Рекурсивные функции

Рекурсивная (самовызываемая) это функция, которая прямо или косвенно вызывает сама себя.

В ***косвенно рекурсивной*** функции находится обращение к другой функции, содержащей прямой или косвенный вызов первой.

Если в функции используется ее вызов, то это ***прямая рекурсия***.

При каждом обращении к рекурсивной функции в памяти создается новый набор объектов, использующихся в ее коде.

Рекурсивные алгоритмы эффективны в задачах, где рекурсия присутствует в определении обрабатываемых данных, поэтому они чаще всего используются для динамических данных с рекурсивной структурой (линейные и нелинейные списки).

В рекурсивных функциях необходимо выполнять следующие правила:

– при каждом вызове в функцию передавать измененные данные (параметры);

– рекурсивный процесс шаг за шагом должен упрощать задачу так, чтобы для нее появилось нерекурсивное решение.

***Пример* 1**. Заданы два числа *a* и *b*, большее из них разделить на меньшее, используя рекурсию.

```
double fun_rec (double, double);
void main (void)
{
 double a, b;
 cout << " Input a, b : ";
 cin >> a >> b;
 cout << " Result =  " << fun_rec (a, b) << endl;
}
```

```
double fun_rec ( double a, double b)   {
    if ( a < b ) return fun_rec ( b, a );
    else  return a / b;
}
```

Если $a \geq b$, условие не выполняется и функция возвращает нерекурсивный результат a / b.

Если условие выполнилось, то функция *fun_rec* обращается сама к себе, аргументы в вызове меняются местами и последующее обращение приводит к тому, что условие снова не выполняется и функция возвращает нерекурсивный, фактически равный b / a.

Пример* 2**. Функция для вычисления факториала ***неотрицательного значения ***k*** (для отрицательных значений можно добавить проверку до вызова функции):

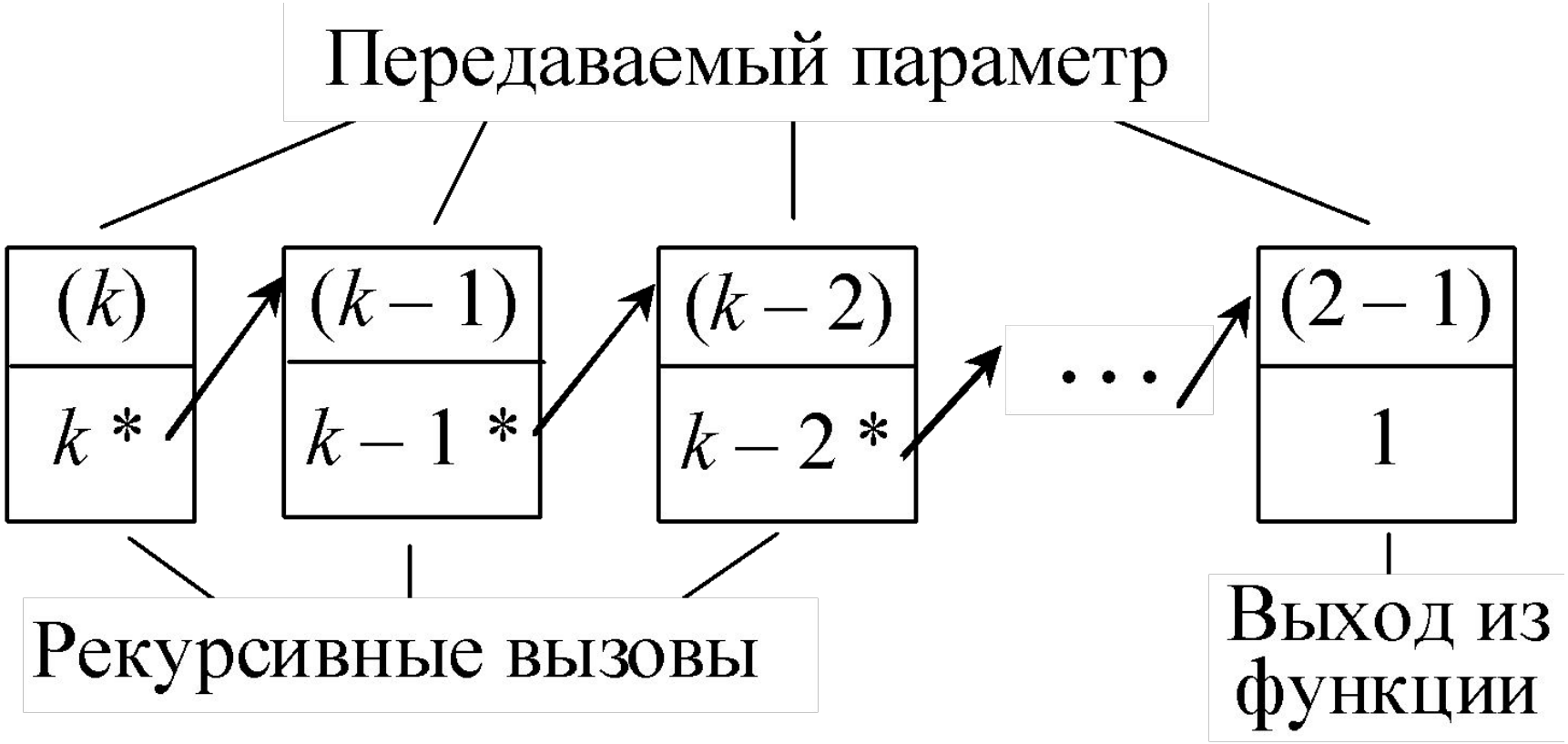
```
double fact_rec (int k)  {
 if ( k < 2 ) return 1;
 else
     return   k * fact_rec ( k – 1);
}
```

Для значений $k < 2$ (напомним, что $0! = 1$) функция возвращает **1**, в противном случае вызывается та же функция с измененным параметром $(k - 1)$ и результат умножается на текущее значение k.

Процесс выполняется до тех пор пока очередное уменьшенное на **1** значение k не станет меньше **2** и не приведет не к очередному вызову, а к выходу из функции, т.е. не выполнится

```
if ( k < 2 ) return 1;
```

Схема выполнения функции ***fact_rec***



Выполнив выход из функции, образуется следующая цепочка вычислений

1 * 2 * 3 * ... * (*k* – 2) * (*k* – 1) * *k*

Последнее значение **1** – результат выполнения условия ***k* < 2** при ***k* = 1**, т.е. последовательность рекурсивных обращений к функции ***fact_rec*** прекращается при вызове ***fact_rec* (1)**.

Именно этот вызов приводит к последнему значению **1** в произведении, т.к. последнее выражение, из которого вызывается функция, имеет вид: **2 * *fact* (2 – 1)**.

Блок-схема может иметь следующий вид