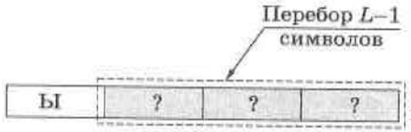
Рекурсивный перебор

В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все слова, состоящие из L букв, которые можно построить из букв этого алфавита.

Это типичная задача на перебор вариантов, которую удобно свести к задаче меньшего размера. Будем определять буквы слова последовательно, одну за другой. Первая буква может быть любой из четырёх букв алфавита. Предположим, что сначала первой мы поставили букву 'Ы'. Тогда для того, чтобы получить все варианты с первой буквой 'Ы', нужно перебрать все возможные комбинации букв на оставшихся $L - 1$ позициях:



Далее поочерёдно ставим на первое место все остальные буквы, повторяя процедуру:

```
алг перебор L символов
нач
 w[1]:='Ы'; перебор последних L-1 символов
 w[1]:='Ш'; перебор последних L-1 символов
 w[1]:='Ч'; перебор последних L-1 символов
 w[1]:='О'; перебор последних L-1 символов
кон
```

Здесь через w обозначена символьная строка, в которой хранится рабочее слово. Таким образом, задача для слов длины L свелась к 4 задачам для слов длины $L - 1$. Как вы знаете, такой прием называется *рекурсией*, а процедура — рекурсивной.

Когда рекурсия должна закончиться? Тогда, когда все символы будут расставлены. При этом нужно вывести получившееся слово на экран и выйти из процедуры.

Подсчитаем количество всех возможных слов длины L. Очевидно, что слов длины 1 всего 4. Добавляя ещё одну букву, получаем $4 \cdot 4 = 16$ комбинаций, для трёх букв — $4 \cdot 4 \cdot 4 = 64$ слова и т. д. Таким образом, из четырёх букв можно составить 4^L различных слов длины L.

В основной программе построим слово (символьную строку) *word* нужной длины (что в ней будет записано, не имеет значения). Процедуре TumbaWords передаётся алфавит в виде символьной константы, слово *word* и число уже установленных символов (в начале — 0):


```
алг ЫШЧО
нач
  лит word = '...'   | из L символов
  TumbaWords('ЫШЧО', word, 0)
кон
```

В процедуре используется описанный выше рекурсивный алгоритм:

```
алг TumbaWords(лит A, w0, цел N)
нач
  если N = длин(w0) то | слово построено
    вывод w0, нс
    выход
  все
  цел i
  лит w
  w:=w0
  нц для i от 1 до длин(A)
    w[N+1]:=A[i]
    TumbaWords(A, w, N+1) | рекурсия
  кц
кон
```

Если условие в начале процедуры ложно (не все символы расставлены), в цикле перебираем все символы алфавита и поочерёдно ставим их на первое свободное место, а затем вызываем рекурсивно эту же процедуру, увеличивая третий аргумент на 1. Поскольку школьный алгоритмический язык не позволяет менять переменную-аргумент, мы вынуждены ввести дополнительную локальную переменную w.

Приведём аналогичную программу на Паскале:

```
program YSCHO;
var word: string;
procedure TumbaWords(A, w: string; N: integer);
var i: integer;
```

```
begin
  if N=Length(w) then begin
    writeln(w);
    exit
  end;
  for i:=1 to Length(A) do begin
    w[N+1]:=A[i];
    TumbaWords(A, w, N+1)
  end
end;
begin
   word:= '...';
   TumbaWords('ЫШЧО', word, 0)
end.
```

Сравнение и сортировка строк

Строки, как и числа, можно сравнивать. Для строк, состоящих из одних букв (русских или латинских), результат сравнения очевиден: меньше будет та строка, которая идет раньше в алфавитном порядке. Например, слово «паровоз» будет «меньше», чем слово «пароход»: они отличаются в пятой букве: 'в' < 'х'. Более короткое слово, которое совпадает с началом более длинного, тоже будет стоять раньше в алфавитном списке, поэтому 'пар' < 'парк'.

Но откуда компьютер «знает», что такое алфавитный порядок? И как сравнивать слова, в которых есть строчные и заглавные буквы, а также цифры и другие символы? Что больше, 'ПАР', 'Пар' или 'пар'? Оказывается, при сравнении строк используются коды символов. Тогда получается, что:

'ПАР' < 'Пар' < 'пар'.

Возьмём пару 'ПАР' и 'Пар'. Первый символ в обоих словах одинаков, а второй отличается — в первом слове буква заглавная, а во втором — такая же, но строчная. Во всех современных таблицах символов (включая UNICODE) заглавные буквы стоят раньше строчных и поэтому имеют меньшие коды. Поэтому 'А' < 'а', 'П' < 'а' и 'ПАР' < 'Пар' < 'пар'.

А как же с другими символами (цифрами, латинскими буквами)? Цифры стоят в кодовой таблице по порядку, причём раньше, чем латинские буквы; латинские буквы — раньше, чем русские;

заглавные буквы (русские и латинские) — раньше, чем соответствующие строчные. Поэтому:

'5STEAM' < 'STEAM' < 'Steam' < 'steam' < 'ПАР' < 'Пар' < 'пар'.

Сравнение строк используется, например, при сортировке. Рассмотрим такую задачу: надо ввести с клавиатуры 10 фамилий и вывести их на экран в алфавитном порядке.

Для сортировки удобно выделить массив строк, который в школьном алгоритмическом языке объявляется как **литтаб**, а в Паскале — **array of string**. Ввод и вывод выполняются с помощью стандартных циклов, а сортировка — любым известным способом, например методом пузырька:

```
алг Сортировка строк
нач
  цел i, j, N=10
  литтаб S[1:N]
  лит s1
  нц для i от 1 до N
    ввод S[i]
  кц
  нц для i от 1 до N-1
    нц для j от N-1 до i шаг -1
      если S[j+1]<S[j] то
        s1:=S[j]
        S[j]:=S[j+1]
        S[j+1]:=s1
      все
    кц
  кц
  нц для i от 1 до N
    вывод S[i], нс
  кц
кон
```

```
program StringSort;
const N=10;
var i,j: integer;
  s1: string;
  S: array[1..N] of string;
begin
  for i:=1 to N do
    readln(S[i]);
  for i:=1 to N-1 do
   for j:=N-1 downto i do
    if S[j+1]<S[j] then begin
      s1:=S[j];
      S[j]:=S[j+1];
      S[j+1]:=s1
    end;
  for i:=1 to N do
    writeln(S[i])
end.
```

Вопросы и задания

1. Что такое символьная строка?
2. Почему неудобно заменять строки массивами символов?
3. Как объявляются строки в школьном алгоритмическом языке и в Паскале?
4. Как обращаться к элементу строки с заданным номером?
5. Как вычисляется длина строки?
6. Что обозначает операция «+» применительно к строкам?
7. Перечислите основные операции со строками и соответствующие им стандартные функции.
8. Как определить, что при поиске в строке образец не найден?
9. Чем различаются средства школьного алгоритмического языка и Паскаля для работы со строками?
10. Как преобразовать число из символьного вида в числовой и обратно?
11. Почему строку не всегда можно преобразовать в число? Как определить, что преобразование закончилось неудачно?
12. Объясните выражение «рекурсивный перебор».
13. Сравните на примерах рекурсивные и нерекурсивные методы решения переборных задач.

Подготовьте сообщение

а) «Символьные переменные в языке Си»
б) «Символьные строки в языке Python»

Задачи

1. Напишите программу, которая во введённой символьной строке заменяет все буквы «а» на буквы «б» и наоборот, заглавные — на заглавные, строчные — на строчные. При вводе строки 'абсАБС' должен получиться результат 'басБАС'.
2. Введите символьную строку и проверьте, является ли она *палиндромом* (палиндром читается одинаково в обоих направлениях, например: «казак»).
3. Введите адрес файла и «разберите» его на части, разделённые знаком «/». Каждую часть выведите в отдельной строке.
4. Введите строку, в которой записана сумма натуральных чисел, например '1+25+3'. Вычислите это выражение.
5. Введите с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Выведите инициалы и фамилию. Например, при вводе строки 'Иванов Пётр Семёнович' должно получиться 'П.С. Иванов'.

6. Разберитесь, как работает ещё одна функция замены:

```
алг лит ReplaceBad(лит s0, sOld, sNew)
нач
  лит s
  s:=s0
  цел p, len
  len:=длин(sOld)
  p:=позиция(sOld,s)
  нц пока p>0
    удалить(s, p, len)
    вставить(sNew, s, p)
    p:=позиция(sOld, s)
  кц
  знач:=s
кон
```

Приведите пример входных данных, при которых эта функция работает неправильно.

7. Напишите рекурсивную версию процедуры `replaceAll`. Сравните две версии. Какую из них вы рекомендуете выбрать и почему?
8. Напишите функцию, которая изменяет в имени файла расширение на заданное (например, на bak). Функция принимает два параметра: имя файла и нужное расширение. Учтите, что в исходном имени расширение может быть пустым.
9. Напишите функцию, которая определяет, сколько раз входит в символьную строку заданное слово.
10. С клавиатуры вводится число N, обозначающее количество футболистов команды «Бублик», а затем — N строк, в каждой из которых — информация об одном футболисте в таком формате:

<Фамилия> <Имя> <год рождения> <голы>

Данные разделяются одним пробелом. Нужно подсчитать, сколько футболистов, родившихся в период с 1998 по 2000 г., не забили мячей вообще.

11. В условиях задачи 10 определите фамилию и имя футболиста, забившего наибольшее число голов, и количество забитых им голов.
12. В условиях задачи 10 выведите в алфавитном порядке фамилии и имена всех футболистов, которые забили хотя бы один гол. В списке не более 100 футболистов.
13. Измените программу рекурсивного перебора так, чтобы длину слова можно было ввести с клавиатуры.
14. Выведите на экран все слова из L букв, в которых буква «Ы» встречается более 1 раза, и подсчитайте их количество.

15. Выведите на экран все слова из L букв, в которых есть одинаковые буквы, стоящие рядом (например, «ЫШШО»), и подсчитайте их количество.
16. В языке племени «тумба-юмба» запрещено ставить две гласные буквы подряд. Выведите все слова длины L, удовлетворяющие этому условию, и найдите их количество.

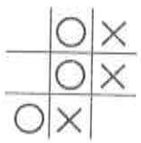
*17. Напишите программу перебора слов заданной длины, не использующую рекурсию. Попробуйте составить функцию, которая на основе некоторой комбинации вычисляет следующую за ней.

*18. Перестановки. К вам пришли L гостей. Напишите программу, которая выводит все *перестановки* — способы посадить их за столом. Гостей можно обозначить латинскими буквами.

§ 67 Матрицы

Что такое матрицы?

Многие программы работают с данными, организованными в виде таблиц. Например, при составлении программы для игры в крестики-нолики нужно запоминать состояние каждой клетки квадратной доски. Можно поступить так: пустым клеткам присвоить код «–1», клетке, где стоит нолик, — код 0, а клетке с крестиком — код 1. Тогда информация о состоянии поля может быть записана в виде таблицы (рис. 8.14).



	1	2	3
1	–1	0	1
2	–1	0	1
3	0	1	–1

Рис. 8.14

Такие таблицы называются **матрицами** или **двумерными массивами**. Каждый элемент матрицы, в отличие от обычного (линейного) массива, имеет два индекса — номер строки и номер столбца. На рисунке 8.14 серым фоном выделен элемент, находящийся на пересечении второй строки и третьего столбца.

Матрица — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т. д.). Каждый элемент матрицы имеет два индекса — номера строки и столбца.

При объявлении матриц указывают два диапазона индексов (для строк и столбцов):

```
цел N=3, M=4
целтаб A[1:N,1:M]
вещтаб X[-3:0,-8:M]
логтаб L[1:N,0:1]
```

```
const N=3; M=4;
var A: array[1..N,1..M] of integer;
    X: array[-3..0,-8..M] of double;
    L: array[1..N,0..1] of boolean;
```

Так же, как и для одномерных массивов, в языке Паскаль индексами матриц могут быть значения любого порядкового типа, например символы:

```
var Q: array['a'..'z',1..10] of integer;
```

Каждому элементу матрицы можно присвоить любое значение, допустимое для выбранного типа данных. Поскольку индексов два, для заполнения матрицы нужно использовать вложенный цикл. Далее в примерах будем считать, что объявлена матрица из N строк и M столбцов, а i и j — целочисленные переменные, обозначающие индексы строки и столбца. В следующем примере матрица заполняется случайными числами и выводится на экран:

```
нц для i от 1 до N
  нц для j от 1 до M
    A[i,j]:=irand(20,80)
    вывод A[i,j]:3
  кц
  вывод нс
кц
```

```
for i:=1 to N do begin
  for j:=1 to M do begin
    A[i,j]:=random(61)+20;
    write(A[i,j]:3)
  end;
  writeln
end;
```

Такой же двойной цикл нужно использовать для перебора всех элементов матрицы. Вот как вычисляется сумма значений всех элементов:

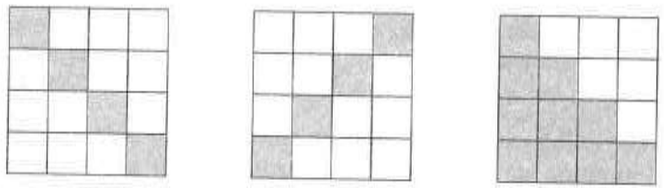
```
s:=0
нц для i от 1 до N
  нц для j от 1 до M
    s:=s+A[i,j]
  кц
кц
```

```
s:=0;
for i:=1 to N do
  for j:=1 to M do
    s:=s+A[i,j];
```

Обработка элементов матрицы

Покажем, как можно обработать (например, сложить) некоторые элементы **квадратной матрицы** A, содержащей N строк и N столбцов.

На рисунке 8.15, *а* выделена главная диагональ матрицы, на рис. 8.15, *б* — вторая, побочная диагональ, на рис. 8.14, *в* — главная диагональ и все элементы под ней.



а *б* *в*

Рис. 8.15

Главная диагональ — это элементы $A[1,1]$, $A[2,2]$, ..., $A[N,N]$, т. е. элементы, у которых номер строки равен номеру столбца. Для перебора этих элементов нужен один цикл:

```
нц для i от 1 до N
  | работаем с A[i,i]
кц
```

```
for i:=1 to N do begin
  { работаем с A[i,i] }
end;
```

Элементы **побочной диагонали** — это $A[1,N]$, $A[2,N-1]$, ..., $A[N,1]$. Заметим, что сумма номеров строки и столбца для каждого из этих элементов равна $N + 1$, поэтому получаем такой цикл перебора:

```
нц для i от 1 до N
  | работаем с A[i,N+1-i]
кц
```

```
for i:=1 to N do begin
  { работаем с A[i,N+1-i] }
end;
```

В случае обработки всех элементов на главной диагонали и под ней (см. рис. 8.15, *в*) нужен вложенный цикл: номер строки будет меняться от 1 до N, а номер столбца для каждой строки i — от 1 до i:

```
нц для i от 1 до N
  нц для j от 1 до i
    | работаем с A[i,j]
  кц
кц
```

```
for i:=1 to N do
  for j:=1 to i do begin
    { работаем с A[i,j] }
  end
```


Чтобы переставить строки или столбцы, достаточно одного цикла. Например, переставим строки 2 и 4, используя вспомогательную целую переменную *c*:

```
нц для j от 1 до M
 c:=A[2,j]
 A[2,j]:=A[4,j]
 A[4,j]:=c
кц
```

```
for j:=1 to M do
 c:=A[2,j];
 A[2,j]:=A[4,j];
 A[4,j]:=c
end;
```

Вопросы и задания

1. Что такое матрицы? Зачем они нужны?
2. Сравните понятия «массив» и «матрица».
3. Как вы думаете, можно ли считать, что первый индекс элемента матрицы — это номер столбца, а второй — номер строки?
4. Могут ли индексы элементов матрицы принимать отрицательные и нулевые значения?
5. Что такое главная и побочная диагонали матрицы?
6. Почему суммирование элементов главной диагонали требует одиночного цикла, а суммирование элементов под главной диагональю — вложенного?

Подготовьте сообщение

а) «Матрицы в языке Си»
б) «Матрицы в языке Python»

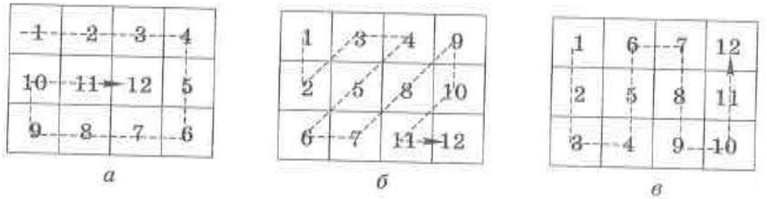
Задачи

1. Напишите программу, которая находит минимальный и максимальный элементы матрицы и их индексы.
2. Напишите программу, которая находит минимальный и максимальный элементы из элементов матрицы с чётными положительными значениями и их индексы. Учтите, что таких элементов в матрице может и не быть.
3. Напишите программу, которая выводит на экран строку матрицы, сумма значений элементов которой наибольшая.
4. Напишите программу, которая выводит на экран столбец матрицы, сумма значений элементов которого наименьшая.
5. Напишите программу, которая заполняет матрицу случайными числами, а затем записывает нули во все элементы выше главной диагонали.

6. Напишите программу, которая заполняет матрицу случайными числами, а затем записывает нули во все элементы выше побочной диагонали.
7. Напишите программу, которая заполняет матрицу размером 7 × 7 случайными числами, а затем записывает в элементы, отмеченные на рисунках серым фоном, число 99.

а *б* *в*

8. Заполните матрицу, содержащую N строк и M столбцов, натуральными числами по спирали и змейкой, как на рисунках.



а *б* *в*

9. Заполните квадратную матрицу случайными числами и выполните её *транспонирование*: так называется процедура, в результате которой строки матрицы становятся столбцами, а столбцы — строками.

1	2	3
4	5	6
7	8	9

→

1	4	7
2	5	8
3	6	9

10. Пиксели рисунка закодированы числами от 0 до 255 (обозначающими яркость пикселей) в виде матрицы, содержащей N строк и M столбцов. Нужно преобразовать рисунок в чёрно-белый по следующему алгоритму:
    - вычислить среднюю яркость пикселей по всему рисунку;
    - все пиксели, яркость которых меньше средней, сделать чёрными (записать код 0), а остальные — белыми (код 255).

11. Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните отражение рисунка сверху вниз, как показано на рисунке.

1	2	3
4	5	6
7	8	9

→

7	8	9
4	5	6
1	2	3

12. Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните поворот рисунка вправо на 90 градусов, как показано на рисунке.

1	2	3
4	5	6
7	8	9

→

7	4	1
8	5	2
9	6	3

*13. Напишите программу, которая играет с человеком в крестики-нолики.
*14. В матрице, содержащей N строк и M столбцов, «записана» карта островного государства Лимония (нули обозначают море, а единицы — сушу). Все острова имеют форму прямоугольника. Напишите программу, которая по готовой карте определяет количество островов.

§ 68 Работа с файлами

Как работать с файлами?

Файл — это набор данных на диске, имеющий имя. Файлы часто рассматривают как сложный тип данных, в котором используется последовательный доступ, т. е. данные читаются последовательно, как на магнитной ленте. Тем не менее современные языки программирования содержат функции для произвольного доступа, позволяющие перейти сразу в заданную позицию файла.

С точки зрения программиста, бывают файлы двух типов:

1) **текстовые**, которые содержат текст, разбитый на строки; таким образом, из всех специальных символов в текстовых файлах могут быть только символы перехода на новую строку;

2) **двоичные**, в которых могут содержаться любые данные и любые коды без ограничений; в двоичных файлах хранятся рисунки, звуки, видеофильмы и т. д.

Мы будем рассматривать только текстовые файлы.

Работа с файлом из программы включает три этапа. Сначала надо **открыть** файл, т. е. сделать его активным для программы. Если файл не открыт, то программа не может к нему обращаться. При открытии файла указывают режим работы: чтение, запись или добавление данных в конец файла. Чаще всего открытый файл блокируется так, что другие программы не могут использовать его. Когда файл открыт (активен), программа выполняет все необходимые операции с ним. После этого нужно **закрыть** файл, т. е. освободить его, разорвать связь с программой. Именно при закрытии все последние изменения, сделанные программой в файле, записываются на диск.

Такой принцип работы иногда называют «принципом сэндвича», в котором три «слоя»: «хлеб», затем «начинка», и потом снова «хлеб» (рис. 8.16).

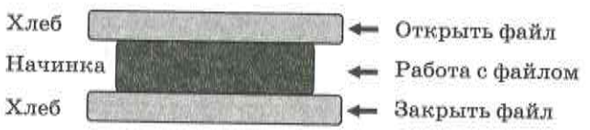


Рис. 8.16

В большинстве языков программирования с файлами работают через вспомогательные переменные (их называют указателями, идентификаторами и т. п.).

Например, в школьном алгоритмическом языке существуют стандартные функции:

- `открыть на чтение`
- `открыть на запись`
- `открыть на добавление`

которые принимают имя файла (символьную строку) и возвращают ссылку на открытый файл (ключ файла), которая записывается в специальную переменную типа **файл**. Команда `закрыть` закрывает открытый файл:

```
файл Fin, Fout
Fin:=открыть на чтение('input.txt')
Fout:=открыть на запись('output.txt')
| здесь работаем с файлами
закрыть(Fin)
закрыть(Fout)
```


После закрытия файла файловую переменную можно использовать повторно, для работы с этим или другим файлом. После окончания работы программы все открытые файлы закрываются автоматически.

Если файл, который открывается на чтение, не найден, возникает ошибка. Если существующий файл открывается на запись, его содержимое уничтожается.

