№ 44 (333) 23–30 ноября 2001

ИНФОРМАТИКА

Еженедельная газета Объединения педагогических изданий «ПЕРВОЕ СЕНТЯБРЯ»
ПОДПИСКА: (095) 249-47-58

“Неограненный алмаз”

Год назад, в ноябре 2000 г., появился процессор Pentium 4 (с тактовыми частотами 1,4 и 1,5 ГГц)

“Pentium 4 — это неограненный алмаз среди микропроцессоров... Различные инновации в его архитектуре направлены на достижение высоких частот... Однако компании Intel еще предстоит доказать, что превосходство в тактовой частоте означает выигрыш в производительности” [1].

Окончание читайте на с. 32

Читайте в номере

ЧТО *"Информатика" предложит подписчикам в 2001/2002 учебном году*

Как и в прошлом году, мы предложим подписчикам **две серии специальных тематических выпусков — зимнюю, январскую, и летний трехмесячный марафон, состоящий из 12 номеров**. Летняя серия получит традиционное название "Жаркое лето-2002", а серия январских номеров будет называться "Снег идет".

Новый проект, который стартовал в октябре (№ 37/2001), связан с нашей новой рубрикой **"Страницы повышения квалификации"**. Учителю информатики как никому другому необходимо постоянно поддерживать свою "форму". Но далеко не всегда есть возможность пройти *хорошие* курсы повышения квалификации. "Информатика" предлагает подписчикам несколько лекционных курсов для повышения квалификации, которые публикуются на страницах газеты. Судя по отзывам на первые публикации, эти материалы не просто интересны, но очень полезны нашим читателям.

В первом полугодии 2002 года начнется публикация материалов, которые все мы давно ждем. Это **задачник по Excel**, который готовится специально для "Информатики" (в этом и предыдущем номерах мы представили несколько фрагментов нового задачника).

Еще одна **новая рубрика**, которая появится во втором полугодии, — **"Начните с простого"**. Статьи этой рубрики, написанные простым языком, ориентированы на тех, кто начинает знакомство с вопросом "с нуля". В них вы познакомитесь с множеством интересных и якобы сложных (ведь дело в том, как объяснить!) вопросов: Perl, PHP, Visual Basic, Delphi, SQL и многими другими.

План публикаций лекций курса "Современные педагогические технологии и частные методики обучения информатике" на "Страницах повышения квалификации".

Лекции читает И.Н. Фалина

Номер лекции	Номер газеты
1	37/2001
2	39/2001
3	41/2001
4	43/2001
5	45/2001
6	47/2001
7	5/2002
8	7/2002
9	9/2002
10	11/2002
11	13/2002
12	15/2002

Уважаемые читатели, пожалуйста, обратите внимание на то, что лекции 7—12 будут опубликованы в первом полугодии 2002 г.

План публикаций лекций курса "Олимпиады по информатике. Пути к вершине" на "Страницах повышения квалификации".

Лекции читает Е.В. Андреева

Номер лекции	Номер газеты
1	38/2001
2	40/2001
3	42/2001
4	44/2001
5	46/2001
6	48/2001
7	6/2002
8	8/2002
9	10/2002
10	12/2002
11	14/2002
12	16/2002

Уважаемые читатели, пожалуйста, обратите внимание на то, что лекции 7—12 будут опубликованы в первом полугодии 2002 г.

План публикаций лекций курса "Введение в специальность "Учитель информатики" на "Страницах повышения квалификации".

Лекции читает А.Г. Гейн

Номер лекции	Номер газеты
1	6/2002
2	8/2002
3	10/2002
4	12/2002
5	14/2002
6	16/2002

Уважаемые читатели, пожалуйста, обратите внимание на то, что все лекции будут опубликованы в первом полугодии 2002 г.

Уважаемые читатели! Газета "Информатика" распространяется только по подписке, поэтому не забудьте подписаться на нашу газету.

Ф. СП-1

Министерство связи Российской Федерации "Роспечать"

АБОНЕМЕНТ на газету **ИНФОРМАТИКА** — **32291** (индекс издания)

наименование издания | Количество комплектов

на 20__ год по месяцам

1	2	3	4	5	6	7	8	9	10	11	12

Куда (почтовый индекс) (адрес)

Кому (фамилия, инициалы)

ДОСТАВОЧНАЯ КАРТОЧКА

ПВ | место | литер — на газету **32291** (индекс издания)

ИНФОРМАТИКА (наименование издания)

Стоимость: подписки _____ руб. | переадресовки _____ руб. — Количество комплектов

на 20__ по месяцам

1	2	3	4	5	6	7	8	9	10	11	12

Куда (почтовый индекс) (адрес)

Кому (фамилия, инициалы)

Олимпиады по информатике. Пути к вершине

Лекции читает Е.В. Андреева

Лекция 4. Алгоритмы поиска в олимпиадных задачах

Данную тему следует рассматривать в двух аспектах. Во-первых, при решении различных олимпиадных задач алгоритмы поиска приходится использовать в качестве технических элементов для реализации собственно алгоритма решения той или иной задачи. Во-вторых, задача сама по себе может требовать построения оптимального в смысле определенных требований алгоритма поиска. Подход в каждом из двух упомянутых случаев должен быть различным.

Алгоритмы поиска в неупорядоченных одномерных массивах

Рассмотрим сначала нюансы реализации "технических" задач поиска, которые встречаются в различных алгоритмах. Начнем с, казалось бы, тривиальной задачи поиска элемента в неупорядоченном массиве. Во всех примерах, относящихся к поиску в одномерном массиве, будем использовать следующую переменную a:

```
a:array[0..N] of <скалярный тип>;
```

— при этом собственно элементы массива, которые мы будем рассматривать, пронумерованы от 1 до *N*, а нулевой элемент будем использовать как вспомогательный в случае необходимости. Конкретный же тип элемента в большинстве описываемых алгоритмов не важен, он может быть как любым числовым, так и символьным или даже строковым. Алгоритм поиска в массиве элемента, значение которого равно *K*, может выглядеть так:

```
i := 0;
repeat
  i := i + 1
until (i = N) or (a[i] = K);
if a[i] = K then write(i)
           else write(0)
{следующее неверно (!!!):
if i = N then write(0)
         else write(i) }
```

При отсутствии в массиве элемента с искомым значением *K* печатается значение нулевого индекса. Оказывается, что и такую достаточно простую программу можно упростить, заодно продемонстрировав так называемый "барьерный" метод, который часто применяется в программировании в том числе и олимпиадных задач. В данном случае он заключается в том, что мы можем занести в дополнительный элемент массива (например, нулевой) искомое значение *K*, избавившись тем самым в условии окончания цикла от проверки на выход за границу массива:

```
a[0] := K;
i := N;
while (a[i] <> K) do
  i := i - 1;
write(i)
```

Дело не только в том, что эта программа проще и эффективней. В ней практически невозможно сделать ошибку.

Другой полезный прием можно показать на задаче поиска максимального и минимального значений в массиве. Состоит он в том, что при любом поиске в массиве искать следует не значение искомого элемента, максимума, минимума и т.п., а его индекс. Тогда, решая одну задачу, мы, по сути дела, решаем сразу две: определяем не только наличие искомого элемента, значение максимума или минимума, но и их местоположение в массиве. Программа при этом не только не усложняется, но зачастую становится даже короче:

```
imax := 1;
imin := 1;
for i := 2 to N do
  if a[i] < a[imin] then imin := i else
  if a[i] > a[imax] then imax := i
```

Заметим, что в качестве начальных значений для минимума и максимума следует использовать именно значение первого элемента массива, а не максимальное и минимальное значения в типе элементов, поскольку последнее может привести к ошибке. Так, следующая программа не находит значение максимума для убывающего массива целых чисел (!!!):

```
max := -MaxInt - 1;
{минимальное значение типа integer}
min := MaxInt;
{максимальное значение типа integer}
for i := 1 to N do
  if a[i] < min then min := a[i] else
  if a[i] > max then max := i
```

Казалось бы, на этом рассмотрение алгоритмов поиска в неупорядоченном массиве можно завершить. Однако именно с одновременного поиска минимума и максимума можно начать класс алгоритмов поиска более эффективных, чем приведенные выше стандартные алгоритмы. Причем именно при решении олимпиадных задач их знание может пригодиться в пер-

План публикаций лекций курса "Олимпиады по информатике. Пути к вершине" на "Страницах повышения квалификации".

Номер лекции	Номер газеты
1	38/2001
2	40/2001
3	42/2001
4	44/2001
5	46/2001
6	48/2001
7	6/2002
8	8/2002
9	10/2002
10	12/2002
11	14/2002
12	16/2002

Уважаемые читатели, пожалуйста, обратите внимание на то, что лекции 7—12 будут опубликованы в первом полугодии 2002 г.

вую очередь. Итак, в приведенном выше алгоритме поиска максимального и минимального элементов в массиве в худшем случае выполняется $2N - 2$ сравнений. Покажем, что ту же задачу можно решить за $3\lceil N/2\rceil - 2$ сравнений[*]. Пусть у нас имеется четное число элементов. Разобьем их на пары и в каждой из $N/2$ пар за одно сравнение определим, какой элемент больше, а какой меньше. Тогда максимум можно искать уже любым способом только из наибольших элементов в парах, а минимум — среди наименьших. Общее число сравнений при этом равно

$$N/2 + 2(N/2 - 1) = 3N/2 - 2.$$

Для нечетного числа элементов элемент, не попавший ни в одну из пар, должен участвовать в поиске как максимума, так и минимума.

Еще большей эффективности можно добиться при одновременном поиске максимального и второго по величине числа. Для этого организуем поиск максимума по схеме "теннисного турнира", а именно: разобьем элементы на пары и определим в каждой из пар больший элемент, затем разобьем на пары уже эти элементы и т.д. В "финале" и будет определен максимум. Количество сравнений в таком алгоритме, как и в традиционной схеме, равно $N-1$. Однако максимальный элемент участвовал при этом в $\lceil \log_2 N\rceil$ сравнениях, причем одно из них проходило обязательно со вторым по величине элементом. Таким образом, теперь для поиска этого элемента потребуется всего $\lceil \log_2 N\rceil - 1$ сравнение (!!!). Попробуйте самостоятельно построить эффективный алгоритм для поиска уже первых трех по величине элементов.

В данном контексте можно поставить также задачу поиска i-го по счету элемента, называемого i-й *порядковой статистикой*. Кроме максимума и минимума, специфической порядковой статистикой является *медиана* — элемент с номером $(N + 1)/2$ для нечетных N и два соседних элемента для четных. Конечно, задачу поиска любой порядковой статистики можно решить, предварительно отсортировав массив. Но, как будет показано ниже, оптимальные по количеству сравнений универсальные (то есть пригодные для произвольных массивов) алгоритмы сортировки выполняют порядка $N\log_2 N$ сравнений. А задачу поиска i-й порядковой статистики можно решить, производя $O(N)$ сравнений. Алгоритм, который гарантирует эту оценку, достаточно сложен, он подробно изложен в [1, 2, 3]. Мы же приведем схему алгоритма, который в худшем случае не является линейным, однако обычно работает существенно быстрее, следовательно, именно его и нужно использовать в практике реального программирования, в том числе и олимпиадных задач:

```
Алгоритм Выбор(A, L, R, i)
{выбирает между L-м и R-м элементами массива A i-й по счету
 в порядке возрастания элемент}
1. if L = R then результат – A[L], конец;
2. Q := L + random(R - L + 1)
   {Q – случайный элемент между L и R}
3. Переставляем элементы от L до R в A так, чтобы сначала шли
   элементы, меньшие A[Q], а затем все остальные, первый элемент
   во второй группе обозначим K.
4. if i ≤ (K – L) then Выбор(A, L, K – 1, i)
                  else Выбор(A, K, R, i – (K – L))
5. конец.
```

Оптимальную реализацию пункта 3 можно заимствовать из алгоритма так называемой *быстрой сортировки* (см., например, [4]).

Наконец, рассмотрим задачу поиска в последовательности из N чисел, хранения которой не требуется вообще, следовательно, ее длина может быть очень велика. Пусть известно, что в этой последовательности встречаются по одному разу все числа от 0 до N, кроме одного. Это пропущенное число и требуется найти. Вот возможное решение такой задачи:

```
S := 0;
for i := 1 to N do
begin
  read(a); S := S + a
end;
writeln(N * (N + 1) div 2 - S)
```

Данную программу легко переписать так, чтобы она работала и для значений N, превосходящих максимальное представимое целое число. Для этого следует использовать переменные типа **extended**, а цикл **for** заменить на **while**. Используя аналогичный прием, попробуйте решить следующую задачу. Пусть N — количество чисел, нечетно и известно, что среди вводимых N чисел каждое имеет ровно одно равное себе, а одно число — нет. Найдите это число.

Указание. В данном случае можно воспользоваться свойством логической функции "исключающее или", обозначаемой в Паскале как **xor**: a **xor** b **xor** b ≡ a.

О других алгоритмах поиска в одномерных массивах можно прочитать, например, в [2, 3, 5, 6]. Особый интерес среди них представляет задача международной олимпиады по информатике 2000 года, посвященная нахождению медианы в массиве, причем не путем сравнения элементов между собой, а с помощью операции определения среднего элемента среди трех произвольных элементов массива, выполняемой библиотечным модулем, предоставленным участникам олимпиады.

[*] Обозначение $\lceil\ \rceil$ соответствует для неотрицательных чисел округлению до ближайшего целого числа, большего или равного выражению в указанных скобках, в отличие от целой части, где округление производится до ближайшего целого, меньшего или равного рассматриваемому выражению.

Поиск в упорядоченных массивах

Под упорядоченными в дальнейшем будут пониматься неубывающие массивы, если не оговорено иное. То есть $a[1] \le a[2] \le ... \le a[N]$.

Сначала приведем пример реализации широко известного алгоритма двоичного (бинарного) поиска элемента, равного K, в уже упорядоченном массиве. Оказывается, досрочный выход из цикла в случае нахождения элемента выигрыша по скорости практически не дает, а лишние проверки делают программу более громоздкой. Поэтому рекомендуется производить поиск, пока диапазон рассматриваемых элементов состоит более чем из одного элемента.

```
L := 1; R := N + 1;
while L < R do
begin
  m := (L + R)div 2;
  if a[m] < K then L := m + 1
              else R := m
end;
if a[m] = K then write(m) else write(0)
```

На полуфинале чемпионата мира по программированию среди студенческих команд вузов, проходившем в Санкт-Петербурге в 2000 году, предлагалась следующая обратная двоичному поиску задача (см. сайт **neerc.ifmo.ru**, на котором можно найти тесты и ответы к ним для указанной задачи). Известно, что алгоритм бинарного поиска, аналогичный приведенному выше, но заканчивающий свою работу в случае досрочного обнаружения элемента, за Q сравнений определил, что искомым является элемент с номером I. Какова могла быть при этом размерность массива (указать все допустимые диапазоны значений N)? Несмотря на кажущуюся сложность при заданных в условии ограничениях, задача решалась путем простого перебора различных значений N и обращения с каждым из этих значений к функции бинарного поиска. Конструктивный же подход к решению задачи намного сложнее. Однако и в случае подбора можно использовать немало интересных фактов. Например, длина каждого из диапазонов возможных значений N является степенью двойки, а каждый следующий диапазон не меньше предыдущего. Это позволяет значительно сократить количество рассматриваемых значений N. Кроме того, максимальное допустимое значение для N легко найти аналитически.

Рассмотрим теперь задачу поиска в упорядоченном массиве наибольшего "равнинного" участка. То есть требуется найти такое число p, что в массиве имеется последовательность из p равных элементов и нет последовательности из $p + 1$ равных по значению элементов (см., например, [7]). Оказывается, существует алгоритм решения этой задачи, количество операций в котором может оказаться существенно меньше, чем N. Так, если мы уже нашли последовательность из p_1 равных между собой чисел, то другую последовательность имеет смысл теперь рассматривать только если ее длина не менее $p_1 + 1$. Поэтому если a[i] — первый элемент предполагаемой новой подпоследовательности, то сравнивать его надо сразу с элементом a[i + p], где p — максимальная величина для уже рассмотренных подпоследовательностей из равных элементов. Приведем фрагмент программы, решающий данную задачу. В качестве результата мы получаем длину максимальной подпоследовательности p, номер элемента, с которого эта подпоследовательность начинается, k и значение элементов в найденной подпоследовательности:

```
p := 1; k := 1;
i := 1; f := false;
while i + p <= N do
  if a[i + p] = a[i] then
  begin
    p := p + 1; f := true
  end
  else if f then
         begin
           k := i; i := i + p; f := false
         end
       else i := i + 1;
writeln(p,' ',k, ' ',a[k])
```

В [7] можно найти еще одну интересную задачу под названием "Жулик на пособии" поиска в упорядоченных последовательностях уже практически какой угодно длины. Пусть имеются три упорядоченных по алфавиту списка из фамилий людей, получающих пособие по безработице в трех различных местах. Длина каждого из списков может быть как угодно большой (каждый из списков можно хранить в отдельном файле). Известно, что по крайней мере одно лицо фигурирует во всех трех списках. Требуется написать программу поиска такого лица, порядок количества операций в которой не будет больше, чем сумма длин всех трех списков. Приведем фрагмент программы, работающий с тремя файлами, обращение к элементам которых (они могут быть любого типа, в том числе и **string**, к элементам которого в Паскале применимы операции сравнения, чтения и записи) из программы происходит с помощью файловых переменных x, y, z типа **text**:

```
readln(x,p); readln(y,q); readln(z,r);
while not((p = q)and(q = r)) do
begin
  if p < q then readln(x,p)
  else if q < r then readln(y,q)
       else if r < p then readln(z,r)
end;
writeln(p);{p=q=r}
```

Покажите, что для поставленной задачи чтение из файла всегда будет производиться корректно и на каждом шаге цикла значение одной из трех переменных p, q, r будет изменено. Кроме того, докажите, что с помощью этой программы всегда будет найден именно минимальный из элементов, присутствующих во всех трех файлах.

