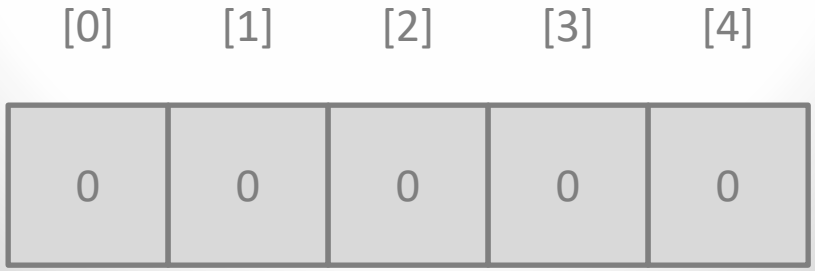
Arrays y Algoritmos

Asignación de valores

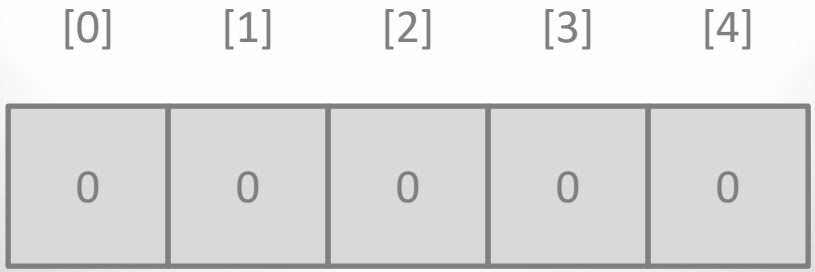
```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

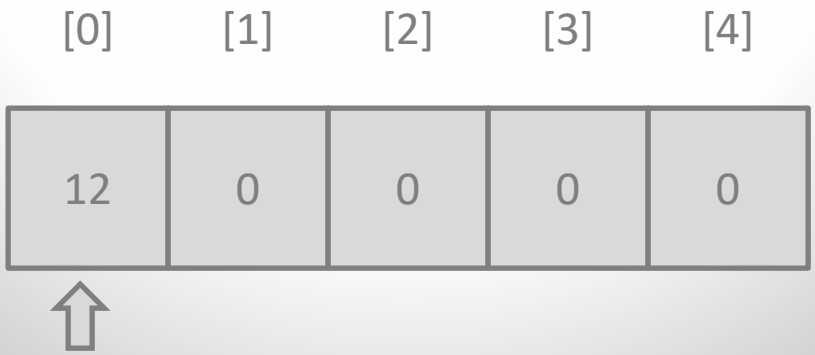
i=0



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

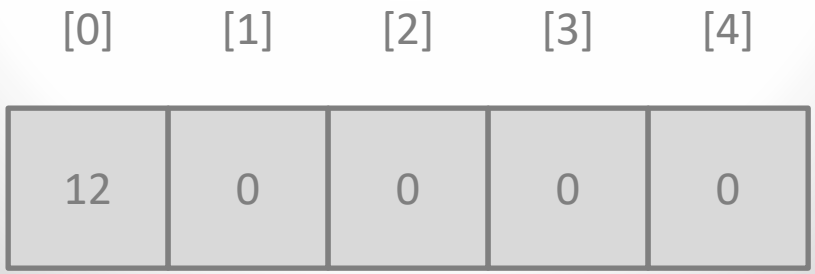
i=0



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

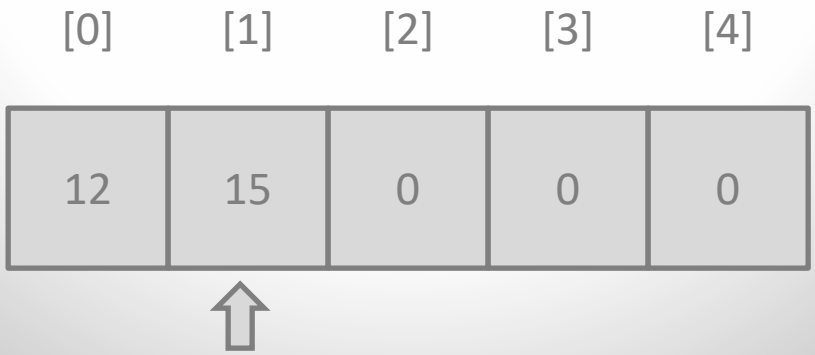
i=1



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

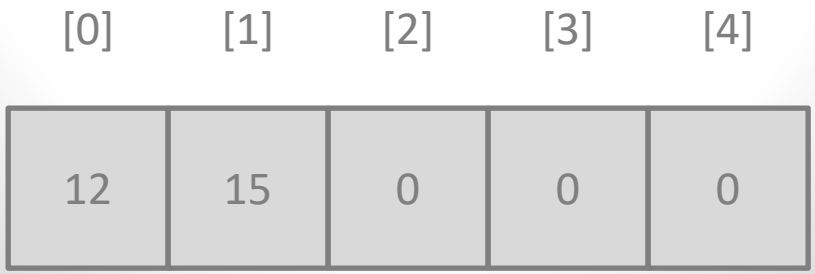
i=1



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

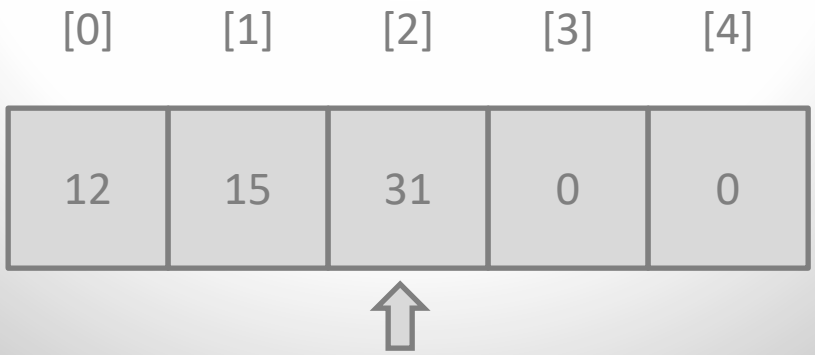
i=2



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

i=2



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

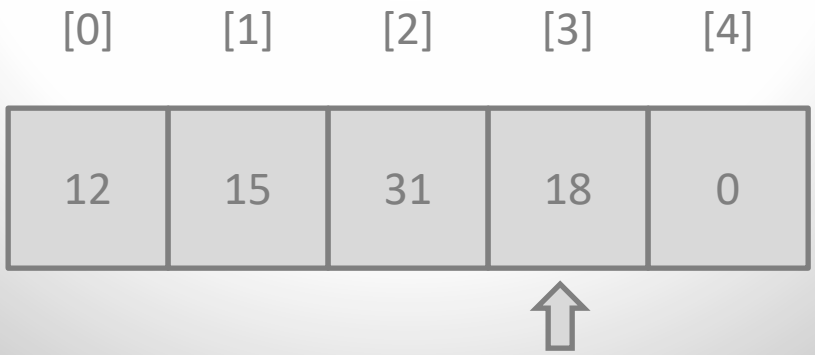
i=3

[0]	[1]	[2]	[3]	[4]
12	15	31	0	0

Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

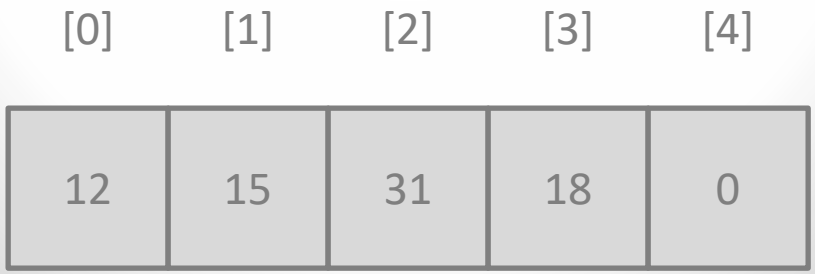
i=3



Asignación de valores