Чтение из текстовых файлов выполняет команда `ввод`, а запись — команда `вывод`. При работе с файлами на первом месте в списке аргументов нужно указать файловую переменную, в которой записан ключ открытого файла. В остальном чтение и запись происходит так же, как и для стандартных устройств — клавиатуры и текстового экрана (эта пара устройств называется *терминалом* или *консолью*).

Если в переменных *Fin* и *Fout* записаны ключи файлов, открытых соответственно на ввод и на вывод, можно написать так:

```
ввод Fin, a, b
вывод Fout, a, '+', b, '=', a+b
```

У нас получилась версия программы, которая складывает два числа, прочитанные из одного файла, и записывает результат в другой файл.

Как правило, текстовый файл — это «устройство» последовательного доступа к данным. Это значит, что для того, чтобы прочитать 100-е по счёту значение из файла, нужно сначала прочитать предыдущие 99. В своей внутренней памяти система хранит положение **указателя** (файлового курсора), который определяет текущее место в файле. При открытии файла указатель устанавливается в самое начало файла, при чтении смещается на позицию, следующую за прочитанными данными, а при записи — на следующую свободную позицию.

Если нужно повторить чтение с начала файла, нужно закрыть его, а потом снова открыть. В школьном алгоритмическом языке вместо этого используется команда `начать чтение`:

```
начать чтение(Fin)
```

Когда файловый курсор указывает на конец файла, логическая функция конец файла возвращает значение «истина»:

```
если конец файла (Fin)
  вывод 'Данные кончились'
все
```

В языке Паскаль для работы с текстовыми файлами используются файловые переменные типа Text:

```
var Fin, Fout: Text;
```

Сначала их нужно с помощью процедуры Assign связать с именами файлов, к которым будем обращаться:

```
Assign(Fin, 'input.txt');
Assign(Fout, 'output.txt');
```

Эти файлы ещё не открыты, и работать с ними нельзя. Для открытия файлов в разных режимах существуют стандартные процедуры

- Reset — открыть на чтение,
- Rewrite — открыть на запись,
- Append — открыть на добавление.

У них один параметр — файловая переменная, которая была предварительно связана с именем файла. После окончания работы файлы закрываются командой Close:

```
Reset(Fin);
Rewrite(Fout);
  {здесь работаем с файлами}
Close(Fin);
Close(Fout);
```

Для чтения и записи используются процедуры read, readln, write и writeln, в которых в качестве первого аргумента указывают файловую переменную:

```
readln(Fin, a, b);
writeln(Fout, a, '+', b, '=', a+b);
```


Напомним, что readln, в отличие от read, читает всё до конца строки. В приведённом выше примере после прочтения значений переменных *a* и *b* все оставшиеся символы до конца строки игнорируются, а следующий вызов read или readln берёт данные уже с новой строки.

Функция Eof (англ. **EOF** — *end of file* — конец файла) возвращает значение True (истина), если указатель стоит на последней позиции файла:

```
if Eof(Fin) then
  write('Данные кончились');
```

Неизвестное количество данных

Предположим, что в текстовом файле записано в столбик неизвестное количество чисел и требуется найти их сумму. В этой задаче не нужно одновременно хранить все числа в памяти (и не нужно выделять массив!), достаточно читать по одному числу:

```
нц пока не конец файла
  прочитать число из файла
  добавить его к сумме
кц
```

Далее будем считать, что файловая переменная *Fin* связана с файлом, открытым на чтение. Основная часть программы (без объявления переменных, открытия и закрытия файлов) выглядит так:

```
S:=0
нц пока не конец файла(Fin)
  ввод Fin, x
  S:=S+x
кц
```

```
S:=0;
while not Eof(Fin) do begin
  readln(Fin, x);
  S:=S+x
end;
```

Обработка массивов

В текстовом файле записано не более 100 целых чисел. Требуется вывести в другой текстовый файл те же числа, отсортированные в порядке возрастания.

Особенность этой задачи в том, что для сортировки нам нужно удерживать в памяти все числа, т. е. для их хранения нужно вы-

делить массив. Косвенно на это указывает ограничение — чисел не более 100. Поэтому массив должен иметь не менее 100 элементов:

```
цел MAX=100
целтаб A[1:MAX]
```

```
const MAX=100;
var A: array[1..MAX] of integer;
```

Основная интрига состоит в том, что количество чисел точно не известно. Следовательно, нам нужно считывать значения и записывать их последовательно в первые ячейки массива, пока данные не закончатся — тогда цикл чтения останавливается. Кроме того, нужно сделать защиту от слишком большого количества данных: если 100 чисел уже записаны в массив, цикл должен остановиться. Ниже приведены циклы чтения на школьном алгоритмическом языке:

```
N:=0
нц пока не конец файла(Fin) и N<MAX
  N:=N+1
  ввод Fin, A[N]
кц
```

и на Паскале:

```
N:=0;
while (not Eof(Fin)) and (N<MAX) do begin
  N:=N+1;
  readln(Fin, A[N])
end;
```

Целая переменная N служит счётчиком прочитанных из файла чисел.

Теперь нужно отсортировать первые N значений массива A (этот код вы уже можете написать самостоятельно) и вывести их во второй файл, открытый на запись:

```
Fout:= открыть на запись('output.txt')
нц для i от 1 до N
  вывод Fout, A[i], нс
кц
закрыть(Fout)
```


Вариант на Паскале:

```
Assign(Fout, 'output.txt');
Rewrite(Fout);
for i:=1 to N do
  writeln(Fout, A[i]);
Close(Fout);
```

Обработка строк

Как известно, современные компьютеры бóльшую часть времени заняты обработкой символьной, а не числовой информации. Предположим, что в текстовом файле записаны данные о собаках, привезённых на выставку: в каждой строчке кличка собаки, её возраст (целое число) и порода, например,

```
Мухтар 4 немецкая овчарка
```

Элементы отделяются друг от друга одним пробелом. Нужно вывести в другой файл сведения о собаках, которым меньше 5 лет.

В этой задаче данные можно обрабатывать по одной строке (не нужно загружать все строки в оперативную память):

```
нц пока не конец файла (Fin)
  прочитать строку из файла Fin
  разобрать строку – выделить возраст
  если возраст<5 то
    записать строку в файл Fout
  все
кц
```

Здесь, как и раньше, *Fin* и *Fout* — файловые переменные, связанные с файлами, открытыми на чтение и запись соответственно.

Будем считать, что все данные корректны, т. е. первый пробел отделяет кличку от возраста, а второй — возраст от породы. Тогда разбор строки можно выполнить так:

```
найти в строке пробел
удалить из строки кличку с первым пробелом
найти в строке пробел
выделить возраст перед пробелом
преобразовать возраст в числовой вид
```

Для этого используются стандартные функции для работы с символьными строками (см. § 66):

```
лит s, sAge
цел age, p
лог OK
...
p:=позиция(' ', s)
s:=s[p+1:длин(s)]
p:=позиция(' ',s)
sAge:=s[1:p-1]
age:=лит_в_цел(sAge, OK)
```

```
var s, sAge: string;
 age, p, r: integer;
...
p:=Pos(' ', s);
Delete(s, 1, p);
p:=Pos(' ', s);
sAge:=Copy(s, 1, p-1);
Val(sAge, age, r);
```

В результате работы этого фрагмента возраст собаки оказывается в переменной age. Конечно, при этом исходная строка «портится», поэтому все операции нужно выполнять с её копией. Основной цикл будет выглядеть так:

```
лит s, s0
нц пока не конец файла(Fin)
  ввод Fin, s0
  s:=s0
    | обработка строки s
  если age<5 то
    вывод Fout, s0, нс
  все
кц
```

```
var s, s0: string;
...
while not Eof(Fin) do begin
  readln(Fin, s0);
  s:=s0;
    {обработка строки s}
  if age<5 then
    writeln(Fout, s0)
end;
```

Вопросы и задания

1. Чем различаются текстовые и двоичные файлы по внутреннему содержанию? Можно ли сказать, что текстовый файл — это частный случай двоичного файла?
2. Объясните «принцип сэндвича» при работе с файлами.
3. Как вы думаете, почему открытый программой файл, как правило, блокируется, и другие программы не могут получить к нему доступ?
4. Почему рекомендуется вручную закрывать файлы, хотя при закрытии программы они закроются автоматически? В каких ситуациях это может быть важно?
5. Что такое файловая переменная? Почему для работы с файлом используют не имя файла, а файловую переменную?
6. В каком случае одна и та же файловая переменная может быть использована для работы с несколькими файлами, а в каком — нет?

7. Что такое последовательный доступ к данным?
8. Как можно начать чтение данных из файла с его начала?
9. Как определить, что данные в файле закончились?
10. В каких случаях нужно знать максимальное количество данных в файле, а в каких — нет?
11. В каких случаях нужно открывать одновременно несколько файлов?

Подготовьте сообщение

а) «Работа с файлами в языке Си»
б) «Работа с файлами в языке Python»

Задачи

1. Напишите программу, которая находит среднее арифметическое всех чисел, записанных в файле в столбик, и выводит результат в другой файл.
2. Напишите программу, которая находит минимальное и максимальное среди чётных положительных чисел, записанных в файле, и выводит результат в другой файл. Учтите, что таких чисел может вообще не быть.
3. В файле в столбик записаны целые числа. Напишите программу, которая определяет длину самой длинной цепочки идущих подряд одинаковых чисел и выводит результат в другой файл.
4. В файле записано не более 100 чисел. Отсортировать их по возрастанию последней цифры и записать в другой файл.
5. В файле записано не более 100 чисел. Отсортировать их по возрастанию суммы цифр и записать в другой файл.
6. В двух файлах записаны отсортированные по возрастанию массивы неизвестной длины. Объединить их и записать результат в третий файл. Полученный массив также должен быть отсортирован по возрастанию.
7. Дополните решение задачи о собаках, так чтобы программа обрабатывала ошибки в исходных данных. При любых ошибках программа не должна завершаться аварийно.
8. В исходном файле записана речь подростка, в которой часто встречается слово-паразит «короче», например: «Мама, короче, мыла, короче, раму». Убрать из текста все слова-паразиты (должно остаться «Мама мыла раму»).
9. Прочитать текст из файла и подсчитать количество слов в нём.
10. Прочитать текст из файла и вывести в другой файл только те строки, в которых есть слова, начинающиеся с буквы «А».
11. Прочитать текст из файла и вывести в другой файл в столбик все слова, которые начинаются с буквы «А».

12. Прочитать текст из файла, заменить везде слово «паровоз» на слово «пароход» и записать в другой файл.
13. В файле записаны данные о результатах сдачи экзамена. Каждая строка содержит фамилию, имя и количество баллов, разделённые одним пробелом:

```
<Фамилия> <Имя> <Количество баллов>
```

Вывести фамилии и имена тех учеников, которые получили больше 80 баллов.
14. В задаче 13 добавить к списку нумерацию, например:

```
1) Иванов Вася
2) Петров Петя
```

15. В задаче 14 сократить имя до одной буквы и поставить перед фамилией:

```
1) В. Иванов
2) П. Петров
```

16. В задаче 15 отсортировать список по алфавиту (по фамилии).
*17. В задаче 15 отсортировать список по убыванию полученного балла (вывести балл в выходной файл).

Практические работы к главе 8[1]

Работа № 25 «Простые вычисления»
Работа № 26 «Ветвления»
Работа № 27 «Сложные условия»
Работа № 28 «Множественный выбор»
Работа № 29 «Задачи на ветвления»
Работа № 30 «Циклы с условием»
Работа № 31 «Циклы с условием-2»
Работа № 32 «Циклы с переменной»
Работа № 33 «Вложенные циклы»
Работа № 34 «Процедуры»
Работа № 35 «Процедуры с изменяемыми параметрами»
Работа № 36 «Функции»
Работа № 37 «Логические функции»
Работа № 38 «Рекурсия»
Работа № 39 «Стек»
Работа № 40 «Перебор элементов массива»
Работа № 41 «Линейный поиск»
Работа № 42 «Поиск максимального элемента массива»
Работа № 43 «Алгоритмы обработки массивов»

[1] Обратите внимание: с 2017 года поменялась нумерация практических работ на авторском сайте.

Работа № 44 «Отбор элементов массива по условию»
Работа № 45 «Сортировка. Метод пузырька»
Работа № 46 «Сортировка. Метод выбора»
Работа № 47 «Быстрая сортировка»
Работа № 48 «Двоичный поиск»
Работа № 49 «Посимвольная обработка строк»
Работа № 50 «Функции для работы со строками»
Работа № 51 «Преобразования "строка-число"»
Работа № 52 «Строки в процедурах и функциях»
Работа № 53 «Рекурсивный перебор»
Работа № 54 «Сравнение и сортировка строк»
Работа № 55 «Обработка символьных строк: сложные задачи»
Работа № 56 «Матрицы»
Работа № 57 «Обработка блоков матрицы»
Работа № 58 «Файловый ввод и вывод»
Работа № 59 «Обработка массивов из файла»
Работа № 60 «Обработка смешанных данных из файла»

ЭОР к главе 8 на сайте ФЦИОР (http://fcior.edu.ru)

- Понятие алгоритма, его свойства. Линейный алгоритм
- Основные структуры данных
- Реализация основных алгоритмических конструкций
- Алгоритмы сортировки
- Алгоритмы поиска
- Организация работы с текстовыми файлами

Самое важное в главе 8

- Алгоритмы могут записываться на псевдокоде, в виде блок-схем и на языках программирования. Алгоритм, записанный на языке программирования, называется программой.
- Данные, с которыми работает программа, хранятся в переменных. Переменная — это величина, которая имеет имя, тип и значение. Значение переменной может изменяться во время выполнения программы.
- В языках программирования различают простые типы данных (целые и вещественные числа, символы, логические значения) и сложные (составные), например, массивы, символьные строки, файлы и др.

- Любой алгоритм можно записать, используя последовательное выполнение операторов, ветвления и циклы. Ветвления предназначены для выбора вариантов действий в зависимости от выполнения некоторых условий. Цикл — это многократное повторение одинаковых действий.
- Подпрограммы — это вспомогательные алгоритмы, которые могут многократно вызываться из основной программы и других подпрограмм. Подпрограммы-процедуры выполняют описанные в них действия, а подпрограммы-функции в дополнение к этому возвращают результат этих действий (число, символ, логическое значение и т. д.). Данные, передаваемые в подпрограмму, называют аргументами. В подпрограмме эти данные представлены как локальные переменные, которые называются параметрами подпрограммы.
- Рекурсивные алгоритмы основаны на последовательном сведении исходной задачи ко всё более простым задачам такого же типа (с другими значениями параметров). Рекурсия может служить заменой циклу. Любой рекурсивный алгоритм можно записать без рекурсии, но во многих случаях такая запись более длинная и менее понятная.
- Массив — это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка массива имеет уникальный индекс (как правило, это номер элемента).
- Сортировка — это расстановка элементов массива в заданном порядке. Цель сортировки — облегчить последующий поиск. Для отсортированного массива можно применить двоичный поиск, который работает значительно быстрее, чем линейный.
- Символьная строка — это последовательность символов, расположенных в памяти рядом (в соседних ячейках). Строка представляет собой единый объект, она может менять свою длину во время работы программы.
- Матрица — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т. д.). Каждый элемент матрицы имеет два индекса; в большинстве языков программирования это номера строки и столбца.
- До выполнения операций с файлом нужно открыть файл (сделать его активным), а после завершения всех действий — закрыть (освободить).

Глава 9
Решение вычислительных задач на компьютере

§ 69
Точность вычислений

«Недостатки математического образования с наибольшей отчётливостью проявляются в чрезмерной точности численных расчётов», — писал выдающийся немецкий математик первой половины XIX века Карл Фридрих Гаусс.

Все практические расчёты выполняются неточно, с некоторой погрешностью (ошибкой, отклонением от истинного значения). В первую очередь это связано с тем, что неточно известны исходные данные, которые получаются в результате измерений.

Погрешности измерений

Окружающие нас предметы имеют различные числовые характеристики (длину, массу, объём и др.), которые часто приходится измерять для решения практических задач. Для измерений используются приборы, каждый из которых имеет определённую точность. Это значит, что с помощью данного прибора невозможно зафиксировать изменение величины, меньшее, чем *цена деления шкалы* этого прибора. Поэтому измеренное значение величины всегда отличается от точного (истинного), разность между ними называют **погрешностью измерения**.

Пусть нужно найти толщину дна кружки, используя только линейку с ценой деления 1 мм. Линейкой можно измерить высоту кружки снаружи и глубину внутри (рис. 9.1). При этом точность измерений будет не выше чем 1 мм (0,1 см). Например, если измеренная высота кружки оказалась примерно 8,2 см, на самом деле она может быть от 8,1 до 8,3 см. Если измеренная глубина равна 7,8 см, фактическая может быть от 7,7 до 7,9 см.

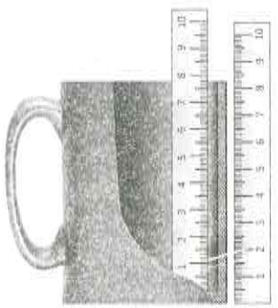


Рис. 9.1

Используя данные измерений, можно найти толщину дна как разность

$$8{,}2 - 7{,}8 = 0{,}4 \text{ см}.$$

Это не означает, что толщина дна действительно такая. Действительно, с учётом ошибок измерений она может быть равна как 8,1 – 7,9 = 0,2 см, так и 8,3 – 7,7 = 0,6 см. Таким образом, реальная толщина может быть от 0,2 до 0,6 см (разница в 3 раза!), и в полученном ответе (0,4 см) нет ни одной верной значащей цифры! Обычно в этом случае пишут ответ в виде 0,4 ± 0,2 см.

В приведённом примере погрешность 0,2 см — это так называемая **абсолютная погрешность** Δx. Для оценки качества измерений чаще используют **относительную погрешность**, которая вычисляется как отношение абсолютной погрешности Δx к истинному значению величины x^*:

$$\delta_x = \frac{\Delta x}{x^*} \cdot 100\%.$$

Поскольку истинное значение, как правило, неизвестно, его обычно заменяют на полученный результат изменений[1]. В данном случае относительную погрешность можно оценить как

$$\delta_x = \frac{0{,}2}{0{,}4} \cdot 100\% = 50\%.$$

Это очень большое значение, которое говорит о низкой точности измерений.

Погрешности вычислений

Пусть нужно вычислить площадь сечения цилиндра, диаметр которого D = 1,2 см известен с точностью 0,1 см. По известной формуле площади круга получаем (например, на калькуляторе):

Рис. 9.2

$$S = \frac{\pi \cdot D^2}{4} = 1{,}130973355923255658465516179806...$$

Значит ли это, что мы нашли площадь с такой точностью? Конечно, нет. Вспомним, что диаметр был измерен с точностью 0,1 см, т. е. он мог быть на самом деле равен как 1,1 см, так и 1,3 см. В этих «крайних» случаях получаем площадь

$$S_{\min} = \frac{\pi \cdot 1{,}1^2}{4} = 0{,}950... \text{ и } S_{\max} = \frac{\pi \cdot 1{,}3^2}{4} = 1{,}327...$$

Таким образом, следует записать ответ в виде $S \approx 1{,}1 \pm 0{,}2$ см^2. Относительную погрешность результата можно оценить как

$$\delta_x = \frac{0{,}2}{1{,}1} \cdot 100\% \approx 18\%.$$

Заметим, что мы не учитывали погрешность, связанную с неточностью задания иррационального числа π.

[1] На практике вместо x обычно используют среднее значение, полученное в результате серии измерений одной и той же величины.

Все практические расчёты выполняются неточно. Погрешность результата вычислений определяется в первую очередь погрешностью исходных данных.

Теперь вернёмся к расчётам с помощью компьютера. Как вы знаете из главы 4, данные записываются в память в двоичном коде ограниченной длины, при этом практически все вещественные числа хранятся неточно. При выполнении вычислений погрешности накапливаются, поэтому при сложных расчётах может получиться неверный ответ. Например, с точки зрения точности очень плохо, если ответ — это небольшое (по модулю) число, которое вычисляется как разность двух неточных больших чисел (вспомните пример с кружкой!).

Погрешность резко возрастает при делении на неточное малое по модулю число. Предположим, что нужно вычислить значение

$$x = \frac{a}{b} - \frac{c}{d},$$

причём a, b, c и d — вещественные числа, которые получены в результате вычислений с погрешностью 0,001:

$$a = 1000 \pm 0{,}001; \quad b = 0{,}002 \pm 0{,}001; \quad c = 1000 \pm 0{,}001; \quad d = 0{,}003 \pm 0{,}001.$$

Легко проверить, что вычисленное значение x может находиться в интервале от −166 667 до 750 000, т. е. относительная

погрешность превышает 300%! Такой метод расчётов **вычислительно неустойчив**: малые погрешности в исходных данных могут привести к огромным погрешностям в решении.

Подводя итог, можно выделить несколько источников погрешностей при компьютерных вычислениях:

- неточность исходных данных;
- неточность записи вещественных чисел в двоичном коде конечной длины;
- погрешности приближённого вычисления некоторых стандартных функций (например, $\sin x$ или $\cos x$);
- накопление погрешностей при арифметических действиях с неточными данными;
- собственная погрешность используемого метода (например, для приближённых методов, рассматриваемых в следующем параграфе).

Проблемы, возникающие при вычислениях с конечной точностью, изучает **вычислительная математика**, задача которой — разработать **вычислительно устойчивые методы** решения задач, при которых небольшие погрешности исходных данных мало влияют на результат. Иногда этого удаётся добиться простым изменением порядка действий или преобразованием формул.

