

# *Algorithms & Programming* *Programming Basics*

C/C++/Kotlin programming  
(p.3 – Functions - Recursion)



**C/C++**



Yevhen Berkunskyi, NUoS  
eugeny.berkunsky@gmail.com  
<http://www.berkut.mk.ua>

# Recursion

What is recursion ?

Let's use Google?..



# Recursion

## Recursion is a ... ?

recursion - Google Search

google.com/search?q=recursion&loq=re&aq=chrome.0.69i59j69i57j69i60j69i61j69i60j69i65j69i60l2.1093j0j1...

Історія Миколаївщ... м.Миколаїв - I ♥ M... draw.io e-olymp/solutions SQL Олімпіади по ин... databases Emoji for apps Другие закладки

Google recursion

All Images News Books Videos More Tools

About 165,000,000 results (0.45 seconds)

Did you mean: **recursion**

<https://en.wikipedia.org/wiki/Recursion>

**Recursion - Wikipedia**

Recursion (adjective: **recursive**) occurs when a thing is defined in terms of itself or of its type. Recursion is used in a variety of disciplines ranging ...

Recursion (computer science) · Recursion (disambiguation) · Category:Recursion

People also ask

- What is recursion with example?
- What do u mean by recursion?
- What is recursion in C ++?
- What is recursive thinking?

**Recursive Functions**

```
int recursion ( x )  
{  
    if ( x==0 )  
        return;  
    recursion ( x-1 );  
}
```

**Recursion in C**

```
1 )  
int fun ( )  
{  
    return 1;  
}  
return 1 + fun( n-1 );  
  
( ) {  
    n = 3;  
    if ("end", fun(n));  
}
```