```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

i=4



Asignación de valores

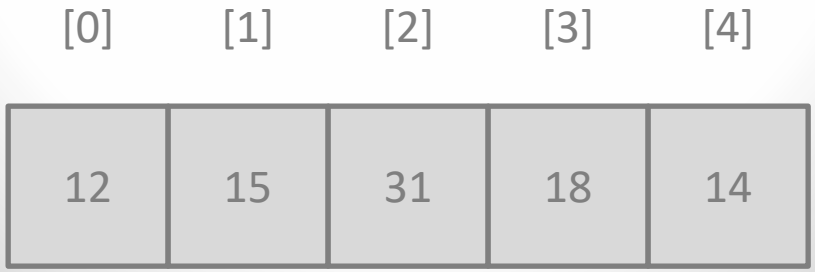
```
const short SIZE=5;
short ages[SIZE]={0};
short sum=0;
short average=0;

for(short i=0; i<SIZE; i++)
{
    cout << "Person " << i+1 << ", enter your age: ";
    cin >> ages[i];
}
```

i=4

Acumulación

```
...
for(short i=0; i<SIZE; i++)
    sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

i=0, sum=0



Acumulación

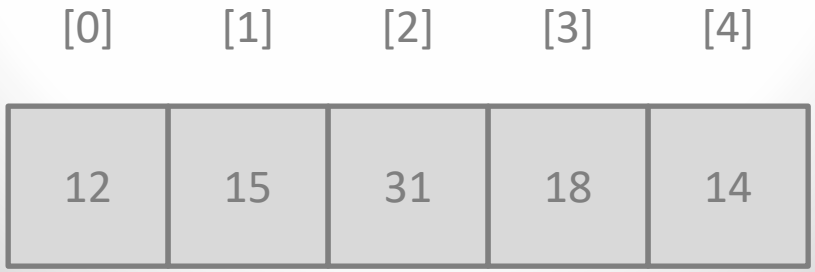
```
...
for(short i=0; i<SIZE; i++)
  ⇨ sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

i=0, sum=12

Acumulación

```
...
for(short i=0; i<SIZE; i++)
    sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

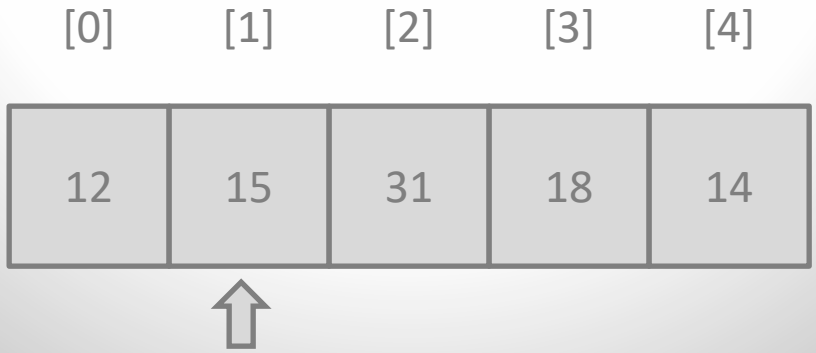
i=1, sum=12



Acumulación

