

# Основы программирования

## Рекуррентные вычисления

# Рекуррентная последовательность

Числовая последовательность  $\{x_k\}$  называется **рекуррентной ранга  $p$** , если

$$\begin{cases} x_k = a_k, & k = 0, 1, \dots, p-1, \\ x_k = f(k, x_{k-1}, x_{k-2}, \dots, x_{k-p}), & k = p, p+1, \dots \end{cases}$$

где  $a_0, a_1, \dots, a_{p-1}$  – константы, а  $f$  – функция

# Пример рекуррентной последовательности

Функция  $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$  (факториал)  
представляется рекуррентным  
соотношением ранга 1:

$$\begin{cases} F_0 = 1, \\ F_i = F_{i-1} * i, \quad i = 1, 2, \dots, n. \end{cases}$$

```
int n, F; cin >> n;  
F = 1;  
for (int i = 1; i <= n; i++)  
    F = F * i;
```

# Примеры тестов

Тесты по методу **черного** ящика:

- минимально возможное  $n=0$
- на 1 больше минимально возможного,  $n=1$
- другое значение  $n$ , например,  $n=6$

Тесты по методу **белого** ящика:

- такое  $n$ , чтобы цикл ни разу не выполнялся,  $n=0$
- такое  $n$ , чтобы цикл выполнялся 1 раз,  $n=1$
- несколько большее значение  $n$ , например,  $n=6$

Т.е. все тесты:  $n=0$ ,  $n=1$ ,  $n=6$

На выходе, соответственно:  $F=1$ ,  $F=1$ ,  
 $F=720$

# Трудоёмкость алгоритма

**Элементарный шаг** – это действие, время выполнения которого не зависит от числа входных переменных и их значений, например:

- один или фиксированное число выполняемых подряд операторов присваивания
- проверка условия в условном операторе или цикле
- вызов функции и возврат из функции

# Трудоёмкость алгоритма

**Трудоёмкость** – это функция зависимости количества элементарных действий от входного параметра  $n$  при  $n \rightarrow \infty$ .

**Например**, для цикла, который выполняется  $n$  раз, трудоёмкость  $T(n) = A \cdot n + B$ . Здесь коэффициент  $B$  – это число шагов до первой проверки условия, а  $A$  – число элементарных шагов при однократном выполнении цикла.

Коэффициенты зависят от компьютера и качества трансляции, поэтому на практике важно не точное значение, а **порядок роста**  $T(n)$  при  $n \rightarrow \infty$ . В нашем случае  **$T(n)$  растёт линейно относительно  $n$** , и это обозначается в виде:  **$T(n) = O(n)$**

# Числа Фибоначчи

Числа Фибоначчи задаются рекуррентной последовательностью ранга 2:

$$\begin{cases} f_0 = 0, f_1 = 1, \\ f_k = f_{k-1} + f_{k-2}, \quad k = 2, 3, \dots, \end{cases}$$

```
int n, f0, f1, f2, k; cin >> n;
f0 = 0; f1 = 1;
for (k = 2; k <= n; k++)
{
    f2 = f0 + f1; f0 = f1; f1 = f2;
}
```

# Числа Фибоначчи

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Значения быстро возрастают:

$$f_{20} = 6765, f_{30} = 832040, f_{40} = 102334155$$

Максимальное число Фибоначчи, представимое типом **int**:  $f_{47}$

Существует **предел отношения** двух последовательных чисел Фибоначчи:

число  $\phi = \lim_{k \rightarrow \infty} (f_{k+1} / f_k) = (1 + \sqrt{5}) / 2 \approx 1,618$

называют **ЗОЛОТЫМ СЕЧЕНИЕМ**



# Приближенное вычисление предела последовательности

Последовательность может иметь **предел** при  $n \rightarrow \infty$ . Например, некоторый конечный предел может иметь последовательность сумм

$$\begin{cases} S_1 = f_1, \\ S_k = S_{k-1} + f_k, \quad k = 2, 3, \dots \end{cases}$$

где слагаемые  $f_k$  стремятся к нулю при  $k \rightarrow \infty$ .

$$\begin{cases} f_1 = a, \\ f_k = p(k, f_{k-1}), \quad k = 2, 3, \dots \end{cases}$$

# Приближенное вычисление предела последовательности

```
double eps, a, f, S;  
int k;  
cin >> a >> eps;  
k = 1; S = f = a;  
while (abs(f) >= eps)  
{  
    k++;  
    f = p(k, f);  
    S += f;  
}
```

# Приближенное значение функции $\sin x$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!}$$

**Рекуррентное соотношение для элементов суммы:**

$$\begin{cases} f_1 = x, \\ f_k = -f_{k-1} \frac{x^2}{(2k-1)(2k-2)}, \quad k = 2, 3, \dots \end{cases}$$

# Алгоритм вычисления $\sin x$

```
double eps, x, f, S;  
int k; cin >> x >> eps;  
k = 1; S = f = x;  
while (abs(f) >= eps)  
{  
    k++;  
    f = -f*x*x / (2*k-1) / (2*k-2);  
    S += f;  
}
```

Количество  $k$  выполнений цикла:  
если  $|x| \leq 1$ ,  $\varepsilon \leq 1/2$ , то  $k \leq \log_2 1/\varepsilon$

# Рекуррентная последовательность Герона

$$\begin{cases} s_0 = 1, \\ s_k = \frac{1}{2} \left( s_{k-1} + \frac{a}{s_{k-1}} \right), \quad k = 1, 2, \dots \end{cases}$$

Можно доказать, что  $\lim_{k \rightarrow \infty} s_k = \sqrt{a}$  при  $a \geq 0$ .

# Вычисление квадратного корня по формуле Герона

```
double a, eps, s, s1, e;
cin >> a >> eps;
s = (1 + a) / 2;
for (e = eps; e >= eps; )
{
    s1 = (s + a / s) / 2;
    e = abs(s - s1);
    s = s1;
}
```

**Трудоемкость:**  $O(\log((1+a)/eps))$