Tableaux

24 septembre 2025

Lycée Louis-le-Grand

Comportement indéfini (UB, undefined behavior)

« Anything at all can happen; the Standard imposes no requirements. The program may fail to compile, or it may execute incorrectly (either crashing or silently generating incorrect results), or it may fortuitously do exactly what the programmer intended. »

Exemple de raisons possibles

- terminer un programme par une ligne non vide ;
- tenter d'obtenir la valeur d'une variable non-initialisée ;
- tenter d'accéder, en lecture ou en écriture, à une case d'un tableau avec un index négatif ou bien supérieur ou égal à la taille du tableau ;
- parvenir au bout d'une fonction (autre que `main`) déclarée comme renvoyant une valeur sans rencontrer de **`return`** et utiliser son résultat ;
- effectuer une opération arithmétique illégale (telle qu'une division par 0) ;
- effectuer un calcul (ou une conversion) dont le résultat n'est pas représentable par le type manipulé (par exemple un calcul sur des **`int`** excédant `INT_MAX`) ;

Un exemple

```c
int cherry;

if (cherry == 42) { ... }  // UB !
```

Un exemple


```c
int cherry;

if (banana) { cherry == 42; }

if (cherry == 42) { ... }  // UB si !banana
```



```c
int cherry;

if (banana) {
    cherry == 42;
    ...
}
```

Exemple d'optimisation

```c
int foo(int i) {
    if (i != 0) {
        printf("Hello World!");
    }
    return 100/i;
}
```

Exemple d'optimisation

```c
int foo(int i) {
    // if (i != 0) {
    printf("Hello World~!");
    // }
    return 100/i;
}
```

Exemple d'optimisation qui tourne mal


```c
void create_sandbox(void) {
    // [création d'un "bac à sable"]
    // stuff
    a = a<<32 + b;
    // more stuff
}

void protected_execution(void) {
    create_sandbox();

    execute_dangerous_code();
}
```

Exemple d'optimisation qui tourne mal


```c
void create_sandbox(void) {
    // Nop !
}

void protected_execution(void) {
    // create_sandbox();

    execute_dangerous_code();
}
```

Exemple d'optimisation

```
mango * 2 / 2
```
$\longmapsto$
```
mango
```

```
cherry + 1 > cherry
```
$\longmapsto$
```
true
```

Car si ça déborde, on fait ce que l'on veut !

Pas d'affectations pour un tableau

```c
arr = ...  // Illégal si arr est un tableau
```

Pas d'affectations pour un tableau

```
arr = ...  // Illégal si arr est un tableau
```



```
for (int i=0; i<n; ++i) {
    arr1[i] = arr2[i];      // arr1 ← arr2
}
```

Pas de test d'égalité pour un tableau

```
arr1 == arr2  // Pas ce que l'on souhaite
```

Pas de test d'égalité pour un tableau

```
arr1 == arr2  // Pas ce que l'on souhaite
```



```
arr1[0] == arr2[0] && arr1[1] == arr2[1] && ...
```

Pas de test d'égalité pour un tableau

```
arr1 == arr2  // Pas ce que l'on souhaite
```

```
bool content_equals = true;
for (int i=0; i<n; ++i) {
    content_equals = content_equals
                              && arr1[i] == arr2[i];
}
// content_equals contient true si et seulement si
// les contenus des cases sont égaux deux à deux
```

Pas de test d'égalité pour un tableau

```c
arr1 == arr2  // Pas ce que l'on souhaite
```

```c
bool content_equals = true;
for (int i=0; i<n && content_equals; ++i) {
    content_equals = content_equals
                              && arr1[i] == arr2[i];
}
// content_equals contient true si et seulement si
// les contenus des cases sont égaux deux à deux
```

Pas de test d'égalité pour un tableau

```
arr1 == arr2  // Pas ce que l'on souhaite
```

```
bool content_equals = true;
for (int i=0; i<n && content_equals; ++i) {
    content_equals = arr1[i] == arr2[i];
}
// content_equals contient true si et seulement si
// les contenus des cases sont égaux deux à deux
```

Pas de tableau en argument

Une fonction ne peut pas prendre un tableau en argument !

