Algoritmos de ordenación

Ordenación de listas

Los datos de la lista deben poderse comparar entre sí

Sentido de la ordenación:

- ✓ Ascendente (de menor a mayor)
- ✓ Descendente (de mayor a menor)

Algoritmos de ordenación básicos:

- ✓ Ordenación por *inserción*
- ✓ Ordenación por *selección directa*
- ✓ Ordenación por el *método de la burbuja*

Los algoritmos se basan en comparaciones e intercambios

Hay otros algoritmos de ordenación mejores

Luis Hernández Yáñez

Fundamentos de la programación

Algoritmo de ordenación por inserción

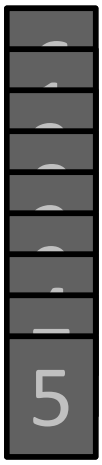
Luis Hernández Yáñez

Ordenación por inserción

Algoritmo de ordenación por inserción

Partimos de una lista vacía

Vamos insertando cada elemento en el lugar que le corresponda



Baraja de nueve cartas numeradas del 1 al 9

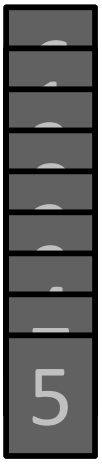
Las cartas están desordenadas

Ordenaremos de menor a mayor (ascendente)

Luis Hernández Yáñez

Ordenación por inserción

Algoritmo de ordenación por inserción



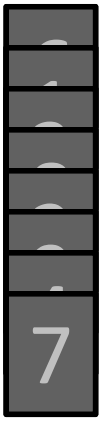
Colocamos el primer elemento en la lista vacía

Lista ordenada:

Luis Hernández Yáñez

Ordenación por inserción

Algoritmo de ordenación por inserción



El 7 es mayor que todos los elementos de la lista
Lo insertamos al final

Lista ordenada:

Luis Hernández Yáñez

Ordenación por inserción

Algoritmo de ordenación por inserción

Primer elemento (5) mayor que el nuevo (4):
Desplazamos todos una posición a la derecha
Insertamos el nuevo en la primera posición

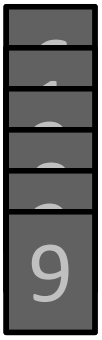
Hemos insertado el elemento en su lugar

Lista ordenada:

Luis Hernández Yáñez

Ordenación por inserción

Algoritmo de ordenación por inserción

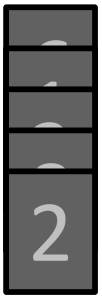


9 es mayor que todos los elementos de la lista
Lo insertamos al final

Lista ordenada:

Ordenación por inserción

Algoritmo de ordenación por inserción

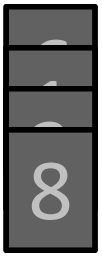


Primer elemento (4) mayor que el nuevo (2):
Desplazamos todos una posición a la derecha
Insertamos el nuevo en la primera posición

Lista ordenada:

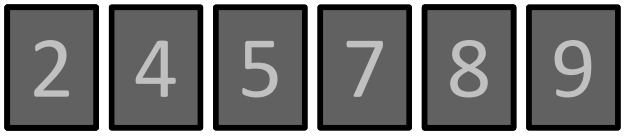
Ordenación por inserción

Algoritmo de ordenación por inserción



El 9 es el primer elemento mayor que el nuevo (8):
Desplazamos desde ese hacia la derecha
Insertamos donde estaba el 9

Lista ordenada:



Luis Hernández Yáñez

Ordenación por inserción

Algoritmo de ordenación por inserción

Segundo elemento (4) mayor que el nuevo (3):
Desplazamos desde ese hacia la derecha
Insertamos donde estaba el 4

Lista ordenada:

Luis Hernández Yáñez

Ordenación por inserción

Algoritmo de ordenación por inserción

Primer elemento (2) mayor que el nuevo (1):
Desplazamos todos una posición a la derecha
Insertamos el nuevo en la primera posición

