Ministerul Educației al Republicii Moldova

Анатол Гремальски
Сержиу Корлат
Андрей Брайков

ИНФОРМАТИКА

Учебник для XII класса

Ştiinţa
2015

CZU 004 (075.3)
Г 80

Elaborat în baza curriculumului disciplinar în vigoare și aprobat prin Ordinul ministrului educației (nr. 399 din 25 mai 2015). Editat din mijloacele financiare ale *Fondului special pentru manuale*.

Comisia de evaluare: *Gheorghe Chistrugă*, prof. școlar, gr. did. superior, Liceul Teoretic „Mihai Eminescu", Drochia; *Elena Doronina*, prof. școlar, gr. did. superior, Liceul Teoretic „Natalia Gheorghiu", Chișinău; *Svetlana Rudoi*, prof. școlar, gr. did. superior, Liceul Teoretic „Mihai Eminescu", Chișinău

Recenzenți: *Tatiana Baciu*, dr. în psihologie, Universitatea Pedagogică de Stat „Ion Creangă"; *Alexei Colîbneac*, prof. universitar, șef catedră „Grafică", Academia de Muzică, Teatru și Arte Plastice, maestru în arte; *Tatiana Cartaleanu*, dr. în filologie, Universitatea Pedagogică de Stat „Ion Creangă"

Acest manual este proprietatea Ministerului Educației al Republicii Moldova

Liceul _____
Manualul nr. _____

Anul de folosire	Numele de familie și prenumele elevului	Anul școlar	Aspectul manualului	
			la primire	la restituire
1				
2				
3				
4				
5				

Dirigintele controlează dacă numele elevului este scris corect.
Elevul nu trebuie să facă niciun fel de însemnări în manual.
Aspectul manualului (la primire și la restituire) se va aprecia cu calificativele: *nou, bun, satisfăcător, nesatisfăcător*.

Traducere din limba română: *Irina Ciobanu*
Redactor: *Valentina Rîbalchina*
Corector: *Tatiana Bolgar*
Redactor tehnic: *Nina Duduciuc*
Machetare computerizată și coperta: *Olga Ciuntu*

Întreprinderea Editorial-Poligrafică *Știința*,
str. Academiei, nr. 3; MD-2028, Chișinău, Republica Moldova;
tel.: (+373 22) 73-96-16; fax: (+373 22) 73-96-27;
e-mail: prini@stiinta. asm.md; prini_stiinta@yahoo.com;
www.stiinta.asm.md

Descrierea CIP a Camerei Naționale a Cărții
Гремальски, Анатол
Информатика: Учеб. для 12 кл. / Анатол Гремальски, Сержиу Корлат, Андрей Брайков; trad. din lb. rom: Irina Ciobanu; Min. Educației al Rep. Moldova. – Ch.: Î.E.P. *Știința*, 2015 (Tipogr. „BALACRON" SRL) – 144 p.

ISBN 978-9975-67-999-2

004 (075.3)

Imprimare la Tipografia „BALACRON" SRL, str. Calea Ieșilor, 10, MD-2069, Chișinău, Republica Moldova
Comanda nr. 646

ISBN 978-9975-67-999-2

ОГЛАВЛЕНИЕ

ГЛАВА 1

ЭЛЕМЕНТЫ МОДЕЛИРОВАНИЯ

Изучив данную главу, вы сможете:

- уточнять смысл понятий *модель* и *моделирование*;
- классифицировать модели предметов, процессов, явлений;
- применять и создавать математические модели;
- отличать аналитические решения от имитационных, полученных путем вычислительных экспериментов;
- объяснять способы получения аналитических решений и решений, найденных путем вычислительного эксперимента;
- объяснять взаимодействие между математической моделью, алгоритмом и программой;
- планировать процесс решения задачи на компьютере.

1.1. Понятие модели. Классификация моделей

Имитация процессов, предметов или явлений природы присуща человеческому обществу на протяжении всего его исторического развития. Первые рисунки, сделанные людьми каменной эпохи на стенах пещер, являются также и первой попыткой отобразить реальные предметы и явления природы с помощью изображений (*рис. 1.1*).

Глобус-макет нашей планеты также является имитацией реально существующего предмета. С его помощью мы узнаем о форме Земли, о ее движении, о местоположении континентов, океанов, стран и городов (*рис. 1.2*). Но элементы макета не представляют соответствующий ему реальный предмет полностью. Так, в случае глобуса-макета мы имеем дело со сферическим телом, через центр которого проходит ось, вокруг которой возможно вращение. На его поверхности напечатана различная

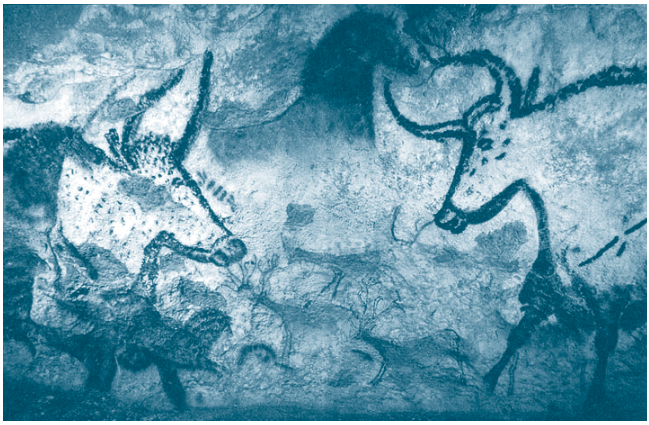


Рис. 1.1. Зооморфное изображение времен палеолита

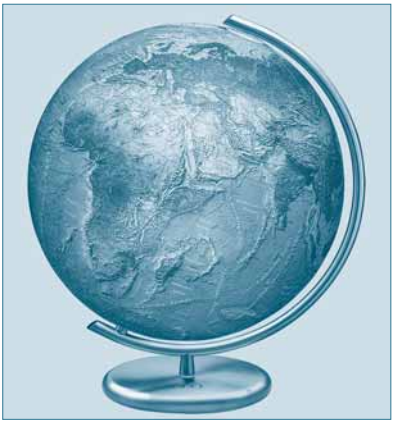


Рис. 1.2. Глобус-макет Земли

информация о планете Земля. Глобус-макет отражает лишь определенные черты реального космического объекта, но отличается от него размерами, физическими свойствами, структурой и т.д. Глобус-макет – это *модель*, являющаяся упрощенным представлением *Терры*, позволяющая изучать лишь некоторые ее свойства.

Модель – это материальная либо идеальная, логико-математическая система, с помощью которой, путем аналогии, могут быть изучены свойства и действия над реальной, значительно более сложной системой.

С давних времен модели используют для изучения природных явлений, сложных объектов и процессов, таких как молекулы, атомы, солнечная система, вселенная, атомный реактор, небоскреб и т.д. Удачная модель более удобна для исследований, нежели реальный объект. Более того, некоторые предметы и явления не могут быть изучены в оригинале. Например, сложно осуществить опыты над экономикой страны, невозможно осуществить опыты над планетами солнечной системы либо над уже прошедшими событиями.

Другим очень важным аспектом моделирования является возможность выделить, сосредоточить внимание на тех свойствах изучаемого реального объекта, которые определяют его суть. Модели также позволяют обучать правильно использовать реальные объекты путем отработки, с их помощью, реакций, необходимых в различных ситуациях.

Опыты над реальными объектами могут быть невозможными либо опасными (продолжительность процесса во времени, возможность разрушения объекта). В случае изучения динамических объектов, характеристики которых меняются во времени, особую важность приобретает возможность прогнозирования состояния объекта под воздействием определенных факторов.

В целом, хорошо построенная модель позволяет получить новые знания об изучаемом реальном объекте.

Процесс построения модели называется моделированием.

Существует несколько типов моделирования, которые могут быть объединены в две большие группы: *материальное моделирование* и *идеальное моделирование*.

В случае ***материального моделирования*** изучение оригинала осуществляется с помощью другого, более простого, материального предмета, называемого моделью, который обладает основными геометрическими, физическими, динамическими и функциональными характеристиками реального предмета. Примеры: макеты зданий, самолетов, военной техники и т.д.

В случае ***идеального моделирования*** изучение оригинала осуществляется путем представления его свойств с помощью определенных концептов, схем, планов, структур, существующих лишь в воображении человека.

В общем, *идеальное моделирование* основывается на интуитивном представлении об изучаемом предмете. Например, опыт жизни каждого человека может служить личностной моделью окружающего мира. Моделирование называют *символьным* в случае, когда используются различные символы, схемы, графики, формулы. В категории символьного моделирования особое место занимает *математическое моделирование*. В данном случае изучение реального объекта осуществляется с помощью модели, описанной математическими средствами и понятиями. Класси-

ческим примером математического моделирования является описание и изучение математическими средствами основных законов механики Ньютона.

Для того чтобы изучить какой-либо процесс или явление, вовсе недостаточно создать его модель. В общем случае, модель описывает определенные связи, взаимодействия, свойства оригинала. Цель моделирования, однако, состоит в получении новых данных об оригинале с помощью модели.

Вопросы и упражнения

1. Объясните понятие *модель*. Приведите примеры реально существующих в окружающем мире предметов и их моделей. В каждом конкретном случае уточните характеристики реальных предметов, которые отражены в их модели.
2. Сформулируйте понятие *материальная модель*. Приведите примеры материальных моделей. Обоснуйте необходимость применения материальных моделей в различных областях науки и техники.
3. Сформулируйте понятие *идеальная модель*. Приведите примеры идеальных моделей. Обоснуйте необходимость применения идеальных моделей.
4. Исследователь Е. Резерфорд предложил в 1911-м году „планетарную (ядерную) модель атома". Объясните смысл слова „модель" в предложении Резерфорда, а также обоснуйте, почему эта модель называется планетарной.

5. Объясните смысл следующего высказывания: „Пиктограммы графического интерфейса пользователя представляют собой модели реальных объектов, обрабатываемых программными продуктами".

1.2. Математические модели и математическое моделирование

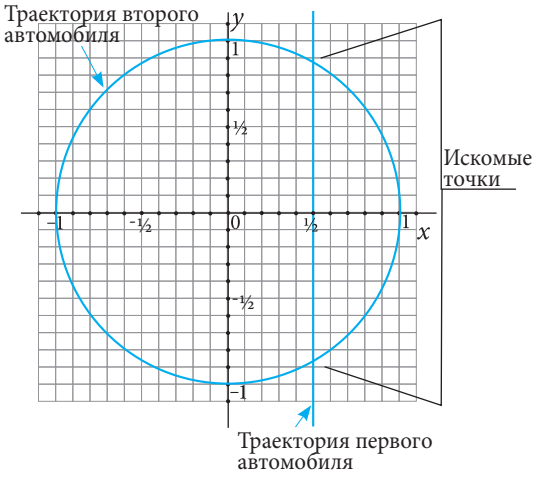
Одной из базовых целей информатики, как межпредметной науки, является разработка методов решения сложных исследовательских и вычислительных задач с помощью компьютера. Первоначально информатика развивалась как область прикладной математики. Первые задачи, решаемые средствами информатики, были чисто математическими, а их решение сводилось к осуществлению большого числа сложных вычислений. В настоящее время информатика стала независимой наукой, основанной на математических законах, обладающей собственными методами и объектами исследований. Информатика изучает и решает сложные задачи из области математики, физики, химии, биологии, экономики, экологии, филологии и социологии. Однако независимо от области, к которой принадлежит задача, информатика при ее решении основывается на математике. И, как следствие, для решения любой задачи на компьютере сначала необходимо с помощью математических понятий описать процессы и явления, происходящие в ней. В частности, это могут быть функции, уравнения, неравенства, системы уравнений и др.

Математическая модель представляет собой описание изучаемого процесса либо явления с помощью математических понятий.

Пример 1: Пусть заданы два автомобиля. Один из них движется прямолинейно, траектория его движения описывается уравнением $x = 1/2$. Второй автомобиль перемещается по круговой траектории, описываемой окружностью с единичным радиусом и центром, совпадающим с началом системы координат. Требуется определить координаты точек возможного столкновения автомобилей.

Абстрагируя задачу, получим следующую формулировку: одна материальная точка движется по траектории, описываемой уравнением $x = 1/2$. Другая материальная точка движется по траектории, описываемой уравнением $x^2 + y^2 = 1$. Требуется найти решение системы уравнений:

$$\begin{cases} x = \dfrac{1}{2} \\ x^2 + y^2 = 1. \end{cases}$$



Графически данная задача представлена на схеме справа.

Алгоритм решения задачи:

1. Примем $x = 1/2$.

2. Подставим это значение во второе уравнение системы и вычислим

$$y_1 = \sqrt{1 - \frac{1}{4}}; \qquad y_2 = -\sqrt{1 - \frac{1}{4}}.$$

3. Получим следующее решение задачи: $\left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right)$; $\left(\frac{1}{2}, -\frac{\sqrt{3}}{2}\right)$.

Пример 2: На гладком столе находится металлический шар, прикрепленный к пружине (*рис. 1.3*). Пружина сжимается, без риска потери упругости, а затем отпускается. Требуется определить координаты шара через t секунд.

Если k – коэффициент упругости пружины, m – масса шара, x – величина упругой деформации пружины, то на основании *закона Гука* и *второго закона Ньютона* математическая модель системы *шар–пружина* имеет форму:

$ma = -kx$, где a – ускорение.

Обозначим через x_0 положение шара после сжатия пружины (первоначальная упругая деформация) (*рис. 1.4*). Требуется определить положение шара (величину упругой деформации) через t секунд. Для этого преобразуем предыдущую модель таким образом, чтобы отразить зависимость величины x (упругой деформации) от времени. Воспользуемся законом гармонического движения, предположив, что деформации в нашем случае малы и что силой трения можно пренебречь:

$$x(t) = x_0 \cos \sqrt{\frac{k}{m}} t \cdot$$

Данная формула позволяет определить положение шара x (упругую деформацию пружины) в любой момент времени t, в случае, если известны величины k, m и x_0.

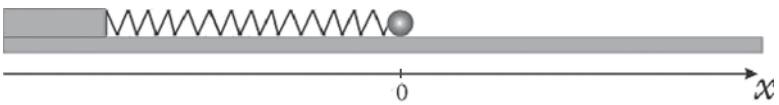


Рис. 1.3. Начальное состояние системы

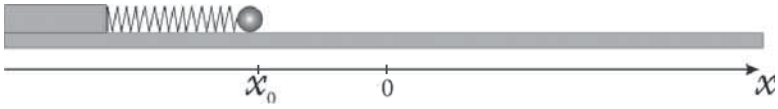


Рис. 1.4. Состояние системы после сжатия пружины

Таким образом, общая задача определения упругой деформации в любой момент времени может быть решена с помощью полученной выше формулы и следующей программы:

```
program cn01;
var
t, x0,k,m,x: real;
begin
   readln(x0,k,m,t);
   x:=x0*cos(sqrt(k/m*t));
   writeln('x=', x:0:6);
end.
```

Вопросы и упражнения

❶ Дайте определение понятию *математическая модель*. Является ли такая модель материальной?

❷ Укажите несколько областей применения математических моделей. Обоснуйте необходимость их использования.

❸ Обоснуйте необходимость преобразования математических моделей в компьютерные программы, а также необходимость применения этих программ.

❹ Напишите программу, которая, в условиях задачи из *примера 2*, будет подсчитывать упругую деформацию пружины x через каждую секунду с момента начала движения и до момента t (t имеет целое значение). Значения x_0, k, m, t считываются с клавиатуры.

❺ Радиоактивный изотоп плутония-235 имеет период полураспада, равный 26 минутам. За этот период половина начального количества изотопов исчезает, распавшись на другие химические элементы. Опишите математическую модель, позволяющую подсчитать число циклов полураспада, необходимое для исчезновения k процентов от начального количества изотопов.

❻ Эффект некоторого лекарства подсчитывается по формуле $r_k = \alpha r_{k-1} + 0{,}4^k$, где r_k – концентрация активных веществ через k часов после приема лекарства. Первоначально $r_0 = 1$ $(0 < \alpha < 1)$. Из формулы следует, что r_k достигает своего максимального значения r_m по истечении m часов, затем его значение начинает уменьшаться. Напишите программу для определения того, через сколько часов эффект от приема лекарства достигнет максимального значения. Действительное число α считывается с клавиатуры.

1.3. Аналитические и имитационные решения

Для начала проанализируем следующую задачу:

Пример 1: Бассейн объемом в 100 м^3, первоначально содержащий 20 м^3 воды, заполняется с помощью насоса мощностью 15 м^3/час. Одновременно через устройство для фильтрации из бассейна вытекают 5 м^3/час воды. Требуется определить, через сколько часов бассейн наполнится.

Несомненно, существует несколько способов решения данной задачи. Один из самых простых состоит в имитации процесса накапливания воды в бассейне через каждый час с помощью следующей таблицы.

Время (часы)	Количество накачиваемой воды	Количество вытекшей воды	Количество воды в бассейне
0	0	0	20
1	15	5	30
2	30	10	40
3	45	15	50
4	60	20	60
5	75	25	70
6	90	30	80
7	105	35	90
8	120	40	100

Решение задачи было получено путем достаточно большого числа последовательных расчетов, с помощью которых объем воды в бассейне динамически реконструировался через каждый час. Безусловно, это не самый эффективный метод решения: потребовалось большое количество промежуточных результатов, а также неоправданно большое число операций.

Другое решение основано на применении формулы, позволяющей произвести прямой расчет конечного результата. Необходимое для наполнения бассейна водой время определяется как отношение разницы общего и начального объемов воды в бассейне к количеству прироста воды в течение часа:

$$t = \frac{100 - 20}{15 - 5} = 8.$$

В первом случае для решения был использован часто повторяющийся процесс, при котором объем воды определялся через каждый интервал времени (1 час), а новый результат рассчитывался с учетом данных, полученных на предыдущем этапе. Другими словами, был смоделирован процесс этапов наполнения бассейна. Такой процесс называется *имитацией*.

Имитация является техникой решения задач, основанной на применении математических и логических моделей, которые описывают поведение реальной системы в пространстве и/или во времени.

Имитационная модель – это модель для решения задачи, основанного на технике имитации. Решения, полученные в процессе имитации, называются имитационными решениями.

Обычно, имитационные модели используют в тех случаях, когда необходимы не только конечные, но и промежуточные результаты.

Второй способ решения рассмотренной задачи основан на прямых расчетах по формуле:

$$t = \frac{V_{\text{о.б.}} - V_{\text{н.к.}}}{C_{\text{н.}} - C_{\text{с.}}},$$ где

$V_{\text{о.б.}}$ – общий объем бассейна,
$V_{\text{н.к.}}$ – начальное количество воды в бассейне,
$C_{\text{н.}}$ – мощность насоса,
$C_{\text{с.}}$ – пропускная способность стока.

Формула позволяет вычислить время наполнения *бассейна любого объема, с произвольным начальным количеством воды, насосом любой мощности, превосходящей пропускную способность стока*, которая, в свою очередь, тоже является произвольной.

Метод решения задач, основанный на применении формул, позволяющих делать прямой расчет конечного результата, не требующий анализа промежуточных состояний, называется аналитическим методом. Решения, полученные таким методом, называются аналитическими решениями.

Пример 2: Требуется написать программу для вычисления суммы первых n членов геометрической прогрессии, зная величину первого члена a_1, ($a_1 > 0$) и знаменатель прогрессии q, ($q < 1$).

Из математики известна формула $S_n = \frac{a_1(q^n - 1)}{q - 1}$, позволяющая напрямую, без нахождения всех членов прогрессии, вычислять требуемую сумму. Известна также и рекуррентная формула $a_n = q \times a_{n-1}$ для вычисления n-го члена прогрессии ($n \geq 2$).

Следовательно, необходимый результат можно получить с помощью итеративного процесса, подсчитывая по рекуррентной формуле каждый последующий член прогрессии и добавляя его к сумме $S_n = a_1 + a_2 + \ldots + a_{n-1} + a_n$.

Прямой расчет суммы

```
program cn02;
var S,a,q : real;
   n : integer;
begin
   readln(a, q, n);
   S:=a*(exp(n*ln(q))-1)/(q-1);
   writeln('S=', S:0:6);
end.
```

Итеративный расчет суммы

```
program cn03;
var S,a,q : real;
   i, n : integer;
begin
   readln(a, q, n);
   S:=0;
   for i:=1 to n do
      begin
         S:=S+a;
         a:=a*q;
      end;
   writeln('S=', S:0:6);
end.
```

Каждый из приведенных выше методов имеет свои достоинства и недостатки. В некоторых задачах бывает достаточно сложно, а иногда и невозможно определить аналитическую формулу (например, координаты кометы или астероида в зависимости от времени). В других случаях бывает достаточно сложно создать адекватную имитационную модель, даже если использовать очень большое число промежуточных расчетов.

Аналитическое решение позволяет делать прямой расчет конечного результата, однако, не позволяет исследовать динамику его построения. Использование итеративного процесса (имитационного решения) позволяет динамически конструировать решение, в зависимости от данных задачи, а также на каждом новом шаге проводить проверку полученного промежуточного результата. В то же время, для получения имитационных решений требуется значительно большее количество операций, нежели в случае аналитических решений.

Выбор метода решения задачи зависит от многих факторов. Приведем основные из них:

- возможность нахождения аналитического решения;
- необходимость исследования промежуточных решений (состояний);
- время, необходимое для расчетов (в случае имитационного решения);
- погрешность имитационного решения (разница между точным решением и решением, полученным с помощью вычислительного эксперимента).

Вопросы и упражнения

❶ Дайте определение понятию *имитация*. Что такое имитационное решение?
❷ Что понимается под *аналитическим методом* решения задачи? Что представляет собой аналитическое решение? Какими свойствами обладают *аналитические решения*?
❸ Перечислите свойства *имитационных решений*. Какие из этих свойств предполагают использование компьютера для нахождения такого типа решений?
❹ Разработайте имитационный метод для нахождения *n*-го элемента числового ряда 1, 2, 3, 5, 8, 13, 21, ... Существует ли аналитический метод решения данной задачи?
❺ Решите следующие задачи имитационным методом:
a) Божья коровка в течение дня поднимается по столбу на 5 м, а в течение ночи – опускается по нему на 3 м. Подъем она начинает утром. Высота столба равна 15 м. Через сколько дней божья коровка достигнет вершины столба?
b) В условиях предыдущей задачи божья коровка начинает движение утром, с высоты равной 6 м.
c) В условиях предыдущей задачи божья коровка начинает движение сразу после наступления ночи.
❻ Напишите программу для вычисления суммы первых *n* членов арифметической прогрессии, зная величину первого члена a_1, ($a_1 > 0$) и шаг *r*, ($r > 0$).

1.4. Этапы решения задачи на компьютере

Инструменты информатики позволяют решать задачи как аналитическими, так и имитационными методами. Решение любой задачи, независимо от применяемого метода, состоит из нескольких этапов. Каждый из этих этапов очень важен.

Анализ задачи. Это этап изучения содержания задачи. На этом этапе определяется набор начальных данных и ожидаемый результат, устанавливаются связи между начальными данными и результатом. Также тут устанавливаются дополнительные ограничения на начальные данные и результаты.

Создание математической модели задачи. На этом этапе начальные данные описываются с помощью математических структур. Описываются, применяя язык математики, отношения, позволяющие получить из начальных данных результат. В зависимости от задачи эти отношения могут быть рекуррентными (создается имитационная модель), либо позволяющими прямой расчет результата (аналитическая модель). На этом же этапе (если необходимо) задача разбивается на подзадачи. В данном случае математическая модель создается для каждой подзадачи в отдельности.

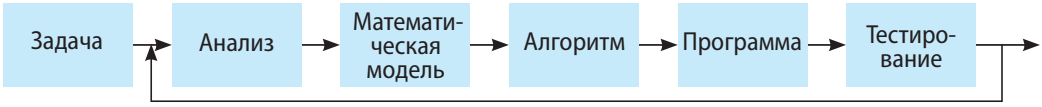
Разработка алгоритма. Алгоритм решения задачи на компьютере содержит множество команд, описанных в установленной форме (псевдокод, логическая схема и др.), которые необходимо выполнить для решения задачи, а также порядок их выполнения (шаги алгоритма). Если задача была разбита на подзадачи, то дополнительно к описанию подалгоритмов необходимо описать алгоритм, определяющий способы и условия их вызова.

Написание программы. Для того чтобы решение задачи было выполнено на компьютере, алгоритм необходимо преобразовать в форму, ему понятную – ***программу***, записанную на ***языке программирования***. Шаги алгоритма представляются в виде операторов языка программирования, а порядок выполнения определяется их

последовательностью и структурой. Начальные и промежуточные данные описываются с помощью структур данных, допустимых в языке программирования. В процессе написания программы возможны как синтаксические, так и семантические ошибки. Процесс их исправления также является частью этапа написания программы. Данный этап считается завершенным тогда, когда компиляция либо интерпретация программы проходит без ошибок.

Тестирование программы. Успешное компилирование еще не является гарантом правильного решения задачи. Для проверки правильности решения проводится ряд тестов, устанавливающих правильность результатов, генерируемых программой, в зависимости от простых, средней сложности и экстремальных начальных данных. Если для всех проведенных тестов программа генерирует правильный результат, то можно предположить, что задача решена верно. Если же в результате тестирования полученные результаты отличаются от ожидаемых, то необходимо возобновить решение задачи, начиная с первого этапа – *анализа задачи*.

Процесс решения задачи на компьютере можно проиллюстрировать с помощью следующей схемы:



Пример:

Задача:

В лабораторных условиях популяция вирусов, первоначально состоящая из N единиц и помещенная в стерильные условия, уменьшается на 50 процентов в течение каждого часа, если число вирусов на начало часа четно, либо вырастает на единицу, если это число на начало часа нечетно. В момент, когда число вирусов становится меньше числа C – критического для выживания, данная популяция полностью исчезает.

Задание: Напишите программу, с помощью которой будет определено время в часах, необходимое для полного исчезновения в лабораторных условиях популяции из N ($N < 32000$) вирусов, имеющих критическое для выживания число C ($1 < C < N$).

Анализ задачи

Популяция, состоящая из четного числа вирусов, уменьшается вдвое в течение часа. Популяция, состоящая из нечетного числа вирусов, вырастает на единцу и становится четной. Повторный рост популяции невозможен, а ее уменьшение вдвое происходит минимум один раз за каждые два часа. Следовательно, значение C ($C > 1$) будет достигнуто за конечное число часов.

Математическая модель

Число популяции в момент времени $t = 0$ задано числом $N_0 = N$.

Число популяции по истечении t часов стерилизации ($t > 0$) задается рекуррентной формулой:

$$N_t = \begin{cases} \dfrac{N_{t-1}}{2}, & \text{если } N_{t-1} \text{ четное число;} \\ N_{t-1} + 1, & \text{если } N_{t-1} \text{ нечетное число.} \end{cases}$$

Алгоритм

Шаг 0. Инициализация: Вводятся значения N, C. $t \Leftarrow 0$*.

Шаг 1.

a) По истечении одного часа: $t \Leftarrow t+1$.

b) Ремоделирование популяции: если $N \bmod 2 = 0$, $N \Leftarrow N \operatorname{div} 2$, иначе $N \Leftarrow N+1$.

Шаг 2. Проверка условия выживания: если $N < C$, выводится значение t.

КОНЕЦ. В противном случае возобновляется выполнение *шага 1*.

Программа

```
program cn04;
var N,C,t: integer;
begin
   readln(N, C); t:=0;
   while (N>=C) do begin
      t:=t+1;
      if N mod 2 = 1 then N:=N+1 else N:= N div 2;
      end;
   writeln(t);
end.
```

Тестирование

Для проверки правильности решения были использованы следующие наборы данных:

Начальные данные	Результат	Начальные данные	Результат	Начальные данные	Результат	Начальные данные	Результат
512 2	9	31999 16	15	331 330	2	332 330	1
25768 235	10	31999 2	18	31997 2	19	3 2	3

Вопросы и упражнения

❶ Перечислите этапы решения задачи на компьютере. Объясните необходимость каждого этапа.

❷ Какие методы описания алгоритма решения задачи вы знаете? Приведите примеры.

❸ Каковы последствия разбиения задачи на подзадачи? Приведите примеры задач, которые могут быть разбиты на подзадачи. Укажите две задачи, содержащие одинаковые подзадачи.

❹ Для следующих задач разработайте математические модели:

a) Известны координаты трех вершин прямоугольника со сторонами, параллельными осям системы координат. Требуется определить координаты четвертой вершины.

b) Решите предыдущую задачу в условиях, когда стороны прямоугольника могут располагаться произвольно относительно осей системы координат.

❺ Для следующих задач опишите алгоритмы решения, воспользовавшись для этого известными методами:

a) Задана последовательность целых чисел, количество которых не превосходит 100. Требуется расположить элементы последовательности в возрастающем порядке.

b) Задана последовательность целых чисел, количество которых не превосходит 100. Требуется определить за один проход элемент с максимальным значением, а также число его повторений в последовательности.

* Здесь и далее выражение $a \Leftarrow b$ означает: a получает значение b.

❻ Напишите программы для решения задач из упражнений 4 и 5.
❼ Придумайте наборы тестов для программ, написанных в упражнении 6.

Контрольный тест

Ⅰ Установите значение истинности следующих утверждений:

1. Модель является точной копией оригинала и обладает всеми его свойствами:
a) истина; b) ложь.

2. Математическая модель является идеальной моделью:
a) истина; b) ложь.

3. Математическая модель может быть использована только для решения математических задач:
a) истина; b) ложь.

4. Имитационное решение задачи позволяет прямой расчет результатов с помощью начальных данных:
a) истина; b) ложь.

5. В процессе решения задачи на компьютере описание алгоритма предшествует этапу разработки математической модели:
a) истина; b) ложь.

6. Правильный результат, полученный на одном наборе тестовых данных, гарантирует правильность результатов, выдаваемых программой, для любого другого набора данных:
a) истина; b) ложь.

Ⅱ Выберите правильный вариант определения:

1. Моделирование – это процесс:
a) использования модели;
b) построения модели;
c) представления модели;
d) разложения оригинала на элементарные части.

2. Математическая модель – это:
a) описание разговорным языком некоторого математического понятия;
b) множество геометрических характеристик некоторой материальной модели;
c) описание некоторого процесса или явления с помощью математических понятий;
d) материальная модель некоторого тела или геометрической фигуры.

3. Аналитический метод решения задачи – это метод:
a) который позволяет производить расчет конечного результата путем исследования промежуточных состояний и результатов;
b) состоящий в выборе правильного результата из конечного числа возможных решений;
c) который позволяет прямой расчет конечного результата без исследования промежуточных состояний и результатов;
d) состоящий в произвольном выборе допустимого решения.

Ⅲ Опишите этапы решения на компьютере задачи нахождения значения a^b для целых чисел $0 < a, b < 10$.

ГЛАВА 2

ПОГРЕШНОСТИ ЧИСЛЕННЫХ МЕТОДОВ

Изучив данную главу, вы сможете:

- идентифицировать точные и приближенные значения;
- вычислять абсолютные и относительные погрешности;
- определять источники погрешностей в задачах на моделирование;
- идентифицировать типы возможных погрешностей в зависимости от природы решаемых задач.

2.1. Приближенные числа. Абсолютная и относительная погрешности

Число a называется приближением числа A, если их значения отличаются незначительно и в расчетах число a может заменить A. Если имеет место отношение $a < A$, a называется приближением с недостатком, если же $a > A$ – приближением с избытком. Например, для числа π значение 3,14 является приближением с недостатком, а 3,142 – приближением с избытком. Действительно $3{,}14 < \pi < 3{,}142$. Приближенные значения возникают в результате проводимых измерений, а разница между приближенным и точным значениями может быть обусловлена множеством факторов: температурными условиями, давлением, влажностью, качеством измерительных приборов, квалификацией специалиста, проводящего измерения и др.

Инфо+

Представление чисел в компьютере предполагает использование приближенных чисел с конечным количеством цифр. Участвующие в записи числа цифры, начиная с первой, отличной от нуля, называются *значащими цифрами*.

Пример: 0,04708 имеет четыре значащих цифры. Первые два нуля лишь фиксируют положение десятичной запятой.

Приближение $a(a_n a_{n-1} a_{n-2} \ldots a_0)$ точного числа A имеет k значащих цифр $a_i a_{i-1} \ldots a_{i-k+1}$, если:

$$|A - a| \le \frac{1}{2} 10^{i-k+1}.$$

Под погрешностью Δ_a понимается разность $A-a$ (иногда и $a-A$). В зависимости от значений a и A, Δ_a может быть отрицательной либо положительной. Для получения точного числа A к приближенному числу a добавляется значение погрешности Δ_a:

$$A = a + \Delta_a.$$

Часто бывает известно приближение a и неизвестен знак приближения. В таких случаях используют *абсолютную погрешность*, определяемую следующим образом:

Абсолютная погрешнось Δ приближенного значения a – это модуль разницы между точным значением A и приближенным значением a:

$$\Delta = |A - a|.$$

Абсолютная погрешность не является достаточным показателем для оценивания точности расчетов либо измерений. Пусть, в результате измерения двух балок длиной в 20 м и 6 м, были получены, соответственно, результаты в 20,5 м и 6,2 м.

Несмотря на то, что абсолютная погрешность в первом случае больше, очевидно все же, что измерения первой балки были более точными, чем измерения второй. Для того чтобы определить качество измерений (расчетов), абсолютная величина ошибки относится к единице длины (или, в общем случае, к соответствующей единице измерения).

Относительная погрешность δ приближенного значения – это отношение абсолютной погрешности Δ к модулю точного числа A $(A \neq 0)$:

$$\delta = \frac{\Delta}{|A|}.$$

Пример: Длина балки равна 100 см. В результате ее измерения было получено значение 101 см. Расстояние между пунктами A и B равно 3000 м. В результате его измерения было получено значение в 2997 м. Необходимо определить относительную погрешность для каждого измерения и установить более точное из них.

Решение:

для первого измерения $\Delta = |100 - 101| = 1$ см, $\delta = \frac{1}{100} = 0,01$;

для второго измерения $\Delta = |3000 - 2997| = 3$ м, $\delta = \frac{3}{3000} = 0,001$.

Так как относительная погрешность второго измерения меньше, оно является более точным.

Вопросы и упражнения

1. Поясните понятие погрешности. Приведите примеры возникновения погрешностей в реальных ситуациях.
2. Дайте определение понятию *абсолютная погрешность*. Приведите примеры.
3. Дайте определение понятию *относительная погрешность*. Приведите примеры. Как изменится формула для расчета относительной погрешности, если она должна выражаться в % от исследуемого точного значения?
4. Длина пути между двумя населенными пунктами согласно дорожному указателю равна 230 км. Преодолев этот путь на автомобиле, с помощью измерительного прибора было зафиксировано расстояние в 230,7 км. Считая значение на указателе точным, определите абсолютную и относительную погрешность данного измерения.
5. Балка, точная длина которой равна 100 см, была измерена с абсолютной погрешностью, равной 2 см. Каковы возможные значения длины, полученные в результате измерений?
6. Точный объем сосуда равен 20 л. Измерения объема были проведены с относительной погрешностью, равной 0,001. Каковы возможные значения объема, полученные в результате измерений?

2.2. Источники вычислительных погрешностей

Источники погрешностей, появляющихся в процессе решения задач, очень разнообразны. Знание этих источников позволяет избежать их и, таким образом, минимизировать накопительный эффект погрешностей. Наиболее часто встречаемые типы погрешностей:

a) *погрешности задачи*;

b) *погрешности метода*;

c) *погрешности входных данных*;

d) *погрешности приближения*;

e) *погрешности округления*.

Погрешности задачи. Этот тип погрешностей появляется в результате того, что математическая модель не описывает исследуемый объект абсолютно точно. Так, в *примере 2* (1.2. Математическая модель и математическое моделирование) для построения математической модели системы *шар–пружина* было использовано уравнение гармонических колебаний в *условиях малых деформаций* и *отсутствия силы трения*. Следовательно, полученный с помощью формулы результат будет отличен от точного, и чем больше будет сила трения и деформация пружины в реальной системе, тем более значительной будет погрешность.

Погрешности метода. Этот тип погрешностей возникает в результате невозможности нахождения точного метода решения задачи, либо в результате ограничений, приводящих к необходимости использовать для решения задачи менее точные методы. В этом случае используются эвристические методы, что может привести к значительным различиям между генерируемым и точным решением задачи.

Хорошим примером для иллюстрации вышесказанного может служить решение *задачи о рюкзаке* методом Гриди (Greedy).

Вспомним!

Задача о рюкзаке:

Пусть дан рюкзак объемом S и n предметов объемами v_i, $i = 1...n$ и стоимостью c_i, $i = 1...n$. Необходимо заполнить рюкзак предметами из данного набора таким образом, чтобы суммарная стоимость положенных в рюкзак предметов была максимально возможной.

Если применить для решения задачи метод Гриди, в большинстве случаев полученное решение будет отличаться от оптимального. Так, для набора из пяти предметов объемами 5, 7, 13, 20, 10 и стоимостью, соответственно, 4, 8, 15, 23, 5, а также рюкзака объемом 30 единиц, метод Гриди сгенерирует решение, равное 27 (предметы 3, 2, 1). (Сортировка предметов производится в порядке убывания отношения стоимость/объем).

В действительности оптимальное решение равно 31 (предметы 2 и 4).

Погрешности входных данных. Математическое моделирование часто основывается на результатах, полученных опытным путем, то есть на числовых последовательностях, полученных в результате измерений. Эти значения не являются точными (например: расстояние, масса, скорость).

Пусть некоторое тело движется по траектории, описываемой на отрезке [0,1] функцией $f(x) = x^2 + x + 1$. Известно, что значение аргумента x вычисляется с абсолютной погрешностью, не превосходящей 0,01. Следовательно, если z – точное значение аргумента, то $|z - x| < 0{,}01$.

В этих условиях можно определить, в какой мере погрешность измерения величины x влияет на результаты вычислений:

$$|f(x) - f(z)| = |x^2 + x + 1 - z^2 - z - 1| =$$
$$= |x^2 + x - z^2 - z| = |(x - z)(x + z + 1)|.$$

Так как $x + z + 1 \le 3$ и $|z - x| < 0,01$, следует $|f(x) - f(z)| \le 0,03$.

Разность между значением функции для точного (z) и приближенного (x) аргументов не превосходит постоянное число. Отсюда следует, что

погрешность результата не зависит от x, а только от точности, с которой он был измерен, и функция является устойчивой к погрешностям.

Погрешности приближения. Это тип погрешностей, генерируемых некоторыми математическими определениями и понятиями. Их наличие приемлемо в задачах, использующих понятия предела, сходимости и др. Появление этого типа погрешностей обосновано самой структурой определений, содержащих в себе элементы приближения.

Пример: Последовательность $\{x_n\}$ является сходящейся и имеет предел x, если для любого $\varepsilon > 0$, $\varepsilon \to 0$, существует ранг $n(n_\varepsilon)$ такой, что $\forall\ n > n_\varepsilon\ |x - x_n| < \varepsilon$.

С точки зрения численного анализа ε отражает точность, с которой x_n приближает x.

В процессе вычисления предела используется последовательное приближение к точному пределу, который, однако, не достигается. Процесс прерывается тогда, когда разница (погрешность) становится меньше максимально допустимой погрешности (ε).

Необходимо отметить, что результаты, полученные численными методами, допустимы лишь в качестве количественных и не могут служить доказательством определенных математических высказываний.

Погрешности округления. Это особый тип погрешностей, обоснованный тем фактом, что в компьютере числовые значения, участвующие в расчетах, сохраняются с ограниченным количеством десятичных цифр после запятой. В качестве примера может служить константа π, значения тригонометрических функций и др.

Пусть $A = \{a_1, a_2, a_3, \ldots, a_n\}$ – множество всех чисел, которые могут быть представлены в компьютере. (Считается, что $a_1 < a_2 < a_3 < \ldots < a_n$.)

Любое из чисел $\frac{a_i + a_{i+1}}{2}$ отсутствует в заданном множестве A. В случае появления подобной ситуации при расчетах возникает погрешность, обусловленная заменой действительного результата – его самым близким приближением из данного множества A. Эта процедура называется *округлением*.

Вспомним!

Считается, что функция $f(x)$: $I \to R$ обладает свойством Лившица, если существует константа $m > 0$, такая, что: $|f(x) - f(z)| \le m^* |z - x|$, $\forall\ x, z \in I$.

Считается, что определение значений функции $f(x)$ устойчиво к погрешностям, если $f(x)$ обладает свойством Лившица.

Другими словами, устойчивость функции к погрешностям предполагает малые изменения значений функции при малых изменениях аргумента.

Инфо+

Приближение значений членов ряда $1/n$ к его пределу – 0.

$n = 1$	$1/n = 1$
$n = 2$	$1/n = 0{,}5$
…	
$n = 501$	$1/n = 0{,}001996$
$n = 502$	$1/n = 0{,}001992$
$n = 503$	$1/n = 0{,}001988$
…	
$n = 999$	$1/n = 0{,}001001$
$n = 1000$	$1/n = 0{,}001000$
$n = 1001$	$1/n = 0{,}000999$
…	

Пример: Пусть в процессе вычислений можно оперировать лишь тремя цифрами после запятой. В этом случае для чисел:
$a = 0{,}2334$, $b = 0{,}2331$, $c = 0{,}233$

получим $rd(rd(a+b)+c) =$
$= rd(rd(0{,}4665)+0{,}233) =$
$= rd(0{,}467+0{,}233) =$ **0,7**;

$rd(a+rd(b+c)) =$
$= rd(0{,}2334+rd(0{,}4661)) =$
$= rd(0{,}2334+0{,}466) =$
$= rd(0{,}6994) =$ **0,699**.

Функция округления, реализованная в компьютере, определена следующим образом:

$$\forall x \in R,\ a_i \in A,\ a_{i+1} \in A \quad x \in (a_i,\ a_{i+1})$$

$$rd(x) = a_i \text{ если } x \in \left(a_i, \frac{a_i + a_{i+1}}{2} \right),$$

$$rd(x) = a_{i+1} \text{ если } x \in \left[\frac{a_i + a_{i+1}}{2}, a_{i+1} \right).$$

Применение функции округления приводит к отклонениям в основных законах арифметических операций, которые уже не являются ассоциативными, дистрибутивными.

В случае округления результатов вычислений, погрешности по модулю не превосходят значение $0{,}5 \times 10^{-n}$ (эти погрешности называются *абсолютными погрешностями округления*), где n есть число значащих цифр (которые могут быть восприняты) в процессе вычислений. Погрешности округления могут быть как положительными, так и отрицательными. В случае их чередования имеет место процесс погашения, и, как результат, конечная погрешность не возрастает вместе с числом произведенных вычислений.

Вопросы и упражнения

❶ Укажите основные источники погрешностей. Приведите примеры.

❷ Приведите примеры погрешностей, обусловленных невозможностью точной формулировки задачи. Аргументируйте невозможность точной формулировки задачи.

❸ Какие функции обладают свойством устойчивости вычисляемых значений к погрешностям?

❹ Является ли устойчивым к погрешностям вычисление значений линейной функции? А функции $y = \sqrt{x},\ \ x \in [1,2]$?

❺ Объясните суть погрешностей приближения.

❻ Решение некоторой задачи вычисляется итеративно, с помощью формулы $x_0 = 0,\ \ x_{i+1} = \sqrt{2 + x_i}$. Достигнет ли точного результата последовательность вычисляемых значений? Определите опытным путем либо аналитически точное решение. Через какое количество итераций абсолютная погрешность решения станет меньше 0,0001?

❼ Почему результаты, полученные численными методами, не могут служить доказательством истинности математических высказываний?

❽ Какова причина возникновения погрешностей округления? Приведите примеры.

❾ Напишите программу, вычисляющую произведение и частное чисел 1,00000001 и 0,999999999. Значения сохраните с помощью переменных типа `real`. Проанализируйте полученные результаты. Объясните причины появления погрешностей.

Контрольный тест

1. Абсолютная погрешность Δ приближенного значения точной величины A задается формулой:

a) $\Delta = |A - a|$;
b) $\Delta = |A| - |a|$;
c) $\Delta = |a| - |A|$;
d) $\Delta = a - A$.

2. Частота вещания радиостанции равна 105,2 МГц. При настройке радиоприемника на данную радиостанцию частота вещания составила 105,25 МГц. Определите абсолютную и относительную погрешности, с которыми была установлена частота вещания. Как скажутся последствия данной погрешности на работе радиоприемника?

3. Определите количество значащих цифр для каждого из приведенных ниже чисел:

a) 0,375;
b) 0,000672;
c) –0,1233;
d) –0,0022222;
e) 0,010101.

4. Погрешности приближения – это погрешности, возникающие и результате:

a) неполной математической модели;
b) недостаточности входных данных;
c) специфики представления чисел в компьютере;
d) приближенного метода решения;
e) математических определений и понятий, содержащих элементы приближения.

5. В результатах расчетов, производимых при финансовых транзакциях, можно оперировать, максимум, двумя цифрами после запятой. Для чисел $a = 0{,}113$, $b = 0{,}162$, $c = 0{,}21$ вычислите:

a) $rd(rd(a+b)+c)$;
b) $rd(a+rd(b+c))$.

Какой тип погрешности является причиной получения различных результатов?

Тематическое исследование

Пусть дан контейнер объемом в 250 единиц и 5 предметов с объемами 120, 40, 40, 100, 150 единиц и стоимостью в 150, 60, 80, 120, 180. Требуется поместить в контейнер предметы из данного набора таким образом, чтобы их общая стоимость была максимально возможной.

a) Найдите решение задачи методом Гриди. (Упорядочивание предметов произведите по убыванию отошения стоимость/объем.)
b) Найдите точное решение задачи методом полного перебора.
c) Определите абсолютную и относительную погрешность частного решения, найденного методом Гриди.
d) Обоснуйте причины возникновения погрешностей.

ГЛАВА 3

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ АЛГЕБРАИЧЕСКИХ И ТРАНСЦЕНДЕНТНЫХ УРАВНЕНИЙ

Изучив данную главу, вы сможете:

- определять наличие решений алгебраических и трансцендентных уравнений на заданном интервале;
- изолировать интервалы в области определения функции $f(x)$, которые содержат ровно по одному корню уравнения $f(x) = 0$;
- применять алгоритмы половинного деления, хорд и касательных для решения алгебраических и трансцендентных уравнений;
- писать программы для решения алгебраических и трансцендентных уравнений методами половинного деления, хорд и касательных;
- комбинировать изученные методы с целью создания эффективных алгоритмов для решения алгебраических и трансцендентных уравнений и писать программы, реализующие эти алгоритмы.