Вопросы и задания

1. Как вы понимаете приведённое в начале параграфа высказывание К. Ф. Гаусса?
2. Какие величины можно измерять? Какие приборы для этого используются? Приведите примеры.
3. Какова цена деления у ваших наручных часов?
4. Как определить цену деления для приборов с цифровыми индикаторами? Приведите примеры.
5. Что такое абсолютная и относительная погрешности? Какое из этих значений более важно в практических задачах?
6. Что такое вычислительно неустойчивый метод?
7. Перечислите источники погрешностей при компьютерных вычислениях.
8. Какие задачи изучает вычислительная математика?

Подготовьте сообщение

а) «Абсолютная и относительная погрешность»
б) «Вычислительная устойчивость методов»

Задачи

1. Как изменятся абсолютная и относительная погрешности результата, если при вычислении площади сечения (см. задачу в тексте параграфа) в качестве π использовать число 3,14?
2. Радиус шарика R измерили с точностью 0,1 см и получили 1,2 см. Затем рассчитали его объём по формуле
$$V = \frac{4 \cdot \pi \cdot R^3}{3} = 7{,}23822947387088...$$
Запишите ответ в этой задаче, оставив нужно количество значащих цифр.
3. С помощью рулетки размеры бруса (220 см × 11 см × 10 см) измерили с точностью 1 см. Определите его объём. Запишите ответ с учётом точности полученного результата.

*4. Пётр и Павел измеряют плотность меди. У них есть медные бруски разной величины, линейка и весы. Пётр взял брусок с размерами (по результатам измерений) 2 × 2 × 2 см, а Павел — с размерами 5 × 5 × 5 см. При измерении массы брусков у Петра получилось 70 г, а у Павла — 1120 г. Погрешность измерения длины — 1 мм, погрешность измерения массы — 10 г. С какой абсолютной и относительной погрешностью определили плотность меди Пётр и Павел? Какой брусок нужно было выбирать, чтобы погрешность была наименьшей?

§ 70 Решение уравнений

Приближённые методы

На уроках математики вас учили искать решение уравнения в виде формулы, выражающей неизвестную величину через известные. Например, решение уравнения $ax + b = 1$ при $a \neq 0$ можно записать в виде $x = (1 - b)/a$. Такое решение называется **аналитическим**, оно может быть использовано для теоретического исследования (изучения влияния исходных данных на результат).

Однако не все уравнения можно (на современном уровне развития математики) решить аналитически. Иногда решение есть, но очень сложное. Например, уравнение $x = \cos x$ так просто не решается. В этом случае приходится использовать другие методы решения, например графический: построить по точкам графики функций, стоящих в левой и правой частях равенства, и посмотреть, где они пересекаются (рис. 9.3). Затем решение можно уточнять,

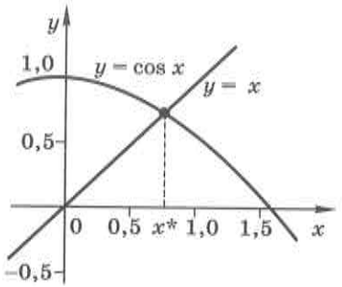


Рис. 9.3

уменьшая шаг при построении графика до получения требуемой точности.

Если нужна высокая точность, графический метод требует очень большого объёма вычислений, который имеет смысл поручить компьютеру. Однако нужно как-то учесть, что компьютер не способен «посмотреть» на график, и для уточнения решения может использовать только числовые данные. Компьютерный алгоритм решения уравнения может выглядеть примерно так:

1) выбрать отрезок $[a_0, b_0]$ для поиска решения (обычно предполагается, что на этом отрезке решение есть, и притом только одно);
2) с помощью некоторого алгоритма уточнить решение, перейдя к меньшему отрезку $[a, b]$;
3) повторять шаг 2, пока длина отрезка (содержащего решение) не станет достаточно мала.

Здесь не совсем ясно, что значит «пока длина отрезка не станет достаточно мала». Обычно задается нужная точность ε: это означает, что отклонение полученного решения от истинного x^* не должно быть больше ε. Если корень уравнения находится на отрезке $[a, b]$, то в качестве решения лучше всего взять его середину $(a+b)/2$. В этом случае погрешность будет минимальной: не больше половины длины отрезка. Поэтому цикл нужно остановить, когда длина отрезка станет меньше, чем 2ε.

Часто используется другой вариант, когда начальный отрезок не нужен, а требуется знать только одну точку вблизи решения:

1) выбрать **начальное приближение** x_0 около решения;
2) с помощью некоторого алгоритма перейти к следующему приближению x, которое находится ближе к точному решению x^*;
3) повторять шаг 2, пока на очередном шаге решение не изменится на величину, меньшую, чем допустимая погрешность ε.

Подобные методы решения уравнений называются **приближёнными**. Их суть в том, что решение последовательно уточняется до тех пор, пока не будет найдено с требуемой точностью. Поскольку при каждом уточнении выполняются одинаковые действия, можно назвать такие методы **итерационными** (от лат. *iteratio* — повторение).

Приближённые методы имеют ряд **недостатков**:

- получается приближённое решение, а не точное; это значит, что нельзя написать $x^* = 1{,}2345$, нужно использовать знак приближённого равенства: $x^* \approx 1{,}2345$ (отметим ещё раз, что практически все вычисления с дробными числами на компьютере выполняются неточно);
- мы получаем не формулу, а число, по которому невозможно оценить, как меняется решение при изменении исходных данных (сложно выявить зависимость от параметра);
- объём вычислений может быть слишком велик, часто это не позволяет использовать приближённые методы в системах реального времени;
- не всегда можно оценить погрешность результата.

Однако в некоторых практических случаях приближённые методы более полезны, чем аналитические. Они обладают следующими **достоинствами**:

- часто получение аналитического решения невозможно или требует значительных усилий, тогда как приближенные методы позволяют достаточно быстро решить задачу с заданной точностью;
- при компьютерных расчётах (с конечной точностью) вычисления по «точным» аналитическим формулам часто могут давать очень неточный результат из-за вычислительной неустойчивости метода. Нередко для таких задач (например, для решения систем линейных уравнений) специально разрабатываются приближённые методы, которые дают значительно более точное решение.

Метод перебора

Как принято в вычислительной математике, далее мы будем рассматривать уравнение общего вида $f(x)=0$, к которому можно привести любое заданное уравнение с одним неизвестным. Например, для уравнения $x=\cos x$ получаем $f(x)=x-\cos x$.

Предположим, что нужно найти решение уравнения $f(x)=0$ с точностью ε, причём известно (например, видно на графике), что решение находится справа от некоторой точки $x=a$. В этом случае можно разбить всю область, где может быть решение, на узкие полоски шириной $\delta=2\varepsilon$, и выбрать такую полоску, где график функции пересекает ось OX (рис. 9.4).

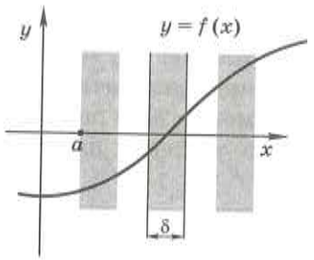


Рис. 9.4

Для того чтобы поручить решение этой задачи компьютеру, нужно ответить на два вопроса:

1) Как с помощью математических операций определить, что в полосе $[x, x+\delta]$ есть решение?
2) Что считать решением уравнения, когда такая полоса определена?

Проще ответить на второй вопрос: лучше всего взять в качестве решения середину полосы, т. е. точку $x_* = x + \varepsilon$ (в этом случае погрешность будет не больше, чем ε).

Для того чтобы определить, есть ли решение на отрезке $[x, x+\delta]$, сравним значения функции на концах этого отрезка. Рассмотрим три случая, показанные на рис. 9.5, *а–в*.

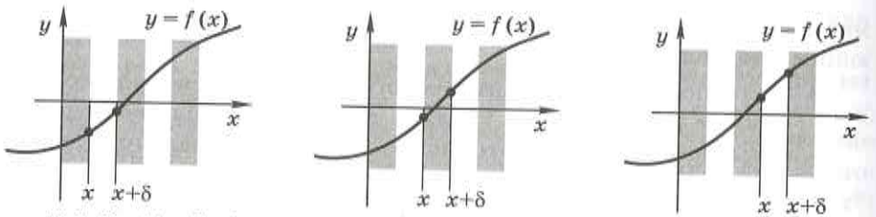


Рис. 9.5

Несложно понять, что если график пересекает ось OX, то на концах отрезка функция имеет разные знаки, т. е. $f(x) \cdot f(x+\delta) \le 0$. При этом важно, чтобы график не имел разрывов.

Если непрерывная функция имеет разные знаки на концах отрезка $[x_1, x_2]$, то в некоторой точке внутри этого отрезка она равна нулю.

Таким образом, нужно найти отрезок шириной 2ε, на концах которого функция имеет разные знаки, и взять в качестве решения его середину. Решение этой задачи при $a = 0$ может выглядеть, например, так (слева приведена программа на школьном алгоритмическом языке, а справа — на языке Паскаль):


```
алг Перебор
нач
 вещ eps, x, delta
 eps:=0.001
 x:=0
 delta:=2*eps
 нц пока f(x)*f(x+delta)>0
  x:=x+delta
 кц
 вывод 'x = ', x+eps
кон
алг вещ f(вещ x)
нач
 знач:=x-cos(x)
кон
```

```
program Perebor;
const eps=0.001;
var x, delta: real;
function f(x: real):real;
begin
 f:=x-cos(x)
end;
begin
 x:=0;
 delta:=2*eps;
 while f(x)*f(x+delta)>0 do
  x:=x+delta;
 writeln('x=',(x+eps):6:3)
end.
```

Здесь заданная точность ε хранится в виде константы *eps*, а вычисление функции $f(x)$ оформлено в виде подпрограммы-функции `f`. Поиск решения начинается с нуля (в других задачах начальное значение может быть другим). Цикл останавливается, когда для очередного отрезка получаем $f(x) \cdot f(x+\delta) \le 0$.

Нужно понимать, что при вычислениях на реальном компьютере мы не можем задавать произвольно малое значение ε. Точность ограничена типом данных, с помощью которого выполняются вычисления. В практических расчётах чаще всего используются данные с двойной точностью (англ. *double*), для которых предельная точность близка к 10^{-16} (вспомните материал § 26 и 29).

Обратите внимание, что в этом простейшем варианте программа зациклится, если справа от нуля решения нет. Подумайте, как изменить программу так, чтобы зацикливания не было.

Кроме того, в приведённом алгоритме есть и ещё одна возможная неточность: подумайте, что случится, если случайно получится $f(x) = 0$ или $f(x+\delta) = 0$. Поскольку условие цикла при этом нарушается, цикл закончится, и будет получен результат с требуемой точностью ε. Однако в этом случае можно было бы определить решение более точно, что вам и предлагается сделать самостоятельно.

Главный недостаток метода перебора — большое количество операций. Например, для решения уравнения с точностью 0,001 может понадобиться несколько тысяч вычислений значения функции $f(x)$. Поэтому сначала можно использовать перебор с достаточно большим шагом, чтобы *отделить корни*, т. е. найти интервалы, в каждом из которых есть только один корень. После

где $g \approx 9{,}81$ м/с^2 — ускорение свободного падения. Задача сводится к тому, чтобы найти два неизвестных, t и α, при которых $x = S$ и $y = H$:

$$S = v_0 \cdot t \cdot \cos\alpha, \quad H = v_0 \cdot t \cdot \sin\alpha - \frac{gt^2}{2}.$$

Время t можно сразу выразить из первого уравнения:

$$t = \frac{S}{v_0 \cos\alpha}.$$

Подставляя этот результат во второе уравнение, получаем уравнение с одним неизвестным α, которое можно привести к стандартному виду $f(\alpha) = 0$, где:

$$f(\alpha) = S \cdot tg\alpha - \frac{g \cdot S^2}{2v_0^2 \cos^2\alpha} - H = 0.$$

Решать его аналитически достаточно сложно[1], поэтому мы найдём приближённое решение. При вычислении тригонометрических функций угол измеряется в радианах, поэтому нужно искать решение в диапазоне углов α от 0 до $\pi/2$.

Мы не знаем, сколько решений имеет уравнение, поэтому изменим метод перебора так, чтобы найти все решения. Цикл `while` не будет останавливаться на первом найденном решении, а будет продолжаться, пока угол не станет больше $\pi/2$. Если в какой-то полосе есть решение, вычисляем угол в градусах и выводим его на экран. Приведём основные части программ на школьном алгоритмическом языке:

```
pi:=3.1415926
u:=0
delta:=2*eps
нц пока u<pi/2
  если f(u)*f(u+delta)<=0 то
    вывод 'Угол: ', (u+eps)*180/pi, ' градусов', нс
  все
  u:=u+delta
кц
```

и на Паскале:

```
u:=0;
delta:=2*eps;
while u<pi/2 do begin
  if f(u)*f(u+delta)<=0 then begin
    alpha:=(u+eps)*180/pi;
    writeln('Угол: ', alpha:4:1, ' градусов')
  end;
  u:=u+delta
end;
```

[1] Хотя можно, например, с помощью замены $z = \frac{1}{\cos^2\alpha}$.

В переменной u хранится угол в радианах, а в переменной *alpha* — угол в градусах. Если запустить эту программу, мы увидим, что уравнение имеет два решение — углы примерно равны 35,6° и 65,8°.

Попробуйте применить в этой же задаче метод деления отрезка пополам. Подумайте, с какими проблемами мы здесь сталкиваемся, и почему они возникают.

Повторите вычисления для начальных скоростей 10 м/с и 20 м/с и объясните полученные результаты.

Использование табличных процессоров

Для численного решения уравнений можно использовать табличный процессор, например OpenOffice Calc (или LibreOffice Calc) или Microsoft Excel. Обычно сначала строится график функции, который позволяет определить количество решений уравнения и их примерное расположение; затем используется модуль «Поиск решения». Далее мы будем рассматривать программу Calc из пакета OpenOffice, указывая на незначительные отличия Excel.

Введём исходные данные для рассмотренной задачи бросания мяча, как показано на рис. 9.8. Для того чтобы формулы выглядели более привычно, дадим ячейкам B1, B2 и B3 имена S, H и v (их можно ввести в левом верхнем углу таблицы).

В столбце A заполним ряд значений углов от 0° до 85° с шагом 5°. Для этого введём два первых значения, выделим эти ячейки и «растянем» за маркер заполнения (квадратик в правом нижнем углу выделенной части, рис. 9.9).

Рис. 9.8

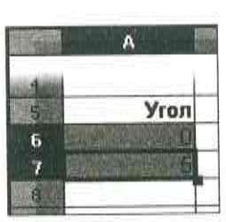


Рис. 9.9

Добавим столбцы, в которых для каждого угла будут вычисляться его значение в радианах (с помощью стандартной функции RADIANS, в русской версии Excel — РАДИАНЫ), время полёта, координата y и значение функции $f(\alpha)$ (рис. 9.10).

	A	B	C	D	E
1	Расстояние	10			
2	Разница высот	2			
3	Скорость	12			
4					
5	Угол	Угол(рад)	Время	y	f(alpha)
6	0	=RADIANS(A6)	=S/v/COS(B6)	=v*SIN(B6)*C6-9,81*C6^2/2	=D6-H
7	5				
8	10				

Рис. 9.10

Формулы в ячейках B6:E6 можно просто «растянуть» (скопировать) вниз за маркер заполнения. Обратите внимание, что в формулах мы используем имена ячеек S, H и v. Это абсолютные ссылки, не меняющиеся при копировании; например, вместо имени S можно было бы написать адрес $B$1, но это было бы менее понятно.

Теперь построим график функции $f(\alpha)$. Сначала нужно выделить данные в столбцах A и E, это можно сделать, если удерживать нажатой клавишу Ctrl. Затем строим диаграмму типа **Диаграмма XY** (в Excel — диаграмма **Точечная**) — рис. 9.11. График функции пересекает ось OX в двух точках, т. е. уравнение $f(\alpha) = 0$ имеет два решения, одно около 35°, второе — около 65°.

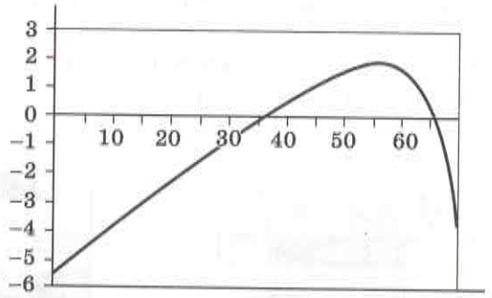


Рис. 9.11

Теперь уточним решение, используя возможности табличного процессора, в котором реализован один из приближённых методов решения уравнений. Для этого нужно знать **начальное приближение** α_0 — значение неизвестной величины, достаточно близкое к решению. По графику мы определили, что первый раз график пересекает ось OX для значения угла около 35°, поэтому можно взять $\alpha_0 = 35°$. Запишем это значение в свободную ячейку, например в H2, и добавим недостающие формулы так, чтобы получить значение функции $f(\alpha)$ в ячейке L2 (рис. 9.12).

	H	I	J	K	L
1	Угол	Угол(рад)	Время	y	f(alpha)
2	35				

Рис. 9.12

Задача подбора параметра формулируется так: «*установить в ячейке ... значение ..., изменяя значение ячейки ...*». Например, в нашем случае нужно установить в ячейке L2 значение 0, изменяя H2. Ячейка L2 называется **целевой**, потому что наша *цель* — получить в ней определённое значение (ноль). Ячейка H2 — это **изменяемая ячейка**. В главном меню выбираем пункт **Сервис, Подбор параметра** и вводим эти данные (рис. 9.13).

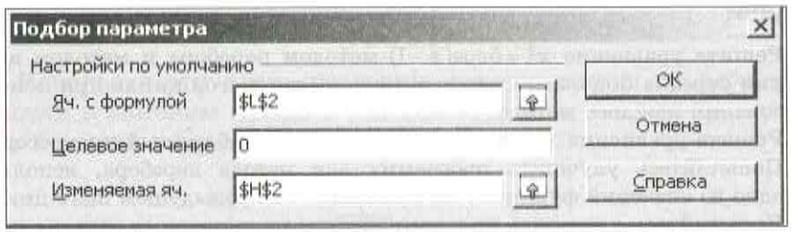


Рис. 9.13

После нажатия на кнопку **ОК** найденное решение уравнения будет записано в ячейку H2.

Как же найти второе решение? Для этого нужно выбрать другое начальное приближение, например $\alpha_0 = 70°$, в остальном порядок действий не меняется. Сделайте это самостоятельно.

Проверьте, что будет происходить при изменении начальной скорости до 10 м/с и до 20 м/с. Попробуйте объяснить эти результаты с точки зрения физики.

Вопросы и задания

1. Какие методы называются приближёнными? В каких случаях они используются?
2. Что такое итерационный метод?
3. Сравните приближённые и аналитические методы решения уравнений. В чём достоинства и недостатки каждого подхода?
4. Объясните, в чём заключается метод перебора. В чём его недостатки?
5. Как с помощью математических операций определяют, есть ли решение уравнения на заданном отрезке? В каких случаях такой подход не сработает?
6. Как избежать зацикливания в методе перебора?
7. В задаче из примера «Полёт мяча» попробуйте уточнить диапазон допустимых углов α, исключив из него углы, при которых решения заведомо нет (учтите действие силы тяжести).
7. Объясните, почему методом перебора при ширине полосы $\delta = 2\varepsilon$ можно найти решение с точностью ε.
8. Что такое отделение корней и уточнение корней?
9. Объясните изменения, сделанные в первоначальной программе для решения уравнения методом перебора, которые позволили в одном цикле найти все решения на заданном отрезке.
10. Объясните, как работает метод деления отрезка пополам. Сравните его с методом перебора.

Задачи

1. Решите уравнение $x^2 = 5\cos(x-1)$ методом перебора и методом деления отрезка пополам. Сравните количество шагов цикла при использовании каждого метода.
2. Решите уравнение $x^2 = 5\cos(x-1)$, используя табличный процессор.

*3. Попытайтесь улучшить программу для метода перебора, используя одно из значений функции, вычисленных на предыдущем шаге цикла.

*4. Попытайтесь улучшить программу для метода деления отрезка пополам, используя одно из значений функции, вычисленных на предыдущем шаге цикла.

5. Экспериментально (пробуя различные значения ε) определите, с какой точностью можно найти решение уравнения $x^2 = 5\cos(x-1)$ в вашей среде программирования.
6. Решите задачу из примера «Полёт мяча» с помощью собственной программы, а затем с помощью табличного процессора. Обсудите преимущества и недостатки этих способов решения.

*7. Используя замену $z = \dfrac{1}{\cos^2\alpha}$, постройте аналитическое решение уравнения из примера «Полёт мяча». Для практических вычислений используйте электронные таблицы. Сравните точное и численное решения.

§ 71 Дискретизация

Вычисление длины кривой

Определим длину траектории L, по которой летит шарик, брошенный под углом к горизонту (см. задачу в § 70). Это не так просто, потому что траектория — кривая линия.

Постараемся как-то свести задачу к более простой, которую мы умеем решать. Если бы линия состояла из отдельных отрезков, её длину можно было бы точно вычислить с помощью теоремы Пифагора. Например, длина ломаной на рис. 9.14 равна:

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}\,.$$

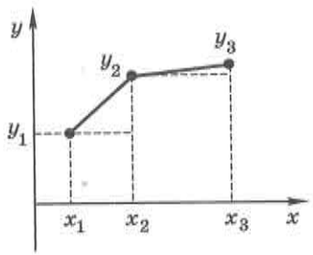


Рис. 9.14

Используя эту идею, разделим кривую линию на небольшие участки и заменим каждый участок отрезком так, чтобы получилась ломаная (штриховая линия на рис. 9.15).