**C Programming**

**Recursion**

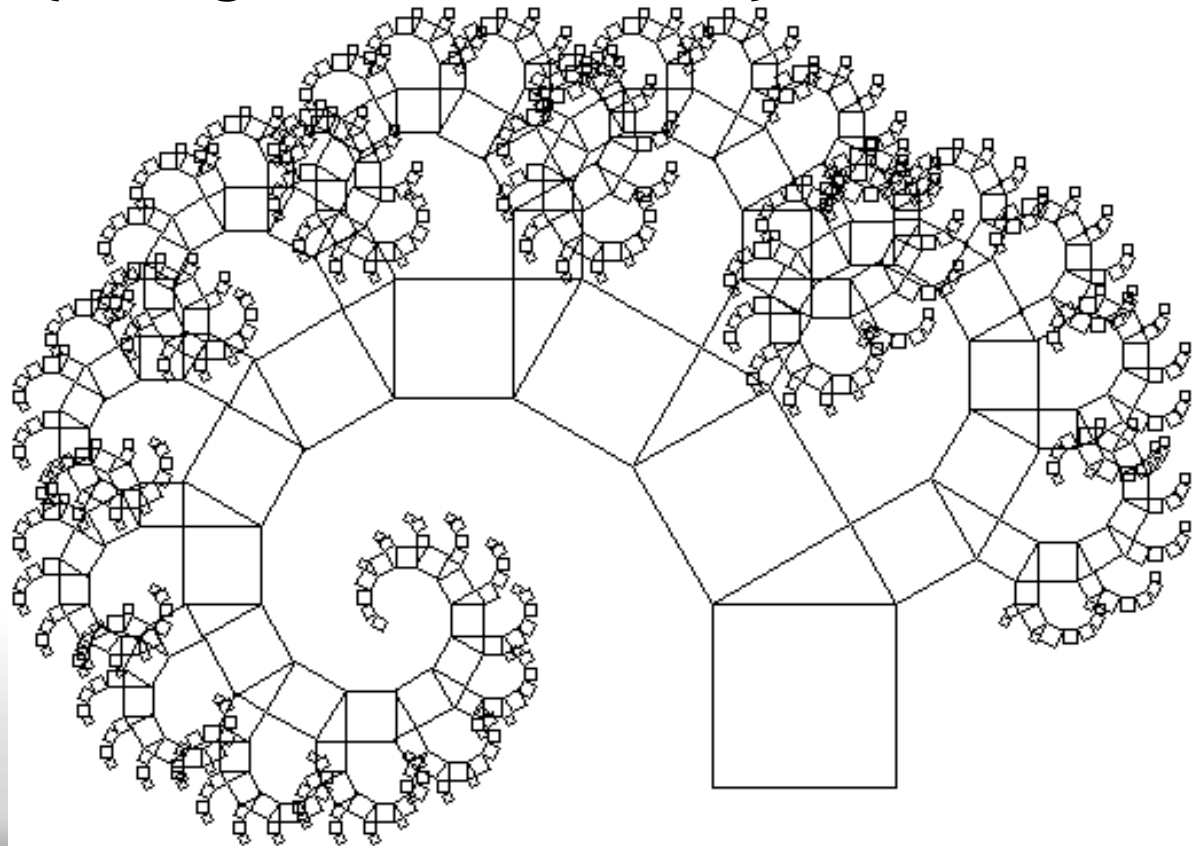
Computer science

Iterative Approach

More images

# Recursion

**Recursion** is the definition of a part of a function through itself, that is, it is a function that calls itself, directly (in its body) or indirectly (through another function).



# Recursion

Typical recursive tasks are tasks:

- Calculating  $n!$
- Finding Fibonacci numbers.

Such tasks have already been solved by us, but only using cycles, that is, iteratively.

Generally speaking, everything that is solved iteratively can be solved recursively, that is, using a recursive function.



# Recursion

Calculating factorial:

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

Iterative (looping) process:

```
int f = 1;  
for (int i = 1; i <= n; i++) {  
    f *= i;  
}
```

# Recursion

Calculation of factorial

$$n! = \begin{cases} 1, & \text{if } n = 0 \text{ or } n = 1 \\ (n - 1)! * n, & \text{if } n > 1 \end{cases}$$

Recursive process:

```
int f(int n) {  
    if ( n == 0 || n == 1 ) return 1;  
    else return n * f(n-1);  
}
```



# Calculation of factorial





# Tail Recursion

Tail recursion is defined as a recursive function in which the recursive call is the last statement that is executed by the function. So basically nothing is left to execute after the recursion call.

$$f(n, a) = \begin{cases} a, & n = 0 \\ f(n - 1, n * a), & n > 0 \end{cases}$$

```
int Factorial(int n, int a = 1) {  
    // return condition  
    if (n==0)  
        return a;  
    // tail recursive call  
    return Factorial(n - 1, n * a);  
}
```

## Fibonacci numbers:

$$f_i = \begin{cases} 1, & \text{if } i = 0 \text{ or } i = 1 \\ f_{i-1} + f_{i-2}, & \text{if } i > 1 \end{cases}$$

