

Рекурсивные функции

Дисциплина: Конструирование программ и языки программирования

*«Чтобы понять рекурсию, нужно
сначала понять рекурсию».*

Рекурсия — это такая организация выполнения работы функции, при которой данная функция вызывает сама себя.

Функции, которые во время выполнения вызывают сами себя называют **рекурсивными**.

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

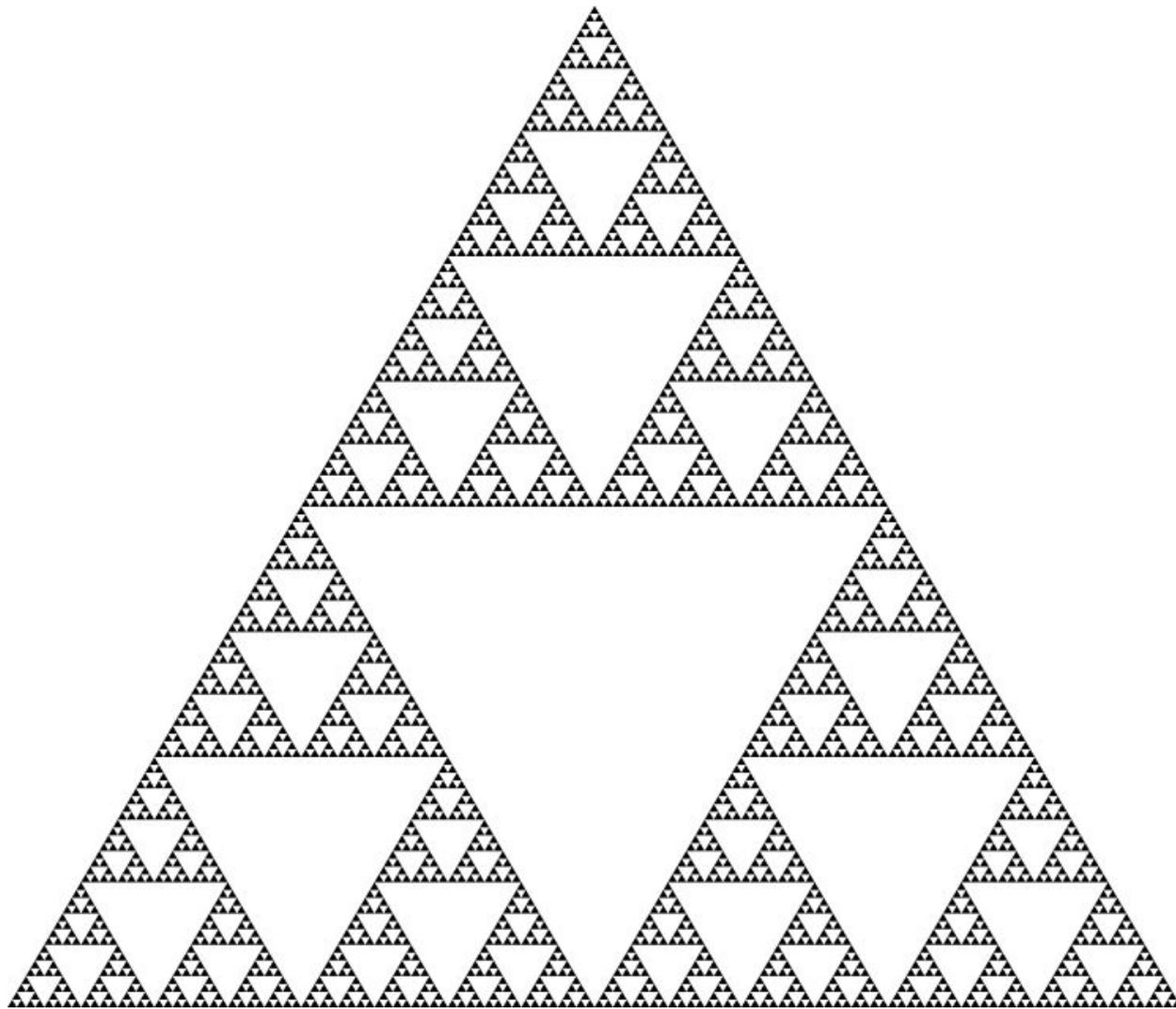
screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// - VLC media player

