№ 18 (211) май 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Электронные версии газеты “Первое сентября” и приложений http://www.1september.ru

«Комтек-99»: картинки с выставки

«Комтек» этого года стал, пожалуй, первой крупной компьютерной выставкой, на которой в полной мере сказался августовский кризис. Сентябрьский Softool в этом смысле был менее показателен, ибо участие в нем было оплачено задолго до 17 августа и целый ряд фирм выставлялся “по инерции”: не пропадать же деньгам.

Итак, начали: очередь (очень небольшая), касса (билет 40 рублей) и – вперед.

Так что «Комтек» этого года был куда меньше и скромнее предыдущих. Но не надо думать, что все те, кто не принял участия в выставке, уже никогда и ни в чем участия не примут. Вовсе нет. Целый ряд известных российских фирм, в основном российские производители программного обеспечения, продолжают жить... ну, не поживать, а скорее выживать, но жить. Вероятно, они посчитали, что в нынешней ситуации есть более насущные проблемы, требующие денежных вложений. А с выставочной мишурой можно годик подождать.

Понимая, что большая часть наших читателей не посещала выставку, и руководствуясь известным соображением, что лучше один раз увидеть, мы решили подготовить краткий фоторепортаж о «Комтеке». Конечно, не много из того, что представлено на фотографиях, даже в отдаленном будущем окажется в наших школах (да и не все так уж в школе и нужно). Но посмотреть ведь все равно интересно?

В этом году на выставке работали всего два павильона.

Охрана по неизвестным причинам крайне недоброжелательно относилась к панорамным снимкам.

Чем бы еще поразить посетителей (и конкурентов!)? Вроде и принтеры A0 уже имеются, и "плоские" 19-дюймовые мониторы не редкость, и клавиатуры "ломают" все кому не лень...

НАШИ ДЕТИ БУДУТ ЖИТЬ В XXI ВЕКЕ

Системы счисления и компьютерная арифметика

Е.В. АНДРЕЕВА, И.Н. ФАЛИНА

Начало в № 14, 15, 16, 17/99.

В этом выпуске завершена публикация главы 3 и помещена

Глава 4. “Смешанные и нетрадиционные системы счисления”.

Продолжение следует

Логические задачи на вступительных экзаменах по информатике

В.И. РАКИТИН

Продолжение. Начало в № 17/99

2.3. Построение логического выражения по условиям, заданным в виде текста

Задача 4. Пусть три человека *A, B, C* являются участниками соревнований. Составить логические функции, соответствующие высказываниям:

1) все трое займут призовые места,
2) ни один из них не займет призового места,
3) только один из них займет призовое место,
4) двое займут призовые места,
5) хотя бы один из них займет призовое место,
6) если или *A* или *B* не займут призового места, то *C* займет,
7) если *B* займет призовое место, то или *C* займет призовое место, или *A* не займет.

Замечание. Значение построенной логической функции должно быть равно 1 для верного высказывания в соответствующем случае.

Решение

Обозначим через *A, B, C* логические переменные, обозначающие, что участники соревнований *A, B, C* занимают независимо друг от друга призовые места. Призовые места соответствуют значениям переменных $A=1$, $B=1$, $C=1$; для неудач $A=0$, $B=0$, $C=0$.

1) $f_1(A,B,C) = A \cdot B \cdot C$ — функция равна 1 только для одной комбинации переменных $A=B=C=1$;

2) $f_2(A,B,C) = \overline{A} \cdot \overline{B} \cdot \overline{C}$ — функция равна 1 только для одной комбинации переменных $A=B=C=0$;

3) $f_3(A,B,C) = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C$ — функция равна 1 для комбинации переменных $A=1$, $B=C=0$, для $A=0$, $B=1$, $C=0$ и для $A=B=0$, $C=1$: каждому из трех слагаемых соответствует только один призер;

4) $f_4(A,B,C) = A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C$ — функция равна 1 для комбинации переменных $A=B=1$, $C=0$, для $A=0$, $B=C=1$ и для $A=1$, $B=0$, $C=1$: каждому из трех слагаемых соответствуют два призера;

5) $f_5(A,B,C) = f_3(A,B,C) + f_4(A,B,C) + f_1(A,B,C) = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot C$ — это или 1 призер, или 2 призера, или 3 призера; функция не равна 1 только для комбинации $A=B=C=0$ и может быть построена как: $f_5(A,B,C) = \overline{\overline{A} \cdot \overline{B} \cdot \overline{C}}$ — отрицание комбинации "нет ни одного призера" или $f_5(A,B,C) = A + B + C$ — хотя бы один призер;

6) $f_6(A,B,C) = (\overline{A} + \overline{B} \Rightarrow C) = \overline{\overline{A} + \overline{B}} + C = AB + C = C + AB$ — функция равна 1 — утверждение верно, когда либо *C* — призер при любых успехах или неудачах *A* и *B*, либо призерами окажутся одновременно *A* и *B* при любых исходах для *C*;

7) $f_7(A,B,C) = (B \Rightarrow (C + \overline{A})) = \overline{B} + C + \overline{A} = \overline{A} + \overline{B} + C$ — утверждение верно, когда функция равна 1 (рассмотрите все возможные комбинации).

2.4. Построение логической функции по таблице истинности

Задача 5. Построить логическую функцию, значение которой равно 1 на следующих комбинациях трех логических переменных *A, B* и *C*: $A=0$, $B=0$, $C=1$; $A=1$, $B=0$, $C=0$; $A=1$, $B=0$, $C=1$. При остальных значениях переменных значение функции равно 0.

Решение

Из таблицы видно, что функция строится в виде суммы трех произведений переменных и их отрицаний. Каждое произведение имеет значение, равное 1 на наборе переменных, определенном в условии задачи. Легко убедиться, что так построенная функция имеет значение, равное 0, на остальных наборах переменных.

№ набора	*A*	*B*	*C*	$f(A,B,C)$	
0	0	0	0	0	0
1	0	0	1	1	$\overline{A} \cdot \overline{B} \cdot C$
2	0	1	0	0	0
3	0	1	1	0	0
4	1	0	0	1	$A \cdot \overline{B} \cdot \overline{C}$
5	1	0	1	1	$A \cdot \overline{B} \cdot C$
6	1	1	0	0	0
7	1	1	1	0	0

$$f(A,B,C) = \overline{A} \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C$$

Изменим формулировку задачи и построим логическую функцию, значение которой равно 0 на следующих комбинациях трех логических переменных *A, B* и *C*:
$A=B=C=0$;
$A=0$, $B=1$, $C=0$;
$A=0$, $B=C=1$;
$A=B=1$, $C=0$;
$A=B=C=1$
(противоположные комбинации). При остальных значениях переменных значение функции равно 1.

Требуемая функция может строиться теперь в виде произведения сумм из трех слагаемых переменных и их отрицаний. Каждая сумма имеет значение, равное 0, на наборе переменных, определенном в условии новой задачи.

Так, построенная функция имеет значение, равное 0 на заданных комбинациях переменных, и 1 — на остальных наборах переменных.

№ набора	*A*	*B*	*C*	$f(A,B,C)$	
0	0	0	0	0	$A + B + C$
1	0	0	1	1	1
2	0	1	0	0	$A + \overline{B} + C$
3	0	1	1	0	$A + \overline{B} + \overline{C}$
4	1	0	0	1	1
5	1	0	1	1	1
6	1	1	0	0	$\overline{A} + \overline{B} + C$
7	1	1	1	0	$\overline{A} + \overline{B} + \overline{C}$

$$f(A,B,C) = (A + B + C)(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C})$$

При одинаковых таблицах истинности для функции имеем два различных ее представления:

$$f(A,B,C) = (A + B + C)(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C}),$$

$$f(A,B,C) = \overline{A} \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C.$$

Задача 6. Построить логическую функцию, значение которой равно 1 на наборах битов, определяемых двоичными представлениями чисел: $X = 0_{10}$; $Y = 2_{10}$; $Z = 7_{10}$. На остальных наборах функция должна быть равна 0.

Решение

Используя двоичную систему счисления, представим данные числа в формате из трех битов: $X=0_{10}=000_2$; $Y=2_{10}=010_2$; $Z=7_{10}=111_2$. Далее строим функцию $f(A,B,C)$ в виде суммы трех произведений переменных *A, B, C* и их отрицаний, как в предыдущей задаче. Каждое произведение имеет значение, равное 1 на наборе битов, определенных заданными числами, т.е. $f(X) = f(Y) = f(Z)=1$, или $f(0) = f(2) = f(7)=1$, или $f(000) = f(010) = f(111)=1$.

Построение функции

	A	*B*	*C*	$f(A,B,C)$	
$X = 0_{10}$	0	0	0	1	$\overline{A} \cdot \overline{B} \cdot \overline{C}$
$Y = 2_{10}$	0	1	0	1	$\overline{A} \cdot B \cdot \overline{C}$
$Z = 7_{10}$	1	1	1	1	$A \cdot B \cdot C$

$$f(A,B,C) = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

Продолжение следует

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Д.М. ЗЛАТОПОЛЬСКИЙ

Окончание. Начало в № 17/99

В предыдущем номере были рассмотрены основные понятия динамического программирования, приведены условия трех задач и рассмотрено решение одной из них. В этом номере рассматриваются решения двух оставшихся задач.

Задача 2.

Введем обозначения:

`Q` — грузоподъемность машины;

`N` — количество предметов;

`W[1:N]` , `C[1:N]` — вес и стоимость предметов.

Напомним, что состоянием системы в данной задаче является вес, который еще можно взять в машину после предыдущих шагов (остаточная грузоподъемность). Обозначим состояние как `rest`. Грузоподъемность машины и веса предметов считаются целыми числами, поэтому возможные состояния принимают все целочисленные значения от 0 до `Q`. Управление в каждом состоянии и для каждого шага заключается в ответе на вопрос: берем данный предмет или не берем, что кодируется соответственно единицей и нулем. Будем хранить значения условных оптимальных управлений в двумерном массиве `Contr[0:Q, 1:N]`.

Для хранения условных оптимальных выигрышей, уже найденных на шаге `Nstep`, используем массив `S[0:Q]`, а при переходе к предыдущему шагу `(Nstep - 1)` условные оптимальные выигрыши заносятся в массив `Snew[0:Q]`. Причем после заполнения массива `Snew` значения его элементов переписываются в массив `S` и используются на другом шаге.

Как мы помним, поиск условных оптимальных управлений и выигрышей надо начинать с последнего шага. Здесь управление вынужденное: если вес `N`-го предмета не больше остаточной грузоподъемности, то предмет берется, а в противном случае — не берется, поскольку это невозможно. Запишем сказанное более формально:

```
если W[ N] > rest то Contr[ rest, N] := 0; S[ rest ] := 0
иначе нач Contr[ rest, N] := 1; S[ rest ] := C[ N] все
```

Отсюда и находим указанные величины при всех возможных значениях состояния `rest`.

Теперь приступаем к нахождению условных оптимальных управлений и выигрышей на всех промежуточных шагах от предпоследнего до второго. Пусть на некотором шаге `(Nstep + 1)` условные оптимальные выигрыши уже найдены и хранятся в массиве `S`. Определим их для предыдущего шага `Nstep`. По-прежнему, если вес предмета с номером `Nstep` больше остаточной грузоподъемности `rest`, то предмет не берется, иначе выясняем, нужно ли его брать. Находим условные выигрыши `Syes`, `Sno` для обоих вариантов управления и выбираем тот, при котором условный выигрыш больше. Более подробная запись сказанного:

```
если W[ Nstep] > rest
  | то
  | Contr[ rest, Nstep] := 0; Snew[ rest ] := 0
  | иначе
  |    Sno := S[ rest ]
  |    Syes := S[ rest - W[ Nstep] ] + C[ Nstep ]
  |    если Syes > Sno
  |    то
  |    Contr[ rest, Nstep] := 1; Snew[ rest ] := Syes
  |    иначе Contr[ rest, Nstep] := 0; Snew[ rest ] := Sno  все
все
```

Выполняем эти операции при всех возможных значениях состояния `rest`, после чего копируем массив `Snew` в `S`.

Наконец, рассмотрим первый шаг. Для него существует лишь одно состояние `Q`. Аналогично предыдущему находим оптимальное управление на первом шаге `Contr[Q, 1]`.

Осталось определить оптимальные управления на следующих шагах. Для этого, зная оптимальное управление и состояние на предыдущем шаге, находим состояние на очередном шаге и берем условное оптимальное управление этого состояния и шага.

Решение этой задачи рассмотрено в [7]. Там же представлена программа на Паскале. Поэтому здесь мы приведем программы на других языках программирования.

Школьный алгоритмический язык

```
цел Q, N | Глобальные параметры задачи
алг Задача 2
нач цел  Nstep, rest,MaxS
  | Q:= 45 | Грузоподъемность автомобиля
  | N:= 6   | Число предметов
  | Описываем массивы
  | цел таб W[1: N], C [1: N], Contr[0: Q, 1: N], S[0: Q], Snew[0: Q]
  |Заполняем массивы W и C
  | W[1]:=4; W[2]:=7; W[3]:=11; W[4]:=12; W[5]:=16; C[6]:=27
  | C[1]:=7; C[2]:=10; C[3]:=15; C[4]:=20; C[5]:=27; C[6]:=34
  |Определяем условные оптимальные параметры
  |Последний шаг
  | нц для rest от 0 до Q
  |   | если W[N] > rest  |взять последний предмет нельзя
  |   |   | то
  |   |   | Contr[ rest, N]:=0;  S[rest]:=0
  |   |   | иначе            |можно взять последний предмет
  |   |   | Contr[ rest, N]:=1;  S[rest]:=C[N]
  |   | все
  | кц
```

```
|Остальные шаги (кроме первого)
нц для Nstep от N-1 до 2  шаг  -1
  | нц для rest от 0 до Q
  |   | если W[Nstep] > rest
  |   | то       |взять предмет с номером Nstep нельзя
  |   |  | Contr[rest, Nstep]:=0; Snew[rest]:=0
  |   |  | иначе   |можно
  |   |  |с помощью процедуры NewControl определяем, стоит ли это делать
  |   |  | NewControl(rest, Nstep, W, C, S, Contr[rest, Nstep],
  |   |  | Snew[rest])
  |   | все
  | кц
  | нц для rest от 0 до Q
  |  | S[rest]:=Snew[rest]    |Переписываем Snew в S
  | кц
кц
|Первый шаг
|Возможное состояние одно - можно взять Q единиц груза
   NewControl( Q, 1, W, C, S, Contr[Q, 1], MaxS])
|Печатаем максимальный выигрыш
вывод нс, "Максимальная суммарная стоимость: ", MaxS
| Определяем вариант управления, при котором он достигается
вывод нс, "Оптимальное управление:"
rest:=Q
нц для Nstep от 1 до N
  | если Contr[rest, Nstep]=1
  |   | то
  |   | вывод нс, Nstep, "-й предмет берем"
  |   | rest:=rest-W[Nstep]
  |   | иначе
  |   | вывод нс, Nstep, "-й предмет не берем"
  | все
кц
кон

алг NewControl(арг цел  R, j, цел таб W[1:N], C[1:N], S[0:Q] рез
цел Control, NewS)
| Выбор условного оптимального управления из двух возможных:
|брать или не брать предмет, а также определение условного
|оптимального выигрыша
нач цел Syes, Sno
  | Sno:=S[R]
  | Syes:=S[R-W[j]] + C[j]
  | если Sno > Syes
  |   | то
  |   | Control:=0; NewS:=Sno
  |   | иначе
  |   | Control:=1; NewS:=Syes
  | все
кон
```

