16 (повышенный уровень, время – 5 мин)

Тема: Рекурсия. Рекурсивные процедуры и функции

Что проверяется:

Вычисление рекуррентных выражений

1.5.3. Индуктивное определение объектов.

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Что нужно знать:

- для того, чтобы задать рекурсивную функцию, нужно определить
  1) условие окончания рекурсии, то есть значения параметров функции, для которых значение функции известно или вычисляется без рекурсивных вызовов;
  2) рекуррентную формулу (или формулы), с помощью которых значение функции для заданных значений параметров вычисляется через значение (или значения) функции для других значений параметров (то есть, с помощью рекурсивных вызовов)
- задачи, в которых требуется найти значение заданной рекурсивной функции при известных значениях параметров можно решать с помощью ручных вычислений, используя электронные таблицы или с помощью своей программы; последние два способа обычно более эффективны
- функцию

  $F(n) = 1$ при $n \le 1$

  $F(n) = n + 1 + F(n–1)$, при $n > 1$

  можно реализовать следующим образом в виде функций на языках программирования:

  Python:

```
def F( n ):
  if n <= 1:
    return 1
  else:
    return n + 1 + F(n-1)
```

  Паскаль:

```
function F( n: integer ): integer;
begin
  if n <= 1 then
    Result := 1;
  else
    Result := n + 1 + F(n-1);
end;
```

  C++:

```
int F( int n )
{
  if( n <= 1 )
    return 1;
  else
    return n + 1 + F(n-1);
}
```

Пример задания:

P-05 (И.В. Свиридкин). Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 2n$ при $n \le 5$

$F(n) = F(n–2) + 3·F(n/2) + n$, если $n > 5$ и чётно,

$F(n) = F(n–1) + F(n–2) + F(n–3)$, если $n > 5$ и нечётно.

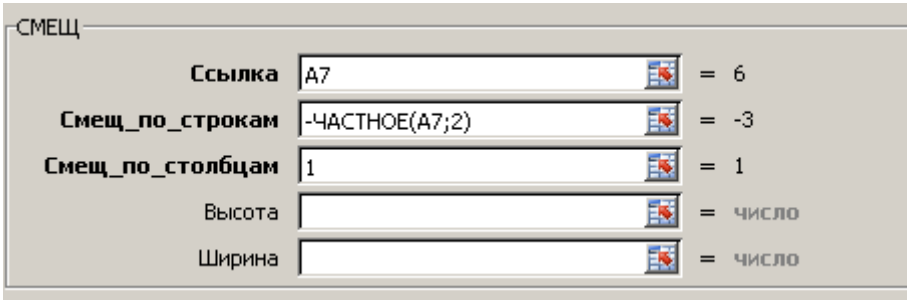
Чему равно значение функции $F(99) + F(100)$?

Решение (электронные таблицы, И.В. Свиридкин):

1) для больших значений n вычислить нужное значение на бумаге очень затруднительно, поэтому будем решать, используя компьютер, а именно, электронные таблицы
2) выражение $F(n) = F(n–1)$ легко реализовать:

	A	B	C
1	n	F(n)	
2	1	1	
3	2	2	
4	3	3	
5	4	=B4	
6	5		

3) но если в выражении встретится деление ($F(n/2)$), то этот способ решения уже не подходит; такие задачи можно решить с помощью функции **СМЕЩ** из категории ССЫЛКИ И МАССИВЫ:
=СМЕЩ (<Ссылка>;<Смещение по строкам>;< Смещение по столбцам>)
Ссылка – адрес ячейки от которой отсчитывается смещение
Смещение по строкам – на сколько строк вверх или вниз сместить
Смещение по столбцам – на сколько столбцов влево или вправо сместить
4) сначала заполним значения для аргумента n от 1 до 100 (используем прогрессию), затем введём первые пять значений $F(n)$
5) запишем в ячейку **B7** введём выражение для $F(n) = F(n–2) + 3·F(n/2) + n$ при чётных $n \geq 5$, используя функцию **СМЕЩ**:



6) в итоге получим:

5	4	8
6	5	10
7	6	=B5+3*СМЕЩ(A7;-ЧАСТНОЕ(A7;2);1)+A7

7) Примечание (**К. Поляков**): можно также использовать формулу, в которой базовая ячейка – A1 (ссылка на неё должна быть абсолютной):
=B5+3*СМЕЩ(A1;ЧАСТНОЕ(A7;2);1)+A7
8) Следующее выражение $F(n) = F(n–1) + F(n–2) + F(n–3)$ при нечётных $n \geq 5$ вводим в ячейку **B8**:

	A	B	C
6	5	10	
7	6	32	
8	7	=B7+B6+B5	

9) далее выделяем диапазон **B7:B8** и копируем формулы до конца столбца
10) находим значение $F(99) + F(100)$

98	97	111195254	
99	98	10990672	
100	99	130706954	
101	100	11432540	142139494
102			

11) Ответ: 142139494.
12) Примечание: можно оба выражения записать в одну строку используя функцию **ЕСЛИ**:

	A	B	C	D	E	F
6	5	9				
7	6	=ЕСЛИ(ОСТАТ(A7;2)=0;B5+3*СМЕЩ(A7;-A7/2;1)+A7;B6+B5+B4)				
8	7	22				

Решение (программа на Python):

1) задача легко решается простой программой на Python с рекурсивной функцией:

```
def F( n ):
  if n <= 5: return 2*n;
  if n % 2 == 0:
    return F(n-2) + 3*F(n//2) + n
  else:
    return F(n-1) + F(n-2) + F(n-3)

print( F(99) + F(100) )
```

2) Ответ: 142139494.

Ещё пример задания:

P-04 (демо-2021). Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 1$ при $n = 1$

$F(n) = n + F(n–1)$, если n чётно,

$F(n) = 2 \cdot F(n–2)$, если $n > 1$ и n нечётно.

Чему равно значение функции $F(26)$?

Решение (ручной счёт от последнего значения):

3) чтобы вычислить $F(26)$, используем формулу для чётных n:

$F(26) = 26 + F(25)$

4) нам неизвестно значение $F(25)$, поэтому, применяя формулу для нечётных n, находим

$F(25) = 2 \cdot F(23)$

5) далее так же придётся написать формулы для вычисления $F(23)$, $F(21)$, …, $F(3)$, и в конце мы получим

$F(3) = 2 \cdot F(1)$

6) значение $F(1) = 1$ нам задано, подставляя его в предыдущую формулу, находим

$F(3) = 2 \cdot F(1) = 2$

7) теперь значение подставляем в формулу для $F(5)$, потом найденное значение $F(5)$ – в формулу для $F(7)$, и т.д.
8) после продолжительных вычислений получим $F(26) = 4122$
9) Ответ: 4122.

Решение (ручной счёт от первого значения):

1) примерно то же самое можно сделать, начиная вычисления с малых значений n
2) сразу записываем в таблицу известное значение $F(1) = 1$, затем последовательно вычисляем

   $F(2) = 2 + F(1) = 3$ по формуле для чётных n

   $F(3) = 2 \cdot F(1) = 2$ по формуле для нечётных n

   $F(4) = \ldots$

   ...

   $F(26) = 26 + F(25) = 4122$
3) результаты вычислений удобно хранить в виде таблицы:

n	1	2	3	4	5	6	7	8	9	…	26
$F(n)$	1	3	2	6	4	10	8	16	16		4122

4) недостаток этого метода в том, что мы вычисляем все значения $F(n)$ подряд, хотя для нахождения $F(26)$ нам не нужны значения $F(n)$ при чётных n
5) Ответ: 4122.
6) (**И. Степанов, Челябинск**) в данной конкретной задаче можно рассмотреть сначала только нечётные значения n, которые вычисляются по формуле $F(n) = 2 \cdot F(n–2)$
7) из это формулы видно, что значения $F(n)$ при нечётных n – это степени двойки:

   $F(3) = 2 \cdot F(1) = 2$

   $F(5) = 2 \cdot F(3) = 4$

   $F(7) = 2 \cdot F(5) = 8$

   ...

   $F(25) = 2 \cdot F(23) = 4096$
8) тогда по формуле для чётных n получаем

   $F(26) = 26 + F(25) = 4122$
9) Ответ: 4122.

Общий недостаток первых двух методов – **большой объём ручных вычислений**, который приводит к большой вероятности ошибки.

Решение (использование табличного процессора):

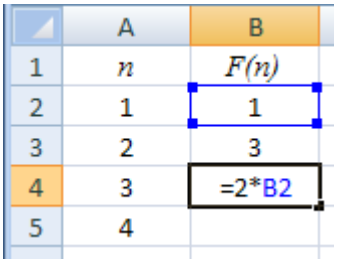
1) для выполнения большого объёма вычислений можно использовать табличный процессор; удобнее строить таблицу из двух столбцов (хотя можно, конечно использовать и две строки) – n и $F(n)$
2) сразу записываем известное значение $F(1) = 1$

	A	B
1	n	$F(n)$
2	1	1
3	2	
4	3	

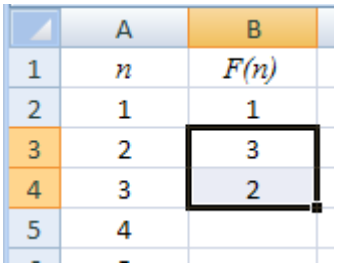
3) в ячейку B3, где вычисляется $F(2)$, записываем формулу для чётных n; здесь соответствующее значение n хранится в A3, а значение $F(n–1) = F(1)$ – в ячейке B2

	A	B
1	n	$F(n)$
2	1	1
3	2	=A3+B2
4	3	
5	4	
6	5	
7	6	

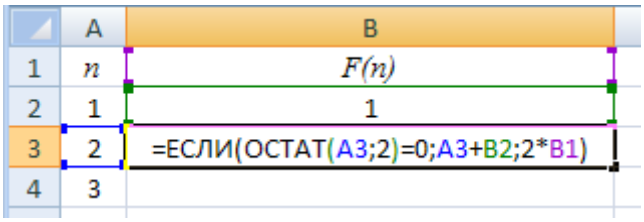
4) в ячейку B4 , где вычисляется $F(3)$, записываем формулу для нечётных n; для этой ячейки соответствующее значение $F(n–2) = F(1)$ – в ячейке B2



5) поскольку формулы чередуются (для чётных и нечётных n), выделяем **две** введённые формулы и «протягиваем» их, копируя вниз до строки с $n = 24$:



6) напротив значения $n = 26$ считываем ответ – 4122
7) Ответ: 4122.
8) можно обойтись и одной формулой, только в ней придётся использовать функцию ЕСЛИ:



Функция ОСТАТ вычисляет остаток от деления n на 2; если этот остаток равен нулю, то значение n чётное.

