

# Алгоритмизация и программирование

Программирование на C/C++  
(ч.3 – массивы, функции, рекурсия)



Беркунский Е.Ю., кафедра ИУСТ, НУК  
eugeny.berkunsky@gmail.com  
<http://www.berkut.mk.ua>

# Одномерные массивы в C++

- Одномерный массив — массив, с одним параметром, характеризующим количество элементов одномерного массива.
- Фактически одномерный массив — это массив, у которого может быть только одна строка, и n-е количество столбцов.
- Столбцы в одномерном массиве — это элементы массива.
- На рисунке показана структура целочисленного одномерного массива **a**.
- Размер этого массива — 16 ячеек.

5	-12	-12	9	10	0	-9	-12	-1	23	65	64	11	43	39	-15
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]	a[12]	a[13]	a[14]	a[15]

# Одномерные массивы в C++

5	-12	-12	9	10	0	-9	-12	-1	23	65	64	11	43	39	-15
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]	a[12]	a[13]	a[14]	a[15]

- Заметьте, что максимальный индекс одномерного массива **a** равен 15, но размер массива 16 ячеек, потому что нумерация ячеек массива всегда начинается с 0.
- Индекс ячейки – это целое неотрицательное число, по которому можно обращаться к каждой ячейке массива и выполнять какие-либо действия над ней (ячейкой).

```
//синтаксис объявления одномерного массива в C++:  
/*тип данных*/ /*имя массива*/[/*размерность*/];  
//пример объявления одномерного массива, изображенного на рисунке  
int a[16];
```

# Одномерные массивы в C++

```
//синтаксис объявления одномерного массива в C++:  
/*тип данных*/ /*имя массива*/[/*размерность*/];  
//пример объявления одномерного массива, изображенного на рисунке  
int a[16];
```

- Всегда сразу после имени массива идут квадратные скобки, в которых задаётся размер одномерного массива, этим массив и отличается от всех остальных переменных.

```
//ещё один способ объявления одномерных массивов  
int mas[10], a[16];
```

- Объявлены два одномерных массива **mas** и **a** размерами 10 и 16 соответственно.
- При таком способе объявления все массивы будут иметь одинаковый тип данных, в нашем случае - **int**.

# Одномерные массивы в C++

```
// массивы могут быть инициализированы при объявлении:  
int a[10] = { 5, -12, -12, 9, 10, 0, -9, -12, -1, 7};
```

- Инициализация одномерного массива выполняется в фигурных скобках после знака **равно**, каждый элемент массива отделяется от предыдущего запятой.

```
int a[] = { 5, -12, -12, 9, 10, 0, -9, -12, -1, 7};
```

- В данном случае компилятор сам определит размер одномерного массива.
- Размер массива можно не указывать только при его инициализации, при обычном объявлении массива обязательно нужно указывать размер массива.

Рассмотрим пример программы:

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

int main()
{
    cout << "obrabotka massiva" << endl;
    int array1[16] = { 5, -12, -12, 9, 10, 0, -9,
                     -12, -1, 23, 65, 64, 11, 43, 39, -15 };
    cout << "index" << "\t\t" << "element massiva" << endl;
    for (int counter = 0; counter < 16; counter++)
    {
        cout << "array1[" << counter << "]" << "\t\t"
             << array1[counter] << endl;
    }
    return 0;
}
```

# Функции в C++

- Опытные программисты знают, что для написания больших программ необходимо пользоваться функциями.
- Программа будет состоять из отдельных фрагментов кода, под отдельным фрагментом кода понимается функция.
- Отдельным, потому, что работа отдельной функции практически не зависит от работы какой-нибудь другой.
- Алгоритм в каждой функции должен быть функционально достаточен и максимально независим от других алгоритмов программы.

# Функции в C++

- Функция (в программировании) — это фрагмент кода или алгоритм, реализованный на каком-то языке программирования, с целью выполнения определённой последовательности операций.
- Однажды написав функцию, её можно будет с лёгкостью переносить в другие программы.

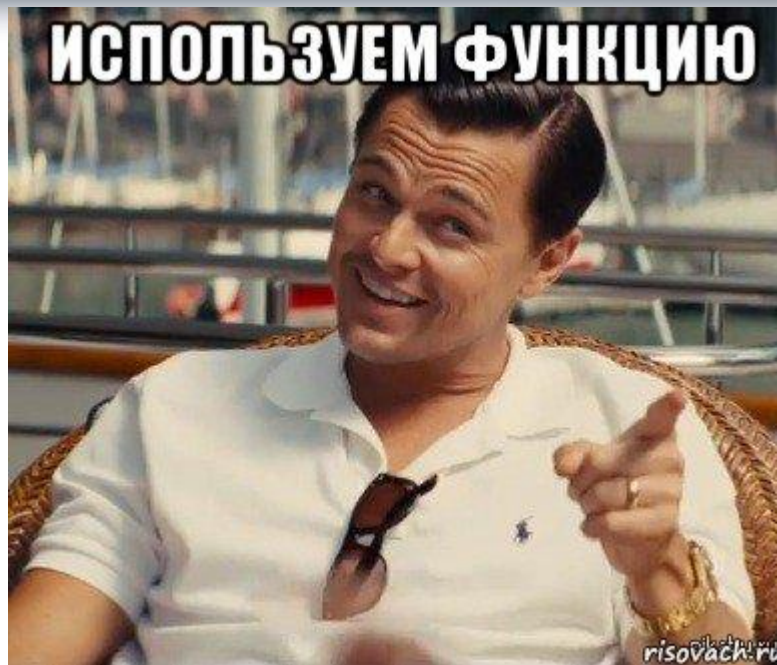




# Функции в C++

- Функции позволяют сделать программу модульной, то есть разделить программу на несколько маленьких подпрограмм (функций), которые в совокупности выполняют поставленную задачу.
- Еще одно преимущество функций в том, что их можно многократно использовать.
- Данная возможность позволяет многократно использовать один раз написанный код, что в свою очередь, намного сокращает объем кода программы!

# Функции в C++



- Чтобы воспользоваться функцией, определённой в заголовочном файле, нужно его подключить.
- Например, чтобы воспользоваться функцией, которая возводит некоторое число в степень, нужно подключить заголовочный файл `<cmath>` и использовать функцию `pow()` в программе.

# Пример программы

