

Universidad de Guanajuato División de Ingenierías
Campus Irapuato Salamanca (DICIS)

Algoritmos y estructura de datos
Carlos Hugo García Capulín

Tarea No. 14
Reporte Ordenamiento por Inserción directa

Jair Chávez Islas
03/Diciembre/2021

Problema

El método de ordenación por inserción directa consiste en recorrer todo el array comenzando desde el segundo elemento hasta el final. Para cada elemento, se trata de colocarlo en el lugar correcto entre todos los elementos anteriores a él o sea entre los elementos a su izquierda en el array.

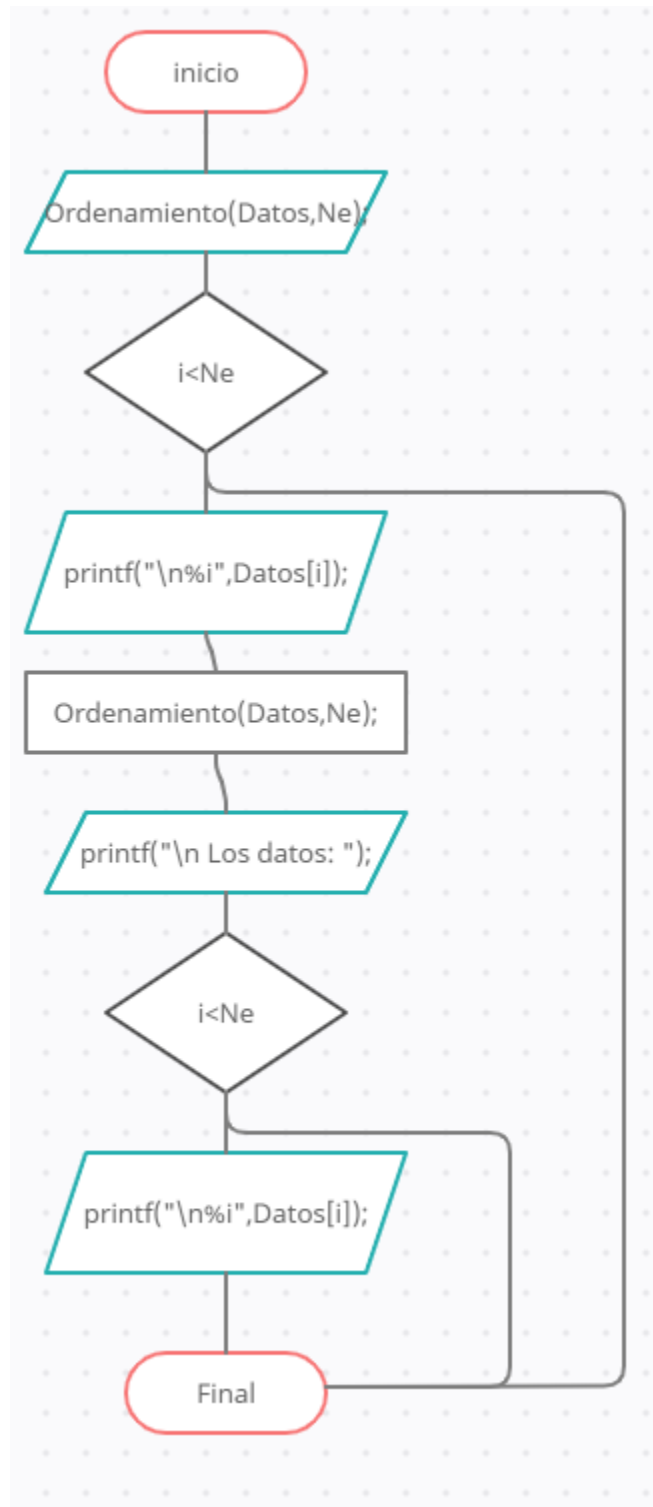
30	15	2	21	44	8	Array original
30	15	2	21	44	8	Se empieza por el segundo elemento. Se compara con el primero. Como $15 < 30$ se desplaza el 30 hacia la derecha y se coloca el 15 en su lugar
15	30	2	21	44	8	Seguimos por el tercer elemento. Se compara con los anteriores y se van desplazando hasta que el 2 queda en su lugar.
15	30	2	21	44	8	
2	15	30	21	44	8	Continuamos por el cuarto elemento. Se compara con los anteriores y se van desplazando hasta que el 21 queda en su lugar.
2	15	30	21	44	8	
2	15	21	30	44	8	Lo mismo para el quinto elemento
2	15	21	30	44	8	En este caso ya está en su posición correcta respecto a los anteriores.
2	15	21	30	44	8	Y finalmente se coloca el último elemento
2	15	21	30	44	8	El array queda ordenado

Suponiendo que tengamos los siguientes datos: 15, 67, 8, 16, 44, 27, 12, 35.

