М. С. Долинский
Гомельский государственный университет имени Франциска Скорины, Республика Беларусь

РЕШЕНИЕ РЕКУРСИВНЫХ ЗАДАЧ ПО ОПРЕДЕЛЕНИЮ

Аннотация
В статье описана методика изучения темы «Решение рекурсивных задач по определению» при подготовке школьников к олимпиадам по информатике. Технической основой является разработанная под управлением автора инструментальная система дистанционного обучения (http://dl.gsu.by).
Ключевые слова: рекурсия, обучение программированию, олимпиады по информатике, инструментальная система дистанционного обучения.
DOI: 10.32517/2221-1993-2020-19-2-60-66

Введение

С сентября 1996 года на базе средней школы № 27 г. Гомеля (Республика Беларусь), а сентября 1999 года дополнительно и на базе сайта дистанционного обучения DL.GSU.BY ведется работа [3, 4] по факультативному изучению информатики и программирования школьниками разных возрастов. Ключевая особенность этого обучения — его раннее начало, фактически с первого класса. Как следствие, уже в четвертом—шестом классах есть ученики, которым имеет смысл объяснять такие темы, как рекурсия. Поскольку традиционные подходы рассчитаны на обучение как минимум старшеклассников, то приходится их модифицировать — делать объяснение более простым и наглядным, медленнее продвигаться по учебному материалу, явно выделяя и обозначая все этапы этого продвижения.

Введение в решение задач с помощью рекурсии изложено в работе [1]. Решение задач с помощью рекурсивной генерации таких комбинаторных объектов, как множество всех подмножеств, сочетания, перестановки, перестановки с повторениями, размещения, описано в работе [2]. В данной статье читателям предлагается материал по теме «Решение рекурсивных задач по определению», включающий условия задач и примеры их решения.

Проверка решений осуществляется автоматически на сайте DL.GSU.BY.

Вдумчивым читателям предлагаем после чтения условия задачи попытаться решить ее самостоятельно; если же это сделать не удалось, рекомендуем предпринять аналогичную попытку перед чтением текста решения каждой задачи, воспользовавшись приведенными рекомендациями по алгоритму решения задачи.

Все задачи взяты из американских соревнований по информатике для школьников и студентов USACO (The United States of America Computing Olympiad — компьютерные олимпиады США). Первые две цифры в названии задачи означают год проведения олимпиады, затем три буквы — месяц, и последняя буква означает дивизион (уровень сложности задач): B — Bronze (легче), S — Silver (сложнее).

Задача 1. 11_JanB. Symmetry

Фермер Джон любит симметрию и сейчас размещает своих коров на прямоугольной решетке из $n \times m$ ячеек ($1 \le n \le 1\,000\,000\,000$, $1 \le m \le 1\,000\,000\,000$).

Чтобы обеспечить симметрию, фермер Джон размещает коров следующим образом.

Он ставит корову в самый центр решетки. Если такого нет, процесс прекращается.

Затем он разделяет поле на четыре меньших поля одинакового размера, разделяемых строкой и столбцом ячейки, в которую он поставил корову. Далее он продолжает процесс для каждого из вновь полученных полей, до тех пор пока не появится центральная ячейка или поле не может быть разделено на четыре подполя.

Например, если $n = 7$, а $m = 15$, то фермер Джон поставит корову в строку 4, колонку 8 и разделит поле на четыре новых поля размером 3×7 каждое. В каждом из этих четырех полей фермер Джон поставит корову

Контактная информация

Долинский Михаил Семенович, канд. тех. наук, доцент, доцент кафедры математических проблем управления и информатики, Гомельский государственный университет имени Франциска Скорины, Республика Беларусь; *адрес:* 246000, Республика Беларусь, г. Гомель, ул. Советская, д. 104; *e-mail:* dolinsky@gsu.by

M. S. Dolinsky
Francisk Skorina Gomel State University, The Republic of Belarus

SOLVING RECURSIVE PROBLEMS BY DEFINITION

Abstract
The article describes the methodology for studying the theme "Solving recursive problems by definition" in preparing students for Olympiads in informatics. Distance learning system DL.GSU.BY is the effective technical base for teaching.
Keywords: recursion, teaching for programming, Olympiads in informatics, distance learning tools.

