«Мир информатики»
Журнал для тех, у кого информатика – любимый школьный предмет
Выпуск № 79, июнь 2023 г.

ЕГЭ

Д.М. Златопольский

Методика выполнения задания 14 демонстрационного варианта ЕГЭ по информатике 2023 года

Общие вопросы

В Спецификация контрольных измерительных материалов для проведения в 2023 году единого государственного экзамена по информатике [1] в качестве проверяемого элемента содержания применительно к заданию 14 указывается «Знание позиционных систем счисления».

Уровень сложности задания – П (повышенный).

Макс. балл за выполнение задания – 1.

Условие [2]

Операнды арифметического выражения записаны в системе счисления с основанием 15.

$123x5_{15} + 1x233_{15}$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 15-ричной системы счисления.

Определите *наименьшее* значение x, при котором значение данного арифметического выражения кратно 14. Для найденного значения x вычислите частное от деления значения арифметического выражения на 14 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

Решение

Решение данной задачи[1] облегчается тем, что в условии использована 15-ричная система счисления и речь идет о кратности числу 14. Дело в том, что в любой системе с основанием q признак делимости числа на $q - 1$ аналогичен признаку делимости десятичных чисел на 9 (вспомните его!), то есть q-ричное число кратно числу $q - 1$, если сумма его цифр кратна $q - 1$.

В нашем случае $q = 15$. Сумма цифр операндов заданного арифметического выражения равна $20 + 2x$. Наименьшее значение x, при котором эта сумма кратна 14, – 4.

Это значит, что операнды равны 12345 и 14233, а их 15-ричная сумма – 26578. Переведя её в десятичную систему (удобно использовать электронную таблицу – см. ниже), получим 122738. Частное от деления этого числа на 14 равно 8767.

Ответ: 8767.

[1] О других возможных вариантах экзамена см. дополнение к статье.

Методика перевода целых недесятичных чисел в десятичную систему счисление с помощью электронной таблицы

Оформим лист электронной таблицы следующим образом. В первой строке будем записывать недесятичные цифры (последняя цифра – в столбце F, во второй – соответствующие весомости разрядов[2]:

	A	B	C	D	E	F	G
1	Цифры						
2	Весомость разрядов						
3							

Применительно к рассмотренному заданию лист примет вид:

	A	B	C	D	E	F	G
1	Цифры		2	6	5	7	8
2	Весомость разрядов		50625	3375	225	15	1
3							

Обратим внимание на то, что вручную значения весомостей рассчитывать и вводить в ячейки нерационально. Лучше в ячейку G2 ввести значение 1, а в ячейку F2 – формулу:

= 15^3 * G2

которую затем распространить (скопировать) на остальные ячейки строки 2.

После этого искомое десятичное значение можно быть получено в одной из ячеек с использованием функции СУММПРОИЗВ:

= СУММПРОИЗВ(B1:G1;B2:G2)

Дополнение

Другие варианты задания и методы их решения

Выше уже отмечалось, в условии задания из [2] использована 15-ричная система счисления и речь идет о кратности числу 14. Возможны также варианты задания, в котором при системе счисления с основанием q речь может идти о делимости не на число q – 1. Кроме того, возможны варианты, в которых требуется найти максимальное из некоторых значений.

Пример

Операнды арифметического выражения записаны в системе счисления с основанием 12.

$5x361_{12} + 133x5_{12}$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 12-ричной системы счисления.

Определите наибольшее значение x, при котором значение данного арифметического выражения кратно 18. Для найденного значения x вычислите частное от деления значения арифметического выражения на 18 и укажите его в ответе в десятичной системе счисления.

В таких случаях описанная методика выполнения задания неприменима. Возможны два основных метода решения:

1) в среде электронной таблицы;

2) путём разработки программы.

---

[2] На экзамене с целью экономии времени текст в столбце A можно не записывать.

[3] Значение 15 используется в рассматриваемом случае!

Решение в среде электронной таблицы

Приведем лист в готовом виде[4]:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Цифры первого операнда						Цифры второго операнда						
2	Значение x	5	x	3	6	1		1	3	3	x	5	Сумма	
3	0	103680	0	432	72	1		20736	5184	432	0	5	130542	
4	1	103680	1728	432	72	1		20736	5184	432	12	5	132282	
…														
13	10	103680	17280	432	72	1		20736	5184	432	120	5	147942	
14	11	103680	19008	432	72	1		20736	5184	432	132	5	149682	
15	Весомость	20736	1728	144	12	1		20736	1728	144	12	1		

Ясно, что рассмотрены все возможные значения цифры *x* в столбце A (их удобно получить, используя так называемое «автозаполнение ячеек»).

Весомости разрядов в строке 15 записываются аналогично тому, как это делалось выше. В ячейках диапазона B3:L14 записаны формулы для расчета произведения всех цифр операндов на весомости соответствующих разрядов. Причем для этого можно вручную ввести формулы только в две ячейки:

- в ячейку B3 вводится формула =B$2*B$15, которая затем копируется в ячейки, выделенные синим цветом;
- в ячейку C3 вводится формула =$A3*C$15, которая затем копируется в ячейки, выделенные желтым цветом.

(Обратите внимание на использование в формулах так называемых «смешанных» или «комбинированных» ссылок.)

В столбце M записаны формулы, суммирующие все произведение в каждой строке, а в столбце N – частное от деления суммы на 18. Максимальное целое значение частного в столбце N и будет являться искомым числом.

Решение с помощью программы

На школьном алгоритмическом языке (система программирования КуМир) программа, основанная на алгоритме решения задачи в среде электронной таблицы, имеет вид:

```
алг
нач цел x, сумма
  нц для x от 0 до 11 | Для всех возможных значений цифры x
    | Рассчитываем сумму произведений цифр на весомость разряда
    сумма := 5 * 12 ** 4 + x * 12 ** 3 + 3 * 12 ** 2 + 6 * 12 + 1 +
             1 * 12 ** 4 + 3 * 12 ** 3 + 3 * 12 ** 2 + x * 12 + 5
    | Выводим частное от деления суммы на 18
    вывод нс, сумма/18
  кц
кон
```

где знак ** соответствует операции возведения в степень.

