Lecture 6

Functions

What is a function?

A function is a *block of code* which only runs when it is *called*.

You can pass data, known as parameters, into a function.

Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

http://ejudge.kz/reference/en/cpp/language/functions.html

Function return types

That return a value:

- int
- double
- bool
- string
- …
- etc, *any type you want*

That return nothing:

- void

Function examples

- `max()`
- `min()`
- `swap()`

`<cmath>`

- `sqrt()`
- `log2()`
- `pow()`

- `tolower()`
- `toupper()`
- `isalpha()`
- `isdigit()`
- `isalnum()`
- `ispunct()`

Calling a function

Examples:

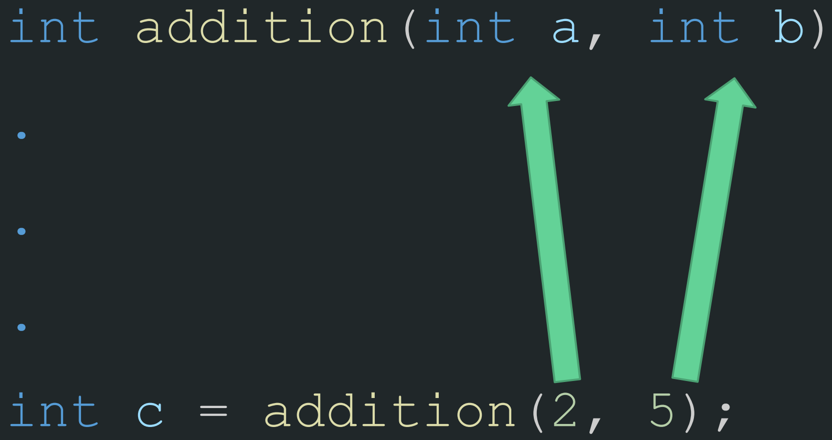
`max(4, 5);` – returns the larger number of the two given

`sqrt(9);` – returns the square root of the given number

`pow(2, 6);` – returns the first number to the power of the second number (2 to the power of 6)

Calling a function

Examples:

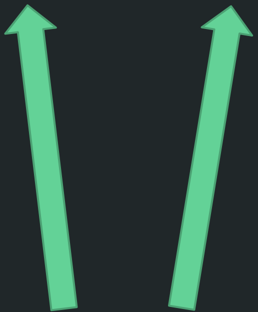


```
int addition(int a, int b)
.
.
.
int c = addition(2, 5);
// c = 7;
```

Calling a function

Examples:

```
int addition(int a, int b)
.
.
.
int c = addition(2, 5);
// c = 7;
```



Order of parameters and their number should be exactly the same as in the function declaration.

Otherwise, you will get an error.