# Fibonacci numbers

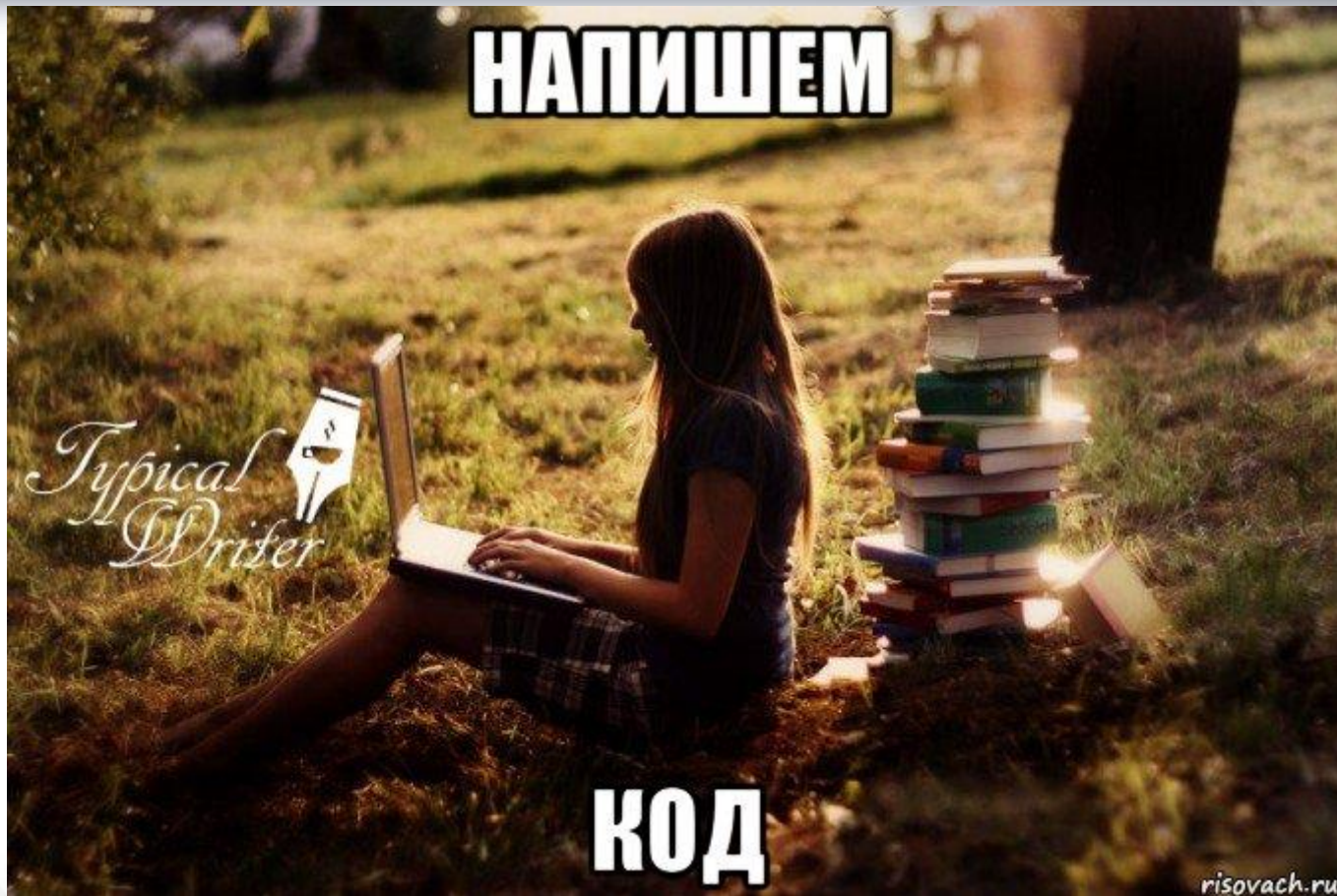
## Simple recursion:

```
int fib(int i) {  
    if (i == 0 || i == 1) return 1;  
    else return fib(i - 1) + fib(i - 2);  
}
```