```
// 1 - подключаем заголовочный файл <cmath>
// который содержит прототипы основных
// математических функций
#include <cmath>

int main() {
    // 2 - запуск функции
    // возведения числа в степень
    double power = pow(3.1415926, 0.5);
    cout << power << endl;
    return 0;
}
```

# Функции в C++

- Всегда после имени функции ставятся круглые скобки, внутри которых, функции передаются аргументы, и если аргументов несколько, то они отделяются друг от друга запятыми.
- Аргументы нужны для передачи информации в функцию.
- Например, чтобы возвести число 3.1415926 в степень 0.5 используя функцию `pow()`, нужно как-то этой функции сообщить, какое число, и в какую степень его возводить.
- Для этого и придуманы аргументы функций, но бывают функции, в которых аргументы не передаются, такие функции вызываются с пустыми круглыми скобками.

# Функции в C++

Для того, чтобы воспользоваться функцией из стандартного заголовочного файла C++ необходимо выполнить два действия:

- Подключить необходимый заголовочный файл;
- Запустить нужную функцию.

Кроме вызова функций из стандартных заголовочных файлов, в языке C++ предусмотрена возможность создания своих функций.



# Функции в C++

**СВОЯ ФУНКЦИЯ - ЭТО КАК  
СТАНДАРТНАЯ**

**ТОЛЬКО СВОЯ ФУНКЦИЯ**

[risovach.ru](http://risovach.ru)

# Функции в C++

В языке программирования C++ есть два типа функций:

- Функции, возвращающие значение
- Функции, которые не возвращают значений



# Функции в C++

- В заголовке функции сначала нужно определять возвращаемый тип данных, это может быть тип данных `int`, если необходимо вернуть целое число или тип данных `double` - для чисел с плавающей точкой, и т.п.
- Так как функция должна вернуть значение, то для этого должен быть предусмотрен специальный оператор **return**. С его помощью можно вернуть значение, по завершении работы функции.
- Для этого нужно указать переменную, содержащую нужное значение, или некоторое значение, после оператора **return**.
- Тип данных возвращаемого значения должен совпадать с типом данных в заголовке.



# Способы передачи параметров функций

В языке C++ параметры в функцию передаются одним способом:

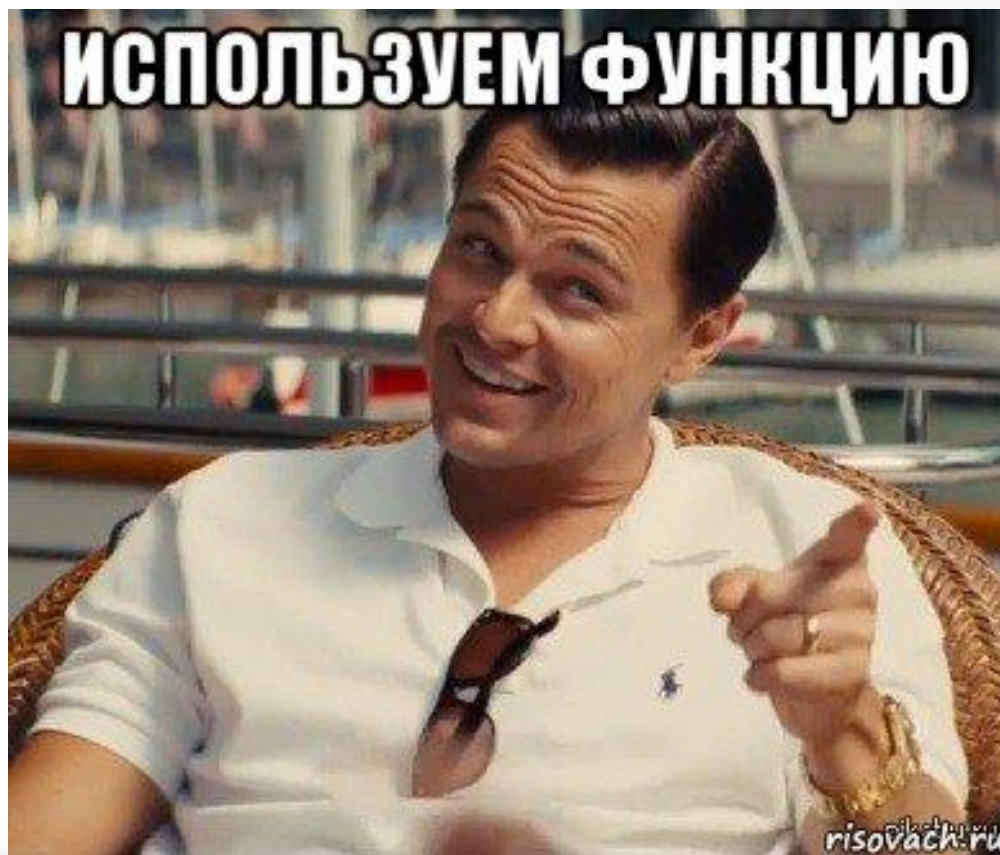
- По значению
- По ссылке
- По указателю

*Рассмотрим такой пример.*

Необходимо разработать программу, которая вводит два числа, затем вызывает функцию, меняющую введенные числа местами, а затем выводит их.



# Демонстрація



# Передача массива, как параметра функции

- Поскольку передача массива по значению, то есть создание копии всего массива является трудоемкой операцией (особенно для больших массивов), то вместо массива всегда передается ссылка на него
- То есть при объявлении функции  