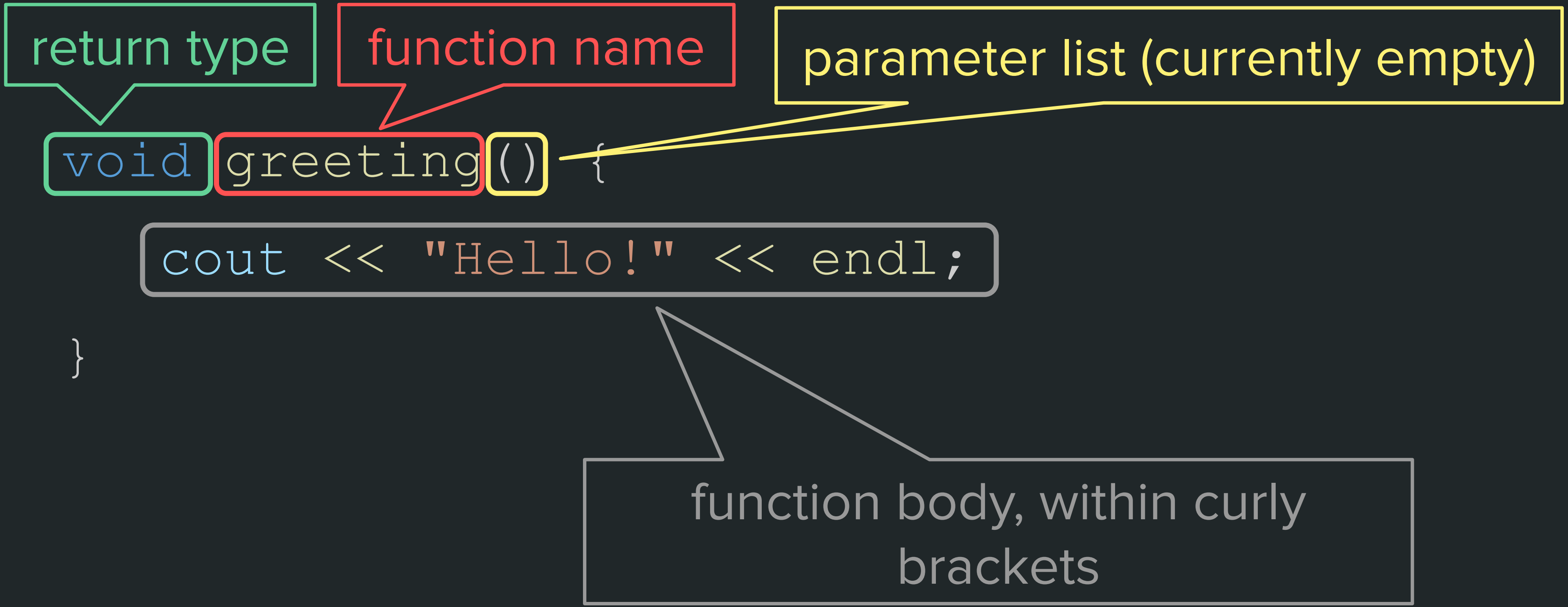
Local and global variables

Two types of variable scopes:

- Local Variables
- Global Variables

Function example

```
void greeting() {
    cout << "Hello!" << endl;
}
```



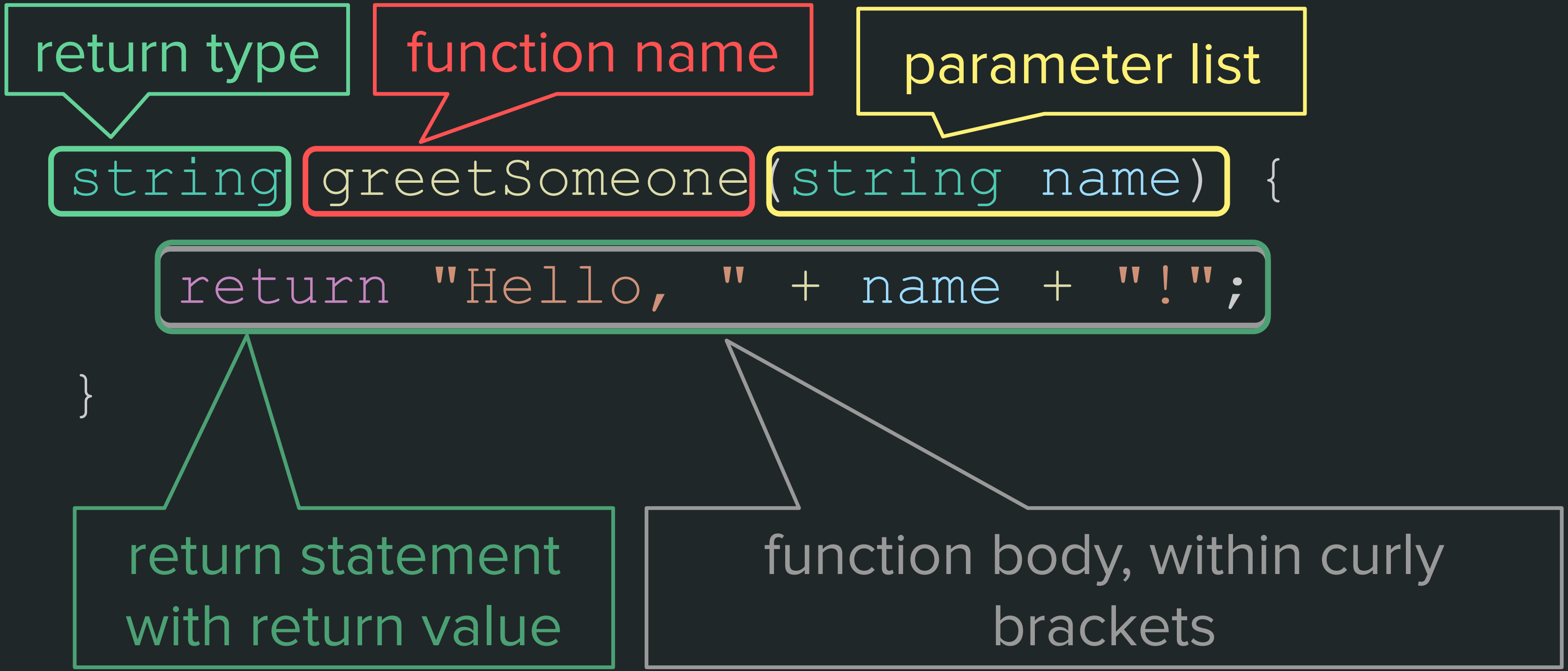
return type

function name

parameter list (currently empty)

```
void greeting() {
    cout << "Hello!" << endl;
}
```

function body, within curly brackets



```
string greetSomeone(string name) {
    return "Hello, " + name + "!";
}
```