Наконец, рассмотрим задачу поиска элемента, значение которого равно K, в двухмерном массиве, упорядоченном по строкам и столбцам: a[i,j] ≤ a[i,j+1], a[i,j] ≤ a[i+1,j], 1 ≤ i < N, 1 ≤ j < M. Данная задача решается за $M + N$ действий, а не за $M \cdot N$, как в произвольном массиве. Поиск следует начинать с элемента a[N,1]. Этот элемент самый маленький в своей

строке и самый большой в своем столбце (в математике подобные элементы называют "седловыми точками"). Поэтому если он окажется меньше, чем K, то из рассмотрения первый столбец можно исключить, а если больше — из рассмотрения исключается последняя строка и т.д. Вот возможная реализация этого алгоритма:

```
i := N; j := 1;
while (i > 0) and (j < M + 1) and
      (a[i,j] <> K) do
  if a[i,j] < K then j := j + 1
  else i := i - 1;
if (i > 0) and (j < M + 1) then writeln(a[i,j])
```

Программа могла бы быть еще короче и эффективнее, если бы в ней использовался упоминавшийся выше барьерный метод. В данном случае для организации барьера требуется дополнить массив нулевой строкой и $(M+1)$-м столбцом. Во все созданные барьерные элементы следует поместить значение K. Тогда условие в цикле можно сократить до следующего: `a[i,j] <> K`.

Алгоритмы поиска и задачи на взвешивания

Появление подобных задач, например, задачи определения фальшивой монеты, в контексте рассмотрения алгоритмов поиска не случайно. Во-первых, поиск и в этом случае осуществляется путем операций сравнения, правда, уже не только одиночных элементов, но и групп элементов между собой. Во-вторых, как будет показано ниже, в отличие от олимпиад по математике для младших школьников задачи на взвешивания вполне можно решать конструктивно.

В качестве примера рассмотрим задачу, предлагавшуюся на теоретическом туре одной из региональных олимпиад по информатике. Пусть у нас имеется 12 монет, одна из которых фальшивая, по весу отличающаяся от остальных монет, причем неизвестно, легче она или тяжелее. Требуется за три взвешивания определить номер фальшивой монеты (попробуйте решить эту задачу самостоятельно, и вы убедитесь, что это совсем не просто, а порой вообще кажется невозможным). Введем следующие обозначения. Знаком "+" будем обозначать монеты, которые во время текущего взвешивания следует положить на весы, причем если монета на весах уже была, то на ту же самую чашу, на которой эта монета находилась во время своего предыдущего взвешивания. Знаком "—" будем обозначать монеты, которые следует переложить на противоположную чашу весов, по отношению к той, на которой они находились (каждая в отдельности), заметим, что если монета на весах еще не была, то знак "—" к ней применен быть не может. Наконец, знаком "0" — монеты, которые в очередном взвешивании не участвуют. Тогда существует 14 различных способов пометки монет для трех взвешиваний:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14
+ + + + + + + + +  0  0  0  0  0 —
                                  первое взвешивание
+ + + - - - 0 0 0  +  +  +  0  0 —
                                  второе взвешивание
+ - 0 + - 0 + - 0  +  -  0  +  0 —
                                  третье взвешивание
```

Из полученной таблицы вычеркнем 2 столбца так, чтобы в каждой из трех строк количество ненулевых элементов оказалось четным (ведь мы не можем во время одного взвешивания положить на две чаши весов нечетное число монет). Это могут быть, например, столбцы 4 и 14. Теперь будем взвешивать 12 монет так, как это записано в оставшихся 12 столбцах. То есть в первом взвешивании будут участвовать 8 произвольных монет, во втором — 3 монеты следует с весов убрать, 2 — переложить на противоположные по отношению к первому взвешиванию чаши весов и 3 монеты положить на весы впервые (на свободные места так, чтобы на каждой из чаш вновь оказались по 4 монеты). Согласно схеме проведем и третье взвешивание, опять располагая на каждой чаше весов по 4 монеты. Результат каждого взвешивания в отдельности никак не анализируется, а просто записывается. При этом равновесие на весах всегда кодируется нулем, впервые возникшее неравновесное состояние — знаком "плюс", если при следующем взвешивании весы отклонятся от равновесия в ту же самую сторону, то результат такого взвешивания также кодируется плюсом, а если в другую сторону — то минусом. Например, записи "=<<" и "=>>" кодируются как "0++", а записи "<=>" и ">=<" — как "+0–". Так как мы не знаем, легче или тяжелее остальных монет окажется фальшивая, то нам важно, как изменялось состояние весов от взвешивания к взвешиванию, а не то, какая именно чаша оказывалась тяжелее, а какая легче. Поэтому два на первый взгляд различных результата трех взвешиваний в этом случае кодируются одинаково. После подобной записи результатов взвешиваний фальшивая монета уже фактически определена. Ею оказывается та, которой соответствует такой же столбец в таблице, как и закодированный нами результат трех взвешиваний. Для первого из примеров это монета, которая участвовала во взвешиваниях по схеме, указанной в 10-м столбце таблицы, а для второго — в 8-м. В самом деле, состояние весов в нашей задаче меняется в зависимости от того, где оказывается фальшивая монета во время каждого из взвешиваний. Поэтому монета, "поведение" которой согласуется с записанным результатом взвешиваний, такой результат и определяет.

Анализ таблицы показывает, что эту задачу можно решить не только для 12, но и для 13 монет. Для этого следует исключить из рассмотрения любой не содержащий нулей столбец, например, все тот же четвертый. В остальном все действия остаются неизменными. Для произвольного числа монет $N > 2$ количество взвешиваний определяется по формуле $\lceil \log_3(2 \cdot N + 1) \rceil$ (за одно взвешивание задача не разрешима ни для какого количества монет!!!), но подход к решению задачи при этом не изменится.

Попробуйте теперь решить задачу, которая предлагалась в 1998 году на уже упоминавшемся выше полуфинале чемпионата мира по программированию среди студенческих команд вузов. В ней также требовалось определить номер фальшивой монеты, вес которой отличался от остальных. Но все взвешивания уже были проведены, а их результаты записаны. Число взвешиваний

являлось входным параметром в задаче (оно могло быть и избыточным по сравнению с описанным выше оптимальным алгоритмом определения номера фальшивой монеты). При этом в каждом из взвешиваний могло участвовать любое четное количество имеющихся монет (сколько и какие именно — известно). Результаты записывались с помощью символов "<", ">" и "=".

Еще одна задача на взвешивания рассмотрена в [8]. В общем случае в ней требуется найти набор из минимального количества гирь, такой, что с его помощью можно взвесить любой груз, весящий целое число килограммов, в диапазоне от 1 до N кг. При необходимости гири можно располагать на обеих чашах весов. Так, для $N = 40$ это гири 1, 3, 9 и 27 кг.

Поиск подстроки в строке

Формализовать эту задачу можно следующим образом. Пусть задан массив s из N элементов (строка) и массив p из M элементов (подстрока), причем $0 < M \le N$. Требуется обнаружить первое непрерывное вхождение p в s. Эта задача на практике встречается очень часто. Так, в большинстве текстовых редакторов реализована операция поиска по образцу, которая практически полностью совпадает с описанной задачей. Если размер массива s — N не превосходит 255, а тип его элементов — `char`, то в Турбо Паскале такой поиск можно выполнять с помощью стандартной функции `Pos(p,s)`. Однако в общем случае ее приходится реализовывать самостоятельно. Прямой поиск, основанный на последовательном сравнении подстроки сначала с первыми M символами строки, затем с символами с номерами 2 .. $M + 1$ и т.д., в худшем случае произведет порядка $N \cdot M$ сравнений. Но для этой задачи известен алгоритм Боуера и Мура (см., например, [5]), который для произвольных строк выполняет не намного более N/M сравнений. То есть разница в вычислительной сложности составляет M^2 (!!!). Рассмотрим последний алгоритм, на примере которого также можно показать, что использование небольшого количества дополнительной памяти (в данном случае вспомогательного массива, размер которого равен размеру алфавита строк) позволяет существенно ускорить выполнение программы.

Перед фактическим поиском для всех символов, которые могут встретиться в строке, вычисляется и запоминается в массиве `d` расстояние от самого правого вхождения этого символа в искомую подстроку до ее конца. Если же какого-то символа из алфавита строки в подстроке нет, то такое расстояние считается равным длине подстроки M. Посимвольное же сравнение подстроки с некоторым фрагментом строки производится не с начала, а с конца искомой подстроки (образца). Если какой-либо символ образца не совпадает с соответствующим символом фрагмента строки, а `x` — последний символ фрагмента строки, то образец можно сдвинуть вдоль строки вправо на `d[x]` символов. Если большинство символов в строке отличны от символов подстроки, то сдвиг будет происходить на M элементов, что и обеспечит приведенную выше сложность алгоритма. Покажем работу алгоритма на примере поиска слова `коала` в строке:

```
кокаколулюбитикоала.
коала
    коала
     коала
          коала
               коала
```

Здесь подчеркнуты символы, которые участвовали в сравнениях. Сдвиги определялись такими значениями массива `d: d['к'] = 4, d['л'] = 1, d['ю'] = 5`. Если бы последней в рассматриваемом фрагменте строки оказалась буква `а`, то величина сдвига была бы равна 2, так как в образце есть еще одна такая буква, отстоящая от конца на 2 символа, а при ее отсутствии сдвиг был бы равен 5. Приведем теперь возможную реализацию описанного алгоритма, для простоты считая, что размер подстроки не превосходит 255, что не снижает общности этой программы:

```
const nmax = 10000;
var p : string; {подстрока}
  s : array[1..nmax] of char; {строка}
  d : array[char] of byte; {массив сдвигов}
  c : char;
  m,i,j,k : integer;
begin
  …{задание строки и подстроки}
  m := length(p);{длина подстроки}
  for c := chr(0) to chr(255) do d[c] := m;
  for j := 1 to m - 1 do d[p[j]] := m - j;
  {массив d определен}
  i := m + 1;
  repeat {выбор фрагмента в строке}
    j := m + 1; k := i;
    repeat {проверка совпадения}
      k := k - 1; j := j - 1
    until (j < 1) or (p[j] <> s[k]);
    i := i + d[s[i - 1]];{сдвиг}
  until (j < 1) or (i > nmax + 1);
  if j < 1 then write(k + 1) else write(0)
end.
```

Приведенный алгоритм не дает выигрыша только в одном случае — когда количество частичных совпадений искомой подстроки с фрагментами текста достаточно велико. Это возможно, например, при чрезвычайной ограниченности алфавита, из символов которого составляются строки. Тогда следует применять алгоритм Кнута — Мориса — Пратта, описанный в [5], или комбинацию из двух алгоритмов.

Рассмотренную проблему не следует путать с такой задачей. Пусть заданы массив s из N элементов и массив p из M элементов, причем $0 < M \le N$. Требуется выяснить, можно ли из первого массива вычеркнуть некоторые члены так, чтобы он совпал со вторым. Число операций в данном случае имеет порядок $N + M$.

Литература

1. *Ахо А.А., Хопкрофт Д.Э., Ульман Д.Д.* Структуры данных и алгоритмы. М.: Вильямс, 2000.
2. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 2000.
3. *Окулов С.М.* Основы программирования. "Информатика" № 27, 2001.
4. *Окулов С.М.* Сортировка и поиск. "Информатика" № 35, 2000.
5. *Вирт Н.* Алгоритмы и структуры данных. М.: Мир, 1989.
6. *Шень А.* Программирование: теоремы и задачи. М.: МЦНМО.
7. *Грис Д.* Наука программирования. М.: Мир, 1984.
8. *Андреева Е., Фалина И.* Системы счисления и компьютерная арифметика. М.: Лаборатория базовых знаний, 2000.

Решение задач XIII международной олимпиады

Е.В. Андреева,
Москва

Окончание. См. № 37, 40, 42, 43/2001

Double Crypt

Улучшенный стандарт шифрования (УСШ) основан на новом алгоритме шифрования. Этот алгоритм работает с тремя блоками информации — p, k и c, размер каждого из которых равен 128 битам. По заданному сообщению p и ключу k функция шифрования E возвращает зашифрованный блок c:

$$c = E(p, k).$$

Обратной функцией к функции шифрования E является функция дешифрования D, то есть справедливо следующее:

$$D(E(p, k), k) = p, \qquad E(D(c, k), k) = c.$$

Назовем двойным УСШ алгоритм, в котором функция E применяется последовательно два раза, используя два независимых ключа — k_1 и k_2, — сначала k_1, затем k_2:

$$c_2 = E(E(p, k_1), k_2).$$

В этой задаче также задается число s. Во всех ключах только $4 \cdot s$ левых битов являются значимыми, остальные $(128 - 4 \cdot s)$ правых битов всегда заполнены нулями.

Вам необходимо восстановить пары ключей, которые использовались при шифровании некоторого сообщения двойным УСШ. Вам дается как исходный текст сообщения p, так и соответствующий ему результат двойного шифрования — текст c_2, а также значение s, определяющее структуру ключей шифрования.

Вам дана библиотека, в которой имеются функции шифрования и дешифрования УСШ.

Вы должны представить для оценки восстановленные ключи, а не восстанавливающую их программу.

Ввод

Вам даны десять входных текстовых файлов с именами от `double1.in` до `double10.in`. Каждый из этих файлов состоит из трех строк. Первая строка содержит целое число s, вторая строка — исходный текст сообщения p, третья строка — зашифрованный текст c_2, полученный из p алгоритмом двойного шифрования УСШ. Тексты исходного и зашифрованного сообщений записаны в виде строк из 32 шестнадцатеричных цифр (0 .. 9 , A .. F). В данной вам библиотеке есть функция для перевода таких строк в 128-битный блок информации. Для всех входных файлов решение задачи существует.

Вывод

Вы должны послать на проверку десять выходных файлов, соответствующих данным входным файлам. Каждый выходной файл должен состоять из трех строк. Первая строка должна содержать текст

```
#FILE double I,
```

где `I` обозначает номер соответствующего входного файла. Вторая строка должна содержать ключ k_1, записанный в виде 32 шестнадцатеричных цифр ('0'..'9', 'A'..'F'), а третья — записанный в том же формате ключ k_2, при этом должно выполняться

$$c_2 = E\,(E(p, k_1), k_2).$$

В данной вам библиотеке есть функции для перевода 128-битных блоков информации в строки из 32 шестнадцатеричных цифр. Если задача имеет несколько решений, следует отослать для проверки одно из них.

Пример

В качестве примера ниже представлен входной файл с номером 0.

DOUBLE0.IN

```
1
00112233445566778899AABBCCDDEEFF
6323B4A5BC16C479ED6D94F5B58FF0C2
```

Возможный выходной файл

```
#FILE double 0
A0000000000000000000000000000000
70000000000000000000000000000000
```

Библиотека для FreePascal:

```
type
  HexStr = string [32];   { only '0'..'9', 'A'..'F' }
  Block  = array [0..15] of byte;   { 128 bits }

procedure HexStrToBlock(const hs : HexStr; var b : Block);
procedure BlockToHexStr(const b : Block; var hs : HexStr);
procedure Encrypt(const p, k : Block; var c : Block); { c = E(p,k) }
procedure Decrypt(const c, k : Block; var p : Block); { p = D(c,k) }
```

Библиотека для GNU C/C++:

