — наименьшая единица измерения информации и обозначается двоичным числом. Более крупной единицей изменения объема информации принято считать 1 байт, который состоит из 8 бит. 1 байт = 8 бит.

ЭТО ИНТЕРЕСНО!

В 1946 г. математик из Принстонского университета Джон Таки впервые использовал в одной из своих статей термин «bit» (бит).

Кодирование текстовой информации

Для кодирования текста, вводимого в компьютер, используется самый простой способ кодировки: каждому символу ставится в соответствие двоичное число. То есть, когда вы нажимаете какую-либо клавишу на клавиатуре, сигнал посылается в компьютер в виде двоичного числа. Для кодирования каждого символа требуется количество информации, равное 8 битам, т. е. длина двоичного кода знака составляет восемь двоичных знаков. Например, при нажатии на клавишу «1» получается двоичный код 00110001, а при нажатии на клавишу «2» — код 00110010. Каждому знаку необходимо поставить в соответствие уникальный двоичный код в интервале от 00000000 до 11111111 (в десятичном коде от о до 255) (таблица 16.1).

Таблица 16.1

Кодирование знаков

Двоичный код	Десятичный код	КОИ-8	Windows	Mac	ISO
00000000	0				
...	...	...			
00001000	8	клавиша Backspase			
...	...	...			
00001101	13	клавиша Enter			
...	...	...			
00100000	32	клавиша Пробел			
...	...	...			
01011010	90	Z			
...	...				
10000000	128	-	Ъ	А	к
...	...	...	...	...	...
11001100	204	л	М	|	Ь
...	...	...	...	...	...
11111111	255	ь	я	неразде-ляемый пробел	п

Человек различает символы по их начертанию, а компьютер — по их двоичным кодам. То есть на экране компьютера — символы, а в памяти компьютера — двоичные коды.

В процессе вывода символа на экран компьютера производится обратное кодирование, т. е. преобразование двоичного кода символа в его изображение.

В зависимости от нажатой клавиши получается тот или иной двоичный код. Правила соответствия или правила кодировки записываются в таблицу, которая называется *кодовой*.

Кодовая таблица — это таблица, которая устанавливает соответствие между символами алфавита и двоичными числами. Эти числа называются *кодами символов* и отвечают внутреннему представлению символов в компьютере.

Различные кодировки знаков. Первые 33 кода в кодовой таблице (десятичные коды с 0 по 32) соответствуют операциям (перевод строки, удаление символа и т. д.).

Десятичные коды с 33 по 127 соответствуют знакам латинского алфавита, цифрам, знакам арифметических операций и знакам препинания.

Десятичные коды с 128 по 255 являются национальными, т. е. в различных национальных кодировках одному и тому же коду соответствуют разные знаки. Сегодня существует пять различных кодовых таблиц для русских букв (КОИ-8, MS-DOS, Windows, Mac, ISO).

В последние годы широкое распространение получил новый международный стандарт кодирования текстовых символов ***Unicode***.

Таким образом, в настоящее время имеется шесть различных кодировок, в которых один и тот же символ имеет различный код (табл. 16.2).

Таблица 16.2

Десятичные коды некоторых знаков в различных кодировках

Символ	КОИ-8	Windows	Mac	ISO	MS-DOS	Unicode
А	225	192	128	176	128	1040
В	247	194	130	178	130	1042
Э	252	221	157	205	157	1069
я	241	255	223	239	239	1103

Вы не должны переживать о перекодировках текстовых документов, так как это делают специальные программы-конверторы, встроенные в операционную систему и приложения.

Контрольные вопросы

1. Что такое *код* и в чем состоит кодирование информации?
2. Приведите несколько примеров из жизни, связанных с кодировкой?
3. По каким правилам она производится в каждом примере?

Практикум

Уровень А

В текстовом режиме экран монитора компьютера обычно разбивается на 25 строк по 80 символов в строке. Определите объем текстовой информации, занимающей весь экран монитора, в кодировке *Unicode*.

Уровень В

Определите, чему равен информационный объем следующего высказывания, закодированного с помощью 16-битной кодировки *Unicode*.

Уровень С

При кодировке сообщения на русском языке из 16-битного кода *Unicode* в 8-битную кодировку *KOI8-R* оно уменьшилось на 480 битов. Какова длина сообщения в символах?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

Проверь себя!

1. В комнате, закрывающейся на замок, компания хранит архивные копии базы данных и другой серверной информации. Замок на двери — это:
   а) административное средство защиты информации;
   б) техническое средство защиты информации;
   в) не является средством защиты информации;
   г) самое распространенное средство защиты информации;
   д) незапатентованное средство защиты информации.

2. Что позволяет сделать функция «Защита документа» в программе Word:
   а) защитить документ от просмотра и редактирования без пароля, но не защищает от вируса;
   б) ограничить возможность изменения документа без пароля;
   в) защитить документ от вирусов;
   г) защитить документ от копирования;
   д) защитить документ от просмотра?
3. Предоставление доступа пользователю, программе или процессу — это:
   а) авторизация;
   б) идентификация;
   в) аутентификация;
   г) модификация;
   д) фальсификация.
4. Процедура установления подлинности пользователя, программы, устройства или данных (информации, получаемого сообщения, ключа) — это:
   а) модификация;
   б) фальсификация;
   в) аутентификация;
   г) идентификация.
5. Переведите числа из одной системы счисления в другую: $24_{10} = ?_2$:
   а) 11100_2;
   б) 11000_2;
   в) 10100_2;
   г) 11101_2.
6. Переведите числа из одной системы счисления в другую: $110111_2 = ?_{10}$:
   а) 23_{10};
   б) 47_{10};
   в) 55_{10}
   г) 82_{10}
7. Как представляется число в двоичной системе счисления:
   а) как сумма степени двойки;
   б) как сумма степени двойки и записываются степени;
   в) как сумма степени двойки и записываются коэффициенты такого представления;
   г) как сумма двоек?
8. Сколько цифр в двоичной системе счисления:
   а) 5;
   б) 7;

в) 2;
г) 10?

9. Объясните, почему следующие предложения не являются высказываниями:
а) какого цвета этот дом;
б) число X не превосходит единицы;
в) $4X + 3$;
г) посмотрите в окно;
д) пейте томатный сок;
е) вы были в театре;
ж) сумма числа 5 и X равна 10?

10. Какие из следующих предложений являются истинными, а какие ложными высказываниями:
а) город Париж — столица Франции;
б) число 2 является делителем числа 7;
в) $3 + 5 = 2 * 4$;
г) $2 + 6 > 10$;
д) сканер — это устройство, которое может напечатать на бумаге то, что изображено на экране компьютера;
е) II + VI > VIII;
ж) сумма чисел 2 и 6 больше числа 8;
з) мышка — устройство ввода информации?

11. Приведите по два примера истинных и ложных высказываний из:
а) биологии;
б) географии;
в) информатики;
г) истории;
д) литературы;
е) математики;
ж) казахского языка.

12. Постройте отрицания следующих высказываний:
а) число 1 есть составное число;
б) натуральные числа, оканчивающиеся цифрой 0, являются простыми числами;
в) неверно, что число 3 не является делителем числа 198;
г) неверно, что любое число, оканчивающееся цифрой 4, делится на 4;
д) некоторые млекопитающие не живут на суше.

РАЗДЕЛ 3

Алгоритмизация и программирование

Из данного раздела вы узнаете:

- функции и процедуры;
- как работать со строками;
- виды сортировок: метод пузырька, линейный поиск, двоичный поиск.

Вы научитесь:

- 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 поиска на графах для решения практических задач.

§17-18 Пользовательские функции и процедуры

Вы научитесь:

- писать код на языке программирования, используя функции и процедуры.

Ключевые понятия:

- процедура
- функция
- имя функции
- тело функции
- виды функции

С функциями вы уже знакомы, потому что применяли стандартные функции языка программирования (например, input, int, randint).

Нужны ли функции в программировании?

Функции в программировании представляют собой изолированный блок кода, к которому можно обращаться несколько раз в течение выполнения программы.

Зачем нужны такие блоки инструкций?

Прежде всего, чтобы сократить объем исходного кода, намного удобнее вынести повторяющиеся выражения в отдельный блок, и затем, при необходимости, к нему обращаться.

Функция — это подпрограмма, которая может манипулировать данными и возвращать некоторое значение. Результатом функции может быть число, строка или другой объект.

Рассмотрим пример функции вычисления суммы цифр числа. В данном случае используется следующий алгоритм (рис. 17.1):

```
sum=0
while n!=0:
    sum+=n%10
    n=n//10
```

Рис. 17.1. Пример функции вычисления длины цифр числа

Для получения последней цифры числа (которая добавляется к сумме) берем остаток от деления числа на 10. Затем последнюю цифру отсекаем и переходим к следующей цифре. Цикл продолжается до тех пор, пока значение n не становится равно нулю.

Как указать программе, чему равно значение функции?

Для этого будем использовать оператор return, после которого записывают значение-результат:

return в переводе с английского означает «вернуть».

```
def sumDigits( n ):
    sum = 0
    while n!= 0:
    sum += n % 10
    n = n // 10
    return sum
# основная программа
print ( sumDigits(12345) )
```

Рис. 17.2. Вид программного кода

Поясним представленный программный код (рис. 17.2). `def` — это инструкция (команда) языка программирования Python, которая позволяет создавать функцию. `sumDigits` — это имя функции, которое (так же, как и имена переменных) может быть почти любым, но желательно смысловым. Двоеточие (:) обозначает окончание заголовка функции. Далее после заголовка следует тело функции. В конце тела функции, как вы обратили внимание, присутствует инструкция `return` (которая может и отсутствовать). Данная инструкция и передает то или иное значение в основную ветку программы. Если бы отсутствовала инструкция `return`, то в основную программу ничего бы не возвращалось и переменной `n` ничего не присваивалось.

Выражения тела функции выполняются лишь тогда, когда она вызывается в основной ветке программы. Так, например, если функция присутствует в исходном коде, но нигде не вызывается в нем, то содержащиеся в ней инструкции не будут выполнены ни разу.

Далее рассмотрим, что такое *процедура*.

Представим такую ситуацию, что в нескольких местах программы требуется выводить на экран сообщение об ошибке: «Ошибка программы». Это можно сделать, например, так:

```
cout << «Ошибка программы»;
```

Этот оператор вывода можно вставить везде, где нужно вывести сообщение об ошибке. Но есть определенные сложности. Во-первых, строка-сообщение хранится в памяти много раз. Во-вторых, если надо будет поменять текст сообщения, нужно будет искать эти операторы вывода по всей программе. Для этого и предназначены процедуры — вспомогательные алгоритмы, которые выполняют некоторые действия.

Рассмотрим фрагмент программы с использованием процедуры (рис. 17.3):

```
def Error():
    print ( "Ошибка программы" )
n = int ( input() )
if n < 0:
Error()
```

Рис. 17.3. Фрагмент программы

Процедура (выделена полужирным шрифтом) начинается с ключевого слова `def`. `Error` — имя процедуры, затем ставим пустые скобки (но они могут быть и не пустыми) и двоеточие.

Процедура должна быть определена к моменту ее вызова, т. е. должна быть выполнена инструкция `def`, которая создает объект-процедуру в памяти. Если процедура вызывается из основной программы, то нужно поместить ее определение раньше точки вызова.

В заголовке процедуры и при ее вызове после имени процедуры нужно ставить круглые скобки (в нашем примере они пустые).

Таким образом, процедуры сокращают код программы, если какие-то операции выполняются несколько раз.

Функция и процедура — это вспомогательные алгоритмы, которые принимают аргументы. Но, в отличие от процедуры, функция всегда возвращает значение-результат. Результатом может быть число, символ или объект другого типа.

Контрольные вопросы

1. Что такое *функция*? Чем она отличается от процедуры?
2. Как оформляются функции в тексте программы?
3. Как по коду программы определить, какое значение возвращает функция?
4. В чем смысл использования процедур?
5. Достаточно ли включить процедуру в текст программы, чтобы она «сработала»?

Практикум

Уровень А

1. Напишите программу, в которой бы в теле функции производились вычисления со значениями трех переменных.

2. Напишите программу с использованием функции, которая вычисляет количество цифр числа.

Уровень В

Напишите программу, которая будет принимать у пользователя число и выводить среднее арифметическое всех введенных за сеанс чисел. Программа должна запрашивать у пользователя числа до тех пор, пока пользователь не введет 0.

Уровень С

1. На соревнованиях выступление спортсмена оценивают пять экспертов, каждый из них выставляет оценку в баллах (целое число). Для получения итоговой оценки лучшая и худшая из оценок экспертов отбрасываются, а для оставшихся трех находится среднее арифметическое. Напишите функцию, которая вводит пять оценок экспертов и возвращает итоговую оценку выступления спортсмена.

2. Придумайте программу, в которой бы присутствовали две функции, основная ветка программы содержала бы не менее 10 строк, и каждая функция вызывалась как минимум один раз.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§19 Пользовательские функции и процедуры. Примеры

Вы научитесь:

- писать код на языке программирования, используя функции и процедуры.

Ключевые понятия:

- процедура
- функции
- имя функции
- тело функции
- виды функции

Есть несколько причин, на основании которых программу стоит разбивать на функции.

• Создание новой функции предоставляет вам возможность присвоить имя группе инструкций. Это позволит упростить чтение, понимание и отладку программы.

• Функции позволяют сократить код программы, благодаря ликвидации повторяющихся участков кода. Позже вы сможете вносить коррективы только в одном месте.

• Разбиение длинной программы на функции позволяет одновременно отлаживать отдельные части, а затем собрать их в единое целое.

• Хорошо спроектированная функция может использоваться во множестве программ. Однажды написав и отладив функцию, вы можете использовать ее множество раз.

В *Python* содержится математический *модуль* (*module*), который предоставляет большинство популярных математических функций. Перед тем, как его использовать, нам необходимо его импортировать:

```
>>> import math.
```

Эта *инструкция* создает модульный *объект* (*module object*), который называется *math*. Если вы выведете на экран модульный *объект*, то получите некоторую информацию о нем:

```
>>> print math
<module 'math' (built-in)>.
```

Модульный *объект* содержит функции и переменные, определенные в объекте. Для получения доступа к одной из этих функций вам необходимо задать *имя модуля* и *имя функции*, разделенные точкой (*period*). Этот формат называется *точечной нотацией* (*dot notation*).

```
>>> ratio = signal_power / noise_power
>>> decibels = 10 * math.log10(ratio)
>>> radians = 0.7
>>> height = math.sin(radians).
```

В первом примере вычисляется логарифм *по* основанию 10 отношения «сигнал-шум». *Модуль math* также предоставляет функцию log, которая вычисляет логарифмы *по* основанию e.

Во втором примере вычисляется синус radians. *Имя переменной* подсказывает, что sin и другие тригонометрические функции (cos, tg и т. д.) принимают аргументы в радианах. Для перевода градусов в радианы разделим на 360 и умножим на 2*pi:

```
>>> degrees = 45
>>> radians = degrees / 360.0 * 2 * math.pi
>>> math.sin(radians)
0.707106781187
```

Выражение math.pi получает *значение* переменной pi из модуля math. Значением этой переменной является приближенное с точностью до 15 знаков.

Добавление новых функций

Ранее мы использовали встроенные в *Python* функции. Теперь мы рассмотрим, как добавлять новые функции. *Определение функции* (*function* definition) задает *имя новой функции* и последовательность инструкций, которые выполняются, когда *функция* вызывается. Как только мы определили функцию, мы можем многократно ее использовать. Например,

```
def print_lyrics():
print "I'm a lumberjack, and I'm okay."
print 'I sleep all night and I work all day.'
```

def — это ключевое *слово*, которое показывает, что далее следует *определение* функции. *Имя функции* — print_lyrics. Правила наименования функций такие же, как для переменных: возможны буквы, числа и некоторые знаки пунктуации, но в начале не может быть число. Вы не можете использовать ключевые (зарезервированные) слова в качестве имен функций. Имена функций и переменных не должны совпадать.

Пустые скобки после имени указывают на то, что *функция* не принимает аргументов. Позже мы рассмотрим функции, которые принимают входные аргументы.

Первая строка определения функции называется *заголовком* (*header*), оставшаяся часть — *телом* (*body*) *функции*. Заголовок заканчивается двоеточием, *тело функции* имеет *отступ*. По договоренности, *отступ* всегда является четырьмя пробелами. *Тело функции* может содержать любое количество инструкций.

Строки, которые мы выводим на экран, заключены в двойные кавычки. Одиночные и двойные кавычки взаимозаменяемы, большинство людей используют одиночные кавычки, за исключением тех случаев, когда одиночные кавычки встречаются внутри строки.

Если вы будете набирать функцию в интерактивном режиме, то *Python* для тела функции сделает отступы, а в конце необходимо будет ввести пустую строку.

Определение функции создает переменную с таким же именем.

```
>>> print print_lyrics
<function print_lyrics at 0xb7e99e9c>
>>> print type(print_lyrics)
<type 'function'>.
```

Значением print_lyrics является функциональный *объект* (*function object*), который имеет тип *'function'*.

Синтаксис вызова новой функции похож на вызов встроенной функции:

```
>>> print_lyrics()
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
```

Однажды определив функцию, вы можете использовать ее внутри других функций. Для примера повторим строку-припев, воспользовавшись новой функцией:

```
def repeat_lyrics():
        print_lyrics()
        print_lyrics().
```

Затем вызовем repeat_lyrics:

```
>>> repeat_lyrics()
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
```

Если собрать вместе весь описанный выше код, то получим следующее:

```
def print_lyrics():
        print "I'm a lumberjack, and I'm okay."
        print 'I sleep all night and I work all day.'
def repeat_lyrics():
        print_lyrics()
        print_lyrics()
repeat_lyrics().
```

Эта *программа* содержит два определения функций: print_lyrics и repeat_lyrics. Определения функций получают управление подобно другим инструкциям, результатом является создание функционального объекта. Инструкции внутри функции не получат управления, пока *функция* не будет вызвана.

Контрольные вопросы

1. Что такое *функция*? Чем она отличается от процедуры?
2. Как по тексту программы определить, какое значение возвращает функция? Приведите пример.
3. Достаточно ли включить процедуру в текст программы, чтобы она «сработала»?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§20 Работа со строками

Вы научитесь:

- использовать процедуры и функции для обработки строк.

Ключевые понятия:

- символьная строка
- длина строки
- конкатенация
- получение среза
- реверс строки

Что такое *символьная строка*?

Вы уже знаете, что символьная строка — это последовательность символов. В компьютерах используется 256 различных символов, каждый символ имеет свой код (от 0 до 255) по специальной таблице. Строка, как и другие переменные, записывается в память, причем компьютеру все равно, какие данные записаны — для него это набор байтов.

Как же определить, где заканчивается строка? Есть два решения:

1) хранить длину строки в отдельной ячейке;

2) выбрать один особый символ, который будет обозначать конец строки, причем в середине строки этот символ не может встречаться.

Символьная строка — это последовательность символов, расположенных в памяти рядом (в соседних ячейках). Основной тип данных для работы с символьными величинами — это символьные строки (тип string).

Для того чтобы записать в строку значение, используют оператор присваивания:

```
S = "Кайрат играет в футбол";
```

Строка заключается в кавычки или одиночные апострофы. Если строка ограничена кавычками, внутри нее могут быть апострофы, и наоборот.

Для ввода строки с клавиатуры применяют функцию `input`:

```
S = input ("Введите имя: ")
```

Для определения длины строки будем использовать функцию `len`.

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

В данном примере в переменную `n` записывается длина строки `s`.

```
prin (s[5])
```

В следующем примере выделен отдельный символ строки. На экран выводится символ строки `s` с индексом 5.

Рассмотрим программу, которая вводит строку с клавиатуры, заменяет в ней все буквы «а» на буквы «б» и выводит полученную строку на экран (рис. 20.1).

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

Рис. 20.1. Фрагмент программы

Дадим пояснение. В цикле `for c in s` перебираются все символы, которые входят в строку `s`. Каждый их них на очередном шаге записывается в переменную `c`. Затем мы проверяем значение этой переменной: если оно совпадает с буквой «а», то заменяем его на букву «б» и прицепляем в конец новой строки `s1` с помощью оператора сложения.

Какой недостаток в программе вы заметили?

«Многократное» сложение строк в данном примере будет работать очень медленно. Поэтому в задачах, где необходимо заменить символы, лучше использовать стандартную функцию `replace`.

Операции со строками

Библиотека обработки строк обеспечивает много полезных функций для работы со строковыми данными, сравнения строк, поиска в строках символов, разметки строк (разделения строк на логические куски) и определения длины строк.

Рассмотрим некоторые типовые функции работы со строками. Операции со строками сведены в таблицу 20.1.

Таблица 20.1

Операции со строками

Операции со строками	Пример	Результат
1	2	3

'+' используется для объединения (сцепления) строк. Конкатенация (другое название)	`s1 = "Hello"`
	`s2 = "Марк"`
	`s =s1+","+s2+"!"`
		в строке s будет записано "Привет, Марк!

операция получения среза, используется для выделения части строки (подстроки)	`s = "0123456789"`
	`s1 = s[3:8]`
	означает символы строки s с 3-го по 7-й (то есть до 8-го, не включая его)
		в строке s будет записано значение «34567»

для удаления части строки необходимо составить новую строку, объединяя части исходной строки до и после удаляемого участка	`s = "0123456789"`
	`s1 = s[:3] + s[9:]`
	Срез s[:3] означает «от начала строки до символа s[3], не включая его», а запись s[9:] — «все символы, начиная с s[9] до конца строки».
		в переменной s1 остается значение «0129»

реверс строки (запись наоборот)	`s1 = s[::-1]`	Так как начальный и конечный индексы элементов строки не указаны, задействована вся строка. Число «−1» означает шаг изменения индекса и говорит о том, что все символы перебираются в обратном порядке

методы upper и lower позволяют перевести строку соответственно в верхний и нижний регистр	`s = "aAbBcC"`
	`s1=s.upper()# "AABBCC"`
	`s2=s.lower()# "aabbcc"`	

Продолжение

1	2	3

функция поиска в строках find	s="Нур-Султан мой город."
	n=s.find ("у")#n=2
	if n >= 0:
	print("Номер символа", n)
	else:
	print("Символ не найден.")
		Метод find возвращает целое число — номер символа, с которого начинается образец (буква "у") в строке s. Если в строке несколько образцов, функция находит первый из них. Если образец не найден, функция возвращает «−1». В рассмотренном примере в переменную n будет записано число 2

О других встроенных функциях, которые не представлены в данном учебнике, вы можете узнать при желании в Интернете.

Контрольные вопросы

1. Что такое *символьная строка*?
2. Как задать значение для символьной строки?
3. Как вычисляется длина строки?
4. Как обратиться к элементу строки с заданным номером?
5. Какие основные операции со строками вы знаете? Приведите примеры.

Практикум

Уровень А

Введите с клавиатуры в одну строку свою фамилию, имя, отчество, разделяя их пробелом.

Уровень В

Напишите программу, которая во введенной символьной строке заменяет все буквы «а» на буквы «б» и наоборот, как заглавные, так и строчные. При вводе строки «абсАБС» должен получиться результат «басБАС».

Уровень С

Введите строку, в которой записана сумма натуральных чисел, например, «1 + 25 + 3». Вычислите это выражение.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§21 Работа со строками. Примеры

Вы научитесь:

- использовать процедуры и функции для обработки строк.

Ключевые понятия:

- символьная строка
- длина строки
- конкатенация
- получение среза
- реверс строки

Строка — пример объекта в *Python*. Объекты содержат данные (фактически саму строку) и методы, которые являются функциями, встроенными в *объект* и доступными для любого экземпляра (*instance*) объекта.

В *Python* есть *функция* dir, которая выводит *список* доступных методов для объекта. *Функция* type показывает *тип объекта*:

```
>>> stuff = 'Hello world'
>>> type(stuff)
< type 'str'>
>>> dir(stuff)
['capitalize', 'center', 'count', 'decode', 'encode',
'endswith', 'expandtabs', 'find', 'format', 'index',
'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'partition', 'replace', 'rfind', 'rindex', 'rjust',
'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
>>> help(str.capitalize)
Help on method_descriptor:
capitalize(...)
S.capitalize() -> string
Return a copy of the string S with only its first character
capitalized.
>>>
```

Метод вызывается подобно функции — он принимает на вход аргументы и возвращает *значение*, но различается *синтаксис*. Мы вызываем метод путем добавления имени метода к имени переменной, используя точку в качестве разделителя.

Например, метод *upper* получает на вход строку и возвращает новую строку со всеми буквами в верхнем регистре:

```
>>> word = 'banana'
>>> new_word = word.upper()
>>> print new_word
BANANA
```

Пустые круглые скобки означают, что метод не имеет аргументов. Вызов метода называется *вызовом* (*invocation*), в этом случае мы можем сказать, что вызвали метод *upper* объекта *word*.

Рассмотрим *строковый* метод *find*:

```
>>> word = 'banana'
>>> index = word.find('a')
>>> print index
1
>>>
```

В этом примере мы вызвали метод find объекта word и передали в качестве входного параметра букву, которую ищем. Метод find может искать подстроки, а не только отдельные символы:

```
>>> word.find('na')
2
```

В качестве второго аргумента метод find принимает *индекс*, с которого начинается *поиск* вхождения:

```
>>> word.find('na', 3)
4
```

Одной из распространенных задач является устранение пробелов (пробелов, табуляции или символов перевода строки) с самого начала и до конца строки с помощью метода strip:

```
>>> line = '  Here we go  '
>>> line.strip()
'Here we go'
```

Некоторые методы, такие как startswith возвращают логические значения:

```
>>> line = 'Please have a nice day'
>>> line.startswith('Please')
True
>>> line.startswith('p')
False
```

Предварительно применим метод lower, переводящий строку в нижний *регистр*:

```
>>> line = 'Please have a nice day'
>>> line.startswith('p')
False
>>> line.lower()
'please have a nice day'
>>> line.lower().startswith('p')
True
```

В последнем примере мы вызвали в одном выражении подряд два метода.

Разбор (parsing) строк

Часто нам нужно в строке найти подстроку. Для примера, если мы имеем несколько строк, отформатированных следующим образом:

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

и хотим вытащить только вторую часть адреса (например, uct.ac.za) из каждой строки, это можно сделать с использованием метода find и строкового среза.