Можно также цифры с одинаковой весомостью разрядов суммировать:

```
сумма := (5 + 1) * 12 ** 4 + (x + 3) * 12 ** 3 +
         (3 + 3) * 12 ** 2 + (6 + x) * 12 + 6
```

Можно также переменную сумма не использовать, а рассчитывать и выводить на экран частное от деления «большого» выражения на 18.

[4] На экзамене тексты можно не записывать

Задание для самостоятельной работы

Операнды арифметического выражения записаны в системе счисления с основанием 12.

$25x63_8 + 143x5_8$

В записи чисел переменной *x* обозначена неизвестная цифра из алфавита восьмеричной системы счисления.

Определите наибольшее значение *x*, при котором значение данного арифметического выражения кратно 16. Для найденного значения *x* вычислите частное от деления значения арифметического выражения на 16 и укажите его в ответе в десятичной системе счисления.

Литература

1. Спецификация контрольных измерительных материалов для проведения в 2023 году единого государственного экзамена по информатике (Демоверсии, спецификации, кодификаторы 2023. Информатика). https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

2. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2023 года по информатике (Демоверсии, спецификации, кодификаторы 2023. Информатика). https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

Методика выполнения задания 23 демонстрационных вариантов ЕГЭ по информатике

Д.М. Златопольский

1. Общие вопросы

В качестве проверяемого элемента содержания для задания 23 в Спецификации [1] указывается: «Умение анализировать результат исполнения алгоритма, содержащего ветвление и цикл».

Уровень сложности задания – повышенный.

Максимальный балл за выполнение задания – 1.

Примерное время выполнения задания (мин.) – 8.

Интересно, что в [1] говорится о том, что использование специализированного программного обеспечения для выполнения данного задания не требуется, хотя, по нашему мнению, это не означает возможности его использования.

2. Примеры заданий 23 из демонстрационных вариантов экзамена

Приведем примеры заданий 23 из демонстрационных вариантов ЕГЭ по информатике нескольких последних лет (с 2020–2023 гг.].

2.1. Задание 2020[5], 2021 и 2022 гг. [2]

Условие

Исполнитель преобразует число на экране.

У исполнителя есть две команды, которым присвоены номера:

```
1. Прибавить 1
2. Умножить на 2
```

Программа для исполнителя – это последовательность команд.

[5] В демонстрационном варианте экзамена 2020 года соответствующее задание имело номер 22.

Сколько существует программ, для которых при исходном числе 1 результатом является число 20, при этом траектория вычислений содержит число 10?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы 121 при исходном числе 7 траектория будет состоять из чисел 8, 16 и 17.

2.2. Задание 2023 года [3]

Условие

Исполнитель преобразует число на экране.

У исполнителя есть две команды, которые обозначены латинскими буквами:

```
A. Прибавить 1
B. Умножить на 2
```

Программа для исполнителя – это последовательность команд.

Сколько существует программ, для которых при исходном числе 1 результатом является число 35, при этом траектория вычислений содержит число 10 и не содержит 17?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы ABA при исходном числе 7 траектория будет состоять из чисел 8, 16 и 17.

3. Задания простого типа

Анализ показывает, что для решения указанных задач следует уметь решать более простые задачи, которые могут быть двух типов. Обсудим их.

3.1. Задания простого типа 1

Условие (в упрощенном виде)

У исполнителя есть две команды, которые обозначены латинскими буквами[6]:

```
A. Прибавить 1
B. Умножить на 2
```

Сколько существует программ, для которых при исходном числе 1 результатом является число <указывается значение>?

***

Задания такого типа будем называть «задачи “1-конечное”». Заметим, что они являются частной задачей приведенных выше заданий демонстрационных вариантов экзамена, а также могут являться самостоятельной задачей.

Решение

Задание может быть выполнено различными методами[7].

Один из рациональных методов выполнения задания – с помощью таблицы. Рассуждения следующие.

Если нужно получить число 2 (обозначим такую задачу 1 → 2), то возможны два варианта программы:

1) A (одна команда A);

2) B (одна команда B).

Пусть нужно получить 3. Так 3 – нечетное число, то оно может быть получено только из числа 2 одним способом. Общее количество вариантов для 1 → 2 мы знаем (2). Значит, и для задачи 1 → 3 ответ равен 2.

---

[6] Возможны также аналогичные задания, в которых командам присвоены номера (см. выше).

[7] В их числе – использование электронных таблиц с записью довольно сложных формул (см., например, [4].

Задача 1 → 4. Число 4 – четное, то есть оно может быть получено двумя способами – «через» 2 и «через» 3. Количество вариантов для каждого случая мы уже знаем (2 и 2, соответственно), то есть для задачи 1 → 4 ответ равен 2 + 2 = 4.

Искомые значения для бóльших результатов (и логика их получения) приведена в таблице:

Число	Может быть получено «через»	Общее количество вариантов
2	1	2
3	2	2
4	2 и 3	2 + 2 = 4
5	4	4
6	3 и 5	2 + 4 = 6
7	6	6

Конечно, описанный метод удобно применять, когда число, которое нужно получить, – небольшое.

Универсальные методы выполнения задания заключаются в использовании компьютерных программ.

Первая программа основана на следующих рассуждениях.

Некоторое нечетное число может быть получено только из числа, на 1 меньшего, с помощью команды А (`Прибавить 1`). Это означает, что количество программ для получения нечетного числа равно количеству программ для предыдущего числа (в каждую из них добавится команда А).

Некоторое четное число может быть получено двумя способами:

1) из числа, на 1 меньшего;

2) из числа, в 2 раза меньшего (по команде `Умножить на 2`).

Следовательно, общее количество программ для получения нечетного числа равно сумме количеств программ для двух указанных «предшествующих» чисел числа – в каждую из них добавится:

- команда А в первом случае;
- команда В во втором случае.

Сказанное означает, что в нашей компьютерной программе при расчетах придется обращаться в ранее рассчитанным значениям[8]. В связи с этим все уже рассчитанные значения целесообразно хранить в массиве. При описании программ будем использовать школьный алгоритмический язык (система программирования КуМир). Русский синтаксис этого языка делает приводимые программы максимально понятными и легко переводимыми на любой другой язык программирования. В программе на этом языке примем имя этого массива – *м*, а размер опишем с «запасом»:

```
алг Подсчет_количества_программ
нач цел к, цел таб м[2:100], цел i
```

Примечания

1. *к* – заданное конечное число.

