

Práctica 1: ORDENAMIENTO POR INSERCIÓN.



Grupo: **2CM2**

Profra. Cecilia Alborante Morato

Integrantes:

- Guevara Badillo Areli Alejandra
- Hernández Simón Rebeca

14 de Marzo, 2023

INTRODUCCIÓN

La palabra Algoritmo tiene su origen en el nombre del matemático Persa "Mohamed ibn Musa al Khwarizmi" (825 d.C.). Su apellido fue traducido al latín como Algorismus y posteriormente pasó al español como Algoritmo. Su trabajo con los algoritmos introdujo el método de cálculo utilizando la numeración arábiga y la notación decimal. En computación, los algoritmos son una herramienta que permiten describir claramente un conjunto finito de instrucciones, ordenadas secuencialmente y libres de ambigüedad, que debe llevar a cabo un computador para lograr un resultado previsible. Al diseñar un algoritmo se tiene que tener algunos puntos a consideración:

- Comprender el alcance.
- Identificar los datos de entrada.
- Identificar los datos de salida o resultados.

El algoritmo por inserción es un método de ordenamiento que se encarga de ordenar una pequeña cantidad de elementos, es de los métodos más eficientes y simples si es que se trata de ordenar pocos elementos. El ordenamiento por inserción analiza de izquierda a derecha los datos de una sucesión, este se repite para cada elemento de la lista hasta que todos queden ordenados.

La idea fundamental de este es ir comparando el elemento actual con el elemento antecesor y si es que es menor al elemento actual intercambiar su posición. Se sigue comparando recorriendo el elemento hacia atrás hasta que llegue a la posición que le corresponde. Este se basa en que, a medida que se recorre la lista, se insertan los elementos en la posición correcta para que quede ordenada.

Su complejidad es de tipo $O(n^2)$. Esto lo hace menos eficiente que otros algoritmos de ordenamiento, no obstante, su simplicidad de implementación lo convierte en una alternativa para ordenar una pequeña cantidad de elementos.



ORDENAMIENTO POR INSERCIÓN

Planteamiento de problema:

Realizar un programa que ordene n cantidad de números aleatorios (no más de 50 datos) por ordenamiento por inserción, que imprima los números antes y después de ordenar.

Para la resolución del problema declaramos 3 funciones distintas:

```
6  int entrada(void); //pide el no. de datos al usuario
7  void llenado(int* , int); //llena el arreglo
8  void insercion(int* , int); //ordena el arreglo
```

La función *entrada* es la encargada de solicitar el número de datos al usuario.

Dando una advertencia en caso de que el número de datos que quiera ordenar el usuario sea mayor a 50:

```
28 int entrada() {
29     int n;
30
31     do{
32         printf("Dame el numero datos a ordenar?: "); //solicita el no. de datos a ordenar
33         scanf("%d", &n);
34
35         if(n > 50){ //en caso de que n sea mayor a 50 lanza una alerta
36             printf("\nEl numero de datos no debe de ser mayor a 50!!!!\n\n");
37             system("pause");
38             system("cls");
39         }
40     }while(n > 50); //se repite el proceso hasta que n cumpla las condiciones.
41
42     return n; //no devuelve en numero de datos.
43 }
```

El valor que nos devuelve se guarda en una variable n y con esta se declara el tamaño del arreglo:

```
14 int n = entrada(); //guarda el valor devuelto por la funcion entrada();
15 int numeros[n]; //declara el arreglo de tamaño n.
```

Ahora llamamos a la función *llenado* y le mandamos el arreglo y su tamaño, esta función llenará el arreglo con numero aleatorios en un intervalo de 1 a 99:

```
17     printf("\nNumeros sin Ordenar\n");
18     llenado(numeros, n); //llama la funcion llenado();
```

```
47 void llenado(int* numeros, int n){
48     srand(time(NULL)); //inicializa la funcion rand
49
50     for(int i = 0; i < n; i++){
51         numeros[i] = (rand()%98+1); //se lleva cada indice con un numero aleatorio
52         printf("%d\t", numeros[i]); //imprime el indice que se acaba de llenar
53     }
54 }
```

Finalmente llamamos a la función *inserción* la cual ordenará nuestro arreglo por el método requerido, para esto le mandamos nuestro arreglo ya lleno y el tamaño del mismo:

```

20     printf("\n\nNumeros Ordenados\n");
21     insercion(numeros, n); //llama la funcion insercion();

