Lecture 12

Arrays

CSE115: Computing Concepts

Introduction to Array

- In C, a group of items of the same type can be set up using Array
- An array is a group of consecutive memory locations related by the fact that they all have the same name and the same type.
- The compiler must reserve storage (space) for each element/item of a declared array.
- The size of an array is static (fixed) throughout program execution.
- To refer to a particular location or element in the array, we specify the name of the array (index or subscript) and the position number of the particular element in the array.

Array Declaration

- Array declaration is made by specifying the **data type**, it's **name** and the **number of space (size)** so that the computer may reserve the appropriate amount of memory.
- General syntax:

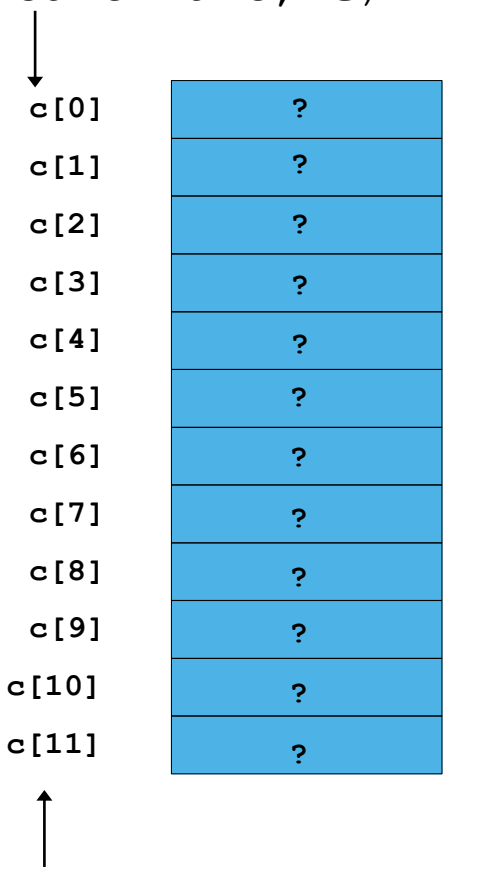
  data_type array_name[size];

- Examples:
  - `int my_array[100];`
  - `char name[20];`
  - `double bigval[5*200];`
  - `int a[27], b[10], c[76];`

Accessing Array Elements

- Declare
  - `int c[12];`
- To refer to an element, specify
  - Array name
  - Position number
- Format:

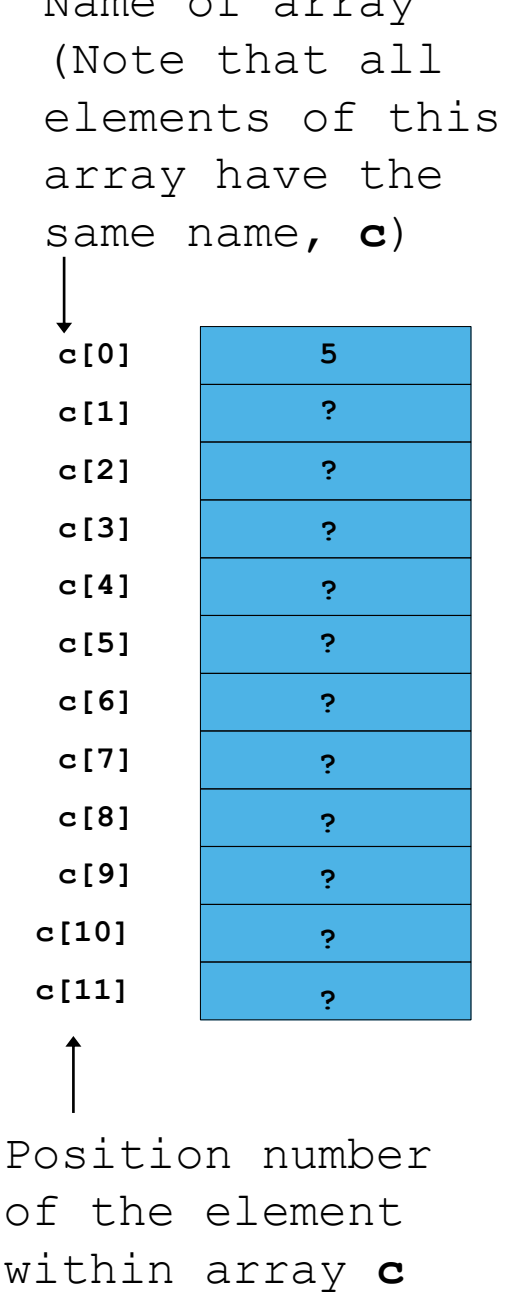
  arrayname **[** *position number* **]**
  - `c[ 0 ], c[ 1 ]...c[ 11 ]`