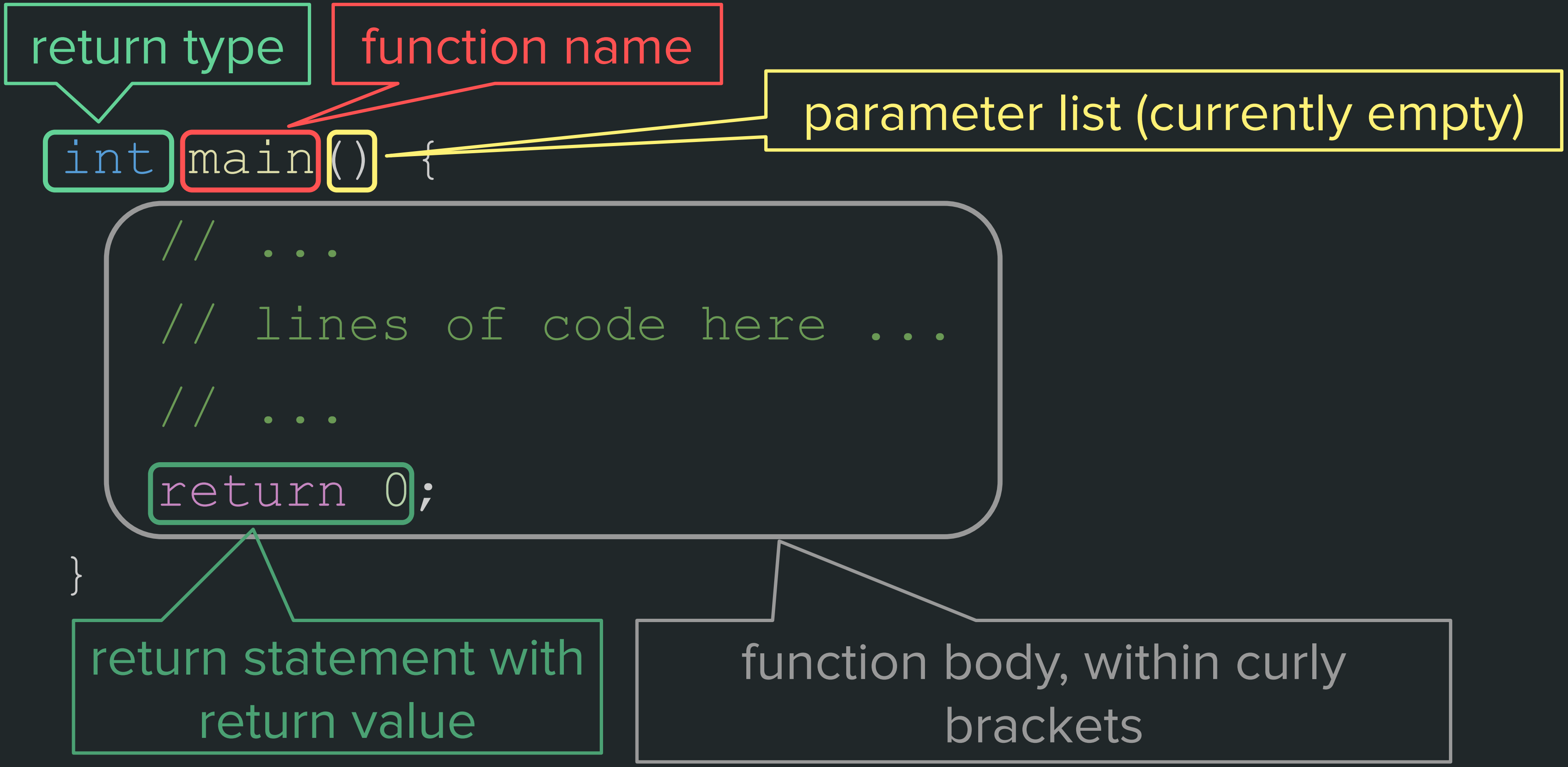
return type

function name

parameter list

return statement with return value

function body, within curly brackets



return type

function name

parameter list (currently empty)

```
int main() {
    // ...
    // lines of code here ...
    // ...
    return 0;
}
```

return statement with return value

function body, within curly brackets

Void return type

```
void greeting() {
   cout << "Hello!" << endl;
   // return "Hello"; - this is a mistake
}
```

Void return type

```
void greeting() {
    cout << "Hello!" << endl;
    // return "Hello"; - mistake if your
function is void
}
```

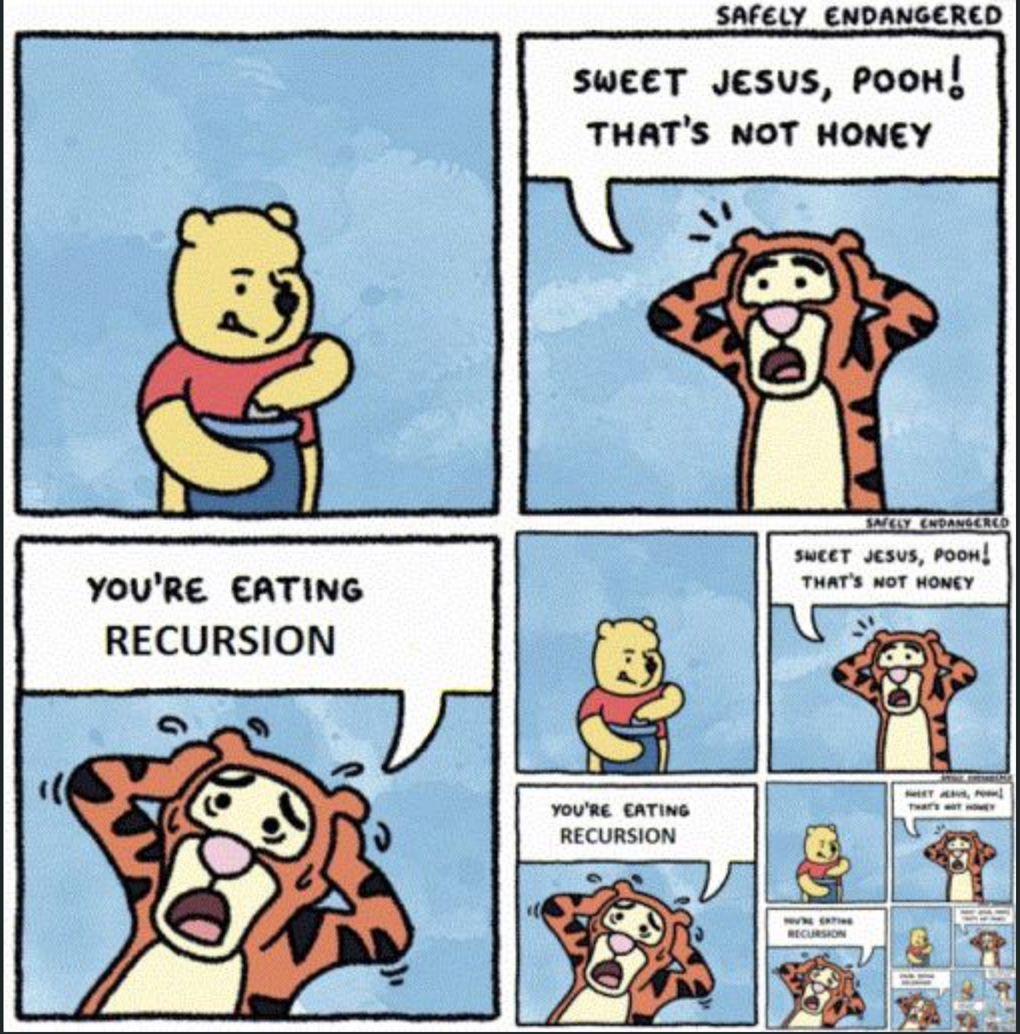
What is recursion?

