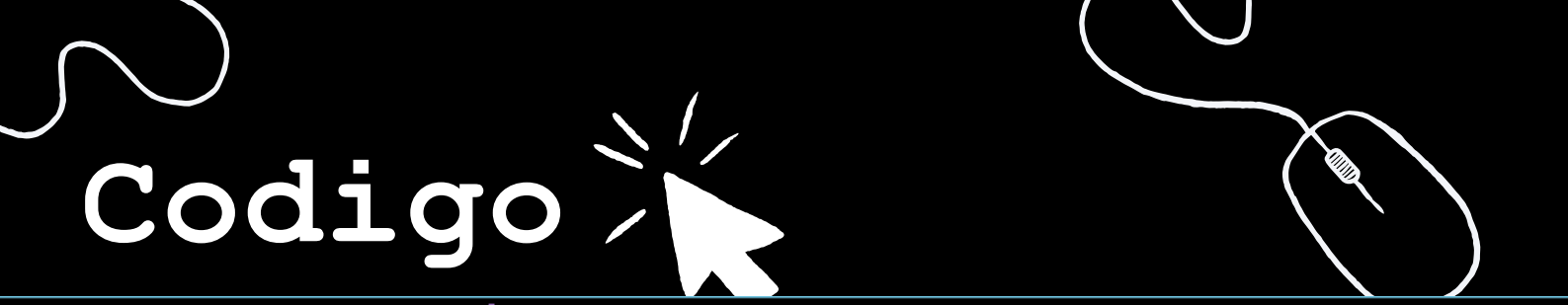
ESTRUCTURA DE DATOS

METODO DE ORDENAMIENTO DE BURBUJA

ENRIQUE HH

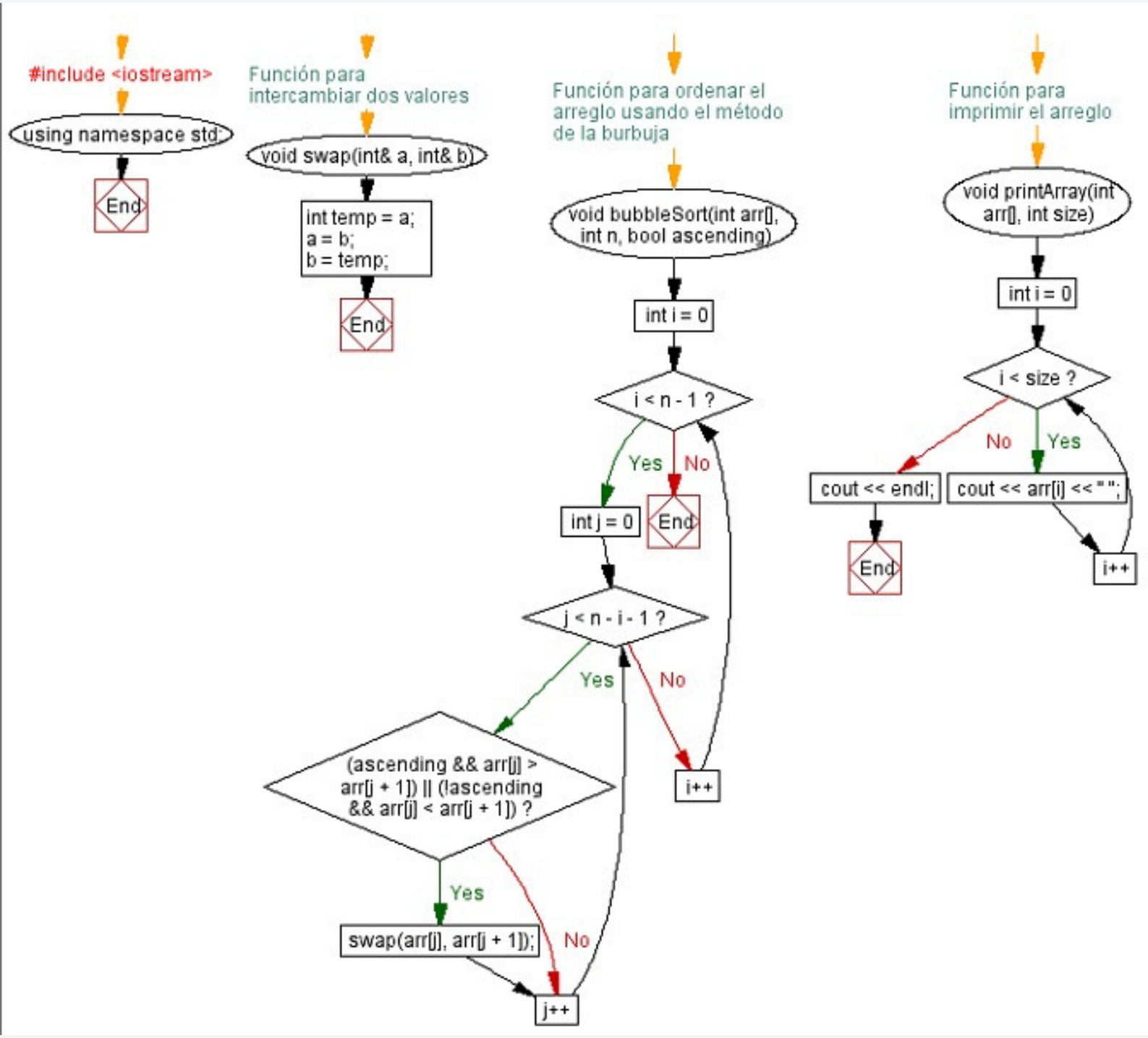
Codigo

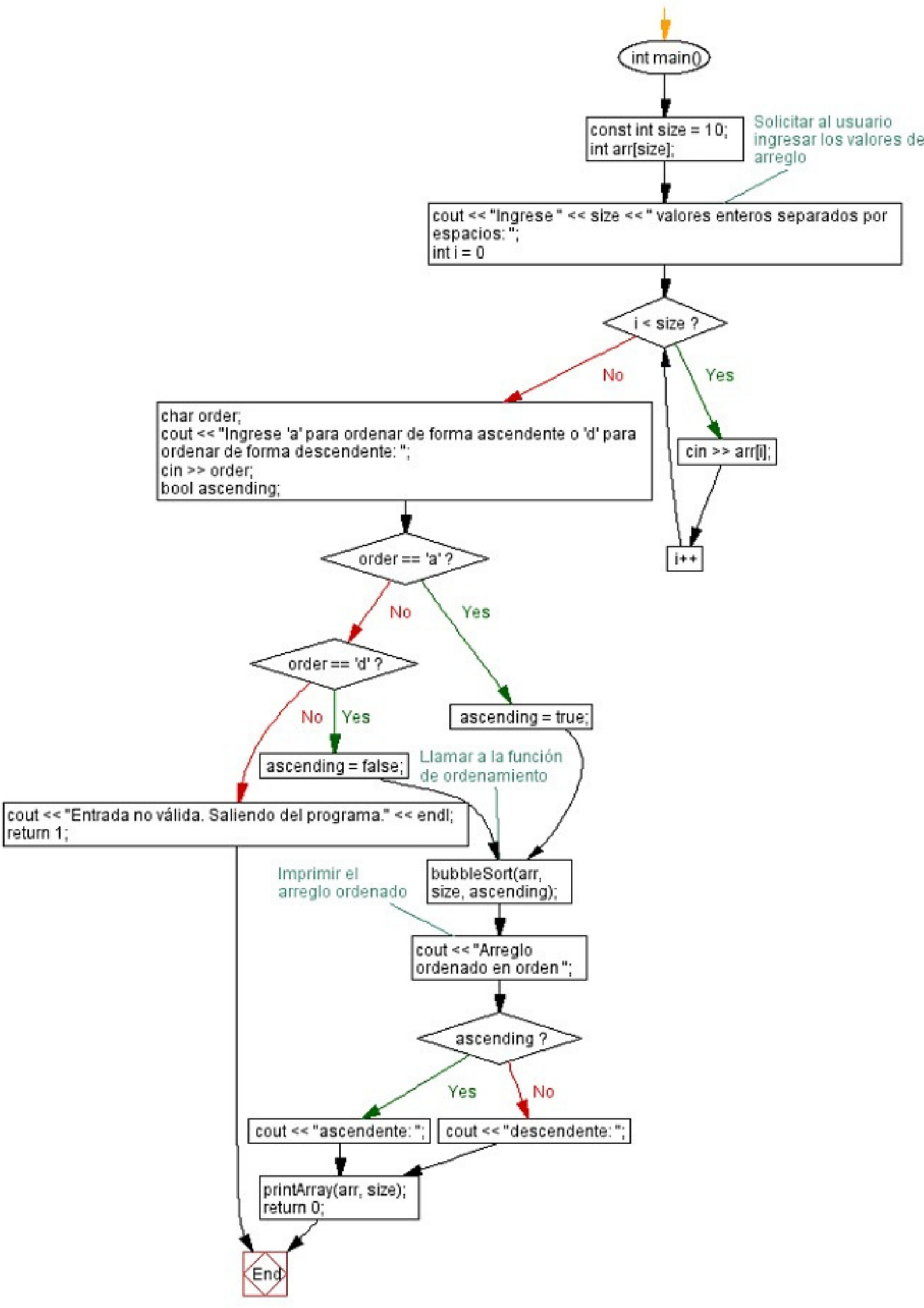


Ejercicio Python > OrdenamientoBurb.cc > printArray(int [], int)

```cpp
#include <iostream>
using namespace std;

void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

void bubbleSort(int arr[], int n, bool ascending) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if ((ascending && arr[j] > arr[j + 1]) || (!ascending && arr[j] < arr[j + 1])) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    const int size = 10;
    int arr[size];

    cout << "Ingrese " << size << " valores enteros separados por espacios: ";
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }

    char order;
    cout << "Ingrese 'a' para ordenar de forma ascendente o 'd' para ordenar de forma descendente: ";
    cin >> order;

    bool ascending;
    if (order == 'a') {
        ascending = true;
    } else if (order == 'd') {
