ECE PARIS • LYON
ÉCOLE D'INGÉNIEURS

Informatique 2

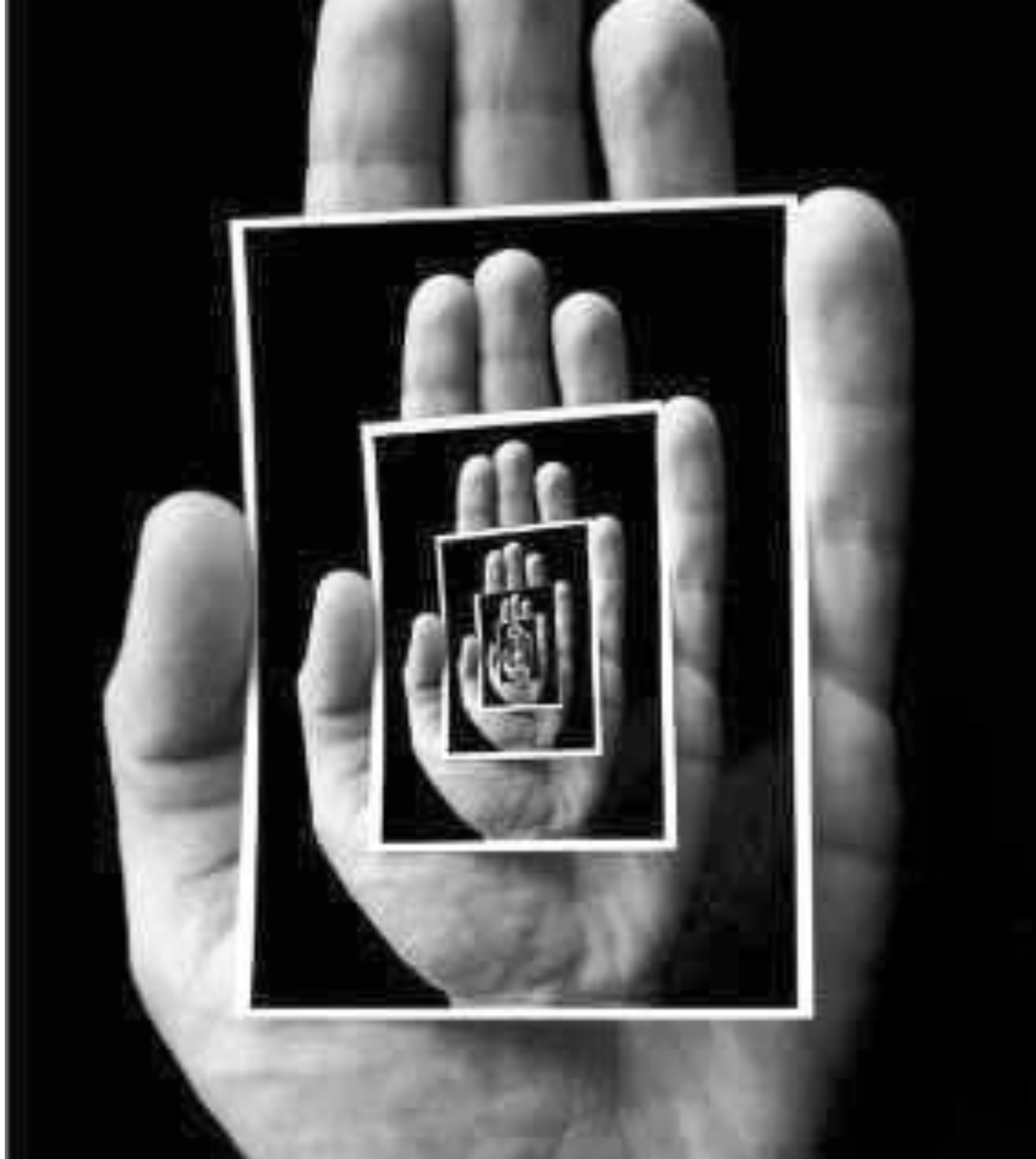
Cours n°11

Récursivité

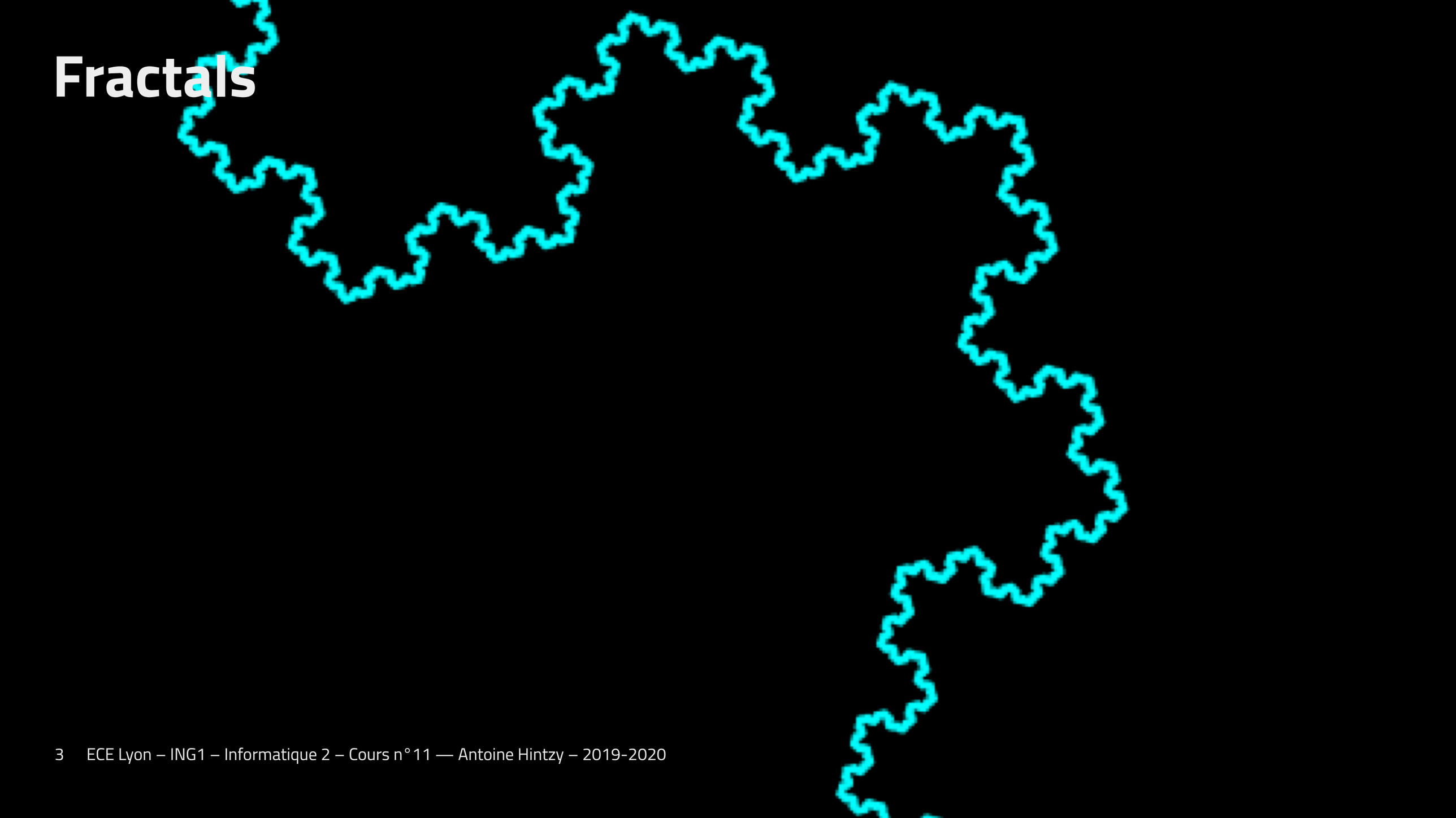
Antoine Hintzy

Présentation

- Acronyme récursif :
  - **Allegro** : **A**llegro **L**ow **LE**vel **G**ame **RO**utines
  - **BING** : **B**ing **I**s **N**ot **G**oogle
  - **GNU** : **G**NU's **N**ot **U**nix
  - etc.
- Mise en abyme :
  - Photo de droite.
  - Logo de la vache qui rit.



Fractals



Définition

Une fonction est récursive si elle s'appelle elle-même.

L'appel de la fonction à l'intérieur d'elle-même est nommé **appel récursif**.

Un appel récursif doit obligatoirement se trouver dans une **instruction conditionnelle** afin d'éviter une **récursivité infinie**.

Factorielle

En mathématiques, la **factorielle** d'un entier naturel n, notée $n!$, est le produit des nombres entiers strictement positifs inférieurs ou égaux à n.

$$n! = \prod_{1 \leqslant i \leqslant n} i = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$$

⚠️ Exception

$$0! = 1$$

Factorielle - Version itérative

```
int factorielle(int n) {
    int res = 1, i = 0;
    for (i = 2; i <= n; i++) {
        res *= i;
    }
    return res;
}
```

Factorielle - Version récursive

```
int factorielleRecursive(int n) {
    if (n <= 1) { // instruction conditionnelle d'arrêt
        return 1;
    } // else
    return n * factorielleRecursive(n - 1); // appel récursif
}
```