```

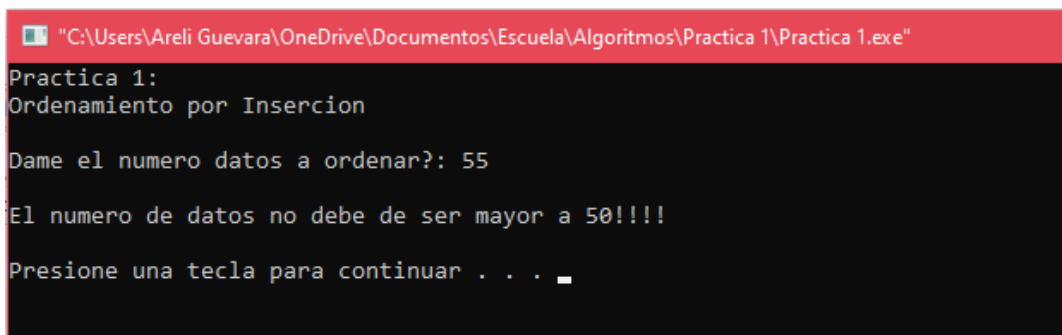
```

54 void insercion(int* numeros, int n){
55     int i, j, aux;
56
57     for(i = 1; i < n; i++){ //indice principal
58         j = i; //igualar el valor de j a i
59         aux = numeros[i]; //le da valor a la variable auxiliar
60
61         while((j > 0) && (numeros[j-1] > aux)){ //verificara si j es mayor a 0 y si el antecesor del numero
62             //a comparar es mayor si esto es verdadero ejecuta el while
63             numeros[j] = numeros[j-1]; //pone al antecesor en el lugar del numero comparado
64             j--; //decrementa uno a j
65         }
66         //cuando el while ya no se cumpla
67         numeros[j] = aux; //coloca el numero comparado en el lugar que le corresponde
68     }
69
70     for(i = 0; i < n; i++)
71         printf("%d\t", numeros[i]); //imprime el arreglo ordenado
72 }

```

RESULTADOS

Cuando el número de datos es mayor a 50, se manda un mensaje en la salida estándar especificando que el dato que se debe ingresar debe ser menor o igual a 50.

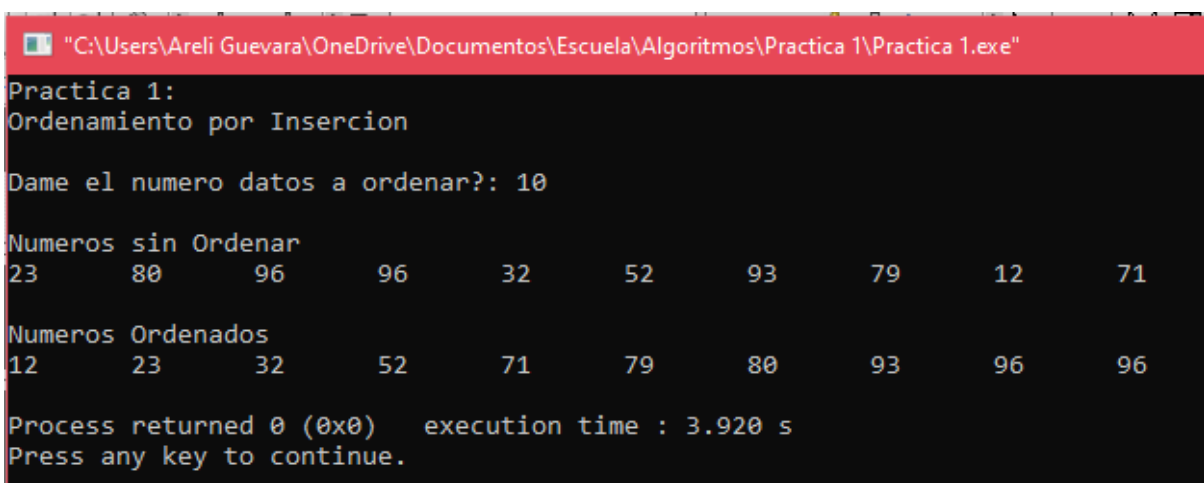


```

"C:\Users\Arelí Guevara\OneDrive\Documentos\Escuela\Algoritmos\Practica 1\Practica 1.exe"
Practica 1:
Ordenamiento por Insercion
Dame el numero datos a ordenar?: 55
El numero de datos no debe de ser mayor a 50!!!!
Presione una tecla para continuar . . .

```

Si el número de datos que se ingresó está en el rango indicado se mostrará el arreglo con números aleatorios de forma desordenada y ordenada respectivamente.



```

"C:\Users\Arelí Guevara\OneDrive\Documentos\Escuela\Algoritmos\Practica 1\Practica 1.exe"
Practica 1:
Ordenamiento por Insercion
Dame el numero datos a ordenar?: 10

Numeros sin Ordenar
23      80      96      96      32      52      93      79      12      71

Numeros Ordenados
12      23      32      52      71      79      80      93      96      96

Process returned 0 (0x0)   execution time : 3.920 s
Press any key to continue.

```