```
...
for(short i=0; i<SIZE; i++)
  ⇨ sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

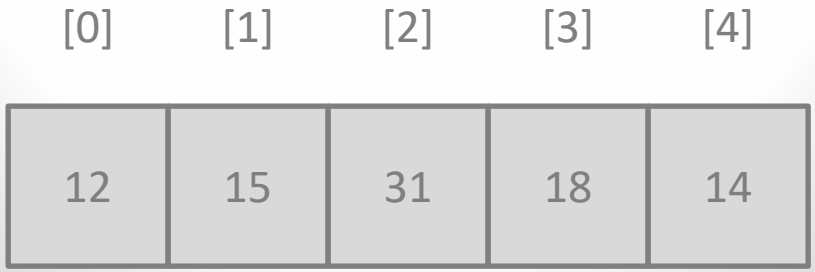
i=1, sum=27



Acumulación

```
...
for(short i=0; i<SIZE; i++)
    sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

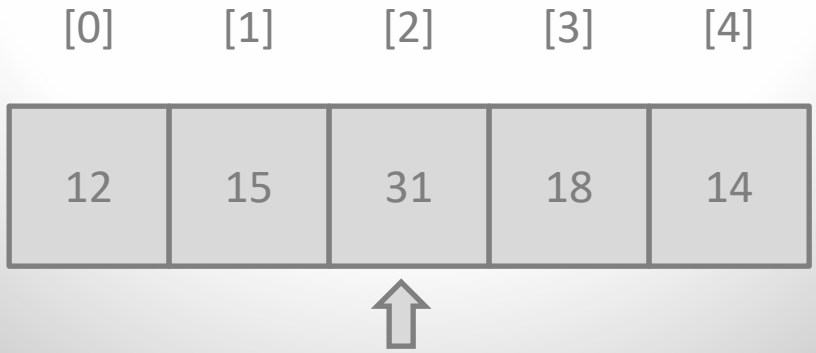
i=2, sum=27



Acumulación

```
...
for(short i=0; i<SIZE; i++)
  ⇨sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

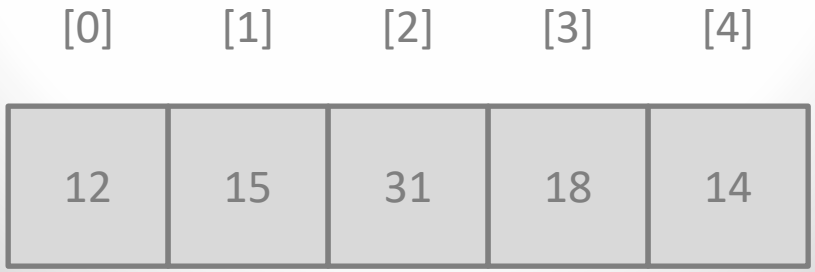
i=2, sum=58



Acumulación

```
...
for(short i=0; i<SIZE; i++)
    sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

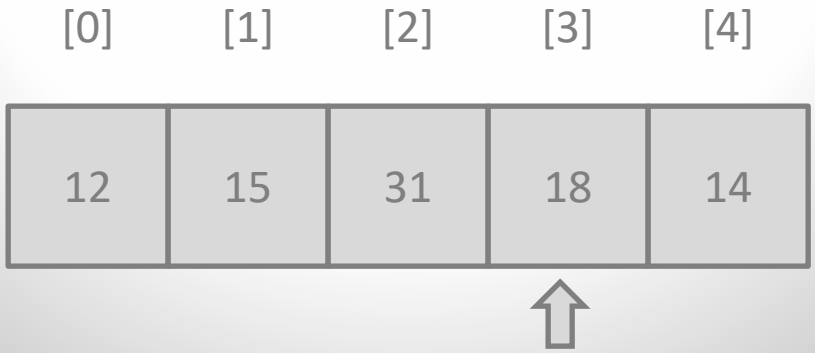
i=3, sum=58



Acumulación

```
...
for(short i=0; i<SIZE; i++)
  ⇨sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

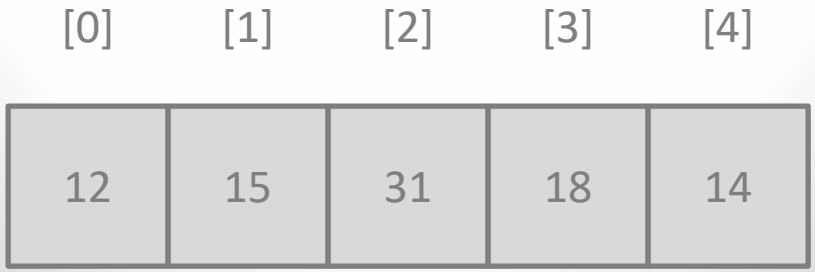
i=3, sum=76



Acumulación

```
...
for(short i=0; i<SIZE; i++)
    sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

i=4, sum=76



Acumulación