2. 2 – начальный индекс элементов массива (в нем будет храниться количество программ для получения числа 2).

---

[8] В таких случаях говорят об использовании так называемых «рекуррентных соотношениях» – формулах, выражающих очередной член последовательности через один или несколько предыдущих членов. Например, рекуррентным соотношением (или рекуррентной формулой) является формула для расчета очередного члена арифметической прогрессии: $a_i = a_{i-1} + d$ (каждый последующий член равен предыдущему, увеличенному на разность прогрессии).

Прежде чем представлять всю программу, заметим, что для чисел 2 и 3 количество программ для их получения мы знаем (их можно определить в уме), а для остальных чисел используем указанные выше закономерности расчетов.

Итак, программа:

```
алг Подсчет_количества_программ
нач цел к, цел таб м[2:100], цел i
  | Ввод конечного числа к
  ввод к
  | Заполнение массива
  | Известные значения
   м[2] := 2
   м[3] := 2
  | Расчеты для остальных чисел
  нц для i от 4 до к
    если mod(i, 2) = 1
      то
         м[i] :=  м[i - 1]
      иначе
         м[i] :=  м[i - 1] +  м[div(i, 2)]
    все
  кц
  | Вывод ответа
  вывод нс, м[к]
кон
```

где *mod* – функция, определяющая остаток от деления её первого параметра на второй, *div* – функция, возвращающая целую часть частного от деления (в других языках программирования для этих расчетов используются не функции, а специальные операции).

Фрагмент программы в теле оператора цикла можно несколько сократить, учитывая, что в обеих ветвях команды **если** есть одно и то же значение (*м*[*i* - 1):

```
…
| Расчеты для остальных чисел
нц для i от 4 до к
  | Значение м[i - 1] учитываем в любом случае
  м[i] := м[i - 1]
  | Для четного к учитываем и второе значение
  если mod(i, 2) = 0
    то
      м[i] := м[i] + м[div(i, 2)]
  все
кц
```

Второй вариант программы использует рекурсивную функцию (напомним, что рекурсивными называют функции, которые обращаются как к вспомогательным к самим себе, но с другими значениями их параметров – см. рубрику «Школа программирования»).).

Рекурсивная функция *Кол(к)*, рассчитывающая количество программ, которыми можно получить число *к*, основана на следующих рассуждениях.

Если *к* – нечетное число, то значение функции совпадает с аналогичным значением для числа *к* – 1, в противном случае оно равно сумме двух значений *Кол(к – 1)* и *Кол(div(к, 2)*). Как принято в рекурсивных функциях, вызовы (обращения) к этой же функции с другими параметрами должны проводиться по условию, которое когда-то станет ложным. Таким условием является следующее: *к* > 2 (при *к* = 2 значение функции известно – оно равно 2).

Соответствующая функция:

```
алг цел Кол2(арг цел к)
нач
  если к > 2
    то | Рекурсивные вызовы функции
      если mod(к, 2) = 1
        то
          знач := Кол2(к - 1) | Значение функции
        иначе
          знач := Кол2(к - 1) + Кол(div(к, 2))
      все
    иначе | Известное значение функции
      знач := 2
  все
кон
```

Как и первую программу, её можно короче:

```
алг цел Кол3(арг цел  к)
нач
  если к > 2
    то
      знач := Кол3(к - 1)
      если mod(к, 2) = 1
        то
          знач := знач + Кол3(div(к, 2))
      все
    иначе
      знач := 2
  все
кон
```

Главная программа:

```
алг Подсчет_количества_программ_3
нач цел к
  ввод к
  вывод нс, Кол3(к)
кон
```

Возможен также рекурсивный вариант, в котором рекурсивные вызовы происходят не с меньшими, а с большими значениями параметров функции[9]. Рассуждения в данном случае такие. От некоторого начального значения *н* можно перейти либо к числу *н* + 1, либо к числу *н* × 2. Для двух указанных чисел дальнейшие действия аналогичны. Это значит, что рекурсивные вызовы для определения количества программ должны быть такими:

```
алг цел Кол4(арг цел н, к)  | Два параметра функции!
…
  знач := Кол4(н + 1, к) + Кол4(н * 2, к)
```

Но при приведенной записи рекурсивные вызовы будут происходить «бесконечно», и их надо ограничить. Можно сказать, что эти вызовы должны происходить, только если *н* < *к*:

```
алг цел Кол4(арг цел н, к)  | Два параметра функции!
нач
  если н < к
```

[9] Автор идеи метода – А. Сидоров [4].

```
    то
      знач := Кол4(н + 1, к) + Кол4(н * 2, к)
```

А что должно быть в противном случае?

Когда первый параметр функции станет равным или больше второго, то рекурсивные вызовы надо прекратить. Какие значения в вызывающую функцию должна вернуть текущая функция в таких случаях? Когда *н* стало равно *к*, то это значит, что получился еще один вариант программы (с последней командой А или В). Когда же очередное *н* превысило *к*, то и тогда вызовы надо, естественно, прекратить, а значение функции принять равным 0.

Вся соответствующая функция имеет вид:

```
алг цел Кол4(арг цел н, к)
нач
  если н < к
    то
      знач := Кол4(н + 1, к) + Кол4(н * 2, к)
    иначе
      если н = к
        то
          знач := 1
        иначе  | н > к
          знач := 0
      все
  все
кон
```

Главная программа:

```
алг Подсчет_количества_программ_4
нач цел к
  ввод к
  вывод нс, Кол4(1, к)
кон
```

Преимуществом последнего варианта функции заключается в том, что не нужно проводить делать проверку чисел на четность и использовать операцию целочисленного деления. Некоторую его громоздкость устраняет такой вариант оформления:

```
алг цел Кол4(арг цел н, к)
нач
  если н = к
    то
      знач := 1
  все
  если н > к
    то
      знач := 0
  все
  знач := Кол4(н + 1, к) + Кол4(н * 2, к)
кон
```

Задание для самостоятельной работы

У исполнителя есть две команды, которые обозначены латинскими буквами:

A. Прибавить 1

B. Умножить на 2

Сколько существует программ, для которых при исходном числе 1 результатом является число 21?