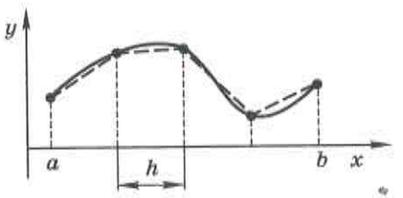


Рис. 9.15

Обычно разбивают исходный отрезок $[a, b]$ (на котором нужно определить длину кривой) на равные участки длины h. Конечно, длина ломаной отличается от длины кривой, но естественно предположить, что при уменьшении шага разбиения h эта разница будет уменьшаться.

Обратим внимание, что мы фактически выполнили **дискретизацию** — представили кривую в виде набора точек, при этом информация о поведении функции между этими точками была потеряна (вспомните оцифровку звука!). Величина h называется **шагом дискретизации**.

Подведём итоги:

- дискретизация позволяет представить задачу в виде, пригодном для компьютерных расчётов;
- при дискретизации часть информации теряется, поэтому все методы, основанные на дискретизации, — приближённые, они решают задачу с некоторой погрешностью;
- чтобы уменьшить погрешность, нужно уменьшать шаг дискретизации (увеличивать количество точек), но при этом возрастёт объём (и время) расчётов;
- при выборе малого шага дискретизации на результат могут сильно влиять погрешности вычислений, вызванные неточностью представления вещественных чисел в памяти компьютера (см. § 29).

Теперь можно составить программу. Будем считать, что константы (или переменные) a, b и h задают границы отрезка и шаг дискретизации. Тогда основная часть программы может выглядеть так на школьном алгоритмическом языке:

```
x:=a; L:=0
нц пока x<b
  y1:=f(x)
  y2:=f(x+h)
  L:=L+sqrt(h*h+(y1-y2)*(y1-y2))
  x:=x+h
кц
вывод 'Длина кривой ', L:10:3
```

и на Паскале:

```
x:=a; L:=0;
while x<b do begin
  y1:=f(x);
  y2:=f(x+h);
  L:=L+sqrt(h*h+(y2-y1)*(y2-y1));
  x:=x+h
end;
writeln('Длина кривой ', L:10:3);
```


Возвращаясь к задаче с шариком, вспомним, что его движение описывается уравнениями:

$$x = v_0 \cdot t \cdot \cos\alpha, \quad y = v_0 \cdot t \cdot \sin\alpha - \frac{gt^2}{2}.$$

Если выразить время из первого уравнения и подставить во второе, получается

$$y = f(\alpha) = x \cdot tg\alpha - \frac{g \cdot x^2}{2v_0^2 \cos^2\alpha}.$$

Это и есть функция, график которой нас интересует. Написать полную программу вы уже можете самостоятельно. Проверьте её работу для разных значений исходных данных (скорости v_0 и угла вылета α).

Вычисление площадей фигур

Применим метод дискретизации в другой достаточно сложной задаче — для вычисления площади фигуры. Пусть фигура, площадь которой нас интересует, ограничена графиками функций $y = f_1(x)$ и $y = f_2(x)$ (рис. 9.16).

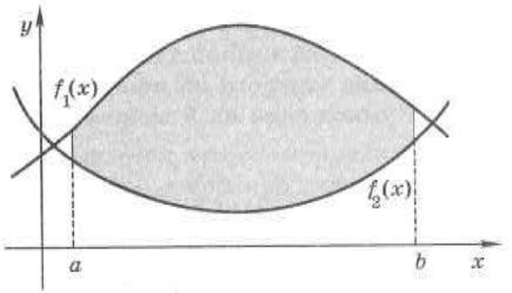


Рис. 9.16

В некоторых простых случаях (но далеко не всегда!) площадь такой фигуры можно вычислить аналитически с помощью методов высшей математики. Мы же будем использовать приближённые методы, применять которые значительно проще.

Как и при вычислении длины кривой, применим *дискретизацию* — разделим фигуру на отдельные полоски и заменим каж-

дую полоску на другую фигуру, для которой мы можем легко найти площадь, например на *прямоугольник* (рис. 9.17).

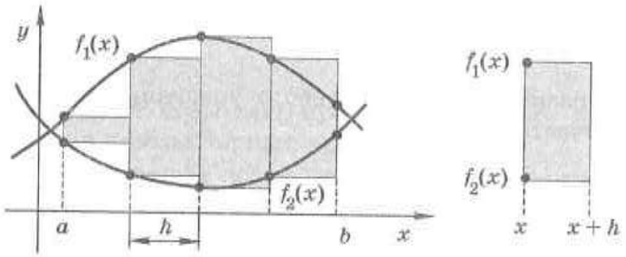


Рис. 9.17

Понятно, что площадь исходной фигуры в общем случае не совпадает с суммой площадей получившихся прямоугольников. Так и должно было быть, ведь при дискретизации информация о поведении функций между точками отсчёта была утеряна. Однако при уменьшении шага дискретизации h эта разница будет уменьшаться.

На рисунке 9.17 высота прямоугольника рассчитывается как разность значений функций на левой границе отрезка (можно использовать и правую границу). На практике обычно вычисляют высоту *в середине отрезка* (в точке $x + h/2$), тогда площадь прямоугольника будет более близка к площади полосы исходной фигуры. Заметим, что ширина каждого из прямоугольников равна h, поэтому в программе умножение на h можно выполнить в конце цикла:

```
S:=0; x:=a
нц пока x<b
  S:=S+f1(x+h/2)-f2(x+h/2)
  x:=x+h
кц
S:=h*S
вывод 'Площадь ', S:8:3
```

```
S:=0; x:=a;
while x<b do begin
  S:=S+f1(x+h/2)-f2(x+h/2);
  x:=x+h
end;
S:=h*S;
writeln('Площадь ', S:8:3);
```

Этот метод называется *методом прямоугольников*.

Вместо прямоугольника для замены удобно применять ещё одну известную фигуру, для которой легко считается площадь, — *трапецию* (рис. 9.18).

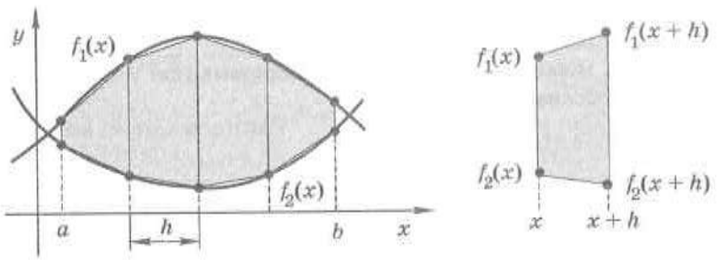


Рис. 9.18

Площадь трапеции равна произведению полусуммы её оснований на высоту, т. е. для элементарной трапеции с левой границей в точке x получаем выражение для площади:

$$\frac{h}{2}\cdot\left[f_1(x) - f_2(x) + f_1(x+h) - f_2(x+h)\right].$$

Простейшая программа выглядит так:

```
S:=0; x:=a
нц пока x<b
  S:=S+f1(x)-f2(x)
  S:=S+f1(x+h)-f2(x+h)
  x:=x+h
кц
S:=h*S/2
вывод 'Площадь ', S:8:3
```

```
S:=0; x:=a;
while x<b do begin
  S:=S+f1(x)-f2(x)
     +f1(x+h)-f2(x+h);
  x:=x+h
end;
S:=h*S/2;
writeln('Площадь ', S:8:3);
```

Этот метод называется *методом трапеций*.

Заметим, что правое основание очередной трапеции совпадает с левым основанием следующей, поэтому можно немного изменить программу, чтобы на каждом шаге цикла вычислялась разность функций только в одной точке. Сделайте это самостоятельно.

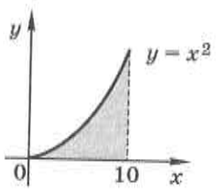
Вопросы и задания

1. Что такое дискретизация?
2. Что даёт дискретизация в рассмотренных в параграфе задачах?
3. Что такое шаг дискретизации? Как должна быть связана его величина с длиной отрезка $[a, b]$?
4. Как зависят точность и время вычислений от выбора шага дискретизации?

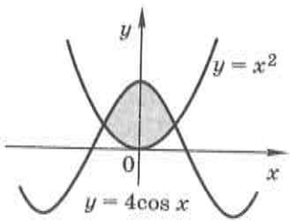
5. Почему методы, основанные на дискретизации, всегда дают приближённый результат?
6. Подумайте, можно ли выполнять дискретизацию с переменным шагом. Ответ обоснуйте.

Задачи

1. Измените программу для вычисления длины кривой так, чтобы на каждом шаге цикла вычислять только одно значение функции.
2. Найдите приближённо длину параболы $y = x^2$ на отрезке $x \in [0, 10]$.
3. Для примера, разобранного в § 70, вычислите длину траектории движения шарика для углов вылета 35,5° и 65,8°. Сравните полученные результаты. Постройте эти траектории с помощью табличного процессора.
4. Решите задачу 3 при разных значениях шага. Какой шаг вы рекомендуете выбрать для этого случая? Почему?
5. В чём заключается дискретизация при вычислении площади?
*6. Измените программу для вычисления площади методом трапеций так, чтобы повторно не вычислять одни и те же величины.
7. Найдите площадь фигуры, ограниченной параболой $y = x^2$ и осью OX, на отрезке $x \in [0, 10]$.



*8. Найдите площадь фигуры, ограниченной графиками функций $y = x^2$ и $y = 4\cos x$.



*9. Найдите площадь фигуры, ограниченной эллипсом

$$\frac{x^2}{4} + \frac{y^2}{9} = 1.$$

*10. Найдите с помощью приближённых методов площадь круга радиуса $R = 2$. Используя это значение, из формулы $S = \pi \cdot R^2$ определите приближенно число π.

§ 72 Оптимизация

Что такое оптимизация?

Оптимизация — это поиск наилучшего (оптимального) решения задачи в заданных условиях.

С точки зрения математики, цель оптимизации — выбрать неизвестную величину x (или несколько неизвестных величин — массив) наилучшим образом.

Чтобы задача оптимизации была корректной, нужно определить **целевую функцию** $f(x)$. Оптимальным называется такое решение, при котором целевая функция достигает максимума (если это «доходы», «прибыль») или минимума («расходы», «потери»):

$$f(x) \to \max \quad \text{или} \quad f(x) \to \min.$$

Чтобы задача оптимизации стала осмысленной, нужно ввести **ограничения**. Например, пусть человек хочет построить загородный дом за минимальную цену (здесь целевая функция — это общая цена строительства, нужно сделать её минимальной). Очевидно, что лучшее решение — не строить дом вообще, при этом расходы будут равны нулю. Такое «оптимальное» решение получено потому, что мы не задали ограничения (например, нужен двухэтажный дом с гаражом, балконом и камином).

Локальные и глобальный минимумы

По традиции в теории оптимизации рассматривают задачу поиска минимума. Если нужно найти максимум, просто меняют знак функции: значение функции $f(x)$ максимально там, где значение функции $(-f(x))$ минимально.

В математике различают **локальный** («местный») и **глобальный** («общий») **минимумы**. В точках x_1, x_2 и x_3 функция, график которой показан на рис. 9.19, имеет локальные минимумы, это значит, что слева и справа от этих точек функция возрастает.

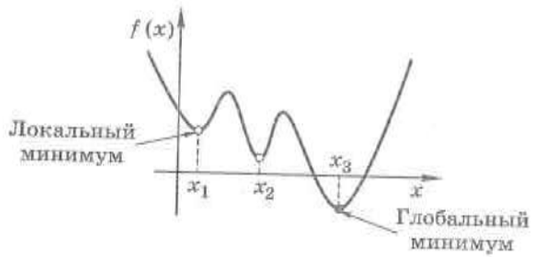


Рис. 9.19

Минимум в точке x_3 — глобальный, потому что здесь функция имеет наименьшее значение во всей рассматриваемой области.

Очевидно, что нас всегда интересует глобальный минимум. Однако большинство существующих методов оптимизации предназначено именно для поиска локальных минимумов[1] вблизи заданной начальной точки (начального приближения). Можно представить себе, что график функции — это срез поверхности, на которую устанавливается шарик в некоторой начальной точке; куда этот шарик скатится, такой минимум и будет найден.

Результат локальной оптимизации зависит от выбранного начального приближения.

Метод дихотомии

Дихотомия (греч. διχοτομια — деление надвое) — это метод поиска минимума функции, который очень напоминает метод деления отрезка пополам (бисекции). Пусть дана непрерывная функция $f(x)$, имеющая на отрезке $[a, b]$ один минимум в точке x^*. Нужно найти это значение с заданной точностью ε.

Как и в методе бисекции, мы последовательно «сжимаем» отрезок, пока его ширина не будет достаточно мала (меньше, чем 2ε). На каждом шаге (рис. 9.20):

1) вычисляется середина отрезка $c = \dfrac{a+b}{2}$;

[1] Для некоторых типов функций существуют методы глобальной оптимизации, но они сложны и выходят за рамки школьного курса.

2) вычисляются координаты двух точек, симметричных относительно середины: $x_1 = c - r$ и $x_2 = c + r$, где $0 < r < (b-a)/2$ — некоторое число;

3) сравниваются значения функции в этих точках: если $f(x_1) > f(x_2)$, то минимум функции находится на отрезке $[x_1, b]$, поэтому отрезок $[a, x_1]$ можно отбросить — переместить левую границу в точку x_1; если $f(x_1) < f(x_2)$, то правая граница отрезка перемещается в точку x_2.

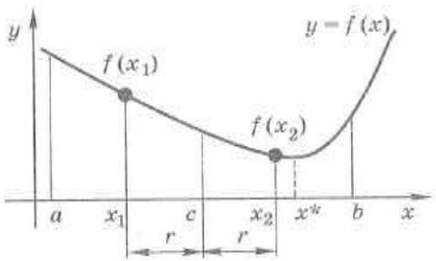


Рис. 9.20

Остаётся один вопрос: как выбирать r на каждом шаге? Проще всего вычислять его по формуле $r = k \cdot (b-a)$, где k — постоянный коэффициент $(0 < k < 0{,}5)$. Тогда ширина отрезка на следующем шаге будет равна

$$\frac{b-a}{2} + k \cdot (b-a) = (0{,}5 + k) \cdot (b-a),$$

т. е. составит $0{,}5 + k$ от первоначальной ширины. Для ускорения поиска выгодно, чтобы это отношение было как можно меньше, т. е. чтобы коэффициент k был возможно ближе к нулю (ноль выбирать нельзя, потому что при этом точки x_1 и x_2 совпадут, и метод не будет работать). Программа может выглядеть примерно так:

```
k:=0.01
delta:=2*eps
нц пока b-a>delta
  r:=k*(b-a)
  c:=(a+b)/2
  x1:=c-r
  x2:=c+r
  если f(x1)>f(x2) то
      a:=x1
```

```
k:=0.01;
delta:=2*eps;
while b-a>delta do begin
  r:=k*(b-a);
  c:=(a+b)/2;
  x1:=c-r;
  x2:=c+r;
  if f(x1)>f(x2) then
      a:=x1
```

```
  иначе b:=x2                    else b:=x2
  все                          end;
кц                             writeln('x =', (a+b)/2:10:3);
вывод 'x =', (a+b)/2:10:3
```

В качестве упражнения можно исследовать работу этой программы при разных значениях k, подсчитав для каждого варианта количество шагов цикла, которое потребуется для получения решения с заданной точностью.

Существует вариант метода дихотомии, при котором на каждом шаге цикла нужно вычислять только одно значение функции (второе «переходит» с предыдущего шага). В этом случае нужно выбирать коэффициент k так, чтобы выполнялось равенство $0{,}5 + k = \frac{1}{\phi}$, где ϕ — отношение «золотого сечения»:

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1{,}618\ldots$$

Пример: оптимальная раскройка листа

Рассмотрим пример практической задачи оптимизации. В углах квадратного листа железа, сторона которого равна 1 м, вырезают четыре квадрата со стороной x. Затем складывают получившуюся развёртку (по штриховым линиям на рис. 9.21), сваривают швы и таким образом получается бак.

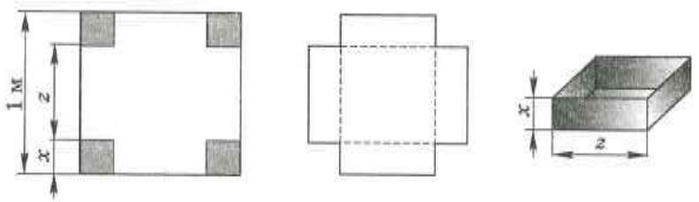


Рис. 9.21

Требуется выбрать размер выреза x так, чтобы получился бак наибольшего объёма.

Для того чтобы грамотно поставить задачу оптимизации, нужно:

1) определить целевую функцию: в данном случае выразить объём бака через неизвестную величину x;
2) задать ограничения на возможные значения x.

Легко видеть, что основание получившегося бака — это квадрат со стороной z (см. рис. 9.21), а его высота равна x. Величина z зависит от x и равна $z = 1 - 2x$, поэтому объём бака вычисляется по формуле $V(x) = x(1-2x)^2$, это и есть целевая функция, для которой нужно найти максимум.

Понятно, что x не может быть меньше нуля. Вместе с тем x не может быть больше, чем половина стороны исходного листа (0,5 м), поэтому ограничения запишутся в виде двойного неравенства: $0 \le x \le 0{,}5$. Заметим, что при $x = 0$ и $x = 0{,}5$ объём бака равен нулю (в первом случае равна нулю высота, во втором — площадь основания).

Таким образом, нужно искать максимум целевой функции $v(x) = x(1-2x)^2$ на отрезке $[0; 0{,}5]$. Для этого можно использовать метод дихотомии (сделайте это самостоятельно). Не забудьте, что приведённый выше вариант программы рассчитан на поиск минимума, а в этой задаче нужно найти максимум.

Использование табличных процессоров

В стандартной поставке OpenOffice Calc модуль оптимизации работает только для линейных функций. Для того чтобы решить рассмотренную выше задачу с баком, нужно использовать модуль Solver for Nonlinear Programming (он входит в стандартную поставку последних версий LibreOffice) и в параметрах модуля оптимизации выбрать один из методов нелинейной оптимизации. В табличном процессоре Excel оптимизация выполняется с помощью стандартной надстройки «Поиск решения».

Сначала построим график функции, как мы делали это при решении уравнения (рис. 9.22).

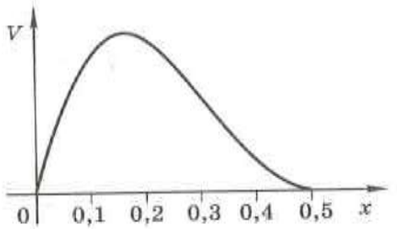


Рис. 9.22

По этому графику видно, что функция имеет единственный максимум в районе точки $x_0 = 0{,}2$. Это значение можно выбрать в качестве начального приближения для поиска.

Выделим в таблице две ячейки (например, E2 и F2), в первую запишем начальное приближение (0,2), а во вторую — формулу для вычисления объёма для этого значения x (рис. 9.23).

	E	F
1	x	Объем
2	0,200	0,072

Рис. 9.23

Задача оптимизации формулируется в виде *«найти максимум (или минимум) целевой функции в ячейке ..., изменяя значения ячеек ... при ограничениях ...»*. В нашей задаче целевая ячейка — F2 (нужно найти её максимум), изменяемая ячейка — E2.

Выбираем в главном меню пункт **Сервис**, **Поиск решения**, в появившемся окне вводим адреса целевой и изменяемой ячеек (для этого можно установить курсор в нужное поле ввода и щёлкнуть по ячейке) и ограничения (рис. 9.24).

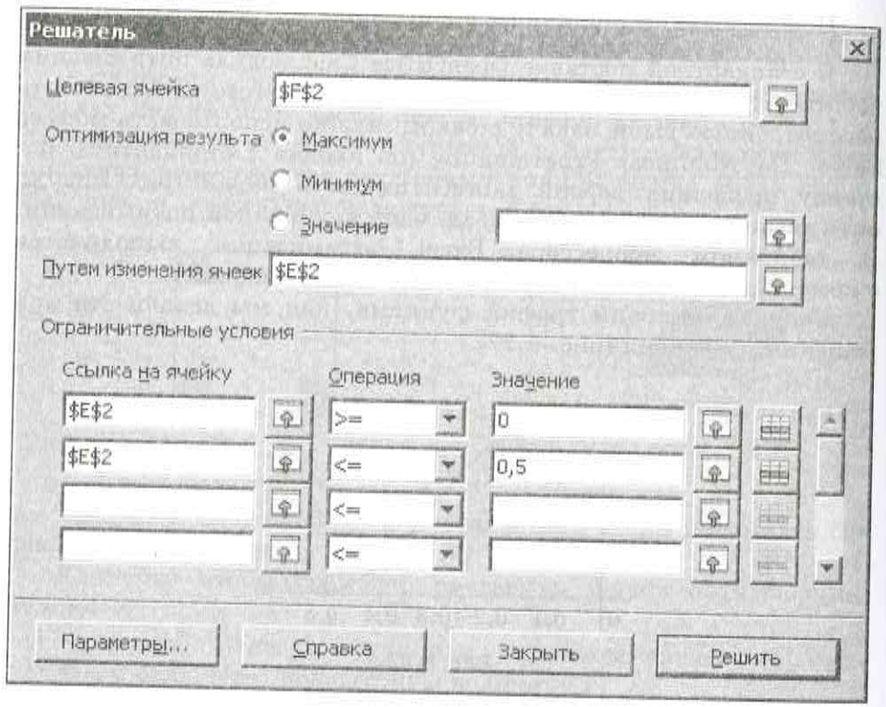


Рис. 9.24

После щелчка на кнопке **Решить** в ячейку E2 будет записано оптимальное значение x, а в целевую ячейку — соответствующее (максимальное) значение объёма.