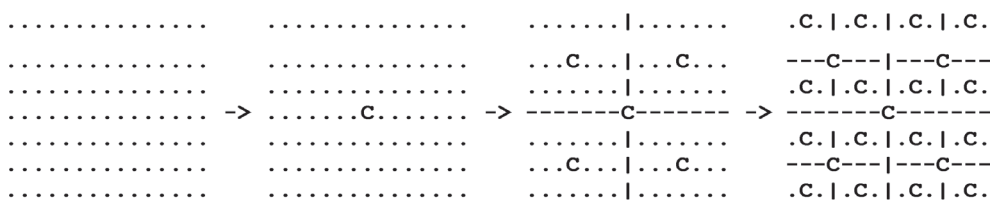

Рис 1. Иллюстрация примера задачи 1

в строку 2, колонку 4 и разделит поле на четыре поля размером 1×3. И т. д.

Этот процесс показан на рисунке 1 (здесь C обозначает корову).

Для такого поля потребуется 21 корова.

Но если, например, $n = m = 5$, потребуется всего одна корова, поскольку после ее установки в центре исходное поле разделится на четыре поля размером 2×2 каждое, которые уже не имеют центральной ячейки.

Помогите фермеру Джону определить, сколько коров ему понадобится.

Формат ввода:

Строка 1: два разделенных пробелом целых числа — n и m.

Формат вывода:

Строка 1: количество коров, которые понадобятся.

Пример ввода (файл sym.in):

7 15

Пример вывода (файл sym.out):

21

Пояснение.

Это задача на написание рекурсивной функции по определению, приведенному в условии задачи. Для получения ответа определяется и используется рекурсивная функция $K(m, n)$, которая сообщает количество коров, которые потребуются для решетки размером m строк и n столбцов. Если n и m нечетные, то мы можем поставить корову в центр и к ответу нужно прибавить 4 — количество коров, которые будут поставлены далее в рекурсивном процессе в каждой из частей. При этом оба параметра (m и n) делятся пополам перед очередным рекурсивным вызовом. Если же одно из чисел m или n четно, корову ставить нельзя, ответ — ноль.

Полный текст решения:

```
var
 n,m : longint;

 function K(n,m:longint):longint;
  begin
   if odd(n) and odd(m)
    then K:=1+4*K(n div 2,m div 2)
    else K:=0;
  end;

begin
 assign(input,'sym.in'); reset(input);
 assign(output,'sym.out'); rewrite(output);
 readln(n,m);
 writeln(K(n,m));
 close(input); close(output);
end.
```

Задача 2. 09_DecB. The Big Dance

Беси и стадо из n ($1 \le n \le 2200$) последовательно пронумерованных от 1 до n коров пришли на танцы, где их ждут быки как партнеры в танцах. Этот танец известен под названием «нечетная корова — лишняя». Коровы выстраиваются в ряд по порядку номеров, и выбирается средняя точка. Она или делит ряд коров ровно пополам, или выбирается так, что первое множество коров ровно на 1 больше, чем второе. Если ровно две коровы остаются во множестве, то они идут танцевать с быками. Если во множестве остается только одна корова, ей дают цветок и отправляют домой. Если во множестве коров больше, чем две, то процесс продолжается опять и опять до получения множеств с одной или двумя коровами. Идентификаторы двух коров перемножаются, и это число прибавляется к глобальной сумме.

Вам дается количество коров, вычислите глобальную сумму, которая получается в результате описанного процесса.

Рассмотрим пример с 11 коровами с номерами от 1 до 11.

На рисунке 2 приведен пример их последовательного разделения. Глобальная сумма будет равна 188.

Формат ввода:

Строка 1: одно целое число — n.

Формат вывода:

Строка 1: одно целое число — глобальная сумма, вычисленная, как описано выше.

Пример ввода (файл bigdance.in):

11

Пример вывода (файл bigdance.out):

188

Пояснение.