Язык Бейсик

```
'Задача 2
DECLARE SUB NewControl (Nstep%, rest%, Control%, NewS%)
DEFINT C-W
N = 6 'Количество предметов
Q = 45 'грузоподъемность машины
DIM SHARED W(1 TO N) 'Вес предметов
DATA 4, 7, 11, 12, 16, 27
FOR i = 1 TO N: READ W(i): NEXT i
DIM SHARED C(1 TO N) 'Стоимость предметов
DATA 7, 10, 15, 20, 27, 34
FOR i = 1 TO N: READ C(i): NEXT i
DIM SHARED S(0 TO Q), Snew(0 TO Q), Contr(0 TO Q, 1 TO N)
'Определяем условные оптимальные параметры
'Последний шаг
FOR rest = 0 TO Q 'Цикл возможных состояний
 IF W(N) > rest THEN 'Брать последний предмет нельзя
  Contr(rest, N) = 0: S(rest) = 0
 ELSE 'Берем последний предмет
  Contr(rest, N) = 1: S(rest) = C(N)
 END IF
NEXT rest         'Цикл возможных состояний
'Остальные шаги (кроме первого)
FOR Nstep = N - 1 TO 2 STEP -1 'Цикл шагов
 FOR rest = 0 TO Q 'Цикл возможных состояний
  IF W(Nstep) > rest THEN 'Брать предмет нельзя
   Contr(rest, Nstep) = 0: Snew(rest) = 0
  ELSE 'выясняем, нужно ли его брать
   CALL NewControl(Nstep, rest, Contr(rest, Nstep), Snew(rest))
  END IF
 NEXT rest         'Цикл возможных состояний
   'Переписываем Snew в S
   FOR rest = 0 TO Q:S(rest) = Snew(rest): NEXT rest
  NEXT Nstep                      'Цикл шагов
  'Первый шаг: Nstep=1. Возможное состояние одно - Q
  CALL NewControl(1, Q, Contr(Q, 1), MaxS)
  'Печатаем найденный оптимальный выигрыш
  PRINT "Максимальная суммарная стоимость: "; MaxS
'Определяем вариант управления, при котором он достигается
PRINT "Оптимальное управление"
rest = Q
FOR Nstep = 1 TO N
 IF Contr(rest, Nstep) = 1 THEN
  PRINT Nstep; "-й предмет берем"
```

Продолжение на с. 4

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Продолжение. Начало на с. 3

```
  rest = rest - W(Nstep)
 ELSE
  PRINT Nstep; "-й предмет не берем"
 END IF
NEXT Nstep
END

SUB NewControl (Nstep%, rest%, Control%, NewS%)
'Выбор условного оптимального управления из двух возможных:
'брать или не брать предмет, а также определение условного
'оптимального выигрыша
DIM Syes AS INTEGER, Sno AS INTEGER
Sno = S(rest%)
Syes = S(rest% - W(Nstep%)) + C(Nstep%)
IF Sno > Syes THEN
 Control% = 0: NewS% = Sno
ELSE
 Control% = 1: NewS% = Syes
END IF
END SUB
```

Язык Си

```
/*Задача 2*/
#include<stdio.h>
#define N 6 /*Количество предметов*/
/*Число возможных состояний=грузоподъемность машины + 1*/
#define State 46
int W[N]={4, 7, 11, 12, 16, 27};/*Вес предметов*/
int C[N]={7, 10, 15, 20, 27, 34};/*Стоимость предметов*/
int S[State],Snew[State], Contr[State][N];
void NewControl(int Nstep, int rest, int *Control, int *NewS)
/*Выбор условного оптимального управления из двух возможных:
  брать или не брать предмет, а также определение условного
  оптимального выигрыша                                    */
{int Syes,Sno;
 Sno=S[rest]; Syes=S[rest-W[Nstep]]+C[Nstep];
 if (Sno>Syes) {*Control=0; *NewS=Sno;}/*Не берем*/
 else {*Control=1; *NewS=Syes;}/*Берем*/
}
void main()
{int Q,Nstep,rest,MaxS;
 Q=State-1;/*Грузоподъемность машины*/
 /*Определяем условные оптимальные параметры
  Последний шаг                              */
 for (rest=0; rest<=Q; rest++)/*Цикл возможных состояний*/
  if (W[N-1]>rest)/*Брать последний предмет нельзя*/
  {Contr[rest][N-1]=0; S[rest]=0;}
  else/*Берем последний предмет*/
  {Contr[rest][N-1]=1; S[rest]=C[N-1];}
 /*Остальные шаги (кроме первого)*/
 for (Nstep=N-2; Nstep>=1; Nstep—)/*Цикл шагов*/
  {for (rest=0; rest<=Q; rest++)/*Цикл возможных состояний*/
    if (W[Nstep]>rest)/*Брать предмет нельзя*/
     {Contr[rest][Nstep]=0; Snew[rest]=0;}
    else/*выясняем, нужно ли его брать*/
     NewControl(Nstep,rest,&Contr[rest][Nstep],&Snew[rest]);
   /*Переписываем Snew в S*/
   for (rest=0; rest<=Q; rest++) S[rest]=Snew[rest];
  }
/*Первый шаг: Nstep=0. Возможное состояние одно - Q*/
NewControl(0,Q,&Contr[Q][0],&MaxS);
/*Печатаем найденный оптимальный выигрыш*/
printf("\nМаксимальная суммарная стоимость: %d",MaxS);
/*Определяем вариант управления, при котором он достигается*/
printf("\nОптимальное управление");
rest=Q;
for (Nstep=0; Nstep<N; Nstep++)
 if (Contr[rest][Nstep]==1)
  {printf("\n%d-й предмет берем",Nstep+1); rest-=W[Nstep];}
  else printf("\n%d-й предмет не берем",Nstep+1);
 }
```

Выполнив любую из этих программ, получим сообщение:

```
Максимальная суммарная стоимость: 69
Оптимальное управление
1-й предмет берем
2-й предмет не берем
3-й предмет берем
4-й предмет берем
5-й предмет берем
6-й предмет не берем
```

Предлагаем читателю убедиться, что любой другой набор предметов с суммарным весом не более 45 обладает суммарной стоимостью, не большей 69.

ЗАДАЧИ

Задача 3

Прежде всего отметим, что исходный запас средств Q, а также средства, вложенные в предприятия, являются целыми величинами. Тогда любое возможное состояние rest, которое (см. *табл.* 1, № 17/99) характеризуется объемом еще не вложенных средств, принимает целые значения в диапазоне от 0 до Q. Для каждого возможного состояния возможны несколько вариантов управления: вложение в то или иное предприятие 0, 1, 2, ..., rest единиц средств. Этим данная задача существенно отличается от предыдущих, где имелось лишь два варианта управления. С другой стороны, задача 3 довольно близка к предыдущей, второй задаче. Поэтому мы воспользуемся уже введенными обозначениями тех величин, которые являются общими для указанных задач: Contr[rest, Nstep] — управление в состоянии rest на шаге Nstep, S[rest] и Snew[rest] — условные оптимальные выигрыши в состоянии rest на шаге Nstep и (Nstep – 1). Новым понятием является функция дохода от вложения средств для данного предприятия. Поскольку вложенные средства принимают все целые значения от 0 до Q, то функции дохода – табличные. Объединим все эти таблицы в двумерный массив Prof[1 : N, 0 : Q].

Тогда Prof[Nstep, X] — доход от вложения X единиц средств в предприятие Nstep.

Напомним, что общий подход к решению подобных задач — последовательно заполнять таблицу 1, ее реализация для данной задачи приведена в таблице 3. Здесь принято: число предприятий N=5, исходный запас Q=10, матрица дохода Prof имеет вид

```
0.0, 0.5, 1.0, 1.4, 2.0, 2.5, 2.8, 3.0, 3.0, 3.0, 3.0
0.0, 0.1, 0.5, 1.2, 1.8, 2.5, 2.9, 3.5, 3.5, 3.5, 3.5
0.0, 0.6, 1.1, 1.2, 1.4, 1.6, 1.7, 1.8, 1.8, 1.8, 1.8
0.0, 0.3, 0.6, 1.3, 1.4, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5
0.0, 1.0, 1.2, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3
```

Таблица 3

Состояние	5-й шаг		4-й шаг		3-й шаг		2-й шаг		1-й шаг	
	УОУ	УОВ	УОУ	УОВ	УОУ	УОВ	УОУ	УОВ	УОУ	УОВ
0	0	0	0	0	0	0	0	0		
1	**(1)**	1	**(0)**	1	0	1	0	1		
2	2	1.2	1	1.3	1	1.6	0	1.6		
3	3	1.3	2	1.6	**(2)**	2.1	0	2.1		
4	4	1.3	3	2.3	2	2.4	0	2.4		
5	5	1.3	3	2.5	1	2.9	5	2.9		
6	6	1.3	4	2.6	2	3.4	5	3.5		
7	7	1.3	4	2.7	2	3.6	5	4.1		
8	8	1.3	5	2.8	2	3.7	5	4.6		
9	9	1.3	5	2.8	4	3.9	7	5.1		
10	10	1.3	5	2.8	5	4.1	**(7)**	5.6	**(0)**	5.6

Значения, соответствующие полученному оптимальному управлению, обведены в кружочки.

При составлении *табл.* 3 оптимальный вариант удобно выбирать с помощью *табл.* 2. Продемонстрируем ее использование на примере определения условного оптимального управления и условного оптимального выигрыша на четвертом (предпоследнем) шаге в состоянии, когда запас средств равен 7 (аналогом *табл.* 2 служит *табл.* 4):

Таблица 4

Управление	Выигрыш от этого управления на данном шаге	Состояние, в которое перейдет система в результате управления	Условный оптимальный выигрыш для состояния в графе 3	Критерий сравнения вариантов управления (сумма граф 2 и 4)
1	2	3	4	5
0	0	7	1.3	1.3
1	0.3	6	1.3	1.6
2	0.6	5	1.3	1.9
3	1.3	4	1.3	2.6
4	1.4	3	1.3	2.7
5	1.5	2	1.2	2.7
6	1.5	1	1.0	2.5
7	1.5	0	0	1.5

Естественно, что к этому моменту 5-й шаг уже оптимизирован, т.е. заполнены соответствующие столбцы табл. 1 (см. ее аналог — табл. 3). Из табл. 4 следует, что условное оптимальное управление в рассматриваемом состоянии равно 4 единицам средств, а соответствующий условный оптимальный выигрыш — 2.7. Эти значения записываются в табл. 3.

Переходим к систематическому изложению алгоритма. В нем существенную роль будет играть процедура

NewControl, (Nstep,rest, Control, NewS)

в которой определяется условное оптимальное управление Control и выигрыш NewS для данного шага Nstep и состояния rest. Требуется проверить все возможные значения управления от 0 до rest. Вначале положим:

```
Xopt := 0; Smax := Prof[Nstep,0]+S[rest]
```

Затем для значений управления X от 1 до rest находим:

```
Sx:=Prof[Nstep,X]+S[rest–X]
если Smax > Sx то Xopt := X; Smax := Sx
```

Осталось присвоить

```
Control := X; NewS := Smax
```

и описание процедуры завершено.

Как мы знаем, алгоритм начинается с последнего шага, где управление вынужденное: оставшийся запас средств надо целиком вложить в последнее предприятие. Другими словами, для всех значений состояния rest от 0 до Q надо выполнить:

```
Contr[rest, N] := rest ; S[rest] := Prof[N, rest]
```

Теперь приступаем к нахождению условных оптимальных управлений и выигрышей на всех промежуточных шагах от предпоследнего до второго. Пусть при некотором (Nstep + 1) они уже найдены. При этом условные оптимальные выигрыши хранятся в массиве S. Определим их для предыдущего шага Nstep, для чего выполняем обращение

```
NewControl(Nstep,rest,Contr[rest,Nstep],Snew[rest])
```

при всех значениях состояния rest от 0 до Q. Затем копируем массив Snew в S.

На первом шаге состояние единственно и равно Q. Поэтому, выполнив обращение

```
NewControl(1, Q,Contr[Q, 1], MaxS),
```

находим оптимальное управление для первого предприятия и максимальный доход MaxS.

Осталось определить оптимальные управления на следующих шагах. Для этого, зная состояние на предыдущем шаге и оптимальное управление, находим их разность. Состояние на очередном шаге равно этой разности. Теперь берем условное оптимальное управление для полученного состояния и очередного шага.

Прежде чем представлять программы, заметим, что с целью их упрощения все исходные данные (исходный запас ресурсов, количество предприятий и функции дохода для каждого предприятия) принимаются известными заранее, т.е. задаются как константы.

Школьный алгоритмический язык

```
цел Q, N | Глобальные параметры задачи
алг Задача 3
нач цел  Nstep, rest, вещ MaxS
  | Q:= 10 | Исходный запас средств
  | N:= 5   | Число предприятий
  | | Описываем массивы
  | вещ таб Prof[1: N,0: Q], S[0: Q], Snew[0: Q]
  | цел таб Contr[0: Q, 1: N]
  | |Заполняем массив функций доходов Prof
  | Prof[1,0]:=0.0; Prof[2,0]:=0.0; Prof[3,0]:=0.0;
  | Prof[4,0]:=0.0; Prof[5,0]:=0.0
  | Prof[1,1]:=0.5; Prof[2,1]:=0.1; Prof[3,1]:=0.6;
  | Prof[4,1]:=0.3; Prof[5,1]:=1.0
  | Prof[1,2]:=1.0; Prof[2,2]:=0.5; Prof[3,2]:=1.1;
  | Prof[4,2]:=0.6; Prof[5,2]:=1.2
  | Prof[1, 3]:=1.4; Prof[2,3]:=1.2; Prof[3,3]:=1.2;
  | Prof[4,3]:=1.3; Prof[5,3]:=1.3
  | Prof[1,4]:=2.0; Prof[2,4]:=1.8; Prof[3,4]:=1.4;
  | Prof[4,4]:=1.4; Prof[5,4]:=1.3
  | Prof[1,5]:=2.5; Prof[2,5]:=2.5; Prof[3,5]:=1.6;
  | Prof[4,5]:=1.5; Prof[5,5]:=1.3
  | Prof[1,6]:=2.8; Prof[2,6]:=2.9; Prof[3,6]:=1.7;
  | Prof[4,6]:=1.5; Prof[5,6]:=1.3
  | Prof[1,7]:=3.0; Prof[2,7]:=3.5; Prof[3,7]:=1.8;
  | Prof[4,7]:=1.5; Prof[5,7]:=1.3
  | Prof[1,8]:=3.0; Prof[2,8]:=3.5; Prof[3,8]:=1.8;
  | Prof[4,8]:=1.5; Prof[5,8]:=1.3
  | Prof[1,9]:=3.0; Prof[2,9]:=3.5; Prof[3,9]:=1.8;
  | Prof[4,9]:=1.5; Prof[5,9]:=1.3
  | Prof[1,10]:=3.0; Prof[2,10]:=3.5; Prof[3,10]:=1.8;
  | Prof[4,10]:=1.5; Prof[5,10]:=1.3
  | |Определяем условные оптимальные параметры
  | |Последний шаг. Вкладываем все, что осталось
  | нц для rest от 0 до Q
  | |    Contr[rest,N]:=rest
  | |    S[rest]:=Prof[N,rest]
  | кц
  | |Остальные шаги (кроме первого)
  | нц для  Nstep от N-1 до 2 шаг  -1
  | |    нц для rest от 0 до Q
  | |    | NewControl(Prof,S,Nstep,rest,Contr[rest,Nstep],Snew[rest])
  | |    кц
  | |    нц для  rest от 0 до Q
  | |    |    S[rest]:= Snew[rest]    |Переписываем Snew в S
  | |    кц
  | кц
  | |Первый шаг: Nstep=1. Возможное состояние одно - Q
  | NewControl(Prof,S,1,Q,Contr[Q,1],MaxS])
  | |Печатаем результат
  | вывод нс, "Максимальный доход: ", MaxS
  | вывод нс, "Для этого необходимо выделить"
  | rest:= Q
  | нц для Nsteo от 1 до N
  | |    вывод нс,  предприятию", Nstep, " - ", Contr[rest,Nstep]
  | |    rest:=rest - Contr[rest,Nstep]
  | кц
кон
```

где NewControl — процедура определения условного оптимального управления и условного оптимального выигрыша в состоянии rest на шаге Nstep (см. *табл.* 2):

```
алг NewControl(арг вещ таб Prof[1: N,0:Q],S[0:Q], цел  Nstep,rest,
рез цел Control, рез вещ  NewS)
нач цел X, Xopt   |X - возможное управление, Xopt - условное
  | оптимальное управление
  | вещ Sx, Smax
  | xopt:=0
  | Smax:=Prof[Nstep,0]+S[rest]
  | нц для X от 1 до rest
  | | Sx:=Prof[Nstep,X]+S[rest-X]
  | | если  Sx > Smax
  | | | то
  | | | Xopt:=X
  | | | Smax:=Sx
  | | все
  | кц
  | Control:=Xopt
  | NewS:= Smax
кон
```