```
typedef char HexStr[33];
/* '0'..'9', 'A'..'F', '\0' - конец строки */
typedef unsigned char Block[16]; /* 128 бит */
void hexstr2block(const HexStr hs, /* результат */ Block b );
void block2hexstr (const Block b, /* результат */ HexStr hs );
void encrypt (const Block p, const Block k, /* результат*/ Block c); /* c = E(p,k) */
void decrypt (const Block c, const Block k, /* результат*/ Block p); /* p = D(c,k) */
```

Ограничения

Для значений s справедливо соотношение $1 \le s \le 5$.

Решение

Очевидно, что перебором всех возможных пар ключей решить данную задачу невозможно. В данном случае применяется так называемый метод "встречи в центре". Заключается он в том, что мы перебираем все возможные значения первого ключа, шифруем сообщение один раз и результат шифровки помещаем в таблицу. Затем перебираем все возможные значения второго ключа и дешифруем с их помощью результат двойного кодирования. Полученное в результате дешифрования сообщение анализируем по уже имеющейся таблице, и если в ней есть такое же сообщение, то это и означает, что с помощью рассматриваемого второго ключа и первого ключа, с помощью которого сообщение в таблицу было занесено, и было осуществлено двойное шифрование исходного текста.

Таким образом, мы получаем не квадратичный, а линейный по количеству различных ключей алгоритм. Проблема заключается только в том, что размер таблицы для хранения всех зашифрованных сообщений слишком большой. Кроме того, операции помещения в таблицу и сравнения сообщения с уже имеющимися в таблице должны осуществляться очень быстро, желательно за время $O(1)$. В таком случае нам может помочь операция хеширования. То есть каждому закодированному сообщению ставится в соответствие некоторое, по возможности уникальное, число из интересующего нас диапазона. Возникающие при этом коллизии (двум сообщениям соответствует одно и то же число) можно разрешать, например, линейным способом — искать в таблице первую свободную ячейку за ячейкой, соответствующей данному хеш-ключу. Чтобы такой процесс не приводил к длительному поиску свободной ячейки в таблице, нужно иметь достаточное их количество, поэтому размер таблицы имеет смысл делать в два раза больше, чем общее количество возможных ключей для шифрования.

Оценим размер таблицы в нашей задаче. В нашем случае мы имеем не более 2^{20} различных ключей, то есть таблица должна иметь по крайней мере 2^{21} ячеек. В каждой ячейке можно хранить как соответствующий ей первый ключ, так и полученное с его помощью зашифрованное сообщение. Однако для получения результата за линейное время вполне достаточно хранить только сам ключ, а соответствующее ему сообщение получать в момент сравнения с результатом дешифрования с помощью второго ключа. Время работы программы при этом увеличится всего в 1,5 раза.

Так как предлагаемая для шифрования функция достаточно эффективна и применяется в реальных задачах шифрования, правда, с тройным, а не двойным кодированием, то в качестве хеш-значения (хеш-ключа) можно взять, например, первые 20 бит закодированного сообщения, определяя по ним номер ячейки в таблице из 2^{21} элементов для хранения соответствующего ключа. Так, если первые 20 бит соответствуют числу i, то ключ можно попытаться поместить в ячейку $2 \cdot i$, а если она уже занята, то последовательно просматривать ячейки $2i + 1$, $2i + 3$ и т.д. При наличии большой оперативной памяти в компьютере и размер таблицы, и размер хеш-ключа можно увеличить; например, при 64 Мб хеширование можно проводить по первым 24 битам в сообщении, что уменьшит число возникающих коллизий.

Лабораторные работы по алгоритмизации

Д.П. Береговой,
Краснодарский край, станица Каневская

Гуманизация образования может позволить себе представить изучение основ информатики в средней общеобразовательной школе без ознакомления учащихся с началами алгоритмизации и программирования. Однако *гуманитаризация* образования требует, чтобы мы настояли на привитии нашим детям хотя бы принципов умения думать.

Из разговора учителей информатики

Предисловие (для преподавателей)

Предлагаемый комплект лабораторных работ (ЛР) предназначен для изучения простейших технологических приемов построения алгоритмов и выработки основных навыков их реализации на ЭВМ. Стратегической целью этой части курса является формирование у школьников алгоритмического стиля мышления при решении сначала специальных, а впоследствии и общих задач.

При подборе заданий лабораторных работ особое внимание уделялось тому, чтобы суть их решения была доступной для понимания. При этом работа должна иллюстрировать тот или иной сугубо технологический способ решения задачи и, соответственно, построения алгоритма. Каждая работа направлена на освоение одного алгоритмического приема. В соответствии с этим принципом в началах алгоритмизации выделены следующие темы:

— ввод-вывод данных (ЛР № 1);
— разветвляющиеся алгоритмы (ЛР № 2);
— циклические алгоритмы (ЛР № 3);
— алгоритмические модули (ЛР № 4);
— простейшая графика (ЛР № 5);
— элементарная обработка строк символов (ЛР № 6).

Предполагается, что каждая тема осваивается за несколько этапов.

1. Сначала, естественно, изучается теоретический материал. Учащимся объясняются назначение приема, типичные случаи его применения, способы формализации и реализации. Думается, у большинства читателей не возникнет трудностей с источниками информации для подготовки такого сорта урока. Мы же используем учебное пособие [1]. Для формализации алгоритмов используется язык блок-схем, который, по мнению автора, является наиболее универсальным и наглядным. Что касается реализации алгоритмов, то, как показала многолетняя практика, выбор языка программирования совершенно не важен, лишь бы он позволял структурировать программу. Поэтому с равным успехом можно пользоваться языками Бейсик, Паскаль или Си. Собственно, одна из наиболее важных идей, которая должна быть привита учащимся, заключается в том, что основой решения любой задачи является алгоритм (например, в виде его графического выражения — блок-схемы, что, на наш взгляд, предпочтительнее), а его реализация — дело аккуратной техники кодирования с использованием любых доступных средств.

2. Далее на демонстрационных примерах подробно анализируются наиболее типичные приемы алгоритмического решения поставленной задачи.

3. Третья часть предусматривает самостоятельное решение задач учащимися, но под управлением преподавателя, так как каждая задача требует применения какого-либо технологического нюанса, далеко не всегда очевидного. Как правило, на уроке рассматриваются 1—2 задания. Оставшиеся задания задаются на дом. При этом можно воспользоваться задачами из Приложения 1.

4. И, наконец, проводится зачетная работа по изученной теме. Как правило, задание выдается заранее, с расчетом, чтобы отчет был подготовлен учащимися дома. Оценивать работу целесообразно в процессе защиты. То есть после демонстрации результатов работы программы (на подготовленном преподавателем наборе тестовых заданий) учащийся не только представляет отчет (Приложение 3), но и должен ответить на несколько вопросов преподавателя (примерные вопросы даны после каждой работы). Как правило, достаточно 3—5 вопросов. Вопросы выбираются либо по усмотрению преподавателя, либо используется "билетная технология". Общая оценка, таким образом, отражает как практическую, так и теоретическую подготовку ученика. Отчеты лучше оформлять в отдельной тетради.

Нумерация заданий в лабораторных работах сквозная:
первая цифра обозначает номер работы;
вторая — тип работы:
1 — демонстрационная,
2 — самостоятельная,
3 — контрольная;
далее следует литера задачи.

Работы № 5 и № 6 (графика и обработка символов) не содержат контрольной части, так как, по всей видимости, для указанных тем невозможно разработать 20—30 вариантов однотипных и, главное, одноуровневых заданий. Поэтому практическая часть работы оценивается по качеству выполнения самостоятельных заданий и проверке теоретических знаний.

Особое внимание нужно уделить качеству оформления лабораторной работы. Поскольку для большинства учащихся понятие технической документации является весьма отвлеченным, то у преподавателя есть прекрасный повод продемонстрировать жесткие промышленные требования к подобного рода документам. Нельзя допускать вольности в оформлении работ. Одна из возможных форм отчета о лабораторной работе приведена в Приложении 3.

Таким образом, на изучение темы затрачивается 3—6 часов, что вполне приемлемо для усвоения соответствующего учебного материала и отработки общих навыков алгоритмизации стандартных задач.

После выполнения работы № 3 (циклические структуры алгоритмов) имеет смысл проверить навыки применения стандартных структур алгоритмов. Примерное содержание соответствующей контрольной работы приведено в Приложении 2.

Литература

1. *Береговой Д.П.* Основы информатики: Учебное пособие для X—XI классов. Краснодар: ОИПЦ, 2001.

2. *Зельднер Г.А.* Microsoft Basic Professional Development System 7.1. Руководство программиста. М.: ABF, 1994.

В этом номере публикуются первые две лабораторные работы. Оставшиеся лабораторные работы и приложения будут опубликованы в следующих (декабрьских) номерах.

Предисловие (для учащихся)

Все вокруг только и говорят о компьютерах, программах и странных людях, которых называют программистами. Хотя, если спросить 100 человек: "Чем занимается программист?" — внятно на этот вопрос ответят не более десяти, а правильно — и того меньше. А между тем, устанавливая на определенное время будильник или составляя свой план действий на ближайшие выходные, мы все становимся программистами. В первом случае мы программируем маленького пластмассово-металлического робота, а во втором — сами себя.

Но, оказывается, не так сложно научить программировать и персональный компьютер. Конечно, предлагаемый вам набор лабораторных работ вовсе не предназначен для того, чтобы сделать из вас профессиональных программистов. Это невозможно в рамках средней школы, да и не нужно — школа-то общеобразовательная. Но представление о том, что такое программирование, вы получите. И для этого не нужно быть семи пядей во лбу. Достаточно лишь добросовестности и аккуратности. Каждая работа, которая вам предлагается, состоит из трех частей:

— первая — демонстрационная;
— вторая — самостоятельная;
— третья — контрольная.

Предполагается, что для их выполнения потребуется внимательно изучить теорию и уметь отвечать на вопросы, образцы которых приведены в конце каждой работы. Но будет просто здорово, если вы попробуете сами себе задавать вопросы и находить на них ответы.

При выполнении самостоятельной работы вы вправе использовать любую литературу, подсказки товарищей, советы учителя. Обратите внимание на оформление контрольной работы (Приложение 3). Помните, что вы работаете с техническим документом и никаких вольностей здесь быть не может. Для отчетов по контрольным работам лучше завести отдельную тетрадь!

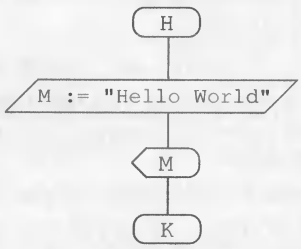
Лабораторная работа № 1

Организация ввода-вывода данных в алгоритмах

Цель работы. Изучить основные приемы ввода-вывода данных в алгоритмах и технологию организации диалогового режима работы ЭВМ.

Техническое задание 1.1а. Разработать алгоритм организации автоматического ввода строки символов `"Hello World!"` и вывода этой строки на экран дисплея.

Алгоритм 1.1а. Это первый, самый важный для будущего программиста алгоритм. Так сказать, крик новорожденного ☺. Но ничего, кроме ввода и вывода данных, в алгоритме нет, поэтому в нем только два блока: ввод и вывод (не считая начала и конца).



— Для автоматического ввода строки символов используем переменную символьного типа, которой и будет присвоено указанное значение.

— Вывод согласно ТЗ осуществляется на дисплей. Поэтому в алгоритме нужно использовать соответствующий блок-символ, в котором будет указано имя переменной, значение которой выводится на экран.

Программа 1.1а. Для ввода информации в одну переменную используем оператор присваивания. Вначале не забудьте определить переменную как символьную! Для этого используется служебное слово `DIM`. Объявление переменной выглядит следующим образом:

`DIM` *<идентификатор>* `AS` *<тип переменной>*

Для символьной переменной, как это имеет место в нашем случае, используется служебное слово `STRING`.

Вывод тоже достаточно прост: блок-символ заменяется на оператор `PRINT`. Так что первая ваша программа будет выглядеть так, как показано ниже. Кстати, полезно начинать программу с очистки экрана (оператор `CLS`). Нельзя заниматься высокоинтеллектуальным трудом (а то, что мы сейчас делаем, без сомнения, является таковым) без предварительной уборки рабочего места. Поздравляю с приобщением к сонму программистов!

```
CLS
DIM Message AS STRING
Message = "Hello World!"
PRINT Message
END
```

Техническое задание 1.1б. Дана группа чисел {—10, 5, 3, 0}. Разработать алгоритм автоматического ввода этой группы чисел в массив. Данные вывести на дисплей: сначала в строку, потом в столбец.

Алгоритм 1.1б. Кроме ввода и вывода данных, в задаче ничего не требуется, так что алгоритм будет состоять из двух операций: автоматического ввода и вывода на дисплей. Особенностью задачи является использование массива.

Поскольку данные в алгоритме должны быть внесены в массив, он должен быть предварительно определен. Всего нужно ввести четыре числовых значения, соответственно и массив объявляется как числовой, одномерный, размером в 4 элемента. Пусть его имя будет `N`. Не забудьте, что по умолчанию элементы массива индексируются с 0! Для обозначения автоматического ввода бу-

дем использовать стандартный блок-символ, внутри которого запишем имена переменных. Здесь это будут элементы массива.

В общем случае формат вывода данных в алгоритме не указывается и реализуется уже в программе средствами языка программирования, поэтому в блок-схеме изображен стандартный дисплей с именами переменных внутри.

Программа 1.1б. Здесь показаны приемы ввода-вывода данных, которые нужно усвоить, поскольку без этих операций не обходится ни одна программа.

Для объявления массива используется то же служебное слово DIM, после чего указываются имя массива и количество элементов. Если вы забудете объявить массив, то скорее всего программа остановится с сообщением об ошибке.

Так как все элементы массива являются целыми числами, имеет смысл и массив объявить целочисленным. Тем самым экономится память и автоматически контролируется тип данных. Да и вообще объявление типа данных — это правило хорошего тона в программировании. В Бейсике для этого используется служебное слово INTEGER или суффикс % у имени массива.

Для автоматического ввода данных в массив можно, конечно, воспользоваться, как и раньше, операцией присвоения. Но для ввода списка в Бейсике (и только в Бейсике) имеется "хитрый" оператор DATA-READ. Следите за тем, чтобы количество значений данных и переменных соответствовало друг другу.

Чтобы вывести данные в строку, нужно в операторе PRINT перечислить выводимые переменные через запятую или точку с запятой. В первом случае данные будут выведены через пробелы, во втором — без пробелов. Так что получается пусть и примитивное, но все же упорядочивание вывода данных.

Вывод данных в столбец смотрится гораздо приятнее. Присмотритесь к синтаксической конструкции, которая позволяет вывести не только значения переменных, но и их названия.

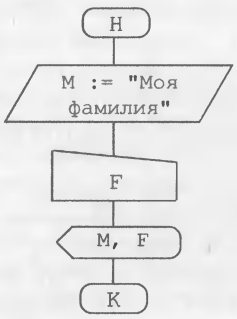
```
CLS
DIM N(3) AS INTEGER
DATA -10, 5, 3, 0
READ N(0), N(1), N(2), N(3)

PRINT N(0), N(1), N(2), N(3)
PRINT "N(0)="; N(0)
PRINT "N(1)="; N(1)
PRINT "N(2)="; N(2)
PRINT "N(3)="; N(3)
END
```

Техническое задание 1.1в. Разработать алгоритм и программу ввода своей фамилии в переменную и вывода ее на экран дисплея в формате:

Моя фамилия: <*Фамилия*>

Алгоритм 1.1в. Здесь также, кроме ввода-вывода данных, ничего нет. Особенностью этой задачи является то, что данные вводятся в двух режимах: автоматическом и ручном. Тем самым мы организуем диалоговый режим работы ЭВМ. Она нас спрашивает, а мы ей отвечаем.



По условию задачи нам придется иметь дело с двумя символьными величинами: константой "Моя фамилия" и переменной, в которой будет содержаться конкретная фамилия. Для сообщения введем переменную M, а для ввода фамилии — F.

Так как значение сообщения есть величина неизменяемая, то вводим ее в автоматическом режиме. Фамилия же будет каждый раз изменяться, значение ее заранее неизвестно, так что необходимо организовать ее ручной ввод с клавиатуры. Появляется новый блок-символ, но его легко запомнить. При некотором воображении можно представить, что это вид клавиатуры сбоку.

Вывод значений переменных производится все тем же оператором PRINT.

Программа 1.1в. В программе используется новый оператор INPUT. Это в переводе на русский означает "ввести". В данном случае значение переменной с помощью клавиатуры вводится в эту самую переменную.

Как мы договорились, сообщение "Моя фамилия" будет находиться в символьной константе. В программе этому соответствует описание

```
CONST Message$ = "Моя фамилия: "
```

Оператор INPUT используется в программе в полной форме. То есть содержит подсказку пользователю о том, что требуется ввести.

Вывод данных осуществляется, как и ранее, оператором PRINT. Поскольку и сообщение, и значение переменной нужно вывести в одну строку, то их идентификаторы в операторе разделяются запятой (или, что то же самое, точкой с запятой).

Для красоты оформления перед выводом данных экран еще раз очищается.