3.1. Отделение корней алгебраических и трансцендентных уравнений

Решить алгебраическое или трансцендентное уравнение (далее просто *уравнение*) $f(x) = 0$ означает определить те значения переменной x, при которых равенство $f(x) = 0$ становится истинным. В случае несложных уравнений их решение можно относительно просто найти аналитическими способами, изучаемыми в лицейском курсе математики. Если же структура уравнения сложная, то решение уравнения становится достаточно трудной процедурой. Более того, в случаях, когда уравнение моделирует определенные реальные ситуации, явления, зависящие от нескольких параметров, а значения этих параметров известны лишь приближенно, понятие точного решения вообще теряет смысл. Поэтому бывает полезно знать и приближенные методы вычисления решения уравнений, а также алгоритмы, их реализующие.

Вспомним!

Если функция $f(x)$ записана либо может быть приведена к форме полинома, уравнение $f(x) = 0$ называется алгебраическим.

Пример: $3x-7=0$.

В противном случае, когда $f(x)$ не полиномиальна, уравнение называется ***трансцендентным***.

Пример: $\sin(2x)+\sqrt{\cos^2 x+e^x}=0$.

Пусть дано уравнение

$$f(x) = 0, \qquad (1)$$

$f(x)$ – функция, определенная и непрерывная на интервале $a \le x \le b$.

Любое значение ξ, для которого выражение $f(\xi) = 0$ является истинным, называется нулём функции $f(x)$ либо корнем уравнения $f(x) = 0$.

В дальнейшем предполагается, что уравнение (1) имеет различные (изолированные) корни, то

есть для каждого корня существует соседняя область, не содержащая других решений.

Таким образом, решение уравнений численными методами разбивается на два этапа:

1. Отделение интервалов, на которых уравнение имеет единственное решение.

2. Уменьшение, насколько это возможно, каждого из изолированных интервалов (если требуется найти все решения уравнения) либо одного из них (если надо определить лишь одно решение).

Аналитический метод. Для аналитического отделения корней можно воспользоваться свойствами производной.

Вспомним!

Теорема. Если функция $f(x)$, непрерывная на отрезке $[a, b]$, принимает на концах отрезка значения, разные по знаку ($f(a) \times f(b) < 0$), тогда на этом отрезке существует по крайней мере одна точка ξ такая, что $f(\xi) = 0$. Если на $[a, b]$ существует производная $f'(x)$, непрерывная и сохраняющая знак, тогда ξ является единственным решением уравнения $f(x) = 0$ на данном отрезке.

Если решение уравнения $f'(x) = 0$ можно легко вычислить, то для отделения корней уравнения $f(x) = 0$ необходимо:

1. Найти различные корни $a \le x_1 \le x_2 \le \ldots \le x_n \le b$ уравнения $f'(x) = 0$.

2. Считая $a = x_0$ и $b = x_{n+1}$, вычислить значения $f(x_0), f(x_1), \ldots, f(x_{n+1})$. Отрезки $[x_i, x_{i+1}]$, $i = 0, \ldots, n$, для которых $f(x_i) \times f(x_{i+1}) < 0$ содержат хотя бы одно решение уравнения $f(x)=0$.

Пример 1: Определите число решений уравнения $e^x + x = 0$:

$$f'(x) = e^x + 1; \quad f'(x) > 0 \ \forall \ x \in R.$$

Так как $\lim\limits_{x \to -\infty} f(x) = -\infty$, $\lim\limits_{x \to \infty} f(x) = \infty$, то заданное уравнение имеет единственное решение.

Пример 2: Отделите корни уравнения $x^3 - 9x^2 + 24x - 19 = 0$ на отрезке $[0, 8]$.

Решение:

$$f(x) = x^3 - 9x^2 + 24x - 19;$$
$$f'(x) = 3x^2 - 18x + 24.$$

Решив уравнение $f'(x) = 0$, получим корни $x_1 = 2, x_2 = 4$.

x	$f(x)$	Знак $f(x)$
0	−19	−
2	1	+
4	−3	−
8	109	+

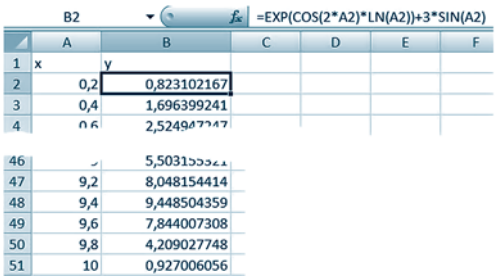
Таким образом, начальное уравнение имеет 3 корня, по одному на каждом из отрезков $[0, 2]$, $[2, 4]$, $[4, 8]$.

Графический метод. Другая возможность отделения корней уравнения $f(x) = 0$ состоит в прямом исследовании графика функции $f(x)$. Для его построения можно использовать как специальные программные приложения[1], так и простые программы, написанные средствами языка програмирования высокого уровня.

[1] MatCAD, Matematica, Derive и др.

Графический метод разделения корней уравнения на заданном отрезке можно реализовать и локально, с помощью электронных таблиц. Для этого достаточно построить таблицу с двумя столбцами. Первый столбец будет представлять собой разбиение отрезка на элементарные отрезки равной длины. Второй столбец будет содержать формулу, которая вычисляет значения функции $f(x)$ для соответствующих значений из первого столбца. На основании данных из столбца, содержащего значения $f(x)$, строится линейная диаграмма, которая является графиком исследуемой функции.

Пример: $f(x) = x^{\cos(2x)} + 3\sin(x)$. Исследуется наличие корней на отрезке [0.2, 10] (*рис. 3.1, а*; *3.1, б*).



	A	B
1	x	y
2	0,2	0,823102167
3	0,4	1,696399241
4	0,6	2,524947247
46		5,503155
47	9,2	8,048154414
48	9,4	9,448504359
49	9,6	7,844007308
50	9,8	4,209027748
51	10	0,927006056

Рис 3.1, а. Функция *f*(*x*) представлена таблично в электронной таблице. Столбец A содержит значения *x* от 0,2 до 10 с шагом 0,2. Столбец B содержит значения *f*(*x*)

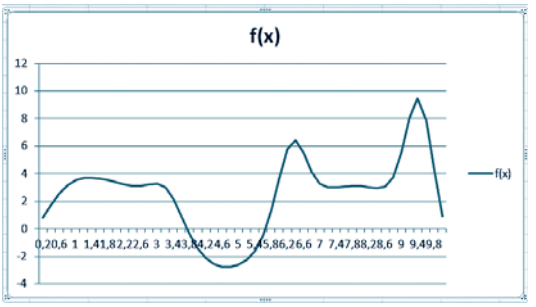


Рис 3.1, б. Функция *f*(*x*) представлена графически на основании данных таблицы. Легко идентифицируются два произвольно выбранных отрезка, каждый из которых содержит ровно по одному корню уравнения *f*(*x*) = 0, например: [3, 5] и [5, 6]

Вопросы и упражнения

❶ Что называется решением уравнения?

❷ Каким условиям должна удовлетворять функция $f(x)$ для того, чтобы на заданном отрезке был хотя бы один корень уравнения $f(x)=0$? А для того, чтобы этот корень был единственным?

❸ Определите аналитически количество действительных корней уравнения:

a) $x^5 - 5x + 7 = 0$; b) $x^3 - 9x^2 + 24x - 13 = 0$.

❹ Напишите программу, которая для функции $f(x)$, непрерывной на $[a, b]$, производит разбиение заданного отрезка на n равных по длине отрезков и выводит в качестве результата все отрезки, на концах которых функция имеет противоположные по знаку значения:

a) $f(x) = x^3 - 7x^2 + 12x - 37$ на $[-10, 10]$, $n = 100$;

b) $f(x) = \sin(3x) + 3\cos(x) - 1$ на $[-2, 2]$, $n = 100$.

❺ Отделите наиболее подходящим способом корни уравнений:

a) $\frac{x^4}{4} + x^3 - \frac{x^2}{2} - 3x - 8 = 0$;

b) $2x^3 - 6x^2 - 48x + 17 = 0$;

c) $x^4 - 14x^2 - 24x - 4 = 0$;

d) $\frac{1}{2} - e^{-x^2} = 0$;

e) $e^x(x^2 - 2x + 2) = 0$;

f) $x[\sin(\ln(x)) - \cos(\ln(x))] = 0$;

g) $\ln(x) + \sqrt{\sin(3x) + 7} - x^2 = 0$.

3.2. Метод половинного деления

Пусть дана функция $f(x)$, непрерывная на отрезке $[a, b]$, и $f(a) \times f(b) < 0$. Требуется определить на отрезке $[a, b]$ одно из решений уравнения

$$f(x) = 0. \qquad (1)$$

Свойства функции гарантируют существование хотя бы одного корня на данном отрезке.

Одним из самых простых методов определения корня уравнения $f(x) = 0$ является *метод половинного деления*. Метод предполагает нахождение значения c – середины отрезка $[a, b]$, затем вычисление значения $f(c)$. Если $f(c) = 0$, тогда c является точным решением уравнения.

В противном случае решение ищется на одном из отрезков $[a, c]$ или $[c, b]$. Оно принадлежит тому отрезку, на концах которого функция имеет противоположные по знаку значения (*рис. 3.2*).

Если $f(a) \times f(c) > 0$, то решение далее ищется на отрезке $[a_1, b_1]$, где a_1 получает значение c, а b_1 – значение b. В противном случае a_1 получает значение a, b_1 – значение c. Процесс деления пополам возобновляется для отрезка $[a_1, b_1]$ и повторяется до тех пор, пока не будет получено точное решение, либо (в подавляющем большестве случаев!) отклонение вычисленного решения c_i от точного станет достаточно малым.

В процессе подряд идущих разбиений получается последовательность отрезков

$[a_0,b_0]$, $[a_1, b_1]$, $[a_2, b_2]$, ..., $[a_i, b_i]$, ...

Для каждого из них имеет место отношение

$$f(a_i) \times f(b_i) < 0,\ i = 0, 1, 2, \ldots . \qquad (2)$$

Оценка погрешности. Так как точное решение ξ уравнения является точкой отрезка $[a_i, b_i]$, то следует, что разность между точным и вычисленным решениями не превосходит длину отрезка. Таким образом, локализация решения на отрезке величиной в ε гарантирует погрешнось вычисления решения, не превосходящую значение ε:

$$|\xi - c_i| < \varepsilon = |b_i - a_i|.$$

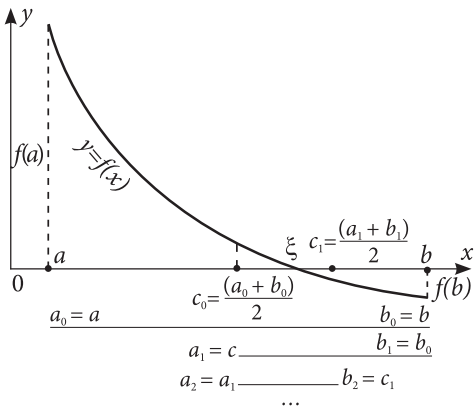


Рис. 3.2. Последовательное вычисление отрезков, содержащих решение уравнения $f(x)=0$

АЛГОРИТМИЗАЦИЯ МЕТОДА

Исходя из математического описания метода половинного деления, можем выделить два разных случая остановки процесса вычислений решения уравнения $f(x) = 0$:

А1. ***Алгоритм вычисления решения для заданного числа последовательных делений пополам:***

Шаг 0. Инициализация: $i \Leftarrow 0$.

Шаг 1. Нахождение середины отрезка $c \Leftarrow \dfrac{a+b}{2}$.

Шаг 2. Уменьшение отрезка, содержащего решение: если $f(c) = 0$, то найдено решение $x = c$. КОНЕЦ.

В противном случае, если $f(a) \times f(c) > 0$, то $a \Leftarrow c$; $b \Leftarrow b$, иначе $a \Leftarrow a$; $b \Leftarrow c$.

Шаг 3. $i \Leftarrow i + 1$. Если $i = n$, то вычисленное решение равно $x = \dfrac{a+b}{2}$. КОНЕЦ.

В противном случае возвращаемся к *шагу 1*.

А2. ***Алгоритм вычисления с заданной точностью*** [2] **ε:**

Шаг 1. Определение середины отрезка $c \Leftarrow \frac{a+b}{2}$.

Шаг 2. Если $f(c) = 0$, то найдено решение $x = c$. КОНЕЦ.
В противном случае, если $f(a) \times f(c) > 0$, то $a \Leftarrow c$; $b \Leftarrow b$, иначе $a \Leftarrow a$; $b \Leftarrow c$.

Шаг 3. Если $|b - a| < \varepsilon$, то вычислено решение $x = \frac{a+b}{2}$ КОНЕЦ.
В противном случае возвращаемся к *шагу 1*.

Пример 1: Определите корень уравнения $x^4 + 2x^3 - x - 1 = 0$ на отрезке [0, 1], произведя 16 последовательных делений пополам.

Так как число последовательных приближений задано, а концы отрезка известны, необходимые присваивания значений переменным производятся в программе.

```
program cn05;
var  a,b,c: real;
     i,n:integer;

function f(x:real):real;
begin f:=sqr(sqr(x))+2*x*sqr(x)-x-1;end;
begin a:=0; b:=1; n:=16;
      for i:=1 to n do
        begin c:=(b+a)/2;
              writeln('i=',i:3,' x=',c:10:8,' f(x)=',f(c):12:8);
              if f(c)=0 then break[3]
              else if f(c)*f(a)>0 then a:=c else b:=c;
                end;
end.
```

Результаты:

```
i= 1 x=0.50000000 f(x)= -1.18750000
i= 2 x=0.75000000 f(x)= -0.58984375
...
i= 15 x=0.86679077 f(x)= 0.00018565
i= 16 x=0.86677551 f(x)= 0.00009238
```

Пример 2: Определите корень уравнения $6\cos(x) + 8\sin(x) = 0$ на отрезке [2, 4] с точностью $\varepsilon = 0{,}00017$.

```
program cn06;
var  a,b,c,eps: real;

function f(x:real):real;
begin f:=6*cos(x)+8*sin(x);end;

begin a:=2; b:=4; eps:=0.00017;
      repeat
        c:=(b+a)/2;
        writeln('x=',c:10:8,' f(x)=',f(c):12:8);
        if f(c)=0 then break
        else if f(c)*f(a)>0 then a:=c else b:=c;
     until abs(b-a)<eps;
end.
```

[2] В данном контексте точность ε означает погрешность вычислений, которая не превосходит значение ε.

Результаты:

```
x=3.00000000 f(x)= -4.81099492
x=2.50000000 f(x)= -0.01908454
…
x=2.49829102 f(x)= -0.00199471
x=2.49816895 f(x)= -0.00077401
```

Вопросы и упражнения

❶ В каких случаях применяются приближенные методы решения алгебраических уравнений?
Опишите метод половинного деления. Какими преимуществами он обладает? А недостатками?
Формула для оценивания погрешности, приведенная в данном параграфе, имеет вид $|\xi - c_i| < \varepsilon = |b_i - a_i|$.
Выразите разность между точным и вычисленным значениями только через значения концов исходного отрезка и число осуществляемых делений пополам.

❷ Опишите алгоритм метода половинного деления.

❸ Определите методом половинного деления решения уравнений:

1. $e^x - x^2 = 0$ на [–1, –0,5];
2. $x^3 - x - 1 = 0$ на [1, 2];
3. $x^3 + 3x^2 - 3 = 0$ на [–3, –2];
4. $x^5 - x - 2 = 0$ на [1, 2].

a) для 10, 20, 40 делений начального отрезка;
b) с точностью ε = 0,001; 0,0001; 0,00001;
c) в условиях предыдущего задания определите число делений пополам, необходимое для достижения требуемой точности.

❹ Определите методом половинного деления решения на отделенных интервалах уравнений, заданных в упражнениях 3, 4, 5, стр. 24, для 10, 20, 30 последовательных делений пополам.

3.3. Метод хорд

Метод половинного деления, несмотря на его простоту, является малоэффективным в случаях, когда решение необходимо получить за небольшое количество итераций, но с большой точностью. В таких случаях более пригоден *метод хорд*, состоящий в разбиении начального отрезка на части, определяемые с помощью точки пересечения хорды – отрезка, соединяющего точки, соответствующие значениям функции на концах отрезка, с осью $0x$.

Пусть дана функция $f(x)$, обладающая следующими свойствами:

1. $f(x)$ непрерывна на отрезке $[a, b]$ и $f(a) \times f(c) < 0$.

2. На отрезке $[a, b]$ существуют $f'(x) \neq 0; f''(x) \neq 0$, непрерывные и сохраняющие свой знак на отрезке $[a, b]$.

Перечисленные свойства гарантируют существование единственного решения уравнения $f(x) = 0$ на $[a, b]$.

Метод хорд предполагает выбор в качестве приближения решения точки пересечения прямой, проходящей через точки $(a, f(a))$ и $(b, f(b))$ с осью $0x$.

Для реализации метода выбирается конец отрезка $[a, b]$, через который будут проведены несколько хорд (*рис. 3.3*). Выбор данного конца определяется условием:

$$f(e) \times f''(e) > 0.$$

Другой конец отрезка $[a, b]$ считается начальным приближением решения: x_0.

Через точки $(e, f(e))$ и $(x_0, f(x_0))$ проводится хорда. Определяется точка x_1, в которой хорда пересекает ось $0x$. Точка x_1 считается следующим приближением решения.

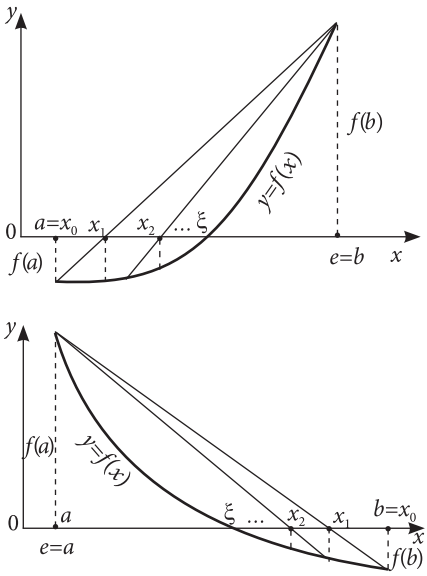


Рис. 3.3. Последовательное приближение к решению уравнения методом хорд

Вспомним!

Теорема Лагранжа

Пусть $f:[a, b] \to R$ – непрерывная и дифференцируемая на отрезке $[a, b]$. Тогда существует значение $c \in (a, b)$, такое, что:

$$f(b) - f(a) = (b-a) f'(c).$$

Процесс повторяется, следующая хорда проходит через точки $(e, f(e))$ и $(x_1, f(x_1))$. Таким образом получается последовательность приближений x_0, x_1, x_2, ... x_i, x_{i+1}, ... x_n ..., предел которой является точным решением уравнения $f(x) = 0$.

Точки e и x_0 известны. Воспользовавшись уравнением прямой, проходящей через две точки, можем определить приближение x_1 ($f(x_1) = 0$):

$$\frac{x - x_0}{e - x_0} = \frac{y - f(x_0)}{f(e) - f(x_0)},$$

откуда

$$x_1 = x_0 - \frac{f(x_0)}{f(e) - f(x_0)}(e - x_0).$$

В общем случае, зная приближение x_{i-1}, можем вычислить следующее приближение x_i с помощью рекуррентной формулы:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f(e) - f(x_{i-1})}(e - x_{i-1}), \qquad (3)$$

$$i = 1, 2, \ldots .$$

Доказано, что последовательность значений x_1, x_2, ... x_i, x_{i+1}, ... x_n ..., вычисленных с помощью формулы (3), сходится к решению ξ уравнения $f(x) = 0$.

Погрешность метода

Из того факта, что последовательность приближений, полученных методом хорд, сходится к точному решению уравнения, можно сделать следующее заключение: погрешность вычисленного решения обратно пропорциональна количеству сделанных итераций. Однако можно вывести и формулу, позволяющую оценить погрешность вычислений.

Пусть $f(x)$ удовлетворяет условиям (1), (2). Если ξ – точное решение уравнения $f(x) = 0$ на отрезке $[a, b]$, а M_1 и m_1 – супремум и инфимум первой производной $f'(x)$ на этом же отрезке, то из теоремы Лагранжа и рекуррентной формулы для вычисления последовательных приближений следует:

$$|\xi - x_i| \le \left|\frac{M_1 - m_1}{m_1}\right| \times |x_i - x_{i-1}| \text{ или } \left|\frac{M_1 - m_1}{m_1}\right| \times |x_i - x_{i-1}| \le \varepsilon. \qquad (4)$$

Следовательно, если требуется вычислить решение с заданной точностью ε, расчеты по формуле (3) будут повторяться до тех пор, пока неравенство (4) не станет истинным.

АЛГОРИТМИЗАЦИЯ МЕТОДА

Применение метода хорд диктует необходимость предварительного исследования функции $f(x)$ с целью установления фиксированного конца отрезка, через который будут проведены все хорды. Число приближений n может быть задано в условии задачи либо получено из некоторых условий.

В обоих случаях вычисления проводятся по формуле (3). В первом случае критерием остановки будет повторение расчетов по формуле n раз; во втором случае – удовлетворение условию (4).

Определение фиксированного конца. Чтобы избежать необходимости вычисления $f''(x)$, поступим следующим образом: определим знак $f(x)$ в c – точке пересечения прямой, проходящей через точки $(a, f(a))$ и $(b, f(b))$, с осью $0x$. **Фиксированным будет такой конец e отрезка $[a, b]$, для которого выполняется условие: $f(e) \times f(c) < 0$.**

А1. ***Алгоритм вычисления для случая заданного числа последовательных приближений***:

Шаг 1. Определение фиксированного конца e и начального приближения x_0:

$$c \Leftarrow a - \frac{f(a)}{f(b) - f(a)}(b - a);$$

если $f(c) \times f(a) < 0$, тогда $e \Leftarrow a$, $x_0 \Leftarrow b$, иначе $e \Leftarrow b$, $x_0 \Leftarrow a$; $i \Leftarrow 0$.

Шаг 2. Вычисление x_{i+1} согласно формуле $x_{i+1} = x_i - \frac{f(x_i)}{f(e) - f(x_i)}(e - x_i)$.

Шаг 3. Если $i + 1 = n$, тогда вычисленное решение $x \Leftarrow x_i$. КОНЕЦ.
В противном случае, $i \Leftarrow i+1$ и возвращаемся к *шагу 2*.

А2. ***Алгоритм вычисления для случая заданной точности ε:***

Так как в формуле оценки погрешности фигурируют значения M_1 и m_1, то в случае, если эти значения не заданы в условии задачи, необходимо аналитически описать функцию $f'(x)$ и определить ее M_1 и m_1.

Шаг 1. Определение фиксированного конца e и начального приближения x_0:

$$c \Leftarrow a - \frac{f(a)}{f(b) - f(a)}(b - a);$$

если $f(c) \times f(a) < 0$, то $e \Leftarrow a$, $x_0 \Leftarrow b$, иначе $e \Leftarrow b$, $x_0 \Leftarrow a$; $i \Leftarrow 0$.

Шаг 2. Вычисление x_{i+1} согласно формуле $x_{i+1} = x_i - \frac{f(x_i)}{f(e) - f(x_i)}(e - x_i)$.

Шаг 3. Если $\left|\frac{M_1 - m_1}{m_1}\right| \times |x_{i+1} - x_i| \le \varepsilon$, то вычисленное решение $x \Leftarrow x_i$. КОНЕЦ.

В противном случае, $i \Leftarrow i+1$ и возвращаемся к *шагу 2*.

Пример 1: Пусть дана функция $f(x) = \ln(x \sin x)$. Необходимо вычислить решение уравнения $f(x) = 0$ на отрезке $[0{,}5; 1{,}5]$ методом хорд, произведя 10 последовательных приближений.

Для этого примера нет необходимости в предварительных математических выкладках.

Так как концы отрезка и число необходимых приближений заданы, соответствующие присваивания будут проведены непосредственно в тексте программы.

```
program cn07;
var   a,b,e,c,x: real;
      n,i: integer;

function f(x:real):real;
begin f:=ln(x*sin(x));end;
begin a:=0.5; b:=1.5; n:=10;
      {определение фиксированного конца e и начального приближения x0}
```

```
      c:=a-(f(a))/(f(b)-f(a))*(b-a);
      if f(c)*f(a)>0 then begin e:=b; x:=a; end
      else begin e:=a; x:=b; end;
        {итеративное вычисление решения}
        for i:=1 to n do
           begin x:= x-(f(x))/(f(e)-f(x))*(e-x);
        writeln(x:10:8,' ',f(x):12:8);
           end;
end.
```

Результаты:

```
i= 1 x=1.27995775 f(x)= 0.20392348
i= 2 x=1.18251377 f(x)= 0.09028687
...
i= 9 x=1.11427651 f(x)= 0.00016577
i= 10 x=1.11420523 f(x)= 0.00006678
```

Пример 2: Пусть дана функция $f(x) = x^4 - 3x^2 + 7{,}5x - 1$. Необходимо вычислить приближенное решение уравнения $f(x) = 0$ на отрезке $[-0{,}5;\ 0{,}5]$ с точностью $\varepsilon = 0{,}0001$, используя метод хорд. Для данной функции на отрезке $[-0{,}5;\ 0{,}5]$ M_1 и m_1 равны, соответственно, 10 и 5.

Для простоты, необходимые присваивания будут реализованы непосредственно в тексте программы.

```
program cn08;
var
   Msup,minf,a,b,e,x,xnou,xvechi,eps: real;
function f(x:real):real;
begin
   f:=sqr(sqr(x))-3*sqr(x)+7.5*x-1;
end;
begin
   a:=-0.5; b:=0.5; eps:=0.0001;
   Msup:=10; minf:=5;
   {определение фиксированного конца и начального приближения}
   x:=a-(f(a))/(f(b)-f(a))*(b-a);
   if f(x)*f(a)>0 then begin e:=b; xnou:=a; end
   else begin e:=a; xnou:=b; end;
   {итеративное вычисление решения}
   repeat
   xvechi:=xnou;
   xnou:= xvechi-(f(xvechi))/(f(e)-f(xvechi))*(e-xvechi);
   writeln(' x=',xnou:10:8,' f(x)=',f(xnou):12:8);
   until abs((Msup-minf)/minf*(xnou-xvechi))<eps;
end.
```

Результаты:

```
x=0.22500000 f(x)= 0.53818789
x=0.15970438 f(x)= 0.12191694
...
x=0.14130134 f(x)= 0.00026052
x=0.14127062 f(x)= 0.00005579
```

Вопросы и упражнения

❶ Объясните суть *метода хорд*. Опишите метод хорд графически.

❷ Как зависит фиксированный конец отрезка от знака $f''(x)$?

❸ Опишите процесс определения фиксированного конца отрезка. Как можно избежать определения $f''(x)$?

❹ Опишите пошаговый алгоритм метода хорд для случая заданного числа итераций.

❺ Определите решения уравнений методом хорд для 10, 20, 30 итераций:

a) $x^3 - 0{,}2x^2 + 0{,}2x - 2{,}1 = 0$ на $[1, 2]$;
b) $5x^3 - 20x + 3 = 0$ на $[0, 1]$;
c) $e^x - x^2 = 0$ на $[-1, 0]$.

❻ Отделите корни следующих уравнений. Решите уравнения методом хорд:

a) $\mathrm{tg}(0{,}55x + 0{,}1) - x^2 = 0$ для 5, 25 итераций;
b) $x^3 - 0{,}2x^2 + 0{,}5x + 1{,}5 = 0$ для 3, 9, 27 итераций.

❼ Определите методом хорд, на отделенных отрезках, решения уравнений из упражнений 3, 4, 5, стр. 24, для 10, 20, 30 итераций. Сравните результаты с полученными в упражнении 4, стр. 27. Объясните расхождения.

❽ Пусть дана функция $f(x) = x[\sin(\ln(x)) - \cos(\ln(x))]$. Определите решение уравнения $f(x) = 0$ на отрезке [2, 3] с точностью $\varepsilon = 0{,}0001$, применив метод хорд.

3.4. Метод Ньютона

Пусть дана функция $f(x)$, обладающая следующими свойствами:

1. $f(x)$ непрерывна на отрезке $[a, b]$ и $f(a)\,f(b) < 0$.

2. На отрезке $[a, b]$ существуют $f'(x) \neq 0, f''(x) \neq 0$, непрерывные и сохраняющие знак на $[a, b]$.

Необходимо решить уравнение $f(x) = 0$ для $x \in [a, b]$. Попытаемся решить задачу путем последовательного построения нескольких касательных к графику функции. Первую касательную проведем через точку $E_0\,(x_0, y_0)$ – конец отрезка $[a, b]$, для которого выполняется условие: $f(x_0) \times f''(x_0) > 0$.

Пусть касательная с номером i пересекает ось $0x$ в точке x_i (*рис. 3.4*). Следующая касательная $(i+1)$проходит через точку E_{i+1} с координатами $(x_i, f(x_i))$ и пересекает ось абсцисс в точке x_{i+1}. Получим последовательность значений $x_0, x_1, x_2, \ldots, x_i, x_{i+1}, \ldots$, сходящуюся к точному решению уравнения $f(x) = 0$. Этот метод решения уравнения $f(x) = 0$ носит название *метода касательных* или *Ньютона* в честь математика, который его изобрел.

Для того чтобы вычислить значения x_1, x_2, … x_i, …, воспользуемся уравнением касательной к функции, проходящей через заданную точку:

$$y - f(x_i) = f'(x_i)(x - x_i). \qquad (5)$$

В общем случае уравнение (5) представляет касательную к функции $f(x)$, проходящую через точку $(x_i, f(x_i))$. Она пересекает ось абсцисс в точке с координатами $(x_{i+1}, 0)$. Отсюда получим:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \qquad (6)$$

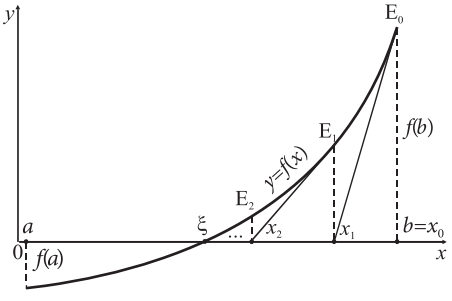


Рис. 3.4. Сходимость последовательности значений $x_0, x_1, x_2, \ldots, x_i, x_{i+1}, \ldots$ к точному решению ξ

Погрешность метода

Итеративный процесс вычисления может быть прерван либо, будучи повторен заданное число раз, либо, при достижении требуемой точности.

Погрешность метода можно оценить с помощью формулы:

$$\varepsilon = |\xi - x_{i+1}| \le \frac{M_2}{2m_1}(x_{i+1} - x_i)^2, \qquad (7)$$

где

x_i, x_{i+1} – два последовательных приближения вычисляемого решения;

M_2 – супремум функции $f''(x)$ на $[a, b]$;

m_1 – инфимум $f'(x)$ на $[a, b]$.

АЛГОРИТМИЗАЦИЯ МЕТОДА

Число последовательных приближений может быть либо вычислено априори, либо определено из некоторого условия. В обоих случаях сначала устанавливаем, какой конец отрезка будет начальным приближением. Вычисление следующего приближения производим по формуле (6). В первом случае условием остановки итеративного процесса будет его повторение заданное количество раз, а во втором – выполнение условия (7).

А1. ***Алгоритм вычислений для заданного количества последовательных приближений:***

Для реализации этого алгоритма достаточно знать аналитическое описание функций $f(x)$ и $f'(x)$. Если описание $f'(x)$ не дается в условии задачи, то его необходимо вывести математически. Начальное приближение определяется подобно тому, как это было сделано в методе хорд.

Шаг 1. Определение начального приближения x_0: $c \Leftarrow a - \dfrac{f(a)}{f(b)-f(a)}(b-a)$;

если $f(c) \times f(a) < 0$, то $x_0 \Leftarrow a$ иначе $x_0 \Leftarrow b$; $i \Leftarrow 0$.

Шаг 2. Вычисляется x_{i+1} по формуле $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$.

Шаг 3. Если $i+1 = n$, то вычисленное решение $x \Leftarrow x_{i+1}$. КОНЕЦ.

В противном случае $i \Leftarrow i+1$, затем возвращаемся к *шагу 2*.

А2. ***Алгоритм вычислений для заданной точности ε:***

В формуле оценки погрешности фигурируют значения M_2 и m_1. Если эти значения не указаны в условии задачи, то их необходимо вычислить, проведя необходимые математические выкладки. Кроме того, необходимы аналитические описания функций $f(x)$ и $f'(x)$.

Шаг 1. Определение начального приближения x_0: $c \Leftarrow a - \dfrac{f(a)}{f(b)-f(a)}(b-a)$;

если $f(c) \times f(a) < 0$, то $x_0 \Leftarrow a$ иначе $x_0 \Leftarrow b$; $i \Leftarrow 0$.

Шаг 2. Вычисляется x_{i+1} по формуле $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$.

Шаг 3. Если $\dfrac{M_2}{2m_1}(x_{i+1} - x_i)^2 \le \varepsilon$, то вычисленное решение $x \Leftarrow x_{i+1}$. КОНЕЦ.

В противном случае $i \Leftarrow i+1$, затем возвращаемся к *шагу 2*.

Пример 1[3]**:** Пусть дана функция $f(x) = x^3 - 2x^2 + x - 3$. Требуется написать программу вычисления решения уравнения $f(x) = 0$ на отрезке [2; 15] для 10 последовательных приближений методом Ньютона.

Предварительная математическая подготовка – определим $f'(x)$:

$$f(x) = x^3 - 2x^2 + x - 3; \qquad f'(x) = 3x^2 - 4x + 1.$$

Так как число приближений изначально известно и концы отрезка заданы, все необходимые присваивания включим прямо в тело программы.

```
program cn09;
var a, b, x, c : real;
   i, n: integer;
function f(z:real):real;
     begin f:=z*z*z-2*z*z+z-3; end;
function fd1(z:real):real;
     begin fd1:=3*z*z-4*z+1; end;
begin   a:=2.1; b:=15; n:=10; i:=0;
        c:=a-(f(a))/(f(b)-f(a))*(b-a);
        if f(c)*f(a)<0 then x:=a else x:=b;
        while i<n do
        begin   i:=i+1;
                x:=x-f(x)/fd1(x);
        writeln('i=',i:2,' x=',x:15:12, ' f=',f(x):15:12);
        end;
end.
```

Результаты:

```
i= 1 x= 10.23214285700   f=869.11072454000
i= 2 x= 7.06207637180    f=256.52261987000
...
i= 9 x= 2.17455942470    f= 0.00000009329
i=10 x= 2.17455941030    f= 0.00000000001
```

Пример 2: Пусть дана функция $f(x) = \cos^2(x) - \frac{x}{4}$. Требуется написать программу для нахождения приближенного решения уравнения $f(x)=0$ на отрезке [2,4;3] с точностью ε=0,0001 методом Ньютона. Для данной функции на отрезке [2,4; 3] значения M_2 и m_1, соответственно, равны 2 и 0,03.

Предварительная математическая подготовка: $f(x) = \cos^2(x) - \frac{x}{4}$; $f'(x) = -\sin(2x) - \frac{1}{4}$.

Так как ε задано, концы отрезка и значения M_2, m_1 – известны, все начальные присваивания производятся прямо в тексте программы.

```
program cn10;
var a, b, xn, xv, M2, m1, e, c : real;
   function f(z:real):real;
   begin f:=cos(z)*cos(z)-z/4;  end;
```

[3] Для всех предложенных примеров и упражнений для функций $f(x)$ предполагается выполнение условий 1 и 2 (стр. 31).

```
function fd1(z:real):real;
   begin fd1:=-sin(2*z)-1/4;  end;
begin    a:=2.4; b:=3; M2:=2; m1:=0.03; e:=0.0001;
         c:=a-(f(a))/(f(b)-f(a))*(b-a);
         if f(c)*f(a)<0 then begin
         xn:=a; xv:=b;
         end
         else begin xn:=b; xv:=a; end;
   while M2*sqr(xn-xv)/(2*m1)>e do
   begin   xv:=xn;
           xn:=xv-f(xv)/fd1(xv);
           writeln(' x=',xn:15:12, ' f=',f(xn):15:12);
   end;
end.
```

Результаты:

```
x= 2.47538619170 f= -0.00078052066
x= 2.47646766320 f= -0.00000027700
x= 2.47646804730 f= 0.00000000000
```

Вопросы и упражнения

❶ Опишите геометрический смысл *метода Ньютона*.

❷ Каким образом для метода Ньютона может быть определено начальное приближение?

❸ Обьясните, почему при правильном выборе начального приближения последовательность приближений, полученная методом Ньютона, сходится к точному решению уравнения.

❹ Напишите программу для определения на заданных отрезках решения приведенных ниже уравнений. Расчеты проводятся для 2, 4, 6 итераций. Применяется метод Ньютона:

a) $4x^4 + 8x^3 - 3x^2 - 7x + 3 = 0$ на $[-1{,}7; -1{,}58]$, $[-1{,}53; -1{,}4]$, $[0{,}4; 0{,}52]$, $[0{,}58; 0{,}8]$;

b) $\sqrt{x + \sqrt{x + \sqrt{x}}} - 12 = 0$ на $[100; 150]$;

c) $(2 - x^2)\cos(x) + 2x\sin(x) = 0$ на $[-4{,}5; -4]$ $[4; 4{,}5]$;

d) $\ln\left(\frac{1}{x} + \ln\left(\frac{1}{x} + \ln\frac{1}{x}\right)\right) - 2 = 0$ на $[0{,}1; 0{,}5]$.

❺ Модифицируйте написанную в упражнении 4 программу таким образом, чтобы расчеты решения проводились с точностью ε = 0,00001 и примените ее для решения следующих уравнений:

a) $2\cos^2(x) - e^{x/2} = 0$ на $[0{,}1; 0{,}74]$, M_2=4, m_1 =0,5;

b) $x^5 - 4x + 9 = 0$ на $[-2; -1]$, M_2=150, m_1 = 1.

❻ Пусть дана функция $f(x) = \sin^2(x) - \frac{x}{2}$. Вычислите приближенное решение уравнения $f(x) = 0$ на отрезке [0,5; 0,7] с точностью ε = 0,00001, применив метод Ньютона.

❼ Отделите корни, а затем найдите решения уравнений методом Ньютона с точностью ε = 0,00001:

a) $x^5 - 80\,x^2 + 89 = 0$; b) $e^x - x^2 = 0$.

Контрольный тест

I Выберите правильный ответ:

1. Решить уравнение $f(x) = 0$ означает найти точки:
a) пересечения графика $f(x)$ с осью $0y$;
b) пересечения графика $f(x)$ с осью $0x$;
c) в которых $f(x)$ не определена.

2. Отделить корни уравнения $f(x) = 0$ означает определить:
a) отрезок на оси $0x$, который содержит все решения уравнения $f(x) = 0$;
b) все отрезки на оси $0x$, содержащие ровно по одному решению уравнения $f(x) = 0$;
c) все отрезки на оси $0x$, обладающие свойством не содержать ни одного решения уравнения $f(x) = 0$;
d) область определения функции $f(x)$.

3. Метод половинного деления для решения алгебраических и трансцендентных уравнений основан на разбиении отрезка, содержащего решение, на части:
a) равные;
b) пропорциональные, определяемые с помощью последовательно проводимых хорд;
c) пропорциональные, определяемые с помощью последовательно проводимых касательных;
d) пропорциональные, определяемые с помощью последовательно проводимых хорд и касательных.

4. Рекуррентная формула вычисления приближения x_i решения уравнения $f(x) = 0$ методом хорд записывается так:

a) $x_{i+1} = x_i + \dfrac{f(x_i)}{f(e) - f(x_i)} \quad (e - x_i)$;

b) $x_{i+1} = x_i - \dfrac{f(x_i)}{f(e) - f(x_i)} \quad (e - x_i)$;

c) $x_{i+1} = x_i - \dfrac{f(e) - f(x_i)}{f(x_i)} \quad (e - x_i)$;

d) $x_{i+1} = x_i - \dfrac{f(x_i)}{f(e) + f(x_i)} \quad (e - x_i)$.

5. Начальным приближением для метода касательных может быть такой конец e отрезка $[a, b]$, содержащий решение, для которого справедливо отношение:
a) $f(e) \times f''(e) < 0$;
b) $f(e) \times f''(e) > 0$;
c) $f(e) \times f''(e) = 0$;
d) $f(e) \times f''(e) \neq 0$.

II Вычислите, воспользовавшись ранее написанными программами:

1. Решение уравнения $\dfrac{-6x^2 + 3^{x - \ln(3x^2 - 2x + 7)}}{4x^2 - x + 12} = 0$ на [10; 15] методом половинного деления, для 5, 10, 20 разбиений начального отрезка, и методом хорд, для 5, 10, 20 итераций. Обьясните разницу в полученных результатах.

2. Решение уравнения $x^2 - \sin 5x = 0$ на [0,5; 0,6] методом касательных, для 2, 4, 6 итераций и с точностью ε: 0,001; 0,00001; 0,0000001.
Считайте: $m_1 = 1{,}2$; $M_2 = 6{,}2$.

ГЛАВА 4

ЧИСЛЕННЫЕ МЕТОДЫ ДЛЯ НАХОЖДЕНИЯ ОПРЕДЕЛИТЕЛЕЙ МАТРИЦ И РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

Изучив данную главу, вы сможете:

- описывать численные методы вычисления определителей и алгоритмы, их реализующие;
- описывать численные методы решения систем линейных уравнений;
- описывать алгоритмы решения систем линейных уравнений методами Крамера и Гаусса;
- писать программы для вычисления определителей *n*-го порядка;
- писать программы для решения систем линейных уравнений порядка *n*;
- комбинировать изученные методы для разработки эффективных алгоритмов решения систем линейных уравнений и писать программы, реализующие эти алгоритмы.

4.1. Численные определители

Пусть дана квадратная матрица порядка n:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ & & \dots & \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix}. \qquad (1)$$

Каждой такой матрице ставится в соответствие числовое значение, называемое *определителем*. Обозначают определитель матрицы A следующим образом: $\det(A)$. Можно дать определение определителя индуктивным способом. Для этого воспользуемся понятием *минора*.

Назовем *минором порядка n–1* элемента $a_{i,j}$ матрицы A ранга $n(n>1)$ определитель матрицы ранга $n–1$, полученной из матрицы A путем исключения строки i и столбца j. Обозначим минор элемента $a_{i,j}$ через $A_{i,j}$, где i – номер строки, а j – номер столбца, на пересечении которых находится элемент $a_{i,j}$.

Так, для вычисления минора $A_{1,2}$ элемента $a_{1,2}$ в приведенной ниже матрице удаляется строка 1 и столбец 2:

$$A = \begin{pmatrix} \cancel{1} & \cancel{3} & \cancel{7} \\ 0 & \cancel{0} & 2 \\ -1 & \cancel{2} & 1 \end{pmatrix}; \quad A_{1,2} = \det \begin{pmatrix} 0 & 2 \\ -1 & 1 \end{pmatrix} = 2.$$

Определителем матрицы ранга *n* называется значение выражения $\sum_{j=1}^{n}(-1)^{1+j}a_{1,j}A_{1,j}$.

Согласно определению

$$\Delta = \det(A) = \begin{vmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{vmatrix} = \sum_{j=1}^{n}(-1)^{1+j}a_{1,j}A_{1,j}.$$

Так, для матрицы 4-го порядка:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix},$$

$$\det(A) = a_{1,1}A_{1,1} - a_{1,2}A_{1,2} + a_{1,3}A_{1,3} - a_{1,4}A_{1,4}.$$

Каждый из миноров $A_{1,j}$, $j = 1, ..., 4$ является определителем матицы 3-го порядка и может быть вычислен прямо по формуле.

Алгоритм вычисления определителя матрицы

Пусть дана матрица (1).

Алгоритм вычисления определителя матрицы порядка *n* основывается на применении определения:

$$\det(A) = \sum_{j=1}^{n}(-1)^{1+j}a_{1,j}A_{1,j}.$$

В этой формуле неизвестными являются миноры элементов из первой строки матрицы. Возьмем произвольный минор $A_{1,j}$. Он является определителем матрицы порядка $n-1$. Для его нахождения необходимо решить такую же задачу, как и исходная, только меньшей размерности. Рассуждая таким образом, в определенный момент возникает необходимость вычислить определители порядка 1, 2 или 3, формулы для прямого вычисления которых известны. Следовательно, может быть применен рекурсивный алгоритм:

Пусть матрица *A* имеет порядок R.

АЛГОРИТМ CRD (*A*, R)

{Рекурсивное вычисление определителя матрицы *A* порядка R.}

Элементарный случай

Если порядок матрицы *A* равен 1, (R = 1), то CRD $\Leftarrow a_{1,1}$, иначе:

Вспомним!

Для матрицы *A* порядка 1, состоящей из единственного элемента $a_{1,1}$, определителем будет значение этого элемента.

Пример: $A = (7)$; $\det(A) = 7$.

Для матрицы порядка 2,

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

определитель равен значению выражения $a_{1,1}a_{2,2} - a_{1,2}a_{2,1}$.

Пример: $A = \begin{pmatrix} 2 & 5 \\ 1 & 7 \end{pmatrix}$; $\det(A) = 9$.

Для матрицы порядка 3,

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

определитель может быть вычислен по правилу Саррюса (правило диагоналей и треугольников). Так, формула для вычисления определителя матрицы 3-го порядка записывается так:

$$\det(A) = a_{1,1}a_{2,2}a_{3,3} + a_{1,3}a_{2,1}a_{3,2} + a_{1,2}a_{2,3}a_{3,1} - a_{1,3}a_{2,2}a_{3,1} - a_{2,1}a_{1,2}a_{3,3} - a_{1,1}a_{2,3}a_{3,2}.$$

Пример:

$$A = \begin{pmatrix} 1 & 3 & 7 \\ 0 & 0 & 2 \\ -1 & 2 & 1 \end{pmatrix}; \det(A) = -10.$$

Правило сходимости:

1. Существует элементарный случай: матрица порядка 1, которой соответствует минор, равный ее элементу.
2. На *k*-м уровне осуществляется *k* вызовов для нахождения *k* определителей порядка $k-1$. Следовательно, процесс вычислений сходится к элементарному случаю.

Случай понижения размерности задачи

1. Значение определителя Δ инициализируется нулем (Δ ⇐ 0).

2. Для всех *j* от 1 до R:

a) Формируется матрица $M_{1,j}$ путем исключения из текущей матрицы *A* первой строки и *j*-го столбца. (Порядок $M_{1,j}$ равен R–1. Данная матрица соответствует минору $A_{1,j}$.)

b) Вычисляется определитель $D_{1,j}$ матрицы $M_{1,j}$ с помощью вызова CRD ($M_{1,j}$, R–1).

c) Обновляется значение $\Delta \Leftarrow \Delta + (-1)^{1+j} \times D_{1,j}$.

3. CRD⇐ Δ . КОНЕЦ.

Реализация алгоритма

Пусть даны объявления:

```
const   nmax=10;
type    matrice=array[1..nmax,1..nmax] of real;
```

Матрица, определитель которой вычисляется, сохраняется в массиве X типа `mat`.

