



# EXERCISES — Bubble Sort

---

version #7be580532266ed398481e31366afcc24b1950c2a



**The way is lit. The path is clear.  
We require only the strength to follow it.**

# Copyright

This document is for internal use at EPITA ([website](#)) only.

Copyright © 2022-2023 Assistants <[assistants@tickets.assistants.epita.fr](mailto:assistants@tickets.assistants.epita.fr)>

## The use of this document must abide by the following rules:

- ▷ You downloaded it from the assistants' intranet.\*
- ▷ This document is strictly personal and must **not** be passed onto someone else.
- ▷ Non-compliance with these rules can lead to severe sanctions.

## Contents

1	Goal	3
2	Prototype	3
3	Example	3

---

\*<https://intra.assistants.epita.fr>

## File Tree

```
bubble_sort/  
├─ bubble_sort.c (to submit)  
└─ bubble_sort.h
```

**Authorized headers :** You are only allowed to use the functions defined in the following headers

- err.h
- errno.h
- assert.h
- stddef.h

**Compilation :** Your code must compile with the following flags

- -std=c99 -pedantic -Werror -Wall -Wextra -Wvla

**Main function :** None

## 1 Goal

The bubble sort is a basic sorting algorithm, easy to implement, but quite slow (worst-case and average complexity  $\mathcal{O}(n^2)$ ). It compares each pair of adjacent items and swaps them if they are in the wrong order.

## 2 Prototype

```
void bubble_sort(int array[], size_t size);
```

## 3 Example

```
[6, 1, 8, 5, 4] -> [1, 6, 8, 5, 4] 6 > 1 Swap them  
[1, 6, 8, 5, 4] -> [1, 6, 8, 5, 4] 6 < 8 No need to swap  
[1, 6, 8, 5, 4] -> [1, 6, 5, 8, 4] 8 > 5 Swap them  
[1, 6, 5, 8, 4] -> [1, 6, 5, 4, 8] 8 > 4 Swap them  
...           Start again until the array is fully sorted
```

Your bubble sort will take two arguments: the array to sort, and its size.

*The way is lit. The path is clear. We require only the strength to follow it.*