Язык Паскаль

```
{Задача 3}
const
N=5;  {Количество предприятий}
Q=10; {Исходный запас средств}
{Функции дохода для каждого предприятия}
Prof:array[1..N,0..Q] of real=(
(0.0, 0.5, 1.0, 1.4, 2.0, 2.5, 2.8, 3.0, 3.0, 3.0, 3.0),
(0.0, 0.1, 0.5, 1.2, 1.8, 2.5, 2.9, 3.5, 3.5, 3.5, 3.5),
(0.0, 0.6, 1.1, 1.2, 1.4, 1.6, 1.7, 1.8, 1.8, 1.8, 1.8),
(0.0, 0.3, 0.6, 1.3, 1.4, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5),
(0.0, 1.0, 1.2, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3));
var
Nstep,rest:integer; MaxS:real;
S,Snew:array[0..Q] of real;
Contr:array[0..Q,1..N] of integer;
procedure NewControl(Nstep,rest:integer;
                     var Control:integer; var NewS:real);
{Определение условного оптимального управления Control и
 выигрыша NewS для данного шага Nstep и состояния rest}
var X,Xopt:integer; Sx,Smax:real;
begin
Xopt:=0; Smax:=Prof[Nstep,0]+S[rest];
for X:=1 to rest do begin
 Sx:=Prof[Nstep,X]+S[rest-X];
 if Sx>Smax then begin Xopt:=X; Smax:=Sx end
                     end;
Control:=Xopt; NewS:=Smax;
end;{NewControl}
BEGIN
{Последний шаг. Вкладываем все, что осталось}
for rest:=0 to Q do begin
 Contr[rest,N]:=rest; S[rest]:=Prof[N,rest];
                     end;
{Остальные шаги (кроме первого)}
for Nstep:=N-1 downto 2 do begin{Цикл шагов}
 for rest:=0 to Q do
 NewControl(Nstep,rest,Contr[rest,Nstep],Snew[rest]);
 S:=Snew;{Переписываем Snew в S}
                       end;{Цикл шагов}
{Первый шаг: Nstep=1. Возможное состояние одно - Q}
NewControl(1,Q,Contr[Q,1],MaxS);
{Печатаем результат}
writeln('Максимальный доход: ',MaxS:4:1);
writeln('Для этого необходимо выделить:');
rest:=Q;
for Nstep:=1 to N do begin
 writeln('предприятию ',Nstep,' - ',Contr[rest,Nstep]);
 rest:=rest-Contr[rest,Nstep];
                       end;
END.
```

Окончание на с. 6

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Окончание. Начало на с. 3—5

Язык Бейсик

```
'Задача 3
DECLARE SUB NewControl (Nstep%, rest%, Control%, NewS!)
DEFINT C,N,Q,R
DEFSNG M,P,S
N = 5  'Количество предприятий
Q = 10 'Исходный запас средств
'Функции дохода для каждого предприятия
DIM SHARED Prof(1 TO N,0 TO Q)
'Записываем массив Prof по строкам
DATA 0.0, 0.5, 1.0, 1.4, 2.0, 2.5, 2.8, 3.0, 3.0, 3.0, 3.0
DATA 0.0, 0.1, 0.5, 1.2, 1.8, 2.5, 2.9, 3.5, 3.5, 3.5, 3.5
DATA 0.0, 0.6, 1.1, 1.2, 1.4, 1.6, 1.7, 1.8, 1.8, 1.8, 1.8
DATA 0.0, 0.3, 0.6, 1.3, 1.4, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5
DATA 0.0, 1.0, 1.2, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3
FOR Nstep = 1 TO N: FOR rest = 0 TO Q
 READ Prof(Nstep,rest)
NEXT rest: NEXT Nstep
DIM SHARED S(0 TO Q), Snew(0 TO Q), Contr(0 TO Q, 1 TO N)
'Последний шаг. Вкладываем все, что осталось
FOR rest = 0 TO Q
 Contr(rest, N) = rest: S(rest) = Prof(N,rest)
NEXT rest
'Остальные шаги (кроме первого)
FOR Nstep = N - 1 TO 2 STEP -1 'Цикл шагов
 FOR rest = 0 TO Q
   CALL NewControl(Nstep, rest, Contr(rest, Nstep), Snew(rest))
 NEXT rest
 'Переписываем Snew в S
 FOR rest = 0 TO Q:S(rest) = Snew(rest): NEXT rest
NEXT Nstep                        'Цикл шагов
'Первый шаг: Nstep=1. Возможное состояние одно - Q
CALL NewControl(1, Q, Contr(Q, 1), MaxS)
'Печатаем результат
PRINT "Максимальный доход: "; MaxS
PRINT "Для этого необходимо выделить:"
rest = Q
FOR Nstep = 1 TO N
 PRINT "предприятию ";Nstep; " - "; Contr(rest,Nstep)
 rest = rest - Contr(rest,Nstep)
NEXT Nstep

SUB NewControl (Nstep%, rest%, Control%, NewS!)
'Определение условного оптимального управления Control и
'выигрыша NewS для данного шага Nstep и состояния rest
DIM X AS INTEGER, Xopt AS INTEGER
DIM Sx AS SINGLE, Smax AS SINGLE
Xopt = 0: Smax = Prof(Nstep%,0) + S(rest%)
FOR X = 1 TO rest%
 Sx = Prof(Nstep%,X) + S(rest% - X)
 if Sx > Smax THEN Xopt = X: Smax = Sx
NEXT X
Control% = Xopt: NewS! =Smax
END SUB
```

Язык Си

```
/*Задача 3*/
#include<stdio.h>
#define N 5 /*Количество предприятий*/
/*Число возможных состояний=Исходный запас средств+1*/
#define State 11
/*Функции дохода для каждого предприятия*/
float Prof[N][State]={
{0.0, 0.5, 1.0, 1.4, 2.0, 2.5, 2.8, 3.0, 3.0, 3.0, 3.0},
{0.0, 0.1, 0.5, 1.2, 1.8, 2.5, 2.9, 3.5, 3.5, 3.5, 3.5},
{0.0, 0.6, 1.1, 1.2, 1.4, 1.6, 1.7, 1.8, 1.8, 1.8, 1.8},
{0.0, 0.3, 0.6, 1.3, 1.4, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5},
     {0.0, 1.0, 1.2, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3}};
 float S[State],Snew[State]; int Contr[State][N];
 void NewControl(int Nstep,int rest,int *Control, float
  *NewS)
  /*Определение условного оптимального управления Control и
    выигрыша NewS для данного шага Nstep и состояния rest*/
 {int X,Xopt; float Sx,Smax;
 Xopt=0; Smax=Prof[Nstep][0]+S[rest];
 for (X=1; X<=rest; X++)
  {Sx=Prof[Nstep][X]+S[rest-X];
   if (Sx>Smax) {Xopt=X; Smax=Sx;}
  }
 *Control=Xopt; *NewS=Smax;
}
void main()
{int Q,Nstep,rest; float MaxS;
 Q=State-1;/*Исходный запас средств*/
 /*Последний шаг. Вкладываем все, что осталось*/
 for (rest=0; rest<=Q; rest++)
 {Contr[rest][N-1]=rest; S[rest]=Prof[N-1][rest];}
 /*Остальные шаги (кроме первого)*/
 for (Nstep=N-2; Nstep>0; Nstep—)/*Цикл шагов*/
  {for (rest=0; rest<=Q; rest++)
   NewControl(Nstep,rest,&Contr[rest][Nstep],&Snew[rest]);
   /*Переписываем Snew в S*/
   for (rest=0; rest<=Q; rest++) S[rest]=Snew[rest];
  }                                /*Цикл шагов*/
 /*Первый шаг: Nstep=0. Возможное состояние одно - Q*/
 NewControl(0,Q,&Contr[Q][0],&MaxS);
 /*Печатаем результат*/
 printf("\nМаксимальный доход: %.1f",MaxS);
 printf("\nДля этого необходимо выделить:");
 rest=Q;
 for (Nstep=0; Nstep<N; Nstep++)
  {printf("\nпредприятию %d - %d",Nstep+1,Contr[rest][Nstep]);
   rest-=Contr[rest][Nstep];
  }
}
```

Выполнив любую из этих программ, получим сообщение:

```
Максимальный доход: 5.6
Для этого необходимо выделить:
предприятию 1 — 0
предприятию 2 — 7
предприятию 3 — 2
предприятию 4 — 0
предприятию 5 — 1
```

Видно, что полученный результат согласуется с *табл.* 3.

В заключение заметим следующее. Конечно, в статье рассмотрены простейшие задачи динамического программирования. Однако и они дают понятие об общей идее метода: пошаговая оптимизация, проводимая в одном направлении (от конца к началу) “условно”, а затем в другом (от начала к концу) — “безусловно”. При этом при выборе условных оптимальных управлений следует соблюдать следующее правило:

Каково бы ни было состояние системы перед очередным шагом, надо выбирать управление на этом шаге так, чтобы выигрыш на данном шаге плюс оптимальный выигрыш на всех последующих шагах был максимальным.

Это правило является общим принципом, лежащим в основе решения всех задач динамического программирования (его часто называют “принципом оптимальности”*):

Фундаментальное изложение метода динамического программирования представлено в [1, 8—10].

ЛИТЕРАТУРА

1. *Вентцель Е.С.* Исследование операций: задачи, принципы, методология. М.: Наука, 1988.
2. Играем с мозаикой списков и слов на Лого. Сборник задач /Составители *Горлицкая С.И., Кузнецова И.Н.* Информатика № 6, 1997.
3. *Кушниренко А.Г., Лебедев Г.В., Сворень Р.А.* Основы информатики и вычислительной техники. М.: Просвещение, 1990.
4. *Эпиктетов М.Г.* Почему школьный алгоритмический? Информатика № 24, 1995.
5. *Зельднер Г.А.* QuickBasik 4.5. М.: ABF, 1994.
6. *Фаронов В.В.* Программирование на персональных ЭВМ в среде Турбо Паскаль. М.: Изд-во МГТУ, 1992.
7. *Ребрина В.А., Попик А.* Динамическое программирование и его применение к решению некоторых задач. Информатика № 13, 1998.
8. *Беллман Р.* Динамическое программирование. М.: Иностранная литература, 1960.
9. *Таха Х.* Введение в исследование операций. М.: Мир, 1985.
10. *Вентцель Е.С.* Элементы динамического программирования. М.: Наука, 1964.
11. Кузницкий Е.М. О динамическом программировании. Информатика № 15, 1996.

* Существуют и другие формулировки принципа оптимальности. Приведем две из них:
1) “любой подпуть оптимального пути является оптимальным” [9];
2) “подстратегия оптимальной стратегии должна быть оптимальной” [11].

ЗАДАЧИ

Выше были приведены примеры чисел, записанных в троичной уравновешенной системе счисления, и каждая из них начиналась цифрой 1. Можно привести примеры чисел, начинающихся с цифры $\bar{1}$:

$\bar{1}\ 1\ \bar{1}\ 0\ 1_3 = -1\cdot 3^4 + 1\cdot 3^3 - 1\cdot 3^2 + 0\cdot 3 + 1\cdot 3^0 = -64_{10}$.

$\bar{1}\ \bar{1}\ 1\ 1\ 1_3 = -1\cdot 3^4 - 1\cdot 3^3 + 1\cdot 3^2 + 0\cdot 3 + 1\cdot 3^0 = -95_{10}$.

Ясно, что всякое число, начинающееся цифрой $\bar{1}$, есть отрицательная величина, а число, начинающееся цифрой 1, — всегда положительная величина.

На эту особенность уравновешенной системы следует обратить внимание. Мы впервые встречаемся с тем, что для обозначения отрицательной величины к числу не требуется присоединять дополнительный знак, а стало быть, при выполнении арифметических операций над числами не требуется анализировать знаки операндов.

В книге В.Н. Касаткина "Новое о системах счисления" приводятся правила выполнения арифметических действий, выписаны таблицы сложения и умножения в уравновешенной троичной системе счисления. В этой же книге даны алгоритмы перевода чисел из троичной уравновешенной системы счисления в традиционную троичную систему и обратно.

Главная особенность уравновешенных систем счисления — перед отрицательными числами не ставится знак "минус" и при выполнении арифметических операций не используется "правило знаков". И это очень привлекательно при конструировании ЭВМ.

В Советском Союзе в 1958 году была построена экспериментальная модель ЭВМ, арифметика которой базировалась на троичной уравновешенной системе счисления. Инициатором разработки этой уникальной машины была группа математиков вычислительного центра Московского Государственного университета имени М.В. Ломоносова во главе с академиком С.Л. Соболевым и при участии Н.П. Брусенцова и С.П. Маслова.

В 1962—1965 годах было выпущено более 50 промышленных экземпляров ЭВМ "Сетунь", работающих в троичной уравновешенной системе. Особенности этой машины до сих пор привлекают внимание ученых и конструкторов. Главный конструктор ЭВМ "Сетунь" Н.П. Брусенцов пишет, что в этом компьютере были реализованы далеко не все полезные свойства трехзначного кода и трехзначной логики. Кроме того, в нем не осуществлялись операции над числами с плавающей запятой, для которых особенно заметны преимущества троичного кода. Несмотря на это, "Сетунь" была значительно дешевле двоичных машин ее класса и превосходила их по быстродействию, демонстрируя выгоды своей системы счисления.

4.5. Контрольные вопросы и упражнения

1. Почему двоичная и двоично-шестнадцатеричная формы записи совпадают, а двоичная и двоично-десятичная — нет?
2. Переведите десятичные числа 645, 383 в двоичную, восьмеричную и шестнадцатеричную системы счисления и заполните следующую таблицу:

	Десятичные	Шестнадцатеричные	Восьмеричные	Двоичные
a	645			
b	383			
$a+b$				
$a-b$				

3. Переведите число $1234{,}5678_9$ в 27-ричную систему счисления, а число $ABCD{,}EF_{16}$ — в восьмеричную. Являются ли указанные пары систем счисления смешанными?
4. Представьте десятичные числа в двоично-десятичной системе:

   34590; 27153; 11911; 101079.

5. Представьте двоично-десятичные числа в десятичной системе счисления:

   0011 0101 1000 0000 1001;
   1001 1000 0001 1000 0111.

6. Любую ли комбинацию из нулей и единиц можно трактовать как двоично-десятичное число?
7. Во сколько раз сократится количество цифр в записи числа, если его перевести из четверичной системы счисления в 64-ричную? А из десятичной в 10 000-ричную?
8. Выпишите четверичное представление для всех цифр алфавита шестнадцатеричной системы счисления и переведите число $12345678{,}9ABCDEF_{16}$ непосредственно в четверичную систему, не используя его двоичное представление в качестве промежуточного.
9. Почему в фибоначчиевой системе счисления можно использовать только две цифры?
10. Определите, начиная с какого десятичного числа представление в факториальной системе окажется короче десятичного?

Системы счисления и компьютерная арифметика

Е.В. АНДРЕЕВА, И.Н. ФАЛИНА

Продолжение. Начало см. в № 14, 15, 16, 17/99

3.5. Перевод конечной десятичной дроби в *P*-ичную

Если у дроби есть ненулевая целая часть, то она переводится из десятичной системы в *P*-ичную отдельно (см. п. 3.4). Сформулируем правила перевода дробной части.

Дана правильная конечная десятичная дробь b. Допустим, что в *P*-ичной системе наша дробь b имеет вид $b=0, b_{-1}b_{-2}\ldots b_{-k}\ldots$ (в *P*-ичной системе дробь может оказаться и бесконечной). Необходимо найти цифры b_{-1}, b_{-2}, ..., b_{-k}, ... Приравняем исходную десятичную дробь b к ее развернутой форме (1.8) в *P*-ичной системе счисления:

$$b=b_{-1}P^{-1}+b_{-2}P^{-2}+\ldots+b_{-k}P^{-k}+\ldots \qquad (3.4)$$

Умножим левую (само число) и правую части выражения (3.4) на P. В правой части получим:

$$b_{-1}+b_{-2}P^{-1}+\ldots+b_{-k}P^{-k+1}\ldots, \qquad (3.5)$$

— значит, первая цифра дробной части числа b в *P*-ичной системе — b_{-1} ($0\le b_{-1}<P$, справедливость этих неравенств следует из того, что поскольку $0\le b<1$, то и $0\le b\cdot P<P$) равна целой части результата умножения десятичной дроби b на P.

Дробную часть результата умножения снова обозначим через b, то есть $b=b_{-2}P^{-1}+\ldots+b_{-k}P^{-k+1}\ldots$, и опять умножим полученное равенство на P. В результате справа получим $b_{-2}+b_{-3}P^{-1}+\ldots+b_{-k}P^{-k+2}\ldots$, и целая часть результата в левой части равна b_{-2} — второй искомой цифре.

Этот процесс необходимо продолжать до тех пор, пока дробная часть результата умножения левой части не будет равна нулю или не будет выделен период из повторяющихся цифр b_{-i}. Иногда процесс можно прервать раньше, когда уже достигнута необходимая ***точность вычислений***.

Почему возникает вопрос о точности вычислений? Просто потому, что иногда можно вполне ограничиться необходимым количеством уже полученных значащих цифр при достижении необходимой точности, до выделения периода (именно так поступают при представлении двоичных дробей в компьютере, подробнее об этом можно прочитать в главе 7).