```
...
for(short i=0; i<SIZE; i++)
  ⇨sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

i=4, sum=90

Acumulación

```
...
for(short i=0; i<SIZE; i++)
    sum+=ages[i];

average=sum/SIZE;
cout << "The average age is " << average;
```

i=4, sum=90

average=18

Búsqueda valor especial

```
short max=ages[0];
```
max=12

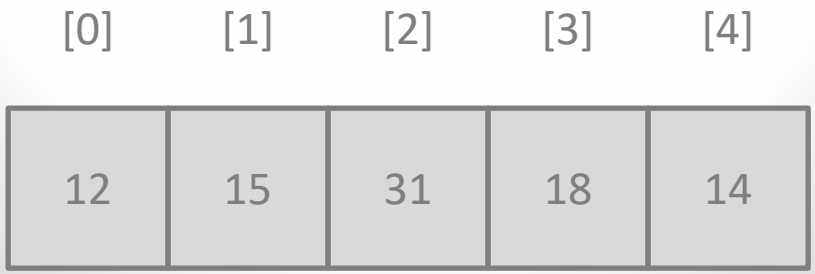
```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
        max=ages[i];

cout << "the max age in the array is " << max;
```

Búsqueda valor especial

```
short max=ages[0];
```
max=12

```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
        max=ages[i];
```
i=1

```
cout << "the max age in the array is " << max;
```

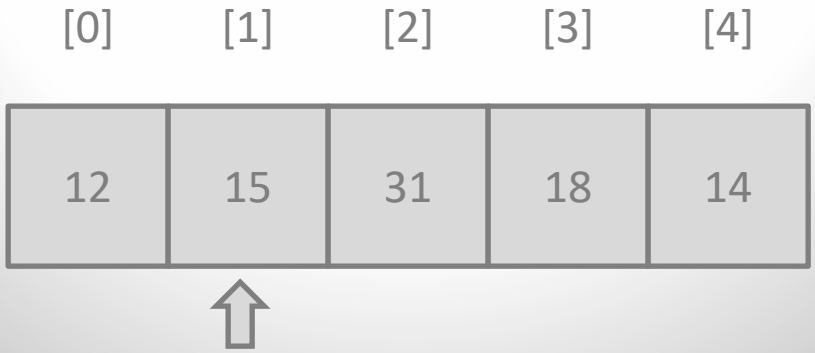


Búsqueda valor especial

```
short max=ages[0];
```
max=12

```
for(short i=1; i<SIZE; i++)
  ⇨if(ages[i]>max)
        max=ages[i];
```
i=1

```
cout << "the max age in the array is " << max;
```

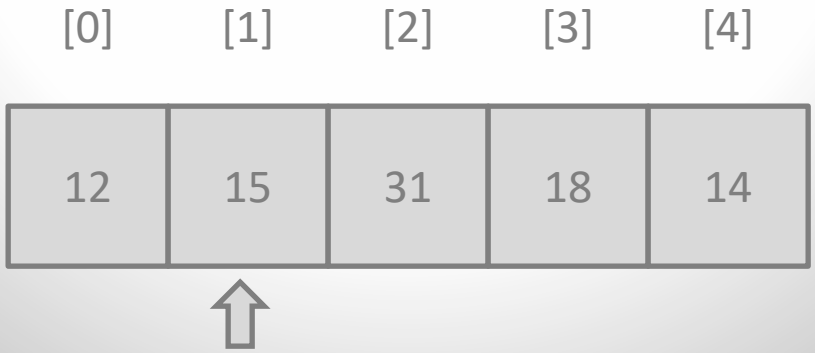


Búsqueda valor especial

```
short max=ages[0];
```
max=15

```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
      ⇨ max=ages[i];
```
i=1

```
cout << "the max age in the array is " << max;
```

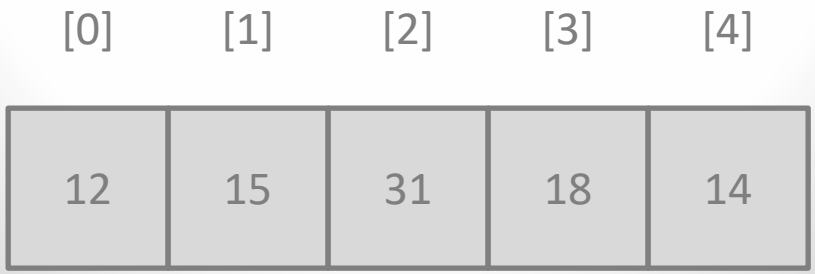


Búsqueda valor especial

```
short max=ages[0];
```
max=15

```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
        max=ages[i];
```
i=2

```
cout << "the max age in the array is " << max;
```

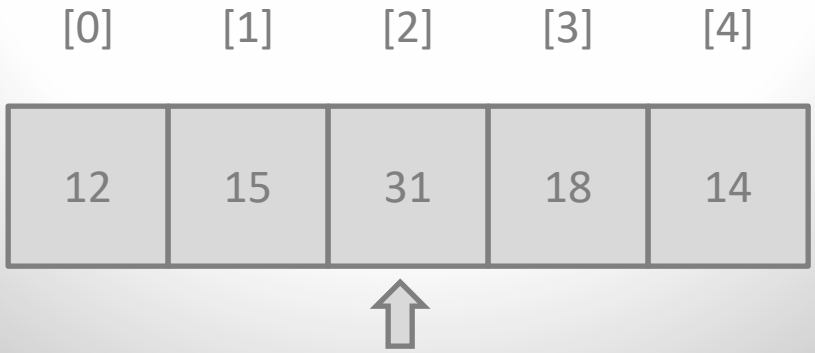


Búsqueda valor especial

```
short max=ages[0];
```
max=15

```
for(short i=1; i<SIZE; i++)
  ⇨if(ages[i]>max)
        max=ages[i];
```
i=2

```
cout << "the max age in the array is " << max;
```