Максимально возможный порядок матрицы определяется константой `nmax`. Минор $M_{1,j}$, генерируемый в процессе расчетов во время текущего вызова, сохраняется в переменной `minor` типа `mat`. Знак значения $(-1)^{1+j}$ CRD ($M_{1,j}$, R–1) определяется четностью переменной *j*, следовательно, вычисленное значение будет прибавляться для нечетных *j* и вычитаться – для четных.

Приведем возможную реализацию на языке Паскаль рекурсивной функции вычисления определителей:

```
function cdet(var x:matrice; t:integer) : real;
   var
   i, j, k: integer;
   s : real;
   minor : matrice;
begin
   if t=1 then cdet:=x[1,1] {элементарный случай}
   else  begin
      s:=0;
         for k:=1 to t do
            begin
      {Исключается строка 1 и столбец k с целью сформирования матрицы,
      соответствующей минору элемента x[1,k]}
      for i:=1 to t-1 do
         for j:=1 to k-1 do
         minor[i,j]:=x[i+1,j];
         for i:=1 to t-1 do
         for j:=k to t-1 do
         minor[i,j]:=x[i+1,j+1];
   {рекурсивный вызов}
   if odd(k) then s:=s+x[1,k]*cdet(minor, t-1)
   else s:=s-x[1,k]*cdet(minor, t-1);
   end;
         cdet:=s;
         end;
end;
```

Число операций, необходимых для рекурсивного вычисления определителя матрицы порядка n, определяется количеством рекурсивных вызовов, а также числом операций, совершаемых во время одного вызова.

Разложение матрицы порядка n по строке предполагает формирование n миноров порядка $n-1$. Далее, при развитии каждого из них появятся $n-1$ миноров порядка $n-2$ и так далее. Общее число вызовов определяется значением выражения $n \times (n-1) \times \times (n-2) \times \ldots \times 3 \times 2 \times 1 = n!$ Количество операций при каждом вызове пропорционально n^2. Следовательно, временная сложность алгоритма равна $O(n^2 n!)$, что делает его малоэффективным для больших значений n.

Существует и алгоритм полиномиальной сложности, позволяющий итеративный расчет определителей. Алгоритм использует элементарные преобразования, приводящие матрицу к треугольному виду. Определитель такой матрицы равен произведению значений элементов, находящихся на главной диагонали.

Для обнуления элементов, расположенных ниже диагонального элемента, в столбце j ($j=1, ..., n-1$), элементы i-й строки ($i = j+1, ..., n$) складываются с элементами строки j, перемноженными на коэффициент $\frac{-a_{i,j}}{a_{j,j}}$. В случае, если элемент $a_{j,j}$ равен нулю, производится попытка поменять местами две строки с номерами j и k ($j < k$) так, чтобы $a_{k,j} \neq 0$. В этом случае необходимо иметь ввиду, что операция перемены местами строк в матрице приводит к изменению знака определителя. Если же такой строки не существует, то можно с уверенностью утверждать, что определитель матрицы равен нулю.

Недостаток метода состоит в том, что производится большое количество операций деления. Это в случае значительных колебаний значений элементов матрицы может привести к значительным погрешностям. Данная ситуация не возникает в случае применения рекурсивного алгоритма.

Приведем возможную реализацию на языке Паскаль итеративной функции вычисления определителей:

```
function CID(x:matrice; r:integer): real;
   var  i,j,k:integer;
        q:real;
begin {CID}
   for i: =1 to r-1 do
      begin
         {проверяется значение диагонального элемента из строки i} if x[i,i]=0
         then {если оно равно нулю, ищется строка для обмена}
            begin
               k:=i;
               for j:=i+1 to r do
                  if x[j,i]<>0 then k:=j;
               {если строка для обмена не найдена}
               if k=i then begin CID:=0; exit; end
               {Иначе строки i, k обмениваются местами}
                     else
                     for j:=1 to r do
                        begin
                           q:=x[i,j];
                           x[i,j]:=x[k,j];
                           x[k,j]:=-q;
                        end;
            end;
```

```
      {преобразование строк с целью обнуления элементов в столбце i,
      расположенных под диагональю}
      for j:=i+1 to r do
         begin
            q:=-x[j,i]/x[i,i];
            for k:=i to r do x[j,k]:=x[j,k]+x[i,k]*q;
         end;
      end;
      {вычисление значения определителя треугольной матрицы}
      q:=1;
      for i:=1 to r do q:=q*x[i,i];
   CID:=q;
end;
```

Вопросы и упражнения

❶ Объясните связь между матрицей и ее определителем.

❷ Напишите функцию для вычисления определителя матрицы 3-го порядка. Воспользуйтесь правилом Саррюса.

❸ Опишите алгоритм вычисления определителя порядка *n* путем развития матрицы по элементам произвольной строки.

❹ Напишите программу для вычисления определителей порядка *n* ($n \le 10$), в которой воспользуйтесь функцией `cdet`, описанной в данном параграфе. Порядок матрицы и значения ее элементов введите с клавиатуры.

❺ Вычислите с помощью написанной программы определители следующих матриц:

a) $\begin{pmatrix} 2 & 5 \\ 4 & -1 \end{pmatrix}$;

b) $\begin{pmatrix} 1 & 0 & 3 \\ 5 & 8 & 2 \\ 6 & -1 & 4 \end{pmatrix}$;

c) $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 0 & 5 \\ 3 & 6 & 9 & 6 \\ 4 & 5 & 8 & 7 \end{pmatrix}$;

d) $\begin{pmatrix} 7 & 2 & 3 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 6 & 0 & 6 \\ 4 & 4 & 3 & 0 \end{pmatrix}$;

e) $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 2 & -3 & 4 & 1 \\ 3 & 4 & -3 & 9 \end{pmatrix}$;

f) $\begin{pmatrix} -4 & 9 & -4 & 6 & -5 \\ -1 & 7 & 5 & 5 & 8 \\ 9 & -6 & 8 & -6 & 2 \\ 2 & 5 & 2 & 3 & 9 \\ 1 & 3 & 1 & 5 & -2 \end{pmatrix}$;

g) $\begin{pmatrix} 1 & -1 & 3 & 2 \\ 1 & -1 & 6 & 5 \\ 1 & -1 & -9 & -10 \\ 1 & -1 & 2 & -1 \end{pmatrix}$;

h) $\begin{pmatrix} 6 & 5 & 3 & -2 & -7 & 1 \\ 1 & 4 & -1 & 8 & -6 & -5 \\ 4 & -3 & 1 & 10 & 9 & -3 \\ 6 & 4 & -6 & 7 & 7 & 9 \\ -8 & 6 & 4 & -1 & -5 & 6 \\ 1 & 4 & -5 & 6 & -6 & 3 \end{pmatrix}$.

❻ В реализованной в упражнении 4 программе, преобразуйте параметр-переменную *x* функции `cdet` в параметр-значение. Обратите внимание на то, как изменятся вычисления определителя одной и той же матрицы. Объясните происшедшие изменения.

❼ Попытайтесь применить написанную программу для вычисления определителя матрицы порядка *n*, $n > 20$, с элементами типа `real`. Что произошло? Объясните.

4.2. Решение систем линейных уравнений методом Крамера

Формулы Крамера[1]

Способность вычисления определителя порядка n с помощью компьютера дает возможность решить целую гамму задач из различных областей. Одним из практических приложений вычисления определителя является решение систем линейных уравнений.

Пусть дана система из n линейных уравнений с n неизвестными:

$$\left.\begin{aligned} &a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1 \\ &a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2 \\ &\ldots\ldots\ldots \\ &a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = b_n \end{aligned}\right\}. \qquad (1)$$

Для записи системы в матричной форме воспользуемся следующими обозначениями:

$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,n} \end{pmatrix}$ – матрица коэффициентов, $b = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}$ – вектор свободных членов, $x = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}$ – вектор неизвестных.

В матричной форме система уравнений (1) может быть записана так:

$$Ax = b. \qquad (2)$$

В случае, если определитель A отличен от нуля, существует обратная матрица A^{-1}. Если умножить обе части равенства (2) на A^{-1}, получим:

$$A^{-1}Ax = A^{-1}b,$$

что эквивалентно

$$x = A^{-1}b. \qquad (3)$$

Эта формула позволяет вычисление решения системы (1) в том случае, если матрица A системы не является вырожденной. Формула вычисления элементов вектора-решения следует непосредственно из (3) и свойств обратной матрицы:

$$x_i = \frac{\Delta_i}{\Delta} \quad i = 1, \ldots, n$$

$$\Delta_i = \begin{vmatrix} a_{1,1} & \ldots & a_{1,i-1} & b_1 & a_{1,i+1} & \ldots & a_{1,n} \\ a_{2,1} & \ldots & a_{2,i-1} & b_2 & a_{2,i+1} & \ldots & a_{2,n} \\ & & & \ldots & & & \\ a_{n,1} & \ldots & a_{n,i-1} & b_n & a_{n,i+1} & \ldots & a_{n,n} \end{vmatrix}. \qquad (4)$$

Вспомним!

Квадратная матрица A называется ***невырожденной***, если ее определитель отличен от нуля.

Обратная матрица матрице A – это такая матрица, которая, будучи умноженной на A (слева либо справа), дает в результате единичную матрицу E. Обозначается такая матрица A^{-1}:

$$AA^{-1} = A^{-1}A = E.$$

[1] *Габриэл Крамер* (1704–1752) – швейцарский математик. Родился в Женеве. Наиболее выдющиеся результаты в математике получены им в области алгебры (исследование алгебраических кривых, правило Крамера) и в области теории вероятностей (решение „Санкт-Петербургского парадокса ").

Δ_i – это определитель матрицы, полученной из A путем замены i-го столбца вектором свободных членов.

Формулы (4) позволяют напрямую вычислить решение системы линейных уравнений (1), так как в них участвуют только величины, вычисленные с помощью матрицы коэффициентов системы (1) и вектора ее свободных членов. Эти формулы называются *формулами Крамера*.

АЛГОРИТМИЗАЦИЯ МЕТОДА

Пусть дана система (1), содержащая n линейных уравнений с n неизвестными. Для решения системы методом Крамера понадобятся следующие структуры данных:

a) Двумерный массив A ($n \times n$) с элементами реального или целого типа (в зависимости от входных данных) – матрица коэффициентов системы.

b) Одномерный массив b с элементами реального или целого типа (в зависимости от входных данных) – вектор свободных членов системы.

c) Одномерный массив x с элементами реального типа – вектор решения системы.

Определение решения системы линейных уравнений с помощью формул Крамера основано на вычислении определителей. Для этого можем воспользоваться рекурсивной функцией CRD (A, n) из предыдущего параграфа, которую включим в алгоритм.

Шаг 1. Вычисляется Δ = CRD(A, n) (A – матрица коэффициентов системы).

Если $\Delta \neq 0$, переходим к *шагу 2*; в противном случае выводим сообщение о невозможности применения метода. КОНЕЦ.

Шаг 2. i: $\Leftarrow$ 1

Шаг 3.

a) Создается копия (C) матрицы коэффициентов системы. $C \Leftarrow A$.

b) В массиве C, i-й столбец заменяется вектором свободных членов b.

c) Считается определитель Δ_i матрицы C, а затем i-я компонента решения:

$$x_i = \frac{\Delta_i}{\Delta}.$$

d) Переходим к *шагу 4*.

Шаг 4. Если $i<n$, то $i \Leftarrow i+1$, переходим к *шагу 3*; в противном случае выводится результат – вектор x. КОНЕЦ.

Примечание. При написании программы копия матрицы коэффициентов может быть получена путем передачи исходной матрицы в подпрограмму в виде параметра-значения. Замена необходимого столбца и вычисление определителя будут произведены в этой подпрограмме.

Пример 1: Написать программу, вычисляющую с помощью формул Крамера решение системы:

$$\begin{cases} 3x_1 + 2x_2 - x_3 = 7 \\ -x_1 + x_2 + x_3 = 4 \\ x_1 - 6x_2 + 2x_3 = -13 \end{cases}$$

Ввод: Коэффициенты и свободные члены системы вводятся путем присваивания в тексте программы.

Вывод: Результаты выводятся на экран.

```
program cn11;
type matrice=array[1..3,1..3] of integer;
     vec=array[1..3] of integer;
     vs=array[1..3] of real;
var  a     : matrice;
     b     : vec;
     sol   : vs;
     p,n   : integer;
     de    : real;

function cdet( x:matrice;t:integer):real;
var  i,j,k,l: integer;
     s      : real;
     minor  : matrice;
begin
   if t=1 then cdet:=x[1,1]
      else
         begin   s:=0;
            for k:=1 to t do
               begin
                  for i:=1 to t-1 do
                     for j:=1 to k-1 do minor[i,j]:=x[i+1,j];
                  for i:=1 to t-1 do
                     for j:=k to t-1 do minor[i,j]:=x[i+1,j+1];
                        if odd(k) then s:=s+x[1,k]*cdet(minor, t-1)
                                  else s:=s-x[1,k]*cdet(minor, t-1);
               end;
            cdet:=s;
         end;
end;

function transforma(x:matrice;t,l:integer):real;
var  i : integer;
begin      for i:=1 to t do x[i,l]:=b[i];
           transforma:=cdet(x,t);
end;
begin
     n:=3;
     a[1,1]:= 3; a[1,2]:= 2; a[1,3]:= -1; b[1]:=  7;
     a[2,1]:=-1; a[2,2]:= 1; a[2,3]:=  1; b[2]:=  4;
     a[3,1]:= 1; a[3,2]:=-6; a[3,3]:=  2; b[3]:=-13;
     de:=cdet(a,n);
        if de<>0 then
           begin   for p:=1 to n do sol[p]:= transforma(a,n,p)/de;
                   for p:=1 to n do writeln('x[',p,']=',sol[p]:0:3);
           end
        else writeln('Calcul imposibil');
end.
```

Результаты:

```
x[1]=1.000
x[2]=3.000
x[3]=2.000
```

Вопросы и упражнения

❶ Для решения каких задач используют формулы Крамера? Каковы условия их применения?

❷ Объясните смысл обозначений Δ и Δ_i из формул Крамера.

❸ Основываясь на примере **cn11** напишите программу для решения системы n линейных уравнений с n ($n \le 10$) неизвестными, использующую формулы Крамера:

a) число уравнений n, значения коэффициентов и свободные члены системы вводятся с клавиатуры;

b) исходные данные считываются из системного текстового файла, в котором в первой строке записано целое число n – количество уравнений системы. Каждая из следующих n строк содержит по n+1 целых чисел, разделенных пробелом, – коэффициенты уравнений и свободные члены в порядке появления уравнений в системе.

❹ Решите с помощью программы из упражнения 3 следующие системы уравнений:

a)
$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 10 \\ x_1 - x_2 + x_3 - x_4 = -2 \\ 2x_1 - 3x_2 + 4x_3 + x_4 = 12 \\ 3x_1 + 4x_2 - 3x_3 + 9x_4 = 38 \end{cases}$$

b)
$$\begin{cases} -4x_1 + 9x_2 - 4x_3 + 6x_4 - 5x_5 = -8 \\ -x_1 + 7x_2 + 5x_3 + 5x_4 + 8x_5 = 199 \\ 9x_1 - 6x_2 + 8x_3 - 6x_4 + 2x_5 = 52 \\ 2x_1 + 5x_2 + 2x_3 + 3x_4 + 9x_5 = 198 \\ x_1 + 3x_2 + x_3 + 5x_4 - 2x_5 = 33 \end{cases}$$

c)
$$\begin{cases} 6x_1 + 5x_2 + 3x_3 - 2x_4 - 7x_5 + x_6 = -48 \\ x_1 + 4x_2 - x_3 + 8x_4 - 6x_5 - 5x_6 = -106 \\ 4x_1 - 3x_2 + x_3 + 10x_4 + 9x_5 - 3x_6 = 70 \\ 6x_1 + 4x_2 - 6x_3 + 7x_4 + 7x_5 + 9x_6 = 64 \\ -8x_1 + 6x_2 + 4x_3 - 1x_4 - 5x_5 + 6x_6 = 25 \\ x_1 + 4x_2 - 5x_3 + 6x_4 - 6x_5 + 3x_6 = -82 \end{cases}$$

d)
$$\begin{cases} x_1 - x_2 + 3x_3 + 2x_4 = 1 \\ x_1 - x_2 + 6x_3 + 5x_4 = 0 \\ x_1 - x_2 - 9x_3 - 10x_4 = 5 \\ x_1 - x_2 + 2x_3 - x_4 = \frac{4}{3} \end{cases}$$

e)
$$\begin{cases} x_1 - 2x_2 - 8x_3 + 3x_4 - 4x_5 - 2x_6 + 2x_7 = -104 \\ 5x_1 + 8x_2 + x_3 - 5x_4 + 2x_5 - x_6 - 8x_7 = 94 \\ 2x_1 - 4x_2 + 7x_3 + 3x_4 + 7x_5 + 10x_6 + 9x_7 = 159 \\ -4x_1 + 2x_2 - x_4 + 3x_5 + 6x_6 - 8x_7 = -80 \\ -x_1 - x_2 + 8x_3 + 7x_4 + 2x_5 - 6x_6 + 2x_7 = 204 \\ 7x_1 + x_2 + 9x_3 - 9x_4 + 6x_5 - 6x_6 = 209 \\ -5x_1 + 9x_2 - 6x_3 - 5x_4 + 10x_5 - x_6 - 9x_7 = -29 \end{cases}$$

f)
$$\begin{cases} -8x_1 + 5x_2 + 10x_3 - 8x_4 + 7x_5 = 106 \\ 4x_1 - 8x_2 - 4x_3 - 3x_4 + 3x_5 = 39 \\ 6x_1 - 4x_2 + 10x_3 + 6x_4 - 7x_5 = 97 \\ 8x_1 + 2x_2 + 4x_3 - 8x_4 - 7x_5 = 119 \\ -6x_1 - 8x_2 + 9x_3 - 6x_4 - 3x_5 = 10 \end{cases}$$

g)
$$\begin{cases} -2x_1 + 6x_3 - 4x_4 = 68 \\ 4x_1 - 8x_2 - 2x_3 + 10x_4 = -78 \\ -9x_2 - 9x_3 + 5x_4 = -128 \\ -5x_1 + 7x_2 + 3x_3 + 4x_4 = 27 \end{cases}$$

4.3. Решение систем линейных уравнений методом Гаусса

Одним из эффективных способов решения систем линейных уравнений является метод последовательного исключения неизвестных, получивший также название как *метод Гаусса*[2]. Данный метод может применяться в случаях, когда система имеет единственное либо бесконечное множество решений (во втором случае будет найдено частное решение уравнения).

Пусть дана система (1) из n линейных уравнений с n неизвестными с расширенной матрицей (1а):

$$\begin{cases} a_{1,1}x_1+\cdots+a_{1,i}x_i+\ldots+a_{1,n}x_n=b_1 \\ a_{2,1}x_1+\cdots+a_{2,i}x_i+\ldots+a_{2,n}x_n=b_2 \\ \cdots \\ a_{i,1}x_1+\cdots+a_{i,i}x_i+\ldots+a_{i,n}x_n=b_i \\ \cdots \\ a_{n,1}x_1+\cdots+a_{n,i}x_i+\ldots+a_{n,n}x_n=b_n \end{cases} \quad (1) \qquad \left(\begin{array}{ccccccc} a_{1,1} & \ldots & a_{1,i} & \ldots & a_{1,n} & b_1 \\ a_{2,1} & \ldots & a_{2,i} & \ldots & a_{2,n} & b_2 \\ & & \cdots & & & \\ a_{i,1} & \ldots & a_{i,i} & \ldots & a_{i,n} & b_i \\ & & \cdots & & & \\ a_{n,1} & \ldots & a_{n,i} & \ldots & a_{n,n} & b_n \end{array}\right) \quad (1а)$$

Метод Гаусса предполагает:

- исключение переменной x_1 из всех уравнений системы, начиная со второго;
- исключение переменной x_2 из всех уравнений системы, начиная с третьего;

...

- исключение переменной x_i из всех уравнений системы, начиная с $(i+1)$-го;

...

- исключение переменной x_{n-1} из n-го уравнения системы.

$$\begin{cases} a^*_{1,1}x_1+a^*_{1,2}x_2+\ldots+a^*_{1,i}x_i+\ldots+a^*_{1,n}x_n=b^*_1 \\ a^*_{2,2}x_2+\ldots+a^*_{2,i}x_i+\ldots+a^*_{2,n}x_n=b^*_2 \\ \cdots \\ a^*_{i,i}x_i+\ldots+a^*_{i,n}x_n=b^*_i \\ \cdots \\ a^*_{n,n}x_n=b^*_n \end{cases} \quad (2)$$

Выражаясь математическим языком, необходимо получить эквивалентную системе (1) систему следующего вида:

Решение системы типа (2) может быть получено, начиная с компоненты x_n, вычисляемой непосредственно из последнего уравнения системы. Вообще говоря, зная компоненты x_{i+1}, x_{i+2} ... x_n, из i-го уравнения системы (2) вычисляется компонента x_i.

Таким образом, процесс решения системы линейных уравнений методом Гаусса разбивается на два этапа:

Этап 1 (прямой) – преобразование системы (1) в эквивалентную систему (2) путем элементарных преобразований матрицы (1а);

Этап 2 (обратный) – вычисление компонент решения системы (2).

В случае, когда система имеет бесконечное множество решений (содержит свободные переменные), метод позволяет нахождение частного решения путем прямого присваивания частных значений свободным переменным.

[2] *Карл Фридрих Гаусс* (1777–1855) – немецкий математик и исследователь. Родился в городе Брауншвейг. На протяжении жизни занимался важными исследованиями в областях теории чисел, статистики, математического анализа, геометрии, геодезии, геофизики, электростатики, астрономии и оптики.

АЛГОРИТМИЗАЦИЯ МЕТОДА

Пусть дана система (1) из n уравнений с n неизвестными. Для решения системы методом Гаусса понадобятся следующие структуры данных:

a) двумерный массив A ($n \times n+1$) с элементами реального типа – расширенная матрица коэффициентов и свободных членов системы;

b) одномерный массив x с n элементами реального типа – вектор-решение системы.

Решение системы линейных уравнений методом Гаусса означает серию последовательных преобразований элементов расширенной матрицы.

Прямой ход (этап 1)

Шаг 1. $i \Leftarrow 1$.

Шаг 2.

a) Если $a_{ii} = 0$, строка i обменивается местами со строкой $r (r > i)$, для которой $a_{ri} \neq 0$. Если такой строки не существует, то переходим к *шагу 3*.

b) Для каждой из строк j от $i+1$ до n повторяется процедура (P):

I. вычисляется $k = \dfrac{-a_{j,i}}{a_{i,i}}$;

II. для всех l от 1 до $n+1$ $a_{j,l} \Leftarrow a_{j,l} + k \times a_{i,l}$.

Шаг 3. Если $i < n$, то $i \Leftarrow i+1$, затем возврат к *шагу 2*; в противном случае, возврат к *шагу 4*. {Переход к *шагу 4* означает конец прямого и начало обратного этапа метода Гаусса.}

Обратный ход (этап 2)

Шаг 4. Для i от n до 1 вычисляется:

$$x_i = \begin{cases} \dfrac{b_i^* - \sum\limits_{j=i+1}^{n} a_{i,j}^* x_j}{a_{i,i}^*} & a_{i,i} \neq 0 \\ 0 & a_{i,i} = 0 \end{cases}$$

Шаг 5. Для i от 1 до n выводятся значения x_i. КОНЕЦ.

Пример: Пусть дана система из n ($n < 20$) линейных уравнений с действительными коэффициентами, имеющая вид:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,i}x_i + \ldots + a_{1,n}x_n = b_1 \\ \cdots \\ a_{i,1}x_1 + a_{i,2}x_2 + \cdots + a_{i,i}x_i + \ldots + a_{i,n}x_n = b_i \\ \cdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,i}x_i + \ldots + a_{n,n}x_n = b_n \end{cases}$$

Напишите программу для решения такой системы методом Гаусса.

Ввод: Коэффициенты и свободные члены системы считываются из текстового файла IN.TXT со следующей структурой (значения элементов отделены пробелом):

n			
$a_{1,1}$	...	$a_{1,n}$	b_1
$a_{2,1}$	...	$a_{2,n}$	b_2
	...		
$a_{n,1}$	...	$a_{n,n}$	b_n

Вывод: Вычисленное решение выводится на экран.

Решение: Условия задачи ставят нас перед необходимостью применить алгоритм решения для общего случая. При наличии свободных переменных им присваивается значение 0.

```
program cn12;
const nmax=10;
type    matrice=array[1..nmax,1..nmax] of real;
        vect=array[1..nmax] of real;
var     a:matrice;
        s :vect;
        i,n:integer;

procedure citeste(var x:matrice; var t:integer);
   var  i, j: integer; f: text;
   begin {citeste}
      assign (f, 'in.txt'); reset(f);
      readln(f, t);
   for i:=1 to t do
   begin
   for j:=1 to t do read(f,x[i,j]);
   readln(f, x [i,t+1]);
   end;
   close(f);
   end; {citeste}

procedure direct(var x:matrice;t:integer);
     label linie_urmatoare;
     var  i,j,k,l:integer; r:real;
     begin {direct}
        for i:=1 to t-1 do
           begin
              if x[i,i]=0 then
                 begin
                    k:=i;
                    for j:=i+1 to n do if x[j,i]<>0 then k:=j;
                    if k=i then goto linie_urmatoare
                           else
                              for j:=1 to t+1 do
                              begin
                                 r:=x[i,j]; x[i,j]:=x[k,j]; x[k,j]:=r;
                              end;
                 end;
              for j:=i+1 to t do
                 begin   r:=-x[j,i]/x[i,i];
                         for k:=i to t+1 do  x[j,k]:=x[j,k]+x[i,k]*r;
                 end;
     linie_urmatoare: end;
   end; {direct}

procedure invers (var q:vect);
   var i,j: integer;
   s: real;
```

```
begin
   for i:=n downto 1 do
   begin
      s:=0;
      for j:=i+1 to n do s:=s+a[i,j]*q[j];
      if a [i,i]<>0 then q[i]:=(a[i,n+1] -s)/a[i,i] else q[i]:=0;
   end;
end;
begin   citeste(a,n);
        direct(a,n);
        invers(s);
        for i:=1 to n do writeln('x[',i,']=',s[i]:0:3);
end.
```

Тестовые данные:

$$\begin{cases} 3x_1 + 2x_2 - x_3 = 7 \\ 6x_1 + 4x_2 - 2x_3 = 14 \\ x_1 - 6x_2 + 2x_3 = -13 \end{cases}$$

Результаты:

```
x[1]=0.800
x[2]=2.300
x[3]=0.000
```

$$\begin{cases} -8x_1 + 9x_2 - x_3 + 9x_4 - 5x_5 = -111 \\ -4x_1 - 9x_2 + 9x_3 + 2x_4 - 2x_5 = 88 \\ -4x_1 - 4x_2 + 7x_3 + 3x_4 + 4x_5 = 64 \\ -3x_1 + 6x_2 - 7x_3 - 6x_4 + 9x_5 = -110 \\ x_1 - 9x_2 - 9x_3 - 7x_4 - 6x_5 = -82 \end{cases}$$

```
x[1]=9.000
x[2]=-3.000
x[3]=11.000
x[4]=1.000
x[5]=2.000
```

Примечания:

Заметим, в данном случае имеются 2 уравнения с пропорциональными коэффициентами. Программа распознает свободную переменную `x[3]`, которой присваивается значение 0.

Определитель данной матрицы отличен от 0. Следовательно, существует единственное решение системы. Путем прямой проверки можно установить правильность решения, сгенерированного программой.

Вопросы и упражнения

❶ Опишите этапы алгоритма решения системы *n* линейных уравнений с *n* неизвестными методом Гаусса.

❷ Оцените объем памяти, необходимый для решения системы *n* линейных уравнений с *n* неизвестными методом Гаусса.

❸ Напишите программу для решения системы *n* ($n \le 10$) линейных уравнений с *n* неизвестными методом Гаусса.

❹ Оцените временную сложность алгоритма решения системы *n* линейных уравнений с *n* неизвестными методом Гаусса. Воспользуйтесь программой `cn12`.

❺ Модифицируйте программу `cn12` так, чтобы она позволяла решать системы уравнений, заданные в виде:

a)
$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n}x_n = b_1 \\ a_{2,2}x_2 + a_{2,3}x_3 + \ldots + a_{2,n}x_n = b_2 \\ \ldots \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ a_{n,n}x_n = b_n \end{cases}$$

b)
$$\begin{cases} a_{1,1}x_1 = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 = b_2 \\ \ldots \\ a_{n-1,2}x_2 + a_{n-1,3}x_3 + \ldots + a_{n-1,n}x_n = b_{n-1} \\ a_{n,1}x_1 + a_{n,2}x_2 + a_{n,3}x_3 + \ldots + a_{n,n}x_n = b_n \end{cases}$$

❻ Модифицируйте программу `cn12` так, чтобы она позволяла генерировать случайные значения свободных переменных.

Контрольный тест

Выберите правильный ответ:

1. Определитель – это числовая характеристика, которой обладает матрица:
a) только ранга 1;
b) любого ранга;
c) ранга, меньшего либо равного 3.

2. Для вычисления определителей могут использоваться рекурсивные алгоритмы:
a) истина; b) ложь.

3. Для сохранения матрицы размерностью 10 × 10, с элементами типа `real`, в памяти компьютера понадобятся:
a) 640 байт;
b) 1 024 байт;
c) 600 байт;
d) 400 байт;
e) 200 байт.

4. Метод Крамера может применяться только в случае, если определитель матрицы коэффициентов системы:
a) отличен от 0;
b) равен 0;
c) положителен;
d) является целым числом.

5. Пусть Δ – определитель матрицы коэффициентов системы n линейных уравнений с n неизвестными , а Δ_1 – определитель матрицы коэффициентов системы, в которой первый столбец заменили вектором свободных членов. Метод Крамера использует следующую формулу для вычисления компоненты x_1 вектора-решения:

a) $x_1 = \dfrac{\Delta}{\Delta_1}$;
b) $x_1 = \dfrac{\Delta_1}{\Delta}$;
c) $x_1 = \dfrac{(\Delta_1 - \Delta)}{\Delta}$;
d) $x_1 = \dfrac{(\Delta_1 - \Delta)}{\Delta_1}$.

6. Метод Гаусса решения системы n линейных уравнений с n неизвестными состоит из:
a) одного этапа;
b) двух этапов;
c) трех этапов;
d) количество этапов невозможно оценить априори.

7. В методе Гаусса, реализуя этап прямого хода решения системы n линейных уравнений с n неизвестными, переменная x_i исключается из всех уравнений с номерами:
a) от 1 до i;
b) от 1 до i +1;
c) от i до n;
d) от i+1 до n.

8. Пусть дана система линейных уравнений, в которой $a_{i,i} \neq 0$ $(i = 1, ..., n)$.

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n-1}x_{n-1} + a_{1,n}x_n = b_1 \\ a_{2,2}x_2 + a_{2,3}x_3 + \ldots + a_{2,n-1}x_{n-1} + a_{2,n}x_n = b_2 \\ \ldots \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ a_{n,n}x_n = b_n \end{cases}$$

Метод Гаусса предполагает нахождение компоненты x_i вектора-решения по формуле:

a) $x_i = \dfrac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$;

b) $x_i = \dfrac{b_i + \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$;

c) $x_i = \dfrac{a_{ii}}{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}$;

d) $x_i = \dfrac{-b_i + \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}$.

II Решите, с помощью написанных ранее программ, предложенные ниже задачи. Для каждого вычисленного решения выведите на экран строку, содержащую слово „Răspuns:", затем, после пробела, сгенерированное программой решение:

1. Определители матриц:

a) $\begin{bmatrix} 1 & 4 & 0 \\ 2 & 3 & 7 \\ -4 & 1 & -2 \end{bmatrix}$;

b) $\begin{bmatrix} 4 & -1 & 2 & 5 \\ 2 & 2 & 1 & 4 \\ 0 & 3 & 5 & 6 \\ 2 & -1 & 4 & 1 \end{bmatrix}$.

2. Решения систем уравнений (методом Крамера):

a) $\begin{cases} -6x_1 - 3x_2 - 3x_3 = -63 \\ -7x_1 + 8x_2 + 6x_3 = 10 \\ 2x_1 + 6x_2 - 7x_3 = -64 \end{cases}$

b) $\begin{cases} 9x_2 - 8x_3 + 2x_4 = -37 \\ 8x_1 - 7x_2 + 9x_3 + 8x_4 = 27 \\ -3x_1 - 7x_2 + 3x_3 + 4x_4 = 12 \\ -x_1 - 9x_3 + x_4 = -69 \end{cases}$

3. Решения систем уравнений (методом Гаусса):

$$\begin{cases} 5x_2 + x_4 + 5x_5 = 89 \\ -8x_1 + 4x_2 - x_4 + 4x_5 = 87 \\ 10x_1 + 3x_2 - 9x_3 - 3x_4 - 2x_5 = -13 \\ -7x_1 + 6x_2 - 6x_3 + 7x_4 + 7x_5 = 217 \\ -7x_1 + 6x_2 + 8x_3 - 7x_4 + x_5 = -1 \end{cases}$$

ГЛАВА 5

ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

Изучив данную главу, вы сможете:

- описывать алгоритм вычисления определенного интеграла методом прямоугольников (трапеций);
- писать подпрограммы для вычисления определенного интеграла методом прямоугольников (трапеций);
- идентифицировать задачи из разных областей, для решения которых необходимо вычислять численными методами определенные интегралы.

5.1. Приближенное вычисление определенного интеграла методом прямоугольников

Одним из наиболее часто применяемых приложений численных методов является численное интегрирование. Прямые методы вычисления определенного интеграла не всегда позволяют найти аналитическое решение, часто бывает, что неизвестна формула интегрируемой функции. Обычно в таких случаях задано лишь некоторое число точек, для которых известно значение интегрируемой функции. В данной ситуации вычисление интеграла возможно лишь приближенными методами (предполагается, что интегрируемая функция является непрерывной на отрезке интегрирования).

Из курса математики известно, что геометрическим смыслом определенного интеграла $\int_a^b f(x)dx$ является площадь криволинейной трапеции, определяемой осью $0x$, прямыми $x = a$, $x = b$ и графиком функции $f(x)$ на отрезке $[a, b]$ (*рис. 5.1*).

Точное вычисление площади данной фигуры не всегда возможно. В таких случаях возможным решением могла бы послужить аппроксимация начальной фигуры множеством других фигур, площадь которых легко вычисляется. Тогда значение определенного интеграла (площадь начальной фигуры) приблизительно будет равно сумме вычисляемых площадей данных фигур.

Пусть f – функция, дифференцируемая на отрезке $[a, b]$. Требуется вычислить $\int_a^b f(x)dx$.

Для численного решения данной задачи построим разбиение отрезка $[a, b]$ с помощью последовательности точек вида $a=x_0<x_1<\ldots<x_{n-1}<x_n<=b$. Пары последовательных точек x_0x_1, x_1x_2, … x_ix_{i+1}, …, $x_{n-1}x_n$ определяют n различных отрезков, объединение которых равно $[a, b]$.

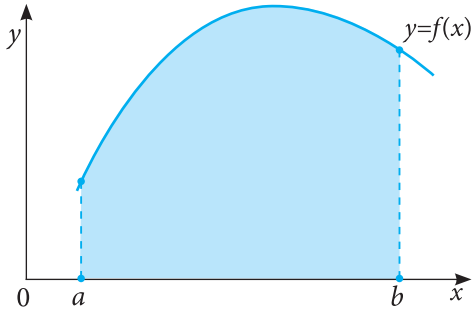


Рис. 5.1. Геометрический смысл определенного интеграла

Для простоты, точки данного разбиения будем считать равноудаленными (*рис. 5.2*). Тогда длина h каждого элементарного отрезка $[x_i, x_{i+1}]$ определяется по формуле

$$h = \frac{|b-a|}{n}.$$

Значения x_i также могут быть выражены через известные величины: $x_i = a + ih$, $i = 0, \ldots, n$.

Зная значения x_i и x_{i+1}, можно вычислить середину каждого элементарного отрезка $z_i = \frac{x_i + x_{i+1}}{2}$.

На каждом из отрезков $[x_i, x_{i+1}]$ площадь криволинейной трапеции аппроксимируется площадью прямоугольника S_i со сторонами h и $f(z_i)$ (*рис. 5.3*):

$$S_i = hf(z_i) = hf\left(a + ih + \frac{h}{2}\right).$$

В данном случае вычисленное значение определенного интеграла I равно сумме площадей прямоугольников и вычисляется по формуле

$$I = h\sum_{i=0}^{n-1} f\left(a + ih + \frac{h}{2}\right),$$

где

n – число разбиений начального отрезка;

h – длина элементарного отрезка;

I – вычисленное значение интеграла.

Таким образом, вычисление интеграла сводится к вычислению значения некоторого арифметического выражения, зависящего лишь от числа разбиений отрезка интегрирования и значения функции в серединах элементарных отрезков. Метод, сводящий вычисление интеграла к вычислению суммы площадей прямоугольников, называется *методом прямоугольников*.

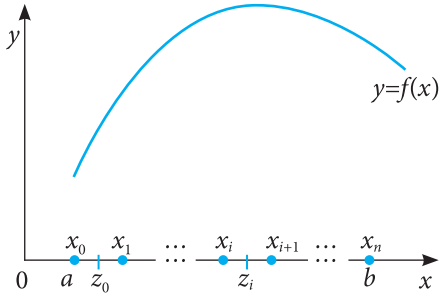


Рис. 5.2. Разбиение $[a, b]$ на элементарные отрезки. Середина элементарного отрезка $[x_i, x_{i+1}]$ – точка $z_i = \frac{x_i + x_{i+1}}{2}$

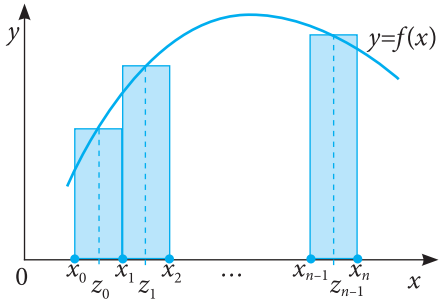


Рис. 5.3. Последовательное построение прямоугольников, аппроксимирующих начальную фигуру. На каждом элементарном отрезке высота прямоугольника равна значению функции $f(x)$ в середине отрезка

Погрешность метода

Обычно, вычисленное значение интеграла отличается от точного, определенного аналитически. Погрешность возникает вследствие того, что на каждом элементарном отрезке функция $f(x)$ аппроксимируется постоянной функцией g, а величина погрешности ε определяется как интеграл от погрешности приближения по следующей формуле:

$$\varepsilon = \left|\int_a^b f(x)dx - I\right| \le (b-a)M\frac{h}{4},$$

где M – супремум $|f'(x)|$ на $[a, b]$, I – вычисленное значение интеграла.

Из приведенной формулы следует важное заключение: погрешность метода прямоуголь-

ников обратно пропорциональна числу разбиений отрезка интегрирования. Так, чтобы полученная в результате вычислений погрешность не превышала заданное значение ε, достаточно разбить отрезок интегрирования на n элементарных отрезков, получив значение n с помощью следующих преобразований:

$$(b-a)M\frac{h}{4}\le\varepsilon \Rightarrow \frac{(b-a)^2M}{4n}\le\varepsilon \Rightarrow n=\left[\frac{(b-a)^2M}{4\varepsilon}\right]+1.$$

АЛГОРИТМИЗАЦИЯ МЕТОДА

Так как для случая вычислений с погрешностью, не превосходящей заданное значение ε, необходимое число разбиений можно установить априори, достаточно реализовать лишь один алгоритм – для заданного числа разбиений n:

Шаг 1. *Инициализация:* Вводятся значения концов отрезка интегрирования a, b и число разбиений n.

Примечание. Для случая вычислений с погрешностью, не превосходящей заданное значение ε, необходимое число разбиений n считается по формуле $n=\left[\frac{(b-a)^2M}{4\varepsilon}\right]+1$.

Шаг 2. Вычисляется длина элементарного отрезка $h=\frac{|b-a|}{n}$. $S\Leftarrow 0$.

Шаг 3. Для всех i от 0 до n–1:

a) Вычисляется значение $z_i \Leftarrow a+ih+\frac{h}{2}$.

b) Вычисляется площадь элементарного прямоугольника: $S_i \Leftarrow f(z_i)\times h$.

c) Вычисленная площадь добавляется к сумме предыдущих: $S \Leftarrow S + S_i$.

Шаг 4. Выводится на экран вычисленная общая площадь S. КОНЕЦ.

Пример 1: Требуется написать программу для вычисления интеграла $\int_1^2 \frac{x^4}{\sqrt{1+x}}\,dx$ для 20, 40, 80 и 160 разбиений отрезка интегрирования методом прямоугольников. Для каждого числа разбиений необходимо вывести вычисленное значение с шестью знаками после запятой.

Решение: Так как число разбиений указано в условии задачи, нет необходимости в предварительной математической обработке.

```
program cn13;
const r=4;
var S, a, b, h : real;
   j,i,n:integer;
function f(x:real): real;
begin
   f:=x*x*x*x/sqrt(1+x);
end;
begin
   a:=1; b:=2; n:=10;
   for j:=1 to r do
   begin
   S:=0; n:=n*2; h:=(b-a)/n;
   for i:=0 to n-1 do
```

```
    S:=S+ h*f(a + i*h + h/2);
    writeln (' n=',n,' I=',S:0:6);
    end;
end.
```

Результаты:

```
n=20 I=3.788513
n=40 I=3.789629
n=80 I=3.789908
n=160 I=3.789977
```

Пример 2: Требуется написать программу для приближенного вычисления интеграла $\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\sin(\cos x^2)}{2} dx$ методом прямоугольников с погрешностью, не превосходящей заданное значение ε. Расчеты будут прерваны тогда, когда выполнится условие: $\frac{\pi}{8} h \le \varepsilon$; $h = \frac{|b-a|}{n}$, где n – число выполненных разбиений.

Переменные a, b, ε задаются непосредственно в тексте программы.

Решение: Из условий задачи определяется число необходимых разбиений:

$$\frac{\pi |b-a|}{8n} \le \varepsilon; \quad \Rightarrow \quad n = \left[\frac{\pi |b-a|}{8\varepsilon} \right] + 1.$$

```
program[1] cn14;
var S, a, b, e, h : real;
   j,i,n:integer;
function f(x:real): real;
begin    f:=sin(cos (x*x))/2;end;
begin    a:=-pi/2; b:=pi/2; e:=0.0001;
         n:=round(pi*(abs(b-a))/(8*e))+1;
         S:=0;
         h:=(b-a)/n;
         for i:=0 to n-1 do
         S:=S+ h*f(a + i*h + h/2);
         writeln ('n=',n,' I=',S:0:6);
end.
```

Результаты:

```
n=12338 I=0.729729
```

Вопросы и упражнения

❶ Аргументируйте необходимость численного интегрирования при вычислении определенных интегралов.

❷ Какой процесс является основой метода прямоугольников для приближенного вычисления интеграла?

❸ Обоснуйте отношение между числом разбиений отрезка интегрирования и погрешностью вычислений методом прямоугольников.

[1] Здесь и в других задачах данного издания используется предопределенная в среде Turbo Pascal функция `pi`. Пользователи других языков программирования либо других компиляторов Паскаля должны явно объявить в программе данную константу. Например: `const pi=3.141`.

❹ Напишите программу для вычисления определенного интеграла методом прямоугольников, для 200, 2000, 20000 разбиений отрезка интегрирования. Для каждого заданного числа разбиений необходимо вывести вычисленное значение интеграла с шестью цифрами после запятой.

a) $\int_{-1}^{1}\left(\frac{1+x^2}{1+x^4}\right)dx;$ c) $\int_{0}^{2\pi}\frac{dx}{(2+\cos x)(3+\cos x)};$ e) $\int_{0}^{1}\frac{dx}{(x+1)\sqrt{x^2+1}};$

b) $\int_{1}^{9}\left(x\sqrt[3]{1-x}\right)dx;$ d) $\int_{0}^{2\pi}\frac{dx}{\sin^4 x+\cos^4 x};$ f) $\int_{0}^{\ln 2}\sqrt{e^x-1}\,dx.$

Объясните расхождения в полученных, для разного числа разбиений, результатах.

Проверьте полученные результаты с помощью доступного on-line приложения для вычисления определенных интегралов: www.solvemymath.com/online_math_calculator/calculus/defnite_integral.

❺ Напишите программу для приближенного вычисления интеграла методом прямоугольников, с погрешностью, не превосходящей заданное значение ε = 0,001; ε = 0,0001; ε = 0,00001. Значение *M* известно априори:

a) $\int_{0}^{3}\left(x^2+3x\right)dx,$ $M=9;$ c) $\int_{1}^{2}\left(x^3-7x\right)dx,$ $M=5;$

b) $\int_{1}^{2}\left(x^3-2x^2+3x\right)dx,$ $M=7;$ d) $\int_{2}^{7}\left(x^2\sqrt{x-1}\right)dx,$ $M=36.$

Для каждого значения погрешности необходимо вывести вычисленное значение интеграла и число разбиений, при котором получено данное приближение.

❻ Границы земельного участка определяются вертикальными прямыми $x=-2$ и $x=0$, сверху – графиком функции $f(x)=x^3-5x+3\cos x$, снизу – графиком функции $f(x)=x^3-x^2$. Напишите программу для вычисления площади участка методом прямоугольников для:
a) 10000 разбиений; b) ε = 0,00001.

5.2. Модификации метода прямоугольников

В некоторых случаях точка, определяющая высоту прямоугольника, аппроксимирующего интеграл на элементарном участке, может не быть серединой отрезка, а одним из его концов. Тогда, если аппроксимация проводится через левые концы (*рис. 5.4*) элементарных отрезков, площадь элементарного прямоугольника $S_i=hf(x_i)$, а определенный интеграл аппроксимируется суммой:

$$I_{st}=\sum_{i=0}^{n-1}hf(x_i)=h\sum_{i=0}^{n-1}f(a+ih),$$

где $h=\frac{|b-a|}{n}$, n – число разбиений промежутка интегрирования, а применяемый метод называется *методом левых прямоугольников*.

Если аппроксимация проводится через правые концы (*рис. 5.5*) элементарных отрезков, площадь элементарного прямоугольника $S_i = hf(x_{i+1})$, а определенный интеграл аппроксимируется:

$$I_{dr}=\sum_{i=1}^{n}hf(x_i)=h\sum_{i=1}^{n}f(a+ih),$$

где $h=\frac{|b-a|}{n}$, n – число разбиений промежутка интегрирования, а применяемый метод называется *методом правых прямоугольников*.