Надстройка «Поиск решения» позволяет:

- находить максимум или минимум целевой функции;
- решать уравнения, задавая желаемое значение целевой функции;
- использовать несколько изменяемых ячеек и диапазонов, например запись A2:A6;B15 в списке изменяемых ячеек означает «изменять все ячейки диапазона A2:A6 и ячейку B15»;
- использовать ограничения типа «меньше или равно», «больше или равно», «равно», «целое» и «двоичное» (только 0 или 1).

Кнопка **Параметры** позволяет опытным пользователям менять настройки алгоритма оптимизации.

Вопросы и задания

1. Что такое оптимизация?
2. Что такое целевая функция?
3. Какое решение называется оптимальным?
4. Почему выражение «самый оптимальный» безграмотно?
5. Что можно сказать о рекламной фразе «Этот крем обеспечивает оптимальный цвет лица»?
6. Зачем нужны ограничения в задаче оптимизации?
7. В чём разница между понятиями «локальный минимум» и «глобальный минимум»?
8. Что такое начальное приближение?
9. Почему результат решения задачи оптимизации чаще всего зависит от выбора начального приближения?
10. Объясните принцип работы метода дихотомии.
11. Обязательно ли при использовании метода дихотомии брать пробные точки симметрично относительно середины отрезка? Ответ обоснуйте.
12. Когда метод дихотомии не будет работать (может выдать неверный ответ)?
13. Подумайте, можно ли задачу решения уравнения сформулировать как задачу оптимизации.

Подготовьте сообщение:

а) «Оптимизация для функции двух переменных»
б) «Оптимизация методом случайного поиска»
в) «Программное обеспечение для решения задач оптимизации»

Задачи

1. Примените метод дихотомии для решения задачи оптимальной раскройки, которая разобрана в параграфе. Решите задачу, используя разные значения коэффициента k, и постройте график зависимости количества шагов цикла от k.

*2. Напишите программу, которая реализует метод «золотого сечения» (на каждом шаге вычисляется только одно значение функции).

*3. Банка имеет форму цилиндра[1], полная площадь её поверхности (боковая поверхность и два круга-основания) равна 100 см^2. Определите радиус и высоту банки, которая при этих условиях имеет максимальный объём.

*4. Банка имеет форму цилиндра, её объём равен 500 см^3. Определите радиус и высоту банки, которая при этих условиях имеет минимальную площадь полной поверхности.

5. Некоторая фирма хочет провести рекламную кампанию в газетах. Данные о цене рекламного объявления и тиражах газет внесены в таблицу:

Газета	Тираж	Цена одного объявления, руб.	Кол-во объявлений	Расходы, руб.	Охват
Ведомости	10 000	1000	1	1000	10 000
Туризм	3500	570	2	1140	7000
Спорт	6000	700	3	2100	18 000
Правда	20 000	1250	4	5000	80 000
Всего				9240	115 000

В каждую газету нужно дать не менее одного и не более 6 объявлений. С помощью надстройки «Поиск решения» табличного процессора определите, сколько объявлений нужно дать в каждую газету, чтобы обеспечить общий охват не менее 200 000 человек и при этом израсходовать как можно меньше денег.

6. В условиях задачи 5 определите, сколько объявлений нужно дать в каждую газету, чтобы обеспечить наибольший общий охват и при этом израсходовать не более 15 000 рублей.

[1] Задачи 3 и 4 предложены В. Я. Лаздиным.

§ 73 Статистические расчёты

Статистика — это наука, которая изучает методы обработки и анализа больших массивов данных.

Табличные процессоры стали незаменимым инструментом для решения этих задач. И Excel, и Calc содержат несколько десятков встроенных статистических функций.

Свойства ряда данных

Простейшая задача статистики — изучить свойства одного ряда данных. **Ряд данных** X длиной n — это множество значений $x_1, x_2, \ldots, x_n$. Для ряда можно определить количество элементов, их сумму, среднее значение, минимальное и максимальное значения и т. д. Далее мы приводим как английские названия функций (для Calc), так и русские (для русской версии Excel).

Пусть ряд данных записан в ячейках A1:A20. Его сумма, среднее, минимальное и максимальное значения могут быть определены с помощью следующих формул:

сумма:	=SUM(A1:A20)	=СУММ(A1:A20)
среднее:	=AVERAGE(A1:A20)	=СРЗНАЧ(A1:A20)
минимальное:	=MIN(A1:A20)	=МИН(A1:A20)
максимальное:	=MAX(A1:A20)	=МАКС(A1:A20)

Функции SUM (русское название — СУММ) и AVERAGE (СРЗНАЧ) учитывают только числа в указанном диапазоне (без учёта пустых и текстовых ячеек). Количество числовых значений можно подсчитать с помощью функции COUNT (СЧЁТ), например:

=COUNT(A1:A20) =СЧЁТ(A1:A20)

С помощью функции COUNTIF (СЧЕТЕСЛИ) можно подсчитать число элементов ряда, удовлетворяющих некоторому условию. Например, если в диапазоне A1:A20 записаны школьные отметки, формула

=COUNTIF(A1:A20;"=5")

вычисляет число пятёрок, а формула

=COUNTIF(A1:A20;">3")

вычисляет общее число четвёрок и пятёрок (всех ячеек, значения которых больше 3).

Более сложная характеристика ряда — **среднеквадратическое отклонение** или **стандартное отклонение** σ_x, с помощью которого оценивается «разброс» значений $x_1, x_2, \ldots, x_n$ относительно среднего значения ряда $\overline{x}$:

$$\sigma_x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}.$$

Среднеквадратическое отклонение — это неотрицательная величина (подумайте почему). Чем больше σ_x, тем больше разброс значений относительно среднего. Для вычисления стандартного отклонения в табличных процессорах используется функция STDEVP (СТАНДОТКЛОНП):

=STDEVP(A1:A20) =СТАНДОТКЛОНП(A1:A20)

Условные вычисления

Как вы знаете, в программировании важнейшую роль играют условные операторы (ветвления), позволяющие выбирать один из двух (или нескольких) вариантов обработки данных. В табличных процессорах тоже возможны **условные вычисления**, при которых в ячейку заносится то или иное значение в зависимости от выполнения какого-то условия.

Предположим, что в книжном интернет-магазине «Бука» доставка покупок бесплатна для тех, кто сделал заказ на сумму более 500 рублей, а для остальных доставка стоит 20% от суммы (рис. 9.25).

	A	B	C
1	**Заказ**	**Сумма**	**Доставка**
2	1234	256 руб.	51 руб.
3	1345	128 руб.	26 руб.
4	1456	1 024 руб.	0
5	1565	512 руб.	0
6	1576	345 руб.	69 руб.

Рис. 9.25

Таким образом, есть два варианта вычисления стоимости доставки, поэтому в формулах столбца C нужно использовать ветвление. Например, алгоритм вычисления значения ячейки C2 может выглядеть так: «если B2>500, записать в ячейку 0, иначе записать значение B2*0,2». В программе на Паскале мы бы записали:

```
if B2>500 then
     C2:=0
else C2:=B2*0.2;
```

В табличных процессорах для условных вычислений используют функцию IF (ЕСЛИ):

=IF(B2>500;0;B2*0,2) =ЕСЛИ(B2>500;0;B2*0,2)

У этой функции три аргумента, разделённые точками с запятой:

1) условие (B2>500);
2) значение ячейки в том случае, когда условие истинно (0);
3) значение ячейки в том случае, когда условие ложно (B2*0,2).

Первый аргумент может быть сложным условием, которое строится с помощью функций NOT (НЕ, отрицание), AND (И, логическое умножение) и OR (ИЛИ, логическое сложение). Например, если бесплатная доставка распространяется только на заказы, у которых номер меньше 1500 и сумма больше 500 рублей, в ячейку C2 нужно записать такую формулу:

=IF(AND(A2<1500;B2>500);0;B2*0,2)

Здесь использовано сложное условие AND(A2<1500;B2>500), которое истинно только при одновременном выполнении обоих простых условий: A2<1500 и B2>500.

Второй и третий аргументы функции IF могут содержать вложенные вызовы этой функции. Пусть, например, для заказов стоимостью более 200 рублей (но не больше 500) стоимость доставки составляет 10% от суммы:

```
if B2>500 then
  C2:=0
else
  if B2>200 then
     C2:=B2*0.1
 else C2:=B2*0.2;
```

В табличном процессоре этот алгоритм запишется в виде:

=IF(B2>500;0;IF(B2>200;B2*0,1;B2*0,2))

Связь двух рядов данных

Одна из задач обработки данных — установить взаимосвязь между величинами, процессами, явлениями. Пусть существуют два ряда данных одинаковой длины:

$$x_1, x_2, \ldots, x_n \quad \text{и} \quad y_1, y_2, \ldots, y_n.$$

Например, первый ряд — это температура воздуха за n последних дней, а второй ряд — значения атмосферного давления в те же дни. Требуется определить, есть ли связь между этими рядами, и оценить, насколько она сильная.

Для решения этой задачи чаще всего используется **коэффициент корреляции** (англ. *correlation* — взаимоотношение, связь):

$$\rho_{xy} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sigma_x \cdot \sigma_y}.$$

Здесь $\overline{x}$ и $\overline{y}$ — средние значения рядов, а σ_x и σ_y — их среднеквадратические отклонения.

Величина ρ_{xy} — это безразмерный коэффициент[1], причём можно показать, что всегда $-1 \le \rho_{xy} \le 1$. Если $\rho_{xy} > 0$, то увеличение значения x (в среднем) приводит к увеличению y; если же $\rho_{xy} < 0$, то при увеличении x значение y чаще всего уменьшается.

Чем больше модуль ρ_{xy}, тем сильнее связь между двумя величинами. При $\rho_{xy} = -1$ или $\rho_{xy} = 1$ они строго связаны линейной зависимостью $y = kx + b$, где k и b — некоторые числа. В случае $\rho_{xy} = -1$ эта зависимость убывающая ($k < 0$), а при $\rho_{xy} = 1$ — возрастающая ($k > 0$).

Считается, что между x и y есть сильная связь, если $|\rho_{xy}| > 0{,}5$. При меньших значениях ρ_{xy} делать какие-то далеко идущие выводы не следует (связь слабая или не обнаружена).

Для вычисления коэффициента корреляции в табличных процессорах используется функция CORREL (КОРРЕЛ):

=CORREL(A1:A20;B1:B20) =КОРРЕЛ(A1:A20;B1:B20)

[1] Попробуйте доказать это самостоятельно.

Обратите внимание, что у этой функции два аргумента (два ряда данных одинаковой длины), адреса двух диапазонов отделяются точкой с запятой.

Нужно учитывать, что коэффициент корреляции лучше всего обнаруживает линейную зависимость. Если связь есть, но она далека от линейной, коэффициент корреляции может быть невысок. В таких случаях для установления связи нужно использовать более сложные методы, которые мы здесь рассматривать не будем.

Вопросы и задания

1. Что изучает статистика? Как вы думаете, в чём её задача?
2. Как влияют пустые ячейки в электронной таблице на результат работы функций COUNT и AVERAGE?
3. Чем отличаются функции COUNT и COUNTIF?
4. Что показывает среднеквадратическое отклонение?
5. Что показывает коэффициент корреляции?
6. Для двух рядов коэффициент корреляции равен (–0,5). Что можно сказать о возможной связи этих рядов между собой?
7. Для двух рядов коэффициент корреляции равен 0,1. Что можно сказать о возможной связи этих рядов между собой?

Задачи

1. В электронной таблице значение формулы =SUM(B1:B2) равно 5. Чему равно значение ячейки B3, если значение формулы =AVERAGE(B1:B3) равно 3?
2. В электронной таблице значение формулы =SUM(C3:E3) равно 12. Чему равно значение формулы =AVERAGE(C3:F3), если значение ячейки F3 равно 4?
3. В электронной таблице значение формулы =SUM(A2:D2) равно 12. Чему равно значение формулы =AVERAGE(A2:E2), если значение ячейки E2 равно 13?
4. В электронной таблице значение формулы =AVERAGE(A6:C6) равно (–2). Чему равно значение формулы =SUM(A6:D6), если значение ячейки D6 равно 6?
5. В электронной таблице значение формулы =AVERAGE(B5:E5) равно 10. Чему равно значение формулы =SUM(B5:F5), если значение ячейки F5 равно 1?
6. В электронной таблице значение формулы =AVERAGE(A1:C1) равно 6. Чему равно значение ячейки D1, если значение формулы =SUM(A1:D1) равно 7?

7. В электронной таблице значение формулы =AVERAGE(A3:D4) равно 6. Чему равно значение формулы =AVERAGE(A3:C4), если значение формулы =SUM(D3:D4) равно 6?
8. В электронной таблице значение формулы =AVERAGE(C2:D5) равно 4. Чему равно значение формулы =SUM(C5:D5), если значение формулы =AVERAGE(C2:D4) равно 6?
*9. Как изменится значение ячейки C3, если после ввода формул переместить содержимое ячейки B2 в B3?

	A	B	C
1	1	2	
2	2	6	=COUNT(A1:B2)
3			=AVERAGE(A1:C2)

10. Доставка товара в фирме «Рога и копыта» стоит 200 рублей, если в доме есть лифт. Если лифта нет, подъём на каждый этаж стоит 200 рублей:

	A	B	C	D
1	**Заказ**	**Этаж**	**Лифт**	**Доставка**
2	12	5	нет	1000
3	34	2	да	200
4	56	8	да	200

Какую формулу нужно записать в ячейку D2?

11. При приёме на работу претенденты проходят два тура собеседования. В каждом туре выставляется оценка от 0 до 100 баллов. На работу принимаются те, кто в каждом туре набрал не менее 80 баллов.

	A	B	C	D
1	**Фамилия**	**1-й тур**	**2-й тур**	**Принят**
2	Иванов	80	80	да
3	Петров	90	70	нет
4	Сидоров	85	90	да

Какую формулу нужно записать в ячейку D2?

12. Решите задачу 11 при условии, что на работу принимаются все, кто набрал 90 баллов хотя бы в одном туре собеседования.
13. Сниженный тариф на телефонные разговор — 2 рубля за минуту разговора — действует в рабочие дни после 20 часов и в выходные дни. Обычный тариф — 5 рублей за минуту. Дни в таблице нумеруются с 1 до 7 (1 — понедельник).

	A	B	C	D
1	**Код**	**Время**	**День недели**	**Тариф, руб.**
2	12	21:12:20	2	2
3	34	17:23:50	1	5
4	56	10:21:42	7	2

Какую формулу нужно записать в ячейку D2? Момент времени 20:00:00 в формуле нужно записывать как TIME(20;0;0).

14. При покупке на сумму более 1000 рублей в магазине владельцам дисконтных карт предоставляется скидка 5%.

	A	B	C	D
1	**Код**	**Цена, руб.**	**Дисконтная карта**	**Со скидкой, руб.**
2	12	1 200	нет	1 200
3	34	1 450	да	1 378
4	56	750	да	750

Какую формулу нужно записать в ячейку D2?

15. Во время уценки цена всех товаров, которых хранятся на складе более 6 месяцев, снижается на 20% (если цена товара больше 1000 рублей) или на 10% (если цена меньше 1000 рублей).

	A	B	C	D
1	**Код**	**Цена, руб.**	**Хранится (месяцев)**	**Новая цена, руб.**
2	12	1 200	3	1 200
3	34	1 450	7	1 160
4	56	750	12	675

Какую формулу нужно записать в ячейку D2?

§ 74
Обработка результатов эксперимента

Зачем это нужно?

Многие практические задачи связаны с проведением эксперимента, в результате которого исследователь с помощью измерительных приборов получает массивы данных. Затем эти данные необходимо обработать, для того чтобы выявить закономерности, подтвердить или опровергнуть выводы теории и т. п.

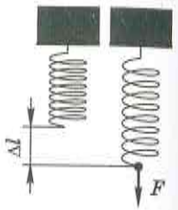


Рис. 9.26

Например, с помощью динамометра и линейки можно экспериментально определить жёсткость пружины (рис. 9.26). Для этого используется закон Гука, связывающий приложенную силу F, жёсткость пружины k и её удлинение Δl линейной зависимостью:

$$F = k \cdot \Delta l.$$

Определив жёсткость пружины k, мы сможем вычислять её удлинение при любой нагрузке (в пределах действия закона Гука), не выполняя новых измерений.

Величина k зависит от материала пружины и её размеров. Для определения жёсткости k на нижний конец пружины подвешивают груз известной массы m (так что сила $F = mg$ тоже известна) и измеряют удлинение пружины Δl. Тогда $k = F/\Delta l$.

Обычно такой эксперимент проводится несколько раз, в результате получается серия значений F_i (для $i = 1, 2, \ldots, n$) и соответствующих им удлинений Δl_i. Для каждой пары рассчитанная жёсткость $k_i = F_i / \Delta l_i$ будет, скорее всего, различной. Конечно, можно принять за k среднее значение по всем полученным измерениям. Однако такой подход не очень хорошо обоснован с научной точки зрения. Поэтому были разработаны другие методы, один из которых рассматривается далее.

Метод наименьших квадратов

Предположим, что есть два ряда данных одинаковой длины: $x_1, x_2, \ldots, x_n$ и $y_1, y_2, \ldots, y_n$. Предполагается, что они связаны линейной зависимостью $y = k \cdot x$, где k — неизвестный коэффициент. Требуется найти *оптимальное* значение k, которое лучше всего соответствует исходным данным.

Поскольку речь идёт о задаче оптимизации, нужно определить функцию, которая позволяет оценить, насколько хорошо выбранная зависимость соответствует исходным данным. Предположим, что выбран некоторый коэффициент k, так что для каждого x_i можно найти соответствующее ему значение функции $Y_i = k \cdot x_i$.

В идеале график функции должен проходить через все точки (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), т. е. при всех i должно выполняться условие $Y_i = y_i$. Однако на практике этого, скорее всего, не будет (рис. 9.27).

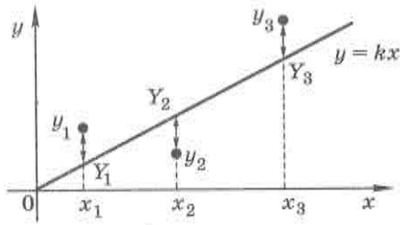


Рис. 9.27

Отклонение полученной линии от исходных данных определяется разностями $Y_i - y_i$: чем они меньше (по модулю), тем лучше соответствие. Однако сумма этих разностей даёт неправильную оценку точности — слагаемые с разными знаками могут скомпенсировать друг друга. Чтобы решить эту проблему, можно сложить квадраты этих величин и выбрать k так, чтобы эта сумма

$$E = \sum_{i=1}^{n} (Y_i - y_i)^2 = (Y_1 - y_1)^2 + (Y_2 - y_2)^2 + \ldots + (Y_n - y_n)^2$$

была минимальной. Такой подход называется **методом наименьших квадратов**. Впервые его предложил немецкий математик *К. Ф. Гаусс*.

Как найти коэффициент k наилучшим образом, т. е. так, чтобы сумма квадратов E была минимальной? Для этого заменим Y_i на $k \cdot x_i$ и раскроем скобки в формуле для вычисления E:

$$(Y_i - y_i)^2 = (k \cdot x_i - y_i)^2 = k^2 x_i^2 - 2k x_i y_i + y_i^2.$$

Группируя слагаемые, содержащие k^2 и k, получаем:

$$E = Ak^2 - Bk + C,$$

где $A = \sum_{i=1}^{n} x_i^2$, $B = 2\sum_{i=1}^{n} x_i y_i$ и $C = \sum_{i=1}^{n} y_i^2$. График зависимости E от k — это парабола, причём её ветви направлены вверх, потому что

$A > 0$ (это сумма квадратов). Вершина параболы (и минимум функции!) находится в точке $k = \frac{B}{2A}$, это и будет оптимальное решение.

Таким образом, алгоритм для определения оптимального значения k приобретает вид:

1) вычислить коэффициенты параболы $A = \sum_{i=1}^{n} x_i^2$ и $B = 2\sum_{i=1}^{n} x_i y_i$;

2) вычислить $k = \frac{B}{2A}$.

Решение получилось простым только потому, что мы выбрали очень простую функцию, линейную с нулевым свободным членом. В более сложных случаях строгое решение задачи оптимизации требует знания высшей математики.

Если исходные данные записаны в массивах $x[1..N]$ и $y[1..N]$, программа для рассмотренного случая выглядит так:

```
A:=0; B:=0
нц для i от 1 до N
  A:=A+x[i]*x[i]
  B:=B+x[i]*y[i]
кц
k:=B/A
```

```
A:=0; B:=0;
for i:=1 to N do begin
  A:=A+x[i]*x[i];
  B:=B+x[i]*y[i];
end;
k:=B/A;
```

Чтобы избавиться от лишних операций, умножение на 2 при вычислении B и деление на 2 при вычислении k не выполняется (применение двух этих операций не меняет результат).

Для решения задачи методом наименьших квадратов можно использовать табличные процессоры с модулем поиска решения. Пусть в результате измерений получены точки (1; 1,1), (2; 1,8) и (3; 3,5). Занесём эти координаты в столбцы A и B, в ячейку B1 запишем начальное приближение для k, а в столбец C — значения функции $y = k \cdot x$ для значений x из столбца A (рис. 9.28).