Эта задача на реализацию рекурсивной процедуры $Rec(n)$, описанной в условии задачи. Основная проблема — формализовать условие (начальное значение ответа = 0, глобальная переменная *Ans* в приведенной далее программе):

- если $n = 1$, то ответ не изменяется;
- если $n = 2$, то ответ увеличивается на произведение этих двух чисел (номеров двух соседних коров с указанной позиции);
- если $n > 2$, то группа из n чисел делится на две почти равные части (из $k1$ и $k2$ чисел), т. е., если n четное, то части равны половине n, а если n нечетное, то первая часть на единицу больше. И к обеим частям вновь применяется этот же рекурсивный алгоритм.

```
1     2     3     4     5     6  |  7     8     9     10     11
    1     2     3  |  4     5     6
        1     2  |  3
              1  2              => 1*2=2 добавляются к сумме -> сумма=2
              3                 => отправили домой с цветком
        4     5  |  6
              4  5              => 4*5=20 добавляется к сумме -> сумма=22
              6                 => отправили домой с цветком
    7     8     9  | 10    11
        7     8  |  9
              7  8              => 7*8=56 добавляется к сумме -> сумма=78
              9                 => отправили домой с цветком
        10    11                => 10*11=110 добавляется к сумме -> сумма=188
```

Рис 2. Иллюстрация примера задачи 2

На языке программирования это может быть записано так:

```
procedure Rec(n,p:longint);
 var
  k1,k2 : longint;
 begin
  if n=1 then exit;
  if n=2 then begin inc(ans,p*(p+1)); exit; end;
  k1:=n div 2; k2:=k1;
  if odd(n) then inc(k1);
  Rec(k1,p); Rec(k2,p+k1);
 end;
```

Полный текст решения:

```
var
 n,ans : longint;

 procedure Rec(n,p:longint);
  var
   k1,k2 : longint;
  begin
   if n=1 then exit;
   if n=2 then begin inc(ans,p*(p+1)); exit; end;
   k1:=n div 2; k2:=k1;
   if odd(n) then inc(k1);
   Rec(k1,p); Rec(k2,p+k1);
  end;

begin
 assign(input,'bigdance.in'); reset(input);
 assign(output,'bigdance.out'); rewrite(output);
 readln(n);
 ans:=0;
 Rec(n,1);
 writeln(ans);
 close(input); close(output);
end.
```

Задача 3. 08_AprS. Roads Around The Farm

Коровы фермера Джона решили исследовать территорию вокруг фермы. Изначально все n коров ($1 \le n \le 1\,000\,000\,000$) движутся вниз по дороге в одной большой группе. Когда они встречают развилку на дороге, то иногда предпочитают разбиться на две меньшие (непустые) группы, каждая из которых движется далее вниз по одной из дорог. Когда какая-то из групп вновь добирается до развилки, она снова разбивается на две группы, и т. д.

Коровы избрали оригинальный способ разбиваться на группы: если они могут разделиться на две группы так, что количества коров в этих группах различаются ровно на k ($1 \le k \le 1000$), тогда они делятся. А иначе прекращают движение вперед.

Предполагая, что всегда есть новая развилка на дороге, вычислите финальное количество групп коров.

Формат ввода:

Строка 1: два разделенных пробелом целых числа: n k.

Формат вывода:

Строка 1: одно целое число, представляющее получившееся количество групп коров.

Пример ввода (файл ratf.in):

6 2

Пример вывода (файл ratf.out):

3

Пояснение.

Рассмотрим приведенный пример. Всего есть шесть коров, и разность количеств в группах равна двум. Всего есть три финальных группы (с двумя, одной и тремя коровами в них):

```
  6
 / \
2   4
   / \
  1   3
```

Это задача на реализацию рекурсивной функции по определению, приведенному в условии задачи.

Если $(n - k)$ нечетное или $n \le k$, то группа одна, иначе рекурсивно вызываем процедуру для двух новых групп:

```
(n-k) div 2
```

и

```
k+(n-k) div 2
```

Далее приводятся две реализации решения — с рекурсивной процедурой и рекурсивной функцией. Первое легче отлаживать в случае необходимости, второе выглядит компактней.

Полный текст решения с рекурсивной процедурой:

```
var
 n,k,ans : longint;

 procedure Rec(n:longint);
  begin
   if (n<=k) or odd(n-k)
    then inc(ans)
    else  begin
           Rec((n-k) div 2);
           Rec(k+(n-k) div 2);
          end;
  end;

begin
 assign(input,'ratf.in'); reset(input);
 assign(output,'ratf.out'); rewrite(output);
 readln(n,k);
 ans:=0;
 Rec(n);
 writeln(ans);
 close(input); close(output);
end.
```

Полный текст решения с рекурсивной функцией:

```
var
 n,k : longint;

 function Kgr(n:longint):longint;
  begin
   if (n<=k) or odd(n-k)
    then Kgr:=1
    else Kgr:=Kgr((n-k)div 2)+Kgr(k+(n-k)div 2);
  end;

begin
 assign(input,'ratf.in'); reset(input);
 assign(output,'ratf.out'); rewrite(output);
 readln(n,k);
 writeln(Kgr(n));
 close(input); close(output);
end.
```