Travailler sur un tableau

En général :


```c
// Préparation

for (int i=0; i<n; ++i) {
    // [bon endroit pour un invariant]

    // Travail sur arr[i]
}

// Clôture
```

Travailler sur un tableau

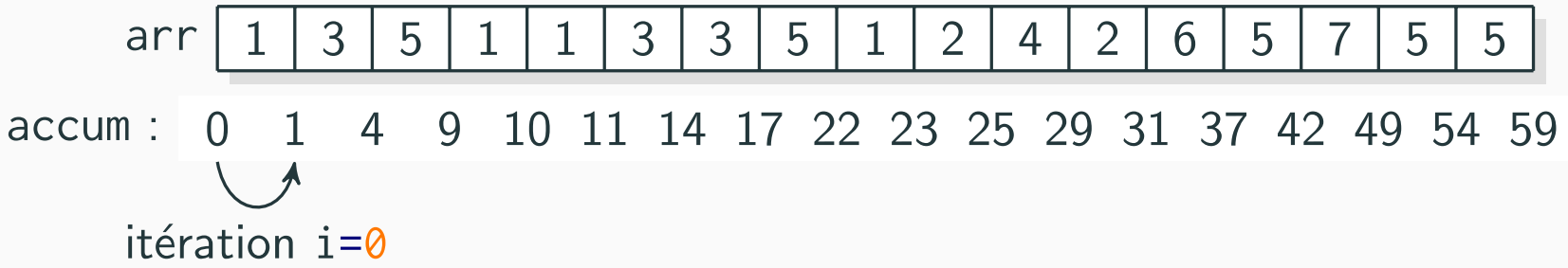
Les exemples qui suivent sont des « briques de bases »

Il faut que le *principe* devienne une évidence !

- résoudre le problème sur papier
- le traduire en C (immédiat)
- documenter le code