	A	B	C
1	k	1,000	
2	E	=SUMXMY2(B5:B7;C5:C7)	
3			
4	x	y	Y
5	1	1,1	=$B$1*A5
6	2	1,8	=$B$1*A6
7	3	3,5	=$B$1*A7

Рис. 9.28

Величина E — это сумма квадратов разностей двух рядов, которая вычисляется с помощью функции SUMXMY2 (СУММКВРАЗН). У этой функции два аргумента — ряд y (измеренные значения в столбце B) и ряд Y (вычисленные значения функции в столбце C). Задача оптимизации — найти минимальное значение E (в ячейке B2), изменяя значение k в ячейке B1 — решается с помощью надстройки «Поиск решения».

Восстановление зависимостей

Пусть заданы пары значений x и y, и предполагается, что они связаны некоторой зависимостью $y = f(x)$. Требуется найти функцию $f(x)$ для того, чтобы вычислять значения y в тех точках, где они неизвестны. Такая задача называется **задачей восстановления зависимости**.

Если вид функции не задан, эта задача некорректна, потому что через заданные точки можно провести сколько угодно различных линий (графиков функций), и невозможно сказать, какая из них лучше подходит (рис. 9.29).

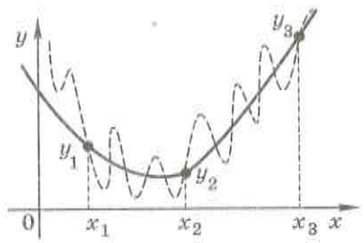


Рис. 9.29

Поэтому для того, чтобы сделать задачу осмысленной, нужно заранее задать вид функции, так что останется только найти её неизвестные коэффициенты.

Откуда взять вид функции? В некоторых случаях он известен из физических законов, описывающих явление (так было в примере с исследованием закона Гука). Иногда вид зависимости можно определить по внешнему виду расположения точек. Также можно попробовать функции разного типа и выбрать лучший вариант. Часто используют следующие типы функций (рис. 9.30):

- линейную $y = a \cdot x + b$;
- логарифмическую $y = a \cdot \ln x + b$;
- показательную (экспоненциальную) $y = a \cdot b^x$;
- степенную $y = a \cdot x^b$.

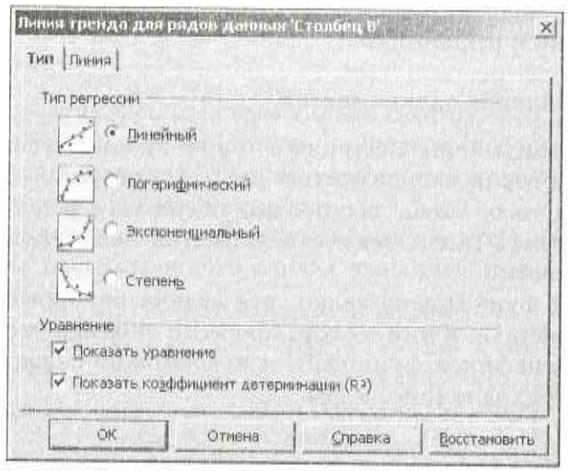


Рис. 9.30

Задача сводится к тому, чтобы выбрать коэффициенты a и b наилучшим образом. Для её решения «вручную» нужно применять методы вычислительной математики, выходящие за рамки школьного курса. Однако в современных табличных процессорах есть встроенные возможности для решения задачи восстановления зависимостей. Полученные графики оптимальных функций называются **линиями тренда** (англ. *trend* — основное направление развития).

Сначала нужно ввести исходные данные (координаты точек) в таблицу и построить по ним диаграмму типа **Диаграмма XY** (в Excel — диаграмма **Точечная**). Лучше оставить на диаграмме только точки, не соединяя их линией.

Для того чтобы построить линию тренда, надо щёлкнуть правой кнопкой мыши на одной из точек и выбрать пункт **Вставить линию тренда** из контекстного меню. В появившемся окне можно выбрать вид зависимости. Если установить флажок **Показать уравнение**, уравнение линии тренда будет показано на диаграмме. Флажок **Показать коэффициент детерминации (R^2)** позволяет

увидеть, насколько точно полученная линия соответствует исходным данным. Коэффициент R^2 вычисляется по формуле

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - Y_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2},$$

где через $\overline{y}$ обозначено среднее значение ряда y. Дробь, которая вычитается из единицы, — это дисперсия ошибки приближения, делённая на дисперсию ряда y. Чем меньше это отношение (и чем больше коэффициент R^2), тем лучше найденная зависимость соответствует исходным данным. В идеальном случае $R^2 = 1$, при этом $y_i = Y_i$ для всех i, т. е. все значения функции совпадают с заданными.

Из приведённой формулы видно, что R^2 имеет наибольшее значение, когда сумма квадратов отклонений $E = \sum_{i=1}^{n}(y_i - Y_i)^2$ минимальна. Это значит, что задача поиска максимума R^2 решается методом наименьших квадратов. Если нужно найти неизвестные коэффициенты функции, которая не входит в стандартный набор (например, $y = a \cdot \sin bx + c$), можно применить метод наименьших квадратов с помощью надстройки «Поиск решения».

Прогнозирование

Во многих задачах (например, в экономике) в результате обработки данных нужно сделать **прогноз**. Если найдена зависимость $y = f(x)$, эта задача решается просто — нужно найти значения функции для тех значений x, которые нас интересуют. Однако может получиться так, что функция, которая очень хорошо соответствует имеющимся данным (см. функцию $y = f(x)$ на рис. 9.31), оказывается непригодна для прогноза.

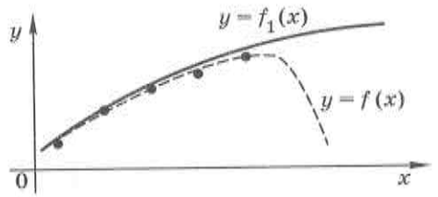


Рис. 9.31

В таких случаях для решения задачи прогнозирования нужно выбирать другую функцию, которая дает меньшее значение R^2, но лучше показывает закономерность изменения величины (например, возрастающий характер функции).

Вопросы и задания

1. Объясните, почему экспериментальные исследования требуют специальных методов обработки данных.
2. Объясните суть метода наименьших квадратов. Почему можно считать такой подход решением задачи оптимизации?

Подготовьте сообщение

а) «Интерполяция»
б) «Экстраполяция»
в) «Аппроксимация»

Задачи

1. Результаты серии измерений, сделанных для определения жёсткости пружины на основе закона Гука, записаны в таблицу:

F, H	1	3	6	10
Δl, м	0,028	0,070	0,170	0,260

Используя метод наименьших квадратов, определите жёсткость пружины. Решите задачу разными способами (с помощью собственной программы и табличного процессора), сравните результаты.

*2. Используя те же данные, что и в задаче 1, найдите жёсткость пружины другим способом. Введём величину, обратную жёсткости: $\eta = 1/k$, тогда $\Delta l = \eta \cdot F$. Сначала, применив метод наименьших квадратов, найдите η, а затем рассчитайте $k = 1/\eta$. Сравните результат с тем, что получилось в задаче 1. Объясните расхождение.

3. Почему задача восстановления зависимости некорректна, если не задан вид функции?

4. Доходы начинающей фирмы (в тысячах рублей) за первые 5 лет работы приведены в таблице:

Год	1	2	3	4	5
Доход	93	187	270	321	350

С помощью табличного процессора найдите зависимость дохода от года работы (выберите лучший из стандартных вариантов, с наибольшим значением R^2). С помощью этой зависимости сделайте прогноз развития фирмы на 2 года вперед.

*5. При изучении волн измерили отклонение уровня поверхности воды y от «нулевого» уровня в одной точке в разные моменты времени t:

t	0	0,2	0,4	0,6	0,8	1
y	1	2	1,2	−0,6	−2	−1,5

Предполагается, что волна описывается формулой $y = a \cdot \sin(b \cdot t + c)$, где a, b и c — некоторые числа. Определите неизвестные коэффициенты методом наименьших квадратов с помощью табличного процессора. В качестве начального приближения можно выбрать $a \approx 1$; $b \approx 4$; $c \approx 1$.

Практические работы к главе 9

Работа № 61 «Решение уравнений методом перебора»
Работа № 62 «Решение уравнений методом деления отрезка пополам»
Работа № 63 «Решение уравнений в табличных процессорах»
Работа № 64 «Вычисление длины кривой»
Работа № 65 «Вычисление площади фигуры»
Работа № 66 «Оптимизация. Метод дихотомии»
Работа № 67 «Оптимизация с помощью табличных процессоров»
Работа № 68 «Статистические расчеты»
Работа № 69 «Условные вычисления»
Работа № 70 «Метод наименьших квадратов»
Работа № 71 «Линии тренда»

ЭОР к главе 9 на сайте ФЦИОР (http//fcior.edu.ru)

- Точность вычислений
- Особенности выполнения вычислений на ЭВМ. Потеря точности
- Уравнения с одной переменной. Корни уравнения. Линейные уравнения
- Дискретизация сигналов
- Задачи оптимизации. Динамическое программирование
- Основные алгоритмы работы со структурами данных

Самое важное в главе 9

- Погрешность вычислений — это разница между вычисленным значением величины и её точным (истинным) значением. Погрешность бывает абсолютная и относительная (в процентах). Для оценки точности измерений и расчётов более полезна относительная погрешность.
- Точность вычислений определяется точностью исходных данных, погрешностями метода и погрешностью вычислений (действий с приближёнными числами).
- Компьютер позволяет легко находить приближённые решения многих задач, которые трудно решаются аналитически (или не решаются вообще).
- Многие приближённые методы решения уравнений — итерационные, т. е. основаны на многократном повторении некоторого алгоритма, который позволяет на каждом шаге приближаться к точному решению. Процедура заканчивается, когда разница между двумя последовательными приближениями становится меньше заданной величины ε.
- Дискретизация — необходимый этап подготовки вычислительных задач для решения на компьютере. Чтобы повысить точность, нужно уменьшать шаг дискретизации, но это увеличивает объём вычислений.
- Оптимизация — это поиск наилучшего решения в заданных условиях. В задачах оптимизации необходимо ввести целевую функцию, для которой требуется найти минимум или максимум. Кроме того, в большинстве практических случаев нужно задать ограничения на допустимые значения переменных.
- Большинство приближённых методов оптимизации позволяет искать только локальный минимум (максимум), в этом случае результат оптимизации зависит от выбора начального приближения.
- Цель статистических расчётов — выявить закономерности путём математической обработки больших массивов данных.
- Для того чтобы задача восстановления зависимости по точкам была корректной, необходимо заранее выбрать вид функции, так что останется определить лишь её неизвестные коэффициенты.

Глава 10
Информационная безопасность

§ 75
Основные понятия

Зависимость современных организаций от компьютерных технологий стала настолько сильной, что вывод из строя компьютерной сети или программного обеспечения может остановить работу предприятия. Чтобы этого не произошло, нужно соблюдать правила информационной безопасности.

Информационная безопасность — это защищённость информации от любых действий, в результате которых владельцам или пользователям информации может быть нанесён недопустимый ущерб. Причиной такого ущерба может быть искажение или утеря информации, а также неправомерный доступ к ней.

Прежде всего в защите нуждаются государственная и военная тайны, а также коммерческая, юридическая и врачебная тайны. Необходимо защищать личную информацию: паспортные данные, данные о банковских счетах, пароли на сайтах, а также любую информацию, которую можно использовать для шантажа, вымогательства и т. п.

Конечно, невозможно защититься от любых потерь, поэтому задача состоит в том, чтобы исключить именно *недопустимый* ущерб. С точки зрения экономики, средства защиты не должны стоить больше, чем возможные потери.

Защита информации — это меры, направленные на то, чтобы не потерять информацию, не допустить её искажения, а также не допустить, чтобы к ней получили доступ люди, не имеющие на это права. В результате нужно обеспечить:

- доступность информации — возможность получения информации за приемлемое время;
- целостность (отсутствие искажений) информации;
- конфиденциальность информации (недоступность для посторонних).

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

Нарушения **целостности информации** — это кража или искажение информации, например подделка сообщений электронной почты и других цифровых документов.

Конфиденциальность информации нарушается, когда информация становится известной тем людям, которые не должны о ней знать (происходит перехват секретной информации).

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

- в сети работает много пользователей, их состав меняется;
- есть возможность незаконного подключения к сети;
- существуют уязвимости в сетевом программном обеспечении;
- возможны атаки взломщиков и вредоносных программ через сеть.

В России вопросы, связанные с защитой информации, регулирует **закон «Об информации, информационных технологиях и о защите информации»**.

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

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

Организационные средства включают:

- распределение помещений и прокладку линий связи таким образом, чтобы злоумышленнику было сложно до них добраться;
- политику безопасности организации.

Серверы, как правило, находятся в отдельном (охраняемом) помещении и доступны только администраторам сети. Важная информация должна периодически копироваться на резервные носители (диски или магнитную ленту) для сохранения её в случае сбоев. Обычные сотрудники (не администраторы):

- имеют право доступа только к тем данным, которые им нужны для работы;
- не имеют права устанавливать программное обеспечение;
- раз в месяц должны менять пароли.

Самое слабое звено любой системы защиты — это человек. Некоторые пользователи могут записывать пароли на видном месте (чтобы не забыть) и передавать их другим, при этом возможность незаконного доступа к информации значительно возрастает. Поэтому очень важно обучить пользователей основам информационной безопасности.

Большинство утечек информации связано с *инсайдерами* (англ. *inside* — внутри) — недобросовестными сотрудниками, работающими в фирме. Известны случаи утечки закрытой информации не через ответственных сотрудников, а через секретарей, уборщиц и другой вспомогательный персонал. Поэтому ни один человек не должен иметь возможности причинить непоправимый вред (в одиночку уничтожить, украсть или изменить данные, вывести из строя оборудование).

Вопросы и задания

1. Что такое информационная безопасность?
2. Что входит в понятие «защита информации»?
3. На какие группы делятся средства защиты информации?
4. Какие меры безопасности обычно применяются в организациях?
5. Почему при объединении компьютеров в сеть безопасность снижается?
6. Кто такие инсайдеры?

§ 76
Вредоносные программы

Что такое компьютерный вирус?

Компьютерный вирус — это программа, способная создавать свои копии (не обязательно точно совпадающие с оригиналом) и внедрять их в файлы и системные области компьютера. При этом копии могут распространяться дальше.

Как следует из этого определения, основная черта компьютерного вируса — это способность распространяться при запуске.

Вирус — это один из типов вредоносных программ. Однако очень часто вирусами называют любые вредоносные программы (англ. *malware*).

Вредоносные программы — это программы, предназначенные для незаконного доступа к информации, для скрытого использования компьютера или для нарушения работы компьютера и компьютерных сетей.

Зачем пишут такие программы? Во-первых, с их помощью можно получить управление компьютером пользователя и использовать его в своих целях. Например, через заражённый компьютер злоумышленник может взламывать сайты и незаконно переводить на свой счёт деньги. Некоторые программы блокируют компьютер и для продолжения работы требуют отправить платное SMS-сообщение.

Зараженные компьютеры, подключенные к сети Интернет, могут объединяться в сеть специального типа — **ботнет** (от англ. *robot* — робот и *network* — сеть). Такая сеть часто состоит из сотен тысяч компьютеров, обладающих в сумме огромной вычислительной мощностью. По команде «хозяина» ботнет может организовать атаку на какой-то сайт. В результате огромного количества запросов сервер не справляется с нагрузкой, сайт становится недоступен, и бизнесмены несут большие денежные потери. Такая атака называется **DoS-атакой**[1] (англ. **DoS** — *Denial of Service* — отказ в обслуживании). Кроме того, ботнеты могут использоваться для подбора паролей, рассылки **спама** (рекламных электронных сообщений) и другой незаконной деятельности.

Во-вторых, некоторые вредоносные программы предназначены для **шпионажа** — передачи по Интернету секретной информации с вашего компьютера: паролей доступа к сайтам, почтовым ящикам, учётным записям в социальных сетях, банковским счетам и электронным платёжным системам. В результате таких краж пользователи теряют не только данные, но и деньги.

В-третьих, иногда вирусы пишутся ради самоутверждения программистами, которые по каким-то причинам не смогли применить свои знания для создания полезного ПО. Такие программы нарушают нормальную работу компьютера: время от времени перезагружают его, вызывают сбои в работе операционной системы и прикладных программ, уничтожают данные.

[1] Здесь речь идёт, строго говоря, о распределённой DoS-атаке (англ. **DDoS** — *Distributed DoS*), которая проводится сразу со многих компьютеров.

Наконец, существуют вирусы, написанные ради шутки. Они не портят данные, но приводят к появлению звуковых или зрительных эффектов (проигрывание мелодии; искажение изображения на экране; кнопки, убегающие от курсора и т. п.).

Создание и распространение компьютерных вирусов и вредоносных программ — это уголовные преступление, которое предусматривает (в особо тяжких случаях) наказание до 7 лет лишения свободы (Уголовный кодекс РФ, статья 273).

Признаки заражения вирусом:

- замедление работы компьютера;
- уменьшение объема свободной оперативной памяти;
- зависание, перезагрузка или блокировка компьютера;
- ошибки при работе ОС или прикладных программ;
- изменение длины файлов, появление новых файлов (в том числе скрытых);
- рассылка сообщений по электронной почте без ведома автора.

Для того чтобы вирус смог выполнить какие-то действия, он должен оказаться в памяти в виде программного кода и получить управление компьютером.

Поэтому **вирусы заражают** не любые данные, а только **программный код**, который может выполняться. Например:

- исполняемые программы (с расширениями exe, com);
- загрузочные секторы дисков;
- пакетные командные файлы (bat);
- драйверы устройств;
- библиотеки динамической загрузки (dll), которые используются прикладными программами;
- документы, которые могут содержать **макросы** — небольшие программы, выполняющиеся при нажатии на клавиши или выборе пункта меню; например, макросы нередко используются в документах пакета Microsoft Office;
- веб-страницы (в них можно внедрить программу-**скрипт**, которая выполнится при просмотре страницы на компьютере пользователя).

В отличие от кода программ, файлы с данными (например, тексты, рисунки, звуковые и видеофайлы) только обрабатываются, но не выполняются, поэтому заложенный в них код никогда не должен получить управление компьютером. Однако из-за ошибок в программном обеспечении может случиться так, что специально подобранные некорректные данные вызовут сбой программы обработки и выполнение вредоносного кода[1]. Таким образом, существует некоторый шанс, что вредоносная программа, внедрённая в рисунок или видеофайл, все-таки запустится.

Сейчас существуют два основных источника заражения вредоносными программами — флэш-диски и компьютерные сети. **Компьютер может быть заражён при:**

- запуске заражённого файла;
- загрузке с заражённого диска CD/DVD или флэш-диска;
- автозапуске заражённого диска CD/DVD или флэш-диска (вирус автоматически запускается из файла autorun.inf в корневом каталоге диска);
- открытии заражённого документа с макросами;
- открытии сообщения электронной почты с вирусом или запуске заражённой программы, полученной в приложении к сообщению;
- открытии веб-страницы с вирусом;
- установке активного содержимого для просмотра веб-страницы.

Кроме того, есть вирусы-черви, которые распространяются по компьютерным сетям без участия человека. Они могут заразить компьютер даже тогда, когда пользователь не сделал никаких ошибочных действий.

Типы вредоносных программ

К вредоносным программам относятся компьютерные вирусы, черви, троянские программы и др. По **«среде обитания»** обычно выделяют следующие типы вирусов:

- **файловые** — внедряются в исполняемые файлы, системные библиотеки и т. п.;

[1] В 2002 г. был обнаружен вирус, который внедрялся в рисунки формата JPEG. Однако он получал управление только из-за ошибки в системной библиотеке Windows, которая была быстро исправлена.

- **загрузочные** — внедряются в загрузочный сектор диска или в главную загрузочную запись жёсткого диска (англ. **MBR** — *Master Boot Record*); опасны тем, что загружаются в память раньше, чем ОС и антивирусные программы;
- **макровирусы** — поражают документы, в которых могут быть макросы;
- **скриптовые вирусы** — внедряются в командные файлы или в веб-страницы (записывая в них код на языке VBScript или JavaScript);
- **сетевые вирусы** — распространяются по компьютерным сетям.

Некоторые вирусы при создании новой копии немного меняют свой код, для того чтобы их было труднее обнаружить. Такие вирусы называют **полиморфными** (от греч. πολυ — много, μορφη — форма, внешний вид).

Червь — это вредоносная программа, которая распространяется по компьютерным сетям. Наиболее опасны **сетевые черви**, которые используют «дыры» (ошибки в защите, уязвимости) операционных систем и распространяются очень быстро без участия человека. Червь посылает по сети специальный пакет данных — **эксплойт** (англ. *exploit* — эксплуатировать), который позволяет выполнить код на удалённом компьютере и внедриться в систему.

Как правило, вскоре после обнаружения уязвимости выпускается обновление программного обеспечения («заплатка», «патч»); если его установить, то червь становится неопасен. К сожалению, системные администраторы не всегда вовремя устанавливают обновления. Это приводит к эпидемиям сетевых червей, которые по статистике вызывают наибольшее число заражений. Заражённые компьютеры используются для рассылки спама или массовых DoS-атак на сайты в Интернете.

Почтовые черви распространяются как приложения к сообщениям электронной почты. Они представляют собой программы, которые при запуске заражают компьютер и рассылают свои копии по всем адресам из адресной книги пользователя. Из-за этой опасности многие почтовые серверы (например, mail.google.com) не разрешают пересылку исполняемых файлов.

Чтобы заставить пользователя запустить червя, применяются методы **социальной инженерии**: текст сообщения составляется так, чтобы заинтересовать человека и спровоцировать его на

запуск программы, приложенной к письму. В некоторых случаях программа-вирус упакована в архив и защищена паролем, но находится немало людей, которые распаковывают его (пароль указывается в письме) и запускают программу. Часто в почтовых сообщениях содержится только ссылка на сайт, содержащий вирус.