Во-первых, мы находим положение at-символа (@) в строке. Затем находим позицию первого пробела после at-символа. После этого с использованием строкового среза извлекаем часть строки, которую ищем:

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan 5
09:14:16 2008'
>>> atpos = data.find('@')
>>> print atpos
21
>>> sppos = data.find(' ',atpos)
>>> print sppos
31
>>> host = data[atpos+1:sppos]
>>> print host
uct.ac.za
```

Мы используем версию метода *find*, позволяющую нам задать позицию в строке, начиная с которой хотим выполнять *поиск*.

Оператор форматирования

Оператор форматирования % позволяет создавать строки, заменяя часть строки с данными, хранящимися в переменных. Когда он применяется к целым числам — это операция взятия по модулю.

Первым оператором является строка форматирования (*format string*), которая содержит одну или более последовательностей форматирования (*format* sequences), определяющих *форматирование* второго оператора.

Результатом является строка.

Например, последовательность форматирования '%d' означает, что второй оператор должен быть отформатирован как целочисленное *значение* (d от decimal):

```
>>> camels = 42
>>> '%d' % camels
'42'
```

Результатом является строка '42', которую не следует путать с целочисленным значением 42.

Последовательность форматирования может появиться в любой части строки, поэтому возможно встраивать последовательность в предложение:

```
>>> camels = 42
>>> 'I have spotted %d camels.' % camels
'I have spotted 42 camels.'
```

Если есть более чем одна форматируемая последовательность в строке, второй *аргумент* должен быть кортежем (*tuple*). Каждая форматируемая последовательность сочетается *по* порядку с элементом кортежа. В следующем примере используется '%d' — *форматирование* для целочисленных значений, '%g' — для чисел с плавающей точкой, и '%s' — для форматирования строки:

```
>>> 'In %d years I have spotted %g %s.' % (3, 0.1,
'camels')
'In 3 years I have spotted 0.1 camels.'
```

Количество элементов в кортеже должно совпадать с количеством форматируемых последовательностей в строке. Кроме того, типы элементов должны соответствовать форматируемым последовательностям:

```
>>> '%d %d %d' % (1, 2)
TypeError: not enough arguments for format string
>>> '%d' % 'dollars'
TypeError: illegal argument type for built-in operatio
```

Контрольные вопросы

1. Как вычисляется длина строки?
2. Какие основные операции со строками и соответствующие им стандартные функции вы знаете?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§22 Работа с файлами

Вы научитесь:

- использовать файлы для чтения и записи информации.

Ключевые понятия:

- текстовые файлы
- двоичные файлы

Наверняка при работе с различными прикладными программами вы заметили, что вводить с клавиатуры большое количество данных очень утомительно, особенно если это надо делать несколько раз. В таких случаях создают файл на диске, в который записывают нужные данные, а программа сама читает данные из файла.

Файл — вещь универсальная. Бывают *текстовые* файлы (в которых можно записывать только буквы, цифры, скобки и т. п.) и *двоичные* (в которых могут храниться любые символы из таблицы). Мы рассмотрим только текстовые файлы.

В Python каждый файл рассматривается как последовательный поток байтов. Поток, передаваемый программе, называется *входным*, а поток, передаваемый программой, — *выходным*.

Для работы с файлом в языках программирования используется специальная переменная, которая называется *указателем на файл*. Это адрес блока данных в памяти, в котором хранится вся информация об открытом файле. В Python используется функция open, которая открывает файл и возвращает файловый указатель — переменную через которую идет вся работа с файлом.

Функция open принимает два параметра: имя файла (или путь к файлу) и режим открытия файла:

«r» — чтение;
«w» — запись в новый файл;
«a» — добавление в конец файла.

Иногда программа не может открыть файл. Если файл не открывается на чтение, это возможно в следующих случаях:

— неверно задано имя файла или файла нет на диске;

— файл используется другой программой и заблокирован.

Если файл открывается на запись, операция может закончиться неудачно, если:

— на диске нет места;

— файл защищен от записи;

— неверно задано имя файла (например, оно содержит две точки, знак вопроса и т. п.).

Метод close закрывает файл (рис. 22.1):

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

Рис. 22.1

Если файл не удалось открыть, не найден, возникает ошибка. Если существующий файл открывается на запись, его содержимое уничтожается. Рассмотрим основные функции для чтения/записи из/в файл (таблица 22.1).

Таблица 22.1

Основные функции для чтения или записи

Описание метода	Пример
`readline` — чтение одной строки из текстового файла	`s = Fin.readline()`
`split` — разбивает строку по пробелам и строит список из соответствующих «слов»	`s = Fin.readline().split()`
`int` — применяется, когда необходимо выполнить вычисления с символьными строками	`a, b = int(s[0]), int(s[1])`
`write` — для вывода строки в файл	`Fout.write ( "{:d} + {:d} = {:d}\n".format(a, b, a+b) )`

Контрольные вопросы

1. Чем отличаются текстовые и двоичные файлы по внутреннему содержанию?
2. Какая переменная используется для работы с файлом?
3. Назовите основные функции для чтения или записи из (в) файла.

Практикум

Уровень А

Напишите программу, которая находит среднее арифметическое всех чисел, записанных в файле в столбик, и выводит результат в другой файл.

Уровень B

Напишите программу, которая находит минимальное и максимальное среди четных положительных чисел, записанных в файле, и выводит результат в другой файл. Учтите, что таких чисел может вообще не быть.

Уровень C

В файле в столбик записаны целые числа. Напишите программу, которая определяет длину самой длинной цепочки идущих подряд одинаковых чисел и выводит результат в другой файл.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§23 Работа с файлами. Примеры

Вы научитесь:

- использовать файлы для чтения и записи информации.

Ключевые понятия:

- текстовые файлы
- двоичные файлы

Чтение файлов

До тех пор, пока *дескриптор* файла не содержит данные, создадим конструкцию с оператором for, читая в цикле и увеличивая *счетчик* для каждой строки в файле.

```
>>> fhand = open('mbox.txt')
>>> count = 0
>>> for line in fhand:
count = count + 1
>>> print('Line Count:', count)
Line Count: 132045
```

Мы можем использовать *дескриптор* файла как последовательность в цикле for. Наш цикл считает количество строк в файле и выводит окончательное *значение* на экран. Перевод *цикла* for на человеческий язык: «для каждой строки в файле, представленной файловым дескриптором, увеличить переменную *count* на единицу».

Причина, *по* которой *функция open* не читает весь *файл*, заключается в том, что *файл* может иметь гигабайты данных. *Функция open* занимает одинаковое количество времени, независимо от размера файла.

В случае, когда *файл* читается с использованием *цикла* for, *Python* заботится о разбиении данных в файле на отдельные строки, используя символ новой строки. *Python* читает каждую строку, используя новую строку, и включает новую строку как последний символ в переменную *line* для каждой итерации *цикла* for.

Поэтому цикл for читает данные по одной строке за раз, это эффективное чтение и подсчет строк в большом файле без загрузки всей памяти для хранения данных. *Программа*, написанная выше, может считать строки в файле любого размера, используя немного памяти для чтения каждой строки, подсчета и их сброса.

Если вы знаете, что *файл* является относительно небольшим по сравнению с размером вашей оперативной памяти, вы можете прочитать весь *файл* в одну строку, используя метод *read* для дескриптора файла.

```
>>> fhand = open('mbox-short.txt')
>>> inp = fhand.read()
>>> print(len(inp))
94626
>>> print(inp[:20])
From stephen.marquar
```

В этом примере все содержимое (94626 символов) файла *mbox-short.txt* прочитано непосредственно в переменную inp. Мы воспользовались строковым срезом для печати 20 символов строки данных, хранящейся в inp.

Помните, что такой способ использования функции *open* возможен, только если *файл* данных будет помещаться в оперативную *память* вашего компьютера, иначе используйте циклы for или *while*.

Поиск через файл

Когда вы ищете данные через *файл*, это очень общий *шаблон* чтения через *файл*, игнорирующий большинство строк и обрабатывающий строки, которые соответствуют определенному критерию. Мы можем объединить *шаблон* для чтения файла со строковыми методами, построив простой механизм поиска.

К примеру, если мы хотим прочитать *файл* и вывести на экран строки, которые начинаются с префикса "*From:*", мы можем воспользоваться строковым методом startswith, выбрав строки с желанным префиксом.

```
>>> fhand = open('mbox-short.txt')
>>> for line in fhand:
if line.startswith('From:') :
```

```
print(line)
```

Программа выполняется, получаем следующий результат:

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
...
```

Мы получили желаемый результат. Но откуда пустые строки? Это результат невидимости символа новой строки. Каждая строка оканчивается новой строкой, так как *функция print* печатает строку из переменной *line*, которая включает новую строку и затем *функция print* добавляет другую новую строку, в результате мы наблюдаем эффект двойного пробела.

Мы могли бы использовать *строковый* срез, чтобы напечатать все, кроме последнего символа, но простой подход заключается в использовании метода rstrip, который удаляет пробелы в правой части строки следующим образом:

```
>>> for line in fhand:
        line = line.rstrip()
        if line.startswith('From:'):
            print(line)
```

Получим следующий результат:

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
...
```

Вы можете усложнить структуру поиска, воспользовавшись инструкцией *continue*. Основная идея поиска в цикле — вы ищете «интересные» строки и пропускаете «неинтересные». И когда находите интересную строку – выполняете с ней какие-то действия.

Мы можем построить цикл *по* следующему шаблону, пропуская неинтересные строки:

```
>>> for line in fhand:
        line = line.rstrip()
        if not line.startswith('From:'):
            continue
        print(line)
```

Вывод программы совпадает с предыдущим.

Мы можем использовать *строковый* метод *find* для имитации поиска в текстовом редакторе. Так как *find* ищет вхождения строки в другой

строке и либо возвращает позицию строки или — 1, если строка не найдена, мы можем написать следующий цикл, чтобы показать строки, которые содержат "@uct.ac.za" (т. е. письма приходят из университета Кейптауна в Южной Африке):

```
>>> fhand = open('mbox-short.txt')
>>> for line in fhand:
        line = line.rstrip()
        if line.find('@uct.ac.za') == -1 :
            continue
        print(line)
```

Получили следующий результат:

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
X-Authentication-Warning: nakamura.uits.iupui.edu: apache
set sender to
stephen.marquard@uct.ac.za using -f
From: stephen.marquard@uct.ac.za
Author: stephen.marquard@uct.ac.za
From david.horwitz@uct.ac.za Fri Jan 4 07:02:32 2008
X-Authentication-Warning: nakamura.uits.iupui.edu: apache
set sender to
david.horwitz@uct.ac.za using -f
From: david.horwitz@uct.ac.za
Author: david.horwitz@uct.ac.za
r39753 | david.horwitz@uct.ac.za | 2008-01-04 13:05:51
+0200 (Fri, 04 Jan
2008) | 1 line
From david.horwitz@uct.ac.za Fri Jan 4 06:08:27 2008
X-Authentication-Warning: nakamura.uits.iupui.edu: apache
set sender to
david.horwitz@uct.ac.za using -f
From: david.horwitz@uct.ac.za
```

Контрольные вопросы

1. Что такое *файловая переменная*? Почему для работы с файлом используют не имя файла, а файловую переменную?
2. Как можно начать чтение данных из файла с его начала?
3. Как определить, что данные в файле закончились?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§24-25 Методы сортировки

Вы научитесь:

- реализовывать алгоритмы сортировки для решения практических задач.

Ключевые понятия:

- понятие сортировки
- метод пузырька
- линейный поиск
- двоичный поиск

Сортировка — это расстановка элементов некоторого списка в заданном порядке.

Существуют разные виды сортировки (по алфавиту, по датам и т. д.), они отличаются лишь процедурой сравнения элементов. Мы рассмотрим простейший вариант сортировки — расстановку элементов массива в порядке возрастания. Программисты придумали множество методов сортировки. Они делятся на две группы:

— понятные, но неэффективные;

— эффективные, но непонятные (быстрая сортировка и т. п.).

Пока мы будем изучать только методы из первой группы, которых достаточно для решения простых задач.

Метод пузырька

Название этого метода произошло от известного физического явления. Идея — пузырек воздуха в стакане воды поднимается со дна вверх.

Для массивов — самый маленький («легкий») элемент перемещается вверх («всплывает»).

Делается это так:

— начиная снизу, сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами;

— за один проход по массиву ***один*** элемент (самый маленький) становится на свое место (рис. 24.1).

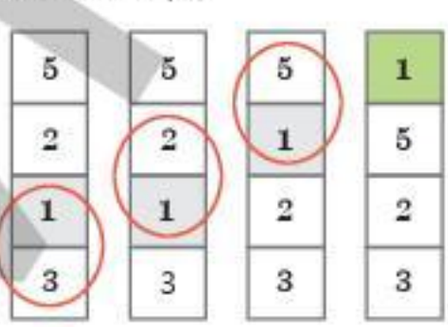


Рис. 24.1. 1-й проход

Далее так же рассматриваем следующую пару элементов и т. д. (рис. 24.2, 24.3)

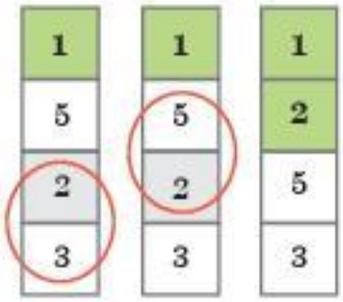


Рис. 24.2. 2-й проход

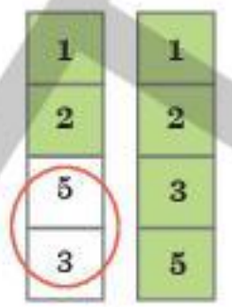


Рис. 24.3. 3-й проход

Для сортировки массива из N элементов нужен N-1 проход (достаточно поставить на свои места $N-1$ элементов).

Метод пузырьковой сортировки представлен в коде (рис. 24.4).

```
#создаем список
li = [0, 5, 8, 4, 9, 3]
#вычисляем длину списка
n = len(li)
#внешний цикл отсчитывает количество "проходов" по списку
for j in range(0,n-1):
    #вложенный цикл сравнивает i-й с i+1-м элементом и при необ-
ходимости меняет их местами
    #количество сравнений каждый раз уменьшается на величину j
    for i in range(0,n-j-1):
        if li[i] < li[i+1]:
            li[i],li[i + 1] = li[i + 1], li[i]
    print(j+1, "-й проход цикла - ",end=" ")
    print(li)
print(li)
```

Рис. 24.4. Фрагмент программы

Метод пузырька работает медленно, особенно на больших массивах. Доказано, что при увеличении размера массива в 10 раз время выполнения программы увеличивается в 100 раз (метод имеет порядок N2).

Линейный поиск

Линейный поиск сравнивает каждый элемент массива с ключом поиска. Поскольку массив не упорядочен, вполне вероятно, что отыскиваемое значение окажется первым же элементом массива. Но программа должна сравнить с ключом поиска половину элементов массива.

В данном алгоритме рассматривается левая и правая граница отрезка массива, где находится нужный нам элемент. Поиск начинается с первого элемента отрезка. Если искомое значение не равно значению функции в данной точке, то осуществляется переход к следующей

точке. Таким образом, в результате каждой проверки область поиска уменьшается на один элемент.

```
def linearSearch(li, x):
    i = 0
    length = len(li)
    while i<length and x!= li[i]:
        i+=1
    return i if i<length else None
```

Рис. 24.5. Отрезок линейного поиска

Метод линейного поиска хорошо работает для небольших или для несортированных массивов (рис. 24.5). Однако для больших массивов линейный поиск неэффективен.

Никлаус Вирт модернизировал данный метод, который позволяет избавиться от одного сравнения на каждом витке цикла. А именно от проверки окончания строки. Для этого мы добавляем искомый элемент в самый конец, что гарантирует остановку по одному условию, а финальная проверка на позицию найденного элемента покажет, подставной он или нет (рис. 24.6).

Никлаус Вирт (1934 г.) — швейцарский ученый, один из известнейших теоретиков в области разработки языков программирования

```
def linearSearchVirt(li, x):
    i = 0
    length = len(li)-1
    while x != li[i]:
        i+=1
    return i if i<length else None
```

Рис. 24.6. Добавление искомого элемента

Далее рассмотрим более эффективный метод — метод двоичного поиска.

Двоичный поиск в массиве

Алгоритм двоичного поиска исключает половину еще непроверенных элементов массива после каждого сравнения. Алгоритм определяет местоположение среднего элемента массива и сравнивает его с ключом поиска. Если они равны, то ключ поиска найден и выдается индекс этого элемента. В противном случае задача сокращается на половину элементов массива. Если ключ поиска меньше, чем средний элемент массива, то дальнейший поиск осуществляется в первой половине массива, а если больше, то во второй половине. Если ключ поиска не

совпадает со средним элементом выбранного подмассива (части исходного массива), то алгоритм повторно применяется и сокращает область поиска до четверти исходного массива. Поиск продолжается до тех пор, пока ключ поиска не станет равным среднему элементу или пока оставшийся подмассив содержит хотя бы один элемент, не равный ключу поиска (т. е. пока не найден ключ поиска).

Рассмотрим следующий код (рис. 24.6):

```
li = [0,3,5,7,10,20,28,30,45,56]
x = 45
i = 0
j = len(li)-1
m = int(j/2)
while li[m] != x and i < j:
    if x > li[m]:
        i = m+1
    else:
        j = m-1
    m = int((i+j)/2)


if i > j:
    print('Элемент не найден')
else:
    print('Индекс элемента: ', m)
```

Рис. 24.6. Добавление искомого элемента

ЭТО ИНТЕРЕСНО!

Двоичный поиск — очень мощный метод. Если, например, длина массива равна 1023, после первого сравнения область сужается до 511 элементов, а после второго — до 255. Легко посчитать, что для поиска в массиве из 1023 элементов достаточно 10 сравнений.

Контрольные вопросы

1. Что такое *сортировка*?
2. Как сортируются элементы массива?

Практикум

Уровень А

1. Составьте блок-схему алгоритма поиска элемента в массиве.
2. Введите программный код, представленный в параграфе, скомпилируйте и запустите программы.

Уровень В

1. Составьте блок-схему алгоритма формирования двумерного массива, элементы которого вводятся с клавиатуры.
2. Напишите программу, которая позволяет найти сумму элементов массива.

Уровень С

1. Примените программы сортировки для упорядочения одномерного массива, содержащего натуральные числа.
2. Напишите программу, которая позволяет вычислить минимальное и максимальное значения в массиве данных.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§26-28 Методы сортировки. Примеры

Вы научитесь:

- реализовывать алгоритмы сортировки для решения практических задач.

Ключевые понятия:

- сортировка
- метод пузырька
- линейный поиск
- двоичный поиск

Сортировка сложных структур с использованием ключа

Это нормально работать с вещами, у которых по природе есть определенный порядок, вроде чисел или строк, но что делать с более сложными структурами? Здесь функция **sort()** демонстрирует свое великолепие.

Функция sort() принимает ключ в качестве опционально названного параметра. Этот ключ сам по себе должен быть функцией, которая принимает один параметр и затем используется функцией sort() для определения значения в целях дальнейшей сортировки. Например, у нас есть класс Person с такими атрибутами, как имя и возраст:

```
class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __repr__(self):
        return "" % (self.name, self.age)
```

Функция __repr__ является специальной функцией, которая используется для переопределения того, как объект будет представлен в интерпретаторе Python.

Причина, по которой мы определили функцию – это **выделение порядка сортировки**. По умолчанию, представление определенных пользователем объектов выглядит примерно так: «<__main__.Person object at 0xb7083ccc>». Если оставить все как есть, то отличать различные экземпляры в будущих примерах будет несколько затруднительно для нас.

Давайте создадим список людей:

```
jack = Person('Jack', 19)
adam = Person('Adam', 43)
becky = Person('Becky', 11)
people = [jack, adam, becky]
```

Сама по себе **функция sort()** не знает, что делать со списком людей:

```
a = sort(people)
print(a) # [<name: Jack, age: 19>, <name: Adam, age: 43>,
<name: Becky, age: 11>]
```

Однако мы можем указать **функции sort()**, какой атрибут сортировать, задав ключ. Давайте определим это в следующем примере:

```
def byName_key(person):
    return person.name
```

Функция ключа должна принять один аргумент и выдать значение, на котором базируется сортировка. Функция sort() должна вызвать **функцию key** в каждом элементе используемой итерируемой, и применить значение выдачи при сортировке списка.

```
a = sort(people, key = byName_key)
print(a) # [<name: Adam, age: 43>, <name: Becky, age:
11>, <name: Jack, age: 19>]
```

Обратите внимание на то, что мы передаем ссылку на саму функцию, не вызывая ее, и передаем ссылку к ее возвращаемому значению. Это очень важный момент. Помните, **sort()** будет использовать функцию key, вызывая ее в каждом элементе итерируемой.

Давайте взглянем на еще один код, на этот раз определяем возраст как значение для сортировки:

```
def byAge_key(person):
    return person.age
a = sort(people, key = byAge_key)
print(a) # [<name: Becky, age: 11>, <name: Jack, age: 19>, <name: Adam, age: 43>]
```

Обратная сортировка

Функция sort() намного упрощает **сортировку в обратном порядке**. Функция принимает опциональный параметр под названием **reverse**, который действует по строгой логике:

```
data = [3, 2, 5 ,4, 7, 1]
a = sorted(data, reverse = True)
print(a) # [7, 5, 4, 3, 2, 1]
data = ('Zane', 'Bob', 'Janet')
b = sorted(data, reverse = True)
print(b) # ['Zane', 'Janet', 'Bob']
```

Сортировка с использованием функции attrgetter

В этот раз возвращаемый список отсортирован по возрасту, как мы и ожидали. Фактически, сортировка по определенному атрибуту объекта — это простая задача Python, которую может выполнить стандартная библиотека, благодаря функции, способной генерировать функции ключей:

```
from operator import attrgetter
```

Результат вызова **attrgetter()** — это функция, схожая с предыдущими двумя, которые мы только что рассмотрели. Мы определяем имя атрибута для выборки, после чего **attrgetter** генерирует функцию, которая принимает объект и возвращает определенный атрибут из этого объекта.

```
getName = attrgetter('name')
result = getName(jack)
print(result) # 'jack'
```

Таким образом, **attrgetter**(name) возвращает функцию, которая ведет себя так же, как и определенная ранее нашей функцией **byName_key()**:

```
result = sort(people, key = attrgetter('name'))
print(result) # [<name: Adam, age: 43>, <name: Becky, age: 11>, <name: Jack, age: 19>]
```

Функция **attrgetter(age)** возвращает функцию, которая ведет себя так же, как и определенная ранее нашей функцией byAge_key():

```
result = sort(people, key = attrgetter('age'))
print(result) # [<name: Becky, age: 11>, <name: Jack, age: 19>, <name: Adam, age: 43>]
```

Предварительное использование key в функции сортировки

До сих пор нашими ключевыми функциями были простые **считыватели атрибутов**, но они также могут вычислять значения для сортировки. Давайте взглянем на еще один пример. На этот раз мы определим **класс Snake**:

```
class Snake(object):
    def __init__(self, name, toxicity, aggression):
        self.name = name
        self.toxicity = toxicity
        self.aggression = aggression
    def __repr__(self):
        return "<%s>" % self.name
```

У нашей змеи есть имя, toxicity (токсичность, мерило того, насколько токсичен ее яд) и agression (степень агрессивности, представленная в виде числа от 0 до 1, которое указывает на вероятность того, что змея нападет).

```
gardenSnake = Snake('gardenSnake', 10, 0.1)
rattleSnake = Snake('rattleSnake', 100, 0.25)
kingCobra = Snake('kingCobra', 50, 1.0)
snakes = [rattleSnake, kingCobra, gardenSnake]
```

Теперь предположим, что мы можем подсчитать, насколько опасная змея, основываясь на показателях токсичности и агрессивности, и можем отсортировать список змей по степени их опасности:

```
def byDangerous_key(snake):
    return snake.toxicity * snake.aggression
result = sort(snakes, key = byDangerous_key)
print(result) # [<gardenSnake>, <rattleSnake>, <kingCobra>]
```

Змеи отсортированы в ожидаемом нами порядке (несмотря на то, что гремучая змея (rattlesnake) более ядовита, чем кобра (kingCobra), уровень агрессивности кобры делает ее более опасной).

Случайная сортировка

Ключи не обязаны иметь какую-либо связь с сортируемыми элементами (однако это не самый продуктивный способ сортировать что-либо). Мы можем создать **случайный порядок** со следующим ключом:

```
from random import random
```

```
def randomOrder_key(element):
    return random()
```

Функция random() — это часть стандартной библиотеки, которая выдает числа в случайном порядке от 0 до 1. Сортировка с использованием данного ключа выдает, случайный порядок:

```
a = sorted(snakes, key = randomOrder_key)
print(a) # [<kingCobra>, <gardenSnake>, <rattleSnake>]

b = sort(snakes, key = randomOrder_key)
print(b) # [<rattleSnake>, <kingCobra>, <gardenSnake>]
```

Контрольные вопросы

1. На какой идее основан метод пузырька? Метод выбора?
2. Сравните на примере метод пузырька и метод выбора. Какой из них требует меньше перестановок?

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§29-30 Алгоритмы на графах

Вы научитесь:

- реализовывать алгоритмы поиска на графах для решения практических задач.

Ключевые понятия:

- дерево
- двоичное дерево
- иерархия
- поиск в ширину
- поиск в глубину

Дерево — одна из наиболее распространенных структур в информатике (рис. 29.1). Деревья можно использовать для хранения информации и как способ представления иерархически связанных данных, обеспечивающий быстрый поиск.

Дерево состоит из узлов и соединяющих их ребер. В Python нет специальной встроенной структуры данных, которые бы соответствовали дереву. Для описания деревьев в Python используют вложенные списки.

В данном параграфе рассмотрим двоичное дерево (рис. 29.2).

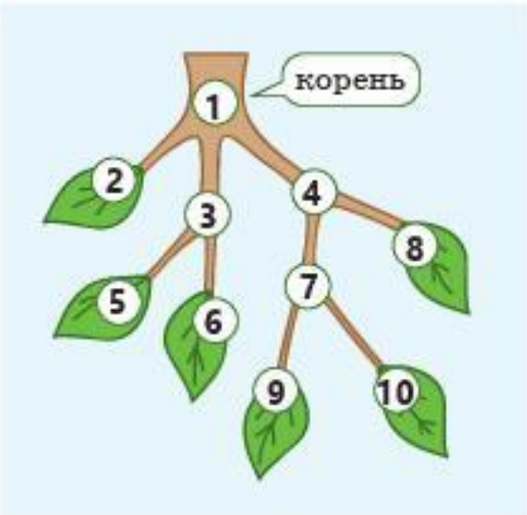


Рис. 29.1. Структура дерева

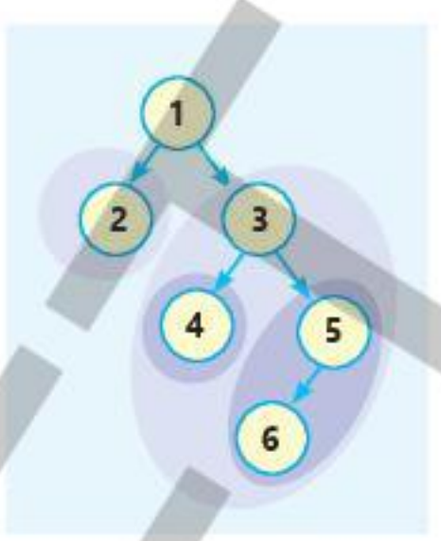


Рис. 29.2. Пример двоичного дерева

Двоичное (бинарное) дерево — это дерево, в котором каждый узел имеет не более двух потомков.