Напомним: раньше уже доказывалось, что при переводе конечной и даже периодической десятичной дроби в *P*-ичную систему всегда получится либо конечная, либо периодическая *P*-ичная дробь.

Сформулируем описанные выше правила перевода десятичных дробей в *P*-ичную систему в виде алгоритма.

Алгоритм перевода правильной конечной десятичной дроби в *P*-ичную систему счисления:

1) *умножим исходное число на P (основание новой системы счисления). Целая часть полученного произведения является первой цифрой после запятой в искомом числе (целая часть может быть как равна нулю, так и быть больше девяти, но она всегда меньше, чем P, это позволяет записать ее в виде ровно одной цифры P-ичной системы счисления);*
2) *дробную часть произведения снова умножим на P, целую часть полученного числа заменяем на цифру в P-ичной системе и приписываем ее справа к результату;*
3) *выполняем пункт 2 до тех пор, пока дробная часть произведения не станет равной нулю или не выделится период (дробная часть окажется равной уже получавшейся ранее дробной части произведения).*

Пример 3.18. Переведем число 0,375 в двоичную систему счисления.

0,375 · 2=0,75 **0** — первая цифра результата
0,75 · 2=1,5 **1** — вторая цифра результата
0,5 · 2=1,0 **1** — последняя цифра результата

Ответ: $0{,}375=0{,}011_2$.

Пример 3.19. Переведем число 0,515625 в четверичную систему счисления.

0,515625 · 4=2,0625 **2** — первая цифра результата
0,0625 · 4=0,25 **0** — вторая цифра результата
0,25 · 4=1,0 **1** — последняя цифра результата

Ответ: $0{,}53125=0{,}201_4$.

Пример 3.20. Переведем число 0,109375 в шестнадцатеричную систему.

0,109375 · 16= 1,75	**1** — первая цифра результата
0,75 · 16=12,0	$C_{16}=12_{10}$ является последней цифрой искомой дроби

Ответ: $0,109375=0,1C_{16}$.

Пример 3.21. Переведем в пятеричную систему счисления число 0,123.

0,123 · 5=0,615	**0**
0,615 · 5=3,075	**3**
0,075 · 5=0,375	**0**
0,375 · 5=1,875	**1**
0,875 · 5=4,375	**4**

Дробная часть последнего произведения равна уже встречавшейся ранее дробной части, следовательно, последние две цифры образуют период пятеричной дроби.

Ответ: $0,123=0,030(14)_5$.

Пример 3.22 (*Московский Государственный университет экономики, статистики и информатики, 1996 г.*).

Десятичное число 20,45 перевели в четверичную систему счисления. Найти 1999-ю цифру после запятой.

Решение. Поскольку надо найти 1999-ю цифру после запятой, достаточно перевести в четверичную систему счисления дробную часть, то есть число 0,45. Имеем:

0,45 · 4=1,8
0,8 · 4=3,2
0,2 · 4=0,8
0,8 · 4=3,2 (дробная часть совпала с уже встречавшейся ранее).

Получили бесконечную дробь с периодом (30) и непериодической частью, равной 1.

Таким образом, $0,45_{10}=0,1(30)_4$.

Найдем теперь 1999-ю цифру этого числа. Первая цифра после запятой — единица; остаются еще 1998 цифр, находящихся в периодической части. Число 1998 четное, т.е. последовательность из двух цифр (30) повторится целое число раз. Поэтому 1999-й цифрой будет 0.

Ответ: 1999-я цифра — ноль.

3.6. Перевод бесконечной периодической десятичной дроби в *P*-ичную

Дана бесконечная периодическая десятичная дробь. Требуется перевести ее в *P*-ичную систему счисления.

Для решения задачи воспользуемся алгоритмом, приведенным в параграфе 3.5. Но для получения правильного результата необходимо уметь умножать период дроби на произвольное число.

Правило умножения периода правильной дроби на произвольное число:

1) *если при умножении периода дроби на некоторое число количество цифр в произведении равно количеству цифр в периоде исходного числа, то период результата равен полученному произведению. Далее переходим к (6). Если же при умножении периода количество цифр результата превышает количество цифр периода, переходим к (2);*
2) *"лишними" цифрами будем считать n—k первых слева цифр результата, где n — количество цифр в результате, k — количество цифр в периоде исходной дроби;*
3) *сложим число, образованное "лишними" цифрами, с числом, образованным правыми k цифрами промежуточного результата;*
4) *если количество цифр в получившемся результате сложения больше, чем k, то процесс следует повторить с (2);*
5) *если количество цифр результата сложения стало равным количеству цифр периода исходной дроби, то период произведения равен последнему результату суммирования;*
6) *непериодическая (целая — для чисто периодических дробей) часть результата равна сумме чисел, образованных из "лишних" цифр каждого этапа.*

Если же у исходной дроби изначально была своя непериодическая часть, то умножить также следует и ее, а затем сложить с результатом умножения периода.

Пример 3.23. Выполним умножение 0,(09) · 8.

Так как в периоде исходной дроби две цифры и в произведении периода (09) на 8 также две цифры, то период результата равен получившемуся произведению.

Ответ: 0,(09) · 8=0,(72).

Пример 3.24. Выполним умножение 0,(7) · 16.

Умножим период дроби на 16: 7 · 16=112. В периоде дроби одна цифра, а в произведении — три цифры. Следовательно, "лишними" являются две левые цифры, образующие число 11. Прибавив 11 к 2, получим 13, что опять превышает исходную длину периода на одну цифру. Вновь сложим "лишнее" число 1 с правой цифрой результата: 1+3=4. Количество цифр результата равно количеству цифр в периоде исходной дроби, следовательно, период произведения равен 4. Целая же часть результата умножения равна 11+1=12 (в процессе ее формирования участвуют только "лишние" цифры).

Ответ: 0,(7) · 16=12,(4).

Пример 3.25. Выполним умножение 0,7(6) · 12.

Сначала умножим на 12 периодическую часть 6 · 12=72. В получившемся произведении две цифры, а в исходном периоде — одна. Следовательно, "лишнее" число равно 7, результат сложения равен 2+7=9. В получившемся числе количество цифр равно количеству цифр в периоде исходной дроби, т.е. период произведения равен 9. Непериодическая часть произведения равна единственному "лишнему" числу — 7. Таким образом, 0,0(6) · 12=0,7(9)=0,8 (здесь использована эквивалентность двух форм записи конечных дробей).

N-разрядное число, "списанное со счетов", оказывается представленным не в виде суммы степеней основания *P*, а является суммой факториалов *n* первых натуральных чисел.

Пример 4.5.

$3221_ф = 3 \cdot 4! + 2 \cdot 3! + 2 \cdot 2! + 1 \cdot 1! = 89_{10}$;

$40301_ф = 4 \cdot 5! + 3 \cdot 3! + 1 \cdot 1! = 499_{10}$.

Эту систему счисления относят к нетрадиционным позиционным и называют ***факториальной системой счисления.***

Алгоритм перевода из десятичной системы счисления в факториальную очень прост. Он аналогичен алгоритму перевода из десятичной системы в *P*-ичную путем деления на основание системы *P*. Отличие в том, что первый раз исходное десятичное число делим на 2, первое частное — на 3, второе частное — на 4 и т.д.

С числами этой системы можно выполнять арифметические действия по правилам, незначительно отличающимся от правил десятичной арифметики.

Аналогично другим системам счисления в факториальной системе можно рассматривать дробные числа.

Каких-либо существенных практических применений этой системы, основанных на необычной сущности цифрового и многочленного представления, по-видимому, нет. Рассмотрение этой системы прежде всего полезно как методический подход в расширении представлений о системах счисления и обобщении принципа позиционности.

К нетрадиционным системам счисления относят и ***фибоначчиеву систему счисления***. Базисом фибоначчиевой системы является последовательность 1, 2, 3, 5, 8, 13, 21, 34, 55, ..., т.е. подряд идущие числа Фибоначчи. В качестве цифр в этой системе счисления используются только цифры 0 и 1.

Приведем примеры чисел, записанных в фибоначчиевой системе счисления:

$37_{10}=34 + 3 = 100000100_ф$;

$25_{10}=21 + 3 + 1 = 1000101_ф$.

Интересно, что в записи чисел не будут стоять две единицы подряд. Предоставим читателю самому разобраться с этим фактом.

Заметим, что системы, аналогичные фибоначчиевой, применяются при шифровании информации. Это практически и теоретически интересные системы записи чисел. Изучение особенностей таких систем продолжается и в наше время.

4.4. Уравновешенная система счисления

В параграфе 4.3 в качестве базиса позиционной системы счисления была взята последовательность чисел, не являющаяся геометрической прогрессией. Цифре дано определение в 1-й части, и другого смысла у нее быть не может. Каждая цифра тем не менее являлась положительным натуральным числом. Но в самом определении цифры (см.: Часть I. Тема 2) нигде не сказано, что она всегда и везде должна быть таковой.

Рассмотрим широко известную задачу "О взвешивании".

Задача. Найти набор из четырех гирь такой, что с их помощью на чашечных весах можно взвесить любой груз массой от 1 до 40 кг включительно. При необходимости гири можно располагать на обеих чашках весов.

Ответ: искомый набор состоит из гирь в 27, 9, 3 и 1 кг.

Запись о взвешивании одного килограмма может быть такой: 0 0 0 1.

Взвешивание двух килограммов требует использования двух гирь: на пустую чашку весов помещаем гирю в 3 кг, а на чашку с грузом — в 1 кг. Результат такого взвешивания можно записать в виде: $0\ 0\ 1\ \bar{1}$.

Взвешивание четырех килограммов можно отразить записью 0 0 1 1.

Более сложно выражается взвешивание груза в 5 кг: $0\ 1\ \bar{1}\ \bar{1}$. Эта запись означает, что на пустую чашку помещена гиря, масса которой равна единице третьего разряда, то есть 9, а на чашку с грузом помещены гири в 1 и 3 кг.

Из приведенных записей ясно, что если над цифрой того или иного разряда стоит черточка, то это означает, что гиря соответствующей массы помещена на чашку с грузом и ее масса вычитается из общей массы. Иначе говоря, цифра $\bar{1}$ есть отрицательная единица. Действительно:

$0\ 1\ \bar{1}\ \bar{1} = 0 \cdot 3^3 + 1 \cdot 3^2 - 1 \cdot 3 - 1 \cdot 3^0 = 5_{10}$.

Приведем несколько других записей результатов взвешивания:

$0\ 1\ \bar{1}\ 0 = 0 \cdot 3^3 + 1 \cdot 3^2 - 1 \cdot 3 + 0 \cdot 3^0 = 6_{10}$.

$0\ 1\ \bar{1}\ 1 = 0 \cdot 3^3 + 1 \cdot 3^2 - 1 \cdot 3 + 1 \cdot 3^0 = 7_{10}$.

$0\ 1\ 0\ \bar{1} = 0 \cdot 3^3 + 1 \cdot 3^2 + 1 \cdot 3 - 1 \cdot 3^0 = 8_{10}$.

Из записей следует, что результат любого взвешивания на чашечных весах выражается числом, записанным в системе счисления с основанием $P=3$ и с алфавитом, состоящим из $\bar{1}$, 0 и 1.

Определение. ***Уравновешенной троичной системой счисления, или троичной системой с симметричным основанием***, называется система с основанием $P=3$ и цифрами $\bar{1}$, 0, 1, где цифра $\bar{1}$ означает "минус единицу".

4.2. Применение теоремы о смешанных системах счисления

Если системы с основаниями P и Q являются смешанными, то, как мы только что убедились, перевод чисел из одной такой системы счисления в другую осуществляется чрезвычайно просто. Он особенно упрощается в обе стороны, если известно представление каждой цифры Q-ичной системы в P-ичной (здесь $Q>P$).

Одно из практических применений теоремы о смешанных системах счисления состоит в том, что *арифметические действия* над числами, записанными в любой системе счисления, можно *выполнять в системе, смешанной с исходной, если последняя по каким-либо причинам более удобна.*

Например, вычисления в 100-ичной системе заменяются на десятичную арифметику (100-ичные числа переводятся в десятичную систему, а результат, при необходимости, может быть снова записан в 100-ичной), а действия с шестнадцатеричными или восьмеричными числами легко заменяются на двоичную арифметику (что активно используется в компьютерной арифметике).

Данную теорему можно также использовать *для сокращенной записи чисел* путем замены системы счисления с меньшим основанием на систему с большим, но таким, чтобы эти системы являлись смешанными. Заметим, что это всегда возможно. Так, если имеется запись числа в P-ичной системе, то можно переписать это же число в систему с основанием $Q=P^m$, уменьшив количество цифр в m раз. Например, при использовании двоичной системы счисления сами числа можно представлять в 256-ричной, сократив количество цифр в записи числа в 8 раз ($256=2^8$).

Но и на этом применение теоремы не исчерпывается. Теорема о смешанных системах счисления может иногда *сделать более рациональным решение задачи перевода чисел из одной системы в другую*, даже если они непосредственно не являются смешанными. В примере 4.3 при переводе чисел из восьмеричной системы в шестнадцатеричную оказалось удобным сначала переписать число в двоичном виде (двоичная система является смешанной как с восьмеричной, так и с шестнадцатеричной). Следующий пример демонстрирует подобную процедуру уже для дробных чисел:

Пример 4.4. Переведем число $BF3{,}6_{16}$ в восьмеричную систему счисления:
$BF3{,}6_{16}=1011|1111|0011{,}0110_2=101|111|110|011{,}011_2=5763{,}3_8$.

Иногда бывает необходимо перевести число из десятичной системы счисления сразу в двоичную, восьмеричную и шестнадцатеричную.

Весьма разумно сначала определить, перевод в какую из перечисленных систем является для вас наиболее простым и удобным. С одной стороны, перевод в шестнадцатеричную систему путем последовательного деления на 16 выполняется меньшим числом действий, а следовательно, вероятность сделать ошибку уменьшается, однако операцию деления на 16 очень уж простой не назовешь. С другой стороны, при переводе в двоичную систему могут применяться и действия, отличные от деления на 2 с остатком, например, выделение максимальной степени двойки. Вполне возможно, что кому-то наиболее простым покажется перевод в восьмеричную систему.

Если вы получили шестнадцатеричное представление исходного числа, то, переписав его в двоичной системе, затем так же легко сможете представить его и в восьмеричной. В случае, когда первичным является двоичное представление, шестнадцатеричная и восьмеричная формы записи получаются из него непосредственно.

Однако наряду с удобством подобный подход имеет и "подводные камни": если ошибка будет сделана при переводе исходного числа в наиболее подходящую в данный момент систему, то она будет растиражирована и для двух других систем счисления, смешанных с первой.

4.3. Факториальная и фибоначчиева системы счисления

В рассмотренных выше системах счисления "вес" единицы любого разряда, кроме первого, всегда равнялся "весу" единицы предшествующего разряда, умноженному на постоянное основание системы P. Можно, однако, представить такой вариант системы счисления, в которой смысл понятия "основание системы счисления" заметно отличается от традиционного. Предлагаемая система является следствием расширения нашего представления о роли основания системы счисления. Существо нового подхода легко представить, если рассмотреть счеты необычной конструкции (*рис.* 1). Они приведены в книге В.Н. Касаткина "Новое о системах счисления".

На нижней проволоке счетов, отведенной для единиц младшего разряда, вес каждой из которых равен единице, помещены две косточки. На следующей проволоке помещены три косточки, на третьей — четыре и т.д., на n-й проволоке — $n+1$ косточка. Так как каждая косточка на второй проволоке заменяет две косточки, расположенные на первой проволоке, то вес ее равен 2. Каждая косточка третьей проволоки заменяет три косточки второй проволоки, и, следовательно, ее вес в $6=3\cdot 2\cdot 1$ раз больше веса косточки на первой проволоке. Это означает, что косточка, расположенная на *n-й проволоке, имеет вес n*! Вес единиц от разряда к разряду растет, но растет неравномерно. Это приводит к представлению числа в виде следующего многочлена:

$$a_n a_{n-1} a_{n-2} \ldots a_2 a_1 = a_n \cdot n! + a_{n-1} \cdot (n-1)! + a_{n-2} \cdot (n-2)! + \ldots + a_2 \cdot 2! + a_1 \cdot 1!$$

Затем умножим на 12 непериодическую часть исходной дроби: $0{,}7\cdot 12=8{,}4$.

Ответ: $0{,}7(6)\cdot 12 = 8{,}4+0{,}8= 9{,}2$.

Теперь перевод в P-ичную систему счисления даже периодических дробей можно осуществлять по правилам параграфа 3.5.