Решение (составление программы – рекурсивная функция):

1) составление программы – самый простой и наглядный способ решения задачи; для этого нужно просто реализовать заданные формулы в виде функции на языке программирования;
2) эта задача не должна представлять какой-то сложности; главная проблема – не получить бесконечную рекурсию; для этого рекомендуется сразу в начале функции записать условие окончания рекурсии, которое задаётся условием «$F(n) = 1$ при $n = 1$», и сразу выйти из функции
3) программа на языке Python:

```
def F( n ):
  if n == 1: return 1
  if n % 2 == 0:
    return n + F(n-1)
  else:
    return 2 * F(n-2)

print( F(26) )
```

4) программа на языке Паскаль:

```
function F( n: integer ): integer;
begin
  if n = 1 then begin
    Result := 1;
```

```
      Exit;
    end;
    if n mod 2 = 0 then
      Result := n + F(n-1)
    else
      Result := 2 * F(n-2);
  end;
  begin
    writeln( F(26) )
  end.
```

5) программа на языке C++:

```
#include <iostream>
using namespace std;

int F( int n )
{
  if( n == 1 ) return 1;
  if( n % 2 == 0 )
    return n + F(n-1);
  else
    return 2 * F(n-2);
}
int main()
{
  cout << F(26);
}
```

Решение (составление программы – динамическое программирование):

1) если рекурсивная функция вызывает сама себя несколько раз (для рассматриваемой задачи это не так, но вполне может быть…), при больших значениях аргумента функция $F(n)$ может вычисляться очень (и даже недопустимо!) долго – это один из недостатков рекурсивной реализации
2) во многих случаях достаточно просто заменить рекурсивную функцию нерекурсивной (в принципе, это можно сделать всегда, вопрос лишь в том, какие усилия придётся для этого приложить)
3) один из простых приёмов – использование динамического программирования, которое позволяет свести вычисление значения функции, заданной рекурсивно, к заполнению массива (таблицы) – так же, как при ручном счёте (этот метод решения задачи описан выше)
4) пусть известно значение n; построим массив **a**, так чтобы значение элемента **a[n]** совпадало со значением $F(n)$
5) учитывая, что в Python и в C++ нумерация элементов массива начинается с нуля, будем использовать фиктивный нулевой элемент – не будем его использовать вообще; согласно условию, в первый элемент нужно записать число 1 («$F(n) = 1$ при $n = 1$»):

```
  a = [0, 1]
```

6) затем перебираем в цикле все значения индексов элементов, начиная с 2 и до n включительно, вычисляя для каждого значение функции; фактически сначала вычисляется $F(2)$ и записывается в элемент массива **a[2]**, затем находим $F(3)$ и записываем в элемент массива **a[3]** и т.д. :

```
for i in range(2, n+1):
  if i % 2 == 0:
```

```
      a.append( i + a[i-1] )
    else:
      a.append( 2*a[i-2] )
```

7) заметьте, что вместо рекурсивных вызовов здесь используются уже готовые значения $F(n)$ для меньших значений n, ранее записанные в массив **a** (см. выделение синим цветом)

8) теперь можно построить функцию, которая возвращает значение **a[n]**:

```
def F( n ):
  a = [0, 1]
  for i in range(2, n+1):
    if i % 2 == 0:
      a.append( i + a[i-1] )
    else:
      a.append( 2*a[i-2] )
  return a[n]
```

9) при вызове

```
print( F(26) )
```

эта функция возвращает тот же результат 4122, что и рекурсивная функция

10) Ответ: 4122.

11) преимущество такого подхода проявится, если функция вызывает сама себя несколько раз; в этом случае каждое значение будет вычисляться только один раз и сохраняться в массиве; когда это значение понадобится снова, оно будет взято сразу из массива в готовом виде (этот приём иногда называют *мемоизацией* – запоминанием)

12) учитывая, что преобразования программы не слишком простые, рекомендуется использовать такой метод на экзамене только тогда, когда «лобовое» решение работает слишком медленно (вы не смогли дождаться результата); помните, что любой шаг в сторону от простейшего решения может стать источником дополнительных ошибок

13) решение на языке Паскаль (элементы массива нумеруются с единицы):

```
function F( n: integer ): integer;
var a: array[1..100] of integer;
    i: integer;
begin
  a[1] := 1;
  for i:=2 to n do
    if i mod 2 = 0 then
      a[i] := i + a[i-1]
    else
      a[i] := 2 * a[i-2];
  Result := a[n];
end;
begin
  writeln( F(26) )
end.
```

14) решение на языке C++

```
#include <iostream>
using namespace std;
int F( int n )
{
  int i, a[100] = {};
  a[1] = 1;
  for( i = 2; i <= n; i++ )
```

```
    if( i % 2 == 0 )
      a[i] = i + a[i-1];
    else
      a[i] = 2 * a[i-2];
  return a[n];
}
int main()
{
  cout << F(26);
}
```

Решение (составление программы – вообще без массива и рекурсии):

1) описанный далее метод решения задачи, в принципе, можно использовать, но при сдаче экзамена он **не имеет никаких преимуществ** перед методом динамического программирования, и в то же время повышает ваши шансы сделать ошибку; поэтому подумайте дважды (трижды, четырежды, …), прежде чем попытаться «блеснуть мастерством», к сожалению, при текущей модели экзамена это никто не оценит…
2) в данной задаче можно заметить, что очередное значение $F(n)$ зависит только от $F(n–1)$ или от $F(n–2)$; следовательно, знания значений $F(n–1)$ и $F(n–2)$ всегда будет достаточно для вычисления $F(n)$
3) поэтому можно вычислить $F(n)$, вообще не используя массив, нужно только хранить значения $F(n–1)$ и $F(n–2)$ в переменных
4) в приведённой далее программе переменные **fn1** и **fn2** хранят соответственно значения $F(n–1)$ и $F(n–2)$; начальное значение **fn1** должно быть равно 1 ($F(1) = 1$), а начальное значение **fn2** можно выбрать любым (например, 0), потому что на первой итерации цикла (при вычислении $F(2)$) оно все равно не используется

```
def F( n ):
  fn2 = 0  # эту строку можно удалить
  fn1 = 1
  for i in range(2, n+1):
    if i % 2 == 0:
      fn = i + fn1
    else:
      fn = 2*fn2
    fn2, fn1 = fn1, fn
  return f
```

Очередное значение $F(n)$ хранится в переменной **fn**. Строка

```
fn2, fn1 = fn1, fn
```

выполняет сдвиг переменных при переходе к следующему значению n: $F(n–1)$ записывается на место $F(n–2)$, а $F(n)$ – на место $F(n–1)$. При вызове

```
print( F(26) )
```

эта функция возвращает тот же результат 2074, что и рекурсивная функция

5) Ответ: 4122.
6) решение на языке Паскаль:

```
function F( n: integer ): integer;
var fn1, fn2, fn, i: integer;
begin
  fn1 := 1;
  for i:=2 to n do begin
```

```
    if i mod 2 = 0 then
      fn := i + fn1
    else
      fn := 2 * fn2;
    fn2 := fn1;
    fn1 := fn;
  end;
  Result := fn;
end;
begin
  writeln( F(26) )
end.
```

7) решение на языке C++

```
#include <iostream>
using namespace std;
int F( int n )
{
  int i, fn, fn1, fn2;
  fn1 = 1;
  for( i = 2; i <= n; i++ ) {
    if( i % 2 == 0 )
      fn = i + fn1;
    else
      fn = 2 * fn2;
    fn2 = fn1;
    fn1 = fn;
    }
  return fn;
}
int main()
{
  cout << F(26);
}
```

Ещё пример задания:

P-03. Определите наименьшее значение n, при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 500000. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующую сумму выведенных чисел.

Python	Паскаль	C++

def F(n):
  print(2*n)
  if n > 1:
  print(n-5)
  F(n-1)
  F(n-2)
	procedure F
	  (n: integer);
	begin
	  writeln(2*n);
	  if n > 1 then begin
	  writeln(n-5);
	  F(n-1);
	  F(n-2);
	  end;
	end;
		void F(int n)
		{
		  cout << 2*n << endl;
		  if(n > 1) {
		  cout << n-5 << endl;
		  F(n-1);
		  F(n-2);
		  }
		}

Для выполнения задания можно также написать программу или воспользоваться редактором электронных таблиц.

Решение (с помощью глобальной переменной):

1) первое, что может прийти в голову – вызывать приведённую процедуру при разных значениях параметра и увеличивать это значение до тех пор, пока сумма выведенных чисел не превысит заданное значение 500000; это тупиковый подход, поскольку чисел очень много и сложение займет очень много времени при низкой вероятности правильного ответа
2) можно попробовать изменить программу так, чтобы сумма выводимых чисел считалась автоматически: добавим в программу глобальную переменную **s** и будем увеличивать её при выводе каждого числа на значение этого числа; при этом для ускорения (значительного!) работы программы сразу закомментируем вывод чисел на экран:

```
def F( n ):
  global s    # если не объявить s глобальной – ошибка!
  # print(2*n)
  s += 2*n
  if n > 1:
    # print(n-5)
    s += n - 5
    F(n-1)
    F(n-2)
```

3) дальше можно написать такую программу и запускать её при различных значениях переменной **n**:

```
n = 15
s = 0
F(n)
print( n, s )
```

4) увеличивая каждый раз значение n на 1, мы в конце концов найдём первое (минимальное) значение n, при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 500000 – это $F(24) = 531864$
5) можно оформить поиск нужного значения n в виде цикла, например, так:

```
n = 0
while True:
  n += 1        # первое значение n = 1
  s = 0         # нужно обнулять сумму перед каждым вызовом
  F(n)          # подсчитали сумму
  if s > 500000: break; # если нашли, выход из цикла
print( n, s )   # вывод результата
```

6) решение на языке Паскаль (строки с выводом чисел закомментированы):

```
var s, n: integer;
procedure F( n: integer );
begin
  // writeln(2*n);
  s := s + 2*n;
  if n > 1 then begin
    // writeln(n-5);
    s := s + n - 5;
    F(n-1);
    F(n-2);
  end;
```

```
end;
begin
  n := 0;
  repeat
    n := n + 1;
    s := 0;
    F(n);
  until s > 500000;
  writeln( n, ' ', s);
end.
```

7) решение на языке C++ (строки с выводом чисел закомментированы):

```
#include <iostream>
using namespace std;
int s;
void F( int n )
{
  // cout << 2*n << endl;
  s += 2*n;
  if( n > 1 ) {
    // cout << n-5 << endl;
    s += n - 5;
    F(n-1);
    F(n-2);
    }
}
int main()
{
  int n = 0;
  do {
    n += 1;
    s = 0;
    F(n);
    }
  while( s <= 500000 );
  cout << n << " " << s;
}
```

8) в этой задаче процедура вызывает сама себя дважды, такая «ветвистость» может значительно увеличить время вычислений; если вы видите, что программа работает очень долго (результата не дождаться), нужно применить другой подход – искать решение через рекуррентную формулу (см. следующее решение) и применять метод динамического программирования, сохраняя в массиве все промежуточные результаты (см. подробности в решении задачи P-01)

9) Ответ: 24 531864.