Factorielle - Version récursive

Version condensée

```
int factorielleRecursive(int n) {
    return (n > 1) ? n * factorielleRecursive(n - 1) : 1;
}
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(4)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1); // 4 * fact(3)
}
```

```
fact(4) = (                fact(4)                )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(4)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1); // 4 * fact(3)
}
```

```
fact(4) = ( 4 * (          fact(3)          ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(3)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1); // 3 * fact(2)
}
```

```
fact(4) = ( 4 * (          fact(3)          ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(3)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1); // 3 * fact(2)
}
```

```
fact(4) = ( 4 * ( 3 * (     fact(2)     ) ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(2)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1); // 2 * fact(1)
}
```

```
fact(4) = ( 4 * ( 3 * (      fact(2)      ) ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(2)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1); // 2 * fact(1)
}
```

```
fact(4) = ( 4 * ( 3 * ( 2 * ( fact(1) ) ) ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(1)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1);
}
```

```
fact(4) = ( 4 * ( 3 * ( 2 * ( fact(1) ) ) ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(1)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1);
}
```

```
fact(4) = ( 4 * ( 3 * ( 2 * ( (  1  ) ) ) ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(1)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1);
}
```

```
fact(4) = ( 4 * ( 3 * ( 2 * (    1    ) ) ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(2)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1);
}
```

```
fact(4) = ( 4 * ( 3 * (         2         ) ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(3)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1);
}
```

```
fact(4) = ( 4 * (                6                ) )
```

Factorielle - Version récursive - Déroulement de 4!

```
int fact(int n) { // fact(4)
    if (n <= 1) {
        return 1;
    }
    return n * fact(n - 1);
}
```

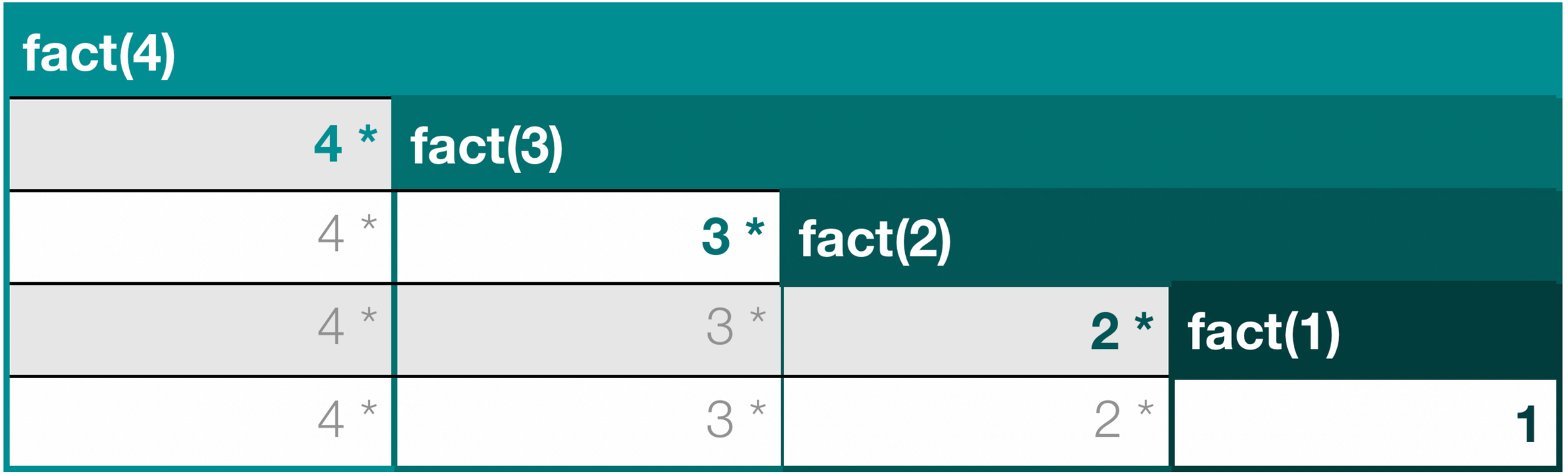
```
fact(4) = (                    24                    )
```

Factorielle - Version récursive - Déroulement de 4!

4! s'exécute donc ainsi :

```
fact(4); /* =>    4 * fact(3)
          * =>    4 * 3 * fact(2)
          * =>    4 * 3 * 2 * fact(1)
          * =>    4 * 3 * 2 * 1
          * =>    4 * 3 * 2
          * =>    4 * 6
          * =>    24
          */
```

Factorielle - Version récursive - Déroulement de 4!



L'appel de `fact(4)` provoque les appels récursifs de `fact(3)`, `fact(2)` et `fact(1)`.

Limites de la récursivité

Grande consommation de mémoire

Lors de l'appel d'une fonction récursive, toutes les variables locales aux fonctions appelées continuent à exister en mémoire tant que les sous-appels récursifs ne sont pas terminés.