Далее приводятся примеры перевода периодических дробей из десятичной системы счисления в P-ичную систему:

Пример 3.26. $0{,}7(6)$ переведем в двенадцатеричную систему счисления:

$0{,}7(6)\cdot 12=\mathbf{9}{,}2$ (первая цифра результата — 9);
$0{,}2\cdot 12=\mathbf{2}{,}4$ (вторая цифра результата — 2);
$0{,}4\cdot 12=\mathbf{4}{,}8$ (третья цифра результата — 4);
$0{,}8\cdot 12=\mathbf{9}{,}6$ (четвертая цифра результата — 9);
$0{,}6\cdot 12=\mathbf{7}{,}2$ (цифрой 7 заканчивается период результата, который начинается со второй его цифры);
$0{,}2\cdot 12=\mathbf{2}{,}4$ (вторая строка совпадает с шестой строкой, т.е. получившаяся дробь является периодической двенадцатеричной дробью).

Ответ: $0{,}7(6)_{10}=0{,}9(2497)_{12}$.

Пример 3.27. $0{,}(7)$ переведем в шестнадцатеричную систему:

$0{,}(7)\cdot 16=\mathbf{12}{,}(4)$ (первая цифра результата — C);
$0{,}(4)\cdot 16=\mathbf{7}{,}(1)$ (вторая цифра результата — 7);
$0{,}(1)\cdot 16=\mathbf{1}{,}(7)$ (третья цифра результата — 1, она же является последней цифрой периода).

Ответ: $0{,}(7)_{10}=0{,}(C71)_{16}$.

Второй способ перевода состоит в том, что любую периодическую дробь можно представить в виде обыкновенной, затем целочисленные числитель и знаменатель перевести в P-ичную систему и уже в ней вновь преобразовать обыкновенную дробь в P-ичную, организовав деление столбиком так, как это показано в параграфе 2.3.

Пример 3.28. $0{,}(3)$ переведем в двоичную систему:

$$0{,}(3)=\frac{1}{3}=\frac{1}{11_2}=0{,}(01)_2.$$

3.7. Перевод чисел из *P*-ичной системы в *Q*-ичную

Обычно сначала число переводят из P-ичной системы в десятичную, используя правила параграфов 3.1—3.3, затем из десятичной системы переводят в Q-ичную систему, по правилам параграфов 3.4—3.6. При таком подходе все вычисления можно производить в привычной для нас десятичной системе счисления.

Для того чтобы сделать перевод непосредственно, минуя десятичную систему, необходимо выписать таблицы умножения и сложения в Q-ичной или в P-ичной системе счисления и проделать действия, аналогичные алгоритмам параграфов 3.1—3.3 или 3.4—3.6 соответственно, только производя все вычисления согласно выписанным таблицам в Q-ичной или, соответственно, в P-ичной системе.

Пример 3.29. $20{,}1_3$ переведем в двоичную систему, производя все арифметические действия в двоичной системе счисления:

$$20{,}1_3=2\cdot 3+\frac{1}{3}=10_2\cdot 11_2+\frac{1}{11_2}=110{,}(01)_2.$$

3.8. Задачи и решения

1. Переведите в десятичную систему счисления числа, записанные в пятеричной системе счисления:
1_5; 301_5; 121234_5.

Решение.
1) Очевидно, что $1_{10}=1_5$;
2) $301_5=3\cdot 5^2+0\cdot 5+1=76_{10}$;
3) $121234_5=1\cdot 5^5+2\cdot 5^4+1\cdot 5^3+2\cdot 5^2+3\cdot 5+4$.
Воспользуемся схемой Горнера для вычисления этого выражения (см. п. 3.1).
$1\cdot 5^5+2\cdot 5^4+1\cdot 5^3+2\cdot 5^2+3\cdot 5+4=$
$=(((((1\cdot 5+2)\cdot 5+1)\cdot 5)+2)\cdot 5+3)\cdot 5+4=$
$=((((7\cdot 5+1)\cdot 5)+2)\cdot 5+3)\cdot 5+4=$
$=(((36\cdot 5)+2)\cdot 5+3)\cdot 5+4=$
$=(182\cdot 5+3)\cdot 5+4=913\cdot 5+4=4569$.
Следовательно, $121234_5=4569_{10}$.

2. Школьный калькулятор работает в троичной системе счисления и для вывода числа на экране имеет только четыре знакоместа. С каким самым большим десятичным числом, переведенным, конечно, в троичную систему счисления, мы можем работать?

Решение. В троичной системе счисления используются цифры 0, 1, 2. Самое большое число, которое можно высветить на экране калькулятора, — 2222. По алгоритму, описанному в п. 3.1, переведем это троичное число в десятичную систему счисления, получим ответ 80.

3. Требуется выбрать 5 различных гирь так, чтобы с их помощью можно было взвесить любой груз до 30 кг включительно при условии, что гири ставятся только на одну чашу весов. (Эта задача приведена в книге знаменитого математика XIII века Леонардо Пизанского. Этой же задачей интересовался Л.Эйлер.)

Решение. Очевидно, что сумма всех гирь должна быть не меньше 30 кг. Но этого, конечно, недостаточно. Чтобы взвесить некоторый груз, помещая гири только на одну чашку весов, надо представить его вес в виде суммы весов имеющихся гирь. При этом каждая гиря, очевидно, должна браться не более одного раза.

Пусть выбранные нами гири имеют вес p_1, p_2, p_3, p_4, p_5. Тогда груз весом $Q\le 30$ кг можно представить таким образом:

$$Q=a_1p_1+a_2p_2+a_3p_3+a_4p_4+a_5p_5,$$

— где каждый коэффициент равен 1, если гиря кладется на весы, или 0 в противном случае.

При такой постановке вопроса видно сходство с представлением числа Q в двоичной системе счисле-

ния. Нужно в качестве веса гирь p_1, p_2, p_3, p_4, p_5 взять степени двойки начиная с нулевой: $p_1=1$, $p_2=2$, $p_3=4$, $p_4=8$, $p_5=16$.

Сумма $p_1+p_2+p_3+p_4+p_5>30$, а любое натуральное число $Q\le30$, как мы знаем, можно представить по формуле (1.7) в виде

$$Q=a_1\cdot2^0+a_2\cdot2^1+a_3\cdot2^2+a_4\cdot2^3+a_5\cdot2^4.$$

Заметим, что получившееся решение задачи не является единственным. Например, вместо гири весом в шестнадцать килограммов можно взять пятнадцатикилограммовую гирю. Для любого груза, имеющего вес, не превосходящий 30 кг, гири, как легко понять, подбираются просто в соответствии с двоичным представлением соответствующего весу числа.

3.9. Контрольные вопросы и упражнения

1. Переведите в десятичную систему счисления числа, записанные в двоичной системе счисления:
   1; 101; 10000; 1000101010; 11001011.
2. Какое максимальное число можно записать в двоичной системе счисления пятью цифрами?
3. Перевести из двоичной системы в десятичную числа $0,0(0011)_2$; $0,(001)_2$. Здесь в скобках указан период бесконечной двоичной дроби.
4. Переведите десятичное число 52 в двоичную, восьмеричную и 11-ричную системы счисления.
5. Переведите следующие числа в десятичную систему счисления: 123_4; $123,4_5$; $203,5_6$; $0,(C)_{16}$.
6. Переведите число 1998 в традиционную периодическую систему счисления с основанием, равным вашему возрасту.
7. Переведите следующие десятичные дроби в троичную и восьмеричную системы счисления: 0,1; 0,3; 0,8; 0,(1); 0,(3); 0,(8); 0,13(18).
8. Переведите в восьмеричную систему конечную шестнадцатеричную дробь $BF3,6_{16}$.
9. Подсчитайте количество используемых операций (сложения и умножения) при вычислении десятичного значения числа *ABCDA*916 по развернутой форме записи и по схеме Горнера. Считаем, что операция возведения в степень производится последовательными умножениями.
10. В задаче о чудаке-математике из темы 3 первой части восстановите все числа в десятичной системе счисления.
11. Фокусник отгадывает задуманное число по спичкам. Загадавший должен в уме делить задуманное число пополам, полученную половину опять пополам и т.д. (для нечетных чисел берется целая часть от деления), и при каждом делении класть перед собой спичку, направленную вдоль, если делится число четное, и поперек, если нечетное. По полученной фигуре фокусник всегда безошибочно отгадывает число. Как он это делает?

Глава 4
Смешанные и нетрадиционные системы счисления

4.1. Теорема о смешанных системах счисления

Определение. Системы счисления с основаниями P и Q назовем ***смешанными***, если $P^m=Q$, где m — натуральное число.

Напомним, что **P—Q-*ичной*** называется позиционная система, в которой каждая цифра числа, заданного в Q-ичной системе, заменяется соответствующим ее представлением в P-ичной системе.

Теорема 2. Для систем счисления с основаниями P и Q ($P<Q$ и $P^m=Q$, где m — натуральное число) вид числа в P-ичной системе счисления совпадает с тем же числом, записанным в смешанной P—Q-ичной системе.

Иными словами, перевод чисел из такой P-ичной системы счисления в Q-ичную и наоборот производится по следующим правилам (сформулируем и докажем их пока только для целых чисел):

1) для того чтобы перевести целое число из системы счисления с основанием P в систему счисления с основанием Q, необходимо запись числа в системе P разбить на группы по m цифр, начиная с правой цифры, и каждую группу из m цифр заменить одной цифрой в системе Q (например: $10101_2=10|101=25_8$ ($P=2$; $Q=8$; $m=3$));

2) для того чтобы перевести целое число из системы счисления с основанием Q в систему счисления с основанием P, необходимо каждую цифру в записи числа в системе Q отдельно перевести в систему с основанием P и дополнить, если это необходимо, полученные числа слева нулями так, чтобы каждое полученное число, за исключением левого, состояло ровно из m цифр (например: $73_{16}=111|0011=1110011_2$ ($P=2$; $Q=16$; $m=4$)).

Доказательство:

1) Запишем исходное число в развернутом виде в системах счисления с основаниями P и Q:

$$a_0+a_1\cdot P+a_2\cdot P^2+\ldots+a_{n1}\cdot P^{n1} = b_0+b_1\cdot Q+b_2\cdot Q^2+\ldots+b_{n2}\cdot Q^{n2}.$$

Преобразуем выражение в левой части равенства следующим образом (разбиваем на группы по m членов):

$$a_0+a_1\cdot P+\ldots+a_{m-1}\cdot P^{m-1}+P^m\cdot\cdot(a_m+a_{m+1}\cdot P+\ldots+a_{2m-1}\cdot P^{m-1})+\ldots P^{2m}\cdot(\ldots)+\ldots=Q^0\cdot(a_0+a_1\cdot P+\ldots+a_{m-1}\cdot P^{m-1})+Q\cdot\cdot(a_m+a_{m+1}\cdot P+\ldots a_{2m-1}\cdot P^{m-1})+Q^2\cdot(\ldots)+\ldots$$

В последнем выражении мы просто заменили P^m на Q. Осталось показать, что любое выражение в скобках строго меньше Q. Для этого каждую цифру a_i P-ичной системы счисления заменим на максимальную цифру алфавита этой системы $[P-1]$:

$$a_0+a_1\cdot P+\ldots+a_{m-1}\cdot P^{m-1}\le$$

$$\le(P-1)(1+P+P^2+\ldots+P^{m-1})=(P-1)\frac{P^m-1}{P-1}<Q.$$

В приведенных преобразованиях была применена формула суммы конечного числа элементов геометрической прогрессии со знаменателем P и первым членом, равным единице. Очевидно, что полученное неравенство справедливо для любого выражения в скобках.

Значит, каждое выражение при степени Q можно записать в виде одной цифры Q-ичной системы счисления (в силу единственности представления натуральных чисел в любой системе счисления):

$$a_0+a_1\cdot P+\ldots+a_{m-1}\cdot P^{m-1}=b_0;$$

$$a_m+a_{m+1}\cdot P+\ldots a_{2m-1}\cdot P^{m-1}=b_1 \text{ и т.д.}$$

Первое утверждение доказано.

2) Представим каждую цифру b_i, $i=0, \ldots, n2$ в представлении исходного числа в Q-ичной системе счисления в P-ичной системе счисления. Так как $b_i<Q=P^m$, то максимальное количество цифр в полученном представлении равно m.

$$b_{0\,0}+b_{0\,1}\cdot P+b_{0\,2}\cdot P^2+b_{0(m-1)}\cdot P^{m-1}+$$
$$+P^m\cdot(b_{1\,0}+b_{1\,1}\cdot P+b_{1\,2}\cdot P^2+b_{1(m-1)}\cdot P^{m-1})+\ldots$$

После раскрытия скобок и в силу единственности представления чисел в P-ичной системе счисления получаем:

$$b_{0\,0}=a_0;$$
$$b_{0\,1}=a_1;$$
$$\ldots;$$
$$b_{1\,0}=a_m;$$
$$b_{1\,1}=a_{m+1}.$$

Примечание. Доказанные утверждения справедливы также и для дробных чисел, но целая и дробная часть числа переводятся в систему, смешанную с исходной, отдельно. Перевод дробной части из Q-ичной системы в P-ичную осуществляется так же, как и для целых чисел. Незначащими в дробной части теперь являются правые нули в P-ичном представлении самой правой цифры дробной части Q-ичного числа. При обратном переводе цифры P-ичной дроби группируются по m штук слева направо, начиная с первой цифры после запятой. Если последняя группа содержит менее m цифр, то к ней добавляют соответствующее количество нулей.

Пример 4.1. Переведем число A,1C16 из шестнадцатеричной системы в четверичную. Шестнадцатеричная и четверичная системы являются смешанными, т.к. $4^2=16$. Следовательно, группировать необходимо по две цифры. Переведем каждую цифру числа, записанного в 16-ричной системе, в четверичную систему, используя десятичную систему как промежуточную: $A_{16}=10_{10}=22_4$; $C_{16}=12_{10}=30_4$; $A,1C_{16}=22,|01|30_4$ (последний незначащий 0 можно опустить).

Следовательно, $A,1C_{16}=22,013_4$.

Пример 4.2. Переведем двоичное число $1010,00011011_2$ в восьмеричную систему. Двоичная и восьмеричная системы являются смешанными, $2^3=8$, т.е. при переводе группировать будем по три цифры двоичного числа $1|010,|000|110|11_2=12,066_8$ (последняя группа двоичных цифр была дополнена нулем справа).

Следовательно, $1010,00011011_2=12,066_8$.

Пример 4.3 (*Московский Государственный университет экономики, статистики и информатики, 1996 г.*).

Сумму восьмеричных чисел 17+1700+170 000+...+ +1 700 000 000 перевели в шестнадцатеричную систему счисления. Найдите в записи числа, равного этой сумме, пятую цифру слева.

Для решения этой задачи воспользуемся теоремой о смешанных системах счисления. Сначала выполним сложение столбиком в восьмеричной системе.

```
  1700000000
+    7000000
     170000
       1700
         17
-----------
 1717171717
```

Переведем полученное значение сначала в двоичную систему. Так как $8=2^3$, то каждая цифра в записи числа в восьмеричной системе заменяется на представление этой цифры в двоичной системе в группе из трех цифр:

$1717171717_8=001|111|001|111|001|111|001|111|001|111_2$.

А теперь число из двоичной системы переведем в шестнадцатеричную систему. Поскольку двоичная и шестнадцатеричная системы являются смешанными ($16=2^4$), то запись числа в двоичной системе необходимо разделить на группы по 4 цифры в каждой, начиная справа, и каждую группу цифр заменить одной цифрой шестнадцатеричной системы счисления:

$$00|1111|0011|1100|1111|0011|1100|1111_2=$$
$$=0F3CF3CF_{16}=F3CF3CF_{16}.$$

Заметим, что самая левая группа в полученном разбиении состоит из двух цифр и соответствует цифре ноль, то есть ее можно отбросить как незначащую. Таким образом, пятая цифра слева в шестнадцатеричной записи числа 1717171717_8 — тройка.