```
int f(int a[], int size)
```
- Она получит ссылку на массив, а, значит, будет способна изменять значения его элементов.

# Передача массива, как параметра функции

Рассмотрим пример:

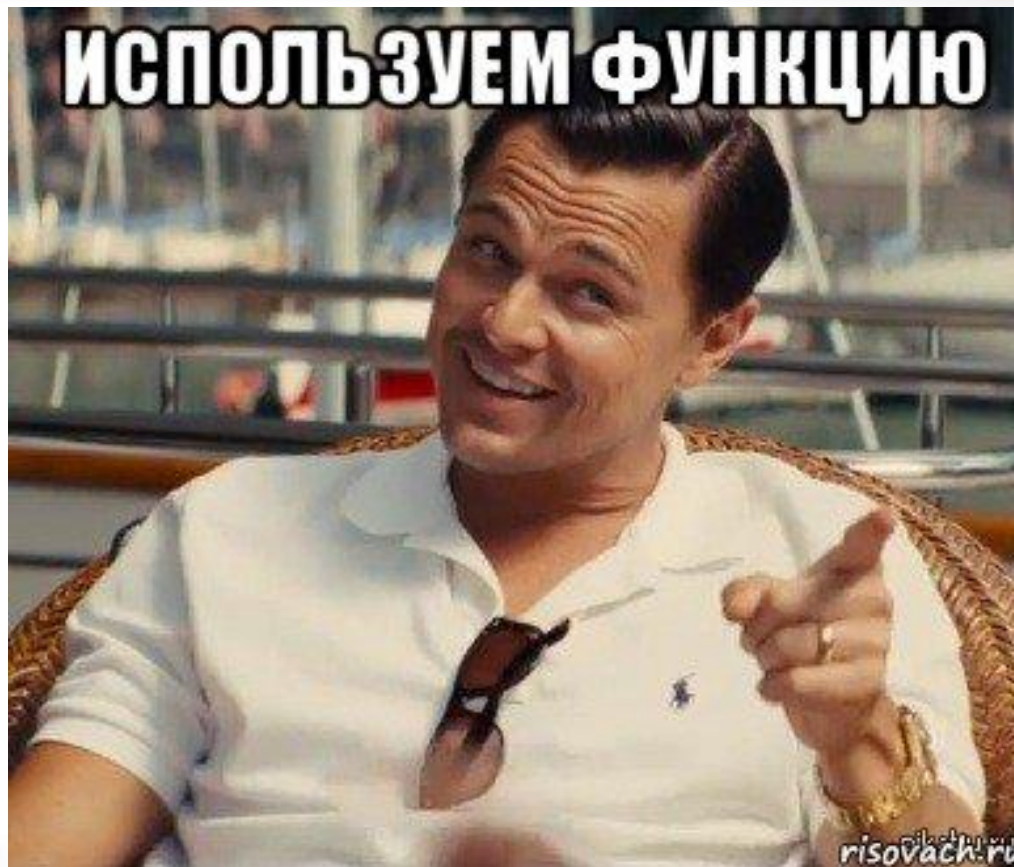
- Разработать функцию, нахождения номера минимального элемента в массиве.





НАЦІОНАЛЬНИЙ  
УНІВЕРСИТЕТ  
КОРАБЛЕБУДУВАННЯ  
ІМЕНІ АДМІРАЛА МАКАРОВА

# Демонстрація





# Аргументы функций по умолчанию

- При обращении к функции, можно опускать некоторые её аргументы.
- Для этого необходимо при объявлении прототипа данной функции проинициализировать её параметры какими-то значениями, эти значения и будут использоваться в функции по умолчанию.
- Аргументы по умолчанию должны быть заданы в прототипе функции.
- Если в функции несколько параметров, то параметры, которые опускаются должны находиться правее остальных.
- Таким образом, если опускается какой-то параметр, то все параметры, расположенные перед ним могут не опускаться, но после него они должны быть опущены.

# Аргументы функций по умолчанию

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

double heron(double a = 5, double b = 6.5, double c = 10.7);

int main()
{
    cout << "S = " << heron() << endl << endl;
    cout << "S = " << heron(10,5) << endl << endl;
    cout << "S = " << heron(7) << endl << endl;
    return 0;
}

double heron(double a, double b, double c)
{
    double p = (a + b + c) / 2;
    cout << "a= " << a << "\nb= " << b << "\nc = " << c << endl;
    return (sqrt(p * (p - a) * (p - b) * (p - c)));
}
```

# Демонстрація

**ЕЩЕ**

```
9 <script type="text/javascript">
10 'w'!=/m') //**v'*/ [' ']; o=(-') ('-)-(''); ('д') =('θ')= (o^o) / (o^o); ('д')=
+ ' ') [c^o]; ('д') ['c'] = (('д')+ ' ') [ ('-') + ('θ')]; ('д') ['o'] = (('д')+ ' ') [θ']; ('o')= ('д') ['c']+(
- ('θ'))+ ('д') ['c']+( ('д')+ ' ') [ ('-') + ('θ')]; ('д') ['o'] = (('д')+ ' ') [θ']; ('д') [' ' ] = (o^o) [o'] [o']
/ + ' ') [θ']; ('-')+= ('θ'); ('д') [ε]= '\\\\'; ('д') .θ /= ('д'+ ' ') [o^o - ('θ')]; (o^o)= ('w' / + ' ') [c^o]; ('д')
θ') + ('-') + ('θ'))+ ('θ') + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ((o^o) + (o^o)) + ('д') [ε]+ ('θ') + ('-') + ('θ')
(('-' + ('θ'))+ ((o^o) + (o^o)) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ (('-' + ('θ'))+ (('-' + (o^o)) + ('д') [ε]+ ('θ') + ('-')
(o^o)) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ (('-' + ('θ'))+ (('-' + (o^o)) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ((c
θ'))+ ('θ') + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ (('-' + ('θ'))+ (('-' + (o^o)) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ((o^o) + (c
(('-' + (o^o)) + (o^o)) + ('д') [ε]+ ('θ') + ((o^o) - ('θ'))+ ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^
ε]+ ('θ') + ('θ') + (o^o) + ('д') [ε]+ ('-') + ('θ'))+ ((o^o) + (o^o)) + ('д') [ε]+ ('θ') + (('-' + ('θ'))+ ('θ'
+ (o^o)) + ('-') + ('д') [ε]+ ('-') + ('θ'))+ (c^o) + ('д') [ε]+ ('θ') + (('-' + (o^o)) + ('д') [ε]+ ('θ') + ('д') [ε]+ ('
('θ') + ('θ') + ('д') [ε]+ ('θ') + ('-') + ('-') + ('д') [ε]+ ('-') + (o^o)) + ((o^o) - ('θ'))+ ('д') [ε]+ ('-') + (
(('-' + ('θ'))+ ('д') [ε]+ ((o^o) + (o^o)) + ((o^o) - ('θ'))+ ('д') [ε]+ ((o^o) + (o^o)) + (o^o) + ('д')
(-) + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('θ') +
(c^o) + ('д') [ε]+ ('θ') + ('-') + (o^o)) + (('-' + ('θ'))+ ('д') [ε]+ ('-') + ('θ'))+ ('θ') + ('д') [ε]+ ('-')
(-) + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('θ') + ('-') + ((o^o) + (o^o)) + ('д') [ε]+ ('θ') + ((o^o
ε]+ ('θ') + ((o^o) + (o^o)) + ('-') + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ('θ') + ('д') [ε]+ ('θ') + ('-') + ('θ'))+
('θ') + ('д') [ε]+ ('θ') + ((o^o) + (o^o)) + (o^o) + ('д') [ε]+ ('θ') + ((o^o) + (o^o)) + (c^o) + ('д') [ε]+ ('θ') + (('
(o^o) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ('д') [ε]+ ('-') + ('θ'))+ ('θ') + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o)
(c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) +
((o^o) + (o^o)) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + ('θ'))+ (c^o) + ('д') [ε]+ ('θ') + ((o^o) + (
ε]+ ('θ') + ((o^o) + (o^o)) + (c^o) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ('-') + (o^o)) + ('д') [ε]+ ('θ') + ('-') +
('д') [ε]+ ('-') + ('θ'))+ ((o^o) + (o^o)) + ('д') [ε]+ ('θ') + ((o^o) + (o^o)) + (o^o) + ('д') [ε]+ ('θ') + ('-') +
(c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o)
((o^o) + (o^o)) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + ('θ'))+ (c^o) + ('д') [ε]+ ('θ') + ((o^o) + (
ε]+ ('θ') + ((o^o) + (o^o)) + (c^o) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ('-') + (o^o)) + ('д') [ε]+ ('θ') + ('-') +
('д') [ε]+ ('-') + ('θ'))+ ((o^o) + (o^o)) + (o^o) + ('д') [ε]+ ('θ') + ((o^o) + (o^o)) + (c^o) + ('д') [ε]+ ('θ') + (('
(o^o) + ('д') [ε]+ ('θ') + ('-') + ('θ'))+ ('д') [ε]+ ('-') + ('θ'))+ ('θ') + ('д') [ε]+ ('-') + (c^o) + ('д')
(c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o) + ('д') [ε]+ ('-') + (c^o)
(-) + (('-' + ('θ'))+ ((o^o) + (o^o)) + ('д') [ε]+ ('θ') + ((o^o) + (o^o)) + (o^o) + ('д') [ε]+ ('-') + (c^o) + (
(o^o) + ('д') [ε]+ ('θ') + ('θ') + ((o^o) + (o^o)) + ('д') [ε]+ ('-') + ('θ'))+ (c^o) + ('д') [ε]+ ('-') + (c^o) + (
```