Media Playback Audio Video Tools View Help

screen:// 1.00x 33:48:20/---



Треугольник Серпинского





Герб Российской Федерации



рекурсия

Поиск

Картинки

Видео

Новости

Книги

Результатов: примерно 276 000

Все страны

Страна: Беларусь

Возможно, вы имели в виду: [рекурсия](#)

Рекурсия

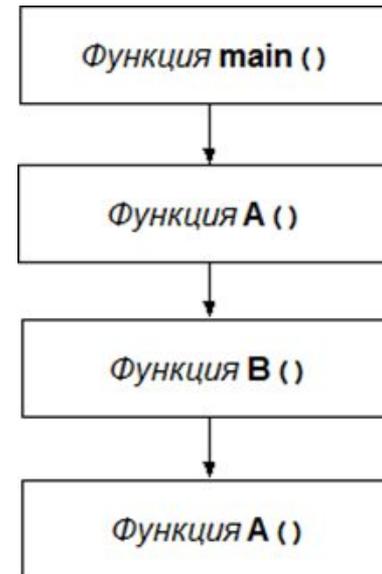
Прямая

Если функция содержит в своем теле вызов самой себя.

```
void f() {  
    .....  
    f();  
    .....  
}
```

Косвенная

Если первая функция вызывает вторую функцию, но при этом в теле второй функции прописан вызов первой.



Вывести на экран сумму чисел от 1 до N:

```
#include <iostream.h>
#include <stdlib.h>

int sum(int n)
{
    if (n==1)
return 1;
else
return sum(n-1)+n;
}

int main()
{
system("cls");
cout<<sum(5);
return 0;
}
```

Рекурсия функции `main()` на языке C++:

Прямая

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!"
<< endl;
    main();
    return 0;
}
```

Косвенная

```
#include <iostream>
using namespace std;
void f();
int main()
{
    cout << "Hello world!" <<
endl;
    f();
    return 0;
}
void f()
{
    main();
}
```

Вычислить факториал $n!$.

($n!$ — это произведение первых n натуральных чисел)