Для хранения данных о двоичном дереве в виде списка используют рекурсивное определение:

— пустое дерево — это двоичное дерево;

— двоичное дерево — это узел и два связанных с ним непересекающихся двоичных поддерева.

Таким образом, двоичное дерево в Python задается списком из трех элементов. Первый элемент — это данные узла (метка), второй — «левый потомок», третий элемент — «правый потомок», допускается, что каждое из поддеревьев может быть пустым.

Рассмотрим следующие примеры:

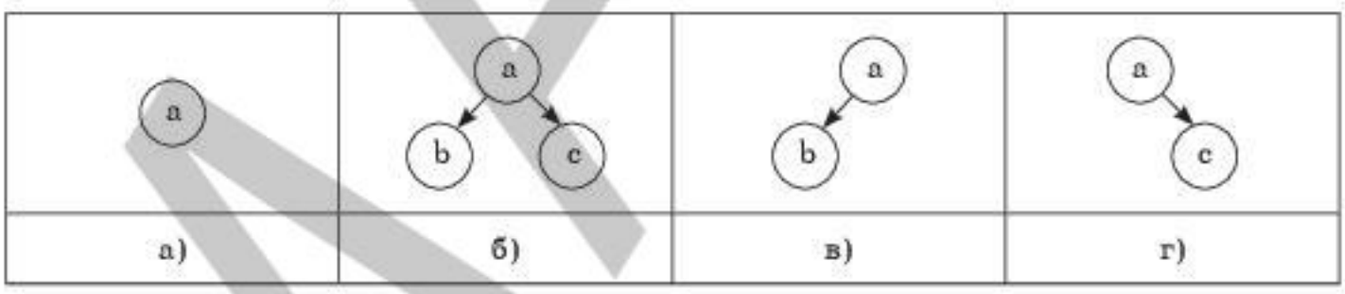


Рис. 29.3. Схемы двоичного дерева

На рисунке 29.3, а) представлено дерево с одной меткой «a» и можно записать в виде [«a», [], []]. Если оба поддерева пустые, их не записывают, запись будет выглядеть как [«a»].

На рисунке 29.3, б) дерево с тремя узлами запишем как [«a», [«b», «c»]]. Деревья на рисунке 29.3, в) и г) соответственно можно записать как [«a», [«b»]] и [«a», [[], «c»]].

Обратите внимание, что в примере г) наличие пустого списка [ ] на месте левого потомка обязательно, иначе узел с меткой «с» будет рассматриваться как левый потомок, а не правый (для многих задач это существенно).

Приведем пример задачи обхода узлов двоичного дерева. Простейший вариант — это обход в порядке КЛП («Корень — Левое поддерево — Правое поддерево»).

Допустим, что двоичное дерево задано в виде списка *T*. Первый элемент этого списка *T* [0], левое поддерево *T* [1], правое поддерево *T* [2]. Для каждого узла необходимо выполнить следующие действия:

— обработать(посетить) узел (для этого будем просто выводить на экран метку узла;

— обойти все поддеревья, сначала левое, потом правое.

Алгоритм обхода получается рекурсивным (рис. 29.4).

```
def DFS ( T ):
    if not T: return
    print(T[0], end=" ")
    for sub in T[1:]:
      DFS ( sub )
```

Рис. 29.4. Алгоритм обхода

Если функции передано пустое дерево, происходит выход (окончание рекурсии). Если дерево непустое, на экран выводится метка узла и затем в цикле выполняется обход всех поддеревьев через рекурсивные вызовы.

Функция *DFS* (depth-first search) — поиск в глубину. То есть при таком обходе мы сначала проходим в глубину дерева по самой левой ветке до ее последнего элемента (листа), а потом возвращаемся назад.

Рассмотрим пример обхода дерева (рис. 29.5), который называется «поиск в ширину» (breadth-first search).

Его суть в том, что сначала обрабатывается корень, затем — его «потомки», потом следуют «потомки потомков» и т. д. Такой обход удобно организовать с помощью очереди:

```
def BFS ( T ):
  if not T: return
  Q = [T]
  while Q:
   node = Q.pop(0)
   print(node[0], end=" ")
   for sub in node[1:]):
     if sub: Q.append ( sub )
```

Рис. 29.5. Пример обхода дерева

Сначала в очередь записывается корень дерева. Затем в цикле, пока очередь не пуста, берем один элемент с начала очереди, посещаем этот узел (выводим его метку на экран) и записываем в очередь ссылки на все его непустые поддеревья. При этом сначала из очереди будет извлечено левое поддерево, а потом — правое.

Контрольные вопросы

1. Что такое *двоичное дерево*?
2. Расскажите суть алгоритма поиска в ширину?
3. Расскажите суть алгоритма поиска в глубину?

Практикум

Уровень А

Составьте дерево игры

Перед двумя игроками лежат две кучки камней, в первой из которых три, а во второй — два камня (рис. 29.6). У каждого игрока неограниченно много камней.

Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в три раза число камней в какой-то куче, или добавляет один камень в какую-то кучу.

Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 16.

Кто выигрывает при безошибочной игре — игрок, делающий первый ход, или игрок, делающий второй ход? Как должен ходить выигрывающий игрок?

Рис. 29.6

Уровень B

Составьте программу для вычисления правильного арифметического выражения, включающего только однозначные числа и знаки операций +, -, * , /.

Уровень C

То же самое, но допускаются также многозначные числа и скобки.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

Проверь себя!

1. Как называется подпрограмма, не возвращающая значение:
   а) процедура;
   б) функция;
   в) структура;
   г) итерация;
   д) рекурсия?
2. В чем особенность подпрограмм-функций:
   а) функция, кроме выполнения определенных действий, возвращает еще значение, которое может быть использовано в правой части оператора присваивания;
   б) у них обязательно должны быть параметры;
   в) у них не должно быть параметров;
   г) у них постоянное число параметров;
   д) у них переменное число параметров?
3. В программе используется символьная строка s:
   s = "Привет от старых штиблет!" Отметьте все правильные операторы, работающие с этой символьной строкой:
   а) s = "12345";
   б) "12345" = s;
   в) s[1] = "0";
   г) s1 = s[:];
   д) s[:] = s1.

5. Что будет выведено на экран после выполнения программы:

а) s = "123";
б) s = s + "0" + s;
в) s = s + s;
г) print (s)?

6. Что будет выведено на экран после выполнения этой программы:

а) s = "0123456";
б) q = "abc";
в) s = s[:2] + s[5:];
г) q = q[0] + s + q[1:];
д) print (q)?

7. Что будет выведено на экран после выполнения этой программы:

а) s = "0123456789";
б) n = s.find("456");
в) q = s[:n+2] + s[n+5:];
г) print (q)?

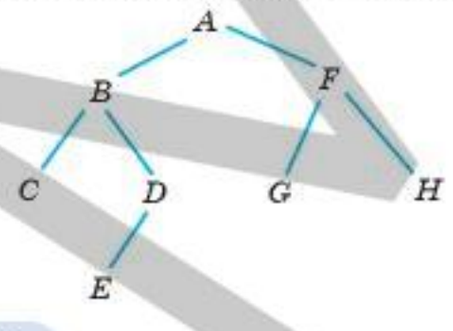
8. Что такое *дерево*:

а) это граф без циклов, одна из вершин в нем считается корнем дерева;
б) это граф без ребер, одна из вершин в нем считается корнем дерева;
в) это граф без вершин, только из ребер;
г) это граф, в котором нет петель;
д) это граф, в котором имеются все возможные между вершинами ребра?

9. Что такое *двоичное* (*бинарное*) *дерево*:

а) это дерево, каждый узел (вершина) которого имеет не более двух потомков;
б) это дерево, каждый узел (вершина) которого имеет по два потомка;
в) это дерево, в котором не более двух ребер;
г) это дерево, в котором ровно два ребра;
д) это дерево, в котором имеется два корня?

10. Выполните прямой обход следующего двоичного (бинарного) дерева:



а) A, B, C, D, E, F, G, H;
б) C, B, E, D, A, G, F, H;
в) C, E, D, B, G, H, F, A;
г) H, G, F, E, D, C, D, A;
д) A, B, F, C, D, G, H, E.

11. Сколько сравнений потребуется при последовательном поиске в массиве из 100 элементов:
а) максимум 100;
б) максимум 20;
в) минимум 5;
г) минимум 25;
д) в среднем 20?

12. Сколько сравнений чисел будет произведено при последовательном поиске числа 4 в массиве чисел 1, 2, 3, 4, 5, 6, 7:
а) 4;
б) 5;
в) 6;
г) 1;
д) 2?

13. Как еще называют двоичный поиск:
а) бинарный поиск, метод деления пополам, дихотомия;
б) четный;
в) двойственный;
г) парный;
д) сдвоенный?

14. Сколько сравнений потребуется при двоичном поиске в массиве из 100 элементов:
а) максимум 7;
б) максимум 5;
в) максимум 2;
г) минимум 5;
д) минимум 2?

15. Какой оператор надо вставить вместо многоточия, чтобы в строке S были записаны символы строки Q в обратном порядке:
а) Q = "0123456789"
б) S = "0"
в) for k in range(1,10)?

РАЗДЕЛ 4

Информационные системы

Из данного раздела вы узнаете:

- что такое Bigdata;
- основные понятия баз данных;
- как работать с базой данных;
- первичный ключ в базе данных;
- что такое форма, отчеты, запросы в базе данных;
- что такое структурированные запросы.

Вы научитесь:

- оценивать положительные и отрицательные стороны использования Bigdata;
- формулировать понятие «реляционная база данных»;
- определять первичный ключ в базе данных;
- определять типы данных полей базы данных;
- создавать однотабличную и многотабличную базу данных;
- создавать форму для ввода данных;
- создавать отчеты, используя извлеченные данные;
- создавать запросы на выборку с помощью конструктора;
- использовать структурированный язык запросов.

§31 Bigdata

Вы научитесь:

- оценивать положительные и отрицательные стороны использования Bigdata.

Ключевые понятия:

- Bigdata — большие данные

Термин Bigdata появился сравнительно недавно, в 2011 г., и с тех пор используется довольно часто.

Так что же такое Bigdata на самом деле?

В различных науках (маркетинге, экономике) используют следующие определения:

— Bigdata — это когда данных больше, чем 500 Гб, 1 Тб.;

— Bigdata — это такие данные, которые невозможно обработать в Excel;

— Bigdata — это такие данные, которые невозможно обработать на одном компьютере.

Bigdata (большие данные) — серия подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объемов для получения воспринимаемых человеком результатов, в условиях распределения по многочисленным узлам.

Будем под Bigdata понимать не какой-то объем данных, а методы их обработки, которые позволяют распределенно обрабатывать информацию.

Эти методы можно применить как к огромным массивам данных (например, содержание всех страниц в Интернете), так и к маленьким, (например, ваш проект по данной теме).

Для чего нужны Bigdata?

Во-первых, хранение и управление объемом данных в сотни терабайт или петабайт, который обычные реляционные базы данных не позволяют эффективно использовать.

Во-вторых, организация неструктурированной информации, состоящей из текстов, изображений, видео и других типов данных.

В настоящее время количество источников стремительно увеличивается, одновременно растут возможности хранения информации, а значит, технологии их обработки становятся все более востребованными.

Начиная с 2000-х годов, происходит переломный момент — цифровые носители получили широкое распространение, тем самым стала доступна информация уже в цифровом виде (рис. 31.1). Отметим, что особый вклад в развитие цифровой эпохи внесли жесткие диски. Удешевление их производства можно считать основным фактором формирования тренда больших данных.

Рис. 31.1. Результаты внедрения Bigdata

Приведем несколько примеров того, что может быть источником данных, для которых необходимы методы работы с большими данными (рис. 31.2):

— GPS-сигналы от автомобилей для транспортной компании;
— оцифрованные книги в Национальной библиотеке;
— информация о транзакциях всех клиентов банка;
— информация о всех покупках в крупной торговой сети и т. д.

Рис. 31.6. Схема работы с большими данными

Алгоритм работы с Bigdata можно описать следующим образом: сбор данных, структурирование полученной информации с помощью отчетов и дашбордов (dashboard), создание инсайтов и контекстов, а также формулирования рекомендаций к действию.

Контрольные вопросы

1. Что понимается под *Bigdata*?
2. Приведите примеры использования больших данных.
3. Опишите алгоритм работы с Bigdata.

Практикум

Уровень А

Разработайте проект «Что такое *дашборд*».

Уровень В

Разработайте проект «Как банки могут использовать Bigdata для предотвращения мошенничества».

Уровень С

Разработайте проект «Как работают с Bigdata телекоммуникационные провайдеры».

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§32–33 Bigdata. Проектная работа

Вы научитесь:

- оценивать положительные и отрицательные стороны использования Big data.

Ключевые понятия:

- Bigdata — большие данные

Темы для проектов

1. Применение больших данных в медицине

Реализация технологий Bigdata в медицинской сфере (рис. 32.1) позволяет врачам более тщательно изучить болезнь и выбрать эффективный курс лечения для конкретного случая. Благодаря анализу информации, медработникам становится легче предсказывать рецидивы и предпринимать превентивные меры. Как результат — более точная постановка диагноза и усовершенствованные методы лечения.

Новая методика дает возможность взглянуть на проблемы пациентов с другой стороны, что приводит к открытию ранее неизвестных источников проблемы.

Рис. 32.1. Технологии в медицинской сфере

Например, некоторые расы генетически более предрасположены к заболеваниям сердца, нежели представители других этнических групп. Теперь, когда пациент жалуется на определенное заболевание, врачи берут во внимание данные о представителях его расы, которые сталкивались с такой же проблемой. Сбор и анализ данных позволяет узнавать о больных намного больше: от предпочтений в еде и стиля жизни до генетической структуры ДНК и метаболитах клеток, тканей, органов.

Еще один пример — случай в Лурдском медицинском центре Богоматери в Нью-Джерси. В то время, как пациент проходил обследование из-за нового приступа судороги, вызванного пропущенным приемом лекарств, врачи обнаружили, что мужчина имеет куда более серьезную проблему со здоровьем. Этой проблемой оказалась фибрилляция предсердий. Диагноз удалось поставить благодаря тому, что сотрудники отделения получили доступ к телефону пациента, а именно к приложению, сопряженному с его фитнес-трекером. Данные с приложения оказались ключевым фактором в определении диагноза, ведь на момент обследования у мужчины никаких сердечных отклонений обнаружено не было.

Это всего лишь два примера, которые показывают, почему использование больших данных в медицинской сфере сегодня играет столь значимую роль.

2. Bigdata — стержень розничной торговли

Например, сети некоторых магазинов с помощью глубинного анализа данных и собственной системы прогнозирования удается с высокой точностью определить вкусы потребителей. За каждым клиентом закрепляется ID, который в свою очередь привязан к кредитке, имени или электронной почте. Идентификатор служит своеобразной корзиной покупок, где хранится информация обо всем, что когда-либо человек приобрел.

3. Большие данные на страже закона и порядка

За последние несколько лет правоохранительным структурам удалось выяснить, как и когда использовать большие данные. Общеизвестным фактом является то, что службы национальной безопасности применяют технологии больших данных, чтобы предотвратить террористические акты. Другие ведомства задействуют прогрессивную методологию, чтобы предотвращать более мелкие преступления.

4. Как технологии Bigdata помогают развиваться городам

Анализ данных также применяется для улучшения ряда аспектов жизнедеятельности городов и стран. Например, зная точно, как и ког-

да использовать технологии Bigdata, можно оптимизировать потоки транспорта. Для этого берется в расчет передвижение автомобилей в режиме онлайн, анализируются социальные медиа и метеорологические данные. Например, концепция умного города, в котором автобусы ждут опаздывающий поезд, а светофоры способны прогнозировать загруженность на дорогах, чтобы минимизировать пробки.

Таблица 32.1

Критерии оценки проекта

Критерий	0 баллов	1 балл	2 балла	3 балла
1	2	3	4	5
Уровень постановки исследовательской проблемы	Работа репродуктивного характера — присутствует лишь информация из других источников, нет обобщений, нет содержательных выводов	Работа в целом репродуктивна, но сделаны неплохие самостоятельные обобщения	Работа частично поисковая — в работе есть проблемы, которые имеют частный характер (не отражающие тему в целом, а касающиеся только каких-то ее аспектов)	Работа исследовательская, полностью посвящена решению одной научной проблемы, пусть не глобального плана, но сформулированной самостоятельно
Актуальность и оригинальность темы	Тема всем известная, изучена подробно, в литературе освещена полно. При этом автор не сумел показать, чем обусловлен его выбор кроме субъективного интереса, связного с решением личных проблем или любопытством	Тема изученная, но в ней появились «белые пятна» вследствие новых данных, либо тема относительно малоизвестная, но проблема «искусственная», не представляющая истинного интереса для науки	Тема с достаточным количеством «белых пятен», либо проблема поставлена достаточно оригинально, вследствие чего тема открывается с неожиданной стороны	Тема малоизученная, практически не имеющая описания, для раскрытия которой требуется самостоятельно делать многие выводы, сопоставляя точки зрения из соседних областей исследования
Логичность доказательства (рассуждения)	Работа представляет собой бессистемное изложение того, что известно автору по данной теме	В работе можно заметить некоторую логичность в выстраивании информации, но целостности нет	В работе либо упущены некоторые важные аргументы, либо есть «лишняя» информация, перегружающая текст ненужными подробностями, но в целом логика есть	Цель реализована последовательно, сделаны необходимые выкладки, нет «лишней» информации, перегружающей текст ненужными подробностями

Продолжение

1	2	3	4	5
Корректность в использовании литературных источников	В работе практически нет ссылок на авторов тех или иных точек зрения, которые местами могут противоречить друг другу и использоваться не к месту	Противоречий нет, но ссылок либо практически нет, либо они делаются редко, далеко не во всех необходимых случаях	Текст содержит наиболее необходимые ссылки на авторов в тех случаях, когда делается информация принципиального содержания (определения, обобщения, описания, характеристика, мнение, оценка и т. д.)	Текст содержит все необходимые ссылки на авторов в тех случаях, когда дается информация принципиального содержания (определения, описания, обобщения, характеристика, мнение, оценка т. д.), при этом автор умело использует чужое мнение при аргументации своей точки зрения, обращаясь к авторитетному источнику
Количество источников	Нет списка литературы	1—2 источника	Список имеет несколько источников, но упущены некоторые важные аспекты рассматриваемой проблемы	Список охватывает все основные источники по данной теме, доступные ученику
Глубина исследования	Работа поверхностна, иллюстративна, источники в основном имеют популярный характер	Работа строится на основе одного серьезного источника, остальные — популярная литература, используемая как иллюстрация	Рассмотрение проблемы строится на содержательном уровне, но глубина рассмотрения относительна	Рассмотрение проблемы строится на достаточно глубоком содержательном уровне
Оформление	Оформление носит абсолютно случайный характер, обусловленный собственной логикой автора	Работа имеет какую-то структуру, но нестрогую	Работа в общем соответствует требованиям, изложенным в следующей графе, но имеет некоторые недочеты, либо одно из требований не выполняется	Работа имеет четкую структуру, обусловленную логикой темы, правильно оформленный список литературы, корректно сделанные ссылки и содержание (оглавление)

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§34–35 Основные понятия баз данных

Вы научитесь:

- объяснять понятие «реляционная база данных»;
- формулировать определения терминов: поле, запись, индекс;
- определять первичный ключ в базе данных.

Ключевые понятия:

- реляционная база данных
- ключевое поле

Каждый человек в жизни постоянно сталкивается с базами данных. Это — различные справочники, каталоги, энциклопедии и т. п. Например, база данных «Записная книжка» хранит информацию о людях, каждый из которых имеет свою фамилию, имя, телефон и т. д.; библиотечный каталог хранит информацию о книгах, каждая из которых имеет свое название, автора, год издания и т. д.

Информация в базах данных хранится в упорядоченном виде. Так, в записной книжке все записи упорядочены по алфавиту, а в библиотечном каталоге — либо по алфавиту (алфавитный каталог), либо по области знания (предметный каталог).

База данных — это организованная структура, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковыми признаками.

Рассмотрим, как же организована база данных? В ней содержатся сведения о большом количестве однотипных объектов. При этом для каждого из объектов существенными являются значения лишь некоторых признаков. (Термин «признак» применяется для обозначения какого-либо признака или свойства — поле).

Например, для базы данных школьной библиотеки (табл. 34.1) объектами являются: художественная и техническая литература, подписки газет, журналов и т. д., а их признаками будут: жанр, название книги, фамилия автора, год издания, количество страниц, есть ли книга в наличии или выдана кому-то из читателей.

Таблица 34.1

Фрагмент базы данных «Школьная библиотека»

Инв. №	ФИО автора	Произведение	Год издания	Кол-во стр.
1054	Чехов А. П.	Дом с мезонином	1983	320
1298	Куприн А. И.	Гранатовый браслет	1980	320
3762	Тургенев И. С.	Вешние воды	1986	560

Особенность базы данных в том, что у всех объектов, входящих в эту базу, количество признаков (полей) одно и то же. Иногда у некоторых из этих признаков может не быть значения (например, при составлении базы данных телефонов учащихся класса у некоторых в поле «телефон» будет прочерк, т. е. отсутствует значение).

Часто некоторые из признаков объявляют ***ключевыми***. Это важно, потому что по ним в дальнейшем можно делать сортировку.

Некоторые из признаков могут быть объявлены обязательно присутствующими. Например, у каждой книги, находящейся в библиотеке, имеется инвентарный номер и код.

Обычно доступ к базе данных есть у достаточно большого количества людей, но среди них всего один лишь человек имеет доступ ко всей базе полностью и при этом единолично вносит в нее произвольные изменения. Кроме данных база содержит методы и средства, позволяющие каждому из сотрудников организации оперировать только с теми данными, на которые у него есть право.

Вся информация в базе данных имеет упорядоченный вид. Чтобы управлять данными, составляющими базу, необходима отдельная программа — система управления базами данных.

Система управления базами данных (СУБД) — это программное обеспечение, с помощью которого реализуется хранение, поиск и обработка данных.

В настоящее время существует несколько классификаций СУБД, различающих архитектурой, внутренним языком программирования, операционной системой, под управлением которой они работают, способом организации хранения и обработки данных, а также другими характеристиками. Наиболее популярными являются реляционные СУБД: Access, FoxPro, Paradox. К более сложным относятся распределенные СУБД, которые предназначены для работы с большими базами данных, распределенными на нескольких серверах (серверы

могут находиться в различных областях). СУБД такого типа являются Oracle, Informix.

Типы баз данных

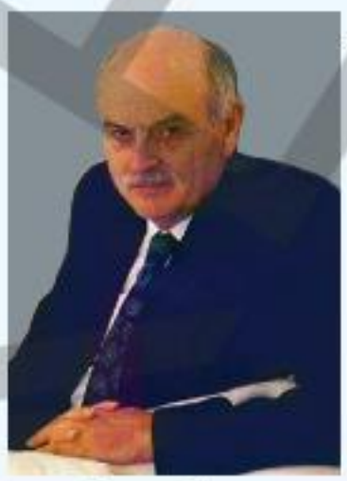
Существует несколько типов баз данных: реляционные (табличные), иерархические и сетевые.

Реляционные (табличные) базы данных. В настоящее время предпочтение отдается реляционным базам данным. Реляционная (от слова *relation* — «отношение») база данных предполагает использование двумерных таблиц (отношений).

ЭТО ИНТЕРЕСНО!

Реляционная модель баз данных была предложена в конце 60-х годов прошлого века Э.Коддом. Согласно его основной концепции, реляционная база данных представляет собой хранилище данных, содержащее набор двумерных взаимосвязанных таблиц.

Доктор университета штата Мичиган (США) Эдгар Кодд (1923–2003) внес огромный вклад в развитие теории реляционных баз данных и средств связи.



Эдгар Кодд

Двумерные таблицы состоят из строк, называемых в терминологии баз данных *записями*, и столбцов, которые именуются *полями*.

В качестве примера рассмотрим таблицу 34.2 «Товары», в которой каждая строка содержит сведения об одном из поставляемых товаров.

Таблица 34.2

Товары

	поле 1	поле 2	поле 3	поле 4	поле 5
№	Код	Поставщик	Тип товара	Ед. измерения	Цена, тенге
1	1	ИП Ахметов	Вафли	коробка (5 кг)	360
2	2	ЗАО «Батыр-Баян»	Мармелад	коробка (10 кг)	420
3	3	ТОО «Азия»	Зефир	коробка (8 кг)	350

В полях (столбцах таблицы 34.2) находятся основные характеристики объекта данных. Каждое из полей однородно, т. е. данные в нем имеют одинаковые тип и длину. Каждое поле таблицы имеет уникальное имя. Поле, значение которого однозначно определяет соответствующую запись, называется *ключевым полем*.

Обратите внимание, что поле «КОД» данной таблицы имеет уникальные значения, которые не повторяются ни в одной из записей, оно и будет иметь статус *ключевого поля*. На роль ключевого поля не подойдет ни поле «Единицы измерения», ни «Цена» (так как могут быть совпадения), ни любое другое поле.

Если ключевое поле одно, то это простой ключ, если ключевых полей несколько, то это составной ключ.

Иерархические базы данных графически могут быть представлены как перевернутое дерево, состоящее из объектов различных уровней. На самом верхнем уровне структуры находится корень дерева, не имеющий вышестоящих узлов. Остальные узлы (порожденные) связаны между собой и всегда имеют исходный узел, находящийся выше.

В качестве примера рассмотрим иерархическую модель предприятия (рис. 34.1), отражающую:

— организационные отношения в производстве;

— распределение ответственности;

— порядок операций.

Рис. 34.1. Иерархическая модель данных

Сетевые базы данных. Сетевая база данных является обобщением иерархической и предполагает организацию данных в виде древовидной структуры, когда любой элемент может быть связан с любым другим элементом (рис. 34.2).