Решение (с помощью функции):

1) можно преобразовать процедуру в функцию, не используя глобальную переменную:

```
def F( n ):
  # print(2*n)
  s += 2*n
  if n > 1:
    # print(n-5)
```

```
    s += n - 5
    return s + F(n-1) + F(n-2)
  return s
```

2) основная программа не требует использования глобальной переменной:

```
n = 0
while True:
  n += 1                  # первое значение n = 1
  s = F(n)                # подсчитали сумму
  if s > 500000: break; # если нашли, выход из цикла
print( n, s )             # вывод результата
```

3) Решение на Паскале:

```
var s, n: integer;
function F( n: integer ): integer;
begin
  // writeln(2*n);
  Result := 2*n;
  if n > 1 then begin
    // writeln(n-5);
    Result := Result + n - 5;
    Result := Result + F(n-1) + F(n-2);
  end;
end;
begin
  n := 0;
  repeat
    n := n + 1;
    s := F(n);
  until s > 500000;
  writeln( n, ' ', s);
end.
```

4) Решение на C++:

```
#include <iostream>
using namespace std;
int F( int n )
{
  // cout << 2*n << endl;
  int s = 2*n;
  if( n > 1 ) {
    // cout << n-5 << endl;
    s += n - 5;
    s += F(n-1) + F(n-2);
    }
  return s;
}
int main()
{
  int n = 0, s;
  do {
    n += 1;
    s = F(n);
    }
```

```
    while( s <= 500000 );
    cout << n << " " << s;
  }
```

5) Ответ: 24 531864.

Решение (с помощью формулы):

1) можно построить рекурсивную функцию $f(n)$, которая определяет сумму выведенных чисел; после этого можно использовать любой из методов вычислений, рассмотренных ниже для задачи P-01 – считать вручную, использовать электронную таблицу или написать свою собственную программу
2) итак, согласно условию, при $n \le 1$ выводится только число $2n$, то есть

   $f(n) = 2n$ при $n \le 1$

   это условие окончания рекурсии
3) основная часть определения рекурсивной функции – рекуррентная формула для остальных случаев; при $n > 1$ процедура печатает число $2n$ сразу при входе в процедуру, затем – ещё число $n–5$ в теле условного оператора, а затем дважды вызывает сама себя с разными значениями параметра, так что

   $f(n) = 2n + n – 5 + f(n–1) + f(n–2)$ при $n > 1$
4) теперь эту функцию можно записать, например, в виде программного кода:

```
def f( n ):
  if n <= 1:
    return 2*n
  else:
    return 2*n + n – 5 + f(n-1) + f(n-2)
```

5) запишем основную программу – алгоритм поиска нужного значения n:

```
n = 0
while True:
  n += 1
  s = f(n)
  if s > 500000: break;
print( n, s )
```

6) решение на языке Паскаль:

```
var s, n: integer;
function f( n: integer ): integer;
begin
  if n <= 1 then
    Result := 2*n
  else
    Result := 2*n + n - 5 + f(n-1) + f(n-2);
end;
begin
  n := 0;
  repeat
    n := n + 1;
    s := f(n);
  until s > 500000;
  writeln( n, ' ', s);
end.
```

7) решение на языке C++:

```
#include <iostream>
```

```
using namespace std;
int f( int n )
{
  if( n <= 1 )
    return 2*n;
  else
    return 2*n + n - 5 + f(n-1) + f(n-2);
}
int main()
{
  int n = 0, s;
  do {
    n += 1;
    s = f(n);
    }
  while( s <= 500000 );

  cout << n << " " << s;
}
```

8) заметим, что когда мы получили математическую форму записи функции (см. пп. 2 и 3), для решения задачи можно использовать любой из методов, описанных в разборе задачи P-01 ниже (например, ручной счёт или электронные таблицы)
9) недостаток такого подхода в том, что он *косвенный*, то есть требует преобразований; поэтому есть неплохой шанс ошибиться при выводе формулы и поэтому получить неверный результат
10) если вы видите, что программа работает очень долго (результата не дождаться), нужно применять метод динамического программирования, сохраняя в массиве все промежуточные результаты (см. подробности в решении задачи P-01)
11) Ответ: 24 531864.

Ещё пример задания:

P-02. Определите, сколько символов * выведет эта процедура при вызове $F(22)$:

Python	Паскаль	C++

def F(n):
  print('*')
  if n >= 1:
  print('*')
  F(n-1)
  F(n-2)
  F(n-3)
	procedure F(n: integer);
	begin
	  write('*');
	  if n >= 1 then begin
	  write('*');
	  F(n-1);
	  F(n-2);
	  F(n-3);
	  end;
	end;
		void F(int n)
		{
		  cout << '*';
		  if(n >= 1) {
		  cout << '*';
		  F(n-1);
		  F(n-2);
		  F(n-3);
		  }
		}

Для выполнения задания можно также написать программу или воспользоваться редактором электронных таблиц.

Решение (с помощью счётчика):

1) может показаться, что эта задача легче, чем задача P-01, разобранная ниже: ведь нам уже дана реализация функции на языке программирования, остаётся только запустить её и посмотреть, что она выведет

2) первое впечатление очень обманчиво; при вызове $F(22)$ программа выводит огромное количество звёздочек (**больше миллиона!**), подсчитать их вручную не представляется возможным
3) можно попробовать изменить программу так, чтобы выводимые звёздочки считались автоматически: добавим в программу глобальную переменную-счётчик **count** и будем увеличивать его при выводе каждой звёздочки:

```
count = 0
def F( n ):
  global count # если не объявить count глобальной – ошибка!
  print('*')
  count += 1
  if n >= 1:
    print('*')
    count += 1
    F(n-1)
    F(n-2)
    F(n-3)

F(22)
print(count)
```

4) запуск программы с вызовов $F(22)$ показывает, что очень много времени занимает вывод звёздочек на экран, мы рискуем не дождаться ответа
5) поэтому убираем из программы операторы вывода в теле функции (например, их можно временно отключить с помощью комментариев):

```
count = 0
def F( n ):
  global count # если не объявить count глобальной – ошибка!
  # print('*')
  count += 1
  if n >= 1:
    # print('*')
    count += 1
    F(n-1)
    F(n-2)
    F(n-3)

F(22)
print(count)
```

6) запустив такую программу, быстро получаем ответ: 1957585
7) решение на языке Паскаль (строки с выводом звёздочек закомментированы):

```
var count: integer;
procedure F( n: integer );
begin
  // write('*');
  count := count + 1;
  if n >= 1 then begin
    // write('*');
    count := count + 1;
    F(n-1);
    F(n-2);
```

```
      F(n-3);
    end;
  end;
  begin
    count := 0;
    F(22);
    writeln(count);
  end.
```

8) решение на языке C++ (строки с выводом звёздочек закомментированы):

```
#include <iostream>
using namespace std;
int count = 0;
void F( int n )
{
  // cout << '*';
  count ++;
  if( n >= 1 ) {
    // cout << '*';
    count ++;
    F(n-1);
    F(n-2);
    F(n-3);
    }
}
int main()
{
  F(22);
  cout << count;
}
```

9) в этой задаче процедура вызывает сама себя трижды, такая «ветвистость» может значительно увеличить время вычислений; если вы видите, что программа работает очень долго (результата не дождаться), нужно применить другой подход – искать решение через рекуррентную формулу (см. следующее решение) и применять метод динамического программирования, сохраняя в массиве все промежуточные результаты (см. подробности в решении задачи P-01)
10) Ответ: 1957585.

Решение (с помощью формулы):

1) возможен другой приём: построить рекурсивную функцию $f(n)$, которая определяет количество выведенных звёздочек; после этого можно использовать любой из методов вычислений, рассмотренных ниже для задачи P-01 – считать вручную, использовать электронную таблицу или написать свою собственную программу
2) итак, согласно условию, при $n < 1$ выводится только одна звёздочка, то есть

   $f(n) = 1$ при $n < 1$

   это условие окончания рекурсии
3) основная часть определения рекурсивной функции – рекуррентная формула для остальных случаев; при $n \geq 1$ процедура печатает одну звёздочку сразу при входе в процедуру, затем – ещё одну в теле условного оператора, а затем трижды вызывает сама себя с разными значениями параметра, так что

   $f(n) = 1 + 1 + f(n–1) + f(n–2) + f(n–3)$ при $n \geq 1$

4) теперь эту функцию можно записать, например, в виде программного кода:

```
def f( n ):
  if n < 1:
    return 1
  else:
    return 2 + f(n-1) + f(n-2) + f(n-3)
```

вызвать и вывести полученный ответ:

```
print( f(22) )
```

5) решение на языке Паскаль:

```
function f( n: integer ): integer;
begin
  if n < 1 then
    Result := 1
  else
    Result := 2 + f(n-1) + f(n-2) + f(n-3);
end;
begin
  writeln( f(22) );
end.
```

6) решение на языке C++:

```
#include <iostream>
using namespace std;
int f( int n )
{
  if( n < 1 )
    return 1;
  else
    return 2 + f(n-1) + f(n-2) + f(n-3);
}
int main()
{
  cout << f(22);
}
```

7) заметим, что когда мы получили математическую форму записи функции (см. пп. 2 и 3), для вычисления $f(22)$ можно использовать любой из методов, описанных в разборе задачи P-01 ниже (например, ручной счёт или электронные таблицы)
8) недостаток такого подхода в том, что он *косвенный*, то есть требует преобразований; поэтому есть неплохой шанс ошибиться при выводе формулы и поэтому получить неверный результат
9) если вы видите, что программа работает очень долго (результата не дождаться), нужно применять метод динамического программирования, сохраняя в массиве все промежуточные результаты (см. подробности в решении задачи P-01)
10) Ответ: 1957585.