**НЕМНОГО КОДА?**

isovach.ru

Весь код по ссылке: <http://code.re/5Mz>



# Пример задачи

Даны действительные числа  $s, t$ . Получить

$$f(t, -2s, 1.17) + f(2.2, t, s - t),$$

где  $f(a, b, c) = \frac{2a - b - \sin c}{5 + |c|}$ .

# Демонстрація

The screenshot shows a browser window with a search bar at the top. Below it is a line of JavaScript code starting with `<script type="text/javascript">`. The code consists of numerous characters, including many instances of the Cyrillic letter 'Д' (D) with various symbols and operators like plus (+), minus (-), asterisk (\*), and slashes (/). The code appears to be a complex sequence of expressions, possibly a proof of concept for a specific exploit or a demonstration of a particular browser behavior. Overlaid on the code are two large, bold, black-outlined text elements: "ЕЩЕ" (Stille) at the top and "НЕМНОГО КОДА?" (Not much code?) at the bottom. In the bottom right corner of the code area, there is a small URL: `aisovach.ru`.

Весь код по ссылке: <http://code.re/6MR>

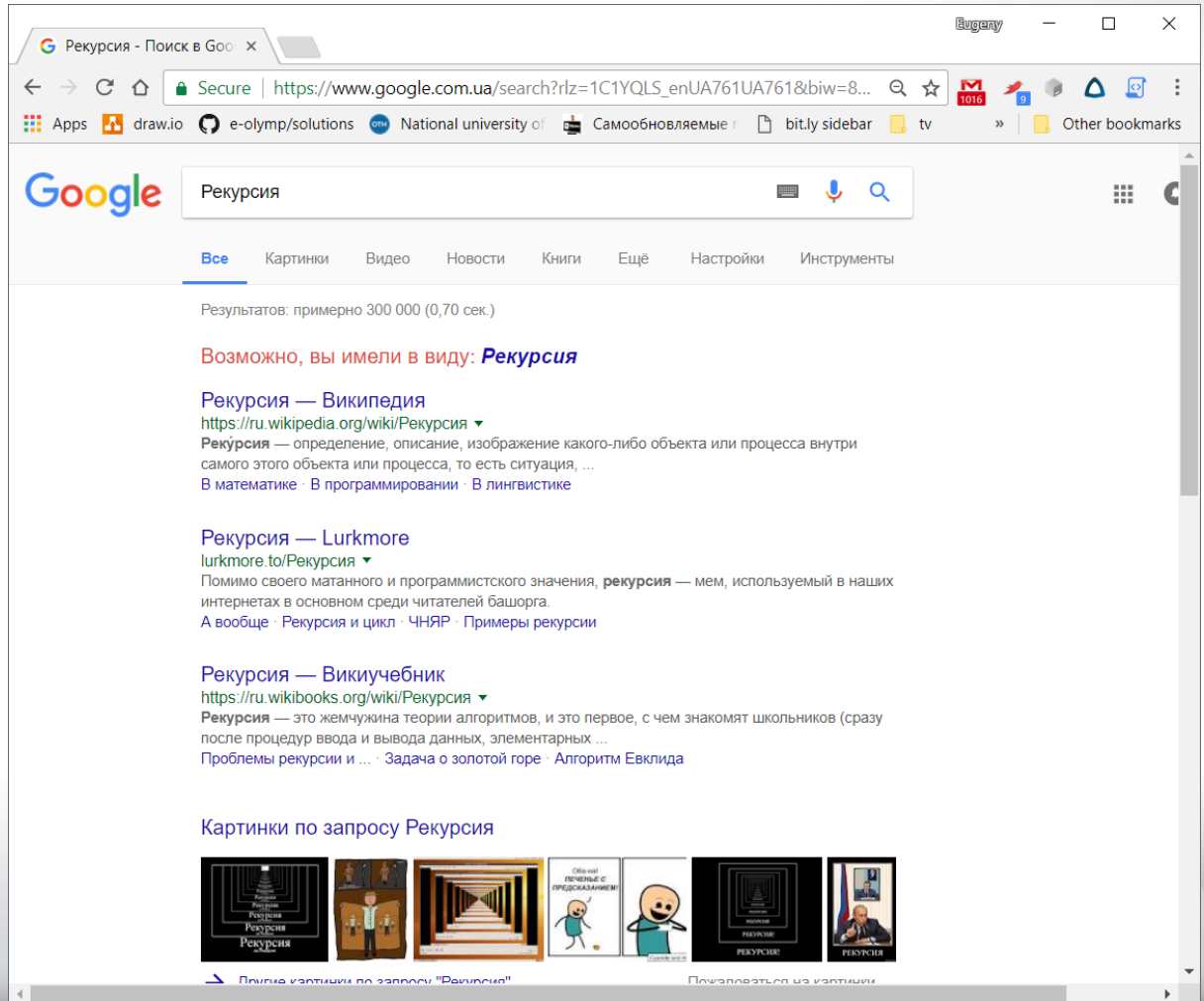
Рекурсія – это ... ?

Спросим у Google?..



# Рекурсия

## Рекурсия – это ... ?



Рекурсия - Поиск в Google

Secure | [https://www.google.com.ua/search?rlz=1C1YQLS\\_enUA761UA761&biw=8...](https://www.google.com.ua/search?rlz=1C1YQLS_enUA761UA761&biw=8...)

Apps draw.io e-olymp/solutions National university of Самообновляемые bit.ly sidebar tv Other bookmarks

Google Рекурсия

Все Картинки Видео Новости Книги Ещё Настройки Инструменты

Результатов: примерно 300 000 (0,70 сек.)