```
#include <iostream>
using namespace std;
int factorial(int n);
int main()
{
    int n = 5;
    int y = factorial(n);
    cout << n<<"! =" << y << endl;
system ("pause");
return 0;
}
int factorial(int n)
{
    int t;
    if(n <=1)
        t = 1;
    else
        t = n * factorial(n - 1);
system ("pause");
return t;
}
```

$$4! = 4 \cdot 3!$$

$$3! = 3 \cdot 2!$$

$$2! = 2 \cdot 1!$$

$$1! = 1$$

Для данного n вычислить число Фибоначчи.

(Каждое последующее число Фибоначчи представляет собой сумму двух предыдущих) - $F(0)=F(1)=1$, $F(N)=F(N-1)+F(N-2)$ при $n>1$.

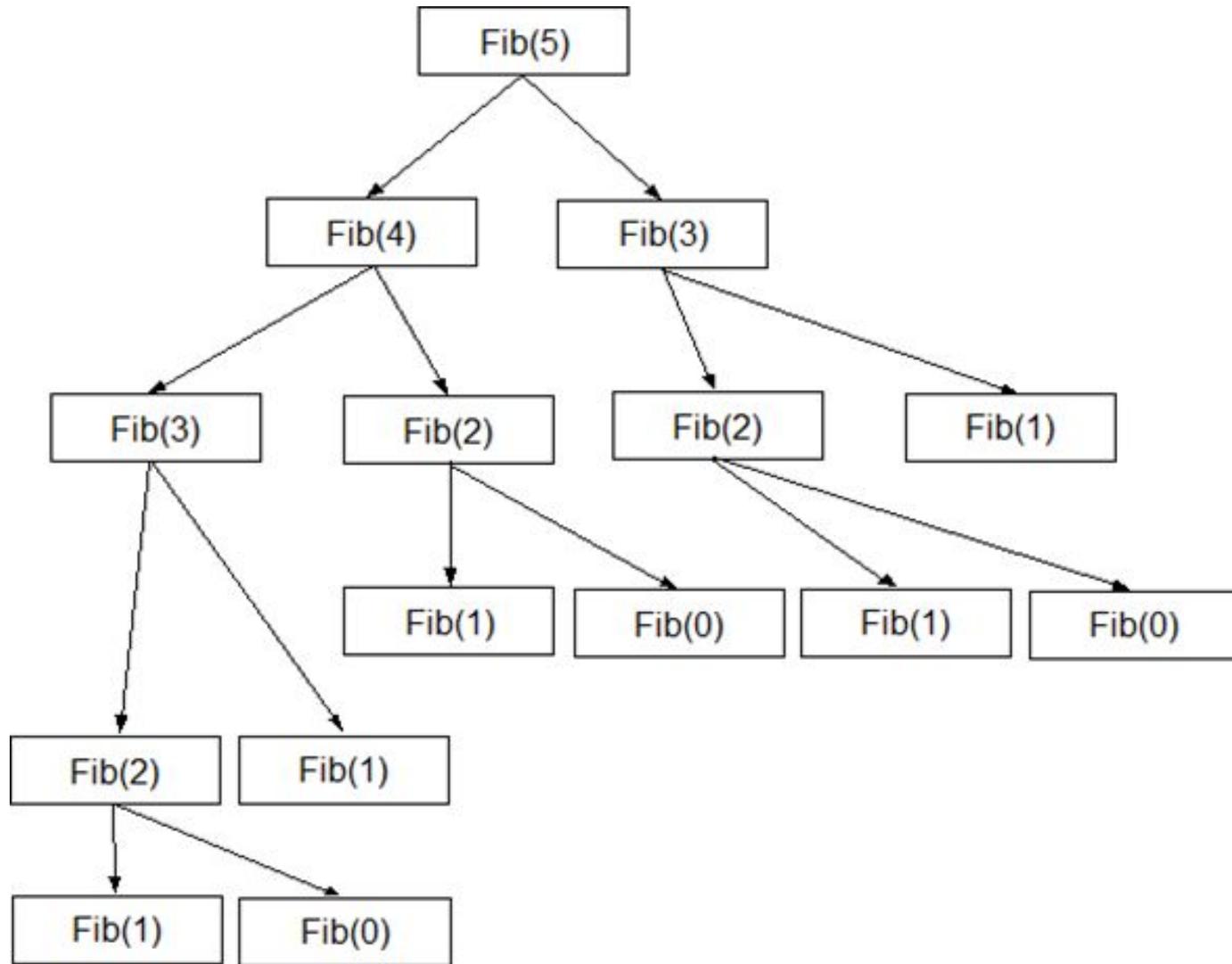
```
#include <iostream>
using namespace std;

int Fib(int n)

{
    if (n==1|| n==2) {
        return 1;}
else {
    return Fib(n-1)+Fib(n-2) ;}

}
int main()
{

cout<<Fib(10) ;
return 0 ;
}
```



Рекурсивный вызов функции Fib (Фибоначчи)

Перевод натурального числа из десятичной системы счисления в двоичную.

```
#include <iostream>
using namespace std;
void DecToBin( int n ) {
    if ( n >= 2 ) {
        DecToBin( n/2 );
    }
    cout << n % 2;
    return;
}

int main () {
    int n;
    cout << "n = ";
    cin >> n;
    cout << n << " (Dec) = ";
    DecToBin( n );
    cout << " (Bin)" << endl;
    return 0;
}
```

n = 65 (Dec) = 100001 (Bin)