Задачи по программированию

Ю.А. СОКОЛИНСКИЙ

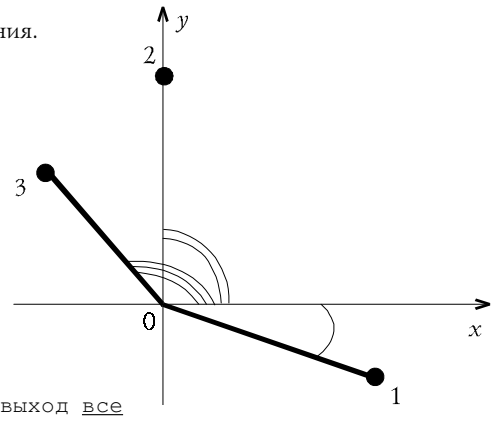
Предлагаются задачи по программированию для работы в классе. Предполагается, что составление программы для большинства задач (а при наличии компьютеров ее компиляция и выполнение) потребует от учащегося 20—30 минут. Более сложные задачи для сильных учащихся помечены звездочкой. Некоторые задачи целесообразно разбить на две части для слабых учащихся. Соответствующие рекомендации даны в тексте. Тематика задач: ветвления, массивы, символьные строки. Приводятся решения задач на языках Паскаль, Бейсик (версия QBasic), Си. При вводе исходных данных, удовлетворяющих некоторому условию, используется цикл с "постусловием": в Паскале это `repeat–until`, в Бейсике `DO–LOOP UNTIL`, в Си `do–while`. Все выходные числовые величины имеют целый тип, что позволяет в программах на Паскале и Си не загромождать вывод чисто техническими деталями.

Тема: "ВЕТВЛЕНИЯ"

Задача 1. На плоскости дана точка с координатами *x, y*, не лежащая на оси *X*, т.е. $y \neq 0$. Возьмем отрезок, соединяющий начало координат с этой точкой. Определить: угол между отрезком и осью *X* — острый, прямой или тупой?

Решение. Эта легкая задача имеет небольшой подвох: ответ не зависит от значения ординаты *y*, лишь бы она отличалась от нуля. Ясно, что при $x>0$ угол острый, при $x=0$ — прямой, при $x<0$ — тупой (см. *рисунок*).

Приведем несколько вариантов ее решения.



Алгоритм 1

```
если x>0 то вывод ("угол острый")
иначе если x=0
       то вывод ("угол прямой")
иначе вывод("угол тупой")
```

Недостаток подобных конструкций очевиден: в них трудно разобраться и немудрено запутаться.

Алгоритм 2

```
если x>0 то вывод ("угол острый"); выход все
если x=0 то вывод ("угол прямой"); выход все
вывод("угол тупой")
```

Этот вариант проще и яснее предыдущего. Здесь надо знать, как выглядит оператор выхода из программы в соответствующем языке. В Паскале это `exit`, в Бейсике — `SYSTEM`, в Си — `return`.

Алгоритм 3

```
если x>0 то вывод ("угол острый");
если x=0 то вывод ("угол прямой");
если x<0 то вывод ("угол тупой");
```

Несомненно, простейший вариант. Подчеркнем, что он совершенно правильный. Его недостаток: если выполняется не самое последнее условие, то происходит проверка заведомо невыполнимых условий. Ясно, что возникающие потери машинного времени весьма малы. Приведем программы, реализующие два последних алгоритма.

Алгоритм 2

Язык Паскаль

```
{Угол между осью X и отрезком, соединяющим начало координат
 и точку на плоскости, острый, прямой, тупой? }
var x:real;
begin
   write('Укажите абсциссу точки '); readln(x);
   if x>0 then begin writeln('Угол — острый'); exit end;
   if x=0 then begin writeln('Угол — прямой'); exit end;
   writeln('Угол — тупой');
end.
```

Язык Бейсик

```
'Угол между осью X и отрезком, соединяющим начало координат
'и точку на плоскости, острый, прямой, тупой?
DIM x AS SINGLE
INPUT "Укажите абсциссу точки "; x
IF x > 0 THEN PRINT "Угол — острый": SYSTEM
IF x = 0 THEN PRINT "Угол — прямой": SYSTEM
PRINT "Угол — тупой"
END
```

Язык Си

```
/*Угол между осью X и отрезком, соединяющим начало координат
 и точку на плоскости, острый, прямой, тупой? */
#include<stdio.h>
void main()
{float x;
 printf("\nУкажите абсциссу точки "); scanf('%f",&x);
 if (x>0)  {printf("\nУгол — острый"); return;}
 if (x==0) {printf("\nУгол — прямой"); return;}
 printf("\nУгол — тупой");
}
```

Алгоритм 3

Язык Паскаль

```
{Угол между осью X и отрезком, соединяющим начало координат
 и точку на плоскости, острый, прямой, тупой? }
var x:real;
begin
  write('Укажите абсциссу точки '); readln(x);
  if x>0 then writeln('Угол — острый');
  if x=0 then writeln('Угол — прямой');
  if x<0 then writeln('Угол — тупой');
end.
```

Язык Бейсик

```
'Угол между осью X и отрезком, соединяющим начало координат
'и точку на плоскости, острый, прямой, тупой?
DIM x AS SINGLE
INPUT "Укажите абсциссу точки "; x
IF x > 0 THEN PRINT "Угол — острый"
IF x = 0 THEN PRINT "Угол — прямой"
IF x < 0 THEN PRINT "Угол — тупой"
END
```

Язык Си

```
/*Угол между осью X и отрезком, соединяющим начало координат
 и точку на плоскости, острый, прямой, тупой? */
#include<stdio.h>
void main()
{float x;
 printf("\nУкажите абсциссу точки "); scanf("%f",&x);
 if (x>0)  printf("\nУгол — острый");
 if (x==0) printf("\nУгол — прямой");
 if (x<0)  printf("\nУгол — тупой");
}
```

Задача 2. На плоскости дана точка с координатами *x, y*, не лежащая на координатных осях, т.е. $x \neq 0$ и $y \neq 0$. Указать, в какой координатной четверти находится точка. Напомним, что первая четверть ограничена лучами, образованными положительной частью координатных осей, и четверти нумеруются против часовой стрелки (см. *рисунок*).

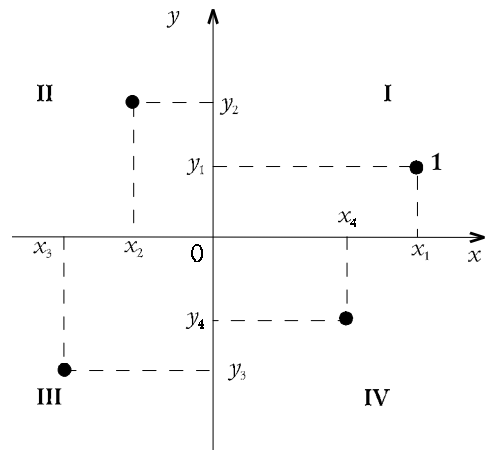
Решение. Воспользуемся простейшим вариантом организации ветвлений:

```
если x>0 и y>0 то вывод ("точка находится в первой четверти");
если x<0 и y>0 то вывод ("точка находится во второй четверти");
если x<0 и y<0 то вывод ("точка находится в третьей четверти");
если x>0 и y<0 то вывод ("точка находится в четвертой четверти").
```

Не забудьте, что ввод исходных данных надо организовать в цикле с проверкой условий `x<>0` и `y<>0`. Приведем соответствующие программы.

Язык Паскаль

```
{В какой четверти находится точка}
var x,y:real;
begin
   repeat {Цикл проверки условия x<>0}
     write('Укажите абсциссу точки, не равную 0 '); readln(x);
   until x<>0;
   repeat {Цикл проверки условия y<>0}
     write('Укажите ординату точки, не равную 0 '); readln(y);
   until y<>0;
   if (x>0) and (y>0) then
     writeln('точка находится в первой четверти');
   if (x<0) and (y>0) then
     writeln('точка находится во второй четверти');
   if (x<0) and (y<0) then
     writeln('точка находится в третьей четверти');
   if (x>0) and (y<0) then
     writeln('точка находится
     в четвертой четверти');
end.
```



Язык Бейсик

```
'В какой четверти находится точка
DEFSNG X-Y
DO 'Цикл проверки условия x<>0
INPUT "Укажите абсциссу точки, не
равную 0 ",x
LOOP UNTIL x <> 0
DO 'Цикл проверки условия y<>0
INPUT "Укажите ординату точки, не
равную 0 ",y
LOOP UNTIL y <> 0
IF x > 0 AND y > 0 THEN
 PRINT "точка находится в первой четверти"
END IF
IF x < 0 AND y > 0 THEN
 PRINT "точка находится во второй четверти"
END IF
IF x < 0 AND y < 0 THEN
 PRINT "точка находится в третьей четверти"
END IF
IF x > 0 AND y < 0 THEN
 PRINT "точка находится в четвертой четверти"
END IF
END
```

Язык Си

```
/*В какой четверти находится точка */
#include<stdio.h>
void main()
{float x,y;
 do /*Цикл проверки условия x<>0 */
  {printf("\nУкажите абсциссу точки, не равную 0 ");
  scanf("%f",&x);
  }
 while (x == 0);
 do /*Цикл проверки условия y<>0 */
  {printf("\nУкажите ординату точки, не равную 0 ");
  scanf("%f",&y);
  }
 while (y == 0);
 if (x>0 && y>0)printf("\nТочка находится в первой четверти");
 if (x<0 && y>0)printf("\nТочка находится во второй четверти");
 if (x<0 && y<0)printf("\nТочка находится в третьей четверти");
 if (x>0 && y<0)printf("\nТочка находится в четвертой четверти");
}
```


Окончание на с. 12

ЗАДАЧИ

Задачи по программированию

Окончание. Начало на с. 11

Задача 3

а) В языке Паскаль имеется функция `round(a)`, которая округляет вещественное число `a` до ближайшего целого.

Например: `round(3.2)=3; round(3.7)=4`.

Используя эту функцию, составить программу на Паскале округления вещественного числа `a` до ближайшего целого, а также до ближайшего целого снизу и сверху.

б) В языке Бейсик имеется функция `INT(a)`, которая округляет вещественное число `a` до ближайшего целого снизу. Например: `INT(3.2) = 3; INT(3.7) = 3`.

Используя эту функцию, составить программу на Бейсике округления вещественного числа `a` до ближайшего целого, а также до ближайшего целого снизу и сверху.

в) В языке Си имеется функция `ceil(a)`, которая округляет вещественное число `a` до ближайшего целого сверху. Например: `ceil(3.2) = 4; ceil(3.7) = 4`.

Используя эту функцию, составить программу на Си округления вещественного числа `a` до ближайшего целого, а также до ближайшего целого снизу и сверху.

Решение. Во-первых, если какое-либо округление вещественного числа не меняет его значения, то число целое и остальные способы округления дадут тот же результат. В противном случае разность между ближайшим целым сверху и ближайшим целым снизу равна единице.

Обозначим через `Near` ближайшее целое к `a`; `NearDn, NearUp` — ближайшее целое к `a` снизу и сверху. Тогда:

```
если Near< a то NearDn=Near; NearUp=Near+1 все
если Near> a то NearUp=Near; NearDn=Near-1 все
```

Наконец, `если a <NearDn+0.5 то Near=NearDn иначе Near=NearUp`

Возможные варианты показаны на *рисунке*.

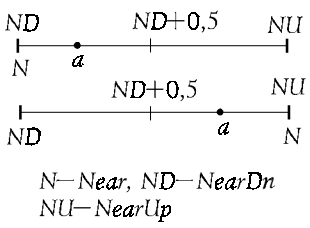
Приведем программы.

Язык Паскаль

```
{Округление до ближайшего целого и до ближайшего целого снизу и сверху}
var a:real;
Near,NearDn,NearUp:integer;
begin
   write('Введите число a '); readln(a);
   Near:=round(a);
   if Near=a then begin NearDn:=Near; NearUp:=Near end;
   if Near<a then begin NearDn:=Near; NearUp:=Near+1 end;
   if Near>a then begin NearDn:=Near-1; NearUp:=Near end;
   writeln('Число ближайшее целое к a ',Near);
   writeln('Число ближайшее целое снизу к a ',NearDn);
   writeln('Число ближайшее целое сверху к a ',NearUp);
end.
```

Язык Бейсик

```
'Округление до  ближайшего целого и
'до ближайшего целого снизу и сверху.
DIM A AS SINGLE
DEFINT N
INPUT "Введите число A ", A
NearDn = INT(A)
IF NearDn = A THEN
   NearUp = NearDn: Near = NearDn
ELSE
 NearUp = NearDn + 1
 IF A < NearDn +.5 THEN Near = NearDn ELSE Near = NearUp
END IF
PRINT "Число ближайшее целое к A "; Near
PRINT "Число ближайшее целое снизу к A "; NearDn
PRINT "Число ближайшее целое сверху к A "; NearUp
END
```



Язык Си

```
/* Округление до ближайшего целого и до ближайшего целого снизу и сверху  */
#include<stdio.h>
#include<math.h>
void main()
{float a;
 int Near,NearDn,NearUp;
 printf("\nВведите число a "); scanf("%f",&a);
 NearUp=ceil(a);
 if (NearUp==a) {Near=NearUp; NearDn=NearUp;}
 else
 {NearDn=NearUp-1;
  if (a<NearDn+.5) Near=NearDn; else Near=NearUp;
 }
printf("\nЧисло ближайшее целое к a %d",Near);
printf("\nЧисло ближайшее целое снизу к a %d",NearDn);
printf("\nЧисло ближайшее целое сверху к a %d",NearUp);
}
```

Задача 4. Даны три целых числа. Выдать их в порядке возрастания.

Решение. Конечно, использование стандартных методов сортировки означало бы в данном случае "стрельбу из пушек по воробьям". Обозначим заданные величины `a, b, c`, а эти же величины, но расположенные в порядке возрастания, — `a1, b1, c1`. Упорядочим сначала `a` и `b`, обозначив меньшее из них как `p`, а большее — `q`.

```
если a<= b то p = a; q = b иначе p = b; q = a все
```

Теперь надо рассмотреть три случая расположения `c` по отношению к `p` и `q`; `c` может быть не больше `p`, находиться между `p` и `q`, быть больше `q`.

```
если c<=p то a1 = c; b1 = p; c1 = q все
  если c>p и c<= q то a1 = p; b1= c; c1= q все
  если c> q то a1 = p; b1= q ; c1= c все
```

12

Прежде чем привести программы, реализующие этот алгоритм, отметим, что в них не контролируется, являются ли вводимые числа целыми. Такой контроль представляет собой самостоятельную и более трудную задачу (подобная проблема рассматривается в задаче 5).

Язык Паскаль

```
{Упорядочить три числа}
var a,b,c,a1,b1,c1,p,q:integer;
begin
   write('Введите три целых числа '); readln(a,b,c);
   if a<=b then begin p:=a; q:=b end else begin p:=b; q:=a end;
   if c<=p then begin a1:=c; b1:=p; c1:=q end;
   if (c>p) and (c<=q) then begin a1:=p; b1:=c; c1:=q end;
   if c>q then begin a1:=p; b1:=q; c1:=c end;
   writeln('Числа в порядке возрастания ',a1,' ',b1,' ',c1);
end.
```

Язык Бейсик

```
'Упорядочить три числа
DEFINT A-C, P-Q
INPUT "Введите три целых числа "; a, b, c
IF a <= b THEN p = a: q = b ELSE p = b: q = a
IF c <= p THEN a1 = c: b1 = p: c1 = q
IF c > p AND c <= q THEN a1 = p: b1 = c: c1 = q
IF c > q THEN a1 = p: b1 = q: c1 = c
PRINT "Числа в порядке возрастания "; a1, b1, c1
END
```

Язык Си

```
/*Упорядочить три числа*/
#include<stdio.h>
void main()
{int a,b,c,a1,b1,c1,p,q;
 printf("\nВведите три целых числа ");
 scanf("%d%d%d",&a,&b,&c);
 if (a<=b) {p=a;q=b;} else {p=b;q=a;}
 if (c<=p) {a1=c; b1=p; c1=q;}
 if (c>p && c<=q) {a1=p; b1=c; c1=q;}
 if (c>q) {a1=p; b1=q; c1=c;}
 printf("\nЧисла в порядке возрастания %d %d %d",a1,b1,c1);
}
```

Задача 5* (Напомним, что здесь и далее звездочкой помечены более сложные задачи.)

Все символы разделены на 4 класса:

1) буквы латинского алфавита (заглавные и строчные);
2) цифры;
3) знаки арифметических действий: "+", "–", "*", "/";
4) прочие символы.

Для заданного символа сообщить, в какой класс он входит.

Указание 1. Буквы латинского алфавита, как заглавные, так и строчные, а также цифры расположены в таблице символов подряд.

Указание 2. Функция перевода символа в верхний регистр в Паскале имеет имя `upcase`, в Бейсике — `UCASE`, в Си — `toupper`.

Решение. Обозначим заданный символ через `Sim`. Прежде всего полезно перевести символ в верхний регистр, поскольку это сократит условие вхождения символа в класс букв. Сформулируем алгоритм, используя условные операторы:

```
если "A"<= Sim и Sim<="Z" то
вывод ("Символ входит в класс букв"); выход все
если "0"<= Sim  и Sim<= "9" то
вывод ("Символ входит в класс цифр"); выход все
если Sim= "+" или Sim="-" или Sim="*" или Sim="/" то
вывод ("Символ входит в класс знаков"); выход все
вывод ("Символ входит в класс прочих символов")
```

Отметим, что использование оператора выхода из программы в этой конструкции обязательно.