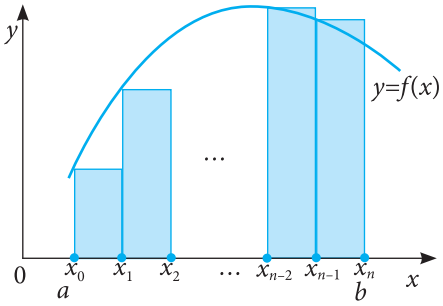


Рис. 5.4. Приближение определенного интеграла левыми прямоугольниками

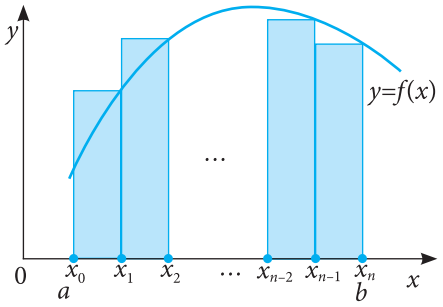


Рис. 5.5. Приближение определенного интеграла правыми прямоугольниками

Погрешность вычислений для модификаций метода прямоугольников

В случае аппроксимации определенного интеграла площадями левых (правых) прямоугольников основная формула оценивания погрешности изменяется незначительно:

$$\left|\int_a^b f(x)dx - I\right| \le (b-a)\,M\,\frac{h}{2},$$

где M – супремум $|f'(x)|$ на $[a, b]$, I – вычисленное значение интеграла путем приближения левыми (правыми) прямоугольниками.

Так же, как и в случае серединных прямоугольников, число разбиений, необходимое для получения решения с погрешностью, не превосходящей заданное значение ε, можно получить из формулы погрешности:

$$n = \left\lceil M\,\frac{(b-a)^2}{2\varepsilon} \right\rceil + 1.$$

АЛГОРИТМИЗАЦИЯ МЕТОДА (левые прямоугольники)[2]

Шаг 1. *Инициализация:* Вводятся значения концов промежутка интегрирования a, b и число требуемых разбиений n.

Примечание: В случае вычислений интеграла с погрешностью, не превосходящей заданную погрешность ε, число разбиений n вычисляется из формулы $n = \left\lceil \frac{(b-a)^2 M}{2\varepsilon} \right\rceil + 1$.

Шаг 2. Вычисляется длина элементарного отрезка $h = \frac{|b-a|}{n}$. $S \Leftarrow 0$.

Шаг 3. Для всех i от 0 до n–1:

a) Вычисляются значения $x_i \Leftarrow a + ih$.

b) Вычисляется площадь элементарного прямоугольника: $S_i \Leftarrow f(x_i) \times h$.

c) Вычисленная площадь добавляется к суммарной площади предыдущих прямоугольников: $S \Leftarrow S + S_i$.

Шаг 4. Выводится вычисленная общая площадь S. КОНЕЦ.

Пример 3: Вычислить определенный интеграл $\int_0^{\pi} (x\sin(x))^2\,dx$ методом прямоугольников (вариация левых прямоугольников) для 10, 100, 1000 разбиений. Присваивание начальных значений осуществляется в тексте программы. Для каждого случая числа разбиений на экран выводятся вычисленное значение интеграла и число разбиений, разделенные пробелом.

[2] Случай правых прямоугольников отличается от приведенного лишь на 3-м шаге. В этом случае индекс i изменяется в диапазоне: от 1 до n.

```
program cn15;
const r=3;
var S, a, b, h : real;
   j,i,n:integer;
function f(x:real): real;
begin    f:=sqr((x* sin(x)));end;
begin    a:=0; b:=pi; n:=1;
         for j:=1 to r do
         begin S:=0; n:=n*10;
         h:=(b-a)/n;
         for i:=0 to n-1 do
         S:=S+ h*f(a + i*h);
         writeln ('n=',n,' I=',S:0:6);
         end;
end.
```

Результаты:

```
n=10 I=4.381796
n=100 I=4.382315
n=1000 I=4.382315
```

Пример 4: Требуется вычислить определенный интеграл $\int_0^1 \sin x \sin 2x \sin 3x dx$ методом прямоугольников (разновидность правых прямоугольников) с погрешностью, не превышающей заданное значение ε = 0,001. Присваивание начальных значений осуществляется в тексте программы. На экран выводятся вычисленное значение интеграла и необходимое число разбиений, разделенные пробелом.

Решение: Необходимое число разбиений можно определить аналитически. В результате элементарных математических выкладок определим значение $M = 6$:

$$n = \left[M \frac{(b-a)^2}{2\varepsilon} \right] + 1.$$

```
program cn16;
var S, a, b, e, h, M : real;
   j,i,n:integer;
function f(x:real): real;
begin  f:=sin(x)* sin(2*x)* sin(3*x);end;
begin  a:=0; b:=1; e:=0.001; M:=6;
       n:=trunc(M*(b-a)*(b-a)/(2*e))+ 1;
       S:=0; h:=(b-a)/n;
       for i:=1 to n do
       S:=S+ h*f(a + i*h);
       writeln ('n=',n,' I=',S:0:6);
end.
```

Результаты:

```
n=3001 I=0.278729
```

Вопросы и упражнения

❶ В чем суть отличий между методом серединных прямоугольников и его модификациями?

❷ Каково отношение между числом разбиений промежутка иртегрирования и погрешностью вычислений в модификациях метода прямоугольников?

❸ Какая из модификаций метода прямоугольников является более точной? Почему?

❹ Напишите программу, вычисляющую определенный интеграл методами серединных, левых и правых прямоугольников. Примените различное число разбиений (например: 10, 100, 1000).

a) $\int_1^3 (x\ln x)^2 dx$; b) $\int_1^6 \left(x\sqrt{x-1}\right) dx$; c) $\int_0^1 \left(x^2\sqrt{1+3x^2}\right) dx$;

d) $\int_0^{\pi} \left(e^x \cos^2 x\right) dx$; e) $\int_0^{\pi} (x\sin x)^2 dx$; f) $\int_0^{2\pi} \frac{dx}{(2+\cos x)(3+\sin x)}$.

Объясните расхождения в полученных, для разного числа разбиений, результатах.

Проверьте полученные результаты с помощью доступного on-line приложения для вычисления определенных интегралов: www.solvemymath.com/online_math_calculator/calculus/defnite_integral.

❺ Напишите программу для приближенного вычисления интеграла методом прямоугольников с погрешностью, не превосходящей заданное значение ε = 0,005; ε = 0,0005; ε = 0,00005. Значение *M* определите аналитически:

a) $\int_1^3 \left(3x^2 + x + 2\right) dx$; b) $\int_1^2 \left(2x^3 - x^2 + 5x\right) dx$; c) $\int_{-3}^{-1} \left(2x^3 - 3x\right) dx$.

Для каждого значения погрешности необходимо вывести вычисленное значение интеграла и число разбиений, при котором получено данное приближение.

❻ Напишите программу, определяющую, какой из следующих определенных интегралов является больше. Примените вариации метода прямоугольников:

a) $\int_0^{\pi} \left(e^{-x^2}\cos^2 x\right) dx$ или $\int_{\pi}^{2\pi} \left(e^{-x^2}\cos^2 x\right) dx$; b) $\int_0^1 \left(e^{-x}\right) dx$ или $\int_0^1 \left(e^{-x^2}\right) dx$.

5.3. Формула трапеций

В предыдущих параграфах для аппроксимации определенного интеграла были применены прямоугольники – геометрические фигуры, площадь которых вычисляется с помощью элементарных формул. Недостатком этого метода является большое число разбиений, необходимое для получения достаточно точных результатов. Более логично было бы аппроксимировать интеграл другими геометрическими фигурами, площадь которых тоже вычисляется с помощью элементарных формул. Одной из таких фигур является трапеция (*рис. 5.6*).

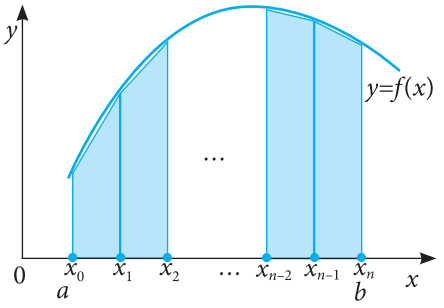


Рис. 5.6. На каждом элементарном отрезке разбиения промежутка [*a, b*] значение определенного интеграла аппроксимируется площадью трапеции. Верхняя сторона трапеции это отрезок с концами в точках с координатами $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$

Для численного интегрирования $\int_a^b f(x)dx$, как и в случае метода прямоугольников, разобьем промежуток интегрирования на *n* равных отрезков. Длина элементарного отрезка определяется по формуле $h = \frac{|b-a|}{n}$, а значения x_i

точек, задающих разбиение, – по формуле $x_i = a + ih$, $i = 0, \ldots, n$.

В результате разбиения получается ряд трапеций с равными высотами (h). Основаниями i-й трапеции служат вертикальные отрезки, соединяющие точки x_i и x_{i+1} с графиком функции $f(x)$. Длины оснований равны, соответственно, $f(x_i)$ и $f(x_{i+1})$ (*рис. 5.7*).

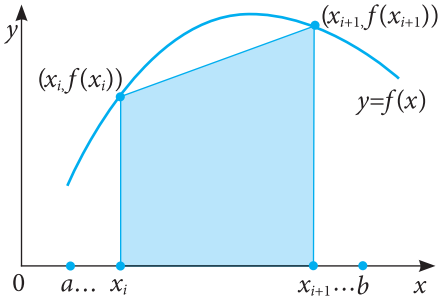


Рис. 5.7. Аппроксимация значения определенного интеграла, на элементарном промежутке, площадью трапеции, определяемой осью $0x$, прямыми $x = x_i$, $x = x_{i+1}$ и отрезком с концами в точках $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$

Площадь элементарной трапеции, определенной точками x_i и x_{i+1} подсчитывается по формуле

$$S_i = h\frac{f(x_i) + f(x_{i+1})}{2}.$$

Следовательно, на промежутке $[a, b]$ вычисленное значение интеграла будет:

$$I_{tr} = \frac{h}{2}\sum_{i=0}^{n-1}\left(f(x_i) + f(x_{i+1})\right).$$

Погрешность метода

Как и в случае метода прямоугольников, погрешность вычисления метода трапеций исследуется путем интегрирования погрешностей, получаемых в результате аппроксимации, на каждом элементарном промежутке функции $f(x)$ некоторой линейной функцией.

Погрешность интегрирования на промежутке $[a, b]$ состоит из суммы погрешностей интегрирования на элементарных промежутках и задается формулой:

$$\left|\int_a^b f(x)dx - I_{tr}\right| \le (b-a)\frac{M}{12}h^2,$$

где M – супремум $f''(x)$ на $[a, b]$, h – длина элементарного отрезка.

В случае, если требуется приближенное вычисление интеграла методом трапеций с погрешностью, не превосходящей заданное значение ε, необходимое число разбиений промежутка интегрирования можно определить априори из отношения $(b-a)\frac{M}{12}h^2 \le \varepsilon$, эквивалентного $n \ge \sqrt{\frac{(b-a)^3 M}{12\varepsilon}}$, откуда $n = \left[\sqrt{\frac{(b-a)^3 M}{12\varepsilon}}\right] + 1$.

АЛГОРИТМИЗАЦИЯ МЕТОДА

Шаг 1. *Инициализация:* Вводятся значения концов промежутка интегрирования a, b и число требуемых разбиений n.

Примечание: В случае вычислений интеграла с погрешностью, не превосходящей заданную величину ε, число разбиений n вычисляется из формулы $n = \left[\sqrt{\frac{(b-a)^3 M}{12\varepsilon}}\right] + 1$.

Шаг 2. Вычисляется длина элементарного отрезка $h = \frac{|b-a|}{n}$. $S \Leftarrow 0$.

Шаг 3. Для всех i от 0 до n–1:

a) Вычисляются значения $x_i \Leftarrow a+ih$; $x_{i+1} \Leftarrow a + (i+1)h$.

b) Вычисляется площадь трапеции: $S_i \Leftarrow h\dfrac{f(x_i)+f(x_{i+1})}{2}$.

c) Вычисленная площадь добавляется к суммарной площади предыдущих трапеций: $S \Leftarrow S + S_i$.

Шаг 4. Выводится вычисленная общая площадь S. КОНЕЦ.

Пример 1: Требуется вычислить определенный интеграл $\int_0^{2\pi} \dfrac{1}{\sin^4 x + \cos^4 x}dx$ методом трапеций для 20, 40, 80 разбиений промежутка интегрирования. Присваивание начальных значений осуществляется в тексте программы. Для каждого случая числа разбений на экран выводятся вычисленное значение интеграла и число разбиений, разделенные пробелом.

Решение: Число разбиений от случая к случаю вычисляется по формуле $n \Leftarrow 2n$.

```
program cn17;
const r=3;
var S, a, b, h : real;
   j,i,n:integer;
function f(x:real): real;
begin    f:=1/(sqr(sqr(sin(x)))+sqr(sqr(cos(x))));end;
begin    a:=0; b:=2*pi; n:=10;
         for j:=1 to r do
         begin
         S:=0; n:=n*2;
         h:=(b-a)/n;
         for i:=0 to n-1 do
         S:=S+ h*(f(a + i*h)+f(a+(i+1)*h))/2;
         writeln ('n=',n,' I=',S:0:11);
         end;
end.
```

Результаты:

```
n=20 I=8.88312405500
n=40 I=8.88576626920
n=80 I=8.88576587630
```

Пример 2: Требуется вычислить определенный интеграл $\int_1^3 (3x^3 - 7x^2 + 2)dx$ методом трапеций с погрешностью, не превышающей значение ε = 0,001. Присваивание начальных значений осуществляется в тексте программы. На экран выводятся вычисленное значение интеграла и число разбиений, необходимое для достижения требуемой точности.

Решение: Необходимое для достижения требуемой точности число разбиений вычисляется по формуле $n = \left[\sqrt{\dfrac{(b-a)^3 M}{12\varepsilon}}\right] + 1$. Путем элементарных математических расчетов устанавливается $M = 42$.

```
program cn18;
var S, a, b, e, h, M : real;
   j,i:integer;
   n:longint;
function f(x:real): real;
begin    f:=3*x*x*x-7*x*x+2;end;
begin    a:=1; b:=3; e:=0.001; M:=42;
         n:=trunc(sqrt(M*(b-a)*(b-a)*(b-a)/(12*e)))+ 1;
         S:=0;
         h:=(b-a)/n;
         for i:=0 to n-1 do
         S:=S+ h*(f(a + i*h)+ f(a + (i+1)*h))/2;
         writeln ('n=',n,' I=',S:0:9);
end.
```

Результаты:

```
n=168 I=3.333852986
```

Вопросы и упражнения

❶ Укажите условия, необходимые для применения метода трапеций.

❷ Чем различаются методы прямоугольников и трапеций?

❸ Опишите алгоритм метода трапеций для приближенного вычисления интеграла.

❹ Какой из изученных численных методов для вычисления определенного интеграла является наиболее точным? Аргументируйте.

❺ Вычислите определенные интегралы методом трапеций для 10, 20, 30 разбиений промежутков интегрирования:

a) $\int_1^4 e^{\sqrt{x}+4}\,dx;$ c) $\int_2^4 \left(x^3+\frac{2x}{x^2+1}\right)dx;$ e) $\int_0^1 \sqrt{\sin x+\cos x}\,dx;$

b) $\int_1^2 \sqrt{3x^2-5x+4}\,dx;$ d) $\int_0^{\pi} (x\sin x)^2\ \cos x\,dx;$ f) $\int_{\frac{2\pi}{3}}^{\pi} \left(tgx^3+\cos x\sin x\right)^2 dx.$

❻ Напишите программу для вычисления определенного интеграла методом трапеций с погрешностью, не превосходящей заданное значение ε = 0,003; ε = 0,0003; ε = 0,00003. Значение M определите аналитически:

a) $\int_1^3 \left(e^x+3\right)dx;$ b) $\int_1^2 \left(3x^2-5x+4\right)dx;$ c) $\int_2^3 \left(x^3+3x-8\right)dx.$

Контрольный тест

I Выберите правильный ответ:

1. Метод прямоугольников позволяет вычисление:
a) определенного интеграла; c) определенного и неопределенного интегралов.
b) неопределенного интеграла;

2. Для одного и того же числа разбиений промежутка интегрирования меньшая погрешность будет получена для:
a) метода серединных прямоугольников; c) метода правых прямоугольников;
b) метода левых прямоугольников; d) метода трапеций.

3. Необходимое число разбиений промежутка интегрирования для приближенного вычисления определенного интеграла с погрешностью, не превосходящей заданное значение ε, можно определить априори:
a) только для метода трапеций; c) для обоих методов;
b) только для метода прямоугольников; d) ни для одного из этих методов.

4. Пусть $\int_a^b f(x)dx$, n – число разбиений промежутка интегрирования, а $h = \frac{|b-a|}{n}$. Установите соответствие между формулами из левого столбца и названиями соответствующих методов, в которых эти формулы используются, из правого столбца:

$I = \frac{h}{2}\sum_{i=0}^{n-1}\left(f(x_i)+f(x_{i+1})\right)$	Серединных прямоугольников
$I = h\sum_{i=0}^{n-1} f\left(a+ih+\frac{h}{2}\right)$	Левых прямоугольников
$I = h\sum_{i=0}^{n-1} f\left(a+ih\right)$	Правых прямоугольников
$I = h\sum_{i=1}^{n} f\left(a+ih\right)$	Трапеций

5. Пусть $\int_a^b f(x)dx$, ε – максимально допустимая погрешность вычислений, M – супремум $|f'(x)|$ на $[a, b]$. Какая из приведенных ниже формул позволяет определить априори необходимое число разбиений промежутка интегрирования, позволяющее вычислить методом серединных прямоугольников приближенное значение интеграла с погрешностью, не превосходящей ε:

a) $n = \left[\sqrt{\frac{(b-a)^3 M}{12\varepsilon}}\right] + 1$; b) $n = \left[\frac{(b-a)^2 M}{4\varepsilon}\right] + 1$; c) $n = \left[\frac{(b-a)^2 M}{2\varepsilon}\right] + 1$.

II Вычислите с помощью написанных ранее программ определенные интегралы. Для каждого вычисленного значения должна выводиться строка, содержащая слово „Răspuns:", после которого, разделенное пробелом, вычисленное решение, с 6-ю десятичными знаками после запятой:

1. Метод левых прямоугольников: $\int_2^4 \sqrt[3]{x^2 - 7x + 8}dx$ для 1000 разбиений промежутка интегрирования.

2. Метод серединных прямоугольников (трапеций)[3]: $\int_0^{\frac{3\pi}{2}} \sqrt{\sin^2 x + x^3 - 8}dx$ для 5000 разбиений промежутка интегрирования.

[3] Дополнительно.

ГЛАВА 6

ОСНОВНЫЕ ПОНЯТИЯ О БАЗАХ ДАННЫХ

Изучив данную главу, вы сможете:

- объяснять смысл понятий *элементарные данные, структура данных, база данных*;
- распознавать элементарные данные, структуры данных и базы данных;
- описывать структуры иерархических, сетевых и реляционных баз данных;
- распознавать концептуальную модель базы данных.

6.1. Понятия и концепты о данных и о базах данных

6.1.1. Элементарные данные и структуры данных

Данное – это модель представления информации, доступной для обработки некоторым процессором (человеком, программой и др.), модель, оперируя с которой можно получить новую информацию.

Элементарное данное – это сущность, неделимая по отношению к представляемой ею информации и по отношению к логическому (программе) либо физическому (ЦПУ) процессору.

С логической точки зрения элементарное данное характеризуется:

a) *идентификатором*;

b) *значением*;

c) *атрибутами*.

Идентификатор данного используется для его вызова (доступа к нему).

Значение данного – это содержимое зоны памяти, в которой это данное размещено. ***Область определения данного*** – это множество значений, которые может принимать в процессе обработки это данное.

Атрибуты данного определяют его специфические характеристики. Одним из наиболее важных атрибутов данного является его ***тип***, определяющий принадлежность к некоторому классу данных. Каждому типу данных соответствует некоторая модель внутреннего представления.

Пример 1: Объявление `var c: integer` в языке Паскаль определяет переменную `c`, которая в любой момент работы программы идентифицирует элементарное данное целого типа, „занимающее" в памяти компьютера зону величиной в два байта. Областью определения типа *integer* является множество чисел {–32 768, –32 767, ..., 32 767}. В результате выполнения оператора `c:= 8` в зоне, выделенной для хранения переменной, запомнится значение 8.

Пример 2: Значения простых типов языка Паскаль являются элементарными данными.

Структура данных – это набор данных, между элементами которых установлены некоторые отношения, определяющие методы их идентификации и обработки. Структура данных может состоять из элементарных данных, а также из других структур. Каждая компонента структуры данных идентифицируется с помощью идентификатора или положения внутри структуры.

Пример 3: Значения типа *record* из Паскаля являются структурами данных, компоненты которых (поля) могут вызываться по имени, то есть по идентификатору поля. Массив – это структура данных, компоненты которой (элементы массива) могут вызываться с помощью указания их положения (то есть индекса) в массиве.

Основными характеристиками некоторой структуры данных являются:

a) *Однородность*. Однородная структура состоит из компонент одинакового типа. Так, записи (значения типа *record*) – это неоднородные структуры, а массивы – однородные.

b) *Способ доступа к компонентам*. К компонентам некоторой стуктуры может быть прямой или последовательный доступ. Например, текстовые файлы являются структурами данных с последовательным доступом.

c) *Стабильность структуры*. Если в процессе обработки с помощью программы в некоторой структуре число элементов и связи между ними остаются неизменными, то такая структура является *статической*. В противном случае структура называется *динамической*. Например, односвязные списки – это динамические структуры, а записи – статические.

d) *Время существования*. Структуры данных могут быть постоянными или временными. Например, внешние файлы являются постоянными структурами, а массивы – временными.

6.1.2. Базы данных

До настоящего момента в школьном курсе информатики нам приходилось обрабатывать различные типы данных и структур, пользуясь средствами языка программирования либо программными приложениями (текстовым редактором, табличным процессором и др.). Наиболее сложной обрабатываемой структурой был файл.

Очевидно, данные о некоторой фирме, компании или предприятии можно сохранить в файле. Например, если речь идет о некотором лицее, то можно создать файлы *Учителя*, *Ученики*, *Классы*, *Предметы* и др., в которых можно сохранять, соответственно, данные об учителях, учениках, классах, учебных предметах и др. Для хранения информации о распределении учителей по классам понадобится еще один файл, *Учитель_Класс*, созданный на основании содержимого файлов *Учителя* и *Классы*. Предположим, что только вступивший в должность завуч создает такой файл. Если данные об учителях в файле *Учителя* содержат ошибки, то эти ошибки повторятся и в файле *Учитель_Класс*. Очевидно, эти же ошибки появятся и в файле *Расписание_уроков*. Обнаружив ошибки, завуч будет вынужден исправлять данные во всех трех файлах. Было бы значительно проще, если бы при исправлении ошибок в файле *Учителя*, автоматически обновлялись все файлы, „использовавшие" эту информацию. Это бы гарантировало ***целостность данных***.

Помимо *сложностей с обновлением*, метод обработки данных с помощью файлов имеет и другие недостатки:

- *Наличие одинаковых данных в нескольких независимых файлах* может привести к их дублированию, а следовательно – к большому расходу памяти.
- *Сложность получения спонтанной информации*, хранящейся в нескольких независимых файлах (например, для получения списка учителей женского пола, преподающих реальные предметы в 10-х классах реального профиля, необходимо обработать, как минимум, файлы *Учителя*, *Классы*, *Предметы*).
- *Несовместимые форматы файлов* замедляют обработку информации (например, файлы, сгенерированные с помощью различных языков программирования, могут иметь различные форматы, следовательно, понадобится программное приложение, преобразующее файлы к одинаковому формату).

Эти и другие проблемы можно решить путем использования структуры типа *база данных*.

База данных – это совокупность информации (данных), специальным образом организованной так, чтобы облегчить ее хранение, обработку и вывод. Информация в базах данных хранится в виде записей или файлов, между которыми существуют логические связи. Кроме, собственно, хранения данных, база данных содержит и их описание: тип, структуру, способ организации и взаимосвязи.

Вопросы и упражнения

❶ Назовите известные вам типы данных. Какие из них являются:
a) простыми типами данных;
b) структурными типами данных?

❷ Выберите элементарные данные: символ, действительное число, строка символов, массив чисел, элемент некоторого множества, целое число, значение *false*, файл.

❸ Какова область определения типа данных:
a) `integer`;
b) `char`;
c) `lit_M`, объявленный следующим образом: `type lit_M = 'A'.. 'Z';`
d) `boolean`?

❹ Охарактеризуйте структуры данных (однородность, способ доступа, стабильность и время существования):
a) множество;
b) массив;
c) запись;
d) односвязный список;
e) файл.

❺ Объясните смысл понятия *база данных*.

❻ Почему хранение и обработка информации о некоторой компании более приемлемы в виде базы данных, нежели в виде системы независимых файлов?

❼ Каковы отличия между базой данных и структурой данных?

6.2. Типы баз данных

Из следующей главы мы узнаем, что одним из этапов разработки базы данных является создание ее *концептуальной модели*, определяющей способ организации данных и не зависящей от физических параметров среды их хранения.

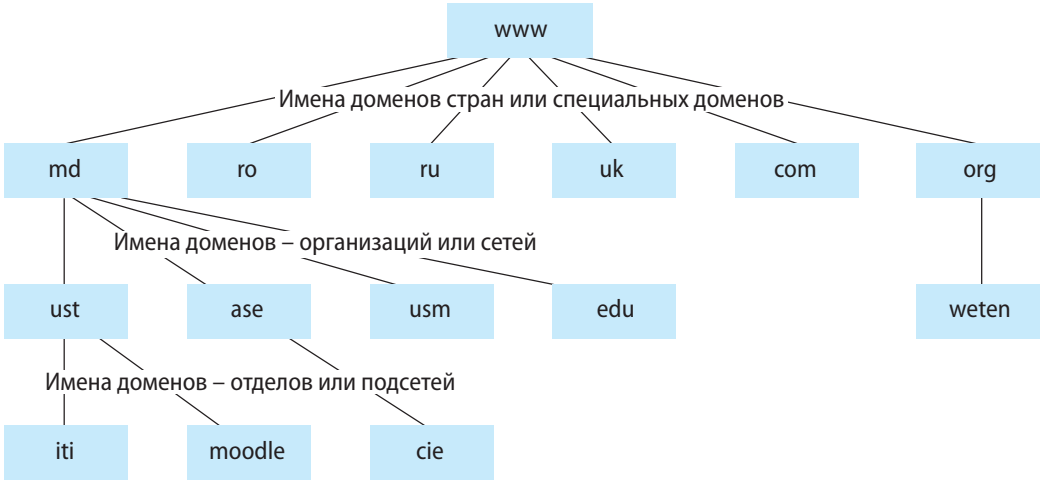
Концептуальная модель представляет собой общее описание базы данных с помощью естественного или математического языка, диаграмм в виде гистограмм и графиков, органиграмм и других средств, понятных тем, кто ее разрабатывает.

Наиболее распространенными концептуальными моделями являются:
a) *иерархическая* (или древовидная);
b) *сетевая*;
c) *реляционная.*

База данных называется *иерархической, сетевой* или *реляционной* в зависимости от ее концептуальной модели.

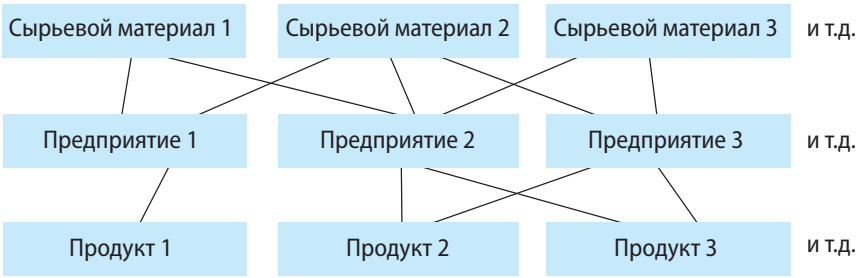
Сущности (коллекции данных) ***иерархической базы данных*** (*hierarhical database)* организованы в виде *узлов*, соединенных ветвями *дерева*. Каждый узел подчинен, максимум, одному узлу уровня, расположенного непосредственно над ним (называемого *материнским узлом*), и может быть связан с ни одним, одним или несколькими узлами (называемыми *дочерними узлами*) на иерархически нижнем уровне.

Пример: База данных с Интернет-адресами имеет следующую иерархическую модель:



В случае ***сетевой базы данных*** (*network database*) сущности также организованы иерархически, однако одному дочернему узлу может соответствовать несколько материнских узлов.

Пример: База данных некоторой индустриальной отрасли может иметь следующую модель:



Реляционная база данных (*relational database*) имеет самую гибкую концептуальную модель организации данных. Обход сущностей в этом случае не является иерархическим.

Реляционная модель проста, однако, очень строга с точки зрения математики. Впервые такая модель была предложена в 1970 году ученым Эдгаром Франк Коддом.*

Реляционная база данных состоит из таблиц, называемых *отношениями*. Каждая таблица состоит из строк и столбцов. Строки называются *записями*, а столбцы – *полями* данных. *Заголовок таблицы* определяет ее структуру. Под *созданием таблицы*, вообще говоря, подразумевается описание ее заголовка. На *рисунке 6.1* представлена часть схемы некоторой реляционной базы данных.

Рис. 6.1

Между таблицами реляционной базы данных существуют взаимосвязи (подробнее об этих связях вы узнаете из последующих глав).

Замечание: В дальнейшем будем рассматривать лишь реляционные базы данных.

Вопросы и упражнения

❶ Опишете структуры баз данных:
a) иерархических;
b) сетевых;
c) реляционных.

❷ Определите тип базы данных, зная что:
a) между любыми двумя сущностями базы существует связь;
b) сущности базы являются узлами двоичного дерева;
c) одна из сущностей связана со всеми остальными и других связей не существует;
d) вся информация базы содержится в десяти различных таблицах.

❸ Представьте схематически иерархическую модель базы данных, содержащей информацию о:
a) лицее;
b) продовольственном магазине;
c) библиотеке.

❹ Решите упражнение ❸ для случая:
a) сетевой модели;
b) реляционной модели.

❺ Определите тип базы данных с номерами телефонов стационарной телефонной связи страны.

* *Эдгар Франк Кодд* (23.08.1923, остров Портленд, Англия – 18.04.2003, Уильямс Айленд, Флорида, США) – американский информатик британского происхожденя. Работая для IBM, изобрел реляционную модель для управления базами данных. За вклад в развитие информатики получил в 1981 году премию Тьюринга, считающуюся „Нобелевской премией в области информатики ".

ГЛАВА 7

СОЗДАНИЕ И УПРАВЛЕНИЕ БАЗАМИ ДАННЫХ

Изучив данную главу, вы сможете:

- описывать этапы разработки базы данных;
- объяснять роль лиц, вовлеченных в процесс разработки базы данных;
- описывать структуру и функции систем управления базами данных;
- распознавать структуру таблиц реляционной базы данных;
- определять тип связи между двумя таблицами;
- выбирать либо создавать поле первичного ключа для таблицы;
- применять некоторые принципы проектирования базы данных при создании ее концептуальной модели.

7.1. Создание базы данных

7.1.1. Основные аспекты

Система базы данных представляет собой систему организации и обработки базы данных. Эта система сформирована, собственно, из базы данных и из совокупности программ, обеспечивающих управление и комплексную обработку данных.

Под **проектированием** или **созданием базы данных** подразумевается процесс создания системы этой базы.

Проектирование включает, как правило, следующие **этапы**:

1. Этап анализа. Анализируется область данных, формулируются требования к системе базы данных со стороны всех категорий ее пользователей, определяются операции, которыми будет управлять эта система. В результате данного анализа создается концептуальная модель базы данных. Также определяется структура основного и вспомогательных интерфейсов системы базы данных.

2. Этап программирования. Создаются логические компоненты (программы): основной интерфейс, приложения для ввода/обновления/обработки данных, приложения для запросов и для вывода информации.

3. Запуск и эксплуатация системы базы данных. База заполняется информацией (записями). Затем осуществляются операции по обновлению, консультации и последующей эксплуатации (а также, развитию) системы базы данных.

Создание и поддержку базы данных обычно осуществляет целая группа лиц: *администраторы* (определяют начальную структуру базы данных, способ хранения информации на физическом уровне; предоставляют пользователям права доступа к базе данных, обеспечивают безопасность данных; изменяют структуру и осуществляют дальнейшее развитие базы данных), *программисты* (пишут программы на языках программирования), *дизайнеры* (создают дизайн интерфейса базы данных) и др.

Замечание: Существуют современные компьютеризированные системы, называемые Системами Управления Базами Данных (СУБД), специально разработанные для создания баз данных. Подробнее о таких системах вы узнаете из следующего параграфа.

7.1.2. Проектирование сущностей реляционной базы данных

В предыдущей главе было отмечено, что сущностями реляционной базы данных являются **таблицы**, в которых хранятся *записи* (строки таблиц).

Заголовок таблицы определяет ее структуру. Под *созданием таблицы* подразумевается, вообще говоря, определение ее заголовка, то есть описание ее полей (столбцов).

Пример: Рассмотрим базу данных *Liceu*, в которой накоплена информация о некотором лицее. Таблица *Elevi* этой базы данных хранит информацию об учениках этого лицея:

Таблица **Elevi**

Cod_elev	Nume_elev	Pren_elev	Cod_clasa	Data_elev	Foto_elev	Telefon	Gen_elev
e001	Bacinschi	Sabina	c01	28.09.1993	Package	29-82-54	F
e002	Belobrov	Andreea	c01	18.10.1993	Package	44-26-48	F
e003	Brîncă	Carmen	c01	20.03.1993	Package	67-46-64	F

При описании каждого поля таблицы необходимо указать его *имя* и *тип*:

a) *имя поля* должно быть уникальным в пределах таблицы; данные, сохраняемые в поле, – однородными;

b) *тип поля* определяет тип его значений (числовой, строка символов, большой текст, календарное число, логический, изображение и др.).

Так, имя первого поля таблицы ***Elevi*** – *Cod_elev*, тип – *Строка символов*, а у пятого поля – имя *Data_elev,* тип – *Календарное число*.

Каждое значение из поля *Cod_elev* таблицы *Elevi* соответствует единственной записи.

Поле, значения которого однозначно определяют каждую запись, называется ***первичным ключом***. Очевидно, поле может быть первичным ключом только тогда, когда его любое значение не пусто и ни разу не повторяется.

Рекомендуется, чтобы в каждой таблице базы данных было поле первичного ключа. Иногда роль первичного ключа могут играть сразу несколько полей (если вместе они однозначно идентифицируют строки таблицы).

Если значения некоторого поля повторяются, то такое поле называется ***вторичным ключом***. Так, в таблице *Elevi* поле *Cod_elev* является первичным ключом, а *Cod_clasa* – вторичным.

Обычно, каждая таблица реляционной базы данных находится в некотором отношении, по крайней мере, с еще одной таблицей этой базы. Существует три типа **отношений между таблицами**:

a) *один-к-одному*;

b) *один-ко-многим*;

c) *многие-ко-многим*.

В отношении типа ***один-к-одному*** (обозначается 1:1) каждой записи одной таблицы соответствует не больше одной записи из другой таблицы, и наоборот. Этот тип отношений возникает, когда таблица с большим количеством полей разбивается на две таблицы. В этом случае в первичных ключах обеих таблиц значения совпадают.

Например, между таблицами *Elevi* и *Adrese_elevi* задано отношение типа *один-к-одному*. Обе таблицы имеют поле первичного ключа *Cod_elev* с соответствующими одинаковыми данными.

Таблица **Adrese_Elevi**

Cod_elev	Loc_elev	Strada_elev	Nr_casa_elev	Ap_elev
e001	Chişinău	Mihail Sadoveanu	40	23
e002	Stăuceni	Viei	9	

Схематически этот тип отношения представлен на *рисунке 7.1*.

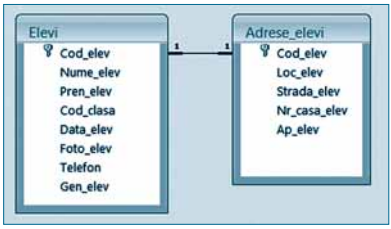


Рис. 7.1

В случае отношения ***один-ко-многим*** (обозначается 1:∞ либо 1:M) каждой записи из одной таблицы могут соответствовать несколько записей из другой таблицы, а каждой записи из второй таблицы соответствует, максимум, одна запись из первой. Для того чтобы установить отношение такого типа, первая таблица должна иметь поле первичного ключа, а вторая таблица – поле вторичного ключа, совместимого с типом первичного ключа первой таблицы.

Например, таблица *Clase* находится в отношении *один-ко-многим* с таблицей *Elevi*, потому что в каждом классе есть „несколько" учеников, а каждый ученик „принадлежит" лишь одному классу.

Таблица **Clase**

Cod_clasa	Anul_de_studii	Nume_clasa	Cod_profil
c01	10	A	p1
c02	10	B	p1
c03	10	C	p2
c04	10	D	p2

Первичный ключ *Cod_clasa* таблицы *Clase* „совместим" со вторичным ключом *Cod_clasa* таблицы *Elevi*. Схематически этот тип отношения представлен на *рисунке 7.2*.

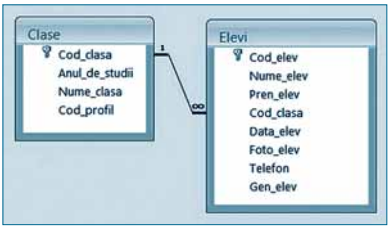


Рис. 7.2

Замечание: Вторичный ключ в случае отношения один-ко-многим называют также *внешним ключом*.

В отношении ***многие-ко-многим*** (обозначается ∞:∞ или M:M) каждой записи из одной таблицы могут соответствовать несколько записей из другой таблицы и наоборот. Например, таблицы *Clase* и *Discipline* находятся в отношеннии *многие-ко-многим,* так как в каждом классе преподаются несколько предметов и каждый предмет преподается в нескольких классах.

Таблица **Discipline**

Cod_disciplina	Nume_disciplina
d01	Matematica
d02	Informatica
d03	Chimia
d04	Fizica

Таблица **Prof_dis_clasa**

Cod_repart	Cod_clasa	Cod_dis	Cod_prof
r001	c01	d01	prof01
r002	c01	d02	prof01
r003	c01	d03	prof02
r004	c01	d05	prof04

Отношение *многие-ко-многим* между двумя таблицами может быть установлено с помощью двух отношений типа *один-ко-многим*. Каждая из двух таблиц связывается отношением *один-ко-многим* с третьей таблицей.

Например, отношение *многие-ко-многим* между таблицами *Clase* и *Discipline* реализуется с помощью таблицы *Prof_dis_clasa* (*рис. 7.3*).

Рис. 7.3

Замечание: Первая запись из таблицы *Prof_dis_clasa* содержит информацию о том, что в классе c01 (то есть в 10-м А классе) предмет d01 (то есть *Matematica*) преподается учителем prof01 (данные об этом преподавателе хранятся в таблице *Profesori*, стр. 74).

7.1.3. Принципы проектирования

Проектируя реляционную базу данных, программисты определяют таблицы таким образом, чтобы соблюдались некоторые принципы (например, чтобы они были приведены к, так называемым, *нормальным формам*).

• Один из этих принципов требует, чтобы данные в полях таблицы были **элементарными** (смотри параграф 6.1 главы 6). Из этих соображений адреса учеников и учителей хранятся в нескольких полях. Очевидно, обработка записей была бы затруднена, если бы весь адрес был записан в одном поле.

• Для того чтобы обновление записей было оптимальным (например, более быстрым), необходимо исключить из таблиц повторяющиеся поля (кроме полей, с помощью которых реализованы отношения между таблицами). Добиваются этого путем **разложения** соответствующей таблицы на несколько таблиц.

Например, данные об учителях, преподающих определенные предметы в определенных классах, накапливаются в четырех таблицах: *Profesori*, *Discipline*, *Clase* и *Prof_dis_clase*. Благодаря тому, что между данными таблицами существуют связи, для замены во всех записях фамилии учителя достаточно изменить ее в таблице *Profesori*.

Вопросы и упражнения

❶ Назовите этапы создания базы данных.
❷ Из каких сущностей состоит реляционная база данных?
❸ Объясните принципы проектирования реляционной базы данных.
❹ Чем отличается отношение M:M от отношения 1:M?
❺ Изучите таблицу *Elevi* базы данных *Liceu* и определите тип значений каждого ее поля.
❻ Выберите первичный ключ для таблицы:
a) *Класс_*12 с полями *Порядковый_номер, Фамилия_ученика, Имя_ ученика, Телефон_ ученика*;
b) *Сотрудники* с полями *Фамилия_сотрудника, Имя_сотрудника, Возраст_сотрудника, Пол_сотрудника, Номер_удостоверения_личности_сотрудника*;
c) *Стоянка_авто* с полями *Марка_авто, Регистрационный_номер_авто, Фото_авто, Фамилия_владельца_авто*;

d) *Страны* с полями *Название_страны, Площадь_поверхности_страны, Столица_страны, Количество_жителей_страны*.

❼ Какой тип отношений можно установить между таблицами:
a) *Вина* и *Производители_вин*;
b) *Книги* и *Авторы*;
c) *Стихи* и *Авторы*;
d) *Специальности* и *Университеты?*

❽ Спроектируйте базу данных, содержащую информацию:
a) из телефонного справочника;
b) о книгах вашей библиотеки;
c) об автомобилях;
d) о великих личностях нашей страны.

7.2. Системы управления базами данных (СУБД)

7.2.1. Основные понятия о системах управления базами данных

Система управления базами данных – это набор программ, позволяющих создание баз данных, ввод и обновление информации, обеспечение контроля доступа к сохраняемым в базе данным, управление данными, а также разработку приложений, использующих информацию из баз данных. СУБД выполняет следующие функции:

a) *определяет базу данных*, в том смысле, что описывает типы и структуры данных, отношения между ними и способы доступа к информации базы данных;
b) *обновляет базу данных*, то есть позволяет добавлять, редактировать и удалять записи;
c) *выполняет запросы к базе данных*, в результате которых данные упорядочиваются либо фильтруются, согласно определенным требованиям, сформулированным пользователями;
d) *создает новые данные*, получаемые путем расчетов, а также путем подведения итогов;
e) *создает отчеты,* простые и сложные, в виде таблиц, графиков, диаграмм и др.;
f) *обеспечивает обслуживание базы данных*, предполагающее восстановление базы в случае появления неисправностей, сжатие (или дефрагментацию), создание резервных копий как самих данных, так и всех объектов из базы данных;
g) *обеспечивает безопасность базы данных,* защиту от неавторизованного доступа, а также устанавливает права полного или частичного доступа.

• Программное приложение СУБД состоит из:
a) *Словаря данных* (*Data Dictionary*), содержащего описание структуры данных, используемых в базе данных;
b) *Языка запросов* (*Query Language*), обеспечивающего доступ к информации базы данных. Наиболее распространенным языком запросов (используемым в СУБД) является язык SQL (*Structured Query Language*).

• Аппаратные средства СУБД могут состоять:
a) *из одного компьютера* или
b) *из сети компьютеров*, в которой на главном компьютере (*сервере*) находятся программные компоненты СУБД, администрирующие базу данных и проверяющие права доступа к данным, а на остальных компьютерах – программы, предназначенные для пользователей.

Наиболее распространенными СУБД являются: Microsoft Office Access, Oracle, FoxPro, Paradox, dBase, Microsoft SQL Server и др.

7.2.2. Система управления базами данных Microsoft Office Access

Microsoft Office Access (далее просто Access) – это система управления реляционными базами данных, работающая в среде Windows. Ее первая версия появилась в 1993-м году. С помощью данной системы можно создавать сложные базы данных, так как для этого имеются все необходимые инструменты. По сравнению с другими СУБД, Access проста и удобна в применении, предоставляя программистам и обычным пользователям целый ряд вспомогательных программ, автоматизирующих разные этапы работы при создании запросов, форм, отчетов и других продуктов баз данных.

Запустить систему Access можно несколькими способами:

– выполнить двойной щелчок, поместив указатель мыши на ссылку Access на рабочем столе Windows;
– выбрать опцию Microsoft Office Access из стартового меню;
– выполнить двойной щелчок, поместив указатель мыши на любую пиктограмму файла с расширением .mdb.

После запуска Access можем **создать базу данных**. Для этого выберем *File* → *New* → *Blanc database*…

Появится окно *File New Database*. В текстовом поле *File name* пишем имя создаваемой базы данных. В раскрывающемся списке *Save in* выбираем папку, в которой будет храниться база.

В окне приложения Access появляется **окно базы данных** с ее именем.

Интерфейсы различных версий Access, несмотря на отличия, содержат следующие основные элементы:

a) заголовок окна Access;
b) панель меню (*File*, *Edit*, *View*, *Insert, Tools*, *Window*, *Help*) окна Access;
c) панель инструментов Access;
d) окно базы данных с закладками для создания и управления объектами семи классов Access (*Tables* – таблицы, *Queries* – запросы, *Forms* – формы, *Reports* – отчеты, *Pages* – страницы, *Macros* – макрокоманды, *Modules* – модули, содержащие операторы языка *Visual Basic for Applications*). Объекты базы данных становятся доступными с помощью соответствующих пиктограмм, расположенных на панели *Objects* окна базы данных;
e) панель меню окна базы данных (*Open*, *Design*, *New*).

Далее объясним как создается и управляется в среде Access некоторая база данных. С этой целью создадим базу данных *Liceu*, в которой будем хранить информацию об учениках и учителях некоторого лицея.

7.2.3. Структура базы данных *Liceu*

База данных *Liceu* будет состоять из 8 связанных между собой таблиц (*рис. 7.4*).