Lista ordenada:

Ordenación por inserción

Algoritmo de ordenación por inserción

El 7 es el primer elemento mayor que el nuevo (6):
Desplazamos desde ese hacia la derecha
Insertamos donde estaba el 7

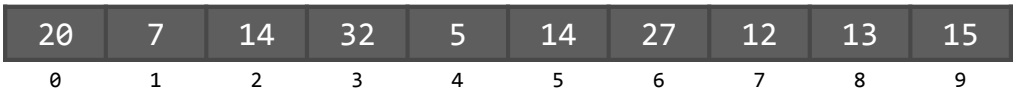
¡¡¡ LISTA ORDENADA !!!

Lista ordenada:

Ordenación por inserción

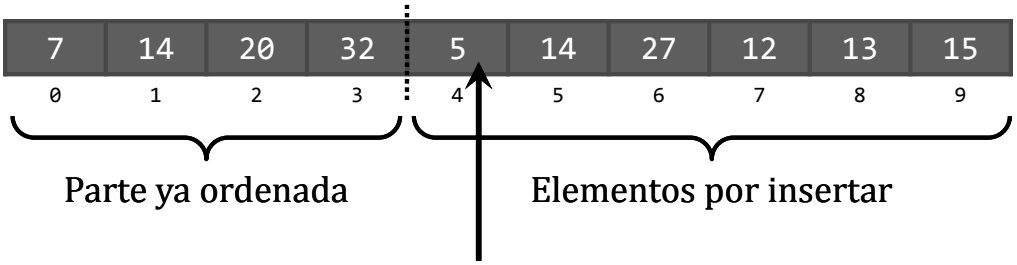
Ordenación de arrays por inserción

El array contiene inicialmente la lista desordenada:



A medida que insertamos: dos zonas en el array

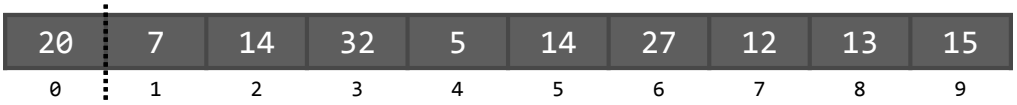
Parte ya ordenada y elementos por procesar



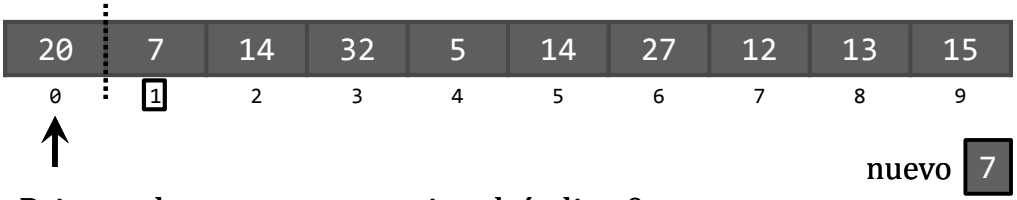
Ordenación por inserción

Ordenación de arrays por inserción

Situación inicial: Lista ordenada con un solo elemento (primero)



Desde el segundo elemento del array hasta el último:
Localizar el primer elemento mayor en lo ya ordenado