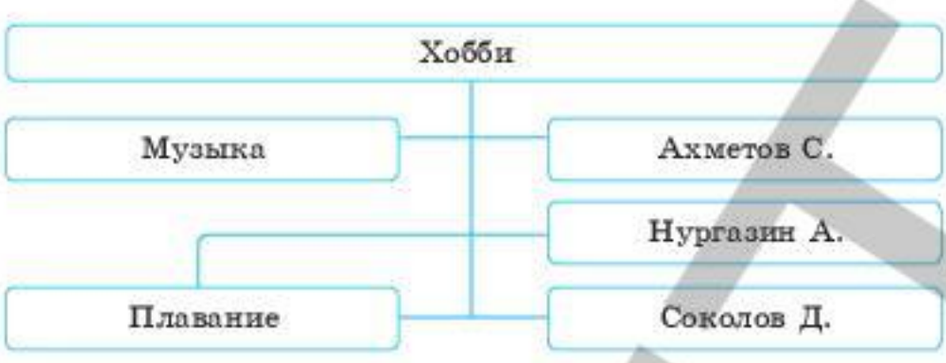


Рис. 34.2. Сетевая модель данных

Недостатком иерархической и сетевой структуры является то, что при добавлении новых вершин или установлении новых связей возникают проблемы потери данных.

Контрольные вопросы

1. Что называется *базой данных*?
2. Что такое *СУБД*?
3. Что такое *реляционные базы данных*?
4. Приведите пример реляционной модели данных.
5. Чем отличается сетевая модель данных от иерархической?
6. Приведите примеры иерархической и сетевой моделей данных.

Практикум

Задание 1

1. Дана сетевая структура базы данных «Вкладчики» (рис. 34.3). Приведите данную структуру к табличному виду.

Рис. 34.3. База данных «Вкладчики»

2. Дана иерархическая структура базы данных «Таблица Менделеева» (рис. 34.4). Приведите данную структуру к табличному виду.

Рис. 34.4. База данных «Таблица Менделеева»

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§36 Первичный ключ в базе данных

Вы научитесь:

- определять первичный ключ в базе данных.

Ключевые понятия:

- база данных
- реляционная база данных
- поле
- запись
- индекс
- первичный ключ

В предыдущем параграфе мы с вами выяснили, что структуру простейшей базы можно представить как прямоугольную таблицу, которая состоит из столбцов и строк. Также вспомним, что вертикальные столбцы называют *полями*, а горизонтальные — *записями*.

Единицей хранимой информации является горизонтальная строка — запись, которая хранит информацию об одном объекте (ученик, книга, товар и др.).

Каждая запись отличается от другой значением хотя бы одного поля, которое называется ***ключом***.

Первичный ключ в базе данных — это поле (или совокупность полей), значение которого не повторяется у разных записей.

Если первичный ключ состоит из одного поля, он называется ***простым,*** если из нескольких полей — ***составным*** ключом.

Таблица 36.1

Фрагмент базы данных «Прогноз погоды»

Дата	Время	Температура	Влажность	Скорость ветра
21.01.2019	12:00	−28	65	6
21.01.2019	16:00	−26	00	5
...	...	...	...	...

Наверное, вы обратили внимание, что в данной таблице 36.1 нельзя выбрать ключ, потому что значение каждого поля повторяется. Однако у разных записей не могут совпадать одновременно значения двух полей: «Дата» и «Время». Они образуют ***составной ключ*** таблицы.

Каждое поле имеет свой формат и тип.

Типы полей

1. *Символьный* — поля этого типа предназначены для хранения в них информации, которая рассматривается как строка символов и может состоять из букв, цифр, знаков препинания и т.п.

2. *Числовой* — поля этого типа предназначены только для хранения чисел.

3. *Дата* — поля этого типа предназначены для хранения каких-либо дат в фиксированном формате: число, месяц, год.

4. *Логический* — поля этого типа предназначены для хранения значений вида «ДА» — «НЕТ» или «ПРАВДА» — «ЛОЖЬ».

Форматы полей

Формат символьного поля определяет число символьных позиций, которое будет занимать поле в записи. Например, если символьное поле имеет формат 10, то его значения в различных записях могут содержать от 0 до 10 символов.

Формат числового поля обычно состоит из двух частей: длины и точности.

Длина — это полное количество символьных позиций, выделяемых под запись числа.

Точность — это количество позиций, выделенное под дробную часть. Следует иметь в виду, что десятичная точка тоже занимает позицию.

Например, формат записи числа 123.45 такой: длина — 6, точность — 2. Целое число, т. е. число без дробной части, имеет точность 3.

Формат логической величины стандартный — 1 символ. Чаще всего используются следующие обозначения: 1 — (истина), 0 — (ложь).

Формат даты обычно имеет длину 8 символов. Более привычен нам стандарт ДД/ММ/ГГ (или ДД.ММ.ГГ, или ДД-ММ-ГГ. Здесь ДД — двузначное обозначение числа, ММ — месяца, ГГ — года. Иногда используется стандарт ММ/ДД/ГГ. Бывают и другие обозначения.

Таким образом, *значения полей* — это некоторые величины определенных типов. От типа величины зависят те действия, которые можно с ней производить. Например, с числовыми величинами можно выполнять арифметические операции, а с символьными и логическими — нельзя.

Контрольные вопросы

1. Объясните значения слов «поле», «запись»?
2. Что понимается под *первичным ключом*?
3. Чем отличается простой ключ от составного? Приведите примеры.
4. Какие типы и форматы полей вы знаете?
5. Какие существуют возможности использования баз данных в повседневной жизни для решения каких-либо проблем?

Практикум

Уровень А

Выберите, какие из этих данных могут быть ключом:

- фамилия;
- имя;
- номер паспорта;
- номер дома;
- регистрационный номер автомобиля;
- город проживания;
- адрес электронной почты;
- дата выполнения работы;
- марка стиральной машины.

Уровень B

Определите типы полей в таблице 36.2.

Таблица 36.2

Погода

День	Осадки	Температура	Давление мм рт. ст.	Влажность %
15.03.2019	Снег ***	–3,5	746	67
16.03.2019	Без осадков	0	750	62
17.03.2019	Снег ****	1,0	746	100

Уровень C

Выберите любую предметную область (страны мира, столицы мира, одноклассники, животные, домашние расходы и т. д.), структурируйте данные (выделите поля), определите тип данных, разработайте структуру таблицы, определите ключевые поля, приведите пример в тетради нескольких записей таблицы. Попытайтесь определить, по каким критериям можно вести поиск в вашей базе данных.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§37–38 Разработка базы данных

Вы научитесь:

- определять типы данных в базе данных;
- создавать однотабличную базу данных;
- создавать многотабличную базу данных.

Ключевые понятия:

- база данных
- реляционная база данных
- поле
- запись
- индекс
- первичный ключ

Таблица является основным объектом базы данных. База данных может состоять из одной или нескольких таблиц. На основании таблиц создаются остальные объекты базы данных.

Для проектирования базы данных необходимо:

1. *Определение цели создания базы данных*

Перед созданием структуры необходимо ответить на вопросы, для чего предназначается создаваемая база данных (описать выбранную предметную область), каковы будут функции и какую информацию она должна содержать. То есть нужно определить основные темы таблиц базы данных и информацию, которую будут содержать поля таблицы.

2. *Определение таблиц данных*

Одним из наиболее сложных этапов в процессе проектирования базы данных является разработка таблиц, так как результаты, которые должна выдавать база данных (отчеты, выходные формы и т. д.), не всегда дают полное представление о структуре таблицы.

При проектировании таблиц рекомендуется руководствоваться следующими принципами:

— информация в таблице не должна дублироваться. Не должно быть повторений и между таблицами;

— если определенная информация в одной таблице, то и изменять ее придется только в одном месте;

— таблица должна содержать информацию на одну тему.

3. *Определение необходимых в таблице полей*

Каждая таблица содержит информацию на отдельную тему, а каждое *поле* в таблице содержит отдельные сведения по теме таблицы.

4. *Задание ключа и определение связей между таблицами*

Для того чтобы связать данные различных таблиц, каждая таблица должна содержать ключевое поле. Значение этого поля позволит выбрать нужную запись в таблице, а также корректно установить связи между таблицами.

5. *Ввод данных и создание объектов базы данных*

Если структуры таблиц отвечают поставленным требованиям, то можно вводить все данные.

Затем можно создавать любые запросы, формы, отчеты, макросы и модули.

Рассмотрим пример. Предположим, что нам необходимо спроектировать базу данных «Библиотека дисков», содержащую сведения об использовании CD и DVD дисков из личной библиотеки пользователя.

Основной задачей при использовании базы данных «Библиотека дисков» является отслеживание выдачи дисков всем желающим клиентам.

Проведя анализ необходимой для хранения информации, попытаемся сначала расположить ее в одной таблице, поля которой разделим на три группы: сведения о клиентах дисков, сведения о наличии дисков и сведения о выдаче дисков. Структура таблицы базы данных в этом случае должна иметь следующий вид (таблица 37.1).

Таблица 37.1

Структура таблицы базы данных «Библиотека дисков»

Имя поля	Тип данных	
Фамилия	Текстовый	Сведения о клиентах дисков
Имя	Текстовый	
Адрес	Текстовый	
Номер телефона	Текстовый	
Адрес электронной почты	Текстовый	
Название диска	Текстовый	Сведения о дисках
Тип диска	Текстовый	
Стоимость диска	Денежный	
Дата выдачи	Дата/время	Сведения о выдаче дисков
Отметка о возврате	Логический	

Работать с такой таблицей достаточно неудобно. Например, при выдаче нескольких дисков для одного клиента будет необходимо многократно повторять информацию о нем: фамилию, имя, адрес и т. д., что приведет к огромному увеличению размера таблицы и может повысить вероятность ошибок при вводе информации и т. д.

При заполнении каждой записи этой таблицы мы видим, что одни поля используются только по одному разу за один визит конкретного клиента, например, название выбранного диска, тогда как другие используются несколько раз: фамилия, адрес клиента. Связано это с тем, что один конкретный клиент может выбрать сразу несколько разных дисков для просмотра.

Для повышения эффективности при работе с создаваемой базой данных необходимо выполнить процесс *нормализации*.

Для этого разделим одну таблицу на три таблицы 37.2, 37.3, 37.4: ***«Клиенты», «Диски», «Выдача дисков»***.

Опишем структуру каждой таблицы.

Таблица 37.2

«Клиенты»

Имя поля	Тип данных
Код клиента	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Адрес	Текстовый
Номер телефона	Текстовый
Адрес электронной почты	Текстовый

Таблица 37.3

«Диски»

Имя поля	Тип данных
Код диска	Счетчик
Название диска	Текстовый
Тип диска	Текстовый
Стоимость диска	Денежный

Таблица 37.4

«Выдача дисков»

Имя поля	Тип данных
Код выдачи	Счетчик
Код клиента	Числовой
Код диска	Числовой
Дата выдачи	Дата/время
Отметка о возврате	Логический

В представленных таблицах впервые присутствуют данные ***Счетчик.***

Он применяется для хранения целых числовых значений, которые Access увеличивает при переходе к каждой новой записи. Счетчик может использоваться в качестве уникального идентификатора записи таблицы, в которой не предусмотрено другой такой величины. В нашем случае *Код клиента, Код диска и Код выдачи* будут уникальными идентификаторами, которые позволят легко отличить одного клиента от другого, один диск от другого и присвоить уникальные специальные значения каждой выдаче дисков.

Поля *Код клиента, Код диска* в дальнейшем будут использованы для установки связей между таблицами, которые мы рассмотрим далее.

Для работы с базой данных будем использовать Microsoft Access. Аналогичные приемы применяются в бесплатной СУБД Base из пакета OpenOffice.org (поэтому практические задания можно выполнять в любой из программ).

В СУБД Microsoft Access вся база данных представляет собой один файл с расширением .accdb. В нем находятся:

— таблицы для хранения данных;

— формы — диалоговые окна, с помощью которых можно вводить и изменять данные;

— запросы — обращения к базе данных, в результате которых отбираются нужные данные или выполняются какие-то другие действия, например, изменение или удаление записей.

— отчеты — шаблоны документов, предназначенные для вывода данных на печать.

После запуска программы Access открывается окно, в котором пользователь должен либо создать новую базу данных, указав местоположение и имя базы данных, либо открыть сохраненную ранее. Основным объектом базы данных являются таблицы, поэтому следующим шагом будет создание таблицы.

СУБД Access позволяет создавать структуру таблицы двумя способами: в режиме Конструктора или Таблицы (путем ввода данных) (рис. 37.1).

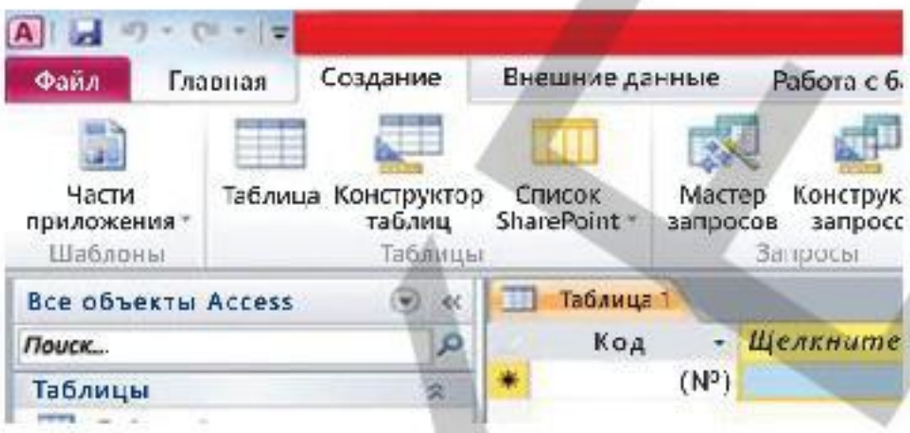


Рис. 37.1. Выбор способа создания таблицы

Создание структуры таблицы в режиме Конструктора

Для создания таблицы в режиме конструктора необходимо:

1. Выполнить команду ***Создание → Конструктор таблиц.*** После чего появится окно конструктора.
2. Набрать имена полей и указать тип данных и свойства (размер ...);
3. В разделе описаний ввести пояснения (по желанию).

В нашем примере таблица *«Диски»* должна содержать четыре поля, для каждого из которых требуется ввести имя, тип данных и в нижней части окна определить свойства поля, как показано на рисунке 37.2.

После ввода полей таблицы, выделите поле ***Код диска*** и в его контекстном меню выберите команду ***Ключевое поле.*** С помощью кнопки ***Сохранить*** на панели быстрого доступа сохраните таблицу под именем «Диски».

Обратите внимание, что активное поле (в нем в данный момент размещен курсор) отмечено слева индикатором — треугольной стрелкой. Свойства активного поля перечислены в нижней части диалога (рис. 37.2).

Типы данных

Тип данных определяется значениями, которые предполагается вводить в поле (столбец), например, текст или число. В Access допускается использование следующих типов данных:

— текстовый;

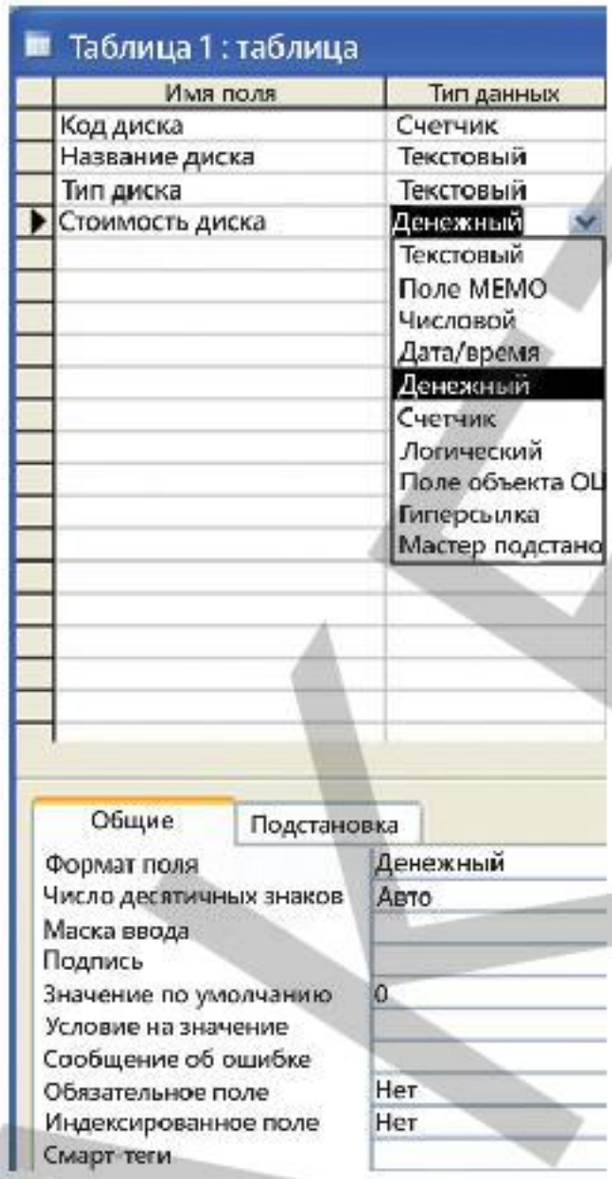


Рис. 37.2. Окно конструктора таблицы

— MEMO (текст большого размера);

— числовой — допускаются целые, дробные, десятичные;

— денежный;

— дата/время;

— счетчик – используется для уникального системного ключа таблицы;

— логический — принимаются два значения «да»—«нет» или «истина»—«ложь».

Ввод и редактирование данных в таблице

После завершения работы по созданию структуры таблицы пользователь может в режиме **Таблицы** приступить к заполнению таблицы данными. Для этого дважды щелкните на нужной таблице в списке таблиц, или в контекстном меню нужной таблицы выберите команду ***Открыть*** (рис. 37.3).

Рис. 37.3. Окно базы данных

Ввод и редактирование данных в таблицу базы данных осуществляется непосредственно в соответствующей ячейке таблицы. Действия по вводу и редактированию аналогичны заполнению и редактированию данных в таблице текстового процессора Word (рис. 37.4).

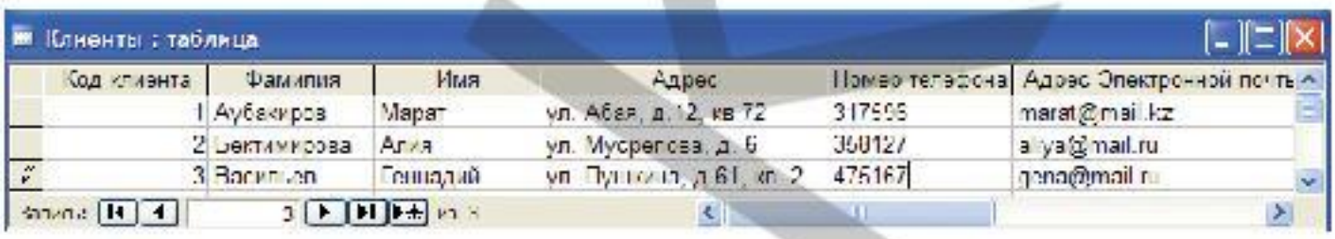


Рис. 37.4. Ввод данных в базу данных «Клиенты»

При вводе и редактировании данных в таблице в поле маркера записи, которое размещается слева от записи таблицы, находятся специальные значки:

▶	Активная запись
*	Пустая запись в конце таблицы
✎	Измененная запись

Если нужно удалить какую-нибудь строку в таблице, щелкните мышью по этой строке в окне конструктора и в контекстном меню нужной строки выберите команду ***Удалить строки***, а для добавления строки, активизируйте поле ниже вставляемой строки и в контекстном меню выберите команду ***Добавить строки***.

При изменении макета таблицы его необходимо сохранять. При вводе данных в режиме таблицы сохранение происходит автоматически.

Связывание таблиц базы данных

Установление связей в Access дает возможность автоматически соединять данные из разных таблиц, таким образом обеспечивается целостность базы данных.

Процесс установки связей между таблицами называют *построением схемы базы данных*.

Для установления связей между двумя таблицами необходимо определить в каждой из них поля для этого связывания. Эти поля не обязательно могут иметь одинаковые имена, но должны содержать однотипные данные.

Пример. Рассмотрим связь между двумя таблицами «Клиенты» и «Выдача дисков» в ранее созданной базе данных «Библиотека дисков». Структура каждой таблицы представлена на рисунке 37.5.

Рис. 37.5. Структура таблиц «Клиенты», «Выдача дисков»

В таблице «Клиенты» ключевым полем является поле **«Код клиента»**. Данное поле является счетчиком и содержит уникальные значения для каждой записи этой таблицы. Поле данной таблицы назовем *первичным ключом*.

Если каждый клиент имеет право выбрать только один диск, то в таблице «Выдача дисков» ключевым полем может являться аналогичное первой таблице поле-счетчик **«Код клиента»**. В таблице «Выдача дисков» ключевое поле **«Код выдачи»** будем называть *внешним ключом*.

В этом случае тип связи, установленной между первичным и внешним ключом, называют «связью «один к одному». Этот тип связи представлен на рисунке 37.6.

Рис. 37.6. Пример связывания таблиц базы данных

В отношении ***«один к одному»*** одной записи в главной таблице соответствует одна запись в связанной таблице.

Если в таблице *«Выдача дисков»* один клиент имеет право сделать несколько заказов, то поле *«Код клиента»* уже не будет уникальным, так как может повторяться многократно. В этом случае тип данных в этом поле может принимать числовые значения, а ключевым полем с уникальными значениями может быть определено поле *«Код выдачи»*. В этом случае тип связи, установленной между одноименными полями *«Код клиента»* в обеих таблицах, называют *связью один ко многим*, как показано на рисунке 37.7.

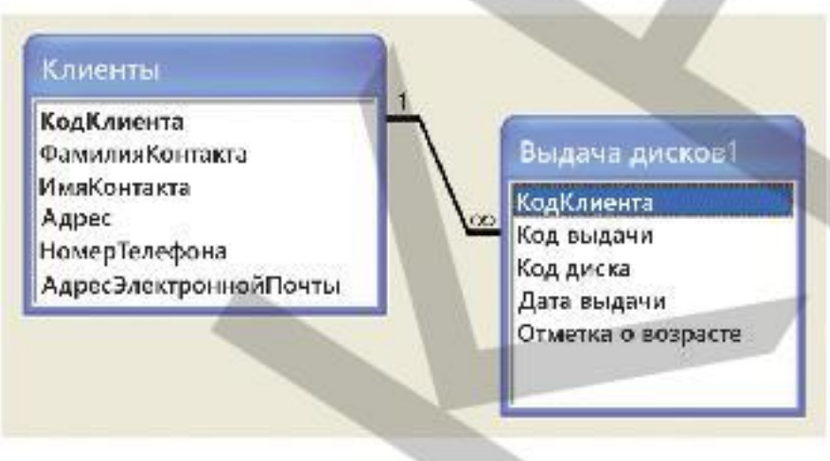


Рис. 37.7. Пример связывания таблиц базы данных

Отношение ***«один ко многим»*** означает, что одной записи таблицы соответствует несколько записей другой таблице.

В реляционных базах данных между таблицами могут также создаваться связи типа ***«многие ко многим»***.

Для создания ***Схемы данных*** в СУБД Access необходимо выполнить следующее:

1. Откройте многотабличную базу данных, для которой между таблицами устанавливаются связи, например, базу данных *«Библиотека дисков»*;

2. Выполните команду ***Работа с базами данных — Схема данных;***

3. Выделите первую таблицу, для которой устанавливается связь в окне ***Добавление таблицы***. Для примера эта таблица — *«Клиенты»;*

4. Щелкните мышью на кнопке ***Добавить***. На экране будет отображена таблица, которую мы добавили, как показано на рисунке 37.8. Затем таким же образом требуется добавить в ***Схему данных*** остальные связываемые таблицы, например *«Выдача дисков»*. Закрыть окно ***Схема данных***;

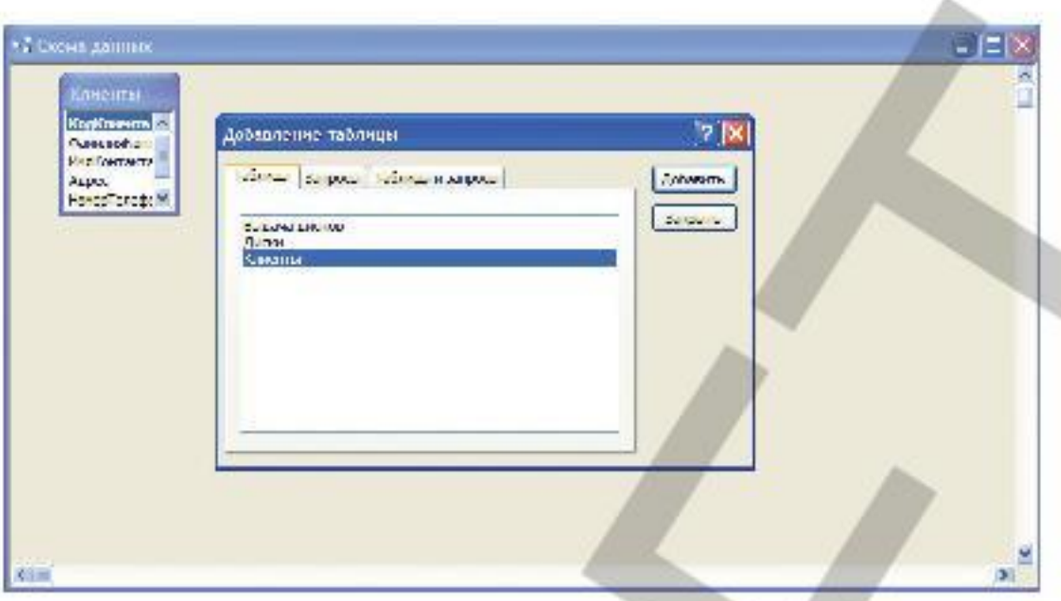


Рис. 37.8. Диалоговое окно «Добавление таблицы»

5. В окне первой таблицы щелкните мышью на поле, которое будете использовать для установки связи, например ***Код клиента***, и перетащите его на совпадающее поле второй таблицы.

В окне ***Изменение связей*** убедитесь, что связаны необходимые поля и установите флажок ***Обеспечение целостности данных***, затем нажмите кнопку ***Создать*** (рис. 37.9).