Ещё пример задания:

P-01. Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 1$ при $n = 1$

$F(n) = n + 2 + F(n–1)$, если n чётно,

$F(n) = 2 \cdot F(n–2)$, если n нечётно.

Чему равно значение функции $F(24)$? Для выполнения задания можно также написать программу или воспользоваться редактором электронных таблиц.

Решение (ручной счёт от последнего значения):

10) чтобы вычислить $F(24)$, используем формулу для чётных n:
$$F(24) = 24 + 2 + F(23)$$
11) нам неизвестно значение $F(23)$, поэтому, применяя формулу для нечётных n, находим
$$F(23) = 2 \cdot F(21)$$
12) далее так же придётся написать формулы для вычисления $F(21)$, $F(19)$, …, $F(3)$, и в конце мы получим
$$F(3) = 2 \cdot F(1)$$
13) значение $F(1) = 1$ нам задано, подставляя его в предыдущую формулу, находим
$$F(3) = 2 \cdot F(1) = 2$$
14) теперь значение подставляем в формулу для $F(5)$, потом найденное значение $F(5)$ – в формулу для $F(7)$, и т.д.
15) после продолжительных вычислений получим $F(24) = 2074$
16) Ответ: 2074.

Решение (ручной счёт от первого значения):

10) примерно то же самое можно сделать, начиная вычисления с малых значений n
11) сразу записываем в таблицу известное значение $F(1) = 1$, затем последовательно вычисляем
$F(2) = 2 + 2 + F(1) = 5$ по формуле для чётных n
$F(3) = 2 \cdot F(1) = 2$ по формуле для нечётных n
$F(4) = \ldots$
…
$F(24) = 24 + 2 + F(23) = 2074$
12) результаты вычислений удобно хранить в виде таблицы:

n	1	2	3	4	5	6	7	8	9	…	24
$F(n)$	1	5	2	8	4	12	8	18	16		2074

13) недостаток этого метода в том, что мы вычисляем все значения $F(n)$ подряд, хотя для нахождения $F(24)$ нам не нужны значения $F(n)$ при чётных n
14) Ответ: 2074.

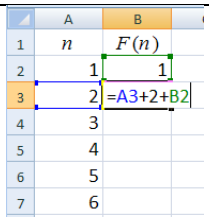
Общий недостаток первых двух методов – **большой объём ручных вычислений**, который приводит к большой вероятности ошибки.

Решение (использование табличного процессора):

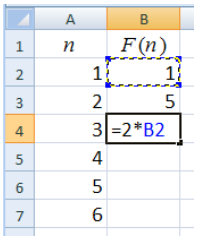
9) для выполнения большого объёма вычислений можно использовать табличный процессор; удобнее строить таблицу из двух столбцов (хотя можно, конечно использовать и две строки) – n и $F(n)$
10) сразу записываем известное значение $F(1) = 1$

	A	B
1	n	$F(n)$
2	1	1
3	2	
4	3	
5	4	
6	5	

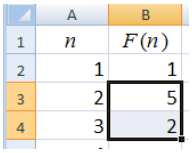
11) в ячейку B3, где вычисляется $F(2)$, записываем формулу для чётных n; здесь соответствующее значение n хранится в A3, а значение $F(n–1) = F(1)$ – в ячейке B2



12) в ячейку B4 , где вычисляется $F(3)$, записываем формулу для нечётных n; для этой ячейки соответствующее значение $F(n–2) = F(1)$ – в ячейке B2



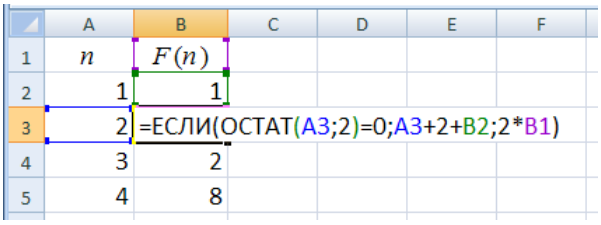
13) поскольку формулы чередуются (для чётных и нечётных n), выделяем **две** введённые формулы и «протягиваем» их, копируя вниз до строки с $n = 24$:



14) напротив значения $n = 24$ считываем ответ – 2074

15) Ответ: 2074.

16) можно обойтись и одной формулой, только в ней придётся использовать функцию ЕСЛИ:



Функция ОСТАТ вычисляет остаток от деления n на 2; если этот остаток равен нулю, то значение n чётное.

Решение (составление программы – рекурсивная функция):

6) составление программы – самый простой и наглядный способ решения задачи; для этого нужно просто реализовать заданные формулы в виде функции на языке программирования;

7) эта задача не должна представлять какой-то сложности; главная проблема – не получить бесконечную рекурсию; для этого рекомендуется сразу в начале функции записать условие окончания рекурсии, которое задаётся условием «$F(n) = 1$ при $n = 1$», и сразу выйти из функции

8) программа на языке Python:

```
def F( n ):
  if n == 1: return 1
  if n % 2 == 0:
    return n + 2 + F(n-1)
  else:
    return 2 * F(n-2)

print( F(24) )
```

9) программа на языке Паскаль:

```
function F( n: integer ): integer;
```

```
begin
  if n = 1 then begin
    Result := 1;
    Exit;
  end;
  if n mod 2 = 0 then
    Result := n + 2 + F(n-1)
  else
    Result := 2 * F(n-2);
end;
begin
  writeln( F(24) )
end.
```

10) программа на языке C++:

```
#include <iostream>
using namespace std;

int F( int n )
{
  if( n == 1 ) return 1;
  if( n % 2 == 0 )
    return n + 2 + F(n-1);
  else
    return 2 * F(n-2);
}
int main()
{
  cout << F(24);
}
```

Решение (составление программы – динамическое программирование):

15) если рекурсивная функция вызывает сама себя несколько раз (для рассматриваемой задачи это не так, но вполне может быть…), при больших значениях аргумента функция $F(n)$ может вычисляться очень (и даже недопустимо!) долго – это один из недостатков рекурсивной реализации
16) во многих случаях достаточно просто заменить рекурсивную функцию нерекурсивной (в принципе, это можно сделать всегда, вопрос лишь в том, какие усилия придётся для этого приложить)
17) один из простых приёмов – использование динамического программирования, которое позволяет свести вычисление значения функции, заданной рекурсивно, к заполнению массива (таблицы) – так же, как при ручном счёте (этот метод решения задачи описан выше)
18) пусть известно значение n; построим массив **a**, так чтобы значение элемента **a[n]** совпадало со значением $F(n)$
19) учитывая, что в Python и в C++ нумерация элементов массива начинается с нуля, будем использовать фиктивный нулевой элемент – не будем его использовать вообще; согласно условию, в первый элемент нужно записать число 1 («$F(n) = 1$ при $n = 1$»):

```
a = [0, 1]
```

20) затем перебираем в цикле все значения индексов элементов, начиная с 2 и до n включительно, вычисляя для каждого значение функции; фактически сначала вычисляется

$F(2)$ и записывается в элемент массива **a[2]**, затем находим $F(3)$ и записываем в элемент массива **a[3]** и т.д. :

```
for i in range(2, n+1):
  if i % 2 == 0:
    a.append( i + 2 + a[i-1] )
  else:
    a.append( 2*a[i-2] )
```

21) заметьте, что вместо рекурсивных вызовов здесь используются уже готовые значения $F(n)$ для меньших значений n, ранее записанные в массив **a** (см. выделение синим цветом)
22) теперь можно построить функцию, которая возвращает значение **a[n]**:

```
def F( n ):
  a = [0, 1]
  for i in range(2, n+1):
    if i % 2 == 0:
      a.append( i + 2 + a[i-1] )
    else:
      a.append( 2*a[i-2] )
  return a[n]
```

23) при вызове

```
print( F(24) )
```

эта функция возвращает тот же результат 2074, что и рекурсивная функция
24) Ответ: 2074.
25) преимущество такого подхода проявится, если функция вызывает сама себя несколько раз; в этом случае каждое значение будет вычисляться только один раз и сохраняться в массиве; когда это значение понадобится снова, оно будет взято сразу из массива в готовом виде (этот приём иногда называют *мемоизацией* – запоминанием)
26) учитывая, что преобразования программы не слишком простые, рекомендуется использовать такой метод на экзамене только тогда, когда «лобовое» решение работает слишком медленно (вы не смогли дождаться результата); помните, что любой шаг в сторону от простейшего решения может стать источником дополнительных ошибок
27) решение на языке Паскаль (элементы массива нумеруются с единицы):

```
function F( n: integer ): integer;
var a: array[1..100] of integer;
    i: integer;
begin
  a[1] := 1;
  for i:=2 to n do
    if i mod 2 = 0 then
      a[i] := i + 2 + a[i-1]
    else
      a[i] := 2 * a[i-2];
  Result := a[n];
end;
begin
  writeln( F(24) )
end.
```

28) решение на языке C++

```
#include <iostream>
using namespace std;
```

```
int F( int n )
{
  int i, a[100] = {};
  a[1] = 1;
  for( i = 2; i <= n; i++ )
    if( i % 2 == 0 )
      a[i] = i + 2 + a[i-1];
    else
      a[i] = 2 * a[i-2];
  return a[n];
}
int main()
{
  cout << F(24);
}
```

Решение (составление программы – вообще без массива и рекурсии):

Описанный далее метод решения задачи, в принципе, можно использовать, но при сдаче экзамена он **не имеет никаких преимуществ** перед методом динамического программирования, и в то же время повышает ваши шансы сделать ошибку; поэтому подумайте дважды (трижды, четырежды, …), прежде чем попытаться «блеснуть мастерством», к сожалению, при текущей модели экзамена это никто не оценит…

1) в данной задаче можно заметить, что очередное значение $F(n)$ зависит только от $F(n–1)$ или от $F(n–2)$; следовательно, знания значений $F(n–1)$ и $F(n–2)$ всегда будет достаточно для вычисления $F(n)$
2) поэтому можно вычислить $F(n)$, вообще не используя массив, нужно только хранить значения $F(n–1)$ и $F(n–2)$ в переменных
3) в приведённой далее программе переменные **fn1** и **fn2** хранят соответственно значения $F(n–1)$ и $F(n–2)$; начальное значение **fn1** должно быть равно 1 ($F(1) = 1$), а начальное значение **fn2** можно выбрать любым (например, 0), потому что на первой итерации цикла (при вычислении $F(2)$) оно все равно не используется

```
def F( n ):
  fn2 = 0  # эту строку можно удалить
  fn1 = 1
  for i in range(2, n+1):
    if i % 2 == 0:
      fn = i + 2 + fn1
    else:
      fn = 2*fn2
    fn2, fn1 = fn1, fn
  return f
```

