Utiliza el algoritmo de ordenamiento burbuja para ordenar los números en orden ascendente. El algoritmo comparará dos elementos adyacentes en la lista, y si están en el orden incorrecto (es decir, el primero es mayor que el segundo), los intercambiará. Este proceso se repetirá varias veces hasta que todos los elementos de la lista estén correctamente ordenados de menor a mayor.

```
Algoritmo burbuja5
    //Este algoritmo compara un par de datos, para ordenarlos y clasificarlos.
    //Entradas: arreglo, i ,  , auxiliar como enteros
    //Salidas: arreglo[i] ordenado
    //Caso de prueba: arreglo[5]= 34, 56, 56 , 1, 237     arreglo [5]= 1 34 56 56 237
    //Definicion de variables
    Definir i , j , arreglo, auxiliar Como Entero;

    //Inicialización
    i← 0;
    j← 0;
    auxiliar← 0;

    //arreglo
    Dimension arreglo[5];

    //Ingresar los elementos al arreglo
    Para i← 0 Hasta 4 Con Paso 1 Hacer
        Escribir "Ingrese un número";
        Leer arreglo[i];
    Fin Para

    Para i←0 Hasta 3 Hacer          //ciclo de recorridos n-1
        Para j← 0 Hasta 3-i  Hacer  //Ciclo para comparar posiciones "hasta 3 - i" optimiza el codigo así en cada iteración se reduce el recorrido
            Si arreglo[j] > arreglo[j+1] Entonces
                auxiliar← arreglo[j];   //Guardo el valor de la posicion 0
                arreglo[j] ← arreglo[j+1]; //Asigno el valor de la posicion 0 al numero que estaba en la posicion 1
                arreglo[j+1]← auxiliar;  //Asigno el valor de la posicion 1 al número que estaba en la posicion 0
            Fin Si
        Fin Para
    Fin Para

    Para i←0 Hasta 4 Hacer
        Escribir sin saltar arreglo[i] " " ;
    Fin Para
FinAlgoritmo
```

```
Algoritmo burbuja5
    //Este algoritmo compara un par de datos, para ordenarlos y clasificarlos.
    //Entradas: arreglo, i ,  , auxiliar como enteros
    //Salidas: arreglo[i] ordenado
    //Caso de prueba: arreglo[5]= 34, 56, 56 , 1, 237
    //Definicion de variables
    Definir i , j , arreglo, auxiliar Como Entero;

    //Inicialización
    i← 0;
    j← 0;
    auxiliar← 0;

    //arreglo
    Dimension arreglo[5];

    //Ingresar los elementos al arreglo
    Para i← 0 Hasta 4 Con Paso 1 Hacer
        Escribir "Ingrese un número";
        Leer arreglo[i];
    Fin Para

    Para i←0 Hasta 3 Hacer          //ciclo de recorridos n-1
        Para j← 0 Hasta 3-i  Hacer  //Ciclo para comparar posiciones "hasta 3 - i" optimiza el codigo así en cada iteración se reduce
            Si arreglo[j] > arreglo[j+1] Entonces
                auxiliar← arreglo[j];   //Guardo el valor de la posicion 0
                arreglo[j] ← arreglo[j+1]; //Asigno el valor de la posicion 0 al numero que estaba en la posicion 1
                arreglo[j+1]← auxiliar;  //Asigno el valor de la posicion 1 al número que estaba en la posicion 0
```

```
PSeInt - Ejecutando proceso BURBUJA5
*** Ejecución Iniciada. ***
Ingrese un número
> 34
Ingrese un número
> 56
Ingrese un número
> 56
Ingrese un número
> 1
Ingrese un número
> 237
1 34 56 56 237 *** Ejecución Finalizada. ***
No cerrar esta ventana   Siempre visible   Reiniciar
```

Razón de mi elección.

Elegí el algoritmo de ordenamiento burbuja porque **es** el primer algoritmo de ordenamiento con el que me encontré y me pareció intuitivo. Es ideal para entender el concepto de comparación e intercambio de elementos, lo que se aplica a muchas situaciones cotidianas. A pesar de que no es el más eficiente en grandes volúmenes de datos