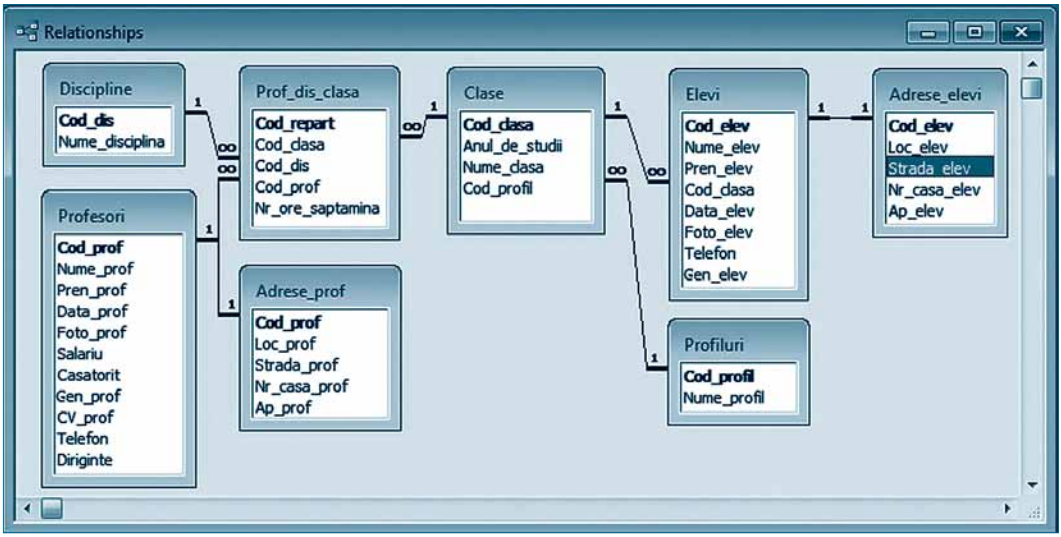
Представим структуру этих 8 таблиц. Заметим, что здесь представлены лишь несколько записей каждой из *таблиц 1–8*.

Таблица 1. **Profiluri**

Cod_profil	Nume_profil
p1	Real
p2	Umanist

Таблица 2. **Discipline**

Cod_dis	Nume_disciplina
d01	Matematica
d02	Informatica



Puc.7.4

Таблица 3. **Profesori**

Cod_ prof	Nume_ prof	Pren_ prof	Data_prof	Foto_ prof	Salariu	CV_prof	Casatorit	Gen_ prof	Telefon	Diri-ginte
prof01	Albu	Ion	01.03.1968	Package	3500	Grad...	Yes	M	64-41-72	CO2
prof02	Moraru	Vasile	11.05.1973	Package	3000	S-a năs...	No	M	29-10-16	

Таблица 4. **Adrese_prof**

Cod_prof	Loc_prof	Strada_prof	Nr_casa_prof	Ap_prof
prof01	Cricova	Vinului	89	
prof02	Chişinău	Grenoble	81	77

Таблица 5. **Clase**

Cod_clasa	Anul_de_studii	Nume_clasa	Cod_profil
c01	10	A	p1
c02	10	B	p1

Таблица 6. **Elevi**

Cod_elev	Nume_elev	Pren_elev	Cod_clasa	Data_elev	Foto_elev	Telefon	Gen_elev
e001	Bacinschi	Sabina	c01	28.09.1998	Package	29-82-54	F
e002	Belobrov	Andreea	c01	18.10.1998	Package	44-26-48	F

Таблица 7. **Adrese_elevi**

Cod_elev	Loc_elev	Strada_elev	Nr_casa_elev	Ap_elev
e001	Chişinău	Mihail Sadoveanu	40	23
e002	Stăuceni	Viei	9	

Таблица 8. **Prof_dis_clasa**

Cod_repart	Cod_clasa	Cod_dis	Cod_prof
r001	c01	d01	prof01
r002	c01	d02	prof01

Замечание: Таблицы базы данных можно скачать по адресу www.bd.ust.md.

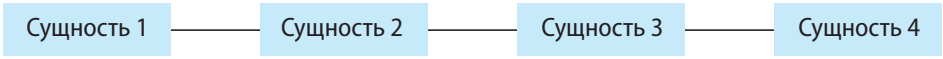
Вопросы и упражнения

❶ Опишите структуру системы управления базами данных.
❷ Объясните функции систем управления базами данных.
❸ Как можно запустить на выполнение СУБД Access?
❹ Проанализируйте интерфейс СУБД Access и дайте характеристику классов объектов Access.
❺ Рассмотрите таблицы базы данных *Liceu*, установите:
a) профиль класса, в котором учится Sabina Bacinschi;
b) фамилию учителя, преподающего информатику в 10-м A классе;
c) название предмета, преподаваемого учителем Ion Albu;
d) адрес классного руководителя 10-го A класса.

Контрольный тест

1. Какие из следующих значений являются элементарными данными в языке Паскаль:
a) строка символов 'Informatica';
b) символьная константа 'E';
c) логическое значение *false*;
d) вектор *V* с компонентами типа *byte;*
e) константа *Pi*;
f) число 32109?

2. Определите тип базы данных, в которой сущности связаны между собой как на рисунке: Аргументируйте ответ.



3. Опишите структуру таблицы, в которой можно хранить оценки по математике, полученные в течение семестра учениками 12-го класса.

4. Выберите первичный ключ для таблицы:
a) *Список учеников 12-го класса* с полями *Порядковый_номер, Фамилия, Имя, Дата_рождения, Пол*;
b) *Web-страницы* с полями *Название_страницы, Адрес_страницы, Язык_страницы, Автор_страницы*;
c) *Фильмы* с полями *Название_фильма, Автор_фильма, Жанр_фильма, Продолжительность_фильма*.

5. Определите тип отношения между таблицами:
a) *Марки_авто* и *Владельцы_авто*;
b) *Учителя* и *Ученики*;
c) *Дети* и *Мамы*;
d) *Страны* и *Столицы*;
e) *Каналы_ТВ* и *Фильмы*.

6. a) Спроектируйте таблицы базы данных *Дневник ученика*, в которой бы хранилась информация из дневника учащегося.
b) Какие типы отношений существуют между спроектированными таблицами?

ГЛАВА 8

ТАБЛИЦЫ – ОСНОВНЫЕ ОБЪЕКТЫ БАЗЫ ДАННЫХ

Изучив данную главу, вы сможете:

- создавать таблицы;
- изменять структуру таблиц;
- задавать свойства полей таблиц;
- задавать условия на значение вводимых в таблицы данных;
- задавать шаблоны, ограничивающие множество символов, вводимых в поле таблицы;
- устанавливать связи между двумя таблицами;
- вводить данные в таблицу;
- редактировать записи таблиц;
- изменять внешний вид таблиц;
- использовать выражения для задания условий проверки данных.

8.1. Создание таблиц

Как уже было отмечено, сущностями базы данных Access являются таблицы. С их помощью создаются другие объекты базы данных: запросы, формы, отчеты и др. Другими словами, реляционная база данных не может существовать без таблиц.

8.1.1. Создание структуры таблицы

Под **структурой таблицы** подразумевается информация, описывающая поля таблицы: их число, тип и свойства каждого поля, поля первичного ключа и др.

Для описания структуры таблицы:

1. Выделяется пиктограмма *Tables* на панели объектов *Objects* окна базы данных (обычно при запуске приложения эта пиктограмма выделяется автоматически). В рабочей области окна появляются опции:

– *Create table in Design view* (создание таблицы в режиме конструктора);
– *Create table by using wizard* (создание таблицы с помощью программы-помощника);
– *Create table by entering data* (создание таблицы путем ввода данных).

Выполним двойной щелчок на первой опции.

Замечание: Этот шаг эквивалентен выбору *New* → *Design view*.

2. Появляется диалоговое окно *Table* с именем по умолчанию *Table1* (*рис. 8.1*). Оно состоит из двух областей:

a) области описания структуры таблицы;
b) области описания свойств поля, выбранного в первой зоне (*Field Properties*).

Область описания структуры таблицы разбита на три столбца:

– *Field Name* (идентификатор поля);
– *Data Type* (тип поля, то есть тип его значений);
– *Description* (описание поля).

Следовательно, для каждого создаваемого поля необходимо уточнить его имя, тип данных и описать его назначение.

Идентификатор поля может содержать любые символы, за исключением символов „.” , „!”, „[”, „]”, пробела в начале и невидимых символов (например, возврат каретки).

Длина идентификатора поля не должна превышать 64 символа.

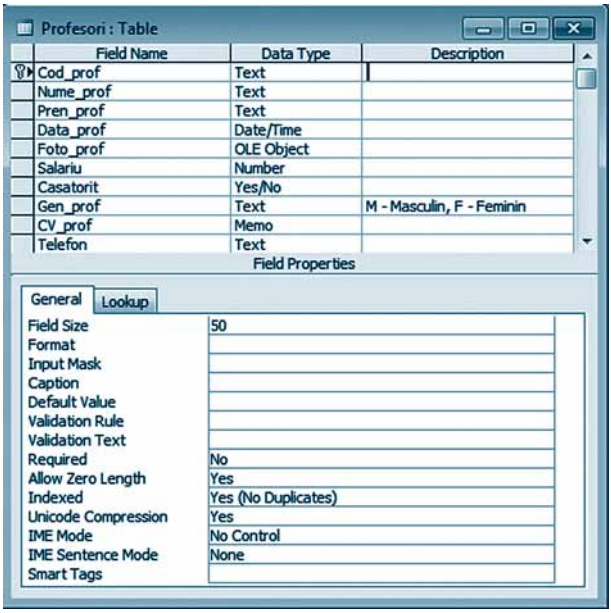


Рис. 8.1

Типы полей, допустимые в Access, представлены в следующей таблице:

Название типа	Описание значений типа
Text	Строка буквенно-цифровых символов. До 255 символов
Memo	Большие тексты. До 64 КБайт
Number	Числа
Date/time	Календарные даты
Currency	Денежные значения
AutoNumber	Натуральные числа 1, 2, 3, ..., автоматически генерируемые в заданном порядке при добавлении новой записи
Yes/No	Значения *Yes* или *No*
OLE Object	Изображения или звуки
Hiperlink	*Web*-адреса

Описание поля (заполнять этот столбец не обязательно) может содержать некоторые пояснения относительно поля.

Свойства полей будут описаны позже.

3. После определения полей **устанавливаем первичный ключ** для таблицы. Для этого выбираем нужное поле (значения которого не повторяются), затем выбираем опцию *Primary Key* из контекстного меню поля (либо нажимаем на соответствующую командную кнопку на панели инструментов Access). В таблице, на *рисунке 8.1*, поле *Cod_prof* было выбрано в качестве первичного ключа. Если поле первичного ключа для таблицы не было выбрано, то система Access предложит сделать это при сохранении таблицы. В качестве поля первичного ключа автоматически будет выбрано первое поле типа *AutoNumber*. Если такого поля в таблице нет, оно будет создано (с именем *ID* по умолчанию).

4. Сохраняем таблицу, выбрав команду *Save* из меню *File*. Появляется окно *Save As*, в котором записываем название таблицы.

Упражнение: Проанализируйте идентификаторы и типы полей таблицы на *рисунке 8.1*. Подобным образом создайте таблицу *Elevi* базы данных *Liceu*, описанной в предыдущей главе.

8.1.2. Свойства полей таблицы

Свойства полей – это характеристики, устанавливающие дополнительный контроль над тем, в каком виде данные поля хранятся, вводятся или выводятся. Свойства зависят от типа поля и уточняются в нижней части окна таблицы (*рис. 8.1*).

1. Свойство **Field Size** определяет формат и размер данных поля и существует только для типов *Text* и *Number*.

– Для типа *Text* допустимыми являются значения от 0 до 255, таким образом, устанавливается максимальная длина строки символов, которую можно сохранить в поле. По умолчанию эта длина равна 50.
– Для типа *Number* можно выбрать одно из значений *Byte*, *Integer*, *Long integer* (значение по умолчанию), *Single*, *Double*, *Replication ID*, *Decimal*.

2. Свойство **Format** с помощью шаблона уточняет способ вывода данных поля. Доступно для всех типов полей, кроме OLE Object.

– В случае типов *Text* и *Memo* шаблон можно создать с помощью следующих символов:

Символ	Описание
@	Текстовый символ или пробел.
&	Текстовый символ не обязателен.
<	Все символы в нижнем регистре.
>	Все символы в верхнем регистре.
–	Выводит символ –.

Пример: Шаблон @@-@@-@@> выведет вместо текста *abcdef* текст *AB-CD-EF*.

– Для типов *Number* и *Currency* можно выбрать одно из значений *General number*, *Currency*, *Euro*, *Fixed*, *Standard*, *Percent* или *Scientific*.
– Формат типа *Date/Time* может быть *General Date*, *Long Date*, *Medium Date*, *Short Date*, *Long Time*, *Medium Time*, *Short Time*. Уточним, что если год записан только двумя цифрами, то значения из интервала [00, 29] соответствуют годам 2000–2029, а значения из интервала [30, 99] – годам 1930–1999.

Пример: При формате *Long Date* значение 28.11.89 появится в виде 28 ноября 1989.

– Тип *Yes/No* допускает один из форматов *Yes/No*, *On/Off*, *True/False*. В последнем случае в поле допускается ввод, соответственно, значений 1 и 0.

3. Свойство **Input Mask** используется для создания шаблона ограничения символов (называется иногда *маской ввода*), которые можно вводить в поле. Для создания шаблонов допустимы следующие символы:

Символ	Описание
0	Цифра (не допускается перед ней знак + или –). Обязательный ввод.
9	Цифра (допускается перед ней знак + или –) либо пробел. Необязательный ввод.
#	Цифра (допускается перед ней знак + или –) либо пробел (при сохранении удаляется). Необязательный ввод.
L	Буква. Обязательный ввод.
?	Буква. Необязательный ввод.
A	Буква или цифра. Обязательный ввод.
a	Буква или цифра. Необязательный ввод.
&	Любой символ или пробел. Обязательный ввод.
C	Любой символ или пробел. Необязательный ввод.
. , : ; - /	Разделители для календарных дат или для классов числа (единицы, тысячи, миллионы, миллиарды и т.д.). Разделители по умолчанию определятся установками в окне *Regional Settings* (можно открыть из панели Control Panel операционной системы *Windows*).
<	Преобразует символы-буквы справа в строчные.
>	Преобразует символы-буквы справа в прописные.
!	Делает обязательным ввод данных справа налево.
\	Выводит лишь символ, следующий после \ (например, шаблон \M выводит букву M).
"Строка символов"	Выводит лишь строку символов (без кавычек).
Password	Вместо введенных символов будут выводиться символы *.

Примеры:

1. Шаблон >L<L0\S допускает строки из 4-х символов, первый из которых – прописная буква, второй – строчная буква, третий – цифра, а последний – буква S.
2. Для записи телефонных номеров только в виде (+373 22) XX-XX-XX можно использовать шаблон "(+373 22)" 00\-00\-00.

Для создания шаблонов можно использовать программу *Input Mask Wizard* (запускается щелчком по кнопке  из кассеты свойства *Input Mask*), предоставляющую 10 полезных форматов масок ввода.

4. Свойство **Caption** задает текст, который будет появляться в заголовке при использовании данного поля в запросах, формах, отчетах. Если это свойство остается незаполненным, то Access использует в качестве заголовка идентификатор поля.

5. Свойство **Default Value** устанавливает значение поля по умолчанию. Это значение появляется автоматически при добавлении новой записи.

6. С помощью свойства **Validation Rule** можно сформулировать условия на значение данных при попытке ввести их в поле. Условия на значение – это выражения, записанные с помощью операндов и функций Access или языка VBA (*Visual Basic for Applications*). Например, условие >= 100 разрешает пользователю ввести в поле числового типа только числа бóльшие либо равные 100.

7. В свойстве **Validation Text** записывается текст, который появится в предупреждающем окне, если введенное в поле значение не удовлетворяет условию проверки, указанному в *Validation Rule*.

8. Свойство **Required** может принимать лишь значения *Yes* и *No*. Значение *Yes* обязывает пользователя заполнить данное поле. Нет смысла заполнять это свойство для поля, выполняющего роль первичного ключа (так как первичный ключ не может быть пустым), либо если задано условие проверки *Is Not Null* (не является пустым).

9. В свойстве **Allow Zero Length** тоже можно использовать лишь значения *Yes* или *No*. Это свойство есть только у полей типа *Text* и *Memo*. При значении *Yes* в поле может быть значение длины 0, то есть пустая строка, даже если свойство *Required* имеет значение *Yes*.

10. Свойство **Indexed** позволяет (для индекса *Yes* (*Duplicates Ok*)) или запрещает (для индекса *Yes* (*No Duplicates*)) повторение значений в поле. Существующий индекс может быть удален, если из раскрывающегося списка свойства выбрать значение *No*. Для первичного ключа индекс *Yes* (*No Duplicates*) появится автоматически (*рис. 8.1*) и будет недоступен для изменения.

11. Свойство **New Values** применяется только для полей типа *AutoNumber*. Для значения *Increment* Access будет генерировать новые значения поля путем прибавления 1 к большему существующему значению. Если же зададим свойству *New Values* значение *Random*, тогда поле будет заполняться случайными значениями.

Вопросы и упражнения

❶ Проанализируйте таблицу *Profesori* базы данных *Liceu* и опишите ее структуру.

❷ Какие типы данных можно хранить в базе данных, созданной в Access?

❸ Каким должен быть тип поля, чтобы в нем можно было хранить фотографии? *Web*-адреса? Чью-либо биографию?

❹ Что необходимо сделать, чтобы добавление новой записи стало невозможным, если пользователь не заполнил последние два ее поля?

❺ Некоторое поле, не являющееся первичным ключом, не позволяет вводить повторяющиеся значения. По какой причине?

❻ Каким образом можно выяснить, есть ли в базе данных *Liceu* информация об учениках с одинаковыми датами рождения?

❼ Каким годам принадлежат даты:
a) 01.01.01; b) 30.12.30; c) 17.12.89; d) 15.04.28?

❽ Создайте таблицу Access, содержащую информацию о музыкальной коллекции. Включите в нее поля для хранения имени исполнителя, студии звукозаписи, года появления, формата (диск, кассета, CD и др.) и рейтинга (например, от 5 до 10).

❾ Создайте таблицу Access, содержащую информацию о коллекции кулинарных рецептов. Включите в нее поля для хранения названий блюд, их типа (первые блюда, гарниры, жаркое, пирожные, десерты и др.), времени приготовления, места происхождения блюд (молдавские, французские, японские и т.д.).

❿ Создайте шаблон, обязывающий пользователя вводить в некоторое поле только целые числа:
a) из интервала [10..99]; b) трехзначные; c) отрицательные, из двух цифр.

⓫ Создайте шаблон, позволяющий вводить в некоторое поле номера удостоверения личности граждан Молдовы. Эти номера начинаются с заглавной буквы, за которой следуют 8 цифр.

8.2. Установление связей между таблицами

Из предыдущей главы вы узнали, что между двумя таблицами может существовать одна из связей: 1 : 1, 1 : M, M : M.

Система Access использует для вывода и для создания связей специальное окно *Relationships*.

Рассмотрим на примере создание связи между двумя таблицами.

Установим связь 1 : M между таблицами *Clase* и *Elevi* базы данных *Liceu*.

1. Щелкните по кнопке на панели инструментов или выберите из меню *Tools* → *Relationships*. Появится окно *Relationships*.

2. Щелкните по кнопке на панели инструментов или выберите из меню *Relationships* → *Show Table*. Появится окно *Show Table* (*рис. 8.2*), из которого поочередно выберите таблицы *Elevi* и *Clase*, подтверждая всякий раз выбор нажатием кнопки *Add*. В окне *Relationships* появляются идентификаторы полей выбранных таблиц (*рис. 8.3*).

3. Первичный ключ каждой таблицы выделен жирным шрифтом. Выделите поле *Cod_clasa* таблицы *Elevi*, затем, держа нажатой левую кнопку мыши, потяните его к полю *Cod_clasa* таблицы *Clase*. Появится окно *Edit Relationships*, в котором автоматически определился тип связи 1 : M (*Relationship Type*: *One-To-Many*). Можно также активизировать следующие характеристики связи (*рис. 8.4*):

a) *Обеспечение целостности данных* (*Enforce Referential Integrity*);

b) *Каскадное обновление связанных полей* (*Cascade Update Related Fields*);

c) *Каскадное удаление связанных полей* (*Cascade Delete Related Records*).

Если активизирован флажок ***Обеспечение целостности данных***, то внешнее поле „подчиненной" таблицы сможет содержать лишь значения из первичного поля главной таблицы. Так, в поле *Cod_clasa* таблицы *Elevi* можно будет вписывать только „коды" классов, введенные в таблицу *Clase*.

Каскадное обновление связанных полей означает, что при изменении некоторого значения *V* в первичном ключе главной таблицы автоматически изменятся все

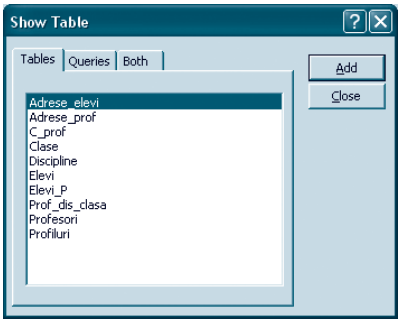


Рис. 8.2

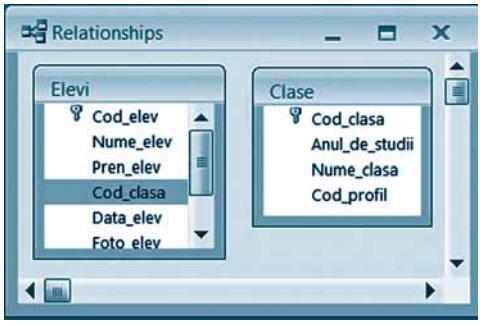


Рис. 8.3

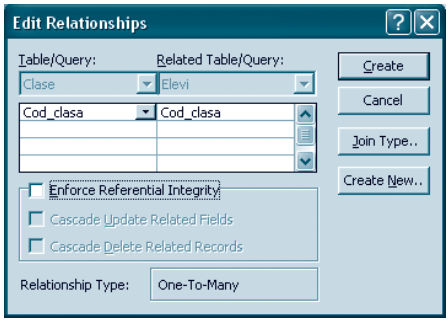


Рис. 8.4

значения *V* во внешнем ключе „подчиненной” таблицы. Например, если в таблице *Clase* заменим значение c01 – „код” 10-го A класса – на c001, то каждое значение c01 поля *Cod_clasa* таблицы *Elevi* заменится на c001.

Если выбрана опция ***Каскадное удаление связанных полей***, то при удалении некоторой записи *X* из главной таблицы в „подчиненной” таблице будут автоматически удалены все записи, содержащие в поле внешнего ключа значения поля первичного ключа записи *X*. Например, если удалим из таблицы *Clase* последнюю запись (с „кодом” c12, соответствующим 12-му D классу), то из таблицы *Elevi* будут „удалены” все ученики 12-го D класса.

Вопросы и упражнения

❶ Опишите алгоритм установления в Access связи между двумя таблицами.
❷ Что означает *целостность данных* в связанных таблицах?
❸ Школьные предметы принадлежат следующим куррикулумным областям:
a) язык и общение;
b) социально-гуманитарное образование (история, география, гражданское воспитание);
c) математика и естествознание (математика, физика, химия, биология, информатика);
d) технология;
e) спорт (физическое воспитание).
Откройте базу данных *Liceu*. Создайте и заполните таблицу для сохранения куррикулумных областей, затем установите связь между этой таблицей и таблицей *Discipline*.
❹ Какова роль характеристики связи *Cascade Update Related Fields*?
❺ Для чего используется характеристика связи *Cascade Delete Related Records*?

8.3. Изменение таблиц

8.3.1. Ввод и редактирование данных

Для того чтобы ввести или отредактировать данные в таблице, ее необходимо открыть в режиме *Datasheet View*, щелкнув на кнопке Open в окне базы данных.

Смена режимов *Datasheet View* (режим редактирования, *рис. 8.5*) и *Design View* (режим проектирования) может осуществляться с помощью кнопки *View* на панели инструментов Access.

Elevi : Table

	Cod_elev	Nume_elev	Pren_elev	Cod_clasa	Data_elev	Foto_elev	Telefon	Gen_elev
+	e218	Pogujanschii	Alexandru	c08	11/3/1996		022-59-68-19	M
+	e219	Pripa	Laura	c08	12/10/1996		022-63-73-42	F
*								

Record: 1 of 219

Рис. 8.5

При редактировании записи, слева от нее появляется **указатель записи (УЗ)**, вид которого зависит от ее состояния. Возможные состояния записи приведены далее в *таблице*.

УЗ	Состояние записи
	Выбрана текущая запись.
	Новая запись, в которую могут быть введены данные.
	Пользователь редактирует запись, однако изменения еще не сохранены.
	Запись блокирована другим пользователем и не может быть изменена (в многопользовательской среде – среде, в которой возможна одновременная работа нескольких пользователей в базе данных).

Для быстрого управления записями можно использовать инструменты из нижней части окна таблицы: 

, имеющие, соответственно, следующие функции (слева направо):

– выбор первой записи;
– выбор записи, предшествующей текущей;
– вывод порядкового номера текущей записи или выбор записи с порядковым номером, указанным в кассете;
– выбор записи, следующей за текущей;
– выбор последней записи;
– добавление новой записи.

Ввод и редактирование записей осуществляется методами, свойственными работе с текстом. Например, данные можно копировать в *Clipboard* и удалять клавишами *Backspace* и *Delete*.

Одно нажатие клавиши *Esc* аннулирует все изменения в текущем поле, а двойное нажатие – в текущей записи.

Не обязательно заполнять информацией все поля записи, за исключением поля первичного ключа (который не может быть пустым) и тех, для которых установлено значение *Yes* свойства *Required*.

Поле можно выбрать, щелкнув на его *заголовок* (ячейка, в которой выведено имя поля), а запись – щелкнув на ее *заголовок* (ячейка, в которой выводится указатель записи). Если, после выбора поля (записи), удерживать кнопку мыши нажатой и потянуть ее указатель к следующему полю (записи), то окажутся выбранными сразу два поля (записи).

Отметим, что **над данными или записями можно осуществлять следующие операции**:

– *добавление новой записи* перед текущей (команда *New Record* из контекстного меню заголовка записи или из меню *Insert* основной панели меню приложения Access);
– *удаление текущей записи* (команда *Delete Record* из контекстного меню заголовка записи или из меню *Edit* основной панели меню приложения Access);
– *заполнение полей*;
– *копирование содержимого некоторой ячейки* (команды *Copy* и *Paste* из контекстного меню ячейки);
– *копирование записи* (команды *Copy* и *Paste* из контекстного меню заголовка записи или из меню *Edit* основной панели меню приложения Access).

***Замечания*:**

1. При выполнении последней операции пользователь обязан изменить в новой записи содержание поля первичного ключа.

2. Некоторые из указанных выше операций могут быть выполнены и с помощью кнопок на панели инструментов Access.
3. Для быстрого выполнения операций можно использовать и комбинации клавиш.
4. Все изменения записи автоматически сохраняются в момент перехода к другой записи или при закрытии таблицы.
5. При вводе данных первыми заполняются главные таблицы, а затем – „подчиненные".

8.3.2. Изменение внешнего вида таблицы

Система Access выводит данные таблицы, соблюдая некоторые установки по умолчанию. Например, записи таблицы выводятся в возрастающем порядке значений поля первичного ключа, а поля следуют в том порядке, в котором были описаны при создании таблицы.

Пользователь может изменить способ представления информации из таблицы. Точнее, допустимы следующие **действия по изменению внешнего вида таблицы**:

a) *изменение порядка вывода полей*;
b) *изменение порядка вывода записей*;
c) *изменение высоты записи и ширины поля*;
d) *скрытие столбцов;*
e) *выборка записей.*

Изменение порядка вывода полей осуществляется путем их перемещения. Чтобы переместить одно или несколько последовательных полей, выделим эти поля, затем, удерживая нажатой левую кнопку мыши, помещаем ее указатель на поле, перед которым хотим разместить выделенные поля.

Чтобы записи появились в порядке возрастания (убывания) значений некоторого поля, выделим это поле, затем щелкнем на кнопке  на панели инструментов (соответственно, на кнопке ) или выберем *Records* → *Sort* → *Sort Ascending* (либо *Records* → *Sort* → *Sort Descending*). Похожим образом осуществляется упорядочивание записей по нескольким полям. Отметим, что в этом случае упорядочивание будет осуществляться в порядке слева направо, то есть значения полей справа будут учитываться лишь тогда, когда значения поля слева совпадают.

Действия по изменению высоты записей и ширины полей похожи на изменения высоты строк и ширины столбцов в редакторе текста или в табличном процессоре.

Чтобы спрятать столбец, выделяем его, затем выбираем команду *Hide Columns* из его меню или из меню *Format*. Для того чтобы показать спрятанные поля, выбираем команду *Unhide Columns* из меню *Format*. Появляется диалоговое окно *Unhide Columns*, в котором активизируем нужные столбцы.

Выборка записей, то есть отбор тех записей, которые удовлетворяют некоторым условиям, осуществляется путем создания **фильтра** (выберем *Records* → *Filter* → *Advansed Filter/Sort*) и его применения (выполним *Filter* → *Apply Filter/Sort*).

Удалить фильтр можно выполнив *Records* → *Remove Filter/Sort*.

***Замечания*:**

1. Изменения внешнего вида таблицы не сохраняются автоматически при закрытии таблицы. Пользователь может сделать это, нажав на кнопку *Save* на панели инструментов или с помощью команды *Save* из меню *Edit*.
2. Изменения внешнего вида таблицы не изменяют ее структуру.

8.3.3. Изменение структуры таблицы

Иногда, в процессе проектирования и даже в процессе эксплуатации базы данных возникает необходимость изменения структуры некоторых таблиц.

Внимание! Эти изменения могут нарушить целостность информации в базе данных. Например, уменьшение размерности текстового поля может привести к частичной потере его значений, а удаление поля первичного ключа приведет к удалению записей из „подчиненной" таблицы. Вообще-то, в зависимости от характеристик связей, система Access может запретить определенные изменения структуры таблицы.

В таких случаях изменение поля первичного ключа или внешнего ключа станет возможным только после удаления связи между таблицами, которую необходимо затем восстановить. Для изменения структуры таблицы ее необходимо открыть в режиме конструктора *Design View*, щелкнув на кнопке 

окна базы данных.

В данном режиме допустимы следующие операции:

a) *добавление нового поля*;
b) *удаление существующего поля*;
c) *изменение идентификатора поля*;
d) *изменение свойств полей*;
e) *установка первичного ключа*;
f) *удаление первичного ключа*.

Операции по изменению структуры таблицы подобны операциям по ее созданию.

8.3.4. Характеристика *Lookup* полей таблицы

Характеристика *Lookup* позволяет присоединить к полю таблицы раскрывающийся список. В этом случае пользователь сможет заполнять данное поле, выбирая нужное значение из списка допустимых. Содержание списка можно загрузить из поля первичного ключа таблицы, связанной с текущей, либо можно создать во время установки характеристики *Lookup*.

Воспользуемся специальной программой – мастером подстановок – для составления раскрывающегося списка для поля *Cod_clasa* таблицы *Elevi*.

1. Откройте таблицу *Elevi* в режиме конструктора. Выберите опцию *Lookup Wizard* из раскрывающегося списка возможных типов поля *Cod_clasa*.

2. Появится первое диалоговое окно *Lookup Wizard*. Выберите первую опцию, определив тем самым, что значения создаваемого списка будут взяты из связанной таблицы. Выбор второй опции означает, что элементы списка будут созданы на следующих этапах.

3. В последующих двух окнах сначала выберите таблицу *Clase*, затем поля *Cod_clasa*, *An_de_studii* и *Nume_clasa*. Значения выбранных полей появятся в созданном списке.

4. В четвертом окне настраивается ширина столбцов списка и задается его имя.

Вопросы и упражнения

❶ Какие операции можно осуществлять с записями таблицы?
❷ Какие изменения могут повлечь за собой изменение структуры таблицы?
❸ Откройте базу данных *Liceu*. Добавьте новое поле *Adresa_Web* в таблицу *Adrese_elevi*. Заполните это поле данными для первых 10 записей.

❹ Откройте таблицу *Elevi* в режиме редактирования:
   a) добавьте в таблицу 5 записей;
   b) удалите предпоследнюю запись;
   c) скопируйте данные первой записи в поля последней записи.

❺ Откройте таблицу *Profesori* в режиме редактирования:
   a) выведите данные таблицы в следующем порядке: фамилия, имя, пол, телефон, дата рождения учителей;
   b) спрячьте поля *Salariu*, *CV_prof* и *Foto_prof*;
   c) верните для просмотра спрятанные поля.

❻ Выведите в алфавитном порядке список учеников базы данных *Liceu*.

❼ Выведите фамилии и имена учеников в убывающем порядке их возраста.

❽ Выведите список учителей базы данных *Liceu* в порядке убывания их заработной платы.

8.4. Создание выражений Access

Анализируя свойство полей *Validation Rule*, было отмечено, что условия на значение данных, вводимых в поле, записываются с помощью выражений Access. Далее вы увидите, что выражения используются и при формировании запросов на выборку данных, и при создании отчетов.

Выражение Access — это сочетание встроенных или пользовательских функций, идентификаторов, операторов и констант.

При вычислении выражения возвращается одно значение. **Идентификатор** Access – это имя некоторого объекта базы данных. Объекты базы данных – это таблицы, поля, запросы, формы, отчеты. Как правило, идентификаторы записываются в квадратных скобках [и].

Идентификатор некоторого „подобъекта" формируется из имени класса объектов и имени „подобъекта", разделенных точкой или восклицательным знаком. Таким образом, каждому объекту базы данных соответствует единственный идентификатор.

Иногда, все же допускается использование в качестве идентификатора имени объекта, если это не создает неоднозначных прочтений.

Примеры: [Elevi].[Nume_elev], [Profesori].[Salariu], [Cod_elev].

8.4.1. Операторы Access

Рассмотрим 6 категорий операторов Access.

• **Арифметические операторы** (+, -, *, /, \, Mod, ^) применяются над числовыми значениями.

Пример: 15 \ 6 возвращает 2, а 15 *Mod* 6 возвращает 3, так как $15 = 6 \cdot 2 + 3$.

• **Операторы сравнения** сравнивают значения двух операндов и возвращают одно из логических значений *True* или *False*. В Access используются такие же операторы сравнения, как и в языке программирования Паскаль: <, <=, =, >=, >, <>.

• **Логические операторы** Access применяются над логическими операндами и тоже совпадают с таковыми из Паскаля: **And**, **Or**, **Not**, **Xor**.

• **Оператор присваивания** = присваивает значение некоторому объекту, переменной или константе.

• **Оператор слияния** + соединяет две символьные строки в одну.

• **Другие операторы.** Приведенные в следующей таблице операторы не входят ни в одну из перечисленных ранее категорий. Выражения, их содержащие, возвращают одно из значений *True* (или –1), *False* (или 0).

Оператор	Описание
Is	Применяется над значением Null (пустое значение) и проверяет, является ли значение пустым или нет.
Like	Определяет, записана ли строка символов в соответствии с заданным в Like шаблоном. Шаблон записывается в кавычках " и ". При его составлении допускаются заменяющие символы (**?** для одного символа, **#** для цифры, * для любого числа символов, в том числе и их отсутствия) и списки значений. Список значений задается в квадратных скобках [и]. Если перед значением записан символ **!**, то принадлежащими списку считаются все значения, кроме тех, перед которыми записан **!**.
In	Определяет принадлежность некоторого значения заданному списку. Значения списка разделяются символом **;**.
Between	Определяет принадлежность некоторого числового значения заданному интервалу.

Примеры:

1. Выражение [*Elevi*].[*Telefon*] *Is Null* возвращает значение *True* только в случае, если в поле *Telefon* таблицы *Elevi,* в текущей записи, ничего не записано.
2. *Like* "*B*ov*" задает все строки символов, начинающиеся с буквы *B* и заканчивающиеся комбинацией букв *ov*. Следовательно, выражение "*Belousov*" *Like* "*B*ov*" возвращает значение *True*.
3. *Like* "[*CK*]#?" задает строки из 3-х символов: первый символ – одна из букв *C* или *K*, второй – некоторая цифра, третий – любой символ.
4. *Like* "*[5*ad–g*]" задает строки, заканчивающиеся цифрой 5 или одной из букв *a*, *d*, *e*, *f*, *g*.
5. *Like* "*[!ae]" задает строки, которые не заканчиваются символами *a* или *e*.
6. Выражение "*Duminica*" *In* ("*Luni*"; "*Marti*"; "*Miercuri*"; "*Joi*"; "*Vineri*") возвращает значение *False*, а выражение 4 *In* (2; 4; 7; 8) – значение *True*.
7. Выражение *Between* 2 *And* 10 эквивалентно выражению >=2 *And* <=10.

Замечание: При записи условий на значение в кассете свойства *Validation Rule* в выражении не записывается первый операнд, так как им считается идентификатор соответствующего поля. Так, правило проверки *In* ("Luni"; "Marti"; "Miercuri"; "Joi"; "Vineri") позволит пользователю вписать в соответствующее поле лишь одно из слов: *Luni*, *Marti*, *Miercuri*, *Joi*, *Vineri*.

8.4.2. Функции Access

Функция возвращает через свое имя некоторое значение. Access предоставляет более 100 встроенных функций для обработки различных типов данных: числовых, строк символов, календарных дат и др. Наиболее часто используемые функции приведены в следующих таблицах:

Функции для обработки календарных дат

Функция	Возвращаемый результат
Date()	Текущая дата
DateAdd(T; N; D)	Календарная дата, получаемая добавлением к дате D или вычитанием из нее N (когда N отрицательно) календарных единиц типа T, где T может быть “yyyy”, “q”, “m”, “ww”, “d”, “h”, что, соответственно, означает: лет, кварталов, месяцев, недель, дней, часов
DateDiff(T; D_1; D_2)	Разность между датами D_1 и D_2, выраженная в единицах типа T
Time()	Текущее время
Now()	Текущие дата и время
Year(D)	Год, записанный 4 цифрами, соответствующий дате D
Month(D)	Порядковый номер месяца в году, соответствующий дате D
Day(D)	Порядковый номер дня в месяце, соответствующий дате D
WeekDay(D)	Порядковый номер дня в неделе, соответствующий дате D (1 – воскресенье, 2 – понедельник и т.д.)
Hour(D)	Час (целое число от 0 до 23), соответствующий календарному значению D

Функции для обработки текста

Функция	Возвращаемый результат
Asc(C)	Код ANSI символа C
Chr(N)	Символ, код ANSI которого равен N
InStr(S_1; S_2)	Позиция, начиная с которой строка S_2 содержится в строке S_1
Mid(S; N_1; N_2)	Подстрока строки S длиной N_2, начиная с позиции N_1. Параметр N_2 может отсутствовать, что означает получение подстроки из S, путем удаления первых N_1–1 символов
LCase(S)	Строка символов, полученная из строки S путем преобразования прописных букв в строчные
UCase(S)	Строка символов, полученная из строки S путем преобразования строчных букв в прописные
Len(S)	Число символов в строке S
LTrim(S)	Строка символов, полученная из строки S путем удаления пробелов, имеющихся в начале
RTrim(S)	Строка символов, полученная из строки S путем удаления пробелов, имеющихся в конце
Trim(S)	Строка символов, полученная из строки S путем удаления пробелов, имеющихся в начале и в конце
Left(S; N)	Строка символов, полученная из первых N символов строки S
Right(S; N)	Строка символов, полученная из последних N символов строки S
Str(X)	Строка символов, полученная из символов числового значения X, записанных в том же порядке
Val(S)	Число, полученное из символов строки S (если строка имеет подходящий формат)

Математические функции

Функция	Возвращаемый результат
Abs(X)	Модуль числа X
Atn(X)	Арктангенс (в радианах) числа X

Avg(*C*)	Среднее арифметическое значений поля *C*
Count(*C*)	Количество ненулевых значений поля *C*
Max(*C*)	Максимальное значение поля *C*
Cos(*X*)	Косинус числа *X*
Exp(*X*)	Значение e^X
Int(*X*)	Целая часть числа *X*
Log(*X*)	Десятичный логарифм числа *X*
Rnd()	Случайное число из интервала от 0 до 1
Sgn(*X*)	0 если число *X* положительно, –1 если *X* отрицательно
Sin(*X*)	Синус числа *X*
Sqr(*X*)	Квадратный корень числа *X*
Tan(*X*)	Тангенс числа *X*

Замечания:

1. Параметрами функций могут быть идентификаторы полей (записанные в скобках [и]).
2. Если параметром фунции является календарная константа, то ее записывают заключенной между символами " и ".

Примеры:

1. *DateAdd*("*d*";–50; *Date*()) возвращает календарную дату, которая была 50 дней назад.
2. *Weekday*("27.09.1993") возвращает значение 2, так как 28 сентября 1993 был понедельник. (Понедельник считается вторым днем недели.)
3. *InStr*("Informatica"; "forma") возвращает 3.
4. *LCase*("INFORMATICA") возвращает текст "informatica".
5. *LTrim*("forma") возвращает строку "forma".
6. *Sgn*(– 20) возвращает значение –1.

Вопросы и упражнения

❶ Для чего в Access используются выражения?

❷ Откройте базу данных *Liceu* и запишите условие на значение для поля *Nr_ore_saptamina* таблицы *Prof_dis_clasa,* которое позволяло бы пользователю вводить в это поле только целые положительные числа.

❸ Какое значение возвратит выражение:

a) *Mid*("Propoziţie"; 3);
b) *Mid*("Calculator"; 1; 3);
c) *Int*(*Rnd*()*50);
d) *Month*("15.11.1990");
e) *Left*("Tractor"; 5);
f) 5 *In* ("4"; "5"; "6"; "7"; "8");
g) *Val*("25") – 25;
h) "R" *Like* "[TR]*";
i) "R" + *Str*(*Date*())?

❹ Напишите выражение, возвращающее:

a) среднюю заработную плату учителей из таблицы *Profesori*;
b) календарную дату, которая будет через 5 недель после текущей даты;
c) порядковый номер дня года для текущей даты;
d) порядковый номер дня недели для 1 января 2000;
e) количество дней между датами 5 марта 2005 и 5 декабря 2005.

❺ Напишете для поля *Gen_prof* таблицы *Profesori* условие на значение, которое позволит записывать в данное поле лишь значения *M* или *F*.

Контрольный тест

1. Определите истинность высказывания: „Значения из таблицы являются частью ее структуры".

a) истина; в) ложь.

2. Впишите пропущенные фрагменты так, чтобы полученное высказывание стало истинным:

a) Свойство ..... устанавливает значение по умолчанию для некоторого поля таблицы.

b) Если значение свойства ..... некоторого поля равно ....., то Access не допустит ввода новой записи до тех пор, пока данное поле текущей записи не будет заполнено.

3. Опишите ограничительные действия, задаваемые шаблоном >L0L 0L09.

4. Создайте шаблон, допускающий в текстовое поле строки, сформированные:

a) из 3 символов, первый из которых – буква, а два последующих – цифры;

b) из, минимум, 4-х и, максимум, 6-и символов, последний из которых является цифрой.

5. Какую из характеристик:

a) Обеспечение целостности данных (*Enforce Referential Integrity*);

b) Каскадное обновление связанных полей (*Cascade Update Related Fields*);

c) Каскадное удаление связанных полей (*Cascade Delete Related Records*)

необходимо активизировать для того, чтобы изменение некоторого значения в поле первичного ключа главной таблицы повлекло за собой соответствующие изменения в „подчиненной" таблице?

6. Пусть дана таблица *Lista* со структурой:

Nume	Prenume	Data naşterii	Raionul	Profesia	Genul
Popa	Ion	12.08.1979	Briceni	profesor	M
...	...	...	...	...	...

Опишите действия, которые необходимо осуществить с данной таблицей, чтобы выводились в указанном порядке только данные полей: *Raionul*, *Data naşterii*, *Prenume*, *Nume*.

7. Впишите вместо точек выражения, чтобы получить истинное высказывание:

a) Выражение …… задает строки символов, начинающиеся с буквы *T* и заканчивающиеся буквой *R*.

b) Выражение …… задает строки символов, начинающиеся с любой, кроме *C*, *D*, *E*, *T*, буквы.

c) Выражение …… задает целые числа, по модулю, большие 15 и меньшие 40.

8. Какое значение возвращает выражение:

a) *DateAdd*("*ww*";1; "01.09.2010");

b) *Year*("01.09.30");

c) "*PAR*" *Like* "[*A-D*] *"?

9. Напишите для поля *Adrese_email* некоторой таблицы условие проверки, позволяющее вводить в данное поле лишь строки, содержащие символ @.

ГЛАВА 9

ЗАПРОСЫ

Изучив данную главу, вы сможете:

- описывать типы запросов;
- создавать запросы (в режиме конструктора и с помощью программы-мастера) на выборку данных из одной или нескольких таблиц;
- выполнять запросы;
- изменять данные в таблицах с помощью запросов;
- получать поля с новыми данными, вычисленными с помощью имеющейся информации;
- создавать итоговые данные из одной или нескольких таблиц;
- компактно представлять подмножества данных из одной или нескольких таблиц.

9.1. Основные понятия о запросах

Системы управления базами данных созданы для хранения и автоматической обработки данных. Даже в случае баз данных с несколькими сотнями записей, поиск вручную информации, удовлетворяющей определенным условиям, является достаточно сложным процессом. На практике существуют базы данных, содержащие миллионы записей. Например, некоторые операторы мобильной телефонной связи Республики Молдова имеют более 1 000 000 абонентов! В парламентских выборах страны участвуют более 1,5 миллиона избирателей. Следовательно, при возможности электронного голосования необходимо обработать более 1,5 миллиона записей базы данных!

Поиск требуемых данных может привести к необходимости просмотра нескольких таблиц. Например, чтобы узнать, какие предметы изучает указанный ученик из базы данных *Liceu*, необходимо проанализировать таблицы *Elevi*, *Clase*, *Prof_dis_clasa* и *Discipline*. Убедимся в этом на практике!

- Проанализируйте базу данных и определите, какие предметы изучает ученик *Dan Moraru*.

Предложенное упражнение – лишь маленький аргумент в пользу необходимости автоматической обработки информации базы данных.

Для быстрого выбора из одной или нескольких таблиц наборов данных, удовлетворяющих определенным условиям, а также для ускорения процесса обновления данных, системы управления базами данных используют *запросы* – заявки на поиск и/или действие в соответствии с заданными условиями.

Запросы – это объекты системы управления базами данных, которые представляют собой заявки на поиск, обработку и/или на изменение данных базы.

Отметим, что при создании запросов, кроме записей из таблиц, в качестве источника информации могут использоваться и результаты других запросов, созданных ранее.

Например, для получения списка преподавателей с максимальной заработной платой, создадим два запроса. Один будет находить максимальное значение *Max* в поле *Salariu,* а второй – сделает выборку записей из таблицы *Profesori,* у которых в поле *Salariu* значение совпадает с величиной *Max* (то есть с результатом первого запроса).

Наиболее часто запросы используются для:
– просмотра некоторого подмножества записей таблицы, без того чтобы открывать эту таблицу;
– вывода в виде одной таблицы данных из нескольких таблиц;
– обновления данных таблиц (изменение, добавление, удаление данных);
– выполнения вычислений над значениями полей и получения новой информации;
– вычисления итогов, средних значений и др.

В зависимости от выполняемых действий и их результатов, *запросы* классифицируются так:
a) *запросы на выборку*;
b) *запросы на удаление определенных записей*;
c) *запросы на изменение определенных записей*;
d) *запросы с вычисляемыми полями*;
e) *запросы с группировкой данных и итогами*;
f) *перекрестные запросы*.