Очередное значение $F(n)$ хранится в переменной **fn**. Строка

```
fn2, fn1 = fn1, fn
```

выполняет сдвиг переменных при переходе к следующему значению n: $F(n–1)$ записывается на место $F(n–2)$, а $F(n)$ – на место $F(n–1)$. При вызове

```
print( F(24) )
```

эта функция возвращает тот же результат 2074, что и рекурсивная функция

4) Ответ: 2074.
5) решение на языке Паскаль:

```
function F( n: integer ): integer;
var fn1, fn2, fn, i: integer;
begin
  fn1 := 1;
  for i:=2 to n do begin
    if i mod 2 = 0 then
      fn := i + 2 + fn1
    else
      fn := 2 * fn2;
    fn2 := fn1;
    fn1 := fn;
  end;
  Result := fn;
end;
begin
  writeln( F(24) )
end.
```

6) решение на языке C++

```
#include <iostream>
using namespace std;
int F( int n )
{
  int i, fn, fn1, fn2;
  fn1 = 1;
  for( i = 2; i <= n; i++ ) {
    if( i % 2 == 0 )
      fn = i + 2 + fn1;
    else
      fn = 2 * fn2;
    fn2 = fn1;
    fn1 = fn;
    }
  return fn;
}
int main()
{
  cout << F(24);
}
```

Задачи для тренировки:

1) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 1$ при $n = 1$
   $F(n) = 2 \cdot F(n–1) + n + 3$, если $n > 1$
   Чему равно значение функции $F(19)$?
2) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 3$ при $n = 1$
   $F(n) = 2 \cdot F(n–1) – n + 1$, если $n > 1$
   Чему равно значение функции $F(21)$?
3) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 2$ при $n = 1$
   $F(n) = F(n–1) + 5n^2$, если $n > 1$
   Чему равно значение функции $F(39)$?
4) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 2$ при $n \le 1$
   $F(n) = F(n–1) + F(n–2) + 2n + 4$, если $n > 1$
   Чему равно значение функции $F(25)$?
5) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 3$ при $n \le 1$
   $F(n) = F(n–1) + 2 \cdot F(n–2) – 5$, если $n > 1$
   Чему равно значение функции $F(22)$?
6) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 2$ при $n \le 1$
   $F(n) = F(n–1) + F(n–2) + 4n$, если $n > 1$
   Чему равно значение функции $F(24)$?
7) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = n$ при $n > 15$
   $F(n) = 2 \cdot F(n+1) + 5n + 2$, если $n \le 15$
   Чему равно значение функции $F(2)$?
8) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = n$ при $n > 18$
   $F(n) = 3 \cdot F(n+1) + n + 8$, если $n \le 18$
   Чему равно значение функции $F(9)$?
9) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = n – 3$ при $n > 16$
   $F(n) = 2 \cdot F(n+1) + 2n + 3$, если $n \le 16$
   Чему равно значение функции $F(2)$?
10) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 2n – 5$ при $n > 12$
   $F(n) = 2 \cdot F(n+2) + n – 4$, если $n \le 12$
   Чему равно значение функции $F(1)$?
11) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:
   $F(n) = 1$ при $n = 1$
   $F(n) = 2 \cdot F(n–1)$, если $n > 1$ и чётно,
   $F(n) = 5n + F(n–2)$, если $n > 1$ и нечётно.
   Чему равно значение функции $F(64)$?
12) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n < 1$

$F(n) = n + 3 \cdot F(n–3)$, если $n \geq 1$ и чётно,

$F(n) = 5n + 2 \cdot F(n–5)$, если $n \geq 1$ и нечётно.

Чему равно значение функции $F(30)$?

13) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 2 \cdot n$ при $n < 3$

$F(n) = 3n + 5 + F(n–2)$, если $n \geq 3$ и чётно,

$F(n) = n + 2 \cdot F(n–6)$, если $n \geq 3$ и нечётно.

Чему равно значение функции $F(61)$?

14) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = –n$ при $n < 0$

$F(n) = 2n + 1 + F(n–3)$, если $n \geq 0$ и чётно,

$F(n) = 4n + 2 \cdot F(n–4)$, если $n \geq 0$ и нечётно.

Чему равно значение функции $F(33)$?

15) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 5–n$ при $n < 5$

$F(n) = 4 \cdot (n – 5) \cdot F(n–5)$, если $n \geq 5$ и делится на 3,

$F(n) = 3n + 2 \cdot F(n–1) + F(n–2)$, если $n \geq 5$ и не делится на 3.

Чему равно значение функции $F(20)$?

16) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 1+2n$ при $n < 5$

$F(n) = 2 \cdot (n + 1) \cdot F(n–2)$, если $n \geq 5$ и делится на 3,

$F(n) = 2 \cdot n + 1 + F(n–1) + 2 \cdot F(n–2)$, если $n \geq 5$ и не делится на 3.

Чему равно значение функции $F(15)$?

17) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n + 3$ при $n < 3$

$F(n) = (n + 2) \cdot F(n–4)$, если $n \geq 3$ и делится на 3,

$F(n) = n + F(n–1) + 2 \cdot F(n–2)$, если $n \geq 3$ и не делится на 3.

Чему равно значение функции $F(20)$?

18) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(1) = G(1) = 1$

$F(n) = 2 \cdot F(n–1) + G(n–1) – 2$, если $n > 1$

$G(n) = F(n–1) + 2 \cdot G(n–1)$, если $n > 1$

Чему равно значение $F(14) + G(14)$?

19) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(1) = G(1) = 1$

$F(n) = 2 \cdot F(n–1) + G(n–1) – 2n$, если $n > 1$

$G(n) = F(n–1) + 2 \cdot G(n–1) + n$, если $n > 1$

Чему равно значение $F(14) + G(14)$?

20) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(1) = G(1) = 1$

$F(n) = 3 \cdot F(n–1) + G(n–1) – n + 5$, если $n > 1$

$G(n) = F(n–1) + 3 \cdot G(n–1) – 3 \cdot n$, если $n > 1$

Чему равно значение $F(14) + G(14)$?

21) Определите, сколько символов * выведет эта процедура при вызове F(28):

Python	Паскаль	C++

`def F( n ):`
`  print('*')`
`  if n >= 1:`
	`procedure F( n: integer );`
	`begin`
	`  write('*');`
		`void F( int n )`
		`{`
		`  cout << '*';`

```
    print('*')
    F(n-1)
    F(n-2)
```

```
  if n >= 1 then begin
    write('*');
    F(n-1);
    F(n-2);
  end;
end;
```

```
  if( n >= 1 ) {
    cout << '*';
    F(n-1);
    F(n-2);
    }
}
```

22) Определите, сколько символов * выведет эта процедура при вызове F(35):

Python	Паскаль	C++

`def F( n ):`
`  print('*')`
`  if n >= 1:`
`  print('*')`
`  F(n-1)`
`  F(n-2)`
`  print('*')`
	`procedure F( n: integer );`
	`begin`
	`  write('*');`
	`  if n >= 1 then begin`
	`  write('*');`
	`  F(n-1);`
	`  F(n-2);`
	`  write('*');`
	`  end;`
	`end;`
		`void F( int n )`
		`{`
		`  cout << '*';`
		`  if( n >= 1 ) {`
		`  cout << '*';`
		`  F(n-1);`
		`  F(n-2);`
		`  cout << '*';`
		`  }`
		`}`

23) Определите, сколько символов * выведет эта процедура при вызове F(40):

Python	Паскаль	C++

`def F( n ):`
`  print('*')`
`  if n >= 1:`
`  print('*')`
`  F(n-1)`
`  F(n-3)`
`  print('*')`
	`procedure F( n: integer );`
	`begin`
	`  write('*');`
	`  if n >= 1 then begin`
	`  write('*');`
	`  F(n-1);`
	`  F(n-3);`
	`  write('*');`
	`  end;`
	`end;`
		`void F( int n )`
		`{`
		`  cout << '*';`
		`  if( n >= 1 ) {`
		`  cout << '*';`
		`  F(n-1);`
		`  F(n-3);`
		`  cout << '*';`
		`  }`
		`}`

24) Определите, сколько символов * выведет эта процедура при вызове F(280):

Python	Паскаль	C++

`def F( n ):`
`  print('*')`
`  if n >= 1:`
`  print('*')`
`  F(n-1)`
`  F(n//3)`
`  print('*')`
	`procedure F( n: integer );`
	`begin`
	`  write('*');`
	`  if n >= 1 then begin`
	`  write('*');`
	`  F(n-1);`
	`  F(n div 3);`
	`  write('*');`
	`  end;`
	`end;`
		`void F( int n )`
		`{`
		`  cout << '*';`
		`  if( n >= 1 ) {`
		`  cout << '*';`
		`  F(n-1);`
		`  F(n/3);`
		`  cout << '*';`
		`  }`
		`}`

25) Определите, сколько символов * выведет эта процедура при вызове F(140):

Python	Паскаль	C++

`def F( n ):`
`  print('*')`
`  if n >= 1:`
`  print('*')`
`  F(n-1)`
`  F(n//2)`
	`procedure F( n: integer );`
	`begin`
	`  write('*');`
	`  if n >= 1 then begin`
	`  write('*');`
	`  F(n-1);`
		`void F( int n )`
		`{`
		`  cout << '*';`
		`  if( n >= 1 ) {`
		`  cout << '*';`
		`  F(n-1);`

	F(n div 2); end; end;	F(n/2); } }

26) Определите наименьшее значение n, при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 1000000. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующую сумму выведенных чисел.

Python	Паскаль	C++

def F(n):
  print(n+1)
  if n > 1:
  print(n+5)
  F(n-1)
  F(n-2)
	procedure F
	  (n: integer);
	begin
	  writeln(n+1);
	  if n > 1 then begin
	  writeln(n+5);
	  F(n-1);
	  F(n-2);
	  end;
	end;
		void F(int n)
		{
		  cout << n+1 << endl;
		  if(n > 1) {
		  cout << n+5 << endl;
		  F(n-1);
		  F(n-2);
		  }
		}

27) Определите наименьшее значение n, при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 1000000. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующую сумму выведенных чисел.

