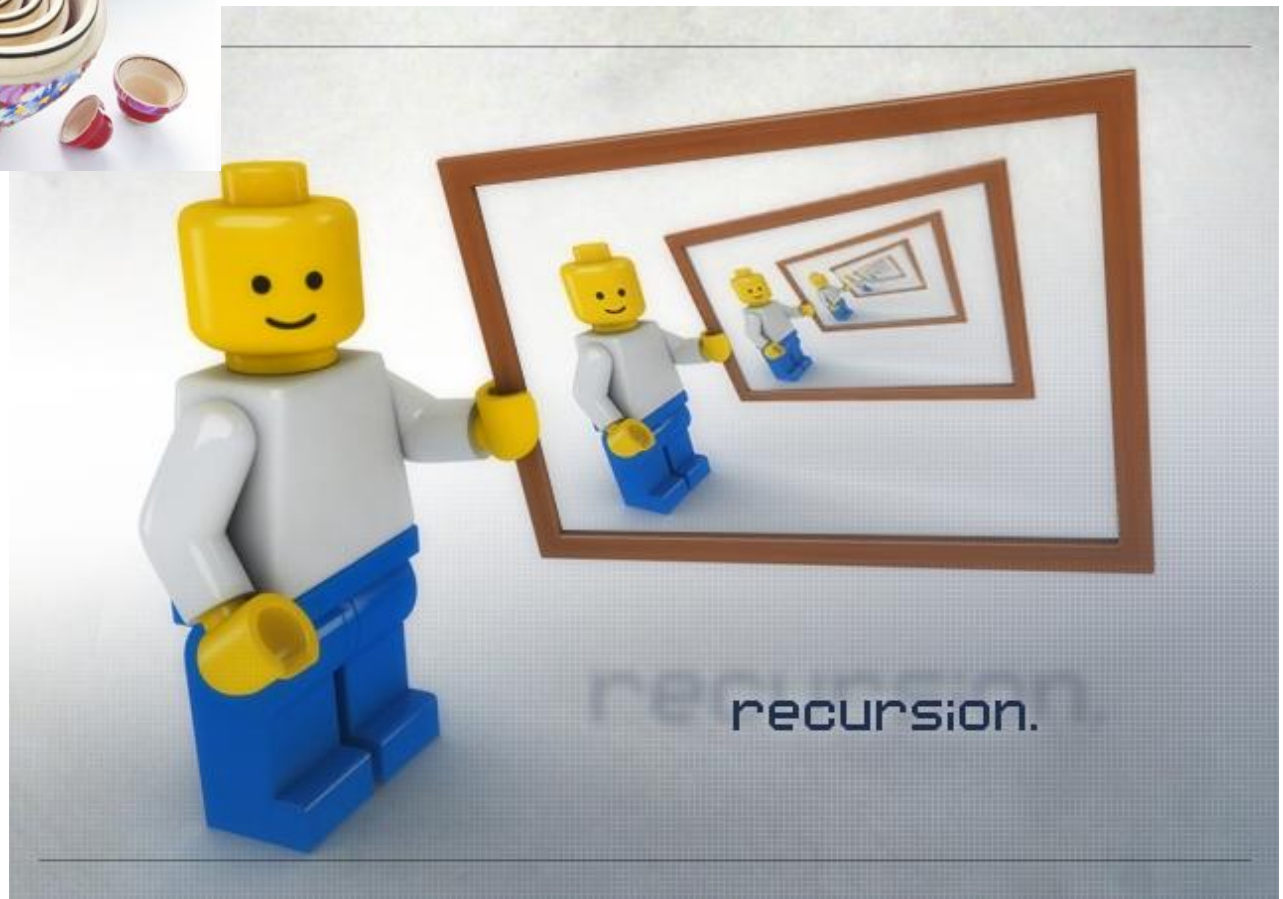
Программирование на языке Си

Рекурсия



Понятие рекурсии

Рекурсивные функции (лат. recursio – возвращение) – в вычислительной математике – функции, определенные на множестве натуральных чисел и принимающие значения того же множества.

Рекурсия – такой способ организации алгоритмического процесса, при котором функция в процессе выполнения входящих в ее состав операторов обращается сама к себе непосредственно либо через другие функции.

Рекурсивный алгоритм – это алгоритм, решающий задачу путем решения одного или нескольких более узких вариантов той же задачи.

Рекурсивные функции - такие функции, которые могут вызывать сами себя. При этом каждый раз под каждый вызов создается совершенно новый набор локальных переменных, отличный от набора вызывающей функции.