Задача 4. 12_FebB. Moo

Последовательность букв определена до бесконечности. Ее начало представлено ниже:

m o o m o o o m o o m o o o o o m o o m o o o m o o m o o o o o o

Эта последовательность проще всего описывается рекурсивно: пусть $s(0)$ — последовательность из трех символов: «m o o»; $s(k)$ получается конкатенацией: копии последовательности $s(k-1)$, затем «m o ... o» с $k+2$ символами «o», и затем еще одна копия последовательности $s(k-1)$.

Например:

$s(0)$ = 'm o o'
$s(1)$ = 'm o o m o o o m o o'
$s(2)$ = 'm o o m o o o m o o m o o o o o m o o m o o o m o o'

Очевидно, что так можно построить строку любой длины.

Требуется предсказать, каким будет символ на позиции n — «m» или «o».

Формат ввода:
Строка 1: одно целое число n ($1 \le n \le 10^9$).

Формат вывода:
Строка 1: единственная строка вывода должна содержать один символ — «m» или «o».

Пример ввода (файл moo.in):
11

Пример вывода (файл moo.out):
m

Пояснение.

Рассмотрим приведенный пример. Требуется предсказать 11-й символ.

Сначала реализация «в лоб».

Формируем (итерационно, не рекурсивно) строку по описанным правилам и выводим символ на введенной позиции (n) этой строки.

Поскольку строка может быть очень длинной, объявляем переменную типа ansistring (до 2 Гбайт). Это решение проходит девять тестов из десяти!

```
var
 s,m : ansistring;
 n,d : longint;
begin
 assign(input,'moo.in'); reset(input);
 assign(output,'moo.out'); rewrite(output);
 readln(n);
 s:='moo'; d:=3; m:='moo';
 while d<n do
  begin
   m:=m+'o';
   s:=s+m+s;
   d:=length(s);
  end;
 writeln(s[n]);
 close(input); close(output);
end.
```

Последний тест не проходит по памяти. На решение задачи отведено 256 мегабайт.

Значит, нам нужно написать решение *без* хранения строки. Как?

Строка по построению (описанному в условии задачи) всегда состоит из *трех* частей — левая, средняя и правая. При этом левая и правая части — одинаковые (предыдущая итерация этой строки), а средняя имеет на первой позиции букву «m», а на всех остальных позициях — букву «o».

Более формально рекурсивное определение конструируемой строки выглядит так:

$s(-1)$ = '' (пустая строка)
$s(0) = s(-1)$ + 'm' + 2 ('o') + $s(-1)$
$s(1) = s(0)$ + 'm' + 3 ('o') + $s(0)$
$s(2) = s(1)$ + 'm' + 4 ('o') + $s(1)$
...
$s(k) = s(k-1)$ + 'm'+ $(k+2)$ символов ('o') + $s(k-1)$

Поэтому рекурсивная идея решения может заключаться в следующем: добраться рекурсивно до средней части строки и вывести «m», если символ находится на первой позиции этой части, или «o» — в противном случае.

Опишем этот алгоритм более детально.

По введенному числу n надо установить минимальный номер итерации k такой, что $n \le Len(k)$ (длина строки $s(k)$) (т. е. символ на позиции n находится внутри этой строки).

Затем выясняем, внутри какой части этой строки находится позиция n:

- левой: $n \le Len(k-1)$;
- средней: $n > Len(k-1)$ и $n \le Len(k-1) + k + 3$;
- или правой: $n > Len(k-1) + k + 3$.

Если в левой, то вызываем рекурсивно описываемую процедуру.

Если в средней, то выводим ответ, как описано ранее.

Если в правой, то сначала вычитаем $Len(k-1)+k+3$, а затем вызываем рекурсивно описываемую процедуру.