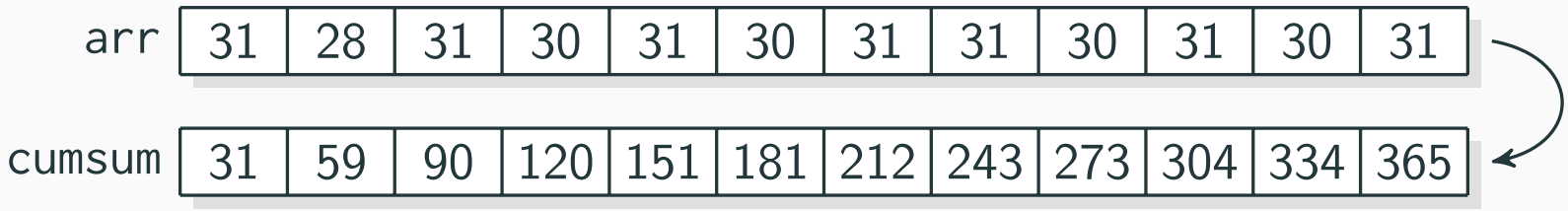
Somme des éléments

Somme des éléments



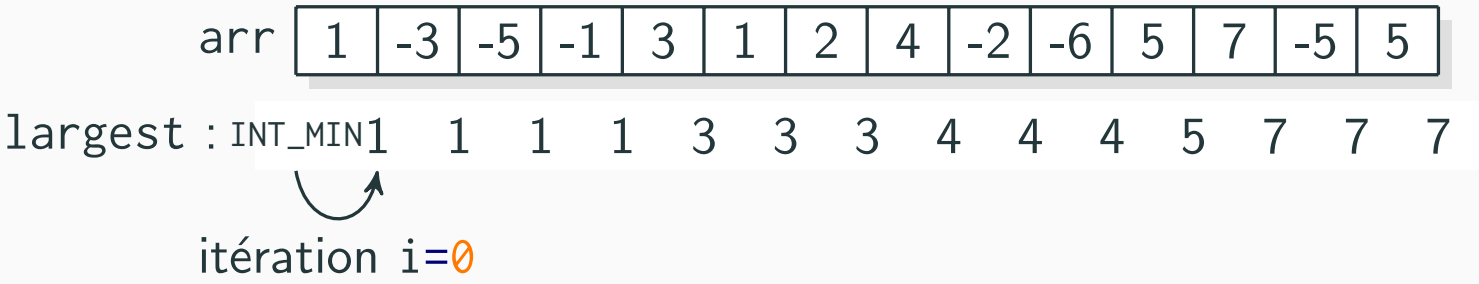
Sommes cumulées

Sommes cumulées

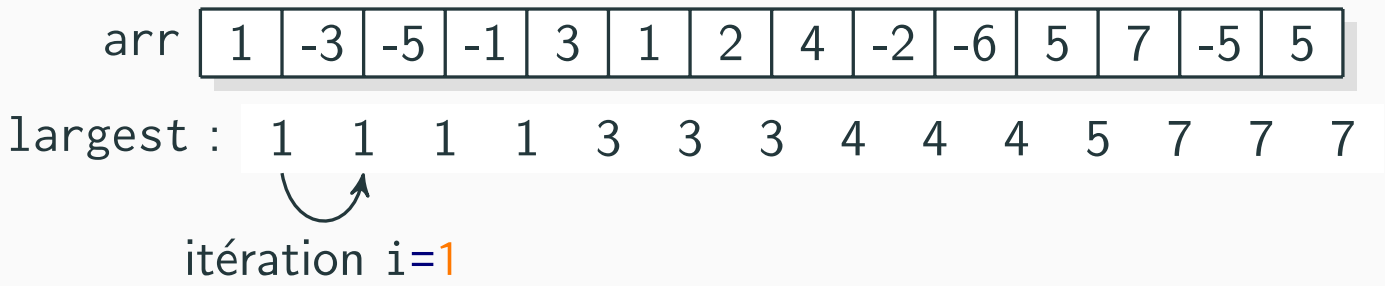


Valeur maximale

Valeur maximale

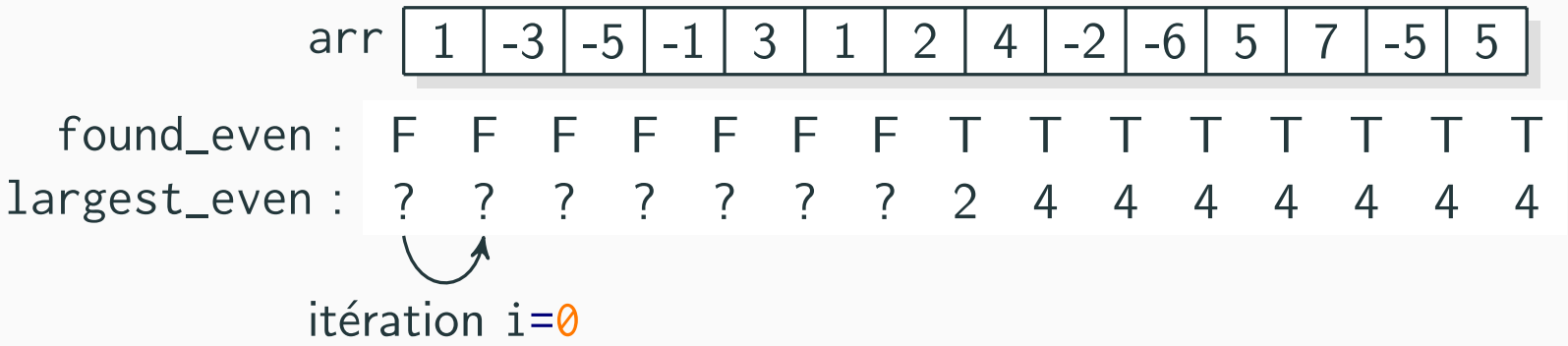


Valeur maximale

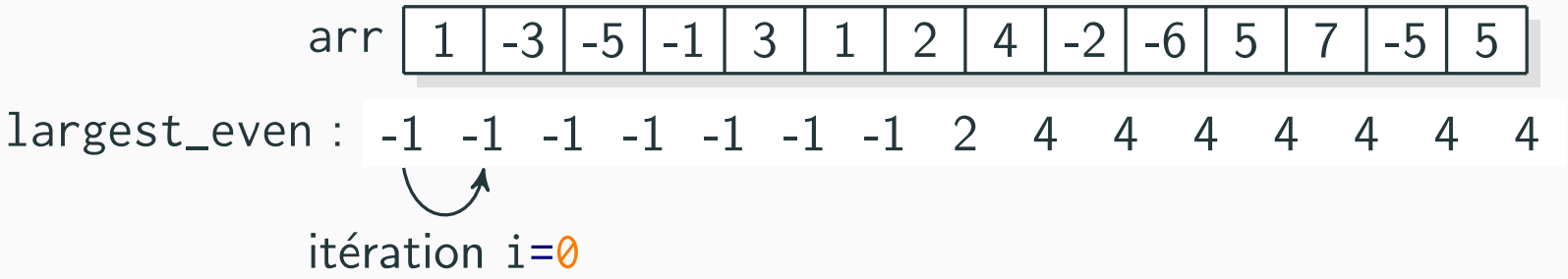


Plus grande valeur paire

Plus grande valeur paire



Plus grande valeur paire



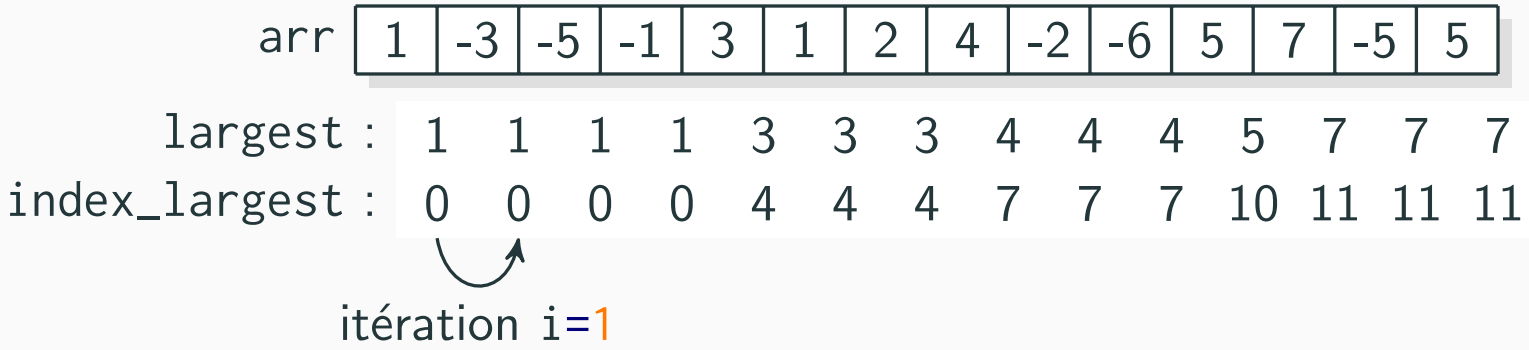
Recherche d'un élément

On cherche un élément `x` pour lequel `looked_for(x)` est `true`

Dans la pratique, peut-être `(x == 42)`, `(x%7 == 0)`, `(x*(x-1) == 1)`...

Recherche du plus grand élément

Recherche du plus grand élément



Vérification de propriétés

Au moins un élément vérifie $\mathscr{P}(x)$

$$\exists x \in \mathrm{E},\ \mathscr{P}(x)$$