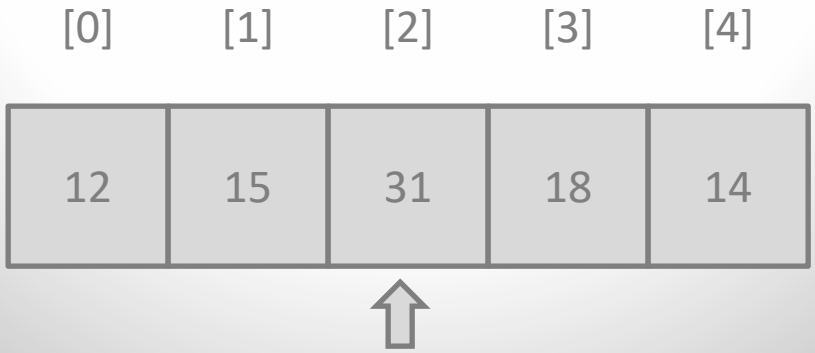


Búsqueda valor especial

```
short max=ages[0];
```
max=31

```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
      ⇨max=ages[i];
```
i=2

```
cout << "the max age in the array is " << max;
```

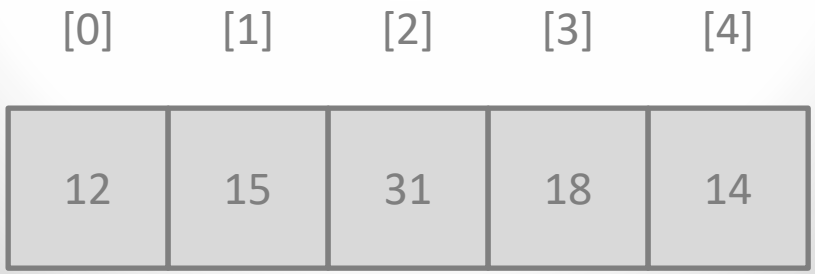


Búsqueda valor especial

```
short max=ages[0];
```
max=31

```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
        max=ages[i];
```
i=3

```
cout << "the max age in the array is " << max;
```

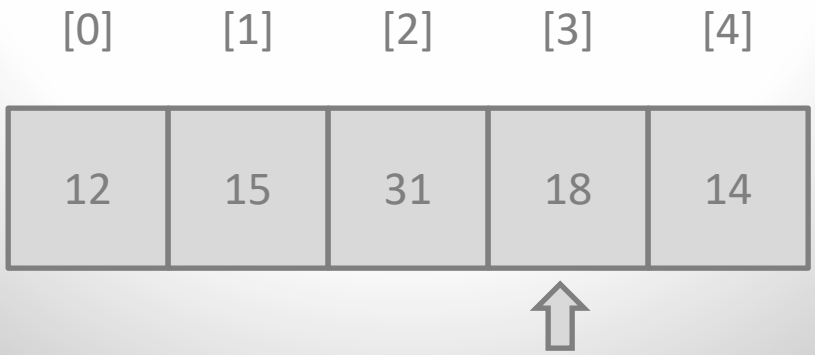


Búsqueda valor especial

```
short max=ages[0];
```
max=31

```
for(short i=1; i<SIZE; i++)
  ⇨if(ages[i]>max)
        max=ages[i];
```
i=3

```
cout << "the max age in the array is " << max;
```



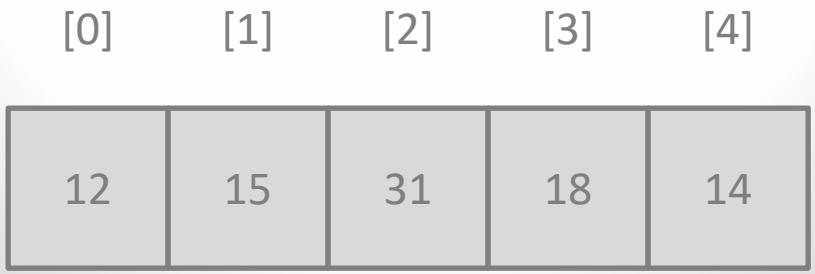
Búsqueda valor especial

```
short max=ages[0];
```

max=31

```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
        max=ages[i];

cout << "the max age in the array is " << max;
```

i=4



Búsqueda valor especial

```
short max=ages[0];
```
max=31

```
for(short i=1; i<SIZE; i++)
  ⇨if(ages[i]>max)
        max=ages[i];
```
i=4

```
cout << "the max age in the array is " << max;
```

Búsqueda valor especial

```
short max=ages[0];
```

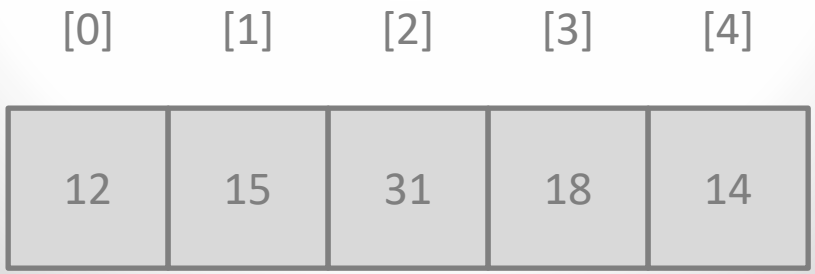
max=31

```
for(short i=1; i<SIZE; i++)
    if(ages[i]>max)
        max=ages[i];

cout << "the max age in the array is " << max;
```

