Рекурсия

CENTRUL DE EXCELENȚĂ ÎN INFORMATICĂ ȘI TEHNOLOGII INFORMAȚIONALE

Шарапановская И

Пример рекурсии

Если у вас жирное пятно на платье, не переживайте. Пятна от масла убираются бензином. Пятно от бензина раствором щёлочи. Щелочь убирается эссенцией. След от эссенции потрите маслом. Ну, а как убрать пятна от масла, вы уже знаете!

Определение

- **Рекурсия** (от латинского *recursio* - возвращение) - это такой способ организации вычислительного процесса, при котором процедура или функция в ходе выполнения составляющих ее операторов обращается сама к себе.

- Другими словами Рекурсия - подпрограмма, которая вызывает саму себя.

Пример вычисление факториала.

- N! = 1*2*3* . . . *(N-2)*(N-1)*N
- 1! = 1
- 0! = 1

- 5! = 1*2*3*4*5 =120

Итеративная функция

- Сначала покажем обычную не рекурсивную функцию для вычисления факториала, которая реализует итеративный алгоритм вычисления

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

int  fact(int n)
{
    int p=1;
    for(int i=1; i<=n; i++)
        p*=i;

    return p;
}

int main()
{
    int N;
    setlocale(0,""); // Включаем кириллицу
    cout << "Введите число для вычисления факториала: ";
    cin >> N;
    cout << "Факториал для числа " << N << " = " << fact(N) << endl << endl;
    return 0;
}
```

```
Введите число для вычисления факториала: 5
Факториал для числа 5 = 120
```

- Вторая функция использует рекурсивные обращения, что делает ее гораздо компактнее, и основана на очевидном соотношении:
- N! = (N-1)!*N
- Иными словами, чтобы получить значение факториала от числа N, достаточно умножить на N значение факториала от предыдущего числа:

- 5! = 4!*5

Рекурсивная функция

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

int fact(int n)
{

    if (n == 0) // если пользователь ввел ноль,
        return 1; // возвращаем факториал от нуля - не удивляетесь, но это 1 =)
    else // Во всех остальных случаях
        return fact(n - 1)*n; // делаем рекурсию.
}

int main()
{
    int N;
    setlocale(0,""); // Включаем кириллицу
    cout << "Введите число для вычисления факториала: ";
    cin >> N;
    cout << "Факториал для числа " << N << " = " << fact(N) << endl << endl; // fact(N) - функц
    return 0;
}
```

```
Введите число для вычисления факториала: 5
Факториал для числа 5 = 120
```

Пример вычислений

- fact(5)=fact(4)*5
- fact(4)=fact(3)*4
- fact(3)=fact(2)*3
- fact(2)=fact(1)*2
- fact(1)=fact(0)*1
- fact(0)=1

- 24*5=120
- 6*4=24
- 2*3=6
- 1*2=2
- 1*1=1
- 1

- Таким образом можем сделать вывод, что рекурсию можем заменить на цикл (итерацию) и наоборот.

- Естественно, напрашивается вопрос – не получим ли мы бесконечный процесс рекурсивного вызова подпрограммы? Когда процесс «самовызова» остановится? **Ответ** – как только будет достигнуто условие, когда мы знаем ответ, не прибегая к рекурсии. Такое условие называется **граничным**.

- **Граничное условие** – условие, при котором решение может быть получено нерекурсивно. Условие, при котором решение выражается с помощью более узкого варианта самого себя называется **рекурсивным** или **общим условием.**

рекурсивный алгоритм

граничное условие

общее условие

Бесконечная рекурсия

- Если граничное условие отсутствует, то получим бесконечную рекурсию – эквивалент бесконечного цикла.
- Бесконечная рекурсия – ситуация при которой подпрограмма бесконечно вызывает саму себя.
- На самом деле, каждый раз, когда подпрограмма вызывает саму себя, используется дополнительная память для хранения новых копий переменных. В конце концов, свободной памяти не останется и программа даст сбой.

Косвенная рекурсия

- Рекурсия может быть как **прямой**, когда программа вызывает саму себя, так и непрямой (**косвенной**) , когда программа вызывает другую программу, а та в свою очередь, вызывает первую программу.
- При объявлении функций при непрямой рекурсии прототип второй функции описывается до описания первой.

Косвенная рекурсия

Образно косвенную рекурсию можно описать так. *Перед зеркалом 1 стоит зеркало 2, в котором отражается само зеркало 1. В последнем видно зеркало 2 и т.д.*

$$f(x) = \begin{cases} 1, & \text{если } x = 0 \\ g(x-1) + x, & \text{если } x > 1 \end{cases}$$

$$g(x) = \begin{cases} x, & \text{если } x = 0 \\ f(x-1) * x, & \text{если } x > 1 \end{cases}$$