Итак, $C(n, k)$ — это символ, который находится на позиции n в строке $s(k)$. Функция, которая вычисляет $C(n, k)$, может быть записана следующим образом:

```
function C(n,k :longint): char;
 begin
  if n>Len(k) then C:=C(n,k+1) else
  if n<=Len(k-1) then C:=C(n,k-1) else
   begin
    n:=n-Len(k-1);
    if n<=k+3
     then  begin
            if n=1
            then C:='m'
            else C:='o';
           end
     else  begin
            n:= n- (k+3);
            C:= C(n,k-1)
           end;
   end;
 end;
```

Здесь $Len(k)$ — это функция, которая вычисляет длину строки $s(k)$. Ее рекурсивная реализация может выглядеть так:

```
function Len(k:longint):longint;
 var
  x : longint;
 begin
  if k=-1
   then Len:=0
   else  begin
          x:=Len(k-1);
          Len:=2*x+k+3;
         end;
 end;
```

Полный текст решения:

```
var
 n,d,i : longint;

function Len(k:longint):longint;
 var
  x : longint;
 begin
  if k=-1
   then Len:=0
   else  begin
          x:=Len(k-1);
          Len:=2*x+k+3;
         end;
 end;

function C(n,k :longint): char;
 begin
  if n> Len(k) then C:=C(n,k+1) else
  if n<= Len(K-1) then C:=C(n,k-1) else
   begin
    n:=n-Len(k-1);
    if n<=k+3
     then  begin
            if n=1
             then C:='m'
            else C:='o';
           end
     else begin
           n:= n- (k+3);
           C:= C(n,k-1)
          end;
   end;
 end;

begin
 assign(input,'moo.in'); reset(input);
 assign(output,'moo.out'); rewrite(output);
 readln(n);
 writeln(C(n,0));
 close(input); close(output);
end.
```

Задача 5. 11_AprB. Skewed Sorting

Имеется 2^n ($1 \le n \le 10$) коров, пронумерованных от 1 до 2^n. Они встали в ряд в случайном порядке. Первая корова имеет номер cow_1, вторая — cow_2, ..., ($1 \le cow\_i \le 2^n$).

Используется следующий алгоритм упорядочения:

1. Если коров больше, чем две, он разделяет их на две равные подгруппы. Сортирует первую подгруппу, используя этот алгоритм. Затем сортирует вторую подгруппу, также используя этот алгоритм.
2. Рассмотрим текущее множество коров, которое должно быть упорядочено. Если первое число второго множества меньше, чем первого, то все элементы второго множества меняются местами со всеми элементами первого множества.

Требуется узнать, какое расстояние пройдут коровы, двигаясь в соответствии с данной «сортирующей» процедурой. Под пройденным расстоянием здесь понимается

количество позиций, которая пройдет корова при переходе от текущей позиции до следующей.

По заданной начальной конфигурации коров обработайте список по алгоритму, описанному выше, и выведите:

1) сумму всех расстояний, пройденных всеми коровами;
2) финальную конфигурацию коров после «сортирующей» процедуры.

Например, рассмотрим ряд из восьми коров:

8 5 2 3 4 7 1 6

Сначала они делятся на две подгруппы для сортировки:

8 5 2 3 | 4 7 1 6

Поскольку в каждой половине больше, чем две коровы, то половины сортируются по отдельности, начиная с первой.

Первая половина (8 5 2 3) сортируется следующим образом:

8 5 | 2 3

В каждой из получившихся половин осталось по две коровы.

Рассмотрим первую пару коров (8 5). Поскольку 5 < 8, мы должны поменять коров в этой паре: 5 8. Расстояние, пройденное каждой коровой, равно 1, поэтому суммарное расстояние равно 2.

Вторая половина (2 3) уже упорядочена, поэтому суммарное расстояние останется 2.

Новая конфигурация этой подгруппы такая:

5 8 | 2 3

Для шага 2 алгоритма для данной подгруппы мы сравниваем две стороны (5 8 и 2 3) лексикографически. Поскольку 2 < 5, мы меняем местами пары элементов:

2 3 5 8

Каждая из этих четырех коров прошла расстояние 2, поэтому всего расстояние 8, и суммарное расстояние становится равным 10.

Рассмотрим вторую половину коров (4 7 1 6). Делим список из четырех коров на две подгруппы:

4 7 | 1 6

Каждая пара (4 7 и 1 6) уже отсортирована.

Сравниваем пары (4 7) и (1 6). Поскольку 1 < 4, мы должны поменять эти две подгруппы:

1 6 4 7

Всего будет сделано 8 шагов, и общее расстояние станет равно 18.

После описанных выше операций список будет выглядеть так:

2 3 5 8 | 1 6 4 7

Пришло время выполнить шаг 2 над группами из четырех элементов.