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

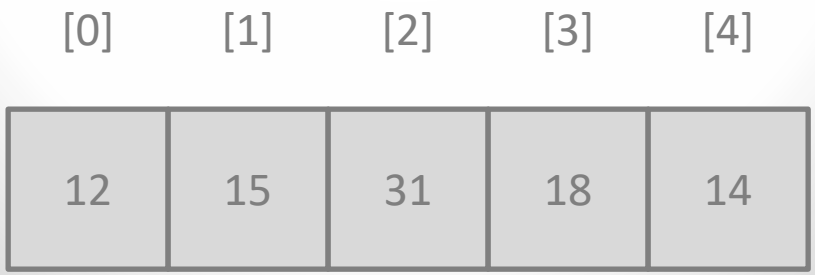


Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=0



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

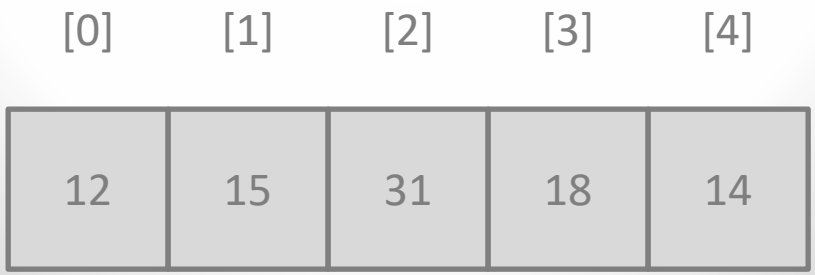
j=0

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=1



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      ⇨if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

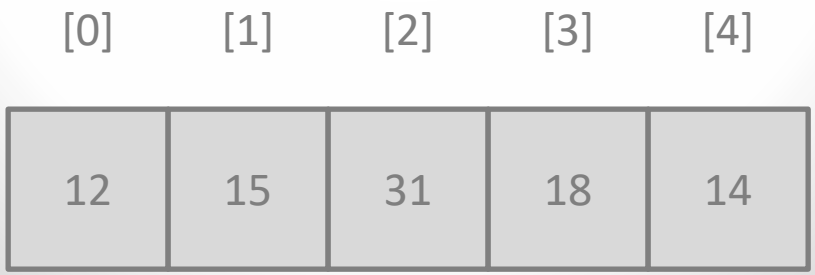
j=1

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=2

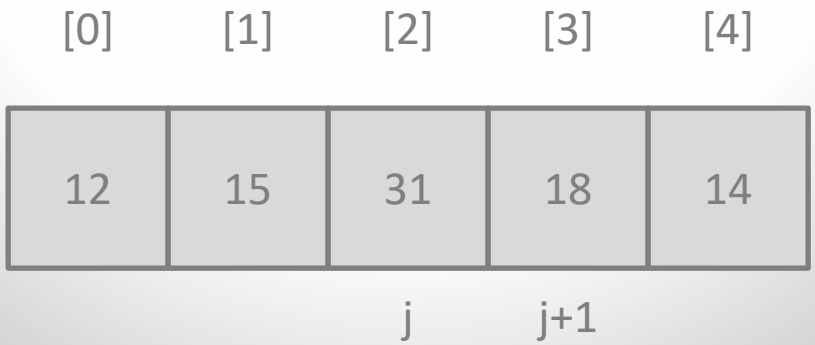


Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      ⇨if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=2



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=2

swap=31

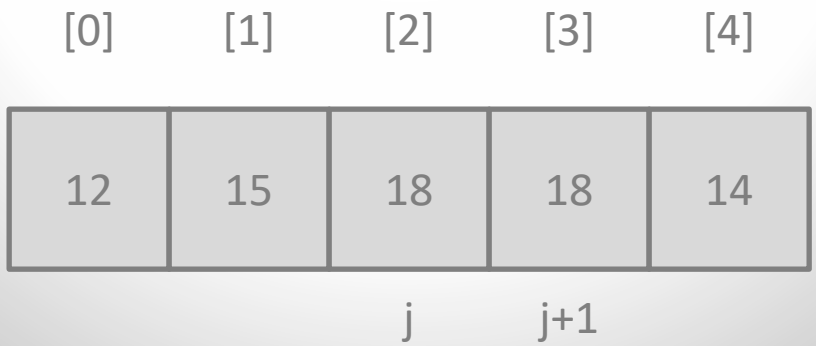
Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=2

swap=31



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=2

swap=31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=3

[0]	[1]	[2]	[3]	[4]
12	15	18	31	14

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=3

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=3

swap=31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=3

swap=31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

j=3

swap=31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=1

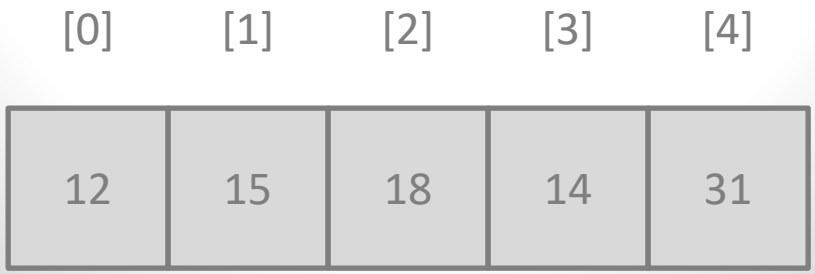
j=4

[0]	[1]	[2]	[3]	[4]
12	15	18	14	31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