Al ordenarlos quedan de la siguiente manera: 8, 12, 15, 16, 27, 35, 44, 65

Solución implementada

Diagrama del programa



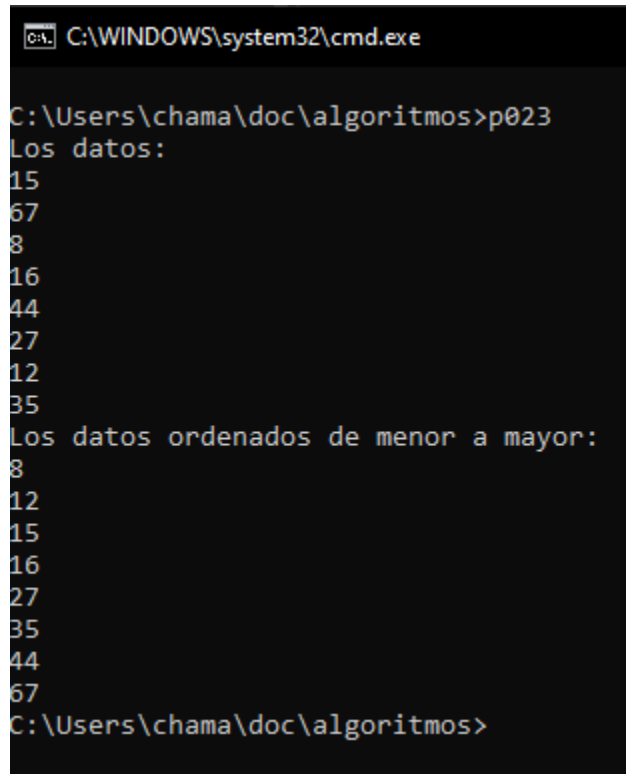
Código comentado del programa

```

1 //Agregamos las librerias necesarias para las funciones que necesitamos
2 #include <stdio.h>
3
4 //Aquí ponemos los prototipos de las funciones utilizadas
5 void Ordenamiento(int *dat, unsigned int Ne);
6
7 //Inicializamos la funcion principal
8 int main()
9 {
10     //Declaracion de variables de esta funcion
11     int Ne=8;
12     int Datos[8]={15, 67, 8, 16, 44, 27, 12, 35};
13     int i;
14
15     //Se imprimen los datos
16     printf("Los datos: ");
17     for(i=0; i<Ne; i++)
18         printf("\n%i", Datos[i]);
19     //Se aplica la funcion de ordenamiento
20     Ordenamiento(Datos, Ne);
21     printf("\nLos datos ordenados de menor a mayor:");
22     //Se imprimen los datos
23     for(i=0; i<Ne; i++)
24         printf("\n%i", Datos[i]);
25     return 0;
26 }
27
28 //Se inicializa la funcion Ordenamiento
29 void Ordenamiento(int *dat, unsigned int Ne)
30 {
31     //Declaracion de las variables de la funcion
32     unsigned int k;
33     unsigned int j;
34     int aux;
35
36     for(j=0; j<(Ne-1); j++)
37     {
38         k=j;
39         while((k>=0)&&(dat[k]>dat[k+1]))
40         {
41             //intercambio de los datos
42             aux=dat[k];
43             dat[k]=dat[k+1];
44             dat[k+1]=aux;
45             k--;
46         }
47     }
48
49 }

```

Pruebas y resultados



```
C:\WINDOWS\system32\cmd.exe

C:\Users\chama\doc\algoritmos>p023
Los datos:
15
67
8
16
44
27
12
35
Los datos ordenados de menor a mayor:
8
12
15
16
27
35
44
67
C:\Users\chama\doc\algoritmos>
```

Los resultados son como los que vimos en la introducción