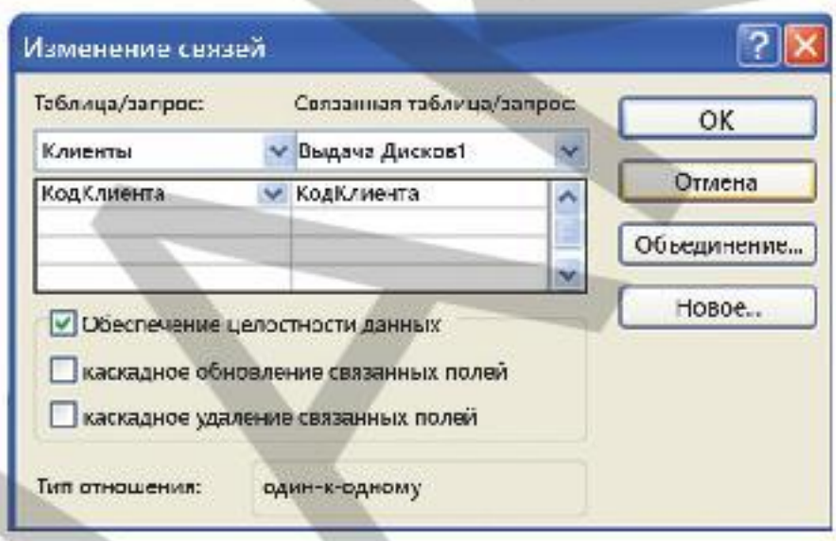


Рис. 37.9. Диалоговое окно «Изменение связей»

В результате выполненных действий в окне ***Схема данных*** два связанных поля соединятся линией, как показано на рисунках 37.8 и 37.9.

В дальнейшем для сохранения созданных связей необходимо щелкнуть мышкой по кнопке ***Сохранить*** и закрыть окно ***Схема данных.***

Созданные связи в дальнейшем можно удалить в окне *Схема данных*, щелкнув мышью по линии связи, чтобы выделить ее, а затем нажать клавишу Delete (Удалить). Двойной щелчок мышью по линии связи приведет к открытию окна ***Изменение связей***, в котором связи могут быть отредактированы.

Контрольные вопросы

1. Назовите этапы проектирования базы данных.
2. Какими принципами нужно руководствоваться при создании таблиц?
3. Как создать структуру таблицы в режиме Мастера?
4. Как используется режим Конструктор для создания структуры таблицы?
5. Что понимается под ключевым полем в таблице? Для чего применяется ключевое поле?
6. Какие типы данных допустимы в СУБД Access?
7. Для чего используется поле Счетчик?
8. Как вводятся и редактируются данные в таблице?
9. Как записи добавляются в таблицу и удаляются из нее?

Практикум

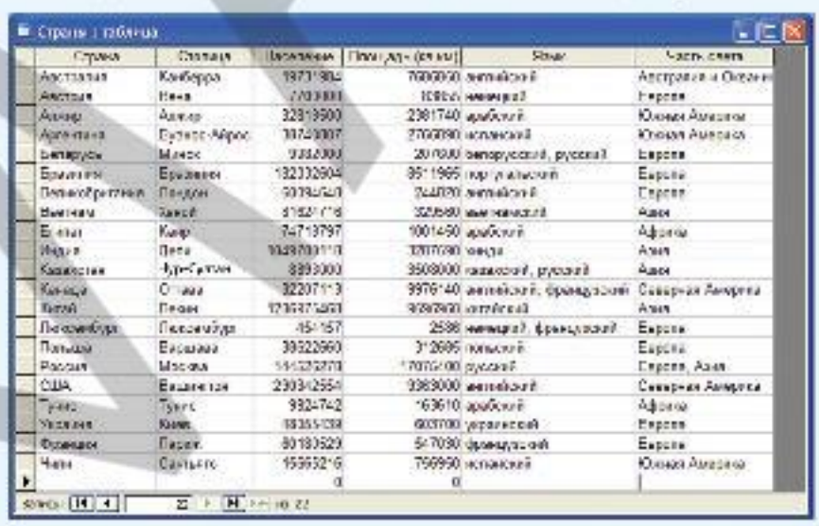
Уровень А

Создание и редактирование однотабличных баз данных.

1. Создайте структуру таблицы базы данных «Ученик», содержащую следующие поля: *код*, *фамилия*, *имя*, *школа*, *класс*, *дата рождения*, *вес*.
2. Определите первичный ключ таблицы.
3. Добавьте в структуру после поля «*дата рождения*» поле «*рост*».

Уровень В

Создайте базу данных «Страны» по образцу (рис. 37.13).



Страна	Столица	Население	Площадь (кв.км)	Язык	Часть света
Австралия	Канберра	19731984	7686850	английский	Австралия и Океания
Австрия	Вена	7703881	83850	немецкий	Европа
Алжир	Алжир	32818500	2381740	арабский	Южная Америка
Аргентина	Буэнос-Айрес	36743007	2766890	испанский	Южная Америка
Беларусь	Минск	9358000	207600	белорусский, русский	Европа
Бразилия	Бразилиа	192332824	8511965	португальский	Европа
Великобритания	Лондон	59778500	244820	английский	Европа
Вьетнам	Ханой	81624716	329560	вьетнамский	Азия
Египет	Каир	74719797	1001450	арабский	Африка
Индия	Дели	1029991015	3287590	хинди	Азия
Казахстан	Нур-Султан	8399300	3568000	казахский, русский	Азия
Канада	Оттава	32207113	9976140	английский, французский	Северная Америка
Китай	Пекин	1216473000	9596960	китайский	Азия
Люксембург	Люксембург	454157	2586	немецкий, французский	Европа
Польша	Варшава	38522960	312685	польский	Европа
Россия	Москва	146393773	17075400	русский	Европа, Азия
США	Вашингтон	239314554	9363000	английский	Северная Америка
Тунис	Тунис	9924742	163610	арабский	Африка
Украина	Киев	48355139	603700	украинский	Европа
Франция	Париж	60183529	547030	французский	Европа
Чили	Сантьяго	15595216	756950	испанский	Южная Америка
		0	0		

Рис. 37.13. База данных «Страны»

Уровень С

Создайте базу данных «Библиотека дисков», содержащую три таблицы «Клиенты», «Диски», «Выдача дисков» (см. таблицы 37.2, 37.3, 37.4).

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§39–41 Разработка базы данных. Проектная работа

Вы научитесь:

- определять типы данных в базе данных;
- создавать однотабличную базу данных;
- создавать многотабличную базу данных.

Ключевые понятия:

- база данных
- реляционная база данных
- поле
- запись
- индекс
- первичный ключ

Человечество из века в век стремилось накопить и сохранить информацию. Яркими примерами древности являются узелковые письма племени майя, наскальные рисунки, рецепты, передаваемые от отца к сыну и т.д. С развитием цивилизации объемы информации увеличились многократно, а значит, возросла потребность накопления, хранения и преобразования информации, наряду с этим появилась необходимость ее сортировки. Повсеместно для этих целей используется компьютерные базы данных, поэтому, выйдя за стены школы, вы должны уметь искать необходимую информацию в базе данных и сортировать ее по определенным условиям. Одним из программных продуктов работы с информацией является Microsoft Access.

Работа над проектом будет осуществляться в группах по 4—5 человек. Вам на выбор предлагается несколько тем для создания базы данных: «Школа», «Мой класс», «Больница», «Отдел кадров» и др. Задача — разработать компьютерную версию базы данных.

Каждый участник группы выполняет свою функцию.

Координатор — хранитель времени — осуществляет взаимодействие между всеми членами группы, следит за тем, чтобы каждый исполнял

свои обязанности и отвечает за своевременность выполнения группой задания.

Редактор — анализатор редактирует и анализирует всю информацию, которую выдает группа.

Исполнитель осуществляет поиск информации, необходимой для решения задачи, поставленной перед группой, и решает эту задачу.

Секретарь — исполнитель делает записи за всю группу и помогает решать поставленную задачу.

Оратор выступает перед классом, предлагая выполненное группой задание.

На всю работу отводится 3 урока (1 урок — знакомство с работой, разбиение на группы, распределение обязанностей в группе, планирование работы; 2 урок — работа по созданию проекта; 3 урок — защита проекта). За это время необходимо:

- повторить этапы разработки базы данных;
- поэтапно разработать презентацию по своей теме;
- создать ее на компьютере;
- рассмотреть возможности поиска и сортировки информации;
- продумать форму защиты проекта.

По завершении работы вы должны презентовать свою базу данных. Это можно сделать любым удобным для вас способом: показать свою базу данных и рассказать о правилах работы в ней. В ходе представления работы обязательно должны рассказать, как выполняли работу, какие идеи возникали и почему от некоторых отказались. Параметры оценки защиты проекта:

- достигнутый результат (из 15 баллов);
- оформление (из 15 баллов);
- представление (из 15 баллов);
- ответы на вопросы (из 15 баллов);
- интеллектуальная активность (из 10 баллов);
- творчество (из 10 баллов);
- практическая деятельность (из 10 баллов);
- умение работать в команде (из 10 баллов).

Рефлексия:

Какая информация вас особенно заинтересовала?
Какие возникли трудности и с кем вы готовы их обсудить?
Какие навыки вы готовы применять уже сейчас?

§42–43 Формы

Вы научитесь:

- создавать форму для вывода данных.

Ключевые понятия:

- базы данных
- структура таблицы
- типы данных
- создание формы

Для организации диалогового интерфейса пользователя, удобства ввода данных в таблицу, а также в случае их редактирования удобно пользоваться *формой*.

Форма — объект базы данных, предназначенный для ввода и отображения информации.

Создать форму можно несколькими способами: с помощью ***Мастера форм, автоформы*** (команда *Форма* на вкладке *Создание*) и *Конструктора форм*.

Рассмотрим создание формы с помощью ***Мастера форм.***

На вкладке *Создание* выбрать команду ***Мастер форм.***

В открывшемся диалоге *Создание формы* (рис. 42.1) и в раскрывающемся списке выберите таблицу, для которой будет создаваться форма.

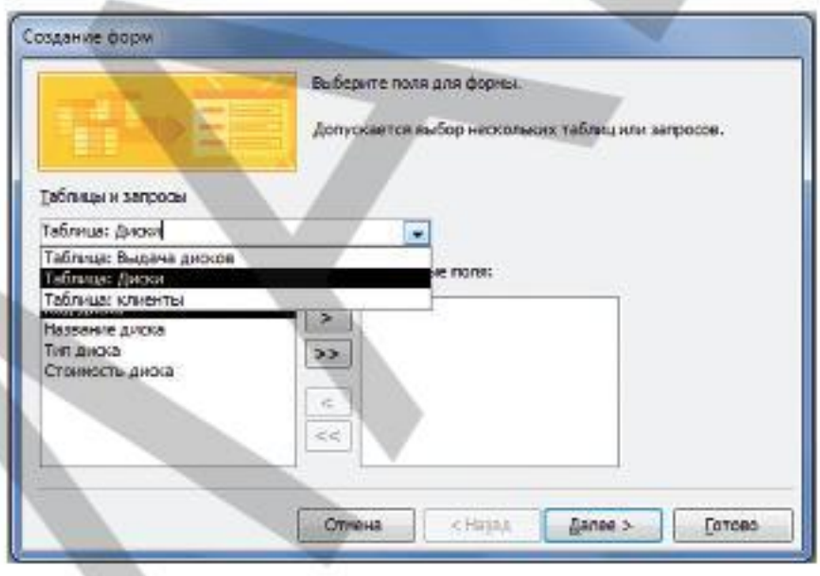


Рис. 42.1. Диалоговое окно «Новая форма»

Следующий шаг: укажите поля, которые будут присутствовать в форме. В списке ***Доступные поля*** для переноса выделенного поля в форму щелкните по кнопке с символом «>». Закончив перенос, нажмите на кнопку *Далее*.

Следующей диалог мастера предназначен для выбора вида формы. По умолчанию предлагается форма, в которой поля ввода размещаются в столбец. Если согласны, то нажмите по кнопке *Далее* (рис. 42.2).

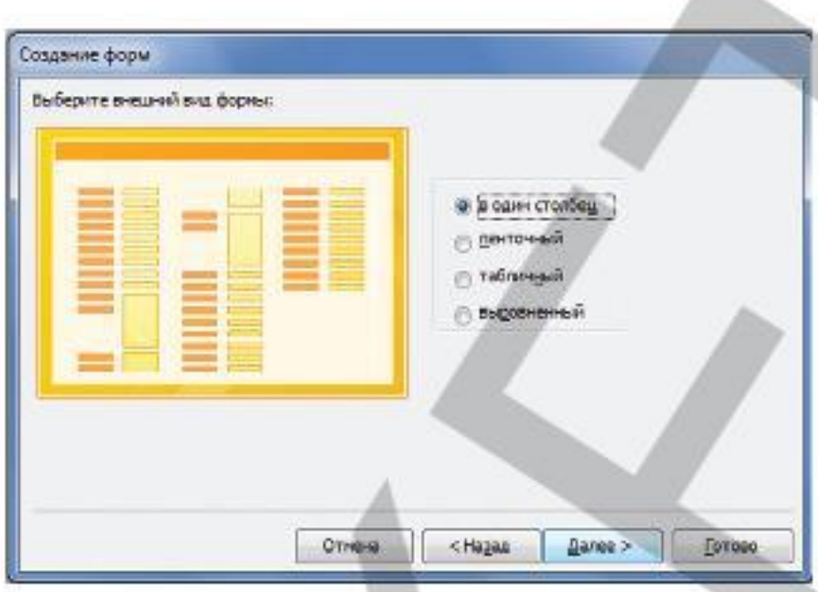


Рис. 42.2. Выбор способа отображения данных

На предложение ввести имя формы введите имя и щелкните мышью *Готово*. Появляется следующая форма, показанная на рисунке 42.3.

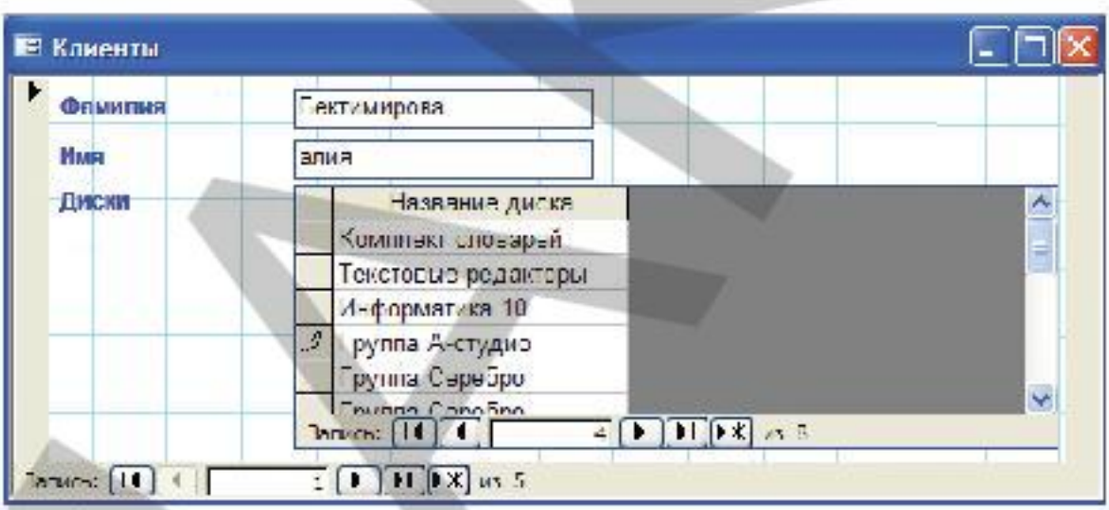


Рис. 42.3. Вид созданной формы

Большая часть данных, входящих в форму, берется из таблицы или запроса.

Щелкая мышью на кнопке ⏮ ⏭, можно вернуться к предыдущей или следующей записи.

Использование фильтров

При работе с большими таблицами базы данных обычно на экране компьютера отображают не все записи, а только группы записей, отобранные по определенным критериям.

Для отбора записей таблицы по определенным критериям в программе Access используются ***фильтры***.

Рассмотрим применение фильтров на конкретной однотабличной учебной базе данных *«Золотые рыбки»*. Таблица этой базы представлена на рисунке 42.4.

Рис. 42.4. База данных «Золотые рыбки»

Программа Access предлагает пользователю различные способы фильтрации: ***по выделенным данным, по исключению выделенных данных, по заданному условию.***

Выполнение фильтрации по выделенным данным рассмотрим на примере 1.

Пример 1. Отобрать записи таблицы, которые содержат сведения о рыбках, место происхождения которых Мексика. Для этого выполним следующие действия:

1. Щелкните мышью на любой ячейке столбца *Место происхождения* со значением *«Мексика»;*

2. Щелкните мышкой по кнопке ***Выделение*** на вкладке ***Главная.***

В результате будут отобраны все записи, имеющие в поле *Место происхождения* значение *«Мексика»*, как показано на рисунке 42.5.

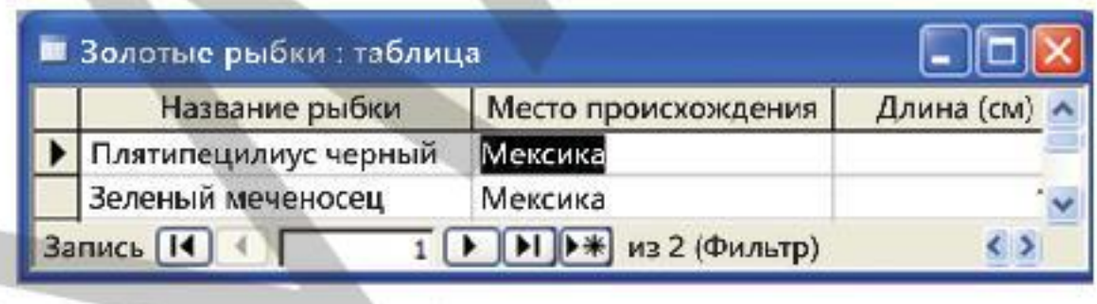


Рис. 42.5. Результат фильтрации

При сохранении таблицы последний установленный фильтр запоминается программой Access и потом может быть снова применен.

Для отключения установленного фильтра можно воспользоваться кнопкой ***Фильтр*** на вкладке ***Главная → Сортировка и Фильтр.*** Повторное нажатие этой кнопки приводит к применению к таблице последнего фильтра.

Фильтрацию по исключению выделенных данных продемонстрируем на примере 2.

Пример 2. Отобрать записи таблицы, которые не содержат сведения о рыбках, место происхождения которых Мексика.

Для этого выполним следующие действия ***Главная → Сортировка и фильтр → Выделение → Не содержит «Мексика».***

В результате будут отобраны все записи, не имеющие в поле *Место происхождения* значение *«Мексика»*, как показано на рисунке 42.6.

Золотые рыбки : таблица

Название рыбки	Место происхождения	Длина (см)
Гамбузия	Южная Америка	
Хаплохилусы	Африка	
Тетра-фон-рио	Южная Америка	
Бархатный гирардинус	Южная Америка	
Макропод	Китай	
Гурами жемчужный	Индия	
Лялиус	Индия	
Неон	Южная Америка	

Запись: 1 из 8 (Фильтр)

Рис. 42.6. Результат фильтрации

Фильтрацию по заданному условию удобно задавать после щелчка мыши по кнопке ***Изменить фильтр*** ***Главная → Сортировка и фильтр → Дополнительно → Изменить фильтр.*** В итоге откроется окно ***Фильтр***, в котором задаются условия.

Пример 3. Отобрать только те записи таблицы, которые содержат сведения о рыбках, место происхождения которых Южная Америка и длина 5 см.

С помощью выпадающих меню в окне *Фильтр* зададим условие *«Длина* (см) = 5 и *Место происхождения = Южная Америка»* (рис. 42.7).

Золотые рыбки : фильтр

Название рыбки	Место происхождения	Длина (см)
	"Южная Америка"	5

Найти | Или

Рис. 42.7. Диалоговое окно фильтрации по заданному условию

После применения фильтра получим таблицу из двух записей, как показано на рисунке 42.8.

Рис. 42.8. Результат фильтрации

Таким образом, форма позволяет не только получить доступ к данным, но и оформить их в виде, удобном для пользователя.

Контрольные вопросы

1. Что такое *формы* и для чего они предназначены?
2. Что служит источником данных для форм?
3. Как изменить фон формы?
4. Как изменить источник данных форм?

Практикум

Уровень А

Для своей базы данных, созданной на прошлых занятиях, постройте форму для просмотра и ввода данных в таблицу.

Уровень В

Постройте форму и оформите ее так, чтобы работа с данными была как можно проще и понятнее.

Уровень С

Проведите исследовательскую работу «Как можно добавить на форму растровые и векторные рисунки?». Попробуйте построить форму, в которой используется графика.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§44–45 Отчеты

Вы научитесь:

- создавать отчеты, используя извлеченные данные.

Ключевые понятия:

- базы данных
- структура таблицы
- типы данных
- создание отчета

Чтобы предоставить в привычном виде данные, которые были собраны в базе, нужно сформировать отчет.

Отчет — это документ, предназначенный для вывода данных на печать.

В данном параграфе рассмотрим ***Мастер отчетов***, ***Конструктор*** и ***Мастер диаграмм***.

Самый простой способ создать отчет, используя Мастер отчета. Построим отчет нашей базы данных и распечатаем на принтере. Для этого:

1. Откройте базу данных. Перейдите на вкладку ***Отчеты***.

2. Щелкните по кнопке ***Создать*** и в открывшемся диалоге *Новый отчет* выберите позицию ***Мастер отчетов***.

3. В раскрывшемся списке укажите вашу таблицу, на основе которой будет создаваться отчет. Щелкните по кнопке *ОК*.

4. В следующем диалоге *Создание отчетов* переместите все поля из списка ***Доступные поля*** в список ***Выбранные поля***, после чего нажмите кнопку ***Далее***.

5. Последующий диалог предназначен для задания уровней группировки для полей отчета. Можете принять установки по умолчанию. Нажмите ***Далее***.

6. Следуйте указаниям мастера, задавая в последующих диалогах порядок сортировки полей, вид макета, стиль оформления отчета, а также имя отчета. Нажмите по кнопке *Готово*.

В результате мы получим отчет, который будет выведен в окно просмотра. В данном окне можно отрегулировать расположение надписей и их формат. Для этого нужно переключиться с помощью кнопки *Вид* в режим конструктора отчетов. Эти операции, выполняемые в отдельном окне конструктора отчетов, просты и интуитивно понятны. Поэтому предлагаем их вам самостоятельно проработать.

Созданный отчет можно распечатать, для этого нужно нажать на кнопку ***Печать*** на панели инструментов.

Контрольные вопросы

1. Для чего предназначен отчет?
2. Через какие действия формируется отчет?

Практикум

Уровень А

Составьте алгоритм формирования отчета в MS Access.

Уровень В

Постройте отчет по своей базе данных и распечатайте его.

Уровень С

В режиме мастера отчетов создайте отчет по двум таблицам.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§46–47 Запросы

Вы научитесь:

- создавать запросы на выборку с помощью конструктора;
- создавать простые и перекрестные запросы;
- задавать условия отбора;
- создавать запросы с вычислением.

Ключевые понятия:

- базы данных
- структура таблицы
- форма
- запрос
- простой запрос
- перекрестный запрос
- создание запроса с вычислением

Поиск информации в базах данных выполняется с помощью запросов.

Запрос — это обращение к СУБД для отбора записей или выполнения других операций.

С помощью ***запроса*** программа Access выбирает и отображает наборы записей из таблиц базы данных, которые отвечают заданным условиям. Запрос может формироваться на основе одной или нескольких связанных таблиц или запросов, построенных ранее.

В большинстве современных СУБД для управления данными (т.е. для составления запросов) используется язык SQL (англ. *Structured Query Language* — «язык структурных запросов»).

Программа Access поддерживает различные типы запросов: *запрос на выборку данных, запрос на добавление и удаление данных, запрос на обновление данных.*

В дальнейшем мы рассмотрим только различные виды запросов на выборку данных.

Запрос на выборку данных по одной таблице

Программа Access поддерживает создание запросов на выборку с помощью *Мастера и Конструктора*. Мастер обеспечивает возможность создания простых запросов.

Рассмотрим создание запросов на основе таблицы *«Золотые рыбки»*, которая представлена на рисунке 46.1.

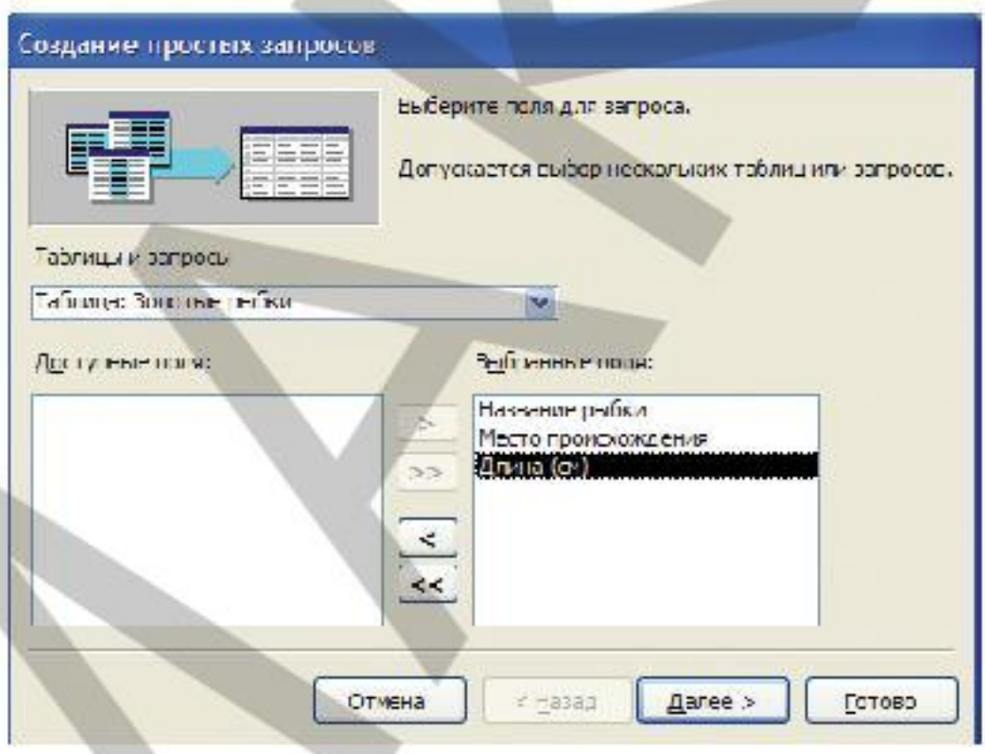


Рис. 46.1. Диалоговое окно «Создание простых запросов»

Пример 1. Создать простой запрос с помощью *Мастера*, который покажет сведения о рыбках, а именно название рыбки, место происхождения и длину.

Для создания запроса необходимо выполнить следующее:

1. Вкладка ***Создание — Мастер запросов.***

2. В открывшемся диалоговом окне выберите ***Простой запрос.***

3. После открытия окна ***Создание просых запросов*** укажите таблицу для создания запроса, например *Золотые рыбки* и из списка ***Доступные поля*** с помощью стрелок перенесите нужные поля *Название рыбки, Место происхождения* и *Длина* в список ***Выбранные поля***, как представлено на рисунке 46.1.