Accessing Array Elements

- Declare
  - `int c[12];`
- To refer to an element, specify
  - Array name
  - Position number
- Format:

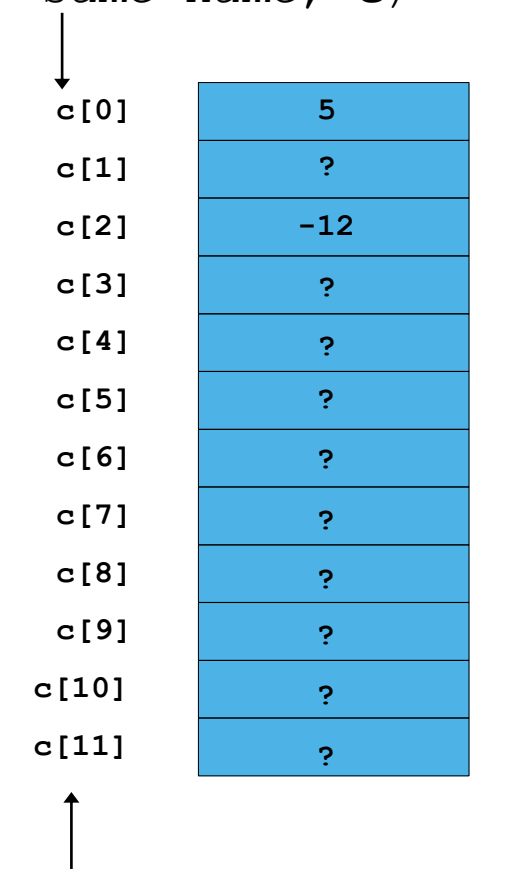
  arrayname **[** *position number* **]**
  - `c[ 0 ], c[ 1 ]...c[ 11 ]`
- Example
  - `c[0] = 5;`



Accessing Array Elements

- Declare
  - `int c[12];`
- To refer to an element, specify
  - Array name
  - Position number
- Format:

  arrayname **[** *position number* **]**
  - `c[ 0 ], c[ 1 ]...c[ 11 ]`
- Example
  - `c[2] = -12;`



Array Initialization

- During compilation
  - `int n[ 5 ] = { 1, 2, 3, 4, 5 };`
    - If not enough initializers, rightmost elements become **0**
    - If too many a syntax error is produced syntax error
  - `int n[ 5 ] = { 0 };`
    - All elements 0
  - `int n[ ] = { 1, 2, 3, 4, 5 };`
    - If size omitted, initializers determine it
    - 5 initializers, therefore 5 element array

Array Initialization

- During execution:
  - Using loop to initialize all elements to zero

```
int arr[3], index;
for (index = 0; index < 3; index++)
      arr[index] = 0;
```

  - Using loop and asking the user to specify the value for each element.

```
int arr[3], index;
for (index = 0; index < 3; index++)
{
      printf ("arr[%d]:",index);
      scanf("%d",&arr[index]);
}
```

Problem

- Take N numbers from the user as input and output them in ascending (or descending) order.