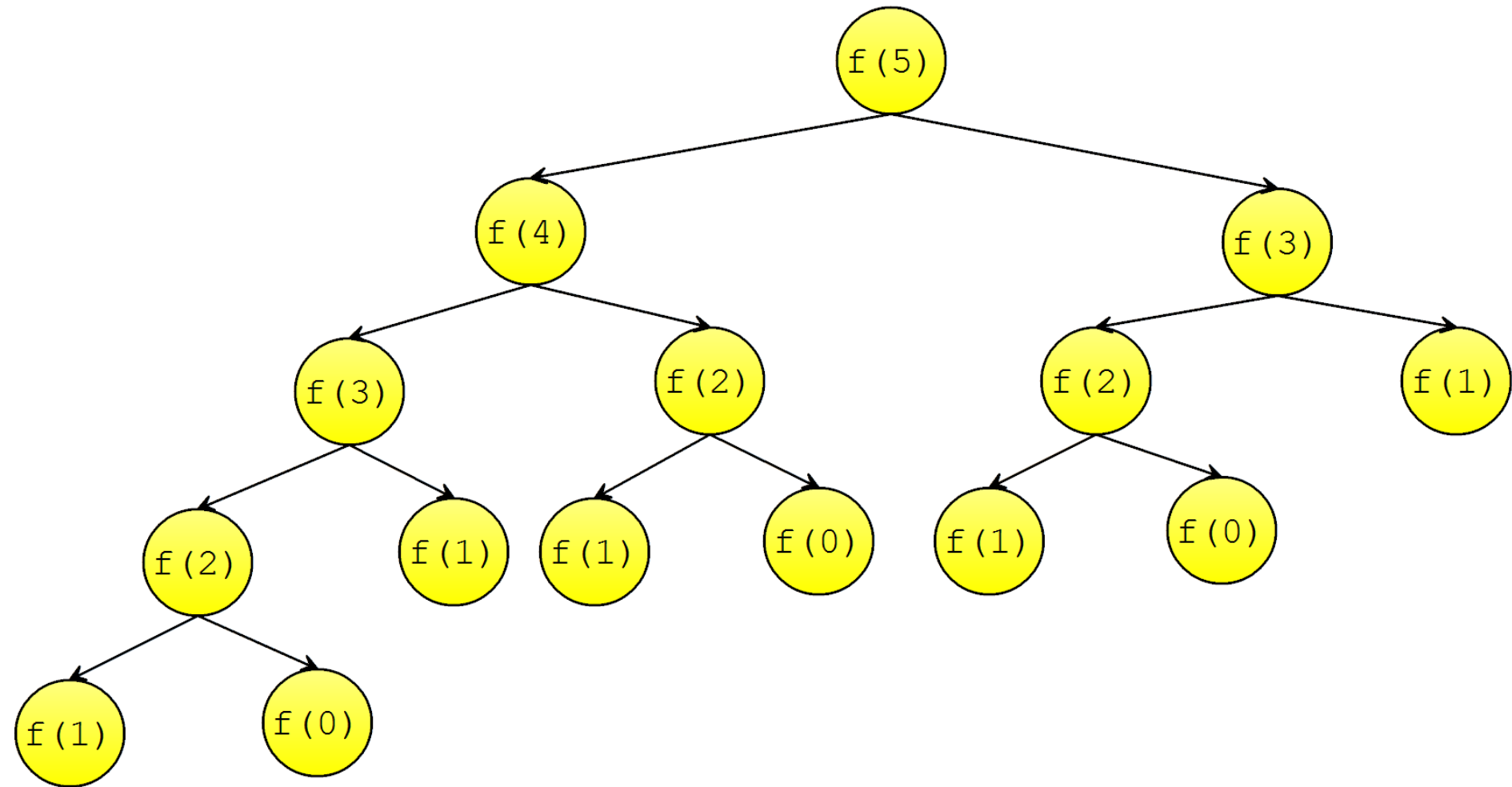
And what is wrong with it?