Виды рекурсии

Рекурсия может быть

прямой или **косвенной**.

Если функция вызывает саму себя, то речь идет о прямой рекурсии.

Если же функция вызывает другую функция, которая затем вызывает исходную функцию, тогда имеет место быть косвенная рекурсия.

Любую рекурсивную функцию можно сделать итеративной, т.е. с использованием циклов.

Рекурсивный подход обычно предпочитается итеративному подходу в случаях:

- когда рекурсия более естественно отражает математическую сторону задачи и приводит к программе, которая проще для понимания и отладки;
- если итеративное решение может не быть очевидным;
- когда используемые данные определены рекурсивно.

Параметры рекурсии

Количество вложенных вызовов функции или процедуры называется **глубиной рекурсии.**

Число рекурсивных вызовов в каждый конкретный момент времени, называется **текущим уровнем рекурсии.**

Задача о вычислении факториала

Факториал n - это произведение всех натуральных чисел до n включительно.

Например:

5! = 5 * 4 * 3 * 2 * 1 = 120
4! = 4 * 3 * 2 * 1 = 24
3! = 3 * 2 * 1 = 6
2! = 2 * 1 = 2
1! = 1
0! = 1

Рекурсия применятся при обработке так называемых рекуррентных формул. Одной из таких формул является,, формула вычисления факториала числа: , где $n! = (n-1)! \cdot n$ $0! = 1$.

Чтобы вычислить факториал на шаге n, надо воспользоваться факториалом, вычисленным на шаге n-1.

Задача о вычислении факториала

Однако мы можем выразить факториал рекурсивно, через другие факториалы.

Чтобы вычислить факториал на шаге n, надо воспользоваться факториалом, вычисленным на шаге n-1.

5! = 5 * 4!

4! = 4 * 3!

3! = 3 * 2!

2! = 2 * 1!

1! = 1 * 0!

0! = 1

Пример использования рекурсии

Первый вариант:

```
int factorial (int i)
{
if (i==0)
    return 1;
else
  {
       i=i*factorial(i-1);
       return i;
  }
}
```

Второй вариант:

```
int factorial(int n)
{
      return !n ? 1 : n * factorial(n - 1);
}
```