```
CLS
CONST Message$ = "Моя фамилия: "
DIM Family AS STRING

INPUT "Введите фамилию: ", Family
CLS
PRINT Message$; Family
END
```

Техническое задание 1.2а. Постройте алгоритм ввода и вывода на дисплей простейшей адресной книги. Данные должны храниться в двух массивах: в первом — фамилия и инициалы, во втором — адрес. Вывод данных организовать в строку в формате:

<*Фамилия*> <*Адрес*>

Рекомендации. Поскольку задание рассчитано на самостоятельное выполнение, ограничимся несколькими советами.

1. Для начала возьмите адреса нескольких знакомых: 4—5, больше не надо.

2. Помните, что массивы нужно предварительно объявить: они должны быть символьными, одномерными. Если это вас не испугает, можете попытаться организовать более продвинутый вариант, использовав двухмерный массив. В нем будут 4—5 строк (по количеству адресов) и два столбца. В первый столбец записывайте фамилию, во второй — адрес.

3. Ввод организуйте автоматически.

4. Не забудьте синтаксис оператора PRINT при выводе данных в строку. Здесь лучше всего использовать вывод через запятую, чтобы разделить фамилию и адрес на колонки. Получится что-то вроде таблицы.

5. Сравните свой алгоритм с алгоритмом 1.16. Внешний вид вашего произведения должен быть точно таким. Поменяется только содержание. То же самое касается и программы.

Техническое задание 1.26. Разработайте алгоритм расчета длины окружности и площади круга произвольного радиуса. Результат выведите на дисплей в формате:

Для радиуса <*R*> м
Длина окружности <*C*> м
Площадь круга <*S*> кв. м

Рекомендации. Изюминка задания состоит в том, что, кроме ввода-вывода данных, необходимо еще организовать и расчеты нужных величин. Для тех, кто забыл, напомним:

$$C_{окр} = 2\pi R,\ S_{кр} = \pi R^2.$$

1. В расчетах будет фигурировать постоянная — число

$$\pi \approx 3{,}14159$$

Понятно, что ее нужно ввести автоматически. Придумайте ей идентификатор. Рекомендуем использовать имя PI. Объявите эту переменную как константу (см. программу 1.1в).

2. Ключевое слово — *произвольный* — должно сразу натолкнуть на мысль, что радиус будет вводиться с клавиатуры в ручном режиме.

3. Структура вашего алгоритма должна быть похожа на алгоритм 1.1в, с той разницей, что между вводом и выводом появится блок расчета $C_{окр}$ и $S_{кр}$. На всякий случай напомним, что в блок-схемах алгоритмов действия обозначаются прямоугольником. Стало быть, все расчетные формулы должны быть записаны в прямоугольниках. В нашем случае в одном прямоугольнике запишутся две формулы.

4. Обратите внимание! При вычислении площади лучше воспользоваться операцией умножения, чем операцией возведения в степень, поскольку операция возведения в степень имеет большую погрешность вычислений.

5. Поэкспериментируйте с числами, символами и знаками препинания, добиваясь красивого оформления результатов расчета. Полезно вернуться к программе 1.1б.

Техническое задание 1.3. Разработать алгоритм расчета выражения. Вид выражения выбирается из таблицы по номеру своего варианта. Выражение должно рассчитываться для произвольных исходных данных. Результат выражения должен быть выведен на экран дисплея в следующем формате:

Расчет выражения вида R = <*вид выражения*>.
Значение выражения R при
A = <*значение*>
B = <*значение*>
C = <*значение*>
равно <*результат*>
Программа разработана <*Фамилия Имя*>

№	Выражение	№	Выражение
1	$r = a(b - 4c)$	16	$r = a + b(-c)$
2	$r = 3a + bc$	17	$r = (a + b)c$
3	$r = -a + 5bc$	18	$r = 5a + 6bc$
4	$r = 2ab - 3c$	19	$r = 3abc$
5	$r = -6a - 2b + 4c$	20	$r = -2a + bc$
6	$r = 5b - 11c$	21	$r = 3a(b - 2c)$
7	$r = 3a + 4c$	22	$r = 2a + 3b + 4c$
8	$r = 4a - b(-c)$	23	$r = 6(-a)(b + c)$
9	$r = 2a + 5b - 3c$	24	$r = (3a + 2b)c$
10	$r = 7b + 2c$	25	$r = 7a + 5b - 3c$
11	$r = -a - b + 3c$	26	$r = 9a - 5(b + 2c)$
12	$r = 5a - 4b(-c)$	27	$r = 3(a + 2b + c)$
13	$r = 4a + 3b - 2c$	28	$r = -a - b - c$
14	$r = 7a + 2b - c$	29	$r = 4(a - 2b) + c$
15	$r = a - 4b + c$	30	$r = 6a + b - c$

Контрольные вопросы

1. Что такое диалоговый режим работы ЭВМ?
2. Запишите общий вид операции присвоения. Приведите примеры.
3. Какие существуют приемы ввода данных? Чем они различаются?
4. Перечислите основные устройства ввода данных.
5. Изобразите блок-символы различных вариантов ввода данных.
6. Перечислите устройства, на которые возможен вывод данных.
7. Изобразите блок-символы операций вывода данных на различные устройства ЭВМ.
8. В каких случаях используется ручной ввод данных?
9. Что такое оперативный вывод данных?
10. В каких случаях необходимо использовать вывод данных в файл?
11. В каких случаях используется вывод данных на принтер?
12. Что такое дружественный интерфейс?
13. Чем диалоговый режим работы ЭВМ отличается от автоматического?
14. Изобразите общую блок-схему алгоритма, организующего диалоговый режим работы ЭВМ.

Лабораторная работа № 2
Разветвляющиеся алгоритмы и их применение

Цель работы. Изучение простейших приемов использования разветвляющихся алгоритмов при решении задач.

Техническое задание 2.1а. Разработать алгоритм расчета выражения

$$R = \frac{x^2 + x - 1}{x^3 - x^2 + 1}$$

для произвольного значения x. Если выражение можно рассчитать, то есть знаменатель не равен 0, вывести на дисплей результат, если нет — вывести сообщение:

При x = <*введенное значение*> R не определено

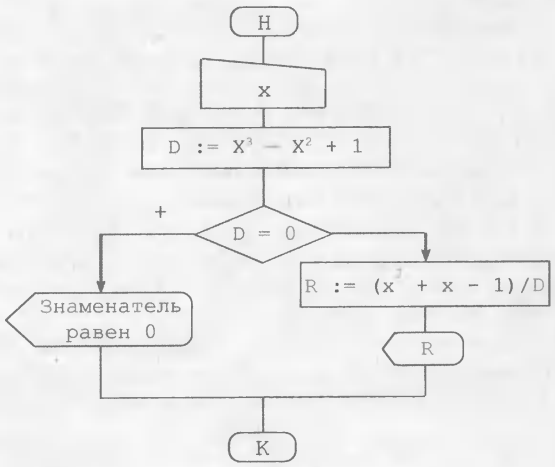
Алгоритм 2.1а. Приведенная задача относится к классу наиболее распространенных — к задаче выбора — и в более общем виде может быть сформулирована так: "В зависимости от некоторого условия надо делать разные действия". В нашем случае надо по-разному реагировать в зависимости от того, принадлежит ли данный аргумент области определения функции или нет.

Такие задачи решаются только с помощью разветвляющихся алгоритмов. Алгоритм, который мы будем строить сейчас, состоит из трех процедур: ввод исходных данных; преобразование данных нужным образом и вывод результата.

Так как в ТЗ сказано, что значение переменной x произвольное, то вводить ее нужно в ручном режиме, т.е. с клавиатуры.

Рассматриваемая функция — дробная. У всякой дроби есть знаменатель. Здесь он довольно сложный, и поскольку значение x произвольное, то невозможно заранее сказать, чему будет равен этот знаменатель. Если же вдруг он окажется равен нулю, то возникнет ошибка "*Division by zero*", которая вызовет *аварийный останов программы*. Порядочная программа так работать не должна. Поэтому, чтобы не утруждать себя решением уравнения третьей степени, просто проверим, чему равен знаменатель, и, если он вдруг окажется нулем, сообщим об этом.

Для этого у нас есть такое мощное средство, как разветвляющаяся структура алгоритма. Как уже было сказано, основой разветвления в алгоритме служит условие, то есть логическое высказывание. В нашем случае нужно проверить, является ли истинным утверждение $x^3 - x^2 + 1 = 0$ при конкретном значении x. И если оно истинно, то вывести предупреждающее сообщение, в противном случае можно спокойно считать дальше и выводить результат. Разумеется, проверку истинности утверждения осуществляет сам компьютер. В блок-схеме значение знаменателя присваивается переменной D, которая в дальнейшем и проверяется на равенство нулю. Это удобно для случаев многократной проверки условий.



Программа 2.1а. Для реализации программы используем условный оператор IF – END IF. Сама же программа может быть оформлена, например, следующим образом:

```
CLS
CONST Message$ = " выражение R не определено."
INPUT "Введите x ", x

D = x ^ 3 - x ^ 2 + 1
IF D = 0 THEN
PRINT "При введенном значении x = ";
      x; Message$
ELSE
R = (x ^ 2 + x - 1) / D
PRINT "При x = "; x; " R ="; R
END IF
END
```

Слово "например" написано не случайно, так как конкретный вид программы во многом зависит от того, насколько хорошо программист (то есть вы!) владеет средствами языка. В данном случае, как уже сказано, используется следующая структура условного оператора:

```
IF <условие> THEN
<"положительная" ветвь алгоритма>
ELSE
<"отрицательная" ветвь алгоритма>
END IF
```

— что в переводе на русский язык обозначает:

```
ЕСЛИ <условие выполняется> ТО
<идем по "положительной" ветви алгоритма>
ИНАЧЕ
<идем по "отрицательной" ветви алгоритма>
КОНЕЦ ЕСЛИ
```

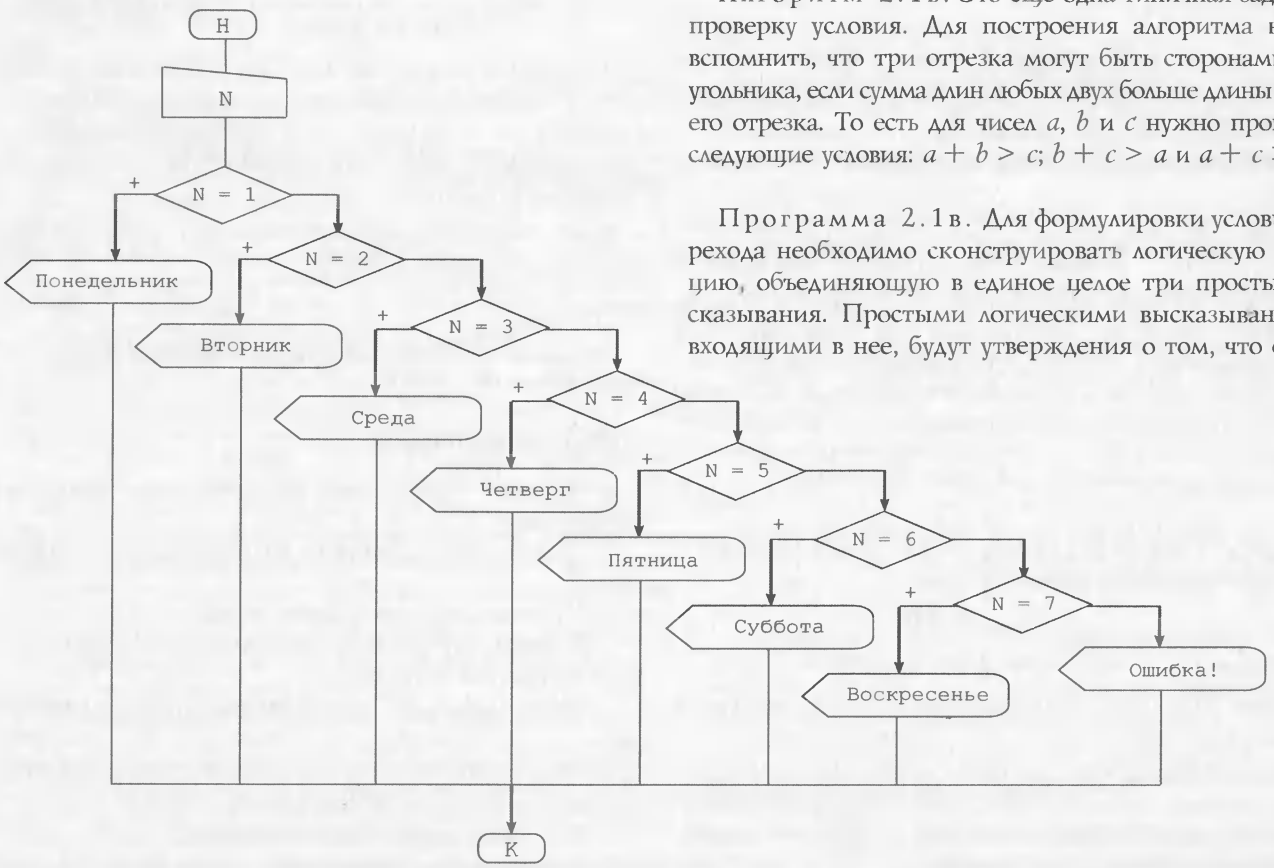
Видно, что ветви алгоритма заключены в так называемые *операторные скобки* IF – END IF. При этом каждая ветвь может содержать любое количество действий (в том числе и условные операторы). Будьте внимательны и не путайте ветви алгоритма: после слова THEN — "положительная" ветвь; после слова ELSE — "отрицательная".

Техническое задание 2.16. Разработать алгоритм, который бы по введенному числу из диапазона [1...7] выводил на дисплей название соответствующего дня недели.

Алгоритм 2.16. Предлагаемая задача относится к категории так называемого множественного выбора и, так же как и предыдущая, является достаточно типичной. В более общем виде задача может звучать так: "Имеется некий признак чего-то. Определить, к какому классу это что-то относится". В данном случае признаком будет введенный номер, а классом — дни недели. Ниже приведена блок-схема алгоритма решения задачи. Несмотря на внешнюю громоздкость, по своей сути он ничем не отличается от алгоритма 2.1а — просто больше условий. Обратите внимание, что если введенное пользователем число не попадает в интервал от 1 до 7, то на экран выводится сообщение об ошибке; это — правило хорошего тона: сообщить пользователю о его вольной или невольной ошибке.

Программа 2.16. В языке Бейсик есть очень удобный оператор, позволяющий реализовать множественный выбор. Оператор этот составной, то есть, так же как и IF – END IF, образован операторными скобками SELECT CASE – END SELECT. Формат оператора приведен ниже:

```
SELECT CASE <идентификатор>
CASE <значение 1>
   <блок операторов>
CASE <значение 2>
    <блок операторов>
  ...
[CASE ELSE
   <блок операторов>]
END SELECT
```



В операторе проверяется значение переменной на соответствие поставленному условию. Здесь <идентификатор> — это имя той самой переменной, значение которой проверяется в CASE. Работа оператора достаточно прозрачна и показана в программе:

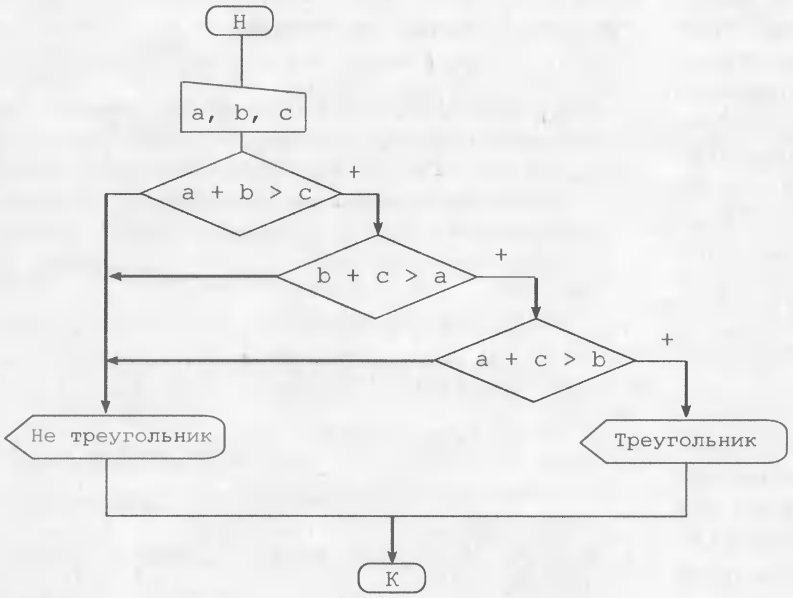
```
CLS
DIM NumberOfDay AS INTEGER
INPUT "Введите номер дня недели ", NumberOfDay
SELECT CASE NumberOfDay
  CASE 1
    PRINT "Это Понедельник"
  CASE 2
    PRINT "Это Вторник"
  CASE 3
    PRINT "Это Среда"
  CASE 4
    PRINT "Это Четверг"
  CASE 5
    PRINT "Это Пятница"
  CASE 6
    PRINT "Это Суббота"
  CASE 7
    PRINT "Это Воскресенье"
  CASE ELSE
    PRINT "Дня с таким номером в неделе не бывает..."
  END SELECT
END
```

Техническое задание 2.1в. Задаются три произвольных числа. Построить алгоритм, который проверяет, могут ли эти числа составить длины сторон треугольника.

Алгоритм 2.1в. Это еще одна типичная задача на проверку условия. Для построения алгоритма нужно вспомнить, что три отрезка могут быть сторонами треугольника, если сумма длин любых двух больше длины третьего отрезка. То есть для чисел a, b и c нужно проверять следующие условия: $a + b > c$; $b + c > a$ и $a + c > b$.

Программа 2.1в. Для формулировки условия перехода необходимо сконструировать логическую функцию, объединяющую в единое целое три простых высказывания. Простыми логическими высказываниями, входящими в нее, будут утверждения о том, что сумма

любых двух сторон больше третьей. При этом для построения треугольника необходимо, чтобы все три высказывания были истинными одновременно. Этого можно достичь, объединив их союзом "и". В программе это делается следующим образом:




```
CLS
  INPUT "Введите значения сторон ", a, b, c
   IF a + b > c AND b + c > a AND  a + c > b THEN
      PRINT "Треугольник"
   ELSE
      PRINT "Не треугольник"
   END IF
END
```

Техническое задание 2.2а. Разработать алгоритм и программу, определяющие четность произвольно введенного целого числа.

Рекомендации. Алгоритм будет содержать всего три операции.

- Ввод произвольного числа. Напоминаем: ключевое слово *произвольный* означает ввод данных в ручном режиме, то есть с клавиатуры.
- Анализ четности введенного числа. Для этого есть много способов, один из них заключается в том, чтобы определить знак выражения $(-1)^n$, где n — введенное число. Если оно четное, то выражение будет положительным, а если число нечетно, то и выражение будет меньше нуля. Получив значение выражения, остается только сравнить его с нулем.
- Вывод сообщения "Четное!" или "Нечетное...".

Техническое задание 2.2б. Разработать алгоритм вычисления выражения вида:

$$y = \begin{cases} \sin x, x \geq 0 \\ \cos x, x < 0 \end{cases}$$

для произвольно введенного значения x. Результат вывести на дисплей.

Рекомендации. Это тоже очень распространенная задача. Часто возникает ситуация, когда в зависимости от значения исходных данных нужно рассчитывать то или иное выражение. Алгоритм решения подобен алгоритму 2.1б, но значительно проще, так как здесь проверяется всего одно условие: больше или меньше нуля введенное в переменную x значение?

Еще один важный момент в этой задаче — вычисление *тригонометрических функций*. Во всех языках программирования аргументы тригонометрических функций выражаются в радианной мере, то есть в долях числа π. А человеку понятнее углы в градусах. Что легче ввести: 45° или 0,7853981... rad? Так что вводить нужно градусы, а потом переводить их в радианы по формуле $a = \alpha \cdot \pi/180$ (a — угол в радианах; α — в градусах). Константу $\pi = 3{,}14159$ лучше ввести заранее.

Техническое задание 2.2в. С клавиатуры вводится один из символов: "П", "С" или "Б". В зависимости от введенного символа алгоритм должен вывести на дисплей числа: 15, 10 и 20 соответственно. Если введен любой другой символ — выдать сообщение об ошибке.

Рекомендации. Стандартная задача на множественный выбор. Повторяет алгоритм 2.1б с точностью до наоборот (в том смысле, что там вводятся числа и выводятся символы, а здесь вводятся символы и выводятся числа). Не забудьте, что ввод производится в символьную переменную, то есть в программе ее нужно либо объявить символьной, либо снабдить символом "$". А введенные значения должны сравниваться с символьными же константами — стало быть, их нужно или взять в кавычки, или присвоить символьным переменным.

Не забудьте аналогично алгоритму 2.1б сообщить об ошибке, если введен какой-либо отличный от заданных символ.

Техническое задание 2.3. Разработать алгоритм и программу, вычисляющие значение выражения (вариант см. в таблице), для произвольно введенных величин. Вывод результата должен быть произведен на дисплей в формате:

При a = *<введенное значение>* значение выражения равно *<результат>*

или

При a = *<введенное значение>* выражение рассчитано быть не может.

Контрольные вопросы

1. В каких задачах используются разветвляющиеся алгоритмы?
2. Чем линейный алгоритм отличается от разветвленного?
3. Что такое условие разветвления?
4. В каком случае ветвь разветвляющегося алгоритма называется положительной?
5. Изобразите блок-символ операции условного перехода.
6. Изобразите полную блок-схему разветвляющегося алгоритма. Укажите ее основные элементы.
7. Что такое множественный выбор?
8. В каких задачах используется множественный выбор?

№	Выражение	№	Выражение	№	Выражение
1	$r = \frac{2a+10}{2a-10}$	11	$r = \sqrt{9-a^2}$	21	$r = \ln(4a-1)$
2	$r = \sqrt{2a-10}$	12	$r = \ln(9-a^2)$	22	$r = \frac{2a+1}{2a}$
3	$r = \ln(2a-10)$	13	$r = \frac{a}{2a+1}$	23	$r = \sqrt{2a}$
4	$r = \frac{1}{1-a^2}$	14	$r = \sqrt{2a+1}$	24	$r = \ln(2a)$
5	$r = \sqrt{1-a^2}$	15	$r = \ln(2a+1)$	25	$r = \frac{1}{1-5a}$
6	$r = \ln(1-a^2)$	16	$r = \frac{25+a^2}{25-a^2}$	26	$r = \sqrt{1-5a}$
7	$r = \frac{5a}{5-a}$	17	$r = \sqrt{25-a^2}$	27	$r = \ln(1-5a)$
8	$r = \sqrt{5-a}$	18	$r = \ln(25-a^2)$	28	$r = \frac{4a}{1-4a}$
9	$r = \ln(5-a)$	19	$r = \frac{4a-1}{4a+1}$	29	$r = \sqrt{1-4a}$
10	$r = \frac{9-a^2}{a}$	20	$r = \sqrt{4a-1}$	30	$r = \ln(1-4a)$

Продолжение следует

Калейдоскоп

"В Космос будут ездить на лифте...

Через пятьдесят лет после того, как над этой идеей перестанут смеяться" — так заявляет известный писатель Артур Кларк.

Еще в 1979 году Кларк описал в одном из своих произведений «космический лифт» — подъемник, перевозящий людей с поверхности Земли на орбитальные станции. Тогда это была всего лишь фантастическая идея, но сейчас, ученые NASA решили вернуться к ней. Они утверждают, что в ближайшие пятьдесят лет земные технологии сделают такую постройку возможной. Путешествие займет около 24 часов, билет будет стоить пять долларов.

Проект выглядит следующим образом: на Земле строится башня высотой около 30 километров, из нее выходит кабель (а фактически труба) длиной около 35 000 километров, именно на такой высоте находится так называемая геостационарная орбита. Чтобы удержать кабель от падения на Землю, на его «свободном конце» придется расположить массивное тело. Сила тяготения, тянущая кабель вниз, и сила, с которой тело будет пытаться оторваться и улететь в Космос, должны взаимно компенсировать друг друга. Таким образом, кабель будет находиться в постоянно натянутом состоянии. На кабеле предполагается разместить от 4 до 6 специальных дорожек, по которым на огромной скорости будут скользить вагоны на электромагнитной тяге. Скорость движения будет достигать нескольких тысяч километров в час.

Правда, пока нет материалов, из которых можно построить подобное сооружение. Углеродные нановолокна, прочность которых в 200 раз превышает прочность стали, пока очень дороги. Основной расчет делается на то, что в ближайшие лет 10—20 технология получения таких волокон значительно удешевится.

По материалам журнала "NewScientist"

Гениальная статистика

Американский ученый Майкл Харт издал книгу «Сто самых важных личностей в истории, расставленных по порядку». Речь идет, как можно догадаться, о самых умных, талантливых, добрых или злых и о самых влиятельных, по мнению автора, людях.

Из ста гениев больше всего ученых и изобретателей — 37. Причем многие из них в первой двадцатке: Гутенберг, Эйнштейн, Пастер, Галилей, Дарвин.

Следующая категория — политики и военачальники в количестве 30 человек.

Затем — философия, литература, искусство и музыка.

Географическое распределение гениев неравномерно. Африка дала лишь 3, Латинская Америка — 1 (Боливара), Азия — 18, Соединенные Штаты — 7. Остальные — 71 человек — уроженцы Европы.

Самый большой вклад в историю сделали британцы, их в списке 18, в частности, 5 — шотландцы.

Вслед за Великобританией идут Германия с Австралией (по 15). Затем 10 французов, 8 римлян и итальянцев, 4 испанца и 3 наших соотечественника: Ленин, Сталин и Петр Великий.

Как объясняет автор, Ленин попал в список не за теорию, а за практику, не за стратегию, а за тактику. Петр, напротив, попал в список не столько за практические реформы, сколько за теоретические прозрения.

Автор также попытался проанализировать семейное положение гениев, несчастные браки и даже рост. Но эти выводы ни к чему не привели.

По материалам газеты "Гудок"

"Калейдоскоп" подготовлен К.Кузнецовым

Решение задач на обработку текста в электронных таблицах

Д.М. Златопольский,
Москва

При изучении электронных таблиц целесообразно наряду с расчетными задачами рассматривать также задачи, связанные с обработкой текста. Это расширит представление учащихся об области возможного применения ЭТ. В статье приведен ряд таких задач.

Прежде чем предъявлять задачи учащимся, следует ознакомить их с основными функциями для работы с текстом, имеющимися в программе Microsoft Excel*:

1. Функция СЦЕПИТЬ

Синтаксис: СЦЕПИТЬ (*текст*1; *текст*2; ...).

В качестве аргументов могут быть указаны текстовые строки, числа или ссылки (адреса), которые ссылаются на одну ячейку. Количество аргументов может достигать 30.

Например, для того чтобы в ячейке B2 получить текст "1 сентября", необходимо записать в ней следующую формулу: =СЦЕПИТЬ(1; " "; B1).

	A	B
1	Месяц	сентября
2	Дата	1 сентября

Вместо функции СЦЕПИТЬ для объединения текстов можно использовать также оператор "&" (без кавычек): = 1 & " " & B2.

2. Функция ДЛСТР

Эта функция возвращает количество символов в текстовой строке.

Синтаксис: ДЛСТР(*текст*).

Аргумент *текст* — это текст, длина которого определяется. Пробелы учитываются.

Примеры

ДЛСТР("Урок информатики") равняется 16;
ДЛСТР("") равняется 0.

3. Функция ПСТР

Возвращает заданное число символов из строки текста, начиная с указанной позиции.

Синтаксис: ПСТР(*текст*; *начальная позиция*; *количество символов*),

— где *текст* — текстовая строка, содержащая извлекаемые символы;

начальная позиция — позиция первого символа, извлекаемого из текста. Первый символ в тексте имеет начальную позицию 1 и т.д.;

количество символов указывает, сколько символов нужно вернуть.

Если *начальная позиция* больше, чем длина текста, то функция ПСТР возвращает строку "" (пустой текст).

Если *начальная позиция* меньше, чем длина текста, но *начальная позиция* плюс *количество символов* превышают длину текста, то функция ПСТР возвращает символы вплоть до конца текста.

Если *начальная позиция* меньше 1 или *количество символов* отрицательно, то функция ПСТР возвращает значение ошибки #ЗНАЧ!

Примеры

ПСТР("Поток жидкости"; 1; 5) равняется "Поток";
ПСТР("Поток жидкости"; 7; 20) равняется "жидкости";
ПСТР("Байт"; 5; 2) равняется "" (пустой текст)

5. Функция ЛЕВСИМВ

Эта функция возвращает первые (слева) символы текстовой строки.

Синтаксис: ЛЕВСИМВ(*текст*; *количество символов*),

— где *текст* — текстовая строка, которая содержит извлекаемые символы;

количество символов — определяет, сколько символов должна извлечь функция ЛЕВСИМВ.

Количество символов должно быть больше или равно нулю.

Если *количество символов* больше длины текста, то функция ЛЕВСИМВ возвращает весь текст.

Если *количество символов* опущено, то оно полагается равным 1.

Примеры

ЛЕВСИМВ("Цена Товара"; 4) равняется "Цена"

Если ячейка A1 содержит строку "Швеция", то: ЛЕВСИМВ(A1) равняется "Ш"

6. Функция ПРАВСИМВ

Возвращает последние (правые) символы текстовой строки.

Синтаксис: ПРАВСИМВ(*текст; количество символов*),

— где текст — текстовая строка, содержащая извлекаемые символы;

количество символов — это количество извлекаемых символов.

Количество символов должно быть больше или равно нулю.

Если *количество символов* опущено, оно полагается равным 1.

* При использовании других программ рассматриваются аналогичные функции, имеющиеся в этих программах.

Если *количество символов* больше, чем длина текста, то функция ПРАВСИМВ возвращает весь текст.

Примеры

ПРАВСИМВ("Продажная Цена"; 4) равняется "Цена"
ПРАВСИМВ("Ассортиментный Номер") равняется "р"

7. Функция НАЙТИ

Функция находит вхождение одной текстовой строки (*искомый текст*) в другую текстовую строку (*просматриваемый текст*) и возвращает положение начала искомого текста относительно крайнего левого символа просматриваемого текста.

Синтаксис: НАЙТИ(*искомый текст*; *просматриваемый текст*; *начальная позиция*),

— где *искомый текст* — искомый текст;

просматриваемый текст — текст, содержащий искомый текст;

начальная позиция — позиция символа, с которой следует начинать поиск. Первый символ в аргументе *просматриваемый текст* имеет номер 1. Если аргумент *начальная позиция* опущен, то он полагается равным 1.

Если *искомый текст* — это "" (пустая строка), то функция НАЙТИ считает подходящим первый символ в просматриваемой строке (то есть возвратит значение аргумента *начальная позиция* или 1).

Искомый текст не должен содержать символов шаблона ("?" и "*").

Примеры

НАЙТИ("М";"Мама мыла раму ") равняется 1.
НАЙТИ("м";"Мама мыла раму ") равняется 3.
НАЙТИ("м";"Мама мыла раму "; 5) равняется 6.

Если *искомый текст* не входит в *просматриваемый текст*, то функция НАЙТИ возвращает значение ошибки #ЗНАЧ!. В таких случаях можно предусмотреть вывод в ячейке соответствующего сообщения, использовав функцию ЕСЛИ, в которой в качестве условия применить функцию ЕОШ (последняя возвращает значение Истина, если в данной ячейке результат имеет значение #ЗНАЧ!).

Например, для того чтобы исключить появление значения ошибки #ЗНАЧ! в ячейке B2, когда в ней вычисляется положение буквы "д" в ячейке A2 (см. ниже), следует записать в ячейке B2 следующую формулу:

=ЕСЛИ(ЕОШ(НАЙТИ("д"; A2)); "Нет такой буквы"; НАЙТИ("д"; A2))

	A	B
1	Использование функции	НАЙТИ
2	абвг	

Для поиска вхождений одной текстовой строки в другую текстовую строку можно также применить функцию ПОИСК, но в отличие от функции НАЙТИ функция ПОИСК не учитывает регистр символов и допускает использование символов шаблона "?" и "*".

Задачи

Задача 1. В ячейке B4 получить текст "Файликов Петя":

	A	B
1	Задача 1	
2	Фамилия ученика:	Файликов
3	Имя ученика:	Петя
4	Фамилия и имя ученика:	

Задача 2. В ячейке B5 получить текст, состоящий из фамилии, имени и отчества сотрудника, разделенных пробелом:

	A	B
1	Задача 2	
2	Фамилия сотрудника:	
3	Имя сотрудника:	
4	Отчество сотрудника:	
5	Фамилия, имя, отчество сотрудника:	

Задача 3. В ячейке B3 получить число символов в строке текста, вводимой в ячейку B2:

	A	B
1	Задача 3	
2	Введите строку символов →	
3	Число символов в строке:	

Задача 4. В ячейке B3 получить слово "форма":

	A	B
1	Задача 4	
2	Исходное слово:	Информатика
3	Полученное слово:	

Задача 5. В ячейке B3 получить слово "Комбинат":

	A	B
1	Задача 5	
2	Исходное слово:	Комбинаторика
3	Полученное слово:	

Задача 6. В ячейке B3 получить слово "байт":

	A	B
1	Задача 6	
2	Исходное слово:	Килобайт
3	Полученное слово:	

Задача 7. В ячейке B4 получить слово "Информация", в ячейке B5 — слово "Оператор":

	A	B
1	Задача 7	
2	Первое исходное слово:	Информатор
3	Второе исходное слово:	Операция
4	Первое полученное слово:	
5	Второе полученное слово:	

Задача 8. В ячейке B5 получить слово "Файл":

	A	B
1	Задача 8	
2	Первое исходное слово:	Фирма
3	Второе исходное слово:	Байт
4	Третье исходное слово:	Паскаль
5	Полученное слово:	

Задача 9. В ячейке B5 получить текст, состоящий из фамилии и инициалов в виде "Иванов Н.И.":

	A	B
1	Задача 9	
2	Фамилия сотрудника:	
3	Имя сотрудника:	
4	Отчество сотрудника:	
5	Фамилия и инициалы сотрудника:	

Задача 10. В ячейках B2 и B3 вводятся слова, состоящие из четного числа букв. Получить в ячейке B4 слово, состоящее из первой половины первого слова и второй половины второго слова:

	A	B
1	Задача 10	
2	Введите первое слово:	
3	Введите второе слово:	
4	Полученное слово:	

Задача 11. В ячейку B2 вводится слово из 9 букв. Поменять местами его трети следующим образом:

а) первую треть слова разместить на месте третьей, вторую треть — на месте первой, третью треть — на месте второй;

б) первую треть слова разместить на месте второй, вторую треть — на месте третьей, третью треть — на месте первой.

Полученные слова получить в ячейках B3 и B4:

	A	B
1	Задача 11	
2	Исходное слово:	
3	Первое полученное слово:	
4	Второе полученное слово:	

Задача 12. В ячейку B2 вводится слово, количество букв в котором кратно трем. Получить в ячейках B3 и B4 слова по правилам, описанным в условии задачи 11.

Задача 13. Текст в ячейке B3 получить по формуле, которую затем распространить (скопировать) на ячейки B4–B10:

	A	B
1		Задача 13
2	Буква	Слово
3	с	соль
4	м	моль
5	н	ноль
6	г	голь
7	р	роль
8	т	толь
9	П	Поль
10	б	боль

Задача 14. Текст в ячейке B4 получить по формуле, которую затем распространить (скопировать) на ячейки B5–B12:

	A	B
1		Задача 14
2	Буква	Слово
3	а	а
4	б	аб
5	в	абв
...	...	...
12	и	абвгдеёжзи

Задача 15. В ячейку B2 вводится слово из восьми букв. В ячейках B4–B11 получить буквы этого слова:

	A	B
1	Задача 15	
2	Исходное слово:	Алгоритм
3	Номер буквы	Буква
4	1	А
5	2	л
6	3	г
...	...	...
11	8	м

Задача 16. Дано слово из пяти букв. Проверить, является ли оно перевертышем ("перевертышем" называется слово, читаемое одинаково как с начала, так и с конца).

Задача 17. Дана строка текста из 20 символов. Определить, сколько раз в ней встречается буква "о".

Задача 18. *Текст* в ячейке A4 получить по формуле, которую затем распространить (скопировать) на ячейки A5–A12:

	A	B
1	Задача 18	
2	№ пп	
3	1.	
4	2.	
5	3.	
...	...	
12	10.	

Задача 19. *Текст* в ячейке A13 получить по формуле, которую затем распространить (скопировать) на ячейки A14–A22:

	A	B
1	Задача 19	
2	№ пп	
3	1.	
4	2.	
5	...	
...	...	
12	10.	
13	11.	
...	...	
22	20.	

Задача 20. В ячейке B2 записано некоторое слово, в котором имеются буквы "а". Найти номер позиции, на которой находится первая буква "а" в слове.

Решить также задачу при условии, что букв "а" в исходном слове может не быть.

Задача 21. В ячейке B2 записаны два слова (начальных пробелов нет). Получить первое слово.

Решение оформить следующим образом

	A	B
1	Задача 21	
2	Заданный текст:	
3	Номер позиции первого пробела:	
4	Первое слово в тексте:	

Задача 22. В ячейке B2 записаны два слова, разделенные одним пробелом (начальных пробелов нет). Получить первое и второе слова.

Решение оформить следующим образом:

	A	B
1	Задача 22	
2	Заданный текст:	
3	Общее число символов в тексте:	
4	Номер позиции пробела:	
5	Первое слово в тексте:	
6	Второе слово в тексте:	

Задача 23. В ячейке записаны фамилия, имя и отчество, разделенные одним пробелом (начальных пробелов нет). Получить фамилию и инициалы этого человека в виде "Иванов Н.В.".

Задача 24. В ячейке записаны фамилия, имя и отчество человека, разделенные одним пробелом (начальных пробелов нет). Получить отдельно фамилию, имя и отчество.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РФ

НАЦИОНАЛЬНЫЙ ФОНД ПОДГОТОВКИ КАДРОВ

ПРОЕКТ

Требования к уровню подготовки выпускников по информатике Обязательный минимум содержания образования по информатике

Департамент образовательных программ и стандартов общего образования Минобразования России представляет проекты **"Обязательного минимума содержания образовательных программ начального, основного и среднего (полного) общего образования"** и **"Требований к уровню подготовки выпускников"**, разработанные для проведения эксперимента по совершенствованию структуры и содержания общего образования, осуществляемого на основании Федеральной программы развития образования, Плана действий Правительства Российской Федерации в области социальной политики и модернизации экономики на 2000—2001 годы.

Обязательный минимум содержания образования задает перечень дидактических единиц содержания образования, которые подлежат обязательному изучению при получении начального, основного и среднего (полного) общего образования.

Обязательный минимум содержания образовательных программ полного среднего образования представлен на двух уровнях — образовательном и профильном. Предполагается, что на общеобразовательном уровне учебные предметы образовательной области будут изучать все учащиеся, не выбравшие конкретный профиль. На профильном уровне учащиеся в соответствии со своими интересами углубленно изучают учебные предметы, ориентирующие их на дальнейшее профессиональное образование.

Требования к уровню подготовки выпускников устанавливают уровень, реально достижимый в практике массового обучения и обеспечивающий права и возможности обучающихся на получение качественного общего образования.