Python	Паскаль	C++

def F(n):
  print(n+1)
  if n > 1:
  print(2*n)
  F(n-1)
  F(n-3)
	procedure F
	  (n: integer);
	begin
	  writeln(n+1);
	  if n > 1 then begin
	  writeln(2*n);
	  F(n-1);
	  F(n-3);
	  end;
	end;
		void F(int n)
		{
		  cout << n+1 << endl;
		  if(n > 1) {
		  cout << 2*n << endl;
		  F(n-1);
		  F(n-3);
		  }
		}

28) Определите наименьшее значение n, при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 5000000. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующую сумму выведенных чисел.

Python	Паскаль	C++

def F(n):
  print(2*n+1)
  if n > 1:
  print(3*n-8)
  F(n-1)
  F(n-4)
	procedure F
	  (n: integer);
	begin
	  writeln(2*n+1);
	  if n > 1 then begin
	  writeln(3*n-8);
	  F(n-1);
	  F(n-4);
	  end;
	end;
		void F(int n)
		{
		  cout << 2*n+1 << endl;
		  if(n > 1) {
		  cout << 3*n-8
		  << endl;
		  F(n-1);
		  F(n-4);
		  }
		}

29) Определите наименьшее значение n, при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 3200000. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующую сумму выведенных чисел.

Python	Паскаль	C++

```
def F( n ):
  print(n-5)
  if n > 1:
    print(n+8)
    F(n-2)
    F(n-3)
```

```
procedure F
    ( n: integer );
begin
  writeln(n-5);
  if n > 1 then begin
    writeln(n+8);
    F(n-2);
    F(n-3);
  end;
end;
```

```
void F( int n )
{
  cout << n-5 << endl;
  if( n > 1 ) {
    cout << n+8 << endl;
    F(n-2);
    F(n-3);
    }
}
```

30) Определите наименьшее значение n, при котором сумма чисел, которые будут выведены при вызове $F(n)$, будет больше 3200000. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующую сумму выведенных чисел.

Python	Паскаль	C++

def F(n):
  print(n*n)
  if n > 1:
  print(2*n+1)
  F(n-2)
  F(n//3)
	procedure F
	  (n: integer);
	begin
	  writeln(n*n);
	  if n > 1 then begin
	  writeln(2*n+1);
	  F(n-2);
	  F(n div 3);
	  end;
	end;
		void F(int n)
		{
		  cout << n*n << endl;
		  if(n > 1) {
		  cout << 2*n+1 << endl;
		  F(n-2);
		  F(n/3);
		  }
		}

31) (**Д.Ф. Муфаззалов**) Определите наименьшее значение n, при котором значение $F(n)$, будет больше числа 320. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующее значение $F(n)$.

Python	Паскаль	C++

def F(n):
  if n>0:
  return n%10*F(n//10)
  else: return 1
	function F
	(n: integer): integer;
	begin
	 if n > 0 then
	  F:= n mod 10*
	  F(n div 10)
	  else
	  F:= 1;
	end;
		int F(int n)
		{
		if(n)
		  return
		  n%10*F(n/10);
		else return 1;
		}

32) (**Д.Ф. Муфаззалов**) Определите наибольшее трехзначное значение n, при котором значение $F(n)$, будет больше числа 7. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующее значение $F(n)$.

Python	Паскаль	C++

def F(n):
  if n<10:
  return n
  else:
  m=F(n//10)
  d=m%10;
	function F(n:
	integer): integer;
	var m,d: byte;
	begin
	 if n < 10 then F:=n
	 else begin
		int F(int n)
		{
		if(n < 10)
		  return n;
		else {
		 int m = F(n/10),

```
    if m<d: return d
    else: return m
```

```
    m:= F(n div 10);
    d:= m mod 10;
    if m < d then F:=d
    else F := m
 end
end;
```

```
     d = m%10;
 if( m < d )
   return d;
 else
  return m;
  }
}
```

33) (**Д.Ф. Муфаззалов**) Определите наименьшее значение n такое, что последнее выведенное число при вызове $F(n)$ будет больше числа 32. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующее значение $F(n)$.

Python	Паскаль	C++

def F(n):
  print(n)
  if n>0:
  d=n%10+F(n//10)
  print(d)
  return d
  else: return 0
	function F(n: integer):
	integer;
	var d:integer;
	begin
	 writeln(N);
	 if n > 0 then begin
	  d := n mod 10+
	  F(n div 10);
	  writeln(d);
	  F := d
	 end
	 else F:= 0;
	end;
		int F(int n)
		{
		cout << n << endl;
		if (n){
		  int d = n % 10 +
		  F(n/10);
		  cout << d << endl;
		  return d;
		  }
		else return 0;
		}

34) (**Д.Ф. Муфаззалов**) Определите наименьшее число n такое, что при вызове $F(n)$ второе выведенное число будет больше числа 51. Запишите в ответе сначала найденное значение n, а затем через пробел – соответствующее значение $F(n)$.

Python	Паскаль	C++

def F(n):
  print(n)
  if n > 0:
  d = (n%10 +
  F(n//10))
  print(d)
  return d
  else:
  return 0
	function f(n: integer):
	integer;
	var d:integer;
	begin
	 writeln(N);
	 if n > 0 then begin
	  d := n mod 10 +
	  F(n div 10);
	  writeln(d);
	  F := d
	 end
	 else F:= 0;
	end;
		int F(int n)
		{
		cout << n << endl;
		if(n) {
		  int d = n%10 +
		  F(n/10);
		  cout << d << endl;
		  return d;
		  }
		else
		  return 0;
		}

35) **(Д.Ф. Муфаззалов, г. Уфа)** Определите наименьшее значение суммы $n+m$ такое, что значение $F(n, m)$ больше числа 15 и выполняется условие $n \neq m$, n и m – натуральные числа. Запишите в ответе сначала значения n и m, при которых указанная сумма достигается, в порядке неубывания, а затем – соответствующее значение $F(n, m)$. Числа в ответе разделяйте пробелом.

Python	Паскаль	C++
def F(n,m): if n<m:	function F(n,m: integer): integer;	int F(int n, int m) {

```
 n,m = m,n
if n != m:
  return F(n-m,m)
else:
  return n
```

```
begin
 if n > m then
  F:= F(n-m,m)
 else
  if n < m then
    F:= F(n,m-n)
  else
    F:= n;
end;
```

```
 if( n > m )
   return F(n-m,m);
 else
   if( n < m )
     return F(m-n,n);
   else
     return n;
}
```

36) **(Д.Ф. Муфаззалов, г. Уфа)** Определите количество различных значений n таких, что n и m – натуральные числа, находящиеся в диапазоне [100; 1000], а значение $F(n, m)$ равно числу 30.

```
def F(n,m):
 if m == 0:
   return n
 else:
   return F(m,n%m)
```

```
function F(n,m:
   integer): integer;
begin
 if m = 0 then
   F:= n
 else
   F:= F(m, n mod m)
end;
```

```
int F(int n, int m)
{
if( m == 0 )
  return n;
else
  return F(m, n%m);
}
```

37) **(Д.Ф. Муфаззалов, г. Уфа)** Определите количество различных натуральных значений n таких, что значение $F(n, 2)$ находится в диапазоне [100; 1000].

```
def F(n,m):
 if m == 0:
   d = 1
 else:
   d = n*F(n, m-1)
 return d
```

```
function F(n,m:
integer): integer;
begin
 if m = 0 then
   F:= 1
 else
   F:= n*F(n,m-1)
end;
```

```
int F(int n, int m)
{
if( m == 0 )
  return 1;
else
  return n*F(n,m-1);
}
```

38) **(Д.Ф. Муфаззалов, г. Уфа)** Определите количество различных значений n таких, что n и m – натуральные числа, а значение $F(n, m)$ равно числу 30.

```
def F(n,m):
 if m == 0:
  d = 0
 else:
  d = n+F(n, m-1)
 return d
```

```
function F(n,m:
integer): integer;
begin
 if m == 0 then
   F:= 0
 else
   F:= n + F(n,m-1)
end;
```

```
int F(int n, int m)
{
if( m == 0 )
  return 0;
else
  return n+F(n,m-1);
}
```

39) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(n) = G(n) = 1$ при $n = 1$

$F(n) = F(n–1) – 2 \cdot G(n–1)$, при $n > 1$

$G(n) = F(n–1) + 2 \cdot G(n–1)$, при $n > 1$

Чему равно значение функции $G(21)$?

40) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(n) = G(n) = 1$ при $n = 1$

$F(n) = F(n–1) – n \cdot G(n–1)$, при $n > 1$

$G(n) = F(n–1) + 2 \cdot G(n–1)$, при $n > 1$

Чему равно значение функции $G(18)$?

41) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(n) = G(n) = 1$ при $n = 1$

$F(n) = F(n–1) – 2 \cdot G(n–1)$, при $n > 1$

$G(n) = F(n–1) + G(n–1) + n$, при $n > 1$

Чему равна сумма цифр значения функции $G(36)$?

42) Алгоритм вычисления функций $F(n)$ и $G(n)$ задан следующими соотношениями:

$F(n) = G(n) = 1$ при $n = 1$

$F(n) = F(n–1) + 3 \cdot G(n–1)$, при $n > 1$

$G(n) = F(n–1) – 2 \cdot G(n–1)$, при $n > 1$

Чему равна сумма цифр значения функции $F(18)$?

43) (**К. Амеличев**) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 3$;

$F(n) = n$ // $4 + F(n–3)$ при $3 < n \le 32$;

$F(n) = 2 \cdot F(n–5)$ при $n > 32$

Здесь // обозначает деление нацело. В качестве ответа на задание выведите значение $F(100)$.

44) (**К. Амеличев**) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 3$;

$F(n) = n * n * n + F(n – 1)$, если $n > 3$ и дает остаток 0 при делении на 3

$F(n) = 4 + F(n // 3)$, если $n > 3$ и дает остаток 1 при делении на 3

$F(n) = n * n + F(n – 2)$, если $n > 3$ и дает остаток 2 при делении на 3

Здесь // обозначает деление нацело. В качестве ответа на задание выведите значение $F(100)$.

45) (**К. Амеличев**) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 10$;

$F(n) = n$ // $4 + F(n–10)$ при $10 < n \le 36$;

$F(n) = 2 \cdot F(n–5)$ при $n > 36$

Здесь // обозначает деление нацело. В качестве ответа на задание выведите значение $F(100)$.

46) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 3$;

$F(n) = 2 \cdot n \cdot n + F(n – 1)$ при чётных $n > 3$;

$F(n) = n \cdot n \cdot n + n + F(n – 1)$ при нечётных $n > 3$;

Определите количество натуральных значений n, при которых $F(n)$ меньше, чем 10^7.

47) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 3$;

$F(n) = F(n – 1) + 2 \cdot F(n / 2)$ при чётных $n > 3$;

$F(n) = F(n – 1) + F(n – 3)$ при нечётных $n > 3$;

Определите количество натуральных значений n, при которых $F(n)$ меньше, чем 10^8.

48) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 3$;

$F(n) = n + F(n – 1)$ при чётных $n > 3$;

$F(n) = n \cdot n + F(n – 2)$ при нечётных $n > 3$;

Определите количество натуральных значений n, при которых $F(n)$ меньше, чем 10^8.

49) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 3$;

$F(n) = 2 \cdot n + F(n – 1)$ при чётных $n > 3$;

$F(n) = n \cdot n + F(n – 2)$ при нечётных $n > 3$;

Определите количество натуральных значений n из отрезка [1; 100], при которых значение $F(n)$ кратно 3.

50) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n$ при $n \le 3$;

$F(n) = n + 3 + F(n-1)$ при чётных $n > 3$;

$F(n) = n \cdot n + F(n-2)$ при нечётных $n > 3$;

Определите количество натуральных значений n из отрезка [1; 1000], при которых значение $F(n)$ кратно 7.

51) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 1$ при $n \le 1$;

$F(n) = n \cdot F(n-1)$ при чётных $n > 1$;

$F(n) = n + F(n-2)$ при нечётных $n > 1$;

Определите значение $F(84)$.

52) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 1$ при $n \le 1$;

$F(n) = n + F(n-1)$ при чётных $n > 1$;

$F(n) = n \cdot n + F(n-2)$ при нечётных $n > 1$;

Определите значение $F(80)$.

53) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n - 5$ при $n > 15$

$F(n) = n \cdot F(n+2) + n + F(n+3)$, если $n \le 15$

Чему равна сумма цифр значения функции $F(1)$?

54) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 2 \cdot n \cdot n \cdot n + n \cdot n$ при $n > 25$

$F(n) = F(n+2) + 2 \cdot F(n+3)$, если $n \le 25$

Чему равна сумма цифр значения функции $F(2)$?

55) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 2 \cdot n \cdot n \cdot n + 1$ при $n > 25$

$F(n) = F(n+2) + 2 \cdot F(n+3)$, если $n \le 25$

Определите количество натуральных значений n из отрезка [1; 1000], при которых значение $F(n)$ кратно 11.

56) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n \cdot n + n$ при $n > 20$

$F(n) = 3 \cdot F(n+1) + F(n+3)$, при чётных $n \le 20$

$F(n) = F(n+2) + 2 \cdot F(n+3)$, при нечётных $n \le 20$

Определите количество натуральных значений n из отрезка [1; 1000], при которых значение $F(n)$ не содержит цифру 1.

57) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n + 2 \cdot n + 1$, при $n > 25$

$F(n) = 2 \cdot F(n+1) + F(n+3)$, при чётных $n \le 25$

$F(n) = F(n+2) + 3 \cdot F(n+5)$, при нечётных $n \le 25$

Определите количество натуральных значений n из отрезка [1; 1000], при которых значение $F(n)$ не содержит цифру 0.

58) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n + 3 \cdot n + 5$, при $n > 30$

$F(n) = 2 \cdot F(n+1) + F(n+4)$, при чётных $n \le 30$

$F(n) = F(n+2) + 3 \cdot F(n+5)$, при нечётных $n \le 30$

Определите количество натуральных значений n из отрезка [1; 1000], при которых значение $F(n)$ содержит не менее двух значащих цифр 0 (в любых разрядах).

59) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n + 5 \cdot n + 4$, при $n > 30$

$F(n) = F(n+1) + 3 \cdot F(n+4)$, при чётных $n \le 30$

$F(n) = 2 \cdot F(n+2) + F(n+5)$, при нечётных $n \le 30$

Определите количество натуральных значений n из отрезка [1; 1000], для которых сумма цифр значения $F(n)$ равна 27.

60) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n + 4 \cdot n + 3$, при $n > 25$

$F(n) = F(n+1) + 2 \cdot F(n+4)$, при $n \le 25$, кратных 3

$F(n) = F(n+2) + 3 \cdot F(n+5)$, при $n \le 25$, не кратных 3

Определите количество натуральных значений n из отрезка [1; 1000], для которых сумма цифр значения $F(n)$ равна 24.

61) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n + 3 \cdot n + 9$, при $n \le 15$

$F(n) = F(n–1) + n – 2$, при $n > 15$, кратных 3

$F(n) = F(n–2) + n + 2$, при $n > 15$, не кратных 3

Определите количество натуральных значений n из отрезка [1; 1000], для которых все цифры значения $F(n)$ чётные.

62) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = 2 \cdot n \cdot n + 4 \cdot n + 3$, при $n \le 15$

$F(n) = F(n–1) + n \cdot n + 3$, при $n > 15$, кратных 3

$F(n) = F(n–2) + n – 6$, при $n > 15$, не кратных 3

Определите количество натуральных значений n из отрезка [1; 1000], для которых все цифры значения $F(n)$ нечётные.

63) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n \cdot n + n \cdot n + 1$, при $n \le 13$

$F(n) = F(n–1) + 2 \cdot n \cdot n – 3$, при $n > 13$, кратных 3

$F(n) = F(n–2) + 3 \cdot n + 6$, при $n > 13$, не кратных 3

Определите количество натуральных значений n из отрезка [1; 1000], для которых все цифры значения $F(n)$ нечётные.

64) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n + 3$, при $n \le 18$

$F(n) = (n \,//\, 3) \cdot F(n \,//\, 3) + n – 12$, при $n > 18$, кратных 3

$F(n) = F(n–1) + n \cdot n + 5$, при $n > 18$, не кратных 3

Здесь «//» обозначает деление нацело. Определите количество натуральных значений n из отрезка [1; 800], для которых все цифры значения $F(n)$ чётные.

65) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n + 15$, при $n \le 5$

$F(n) = F(n \,//\, 2) + n \cdot n \cdot n – 1$, при чётных $n > 5$

$F(n) = F(n–1) + 2 \cdot n \cdot n + 1$, при нечётных $n > 5$

Здесь «//» обозначает деление нацело. Определите количество натуральных значений n из отрезка [1; 1000], для которых значения $F(n)$ содержит не менее двух цифр 8.

66) Алгоритм вычисления функции $F(n)$ задан следующими соотношениями:

$F(n) = n \cdot n + 11$, при $n \le 15$

$F(n) = F(n \,//\, 2) + n \cdot n \cdot n – 5 \cdot n$, при чётных $n > 15$

$F(n) = F(n–1) + 2 \cdot n + 3$, при нечётных $n > 15$

Здесь «//» обозначает деление нацело. Определите количество натуральных значений n из отрезка [1; 1000], для которых значения $F(n)$ содержит не менее трёх цифр 6.

67) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – натуральное число, задан следующими соотношениями:

$F(n) = n + 1$ при $n < 3$,

$F(n) = n + 2*F(n + 2)$, когда $n \geq 3$ и четно,

$F(n) = F(n – 2) + n – 2$, когда $n \geq 3$ и нечетно.

Сколько существует чисел n, для которых значение $F(n)$ определено и будет трехзначным?

68) Алгоритм вычисления функций $F(n)$, где n – натуральное число, задан следующими соотношениями:

$F(n) = n + 1$ при $n < 3$,

$F(n) = F(n – 2) + n – 2$, когда $n \geq 3$ и четно,

$F(n) = F(n + 2) + n + 2$, когда $n \geq 3$ и нечетно.

Сколько существует чисел n, для которых значение $F(n)$ определено и будет пятизначным?

69) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n – 1$ при $n < 4$,

$F(n) = n + 2 \cdot F(n – 1)$, когда $n \geq 4$ и кратно 3,

$F(n) = F(n – 2) + F(n – 3)$, когда $n \geq 4$ и не кратно 3.

Чему равна сумма цифр значения $F(25)$?

70) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = 1$ при $n = 0$,

$F(n) = 2 \cdot F(1 – n) + 3 \cdot F(n – 1) + 2$, когда $n > 0$,

$F(n) = –F(–n)$, когда $n < 0$.

Чему равна сумма цифр значения $F(50)$?

71) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = 5$ при $n = 0$,

$F(n) = 3 \cdot F(n – 4)$, когда $n > 0$,

$F(n) = F(n + 3)$, когда $n < 0$.

Чему равно значение $F(43)$?

72) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = F(n+2) + 2 \cdot F(3 \cdot n)$ при $n \leq 70$,

$F(n) = n – 50$, когда $n > 70$.

Чему равно значение $F(40)$?

73) (**Е. Джобс**) Алгоритмы вычисления функций $F(n)$ и $G(n)$ где n – целое число, заданы следующими соотношениями (// обозначает деление нацело):

$F(n) = n$, при $n < 50$,

$F(n) = 2 \cdot G(50 – n // 2)$, при $n > 49$,

$G(n) = 10$, при $n > 40$,

$G(n) = 30 + F(n + 600 // n)$, при $n < 41$

Чему равно значение $F(80)$?

74) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = 1$, при $n < –100000$,

$F(n) = F(n – 1) + 3 \cdot F(n – 3) + 2$, при $n > 10$,

$F(n) = – F(n – 1)$ для остальных случаев.

Чему равно значение $F(20)$?

75) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 1$,

$F(n) = 1 + F(n / 2)$, когда $n > 1$ и чётное,

$F(n) = 1 + F(n + 2)$, когда $n > 1$ и нечётное.

Назовите минимальное значение n, для которого $F(n) = 16$.

76) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = 1$, при $n \le 1$,

$F(n) = 3 + F(n / 2 – 1)$, когда $n > 1$ и чётное,

$F(n) = n + F(n + 2)$, когда $n > 1$ и нечётное.

Назовите минимальное значение n, для которого $F(n) = 19$.

77) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 1$,

$F(n) = n + F(n / 3)$, когда $n > 1$ и делится на 3,

$F(n) = n + F(n + 3)$, когда $n > 1$ и не делится на 3.

Назовите минимальное значение n, для которого $F(n)$ определено и больше 100.

78) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 1$,

$F(n) = n + F(n / 3 – 1)$, когда $n > 1$ и делится на 3,

$F(n) = n + F(n + 3)$, когда $n > 1$ и не делится на 3.

