acm.mipt.ru

олимпиады по программированию на Физтехе

Степень числа

Для вычисления функции $f(n) = a^n$ есть рекуррентная формула:

$$a^n = a \cdot a^{n-1}, \quad a^0 = 1.$$

Вот программа, которая основана на этой формуле:

```
/*
  Степень числа: простая рекурсия
*/
#include<stdio.h>
double power(double x, long n)
{
   if(n == 0) return 1;
   if(n < 0) return power ( 1 / x, -n);
   return x * power(x, n - 1);
}
void main()
{
    double x;
    long n;
    while (scanf ("%lf %ld", &x, &n) == 2)
       printf("%lf\n", power (x, n));
}
```

Но есть более ``умная'' рекурсивная функция: %$ a^n = \left\{ \begin{array}{l} a\cdot a^{n-1}, \mbox{ если n нечетная,}\\ a^{n/2})^2, \mbox{ если n четная.} \end{array}\right. $%

Например, если обозначть стрелочкой %TO% слово ``сводится к '', то при вычислении a^{12} для первой рекурсии получим цепочку длины 12:

$$a^{12} \to a^{11} \to a^{10} \to a^9 \to a^8 \to a^7 \to a^6 \to a^5 \to a^4 \to a^3 \to a^2 \to a^1 \to a^0.$$

А для второй рекурсии цепочку из 5 шагов: $a^{12} \to a^6 \to a^3 \to a^2 \to a^1 \to a^0.$

Для больших n разница в длине цепочки более разительная. В частности a^{10000} первой рекурсией вычисляется за 10000 шагов, а второй – за 19 шагов.

```
/*
 Программа 2: степень числа – оптимизированная рекурсия.
*/
double power(double x, long n)
{
    double tmp;
   if(n == 0) return 1;
   if(n < 0) return power ( 1 / x, -n);
   if(n % 2) return x * power (x, n - 1);
   return power(x * x, n / 2);
}
```

```
/*
   Программа 3: степень числа – оптимальный алгоритм без рекурсии.
*/
double power(double x, long n)
{
```

Поиск
Поиск
Раздел «Язык Си»
Главная
Зачем учить С?
Определения
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

```
    double a = 1;
    while(n){
        if(n % 2) a *= x;
        x *= x;
        n /= 2;
    }
    return a;
}
```

- Сколько шагов требуется для вычисления a^{30} вторым методом?
- Покажите, что второй алгоритм выполняется за логарифмическое по n число шагов, а точнее ограничено сверху $2 \cdot \log_2 n$ (еще точнее: в точности равно числу знаков в двоичной записи числа n плюс число единичек в этой записи).
- Объясните, как работает программа 3.

-- ArtemVoroztsov - 08 Sep 2004

(c) Материалы раздела "Язык Си" публикуются под лицензией GNU Free Documentation License.