For example, if user input is

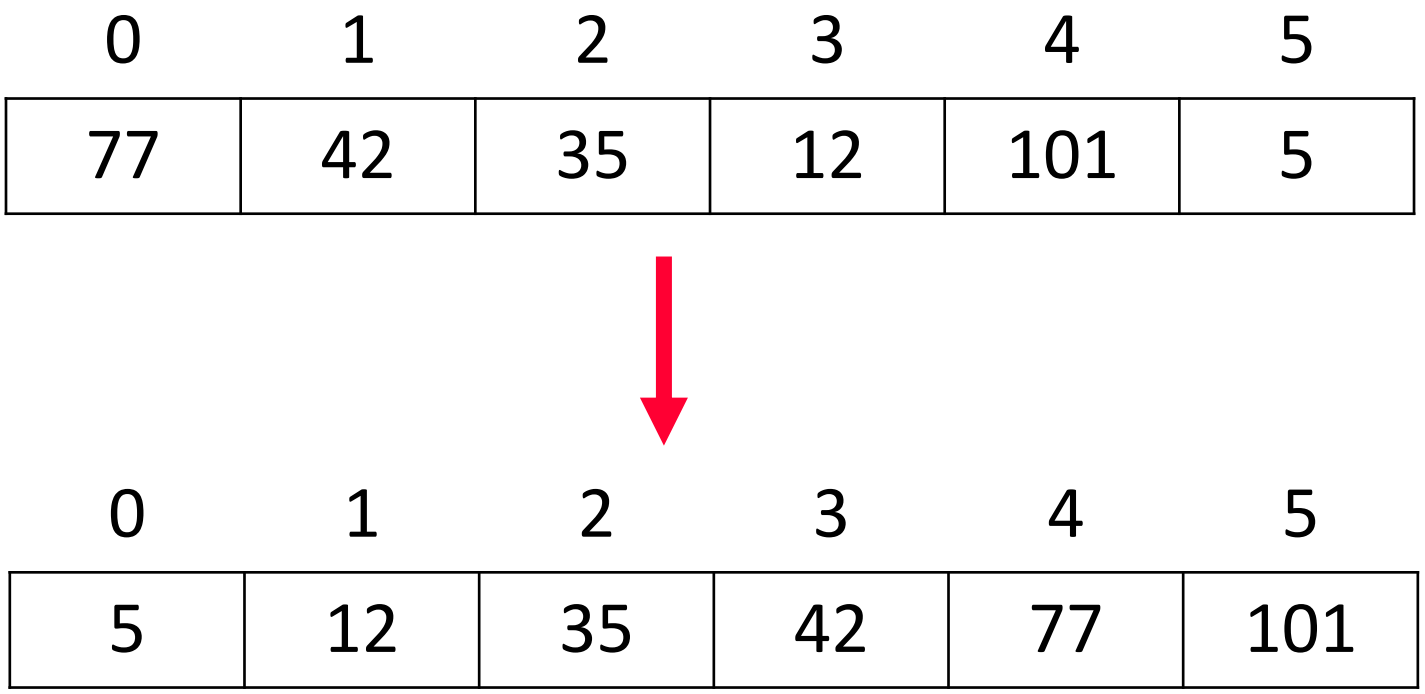
25 14 78 5 90 67 32 85

then your program's output is

5 14 25 32 67 78 85 90

Sorting

- **Sorting takes an unordered collection and makes it an ordered one.**



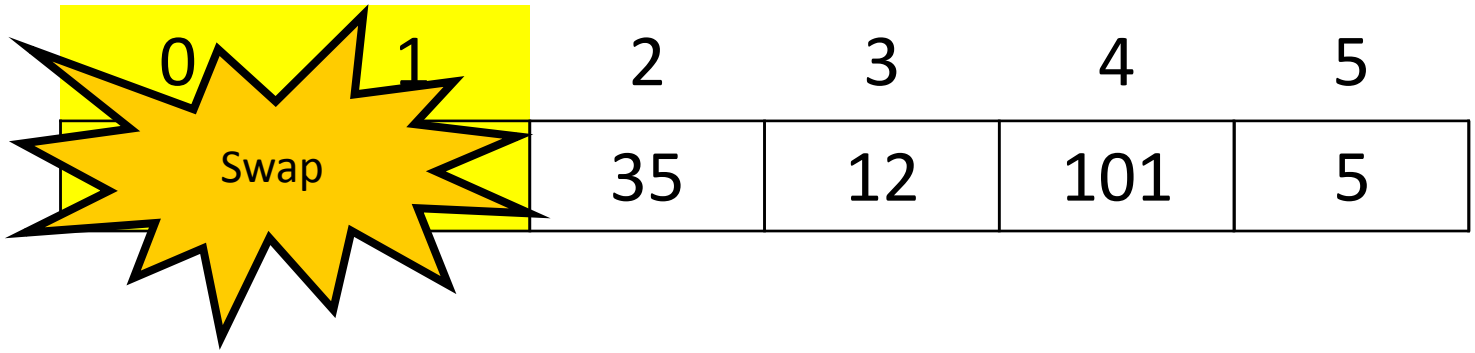
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **"Bubble" the largest value to the end using pair-wise comparisons and swapping**