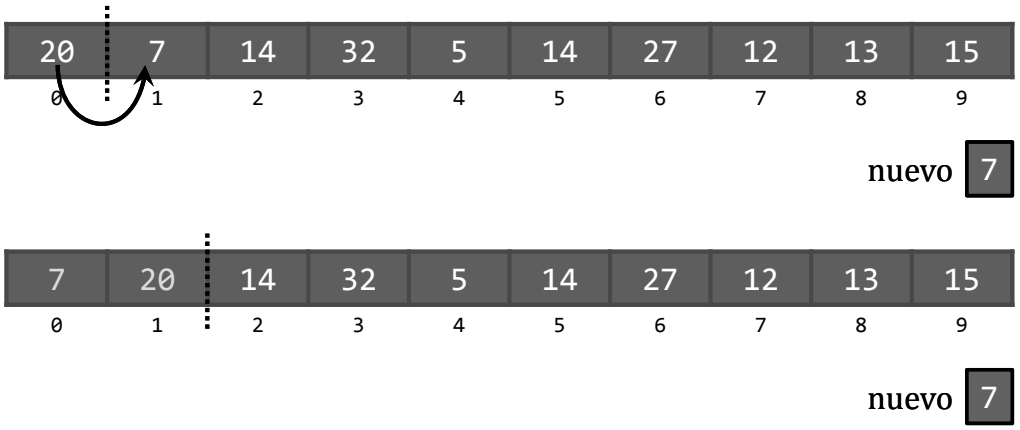
Ordenación por inserción

Ordenación de arrays por inserción

...