После нажатия кнопки ***Готово*** откроется сформированный запрос, содержащий три выбранных столбца (рис. 46.2).

Рис. 46.2. Результат запроса на выборку данных

Аналогичный ***Запрос*** по таблице «Золотые рыбки» может быть создан в режиме ***Конструктора.*** В этом случае говорят, что запрос строится «с нуля». Для этого следует выполнить следующее:

1. Выполните цепочку ***Создание → Конструктор запросов.***

2. В окне ***Добавление таблицы*** выберите таблицу с именем *Золотые рыбки* и щелкните по кнопке ***Добавить.***

3. В результате выполненных действий откроется окно ***Запрос на выборку Конструктора***, как показано на рисунке 46.3. В этом окне

Рис. 46.3. Создание запроса в режиме Конструктора

последовательно двигаясь слева направо щелкните мышью на стрелке в строке ***Поле*** и из открывающегося списка выберите поля, которые хотите добавить в запрос: *Название рыбки, Место происхождения, Длина.*

Установка флажка в каждом отобранном столбце строки ***Вывод на экран*** позволит вывести нужные столбцы. После завершения конструирования запроса его необходимо сохранить.

Результатом выполнения запроса становится новая временная таблица.

Рассмотрим несколько примеров создания составных запросов в окне *Конструктора*.

Пример 2. Создать запрос, который покажет сведения о рыбках, место происхождения которых Индия.

В окне *Конструктора* запроса в строке ***Условие отбора*** необходимо ввести значение «Индия» в поле *Место происхождения*, как показано на рисунке 46.4.

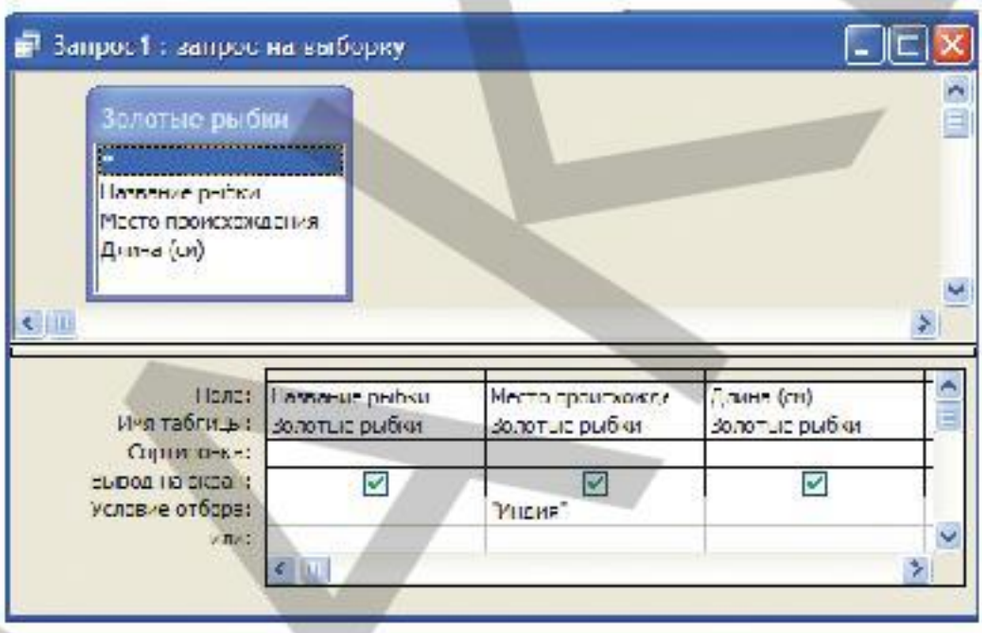


Рис. 46.4. Составной запрос в режиме Конструктора

В итоге получим следующий результат с таблицей, представленной на рисунке 46.5.

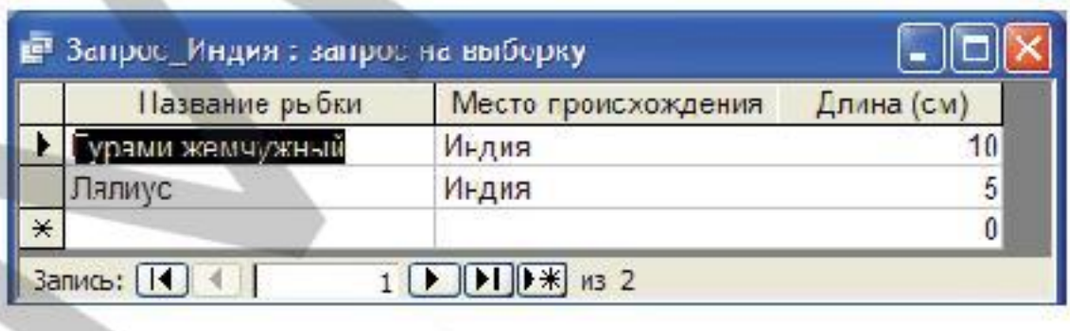


Рис. 46.5. Результат запроса

Пример 3. Создать запрос, который покажет сведения о рыбках, место происхождения которых Индия и длина больше 5 см.

В окне *Конструктора запроса* в строке ***Условие отбора*** дополнительно к запросу примера 2 необходимо ввести значение «>5» в поле *Длина (см)*, как показано на рисунке 46.6. В результате получим таблицу, показанную на рисунке 46.6.

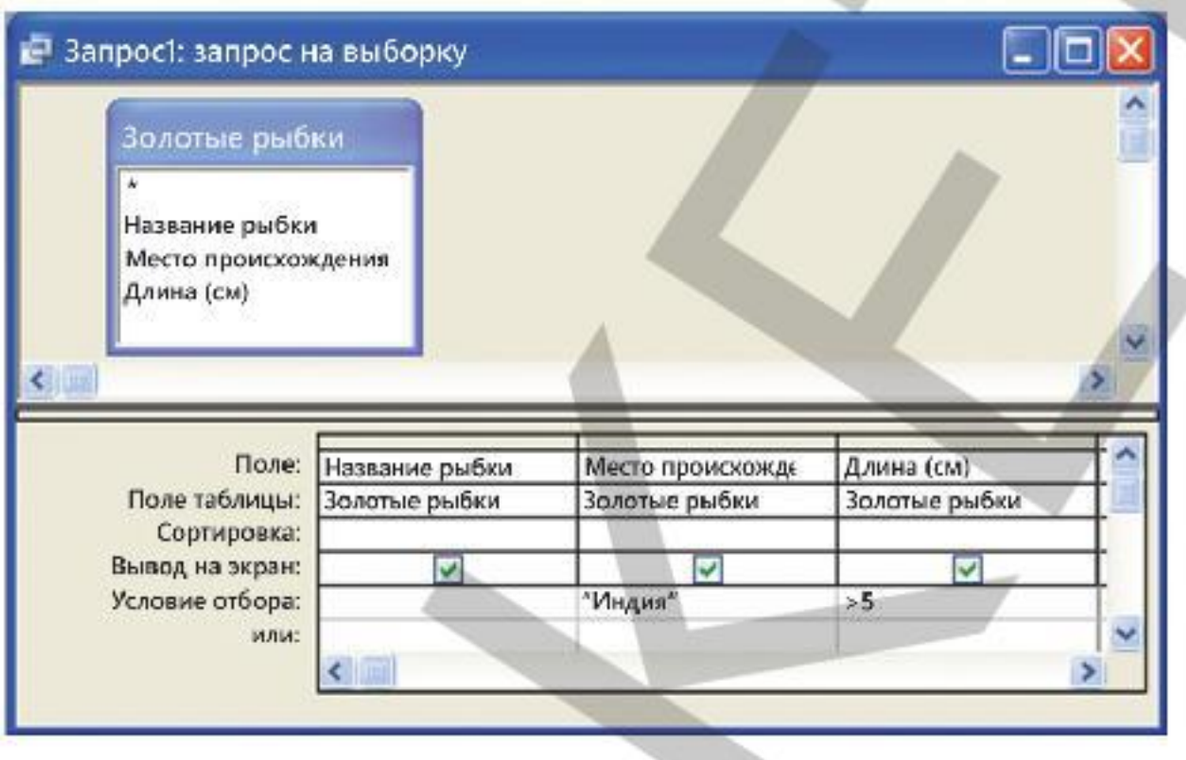


Рис. 46.6. Составной запрос в режиме Конструктора

Структурированный запрос (SQL)

SQL = Structured Query Language — язык структурных запросов для управления данными.

Любой запрос на получение определенных данных из одной или нескольких таблиц выполняется с помощью предложения SELECT. Результатом предложения SELECT является другая таблица.

Для нашей таблицы «Золотые рыбки» он будет выглядеть так (рис. 46.7):

SELECT «Название рыбки», «Место происхождения», «Длина» FROM «Золотые рыбки».

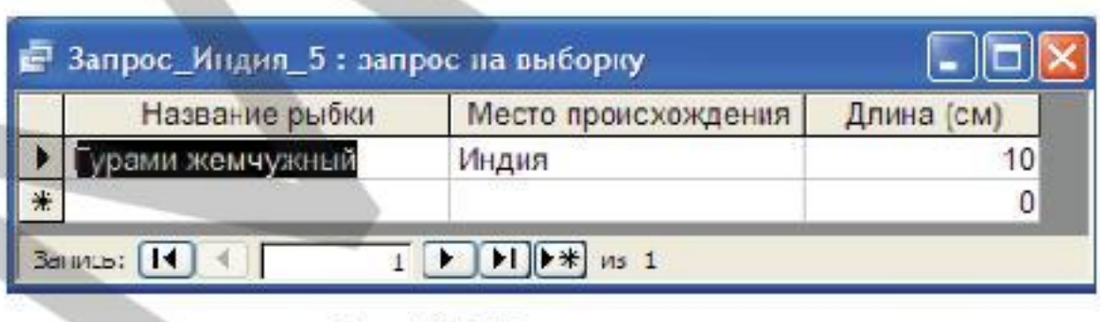


Рис. 46.7. Результат запроса

Контрольные вопросы

1. Для чего используются запросы?
2. Какие типы запросов поддерживает программа Access?
3. Как создается простой запрос на выборку с помощью Мастера?
4. Как строится простой запрос с помощью Конструктора?
5. Какие возможности предоставляет пользователю Конструктор для создания запроса на выборку с условиями?
6. Для чего используются фильтры?
7. Какие операции фильтрации предлагает пользователю программа Access?
8. Что означает фильтрация по выделенным данным?
9. Как выполняется фильтрация по исключению выделенных данных?

Практикум

Уровень А

Откройте ранее созданную базу данных и выполните отбор нужных записей таблицы по выделенным данным.

Уровень В

Откройте ранее созданную базу данных и выполните отбор нужных записей таблицы по исключению выделенных данных и по заданному условию.

Уровень С

Откройте ранее созданную базу данных «Страны», и создайте следующие запросы:

а) простой запрос, содержащий все записи полей *Страна, Столица, Население, Язык;*

б) запрос, который выводит данные о странах с населением меньше 10 млн. человек;

в) запрос, который выводит столицы стран, где говорят на английском языке.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§48–49 Структурированные запросы

Вы научитесь:

- использовать структурированный язык запросов.

Ключевые понятия:

- язык SQL

Язык SQL предназначен для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление).

SQL = *Structured Query Language* — язык структурных запросов для управления данными

У того, кто работает с базами данных довольно часто, возникает вопрос, как выбрать ту или иную информацию из таблицы.

Любой запрос на получение определенных данных из одной или нескольких таблиц выполняется с помощью предложения SELECT. Результатом предложения SELECT является другая таблица.

Простейший запрос для выборки данных из одной таблицы выглядит так:

```
SELECT список полей FROM имя таблицы
```

Пример 1. Для нашей таблицы «Золотые рыбки» из предыдущего параграфа запрос будет выглядеть так:

```
SELECT Название рыбки, Место происхождения, Длина FROM Золотые рыбки.
```

Данный запрос выбирает название рыбки, место происхождения, длину из таблицы «Золотые рыбки».

Важно. SQL игнорирует лишние пробелы и команды перехода на новую строку в тексте и поэтому этот запрос можно записывать в несколько строк.

Наш запрос будет выглядеть следующим образом:

```
SELECT Название рыбки, Место происхождения, Длина
FROM Золотые рыбки
```

Пример 2. Рассмотрим следующий пример:

```
SELECT shortname FROM Золотые рыбки
```

Данный запрос выбирает из таблицы «Золотые рыбки» только краткие названия рыб.

Правила:

1. После SELECT можно указать одно из полей таблицы, несколько полей или все поля, которые имеются в таблице. Можно выбирать только часть полей, имеющихся в таблице базы данных. Это уменьшает

объем информации, которая передается по сети, и обеспечивает большую скорость при работе с удаленными базами данных.

2. Поля могут перечисляться не в том порядке, в котором они перечислены в таблице.

Чем отличаются следующие два запроса в примере 3?

Пример 3.

```
SELECT Название рыбки, Место происхождения, Длина FROM Золотые рыбки
SELECT Длина, Место происхождения, Название рыбки FROM Золотые рыбки
```

Отличаются только тем, что в выходных данных будет другой порядок следования полей.

Вместо списка полей можно указать символ *, который означает, что в выходных данных следует вывести все поля, которые имеются в таблице. При этом порядок следования полей будет тот же, что и в таблице.

Это удобно использовать, если мы не знаем, какие поля есть в таблице.

Пример 4.

```
SELECT * FROM Золотые рыбки
```

Данный запрос выведет все поля из таблицы «Золотые рыбки». Фактически он выведет всю информацию, которая имеется в этой таблице.

Контрольные вопросы

1. Что такое *запрос*? Зачем используются запросы?
2. На каком языке составляются запросы к СУБД?
3. Как изменить существующий запрос?

Практикум

Уровень А

Проведите исследовательскую работу. Какие операции, кроме выборки данных, можно выполнить с помощью SQL-запросов?

Уровень В

Создайте запрос, который отбирает данные о всех рыбках, в названии которых есть буква «а».

Уровень С

Создайте запрос, который отбирает данные о всех рыбках, в названии которых третья буква «а».

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§50–52 Структурированные запросы. Проектная работа

Вы научитесь:

- использовать структурированный язык запросов;
- создавать запросы на выборку с помощью Конструктора;
- создавать простые и перекрестные запросы;
- задавать условия отбора;
- создавать запросы с вычислением.

Ключевые понятия:

- базы данных
- структура таблицы
- форма
- запрос
- простой запрос
- перекрестный запрос
- создание запроса с вычислением

Задание 1. Строительная организация получает строительные изделия и материалы от нескольких поставщиков. В базе данных должны содержаться сведения о:

а) поставщиках (данными о поставщике являются его номер, индекс, наименование, адрес (табл. 50.1));

б) получаемых изделиях (данными об изделии являются его шифр, наименование, единица измерения, количество, поставщик (табл. 50.2)).

Таблица 50.1

№ поставщика	Наименование	Адрес	Кол-во сотрудников
2368	ООО «Гранит»	Ул. Абая, 30	30
3820	ЧП «Аубакиров»	Пр. Гагарина, 26	52
5024	ЗАО «Альфа»	Ул. Астана, 12	108
3015	АО «СТРОЙ»	Ул. Жумабаева, 3	24
9543	ТОО «Проект»	Ул. Достык, 9	185

Таблица 50.2

Изделие

Шифр	Наименование	Ед. изм.	Кол-во	Поставщик
1238	Шлакоблок	шт	50	9543
1237	Цемент	т	18	3820
1247	Стеновая панель	шт	60	5024
7421	Гипсокартон	шт	320	2368
1241	Дверной блок	шт	280	3820
5421	Оконный блок	шт	82	1237
3248	Плита перекрытия	шт	240	3820

1. Напишите запрос, который сокращает количество сотрудников у всех поставщиков на 2.

2. Напишите запрос, увеличивающий количество изделий поставщика ЧП «Аубакиров» на 5 шт.

3. Напишите запрос, который выводит *Наименование*, *Адрес* и *Количество сотрудников* из таблицы *Поставщик*.

4. Напишите запрос, который вывел бы список всех изделий, поставляемых ЗАО «Альфа».

5. Напишите запрос, который вывел бы таблицу *Изделие* со столбцами в обратном порядке.

6. Напишите запрос, извлекающий из таблицы *Изделие* список номеров поставщиков. Номера не должны повторяться.

7. Напишите запрос, выводящий наименование и адрес поставщика, где работает минимальное количество сотрудников.

8. Напишите запрос на создание списка, состоящего из *Наименования изделия* и его *Поставщика* для всех поставщиков, количество сотрудников которых не менее 100.

9. Напишите запрос на удаление всех изделий, поставляемых поставщиком №2450.

Задание 2. Завод-изготовитель поставляет нескольким получателям строительные изделия и конструкции. В базе данных должны содержаться сведения о:

а) получателях (данными о получателе являются его код, наименование, адрес, удаленность от завода (табл. 50.3));

б) поставках (данными о поставке являются ее шифр, наименование изделия, единица измерения, цена единицы измерения, получатель (табл. 50.4)).

Таблица 50.3

№ поставщика	Наименование	Адрес	Кол-во сотрудников
5241	ООО «Гранит»	Ул. Абая, 30	30
3820	ЧП «Аубакиров»	Пр. Гагарина, 26	52
2450	ЗАО «Альфа»	Ул. Астана, 12	108
3054	АО «СТРОЙ»	Ул. Жумабаева, 3	24
1568	ТОО «Проект»	Ул. Достык, 9	185

Таблица 50.4

Шифр	Изделие	Ед. изм.	Цена	Получатель
1238	Шлакоблок	шт	5000	3054
1237	Цемент	т	900	3820
1247	Стеновая панель	шт	6000	2450
7421	Гипсокартон	шт	1020	1568
1241	Дверной блок	шт	28000	1568
5421	Оконный блок	шт	44000	3054
3248	Плита перекрытия	шт	240	3820
5421	Гвозди	кг	70	2450

1. Напишите запрос, который увеличивает *Цену* всех поставок на 1000 тенге.

2. Напишите запрос, передающий поставки от ЗАО «Альфа» в ТОО «Проект».

3. Напишите запрос, который выводит *Наименование*, *Удаленность* и *Адрес* из таблицы *Получатель*.

4. Напишите запрос, который вывел бы всю информацию о поставке ЧП «Аубакиров».

5. Напишите запрос, который вывел бы таблицу *Поставка* со столбцами в обратном порядке.

6. Напишите запрос, извлекающий из таблицы *Поставка* список получателей. Получатели не должны повторяться.

7. Напишите запрос, считающий среднюю цену поставок.

8. Напишите запрос на создание списка, состоящего из названия *Изделия* и *Наименования* его получателя для всех получателей, которые расположены далее 70 км от завода.

9. Напишите запрос на удаление всех поставок получателя с кодом 2450.

Задание 3. Строительное подразделение ведет работу на нескольких объектах. В базе данных должны содержаться сведения:

а) об объектах (данными об объекте являются его номер, наименование, сметная стоимость работ (млн. тенге), процент выполнения работ (табл. 50.5));

б) о поставках ресурсов (данными о поставке ресурсов являются код поставки, наименование ресурса, единица измерения, количество, объект (табл. 50.6)).

Таблица 50.5

Объект

№ объекта	Наименование	Стоимость	Выполнение
1	Поликлиника	150	70
2	Школа	82	60
3	Жилой дом по пр. Абая	140	90
4	Торговый центр	350	20
5	Дет. сад	650	50

Таблица 50.6

Поставка ресурсов

Код	Ресурс	Ед. изм	Количество	Объект
3258	Цемент	кг	680	1
4342	Краска	кг	350	4
5428	Шпатлевка	кг	260	3
5321	Кирпич	$м^3$	36	2
4418	Песок	т	9	4
3021	Известь	т	4	5
5152	Краска	л	120	2

1. Напишите запрос, который увеличивает выполнение по всем объектам на 1%.

2. Напишите запрос, переводящий ресурсы, предназначенные для объекта «Жилой дом по пр. Абая» на объект «Школа».

3. Напишите запрос, который выводит *Наименование*, *Стоимость*, *Выполнение* из таблицы *Объект*.

4. Напишите запрос, который вывел бы список всех поставок ресурсов для объекта *Поликлиника*.

5. Напишите запрос, который вывел бы таблицу *Поставка ресурсов* со столбцами в обратном порядке.

6. Напишите запрос, извлекающий из таблицы *Поставка ресурсов* список объектов. Объекты не должны повторяться.

7. Напишите запрос, выводящий наименование объекта и процент выполнения для объекта, имеющего самую высокую сметную стоимость.

8. Напишите запрос на создание списка, состоящего из *Наименования ресурса* и *Наименования объекта*, где он используется для всех объектов, выполнение по которым не более 60%.

9. Напишите запрос на удаление всех записей из таблицы *Поставка ресурсов*, предназначенных для объекта №1.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

Проверь себя!

1. Определите тип связи между таблицами «Учителя» и «Учащиеся», если один преподаватель обучает разных учеников:
   а) «один-к-одному»; в) «многие-к-одному»;
   б) «один-ко-многим»; г) «многие-ко-многим».
2. Какой из объектов служит для хранения данных в БД:
   а) отчет; в) запрос;
   б) форма; г) таблица?
3. Программа или комплекс программ для администрирования и мониторинга данных называется:
   а) Средой Управления Базами Данных;
   б) Системой Умножения Баз Данных;
   в) Банк данных.
4. Определите связь между таблицами «Город» и «Район», если каждому городу соответствует несколько районов:
   а) «один-к-одному»; в) «многие-к-одному»;
   б) «один-ко-многим»; г) «многие-ко-многим».
5. Тип поля реляционной базы данных определяется:
   а) именем поля; в) именем ячейки;
   б) типом данных; г) типом ключа.
6. Иерархическая база данных — это:
   а) БД, в которой информация организована в виде прямоугольных таблиц;

б) БД, в которой элементы в записи упорядочены, т. е. один элемент считается главным, остальные подчиненными;
в) БД, в которой записи расположены в произвольном порядке;
г) БД, в которой существует возможность устанавливать горизонтальные связи дополнительно к вертикальным иерархическим связям.

7. Формы используются для:
а) ввода данных;
б) выборки данных;
в) вывода данных на печать;
г) просмотра данных.

8. Как исключить наличие повторяющихся записей в таблице:
а) упорядочить строки таблицы;
б) определить ключевое поле;
в) определить внешний ключ;
г) проиндексировать поля таблицы?

9. Столбец однотипных данных в Access называется:
а) записью;
б) запросом;
в) отчетом;
г) полем.

10. Для выборки записей и обновления данных из одной или нескольких таблиц базы данных служат:
а) отчеты;
б) формы;
в) запросы;
г) таблицы.

11. База данных — это:
а) пакет пользовательских программ;
б) совокупность файлов на жестком диске;
в) совокупность сведений, характеризующих объекты, процессы или явления реального мира;
г) система управления базами данных.

12. СУБД — это:
а) система средств администрирования банка данных;
б) система средств архивирования и резервного копирования банка данных;
в) специальный программный комплекс для обеспечения доступа к данным и управления ими;
г) система средств управления транзакциями.

13. Типы данных полей таблицы MS Access (уберите лишнее):
а) логический;
б) счетчик;
в) общий;
г) текстовый.

14. Структура реляционной базы данных (БД) меняется при удалении:
а) одного из полей;
б) одной записи;
в) нескольких записей;
г) всех записей.

15. Какой тип данных для поля таблицы следует выбрать для записи следующего значения (7272) 23-89-49:

а) логический;
б) счетчик;
в) числовой;
г) текстовый?

16. Какой размер указывается по умолчанию для полей текстового типа:

а) 255 символов;
б) 100 символов;
в) 65536 символов;
г) 50 символов?

17. Поле, значение которого не повторяется в различных записях, называется:

а) типом поля;
б) первичным ключом;
в) внешним ключом;
г) составным ключом.

РАЗДЕЛ 5

Веб-проектирование

Из данного раздела вы узнаете:

- назначение языка разметки;
- понятие тегов, элементов, атрибутов;
- структуру HTML-документа;
- простые приемы форматирования текста;
- CSS;
- как связывать страницы с базой данных.

Вы научитесь:

- использовать HTML-теги при разработке web-страниц;
- использовать CSS при разработке web-страниц;
- использовать скрипты при разработке web-страниц;
- применять HTML-теги для вставки мультимедиа объектов на web-страницу;
- использовать готовые скрипты при разработке web-страниц;
- устанавливать связь web-страницы с базой данных.

§53-54 Способы разработки веб-сайтов. HTML (HyperText Markup Language)

Вы научитесь:

- использовать HTML-теги при разработке web-страниц.

Ключевые понятия:

- тег
- атрибуты
- комментарий

Основой всемирной «паутины» World Wide Web является язык гипертекстовой разметки HTML (HyperText Markup Language).

ЭТО ИНТЕРЕСНО!

Концепция гипертекста, лежащая в основе WWW, была предложена Теодором Хольманом Нельсоном в 60-х годах XX в. Дальнейшим развитием гипертекста стала технология гипермедиа, позволяющая связывать гиперссылками не только текстовые фрагменты, но и данные иного типа (графику, звукозаписи, цифровое видео и пр.).

Первая попытка применения идеи гипертекста в Интернете была предпринята Тимом Бернерсом-Ли, сотрудником Лаборатории физики элементарных частиц Европейского центра ядерных исследований (CERN, г. Женева) в 1989 г.

Благодаря языку разметки пользователь может у себя на экране просмотреть web-документ в том виде, в каком его задумал разработчик: с определенными размерами шрифта и разбивкой на абзацы, с заданными размерами и расположением рисунков и др.

HTML — набор соглашений для разметки документов, которые определяют внешний вид документов на экране компьютера при доступе к ним с использованием программы браузера.

Документ, составленный с помощью языка разметки HTML, представляет собой текстовый файл. Такой файл можно набрать и отредактировать в обычном текстовом редакторе (Блокнот или WordPad). Но также существуют более удобные и совершенные программы подготовки HTML-документов (визуальные редакторы HTML и редакторы HTML-текстов).

Как выполняется разметка документа с помощью HTML?

В языке HTML имеется много тегов, среди которых — теги создания заголовка документа, задания параметров шрифта, вставки графических элементов и др.

Тег (tag — указатель, метка) — это фрагмент кода, который описывает определенный элемент документа HTML и заключается в угловые скобки < >.