Данный проект публикуется для организации общественного обсуждения, к которому приглашаются педагоги, ученые, родители.

Замечания и предложения позволят содержательно откорректировать предлагаемый проект.

В ходе обсуждения важно получить ответы на следующие вопросы:

1. Обеспечивает ли проект фундаментальный характер и практико-ориентированную направленность образования?
2. Отвечает ли данное содержание общего образования современным потребностям личности, общества и государства? В какой степени освоение представленного содержания общего образования необходимо для каждого выпускника начальной, основной и полной средней школы?
3. Достаточна ли разгрузка содержания образования? Оправданно ли исключение некоторых вопросов содержания и переноса определенной части содержания в профильное обучение?
4. Как вы предполагаете, реально ли выполнение указанных требований учащимися? Какой объем учебного времени необходим для обеспечения отработки учебных и практических умений, освоения опыта эмоциональной и творческой деятельности?

Просьба замечания и предложения направлять по адресу:

101856, Москва, Чистопрудный бульвар, 6. Департамент образовательных программ и стандартов общего образования Минобразования России.

Департамент образовательных программ
и стандартов общего образования

Предисловие

В настоящем издании представлены проекты "Требований к уровню подготовки выпускников" и "Обязательного минимума содержания образовательных программ" начального, основного и полного среднего образования, разработанные в соответствии с Планом мероприятий Правительства РФ по модернизации образования. Эти документы являются исходной основой создания государственных образовательных стандартов общего среднего образования, которое намечено завершить в этом году.

Традиция задания содержания обучения через перечень дидактических единиц восходит к хорошо известным предметным программам. Достоинством такого подхода является сравнительно прозрачная структура содержания обучения, понятная непрофессионалу. Его недостатком является принципиально недеятельностный характер, затрудняющий четкую характеристику объема усвоенного содержания и операциональное задание планируемых результатов обучения. Иными словами, "Обязательный минимум" представляет собой своеобразный каркас (framework) содержания обучения, выстраиваемый с помощью деятельностных характеристик "Требований".

С другой стороны, задание содержания обучения исключительно на деятельностном языке, позволяющее профессионалу эффективно планировать и контролировать результаты обучения, не отличается наглядностью и понятностью для широкой публики, которая имеет право и желает получить ответ на вопрос: чему учит школа? Таким образом, два взаимосвязанных документа государственного образовательного стандарта — "Обязательный минимум содержания образовательных программ" и "Требования к уровню подготовки выпускников" — взаимно дополняют друг друга.

"Требования к качеству подготовки выпускников" являются лидирующим документом проекта. Они устанавливают уровень подготовки выпускников, реально достижимый в практике массового обучения и обеспечивающий права и возможности обучающихся на получение полноценного и качественного общего образования.

"Обязательный минимум содержания образования" задает перечень дидактических единиц содержания образования, которые подлежат обязательному изучению в начальной, основной и полной средней школе.

Представленная совокупность документов определяет уровень и объем содержания образования, которые **государство обязуется гарантированно предоставить обучающимся во исполнение их конституционного**

права на образование. Иными словами, "Требования" и "Обязательный минимум" описывают, образование какого уровня и объема должно быть предоставлено школьникам **бесплатно**.

При отборе содержания образования всегда противоборствуют две тенденции — консервативная и радикальная. С одной стороны, соображения преемственности и требования разумного консерватизма вынуждают нас корректировать содержание образования только по самым серьезным основаниям. Поэтому любая модернизация содержания школьного образования отправляется от существующего содержания и сохраняет его традиционное ядро. Подобный подход, сберегающий лучшие традиции отечественного образования и отвечающий категорическому императиву "не навреди!", принят и в настоящей работе.

Так, в проекте сохранена традиционная для российской школы **ориентация на фундаментальный характер образования** — будь то высокие духовные стандарты гуманитарного образования или основополагающие конструкции научной картины мира в естественно-научном образовании. Более того, в проекте заложены предпосылки существенного углубления подобной ориентации за счет явно выраженной тенденции гуманитаризации содержания образования.

Однако происходящие сегодня существенные изменения в общественных отношениях, средствах коммуникации и производства требуют адекватного отражения в содержании общего образования. На наших глазах происходит невиданное ускорение информационного взаимодействия людей, сближающее народы и континенты и обеспечивающее процессы глобализации. Для школы это означает необходимость значительного **усиления коммуникативных компетенций**, будь то традиционные аспекты языковых коммуникаций или более современные направления использования информационных технологий.

С этой целью в проекте существенно **усилена филологическая подготовка школьников**. В первую очередь это относится к освоению русского языка — и как родного, и как языка межнационального общения народов России — за счет изучения его на всех ступенях общего образования. Принципиальные изменения предложены по отношению к освоению иностранных языков: проект предусматривает изучение по крайней мере одного иностранного языка на всех ступенях школы.

Проект закладывает серьезные предпосылки **информатизации школы** посредством современной трактовки содержания обучения информационным технологиям, предусматривающей общий подход к изучению информационных процессов, реальное освоение доступной компьютерной и аудиовизуальной техники, их применение в процессе изучения школьных дисциплин. В этом смысле проект учитывает как общую тенденцию компьютеризации образования, так и конкретные планы и программы информатизации российской школы.

Наконец, усиление коммуникативных компетенций требует адекватной ориентации содержания гуманитарного образования: необходимо формировать **социальные умения общения**, связанные с осознанием многообразия позиций, готовностью выслушать и понять другую точку зрения и терпимостью, критическим анализом аргументов и т.п. Соответствующие аспекты заложены в содержание обучения гуманитарным дисциплинам, и в первую очередь обществоведению.

Сущностные изменения в социально-политическом и экономическом устройстве страны, произошедшие в России за последние десять лет, потребовали серьезной коррекции содержания гуманитарных дисциплин. Проект закрепляет такие общепринятые сегодня положения, как отказ от навязывания идеологических доктрин при рассмотрении гуманитарных проблем и анализе социально-экономических явлений и процессов. Это достигается, в частности, посредством снятия идеологических оценочных клише, представлением различных, зачастую противоборствующих, позиций. Тем самым для учащихся создаются возможности самостоятельного суждения и осознанного выбора точки зрения.

В блоке социально-экономических дисциплин значительно усилены вопросы **правового и экономического образования**. В частности, на старшей ступени предусмотрено систематическое изучение вопросов экономики. Экономический аспект подчеркнут и в содержании технологического образования.

Необходимость коррекции содержания образования вызывается не только внешними причинами. В не меньшей степени она продиктована соображениями, связанными с внутренними проблемами самой системы общего образования.

Так, отказ от осуществления всеобщего среднего образования порождает проблему, вытекающую из линейной конструкции многих курсов, которые начинаются на основной и продолжаются на старшей ступени школы. В результате те учащиеся, которые завершают общее образование в основной школе, не получают целостного образования. Поэтому переход к концентрической системе изложения, реализованный в проекте, является вынужденной мерой, обеспечивающей **полноценный характер базового образования**.

Значительно более серьезной проблемой школы является перегрузка содержания образования. Эта проблема обязана своим происхождением предшествующей реформе 60-х гг., проводившейся под лозунгом "приближения содержания школьного образования к уровню современной науки". К сожалению, реализация реформы проходила со значительными издержками, связанными с перенасыщением школьных предметов чрезмерным объемом научного содержания, недоступного для большинства детей.

Педагоги старшего поколения помнят бурную реакцию общества конца 70-х гг. на неутешительные итоги этой реформы в области математики. Отделение математики АН СССР тогда отвергло демагогические доводы, апеллировавшие к необходимости "повышения научного уровня школьной математики", и заняло принципиальную позицию в интересах детей, что позволило существенно разгрузить школьные программы по математике. Этот пример, однако, не оказался заразительным. Не случайно поэтому важнейшим лозунгом реформы 1984 г. стало требование "разгрузки содержания образования от чрезмерно усложненного и второстепенного материала". Однако по многим причинам эта реформа оказалась в значительной степени нереализованной.

Избыточный объем школьных программ далеко не пустяк. Поговорите с учителями любого предмета, и они вам скажут: чтобы выполнить программу, фактически на каждом уроке надо вводить новые понятия, объяснять новые факты, доказывать новые теоремы, запоминать новые правила и т.д. Детям некогда обдумать новый материал, некогда потренироваться в его практическом усвоении, некогда рефлексировать и переживать, некогда решать интересные творческие задачи. Школа превратилась в конвейер знаний, который движется с чрезмерно большой и непосильной для большинства учащихся скоростью, что приводит к стрессовым перегрузкам и возникновению негативных психологических комплексов у добросовестных школьников. Подавляющее же большинство школьников вынуждено игнорировать заметную часть учебного материала, что порождает лавину лжи и обмана как со стороны учащихся, так и со стороны учителей.

Между тем в отечественной педагогике в работах В.В. Краевского, И.Я. Лернера и М.Н. Скаткина давно сформирована прогрессивная концепция содержания образования, утверждающая необходимость освоения школьниками, помимо знаний, умений и ценностных ориентаций, еще и опыта эмоциональной и творческой деятельности. Но, чтобы реализовать эти идеи, необходимо преодолеть односторонний "знаниевый" подход и потеснить знания в пользу других компонентов содержания образования.

В пользу этого говорят и традиции отечественной школы, девизом которой вплоть до последнего сорокалетия было правило "лучше меньше, да лучше". Об этом свидетельствует опыт зарубежной школы, предоставляющей учащимся основной (непрофилированной) школы значительно больше "воздуха", чем обеспечивается развитие детей, а не формальное усвоение большого объема знаний.

Поэтому разгрузка содержания образования необходима прежде всего для **восстановления педагогически и психологически обоснованной структуры содержания образования**. Надо ли объяснять, что только реализация этого подхода способна обеспечить такие направления образования, как нравственное воспитание школьников, реализация развивающего обучения, прикладная и практическая ориентация обучения, осознанное и качественное усвоение знаний, формирование интересов и предпочтений!

С этой целью в проекте осуществлена достаточно серьезная **разгрузка обязательного содержания** путем исключения ряда вопросов, не имеющих общеобразовательного значения. Тем самым создаются условия для повышения качества образования за счет высвобождения учебного времени для отработки учебных и практических умений, освоения опыта эмоциональной и творческой деятельности.

Другое направление разгрузки относится к тому материалу, который, не обладая общеобразовательной ценностью, оказывается полезным или даже необходимым для продолжения образования по данному предмету в высшей школе. Как правило, именно этот "технический" материал и является недоступным для тех детей, сфера интересов которых находится за пределами данного предмета.

С целью решения возникающей таким образом проблемы в проекте реализована **двухуровневая структура содержания образования на старшей ступени** школы, обеспечивающая возможность изучения интересующих старшеклассника предметов на углубленном — профильном — уровне. Именно в профилированную старшую ступень проект переносит "техническое" содержание предметов. Подобная дифференциация обучения на старшей ступени, с одной стороны, освобождает учащихся от изучения ненужных для них специальных вопросов. С другой стороны, она же создает возможности и условия для существенного повышения качества образования по профилю по сравнению с сегодняшней школой, на необходимости чего настаивают представители вузов.

Наконец, проект предусматривает некоторую **дифференциацию** содержания образования с точки зрения требований к его усвоению посредством явного выделения учебного материала, подлежащего обязательному изучению, но не обязательному усвоению. Думается, однако, что в указанном направлении **деятельностной разгрузки** сделан лишь первый, далеко не достаточный шаг. Однако подлинное обеспечение такой разгрузки возможно лишь на основе четко определенных операциональных **критериев успешности усвоения** содержания образования, прямо выходящих на процедуры оценки и аттестации и разработки механизма трансляции этих критериев школьникам. Проект предусматривает создание таких критериев и осуществление на этой основе соответствующей коррекции "Требований", и особенно "Обязательного минимума", что должно стать ближайшим направлением работы авторского коллектива в этом году.

Авторы рассматривают настоящий проект как предварительный. Более того, методология его разработки была сознательно ориентирована на применение итеративной процедуры — своеобразного метода последовательных приближений, в ходе которой проект будет неоднократно уточняться и совершенствоваться.

Важнейшим этапом этой процедуры должно стать общественное обсуждение предлагаемого содержания образования в российской школе. Широкая публикация проекта преследует цель инициировать такое обсуждение. При этом недостаточно ограничиться дискуссиями исключительно внутри профессионального сообщества, ибо содержание проекта выходит за границы собственно педагогических задач. "Рамочные" ограничения образовательных стандартов призваны стать полем просвещенного консенсуса, достигнутого не только представителями разных областей образования, науки и культуры, но и общества в целом.

Общественное обсуждение проекта будет организовано в федеральных округах и регионах в ближайшие несколько месяцев. Мы рассчитываем на помощь и участие коллег, на конструктивную и содержательную критику, которая призвана стать существенным элементом в реализации итеративного подхода. Мы планируем также получить официальные заключения государственных учреждений, научных организаций и вузов страны. Окончательная коррекция проекта будет осуществлена с учетом хода и итогов обсуждений и результатов разработки критериев успешности усвоения содержания образования.

Просьба замечания и предложения направлять по адресу: 101856, Москва, Чистопрудный бульвар, 6. Департамент образовательных программ и стандартов общего образования Министерства образования РФ; или по электронной почте: **firsov@online.ru**.

В.В. Фирсов

Авторский коллектив

Руководитель проекта: В.В. Фирсов.
Информатика: А.А. Кузнецов (рук.), А.Л. Семенов, А.Ю. Уваров.

НАЧАЛЬНАЯ ШКОЛА

Требования к уровню подготовки выпускников

Изучение информатики должно предоставить учащимся возможность:

— получить представления о возможных источниках информации, способах ее поиска; приобрести опыт получения информации из различных источников;

— анализировать и оценивать полученную информацию, используя различные схемы;

— переводить информацию в личные знания для использования в своей деятельности; использовать информацию для принятия самостоятельных решений;

— использовать информацию для создания информационных моделей (схема, таблица и др.) объектов и процессов; приобрести опыт создания простых информационных объектов;

— познакомиться с техническими средствами приема, передачи, хранения и обработки информации: аудио-, видео-, CD-ресурсы, радио, телевидение, телефон (при наличии оборудования);

— получить общее представление о компьютере, правилах поведения в компьютерном классе и правилах работы на компьютере (при наличии оборудования);

— осуществлять компьютерный поиск ключевых слов в текстах, поиск информации на компакт-дисках; создавать (конструировать) на компьютере несложные информационные объекты (при наличии оборудования).

Обязательный минимум содержания образования

Виды информации. Знаки. Слова. Предложения. Тексты. Изображения. Иллюстрации. Аудио- и видеозаписи.

Источники информации. Наблюдения. Книги, газеты, журналы, радио, телевидение, аудио- и видеозаписи и др.

Организация информации. Алфавитный порядок. Составные части книги и их назначение. Каталог. Справочники.

Поиск и анализ информации. Сужение и расширение объема искомой информации. План поиска. Оценка информации.

Применение информации. Создание новых информационных объектов. Обмен информацией.

Информационные модели. Описание предметов, действий. Рассуждение. Схемы.

Технические средства приема, передачи, хранения и обработки информации[1]. Аудио- и видеозаписи, CD-ресурсы. Радио, телевидение, телефон. Компьютер и его составные части. Правила поведения в компьютерном классе и правила работы на компьютере.

Компьютерный поиск ключевых слов в текстах. Поиск информации на компакт-дисках. Создание на компьютере информационных объектов (текстов, изображений).

ОСНОВНАЯ ШКОЛА

Требования к уровню подготовки выпускников

Изучение информатики должно предоставить учащимся возможность:

— получить представление об управлении процессами, обратной связи, автоматическом и программном (компьютерном) управлении, о принципах программного управления физическими объектами; строить простейшие программы формальных исполнителей с использованием базовых конструкций: выбора (ветвления), повторения, именования;

— использовать и строить цепочки, деревья и таблицы для описания объектов информатики, классификации информационных объектов и выбора действий, получить представления о построении математических моделей игровой деятельности;

— получить представление о материальных и информационных моделях, их свойствах; знать основные этапы построения моделей, иметь представление о способах оценки моделей, иметь опыт построения компьютерной визуализации моделей объектов и процессов;

— освоить стандартные массовые средства работы с информационными объектами (текст/гипертекст, звук, фотография, рисунок, чертеж, видеозапись, мультипликация, динамические (электронные) таблицы), создавать и редактировать их с помощью стандартных средств информационных и коммуникационных технологий;