3.2. Задания простого типа 2

Условие

У исполнителя есть две команды, которые обозначены латинскими буквами:

A. Прибавить 1
B. Умножить на 2

Сколько существует программ, для которых при исходном числе 1 результатом является число <*указывается значение*> и при этом траектория вычислений программы не содержит число <*указывается значение*>?

Задания такого типа будем называть «задачи “1-конечное (кроме)”». Они являются частной задачей приведенных выше заданий демонстрационных вариантов экзамена [2021, 2023], а также могут являться самостоятельной задачей.

Самый удобный и логичный способ выполнения таких заданий основан на программе Подсчет_количества_программ_4. Необходимо в рекурсивную функцию *Кол4* включить величину *кроме*. В результате она примет вид:

```
алг цел Кол5(арг цел н, к, кроме)
нач
  если н > к
    то
      знач := 0
  все
  если н = кроме    | Новый фрагмент
    то
      знач := 0     | Дальше также рекурсивных вызовов на будет
  все
  если н = к
    то
      знач := 1
  все
  знач := Кол5(н + 1, к, кроме) + Кол5(н * 2, к, кроме)
кон
```

Можно первые два случая объединить:

```
алг цел Кол5(арг цел н, к, кроме)
нач
  если н > к или н = кроме
    то
      знач := 0
  все
  если н = к
    то
      знач := 1
  все
  знач := Кол5(н + 1, к, кроме) + Кол5(н * 2, к, кроме)
кон
```

Основная часть программы:

```
алг Подсчет_количества_программ_5
нач цел к, кроме
  ввод к
  ввод кроме
  вывод нс, Кол5(1, к, кроме)
кон
```

Задание для самостоятельной работы

У исполнителя есть две команды, которые обозначены латинскими буквами:

```
A. Прибавить 1
B. Умножить на 2
```

Сколько существует программ, для которых при исходном числе 1 результатом является число 33 и при этом траектория вычислений программы не содержит число 13?

4. Решение задач из демонстрационных вариантов экзамена

4.1. Задание из [2]

В задании (см. раздел 2) спрашивается: «Сколько существует программ, для которых при исходном числе 1 результатом является число 20, при этом траектория вычислений содержит число 10?».

Рассуждения такие.

Все программы обязательно должны обеспечивать расчет числа 10. Количество таких программ можно определить одним из рассмотренных выше методов. Оно равно 14.

Вторая часть задачи – определение количества программ для получения числа 20 при начальном числе 10. В данном случае она может быть решена без использования программ. Так как 20 в два раза больше числа 10, то оно может быть получено двумя способами:

1) при умножении на 2 (в каждую из программ с траекторией, включающей 10, добавится команда B);

2) при 10-кратном увеличении на 1 (в каждую из программ с траекторией, включающей 10, добавится 10 команд A).

Это значит, что общее искомое количество программ равно 14 × 2 = 28.

Ответ: 28.

Сделанные рассуждения позволяют сформулировать общее правило для решения задачи вида: «Сколько существует программ, для которых при исходном числе 1 результатом является число *к*, при этом траектория вычислений содержит число *пром*?». Оно заключается в следующем:

1) рассчитывается количество программ для получения из числа 1 числа *пром*;

2) определяется количество программ для получения из числа *пром* числа *к*;

3) искомый результат равен произведению двух найденных значений.

Условимся такие задачи называть «задачи “1-промежуточное-конечное”».

Если в условии фигурируют два обязательных числа в траектории программы, то искомый результат определяется как произведение трех предварительно рассчитанных значений.

Задание для самостоятельной работы

У исполнителя есть две команды, которые обозначены латинскими буквами:

```
A. Прибавить 1
B. Умножить на 2
```

Сколько существует программ, для которых при исходном числе 1:

а) результатом является число 37 и при этом траектория вычислений программы содержит число 15?

б) результатом является число 30 и при этом траектория вычислений содержит числа 10 и 21?

4.2. Задание из [3]

В условии этого задания есть требование о том, что траектория вычислений не должна содержать число 17. Как учесть это требование? Можно ли подсчитать и исключить количество *всех* программ, траектория которых включает число 17? – Нет, нельзя. Так как

траектории должны обязательно включать число 10, то исследовать нужно программы, в которых исходным является число 10, то есть, как и в предыдущем задании, нужно определить количество программ для получения из числа 10 числа 35, кроме тех, траектория которых содержит число 17. Как?

Можно рассуждать так.

Из числа 10 число 35 можно получить:

- по 25 командам А;

- применив команду В к некоторым числам, а затем продолжить программу, записав в ней ряд команд А.

Но в первом случае в траектории программы будет число 17, что недопустимо, то есть соответствующая программа не подходит.

К каким числам может быть применена команда В во втором случае? – К числам 10, 11, …, 16. Таких чисел – 7, то есть количество программ, которое мы должны найти на этом этапе, равно 7, а общее количество программ, искомых при выполнении задания – 14 × 7 = 98 (каждую из 14 программ для получения из числа 1 числа 10, можно дополнить семью способами).

Итак, ответ: 98.

Можно так сформулировать общее правило для решения задачи вида: «Сколько существует программ, для которых при исходном числе 1 результатом является число *к*, при этом траектория вычислений программ содержит число *пром* и не содержит число *кроме*»[10]:

1) рассчитывается количество программ для получения из числа 1 числа *пром*;

2) определяется количество программ для получения из числа *пром* числа *к* без учета программ, траектория которых содержит число *кроме*;

3) искомый результат равен произведению двух найденных значений.

Но как определить второе количество? Анализ показывает, что для этого может быть использована программа с рекурсивной функцией Кол5, в основной части которой вызов функции должен происходить с начальным значением, равным известному числу *пром*:

```
алг
нач цел пром, к, кроме
  ввод пром
  ввод к
  ввод кроме
  вывод нс, Кол5(пром, к, кроме)
кон
```

При этом напомним, что при решении задач типа «Сколько существует программ, для которых при исходном числе 1 результатом является число *к*, при этом траектория вычислений программ содержит число *пром* и не содержит число *кроме*» приходится определять два значения:

- количество программ, для которых при исходном числе 1 результатом является число *пром*;

- количество программ, для которых при исходном числе *пром* результатом является число *к* и при этом траектория вычислений программ не содержит число *кроме*,

которые затем перемножаются.