Vérification de propriétés

Au moins un élément vérifie $\mathscr{P}(x)$

$$\exists x \in \mathrm{E},\ \mathscr{P}(x)$$

Tous les éléments vérifient $\mathscr{P}(x)$

Vérification de propriétés

Au moins un élément vérifie $\mathscr{P}(x)$

$$\exists x \in \mathrm{E},\ \mathscr{P}(x)$$

Tous les éléments vérifient $\mathscr{P}(x)$

$$\forall x \in \mathrm{E},\ \mathscr{P}(x) \quad \equiv \quad \nexists x \in \mathrm{E} \ \mid \ \text{non}\, \mathscr{P}(x)$$

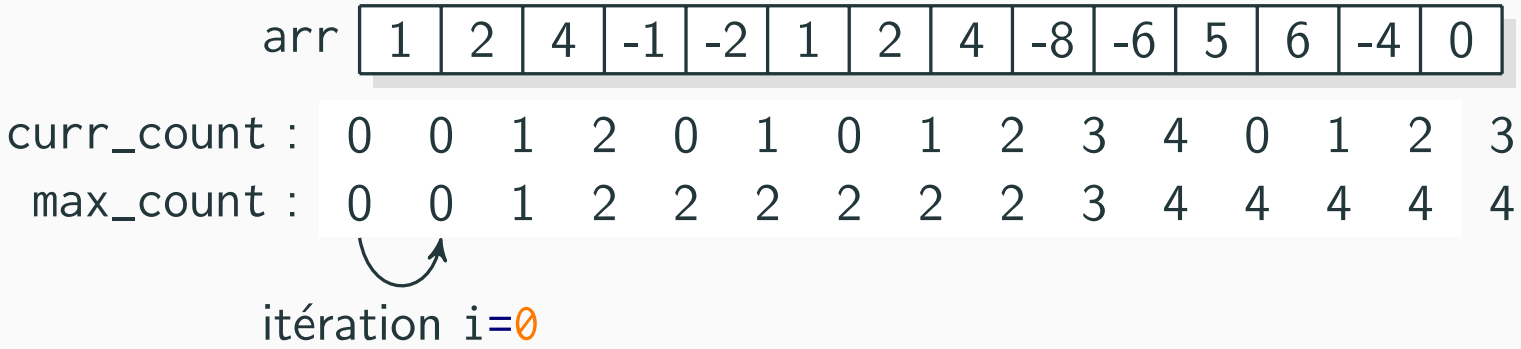
Tableaux ordonnés

On souhaite vérifier

$$\forall i \in [\![1 \,..\, n-1]\!],\ \texttt{tab[i-1]} \leqslant \texttt{tab[i]}$$

Plus longue séquence d'entiers pairs

Plus longue séquence d'entiers pairs



Où commence-t-elle ?

Où commence-t-elle ?

Plus longue séquence croissante

Plus longue séquence croissante