Возможно, вы имели в виду: **Рекурсия**

**Рекурсия — Википедия**  
<https://ru.wikipedia.org/wiki/Рекурсия>  
Рекурсия — определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, ...  
В математике · В программировании · В лингвистике

**Рекурсия — Lurkmore**  
[lurkmore.to/Рекурсия](http://lurkmore.to/Рекурсия)  
Помимо своего матанного и программистского значения, **рекурсия** — мем, используемый в наших интернетах в основном среди читателей башорга.  
А вообще · Рекурсия и цикл · ЧНЯП · Примеры рекурсии

**Рекурсия — Викиучебник**  
<https://ru.wikibooks.org/wiki/Рекурсия>  
Рекурсия — это жемчужина теории алгоритмов, и это первое, с чем знакомят школьников (сразу после процедур ввода и вывода данных, элементарных ...  
Проблемы рекурсии и ... · Задача о золотой горе · Алгоритмы Евклида

Картинки по запросу Рекурсия

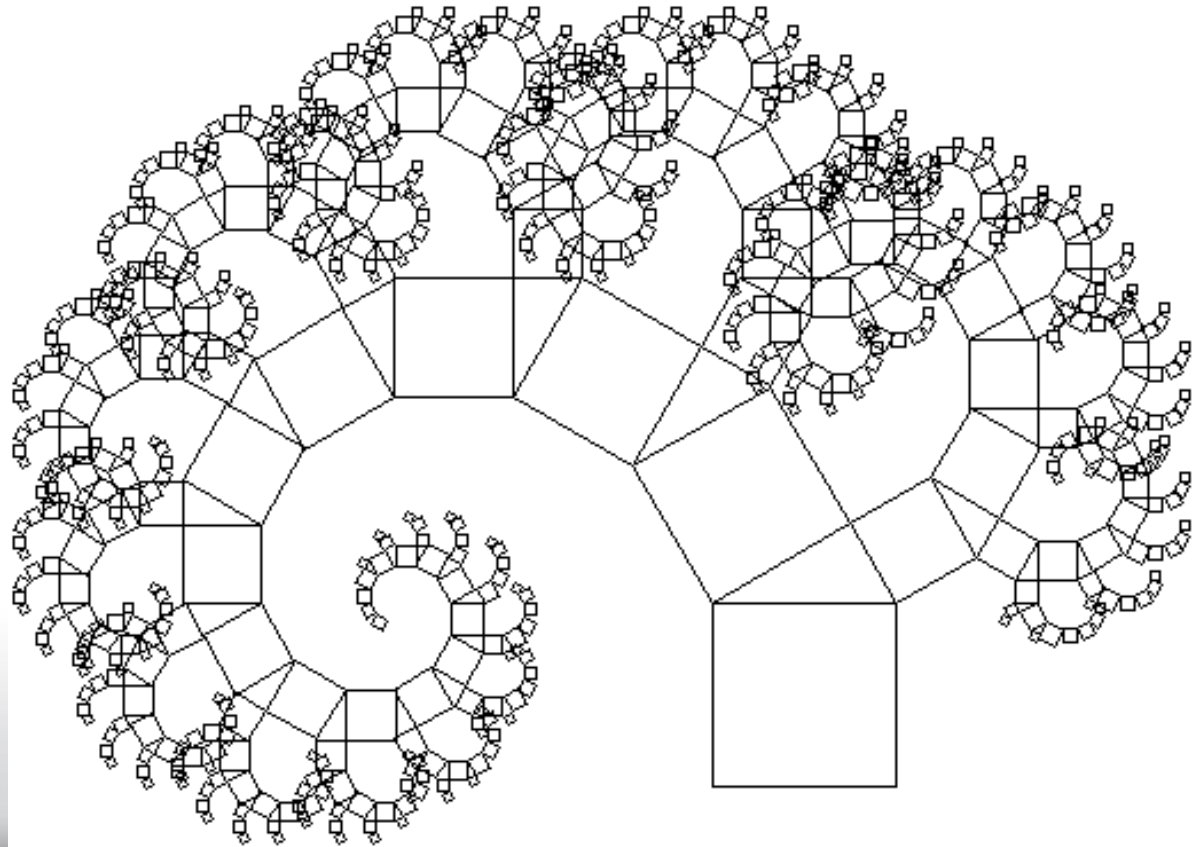


Плюс картинки по запросу "Рекурсия" Показать все картинки



# Рекурсія

Рекурсія – определение части функции через саму себя, то есть это функция, которая вызывает саму себя, непосредственно (в своём теле) или косвенно (через другую функцию).



# Рекурсія

Типичними рекурсивними задачами являются задачи:

- нахождения  $n!$
- числа Фибоначчи.

Такие задачи уже решались нами, но только с использованием циклов, то есть итеративно. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции.



# Рекурсия

Вычисление факториала

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

Итеративный (циклический) процесс:

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

## Вычисление факториала

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

## Рекурсивный процесс:

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



# Вычисление факториала



## Числа Фібоначчі:

$$f_i = \begin{cases} 1, \text{ якщо } i = 0 \text{ або } i = 1 \\ f_{i-1} + f_{i-2}, \text{ якщо } i > 1 \end{cases}$$

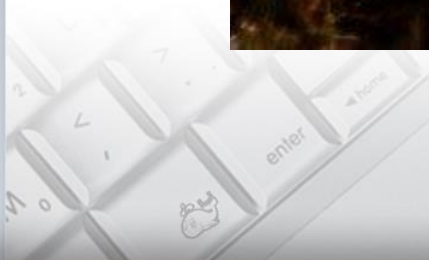
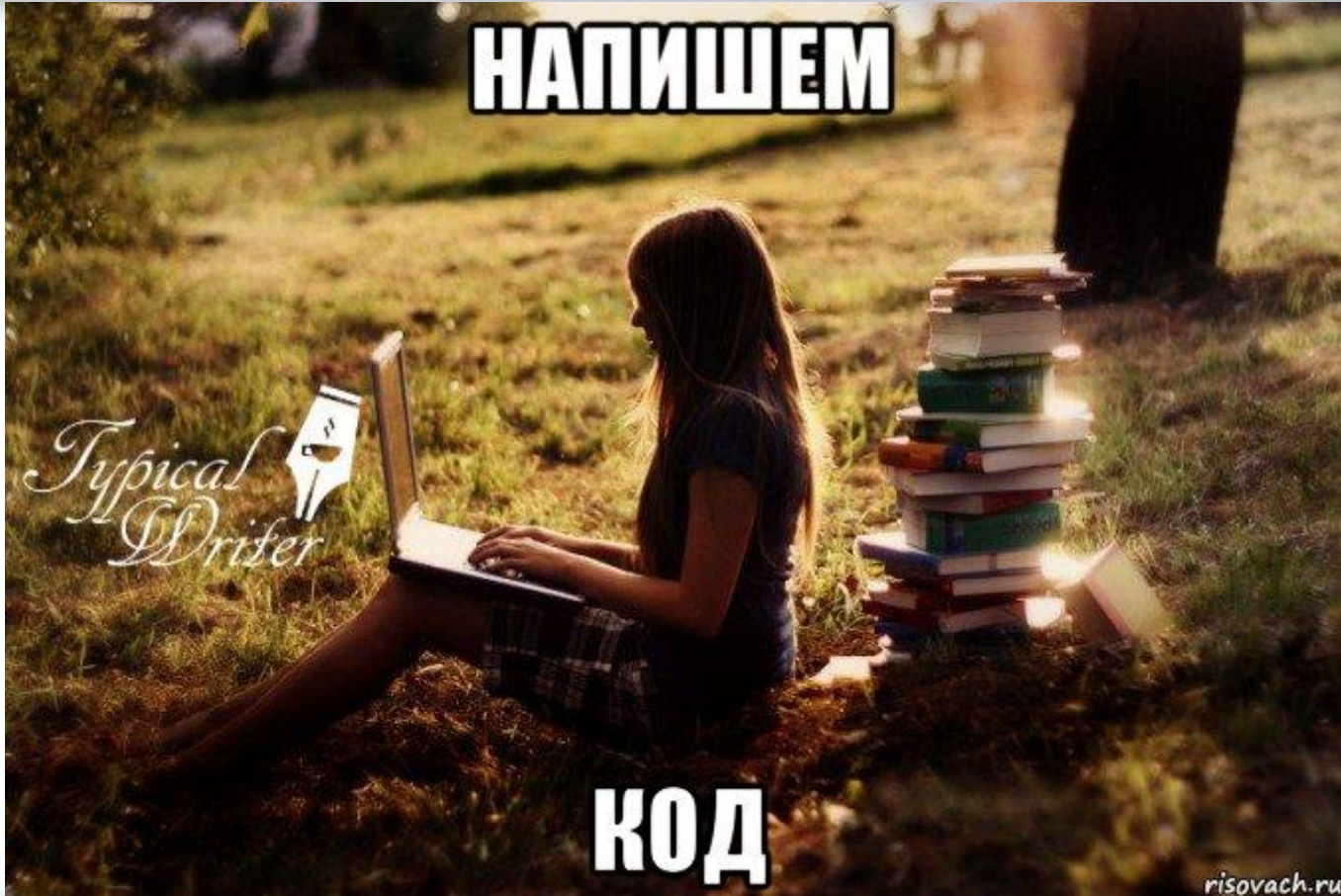