Чтобы для каждой частной задачи не разрабатывать отдельную функцию, можно и для нахождения первого значения использовать функцию *Кол5* с условным значением параметра *кроме*, равного 0:

```
алг
нач цел пром, к, кроме
  ввод пром
```

[10] Такие задачи условимся называть «задачи “1-промежуточное-конечное (кроме)”».

```
    ввод к
    ввод кроме
    вывод нс, Кол5(1, пром, 0) * Кол5(пром, к, кроме)
кон
```

Задание для самостоятельной работы

У исполнителя есть две команды, которые обозначены латинскими буквами:

```
A. Прибавить 1
B. Умножить на 2
```

Сколько существует программ, для которых при исходном числе 1:

а) результатом является число 37 и при этом траектория вычислений программы содержит число 12 и не содержит число 30?

в) результатом является число 35 и при этом траектория вычислений программы содержит число 11 и не содержит число 19?

в) результатом является число 33 и при этом траектория вычислений программы содержит число 9 и не содержит число 15?

5. Другие варианты задач с двумя командами

Выше были рассмотрены программы для исполнителя с двумя командами в его системе команд:

```
A. Прибавить 1
B. Умножить на 2
```

для решения задач четырех видов:

1) из числа 1 надо получить некоторое число *к* (задачи «1- конечное»);

2) из числа 1 надо получить некоторое число *к* и при этом траектория вычислений не должна содержать некоторое число *кроме* (задачи «1-конечное (кроме)»);

3) из числа 1 надо получить некоторое число *к* и при этом траектория вычислений содержит некоторое число *пром* (задачи «1-промежуточное-конечное»);

4) из числа 1 надо получить некоторое число *к* и при этом траектория вычислений должна содержать некоторое число *пром* и не должна содержать некоторое число *кроме* (задачи «1-промежуточное-конечное (кроме)»).

Нетрудно понять, что задачи с двумя командами:

```
A. Прибавить <значение 1>
Б. Прибавить <значение 2>
```

во всех четырех случаях решаются с минимальным изменением приведенных программ.

Задание для самостоятельной работы

У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 3

Сколько существует программ, для которых при исходном числе 1:

а) результатом является число 17?

б) результатом является число 19 и при этом траектория вычислений программы не содержит число 9?

в) результатом является число 41 и при этом траектория вычислений программы содержит число 15?

г) результатом является число 43 и при этом траектория вычислений программы содержит число 11 и не содержит число 25?

Возможны также задания с исполнителем с двумя командами, в которых:

- вместо начального числа 1 указано любое значение *н*;

- вместо команды Прибавить 1 используются команды Прибавить <*значение*> или Вычесть <*значение*>;
- вместо команды Умножить на 2 используются команды Умножить на <*значение*> или Разделить нацело на <*значение*>.

Анализ показывает, что для выполнения заданий рассмотренного типа универсальным является использование функций *Кол4* и *Кол5*[11], в которые вместо начального значения, равного 1, следует записать значение *н*, а также учесть все возможные новые команды.

Например, задача «1-конечное» становится задачей «начальное-конечное», то есть функция *Кол4* должна принять вид:

```
алг цел Кол4_новая(арг цел н, к)[12]
нач
  если н > к
    то
      знач := 0
    иначе
      если н = к
        то
          знач := 1
        иначе | Рекурсивные вызовы
         знач := Кол4_новая(н + увел, к) + Кол4_новая(н * множ, к)
      все
  все
кон
```

где *увел* – величина увеличения в команде Прибавить …, *множ* – значение множителя в команде Умножить … .

Если система команд исполнителя состоит из команд Вычесть <*значение*> и Разделить нацело на <*значение*>, то число-результат выполнения программ будет меньше начального числа, и в рекурсивной функции *Кол4* следует учесть:

- эти команды;
- возможность выхода за меньшее конечное число:

```
алг цел Кол4_новая(арг цел н, к)
нач
  если н < к | Новое условие
    то
      знач := 0
    иначе
      если н = к
        то
          знач := 1
        иначе | Рекурсивные вызовы
          знач := Кол4_новая(н - умен, к) +
                    Кол4_новая(div(н, делит), к)
        все
  все
кон
```

где *умен* – величина уменьшения в команде Вычесть …, *делит* – значение делителя в команде Разделить нацело на … .

---

[11] Кроме их преимуществ по сравнению с программой Подсчет_количества_программ и функцией *Кол_3*, указанных в пункте 3.1, в них упрощается учет известных значений до использования рекуррентных соотношений и рекурсивных вызовов.

[12] На экзамене, с целью сокращения времени, имя функции можно дать более коротким. Это замечание касается и имени функции *Кол5_новая*, которая будет использована далее.

Во всех случаях основная программа примет вид:

```
алг
нач цел н, к
  ввод н
  ввод к
  вывод нс, Кол4_новая(н, к)
кон
```

Решенная ранее задача «1-конечное (кроме)» становится задачей «начальное-конечное (кроме)», и поэтому функция *Кол5* принимает вид:

```
алг цел Кол5_новая(арг цел н, к, кроме)
нач
  если н > к или н = кроме
    то
      знач := 0
  все
  если н = к
    то
      знач := 1
  все
  знач := Кол5_новая(…, к, кроме) + Кол5_новая (…, к, кроме)
кон
```

где вместо многоточия используются действия, соответствующие командам исполнителя.