Запросы на выборку – это запросы, сформулированные на основании логических условий. Они выбирают подмножество данных из одной или нескольких связанных таблиц. Например, поиск учеников, родившихся до 10 января 1992, вывод списка учеников 10-го B класса – это запросы на выборку.

Запросы на удаление определенных записей представляют собой запросы на удаление из таблицы всех записей, удовлетворяющих некоторым логическим условиям. Например, запрос на удаление из таблицы *Elevi* базы данных *Liceu* информации об учениках 12-го класса (в связи с окончанием лицея) представляет собой запрос такого типа.

Запросы на изменение определенных записей изменяют значения некоторого поля таблицы по одному и тому же алгоритму. Например, увеличение на 50% значений поля *Salariu* таблицы *Profesori* может быть осуществлено с помощью запроса такого типа.

Иногда возникает необходимость вывести определенную информацию в новых полях. Например, отдельно подсчитать возраст учащихся. Для этой цели воспользуемся ***запросами с вычисляемыми полями***.

Запросы с группировкой данных и итогами применяются для суммирования данных поля, подсчета средних значений, нахождения наибольшего либо наименьшего значения и др. Например, подсчет общего числа часов за неделю, реализованных в каждом классе базы данных *Liceu,* можно осуществить запросом такого типа.

Перекрестные запросы предназначены для компактного представления информации в виде двумерной таблицы. Например, информация о количестве часов в неделю по каждому предмету, в каждом классе, может быть выведена с помощью перекрестного запроса в виде следующей таблицы:

Anul_de_studii	Nume_clasa	Biologia	Chimia	...
10	A	2	3	...
10	D	1	1	...
11	A	3	2	...
12	D	1	1	...
...	...	...	...	...

Так как результаты запросов зависят от информации, содержащейся в таблицах, изменения, производимые в таблицах, автоматически влекут за собой изменения в запросах (очевидно, после их повторного выполнения).

В свою очередь, в случае запросов a) – d), изменение пользователем данных в запросе может привести к изменениям в таблицах. Из этих соображений результаты запросов a) – d) называют **динамическим набором данных**.

Динамический набор данных не запоминается. Он существует только в момент выполнения запроса.

Для создания нового запроса:

1. Выделяем класс объектов *Queries* на панели объектов *Objects*. В рабочей области окна базы данных появляются опции:

– *Create query in Design view* (создание запроса в режиме конструктора);

– *Create query by using wizard* (создание запроса с помощью программы – мастера).

Выполним двойной щелчок на первой опции.

Замечание: *Шаг 1* эквивалентен выбору *New* → *Design View*.

2. Появляется окно запроса и диалоговое окно *Show Table* (*рис. 9.1*).

Выбираем, поочередно, необходимые таблицы, щелкая всякий раз на кнопке *Add*. Поля выбранных таблиц появляются в окне запроса вместе с графическим представлением связей между таблицами (*рис. 9.2*).

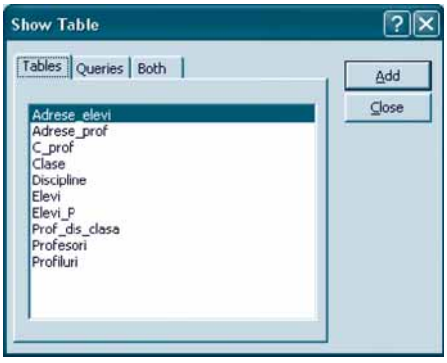


Рис. 9.1

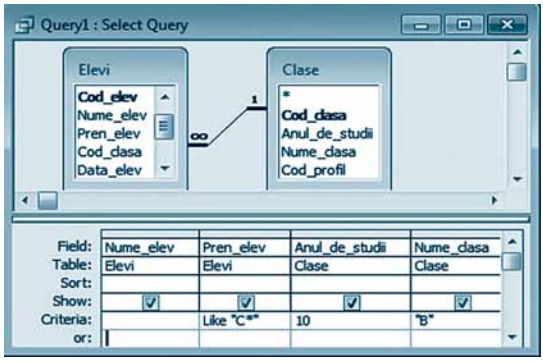


Рис. 9.2

3. Окно запроса разбито на две зоны. В верхней зоне представлены названия полей таблиц. В нижней зоне выводится **бланк запроса QBE**[*], в котором пользователь может использовать примеры частичных объявлений либо выражения для создания запроса. Выбираем поля, которые понадобятся для написания условий выборки, и те, значения которых будут выводиться. Добавить поле можно щелкнув дважды на его идентификаторе либо выделив и перетащив его с помощью мыши из верхней части окна в бланке запроса. Порядок полей, выведенных в бланке запроса, может быть изменен похожим образом, которым мы меняли порядок полей в таблице, открытой в режиме *Datasheet View*.

[*] Характеристика QBE (Query by Example) первоначально была создана для того, чтобы пользователи баз данных, без необходимости знать язык программирования, могли находить и выводить фрагменты данных. Со временем приложение QBE стало предпочитаемым инструментом для написания запросов. Большинство СУБД разработали собственные приложения QBE, позволяющие формулировать широкий спектр запросов.

4. Для каждого поля, добавленного в бланк запроса QBE, помимо (выведенных в строках *Field* и *Table*) его имени и названия таблицы, из которой оно взято, можно дополнительно уточнить:

a) способ сортировки записей по данному полю (строка *Sort*), выбирая одно из значений: *Ascending* (по возрастанию) или *Descending* (по убыванию);
b) вывод или скрытие значений поля при выполнении запроса (строка *Show*);
c) условие выборки, которому будут удовлетворять выводимые значения (строки *Criteria* и *or*).

5. Можем **вывести данные**, указанные в запросе, до того, как сохраним этот запрос, воспользовавшись командой *Datasheet View* из меню *View* или из раскрывающегося списка  на панели инструментов Access.

6. Сохраняем запрос (кнопка *Save* или команда с таким же именем из меню *File*).

Пример: На *рисунке 9.2* представлен в режиме конструктора запрос на выборку списка учеников 10-го B класса, имя которых начинается на букву C. На *рисунке 9.3* показан этот список еще до сохранения запроса.

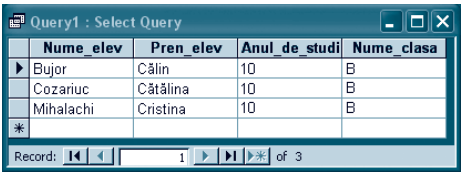


Рис. 9.3

Вопросы и упражнения

❶ С какой целью используются запросы?
❷ Охарактеризуйте основные типы запросов.
❸ Опишите алгоритм создания запроса.
❹ Почему результаты некоторых типов запросов называются *динамическими наборами данных*?
❺ Определите тип следующих запросов:
a) нахождение числа преподавателей мужского и женского пола;
b) вычисление средней заработной платы учителей базы данных *Liceu*;
c) вывод списка девочек из 11-х классов;
d) вычисление числа часов, реализуемых еженедельно каждым учителем;
e) поиск учителей женского пола, преподающих в 10-х классах.
❻ Проанализируйте базу данных *Liceu* и сформулируйте по два запроса:
a) на выборку;
b) на удаление некоторых записей;
c) на изменение некоторых записей;
d) с вычисляемыми полями;
e) с группировкой и итогами;
f) перекрестных.

9.2. Запросы на выборку

Так как запросы на выборку используются наиболее часто, Access устанавливает именно этот тип, по умолчанию, для создаваемого нового запроса. Позже вы увидите, что пользователь должен дополнительно осуществить некоторые действия, чтобы изменить тип запроса.

Итак, чтобы создать запрос на выборку, выберем таблицы и нужные поля, воспользовавшись описанным в предыдущей теме алгоритмом.

9.2.1. Условия выборки

Очень важным моментом в процессе создания запроса на выборку является написание **условия выборки**. Если условие формулируется для одного поля, то логическое выражение, контролирующее вывод данных, вписывается в ячейке *Criteria* для данного поля. Отметим, что оператор *Like* добавляется автоматически самим приложением Access, если в качестве условий выборки используются шаблоны, ограничивающие данные. В частности, в ячейку *Criteria* можно вписывать константы типа, совместимого с типом значений соответствующего поля.

Пример: Для вывода списка учеников по имени Ion, достаточно вписать в ячейку *Criteria* поля *Pren_elev* (из таблицы *Elevi*) строку символов "Ion".

Так как условия выборки являются логическими выражениями Access, для их записи можно использовать функции и операторы, изученные в предыдущей главе, в том числе, и *логические*.

Кроме того, бланк запроса QBE предоставляет помощь в написании составных условий, состоящих из нескольких условий и операторов AND и/или OR. Так:

a) для одного поля можно определить несколько условий выборки, связанных между собой оператором OR: первое условие записывается в строке *Criteria*, все остальные – ниже, по одному в каждой ячейке;

b) два или несколько условий из строки *Criteria* считаются связанными оператором AND.

Примеры:

1. Если в бланке запроса, из предыдущего примера, ниже значения "Ion" (в ячейке *or*) записать "Vasile", то запрос выдаст список учеников, имя которых Ion или Vasile. В случае добавления в ячейку *Criteria* поля *Nume_elev* условия "B*", запрос выдаст список учеников, имя которых Ion, фамилия начинается с буквы "B" и всех учеников по имени Vasile.
2. Запрос на *рисунке 9.4* выдаст список учеников 11-го B класса, родившихся в январе.
3. Запрос на *рисунке 9.5* выдаст список учеников 10–11 классов реального профиля. Такой же результат будет получен, если запишем в строке *Criteria* выражение 10 OR 11.
4. Запрос на *рисунке 9.6* выдаст список учителей с месячной заработной платой большей 2500 леев и меньшей либо равной 3000 леев, в порядке возрастания заработых плат.

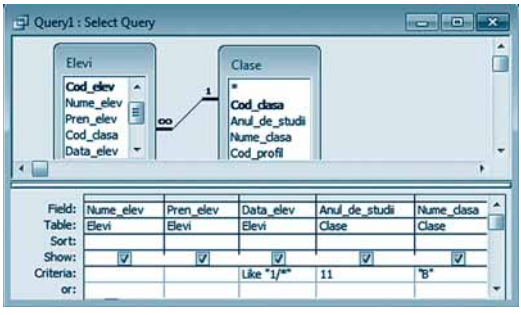


Рис. 9.4

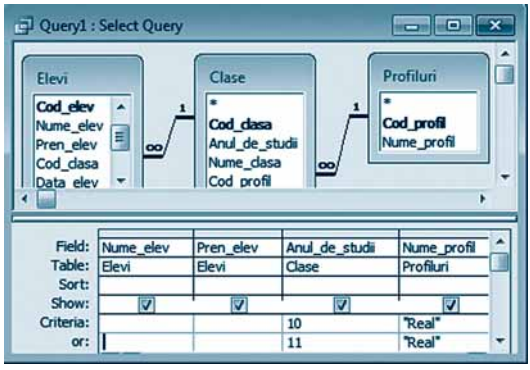


Рис. 9.5

9.2.2. Запросы с параметрами

Очевидно, невозможно описать все условия выборки, которые могут понадобиться пользователям. Более того, некоторые запросы могут отличаться друг от друга только значениями из бланка запроса QBE. Например, запросы, выводящие список предметов, изучаемых в 10-м B классе, соответственно, в 11-м B, отличаются только значениями в ячейке *Criteria* поля *Anul_de_studii*.

В таких случаях можно создать один запрос, в котором, вместо конкретного значения *Criteria* соответствующего поля, вписать *параметр*.

Строка символов, заключенная в квадратные скобки [и], записанная в некоторой ячейке строки *Criteria* воспринимается системой Access как **параметр**.

Для каждого параметра, во время выполнения запроса, сначала появляется диалоговое окно, в котором пользователь вписывает значение параметра (некоторое значение поля, для которого создан параметр), затем выводятся записи, у которых значения из поля с параметром совпадают со значением, указанным пользователем.

Как правило, строка символов, определяющая параметр, является некоторым текстом, подсказывающим пользователю, какое значение он должен ввести в диалоговом окне.

По умолчанию параметр считается типа *Text*. Чтобы изменить тип параметра, необходимо выбрать *Query* → *Parameters*. Появится окно *Query Parameters*, в котором вписываются все параметры и их типы.

Пример: На *рисунке 9.7* представлено окно некоторого запроса на выборку с тремя параметрами, из которых: первый – для поля *Anul_de_studii*, второй – для поля *Nume_clasa*, а третий (типа *Number*) – для поля *Nr_ore_saptamina*. С помощью данного запроса можно вывести список предметов, изучаемых учениками класса, указанного пользователем, с числом часов в неделю, тоже указанным пользователем.

При выполнении запроса, представленного на *рисунке 9.7*, поочередно появятся три диалоговых окна, в которых пользователь сначала введет год обучения (10, 11 или12), затем название класса (A, B, C или D) и, соответственно, число часов в неделю.

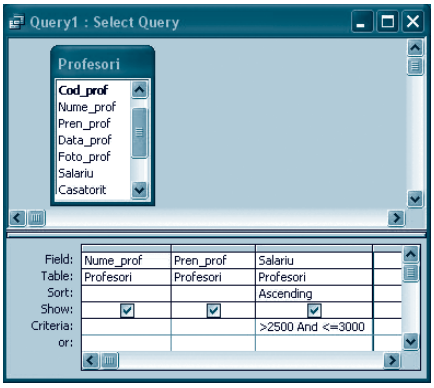


Рис. 9.6

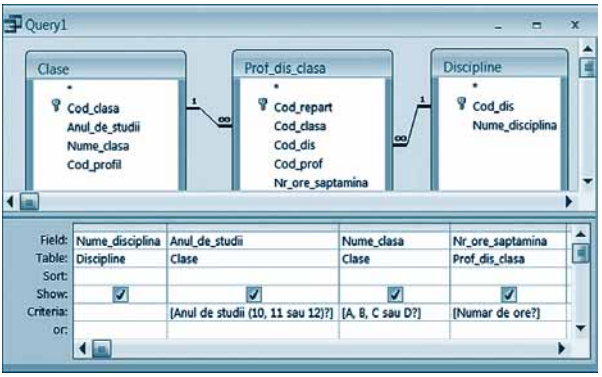


Рис. 9.7

Вопросы и упражнения

❶ Из чего состоят запросы на выборку?
❷ Какова роль условия выборки в запросе?
❸ С какой целью используют параметры в запросе?
❹ Откройте базу данных *Liceu*. Создайте запрос на выборку, который выведет список:
a) девочек;
b) классных руководителей;
c) учителей, родившихся до 12 февраля 1970;
d) учителей, родившихся в марте;
e) мальчиков, в порядке возрастания их возраста;
f) учеников, у которых не указан номер телефона;
g) учителей мужского пола, не достигших возраста 50 лет;
h) учеников, родившихся летом;
i) учителей, номер телефона которых начинается с 022 48.
j) учеников, не живущих в Кишинэу (Chişinău);
k) учителей, чье имя начинается с буквы A или буквы E;
l) учеников, не изучающих философию (Filosofia);
m) учеников, изучающих математику (Matematica) 5 часов в неделю.
❺ Откройте базу данных *Liceu*. Создайте запрос на выборку с параметром, который выведет список:
a) предметов, изучаемых учеником, указанным пользователем;
b) учителей, преподающих в классе, указанном пользователем;
c) учителей, живущих в местности (Loc_prof), указанной пользователем;
e) учителей, преподающих предмет, указанный пользователем.
❻ Сформулируйте и создайте 5 запросов на выборку для базы данных *Liceu*.
❼ Для базы данных *Liceu* сформулируйте и создайте 5 запросов на выборку с параметром.

9.3. Запросы на изменение

Запросы на изменение применяются для создания новых таблиц на основе информации из таблиц, уже существующих в базе данных, а также для осуществления изменений в таблицах базы. В окне базы данных таким запросам соответствует пиктограмма с восклицательным знаком.

Внимание! Запросы на изменение (за исключением создающих таблицы) изменяют содержание таблиц.

9.3.1. Запросы на создание таблиц

Запросы на выборку выводят информацию из таблиц в момент выполнения запроса. Результат такого запроса не сохраняется (в виде таблицы). Для этой цели имеется возможность преобразовать запрос в такой, который сможет сохранить результат в виде новой таблицы, то есть создаст из динамического набора новую таблицу.

Создадим запрос, генерирующий таблицу с данными об учениках 10-х классов.

1. Сначала создаем запрос на выборку необходимых данных (*рис. 9.8*).

2. Из меню *Query* выбираем *Make-Table Query*... Появится окно *Make Table*, где уточняется название новой таблицы (например, *Clasele_10*) и имя базы данных, в которой будет храниться новая таблица. По умолчанию эта таблица сохранится в текущей базе данных.

3. Записываем имя таблицы, нажимаем на кнопку OK, и затем сохраняем запрос.

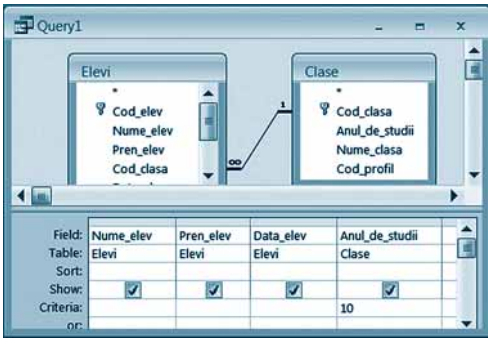


Рис. 9.8

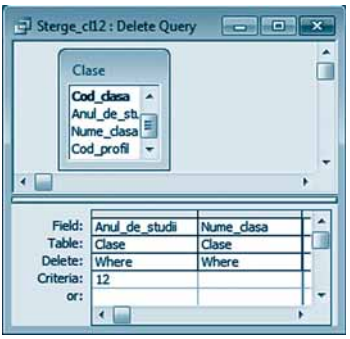


Рис. 9.9

4. Для получения новой таблицы *Clasele_10* выполним этот запрос. Появляется окно с предупреждением. Подтверждаем создание новой таблицы, нажав кнопку *Yes*.

9.3.2. Запросы на удаление записей

Часто возникает необходимость удаления некоторых записей из таблиц. Например, в случае базы данных *Liceu,* допустим, что необходимо в связи с окончанием лицея удалить данные об учениках 12-х классов.

1. Так как между таблицами *Clase* и *Elevi* установлена связь **один-ко-многим** с характеристикой *Каскадное удаление связных полей* (*Cascade Delete Related Records*), достаточно удалить те записи в таблице *Clase*, для которых значение поля *Anul_de_studii* равно 12. Наоборот будет неправильно, потому что таблица *Elevi* является „подчиненной" таблицы *Clase*.

2. Создаем запрос, который отображает список классов. Из меню *Query* выбираем *Delete Query*. В бланке запроса QBE вместо строк *Show* и *Sort* появляется строка *Delete*. В ячейку *Criteria* поля *Anul_de_studii* пишем 12 (*рис. 9.9*).

3. Сохраним запрос. Удаление записей осуществится после выполнения запроса.

Внимание! Так как удаленные записи не могут быть восстановлены, рекомендуется, до того как запрос на выборку будет преобразован в запрос на удаление, вывести его результаты и проверить их правильность.

9.3.3. Запросы на обновление записей

Если записи некоторого поля можно преобразовать по одному и тому же алгоритму, то для автоматизации этих преобразований используется запрос на обновление записей. Сформулируем запрос, который увеличит на 50% значения из поля *Salariu* таблицы *Profesori*.

1. Создаем запрос на выборку, выводящий значения поля *Salariu*.

2. Из меню *Query* выбираем *Update Query*. В бланке запроса QBE вместо строк *Sort* и *Show* появляется строка *Update To*. Запишем выражение [Salariu]*1,5 в ячейке на пересечении строки *Update To* и поля *Salariu* (*рис. 9.10*).

3. Сохраняем запрос. Изменение записей произойдет после выполнения запроса.

Рис. 9.10

Внимание! Запросы на обновление выполняются один раз. При повторном выполнении запроса записи изменяются вновь. Так, если выполним дважды последний запрос, то заработная плата вместо 50% „вырастет” на 125%.

9.3.4. Запросы на добавление новых записей в существующие таблицы

Иногда возникает необходимость добавлять новые записи в одну таблицу из другой. Например, предположим, что существует таблица *Elevi1*, содержащая данные об учениках, родившихся в мае. Мы желаем добавить в эту таблицу данные об учениках таблицы *Elevi,* родившихся в июне. Сделаем это так:

1. Создаем запрос на выборку, выводящий список учеников, родившихся в июне.

2. Из меню *Query* выбираем *Append Query*. Появляется окно *Make Table*. Из раскрывающегося списка *Table Name* выбираем *Elevi1*.

3. Сохраним, а затем выполним запрос.

Вопросы и упражнения

❶ С какой целью используют запросы на изменение?

❷ Опишите алгоритм создания запроса:
a) генерирующего таблицу;
b) на обновление некоторых записей;
c) на удаление некоторых записей;
d) на добавление новых записей в существующие таблицы.

❸ Проанализируйте базу данных *Liceu*. Разработайте запрос на создание таблицы:
a) T1 с данными о преподавателях женского пола;
b) T2 с данными об учениках, родившихся в мае;
c) T3 с данными о классных руководителях;
d) T4 с данными о преподавателях математики и химии;
e) T5 с данными об учениках, не живущих в Кишинэу (Chişinău).

❹ Проанализируйте базу данных *Liceu*. Создайте запрос на удаление из таблицы:
a) T1 данных об учителях, рожденных летом;
b) T2 данных об учениках 11-го класса;
c) T3 данных о классных руководителях классов реального профиля;
d) T4 данных об учителях, не преподающих химию;
e) T5 данных об учениках, живущих в Крикова (Cricova).

❺ Проанализируйте базу данных *Liceu*. Создайте запрос, который:
a) увеличит заработную плату учителей на 500 леев;
b) уменьшит заработную плату на 300 леев учителям, преподающим только один предмет;
c) увеличит заработную плату на 25% учителям, рожденным до 01.01.1960.

❻ Проанализируйте базу данных *Liceu*. Создайте запрос, который добавит в таблицу:
a) T1 данные об учителях мужского пола, преподающих математику;
b) T2 данные об учениках, рожденных летом;
c) T3 данные об учителях, не являющихся классными руководителями, живущих в Кишинэу (Chişinău);
d) T4 данные об учителях, преподающих иностранный язык;
e) T5 данные об учениках 10-го класса, живущих в Кишинэу (Chişinău).

9.4. Итоговые запросы

9.4.1. Запросы с вычисляемыми полями

В предыдущих главах было отмечено, что при проектировании баз данных в таблицы не включаются поля со значениями, которые можно вычислить с помощью уже существующих полей.

По этим соображениям, например, в таблицу *Elevi* не было включено поле *Virsta*. Значения такого поля зависят от содержания поля *Data_elev*. Создадим запрос, выводящий в новом поле возраст учеников базы данных *Liceu*. Этот запрос не затронет содержимое таблицы *Elevi*.

1. Создадим запрос на выборку на основании таблицы *Elevi*. В первые два столбца бланка запроса QBE включим поля *Nume_elev*, *Pren_elev*, а в третьем столбце, в месте для имени столбца, запишем выражение *Virsta*: *DateDiff*("yyyy";[Data_elev];Date()). Заметим, что *Virsta* – это имя нового поля, а функция *DateDiff*(T; D_1; D_2), возвращающая разность, выраженную в календарных единицах T, между датами D_1 и D_2 (см. тему *Функции Access*). Обратите внимание на то, что:

– T равен "yyyy", значит выражает года;

– D_1 равен [Data_elev], то есть дата рождения ученика;

– D_2 равен Date(), то есть текущая дата.

2. Сохраним запрос. Поле *Virsta* является динамическим набором. Оно существует пока выводятся результаты запроса.

9.4.2. Запросы с группировкой данных и итогами

Для получения результатов, основывающихся на записях одной или нескольких таблиц, воспользуемся *запросами с группировкой данных и итогами*.

Определим запрос, выводящий количество учеников каждого класса базы данных *Liceu*.

1. Создадим запрос на выборку на основе таблиц *Clase* и *Elevi*, включив поля *Anul_de_studii*, *Nume_clasa* и *Cod_elev*.

2. Щелкнем на кнопке *Totals* (Σ) на панели инструментов. В бланке запроса QBE появляется строка *Total* (*рис. 9.11*). Заполняем ячейки этой строки:

– в первых двух столбцах, в раскрывающихся списках, выберем значение *Group By* (так как группируем данные по году обучения и по названию класса);

– в третьем столбце выберем функцию *Count* (так как подсчитываем количество значений поля *Cod_elev*).

3. Сохраним запрос. Результат запроса представлен на *рисунке 9.12*.

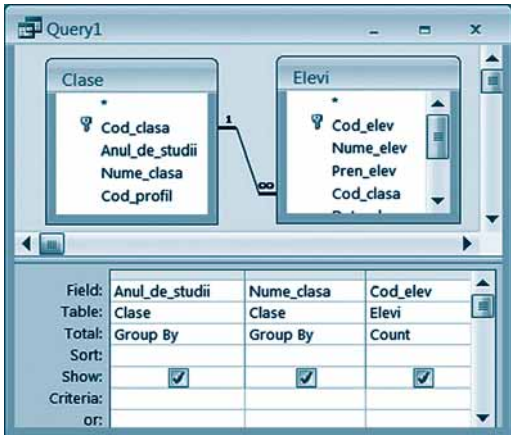
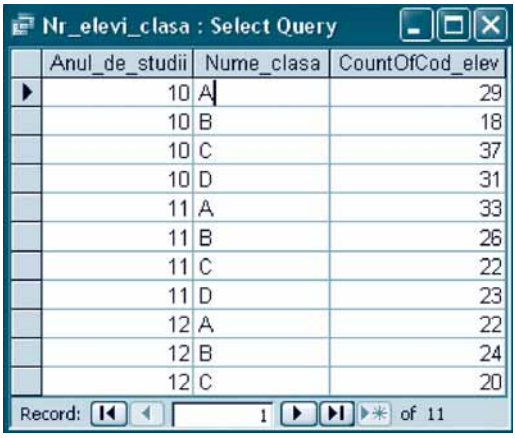


Рис. 9.11



Anul_de_studii	Nume_clasa	CountOfCod_elev
10	A	29
10	B	18
10	C	37
10	D	31
11	A	33
11	B	26
11	C	22
11	D	23
12	A	22
12	B	24
12	C	20

Record: 1 of 11

Рис. 9.12

Замечания:

1. Раскрывающиеся списки ячеек строки *Total* предоставляют различные *глобальные функции* (функции, применимые для групп ячеек с данными) для получения итогов: *Sum*, *Max*, *Min*, *Avg*, *First* и др.
2. В запросах с группировкой данных и итогами тоже можно использовать условия выборки. Например, если для поля *Cod_elev* предыдущего запроса написать условие >25, тогда запрос будет выводить только информацию о классах с числом учеников, большим 25.

9.4.3. Перекрестные запросы

Перекрестный запрос – это выборка данных, записанная в виде двумерной матрицы (то есть в виде таблицы). Такой тип запроса рекомендуется, если необходимо подведение большого числа итогов. При создании перекрестных запросов необходимо иметь ввиду следующие ограничения:

a) названия строк результирующей таблицы могут быть значениями из одного или нескольких полей;
b) названия столбцов результирующей таблицы могут быть значениями лишь одного поля;
c) значения остальных ячеек результирующей таблицы являются результатами вычислений с помощью некоторой глобальной функции;
d) записи результирующей таблицы не могут быть отсортированы по вычисляемым полям.

Составим запрос, который выведет количество часов, предусмотренных еженедельно для каждого предмета в каждом классе базы данных *Liceu*.

1. Создадим запрос на выборку на основании таблиц *Clase*, *Discipline* и *Prof_dis_clasa*, включающий поля *Anul_de_studii*, *Nume_clasa*, *Nume_disciplina* и *Nr_ore_saptamina*.

2. Из меню *Query* выберем *Crosstab Query*. В бланке запроса QBE появляются строки *Total* и *Crosstab* (*рис. 9.13*).

3. Заполним ячейки строк *Total* и *Crosstab* как на *рисунке 9.13*. Таким образом, значения полей *Anul_de_studii* и *Nume_clasa* будут названиями строк результирующей таблицы, значения поля *Nume_disciplina* – названиями столбцов результирующей таблицы, а значения поля *Nr_ore_saptamina* будут просуммированы и заполнят все остальные ячейки результирующей таблицы (*рис. 9.14*).

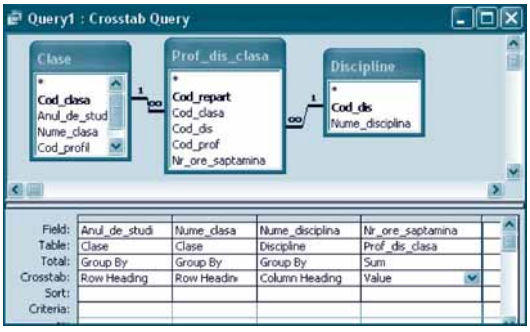


Рис. 9.13

Рис. 9.14

Вопросы и упражнения

❶ Что собой представляют:
a) запрос на вычисление полей; b) запрос с группировкой данных и итогами;
c) перекрестный запрос?

❷ Опишите алгоритм создания:
a) запроса на вычисление полей; b) запроса с группировкой данных и итогами;
c) перекрестного запроса.

❸ Проанализируйте базу данных *Liceu*. Определите с помощью запроса:
a) возраст учеников, выраженный в днях;
b) число преподавателей мужского и женского пола;
c) количество классов каждого из профилей *real* и *umanist*;
d) среднюю месячную заработную плату учителей;
e) число учеников из каждой местности (Loc_elev);
f) число часов, преподаваемое каждым учителем в течение недели;
g) число часов, преподаваемое в каждом классе за неделю;
h) значение минимальной заработной платы;
i) учителей с максимальной заработной платой;
j) количество учителей, имеющих заработную плату ниже средней.

❹ Проанализируйте базу данных *Liceu*. Сформулируйте и реализуйте:
a) два запроса на вычисление полей; b) два запроса с группировкой данных и итогами;
c) два перекрестных запроса.

Контрольный тест

1. Проанализируйте базу данных *Liceu*. Создайте запрос, выводящий список:
a) учеников, имя которых начинается с буквы A;
b) учеников, рожденных в четверг;
c) учителей, преподающих предмет, указанный пользователем.

2. Проанализируйте базу данных *Liceu*. Создайте запрос, который:
a) добавит к полям из таблицы *Elevi* новое поле *Adresa_Web*, составленное из фамилии и имени ученика и последовательности символов „@liceu.md". Например, *Web*-адрес ученика Ion Popa будет Popalon@liceu.md;
b) сгенерирует таблицу *Test* с данными об учениках, не включенных ни в один из классов: 10-й A, а 11-й B, 12-й C (включая данные о *Web*-адресах, полученные в пункте a));
c) удалит из таблицы *Test* данные об учениках из классов 10-го B и 11-го C;
d) обновит *Web*-адреса каждого ученика таблицы *Test*, заменяя подстроку „liceu.md" на „elev.md";
e) добавит в таблицу *Test* данные об учениках женского пола 10-го A класса;
f) увеличит на 40% заработную плату учителям старше 40 лет.

3. Проанализируйте базу данных *Liceu*. Создайте запрос, который выведет число:
a) учеников каждого из профилей *real* и *umanist*;
b) учеников женского и мужского пола.

4. Создайте перекрестный запрос, который выводит число учеников каждого класса базы данных *Liceu* (как на *рис. 9.15*).

Query1 : Crosstab Query

Anul_de_studii	Nume_clasa	Real	Umanist
10	A	29	
10	B	18	
10	C		37
10	D		31
11	A	33	
11	B	26	
11	C		22
11	D		23

Record: 8 of 8

Рис. 9.15

Глава 10

ФОРМЫ И ОТЧЕТЫ

Изучив данную главу, вы сможете:

- распознавать формы и составляющие их элементы;
- создавать формы и подчиненные формы с помощью программы-мастера;
- применять формы для просмотра, обновления, форматирования и проверки правильности данных;
- составлять формы на базе связанных таблиц;
- распознавать отчеты и составляющие их элементы;
- создавать отчеты и подчиненные отчеты с помощью программы-мастера;
- составлять отчеты на базе связанных таблиц;
- группировать данные и подводить итоги внутри отчета.

10.1. Формы

Формы – это объекты базы данных системы Access, предусмотренные специально для создания удобного пользовательского интерфейса, обеспечивающего ввод, редактирование, вывод записей из таблиц и запросов.

Формы увеличивают скорость обработки данных и уменьшают вероятность ошибок. Кроме того, они позволяют выводить данные в более привлекательном формате, нежели в режиме *Datasheet View*.

С помощью форм можно осуществлять проверку данных, вводимых в данную базу из других баз данных, создавать подчиненные формы (формы, входящие в другие формы), поля со списками (для быстрого доступа к записям), *списки опций* и др. Компоненты некоторой формы называются **элементами управления** или **объектами управления**.

10.1.1. Создание формы с помощью программы-мастера

Создадим форму, позволяющую редактировать данные в таблице *Elevi* базы данных *Liceu*.

1. Выделим класс объектов **Forms** на панели объектов *Objects*. В рабочей области окна базы данных появятся опции:

– *Create form in Design view* (создание формы в режиме конструктора);

– *Create form by using wizard* (создание формы с помощью программы-мастера).

Щелкнем дважды на второй опции.

Замечание: *Шаг 1* эквивалентен выбору *New* → *Form wizard*.

2. Появится окно *Form Wizard* (*рис. 10.1*). Из списка *Table/Queries* выделим значение *Table*: *Elevi*. Тогда в списке *Available Fields* появятся идентификаторы таблицы *Elevi*.

Рис. 10.1

Рис. 10.2

3. Выберем поля, которые будут включены в форму, подтверждая выбор кнопкой [>]. Для включения всех полей одновременно, щелкнем на кнопке [>>]. Идентификаторы выбранных полей переходят в список *Selected Fields*. Кнопки [<] и [<<] используют для возврата полей, выбранных ошибочно. После выбора всех полей, щелкнем на кнопке *Next*.

4. Следующее окно *Form Wizard* предоставляет возможность выбора способа вывода полей формы (*рис. 10.2*). Выберем опцию *Columnar* (выбранные на предыдущем шаге поля будут выводиться одно под другим в виде столбца).

5. На следующем шаге выбираем один из допустимых предопределенных стилей формы (*рис. 10.3*).

6. В последнем окне зададим название формы (*Formular_Elevi*). Если выберем опцию *Modify the Form's design* (*рис. 10.4*), то после того как щелкнем на кнопке *Finish*,

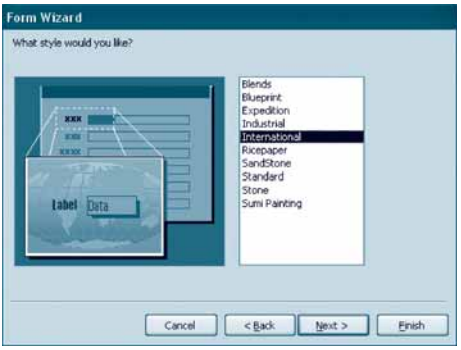


Рис. 10.3

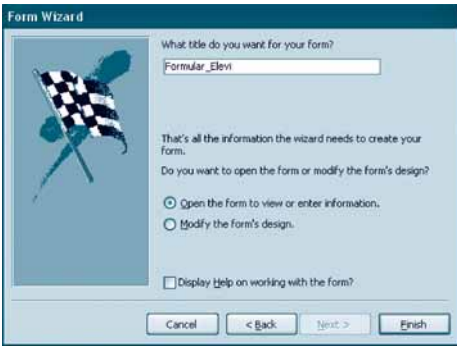


Рис. 10.4

Рис. 10.5

форма откроется в режиме конструктора. В противном случае форма откроется в режиме редактирования данных.

7. Откроем форму в режиме просмотра и редактирования данных (*рис. 10.5*).

Кнопки в нижней части формы используются для пролистывания записей. Так, кнопки |◀, ▶| выводят первую и, соответственно, последнюю запись. Для просмотра предыдущей или последующей записей щелкнем на кнопке ◀ или, соответственно, ▶.

При добавлении новых данных щелкнем на кнопке ▶* либо выбираем *New Record* из меню *Insert*.

Замечания:

1. Форма может быть создана на основе большого числа исходных объектов (таблиц и запросов).
2. Изменения данных в форме приводят к их изменению в исходных объектах.
3. Если хотим вводить данные с помощью формы, она обязательно должна содержать первичный ключ и поля с установленным значением *Yes* для свойства *Required*. Если эти поля не заполнить, то новая запись не будет добавлена.

10.1.2. Создание либо изменение форм в режиме *Design View* (режим конструктора)

Режим *Design View* – это самый исчерпывающий способ создания и изменения форм, потому что он позволяет отредактировать любой элемент управления в форме.

Поверхность формы разбита на несколько зон (*рис. 10.6*). Изменение размеров этих зон осуществляется с помощью мыши.

Нижняя горизонтальная линия и вертикальная линия справа определяют его нижнюю и, соответственно, правую границы.

• Зона **Detail** появляется автоматически при созании или редактировании новой формы. В ней выводятся данные из источников информации для формы.

• Зоны **Form Header** и **Form Footer** не появляются по умолчанию, их наличие возможно при выборе опции *Form Header/Footer* из меню *View*. Эти зоны представляют

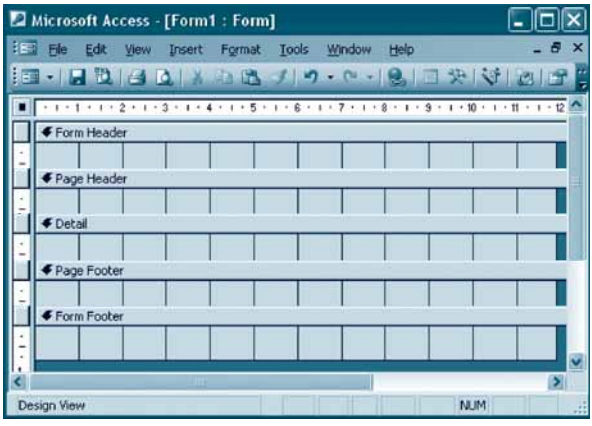


Рис. 10.6

собой *верхний* и *нижний колонтитулы* формы, предназначенные для вывода некоторой полезной информации. Эта информация остается неизменной при смене записей и может, к примеру, содержать рекомендации по применению формы, итоговые данные, данные об авторах, текущее время, дату создания формы и др.

• Зоны **Page Header** (*верхний колонтитул страницы*) и **Page Footer** (*нижний колонтитул страницы*) также выводятся явно, при подключении опции *Page Header/Footer* из меню *View*.

• *Page Header* выводит информацию в верхней части, а *Page Footer* – в нижней части каждой страницы формы.

Замечание: Верхний и нижний колонтитулы добавляются или удаляются только парами.

Помимо рабочей поверхности формы окно *Form Design* содержит панели инструментов *Form Design* и *Toolbox*.

• Панель *Form Design* применяется для быстрого выполнения некоторых действий, без того, чтобы обращаться к основному меню окна конструктора формы.

• Панель *Toolbox* применяется для добавления элементов управления на поверхность формы.

Проанализируем некоторые ее инструменты:


Select Objects не используется для помещения элемента управления, а только для выбора (если он активирован) уже существующих в форме элементов.


Control Wizard (когда активирован) позволяет использовать программу-мастер для создания некоторых элементов управления.


Label создает текстовое окно, в котором можно вывести статический текст (пользователь не сможет изменить этот текст в режиме редактирования и просмотра).


Text Box создает текстовое окно, в котором пользователь сможет изменять текст.


Option Group создает прямоугольную кассету, в которую можно поместить выключатели, переключатели и флажки. В случае переключателей пользователь сможет выбирать одновременно только одну из предложенных опций.


Toggle Button создает выключатель – кнопку для активации или дезактивации некоторого состояния.


Option Button создает кнопку (переключатель), также применяемую для включения или выключения некоторого состояния (или опции). Обычно, используется целая группа кнопок такого типа для организации исключающих опций.


Check Box создает флажок для подтверждения или запрета некоторого состояния.


Combo Box создает поле со списком, сформированное из текстового поля, в которое пользователь может вписать значение или выбрать из раскрывающегося списка.


List Box создает открытый список, из которого можно выбрать одно значение.


Command Button создает командную кнопку, с помощью которой может быть выполнена некоторая команда.


Image отображает нередактируемый рисунок, не являющийся объектом OLE.


Unbound Object отображает свободный объект OLE (например, созданный с помощью Microsoft Draw), который не является значением поля типа OLE некоторой записи.

Bound Object отображает объект OLE – содержимое некоторого поля типа OLE (например, рисунок).

Page Breack создает разрыв страницы – начало новой страницы при печати.

Tab Control создает вкладки в форме, на каждой из которых могут размещаться свои элементы управления. С его помощью можно реализовать переключение между формами.

Subform/Subreport создает подчиненную форму или подчиненный отчет.

Line создает отрезки прямых, используемые для разграничения разделов и дизайна (цвет и толщину линий можно менять).

Rectangle создает прямоугольники, используемые для дизайна.

More Controls выводит инструменты для создания других элементов управления.

• С точки зрения функциональности, различают следующие **категории элементов управления**:

a) *связанные элементы управления*;

b) *вычисляемые элементы управления*;

c) *независимые элементы управления*.

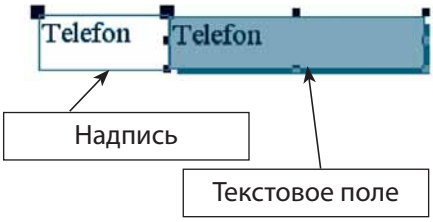


Рис. 10.7

Связанные элементы управления служат для вывода и редактирования данных из полей таблиц и запросов, на основе которых была построена форма. Связанный элемент управления состоит из поля (в котором выводятся данные) и надписи, ассоциируемой с этим полем (в которой выводится пояснительный текст, например, имя поля). По умолчанию, надпись связанного поля совпадает с идентификатором поля, значения которого выводятся в поле данных этого элемента управления (*рис. 10.7*). В режиме *Datasheet View* в поле, изображенном на *рисунке 10.7* формы, созданной ранее (*рис. 10.5*), будет отображаться номер телефона.

Можно менять размеры и перемещать компоненты элементов управления. Если при позиционировании индикатора мыши над элементом появляется изображение ладони, то методом *Drag & Drop* можно перемещать одновременно обе компоненты элемента управления. Поместив указатель мыши над одной из двух контактных точек в левом верхнем углу каждой компоненты, он приобретает форму указательного пальца. В этом случае можно перемещать отдельно каждую компоненту элемента управления.

Добавлять связанные элементы управления в форму можно путем перетаскивания нужного элемента из списка, открывающегося опцией *Field List* из меню *View*. Можно это сделать и путем выбора на панели инструментов *ToolBox* элемента *Text Box* ab|, размещения на свободном от других элементов месте на поверхности формы и записи в кассету *Unbound* имени нужного поля.

Несмотря на свое название, в текстовом поле можно вывести, кроме текстов, числовые, календарные и другие данные.

По умолчанию, Access выравнивает текст по левой границе поля, а числа – по правой.

Если в текстовом поле должен выводиться большой текст (например, содержимое полей типа *Memo*), то его можно увеличить до нужных размеров. Можно также снабдить его линейкой прокрутки. Для этого щелкнем мышью на кнопке *Properties*, затем выделим значение *Vertical* свойства *Scroll Bars* из закладки *Format* (в появившемся окне, *рис. 10.9*).

Вычисляемые элементы управления применяются для вывода значений выражений и могут обрабатываться точно так же, как и связанные элементы управления. Добавить такой элемент на поверхность формы можно с помощью инструмента *Text Box*, только в текстовом поле *Unbound* записывается выражение (после знака =), значение которого появится в этом поле при просмотре.

Например, элемент управления на *рисунке 10.8*, будучи добавленным в ранее созданную форму, будет выводить возраст учеников.

Независимые элементы управления не связаны с информацией в таблицах или запросах, а значит, не меняются при переходе от одной записи к другой. Применяются для создания дизайнерских эффектов.

Рис. 10.8

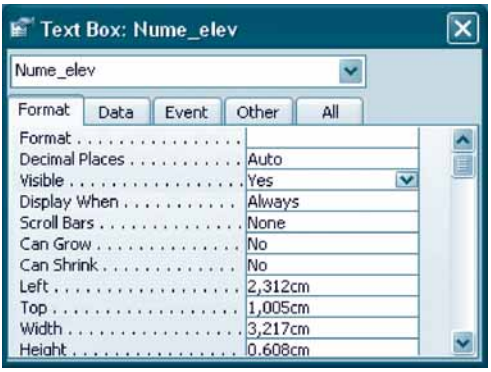


Рис. 10.9

• **Свойства элемента управления** определяют различные его характеристики: цвет, размеры, источник данных, позицию на поверхности формы, реакцию на определенные действия пользователя и др.

Изменить свойства элемента управления можно в специальном окне, появляющемся при двойном нажатии кнопки мыши, выбрав ею нужнй элемент (*рис. 10.9*). Все свойства сгруппированы в четыре категории (закладки): *Format*, *Data*, *Event*, *Other*. Группа *All* включает все свойства, упорядоченные в алфавитном порядке.

- Группа **Format** включает свойства, определяющие внешний вид объекта (размеры, цвет, формат и др.).
- Группа **Data** включает свойства относительно источника записей, управляемых данным элементом.
- Группа **Event** включает список событий (действия со стороны пользователя или приложения), на которые может реагировать элемент.
- Группа **Other** включает все остальные свойства. Они относятся к окнам операционной системы Windows.

• Форму можно распечатать (как и любой другой объект базы данных) используя кнопку *Print* на панели инструментов или опцию *Print* из меню *File*.

Для того чтобы просмотреть формуляр в том виде, в каком он будет распечатан, воспользуйтесь кнопкой *Preview* на панели инструментов или выберите опцию *Preview* из меню *File*.

10.1.3. Подчиненные формы

Подчиненная форма – это форма, включенная в другую форму. Включить другую форму в данную можно с помощью программы-мастера или путем самостоятельного проектирования. Мы рассмотрим только случай применения мастера, который является доступным, если выбран инструмент *Control Wizard* в кассете *Toolbox*.

Пусть дана форма *Clase*, представленная на *рисунке 10.10,* для вывода и редактирования данных о классах базы данных *Liceu*. Включим в нее подчиненную форму, в которой будут отображаться данные об учениках выбранного, в основной форме, класса.