        ascending = false;
    } else {
        cout << "Entrada no válida. Saliendo del programa." << endl;
        return 1;
    }

    bubbleSort(arr, size, ascending);

    cout << "Arreglo ordenado en orden ";
    if (ascending) {
        cout << "ascendente: ";
    } else {
        cout << "descendente: ";
    }
    printArray(arr, size);

    return 0;
}
```

Diagrama


#include <iostream>
using namespace std;
End
Función para intercambiar dos valores
void swap(int& a, int& b)
int temp = a;
a = b;
b = temp;
End
Función para ordenar el arreglo usando el método de la burbuja
void bubbleSort(int arr[], int n, bool ascending)
int i = 0
i < n - 1 ?
Yes
No
End
int j = 0
j < n - i - 1 ?
Yes
No
(ascending && arr[j] > arr[j + 1]) || (!ascending && arr[j] < arr[j + 1]) ?
Yes
No
swap(arr[j], arr[j + 1]);
j++
i++
Función para imprimir el arreglo
void printArray(int arr[], int size)
int i = 0
i < size ?
No
Yes
cout << endl;
cout << arr[i] << " ";
End
i++


int main()
const int size = 10;
int arr[size];
Solicitar al usuario ingresar los valores de arreglo
cout << "Ingrese " << size << " valores enteros separados por espacios: ";
int i = 0
i < size ?
No
Yes
cin >> arr[i];
i++
char order;
cout << "Ingrese 'a' para ordenar de forma ascendente o 'd' para ordenar de forma descendente: ";
cin >> order;
bool ascending;
order == 'a' ?
No
Yes
order == 'd' ?
No
Yes
ascending = true;
ascending = false;
Llamar a la función de ordenamiento
cout << "Entrada no válida. Saliendo del programa." << endl;
return 1;
Imprimir el arreglo ordenado
bubbleSort(arr, size, ascending);
cout << "Arreglo ordenado en orden ";
ascending ?
Yes
No
cout << "ascendente: ";
cout << "descendente: ";
printArray(arr, size);
return 0;
End

Resultados

Ordenamiento Ascendente

```
Ingrese 10 valores enteros separados por espacios: 451 45 12 85 14 214 35 25 12 14
Ingrese 'a' para ordenar de forma ascendente o 'd' para ordenar de forma descendente: a
Arreglo ordenado en orden ascendente: 12 12 14 14 25 35 45 85 214 451
PS C:\Users\Enrique HH\Documents\Phython\Ejercicio Python\output>
```