Для исполнителя с командами Вычесть <*значение*> и Разделить нацело на <*значение*>, то число-результат выполнения программ будет меньше начального числа, и в рекурсивной функции *Кол5* следует учесть:

- эти команды;
- возможность выхода за меньшее конечное число:

```
алг цел Кол5_новая(арг цел н, к, кроме)
нач
  если н < к или н = кроме | Новое первое простое условие
    то
      знач := 0
  все
  если н = к
    то
      знач := 1
  все
  знач := Кол5_новая(…, к, кроме) + Кол5_новая (…, к, кроме)
кон
```

Основная часть программы в этом случае:

```
алг
нач цел н, к, кроме
  ввод н
  ввод к
  ввод кроме
  вывод нс, Кол5_новая(н, к, кроме)
кон
```

Третья задача «1-промежуточное-конечное» превращается в задачу «начальное-промежуточное-конечное» и может рассматриваться как две отдельные задачи «начальное-промежуточное» и «промежуточное-конечное». Это значит, что искомое количество

программ в общем случае может быть определено как произведение двух значений функции *Кол4_новая*:

```
алг
нач цел н, к, пром
  ввод н
  ввод к
  ввод пром
  вывод нс, Кол4_новая(н, пром) * Кол4_новая(пром, к)
кон
```

И, наконец, задача «1-промежуточное-конечное (кроме)» становится задачей «начальное-промежуточное-конечное (кроме)», которая решается с использованием двух значений одной и той функции *Кол5_новая*:

```
алг
нач цел н, к, пром, кроме
  ввод н
  ввод к
  ввод пром
  ввод кроме
  вывод нс, Кол5_новая(н, пром, 0) * Кол5_новая(пром, к, кроме)
кон
```

В заключение заметим, что возможны также задания в которых фигурируют исполнители с тремя командами, например:

```
1. Прибавить 1
2. Прибавить 2
3. Умножить на 3
```

Методика выполнения таких заданий по сути аналогична рассмотренным методикам с использованием функций *Кол4_новая* и *Кол5_новая*.

Задания для самостоятельной работы

1. У исполнителя есть две команды, которым присвоены номера:

```
1. Прибавить 1
2. Умножить на 3
```

Сколько существует программ, для которых при исходном числе 2 результатом является число 20?

2. У исполнителя есть две команды, которым присвоены номера:

```
1. Прибавить 3
2. Умножить на 3
```

Сколько существует программ, для которых при исходном числе 5 результатом является число 45 и при этом траектория вычислений программ не содержит число 14?

3. У исполнителя есть две команды, которым присвоены номера:

```
1. Прибавить 2
2. Умножить на 3
```

Сколько существует программ, для которых при исходном числе 3 результатом является число 60 и при этом траектория вычислений программ содержит число 19?

4. У исполнителя есть две команды, которым присвоены номера:

```
1. Прибавить 3
2. Умножить на 2
```

Сколько существует программ, которые преобразуют исходное число 3 в число 45, и при этом траектория вычислений программ содержит число 12 и не содержит числа 25?

5. У исполнителя есть две команды, которым присвоены номера:

```
1. Прибавить 3
2. Умножить на 2
```

Сколько существует программ, которые преобразуют исходное число 3 в число 45, и при этом траектория вычислений программ содержит число 25 и не содержит числа 8?

6. У исполнителя есть две команды, которым присвоены номера:

```
1. Вычесть 1
2. Разделить нацело на 2
```

Сколько существует программ, которые исходное число 28 преобразуют в число 5?

7. У исполнителя есть две команды, которым присвоены номера:

```
1. Вычесть 2
2. Разделить нацело на 3
```

Сколько существует программ, которые исходное число 44 преобразуют в число 7 и при этом траектория вычислений программ не содержит число 16?

8. У исполнителя есть две команды, которым присвоены номера:

```
1. Вычесть 5
2. Разделить нацело на 2
```

Сколько существует программ, которые исходное число 66 преобразуют в число 8 и при этом траектория вычислений программ содержит число 25?

9. У исполнителя есть две команды, которым присвоены номера:

```
1. Вычесть 3
2. Разделить нацело на 3
```

Сколько существует программ, которые исходное число 86 преобразуют в число 5 и при этом траектория вычислений программ содержит число 20 и не содержит число 28?

Литература

1. Спецификация контрольных измерительных материалов для проведения в 2023 году единого государственного экзамена по информатике (Демоверсии, спецификации, кодификаторы 2023. Информатика). https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

2. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2022 года по информатике (Демоверсии, спецификации, кодификаторы 2022. Информатика). https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

3. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2023 года по информатике (Демоверсии, спецификации, кодификаторы 2023. Информатика). https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

4. Сайт К.Ю. Полякова. https://kpolyakov.spb.ru/school/ege.htm

О рекурсивных функциях

Обращаем внимание читателей, что в задания, связанные с рекурсивными функциями, представлены в демонстрационных вариантах ЕГЭ по информатике.

***

«Рекурсивными»[13] называют функции, которые используют (вызывают) как вспомогательную саму себя. Непонятно? Тогда – по порядку.

Что делают известные вам стандартные функции – sin, sqrt и другие при их использовании в программах? – Возвращают какое-то значение (результат расчетов или т. п.). Так вот, с такой же целью можно также создавать и использовать собственные, нестандартные, функции. Пусть, например, в программе надо рассчитать значения x:

$$x = \frac{2+\sqrt{2}}{5+\sqrt{5}} + \frac{5+\sqrt{5}}{13+\sqrt{13}} + \frac{11+\sqrt{11}}{8+\sqrt{8}} + \frac{14+\sqrt{14}}{7+\sqrt{7}}$$

Видно, что в выражении имеются похожие фрагменты – дроби. Желательно для расчета дробей оформить функцию, которая вычисляла бы значения отдельных дробей вида $\frac{a+\sqrt{a}}{b+\sqrt{b}}$ при любых значениях a и b, а затем использовать ее для расчетов.

Правила оформления функций в разных языках программирования различаются. В школьном алгоритмическом языке общий вид функций такой[14]:

```
алг <тип результата> <имя функции> (<список формальных параметров>)
нач <описания переменных величин используемых в функции>
  <тело функции>
кон
```

причем последним оператором тела функции должен быть оператор присваивания, в левой части которого должно быть указано служебное слово знач.

В нашем случае функция для расчета значения дробей указанного чуть выше вида оформляется так:

```
алг вещ Др(арг цел a, b)
нач
  знач := (a + sqrt(a))/(b + sqrt(b))
кон
```

Как и стандартные, функция, созданная программистом, используется в правой части оператора присваивания[15]. Для этого надо указать ее имя, а в скобках – значения фактических параметров:

```
x := Др(2,5) + Др(5,13) + Др(13, 8) + Др(14,7)
```

Видно, что благодаря использованию функции, оператор для расчета значения x гораздо компактнее такого варианта:

---

[13] От латинского *recursio* – возвращение.