Recursion, by definition, is “when a thing is defined in terms of itself.”

A recursive function is a function that calls itself, either directly, or indirectly (through another function).

What is recursion?

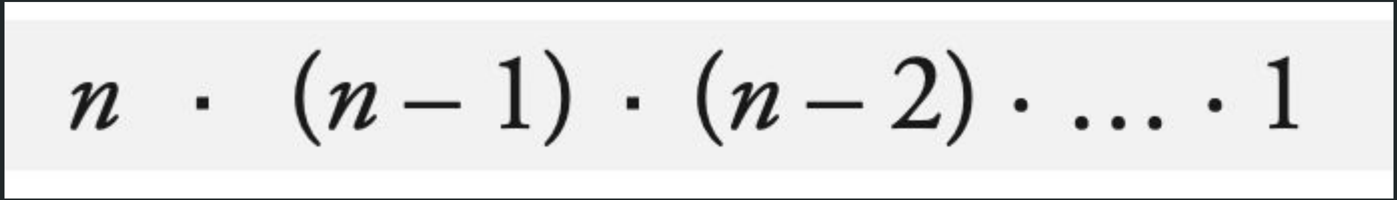
Example:



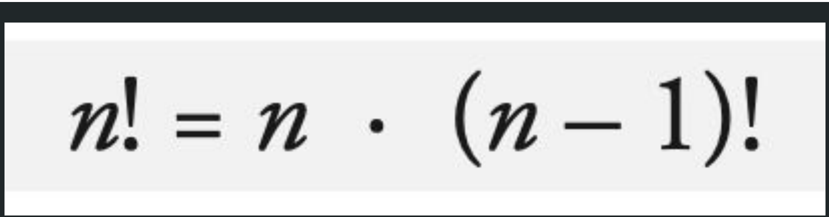
Key components of recursion

- Base case
  - condition to which the recursion converges
  - when it is reached, the recursion stops
- Problem reduction
  - with each recursive function call the problem should become smaller and be closer to the base case