0	1	2	3	4	5
77	42	35	12	101	5

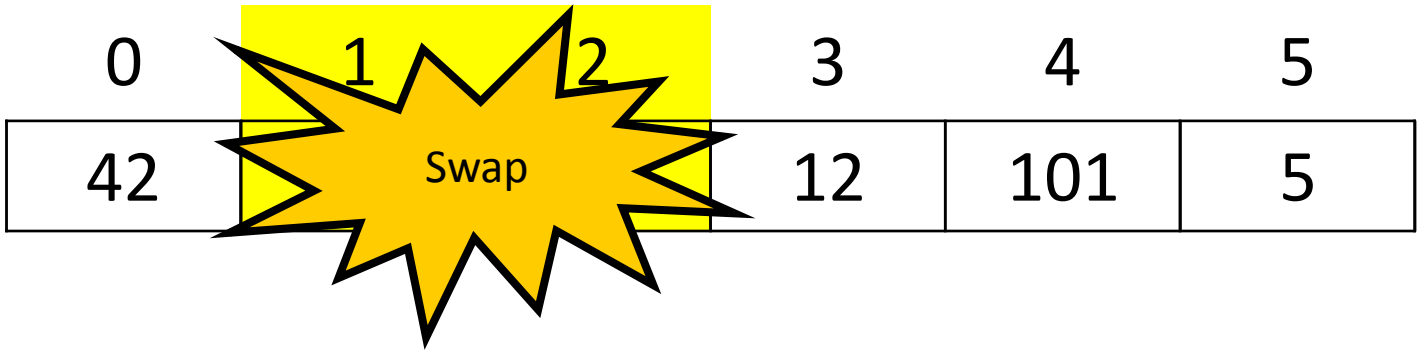
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



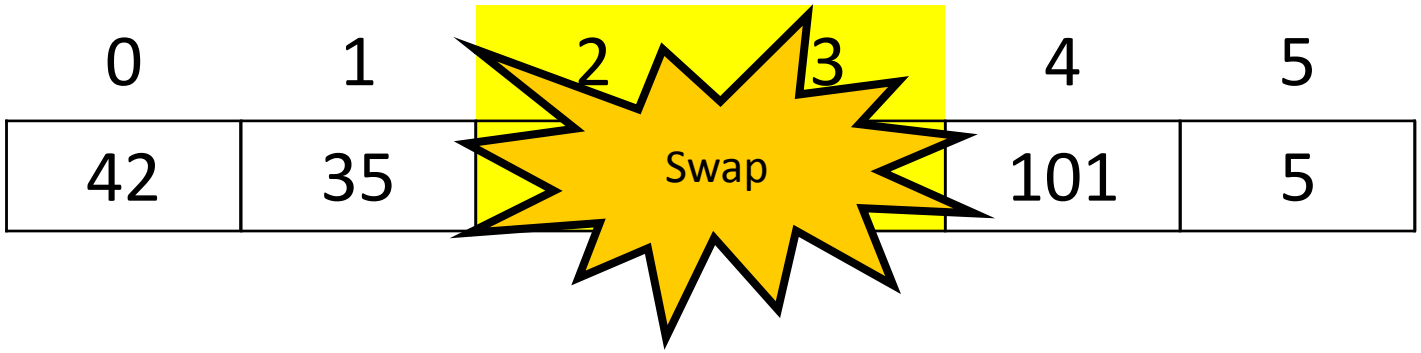
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