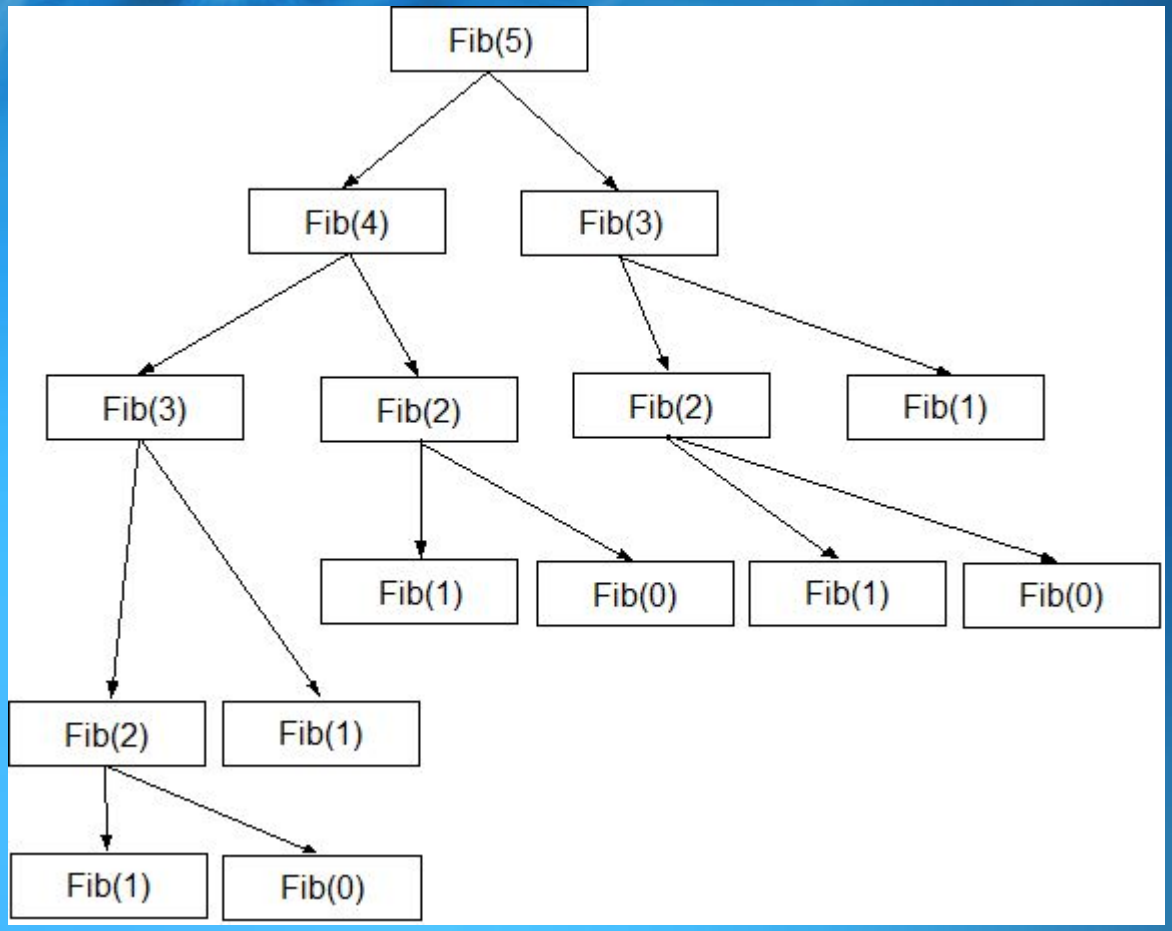
```
#include <iostream>
using namespace std;
int g(int x);
int f(int x)
{
    if (x == 0)   return 1;
    else return g(x - 1)+x;
}

int g(int x)
{
    if (x == 0)   return x;
    else return f(x - 1)*x;
}

int main()
{
    int N;
    setlocale(0,"");
    cout << "Введите число  ";
    cin >> N;
    cout << "Ответ " << f(N) << endl;
}
```

```
Введите число  5
Ответ 25
```

Числа Фибоначчи



- F(0)=0
- F(1)=1,
- F(N)=F(N-1)+F(N-2) при n>1.
- 0 1 1 2 3 5 8 13 21 34 55

Числа Фибоначчи

ИТЕРАЦИЯ

```
int fib(int n)
{
    int a=0;
    int b=1;
    int c;
    if(n==0) return 0;
    if(n==1) return 1;
    for(int i=2; i<=n; i++)
    {
        c=a+b;
        a=b;
        b=c;
    }
    return c;
}
```

Числа Фибоначчи

```
int fib(int n)
{
    if(n<=1) return n;
    return fib(n-1)+fib(n-2);
}
```

- Fib(5) = Fib(4)+Fib(3)
- Fib(4) = Fib(3)+Fib(2)
- Fib(3) = Fib(2)+Fib(1)
- Fib(2) = Fib(1)+Fib(0)
- Fib(1) = 1
- Fib(0) = 0

Степень числа

- $a^b = a*a*\ldots*a$ - b *раз*

или

- $a^b = a^{b-1}*a$

```
int power(int a, int b)
{
    if (b == 0)   return 1;
    else return power(a, b - 1)*a;
}
```

ВСЁ!