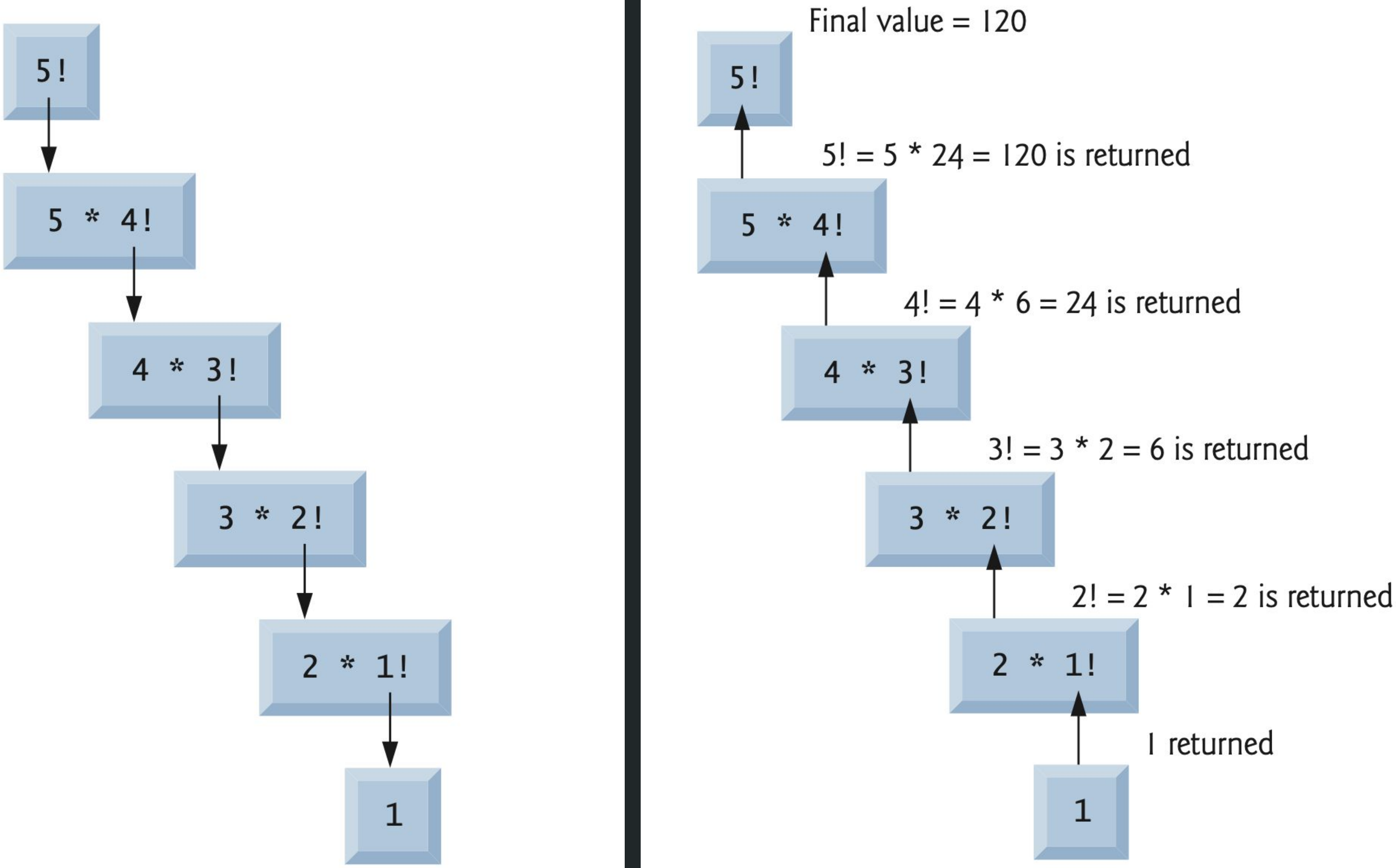
Factorial



$$n \;\cdot\; (n-1) \cdot (n-2) \cdot \ldots \cdot 1$$



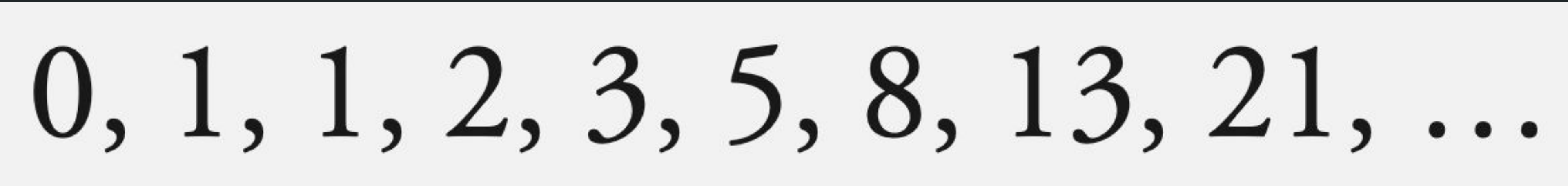
$$n! = n \;\cdot\; (n-1)!$$


5!
5 * 4!
4 * 3!
3 * 2!
2 * 1!
1
Final value = 120
5!
5! = 5 * 24 = 120 is returned
5 * 4!
4! = 4 * 6 = 24 is returned
4 * 3!
3! = 3 * 2 = 6 is returned
3 * 2!
2! = 2 * 1 = 2 is returned
2 * 1!
1 returned
1