[14] С правилами оформления собственных функций в языке программирования, который вы используете, ознакомьтесь самостоятельно.

[15] Конечно, нестандартная функция может быть записана в условиях (логических выражениях), в операторах вывода на экран и т. д.

```
      x := (2 + sqrt(2))/(5 + sqrt(5)) + (5 + sqrt(5))/(13 + sqrt(13)) +
(13 + sqrt(13))/(8 + sqrt(8)) + (14 + sqrt(14))/(7 + sqrt(7))
```

Кроме того, в последнем варианте выше вероятность появления ошибки при записи оператора.

Из описания функции видно, что в ней в качестве вспомогательной используется стандратная функция sqrt. Так вот, в собственной функции в качестве вспомогательной может быть использована эта же самая функция (такое использование и является примером рекурсии, обращению функции к самой себе). В качестве примера приведем функцию для расчета факториала натурального числа *n* (факториал числа *n*, обозначаемый *n*!, равен $1 \times 2 \times 3 \times \ldots \times n$). Такую функцию можно создать, имея в виду, что $n! = (n-1)! \times n$. На школьном алгоритмическом языке соответствующая функция имеет вид:

```
      алг цел Факториал(цел n)
нач
        знач := Факториал(n - 1) * n        | Рекурсивный вызов функции
Факториал
        | Служебное слово знач означает "значение функции"
      кон
```

На самом деле эта функция оформлена не по правилам и при выполнении программы появится сообщение об ошибке. Дело в том, что, при каждом вызове функции для нее отводится место в оперативной памяти, которое «освобождается» после завершения ее (функции) работы. Но если во вспомогательной функции имеется рекурсия, то вызовы функции будут продолжаться до тех пор, когда место в памяти будет исчерпано. Чтобы устранить этот недостаток, необходимо так оформлять функции, чтобы рекурсивные вызовы осуществлялись по условию, которое когда-то станет ложным. Например, для приведенной функции это условие записывается так:

```
    алг цел Факториал(цел n)
    нач
      если n > 1
        то
          | Рекурсивный вызов функции Факториал
          знач := Факториал(n - 1) * n
        иначе | Значение функции известно (рекурсивного вызова нет)
          знач := 1
      все
    кон
```

или

```
      алг цел Факториал(цел n)
      нач
        если n = 1
          то
            знач := 1   | Известное значение функции
          иначе         | Рекурсивный вызов функции Факториал
            знач := Факториал(n - 1) * n
        все
      кон
```

Алгоритм вычисления значения функции F(n), использующий рекурсию, может быть описан по-другому – с указанием двух или более вариантов значения, например, в виде:

```
F(2) = 10
F(n) = F(n – 1) * 3 при при n > 2
```

или

```
F(3) = 5
F(n) = F(n – 1) + 5 при n > 3 и n, кратном 3
F(n) = F(n – 1) – 3 при n > 3 и n, не кратном 3
```

или

```
F(12) = 15
F(n) = F(n + 1) – 1 при n < 12

F(10) = 5
F(n) = F(n + 1) + 5 при n < 3 и n, кратном 2
F(n) = F(n + 1) – 3 при n < 3 и n, не кратном 2
```

и т. п.

Видно, что в двух первых случаях рекурсивные вызовы осуществляются с меньшими значениями аргумента, в остальных – с большими. Во всех случаях когда-то рекурсивные вызовы прекратятся (убедитесь в этом!) и начнется передача результатов в «вызывающую» функцию.

Пример работы программы при вычислении значения факториала числа 4 показан на рис. 1–2.

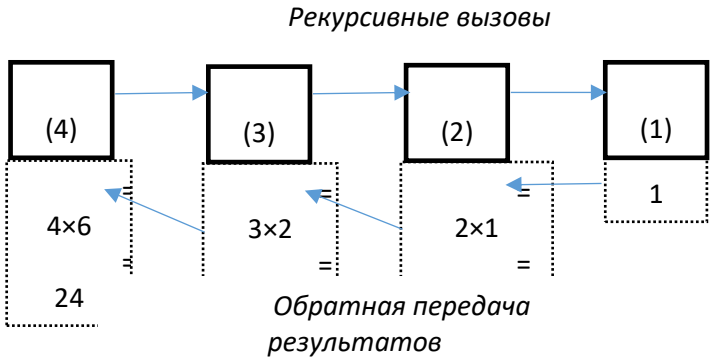


Рис. 1

F(n)	F(4)	F(3)	F(2)	F(1)
Значение функции	24 ← 4 × 6	6 ← 3 × 2	2 ← 2 × 1	1
n	4	3	2	1

Рис. 2

Ответ: 24.

Рассмотрим задачу, связанную с ЕГЭ косвенно, но очень показательную: «Определить 50-й член последовательности Фибоначчи (последовательность Фибоначчи образуют числа 1, 1, 2, 3, 5, 8, 13, ...)».

Достаточно очевидный метод решения задачи такой – использовать массив из 50 элементов, записать в него значения двух первых элементов (равных 1), а затем последовательно вычислять остальные как сумму двух предшествующих значений:

```
алг Расчет
нач цел таб мас[1 : 50], цел к
  мас[1] := 1
  мас[2] := 1
  нц для к от 3 до 50
    мас[i] := мас[i – 2] + мас[i – 1]
  кц
  вывод мас[50]
кон
```

Недостатком такого метода является то, что, хотя нас интересует лишь один член последовательности, в процессе выполнения программы компьютер вычисляет и хранит в памяти все члены последовательности от первого до 50-го.

Этого недостатка лишен второй метод вычислений – без использования массивов. Действительно, если для вычисления очередного члена последовательности Фибоначчи нужно знать только значения двух предыдущих, то можно не запоминать все члены последовательности в массиве, а хранить две величины:

`пред` – элемент, предшествующий очередному члену последовательности;

`предпред` – элемент, предшествующий элементу `пред`.

Кроме них, используем в программе величину `очер` – очередной рассчитываемый элемент последовательности.