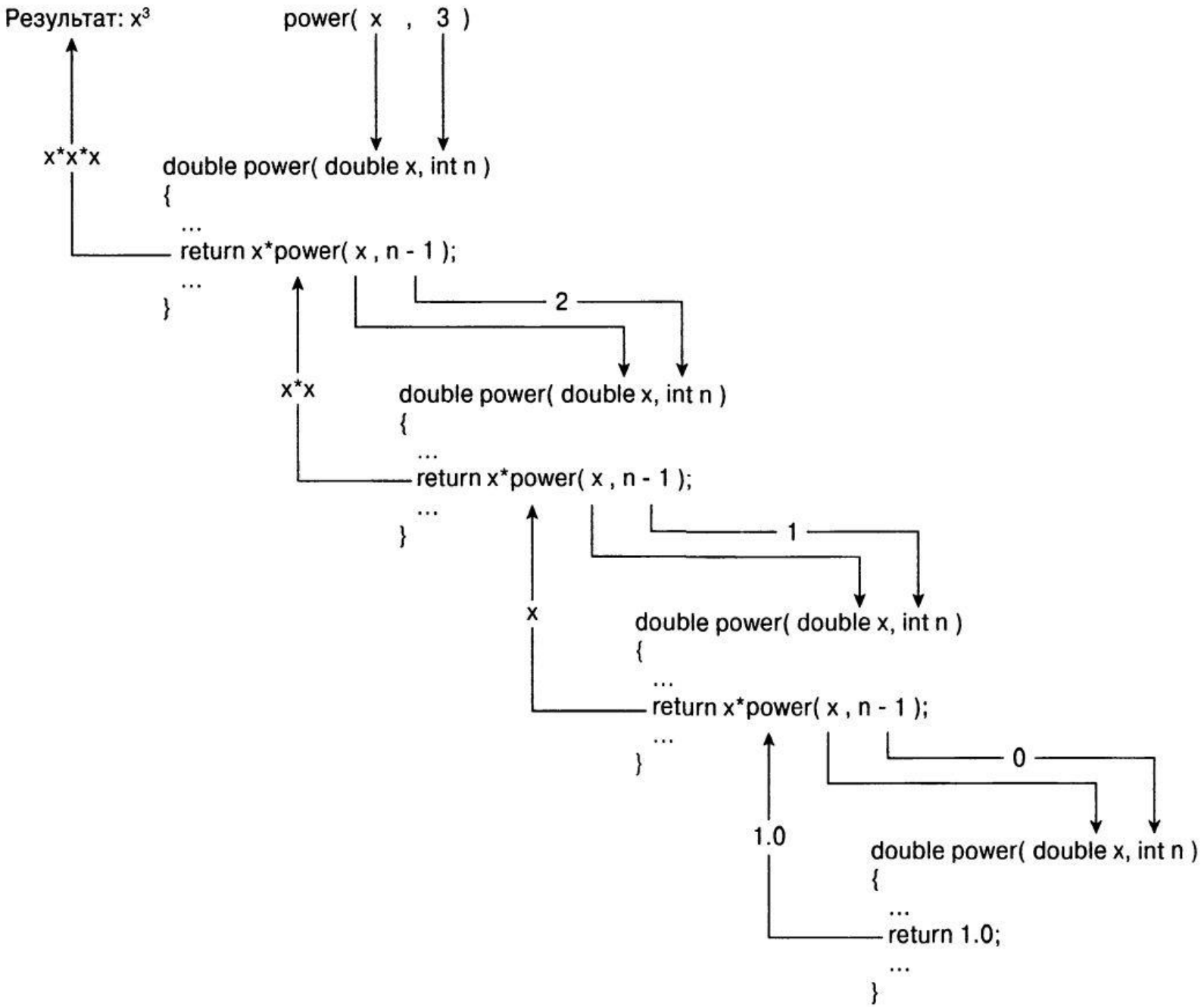
Пример использования рекурсии

Пример 2: Написать программу возведения значения в целочисленную степень - X^n. Это эквивалентно x, умноженному на себя n раз. Функция работает с отрицательными степенями, т.е. X^{-n} эквивалентно $1/x^n$

```
double power (double x, int n);
int main (void)
{
    double x=2.0;
    double result=0.0;
    for (int i=-3; i<=3; i++)
    {   result=power(x,i);
        printf("%lf в степени %d=%lf",x,i,result);
    }
}
double power (double x, int n)
{   if (n<0)
        {   x=1.0/x;
            n=-n;
        }
    if (n>0)  return x*power (x,n-1);
    else   retutn 1.0;
}
```

Вывод этой программы будет таким:

```
2 в степени -3 равно 0.125
2 в степени -2 равно 0.25
2 в степени -1 равно 0.5
2 в степени 0 равно 1
2 в степени 1 равно 2
2 в степени 2 равно 4
2 в степени 3 равно 8
```


Результат: x3
power(x , 3)
x*x*x
double power(double x, int n)
{
...
return x*power(x , n - 1);
...
}
2
x*x
double power(double x, int n)
{
...
return x*power(x , n - 1);
...
}
1
x
double power(double x, int n)
{
...
return x*power(x , n - 1);
...
}
0
1.0
double power(double x, int n)
{
...
return 1.0;
...
}

Пример использования рекурсии

Пример 3: В следующей программе происходит вывод на экран целых чисел 10, 9, …, 1 с использованием рекурсивной функции Print(). Данная функция в качестве аргумента получает число, которое необходимо вывести на экран. После того как число напечатано, данная функция вызывает саму себя с аргументом, на единицу меньшим только что напечатанного. Функции Print() завершается в случае, когда она в качестве параметра получает число, меньшее единицы.

```
#include <stdio.h>
void Print(int);
void Print(int i)
{
if (i >= 1)
{ printf("%d\n", i);
  Print(i - 1);
}
}
int main( )
{
Print(10);
return 0;
}
```

Преимущества и недостатки рекурсии

Использование рекурсии может сократить размер исходного кода программы и сделать код более элегантным и понятным. Также некоторые динамические информационные структуры легче реализуются с помощью рекурсии.

Однако рекурсия имеет и свои недостатки:

- алгоритмы, использующие рекурсию весьма требовательны к памяти.
- вернемся к примеру с рекурсивным вычислением факториала. При каждом вызове компилятор генерирует копии аргументов функции и отслеживает место в памяти, куда нужно вернуть значение. При каждом возвращении return. Однако чем большее число мы будем вычислять, тем больше нам потребуется памяти.

Таким образом, всегда полезно подумать о замене рекурсии на циклические алгоритмы.

Однако в некоторых случаях решение задачи без рекурсии может быть чрезвычайно сложным и прирост производительности не будет стоить потраченных усилий.