— пользоваться на базовом уровне компьютером и типовым периферийным оборудованием (сканер, цифровая камера, принтер, мультимедийный проектор), стандартным компьютерным графическим интерфейсом;

— понимать специфику информационных систем массовой и индивидуальной коммуникации (почта, телефон, Интернет), уметь искать сведения, пользуясь информационными ресурсами библиотек, Интернета;

— получить представления о роли информатики и информационных технологий в развитии современной цивилизации, информационной инфраструктуре общества, юридических, этических и моральных нормах работы с информационными объектами; об информационной безопасности общества и личности, необходимости самоограничения человека, живущего в условиях избытка информации.

Обязательный минимум содержания образования

Теоретическая информатика

Имя и значение. Логические значения. Операции "Все", "Существует", отрицание. Свойства. Классификации.

Цепочки (конечные последовательности) в описании объектов информатики, в том числе естественных и искусственных языков. Таблицы как способ представления информации. Деревья возможностей, классификации, выбора действий, предков, потомков, ссылок, подзадач, формул. Древовидная система хранения информационных объектов (система вложенных "папок" файловой системы). Игры с полной информацией, дерево игры. Поиск стратегии выигрыша.

Действия и команды. Обратная связь. Исполнители, система команд исполнителя. Программа. Прямое и программное управление информационными и физическими объектами. Построение программ для решения задач в графической среде, выбор, повторение, присваивание значения имени в ходе выполнения программы. Имена программ. Разбиение задачи на подзадачи.

Материальные и информационные модели, свойства моделей. Дискретное представление объектов и процессов. Адекватность модели объекту и целям моделирования. Количественная и качественная оценка модели, адекватность объекту и цели моделирования. Компьютерные модели физических процессов, процессов в экономике и экологии, их визуализация.

Информационные технологии и средства информатизации

Процессор, память, внешние устройства. Способы хранения непрерывной и цифровой (дискретной) информации. Хранимая программа, пошаговое выполнение, взаимодействие между устройствами, шина передачи данных. Интерфейс пользователя. Работа на клавиатуре.

Оценка объемов информации (текст, звук, рисунок, видео). Оценка скорости передачи и обработки информации.

Принципы построения информационных систем вещания (телевидение, радио, газета), информационные ресурсы Интернета. Использование средств индивидуальной коммуникации (почта, телефон, электронная почта в Интернете).

Средства и способы работы с информационными объектами (текст/гипертекст, звук, фотография, рисунок, чертеж, видеозапись, мультипликация, динамические/электронные таблицы). Этапы создания информационных объектов. Фиксация объектов и процессов окружающего мира в информационных объектах. Поиск, отбор информационных объектов.

Деятельность с информационными объектами — создание, обработка (редактирование), передача другому (создание и проведение выступлений, размещение информационных объектов в Интернете) — в рамках различных областей образовательного процесса. Использование информационной среды школы, планирование и организация индивидуальной и коллективной деятельности с использованием средств ИКТ.

Социальная информатика

Восприятие и обработка информации человеком, понимание. Человеческая память. Порог восприятия и порог различения. Шкала интенсивности (громкости, яркости).

Информационное общество. Информационные ресурсы общества. Правовые и этические нормы информационной деятельности. Информационная безопасность. Информационная экология.

СРЕДНЯЯ (ПОЛНАЯ) ШКОЛА

Общеобразовательный уровень

Требования к уровню подготовки выпускников

Изучение информатики должно предоставить учащимся возможность:

[1] При наличии соответствующего оборудования.

— развить представления об управлении процессами, обратной связи, автоматическом и программном (компьютерном) управлении, о принципах программного управления физическими объектами; строить несложные программы формальных исполнителей с использованием базовых конструкций: выбора (ветвления), повторения, именования;

— получить представления о материальных и информационных моделях, их свойствах, рассуждать об адекватности модели моделируемому объекту и целям моделирования; освоить основные этапы построения моделей, получить представление о способах оценки моделей, приобрести опыт построения компьютерной визуализации моделей и процессов в области физики, экономики и экологии;

— уверенно владеть стандартными массовыми средствами работы с информационными объектами (текст/гипертекст, звук, фотография, рисунок, чертеж, видеозапись, мультипликация, динамические (электронные) таблицы), создавать и редактировать их с помощью стандартных средств информационных и коммуникационных технологий;

— свободно пользоваться компьютером и типовым периферийным оборудованием (сканер, цифровая камера, принтер, мультимедийный проектор), стандартным компьютерным графическим интерфейсом;

— понимать специфику информационных систем массовой и индивидуальной коммуникации (почта, телефон, Интернет), уметь искать сведения, пользуясь информационными ресурсами библиотек, Интернета, осознанно пользоваться технологиями личной коммуникации (презентации) при подготовке и выполнении выступлений, технологиями размещения информации в Интернете;

— познакомиться с основными шагами процедуры разработки информационных объектов и систем, иметь личный опыт этой работы на примере решения задач развития/поддержания информационной среды образовательного учреждения;

— получить представления о роли информатики и информационных технологий в развитии современной цивилизации, информационной инфраструктуре общества, юридических, этических и моральных нормах работы с информационными объектами; об информационной безопасности общества и личности, необходимости самоограничения человека, живущего в условиях избытка информации.

Обязательный минимум содержания образования

Теоретическая информатика

Управление процессами, обратная связь, автоматизированное управление, использование компьютера. Исполнители. Команды исполнителя. Прямое и программное управление объектами. Понятие алгоритма, его свойства и способы записей. Исполнение алгоритма. Программа. Построение программ для решения задач в графической среде, присваивание значения имени в ходе выполнения программы. Рекурсия.

Цепочки (конечные последовательности) в описании объектов информатики, в том числе естественных и искусственных языков. Деревья классификации, выбора действий, предков, потомков. Древовидная система хранения информационных объектов (система вложенных "папок" файловой системы). Игра с полной информацией, дерево игры. Стратегия выигрыша.

Модели материальные и информационные, свойства моделей. Адекватность модели объекту и целям моделирования. Основные этапы построения моделей, формализация. Количественная и качественная оценки модели, адекватность объекту и цели моделирования. Компьютерные модели процессов в экономике и экологии, их визуализация. Компьютерные модели физических процессов.

Информационные технологии и средства информатизации

Основные принципы работы компьютера. Процессор, память, внешние устройства. Формирование изображения на мониторе, в проекторе и принтере. Способы хранения информации на машинных носителях. Интерфейс пользователя. Оценка объемов информации (текст, звук, рисунок, видео). Скорость передачи и обработки информации.

Принципы построения информационных систем вещания (телевидение, радио, газета) и индивидуальной коммуникации (почта, телефон, Интернет). Информационные ресурсы Интернета.

Средства и способы работы с информационными объектами (текст/гипертекст, звук, фотография, рисунок, чертеж, видеозапись, мультипликация, динамические/электронные таблицы). Деятельность с информационными объектами: создание, фиксация, обработка (редактирование), передача другому (презентационная графика), размещение в Интернете.

Процедуры поиска, сбора, организации информации. Создание информационных объектов (потребность — замысел — проект — прототип — продукт — усовершенствованный продукт).

Информационная среда школы как информационная система: структура, функционирование, проектирование.

Планирование индивидуальной и коллективной деятельности, использование информационных технологий (средства управления проектом). Организация коллективной деятельности с использованием информационных технологий (компьютерная сеть).

Социальная информатика

Информационное общество. Информационные ресурсы общества. Правовые и этические нормы информационной деятельности. Информационное пространство и индивидуальная коммуникация, их безопасность. Информационная безопасность. Информационная экология.

Профильный уровень

Требования к уровню подготовки выпускников

Изучение информатики должно предоставить учащимся возможность:

— развить представления об управлении процессами, обратной связи, автоматическом и программном (компьютерном) управлении, о принципах программного управления физическими объектами; строить программы формальных исполнителей с использованием базовых конструкций: выбора (ветвления), повторения, именования;

— использовать и строить цепочки (конечные последовательности) и таблицы для описания объектов информатики, в том числе естественных и искусственных языков; использовать и строить деревья (списки) для классификации информационных объектов, выбора действий, описания отношений родства; работать с древовидными структурами при обращении к файловой системе компьютера;

— получить представления о материальных и информационных моделях, их свойствах, рассуждать об адекватности модели моделируемому объекту и целям моделирования; освоить основные этапы построения моделей, получить представление о способах оценки моделей, приобрести опыт построения компьютерной визуализации моделей и процессов в области физики, экономики и экологии; получить представления о построении математических моделей игровой деятельности, выигрышных стратегиях в играх с полной информацией;

— получить представления об универсальных алгоритмах, сложности вычислений, кодировании и сложности информационных объектов, невозможности и нереальности некоторых алгоритмов, о доказательности правильности алгоритмов; освоить конкретные алгоритмы сортировки и перебора, в том числе описанные рекурсивно;

— уверенно владеть стандартными массовыми средствами работы с информационными объектами, создавать и редактировать их с помощью стандартных средств информационных и коммуникационных технологий;

— свободно пользоваться компьютером и типовым периферийным оборудованием (сканер, цифровая камера, принтер, мультимедийный проектор), стандартным компьютерным графическим интерфейсом;

— понимать специфику информационных систем массовой и индивидуальной коммуникации (почта, телефон, Интернет), уметь искать сведения, пользуясь информационными ресурсами библиотек, Интернета, осознанно пользоваться технологиями личной коммуникации (презентации) при подготовке и выполнении выступлений, технологиями размещения информации в Интернете;

— использовать правила защиты информации и средств информатизации от возможного проникновения вирусов, уметь использовать антивирусные программы;

— ознакомиться с основными шагами процедуры разработки информационных объектов и систем, накопить личный опыт этой работы на примере решения задач развития/поддержания информационной среды образовательного учреждения; планировать индивидуальную и коллективную деятельность с использованием программных инструментов поддержки управления проектом;

— получить представления о роли информатики и информационных технологий в развитии современной цивилизации, информационной инфраструктуре общества, юридических, этических и моральных нормах работы с информационными объектами; об информационной безопасности общества и личности, необходимости самоограничения человека, живущего в условиях избытка информации;

— овладеть информационными технологиями, математическими моделями и социальными аспектами информатики в приложении к профилю обучения.

Обязательный минимум содержания образования

Теоретическая информатика

Основные информационные процессы и их свойства, количественная оценка информации. Автоматизированное управление, обратная связь. Использование компьютера.

Исполнитель. Команды исполнителя. Прямое и программное управление объектами. Понятие алгоритма, его свойства и способы записей. Выполнение алгоритмов. Программа. Построение программ для решения задач в графической среде, присваивание значения имени в ходе выполнения программы. Рекурсия.

Имя и значение. Логические значения. Операции "Все", "Существует", отрицание. Свойства. Классификации. Таблицы.

Цепочки (конечные последовательности) в описании объектов информатики, в том числе естественных и искусственных языков. Деревья классификации, выбора действий, предков, потомков. Древовидная система хранения информационных объектов (система вложенных "папок" файловой системы). Игра с полной информацией, дерево игры. Стратегия выигрыша. Вероятность и случайность в описании реальности.

Моделирование как метод научного познания. Модели материальные и информационные, свойства моделей. Адекватность модели объекту и целям моделирования. Основные этапы построения моделей, формализация. Количественная и качественная оценки модели, адекватность объекту и цели моделирования.

Алгоритмы. Алгоритмическая сложность информационного объекта. Оценка сложности вычисления.

Математические модели и алгоритмы: алгоритмы сжатия информации, алгоритмы перебора вершин дерева, оптимизационные задачи, интервальный анализ, разностные уравнения. Математические модели экономических, экологических и физических процессов, их программная реализация.

Информационные технологии и средства информатизации

Основные принципы работы компьютера. Процессор, память, внешние устройства, хранимая программа, шина. Формирование изображения на мониторе, в проекторе и принтере. Способы хранения информации на машинных носителях. Интерфейс пользователя. Работа на клавиатуре.

Оценка объемов информации (текст, звук, рисунок, видео). Скорость передачи и обработки информации.

Принципы построения информационных систем вещания (телевидение, радио, газета) и индивидуальной коммуникации (почта, телефон, Интернет). Информационные ресурсы Интернета.

Средства и способы работы с информационными объектами (текст/гипертекст, звук, фотография, рисунок, чертеж, видеозапись, мультипликация, динамические/электронные таблицы). Деятельность с информационными объектами: создание, фиксация, обработка (редактирование), передача другому (презентационная графика), размещение в Интернете.

Процедуры поиска, сбора, организации информации. Разработка информационных объектов. Информационная среда школы как информационная система: структура, функционирование, проектирование.

Планирование индивидуальной и коллективной деятельности, использование информационных технологий (средства управления проектом). Организация коллективной деятельности с использованием информационных технологий (компьютерная сеть).

Восприятие и обработка информации человеком. Порог восприятия и порог различения. Шкала интенсивности (громкости, яркости).

Физические и технические основы современных информационных технологий (полупроводниковый переход, триггер, лазерная запись, жидкие кристаллы, радио и оптоволоконная связь). Представление данных в компьютере. Операционные системы и прикладные программы.

Внеклассная работа

Д.М. Златопольский,
Москва

Работа со стеком

Вам, очевидно, известно, что стеком называется структура данных, доступ к элементам которой основан на принципе "первым пришел — последним вышел".

Перед вами условный стек, элементы которого — термины, относящиеся к информатике. Необходимо выполнить приведенные ниже предписания и определить состояние стека.

ЕВКЛИД
ГЕРОН
ГИГАБАЙТ
СКАНЕР
ПРЕЗЕНТАЦИЯ
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

Начальное состояние

1. Если на вершине стека — фамилия древнегреческого философа, именем которого названа известная теорема о сторонах прямоугольного треугольника, то извлечь ее из стека.

2. Если на вершине стека — фамилия древнегреческого математика, автора алгоритма нахождения наибольшего общего делителя двух целых чисел, то извлечь ее из стека и добавить в стек название основного электронного устройства компьютера, управляющего его работой, иначе извлечь из стека 4 элемента.

3. Если на вершине стека — компьютерный слайд-фильм, создаваемый программой Microsoft PowerPoint, то добавить в стек термин, которым называют несколько транзисторов, объединенных в схему, осуществляющую элементарные операции по обработке информации; затем добавить в стек название места подключения внешних устройств к внутренней шине микропроцессора; затем — название символа, используемого в программе для разделения целой и дробной частей числа, иначе извлечь из стека один элемент.

4. Если на вершине стека — название устройства для приема-передачи компьютерных данных по телефонным линиям, то добавить название логической операции, которую называют также логическим умножением, в противном случае извлечь из стека один элемент.

5. Если на вершине стека — единица измерения информации, равная 1024 килобайтам, то добавить в стек название алгоритмической конструкции, обеспечивающей повторение одних и тех же операций; а затем название логической операции, которую называют также логическим сложением.

6. Если на вершине стека — логическая операция, для обозначения которой используется слово AND, то извлечь из стека два элемента.

7. Если на вершине стека — единица измерения информации, равная 2^{20} килобайтам, то извлечь из стека три элемента, иначе извлечь из него все элементы.

8. Если стек пуст, то добавить в него поочередно название компьютера, предоставляющего свои услуги другим компьютерам в сети, и слово СЧЕТЫ, иначе добавить в стек название вычислительного устройства у древних греков и римлян, похожего на счеты, и слово АРИФМОМЕТР.

9. Если на вершине стека — название механического вычислительного устройства, то извлечь из стека два элемента, иначе добавить в стек название программы, обладающей способностью к самовоспроизведению и выполняющей нежелательные действия.

10. Если на вершине стека — фамилия изобретателя системы кодирования информации, использующей 2 символа — точку и тире, то извлечь из стека два элемента, иначе добавить в него слово ДЛИНА.

11. Если на вершине стека — характеристика файла, переменной величины или константы, то извлечь из стека все элементы.

12. Если стек не пуст, то добавить в него поочередно слова:

ФОРМАТИРОВАНИЕ, БИТ, ИНФОРМАТИКА, ПРИНТЕР, ПОДПРОГРАММА,

иначе добавить в стек слова:

ДЕВЯТЬ, СКАНЕР, ИНТЕРНЕТ, КУРСОР.

В стеке находятся термины (начиная с вершины):

1. КУРСОР — указатель места на экране.
2. ИНТЕРНЕТ — глобальная компьютерная сеть.
3. СКАНЕР — устройство ввода информации в компьютер.
4. ДЕВЯТЬ — цифра десятичной системы счисления.

Состояние стека после выполнения каждого шага алгоритма

ЕВКЛИД
МОДЕМ
ГИГАБАЙТ
СКАНЕР
ПРЕЗЕНТАЦИЯ
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 1

ПРОЦЕССОР
МОДЕМ
ГИГАБАЙТ
СКАНЕР
ПРЕЗЕНТАЦИЯ
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 2

МОДЕМ
ГИГАБАЙТ
СКАНЕР
ПРЕЗЕНТАЦИЯ
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 3

КОНЪЮНКЦИЯ
МОДЕМ
ГИГАБАЙТ
СКАНЕР
ПРЕЗЕНТАЦИЯ
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 4