Сначала имеем:

```
пред := 1
предпред := 1
```

Затем рассчитываем очередной (третий) элемент:

```
след := пред + предпред,
```

после чего «готовимся» к расчету следующего элемента:

```
предпред := пред  | Именно в
пред := след      | таком порядке!
```

и определяем его:

```
очер := пред + предпред
```

и т. д.

Ясно, что для определения 50-го члена последовательности необходимо провести 48 повторений описанных действий. Итак, программа:

```
алг Расчет_2
нач очер, пред, предпред, к
  пред := 1
  пред := 1
  нц для к от 1 до 48
    очер := пред + предпред
    предпред := пред
    пред := очер
  кц
  вывод очер
кон
```

Для решения обсуждаемой задачи может быть использована рекурсия. Вот как просто и логично выглядит рекурсивный вариант функции:

```
алг цел Фиб(цел к)
нач
  если к = 1 или к = 2
    то
      знач := 1
    иначе
      знач := Фиб(к - 2) + Фиб(к - 1)
  все
кон
```

Такое оформление функции полностью соответствует закону построения последовательности Фибоначчи – очередной элемент последовательности равен сумме двух предыдущих. При нем не требуется применять оператор цикла и думать над последовательностью расчета значений `пред` и `предпред`.

Обращаем внимание на то, в созданной функции имеют место два рекурсивных вызова.

Разработаем рекурсивные функции для случаев, когда функция задана с указанием двух или более вариантов значения (см выше).

Задача 1

Функция F(n), где n – натуральное число, задана следующими соотношениями:

```
F(1) = 10
F(n) = F(n – 1) + 3 при n > 1
```

Чему равно значение функции `F(10)`?

Решение

Соответствующая функция в программе, использующая рекурсию, оформляется следующим образом:

```
алг цел F(цел n)
нач
  если n <= 2
    то
      знач := 10
    иначе
      знач := F(n – 1) * 3
  все
кон
```

а основная программа для нахождения искомого значения:

```
алг цел F(цел n)
нач
  вывод F(10)
кон
```

Задача 2

Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

```
F(3) = 5
F(n) = F(n – 1) + 5 при n > 3 и n, кратном 3
F(n) = F(n – 1) – 3 при n > 3 и n, не кратном 3
```

Чему равно значение `F(16)`?

Вся программа решения:

```
алг цел F(цел n)
нач
  вывод F(10)
кон
алг цел F (цел n)
нач
  если n = 3
    то
      знач := 5
    иначе
      если mod(n, 3) = 0
        то
          знач := F(n – 1) + 5
```

```
            иначе
              знач := F(n - 1) - 3
          все
      все
кон
```

Задания для самостоятельной работы

1. Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

```
F(12) = 15
F(n) = F(n + 1) - 1 при n < 12
```

Чему равно `F(0)`?

2. Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

```
F(12) = 25
F(13) = 7
F(n) = F(n + 1) - 1 при n < 12
```

Чему равно `F(7)`?

История информатики

Один из первых компьютеров

Ответы, решения, разъяснения

к заданиям, опубликованным в журнале «Мир информатики» № 76 (март 2023 г.)

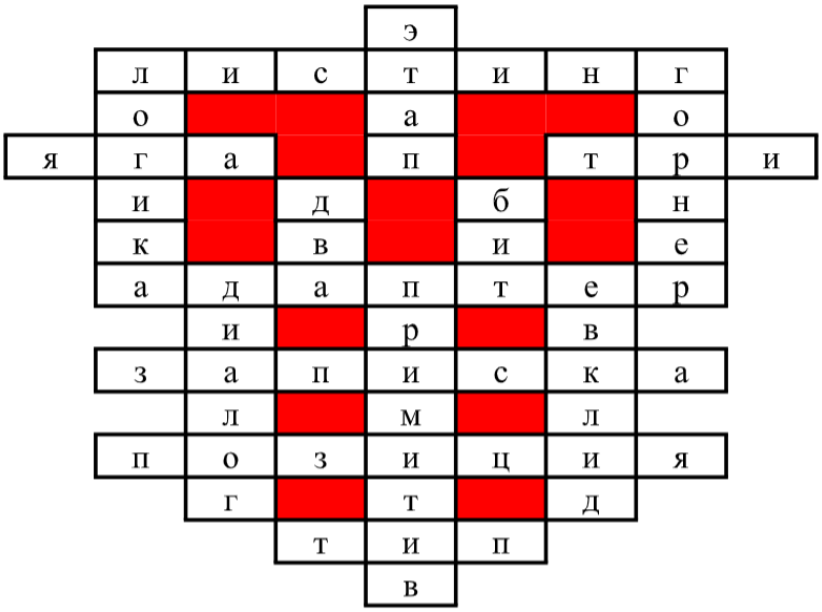
Головоломка «Шесть ребусов»

Ответы

Ребус 1 – ПЕРЕДАЧА. Ребус 2 – ОКНО. Ребус 3 – КЛАВИША. Ребус 4 – ИНТЕРНЕТ. Ребус 5 – КОМПЬЮТЕР. Ребус 6 – МОНИТОР.

Кроссворд

Ответы



Ответы и решения прислали:

– Авакиян Евгений, Горелкин Роман, Медведев Владимир, Пантюшкин Георгий, Попкова Мария, Резцова Анастасия, Резцов Владимир, Устинов Данил b Ягиева Гуля, средняя школа села Ириновка, Новобурасский р-н Саратовской обл., учитель **Брунов А.С.**;

— Аржанов Максим, Бирюков Максим и Шаронова Анна , г. Пенза, школа № 73, учитель **Диков А.В.**;

— Велихов Иван и Ярославцев Никита, г. Нижний Новгород, школа № 54, учитель **Дологова Г.А.**;

— Зубрицкий Антон и Поливанов Никита, Москва, школа № 1530 «Школа Ломоносова», учитель **Каменецкая Т.С.**;

— Иванова Виолетта и Левченко Ирина, средняя школа села Восточное Нижегородской обл., учитель **Макарчук В.А.**;

– Никифоров Никита, Владимирская обл., г. Александров, школа № 1, учитель **Ахсахалян С.Д.**;

—Норбеков Адам и Турков Андрей, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**

Спасибо всем!