Интернет-страничка состоит из нескольких частей: заголовка <HEAD> и тела <BODY>. Данную структуру в простейшем виде представим в таблице 53.1:

Таблица 53.1

Таблица тегов

<HTML>	начало контейнера HTML-документа
<HEAD>	начало контейнера заголовка
<TITLE>	начало контейнера строки — название страницы
...	строка названия страницы
</TITLE>	конец контейнера строки — название страницы
</HEAD>	конец контейнера заголовка
<BODY>	начало контейнера тела страницы
...	тело (все содержимое) страницы
</BODY>	конец контейнера строки
</HTML>	конец контейнера HTML-документа

Рассмотрим пример (добавленные к исходному тексту теги выделены жирным цветом).

<HTML>
<HEAD>
<TITLE>Привет! Hello! **</TITLE>**
HEAD>
<BODY> Это моя первая WEB-страница! Может быть, она не очень красивая, но она работает!!!
</BODY>
</HTML>

При необходимости в HTML-текст можно добавлять комментарии, состоящие из любых символов, заключенных в «полутеги» <! -- и -->.

Часто теги, помимо имени, содержат дополнительные элементы, которые называются *атрибутами*. Например, если в тег тела документа <BODY> ввести дополнительный элемент:

<BODY bgcolor= «yellow»>, то это будет означать, что документ должен отображаться на желтом фоне. Слово bgcolor является атрибутом, а yellow — значение атрибута.

Контрольные вопросы

1. Каково назначение языка разметки HTML?
2. Что понимается под *HTML-документом*?
3. Что такое *теги* и *элементы HTML*?
4. Что такое *заголовок* и *тело документа*?
5. Каково назначение атрибутов и в какой части кода они размещаются?
6. Как создать комментарий?

Практикум

Уровень А

Откройте текстовый редактор Блокнот и наберите текст, представленный в параграфе учебника. Сохраните файл index.html. Затем запустите созданный и сохраненный файл. Посмотрите, что у вас получилось. Обсудите полученный результат, сделайте выводы.

Уровень В

Заполните таблицу 53.2, используя интернет-ресурсы.

Таблица 53.2

Пример практикума

Теги	Назначение
<HTML>	Открытие документа
<HEAD>	
<BODY>	
 </BR>	
<HR></HR>	
<CENTER></CENTER>	
<ADDRESS>	
<B></B>	

Продолжение

<I></I>	
<SUB>	
<SUP>	
<FONT SIZE = «x»></FONT>	
<FONT COLOR= «x»>	
<FONT FACE= «ARIAL»>	
<UL> <LI>Элемент 1 <LI>Элемент 2 <LI>Элемент 3 </UL>	
<OL> <LI>Элемент 1 <LI>Элемент 2 <LI>Элемент 3 </OL>	
<IMG SRC= «URL»>	
<BODY BACKGROUND= «URL»>	
<BODY BGCOLOR= «x»>	
<BODY TEXT= «x»>	
<BODY VLINK= «x»>	
<BODY LINK= «x»>	
<BODY ALINK= «x»>	

Уровень С

Какой вариант HTML-кода для пустой web-страницы — правильный?

а)	б)	в)
<HTML> <HEAD> <TITLE> </HEAD> <BODY> </BODY> </HTML>	<HTML> <HEAD> <TITLE> </ TITLE > </HEAD> <BODY> </BODY> </HTML>	<HTML> <HEAD> <TITLE> </ TITLE > <BODY> </BODY> </HTML>

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§55 Форматирование текста (шрифт, абзац, списки)

Вы научитесь:

- использовать CSS при разработке web-страниц.

Ключевые понятия:

- форматирование строк, абзацев, заголовков в HTML-документе

Формирование абзацев и строк

Текст, который вы хотите поместить на страничке, может быть заготовлен заранее, и тогда в него добавляются требуемые теги. Для создания абзаца в языке HTML предусмотрено несколько возможностей. Простейшая из них — это использование тегов <P> и </P>, между которыми помещается текст абзаца.

Для перевода строки применяется тег
. Этот тег заставляет браузер перенести текст на новую страницу, что особенно удобно при наборе стихов.

Задание заголовков

Чтобы объявить выбранный абзац заголовком или подзаголовком (выбрав для него более крупный или жирный шрифт, чем для остального текста), достаточно заключить весь этот абзац в контейнер <H*> ... </H*>, где на место * записывается цифра.

Цифра 1 задает шрифт наибольшего размера — для заголовка;

2 — чуть меньший (для подзаголовков);

3 — еще меньше и т. д.

Выравнивание абзацев

Абзацы, которые задаются тегами <P> и
, по умолчанию выравниваются по левому краю страницы. С помощью атрибута aling. Значение aling = «center» будет задавать выравнивание по центру, aling = «right» — выравнивание по правому краю страницы, соответственно aling = «left» — выравнивание по левому краю.

Изменение параметров шрифта: вид (гарнитура), размер символов, цвет

Параметры шрифта, используемого для отображения текста на web-страницах задаются при помощи контейнера <FONT> ... </FONT>. Для этого элемента предусмотрены следующие атрибуты:

— face — гарнитура шрифта или список допустимых шрифтов;

— color — цвет шрифта;

— size — размер шрифта.

Например, добавим в наш пример текст оранжевого цвета шрифтом Arial. Для этого необходимо ввести в документ следующий тег:

<FONT face = «Arial» color = «orange»> </FONT>.

Чтобы задать то или иное начертание шрифта, необходимо использовать следующие теги: <B> ... </B> — для задания полужирного текста; <I> ... </I> — для задания курсивного текста и <U> ... </U> — для подчеркнутого текста.

Задание цвета шрифта и фона страницы

Если нужно определить цвет шрифта для всей страницы, то используют атрибут text в теге <BODY>. Например, тег вида

<BODY text = «red»> — задает для всего текста красный цвет.

Цвет фона всего HTML-документа определяется атрибутом bgcolor тега <BODY>. Например, следующий тег назначает оливковый цвет для фона:

<BODY bgcolor = «olive»>

Таблица 55.1

Теги для оформления текста

<HR>	Позволяет создать выпуклую горизонтальную линию для визуального выделения фрагментов текста на экране. Этот тег не требует закрывающего тега
<P> текст </P>	Формирует отдельный абзац: — перед абзацем добавляется небольшой отступ; — абзац по умолчанию выравнивается по левому краю; — между словами помещается ровно по одному пробелу (независимо от того, сколько их поставлено в исходном тексте); — перенос текста происходит автоматически, если очередное слово не умещается в экранной строке
<P aling= center> </P> <P aling= left> </P> <P aling= right> </P> <P aling= justify> </P>	Параметр **aling** задает выравнивание абзаца: — **center** — центрирование всех строк абзаца, в том числе при наличии принудительных разрывов строк с помощью тега ; — **left** — выравнивание по левому краю; — **right** — выравнивание по правому краю; — **justify** — выравнивание по ширине
<HN> текст заголовка </HN> <H1> текст заголовка </H1> <H2> текст заголовка </H2> <H3> текст заголовка </H3> <H4> текст заголовка </H4> <H5> текст заголовка </H5> <H6> текст заголовка </H6> <HN> aling= justify </HN>	Заголовки разделяют информацию на отдельные логические части и тем самым существенно улучшают восприятие. Заголовки в HTML бывают разными по значению (по уровню). Заголовок ... — самый крупный, ... — самый мелкий
текст	Полужирное начертание текста
текст	Курсивное начертание текста
<B> <I> текст </I></B>	Комбинация полужирного и курсивного начертания текста

Контрольные вопросы

1. Назовите известные вам элемента, формирующие отдельную строку (абзац) в HTML-документе?
2. Какие элементы задают иерархические заголовки?
3. Какой атрибут применяется для выравнивания абзацев?
4. С помощью каких тэгов и атрибутов задаются параметры шрифта?
5. Какие тэги задают курсивный и подчеркнутый шрифт?
6. Как задать цвет шрифта, отображаемого на странице?

Практикум

Уровень А

Примените различные элементы оформления на своей страничке.

Уровень В

Самостоятельно рассмотрите и примените на своей страничке маркированный и нумерованный список.

Уровень С

Выполните шрифтовое и фоновое оформление страницы.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§56-57 Таблицы

Вы научитесь:

- использовать HTML-теги при разработке web-страниц;
- создавать таблицы в Web-документах;
- форматировать таблицы;
- объединять ячейки в таблицах.

Ключевые понятия:

- элементы таблицы
- форматирование таблиц
- объединение ячеек в таблицах

Таблицы в web-документах применяются не только для размещения табличных данных. Они служат для вставки изображений и ссылок, для рациональной компоновки web-страниц.

Элементы таблицы

Таблицы строятся по принципу вложения и вводятся на web-страницу с помощью ряда элементов. Каждая таблица начинается открывающим тегом <TABLE> и заканчивается тегом </TABLE>. Создаваемая таблица как бы разворачивается по строкам, а строки заполняются ячейками. При этом внутрь тегов <TABLE>…</TABLE> могут вставляться следующие элементы:

TR — элемент создания строки;

TD — элемент, определяющий содержимое ячейки;

TH — элемент, определяющий ячейку заголовка.

Например, для создания таблицы 3x2 используется следующий шаблон:

```
<TABLE>
<TR><TD> . . . </TD><TD> . . . </TD></TR>
<TR><TD> . . . </TD><TD> . . . </TD></TR>
<TR><TD> . . . </TD><TD> . . . </TD></TR>
</TABLE>
```

По этому шаблону составим таблицу, например телефонов ваших друзей:

```
<TABLE border>
<TR><TD>  Фамилия, имя </TD><TD> Телефон </TD></TR>
<TR><TD> Аубакиров Данияр </TD><TD> 8(777)6572394 </TD></TR>
<TR><TD> Воробьев Кирилл </TD><TD> 8(707)5439715 </TD></TR>
<TR><TD> Гаппасов  Рамиль</TD><TD> 8(777)3456815 </TD></TR>
<TR><TD> Темирбаев Серик </TD><TD> 8(702)6598102 </TD></TR>
</TABLE>
```

Как вы обратили внимание, в тег <TABLE> введен атрибут border, задающий внешнюю и внутренние рамки таблицы толщиной 1 пиксель. Данная таблица на web-странице будет иметь вид в соответствии с рисунком 56.1.

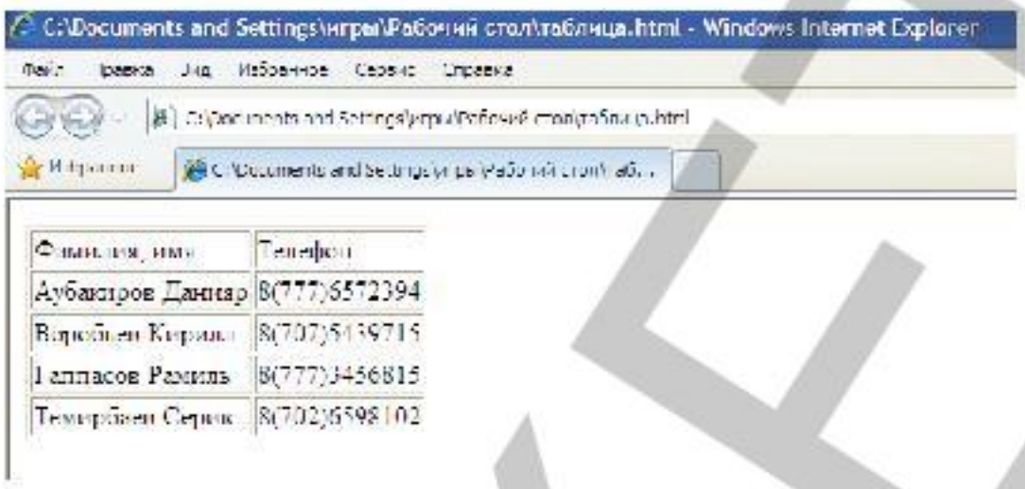


Рис. 56.1. Пример таблицы с внешними и внутренними рамками

Задание параметров таблицы

В нашем примере таблица имеет ширину столбцов, равную максимальной длине текста в ячейках. Таблица выровнена по левому краю браузера, а ее содержимое отображается шрифтом Times New Roman. Содержимое в ячейках заголовка выравнивается по центру, а в других ячейках — по левой границе. Чтобы изменить эти установки, принятые по умолчанию, используются различные атрибуты, представленные в таблице 56.1.

Таблица 56.1

Атрибуты элемента TABLE

Название атрибута	Описание
width — задает ширину таблицы	Его значение выражается в пикселях или в процентах (от полной ширины браузера). Например, тег <TABLE = «40%»> задает таблицу с длиной всех строк, равной 40% от ширины окна. Задание ширины в процентах предпочтительнее, поскольку строки таблицы полностью отображаются в окне браузера (без прокрутки)
Aling — задает выравнивание таблицы в документе	Этот атрибут может принимать одно из трех значений: ***left*** — размещение таблицы вдоль левое края документа; ***center*** — по центу документа; ***right*** — вдоль правого края
border — задает вывод рамок таблицы	Если значение этого атрибута не определено, например, <TABLE border>, все рамки будут иметь толщину 1 пиксель. Если задать, например, border = 5, то толщина пикселей будет присвоена только внешней рамке. Толщина внутренних рамок по-прежнему будет равняться 1 пиксель.

Цвет в таблицах

В таблице 56.2 опишем атрибуты, которые управляют цветом таблиц.

Таблица 56.2

Атрибуты управления цветом таблиц

Название атрибута	Описание
bgcolor — определяет цвет фона в таблице	В зависимости от того, в какой тег этот атрибут вводится (<TABLE>, <TR>, <TH> или <TD>), будет задан фон всей таблицы, фон строки, фон ячейки заголовка или фон данных. Например, тег <TABLE bgcolor = «red»> назначает красный фон всей таблицы, а тег <TD bgcolor = «yellow»> задает желтый фон ячейки данных
bordercolor — назначает цвет рамок таблицы	Если атрибут bordercolor вставить в тег <TABLE>, то он будет действовать, когда у таблицы имеются рамки, то есть при наличии атрибута bordercolor. Если же нужно задать цвет лишь определенных ячеек, атрибут bordercolor помещается в теги <TR>, <TH> или <TD>. Например, тег <TR bordercolor = «FF0000»> задает красные границы всех ячеек строки

Объединение ячеек таблицы

В языке HTML предусмотрена возможность объединения смежных ячеек. Для этого в начальных тегах <TH> или <TD> применяются следующие атрибуты, описанные в таблице 56.3.

Таблица 56.3

Атрибуты объединения ячеек

Название атрибута	Описание
rowspan — объединяет ячейки смежных строк	Значение атрибута задает количество объединяемых ячеек. Например, начальный тег ячейки <TD rowspan = 2> устанавливает объединение двух ячеек из смежных строк
colspan — объединяет ячейки смежных столбцов	Например, <TD colspan = 3> формирует одну ячейку данных из трех ячеек смежных столбцов

Если применить одновременно оба атрибута — rowspan и colspan, получим объединенную ячейку из смежных строк и столбцов. Например, тег <TD rowspan = 2 colspan = 4> задает ячейку, расположенную на пересечении двух строк и четырех столбцов.

Контрольные вопросы

1. Перечислите HTML-элементы, с помощью которых создаются таблицы.
2. Каковы параметры таблицы, принятые по умолчанию?
3. Как вставить в web-документ таблицу 3x3? Запишите HTML-код.
4. Как задать ширину таблицы?
5. Как выровнять таблицу по центру документа, по правому краю?
6. Какими атрибутами задаются толщина и цвет рамок таблицы?
7. Как задать заливку ячеек строки определенным цветом?
8. Запишите значения атрибутов для выравнивания содержимого ячеек по правому и по верхнему краям.
9. С помощью каких атрибутов выполняется объединение ячеек таблицы?

Практикум

Уровень А

Добавьте на свою web-страницу таблицу 3x2 и заполните ее.

Уровень В

Выполните цветовое оформление таблицы на web-странице.

Уровень С

Объедините несколько ячеек в таблице.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§58-60 CSS (Cascading Style Sheets)

Вы научитесь:

- использовать CSS при разработке web-страниц.

Ключевые понятия:

- технология CSS

CSS был введен вместе с HTML, чтобы обеспечить более лучший способ оформлять HTML-элементы.

Каскадные таблицы стилей (Cascading Style Sheets) — это технология управления дизайном web-страницы, которая позволяет упростить процесс редактирования и форматирования содержимого страницы.

Для чего нужна технология CSS?

Давайте представим такую ситуацию, что ваш сайт состоит из 50 страниц, и вы решили изменить цвет текста, фон, размер заголовков каждого документа и др. Используя CSS, это можно сделать за несколько секунд. Отредактировав всего одну строку CSS файла, можно поменять вид всех заголовков или таблиц сайта. Это позволяет разработчику сосредоточиться на дизайне, а не на рутинной работе.

Для этого необходимо изменить значения соответствующих атрибутов во внешнем CSS-файле.

Например, нужно сделать все заголовки h2 серыми.

В HTML:

```
<h2><font color=«gray»> Заголовок</font></h2>
```

Если элементов h2 много, то это утомительный процесс. А если потребуется потом поменять цвет на зеленый?

В CSS делаем одну запись:

```
h2 {color: gray;} —
```

и все заголовки — серые, а при необходимости, если поменять их на зеленые, то изменяем всего одно слово.

Структура правила представлена на рисунке 58.1

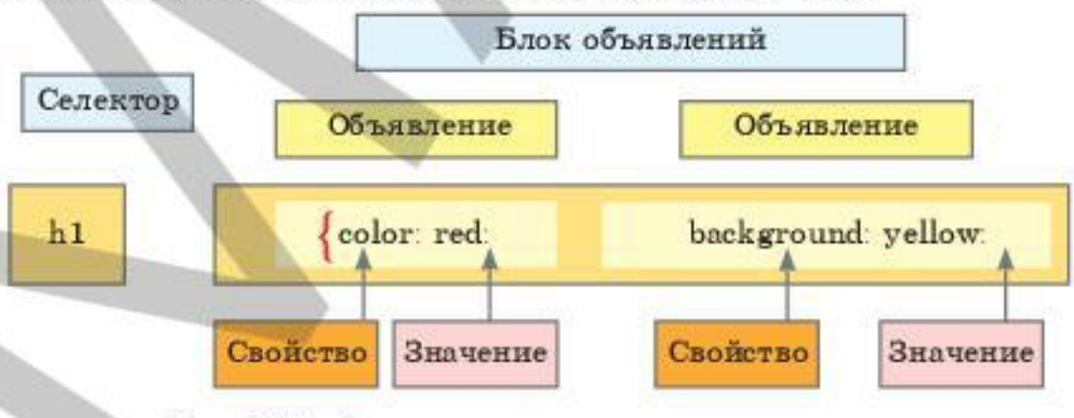


Рис. 58.1. Фрагмент редактирования страницы

Селектор — это чаще всего (но не всегда) элемент HTML — **h1, p, em, table**

```
P {color: silver;}      em {background: red;}
```

В блок объявлений входит одно или несколько объявлений. После «:» и «;» может быть произвольное количество пробелов.

Значение — это либо одно ключевое слово, либо несколько допустимых ключевых слов, разделенных пробелами:

```
P {font: medium Helvetica;}
```

Если указать неверное свойство или значение — все объявление будет проигнорировано целиком.

Например, у нас имеется список следующего вида:

```
<UL>
<LI>Айгерим</LI>
<LI>Ержан</LI>
<LI>Иван</LI>
<LI>Ильдар</LI>
</UL>
```

Результатом данного списка будет следующее:

- Айгерим
- Ержан
- Иван
- Ильдар

Применим каскадные таблицы стилей:

```
<TITLE>Современные уроки CSS</TITLE>
<STYLE type=«text/css»>
<ul> <li> {list-style:none; display:inline; background-color:#99ccff; padding:2px}
</STYLE>
</HEAD>
<BODY>
<ul>
<LI> Айгерим </LI>
<LI> Ержан </LI>
<LI> Иван </LI>
<LI> Ильдар </LI>
</UL>
</BODY>
```

В результате получим следующее:

Айгерим	Ержан	Иван	Ильдар

Таблица 58.1

Свойства шрифта и абзаца

font-family	Указывает шрифт или шрифтовое семейство, которым будет отображаться текст (иначе шрифт выбирается браузером пользователя). Так как у пользователя может не оказаться нужного шрифта, целесообразно указывать несколько типов шрифтов в порядке их предпочтения. Например, **p {font-family: Verdana, sans-serif}**
font-weight	Определяет степень жирности шрифта: **lighter** — обычный, **bold** — полужирный, **bolder** — жирный. Жирность шрифта также может определяться числами от 100 до 900. Например, **p {font-family: bolder}**
font-size	Определяет размер шрифта. Значение может указываться в процентах, пикселях и пунктах, а также значениями **smaller** — очень маленький, **small** — небольшой, **medium** — средний, **large** — большой, **larger** — очень большой. Например, **h1 {font-size: 200%}** **h2 {font-size: 200px** **h3 {font-size: 400pt}**
font-variant	Определяет вид текста обычный или написание капителью — заглавными буквами с размерами строчных. Допустимые значения: **normal** или **small-caps**. Если значение не указано, то по умолчанию используется значение **normal**
font-style	Определяет курсивное начертание шрифта: **normal** — без изменений, **italic** — курсив
text-decoration	Определяет такие эффекты оформления шрифта, как подчеркивание и перечеркивание. Допустимые значения: **non** — нет, **overline** — подчеркивание сверху, **underline** — подчеркивание снизу, **line-through** — перечеркивание. Например, **h1 {text-decoration: underline}**
text-align	Определяет выравнивание абзаца. Например, **p {font-aling: left}** **p {font-aling: right}** **p {font-aling: justify}** **p {font-aling: center}**
text-indent	Устанавливает отступ для первой («красной») строки. **Например,** **p {font-indent: 60pt}**
text-transform	Определяет различные режимы преобразования текста. Допустимые значения: **capitalize** — делает заглавной первую букву каждого слова, **uppercase** — делает все буквы заглавными, **lowercase** — делает все буквы строчными, **none** — снимает все установки

Практикум

Уровень А

Каскадные таблицы стилей позволяют разделить содержание и оформление web-документа так же, как стили в любом текстовом редакторе. С их помощью определяется внешний вид отображаемого документа: цвет текста, шрифт (гарнитура), выравнивание абзацев и многое другое. Для каждого элемента, задаваемого каким-либо тегом HTML, можно определить свой стиль отображения в окне браузера, меняя соответствующие установки форматирования, предусмотренные по умолчанию. Сведения о стилевой разметке документа содержатся внутри контейнера **<style>… </style>**, расположенного в разделе **<head>** документа HTML.

Пример:

```
<HTML>
<HEAD>
<TITLE> … </TITLE>
<STYLE type= «text/
css»>
<! -
H1{
  font-family:Verdana
  }
</STYLE>
</HEAD>
<BODY>
……
</BODY>
</HTML>
```

Атрибут type указывает на используемый тип таблиц стилей, в данном случае на **CSS**. Здесь речь идет об одном стиле, применяемом ко всем тегам **h1** для изменения шрифта заголовка на **Verdana**

Первая часть стиля называется *селектором*. Он определяет область применения данного стиля (в нашем случае селектор **h1** указывает, то этот стиль будет применен к тексту внутри всех контейнеров **h1** в составе web-документа).

Фигурные скобки ограничивают тело стиля. Внутри него содержится строка со свойствами и их значениями: свойство — перед символом двоеточия, а соответствующее значение — после двоеточия.

Уровень В

Примените стили к тексту **h1, h2, p** так, чтобы отформатировать информацию по образцу.

<STYLE> <! – H1 { font-family: Courier New; font-size: 20pt; font-weight: bolder ; text-align: center; } H2 { font-family: Courier New; font-size: 14pt; font-weight: bolder ; text-align: justify; } P { font-family: Arial; font-size: 18pt; text-align: left; } --> </STYLE>	Для стиля **h1** задан шрифт **Courier New**, размер — **20** пунктов, жирный шрифт, выравнивание по центру Для стиля **h2** задан шрифт **Sans-serif**, размер — **14** пунктов, жирный шрифт, выравнивание по ширине Для стиля **p** задан шрифт **Arial**, размер — **18** пунктов, выравнивание по левому краю

Образец

Растровая и векторная графика

Растровая графика

Растровое изображение формируется как матрица точек различного цвета (пикселей), которые образуют строки и столбцы. Каждый пиксель может принимать любой цвет из палитры, содержащей десятки тысяч или даже десятки миллионов цветов, поэтому растровые изображения обеспечивают высокую точность передачи цветов и полутонов.

Векторная графика

Векторные изображения формируются из объектов (точка, линия, окружность, прямоугольник и др.), которые называются графическими примитивами. Для каждого примитива задаются опорные координаты и цвет.

Уровень С

Добавьте CSS в теги в HTML-документ. Поэкспериментируйте со своей таблицей. Посмотрите, что у вас получилось.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§61 Внедрение мультимедиа

Вы научитесь:

- применять HTML-теги для вставки мультимедиа-объектов на web-страницу.

Ключевые понятия:

- якорь
- якорный элемент

Ну вот, наконец, мы подошли к самому главному — к тому, что отличает интернет-страницы от простого текста с иллюстрациями. Ссылка (гиперссылка) — это возможность «связать» любой находящийся на данной странице фрагмент текста (слово, фразу или целый абзац) либо рисунок с другой интернет-страницей или файлом данных (например, графическим файлом GIF или JPEG). Поэтому при щелчке мышью на таком тексте или рисунке вы автоматически переходите на указанную в ссылке страницу. В случае, когда страница находится на другом сервере, с ней автоматически устанавливается связь, если она доступна. Фрагменты текста, являющиеся ссылками, на странице, как правило, визуально выделяются цветом (обычно синим) и подчеркиванием.

Отправная точка ссылки задается тегом <A>. Имя этого тега происходит от первой буквы слова ***anchor*** — ***якорь***. Сам элемент A называется элементом привязки, или *якорным элементом*.

Внутри тега <A> ставится обязательный атрибут ***href***, с помощью которого определяется точка назначения ссылки (целевой ресурс). Между тегами <A> и </A> размещается текст ссылки или элемент рисунка. Так простейшая ссылка может выглядеть как:

```
<A href= «rest.html»> Мои каникулы </A>
```

На web-странице эта ссылка будет отображаться в виде текста «Мои каникулы». При щелчке мышью на этой ссылке будет загружаться HTML-файл rest.html.

Обратите внимание, что в атрибуте href указано только имя файла, что отвечает относительной ссылке на файл, который размещен, в той же папке, что и исходный документ.

Если нужно сослаться на ресурс, размещенный в WWW, то в атрибуте href указывается URL этого ресурса, например,

```
<A href= «http://www.samsung.com»> Продукция Samsung </A>
```

В качестве значения атрибута href можно указать ресурс mailto (вызов протокола электронной почты SMPT). Например, ссылка вида <A href= «mailto://mektep@mail.ru»> Письмо </A> позволит по-

сетителю вашей страницы непосредственно перейти к созданию и отправке сообщения по адресу mektep@mail.ru.