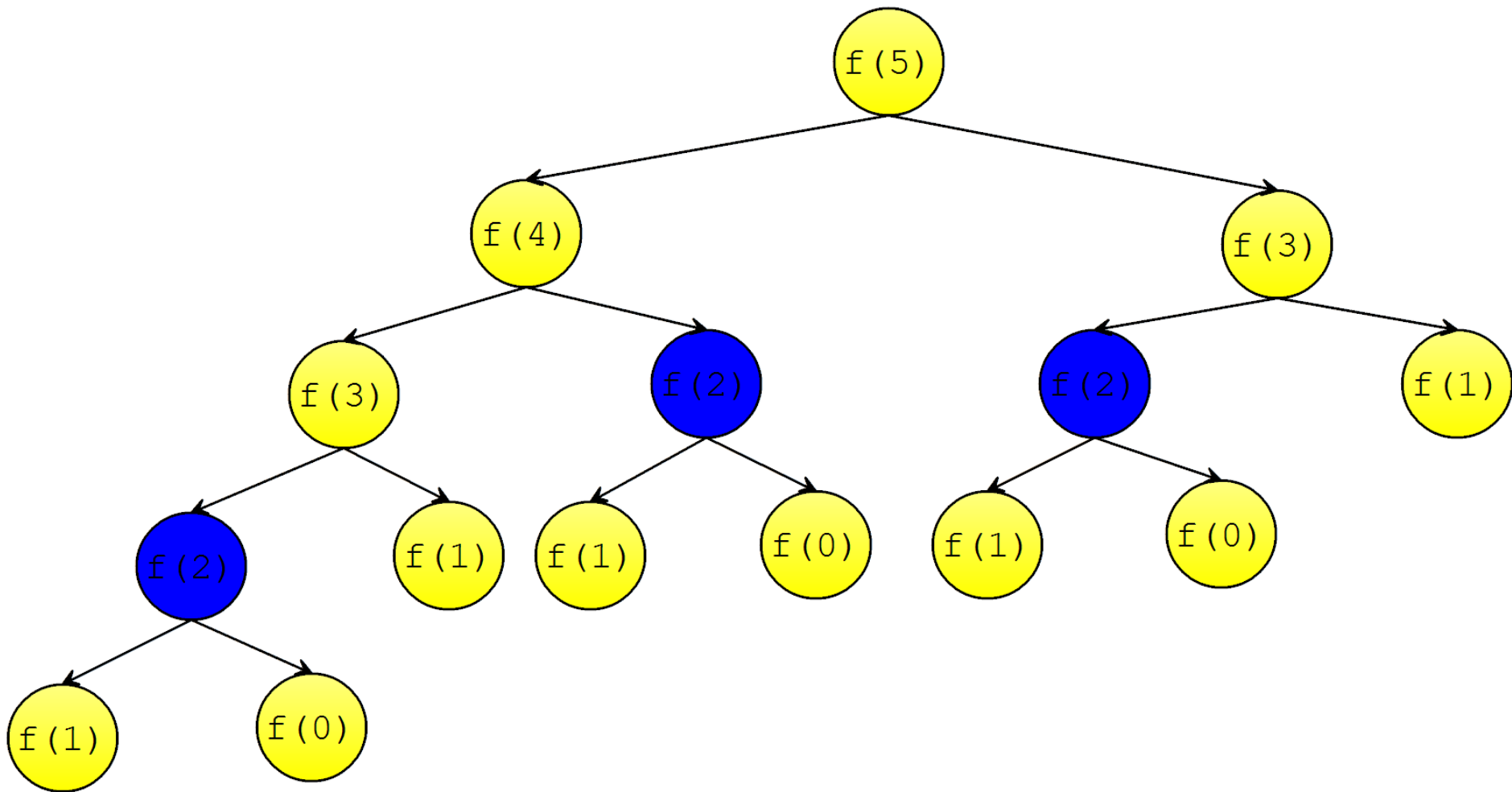
# Demo



# Recursion

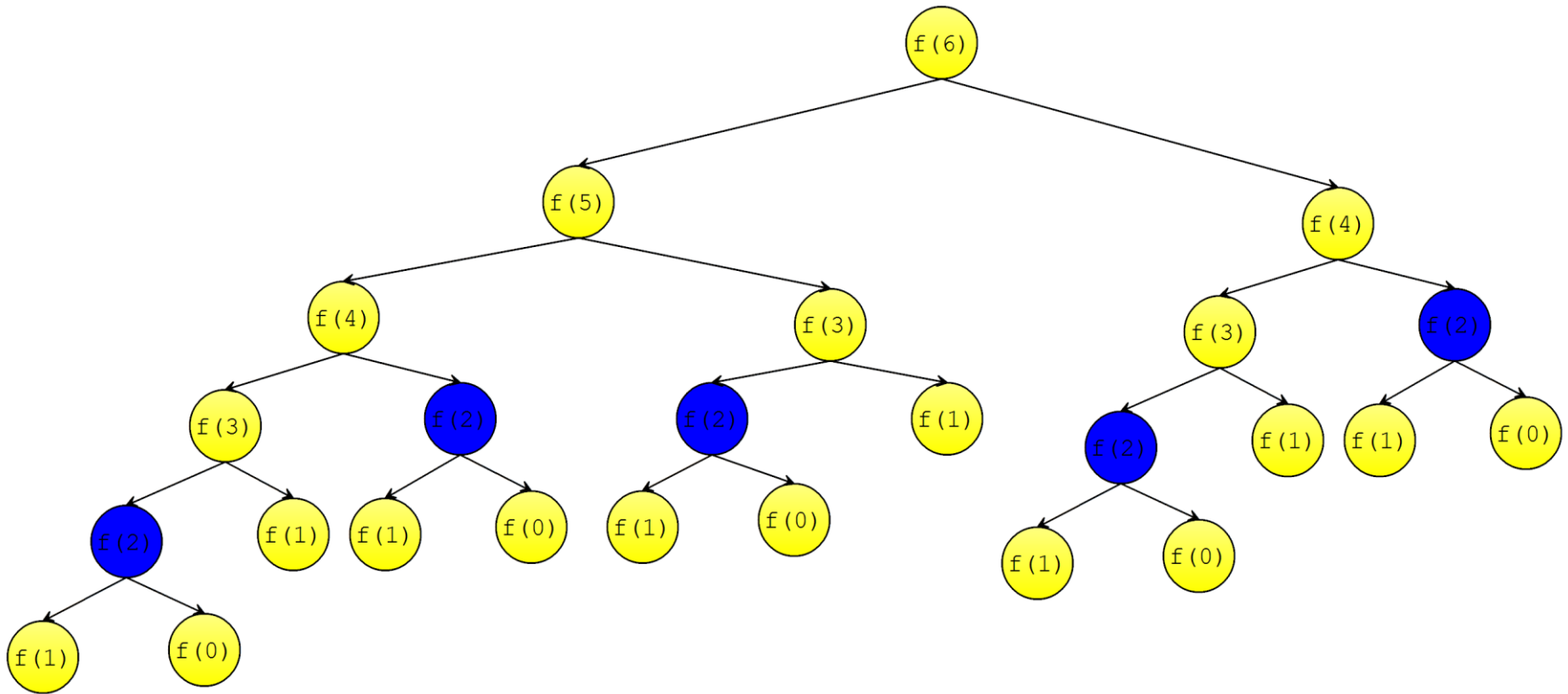


# Recursion





# Recursion



We have a problem. How can we solve it?

**ЕЩЕ**

A meme image featuring a large, bold, white Russian text "НЕМНОГО КОДА?" (A little code?) overlaid on a background of a JavaScript code snippet. The code is a long, repetitive string of escaped characters and symbols, likely a placeholder or a joke about obfuscation. The background is dark with light-colored text, and the overall aesthetic is that of a popular internet meme.

# НЕМНОГО КОДА?

risovach.ru

# Tail recursion

```
int Fib(int n, int a = 0, int b = 1)
{
    // return condition
    if (n == 1)
        return b;
    else
        // tail recursive call
        return Fib(n-1, b, a+b);
}
```

# Recursion

And one more difficult problem.

And more interesting 😊

«Tower of Hanoi».