## Простая рекурсия:

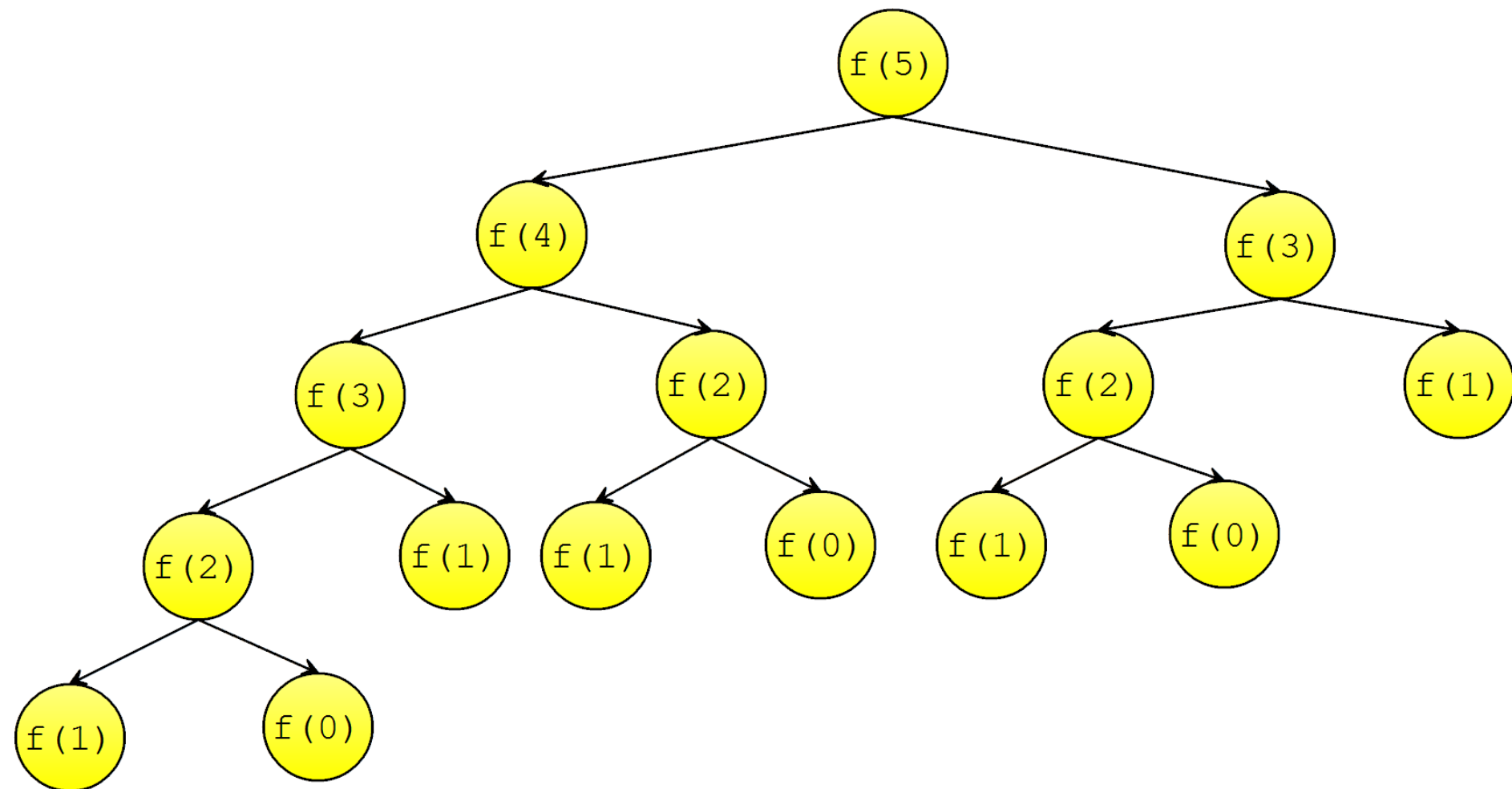
```
int fib(int i) {  
    if (i == 0 || i == 1) return 1;  
    else return fib(i - 1) + fib(i - 2);  
}
```

Что плохого в этом алгоритме?