КОНЪЮНКЦИЯ
МОДЕМ
ГИГАБАЙТ
СКАНЕР
ПРЕЗЕНТАЦИЯ
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 5

ГИГАБАЙТ
СКАНЕР
ПРЕЗЕНТАЦИЯ
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 6

МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 7

АРИФМОМЕТР
АБАК
МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 8

МОРЗЕ
АЛГОРИТМ
ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 9

ИМЯ
ПАСКАЛЬ
АЛГОРИТМ

После шага 10

Стек пуст

После шага 11

КУРСОР
ИНТЕРНЕТ
СКАНЕР
ДЕВЯТЬ

После шага 12

Третий — лишний

Для каждого из выделенных курсивом терминов приведены три определения, одно из которых не соответствует термину. Необходимо указать номер неверного определения.

1. *Автомат* — это...

а) устройство, машина или система, выполняющие действия без участия человека;

б) программа, постоянно находящаяся в оперативной памяти компьютера и автоматически выполняемая при возникновении определенных условий;

в) огнестрельное оружие.

2. *Бегунок* — это...

а) указатель места ввода символа на экране;

б) подвижный элемент распределителя зажигания в карбюраторном двигателе;

в) элемент управления на полосе прокрутки в окне текстового редактора Microsoft Word.

3. *Интерпретация* — это...

а) истолкование, раскрытие смысла или содержания чего-нибудь;

б) процесс сохранения файла на диске в сжатом виде;

в) один из способов трансляции программы.

4. *Ключ* — это...

а) предмет, с помощью которого открывают замок;

б) поле в базе данных, по которому проводится, например, ее сортировка;

в) условие, по которому проводится отбор записей в базе данных.

5. *Магистраль* — это...

а) совокупность линий связи процессора, оперативной памяти и устройства ввода-вывода;

б) провод, питающий электрическим током устройства компьютера;

в) широкая и прямая улица или дорога.

6. *Окно* — это...

а) остекленное отверстие в стене;

б) часть экрана, занимаемая приложением или документом Windows;

в) прямоугольник, обрамляющий таблицу.

7. *Плата* — это...

а) металлическая пластинка на задней панели системного блока персонального компьютера, закрывающая отверстие в нем;

б) отдача денег за проезд в общественном транспорте;

в) деталь системного блока персонального компьютера в виде пластины, на которой установлены микросхема и другие элементы.

8. *Приложение* — это...

а) программа, созданная для работы под управлением операционной системы Windows;

б) в грамматике — определение, выраженное существительным;

в) программный модуль, в котором собраны вспомогательные процедуры и функции.

9. *Пробел* — это...
а) недостаток в знаниях;
б) незаполненный разъем расширения в материнской (системной) плате компьютера;
в) название клавиши.

10. *Программа* — это...
а) синоним слова "файл";
б) перечень номеров, включенных в концерт;
в) алгоритм, записанный на языке, понятном ЭВМ.

11. *Пункт* — это...
а) раздел меню;
б) часть программы;
в) пронумерованная часть текстового документа.

12. *Разметка* — это...
а) синоним слова "форматирование", подготовка диска к работе;
б) знаки на покрытии автомобильной дороги;
в) шаблон, образец текстового документа.

13. *Режим* — это...
а) распорядок дел, действий;
б) совокупность особенностей трансляции программы;
в) совокупность характеристик работы монитора компьютера.

14. *Узел* — это...
а) точка графа (граф можно представить в виде конечного числа точек на плоскости, соединенных отрезками кривых линий);
б) место в программе, в котором происходит вызов вспомогательной процедуры;
в) место, где туго соединены концы чего-нибудь, например, шнурков.

15. *Указатель* — это...
а) отметка в тексте программы, с помощью которой можно перейти в это место;
б) дорожный знак;
в) в Паскале — переменная, значением которой является адрес байта памяти.

16. *Фильтр* — это...
а) поле в базе данных, по которому проводится, например, ее сортировка;
б) условие, по которому проводится отбор записей в базе данных;
в) устройство для очистки воды, воздуха и др.

Ответы: 1 — б, 2 — а, 3 — б, 4 — в, 5 — б, 6 — в, 7 — а, 8 — в, 9 — б, 10 — а, 11 — б, 12 — в, 13 — б, 14 — б, 15 — а, 16 — а.

Омонимы

Вы, конечно, знаете, какие слова называются омонимами (слова, совпадающие по звучанию, но полностью различающиеся по значению). Перед вами две колонки, в которых даны определения некоторых терминов (в левой колонке приведены определения терминов, касающихся информатики). Необходимо найти пары определений (по одному из каждой колонки), относящиеся к терминам-омонимам.

1. Поле в базе данных, по которому проводится, например, ее сортировка	1. Перечень видов спорта, включенных в соревнования
2. В программировании — тип данных, при помощи которого реализуется последовательность однородных элементов	2. Увеличение ширины чего-либо
3. Деталь системного блока персонального компьютера в виде пластины, на которой установлены микросхема и другие элементы	3. Большое пространство, однородное по каким-либо признакам
4. Алгоритм, записанный на языке, понятном ЭВМ	4. Процесс "заводки" двигателя автомобиля
5. Язык программирования, используемый для решения задач искусственного интеллекта	5. Листок с наименованием товара и другими сведениями о нем
6. Кнопка на рабочем столе Windows	6. Вступительная часть литературного произведения
7. Часть имени файла	7. Широкая и прямая улица или дорога
8. Условие, по которому проводится отбор записей в базе данных	8. Отдача денег за проезд в общественном транспорте
9. Значок на экране, щелкнув мышью на котором, можно открыть некоторую программу, документ или папку	9. Истолкование, раскрытие смысла или содержания чего-нибудь
10. Разновидность трансляции программы, при которой действия (операторы) выполняются по мере перевода их на машинный язык	10. Недостаток в знаниях
11. Совокупность линий связи процессора, оперативной памяти и устройства ввода-вывода	11. Предмет, с помощью которого открывают замок
12. Так называют программу, созданную для работы под управлением операционной системы Windows	12. Устройство для очистки воды, воздуха и др.
13. Название клавиши	13. В грамматике — определение, выраженное существительным

Ответы: 1 — 11 (ключ), 2 — 3 (массив), 3 — 8 (плата), 4 — 1 (программа), 5 — 6 (Пролог), 6 — 4 (пуск), 7 — 2 (расширение), 8 — 12 (фильтр), 9 — 5 (ярлык), 10 — 9 (интерпретация), 11 — 7 (магистраль), 12 — 13 (приложение), 13 — 10 (пробел). Первыми указаны номера определений в левой колонке.

Компактный и легко расширяемый

Чарльз Мур

"Форт — язык программирования, позволяющий в сжатые сроки разрабатывать надежные программы. На Форте создавалось программное обеспечение космического корабля Space Shuttle, искусственных спутников Земли, международных аэропортов. Форт обладает исключительными возможностями..." [1].

"Форт — ...очень компактный, легко расширяемый язык высокого уровня общего назначения... Широко применяется для управляющих систем" [2].

"Форт — необычный язык. Он "попирает" многие устоявшиеся правила программирования" [3].

Основным понятием языка Форт (Forth) является слово, под которым понимается любая последовательность символов (включая знаки препинания, но исключая пробелы). Язык имеет простой синтаксис. Программа представляет собой последовательность отдельных слов и очень напоминает текст печатного издания. Одна из особенностей Форта — компактность. Другое свойство, отличающее Форт от других языков, — легкая "расширяемость": программист может без труда определять новые ключевые слова или команды [1—10].

Вначале создатель языка Чарльз Мур хотел дать своему детищу название Fourth (четвертый), поскольку ему казалось, что мощь языка настолько велика, что он на целое поколение опередил языки для использовавшихся тогда компьютеров третьего поколения. (По словам Мура, новый язык повысил производительность его работы в 10 раз.) Однако машина IBM 1130, на которой Мур в то время работал, не позволяла употреблять идентификаторы, содержащие более пяти символов, и ему пришлось сократить название языка до Forth (вперед). Позже Мур охарактеризовал это как "утонченную игру слов" [4].

Впервые Форт стал активно применяться в начале 1970-х годов, когда Мур работал в Национальной радиоастрономической обсерватории в Аризоне.

Совместно с Элизабет Разер, отвечавшей за программное обеспечение обсерватории, он использовал Форт при подготовке серии программ для мини-компьютеров, которые, в частности, управляли в реальном масштабе времени системой наведения 11-метрового телескопа [4, 5]. Программы и сама система оказались столь удачными, что в 1973 году Мур, Разер и их руководитель Нед Конклин решили создать компанию Forth, Inc., которая стала продавать системы не только для обсерваторий, но и для других объектов, где требовалось управление в реальном масштабе времени.

Одна из систем такого рода — для автоматического управления видеокамерами — была установлена на подводном аппарате "Арго", использовавшемся в 1985 году при поисках в водах Северной Атлантики затонувшего в 1912 году океанского лайнера "Титаник". (Тогда после 16 дней упорных поисков франко-американская научная экспедиция обнаружила "Титаник" на глубине более трех километров.)

Хотя фирма не проявляла особого интереса к рынку персональных компьютеров, Форт вызвал своего рода ажиотаж среди любителей таких машин. В числе этих любителей был молодой инженер Ким Харрис. Большое впечатление на него произвело то, как на семинаре, проводимом компанией, один из демонстраторов за 15 минут составил простую программу для исполнения компьютерной музыки. Харрис знал квалифицированного любителя, который трудился больше года, чтобы получить подобную программу на ассемблере. "Это было подобно чуду, — вспоминал он, — и я увидел его собственными глазами" [4].

В 1977 году Харрис организовал группу под названием FIG (от *FORTH Interest Group*), которая уже почти через полгода создала упрощенный интерпретатор языка Форт для персональных компьютеров, получивший название FIG-FORTH. В дальнейшем были созданы и другие версии Форта для микрокомпьютеров, однако очень многие пользователи отдавали предпочтение системе FIG-FORTH.

У нас первая книга, посвященная языку Форт, вышла в 1988 году (см. [11]), хотя работы, связанные с его реализацией на отечественных компьютерах, начались раньше...

Литература

1. *Шереметьев К.П.* Форт. Спецвыпуски // Информатика, № 45, 47/95.
2. *Пройдаков Э.М., Теплицкий Л.А.* Англо-русский толковый словарь по вычислительной технике, Интернету и программированию. Изд. 2-е, испр. и доп. М.: Издательско-торговый дом "Русская редакция", 2000.
3. *Броуди Л.* Начальный курс программирования на языке Форт: Пер. с англ. М.: Финансы и статистика, 1990.
4. Язык компьютера: Пер. с англ. М.: Мир, 1989.
5. *Малыхина М.П., Частиков А.П.* Языки программирования: Форт // Новое в жизни, науке, технике. Сер. "Вычислительная техника и ее применение", № 12/90.
6. *Шереметьев К.П.* Язык программирования Форт // Информатика, № 13/95.
7. *Левин Д.С., Сенокосов А.И.* Язык Форт // Информатика, № 31/96.
8. "Четвертый" // Информатика, № 45/2000.
9. Толковый словарь по вычислительной технике (Microsoft Corporation): Пер. с англ. М.: Издательский отдел "Русская редакция" ТОО "Channel Trading Ltd.", 1995.
10. Толковый словарь по вычислительным системам / Под ред. В.Иллингуорта и др.: Пер. с англ. М.: Машиностроение, 1990.
11. *Баранов С.Н., Ноздрунов Н.Р.* Язык Форт и его реализация. Л.: Машиностроение, 1988.

"Неограненный алмаз"

Окончание. Начало на с. 1

Pentium — "торговая марка, под которой выпускается серия процессоров для персональных компьютеров; принадлежит фирме Intel" [2]. Серия включает в себя: *Pentium* (появился в 1993 г.), *Pentium Pro* (1995 г.), *Pentium MMX* (1997 г.), *Pentium II* (1998 г.), *Pentium III* (1999 г.), *Pentium 4* (2000 г.).

В процессоре *Pentium* "используется скоростная версия технологии компьютера с полным набором команд (*Complex Instruction Set Computer*, CISC), суперскалярная архитектура (с двумя конвейерами) и 64-разрядная внутренняя шина данных", хотя внутренние регистры и адреса являются 32-разрядными [3—5]. [Компьютер с полным (обычным) набором команд — классический вариант компьютера, имеющий обширный набор достаточно сложных команд; суперскалярный — имеющий несколько конвейеров и способный исполнять более одной команды за один такт.]

Pentium Pro — существенно усовершенствованная версия процессора Pentium. При его создании были применены два подхода для увеличения производительности [6, 7]. Один из них предусматривает повышение тактовой частоты, что способствует увеличению количества операций, которое центральный процессор способен выполнить за единицу времени. Другой путь — увеличение степени "параллелизма" процессора — способствует выполнению большего количества операций в течение каждого периода тактовой частоты.

В процессоре *Pentium MMX* использован "набор из 57 дополнительных команд для ускорения работы с графикой, звуком и видео" [8]. Традиционные 32-разрядные процессоры способны выполнять сложение двух 8-разрядных чисел, размещая каждое из них в младших разрядах 32-разрядных регистров. При этом 24 старших разряда регистров не употребляются, и потому, например, при одной операции сложения ADD осуществляется просто сложение двух 8-разрядных чисел. Команды MMX оперируют сразу с 64 разрядами, где могут храниться восемь 8-разрядных чисел, причем имеется возможность выполнить их сложение с другими 8-разрядными числами в процессе одной операции ADD. Регистры MMX могут употребляться также для одновременного сложения четырех 16-разрядных слов или двух 32-разрядных длинных слов. Такой принцип получил название SIMD (*Single Instruction/Multiple Data* — "один поток команд /много потоков данных") [9—12].

Pentium II — "логическое продолжение" и значительно усовершенствованная версия процессора Pentium. Здесь также употребляется набор "мультимедийных команд", а кроме того, реализованы еще два технологических достижения фирмы Intel: архитектура так называемого динамического выполнения (Dynamic Execution) и архитектура двойной независимой шины (Dual Independent Bus), состоящей из отдельных системной шины и шины кэша [2]. Кстати, процессор Celeron [в названии употреблен английский корень *celer* ("быстрый") и окончание *-on*] был предложен фирмой Intel как "дешевая альтернатива" Pentium II.

Процессор *Pentium III* — это несколько модифицированный Pentium II, причем "его название можно отнести к чисто рекламному трюку" [13].

Что же касается процессора *Pentium 4*, то здесь применена архитектура, которую специалисты Intel назвали NetBurst Micro-architecture[1]. Она призвана ускорить работу прикладных программ, выдающих данные большими "пакетами". "Эта архитектура — новая ступень в развитии процессора: от обычного исполнения (оптимизированного для деловых приложений) к мультимедийному" [1].

Например, в процессоре Pentium 4 впервые используется Streaming SIMD Extensions 2 (SSE2)[2] — набор новых мультимедийных и графических команд, служащих для ускорения работы различных прикладных программ. Так, определенные команды SSE2 сокращают время кодирования и обработки речи, видеоизображений, работы с ресурсами Internet (для чего, вообще говоря, и был создан данный процессор). "Все конструктивные особенности нового процессора... нацелены на достижение высокой производительности при обработке мультимедиаданных... Но то, что хорошо для работы с мультимедиа, не обязательно приносит пользу при выполнении стандартных офисных программ, применяемых наиболее часто" [14]. Следует также учитывать, что разработчики программного обеспечения стараются не ориентироваться на самые совершенные компьютеры. Для таких программ трудно найти покупателей. "Программные средства, рассчитанные на новые аппаратные средства, появляются лишь после того, как в руках пользователей оказывается достаточное количество быстродействующих ПК".

Литература

1. *Эссекс Д.* Pentium 4: прорыв или провал? // Мир ПК, № 2/2001.
2. *Мостицкий И.Л.* Новейший англо-русский толковый словарь по современной электронной технике. М.: ЛУЧШИЕ КНИГИ, 2000.
3. *Шниер М.* Толковый словарь компьютерных технологий: Пер. с англ. Киев: ДиаСофт, 2000.
4. *Фафенбергер Б., Уолл Д.* Толковый словарь по компьютерным технологиям и Internet: Пер. с англ. Изд. 6-е. Киев: Диалектика, 1996.
5. Пентиумы // Информатика, № 17/2001.
6. *Рапли С., Клаймен Д.* P6: процессор нового поколения // PC Magazine/Russian Edition, № 12/95.
7. Pentium для ПРОфессионалов // Информатика, № 40/99.
8. *Пройдаков Э.М., Теплицкий Л.А.* Англо-русский толковый словарь по вычислительной технике, Интернету и программированию. Изд. 2-е, испр. и доп. М.: Издательско-торговый дом "Русская редакция", 2000.
9. *Частиков А.П.* От калькулятора до супер-ЭВМ // Новое в жизни, науке, технике. Сер. "Вычислительная техника и ее применение", № 1/88.
10. *Мец К.* MMX: успешный дебют // PC Magazine/Russian Edition, № 4/97.
11. *Клаймен Дж., Стам Н.* Как работает технология MMX // PC Magazine/Russian Edition, № 4/97.
12. Мультимедийный процессор // Информатика, № 9/2001.
13. *Сенокосов А.И.* Что там inside? Заметки о новом процессоре Pentium III // Информатика, № 17/99.
14. *Мец К.* Быстродействие ЦП: где разумный предел? // PC Magazine/Russian Edition, № 4/2001.

[1] NetBurst — "сетевой пакет".

[2] Streaming SIMD Extensions (SSE) — *потоковые расширения SIMD*, используются в процессорах Pentium III.

Гл. редактор
С.Л. Островский
Зам. гл. редактора
А.И. Сенокосов
Редакция:
Е.В. Андреева
Н.Л. Беленькая
Л.Н. Картвелишвили
Н.П. Медведева
Дизайн и верстка:
Н.И. Пронская
Корректоры:
Е.Л. Володина,
С.М. Подберезина

выходит четыре раза в месяц
При перепечатке ссылка на ИНФОРМАТИКУ обязательна, рукописи не возвращаются

Адрес редакции и издателя:
121165, Киевская, 24
тел. 249-48-96
Отдел рекламы
тел. 249-98-70

Учредитель: ООО "Чистые пруды"

Зарегистрировано в Министерстве РФ по делам печати. ПИ № 77-7230 от 12.04.2001.
Отпечатано в ОИД "Медиа-Пресса", 125993, ГСП-3, Москва, А-40, ул. "Правды", 24.
Тираж 6000 экз.
Срок подписания в печать по графику 14.11.2001.
Номер подписан 14.11.2001
Заказ № 17485
Цена свободная

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта изданий 32744

Тел.: (095)249-31-38, 249-33-86. Факс (095)249-31-84

Internet: inf@1september.ru
WWW: http://www.1september.ru

Главный редактор комплекта изданий А.С. Соловейчик

Английский язык (Е.В. Громушкина), индекс подписки — 32025; **Библиотека в школе** (О.К. Громова), индекс подписки — 33376; **Биология** (Н.Г. Иванова), индекс подписки — 32026; **Воскресная школа** (монах Киприан (Ященко), индекс подписки — 32742; **География** (О.Н. Коротова), индекс подписки — 32027; **Здоровье детей** (А.У. Лекманов), индекс подписки — 32033; **Информатика** (С.Л. Островский), индекс подписки — 32291; **Искусство** (Н.Х. Исмаилова), индекс подписки — 32584; **История** (А.Ю. Головатенко), индекс подписки — 32028; **Литература** (Г.Г. Красухин), индекс подписки — 32029; **Математика** (И.Л. Соловейчик), индекс подписки — 32030; **Начальная школа** (М.В. Соловейчик), индекс подписки — 32031; **Немецкий язык** (М.Д. Бузоева), индекс подписки — 32292; **Русский язык** (Л.А. Гончар), индекс подписки — 32383; **Спорт в школе** (Н.В. Школьникова), индекс подписки — 32384; **Управление школой** (А.И. Адамский), индекс подписки — 32652; **Физика** (Н.Д. Козлова), индекс подписки — 32032; **Французский язык** (Г.А. Чесновицкая), индекс подписки — 33371; **Химия** (О.Г. Блохина), индекс подписки — 32034; **Школьный психолог** (М.Н. Сартан), индекс подписки — 32898.