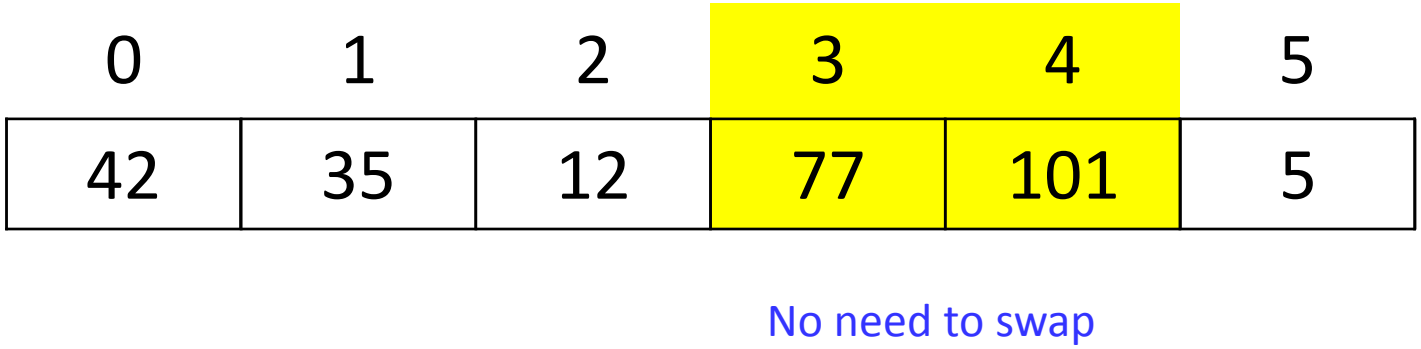
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



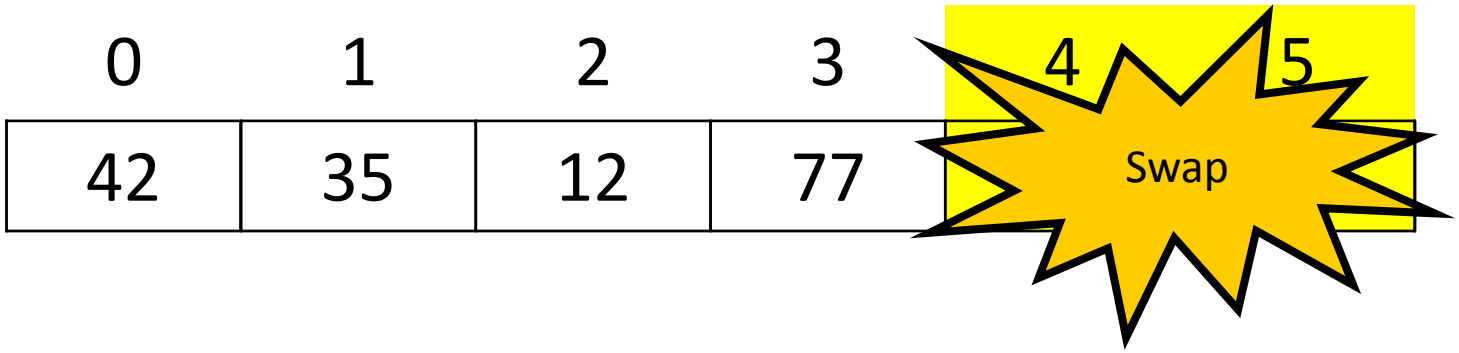
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



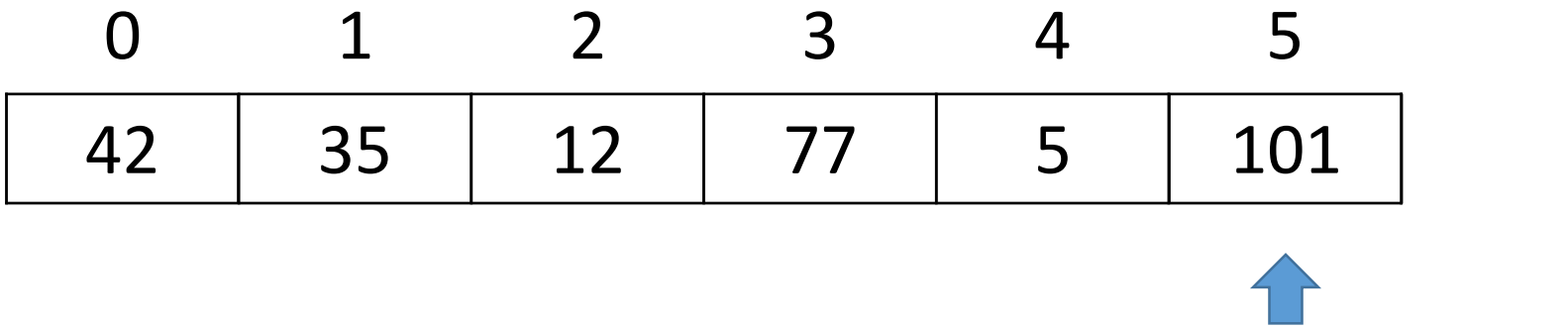
"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



"Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**

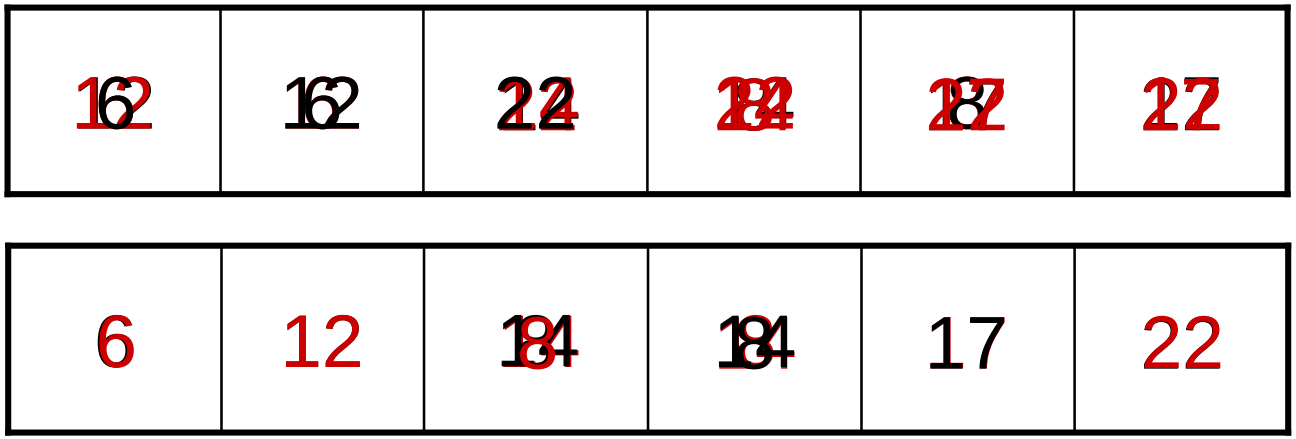


The “Bubble Up” Algorithm

```
index = 0
last_index = n - 1

While(index < last_index)
   if(A[index] > A[index + 1]) then
      Swap(A[index], A[index + 1])
   endif
   index = index + 1
endwhile
```

Bubble Sort

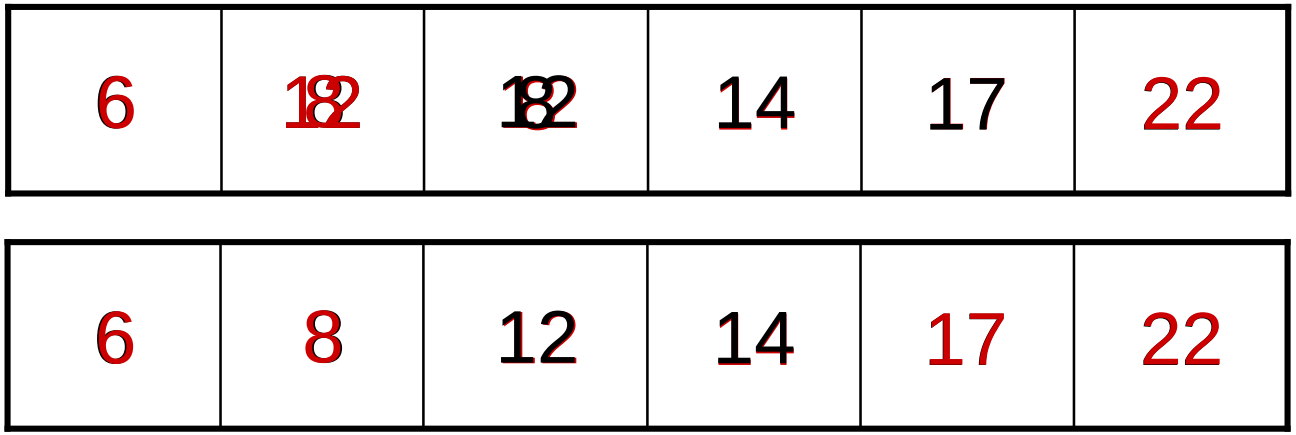


Given n numbers to sort:

Repeat the following n-1 times:

- For each pair of adjacent numbers:
  - If the number on the left is greater than the number on the right, swap them.