The objective of the puzzle is to move the entire stack to the last rod, obeying the following rules:

- Only one disk may be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
- No disk may be placed on top of a disk that is smaller than it.

# Recursion «Tower of Hanoi»



# Tower of Hanoi

```
#include <iostream>

using namespace std;

void p(int n, int a, int b, int c) {
    if (n == 1) {
        cout << a << "->" << b << "\n";
    } else {
        p(n-1, a, c, b);
        p(1, a, b, c);
        p(n-1, c, b, a);
    }
}

int main() {
    int n;
    cin >> n;
    p(n, 1, 2, 3);
    return 0;
}
```



# Recursion



# Optimistic final 😊





НАЦІОНАЛЬНИЙ  
УНІВЕРСИТЕТ  
КОРАБЛЕБУДУВАННЯ  
ІМЕНІ АДМІРАЛА МАКАРОВА

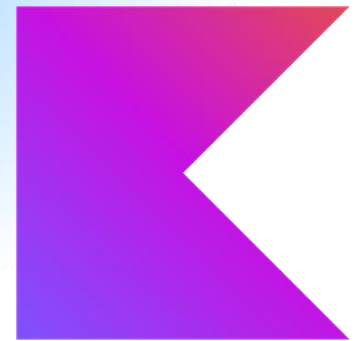


# *Algorithms & Programming* *Programming Basics*

C/C++/Kotlin programming  
(p.3 – Functions - Recursion)



**C/C++**



Yevhen Berkunskyi, NUoS  
eugeny.berkunsky@gmail.com  
<http://www.berkut.mk.ua>