Вставка изображения

Трудно найти в WWW страницу, на которой не было бы изображений. Вставка изображения на web-страницу выполняется одиночным тегом <IMG>. Внутри этого тега обязательно записывается атрибут scr, содержащий URL-изображения. Название этого атрибута происходит от слова *sourse* — «источник».

Как вы думаете, как разместить картинку на страничке?

Для этого сохраните файл с изображением в определенной папке (например, в той же папке, что и сам HTML-документ), а в документ введите тег <IMG src=«картинка.ipg»>.

По умолчанию браузер отобразит изображение, выровненное по левому краю страницы.

Как вы думаете, какого размера будет изображение на web-странице?

По умолчанию браузером будут использованы действительные размеры изображения, хранящегося в графическом файле. Если нужно изменить эти размеры, то применяются атрибуты width (ширина) и height (высота), входящие в тэг <IMG>. Значения размеров изображения задаются обычно в пикселях, например, width = «133» height «33».

Можно также ширину и высоту задавать в процентах относительно размеров внешнего элемента (страницы).

В Интернете наиболее популярны два графических формата (табл. 61.1):

Таблица 61.1

JPG — для фотографий и сложных по цветовой гамме рисунков с плавными переходами между оттенками **GIF** — для логотипов, надписей, заголовков и рисунков, имеющих четкие цветовые границы	
<IMG>	Тег **<IMG>** — позволяет разместить иллюстрации в нужном месте текста
<IMG **src=«имя файла»**>	Параметр **src** — указывает имя файла с иллюстрацией

Продолжение

<IMG src=«имя файла» alt= «текст_надписи»>	Параметр **alt** — позволяет снабдить иллюстрацию текстовой подписью («всплывающей подсказкой»)
<IMG src=«имя файла» width=значение в пикселях или процентах>	Параметр **width** — задает ширину изображения
<IMG src=«имя файла» height=значение в пикселях>	Параметр **height** — задает высоту изображения
<IMG src=«имя файла» border=значение в пикселях>	Параметр **border** — задает толщину рамки вокруг рисунка
<IMG src=«имя файла» hspace =значение в пикселях>	Параметр **hspace** — задает поля слева и справа от рисунка
<IMG src=«имя файла» vspase=значение в пикселях>	Параметр **vspase** — задает поля сверху и снизу от рисунка
<IMG src=«имя файла» align= top | middle | bottom | left | right>	Параметр **align** — задает положение иллюстрации по отношению к соседним элементам документа (прежде всего — к окружающему тексту): **top** — вертикальное выравнивание по верхнему краю; **middle** — вертикальное выравнивание по центру; **bottom** — вертикальное выравнивание по нижнему краю; **left** — горизонтальное выравнивание по левому краю; **right** — горизонтальное выравнивание по правому краю

Контрольные вопросы

1. Каким элементом HTML задается ссылка?
2. С помощью какого элемента и атрибута на web-страницу вставляется изображение?
3. Как задается размер изображения?
4. Какой графический формат должно иметь изображение, размещаемое на web-странице?

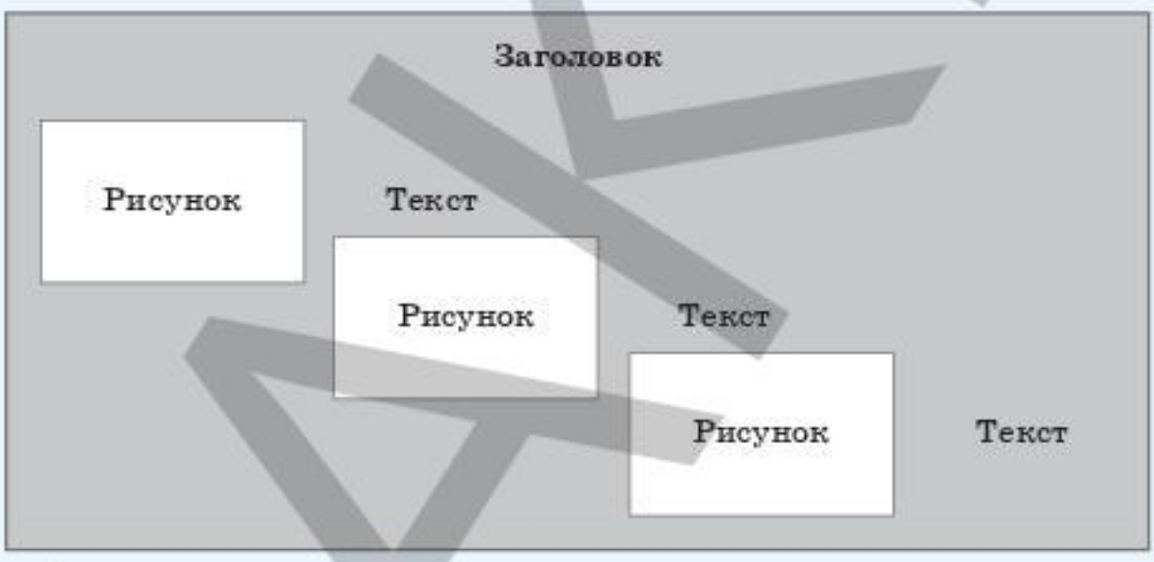
Практикум

Уровень А

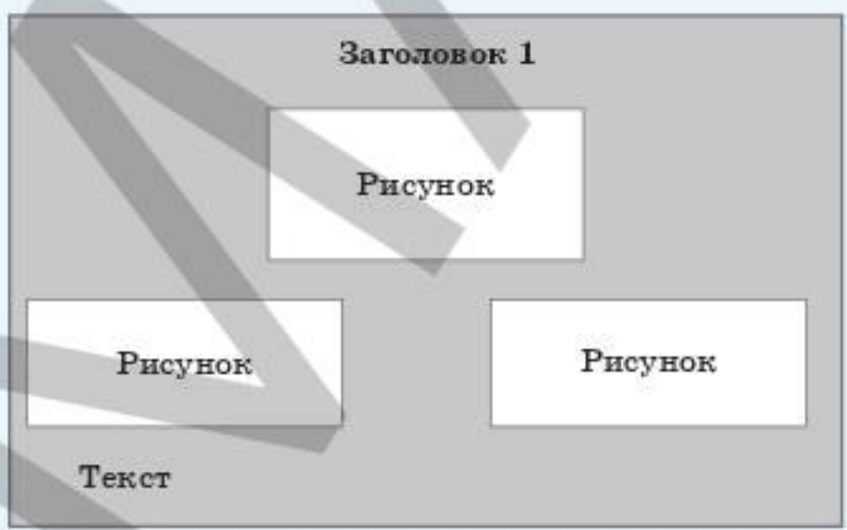
Создайте 3 web-странички по следующим шаблонам:

1.

2.



3.



Компоновка изображений и текста

Обратите внимание на web-страницу, показанную на рисунке 61.1. Устраните ошибку на рисунке 61.1 и преобразуйте web-страницу как на рисунке 61.2.

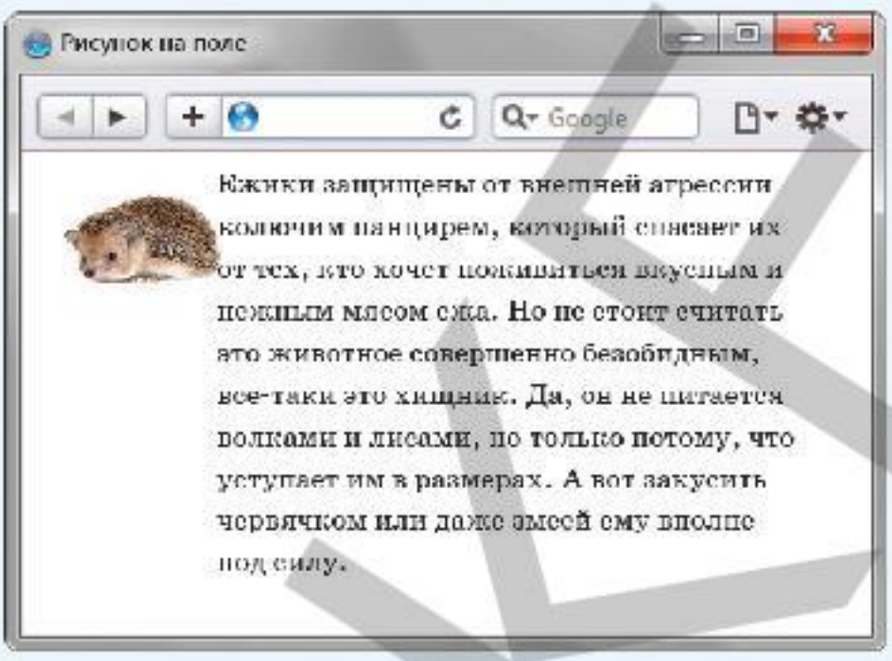


Рис. 61.1. Web-страничка с ошибкой

Как можно устранить этот дефект, задать свободное поле вокруг рисунка (как на рисунке 61.2, представленном ниже)?

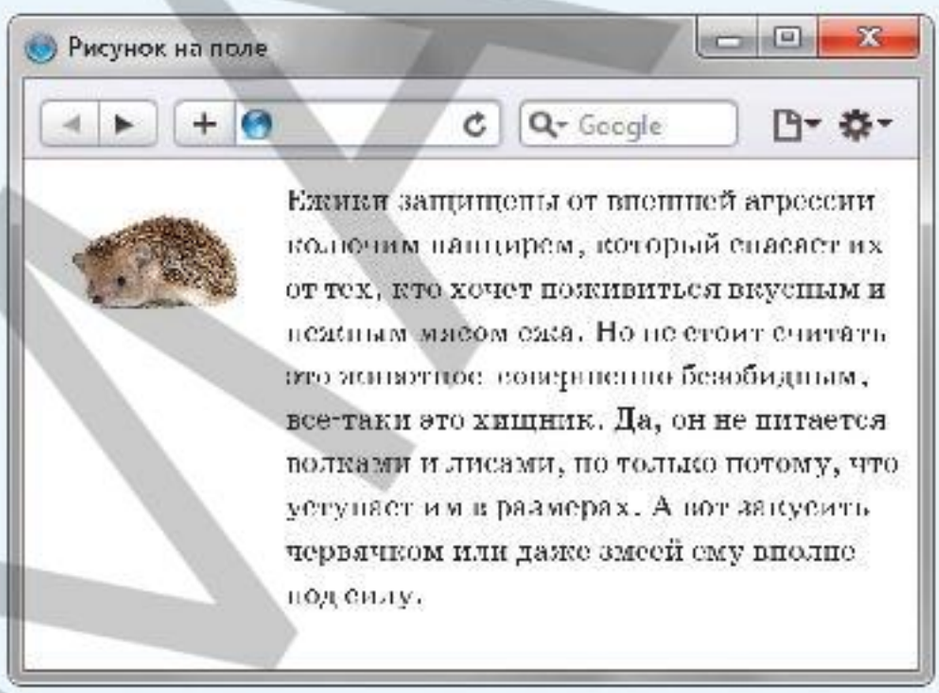


Рис. 61.2. Редактирование web-странички

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§ 62 Использование скриптов

Вы научитесь:

- использовать скрипты при разработке web-страниц.

Ключевые понятия:

- скрипт сайта

Скрипты используются для сбора статистики (о посещениях, просмотрах и т. д.), для оптимизация поиска на сайте, для упрощения изменения структуры сайта, для организации работы форумов.

Скрипт помещается в отдельный файл и загружается на сервер. Когда нужно его выполнить, к файлу приходит обращение.

Скрипт сайта — это код, который внедряется на ресурс и расширяет функциональность портала. Другими словами, скрипт — это исполняемый процесс, запускаемый сервером по специальному запросу, поступающему со страницы web-сервера для выполнения определенной задачи.

Рассмотрим на простом примере. Стиральная машинка-автомат может стирать в нескольких режимах. Для этого ее нужно правильно запрограммировать. При нажатии на кнопку будет запускаться скрипт определенного режима стирки.

На сайте скрипт работает примерно так же: запускается при определенном условии и выполняет свою работу.

Существует несколько языков программирования, с помощью которых делаются скрипты. Например,

- Jscript;
- Python;
- JavaScript;
- PHP;
- Perl;
- AngelScript.

Рассмотрим пример разработки тестов (взаимодействие JavaScript-программы с элементами форм).

Одно из применений JavaScript — это создание на базе HTML электронных учебников, снабженных интерактивными тестами. Это

может быть web-страница, содержащая некоторую форму (включает в себя текст вопросов и диалоговые элементы). Форма позволяет выбрать или ввести ответ.

Рассмотрим один из вариантов построения тестовых форм и соответствующие им Java-скрипты.

Самое простое — создать форму, содержащую список ответов, из которых один правильный, где тексты ответов представляют собой кнопки. В этом случае анализ ответов сводится лишь к вызову по событию *onclick* для каждой из кнопок функции-обработчика *Verno*() или *Neverno*(), выводящих окна *alert* с сообщениями «Ответ правильный» и «Ответ неправильный», соответственно.

Пример формы и скриптовой программы:

```
<H2 ALIGN=CENTER><FONT COLOR= red>
Тест 1<FONT></H2>
<P ALIGN=JUSTIFY><B> Вопрос.</B> Какой тэг служит для
создания горизонтальной отлиновки? </P>
<FROM NAME= «TEST1»>
<INPUT TYPE= BUTTON VALUE= «1» NAME= «otv1»
onClick= «javascript: Neverrno ();»>
&1t; TABLE&GT; <BR>
<INPUT TYPE= BUTTON VALUE= «2» NAME= «otv1»
onClick= «javascript: Verno ();»>
&1t; HR&gt; <BR>
<INPUT TYPE= BUTTON VALUE= «3» NAME= «otv3»
onClick= «javascript: Neverrno ();»>
&1t; LINE&gt; <BR>
<INPUT TYPE= BUTTON VALUE= «4» NAME= «otv4»
onClick= «javascript: neverrno ();»>
&1t; Pow&gt;
</FROM>
…
<SCRIPT LANGUAGE= javascript >
// правильный ответ - №2
}
Function Verno () {

Alert («ответ правильный»);
}
Function Neverno () {
}
Alert («ответ неправильный»);
}
</ SCRIPT >
```

Контрольные вопросы

1. Что такое *скрипт*?
2. Перечислите языки программирования, предназначенные для написания скриптов?

Практикум

Уровень А

Выполните пример из параграфа учебника, представленный выше.

Уровень В

Составьте интерактивный тест, содержащий пять вопросов и пять вариантов ответа.

Уровень С

Составьте интерактивный тест, содержащий пять вопросов и пять вариантов ответа разного типа (ввод ответа в поле ввода текста, вопрос с несколькими вариантами ответа, составление ответа из готовых фрагментов и др.).

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

§63 Связь web-страницы с базой данных

Вы научитесь:

- устанавливать связь web-страницы с базой данных.

Ключевые понятия:

- web-страница
- база данных

В данном параграфе рассмотрим, как связать web-страницы с базой данных. Подсоединиться к базе данных очень просто:

```
db = openDatabase("ToDo", "0.1", "A list of to do items.", 200000);
```

Данный код позволяет создать объект, представляющий базу данных, а если базы данных нет, то она создается. В аргументах указывается имя базы данных, версия, отображаемое имя и приблизительный размер. Отметим, что приблизительный размер не является ограничением. Реальный размер базы данных может изменяться.

Проверить подключение к базе данных можно, используя команду:

```
if(!db){alert("Failed to connect to database.");}
```

Для выполнения запросов к базе данных необходимо предварительно создать транзакцию, вызвав функцию database.transaction(). У нее один аргумент, а именно другая JavaScript функция, принимающая объект транзакции и предпринимающая запросы к базе данных.

SQL-запрос можно выполнить, вызвав функцию executeSql объекта транзакции.

Пример работы функции executeSql:

```
db.transaction(function(tx) {
tx.executeSql("SELECT COUNT(*) FROM ToDo", [], function(result){}, function(tx, error){});
});
```

Далее изменим код таким образом, чтобы при невозможности выборки из таблицы «ToDo» (которой пока не существует), данная таблица создавалась.

```
db.transaction(function(tx) {
tx.executeSql("SELECT COUNT(*) FROM ToDo", [], function (result) { alert('dsfsdf') }, function (tx, error) {
tx.executeSql("CREATE TABLE ToDo (id REAL UNIQUE, label TEXT, timestamp REAL)", [], null, null);
})});
```

Также, используя SQL-запросы, можно добавлять и удалять строки в таблицу.

Практикум

Уровень А

Установите связь между своей web-страницей и базой данных.

Уровень В

Добавьте новые записи в таблицу.

Уровень С

Заполните свободные поля в таблице.

Рефлексия:

- Какая информация вас особенно заинтересовала?
- Какие возникли трудности и с кем вы готовы их обсудить?
- Какие навыки вы готовы применять уже сейчас?

Проверь себя!

1. С помощью какого атрибута можно задать текст для картинки, который будет отображен, если ее не удастся загрузить:
   а) caption; в) alt;
   б) title; г) popup?
2. С помощью какого свойства можно сделать отступы внутри ячейки в таблице:
   а) case; в) margin;
   б) space; г) padding?
3. Как выделить текст курсивом:
   а) <P> курсив </p>; в) <HR> курсив </hr>;
   б) <EM> курсив </EM>; г) <C> курсив </C>?
4. С помощью какого тега нужно задавать подписи к полям формы:
   а) id; в) label;
   б) type; г) field?
5. Как сделать картинку ссылкой:
   а) <IMG link="https://webshake.ru/html-training-course" src="promo.png">;
   б) <A href="https://webshake.ru/html-training-course"><IMG src="promo.png"></A>;
   в) <IMG href="https://webshake.ru/html-training-course" src="promo.png">?

6. Какое число заголовков первого уровня считается допустимым:

а) 3;
в) 4;
б) 1;
г) 2?

7. Как правильно оформить цитату?

а) <BLOCKQUOTE>текст цитаты<cite>автор цитаты</CITE></BLOCKQUOTE>;
б) <BLOCKQUOTE>текст цитаты</BLOCKQUOTE><cite>автор цитаты</CITE>?

8. Как сделать текст жирным:

а)
 жирный </BR>;
в) <P> жирный </P>;
б) <A> жирный </A>;
г) <STRONG> жирный </STRONG>?

9. Как вставить картинку в HTML:

а) <IMAGE>http://site.com/image.jpg</IMAGE>;
б) <IMG>http://site.com/image.jpg</IMG>;
в) <IMAGE source="http://site.com/image.jpg">;
г) <IMG src="http://site.com/image.jpg">?

10. С помощью какого свойства изменяется ширина таблицы:

а) count;
в) length;
б) size;
г) width?

Глоссарий

IP-адрес — уникальный физический адрес компьютера, подключенного к Интернету. Составляется из четырех десятичных чисел, разделенных точкой, — каждое в диапазоне от 0 до 255 (четыре байта). Например, 192.126.0.18.

HTML — набор соглашений для разметки документов, которые определяют внешний вид документов на экране компьютера при доступе к ним с использованием программы браузера.

URL — адрес страницы или файла в Интернете.

Web-страница — основной структурный элемент *«Всемирной паутины»* — документ, который содержит текстовую и (или) графическую информацию, а также ссылки на другие документы в Интернете.

WWW (Word Wide Web) — гипертекстовая информационно-поисковая система в Интернете.

Авторизация — предоставление доступа к какому-либо ресурсу (например, к электронной почте после ввода пароля, разблокировка смартфона после сканирования отпечатка пальца и др.).

Алгоритм — совокупность правил выполнения определенных действий, обеспечивающих решение задачи.

Алфавит — определенный набор символов, а символы называются буквами алфавита.

Аутентификация — процедура проверки подлинности. Другими словами, пользователя проверяют с помощью пароля, письмо проверяют по электронной почте и т. д.

Байт — совокупность из восьми *бит*, воспринимаемая компьютером как единое целое.

База данных (БД) — информационная модель, которая позволяет в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств.

Бит — наименьшая единица информации, известная в природе. Значение бита — 0 или 1, и это можно толковать как альтернативу: «выключено — включено», «нет — да», «ложь — истина».

Гипертекст — документ, содержащий активные ссылки на другие документы.

Данные — обрабатываемая информация, представленная в памяти компьютера в двоичной форме.

Двоичное (бинарное) дерево — дерево, в котором каждый узел имеет не более двух «потомков».

Дизъюнкция — логическая операция, которая каждым двум простым высказываниям ставит в соответствие составное высказывание, являющееся ложным тогда и только тогда, когда все исходные высказывания ложны и являющееся истинным, когда хотя бы одно из исходных высказываний истинно.

Запись базы данных — строка таблицы, содержащая набор значений свойств, размещенный в полях базы данных.

Запрос — обращение к СУБД для отбора записей или выполнения других операций.

Идентификация — процедура распознавания субъекта по его идентификатору. Другими словами, это определение имени, логина или номера и т. д.

Информационная безопасность — состояние сохранности информационных ресурсов и защищенности законных прав личности и общества в информационной сфере.

Информационная модель — совокупность информации об объекте или процессе.

Информационный вес символа — количество информации, которое несет один символ алфавита.

1 бит — информационный вес символа двоичного (двухсимвольного) алфавита.

Код — набор символов (условных обозначений) для представления информации.

Кодирование — процесс представления информации в виде кода.

Кодовая таблица — таблица, которая устанавливает соответствие между символами алфавита и двоичными числами. Эти числа называются кодами символов и отвечают внутреннему представлению символов в компьютере.

Компьютерная этика — дисциплина, которая занимается исследованием поведения людей, использующих компьютер.

Компьютерная сеть — группа компьютеров, объединенных линиями связи.

Конъюнкция — логическая операция, ставящая в соответствие каждым двум простым высказываниям составное высказывание, являющееся истинным тогда и только тогда, когда оба исходных высказывания истинны.

Кроссплатформенная программа — программа, у которой есть версии для различных операционных систем (Windows и Linux).

Лексема — последовательность символов, отделенная символами-разделителями (обычно пробелами или знаками пунктуации).

Логика — наука о высказываниях и их связях.

Логический элемент компьютера — это часть электронной логической схемы, которая реализует элементарную логическую функцию.

Отрицание — логическая операция, которая каждому простому высказыванию ставит в соответствие составное высказывание, заключающееся в том, что исходное высказывание отрицается.

Мультимедиа — специальная технология, которая позволяет с помощью программного обеспечения объединить на твоем компьютере обычную информацию (текст или графику) со звуком и движущимися изображениями (можно создавать свои видеофильмы).

Первичный ключ в базе данных — поле (или совокупность полей), значение которого не повторяется у разных записей.

Поле базы данных — столбец таблицы, содержащий значения определенного свойства.

Программирование — профессиональная деятельность по разработке программного обеспечения компьютеров.

Провайдер — фирма, предоставляющая пользователям выход в Интернет.

Протокол — определенные правила, в соответствии с которыми происходит обмен информацией между компьютерами.

Свойство объекта — совокупность признаков объекта, по которым его можно отличить от других объектов.

Сортировка — (англ. sorting — классификация, упорядочение) — последовательное расположение или разбиение на группы чего-либо в зависимости от выбранного критерия.

Скрипт сайта — код, который внедряется на ресурс и расширяет функциональность портала. Другими словами, скрипт — это исполняемый процесс, запускаемый сервером по специальному запросу, поступающего со страницы web-сервера для выполнения определенной задачи.

Тег (tag — указатель, метка) — фрагмент кода, который описывает определенный элемент документа HTML и заключается в угловые скобки < >.

Форма — объект базы данных, предназначенный для ввода и отображения информации.

Список литературы

1. *Андрианов В. В.* Обеспечение информационной безопасности бизнеса. М.: Альпина Паблишерз, 2011. 373 с.
2. *Бейли Л.* Изучаем SQL. СПб.: Питер, 2012. 592 с.
3. *Бирюков А. А.* Информационная безопасность: защита и нападение. М.: ДМК Пресс, 2012. 474 с.
4. *Бураков П. В., Петров В. Ю.* Введение в системы баз данных. СПб.: Питер. 2015. 186 с.
5. Компьютерные сети. Принципы, технологии, протоколы: Учебник. 5-е изд. СПб.: Питер, 2016. 992 с.
6. *Лутц М.* Изучаем Python. Символ-Плюс, 2018.
7. *Лутц М.* Программирование на Python. Том 1. Символ-Плюс, 2012.
8. *Мейер Э. А.* CSS-каскадные таблицы стилей. Подробное руководство. Символ-Плюс, 2013.
9. *Нестеров С. А.* Базы данных: учебник и практикум. М.: Издательство Юрайт, 2018. 230 с.
10. *Седер Н.* Python. Экспресс-курс. СПб.: Питер, 2018.
11. *Сергеев А.* Основы локальных компьютерных сетей. 2016.
12. *Таненбаум Э., Уэзеролл Д.* Компьютерные сети. 5-е изд. 2016.
13. *Филиппов Б. И.* Информационная безопасность. Основы надежности средств связи. М.: Директ-Медиа, 2019. 241 с.
14. *Фримен Э.* Изучаем HTML, XHTML и CSS. СПб.: Питер, 2019.
15. *Царев Р. Ю.* Программные и аппаратные средства информатики. Красноярск, 2015. 160 с.

Предметный указатель

СОДЕРЖАНИЕ

Раздел 5. ВЕБ-ПРОЕКТИРОВАНИЕ

Учебное издание

Кольева Наталья Станиславовна
Шевчук Елена Владимировна

ИНФОРМАТИКА

Учебник для 10 классов
естественно-математического направления
общеобразовательных школ

Редактор *К. Амирова*
Дизайн макета, иллюстрации *Е. Мельник*
Художн. редактор *Л. Уралбаева*
Техн. редактор *Л. Садыкова*
Корректор *Е. Шумских*
Компьютерная верстка *И. Алмабаевой*

Государственная лицензия № 0000001 выдана издательству
Министерством образования и науки Республики Казахстан 7 июля 2003 года

ИБ № 5882

Подписано в печать 14.06.19. Формат $70\times100^{1}/_{16}$. Бумага офсетная.
Гарнитура «SchoolBook Kza». Печать офсетная. Усл.-печ. л. 14,84.
Усл. кр.-отт. 60,01. Уч.-изд. л. 6,46. Тираж 10 000 экз. Заказ №

Издательство «Мектеп», 050009, Алматы, пр. Абая, 143
Факс: 8(727) 394-42-30, 394-37-58
Тел.: 8(727) 394-41-76, 394-42-34
E-mail: mektep@mail.ru
Web-site: www.mektep.kz