Bubble Sort



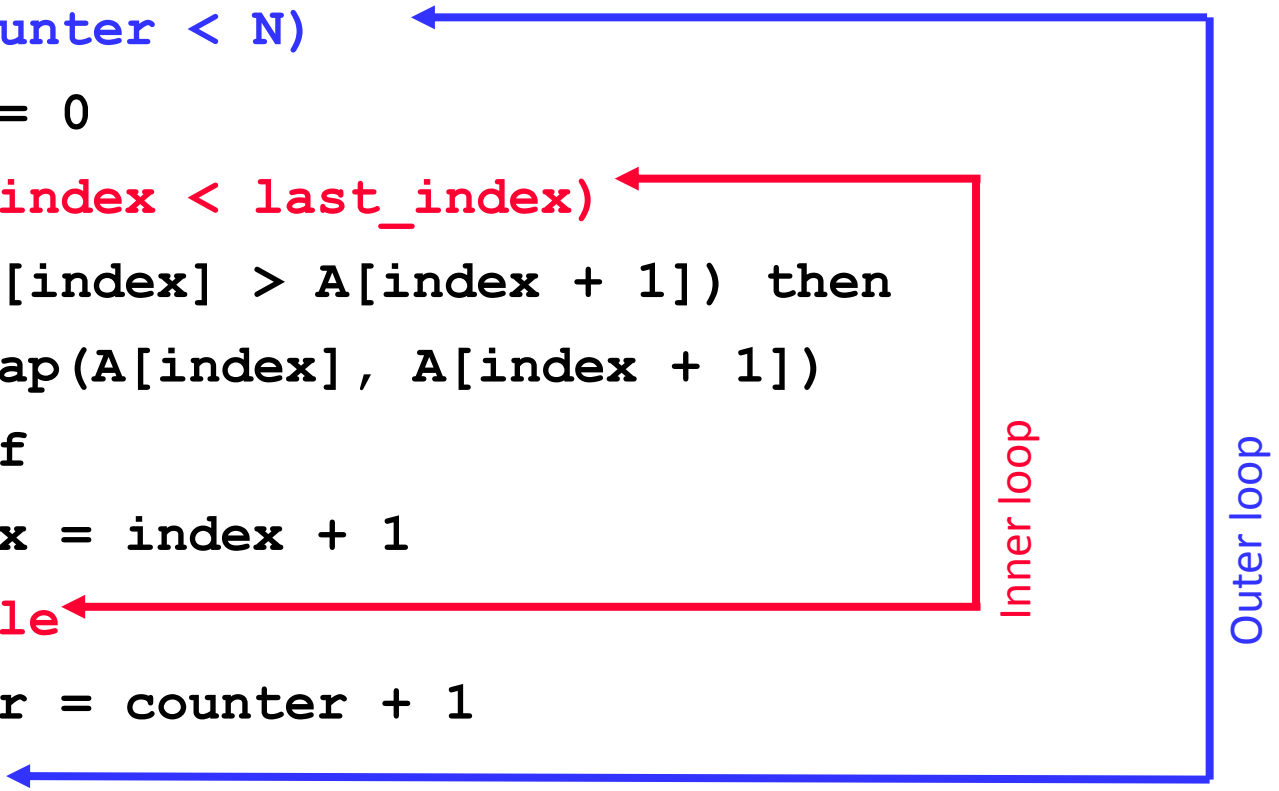
Given n numbers to sort:

Repeat the following n-1 times:

- For each pair of adjacent numbers:
  - If the number on the left is greater than the number on the right, swap them.

Bubble Sort

```
last_index = N - 1
counter = 1
while(counter < N)
  index = 0
  while(index < last_index)
    if(A[index] > A[index + 1]) then
      Swap(A[index], A[index + 1])
    endif
    index = index + 1
  endwhile
  counter = counter + 1
endwhile
```



Home-work

- Implement the Bubble Sort algorithm.