ALGORITMOS DE ORDENAMIENTO

ALGORITMOS DE ORDENAMIENTO

Son funciones que se utilizan para cambiar el orden de los elementos de un arreglo. Hay algunos lenguajes que ya tienen la opción predefinida de “ordenar” sus componentes, pero es muy importante aprender como funcionan.

ORDENAMIENTO TIPO BURBUJA

ORDENAMIENTO BURBUJA

Compara e intercambia repetidamente los elementos hasta ordenar todo el arreglo. Va ordenando casilla por casilla y se tiene que volver a iniciar cuando se reordena una parte.

Se le llama burbuja por el hecho de que los elementos van "burbujeando" hasta el principio o final del arreglo, hasta que todo esté ordenado.

ORDENAMIENTO BURBUJA

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

void burbuja(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Intercambiar elementos si están en el
orden incorrecto
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    burbuja(arr, n);
    cout << "Arreglo ordenado: \n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

ORDENAMIENTO POR SELECCIÓN

ORDENAMIENTO POR SELECCIÓN

Al igual que el ordenamiento en burbuja, se comparan elementos hasta encontrar el orden final, con la diferencia de que va seleccionando los elementos de manera ordenada para rellenar el arreglo final.

Dependiendo de si busca el más pequeño o el más grande, primero escoge el elemento que más cumple con esta condición, para ir buscando los elementos que cumplan en menor medida con el criterio de búsqueda.

ORDENAMIENTO POR SELECCIÓN

ORDENAMIENTO POR SELECCIÓN

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

void seleccion(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        int min_idx = i;
        for (int j = i+1; j < n; j++) {
            if (arr[j] < arr[min_idx])
                min_idx = j;
        }
        // Intercambiar el elemento mínimo encontrado con el
primer elemento no ordenado
        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

int main() {
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    seleccion(arr, n);
    cout << "Arreglo ordenado: \n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

ORDENAMIENTO POR INSERCIÓN

ORDENAMIENTO POR INSERCIÓN

Es el más ordenado y eficiente de los 3. Este algoritmo funciona como si se barajara un conjunto de cartas.

- Comienza desde el segundo elemento de la lista y lo considera como elemento a insertar.
- Compara el elemento a insertar con los elementos anteriores en la lista y los mueve hacia la derecha si son mayores que el elemento a insertar.
- Repite el paso 2 hasta encontrar la posición correcta para insertar el elemento "a insertar" en la lista ordenada.
- Avanza al siguiente elemento y repite los pasos anteriores hasta que todos los elementos estén en su posición ordenada.

ORDENAMIENTO POR INSERCIÓN

ORDENAMIENTO POR INSERCIÓN

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

void insercion(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

         // Mover los elementos del subarreglo ordenado
hacia adelante si son mayores que la clave
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    insercion(arr, n);
    cout << "Arreglo ordenado: \n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

ORDENAMIENTO POR INSERCIÓN

E J E R C I C I O S

- Ingresar un arreglo de 8 números enteros, ordenarlos de manera ascendente y descendente con los 3 algoritmos.
- Ingresar 5 letras mayúsculas, ordenarlos alfabéticamente con los 3 algoritmos.
- Crear un algoritmo que de manera aleatoria genere 15 números. Encontrar el número más grande y el número más pequeño de ese conjunto de números.
- Crear un arreglo de dos dimensiones, con diez casillas (10x2). Ingresar números a todas las casillas, y posteriormente ordenar el arreglo guiándose por el número en la primera casilla.

  Ejemplo:

  15-2 -> 17-4
  13-5 -> 15-2
  5-8 -> 13-5
  9-3 -> 9-3
  17-4 -> 5-8