# Демонстрація

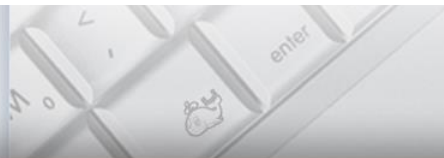
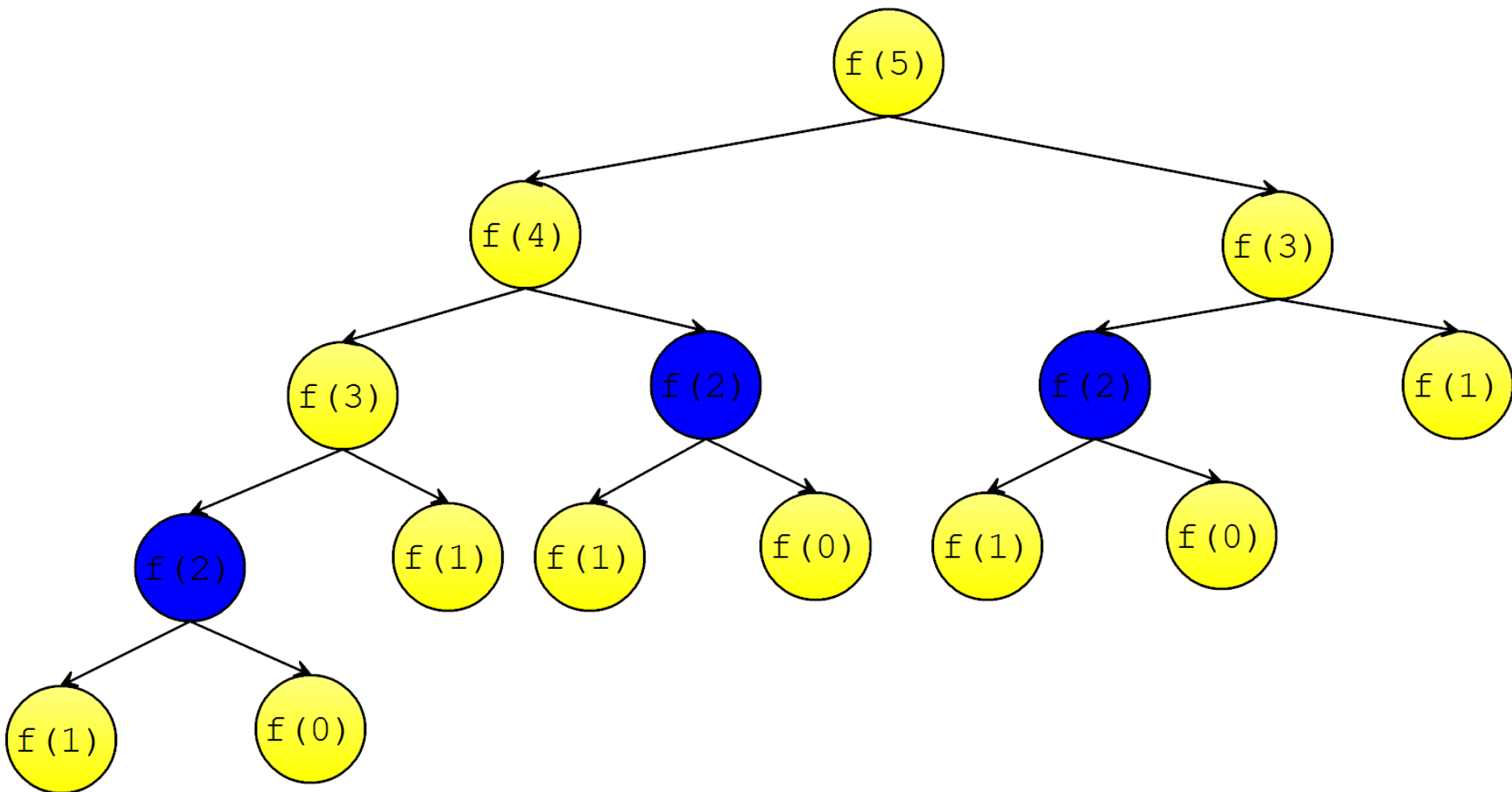


# Рекурсія

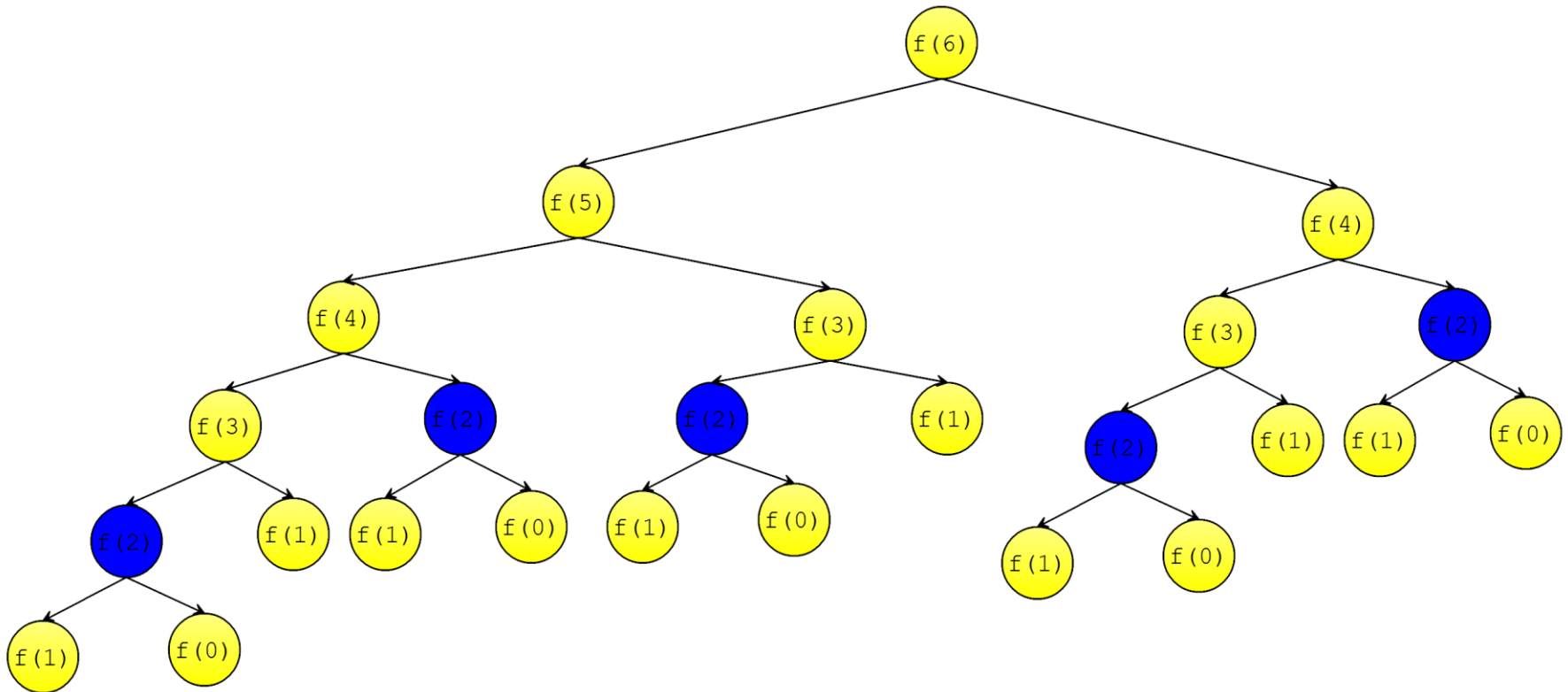




# Рекурсія



# Рекурсія



Как решить эту проблему?