Иногда файл, пришедший как приложение к письму, имеет двойное расширение, например:

```
СуперКартинка.jpg                .exe
```

В самом деле это программа (расширение имени файла exe), но пользователь может увидеть только первые две части имени и попытаться открыть такой «рисунок».

Существуют черви, которые могут распространяться через файлообменные сети, чаты и системы мгновенных сообщений (например, ICQ), но они мало распространены.

Ещё одна группа вредоносных программ — **троянские программы**, или «троянцы» (трояны). Троянский конь — это огромный деревянный конь, которого древние греки подарили жителям Трои во время Троянской войны. Внутри него спрятались воины, которые ночью выбрались, перебили охрану и открыли ворота города. Троянские программы проникают на компьютер под видом «полезных» программ, например кодеков для просмотра видео или экранных заставок. В отличие от вирусов и червей они не могут распространяться самостоятельно и часто «путешествуют» вместе с червями. Среди «троянцев» встречаются:

- **клавиатурные шпионы** — передают «хозяину» все данные, вводимые с клавиатуры (в том числе коды доступа к банковским счетам и т. п.);
- **похитители паролей** — передают пароли, запомненные, например, в браузерах;
- **утилиты удалённого управления** — позволяют злоумышленнику управлять компьютером через Интернет (например, загружать и запускать любые файлы);
- **логические бомбы** — при определённых условиях (дата, время, команда по сети) уничтожают информацию на дисках.

Большинство существующих вирусов написано для ОС Windows, которая установлена более чем на 90% персональных компьютеров.

Известны также вирусы для macOS и Linux, но не каждому удается их запустить. Дело в том, что обычный пользователь (не администратор) в этих операционных системах не имеет права на изменение системных файлов, поэтому macOS и Linux считают защищёнными от вирусов. Кроме того, вирусы часто полагаются на то, что системные функции размещаются в памяти по определённым адресам. При сборке ядра Linux из исходных кодов эти адреса могут меняться, поэтому вирус, работающий на одном дистрибутиве, может не работать на других.

Вопросы и задания

1. Что такое компьютерный вирус? Чем он отличается от других программ?
2. Что такое вредоносные программы? Какие вредоносные программы вы знаете?
3. Перечислите признаки заражения компьютера вирусом.
4. Какие вредные действия могут совершать вредоносные программы?
5. Какие объекты могут быть заражены вирусами?
6. Какие объекты не заражаются вирусами?
7. При каких действиях пользователя возможно заражение вирусом?
8. Является ли создание и распространение вирусов уголовным преступлением?
9. Какие типы вирусов вы знаете?
10. Что означает сокращение MBR?
11. Чем опасны загрузочные вирусы?
12. Что такое макровирусы? Какие файлы они поражают?
13. Что могут заражать скриптовые вирусы?
14. Что такое полиморфные вирусы? Почему их сложно обнаруживать?
15. Что такое сетевой червь?
16. Что такое эксплойт?
17. Почему необходимо сразу устанавливать обновления для операционных систем?
18. С какими целями могут быть использованы компьютеры, заражённые сетевым червем?
19. Почему многие почтовые серверы запрещают пересылку исполняемых файлов?
20. Что такое социальная инженерия? Как она используется авторами вирусов?
21. Что такое троянские программы? Какие типы троянских программ вы знаете?
22. Какие операционные системы лучше защищены от вирусов? Почему?

Подготовьте сообщение:

а) «Сетевые черви»
б) «Социальная инженерия»
в) «Троянские программы»
г) «Вредоносные программы для Linux и MacOS»
д) «Вредоносные программы и закон»

§ 77
Защита от вредоносных программ

Антивирусные программы

Антивирус — это программа, предназначенная для борьбы с вредоносными программами.

Антивирусы выполняют три основные задачи:

1) не допустить заражение компьютера вирусом;
2) обнаружить присутствие вируса в системе;
3) удалить вирус без ущерба для остальных данных.

Код большинства вирусов содержит характерные цепочки байтов — **сигнатуры** (от лат. *signare* — подписать). Если в файле обнаруживается сигнатура какого-то вируса, можно предположить, что файл заражён. Такой подход используется всеми антивирусными программами. Сигнатуры известных вирусов хранятся в базе данных антивируса, которую нужно регулярно обновлять через Интернет.

Современные антивирусы — это программные комплексы, состоящие из нескольких программ. Чаще всего они включают антивирус-сканер (иногда его называют антивирус-доктор) и антивирус-монитор.

Для того чтобы **антивирус-сканер** начал работу, пользователь должен его запустить и указать, какие файлы и папки нужно проверить. Это «защита по требованию». Сканеры используют два основных метода поиска вирусов:

- **поиск в файлах сигнатур вирусов**, которые есть в базе данных; после обнаружения файл с вирусом можно вылечить, а если это не получилось — удалить;

- **эвристический анализ** (греч. ευρηκα — «нашёл!»), при котором программа ищет в файле код, похожий на вирус.

Эвристический анализ часто позволяет обнаруживать полиморфные вирусы (изменяющие код с каждым новым заражением), но не гарантирует это. Кроме того, случаются ложные срабатывания, когда «чистый» файл попадает под подозрение.

Главный недостаток сканеров состоит в том, что они не могут предотвратить заражение компьютера, потому что начинают работать только при ручном запуске.

Антивирусы-мониторы — это программы постоянной защиты, они находятся в памяти в активном состоянии. Их основная задача — не допустить заражения компьютера и получения заражённых файлов извне. Для этого мониторы:

- проверяют «на лету» все файлы, которые копируются, перемещаются или открываются в различных прикладных программах;
- проверяют используемые флэш-диски;
- перехватывают действия, характерные для вирусов (форматирование диска, замена и изменение системных файлов) и блокируют их;
- проверяют весь поток данных, поступающий из Интернета (сообщения электронной почты, веб-страницы, сообщения ICQ).

Мониторы ведут непрерывное наблюдение, блокируют вирус в момент заражения. Иногда они могут перехватить и неизвестный вирус (сигнатуры которого нет в базе), обнаружив его подозрительные действия.

Главный недостаток антивирусов-мониторов — значительное замедление работы системы, особенно на маломощных компьютерах. Кроме того, мониторы фактически встраиваются в операционную систему, поэтому ошибки разработчиков антивируса могут привести к печальным последствиям (вплоть до удаления системных библиотек и вывода ОС из строя). Бывает и так, что при запущенном мониторе некоторые программы работают неправильно или вообще не работают. Тем не менее не рекомендуется отключать монитор, особенно если вы работаете в Интернете или переносите файлы с помощью флэш-дисков.

Современные антивирусы частично защищают компьютер ещё и от:

- **фишинга** — выманивания паролей для доступа на сайты Интернета с помощью специально сделанных веб-страниц, которые внешне выглядят так же, как «официальные» сайты;
- **рекламных баннеров и всплывающих окон** на веб-страницах;
- **спама** — рассылки нежелательных рекламных сообщений по электронной почте.

Большинство антивирусных программ — условно-бесплатные (англ. *shareware*), пробные версии с ограниченным сроком действия можно свободно загрузить из Интернета. Наиболее известны антивирусы AVP (www.kaspersky.ru), DrWeb (www.drweb.com), Nod32 (www.eset.com), McAfee (home.mcafee.com).

На многих сайтах (www.kaspersky.ru, www.freedrweb.com) доступны для скачивания лечащие программы-сканеры, которые бесплатны для использования на домашних компьютерах. В отличие от полных версий, в них нет антивируса-монитора, и базы сигнатур не обновляются.

Существуют антивирусы, бесплатные для использования на домашних компьютерах, например Microsoft Security Essentials (www.microsoft.com), Avast Home (www.avast.com), Avira (www.avira.com), AVG Free (free.grisoft.com). Антивирус ClamAV (www.clamav.net) распространяется свободно с исходным кодом.

На сайтах некоторых компаний можно найти **онлайновые антивирусы** (например, http://www.kaspersky.ru/virusscanner). Они устанавливают на компьютер специальный сканирующий модуль и проверяют файлы и оперативную память. Как правило, онлайновые антивирусы могут обнаружить вирусы, но не удаляют их, предлагая приобрести коммерческую версию.

Брандмауэры

Для защиты отдельных компьютеров и сетей от атак из Интернета (в том числе и вирусных) используются **брандмауэры** (нем. *Brandmauer* — стена между зданиями для защиты от распространения огня). Их также называют **сетевыми экранами** или **фаейрволами** (от англ. *firewall* — противопожарная стена). Брандмауэры запрещают передачу данных по каналам связи, которые часто используют вирусы и программы для взлома сетей.

На рисунке 10.1 показана защита одного компьютера с помощью брандмауэра, точно так же защищаются от угроз из Интернета локальные сети.

Рис. 10.1

Брандмауэр входит в состав современных версий ОС Windows, в ядро Linux также включён встроенный брандмауэр Netfilter. Иногда устанавливают дополнительные брандмауэры, например Agnitum Outpost (www.agnitum.com), Kerio Winroute Firewall (kerio.ru) или бесплатную программу Comodo Personal Firewall (www.personalfirewall.comodo.com).

Меры безопасности

Главный вред, который могут нанести вредоносные программы, — это потеря данных или паролей доступа к закрытой информации.

Чтобы уменьшить возможный ущерб, рекомендуется регулярно делать резервные копии важных данных на дисках CD/DVD или флэш-дисках.

Если вы работаете в сети, желательно включать антивирус-монитор и брандмауэр. Монитор сразу сообщит об опасности, если вставленный флэш-диск содержит вирус. Все новые файлы (особенно программы!) нужно проверять с помощью антивируса-сканера.

Не рекомендуется открывать подозрительные сообщения электронной почты, полученные с неизвестных адресов, особенно файлы-приложения (помните про методы социальной инженерии — заинтересовать жертву и заставить запустить программу). Опасно также переходить по ссылкам в тексте писем, с большой вероятностью они ведут на сайты, зараженные вирусами.

Если компьютер заражён, нужно отключить его от компьютерной сети и запустить антивирус-сканер. Очень часто это позволяет удалить вирус, если его сигнатура есть в базе данных. Если антивирус не был установлен раньше, можно попробовать установить его на заражённый компьютер, но это не всегда приводит к успеху (вирус может блокировать установку антивируса).

Если антивирус-сканер не обнаруживает вирус или не может его удалить, можно попытаться (желательно с другого компьютера) найти в Интернете бесплатную утилиту для лечения с новыми базами сигнатур. Например, утилита CureIt (www.freedrweb.com) не требует установки и может быть запущена с флэш-диска. Даже если удалить вирус не удастся, скорее всего, он будет обнаружен, и программа покажет его название. Следующий шаг — искать в Интернете утилиту для удаления именно этого вируса (например, ряд утилит можно найти на сайте support.kaspersky.ru).

В особо тяжёлых случаях для уничтожения вирусов приходится полностью форматировать жёсткий диск компьютера, при этом все данные теряются.

Вопросы и задания

1. Что такое антивирус? Какие задачи он решает?
2. Что такое сигнатура?
3. Почему нужно регулярно обновлять базы сигнатур антивирусов?
4. Чем отличается антивирус-сканер от антивируса-монитора?
5. Что значит «защита по требованию»?
6. Что такое эвристический анализ? В чём его достоинства и недостатки?
7. Какие функции у антивируса-монитора? Каковы его недостатки?
8. Что такое фишинг?
9. Что такое спам?
10. Какие ограничения есть у пробных версий коммерческих антивирусов?
11. Что такое онлайновый антивирус?
12. Что такое брандмауэр? Зачем он нужен?
13. В чём заключается основной вред, наносимый вирусами? Как можно уменьшить возможные потери?
14. Как можно улучшить безопасность компьютера при работе в сети Интернет?
15. Какие меры безопасности необходимы при работе с электронной почтой?
16. Какие действия можно предпринять, если компьютер заражен вирусом?

Подготовьте сообщение

а) «Бесплатное антивирусное программное обеспечение»
б) «Онлайновые антивирусы»
в) «Настройка брандмауэра»

§ 78 Шифрование

Один из методов защиты информации от неправомерного доступа — это шифрование, т. е. кодирование специального вида.

Шифрование — это преобразование (кодирование) открытой информации в зашифрованную, недоступную для понимания посторонними.

Шифрование применяется, в первую очередь, для передачи секретной информации по незащищённым каналам связи. Шифровать можно любые данные — тексты, рисунки, звук, базы данных и т. д.

Человечество применяет шифрование с того момента, как появилась секретная информация, которую нужно было скрыть от врагов. Первое известное науке шифрованное сообщение — египетский текст, в котором вместо принятых тогда иероглифов были использованы другие знаки.

Методы шифрования и расшифровывания сообщений изучает наука **криптология**, история которой насчитывает около четырех тысяч лет. Она состоит из двух ветвей: криптографии и криптоанализа.

Криптография — это наука о способах шифрования информации.
Криптоанализ — это наука о методах и способах вскрытия шифров.

Обычно предполагается, что сам алгоритм шифрования известен всем, но неизвестен его ключ, без которого сообщение невозможно расшифровать. В этом заключается отличие шифрования от простого кодирования, при котором для восстановления сообщения достаточно знать только алгоритм кодирования.

Ключ — это параметр алгоритма шифрования (шифра), позволяющий выбрать одно конкретное преобразование из всех вариантов, предусмотренных алгоритмом. Знание ключа позволяет свободно зашифровывать и расшифровывать сообщения.

Все шифры (системы шифрования) делятся на две группы — симметричные и несимметричные (с открытым ключом).

Симметричный шифр означает, что и для шифрования, и для расшифровывания сообщений используется один и тот же ключ. В системах **с открытым ключом** используются два ключа — открытый и закрытый, которые связаны друг с другом с помощью некоторых математических зависимостей. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

Криптостойкость шифра — это устойчивость шифра к расшифровке без знания ключа.

Стойким считается алгоритм, который для успешного раскрытия требует от противника недостижимых вычислительных ресурсов, недостижимого объёма перехваченных сообщений или такого времени, что по его истечении защищённая информация будет уже не актуальна.

Шифр Цезаря[1] — один из самых известных и самых древних шифров. В этом шифре каждая буква заменяется на другую, расположенную в алфавите на заданное число позиций k вправо от неё. Алфавит замыкается в кольцо, так что последние символы заменяются на первые. На рисунке 10.2 — пример шифра Цезаря (со сдвигом 3)[2].

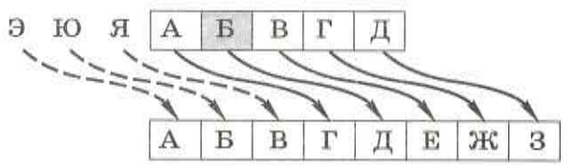


Рис. 10.2

[1] Назван в честь римского императора Гая Юлия Цезаря, использовавшего его для секретной переписки.

[2] Будем считать, что буквы «Е» и «Ё» совпадают.

Знаменитая фраза «ПРИШЕЛ УВИДЕЛ ПОБЕДИЛ» при использовании шифра Цезаря со сдвигом 3 будет закодирована так:

ТУЛЬИО ЦЕЛЗИО ТСДИЗЛО

Если первая буква алфавита имеет код 0, вторая — код 1 и т. д., алгоритм шифрования может быть выражен формулой

$$y = (x + k) \bmod n,$$

где x — код исходного символа, k — величина сдвига, y — код символа-замены, n — количество символов в алфавите, а запись $(x + k) \bmod n$ обозначает остаток от деления $x + k$ на n. Операция взятия остатка от деления необходима для того, чтобы замкнуть алфавит в кольцо. Например, при использовании русского алфавита (32 буквы, без буквы «Ё») для буквы «Я» (код 31) получаем код заменяющего символа $(31 + 3) \bmod 32 = 2$, это буква «В».

Ключом для шифра Цезаря служит сдвиг k, если его знать, то сообщение легко расшифровать. Для этого используется формула

$$x = (y - k + n) \bmod n.$$

Шифр Цезаря относится к шифрам **простой подстановки**, так как каждый символ исходного сообщения заменяется на другой символ из того же алфавита. Такие шифры легко раскрываются с помощью частотного анализа, потому что в каждом языке частоты встречаемости букв примерно постоянны для любого достаточно большого текста.

Значительно сложнее сломать **шифр Виженера**[1], который стал естественным развитием шифра Цезаря. Для использования шифра Виженера используется ключевое слово, которое задает переменную величину сдвига. Например, пусть ключевое слово — «ЗАБЕГ». По таблице (рис. 10.3) определяем коды букв.

0	1	2	3	4	5	6	7	8	9	
А	Б	В	Г	Д	Е	Ж	З	И	Й	…

Рис. 10.3

Получаем: «З» — 7, «А» — 0, «Б» — 1, «Е» — 5, «Г» — 3. Это значит, что для кодирования первой буквы любого текста используется сдвиг 7, для кодирования второй — 0 (символ не меняется) и т. д. Для пятой буквы используется сдвиг 3, а для шес-

[1] Назван по имени *Блеза Виженера*, швейцарского дипломата XVI века.

той — снова 7 (начинаем «проходить» кодовое слова с начала). Фраза «ПРИШЕЛ УВИДЕЛ ПОБЕДИЛ» при использовании шифра Виженера с ключом «ЗАБЕГ» будет закодирована в виде «ЦРЙЭИТ УГНЗМЛ РУДМДЙР».

Шифр Виженера обладает значительно более высокой криптостойкостью, чем шифр Цезаря. Это значит, что его труднее раскрыть — подобрать нужное ключевое слово. Теоретически, если длина ключа равна длине сообщения и каждый ключ используется только один раз, шифр Виженера обладает абсолютной криптостойкостью — взломать его невозможно.

Вопросы и задания

1. Чем различаются понятия «шифрование» и «кодирование»?
2. Что такое ключ?
3. Как называется наука, изучающая методы шифрования?
4. Что такое симметричный шифр? Какая проблема возникает при использовании симметричного шифра, если участники переписки находятся в разных странах?
5. Что такое несимметричные шифры? На чём основана их надёжность?
6. Что такое криптостойкость алгоритма? Какой алгоритм считается криптостойким?

Подготовьте сообщение

а) «Полиграммные шифры подстановки»
б) «Шифрование и закон»
в) «Криптостойкость шифров»
г) «Частотный анализ»

Задачи

1. Зашифруйте с помощью шифра Цезаря со сдвигом 6 высказывание «ЛЮДИ ОХОТНО ВЕРЯТ ТОМУ, ЧЕМУ ЖЕЛАЮТ ВЕРИТЬ».
2. Напишите программу, которая выполняет шифрование строки с помощью шифра Цезаря.

*3. Попытайтесь расшифровать сообщение, закодированное шифром Цезаря с неизвестным сдвигом: «ХШЖНПУТ ФКХКОЙКТ». Для этого можно написать программу.

4. Используя шифр Виженера с ключом «ЛЕНА», зашифруйте сообщение «НЕЛЬЗЯ ОБИЖАТЬ ГОСТЯ».

*5. Измените программу для шифрования с помощью шифра Цезаря так, чтобы учитывать букву Ё.

*6. Измените программу для шифрования с помощью шифра Цезаря так, чтобы учитывать и букву Ё, и пробел.

§ 79
Хэширование и пароли

В современных информационных системах часто используется вход по паролю. Если при этом где-то хранить пароли всех пользователей, система становится очень ненадёжной, потому что «утечка» паролей позволит сразу получить доступ к данным. Вместе с тем кажется, что пароли обязательно где-то нужно хранить, иначе пользователи не смогут войти в систему. Однако это не совсем так. Можно хранить не пароли, а некоторые числа, полученные в результате обработки паролей. Простейший вариант — сумма кодов символов, входящих в пароль. Для пароля «A123» такая сумма равна

$$215 = 65 \text{ (код «A»)} + 49 \text{ (код «1»)} + 50 \text{ (код «2»)} + 51 \text{ (код «3»)}.$$

Фактически мы определили функцию $H(M)$, которая сообщение M *любой* длины превращает в короткий код m. Такую функцию называются **хэш-функцией** (от англ. *hash* — мешанина, крошить), а само полученное число — **хэш-кодом**, **хэш-суммой** или просто **хэшем** исходной строки. Важно, что, зная хэш-код, невозможно восстановить исходный пароль! В этом смысле **хэширование** — это *необратимое* шифрование.

Итак, вместо пароля «A123» мы храним число 215. Когда пользователь вводит пароль, мы считаем сумму кодов символов этого пароля и разрешаем вход в систему только тогда, когда она равна 215. И вот здесь возникает проблема: существует очень много паролей, для которых наша хэш-функция даёт значение 215, например «B023». Такая ситуация — совпадение хэш-кодов различных исходных строк — называется **коллизией** (англ. *collision* — столкновение). Коллизии будут всегда — ведь мы «сжимаем» длинную цепочку байтов до числа. Казалось бы, ничего хорошего не получилось: если взломщик узнает хэш-код, то, зная алгоритм его получения, он сможет легко подобрать пароль с таким же хэшем и получить доступ к данным. Однако это произошло потому, что мы выбрали плохую хэш-функцию.

Математики разработали надёжные (но очень сложные) хэш-функции, обладающие особыми свойствами:

1) хэш-код очень сильно меняется при малейшем изменении исходных данных;

2) при известном хэш-коде m невозможно за приемлемое время найти сообщение M с таким же хэш-кодом ($H(M) = m$);
3) при известном сообщении M невозможно за приемлемое время найти сообщение M_1 с таким же хэш-кодом ($H(M) = H(M_1)$).