Поскольку 1 < 2, мы должны поменять половины. Получим такую конфигурацию:

1 6 4 7 2 3 5 8

Поскольку каждая из 8 коров сделает по 4 шага, на этой итерации получится 32 шага, а общее пройденное расстояние станет равным 50.

Отсюда ответ: 50 и 1 6 4 7 2 3 5 8.

Формат ввода:

Строка 1: одно целое число — n.

Строки 2.. $2^n + 1$: строка $i + 1$ содержит одно целое число — *cow_i*.

Формат вывода:

Строка 1: одно целое число — общее расстояние, пройденное коровами.

Строки 2.. $2^n + 1$: строка $i + 1$ будет содержать одно целое число — номер i-й коровы в конечной конфигурации.

Пример ввода (файл ssort.in):

```
3
8
5
2
3
4
7
1
6
```

Пример вывода (файл ssort.out):

```
50
1
6
4
7
2
3
5
8
```

Пояснение.

Фактически в условиях задачи описывается такой рекурсивный алгоритм.

Пусть

k — количество элементов в группе;

p — позиция первого элемента группы.

Если $k \geq 2$, то:

- разбиваем группу на две;
- применяем рекурсивно алгоритм к обеим группам;
- объединяем две рядом стоящие группы размером k, причем первая группа начинается с элемента p.

На языке программирования Pascal это может быть записано так:

```
procedure Rec(k,p:longint);
 begin
  if k>=2
   then  begin
          k:=k div 2;
          Rec(k,p); Rec(k,p+k);
          Merge(k,p);
         end;
 end;
```

Процедура слияния *Merge* должна сравнить первые элементы в обеих группах. Если первый элемент во второй группе меньше первого элемента в первой группе, то надо поменять группы местами (т. е. поменять местами попарно соответственно все элементы групп). Ответ при этом надо увеличить на $2 \cdot k \cdot k$ (2 группы, k элементов, каждый пройдет по k шагов).

```
 procedure Merge(k,p: longint);
  var
   i,t : longint;
  begin
   if cow[p+k]<cow[p]
    then  begin
           for i:=1 to k do
            begin
             t:=cow[p];
             cow[p]:=cow[p+k];
             cow[p+k]:=t;
             inc(p);
            end;
           inc(ans,2*k*k);
          end;
  end;
```

Полный текст решения:

```
Const
 p2 : array [1..10] of longint =
   (2,4,8,16,32,64,128,256,512,1024);
var
 cow         : array [1..1024] of longint;
 n,i,ans,pn : longint;

 procedure Merge (k,p: longint);
  var
   i,t : longint;
  begin
   if cow[p+k]<cow[p]
    then  begin
           for i:=1 to k do
            begin
             t:=cow[p];
             cow[p]:=cow[p+k];
             cow[p+k]:=t;
             inc(p);
            end;
           inc(ans,2*k*k);
          end;
  end;

 procedure Rec(k,p:longint);
  begin
   if k>=2
    then  begin
           k:=k div 2;
           Rec(k,p); Rec(k,p+k);
           Merge(k,p);
          end;
  end;

begin
 assign(input,'ssort.in'); reset(input);
 assign(output,'ssort.out'); rewrite(output);
 readln(n);
 pn:=p2[n];
 for i:=1 to pn do readln(cow[i]);
 ans:=0;
 Rec(pn,1);
 writeln(ans);
 for i:=1 to pn do writeln(cow[i]);
 close(input); close(output);
end.
```

Заключение

В данной статье представлена методика изучения темы «Решение рекурсивных задач по определению» учениками пятых—восьмых классов, предлагающая наиболее простой, по мнению автора, способ постепенного осознания школьниками механизма рекурсии и способа решения задач с ее помощью. Методика включает в себя последовательность задач в порядке возрастания сложности, снабженных при необходимости предварительными общими пояснениями и последующими полными решениями предлагаемых задач.

Список использованных источников

1. *Долинский М. С.* Введение в решение задач с помощью рекурсивных процедур и функций // Информатика в школе. 2016. № 10.

2. *Долинский М. С.* Введение в решение задач с помощью рекурсивных процедур и функций // Информатика в школе. 2019. № 5.

3. *Долинский М. С.* Гомельская школа олимпиадного программирования // Информатика и образование. 2015. № 7.

4. *Долинский М. С., Кугейко М. А.* Гомельская инструментальная система дистанционного обучения // Информатика и образование. 2010. № 11.