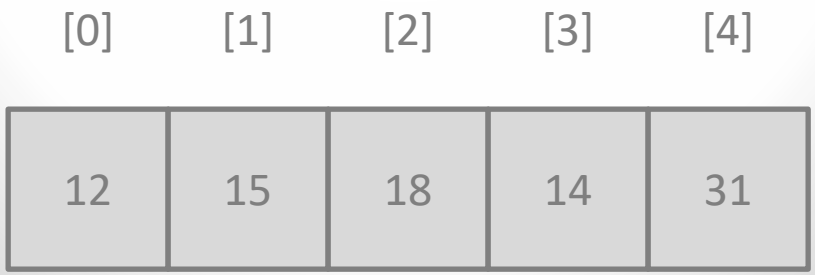


Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=0



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      ⇨if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=0

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=1

[0]	[1]	[2]	[3]	[4]
12	15	18	14	31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      ⇨if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

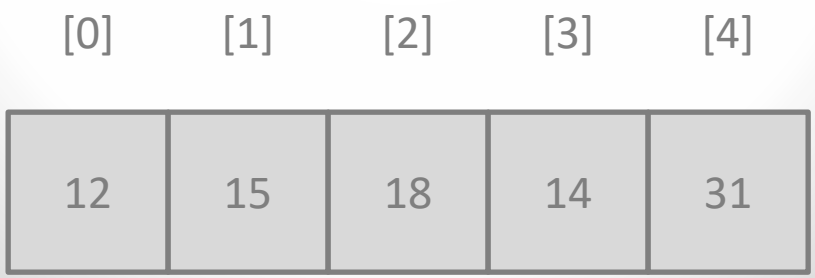
j=1

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=2



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      ⇨if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=2

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=2

swap=18

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=2

swap=18

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

j=2

swap=18

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=2

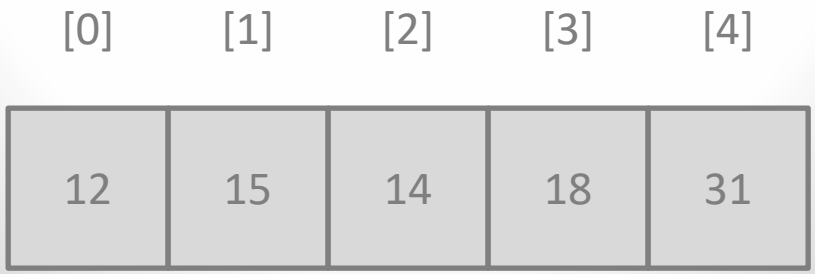
j=3

[0]	[1]	[2]	[3]	[4]
12	15	14	18	31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

j=0

[0]	[1]	[2]	[3]	[4]
12	15	14	18	31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      ⇨if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

j=0

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

j=1

[0]	[1]	[2]	[3]	[4]
12	15	14	18	31

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      ⇨if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

j=1

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

j=1

swap=15

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

j=1

swap=15

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

j=1

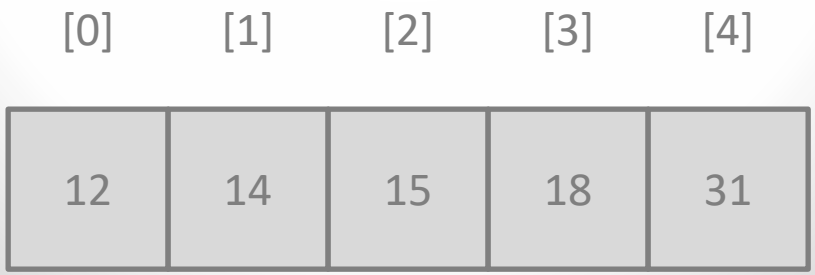
swap=15

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=3

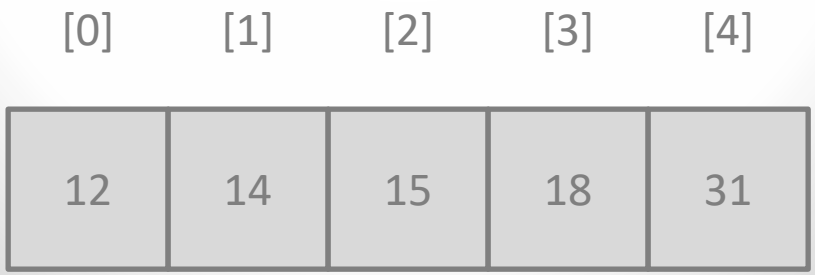
j=2



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=4



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
      if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=4