Desplazar a la derecha los ordenados desde ese lugar

Insertar el nuevo en la posición que queda libre



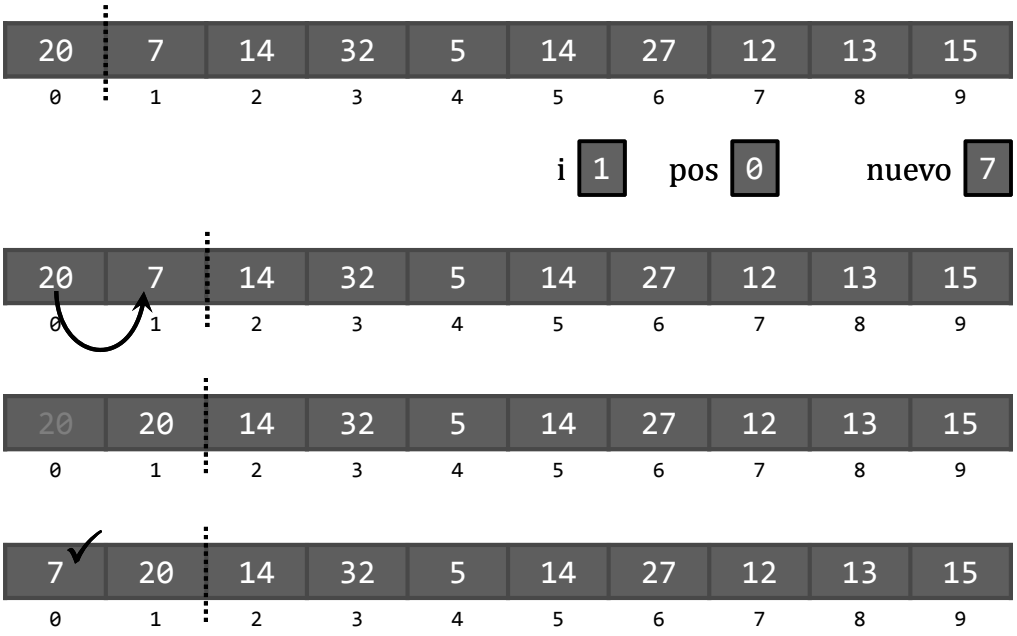
Ordenación de arrays por inserción

Implementación

```
const int N = 15;
typedef int tLista[N];
tLista lista;
```

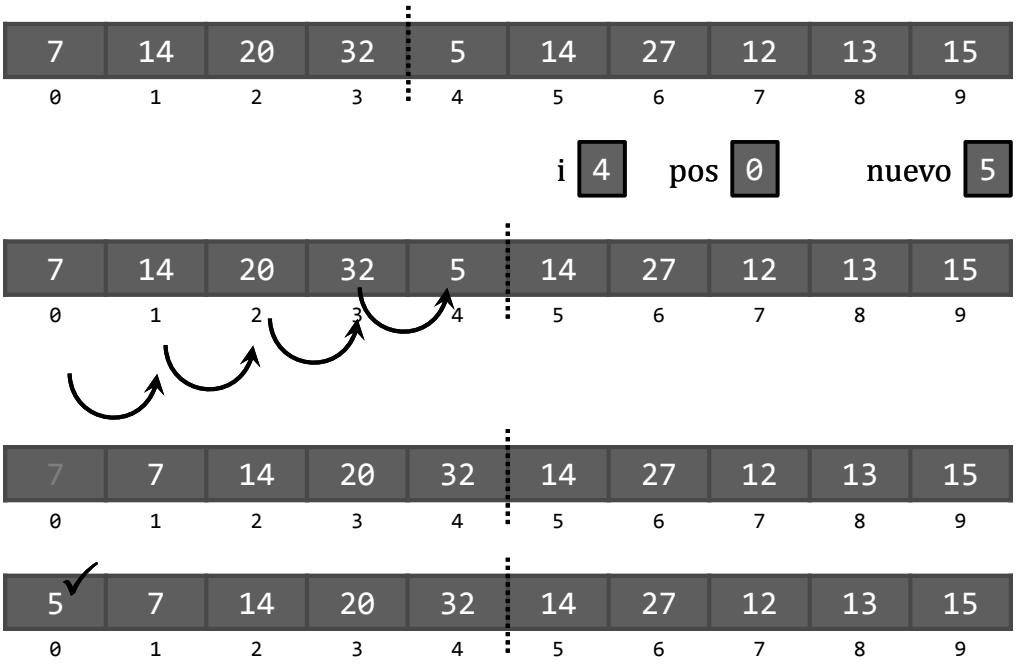
```
...
int nuevo, pos;
// Desde el segundo elemento hasta el último...
for (int i = 1; i < N; i++) {
   nuevo = lista[i];
   pos = 0;
   while ((pos < i) && !(lista[pos] > nuevo)) {
      pos++;
   }
   // pos: índice del primer mayor; i si no lo hay
   for (int j = i; j > pos; j--) {
      lista[j] = lista[j - 1];
   }
   lista[pos] = nuevo;
}
```