Ordenamiento Decendente

```
PS C:\Users\Enrique HH\Documents\Phython\Ejercicio Python\output> & .\'OrdenamientoBurb.exe'
Ingrese 10 valores enteros separados por espacios: 15 248 12 03 520 45 157 24 150 630
Ingrese 'a' para ordenar de forma ascendente o 'd' para ordenar de forma descendente: d
Arreglo ordenado en orden descendente: 630 520 248 157 150 45 24 15 12 3
PS C:\Users\Enrique HH\Documents\Phython\Ejercicio Python\output>
```

Implementacion

El método de ordenamiento de burbuja es un algoritmo simple y fácil de implementar, pero no suele ser la opción más eficiente para grandes conjuntos de datos debido a su complejidad temporal cuadrática. Sin embargo, puede ser útil en ciertas situaciones de la vida cotidiana donde la cantidad de datos es pequeña o la eficiencia no es la principal preocupación. Aquí hay algunos ejemplos de usos y funcionamientos en la vida cotidiana:

1. Pequeños conjuntos de datos: Cuando se trabaja con un número relativamente pequeño de elementos, como una lista de tareas pendientes, una lista de compras, o una lista de contactos telefónicos, el método de burbuja puede ser una opción viable para ordenar los elementos.
2. Ordenamiento visual: Aunque no es práctico para grandes conjuntos de datos, el método de burbuja puede ser útil para demostraciones visuales o propósitos educativos. Por ejemplo, en un aula, se puede usar para enseñar los conceptos básicos de algoritmos de ordenamiento y cómo funcionan.

Implementacion

1. Aplicaciones simples: En aplicaciones o scripts pequeños donde la simplicidad y la facilidad de implementación son más importantes que la eficiencia, el método de burbuja podría ser suficiente para ordenar los datos.
2. Ordenamiento preliminar: Aunque el método de burbuja no es el más eficiente, puede ser útil como una primera etapa de un algoritmo de ordenamiento más complejo. Por ejemplo, podría utilizarse para realizar una clasificación inicial antes de aplicar un algoritmo más rápido como el Quicksort o el Merge Sort.
3. Aprendizaje y práctica: El método de burbuja es a menudo uno de los primeros algoritmos de ordenamiento que se enseñan en cursos de informática. Practicar su implementación y comprender cómo funciona puede ayudar a los estudiantes a comprender los fundamentos de los algoritmos de ordenamiento y desarrollar habilidades de programación.

n resumen, aunque el método de ordenamiento de burbuja no es a opción más eficiente en la mayoría de los casos, tiene su lugar en situaciones donde la simplicidad, la comprensión conceptual y el tamaño de los datos son más importantes que la velocidad de ejecución.

Conclusion

En conclusión, el método de ordenamiento de burbuja es una herramienta útil en ciertos contextos específicos, pero tiene limitaciones en términos de eficiencia y aplicabilidad en conjuntos de datos grandes. Aquí hay algunas conclusiones sobre su uso en este caso particular:

1. **Simplicidad de implementación**: El método de burbuja es fácil de entender e implementar, lo que lo hace adecuado para situaciones donde la simplicidad y la claridad del código son prioritarias.
2. **Eficiencia en conjuntos de datos pequeños**: Aunque su complejidad temporal es, lo que lo hace ineficiente para grandes conjuntos de datos, el método de burbuja puede ser suficientemente rápido para ordenar conjuntos de datos pequeños en la vida cotidiana.
3. **Usos educativos y demostrativos**: El método de burbuja es útil en entornos educativos y demostrativos para enseñar los conceptos básicos de algoritmos de ordenamiento y cómo funcionan.

En resumen, el método de ordenamiento de burbuja tiene su lugar en ciertas situaciones donde la simplicidad y la claridad del código son importantes, pero se debe tener cuidado al utilizarlo en aplicaciones donde la eficiencia es crucial. Es importante evaluar las necesidades específicas del problema y considerar otras opciones de ordenamiento más eficientes si es necesario.