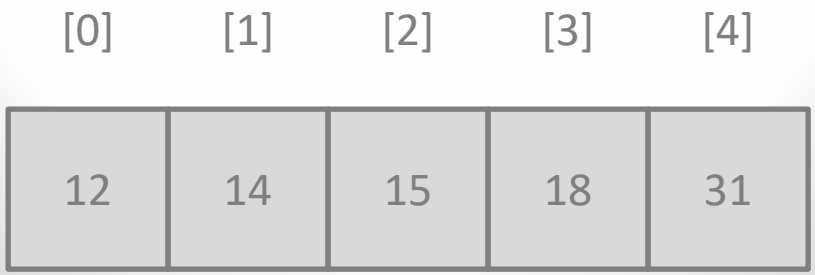
j=0

Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=4

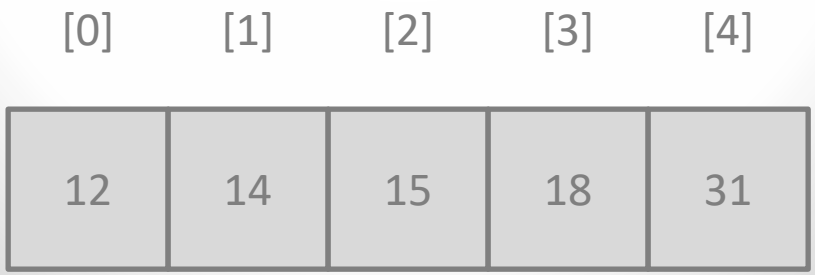
j=1



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

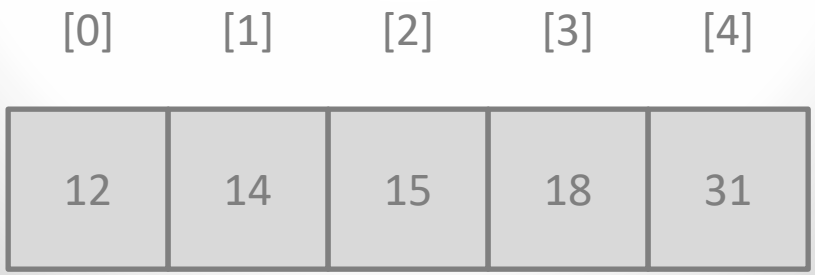
i=5



Ordenación *bubblesort*

```
for(short i=1; i<SIZE; i++)
    for(short j=0; j<SIZE-i; j++)
        if(ages[j] > ages[j+1])
        {
            int swap = ages[j];
            ages[j] = ages[j+1];
            ages[j+1] = swap;
        }
```

i=5



Complejidad

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Arrays como parámetros de funciones

```
void print_array(const float an_array[], const int size)
{
    for(int i = 0; i < size; i++)
        cout << an_array[i] << " ";
    cout << endl;
    return;
}

int main()
{
    const int SIZE = 100;
    float my_array[SIZE];
    ...
    print_array(my_array, SIZE);
}
```

Cómo indicar que pasamos un array?

```
void print_array(const float an_array[], const int size);

int main()
{
    const int SIZE = 100;
    float my_array[SIZE];

    ...
    print_array(my_array[], SIZE);
```

Cómo indicar que pasamos un array?

```
void print_array(const float an_array[], const int size);

int main()
{
    const int SIZE = 100;
    float my_array[SIZE];

    ...
    print_array(my_array[], SIZE);
```

Syntax error !!!

Ejemplo 1

```
void sort_array(float an_array[], const int size)
{
    for(short i = 1; i < size; i++)
        for(short j = 0; j < size-i; j++)
            if(an_array[j] > an_array[j+1])
                swap(an_array[j], an_array[j+1]);
}

int main()
{
    const int SIZE = 100;
    float my_array[SIZE];
    ...
    sort_array(my_array, SIZE);
    ...
```

Ejemplo 2

```
void shift_right(int an_array[], const int size)
{
    for(int i = size - 1; i > 0; i--)
        an_array[i] = an_array[i-1];
    return;
}

int main()
{
    const int SIZE = 100;
    float my_array[SIZE];
    ...
    shift_right(my_array, SIZE);
    ...
```

BEFORE

[0]	[1]	[2]	[3]	[4]
12	14	15	18	31

Ejemplo 2

```
void shift_right(int an_array[], const int size)
{
    for(int i = size - 1; i > 0; i--)
        an_array[i] = an_array[i-1];
    return;
}

int main()
{
    const int SIZE = 100;
    float my_array[SIZE];
    ...
    shift_right(my_array, SIZE);
    ...
```

BEFORE

[0]	[1]	[2]	[3]	[4]
12	14	15	18	31

AFTER

[0]	[1]	[2]	[3]	[4]
12	12	14	15	18

Búsqueda lineal

```
bool is_found (const char an_array[], const int size, const
   char target)
{
    bool found = false;
    int i = 0;
    while( (i < size) && (found == false) )
    {
        if (target == an_array[i])
            found = true;
        i++;
    }
    return found;
}
```

Búsqueda lineal mejorada

```
int is_found (const char an_array[], const int size, const
   char target)
{
    int position_found = -1;
    int i = 0;
    while( (i < size) && (position_found == -1) )
    {
        if (target == an_array[i])
            position_found = i;
        i++;
    }
    return position_found;
}
```

Búsqueda binaria

```
int is_found_binary (const float an_array[], const int size, const
   float target)
{
    int position_found = -1, low = 0, high = size - 1, mid;
    while( (low <= high) && (position_found == -1) )
    {
        mid = (low + high) / 2;
        if (target < an_array[mid])
            high = mid - 1;
        else if (target > an_array[mid])
            low = mid + 1;
        else
            position_found = mid;
    }
    return position_found;
}
```

Complejidad

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)

- Complejidad de búsqueda binaria?