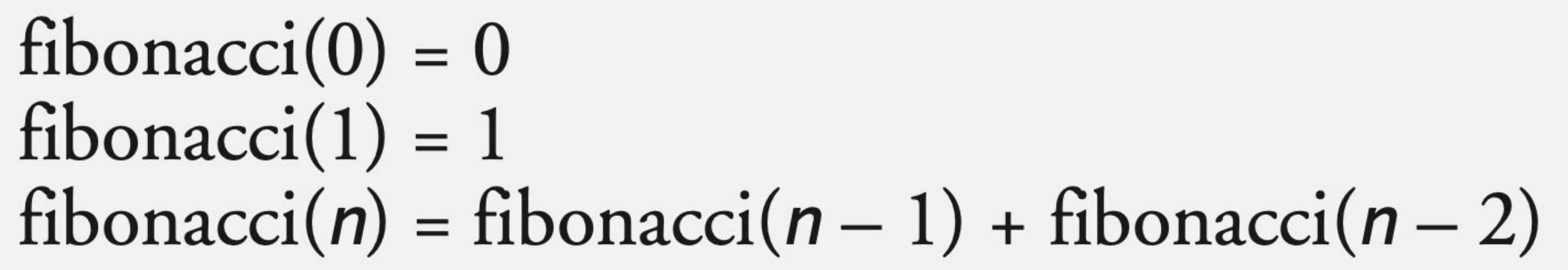
(a) Procession of recursive calls

(b) Values returned from each recursive call

Fibonacci series



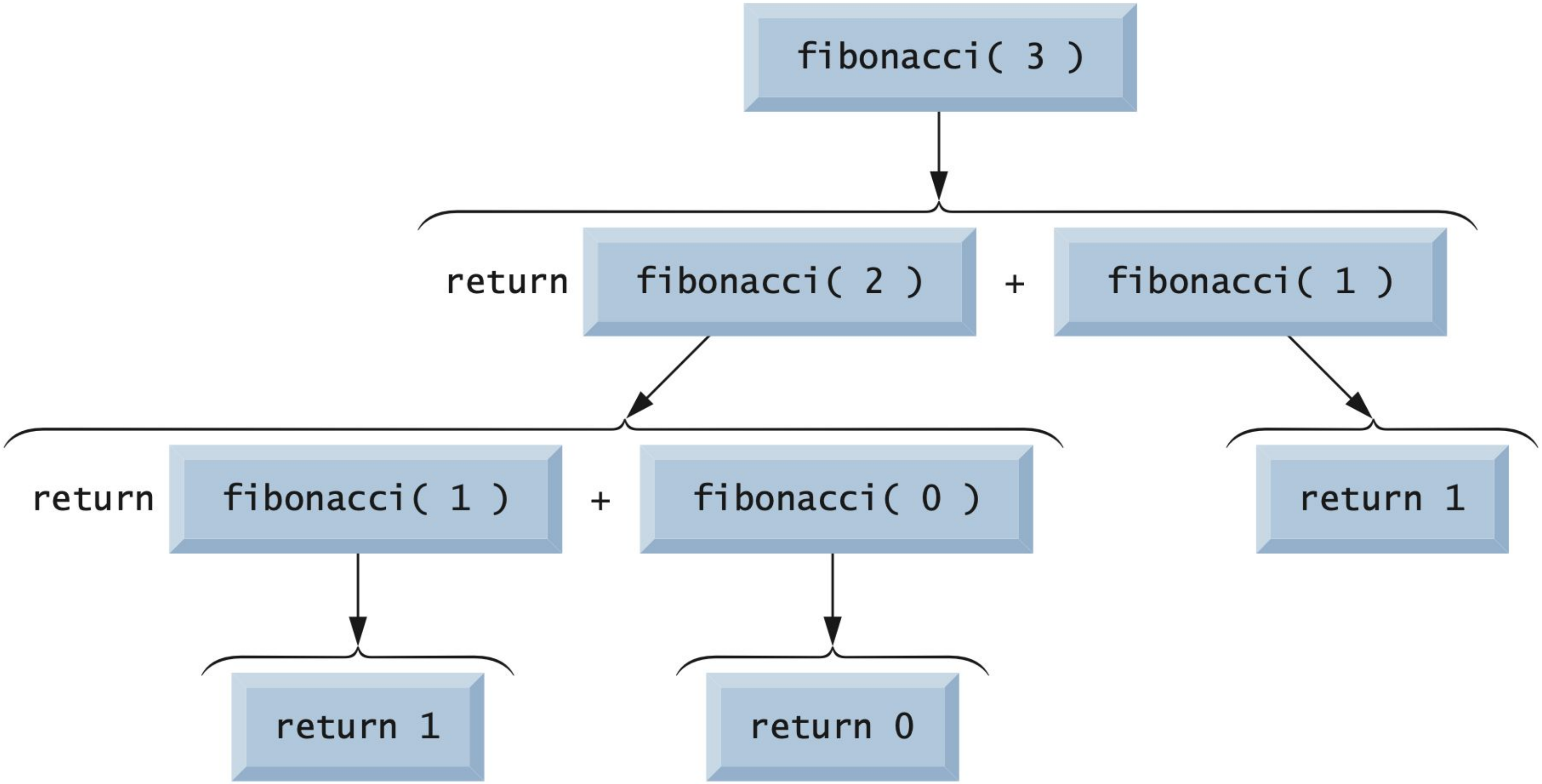
$$0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$$