В языках Паскаль и Бейсик можно вместо условных операторов использовать оператор разветвление (выбор), обладающий значительной наглядностью. В Паскале это `case`, а в Бейсике — `SELECT CASE`. Дело в том, что здесь в качестве "метки", определяющей вариант выбора, можно использовать:

а) последовательность символов, расположенных в таблице символов подряд; при этом указываются только первый и последний символ;

б) несколько символов.

В программе на Паскале используется оператор разветвления, а в программе на Си — условные операторы.

Язык Паскаль

```
{Символ входит в класс букв, цифр, арифметических знаков
 или прочих символов?}
uses crt;
var Sim:char;
begin
   write('Введите символ '); readln(Sim);
   Sim:=Upcase(Sim);
   case Sim of
    'A'..'Z': writeln('Символ '+Sim+' входит в класс букв');
    '0'..'9': writeln('Символ '+Sim+' входит в класс цифр');
    '+','-','*','/':
     writeln('Символ '+Sim+' входит в класс знаков');
   else
    writeln('Символ '+Sim+' входит в класс прочих символов');
   end;
end.
```

Язык Си

```
/*Символ входит в класс букв, цифр, арифметических знаков или прочих символов? */
#include<stdio.h>
#include<ctype.h>
void main()
{char Sim;
 printf("\nВведите символ "); scanf("%c",&Sim);
 Sim=toupper(Sim);
 if ('A'<=Sim && Sim<='Z')
 {printf("\nСимвол %c входит в класс букв":,Sim); return;}
 if ('0'<=Sim && Sim<='9')
 {printf("\nСимвол %c входит в класс цифр",Sim); return;}
 if (Sim=='+' || Sim=='-' || Sim=='*' || Sim=='/')
 {printf("\nСимвол %c входит в класс знаков",Sim); return;}
 printf("\nСимвол %c входит в класс прочих символов",Sim);
}
```


Продолжение в следующем номере

Современные форматы графических файлов

Вечера с Вовой

А.Г. ЛЕОНОВ

Продолжение. Начало в № 17/99

• • •

— Представь, что ты нарисовал треугольник. Для сохранения рисунка в BMP-файле тебе потребуются 1,4 Мб, а если рисовать треугольник векторами, указывая координаты начала и конца каждого отрезка (координаты двухбайтовые)? Какой объем файла потребуется для сохранения такого треугольника?

— 16 байт. — Ответ прозвучал через несколько секунд.

— Но мы будем рассматривать только растровую графику. Теперь еще вопрос. Как ты думаешь, если сохранить в файле только растр картинки, который еще называют растровым массивом, сумела бы программа Ulead Photo Express разобраться, что текст не является картинкой?

— Нет.

— Правильно! Для этого потребуется еще дополнительная информация в файле. Она называется заголовком графического файла. В нем указываются тип файла и некоторая дополнительная информация, например, размер файла. Давай заглянем внутрь BMP-файла. Для примера возьмем файл с фотографией кукол DOLLS.BMP размером 1407 Кб. Откроем его при помощи Ulead Photo Express. Посмотрим характеристики файла (*properties*):

```
Data type:  RGB True color (24-bit)
Ширина:  800 Pixels
Высота:  600 Pixels
Размер:  1406 Кб
Имя:     DOLLS.BMP
Формат:  Windows bitmap
Compression:None
```

— Не обращай пока внимания на "страшное слово" — *compression*. Теперь при помощи специальной программы HEX-viewer "посмотрим" на внутренности файла:

```
000000 42 4D 36 F9 15 00 00 00 00 00 36 00 00 00 28 00 BM6·····6···(·
000010 00 00 20 03 00 00 58 02 00 00 01 00 18 00 00 00 ·····X·········
000020 00 00 00 F9 15 00 25 16 00 00 25 16 00 00 00 00 ·····%···%·····
000030 00 00 00 00 00 00 1D 35 55 1C 36 56 20 39 58 25 ······5U·6V 9X%
000040 3D 5C 28 3F 5F 28 3F 5F 26 3E 5A 24 3A 56 20 35 =\(?_(?_&>Z$:V 5
```

Файл начинается с символов "BM", указывающих на формат файла: Windows bitmap, далее закодирован размер файла: 4 байта — шестнадцатеричные числа: 00, 15, F9, 36. Закодированное число читается справа налево. Теперь возьми бумажку и вычисли размер файла, переведя числа из шестнадцатеричного в десятичное представление, а затем сравни с реальным размером файла.

— Ответ готов, — доложил Вова, проведя необходимые вычисления, — 1440054.

Задача 1.2. Проверьте: не ошибся ли Вова?

— А теперь посмотри, вот таблица, описывающая формат BMP:

Я заполнил лишь первые две колонки: название и длина, а третью оставил Вове. После того как он закончил свою колонку, я дописал комментарии, а он перевел числа из шестнадцатеричной системы в десятичную. Большинство полей формата ему были понятны, однако некоторые требовали пояснения.

— А что такое разрешение? — спросил Вова.

— Когда ты сканируешь фотографию или делаешь снимок цифровым фотоаппаратом, происходит преобразование в дискретную, цифровую форму. Изображение получает определенный размер в пикселях (точках), и, чтобы при этом не потерялась информация о реальном исходном размере фотографии, требуется хранить разрешение, указывающее, сколько точек изображения помещалось в 1 см (или в 1 дюйме). Разрешение (как горизонтальное, так и вертикальное) вычисляется делением получившегося размера в пикселях на соответствующий размер. Если изображение имеет 800 точек по горизонтали, а длина исходной фотографии была 12 см, то при сканировании использовалось разрешение 800/12 = 67 точкам в 1 см. Чем выше разрешение, тем качественнее получится изображение при печати, но тем больше будет растровый массив и, соответственно, файл займет больше места.

Название	Длина (в байтах)	Содержимое файла DOLLS.BMP	Примечание
Заголовок файла (в том числе)	14		
"BM"	2	42 4D	Это код "BM"
Размер файла	4	36 F9 15 00	=1440054
Не используется	4	00 00 00 00	
С какого байта начало растрового массива	4	36 00 00 00	
Доп. информация (всего)	40		
В том числе			
Длина	4	28 00 00 00	=40
Ширина изображения	4	20 03 00 00	=800
Высота изображения	4	58 02 00 00	=600
Спец. информация	2	01 00	
Бит/пиксель	2	18 00	=24 битам на точку
Метод сжатия	4	00 00 00 00	
Длина растрового массива	4	00 F9 15 00	=1440054—54
Горизонтальное разрешение	4	25 16 00 00	
Вертикальное разрешение	4	25 16 00 00	
Число цветов	4	00 00 00 00	нет палитры
Число основных цветов	4	00 00 00 00	нет палитры
Таблица цветов	до 1024	нет	может отсутствовать
Растровый массив	переменная	1D 35 ...	

— Разрешение имеет смысл только при печати. При рисовании картинки никакого разрешения может и не быть. Правда, какое-то разрешение присваивается по умолчанию, ведь предполагается, что нарисованная картинка потом будет напечатана. Понятно?

— Понятно.

— Теперь еще немного про формат BMP.

BMP — это сокращение от BitMaP (битовая карта), "родной" формат графических файлов для Windows, поскольку наиболее близко соответствует внутреннему формату системы, в котором последняя хранит свою графическую информацию. Файлы, представленные в BMP-формате, чаще всего имеют расширение BMP, хотя говорят, что допустимо и другое расширение имени файла — RLE (*run length encoding* — кодирование длины серий). Это расширение указывает на то, что произведено сжатие файла одним из способов сжатия RLE. При этом файл становится меньше, что экономит место на диске. Но об этом позже. В нашем примере информация о цвете точки была закодирована 24 битами...

— То есть точка могла "гореть" одним из 16,7 млн цветов? — спросил Вова.

— Да, но в целях экономии места иногда используют меньшее число бит для кодирования информации о точке.

— А! Знаю-знаю! — Вова захлопал в ладоши. — 1 бит для черно-белых изображений, какая экономия!

Задача 1.3. Подсчитайте, какая экономия при 1 бите на точку (растр по-прежнему 800×600), 4 битах на точку, 8 битах, 16 битах.

— В файлах BMP информация о цвете каждого пикселя кодируется 1, 4, 8, 16 или 24 битами на пиксель. Это число, называемое глубиной представления цвета, и определяет максимальное число цветов в изображении.

— Ага, изображение глубиной 1 бит может иметь всего два цвета: черный и белый, — добавил Вова.

— Но вернемся к формату. Формат собственно данных растрового массива в файле BMP зависит от числа бит, используемых для кодирования данных о цвете каждого пикселя. При 256-цветном изображении под каждый пиксель растрового массива отводится один байт (8 бит). Этот байт не содержит яркостей основных цветов красного, синего и зеленого (RGB)...

— Как было в режиме *true color*?

— Да, но в случае 256-цветного изображения байт пикселя содержит номер цвета в таблице цветов файла. Таким образом, если в качестве первого значения цвета RGB в таблице цветов файла BMP хранится R/G/B=255/0/0, то значению пикселя 0 (в растровом массиве) будет поставлен в соответствие ярко-красный цвет.

Сама таблица цветов, называемая еще палитрой, представляет собой массив. Индекс элемента массива — это номер цвета палитры, а содержимое массива — тройки чисел, описывающие цвет яркостью трех основных компонент R/G/B (красного, зеленого, синего). Как ты видел, 24-битные изображения не имеют таблицы цветов.

Файлы BMP с глубиной 16 бит на пиксель также не имеют таблиц цветов, в них, как в 24-битных файлах, значения пикселей растрового массива непосредственно задают значения цветов RGB. Значения пикселей в растровом массиве хранятся слева направо, начиная (как правило) с нижней строки изображения. Таким образом, в 256-цветном BMP-файле первый байт данных и растрового массива представляет собой индекс для цвета пикселя, находящегося в нижнем левом углу изображения; второй байт представляет индекс для цвета соседнего справа пикселя и т.д. Если число байт в каждой строке нечетно, то к каждой строке добавляется еще байт, чтобы "выровнять" данные растрового массива до четного числа.

— Это все? — с надеждой спросил мой усталый собеседник. — Теперь я все знаю про BMP-формат?

— Могут быть некоторые вариации по хранению различных частей файла; так, растровый массив в некоторых 16- и 256-цветных BMP-файлах может быть сжат алгоритмом RLE. Но об этом в другой раз.

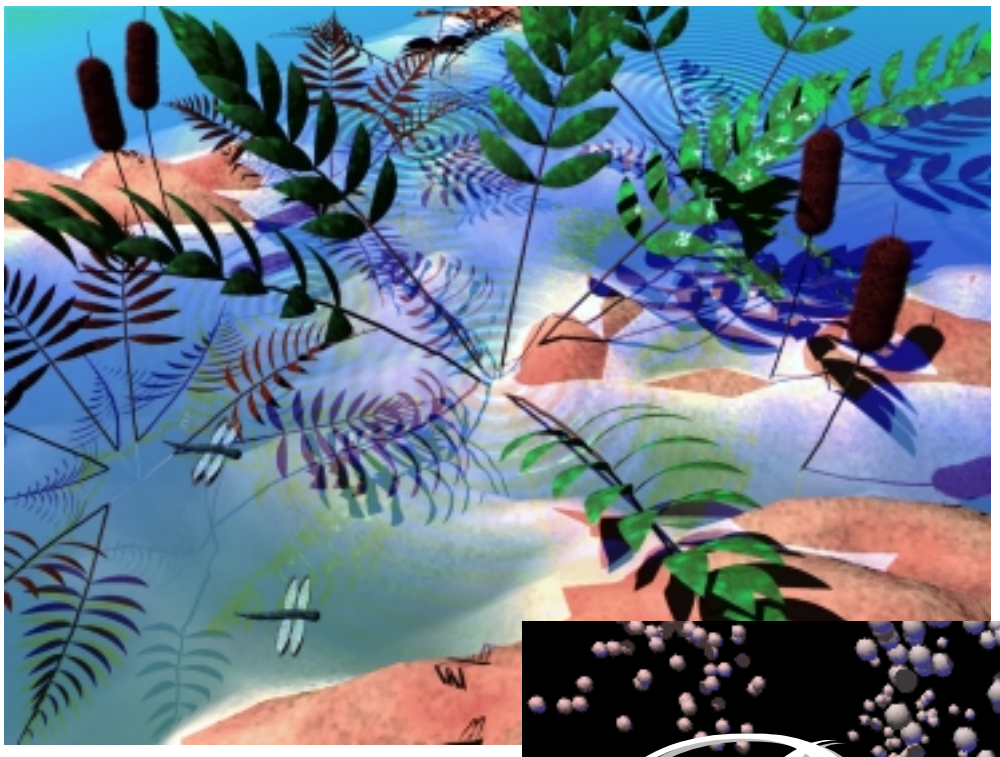
Вова вскочил, потянулся.

— Ну, я пошел, дядя Саша? Завтра зайду?

— Спокойной ночи, Вова. Жду тебя завтра. Будем чай пить и форматы обсуждать. Там еще масса интересного. Хватит не на один вечер.

Продолжение следует

БЕСЕДЫ



А.И. СЕНОКОСОВ

Виртуальная школа

Возможно, не очень многим известно, что, кроме господина Сороса, российскому образованию помогает еще и Государственный департамент США в лице одного из своих подразделений, называемого USIA.

Вот уже второй год среди российских учителей английского языка проводится конкурс, победители которого получают возможность стажировки в США, а в их школы поставляется разнообразная техника. Кому-то достаются компьютеры, кому-то копиры...

В качестве продолжения этого проекта американцы организовали электронную конференцию, на которой учителя бывшего СССР и США могут обсуждать любые интересующие их профессиональные проблемы.

Вот и в этом году одним из участников дискуссии был предложен вопрос:

Какие факторы вашей профессии являются для вас наиболее важными?

Далее необходимо было отранжировать различные факторы, такие, как наличие мотивации у учеников, гарантированная работа, хорошая зарплата и т.п. В частности, одним из таких факторов является техническое оснащение школ.

Одна из американских учительниц, обосновывая первостепенное значение технической оснащенности, рассказала, что в каждом классе их школы стоит компьютер с выходом в Интернет, в школе три компьютерные лаборатории, оснащенные сканерами, цифровыми фотоаппаратами, проекционной техникой, есть компьютеризованная библиотека... Все это техническое изобилие позволяет совершенно по-новому организовать учебный процесс и говорить о принципиально другом подходе к преподаванию.

Мне показалось интересным начать свой рассказ именно с этой исходной посылки.

Представьте на долю секунды, что в вашем распоряжении все то изобилие техники, которое имеет американская школа. Как бы вы им распорядились? Как бы вы выстроили свою "виртуальную школу"?

Нет, само собой, в кабинете (двух, трех) информатики появилась бы суперсовременная техника, которая дала бы наконец возможность перейти к непосредственному изучению передовых офисных технологий.

К информатике мы еще вернемся, а что эта современная техника могла бы дать другим предметам?

Давайте начнем вместе. В каждый класс на каждое рабочее место поставим по компьютеру...

Стоп! Во-первых, этого не даст сделать санэпидстанция, жестко регламентирующая время работы школьников за компьютерами одним часом в неделю, а во-вторых, довольно проблематично подобрать программное обеспечение по любому предмету такое, которое бы в точности соответствовало вашему курсу, и даже если бы за компьютерами можно было работать неограниченно долго, вряд ли бы они использовались больше, чем на 15—20 уроках в год.

Хорошо. Пускай будет, как у них: один компьютер на класс с хорошим экранным проектором и прямым выходом в Интернет... Вот здорово! Идет, скажем, рассказ об эпохе Возрождения — и мы можем показать экспозицию ведущих музеев Европы...

Ну, во-первых, эту экспозицию нужно еще найти. Далеко не факт, что в Интернете вообще есть то, что нужно именно вам на конкретном уроке. Но даже сам поиск — это особое искусство. В качестве примера можно сослаться на эксперимент, проведенный в одной из элитарных школ Екатеринбурга. Учитель предложил завсегдатаям Интернета найти в нем законы Ньютона. Не прошло и пары часов, как задача была решена. Правда, к несчастью, информация оказалась на английском...