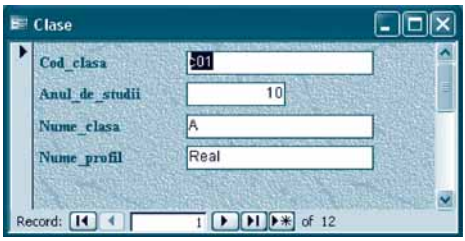


Рис. 10.10

1. Откроем форму *Clase* в режиме *Design*. Активируем сначала кнопку *Control Wizard*, затем, кнопку *Subform/Subreport* из кассеты *Toolbox* и щелкнем мышью на поверхности формы, в месте, где появится подчиненная форма. Появилось окно *SubForm Wizard*, в котором выделим таблицу *Elevi*.

2. В следующем диалоговом окне выделим поля таблицы *Elevi*, а в третьем окне подтвердим название поля связи (Cod_clasa) между основной формой и подчиненной.

3. В последнем окне *SubfForm Wizard* напишем имя подчиненной формы (*Lista_elevi*). Перейдя в режим просмотра *Form View*, увидим список учеников класса, выбранного в основной форме.

Замечание: Подчиненная форма *Lista_elevi* автоматически сохранится на диске при сохранении основной формы.

Вопросы и упражнения

❶ Для каких целей применяются формы?
❷ Какие объекты базы данных могут быть источниками данных для форм?
❸ Охарактеризуйте зоны формы в режиме конструктора.
❹ Какова роль элементов управления формы?
❺ Опишите категории элементов управления.
❻ Каким образом можно изменить свойства некоторого элемента управления?
❼ Проанализируйте базу данных *Liceu*. Создайте с помощью программы-мастера форму для редактирования данных:
a) об учителях базы данных, включая их адрес; b) о преподаваемых предметах базы данных.
❽ Добавьте в созданную ранее форму текстовое поле, в котором появятся текущие дата и время.
❾ Добавьте в созданную ранее форму текстовое поле, в котором появится число дней, оставшихся до конца: a) текущего года; b) текущего учебного года.
❿ Проанализируйте базу данных *Liceu*. Создайте форму, выводящую данные о каждом классе, содержащую подчиненную форму со списком учителей, преподающих в классе, выбранном в основной форме.
⓫ Проанализируйте базу данных *Liceu*. Создайте форму, выводящую данные о каждом классе, содержащую подчиненную форму со списком предметов, изучаемых в классе, выбранном в основной форме.
⓬ Проанализируйте базу данных *Liceu*. Создайте форму, выводящую данные о каждом учителе базы, содержащую подчиненную форму со списком предметов, преподаваемых учителем, выбранным в основной форме.

10.2. Отчеты

Отчеты представляют собой конечный продукт базы данных. В них комбинируются данные из таблиц, запросов и форм для печати либо для записи на диск в формате, удобном, воспринимаемом для чтения. В отличие от форм, отчеты:

a) специально предназначены для печати;

b) не могут менять данные в таблицах или запросах, на основании которых они созданы;

c) не выводят данные в виде таблицы (нет режима типа *Datasheet View*).

Как и в случае форм, отчеты могут создаваться в режиме программы-мастера и в режиме конструктора.

10.2.1. Создание отчета с помощью программы-мастера

1. Выделим класс объектов **Reports** на панели объектов *Objects*. Выберем *New* → *Report Wizard*.

2. Появится окно *Report Wizard* (*рис. 10.11*). Из комбинированного списка *Tables/Queries* поочередно выберем таблицы *Clase*, *Elevi* и запрос *Virsta (см. тему 9.4.1)*, а из кассеты *Available Fields* выберем поля *Anul_de_studii*, *Nume_clasa* (таблицы *Clase*), *Nume_elev*, *Pren_elev* (таблицы *Elevi*) и *Virsta* (запроса *Virsta*).

3. В следующем окне *Report Wizard* выберем поля, по которым будут группироваться записи (*рис. 10.12*).

4. В третьем окне *Report Wizard* указываем поля, по которым записи будут отсортированы (*рис. 10.13*).

С помощью кнопки *Summary Options* можем, для каждого числового поля, указать одну или несколько вычислительных опций: сумма, среднее арифметическое, максимальное или минимальное значение. Для вычисления среднего возраста учеников каждого класса выберем опцию *Avg* (*рис. 10.14*). Если отметить

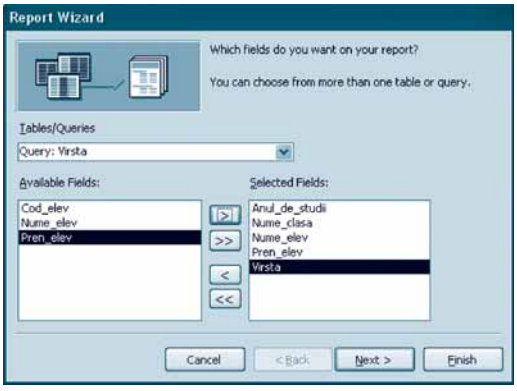


Рис. 10.11

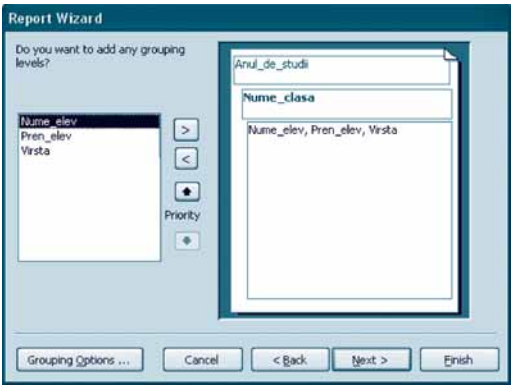


Рис. 10.12

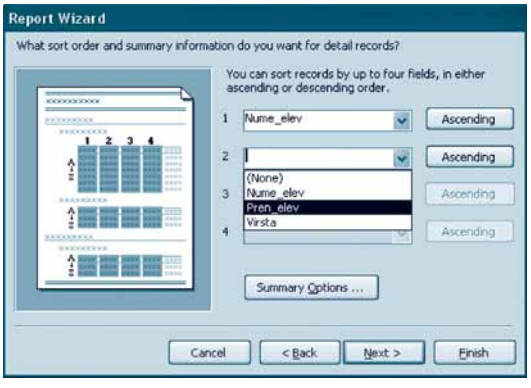


Рис. 10.13

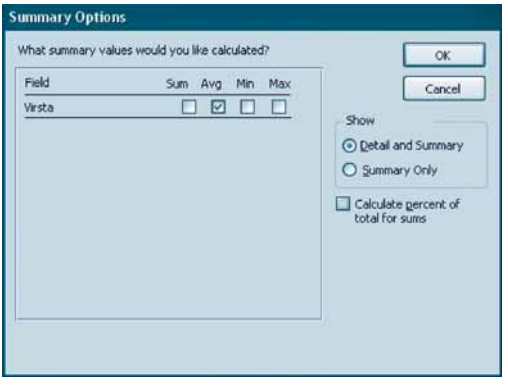


Рис. 10.14

опцию *Summary Only*, то средний возраст учащихся будет подсчитан только в целом по лицею.

5. В следующих трех окнах *Report Wizard* указываем способ размещения записей на странице, стиль и, соответственно, заголовок отчета. На *рисунке 10.15* представлена часть созданного отчета.

Рис. 10.15

Для печати отчета щелкнем по кнопке *Print* на панели инструментов.

Замечание: Создание подчиненных отчетов осуществляется подобно созданию подчиненных форм.

10.2.2. Изменение отчетов в режиме *Design View*

Режим *Design View* для создания и изменения отчетов похож на режим *Design View* форм. Как и формы, отчеты состоят из пяти основных зон: *Report Header*, *Page Header*, *Detail*, *Page Footer* и *Report Foter*. В отчете могут также быть и зоны для групп данных. Например, на *рисунке 10.16* представлен отчет, создание которого было описано ранее, открытый в режиме конструктора. Обратите внимание на то, что кроме 5 основных зон появились зоны *Anul_de_studii Header*, *Nume_clasa Header*, *Anul_de_studii Footer* и *Nume_clasa Footer*. Действия с элементами управления отчета осуществляются подобно таковым из форм.

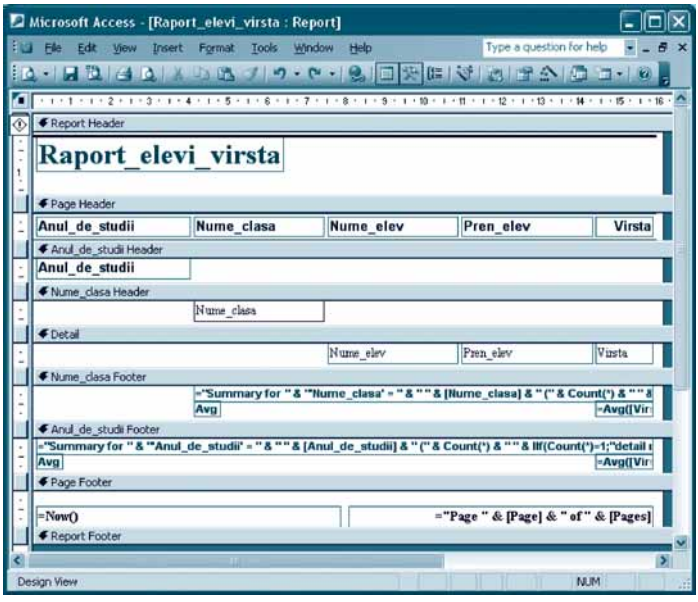


Рис. 10.16

10.2.3. Создание диаграмм в отчетах

Известно, что графическое представление данных воспринимается легче, особенно если сравниваются числовые значения.

В системе управления базами данных диаграммы считаются специальным типом отчета. Можем создать диаграмму на основе таблицы или запроса.

Пример: Создадим диаграмму, представляющую распределение по классам учеников базы данных *Liceu*, основываясь на данных запроса *Nr_elevi_clasa*, созданном ранее в подпараграфе 9.4.2.

1. Выделим класс объектов **Reports** на панели объектов *Objects*. Нажмем на кнопку *New*. Появится окно *New Report*, из которого выберем тип отчета *Chart Wizard* и запрос *Nr_elevi_clasa* из раскрывающегося списка. Подтвердим выбор нажатием кнопки *Ok*.

2. Появилось первое окно *Chart Wizard*, в котором выберем поля *Anul_de_studii*, *Nume_clasa* и *CountOfCod_elev*.

3. В следующем окне *Chart Wizard* укажем тип диаграммы: *Column Chart* (столбиковая диаграмма или гистограмма).

4. В третьем окне *Chart Wizard* (*рис. 10.17*) определяем источники для категорий данных (*Anul_de_studii*), значений данных (*CountOfCod_elev*) и легенду (*Nume_clasa*).

5. В последнем окне *Chart Wizard* задаем имя диаграммы и уточняем, будет или нет выводиться легенда. Получим диаграмму как на *рисунке 10.18*.

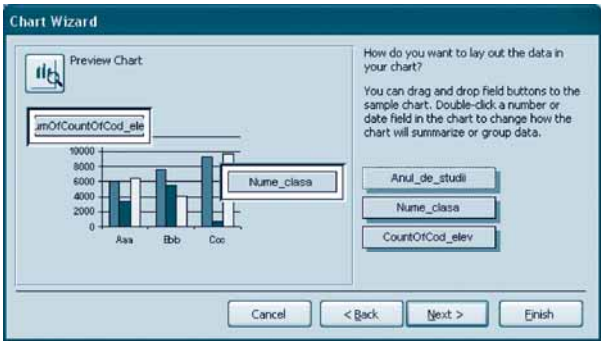


Рис. 10.17

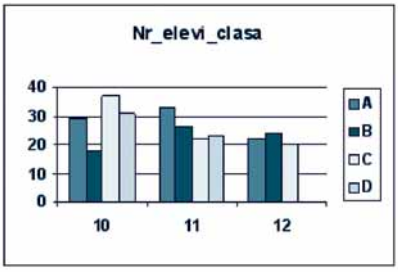


Рис. 10.18

Вопросы и упражнения

❶ Для каких целей применяются отчеты?

❷ Какие объекты базы данных могут служить источниками данных для отчетов?

❸ Какова роль диаграмм в базе данных?

❹ Проанализируйте базу данных *Liceu*. Создайте отчет, в котором выводится:
a) среднее арифметическое часов в неделю для каждого года обучения;
b) среднее арифметическое часов в неделю для каждого профиля;
c) общее количество часов в неделю для каждого года обучения и в целом по лицею.

❺ Проанализируйте базу данных *Liceu*. Создайте диаграмму, представляющую графически число:
a) учащихся кадого профиля;
b) учащихся из каждой местности проживания;
c) учителей каждого пола;
d) часов за неделю, преподаваемых в каждом классе.

❻ Проанализируйте базу данных *Liceu*. Создайте диаграмму, представляющую графически число учеников разного возраста.

10.3. Администрирование баз данных (факультативно)

Для того чтобы база данных нормально функционировала, она должна постоянно развиваться и поддерживаться. Обычно эти операции осуществляет администратор базы данных. В этом параграфе рассмотрим некоторые действия, необходимые для поддержки функциональности баз данных.

10.3.1. Сжатие и восстановление баз данных

• Со временем, в результате добавлений, обновлений, изменений различных объектов базы, растет *размер файла базы данных* и уменьшается *производительность*, то есть быстрота доступа и обработки данных.

Эти последствия объясняются не только ростом объема данных, но и:

– наличием в базе некоторых временных объектов (спрятанных от пользователя), даже если система больше в них не нуждается;
– присутствием в файле базы свободных пространств, получившихся в результате удаления некоторых объектов базы данных.

В таких случаях необходимо *сжатие* базы данных – операции по дефрагментации файла базы, то есть, по удалению внутренних свободных пространств.

• Иногда возникают повреждения файла базы данных. Произойти это может по внешним причинам (например, отключение источника питания, поломки в сети) или в случае одновременного доступа к файлу базы большого числа пользователей.

В этом случае база нуждается в *восстанослении*.

Для **сжатия и восстановления базы данных**:

1. Закроем базу данных, оставив открытым приложение Access.

2. Выполним *Tools* → *Database Utilities* → *Compact and Repair Database*.

3. Появляется диалоговое окно *Database to Compact From,* в котором уточним название базы данных, которая будет сжата и восстановлена. Подтвердим действие нажатием кнопки *Compact*.

4. Появляется диалоговое окно *Compact Database Into,* в котором впишем имя, под которым сохранится сжатая и восстановленная база данных. Подтвердим действие нажатием кнопки *Save*.

10.3.2. Создание резервных копий

Access может восстановить поврежденную базу данных полностью или частично. В последнем случае потерянные данные можно восстановить из резервной копии базы данных.

Следовательно, администратор базы должен периодически создавать такие копии.

Это действие можно осуществить традиционным образом, скопировав файл базы в другую папку, либо выполнив *Tools* → *Database Utilities* → *Back Up Database*.

10.3.3. Обеспечение безопасности данных

В базе данных могут содержаться конфиденциальные данные, но в то же время доступ к ней по сети (возможно и через Интернет) могут иметь многие пользователи.

В этом случае предпринимаются действия по защите от несанкционированного доступа к данным. Более того, различные пользователи могут иметь различные права доступа. Например, некоторые пользователи могут иметь право вводить информацию, другие – только просматривать определенную ее часть.

С этой целью, каждому пользователю или группам пользователей предоставляются:

– уникальное имя для доступа;

– пароль;

– права доступа или владельца.

Самый простой способ защиты базы данных от несанкционированного доступа – это установление пароля, без которого базу невозможно открыть.

Для **создания пароля**:

1. Закроем базу данных, оставив открытым приложение Access.

2. Выполним *File* → *Open*. Выделим файл базы данных, а затем, опцию *Open Exclusive* из раскрывающегося списка *Open*.

3. Выполним *Tools* → *Security* → *Set Database Password*. Появляется диалоговое окно *Set Database Password*, в котором устанавливаем и подтверждаем пароль.

Замечания:

1. Существуют и другие действия по администрированию баз данных с целью защиты данных. Например, данные базы можно **шифровать**, чтобы их невозможно было просматривать без **дешифрования** с помощью текстовых редакторов или специальных программ.
2. Создание групп, паролей и прав доступа осуществляется из каскадного меню *Security* меню *Tools*.

Исследуем, создаем и практикуемся

На *рисунке 10.19* представлены связи между таблицами базы данных „Club sportiv” (поле *Platit* является логическим).

1. Создайте базу данных согласно рисунку и заполните ее записями.

2. Объясните, почему поля **Cod_client**, **Cod_abonament** и **Luna** вместе образуют первичный ключ таблицы *Evidenta*.

3. Создайте запрос, который отобразит:

a) список клиентов, фамилии которых начинаются на буквы A или C;

b) список тренеров, чьи имена состоят по меньшей мере из 5 букв;

c) среднюю заработную плату тренеров;

d) список тренеров с заработной платой большей, чем средняя;

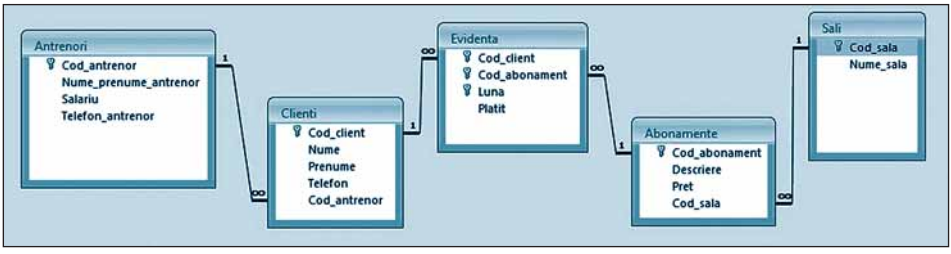


Рис. 10.19

e) список кодов абонементов с максимальной ценой;
f) количество клиентов каждого тренера;
g) имена тренеров с наибольшим числом клиентов;
h) общую сумму денег, которую заплатил каждый клиент;
i) имя клиентов, которые заплатили наибольшую сумму денег (всего);
j) залы, наиболее посещаемые клиентами.

4. Сформулируйте и создайте еще 5 запросов.
5. Создайте формы для ввода и редактирования данных.

Создаем в команде

1. Разработайте концептуальную модель реляционной базы данных с информацией:
   a) о дневнике студента;
   b) о странах Европы;
   c) об автомобилях;
   d) о книгах;
   e) о сайтах.
2. Создайте таблицы разработанной базы данных.
3. Установите первичные ключи и отношения между таблицами базы данных.
4. Заполните каждую таблицу как минимум 5-ю записями.
5. Сформулируйте и выполните как минимум:
   a) 4 запроса на выборку, из которых один – с параметром;
   b) 3 запроса на обновление записей;
   c) 2 итоговых запроса;
   d) один перекрестный запрос;
   e) один запрос на удаление записей.
6. Создайте две формы для добавления/обновления записей в таблицах базы данных.
7. Создайте два отчета со сведениями из базы данных.
8. Создайте диаграмму, которая отражает результаты созданных отчетов.

Контрольный тест

1. Установите истинность высказываний:
   a) При изменении данных в форме эти данные изменяются и в ассоциированной с формой таблице.
   b) Access сохраняет изменения в записях сразу же после их заполнения, даже если закрыть форму без сохранения.
   c) Верхний и нижний колонтитулы формы удаляются или перемещаются только в паре.
2. Как можно изменить размеры текста некоторого элемента управления при помощи кнопок на панели инструментов?
3. Создайте с помощью программы-мастера формуляр для редактирования адресов преподавателей базы данных *Liceu*.
4. Создайте в режиме конструктора форму с именем *Test,* содержащую на зеленом фоне три текстовых поля для редактирования фамилии, имени и даты рождения учеников базы данных *Liceu*.
5. Создайте отчет для базы данных *Liceu,* выводящий число учителей разного пола для каждого преподаваемого предмета.
6. Создайте диаграмму, представляющую графически данные отчета, созданного в упражнении 5.

Глава 11

WEB-ДОКУМЕНТЫ

После изучения данной главы, вы сможете:

- распознавать *Web*-документы;
- проектировать *Web*-документы;
- создавать *Web*-документы с помощью офисных приложений.

11.1. Понятия и концепты

В 10-м классе вы изучили основные понятия из области компьютерных сетей. Познакомились с элементами топологии и архитектуры сетей, технологии взаимодействия в компьютерных сетях, сетью ИНТЕРНЕТ, ее компонентами и гаммой предоставляемых услуг. Вы уже знаете, что информация и управление ею являются наиболее важными элементами глобальной сети, а самым современным сервисом распространения и поиска информации в ИНТЕРНЕТе является сервис WWW (World Wide Web – Всемирная Паутина).

Знакомо вам и понятие *Web*-страницы, а также программы просмотра (навигации). Вы умеете находить *Web*-страницы по их адресам URL (Uniform Resurse Locator), просматривать их информационные элементы, копировать необходимую информацию, представленную в различных форматах. Несомненно, вы заметили, что *Web*-страницы могут содержать большое количество элементов разной природы: тексты, рисунки, звуковые фрагменты и видеофрагменты, командные кнопки и ссылки на другие *Web*-страницы. Такая комбинация элементов возможна благодаря гипертексту и языку HTML (Hyper Text MarkUpLanguage – язык разметки гипертекста).

***Web*-документ – это набор взаимосвязанных ссылками *Web*-страниц, совместно со встроенными информационными элементами и элементами управления. Опубликованный в глобальной сети *Web*-документ называют *Web*-сайтом.**

11.2. Типы *Web*-документов

Наиболее распространенными *Web*-документами являются **документы HTML** – документы, в которых информация, ссылки, элементы управления и метаданные описаны на языке HTML. При открытии файла HTML программа навигации интерпретирует специальные метки, встречаемые в тексте, в соответствии с правилами языка. Документы HTML имеют расширение HTML (иногда HTM, если

операционные системы позволяют использовать в расширении имени файлов не более трех символов). Документ HTML может быть создан с помощью любого редактора текстов и просмотрен с помощью программы навигации (*рис. 11.1*).

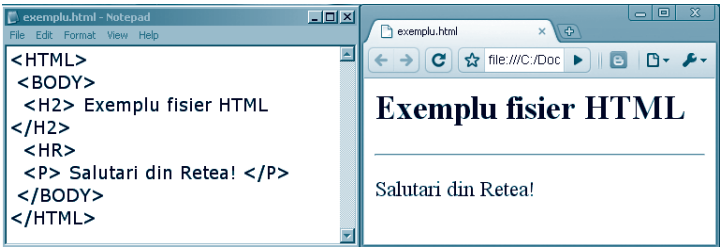


Рис. 11.1. Содержимое файла HTML (слева) и его интерпретация программой навигации (справа)

Многие *Web*-пользователи называют документы, описанные с помощью языка HTML, *документами в формате HTML*.

Документы PHP – это *Web*-страницы, содержащие элементы, созданные с помощью языка PHP (Hypertext Preprocessor). Использование PHP позволяет создание страниц с динамическим содержанием: содержание страницы PHP обрабатывается *Web*-сервером, затем генерируется страница HTML, которая передается на компьютер-клиент, где просматривается с помощью программы навигации. Документы PHP часто используются для динамического просмотра информации из сетевых баз данных.

Документы ASP – это *Web*-страницы, разработанные с помощью технологии Active Server Page. Как и в случае документов PHP они базируются на запрограммированных на языке высокого уровня компонентах, называемых объектами, которые могут передаваться с одной страницы на другую.

Графические документы SVG (Scalable Vector Graphics) реализованы на языке XML* для качественного воспроизведения элементов двумерной графики. Графические инструменты, встроенные в этот язык, позволяют использовать векторную графику в *Web*-документах (*рис. 11.2*).

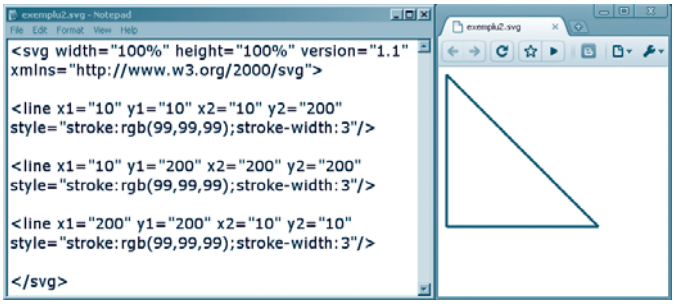


Рис. 11.2. Содержимое графического *Web*-документа файла (слева) и его интерпретация программой навигации (справа)

* Метаязык XML (eXntensible Meta Language) является языком, похожим на HTML, спроектированный с целью передачи данных между приложениями, в том числе по сети.

11.3. Разработка и создание *Web*-документа

Обычно, термин *документ* применяют для отдельного файла, содержащего информацию различной природы, однако, из одной области. *Web*-документ, или *сайт,* в подавляющем большинстве случаев является набором файлов (называемых страницами) с внутренними и внешними ссылками (переходами). В повседневной деятельности вы столкнетесь с необходимостью не только пользоваться *Web*-документами, но и создавать собственные. Современные *Web*-инструменты позволяют автоматизировать, в достаточно большой степени, создание некоторых стандартных страниц, таких как *персональные* страницы в социальных сетях, *блоги, вики*-страницы. В случае документов не совсем стандартных, эти инструменты позволяют лишь спроектировать документ. Будучи предназначенным для широкой публики, *Web*-документ должен быть разработан так, чтобы пользователь имел, по возможности, наиболее удобный и быстрый доступ к информации, структурированной в соответствии с критериями, принятыми в области, которой посвящен сайт. Это приводит нас к выводу, что необходимо *предварительное проектирование Web*-документа.

Независимо от типа *Web*-документа, процесс проектирования начинается с *установления целей* его применения. Установив цели, проще определить тип документа и инструменты, необходимые для его разработки. Например, если вы хотите опубликовать подборку собственных эссе и сочинений ваших друзей, достаточно выбрать страницы HTML. Этот же тип пригоден и для создания не очень большой галереи изображений. Если же вам понадобится открывать изображения геометрических фигур на плоскости, то наиболее удачным окажется выбор документов SVG, что приведет к повышению качества и плавности линий изображений на странице.

После определения целей, можете перейти к следующему этапу – *определению требований,* предъявляемых к документу. Существует несколько категорий требований: относительно содержания (контента) и его дозирования на странице, относительно цветов и формата символов, относительно навигации внутри страниц и перехода с одной страницы на другую. Эти требования специфичны для каждого документа и устанавливаются в зависимости от целевой группы, для которой разрабатыватся документ, и от специфики его контента.

> ***Web*-дизайн – это процесс создания *Web*-страницы или *Web*-документа, который сочетает в себе реализацию этических и эстетических требований, а также требований, предъявляемых к контенту и удобству навигации, создаваемого документа.**

На следующем этапе определяется *структура Web-документа*: структура главной страницы, ее содержание, размещение информативных и управляющих элементов, данные об авторе, структура и содержание подчиненных страниц, способ обращения к этим страницам. Для избежания неудачного размещения элементов и появления неверных ссылок рекомендуется создать графическую схему документа. На схеме будущие страницы представляются в виде узлов, а ориентированные связи между узлами отображают ссылки между страницами документа.

Пример: Допустим, вы хотите создать персональный *Web*-документ, содержащий: личные данные, фотогалерею, отдельные страницы, посвященные учебе, друзьям, хобби и набор ссылок на различные образовательные сайты.

Главная страница будет содержать заголовок (название) документа, ссылки на подчиненные страницы, контактные данные автора и некоторые личные данные. Каждая подчиненная страница будет тематической: лицей, друзья, фотогалерея, хобби. Страница со ссылками, посвященная процессу обучения, будет подчинена странице с данными о лицее (*рис. 11.3*).

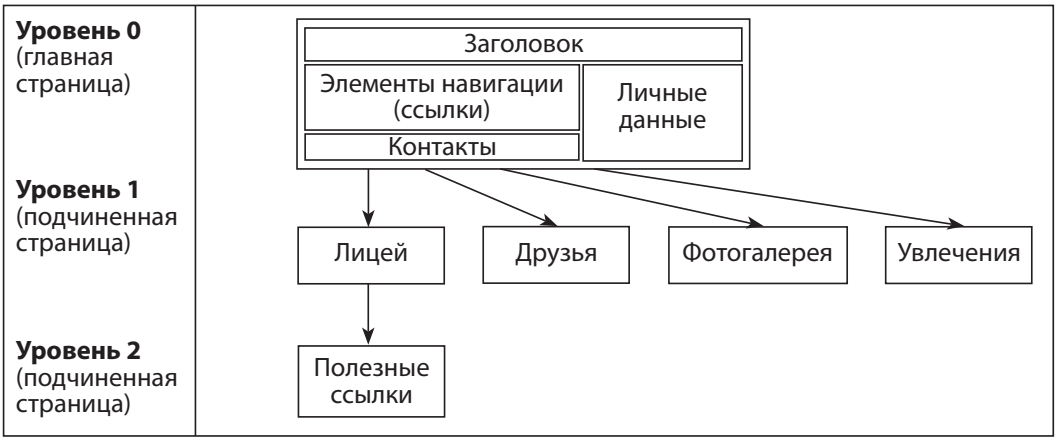


Рис.11.3. Схема структуры персонального сайта. Приведены детали главной страницы

Считается хорошей практикой возможность прямой ссылки на главную страницу с каждой страницы документа. Чтобы не загружать схему структуры *Web*-документа, эти ссылки указывать не обязательно.

После проектирования структуры ссылок можно перейти к уточнению содержания подчиненных страниц. Рекомендуется применение одинаковой модели для всех страниц, с минимальными изменениями, зависящими от контента.

Программирование страниц Web-документа. Это следующий этап разработки. В зависимости от применяемой автором документа техники, этот этап может содержать не только процесс программирования страниц с помощью некоторого языка *Web*-программирования, но и заполнение этих страниц контентом. Если применяются системы управления контентом (такие, как sites.google.com, wordpress.com), элементы программирования становятся менее очевидными, но вместе с тем теряется свобода в организации структуры и содержания документа.

Тестирование Web-документа. Сначала документ просматривается с помощью программ навигации на локальном компьютере. Проверяется правильность вывода с точки зрения структуры, контента, цветов и ссылок. В случае обнаружения ошибок возвращаемся к этапу разработки. После исправления обнаруженных ошибок документ готов к публикации. Для этого его нужно разместить на *Web*-сервере. Если для разработки документа использовались online-системы управления контентом, то размещение на *Web*-сервере является опциональным.

Поддержка и развитие документа. Периодически необходимо обновлять содержимое страниц сайта. Может возникнуть необходимость изменения структуры, дизайна и др. Для сайтов, созданных в *Web*-среде, редактирование производится непосредственно, путем замены старых версий страниц на новые. Если же сайт был создан на локальном компьютере, а затем опубликован, новая версия сначала тестируется, а затем публикуется на *Web*-сервере, заменяя старую версию сайта.

11.4. Создание *Web*-документа с помощью офисных приложений

В повседневной деятельности мы применяем для обработки информации различные прикладные программы: редакторы текстов, графические редакторы, системы создания электронных презентаций, табличные процессоры и другие специализированные приложения.

Файлы, созданные в этих системах, имеют специфические форматы, которые не воспринимаются программами навигации. Однако достаточно часто эта информация должна быть отражена в *Web*-документе или в отдельной *Web*-странице. Простым решением этой проблемы, которое предоставляют большинство приложений общего пользования, является автоматическое создание *Web*-документа, структура и содержание которого добросовестно повторяют оригинальный документ данного приложения. Достигается это простым выбором специальных опций при сохранении документа или путем экспорта.

Стандартные типы *Web*-документов, созданных таким образом, являются **документами HTML**. Для автоматического получения *Web*-документа при его сохранении необходимо выбрать опцию типа документа: *Web Page* (*рис. 11.4*).

Рис. 11.4. Сохранение файлов как *Web*-документов в различных офисных приложениях

Необходимо отметить, что в случае, когда исходный документ содержит неоднородную информацию или элементы, которые не могут быть воспроизведены только инструментами HTML, приложение создает весь набор необходимых подчиненных файлов, помещенных в папку, имя которой формируется из имени сохраняемого документа и добавления к нему суффикса **files** (*рис. 11.5*).

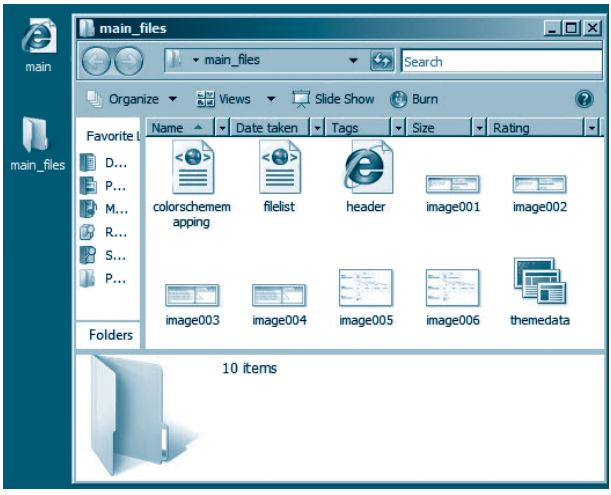


Рис. 11.5. При сохранении файла с именем **main** в формате *Web*-страницы был автоматически создан и каталог **main_files**, содержащий несколько файлов с изображениями и со служебной информацией, необходимыми для правильного представления *Web*-страницы **main**

Большинство офисных приложений предлагают дополнительную возможность генерирования *Web*-документа: *Web Page, Filtered* (отфильтрованная *Web*-страница). Отличие такой страницы от обычной состоит в том, что из нее исключены излишние коды и служебная информация, занимающие достаточно большой объем памяти. Эта опция помогает уменьшить объем *Web*-документа, однако, возникает риск неправильного его отображения в некоторых программах навигации.

Расширение файлов, содержащих *Web*-документ, в обоих случаях, HTML. Расширения дополнительных файлов могут варьировать в зависимости от офисного приложения, в котором генерировался *Web*-документ.

Другой тип *Web*-документов, которые могут генерироваться автоматически с помощью офисных приложений, это ***Web*-архив**. Для создания *Web*-архива из файла специфического типа в меню выбирается опция сохранения документа: *Single File Web Page* (*рис. 11.6*).

Рис. 11.6

Этот тип документа включает все элементы исходного документа, сохраняет отношения между элементами, на которые сделаны внутренние ссылки, и может быть открыт программой навигации на локальном компьютере, на котором находится.

Расширение файлов, содержащих *Web*-архивы, MHT.

Вопросы и упражнения

❶ Как еще называют *Web*-документ? Какие *Web*-документы называют сайтами?

❷ Какие из следующих типов *Web*-документов реализуются с помощью языков программирования высокого уровня:
a) документы HTML;
b) документы ASP;
c) документы PHP;
d) документы SVG?

❸ Какой тип имеют *Web*-документы, позволяющие просмотр форматированного текста с помощью языка разметки?

❹ Какой тип имеют *Web*-документы, позволяющие прямое создание графических элементов в приложении навигации?

❺ Перечислите этапы проектирования *Web*-документа.

❻ Спроектируйте схему структуры некоторого сайта – виртуальной микробиблиотеки с произведениями, максимум, трех авторов. Для каждого автора необходимо поместить на отдельных страницах, самое большее, по два произведения. Главная страница будет содержать список ссылок на страницы каждого автора в отдельности. Страница каждого автора будет содержать ссылки на страницы с их литературными произведениями.

❼ Выберите некоторую область, для которой хотели бы спроектировать *Web*-сайт. Определите цель применения сайта. Спроектируйте схему его структуры.

❽ С помощью редактора текста создайте документ, содержащий: форматированный текст, таблицы, диаграммы, рисунки. Воспользуйтесь опциями сохранения документа и сгенерируйте автоматически *Web*-документ, эквивалентный оригиналу. Чем будет отличаться от *Web*-документа сгенерированный *Web*-архив?

❾ Создайте электронную презентацию, содержащую форматированный текст, таблицы, диаграммы, рисунки. Воспользуйтесь опциями сохранения документа и сгенерируйте автоматически *Web*-документ, эквивалентный оригиналу. Как вы думаете, какие элементы презентации не могут быть сохранены в *Web*-документе?

Глава 12

ЯЗЫК HTML

Изучив данную главу, вы сможете:

- создавать *Web*-страницы по синтаксическим правилам HTML;
- форматировать контент *Web*-документов с помощью средств HTML;
- составлять и организовывать списки с помощью средств HTML;
- создавать и применять внутренние и внешние ссылки в документах HTML;
- вставлять изображения в документы HTML;
- создавать и редактировать таблицы в документах HTML;
- применять таблицы для размещения элементов на странице HTML.

12.1. Общая структура документа HTML

12.1.1. О документах HTML

Как было отмечено ранее, самыми распространенными *Web*-документами являются документы HTML. Наиболее важные характеристики формата HTML – это независимость платформы (один и тот же документ HTML выводится одинаково на разных компьютерах), ссылки, структурированное форматирование. Ссылки значительно упрощают навигацию в большом документе и/или по многим документам сайта. В качестве ссылок может быть использовано слово, фраза, изображение или любой другой элемент *Web*-страницы.

Первая версия HTML была выпущена в 1989 году (созданная Тимоти Бернерс-Ли*). Несмотря на то, что некоторые программы навигации распознают не все ключевые слова языка HTML, он продолжает оставаться одним из самых распространенных и современных языков для создания *Web*-страниц. Файлы HTML имеют текстовый формат ASCII, следовательно, могут быть созданы в любом текстовом редакторе. Отметим, что простую *Web*-страницу легко спроектировать с помощью HTML.

Под **элементом** некоторого документа HTML будем понимать любую компоненту его структуры: таблицу, абзац, список, заголовок, кнопку, текстовое поле, изображение и др.

Для разметки элементов файла HTML применяются ключевые поля, называемые **дескрипторами** или **тэгами** (tags). Любой тэг заключается в символы < и >.

Тэги, обычно, при включении некоторого элемента формируют пару. Тэг конца элемента выглядит так же, как и тэг начала, только перед ключевым словом добавляется символ / (slash). Тэг начала может содержать один или несколько атрибутов, специфицируемых в следующей форме:

<Дескриптор a_1=v_1 a_2=v_2 . . . a_i=v_i >, где v_i – значение атрибута a_i.

Заметим, язык HTML не делает отличия между строчными и прописными буквами.

* *Сэр Тимои (Тим) Бернерс-Ли* (англ. Sir Timothy John (Tim) Berners-Lee; род. 8 июня 1955 г.) – британский ученый, изобретатель Всемирной Паутины (WWW) и действующий глава Консорциума Всемирной Паутины (W3C), организации, „опекающей" *Web*-стандарты.

12.1.2. Общая структура документа HTML

Документ HTML имеет, как правило, следующую структуру:

<HTML>	Начало документа
<HEAD> <TITLE> Заголовок документа </TITLE> </HEAD>	Раздел заголовка документа
<BODY> Тело документа </BODY>	Раздел содержания документа
</HTML>	Конец документа

• Текст, записанный между дескрипторами <TITLE> и </TITLE>, выводится на панели названия окна программы навигации.

• Дескрипторы <BODY> и </BODY> содержат контент документа HTML, который выводится в окне приложения навигации и который виден пользователю.

Отметим несколько атрибутов тэга <BODY>:

– *bgcolor* = "#$n_1n_2n_3n_4n_5n_6$", где $\overline{n_1n_2}$, $\overline{n_3n_4}$, $\overline{n_5n_6}$ являются шестнадцатеричными числами, определяющими интенсивность красного, зеленого и, соответственно, синего цветов для фона страницы. Цвет можно задать и его названием (red, blue, black, yellow, green, cyan, purple, white, gray и др.);
– *text* = "#$n_1n_2n_3n_4n_5n_6$" определяет цвет текста;
– *background* = "URL", где URL – адрес и имя графического файла, чье изображение будет фоном документа;
– *leftmargin* = "*n*", где *n* – натуральное число, определяющее в пикселях расстояние между левой границей окна программы навигации и левой границей содержимого страницы;
– *topmargin* = "*n*", где *n* – натуральное число, определяющее в пикселях расстояние между верхней границей окна программы навигации и верхней границей содержимого страницы.

Пример: Следующий файл HTML, интерпретируемый программой навигации, выведет *Web*-страницу с фоном цвета *aqua* и текстом цвета #FF2233.

```
<HTML>
<Head> <Title> Пример </Title> </Head>
<Body leftmargin="140" topmargin="200" bgcolor="aqua" text="#FF2233">
Привет! Это пример документа .html
</Body>
</HTML>
```

Вопросы и упражнения

❶ Объясните назначение ссылок в некотором документе HTML.
❷ Какой редактор текста можно использовать для создания некоторого файла HTML?
❸ Что имеется ввиду под понятием *элемент документа HTML*?
❹ Какой смысл имеют дескрипторы в документе HTML?
❺ Какие символы используются для выделения дескрипторов?

❻ Чем отличаются тэги начала от тэгов конца элемента?

❼ Какую общую структуру имеет документ HTML?

❽ Для чего и как применяются тэги <TITLE> и </TITLE> ?

❾ Для чего и как применяются тэги <BODY> и </BODY>? Приведите примеры атрибутов тэга <BODY>.

❿ Создайте *Web*-страницу с желтым фоном и зеленым текстом. Содержимое страницы должно выводиться на расстоянии 110 пикселей от левой и 105 пикселей от верхней границы окна программы навигации.

⓫ Создайте *Web*-страницу с синим фоном и белым текстом. Содержимое страницы должно выводиться на расстояние 10 пикселей от левой и 15 пикселей от верхней границы окна программы навигации.

⓬ Что выведет следующий документ HTML?

```
<HTML>
<Head> Что такое файл .html? <Title> ЗАДАЧА </Title> </Head>
<Body>
Какова структура документа .html?
</Body>
</HTML>
```

⓭ Создайте *Web*-страницу, которая выводит на панели названия окна программы навигации текст ПЕРВЫЙ ДОКУМЕНТ HTML, а в качестве фона использует некоторое изображение.

12.2. Форматирование текста

• В HTML используются 6 **уровней заголовка**, специфицируемых тэгами <Hn> и </Hn>, где n может принимать значения от 1 до 6. Каждая из этих пар тэгов задает определенный уровень заголовка. Первый (1) уровень самый крупный, выразительный, а уровню 6 соответствуют самые маленькие и тонкие символы.

Тэги заголовка могут включать атрибут *align* = "*tip*", где *tip* одно из слов *center*, *left*, *right* для определения способа выравнивания текста.

• В отличие от обычного редактора текста, HTML не учитывает длину текста, символ абзаца (Enter) и несколько вподряд идущих пробелов, которые автоматически заменяются одним пробелом. Программа навигации выводит текст с новой строки только если перед этим текстом встречается один из тэгов <P>,
, <PRE>, <DIV> или <CENTER>.

a) Таким образом, **абзацем** будет считаться текст, заключенный между тэгами <P> и </P>. Этот тэг тоже может включать атрибут *align*.

b) Тэг
 осуществляет **переход на новую строку** в текущем абзаце.

c) Текст, заключенный между тэгами <PRE>, </PRE>, будет выводиться в таком формате, в котором он написан, то есть он является **форматированным текстом**.

d) Часто **текстовой блок** форматируется тэгами <DIV>, </DIV>. Тэг <DIV> может иметь атрибуты *align* и *nowrap* (запрещающий программе навигации разрывать строки). Блок <DIV>...</DIV> может содержать внутри другие блоки. В таком случае, выравнивание, указанное атрибутом *align* внешнего блока, распространяется и на все его внутренние блоки.

e) Элементы, заключенные в тэги <CENTER> и </CENTER>, выравниваются по центру.

• Текстовый блок, заключенный между тэгами <NOBR> и </NOBR>, будет выведен в одной строке. Если пользователь уменьшит размеры окна программы навигации, то может случиться, что текст будет обрезан.

• **Горизонтальные линии** применяются для разбиения страницы на логические разделы. В зависимости от необходимости, тэг <HR>, добавляющий горизонтальную линию, может включать атрибуты: *align* (устанавливает способ выравнивания линии); *size* (уточняет толщину линии в пикселях); *width* (уточняет длину линии в пикселях или процентах от ширины окна программы навигации); *color* (уточняет цвет линии); *noshade* (определяет линию без каких-либо теней).

• **Шрифт** текста задается тэгами <FONT> и </FONT>. Его характеристики определяются атрибутами: *face* (задает набор символов и может иметь одно или несколько значений констант, разделяемых запятыми, например: *arial*, *serif*, *cursive*, *monospace*, *fantasy* и др.); *color* (задает цвет символов); *size* (определяет величину символов и может иметь значения 1, 2, ..., 7 или +1, +2, ... (увеличивая величину символов по отношению к текущей), или –1, –2, ... (уменьшая величину символов по отношению к текущей)); *weight* (задает толщину символов, допустимые значения равны 100, 200, ..., 900).

Пример 1 (*рис. 12.1*):

```
<HTML>
<Head>
   <Title> Форматированный текст </Title>
   <H1 align="center"> Кишинэу</H1>
</Head>
<Body leftmargin="20" topmargin="20" bgcolor="lightgreen" text="blue">
   <HR size="5" width="70%" color="black">
   <Font color="purple">
   <H2>Районы Кишинэу:</H2>
      <H3>
      <Pre>
   Центр
      Буюкань
         Рышкань
            Ботаника
               Чокана
      </Pre>
   </H3>
   </Font>
</Body>
</HTML>
```

• Для выделения некоторого фрагмента текста применяются **стили**. Стили делятся на две категории:

a) физические стили, определяющие внешний вид символов;

b) логические стили, выделяющие текст в зависимости от его значения.

Физические стили выводят текст одинаково во всех программах навигации. Для их определения применяют тэги:

<B>, </B> – толщина символов текста увеличена (bold – полужирный);

<I>, </I> – символы текста наклонены (italic – курсив);

<U>, </U> – символы текста подчеркнуты;

<TT>, </TT> – текст, заданный моноширинным шрифтом;
<BIG>, </BIG> – текст, размер символов которого на единицу больше текущего;
<SMALL>, </SMALL> – текст, размер символов которого на единицу меньше текущего;
_, – символы текста выделены как нижний индекс;
[,] – символы текста выделены как верхний индекс;
<S>, </S> – текст, символы которого зачеркнуты линией.
Тэги физических стилей могут быть вложенными.