Ordenación de arrays por inserción



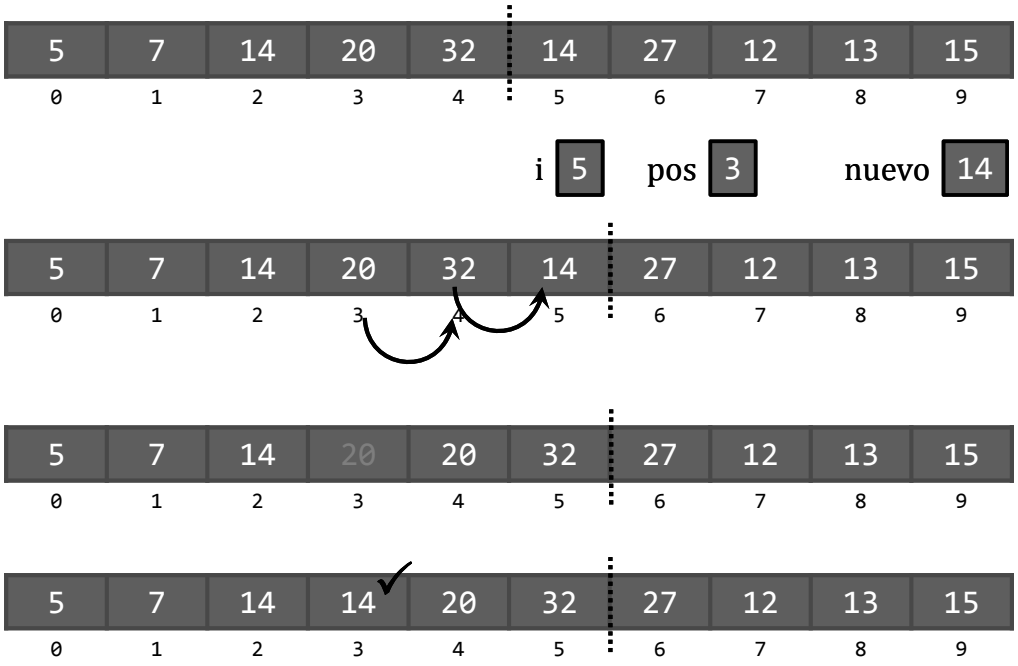
Luis Hernández Yáñez

Ordenación de arrays por inserción



Luis Hernández Yáñez

Ordenación de arrays por inserción



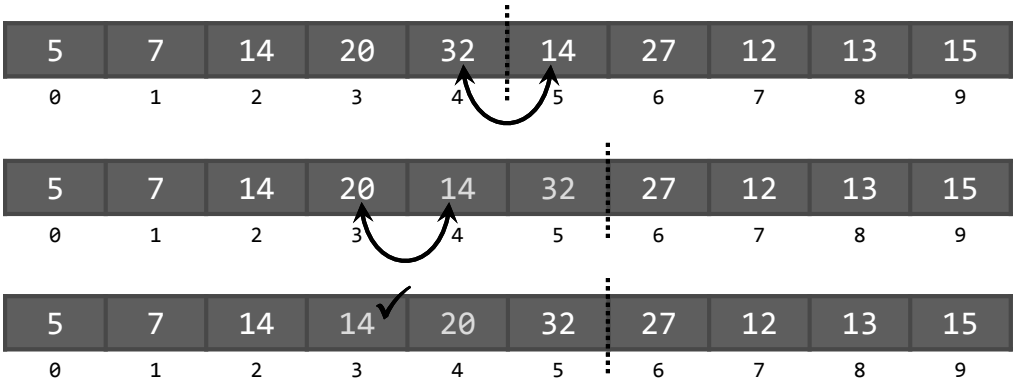
Fundamentos de la programación

Algoritmo de ordenación por inserción con intercambios

Ordenación por inserción con intercambios

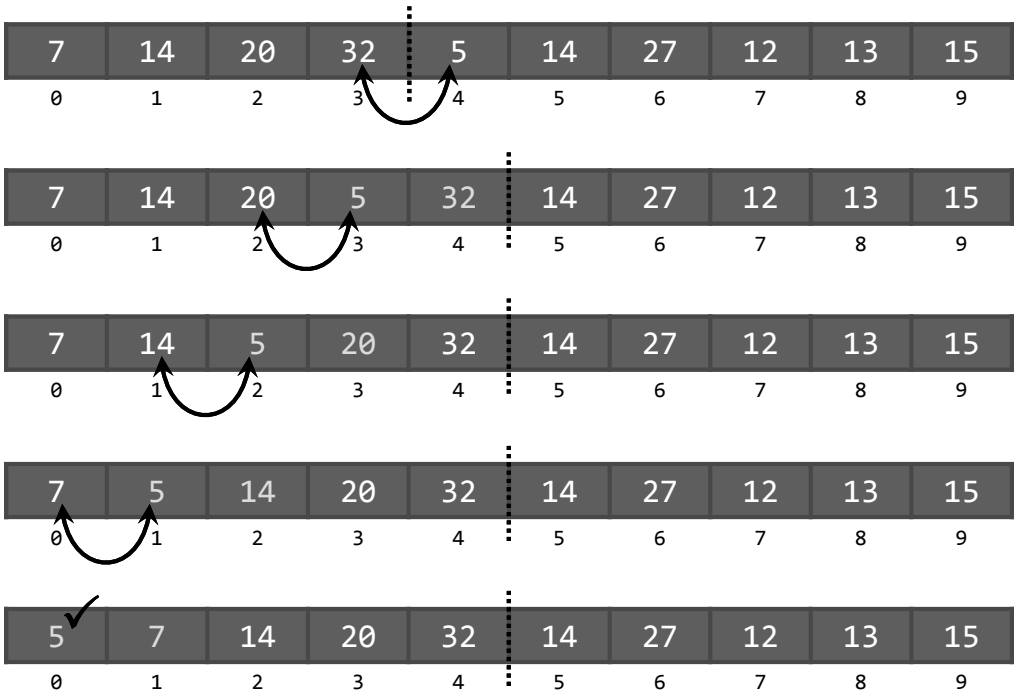
La inserción de cada elemento se puede realizar con comparaciones e intercambios