Назовите минимальное значение n, для которого $F(n)$ определено и больше 1000.

79) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 5$,

$F(n) = n + F(n / 3 + 1)$, когда $n > 5$ и делится на 3,

$F(n) = n + F(n + 3)$, когда $n > 5$ и не делится на 3.

Назовите минимальное значение n, для которого $F(n)$ определено и больше 1000.

80) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 5$,

$F(n) = n + F(n / 3 + 2)$, когда $n > 5$ и делится на 3,

$F(n) = n + F(n + 3)$, когда $n > 5$ и не делится на 3.

Назовите минимальное значение n, для которого $F(n)$ определено и больше 1000.

81) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 5$,

$F(n) = n + F(n / 5 + 1)$, когда $n > 5$ и делится на 5,

$F(n) = n + F(n + 6)$, когда $n > 5$ и не делится на 5.

Назовите минимальное значение n, для которого $F(n)$ определено и больше 1000.

82) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 5$,

$F(n) = n + F(n / 2 – 1)$, когда $n > 5$ и делится на 4,

$F(n) = n + F(n + 2)$, когда $n > 5$ и не делится на 4.

Назовите максимальное значение n, для которого возможно вычислить $F(n)$.

83) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n \le 5$,

$F(n) = n + F(n / 2 – 3)$, когда $n > 5$ и делится на 8,

$F(n) = n + F(n + 4)$, когда $n > 5$ и не делится на 8.

Назовите максимальное значение n, для которого возможно вычислить $F(n)$.

84) (**А. Богданов**) Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$F(n) = n$, при $n < 2$,

$F(n) = F(n / 2) + 1$, когда $n \geq 2$ и чётное,

$F(n) = F(3n + 1) + 1$, когда $n \geq 2$ и нечётное.

Назовите количество значений n на отрезке [1;100], для которых $F(n)$ определено и больше 100.

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

$F(1) = 1$,

$F(n) = F(n / 2) + 1$, когда $n \geq 2$ и чётное,

$F(n) = F(n – 1) + n$, когда $n \geq 2$ и нечётное.

Назовите количество значений n на отрезке [1;100000], для которых $F(n)$ равно 16.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 2) + 1$, когда $n \geq 2$ и чётное,

$F(n) = F(n – 3) + 3$, когда $n \geq 2$ и нечётное.

Назовите количество значений n на отрезке [1;100000], для которых $F(n)$ равно 12.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 3) + 1$, когда $n \geq 2$ и делится на 3,

$F(n) = F(n – 2) + 5$, когда $n \geq 2$ и не делится на 3.

Назовите количество значений n на отрезке [1;100000], для которых $F(n)$ равно 55.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 3) – 1$, когда $n \geq 2$ и делится на 3,

$F(n) = F(n – 1) + 7$, когда $n \geq 2$ и не делится на 3.

Назовите количество значений n на отрезке [1;100000], для которых $F(n)$ равно 35.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 3) – 1$, когда $n \geq 2$ и делится на 3,

$F(n) = F(n – 1) + 17$, когда $n \geq 2$ и не делится на 3.

Назовите количество значений n на отрезке [1;100000], для которых $F(n)$ равно 43.

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

$F(1) = 1$,

$F(n) = F(n / 2) + 1$, когда $n \geq 2$ и чётное,

$F(n) = F(n – 1) + n$, когда $n \geq 2$ и нечётное.

Назовите минимальное значение n, для которого $F(n)$ равно 19.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 2) + 1$, когда $n \geq 2$ и чётное,

$F(n) = F(n – 3) + 3$, когда $n \geq 2$ и нечётное.

Назовите минимальное значение n, для которого $F(n)$ равно 31.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 3) + 1$, когда $n \geq 2$ и делится на 3,

$F(n) = F(n – 2) + 5$, когда $n \geq 2$ и не делится на 3.

Назовите минимальное значение n, для которого $F(n)$ равно 73.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 3) – 1$, когда $n \geq 2$ и делится на 3,

$F(n) = F(n – 1) + 7$, когда $n \geq 2$ и не делится на 3.

Назовите минимальное значение n, для которого $F(n)$ равно 111.

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

$F(n) = 1$, при $n < 2$,

$F(n) = F(n / 3) – 1$, когда $n \geq 2$ и делится на 3,

$F(n) = F(n – 1) + 17$, когда $n \geq 2$ и не делится на 3.

Назовите минимальное значение n, для которого $F(n)$ равно 110.

95) (**А. Богданов**) Алгоритмы вычисления функций $F(n)$ и $G(n)$ заданы следующими соотношениями (здесь // – операция деления нацело, % – остаток от деления):

$F(n) = n$, при $n < 10$,

$F(n) = F(G(n))$, при $n \geq 10$,

$G(n) = n$, при $n < 10$,

$G(n) = n \% 10 + G(n // 10)$, при $n \geq 10$.

Чему равно значение F(12345678987654321)?

96) (**А. Богданов**) Алгоритмы вычисления функций $F(n)$ и $G(n)$ заданы следующими соотношениями (здесь // – операция деления нацело, % – остаток от деления):

$F(n) = n$, при $n < 10$,

$F(n) = n \% 10 + F(n // 10)$, при $n \geq 10$.

$G(n) = n$, при $n < 10$,

$G(n) = G(F(n))$, при $n \geq 10$,

Чему равна сумма значений функции $G(n)$ для всех двузначных n?

97) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 0$,

$F(n) = F(n / 2)$, когда $n > 0$ и делится на 2,

$F(n) = F(n – 1) + 3$, когда $n > 0$ и не делится на 2.

Сколько существует значений n, принадлежащих отрезку [1; 1000], для которых $F(n)$ равно 18?

98) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 0$,

$F(n) = F(n / 2) + 3$, когда $n > 0$ и делится на 2,

$F(n) = 2 \cdot F(n – 1) + 1$, когда $n > 0$ и не делится на 2.

Сколько различных значений может принимать функция $F(n)$ при n, принадлежащих отрезку [1; 1000]?

99) (**А. Богданов**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 0$,

$F(n) = 1$, когда $0 < n < 3$,

$F(n) = F(n - 2) + F(n - 1)$, когда $n \geq 3$.

Определите четыре последние цифры числа F(47).

100) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(n) = n + 3$, при $n \leq 3$

$F(n) = F(n - 2) + n$, при $n > 3$ и четном значении $F(n-1)$,

$F(n) = F(n - 2) + 2 \cdot n$, при $n > 3$ и нечетном значении $F(n-1)$

Определите сумму значений, являющихся результатом вызова функции для значений n в диапазоне [40; 50].

101) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 1$, $F(1) = 3$

$F(n) = F(n-1) - F(n-2) + 3n$, при $n > 1$

Чему равно значение функции F(40)? В ответе запишите только целое число.

102) (**Е. Джобс**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 1$, $F(1) = 3$

$F(n) = F(n - 1) - F(n - 2) + 3n$, при $n > 1$ и n – четно

$F(n) = F(n - 2) - F(n - 3) + 2n$, при $n > 1$ и n – нечетно

Чему равно значение функции F(40)? В ответе запишите только целое число.

103) (**П. Волгин**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 1$

$F(n) = F(n-1)$, при $0 < n \leq 10$

$F(n) = 2{,}2 * F(n-3)$, при $10 < n < 100$

$F(n) = 1{,}7 * F(n-2)$, при $n \geq 100$

Чему равна целая часть значения функции F(22)?

104) (**П. Волгин**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 1$

$F(n) = F(n-1)$, при $0 < n \leq 10$

$F(n) = 2{,}2 * F(n-3)$, при $10 < n < 100$

$F(n) = 1{,}7 * F(n-2)$, при $n \geq 100$

Чему равна сумма цифр целой части F(40)?

105) (**П. Волгин**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 2$

$F(n) = F(n-1)$, при $0 < n \leq 15$

$F(n) = 1{,}6 * F(n-3)$, при $15 < n < 95$

$F(n) = 3{,}3 * F(n-2)$, при $n \geq 95$

Какая цифра встречается чаще всего в целой части значения функции F(33)?

106) (**П. Волгин**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 3$

$F(n) = F(n-1)$, при $0 < n \leq 15$

$F(n) = 2{,}5 * F(n-3)$, при $15 < n < 95$

$F(n) = 3{,}3 * F(n-2)$, при $n \geq 95$

С какой цифры начинается целая часть значения функции F(70)?

107) (**П. Волгин**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 3$

$F(n) = F(n–1)$, при $0 < n \le 15$

$F(n) = 2{,}5*F(n–3)$, при $15 < n < 100$

$F(n) = 3{,}3*F(n–2)$, при $n \ge 100$

С какой цифры начинается дробная часть значения функции F(100)?

108) (**П. Волгин**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(0) = 1$

$F(n) = F(n–1) + F(n–2)$, при чётном $n > 0$

$F(n) = 1{,}5*F(n–1)$, при нечётном $n > 0$

Сколько различных цифр встречается в целой части значения функции F(15)?

109) (**А. Богданов**) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(n) = 0$ при $n \le 2$ и $n = 8$

$F(n) = 1$ при $n = 3$

$F(n) = F(n–2) + F(n–1)$ при $n > 3$ и $n \ne 8$

Для какого значения n значение $F(n)$ будет равно 25?

110) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(n) = 0$ при $n = 0$

$F(n) = F(n/2) – 1$ при $n > 0$ для чётных n

$F(n) = 1 + F(n–1)$ при $n > 0$ для нечётных n

Сколько существует чисел n, меньших 1000, для которых значение $F(n)$ будет равно 0?

111) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(n) = 0$ при $n = 0$

$F(n) = F(n/2) – 2$ при $n > 0$ для чётных n

$F(n) = 2 + F(n–1)$ при $n > 0$ для нечётных n

Сколько существует чисел n, меньших 1000, для которых значение $F(n)$ будет равно –2?

112) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(n) = 0$ при $n = 0$

$F(n) = F(n/2) – 1$ при $n > 0$ для чётных n

$F(n) = 2 + F(n–1)$ при $n > 0$ для нечётных n

Сколько существует чисел n, меньших 1000, для которых значение $F(n)$ будет равно 3?

113) Алгоритм вычисления функции $F(n)$, где n – целое неотрицательное число, задан следующими соотношениями:

$F(n) = 0$ при $n = 0$

$F(n) = F(n/2) – 1$ при $n > 0$ для чётных n

$F(n) = 3 + F(n–1)$ при $n > 0$ для нечётных n

Сколько различных значений может принимать функция $F(n)$ для чисел n, меньших 1000?