Пример 2 *(рис. 12.2)*:

```
<HTML>
<Head>
<Title> Пример 2 </Title>
<H2 align=”center”> Стили HTML</H2>
</Head>
<Body>
<H3>Физические стили:</H3>
<Pre>
   Обычный текст
   <B>Текст с утолщенными символами </B>
   <I>Текст с наклонными символами </I>
   <U> Текст с подчеркнутыми символами </U>
   <TT> Текст, заданный моноширинным шрифтом </TT>
   <big>Текст с увеличенными символами</big>
   <small>Текст с уменьшенными символами </small>
   Y = X<sup>2</sup>
   M = (m<sub>1</sub>, m<sub>2</sub>)
   <s>Перечеркнутый текст </s>
   <B><I>Полужирный с наклоном текст </I></B>
</Pre>
</Body>
</HTML>
```

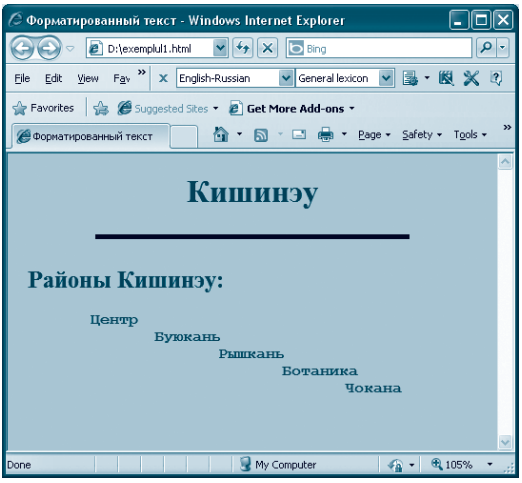


Рис. 12.1

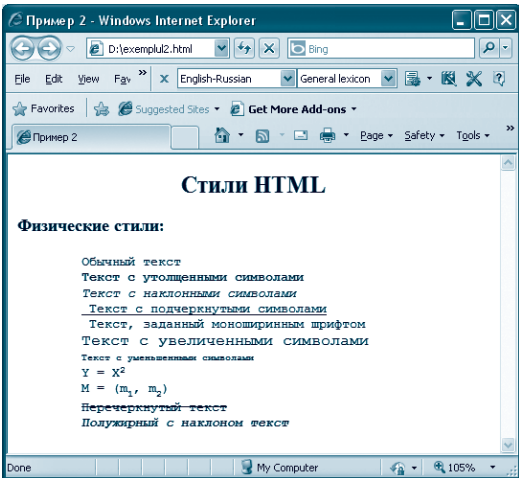


Рис. 12.2

Логические стили зависят от конфигурации программы навигации и предназначены для выделения текста, имеющего разное назначение в контенте *Web*-страницы. Для них используются следующие тэги:

`<EM>`, `</EM>` – выделенный текст (обычно, курсивом);
`<STRONG>`, `</STRONG>` – важный текст (обычно, полужирный);
`<KBD>`, `</KBD>` – введенный с клавиатуры текст (обычно, моноширинный шрифт);
`<CODE>`, `</CODE>` – текст, содержащий код компьютерной программы (обычно, моноширинный шрифт);
`<DFN>`, `</DFN>` – текст-определение (обычно, курсивом);
`<VAR>`, `</VAR>` – идентификатор переменной;
`<ADDRESS>`, `</ADDRESS>` – текст-адрес;
`<CITE>`, `</CITE>` – текст-цитата.

• Одной из возможностей **включения диакритических знаков** румынского языка в документ HTML является прямое кодирование с помощью &-последовательностей (escape-последовательности). Некоторые программы навигации распознают &-последовательности только, если в их конце записан символ **;**.

Буква	**Ă**	**ă**	**Â**	**â**	**Î**	**î**	**Ş**	**ş**	**Ţ**	**ţ**
Код	&#258	&#259	&Acirc	&acirc	&Icirc	&icirc	&#350	&#351	&#354	&#355

&-последовательности применяются также для включения разных специальных символов:

Символ	**&**	**<**	**>**	**“**	**®**	**©**	**пробел**	**§**
Код	&amp	&lt	&gt	&quot	&reg	&copy		&sect

Символ	**±**	**×**	**¦**	**1/2**	**1/4**	**3/4**	**°**
Код	&plusmn	&times	&brvbar	&frac12	&frac14	&frac34	&deg

Пример (*рис. 12.3*):

```
<HTML>
<Head><Title> Пример 3 </Title><Body>
<H4>
Acest capitol v&#259 &icircnva&#355&#259 s&#259 crea&#355i pagini Web
</H4>
</Body>
</HTML>
```

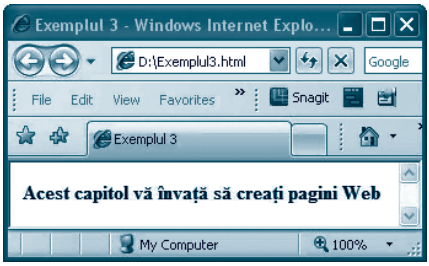


Рис. 12.3

Вопросы и упражнения

❶ Каким тэгом можно указать уровень заголовка?
❷ Сколько уровней заголовка можно использовать в HTML?
❸ Какие атрибуты можно применять в тэге заголовка?
❹ Какими тэгами можно обозначить абзац?
❺ Какие тэги применяют для перехода в новую строку?
❻ Как можно вставить в текст горизонтальную линию?
❼ Какими тэгами можно специфицировать необходимый шрифт?
❽ Перечислите атрибуты тэга <FONT>.
❾ Что такое стиль и для чего применяют стили?
❿ Чем отличаются между собой логические и физические стили?
⓫ Назовите тэги для выделения стилей: a) логических; b) физических.
⓬ Создайте *Web*-страницу магазина так, чтобы название магазина выводилось как заголовок первого уровня, черным цветом. Ниже должны выводиться название категории товаров и названия самих товаров. Название категории – заголовок уровня 3, а названия товаров – форматированный текст. Название магазина отделяется от названия категории товаров горизонтальной линией. Толщина линии – 4, цвет – синий, выравнивание по левой границе, длина – 80% от ширины окна программы навигации.
⓭ Создайте *Web*-страницу, которая выводит на экран две теоремы конгруэнтности треугольников. Слово *Теорема* необходимо написать как заголовок уровня 3, черным цветом, полужирными и подчеркнутыми буквами. Текст обеих теорем должен быть записан курсивом, а между теоремами проведите горизонтальную линию красного цвета, толщиной 2, выровненную по центру, длина которой – 70% от ширины окна программы навигации.
⓮ Создайте *Web*-страницу, выводящую по центру страницы формулу:
a) $(a + b)^2 = a^2 + 2ab + b^2$;
b) $(a_1 + a_2 + a_3)^2 = a_1^2 + a_2^2 + a_3^2 + 2a_1a_2 + 2\,a_1a_3 + 2\,a_2a_3$;
c) $|\,a + b| < |a| + |b|$, для любых двух отличных от нуля чисел a и b.
⓯ Создайте документ HTML выводящий *Web*-страницу, подобную изображенной на рисунке *12.4 a, b*:

a) *b)*

Рис. 12.4

⓰ Создайте *Web*-страницу, на которой написано стихотворение. Название стихотворения должно быть зеленого цвета, выровненное по центру, на желтом фоне, записанное как заголовок уровня 3. Между названием и самим стихотворением проведите горизонтальную линию толщиной 5, занимающую 50% от ширины окна программы навигации. Символы текста должны быть написаны полужирным курсивом.

12.3. Списки

Для организации информации некоторого документа html в структурированном виде можно использовать списки. Списки могут содержать:

a) *Неупорядоченную информацию*;

b) *Упорядоченную информацию*;

c) *Определения*.

Неупорядоченные списки служат для того, чтобы перечислять информацию без внутренней иерархической связи.

- Для обозначения начала и конца такого списка используются тэги <UL> и </UL> (*unordered list* – „неупорядоченный список").
- Перед каждым включаемым в список элементом записывается тэг <LI> (*list item* – „элемент списка").

В каждом из тэгов <UL>, <LI> можно использовать атрибут *type*, который принимает одно из трех значений: *Circle*, *Square*, *Disc*. Это значение определяет маркер перед элементом списка.

Упорядоченные списки (иногда называемые *нумерованными списками*) заключаются в тэги <OL> и </OL> (*ordered list* – „упорядоченный список").

Как и в случае неупорядоченного списка, каждый элемент списка начинается с тэга <LI>. Атрибут *type* тэга <OL> может принимать одно из значений: A, a, I, i, 1.

Если атрибут *type* не специфицирован, то элементы списка нумеруются арабскими цифрами.

Списки определений обычно применяются для организации глоссария. Термины глоссария включаются в список в алфавитном порядке, каждый со своим определением.

• Списки определений заключаются в тэги <DL> и </DL> (*definition list* – „список определений").

• Каждый термин начинается тэгом <DT> (*definition term* – „определяемый термин"), а само определение (описание) термина – тэгом <DD> (*definition description* – „описание определения").

Замечания:

1. Списки могут быть вложенными.
2. Внутри списка можно использовать тэги <P> и
, а также – тэги форматирования текста.
3. Упорядоченный список может прерываться некоторым текстом. В этом случае используют два набора тэгов <OL>: один для первой части списка (до текста) и второй – для продолжения списка (после текста). Для второй части списка в тэг <OL> необходимо включить атрибут *start* со значением, указывающим, с какой величины возобновляется нумерование.

Пример 1 (*рис. 12.5*):

```
<HTML>
<Head> <Title> Неупорядоченные списки </Title> </Head>
<Body>
   <H4> Некоторые области математики </H4>
   <UL>
   <LI> Алгебра
```

```
    <LI> Геометрия
    <LI> Теория дифференциальных уравнений
    <LI> Математический анализ
    </UL>
    <H4> Некоторые области информатики </H4>
    <UL TYPE=square>
    <LI> Теория алгоритмов
    <LI> Web-программирование
    <LI> Теория баз данных
    <LI> Криптография
    </UL>
</Body>
</HTML>
```

Пример 2 (*рис. 12.6*):

```
<HTML>
<Head> <Title> Упорядоченные списки </Title> </Head>
<Body>
    <H4> Некоторые предметы, изучаемые в X классе </H4>
    <OL>
    <LI> Румынский язык
    <LI> Математика
    <LI> Информатика
    </OL>
    <H4> Некоторые предметы, изучаемые в XII классе </H4>
    <OL TYPE=A>
    <LI> Информатика
    <LI> Биология
    <LI> Математика
    </OL>
</Body>
</HTML>
```

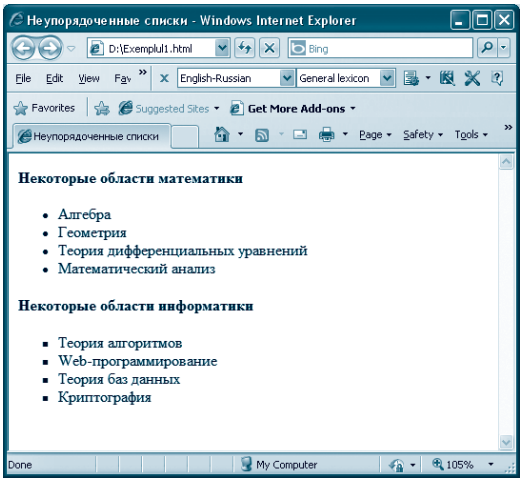


Рис. 12.5

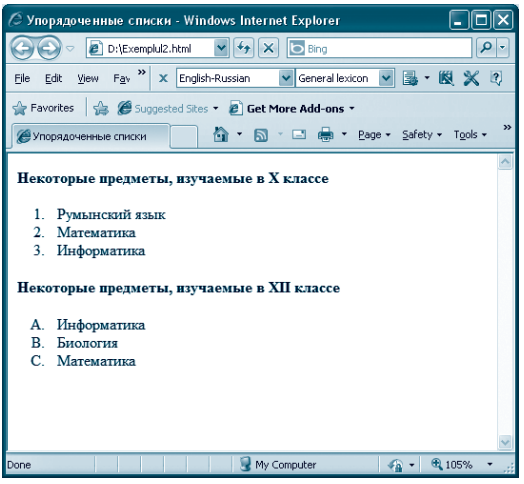


Рис. 12.6

Пример 3 (*рис. 12.7*):

```
<HTML>
<Head> <Title> Список определений </Title> </Head>
<Body>
   <H4> Некоторые понятия информатики </H4>
   <DL>
   <DT> Константа <DD> - величина, чье значение не может меняться
   в процессе исполнения алгоритма
   <DT> Переменная <DD> - величина, чье значение может меняться
   в процессе исполнения алгоритма
   <DT> Идентификатор <DD> - последовательность символов, обычно начинающаяся
   с буквы, за которой могут следовать буквы или цифры, служащие именем
   для констант, переменных, типов, подпрограмм.
   </DL>
   </Body>
</HTML>
```

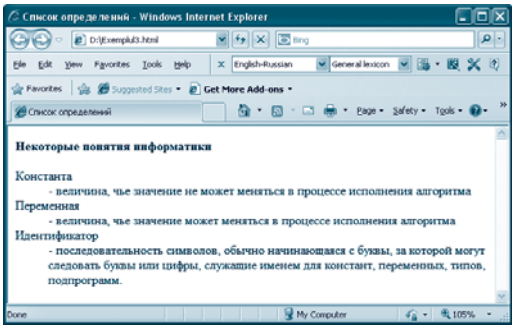


Рис. 12.7

Вопросы и упражнения

❶ Для каких целей используются списки?

❷ Какие типы списков вы знаете?

❸ Назовите тэги для каждого типа списка.

❹ Каким образом специфицируются элементы списков?

❺ Каковы атрибуты этих тэгов и какие значения они могут принимать?

❻ Создайте *Web*-страницу, выводящую заголовок „Любимые предметы" на голубом фоне, уровень заголовка – 4. Ниже заголовка организуйте неупорядоченный список этих предметов. В качестве маркера используйте *квадрат*.

❼ Создайте *Web*-страницу, выводящую заголовок „Предметы, изучаемые в первом семестре 12-го класса", на голубом фоне, уровень заголовка – 3. Ниже заголовка организуйте неупорядоченный список этих предметов. В качестве маркера используйте *круг*. После списка выведите две пустые строки, а затем – неупорядоченный список предметов, по которым сдаются экзамены на степень бакалавра.

❽ Создайте *Web*-страницу, выводящую заголовок „Список учеников 12-го класса", на фоне цвета *aqua*, уровень заголовка – 5, выравнивание – по центру. Ниже заголовка организуйте неупорядоченный список учеников вашего класса. После списка выведите две пустые строки, а затем – неупорядоченный список учеников, которые закончили 11-й класс с общей средней оценкой больше 8.

❾ Создайте *Web*-страницу, выводящую заголовок „ОГЛАВЛЕНИЕ" уровня 3, после которого следуют названия глав некоторого учебника и тем, в них входящих. Используйте упорядоченные и вложенные списки.

❿ Создайте *Web*-страницу, выводящую 2 упорядоченных списка: заголовок первого списка „Пищевые продукты", уровень заголовка – 2, элементы пронумерованы; заголовок второго

списка – „Промышленные товары", уровень заголовка – 3, в качестве маркера элементов используются буквы.

⓫ Создайте *Web*-страницу, выводящую упорядоченный список кулинарных продуктов. Список должен прерываться следующим курсивным текстом: *„Эти кулинарные продукты мне нравятся, а ниже следуют продукты, которые мне не нравятся:"*. Заголовок списка должен быть уровня 4, синего цвета.

⓬ Создайте документ HTML выводящий *Web*-страницу, изображенную на *рисунке 12.8*.

⓭ Создайте *Web*-страницу, выводящую определения 5 геометрических понятий, следующих после заголовка „Геометрические понятия" уровня 3, серого цвета, выровненного по правой границе и отделенного от самих понятий горизонтальной линией черного цвета, толщиной 3, выровненной по центру.

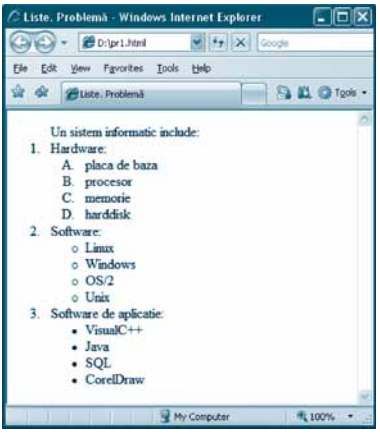


Рис. 12.8

12.4. Ссылки

В параграфе 12.1 было отмечено, что ссылки (называемые иногда связями, гиперссылками или линками (*link*)) представляют собой, пожалуй, самую важную характеристику языка HTML. С помощью ссылок можно очень быстро попасть на другой фрагмент текущего документа или на другой документ (на том же или другом сервере). Ссылки – это *активные зоны Web*-страницы, в том смысле, что при нажатии на них программа навигации обновляет страницу.

• Для выделения ссылки используются тэги <A> (от „ancor" - якорь) и </A>. Атрибут *href* тэга <A> обязателен. Значением этого атрибута является имя файла HTML, с которым устанавливается связь. Записывается это имя между символами " и ". Текст или изображение, заключенные между тэгами <A> и </A>, являются активными зонами.

Так, ссылка, объявленная следующим образом:

<A href=URL> текст или *изображение</A>*,

где URL (*Uniform Resourse Locator* – уникальный идентификатор источника) является именем файла-назначения.

Замечание: В данном параграфе рассмотрим только ссылки, активными зонами которых являются тексты.

Вообще говоря, ссылка может:

– связывать текущий документ с документом из той же папки;
– связывать текущий документ с документом на том же локальном диске;
– связывать текущий документ с другим сайтом;
– связывать текущий документ с другой областью того же документа;
– связывать текущий документ с некоторой областью другого документа;
– запускать на выполнение приложение рассылки электронных сообщений;
– связывать текущий документ с некоторым файлом произвольного формата, с целью создания на диске копии этого файла.

а) Если документ, на который указывает ссылка, находится в той же папке, то URL является его именем.

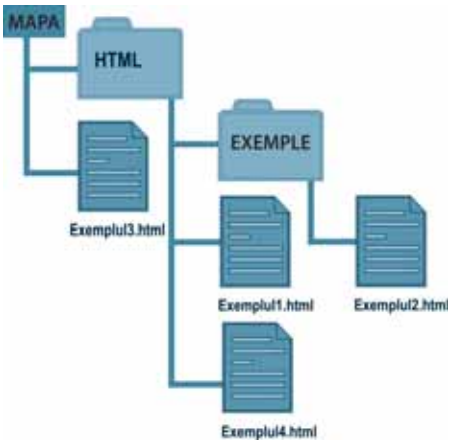
b) Если документ, на который указывает ссылка, находится на том же локальном диске, но в другой папке, то используется относительный адрес.

c) Если документ, на который указывает ссылка, представляет собой другой сайт, то указывается его адрес, записанный между символами ″ и ″.

Пример 1 (*рис. 12.9*):

Представим содержание файла *exemplul4.html*, включающего ссылки на:

- файл *exemplul1.html* (находящийся в той же директории);
- файл *exemplul2.html* (находящийся в субдиректории EXEMPLE текущей директории);
- файл *exemplul3.html* (находящийся в директории MAPA, следовательно, на иерархически более высоком уровне);
- сайт Тираспольского государственного университета: http://www.ust.md.




```
<HTML>
<Head> <Title> Ссылки</Title> </Head>
<Body>
   <H4> <a href = ”exemplul1.html”> Переход к примеру 1 </a>
   <p> <a href = ”EXEMPLE/exemplul2.html”> Переход к примеру 2 </a> </p>
   <p> <a href = ”../exemplul3.html”> Переход к примеру 3 </a> </p>
   <p> <a href = ”http://www.ust.md”> Тираспольский государственный
   университет </a> </p>
   </H4>
</Body>
</HTML>
```

d) Ссылка на некоторый фрагмент текущего документа называется *именованным анкером*. В этом случае URL это имя, присвоенное фрагменту текста, которое записывается после символа #. Для присвоения имени фрагменту текста необходимо создать анкер следующим образом:

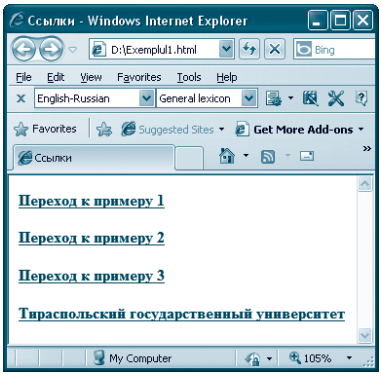


Рис. 12.9

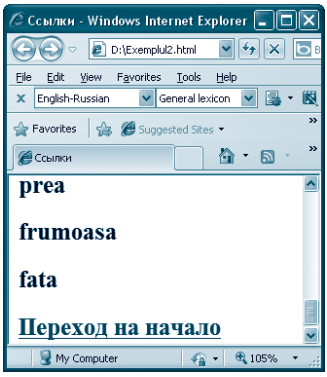


Рис. 12.10

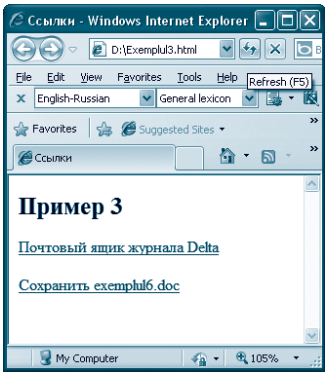


Рис. 12.11

<A *name = имя фрагмента*> </A>, где *имя фрагмента* – это последовательность символов, записанная между ″ и ″.

e) URL фрагмента из другого документа состоит из имени файла, содержащего этот документ, и имени фрагмента, записанного после символа #.

Например, URL фрагмента с именем *capitolul2* файла *exemplul1.html* из текущего каталога выглядит так: `″exemplul1.html#capitolul2″`.

Пример 2 (*рис. 12.10*):

```
<HTML>
<Head> <Title> Ссылки</Title> </Head>
<Body>
<H2>
<a name=”начало”></a>
   <P>A</P> <P>fost</P> <P>odata</P> <P>ca-n</P> <P>povesti</P>
   <P>A</P> <P>fost</P> <P>ca</P> <P>niciodata</P>
   <P>Din</P> <P>rude</P> <P>mari</P> <P>imparatesti</P>
   <P>O</P> <P>prea</P> <P>frumoasa</P> <P>fata</P>
<a href=”#начало”> Переход на начало </a>
</H2>
</Body>
</HTML>
```

f) Для запуска на выполнение некоторого приложения, для рассылки электронных сообщений (предполагается, что такое приложение установлено на компьютере пользователя), применяется следующий анкер:

<A *href = ″mailto: допустимый адрес e-mail″*>.

g) Для создания ссылки на файл произвольного типа, с целью создания его копии на диске или для запуска приложения, его интерпретирующего, используется следующий анкер:

<A href=URL> текст</A>,

где URL – имя файла, а *текст* – активная зона.

Замечание: Тэгу <A> можно задать атрибут *title*, используемый для предоставления дополнительной информации о данной ссылке в момент, когда указатель мыши располагается над активной зоной. Значение атрибута *title* записывается в кавычках ″ и ″.

Пример 3 (*рис. 12.11*):

```
<HTML>
<Head> <Title> Ссылки</Title>
   <H2>Пример 3</H2>
</Head>
<Body>
<a href=mailto: delta_mi@mail.md> Почтовый ящик журнала Delta </a>
<p><a href=”exemplul6.doc” title=”Сохранение на диске файла
exemplul6.doc ”> Сохранить exemplul6.doc </p> </a>
</Body>
</HTML>
```

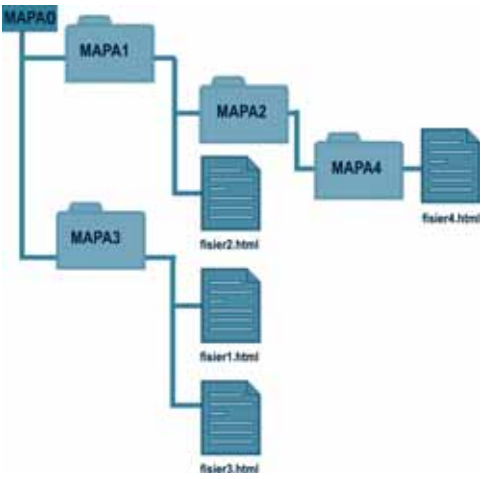
При нажатии на ссылку `Сохранить exemplul6.doc`, появится окно, изображенное на *рисунке 12.12*, позволяющее пользователю либо сохранить на диске копию файла `exemplul6.doc`, либо открыть его с помощью приложения Microsoft Word.

Рис. 12.12

Вопросы и упражнения

❶ Какое значение для *Web*-страниц имеют ссылки?
❷ Какие тэги применяются для объявления ссылок?
❸ Назовите обязательный атрибут для тэга <A>. Какое значение он может иметь?
❹ Что такое активная зона?
❺ Какую общую форму имеет объявление ссылки?
❻ Ссылки на какие элементы *Web*-страницы можно создавать?
❼ Где может находиться документ, на который можно перейти по ссылке?
❽ Что такое *именной анкер*?
❾ Как можно присвоить имя некоторому фрагменту документа HTML?
❿ Объясните, в каких случаях при написании адреса URL используется знак #.
⓫ Как создается связь с некоторым фрагментом текущего документа?
⓬ Из чего состоит адрес URL фрагмента другого документа?
⓭ Как из документа HTML можно запустить на выполнение приложение для рассылки электронных сообщений?
⓮ Создайте *Web*-страницу, которая выведет список изучаемых предметов. Каждый элемент списка будет ссылкой на другой файл (находящийся в той же, где и сама страница, директории), содержащий информацию о соответствующем предмете.
⓯ Создайте *Web*-страницу, которая выведет упорядоченный список с названиями некоторых ВУЗов Республики Молдова. Каждый элемент списка будет ссылкой на сайт соответствующего ВУЗа.
⓰ Создайте *Web*-страницу, которая выведет список лицеев. Каждый элемент списка будет ссылкой на файл (сайт) соответствующего лицея. Созданная *Web*-страница будет сохранена в каталоге LICEE, а каждый файл, на который указывает ссылка, будет сохранен в подкаталоге с названием лицея. Все подкаталоги находятся в каталоге LICEE.
⓱ Проанализируйте изображение из примера 1. Какой адрес URL имеет ссылка на файл:
a) exemplul1.html если ссылка находится в файле exemplul3.html;
b) exemplul2.html если ссылка находится в файле exemplul1.html;
c) exemplul4.html если ссылка находится в файле exemplul2.html;
d) exemplul3.html если ссылка находится в файле exemplul2.html?
⓲ Создайте *Web*-страницу, которая выведет список наименований нескольких книг. Каждое наименование будет ссылкой на аннотацию соответствующей книги, представленную на этой же *Web*-странице.

⓳ Проанализируйте рисунок. Какой адрес URL имеет ссылка на файл:
a) f1.html если ссылка находится в f4.html;
b) f2.html если ссылка находится в f3.html;
c) f4.html если ссылка находится в f2.html;
d) f4.html если ссылка находится в f3.html?



⓴ Проанализируйте рисунок. В каком каталоге находится документ HTML содержащий ссылку, если ее адрес URL:
a) "Mapa4\fisier4.html";
b) "Mapa1\Mapa2\Mapa4\fisier4.html";
c) "..\ Mapa1\Mapa2\Mapa4\fisier4.html";
d) "..\ ..\ Mapa3\fisier3.html"?

㉑ Создайте две *Web*-страницы таким образом, чтобы из первой можно было перейти на некоторые фрагменты во второй странице, а из второй – на некоторые фрагменты первой страницы.

㉒ Создайте *Web*-страницу, которая выведет список со ссылками на несколько файлов разного типа так, чтобы пользователь смог их скопировать на диск или открыть для просмотра.

12.5. Изображения

Помимо своей информационной нагрузки, изображения придают привлекательность *Web*-странице. Все же их использование в документе HTML должно быть рациональным, так как для загрузки, декодирования и вывода каждого изображения требуется время.

Приведем некоторые форматы файлов-изображений, наиболее часто используемые в *Web*-дизайне:

– GIF (*Grafics Interchange Format*) – файлы с расширением .gif;
– JPEG (*Join Photographic Expert Group*) – файлы с расширением .jpeg sau .jpg;
– BMP (*BitMap*) – файлы с расширением .bmp.

Для вставки изображения используется тэг `<IMG>`, с обязательным атрибутом SRC, значением которого является адрес URL изображения:

<IMG SRC = URL>

Синтаксис для URL совпадает с тем, который был рассмотрен в случае создания ссылок (см. параграф 12.4).

Кроме атрибута SRC тэг <IMG> может иметь следующие атрибуты:

• ALIGN, применяется для выравнивания изображения. Может иметь одно из значений: *top* (выравнивание по верху – верхняя часть изображения выравнивается по верхней части текста, предшествующего изображению), *bottom* (выравнивание по основанию – нижняя часть изображения выравнивается по нижней строке текста), *middle* (выравнивание по середине – середина изображения выравнивается по нижней строке текста, предшествующего изображению), *left* (выравнивание по левой границе окна – текст и остальные элементы будут размещены справа), *right* (выравнивание по правой границе окна – текст и остальные элементы будут размещены слева);

• ALT, используется для вывода *пояснительного текста* на месте изображения (в случае, если программа навигации не может загрузить само изображение – по опции, установленной пользователем, либо по другой причине). Значение атрибута (т.е. пояснительный текст) записывается в кавычках;

• WIDTH и HEIGHT, используются для спецификации размеров изображения, в пикселях;

• BORDER, используется для создания вокруг изображения рамки, толщина которой (в пикселях) равна значению атрибута;

• HSPACE и VSPACE, используются для уточнения расстояния, в пикселях, по горизонтали и по вертикали, между изображением и другими элементами страницы.

Пример 1 (*рис. 12.13*):

```
<HTML>
<Head>
   <Title>Пример 1. Изображение</Title>
</Head>
<Body>
   <H3 align=center>Тираспольский государственный университет </H3>
   <P>
   Тираспольский государственный университет (ТГУ) является первым высшим
   учебным заведением Республики Молдова, созданным 1 октября 1930 года.

   <IMG SRC="ust.jpg" align=right hspace=10 vspace=10>

   С 1992 года ТГУ находится в Кишинэу. ТГУ подготовил более 80 000 специалистов
   с высшим образованием. Многие из них стали знаменитостями в различных сферах
   деятельности – в областях среднего и высшего образования, учеными педагогики
   и национальной экономики. Большинство дидактических кадров обладают
   учеными и дидактическими степенями. ТГУ сотрудничает с различными
   учреждениями страны и зарубежья – Румынии, России, Украины, США, Германии,
   Италии, Португалии и др.

</Body>
</HTML>
```

Изображения могут использоваться и в качестве ссылок.

Пример 2 (*рис. 12.14*):

Следующая *Web*-страница выводит 4 изображения-ссылки. При нажатии на одну из ссылок выводится соответствующее изображение в его реальных размерах.

Эта техника используется при создании так называемых „фотогалерей” (ссылки могут быть значительно меньших размеров, чем указываемые ими реальные файлы-изображения). Файлы-изображения находятся в каталоге DESENE текущего каталога.

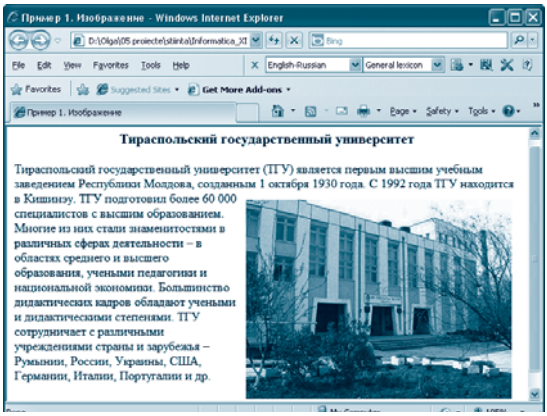


Рис. 12.13

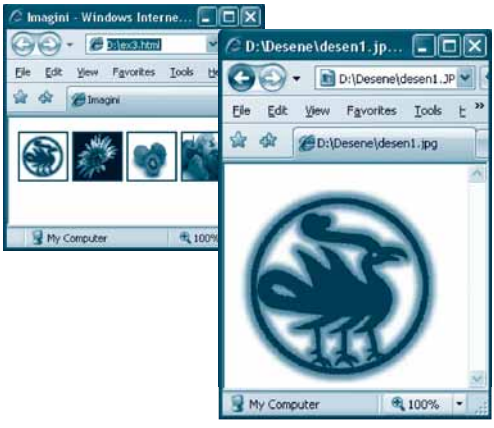


Рис. 12.14

```
<HTML>
<head> <Title>Imagini</Title> </head>
<Body>
<A HREF=”Desene/desen1.jpg”><img src=Desene/desen1.jpg width=
=50 height=50></A>
<A HREF=”Desene/desen2.jpg”><img src=Desene/desen2.jpg width=
=50 height=50></A>
<A HREF=”Desene/desen3.jpg”><img src=Desene/desen3.jpg width=
=50 height=50></A>
<A HREF=”Desene/desen4.jpg”><img src=Desene/desen4.jpg width=5
=0 height=50></A>
</Body>
</HTML>
```

Вопросы и упражнения

❶ Какие форматы файлов-изображений воспринимаются программой *Web*-навигации?

❷ Какой тэг используется для вставки изображения в документ HTML? Какой обязательный атрибут имеет этот тэг?

❸ Какие необязательные атрибуты может иметь тэг для вставки изображений в документ HTML и каковы их возможные значения?

❹ Как можно создать ссылку-изображение?

❺ Создайте *Web*-страницу, выводящую резюме с фотографией его автора, выровненной по центру страницы.

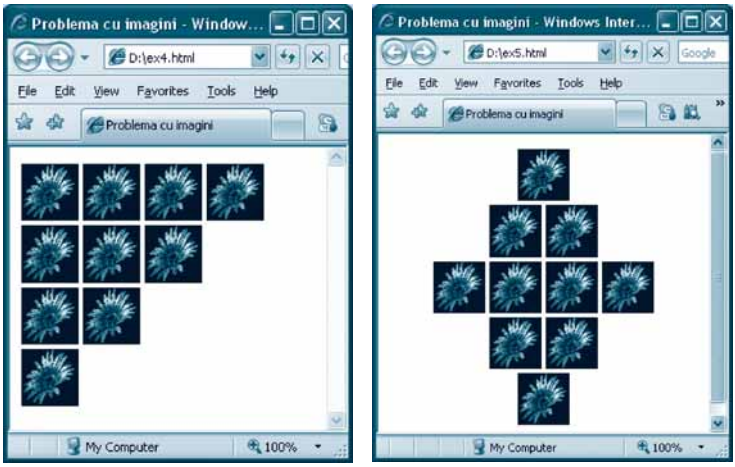
❻ Создайте *Web*-страницу, выводящую несколько изображений-ссылок размерами 5 см x 5 см. При нажатии на каждую ссылку-изображение программа навигации выводит это изображение размером 15 см x 15 см.

❼ Создайте *Web*-страницу, выводящую текст о местности, в которой вы родились. Заголовок – название местности – имеет уровень 5, черного цвета, на сером фоне. Изображение местности, обведенное рамкой, выводится справа от текста.

❽ Создайте *Web*-страницу, выводящую график некоторой функции.
Указание: Построить график можно с помощью программы Microsoft Excel.

❾ Создайте *Web*-страницу, выводящую органиграмму.

❿ Создайте документ HTML, выводящий *Web*-страницу, похожую на приведенную на *рисунке 12.15 a, b* (можно использовать другие изображения):



a) *b)*

Рис. 12.15

12.6. Таблицы

Чаще всего таблицы используют для представления данных в структурированном виде. Вместе с этим, благодаря тому факту, что каждая ячейка таблицы может иметь собственные цвета, стили, эффекты для текста и фона, таблицы успешно используют для решения некоторых дизайнерских проблем.

Таблица обычно имеет:

a) *название* для спецификации названия таблицы;

b) *строки* и *столбцы*;

c) *заголовок* (table header) для описания содержания столбцов и строк таблицы;

d) *ячейки данных*, формирующие строки и столбцы таблицы.

Тэги для определения таблицы в документе .html

Тэги	Описания тэгов
<TABLE> </TABLE>	Определяют таблицу. Тэг <TABLE> может иметь атрибуты: – BORDER, добавляющий рамку таблицы, его значение – целое положительное число – определяет толщину рамки в пикселях (по умолчанию задается значение 1, а значение 0 указывает на визуальное отсутствие рамки); – ALIGN, определяет способ выравнивания таблицы в документе и может иметь одно из значений *left*, *right*, *center*; – BGCOLOR, задает цвет фона для всей таблицы; – CELLSPACING, задает расстояние в пикселях между двумя соседними ячейками таблицы (по умолчанию имеет значение 2); – CELLPADDING, задает расстояние в пикселях между границей ячейки и ее содержимым (по умолчанию имеет значение 1); – WIDTH и HEIGHT, задают ширину и высоту таблицы (значениями могут быть целые положительные числа (число пикселей) либо целые положительные числа меньшие 101, за которыми записан знак % (указывающие, какая часть от ширины и высоты страницы выделяется для таблицы)).
<CAPTION> </CAPTION>	Определяют название таблицы. Могут быть включены во внутрь тэгов <TABLE> и </TABLE>, но не во внутрь описания ячеек или строк таблицы. Тэг <CAPTION> может включать атрибут ALIGN, определяющий позицию названия по отношению к таблице. Допустимые значения атрибута *bottom*, *top*, *left*, *right*.
<TR> </TR>	Задает строку таблицы. Тэг <TR> может иметь атрибуты: – ALIGN для выравнивания по горизонтали (с одним из значений *left*, *center*, *right*); – VALIGN для выравнивания по вертикали (с одним из значений *top*, *middle*, *bottom*).
<TH> </TH>	Определяют ячейку заголовка. По умолчанию, текст такой ячейки выводится полужирными символами и выравнивается по центру. Тэг <TH> может иметь атрибуты: – WIDTH и HEIGHT для установления ширины и высоты ячейки; – ALIGN и VALIGN; – ROWSPAN, объединяет ячейку с несколькими, расположенными ниже, ячейками (значение атрибута равно общему числу объединенных ячеек); – COLSPAN, объединяет ячейку с несколькими, расположенными справа, ячейками (значение атрибута равно общему числу объединенных ячеек); – NOWRAP, запрещает разрыв текстовой строки из ячейки (то есть ширина столбца может стать сколь угодно большой), записывается без значений; – BGCOLOR, определяет цвет фона ячейки (цвет текста можно установить с помощью атрибута *Color* тэга <Font>).
<TD> </TD>	Определяют обычную ячейку с данными. Тэг <TD> может иметь любой из атрибутов тэга <TH>.

Если некоторая ячейка таблицы пуста, то, как правило, она выводится без рамки. Для того чтобы избежать этого, содержанием ячейки можно сделать текст   или
.

Пример (*рис. 12.16*):

```
<HTML>
<Head> <Title>Пример 1. Таблицы</Title> </Head>
<Body>
<Table border=2 width="90%" height="80%" align="center" bgcolor="lightcyan">
<Caption align="left"><Font color="Red" Size=6> Расписание занятий </font>
</Caption>
<Tr><Th Colspan=2>Понедельник<Th>Вторник<Th>Среда<Th>Четверг<Th>Пятница </Tr>
<Tr align="center"><Td>1<Td>Румынский язык<Td>Химия<Td>Физика
   <Td>История<Td>Алгебра </Tr>
<Tr align="center"><Td>2<Td>Английский язык<Td>Физика<Td>Информатика
   <Td>Русский язык<Td>Биология </Tr>
<Tr align="center"><Td>3<Td>Французский язык<Td>Алгебра<Td>Геометрия
   <Td>Музыка<Td> Румынский язык </Tr>
</Table>
</Body>
</HTML>
```

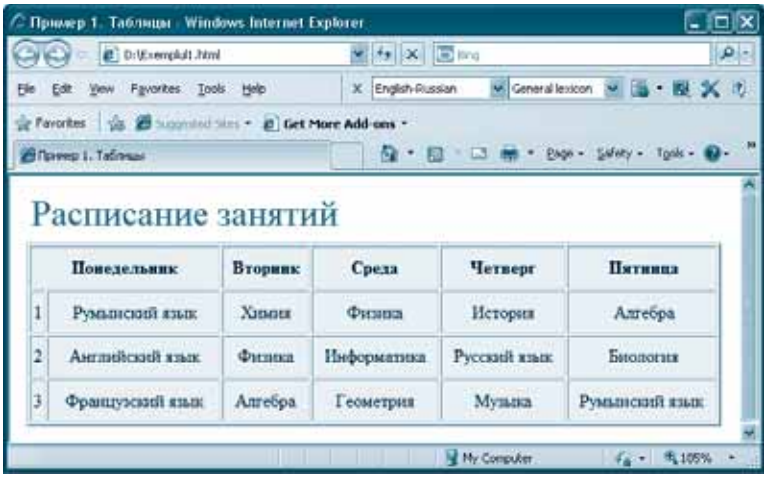


Рис. 12.16

Вопросы и упражнения

❶ Для чего используют таблицы?
❷ Назовите атрибуты тэга <TABLE> и их возможные значения.
❸ Какими атрибутами обладает тэг <CAPTION> и каковы их возможные значения?
❹ Каким образом задается строка таблицы?
❺ Какими тэгами определяется ячейка заголовка таблицы?
❻ Каковы атрибуты и их возможные значения для тэга ячейки заголовка таблицы?
❼ Какими тэгами определяется обычная ячейка данных? Каковы их атрибуты?

❽ Создайте *Web*-страницу, выводящую следующую таблицу:

a)

A	C	E
B	D	F

b)

A	B	C
	D	E

c)

A		D
B	C	E

d)

A	C	E
B		

e)

A	B			C
	D	E	F	
G	H			I

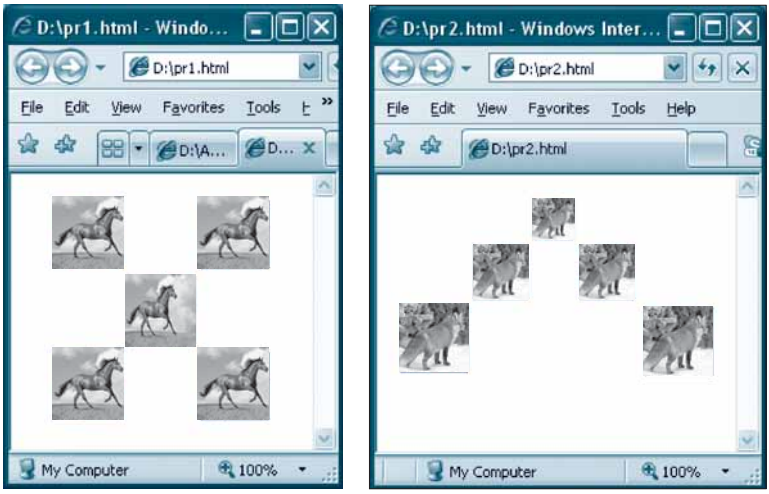
f)

A	C	E
B	D	F
	G	H

g)

A	B	D	E	F	G
	C		H		

❾ Создайте документ HTML, выводящий следующую *Web*-страницу (*рис. 12.17 a, b*) (можно использовать другие изображения):



a) *b)*

Рис. 12.17

❿ Что выведет *Web*-страница со следующим кодом HTML?

a)
```
<HTML>
  <Head><Title>Ce afisam?</Title> </Head>
  <Body>
  <Table align=center border=2 width=100>
     <Tr align=center>
     <Td colspan=2>B</Td><Td rowspan=2>D</Td>
     </Tr>
     <Tr align=center>
     <Td>C</Td><Td>A</Td>
     </Tr>
  </Table>
  </Body>
  </HTML>
```

b)
```
<HTML>
  <Head><Title>Ce afisam?</Title> </Head>
  <Body>
  <Table align=center width=100 border=2>
  <Tr align=center>
     <Td rowspan=2>B</Td><Td>C</Td><Td>A</Td>
  </Tr>
  <Tr align=center>
     <Td colspan=2>D</Td>
  </Tr>
  </Table>
     </body>
  </HTML>
```

⓫ Создайте документ HTML, выводящий следующую таблицу:

Таблица. **Добыча драгоценных камней**

	2010	2012	2014
Рубин	1 483	4 532	6 743
Сапфир		3 835	7 482
Альмаз	683	238	

⓬ Создайте документ, выводящий таблицу в виде:

Домашние животные		Дикие животные		
Могут летать	Не могут летать	Могут летать	Не могут летать	
 Курица	 Кролик	 Ворона	 Волк	Встречаются на территории нашей страны.
 Гусь	 Коза	 Сова	 Лиса	

 Утка	 Конь	 Фламинго	 Слон	Не встречаются на территории нашей страны.
 Перепёлка	 Осёл	 Колибри	 Лев	

Создаем в команде

1. Спроектируйте сайт под одним из приведенных названий:
a) Республика Молдова – европейская страна;
b) Моя родина;
c) Наша школа;
d) Туристические места в Молдове;
e) Монастыри Молдовы;
f) Великие люди;
g) Школьные годы;
h) Фауна Республики Молдова;
i) Молдавская кухня;
j) Наши любимые книги;
k) Престижные профессии;
l) Энциклопедия моды;
m) Юмор – источник хорошего настроения;
n) Успешные выпускники нашего лицея;
o) Качественные отечественные товары;
p) Спорт – это здоровье.

2. Используя язык HTML и различные информационные источники, создайте *Web*-страницы проектируемого сайта. Установите ссылки между ними. Все страницы должны иметь ссылку на главную страницу. Проверьте сайт с различными программами навигации.

3. Создайте и свяжите с сайтом:
а) *Web*-страницу со ссылками на дополнительные источники информации на тему сайта;
б) *Web*-страницу с галереей изображений на тему сайта.

Контрольный тест

1. Установите истинность высказывания:
   a) Текст, записанный между тэгами <HEAD> и </HEAD> документа HTML, будет выведен на панели названия окна программы навигации, в котором интерпретируется этот документ.
   b) Тэг может иметь несколько атрибутов.
   c) Программа навигации будет выводить с новой строки текст, перед которым записан любой из тэгов <P>,
, <HR>.
   d) Текст, записанный между тэгами <DIV> и </DIV>, является форматированным.

2. Выберите тэги для определения физических стилей текста: <B>, </B>, <DFN>, </DFN>, <VAR>, </VAR>, <BIG>, </BIG>, <EM>, </EM>, _,.

3. Напишите код HTML, который, будучи интерпретированным программой навигации, выведет следующие вложенные списки:
   - Фрукты
     a) Яблоки
     b) Груши
     c) Сливы
     d) Абрикосы
   - Овощи
     1. Картофель
     2. Морковь
     3. Лук
   - Молочные продукты
     o Молоко
     o Творог
     o Йогурт

4. Файл f1.html находится в папке A, а файл f2.html – в папке B. Каким будет адрес URL ссылки из файла f1.html на файл f2.html, если:
   a) папки A и B находятся в одном и том же каталоге;
   b) папка A содержится в папке B;
   c) папка B содержится в папке A;
   d) папка A содержится в папке C, входящей в папку B?

5. Прокомментируйте смысл следующего кода HTML:
<A HREF="../TEST/d1.jpg"><img src="d2.jpg" width=50 height=50></A>.

6. Напишите код HTML, который, будучи интерпретирован программой навигации, выведет следующую таблицу:

<table>
<tr><td colspan="2">A</td><td>B</td><td>C</td><td>D</td></tr>
<tr><td colspan="2">E</td><td rowspan="2" colspan="2">F</td><td>G</td></tr>
<tr><td>H</td><td>I</td><td>J</td></tr>
</table>