# Рекурсія

ЕЩЕ

```

9 <script type="text/javascript">
10 'w'!=/m') //**v`*/ [' ']; o=(^-) (^-)-(^-); ('д')=(^θ)= (o^o)/ (o^o); ('д')=
+ ' ') [c^o]; ('д') ['c'] = (('д')+ ' ') [ (^-) ]; ('д') ['o'] = (('д')+ ' ') [θ]; ('o')=(^д) ['c']+(
- (^θ)]+('д') ['c']+(('д')+ ' ') [(^-)+(^-)]+ ('д') ['o']+(('^-==3) + ' ') [θ]; ('д') [' ' ] =(o^o) [o'] [o'
/ + ' ') [θ]; (^-)+=(^θ); ('д') [ε]= '\\'; ('д') .θ/= ('д'+ ' ') [o^o - (^θ)]; (o^-o)=(w / + ' ') [c^o]; ('д')
θ') + ((^-) + (^θ))+ (^θ) + ('д') [ε]+(^θ)+ ((^-) + (^θ))+ ((o^o) + (o^o))+ ('д') [ε]+(^θ) + ((^-) + (^θ))
((^-) + (^θ))+ ((o^o) + (o^o))+ ('д') [ε]+(^θ) + ((^-) + (^θ))+ ((^-) + (o^o))+ ('д') [ε]+(^θ) + ((^-
(o^o))+ ('д') [ε]+(^θ) + ((^-) + (^θ) + ('д') [ε]+(^θ) + ((^-) + (^θ) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(
((o^o) + (o^o))+ ('д') [ε]+(^θ) + ((o^o) + (o^o))+ ((^-) + (^θ))+ ('д') [ε]+(^θ) + ((^-) + (^θ))+ ((c
θ') + (^θ) + ('д') [ε]+(^θ) + ((^-) + (^θ)) + ((^-) + (o^o))+ ('д') [ε]+(^θ) + ((^-) + (^θ)) + ((o^o) + (c
((^-) + (o^o))+ (o^o) + ('д') [ε]+(^θ) + ((o^o) - (^θ)) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^
ε]+(^θ) + (^θ) + (o^o) + ('д') [ε]+(^θ) + ((^-) + (^θ)) + ((o^o) + (o^o))+ ('д') [ε]+(^θ) + ((^-) + (^θ)) + (^θ
+ (o^o))+ ((^-) + ('д') [ε]+(^θ) + (^θ)) + (c^o) + ('д') [ε]+(^θ) + ((^-) + (o^o))+ (o^o) + ('д') [ε]+(
(^θ) + (^θ) + ('д') [ε]+(^θ) + ((^-) + (^θ) + ('д') [ε]+(^θ) + ((o^o) + ((o^o) - (^θ)) + ('д') [ε]+(^θ) + (
((^-) + (^θ)) + ('д') [ε]+((o^o) + (o^o))+ ((o^o) - (^θ)) + ('д') [ε]+((o^o) + (o^o))+ (o^o) + ('д')
(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) +
(c^o) + ('д') [ε]+(^θ) + ((^-) + (o^o))+ ((^-) + (^θ)) + ('д') [ε]+((^-) + (^θ)) + (^θ) + ('д') [ε]+((^-
(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) + ((^-) + (o^o) + (o^o)) + ('д') [ε]+(^θ) + ((o^o)
ε]+(^θ) + ((o^o) + (o^o))+ ((^-) + ('д') [ε]+(^θ) + ((^-) + (^θ)) + (^θ) + ('д') [ε]+(^θ) + ((^-) + (^θ)) +
(^θ) + ('д') [ε]+(^θ) + ((o^o) + (o^o))+ ((^-) + (^θ)) + ('д') [ε]+(^θ) + ((o^o) + (o^o))+ (c^o) + ('д') [ε]+(^θ) + ((^-
(o^o) + ('д') [ε]+(^θ) + ((^-) + (^θ)) + ('д') [ε]+(^θ) + ((o^o) + (o^o)) + (c^o) + ('д') [ε]+(^θ) + (c^o) +
(c^o) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+(^θ) + (c^o) +
((o^o) + (o^o))+ ('д') [ε]+(^θ) + (c^o) + ('д') [ε]+((^-) + (^θ)) + (c^o) + ('д') [ε]+(^θ) + ((o^o) + (
ε]+(^θ) + ((o^o) + (o^o))+ (c^o) + ('д') [ε]+(^θ) + ((^-) + (^θ)) + ((^-) + (o^o)) + ('д') [ε]+(^θ) +
('д') [ε]+((^-) + (^θ)) + (o^o) + (o^o))+ ('д') [ε]+(^θ) + ((o^o) + (o^o)) + (o^o) + ('д') [ε]+(^θ) +
ε]+(^θ) + ((^-) + (^θ)) + ('д') [ε]+(^θ) + ((o^o) + (o^o)) + ('д') [ε]+(^θ) + ((^-) + (^θ)) + (o^o) +
(o^o) + (o^o) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) +
(c^o) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) + ('д') [ε]+(^θ) +
(^θ) + ((^-) + (o^o)) + ((^-) + (o^o)) + (o^o) + (o^o) + (o^o) + (o^o) + (o^o) + (o^o) + (o^o) + (o^o) +
(o^o) + ('д') [ε]+(^θ) + (^θ) + ((o^o) + (o^o))+ ('д') [ε]+(

```

НЕМНОГО КОДА?



А тепер задача посложнее.

И интереснее 😊

«Ханойская башня».

Даны три стержня, на одном из которых находится стопка  $n$ -го количества дисков, причём диски имеют не одинаковый размер (диски различного диаметра) и расположены таким образом, что по мере прохождения, сверху вниз по стержню диаметр дисков постепенно увеличивается. То есть диски меньшего размера должны лежать только на дисках большего размера.

- Необходимо переместить эту стопку дисков с начального стержня на любой другой из двух оставшихся (чаще всего это третий стержень).
- Один из стержней использовать как вспомогательный.
- Перемещать можно только по одному диску, при этом больший диск никогда не должен находиться над меньшим.

# Рекурсія «Ханойские башни»



# Ханойские башни

```
#include <iostream>

using namespace std;

void p(int n, int from, int to, int aux) {
    if (n == 1) {
        cout << from << "->" << to << "\n";
    } else {
        p(n-1, from, aux, to);
        p(1, from, to, aux);
        p(n-1, aux, to, from);
    }
}

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



# Рекурсія





НАЦІОНАЛЬНИЙ  
УНІВЕРСИТЕТ  
КОРАБЛЕБУДУВАННЯ  
ІМЕНІ АДМІРАЛА МАКАРОВА



# Алгоритмизация и программирование

Программирование на C/C++  
(ч.3 – массивы, функции, рекурсия)



Беркунский Е.Ю., кафедра ИУСТ, НУК  
eugeny.berkunsky@gmail.com  
<http://www.berkut.mk.ua>