$$\text{fibonacci}(0) = 0$$

$$\text{fibonacci}(1) = 1$$

$$\text{fibonacci}(n) = \text{fibonacci}(n - 1) + \text{fibonacci}(n - 2)$$


fibonacci(3)
return
fibonacci(2)
+
fibonacci(1)
return
fibonacci(1)
+
fibonacci(0)
return 1
return 1
return 0

Recursion vs. Iteration

Both iteration and recursion are based on a control statement: Iteration uses a repetition structure; recursion uses a selection structure. Both iteration and recursion involve repetition: Iteration explicitly uses a repetition structure; recursion achieves repetition through repeated function calls. Iteration and recursion both involve a termination test: Iteration terminates when the loop-continuation condition fails; recursion terminates when a base case is recognized. Iteration with counter-controlled repetition and recursion both gradually approach termination: Iteration modifies a counter until the counter assumes a value that makes the loop-continuation condition fail; recursion produces simpler versions of the original problem until the base case is reached. Both iteration and recursion can occur infinitely: An infinite loop occurs with iteration if the loop-continuation test never becomes false;infinite recursion occurs if the recursion step does not reduce the problem during each recursive call in a manner that converges on the base case.

Recursion vs. Iteration

- Any problem that can be solved recursively can also be solved iteratively (non-recursively).
- A recursive approach is normally chosen *when the recursive approach more naturally mirrors the problem* and *results in a program that’s easier to understand and debug*.
- Another reason to choose a recursive solution is that an iterative solution is not apparent.

Additional materials, functions

- Paper:
  - C++ How to Program, Seventh Edition, H. M. Deitel, P. J. Deitel:
    - Chapter 6, Sections 6.1 - 6.7, 6.10, 6.12 (available in the KBTU library);
- Digital:
  - informatics.msk.ru:
    - Теоретический материал (C++): Функции - 1
    - Функции и процедуры. Рекурсия
  - w3schools:
    - C++ Functions

Additional materials, recursion

- Paper:
    - C++ How to Program, Seventh Edition, H. M. Deitel, P. J. Deitel:
        - Chapter 6, Sections 6.19 - 6.21 (available in the KBTU library);
- Digital:
    - informatics.msk.ru:
        - Функции: Условия задач
        - Функции и процедуры. Рекурсия
    - w3schools:
        - C++ Function Recursion