Во-вторых, иллюстрации в Интернете редко отличаются большим форматом и выдающимся качеством: никто не собирается выкладывать во всеобщее пользование информацию, имеющую немалую коммерческую стоимость. Опять же в качестве примера могу привести поучительную историю о выпуске мультимедийного диска к 275-летию города Екатеринбурга. Оказалось, что чуть ли не половину стоимости выпуска диска составила оплата права на использование в нем фотоматериалов. Думаю, излишне объяснять то, что на живом реальном уроке нужны именно иллюстрации, а не многокилобайтные текстовые описания.

И в-третьих, для успешного использования Интернета в режиме реального времени урока нужен не просто выход в Интернет, а высокоскоростной выход. Здесь не подходит не только модем, но и значительно более высокие скорости современных российских выделенных каналов. Видимо, минимум, которым можно довольствоваться, в настоящее время составляет 1Мбод. Даже в США далеко не каждая школа имеет такой высокоскоростной канал, не говоря уж о России. Выходить же за американские рамки мы вроде бы не договаривались...

Ну, хорошо. Качественное проекционное оборудование и мощный компьютер позволяют в конце концов использовать тематические компакт-диски по самым разнообразным предметам...

Видимо, и тут нас поджидают сплошные огорчения. Что касается американского рынка, то там на самом деле каталог учебных компакт-дисков насчитывает несколько тысяч наименований. В распоряжении же наших учителей таковых считанные единицы. Да и те, как показывают "Медиаобзоры" "Информатики", как правило, не предназначены для работы в классе на проекционном оборудовании. Исключение, пожалуй, составляют достаточно многочисленные и качественные компакт-диски по истории. Но и тут вряд ли можно говорить об использовании компьютерной техники на большей части уроков. Конечно, можно возразить, что появление в изобилии необходимых дисков — только вопрос времени. Вот еще бы только понять, какого...

Итак, вполне можно прийти к неутешительному выводу о том, что оснащение какой-либо российской школы самой современной компьютерной техникой вряд ли способно само по себе привести к какому-то ощутимому изменению характера образования и его качества.

А, собственно говоря, почему нас должны тревожить другие предметы? В конце-то концов подавляющему числу читателей этой газеты деньги платят именно за преподавание информатики, а тут — все вроде бы идеально.

Но давайте вспомним основной круг задач, поставленных в свое время перед школьным предметом "информатика", прекрасно очерченный в статье А.Ю. Уварова "Чему и как учить на уроках информатики" (см.: "Информатика" № 1/99). Вкратце напомним:

1. Знакомство с компьютером и с процессами информатизации как с необходимым элементом среды обитания ("общекультурная составляющая").
2. Начальная компьютерная грамотность в технологическом смысле (знание текстовых редакторов и т.п.).
3. "Предпрофессиональная" составляющая — подготовка специалистов для промышленности обработки данных в терминах Г.Р. Громова.
4. "Общеобразовательная" составляющая — обучение алгоритмическому мышлению.
5. "Общепедагогическая" составляющая — учитель информатики и его кабинет как проводник новых компьютерных технологий обучения в своей школе.

Пункты 1, 2 и 4 входят в "Образовательный минимум" и поэтому в той или иной степени присутствуют во всех учебниках информатики, одобренных Министерством. Подавляющему большинству учителей информатики совершенно ясно, о чем в них идет речь и какими средствами можно более или менее успешно решить поставленные в них задачи.

Пункт 3 — предмет углубленного курса.

Темным облачком на ясном горизонте представляется лишь пятый пункт, скажем прямо, далеко не самый животрепещущий для тех же учителей информатики. Он оказался вроде бы никому конк-

ретно не нужным. Хотя, как было справедливо замечено в той же статье, надобность в интенсификации учебного процесса и повышении производительности педагогического труда отнюдь не исчезла.

И, как ни удивительно, за последние десять лет альтернативных путей для решения этой проблемы не появилось. Именно несбывшиеся надежды, связанные с "общепедагогической составляющей" заставляют снова и снова возвращаться к треклятому вопросу о месте информатики в школьном образовании и ее интеграции с другими предметами.

Так как же "разъять живую ткань действительности", определить, где кончается область действия информатики и начинается, к примеру, математика?

И тут пришла пора сделать лирическое отступление. В конце XIX века физика тоже переживала один из самых приятных моментов в своей истории. Казалось, основные законы открыты, особых проблем не предвидится, и только пара мелких проблем темными облачками маячила на горизонте. В частности, корпускулярно-волновой дуализм света...

Все мы прекрасно знаем, к чему привели эти облачка. XX век фактически стал веком появления принципиально новой физики. Очень многие проблемы, неразрешимые в рамках классической физики, в том числе и дуализм света, были решены совершенно по-новому. Но "разнималась ли при этом живая ткань действительности"? Как раз нет! Можно сказать, что физика первой из наук перешла не просто к пониманию, но и постулированию того, что впоследствии было названо "волновой парадигмой".

Вкратце ее можно сформулировать, как отказ от однозначного объяснения природы явлений и понимание того, что все процессы в природе взаимозависимы.

К сожалению, осознание "волновой парадигмы" в других науках существенно запоздало. Так, лишь в 70-е годы стремительно набирающая силы экология стала ее явным выражением в биологии. Что же касается общественных наук, то мы становимся живыми свидетелями начала их переосмысления в направлении новой парадигмы.

"В человеке все от наследственности, решительно все, человек на сто процентов определяется генами, родителями, происхождением, предками.

И вместе с тем в человеке все от воспитания, решительно все, он на сто процентов определяется прямым воспитанием и обстоятельствами жизни.

Не половина на половину, а сто процентов на сто процентов". (С.Л. Соловейчик. "Последняя книга".)

Возможно, даже сегодня многими это может быть воспринято, как метафора выдающегося педагога. Однако в рамках волновой парадигмы понимать подобное нужно буквально. Как научную истину. Быть может, немножко непривычную, но в начале века такой же непривычной была истина о том, что свет — "волна и частица одновременно".

Так нужно ли "разнимать живую ткань действительности" для того, чтобы выйти на новый уровень ее осознания и познания? Быть может, не нужно "размежевываться", чтобы впоследствии "интегрироваться"?

Теперь пришла пора вспомнить о "Многотрудной, полной невзгод и опасностей истории компьютеризации школы № 104 города Екатеринбурга..." (см.: "Информатика" № 3, 5, 6, 9 и 11/98).

Напомним, что в ней шла речь именно об интенсификации учебного процесса в школе путем создания локальной сети и использования фактически всего лишь двух самоочевидных программных средств: текстового редактора в качестве электронной доски и программы просмотра графических файлов.

Но даже такой вариант использования компьютерной техники привел, по нашим данным, к увеличению интенсивности урока на 15—20%.

Возникает вполне закономерный вопрос: можно ли опыт работы 104-й школы считать решением пресловутой пятой проблемы? Как ни странно, ответ довольно очевиден — нет! Безусловно, проблема интенсификации, скажем так, сдвигается с мертвой точки, но школьная информатика здесь совершенно ни при чем.

Так, в одном из районов нашего города предлагался совершенно иной подход к организации мультимедиа-кабинетов. На базе райметодкабинета был организован неплохо оснащенный компьютерный центр и предполагалось, что учителя из различных школ делают заказы на компьютерные слайды именно там. А в первую очередь мультимедиа-комплексами были оснащены кабинеты физики в пяти школах.

Такой подход предполагает наличие специалиста (одного на весь район), который занимается исключительно проблемой кабинетов подобного сорта. Вполне возможно, что этот вариант тоже может привести к неплохим результатам.

Рассчитывать же на то, что замордованные безумным количеством часов учителя информатики на голом энтузиазме или за символическую оплату начнут в массовом порядке активно внедрять новые технологии образования, может лишь человек, весьма далекий от реалий современной российской школы. Собственно говоря, А.Ю. Уваров это прекрасно осознает, когда называет "пятую проблему" неразрешимой в существующих рамках.

И все-таки, отвечая на технологические вызовы грядущего нового века, школьная информатика должна выйти на новый уровень своего развития...

Да, кстати, немного об этой заезженно-затасканной фразе. Представляется, что ее тоже нужно понимать буквально. Рассмотрим, например, следующую интересную цепочку вопросов-ответов.

1. В курсе информатики мы изучаем устройство компьютера. А зачем?
2. Для того, чтобы человек понимал основные принципы его работы и мог грамотно общаться с операционной системой. А зачем ему грамотно общаться?
3. Для того, чтобы успешно использовать прикладные программы, тот же текстовый редактор. А зачем ему этот самый текстовый редактор?
4. Для того, чтобы в дальнейшей жизни он смог при необходимости подготовить документ на компьютере.

Стоп. Ответ на этот вопрос уже вне сферы задач, которые решает школа. И, вообще говоря, он, возможно, и не предполагает дальнейших вопросов, хотя один явно напрашивается: а если такой необходимости не будет?

(Понятно, что текстовый редактор всегда можно защитить. Им на самом деле пользуется довольно много людей. А вот если прикинуть, какой процент людей нуждается в своей профессиональной деятельности, скажем, в том, чтобы создавать новые базы данных на "Access"... По самым завышенным оценкам, на долю таковых приходится никак не более 2—3% выпускников, которые к тому же чаще всего оканчивают профильные школы.)

Можно привести еще одну цепочку, тоже относящуюся к информатике:

1. В курсе информатики мы изучаем Систему Команд Бездумного Исполнителя (вопрос везде один и тот же).
2. Для того, чтобы школьники могли на наглядном примере понять, как работают стандартные алгоритмические конструкции.
3. Для того, чтобы развить в себе один из типов мышления — алгоритмическое мышление.
4. Потому что "его развитие представляет самостоятельную ценность, так же как и развитие мышления человека вообще" (см.: *А.Г. Кушниренко, Г.В. Лебедев.* "12 лекций...", "Информатика" № 1/99).

Последний уровень здесь оказался почти таким же, как и в аналогичных построениях для математики, физики, литературы...

Таким образом, информатика оказывается, и совершенно справедливо, в ряду предметов, занимающихся *образованием* (*education*) человека, а не *натаскиванием* (*training*), причем образованием такого сорта, какое не способен дать ни один другой школьный предмет. Автор одной из публикаций в известном компьютерном еженедельнике как-то заметил: "Ни один школьный предмет не учит формулировать задачу для кого-то, там уже все сформулировано для решения задач самим учащимся, и информатика, пойдя по тому же накатанному пути, потеряла (бы) свою индивидуальную сущность, утратив заодно и интегративную функцию, как средства к обучению постановке задач". К сожалению, автор, разуверившись в известной ему школьной информатике, частицу "бы" в своей статье не поставил.

Вернемся к разобранным примерам. В сфере задач школьной информатики оказывались три или четыре уровня вопросов-ответов. Новый уровень, о котором мы начали говорить, предполагает еще один вопрос "а зачем?", на который будет дан относящийся к школе ответ:

...Для того, чтобы сделать существенно более интересным и эффективным процесс школьного обучения.

Или

...Для того, чтобы изменить характер восприятия учениками своей школы, сделав ее более привлекательной за счет существенного расширения возможностей к самовыражению.

Эти два ответа на самом деле просто отражают две ипостаси школьной жизни, которая сводится к учебе и так называемым внеклассным мероприятиям. И, между прочим, это означает, что информатика в школе становится "надпредметом", поскольку только в ее рамках как учебного предмета возможно выйти на такие ответы, реализующие ее интегративную функцию.

Впрочем, если вспомнить основные задачи внедрения автоматизированных систем в производство или управление, они точно такие же:

- Увеличить производительность труда и/или
- Изменить характер труда в сторону его большей привлекательности для человека.

Иными словами, до определенной степени пятая задача в терминах А.Ю. Уварова включает в себя и решение стандартной задачи внедрения АСУ в школе. Новизна заключается лишь в том, что это должно делаться не на кружках или факультативах, а в процессе самого преподавания курса.

Вообще интегративный потенциал современной информатики превосходит все, с чем до сей поры сталкивалось человечество. Видимо, стало уже расхожей фразой, что "будущее общество будет не технократическим, а информационным и высокотехнологическим". Стремительное развитие информатики приводит де-факто к тому, что она становится "наукой наук", и отнюдь не в философском смысле этого слова, а в самом что ни на есть буквальном.

И один из фундаментальных столпов этого развития — создание Единой Мировой Информационной Среды. К сожалению, этот факт в России мало кто способен по достоинству оценить. Большинство лишь недоуменно хмыкнуло над очередной "экстравагантной" выходкой фирмы Microsoft, превратившей Windows 98 в "сплошной Интернет"...

Итак, слово сказано. И теоретически, и технологически в Большой Информатике давно уже есть средство тотальной интеграции. Дело за небольшим — перенести технологии Интернета (или, что то же самое, Интранета — информационной среды одного предприятия или учреждения на базе локальной сети) в школу.

Но ведь американские школы именно этим и занимаются! Фактически они используют Большой Интернет в качестве своей школьной информационной среды, подкрепляя это разнообразными учебными компакт-дисками. К рассмотренным выше убогим реалиям современной российской школы можно добавить лишь то, что англоязычный Интернет НА ДВА ПОРЯДКА более богат, чем русскоязычный. Что же касается материалов, годных для образовательных целей, эта разница еще существеннее.

По счастью, американская школа фактически не имеет такого предмета, как информатика в нашем понимании. Стало быть, если бы нам удалось в рамках Единой Школьной Информационной Среды решить 1, 2, 4 и 5-ю задачи в классификации А.Ю. Уварова, то мы бы получили систему интеграции образовательной информации, ничуть не уступающую американской, а в некотором смысле даже ее превосходящую. Представляется, что на ее базе на самом деле можно было бы решать задачи принципиальной переделки самого характера образования.

Именно такая среда воистину является "виртуальной школой", к созданию которой необходимо стремиться.

Безусловно, читатели вправе усомниться в том, что "виртуальная школа" — нечто большее, чем прекраснодушные ни к чему не обязывающие измышления о "виртуальных красивостях", основанные на не менее "виртуальных предпосылках".

Представляется, что пришла пора вспомнить о "Многотрудной истории..." и подробно описать реально существующую "виртуальную школу".

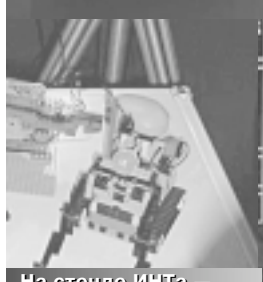
Но об этом — в следующей статье.

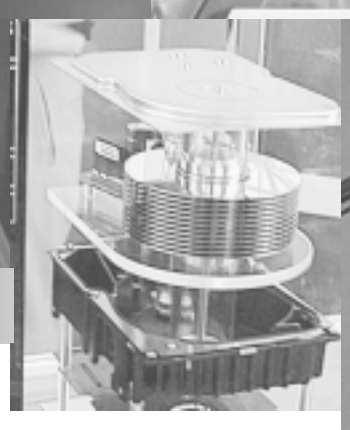
ВОКРУГ ПРЕДМЕТА

«Комтек-99»: картинки с выставки

Одетые в бело-голубой пластик Маки привлекали внимание.

Широкий спектр устройств ввода информации в компьютер.





На стенде ИНТа — программируемые роботы.

Замечательные наглядные пособия. Слева — «препарированная» «Барракуда».

ОБЪЕДИНЕНИЕ ПЕДАГОГИЧЕСКИХ ИЗДАНИЙ "ПЕРВОЕ СЕНТЯБРЯ"

Первое сентября
А.С. Соловейчик
индекс подписки — 32024

Английский язык
Е.В. Громушкина
индекс подписки — 32025

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
Gerolf Demmel
индекс подписки — 32292

Русский язык
Л.А. Гончар
индекс подписки — 32383

Спорт в школе
Н.В. Школьникова
индекс подписки — 32384

Управление школой
Н.А. Широкова
индекс подписки — 32652

Физика
Н.Д. Козлова
индекс подписки — 32032

Химия
О.Г. Блохина
индекс подписки — 32034

Школьный психолог
М.Н. Сартан
индекс подписки — 32898

Гл. редактор
С.Л.Островский
Зам. гл. редактора
Е.Б.Докшицкая

Редакция:
Л.Н.Картвелишвили,
Ю.А.Соколинский,
Н.Л.Беленькая,
Н.П.Медведева
Дизайн и компьютерная верстка:
Н.И.Пронская
Корректоры:
Е.Л.Володина,
С.М.Подберезина

Отпечатано с готовых диапозитивов редакции в ОАО ПО «Пресса-1», 125865, ГСП, Москва, ул. Правды, 24.

Тираж 7000 экз.

Заказ №

121165, Москва, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
для предприятий и организаций 32591
комплекта приложений 32744

Internet: inf@1september.ru
Fidonet: 2:5020/69.32
WWW: http://www.1september.ru

Тел. (095)249 3138, 249 3386, факс (095)249 3184