Здесь выражение «невозможно за приемлемое время» (или «вычислительно невозможно») означает, что эта задача решается только перебором вариантов (других алгоритмов не существует), а количество вариантов настолько велико, что на решение могут уйти сотни и тысячи лет. Поэтому даже если взломщик получил хэш-код пароля, он не сможет за приемлемое время получить сам пароль (или пароль, дающий такой же хэш-код).

Чем длиннее пароль, тем больше количество вариантов. Кроме длины для надёжности пароля важен используемый набор символов. Например, очень легко подбираются пароли, состоящие только из цифр. Если же пароль состоит из 10 символов и содержит латинские буквы (заглавные и строчные) и цифры, перебор вариантов (англ. *brute force* — метод «грубой силы») со скоростью 10 млн паролей в секунду займет более 2000 лет.

Надёжные пароли должны состоять не менее чем из 7–8 символов; пароли, состоящие из 15 символов и более, взломать методом «грубой силы» практически невозможно. Не используйте пароли типа «12345», «qwerty», свой день рождения, номер телефона. Плохо, если пароль представляет собой известное слово, для этих случаев взломщики используют подбор по словарю. Сложнее всего подобрать пароль, который представляет собой случайный набор заглавных и строчных букв, цифр и других знаков[1].

Сегодня для хэширования в большинстве случаев применяют алгоритмы MD5, SHA1 и российский алгоритм, изложенный в ГОСТ Р34.11-94 (он считается одним из самых надёжных). В криптографии хэш-коды чаще всего имеют длину 128, 160 и 256 битов.

Хэширование используется также для проверки правильности передачи данных: различные контрольные суммы — это не что иное, как хэш-коды.

[1] Однако такой пароль сложно запомнить.

Вопросы и задания

1. Что такое хэширование? Хэш-функция? Хэш-код?
2. Какую хэш-функцию вы используете, когда начинаете искать слово в словаре?
3. Что такое коллизии? Почему их должно быть как можно меньше?
4. Какие требования предъявляются к хэш-функциям, которые используются при хранении паролей?
5. Что значит «вычислительно невозможно»?
6. Взломщик узнал хэш-код пароля администратора сервера. Сможет ли он получить доступ к секретным данным на сервере?
7. Какие свойства пароля влияют на его надёжность?
8. Как выбрать надёжный пароль?
9. Какие алгоритмы хэширования сейчас чаще всего применяются?
*10. Предложите какой-нибудь свой метод хэширования. Подумайте, как часто при его использовании могут происходить коллизии.

Подготовьте сообщение

а) «Где используют хэши?»
б) «Хэширование и передача данных»
в) «Хэширование с "солью"»

Задачи

1. Напишите программу, которая запрашивает длину пароля, количество используемых символов и скорость перебора (например, в миллионах вариантов в секунду) и находит время, необходимое для подбора пароля методом «грубой силы».
2. Компьютер выполняет перебор со скоростью 10 млн паролей в секунду. С помощью программы (см. задачу 1) определите, какова должна быть длина пароля, чтобы время его взлома составило не менее 1 месяца, если:
   а) пароль состоит только из цифр;
   б) пароль состоит только из заглавных латинских букв и цифр;
   в) пароль состоит только из латинских букв (заглавных и строчных) и цифр.

§ 80
Современные алгоритмы шифрования

Государственным стандартом шифрования в России является алгоритм, зарегистрированный как ГОСТ 28147-89. Он является **блочным шифром**, т. е. шифрует не отдельные символы, а 64-бит-

ные блоки. В алгоритме предусмотрены 32 цикла преобразования данных с 256-битным ключом, за счёт этого он очень надёжен (обладает высокой криптостойкостью). На современных компьютерах раскрытие этого шифра путём перебора ключей («методом грубой силы») займёт не менее сотен лет, что делает такую атаку бессмысленной.

В США в качестве стандарта принят блочный шифр AES (*англ.* Advanced Encryption Standard, передовой стандарт шифрования), выбранный в 2001 г. по результатам проведённого конкурса. Шифр AES используется также в защищённых беспроводных сетях (Wi-Fi).

В Интернете популярен алгоритм RSA, названный так по начальным буквам фамилий его авторов — Р. Райвеста (R. Rivest), А. Шамира (A. Shamir) и Л. Адлемана (L. Adleman). Это алгоритм *с открытым ключом*, стойкость алгоритма основана на том, что перемножить два очень больших простых числа достаточно просто, а вот разложить такое произведение на простые сомножители очень трудно (эту задачу сейчас умеют решать только перебором вариантов). Поскольку количество вариантов огромно, для раскрытия шифра требуется много лет работы современных компьютеров.

Для применения алгоритма RSA требуется построить открытый и секретный ключи следующим образом.

1. Выбрать два больших простых числа, p и q.
2. Найти их произведение $n = p \cdot q$ и значение $\varphi = (p-1)\cdot(q-1)$.
3. Выбрать число e $(1 < e < \varphi)$, которое не имеет общих делителей с φ.
4. Найти число d, которое удовлетворяет условию $d \cdot e = k \cdot \varphi + 1$ для некоторого целого k.
5. Пара значений (e, n) — это открытый ключ RSA (его можно свободно публиковать), а пара (d, n) — это секретный ключ.

Передаваемое сообщение нужно сначала представить в виде последовательности чисел в диапазоне от 0 до $n - 1$. Для шифрования используют формулу

$$y = x^e \bmod n,$$

где x — исходное сообщение (число), (e,n) — открытый ключ, y — закодированное сообщение (число), а запись $x^e \bmod n$ обозначает остаток от деления x^e на n. Расшифровка сообщения выполняется по формуле

$$x = y^d \bmod n.$$

Это значит, что зашифровать сообщение может каждый (открытый ключ общеизвестен), а прочитать его — только тот, кто знает секретный показатель степени d.

Для лучшего понимания мы покажем работу алгоритма RSA на простом примере. Возьмём $p = 3$ и $q = 7$, тогда находим $n = p \cdot q = 21$ и $\varphi = (p-1)\cdot(q-1) = 12$. Выберем $e = 5$, тогда равенство $d \cdot e = k \cdot \varphi + 1$ выполняется, например, при $d = 17$ (и $k = 7$). Таким образом, мы получили открытый ключ (5, 21) и секретный ключ (17, 21).

Зашифруем сообщение, состоящее из чисел 1, 2 и 3, с помощью открытого ключа (5, 21). Получаем:

$$1 \Rightarrow 1^5 \bmod 21 = 1, \quad 2 \Rightarrow 2^5 \bmod 21 = 11, \quad 3 \Rightarrow 3^5 \bmod 21 = 12,$$

т. е. зашифрованное сообщение состоит из чисел 1, 11 и 12. Зная секретный ключ (17, 21), можно его расшифровать:

$$1 \Rightarrow 1^{17} \bmod 21 = 1, \quad 11 \Rightarrow 11^{17} \bmod 21 = 2, \quad 12 \Rightarrow 12^{17} \bmod 21 = 3.$$

Мы получили исходное сообщение.

Конечно, вы заметили, что при шифровании и расшифровке приходится вычислять остаток от деления очень больших чисел (например, 12^{17}) на n. Оказывается, само число 12^{17} в этом случае находить не нужно. Достаточно записать в обычную целочисленную переменную, например x, единицу, а потом 17 раз выполнить преобразование $x = 12 \cdot x \bmod 21$. После этого в переменной x будет значение $12^{17} \bmod 21 = 3$. Попробуйте доказать правильность этого алгоритма.

Чтобы взломать шифр, злоумышленнику надо узнать секретный показатель степени d. А для этого необходимо найти большие простые числа p и q, такие что $n = p \cdot q$. Если n велико, это очень сложная задача, её решение перебором вариантов на современном компьютере займёт сотни лет. В 2009 г. группа учёных из разных стран в результате многомесячных расчётов на сотнях компьютеров смогла расшифровать сообщение, зашифрованное алгоритмом RSA с 768-битным ключом. Поэтому сейчас надёжными считаются ключи с длиной 1024 бита и более. Если будет построен работающий квантовый компьютер, взлом алгоритма RSA будет возможен за очень небольшое время.

При использовании симметричных шифров всегда возникает проблема: как передать ключ, если канал связи ненадёжный? Ведь, получив ключ, противник сможет расшифровать все дальнейшие сообщения. Для алгоритма RSA этой проблемы нет, сто-

ронам достаточно обменяться открытыми ключами, которые можно показывать всем желающим.

У алгоритма RSA есть ещё одно достоинство: его можно использовать для цифровой подписи сообщений. Она служит для доказательства авторства документов, защиты сообщений от подделки и умышленных изменений.

Цифровая подпись — это набор символов, который получен в результате шифрования сообщения с помощью личного секретного кода отправителя.

Отправитель может передать вместе с исходным сообщением такое же сообщение, зашифрованное с помощью своего секретного ключа (это и есть цифровая подпись). Получатель расшифровывает цифровую подпись с помощью открытого ключа. Если она совпала с незашифрованным сообщением, можно быть уверенным, что его отправил тот человек, который знает секретный код. Если сообщение было изменено при передаче, оно не совпадёт с расшифрованной цифровой подписью. Так как сообщение может быть очень длинным, для сокращения объёма передаваемых данных чаще всего шифруется не всё сообщение, а только его хэш-код.

Во многих современных программах есть возможность шифровать данные с паролем. Например, офисные пакеты OpenOffice и Microsoft Office позволяют шифровать все создаваемые документы (для их просмотра и/или изменения нужно ввести пароль). При создании архива (например, в архиваторах 7-ZIP, WinRAR, WinZip) также можно установить пароль, без которого извлечь файлы невозможно.

В простейших задачах для шифрования файлов можно использовать бесплатную программу Шифровальщик (www.familytree.ru/ru/cipher.htm), версии которой существуют для Linux и Windows. Программы TrueCrypt (www.truecrypt.org), BestCrypt (www.jetico.com) и FreeOTFE (freeotfe.org) создают логические диски-контейнеры, информация на которых шифруется. Свободно распространяемая программа DiskCryptor (diskcryptor.net) позволяет шифровать разделы жёстких дисков и даже создавать шифрованные флэш-диски и диски CD/DVD.

Программа GnuPG (gnupg.org) также относится к свободному программному обеспечению. В ней поддерживаются симметричные и несимметричные шифры, а также различные алгоритмы электронной цифровой подписи.

Вопросы и задания

1. Какой алгоритм шифрования принят в России в качестве государственного стандарта?
2. Что такое блочный алгоритм шифрования?
3. К какому типу относится алгоритм RSA? На чем основана его криптостойкость?
4. Что такое цифровая подпись?
5. Как можно использовать алгоритм RSA для цифровой подписи?

Подготовьте сообщение

а) «Стандарты шифрования разных стран»
б) «Шифрование с открытым ключом: за и против»
в) «Алгоритм Эль-Гамаля»
г) «Цифровая подпись в Российской Федерации»

Задачи

*1. Напишите программу, которая строит открытый и секретный ключи RSA для небольших множителей p и q.
*2. Напишите программу, которая шифрует и расшифровывает сообщения с помощью алгоритма RSA при небольших значениях открытого и секретного ключей.

§ 81 Стеганография

При передаче сообщений можно не только применять шифрование, но и скрывать сам факт передачи сообщения.

Стеганография — это наука о скрытой передаче информации путём скрытия самого факта передачи информации.

Древнегреческий историк Геродот описывал, например, такой метод: на бритую голову раба записывалось сообщение, а когда его волосы отрастали, он отправлялся к получателю, который брил его голову и читал сообщение.

Классический метод стеганографии — *симпатические* (невидимые) *чернила*, которые проявляются только при определенных условиях (нагрев, освещение, химический проявитель). Например, текст, написанный молоком, можно прочитать при нагреве.

Сейчас стеганография занимается *скрытием информации в текстовых, графических, звуковых и видеофайлах* с помощью программного «внедрения» в них нужных сообщений.

Простейшие способ — заменять младшие биты файла, в котором закодировано изображение. Причём это нужно сделать так, чтобы разница между исходным и полученным рисунками была неощутима для человека. Например, в чёрно-белом рисунке (256 оттенков серого) яркость каждого пикселя кодируется 8 битами. Если поменять 1–2 младших бита этого кода, «встроив» туда текстовое сообщение, фотография, в которой нет чётких границ, почти не изменится. При замене одного бита каждый байт исходного текстового сообщения хранится в младших битах кодов 8 пикселей. Например, пусть первые 8 пикселей рисунка имеют такие коды:

10101101	10010100	00101010	01010010	10101010	10101010	10101011	10101111

Чтобы закодировать в них код буквы «И» (11001000_2), нужно изменить младшие биты кодов:

10101101	10010101	00101010	01010010	10101011	10101010	10101010	10101110
1	1	0	0	1	0	0	0

Получателю нужно взять эти младшие биты и «собрать» их вместе в один байт.

Для звуков используются другие методы стеганографии, основанные на добавлении в запись коротких условных сигналов, которые обозначают 1 и 0 и не воспринимаются человеком на слух. Возможна также замена одного фрагмента звука на другой.

Для подтверждения авторства и охраны авторских прав на изображения, видео и звуковые файлы применяют *цифровые водяные знаки* — внедрённую в файл информацию об авторе. Они получили своё название от старых водяных знаков на деньгах и документах. Для того чтобы установить авторство фотографии, достаточно расшифровать скрытую информацию, записанную с помощью водяного знака.

Иногда цифровые водяные знаки делают видимыми (текст или логотип компании на фотографии или на каждом кадре видеофильма). На многих сайтах, занимающихся продажей цифровых фотографий, видимые водяные знаки размещены на фотографиях, предназначенных для предварительного просмотра.

Вопросы и задания

1. Что такое стеганография?
2. Какие методы стеганографии существовали до изобретения компьютеров?
3. Как можно добавить текст в закодированное изображение?
4. На чем основаны методы стеганографии для звуковых данных и видеоданных?
5. Что такое цифровые водяные знаки? Зачем они используются?

Подготовьте сообщение

«Водяные знаки в цифровую эпоху»

§ 82
Безопасность в Интернете

Угрозы безопасности

Если компьютер подключён к Интернету, появляются дополнительные угрозы безопасности. Атаку через сеть могут проводить злоумышленники и **боты** (программы-роботы), находящиеся в других городах и странах. Можно выделить три основные **цели злоумышленников**:

- **использование вашего компьютера** для взлома других компьютеров, атак на сайты, рассылки спама, подбора паролей и т. п.;
- **кража секретной информации** — данных о банковских картах, имён и паролей для входа на почтовые серверы, в социальные сети, платёжные системы;
- **мошенничество** — хищение чужого имущества путём обмана.

Первые две угрозы связаны, главным образом, с вредоносными программами: вирусами, червями и «троянцами», которые позволяют злоумышленнику управлять компьютером через сеть и получать с него данные.

Мошенничество процветает потому, что многие пользователи Интернета очень доверчивы и неосторожны. Классический пример мошенничества — так называемые **нигерийские письма**, приходящие по электронной почте. Пользователя от имени какого-то бывшего высокопоставленного лица просят принять участие в переводе крупных денежных сумм за границу, обещая выплачивать большие проценты. Если получатель соглашается, мошенники постепенно выманивают у него деньги.

Фишинг (англ. *phishing*, искажение слова *fishing* — рыбная ловля) — это выманивание паролей. Для этого чаще всего используются сообщения электронной почты, рассылаемые якобы от имени администраторов банков, платёжных систем, почтовых служб, социальных сетей. В сообщении говорится, что ваш счёт (или учётная запись) заблокирован, и даётся ссылка на сайт, который внешне выглядит как настоящий, но расположен по другому адресу (это можно проверить в адресной строке браузера). Неосторожный пользователь вводит своё кодовое имя и пароль, с помощью которых мошенник получает доступ к данным или банковскому счёту.

Антивирусы и последние версии браузеров содержат специальные модули для обнаружения подозрительных сайтов (*«антифишинг»*) и предупреждают о заходе на такой сайт. Кроме того, нужно помнить, что администраторы сервисов, а тем более работники банков, никогда не просят пользователя сообщить свой пароль по электронной почте.

Мошенничество может быть связано и с вредоносными программами. В 2010 г. несколько миллионов компьютеров в России было заражено троянской программой Winlock, которая блокировала компьютер и требовала отправить платное SMS-сообщение для снятия блокировки.

Правила личной безопасности

Вредоносные программы, распространяющиеся через Интернет, представляют серьёзную угрозу безопасности данных. Нужно помнить, что многих проблем можно избежать, если работать в Интернете только из-под ограниченной учётной записи (без прав администратора). Кроме того, желательно своевременно обновлять программное обеспечение; особенно важно устанавливать «заплатки», связанные с безопасностью.

Чтобы ваши пароли не украли, лучше не запоминать их в браузере (иногда они хранятся в открытом виде и могут быть украдены троянской программой). Заходя под своим именем в закрытую зону сайта с другого компьютера, нужно отмечать флажок **Чужой компьютер**, иначе тот, кто после вас откроет эту страницу на вашем компьютере, сможет получить доступ к вашим данным.

На многих сайтах предусмотрена возможность восстановления пароля по секретному вопросу. Этот вопрос нужно выбирать так, чтобы никто другой не знал ответа на него и, самое важное, не мог его выведать. Например, ответы на вопросы «Как звали вашу первую собаку?», «Какое ваше любимое блюдо?» и т. п. часто

можно найти на персональных страничках авторов в социальных сетях (в заметках, подписях к фотографиям и т. п.). Если мама автора имеет свою страничку, на ней, скорее всего, можно найти её девичью фамилию, поэтому вопрос «Какова девичья фамилия вашей матери?» тоже лучше не использовать.

Нужно понимать, что, размещая какую-то информацию в Интернете, вы делаете её доступной для широкого круга лиц, включая работодателей, полицию, официальные органы и даже преступников. Возможны ситуации, когда эта информация (личные данные, фотографии, высказывания на форумах и в блогах) может быть использована против вас, даже если она находится в закрытом разделе сайта.

Для передачи информации, которую необходимо сохранить в тайне, лучше применять шифрование (например, упаковать данные в архив с паролем).

Наибольший уровень безопасности обеспечивается при денежных расчётах через Интернет: вместо протокола HTTP используют защищенный протокол HTTPS (англ. *Hypertext Transfer Protocol Secure* — безопасный HTTP), который предусматривает шифрование данных (например, с помощью алгоритма RSA). Поэтому нужно проверять, чтобы адрес на странице ввода пароля в таких системах начинался с https://, а не с http://.

Современные молодые люди часто общаются в чатах, форумах и т. п., в том числе с теми, кого они не знают лично. Продолжение такого *виртуального* (компьютерного, электронного) знакомства в реальной жизни весьма опасно, потому что нередко участники чатов и форумов представляются не теми, кем они являются на самом деле.

Вопросы и задания

1. Какие угрозы безопасности существуют при подключении к Интернету?
2. Какие схемы интернет-мошенничества вам известны?
3. Какие меры безопасности нужно соблюдать при работе в Интернете?
4. Как обеспечивается безопасность обмена данными при денежных расчётах в Интернете?

Подготовьте сообщение

а) «Нигерийские письма»
б) «Фишинг»
в) «Безопасность финансовых расчётов в Интернете»

Практические работы к главе 10

Работа № 73 «Использование антивирусных программ»
Работа № 74 «Простые алгоритмы шифрования данных»
Работа № 75 «Современные алгоритмы шифрования и хэширования»
Работа № 76 «Использование стеганографии»

ЭОР к главе 10 на сайте ФЦИОР (http//fcior.edu.ru)

- Организация защиты при работе в сети
- Компьютерные вирусы и антивирусные программы
- Методы и средства защиты программных продуктов
- Сетевые и интернет-технологии. Компьютерная безопасность
- Информационная безопасность

Самое важное в главе 10

- Информационная безопасность — это защищённость информации от любых действий, в результате которых может быть искажена или утеряна информация либо нарушена её конфиденциальность, а владельцам или пользователям информации нанесён недопустимый ущерб.
- Основные угрозы информационной безопасности — выход оборудования из строя, неправомерный доступ к информации, вредоносные программы.
- Слабое звено любой системы информационной защиты — это человек.
- Шифрование — это преобразование открытой информации в зашифрованную, недоступную для понимания посторонних.
- Существуют системы шифрования с открытым ключом, при использовании которых все данные могут передаваться по незащищённому каналу связи.
- Цифровая подпись — это набор символов, который получен в результате шифрования сообщения с помощью личного секретного кода отправителя.

Навигационные значки

Уважаемые ученики! В работе с книгой вам помогут навигационные значки:

— важное утверждение или определение.

— вопросы и задания к параграфу.

— дополнительное разъяснение.

— задания для подготовки к итоговой аттестации.

— к каждой главе учебника рекомендуются:

1) электронные образовательные ресурсы (ЭОР) с сайта Федерального центра образовательных ресурсов (ФЦИОР):
http://fcior.edu.ru
Доступ к ЭОР из каталога ФЦИОР:
http://fcior.edu.ru/catalog/meta/4/mc/discipline%2000/mi/4.06/p/page.html.
Ресурсы размещены в алфавитном порядке, согласно названиям учебных тем;
2) электронное приложение к УМК:
   - практические, самостоятельные и контрольные работы на сайте поддержки учебника
     http://kpolyakov.spb.ru/school/probook/prakt.htm;
   - презентации для проведения уроков на сайте поддержки учебника
     http://kpolyakov.spb.ru/school/probook/slides.htm.

— Проектное или исследовательское задание.
В ходе выполнения проекта (исследования) вы можете:
- подготовить набор полезных ссылок с использованием веб-ресурсов;
- подготовить небольшое выступление с использованием презентации (5–7 мин.);
- оформить доклад и поместить его на сайт школьной конференции;
- подготовить видеоролик;
- разместить материалы проекта (исследования) в коллекции обучающих модулей по предмету на сайте школы.