Desde el segundo elemento hasta el último:
Desde la posición del nuevo elemento a insertar:
Mientras el anterior sea mayor, intercambiar



Luis Hernández Yáñez

Ordenación por inserción con intercambios



Luis Hernández Yáñez

Ordenación por inserción con intercambios

```
const int N = 15;
typedef int tLista[N];
tLista lista;
```

```
...
int tmp, pos;
// Desde el segundo elemento hasta el último...
for (int i = 1; i < N; i++) {
   pos = i;
   // Mientras no al principio y anterior mayor...
   while ((pos > 0) && (lista[pos - 1] > lista[pos])) {
      // Intercambiar...
      tmp = lista[pos];
      lista[pos] = lista[pos - 1];
      lista[pos - 1] = tmp;
      pos--; // Posición anterior
   }
}
```


Ordenación por inserción con intercambios

insercion.cpp

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

const int N = 100;
typedef int tArray[N];
typedef struct { // Lista de longitud variable
   tArray elementos;
   int contador;
} tLista;

int main() {
   tLista lista;
   ifstream archivo;
   int dato, pos, tmp;
   lista.contador = 0;
   ...
```


Ordenación por inserción con intercambios

```
archivo.open("insercion.txt");
if (!archivo.is_open()) {
   cout << "Error de apertura de archivo!" << endl;
}
else {
   archivo >> dato;
   while ((lista.contador < N) && (dato != -1)) {
   // Centinela -1 al final
      lista.elementos[lista.contador] = dato;
      lista.contador++;
      archivo >> dato;
   }
   archivo.close();
   // Si hay más de N ignoramos el resto
   cout << "Antes de ordenar:" << endl;
   for (int i = 0; i < lista.contador; i++) {
      cout << lista.elementos[i] << "  ";
   }
   cout << endl;      ...
```


Ordenación por inserción con intercambios

```
   for (int i = 1; i < lista.contador; i++) {
      pos = i;
      while ((pos > 0)
      && (lista.elementos[pos-1] > lista.elementos[pos]))
      {
         tmp = lista.elementos[pos];
         lista.elementos[pos] = lista.elementos[pos - 1];
         lista.elementos[pos - 1] = tmp;
         pos--;
      }
   }
   cout << "Después de ordenar:" << endl;
   for (int i = 0; i < lista.contador; i++) {
      cout << lista.elementos[i] << "  ";
   }
   cout << endl;
}
return 0;
}
```

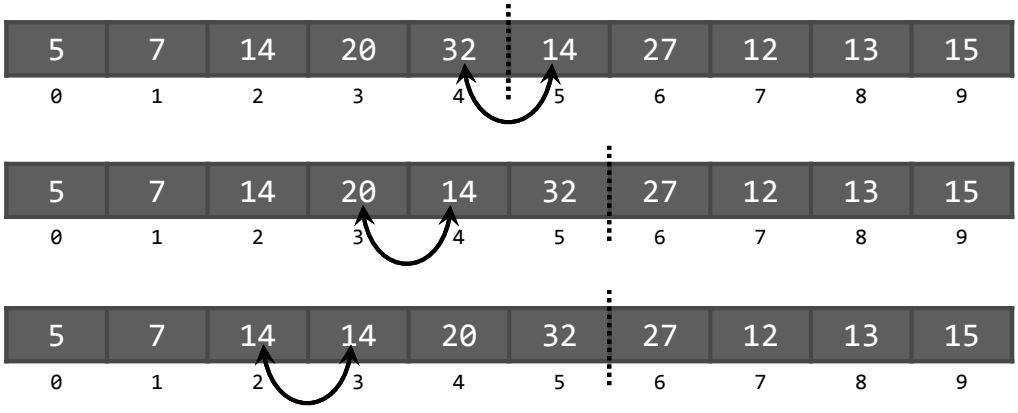

Ordenación por inserción con intercambios

Consideración de implementación

¿Operador relacional adecuado?

`lista[pos - 1]` ¿ > o >= ? `lista[pos]`

Con >= se realizan intercambios inútiles:



¡Intercambio inútil!

Fundamentos de la programación

Claves de ordenación

Ordenación por inserción

Claves de ordenación

Elementos que son estructuras con varios campos:

```
const int N = 15;
typedef struct {
   int codigo;
   string nombre;
   double sueldo;
} tDato;
typedef tDato tLista[N];
tLista lista;
```

Clave de ordenación:

Campo en el que se basan las comparaciones

Luis Hernández Yáñez

Ordenación por inserción

Claves de ordenación

```
tDato tmp;
while ((pos > 0)
      && (lista[pos - 1].nombre > lista[pos].nombre)) {
   tmp = lista[pos];
   lista[pos] = lista[pos - 1];
   lista[pos - 1] = tmp;
   pos--;
}
```

Comparación: campo concreto

Intercambio: elementos completos

Luis Hernández Yáñez

Ordenación por inserción

Claves de ordenación

Función para la comparación:

```
bool operator>(tDato opIzq, tDato opDer) {
   return (opIzq.nombre > opDer.nombre);
}
```

```
tDato tmp;
while ((pos > 0) && (lista[pos - 1] > lista[pos])) {
   tmp = lista[pos];
   lista[pos] = lista[pos - 1];
   lista[pos - 1] = tmp;
   pos--;
}
```


Ordenación por inserción

claves.cpp

Claves de ordenación

```
#include <iostream>
#include <string>
using namespace std;
#include <fstream>
#include <iomanip>
const int N = 15;
typedef struct {
   int codigo;
   string nombre;
   double sueldo;
} tDato;
typedef tDato tArray[N];
typedef struct {
   tArray datos;
   int cont;
} tLista;
...
```


Ordenación por inserción

```
void mostrar(tLista lista);
bool operator>(tDato opIzq, tDato opDer);

int main() {
   tLista lista;
   ifstream archivo;
   lista.cont = 0;
   archivo.open("datos.txt");
   if (!archivo.is_open()) {
      cout << "Error de apertura del archivo!" << endl;
   }
   else {
      tDato dato;
      archivo >> dato.codigo;
      while ((lista.cont < N) && (dato.codigo != -1)) {
         archivo >> dato.nombre >> dato.sueldo;
         lista.datos[lista.cont] = dato;
         lista.cont++;
         archivo >> dato.codigo;
      }
      archivo.close();        ...
```


Ordenación por inserción

```
      cout << "Antes de ordenar:" << endl;
      mostrar(lista);
      for (int i = 1; i < lista.cont; i++) {
      // Desde el segundo elemento hasta el último
         int pos = i;
         while ((pos > 0)
                && (lista.datos[pos-1] > lista.datos[pos])) {
            tDato tmp;
            tmp = lista.datos[pos];
            lista.datos[pos] = lista.datos[pos - 1];
            lista.datos[pos - 1] = tmp;
            pos--;
         }
      }
      cout << "Después de ordenar:" << endl;
      mostrar(lista);
   }
   return 0;
}                                                    ...
```


Ordenación por inserción

```
void mostrar(tLista lista) {
   for (int i = 0; i < lista.cont; i++) {
      cout << setw(10)
           << lista.datos[i].codigo
           << setw(20)
           << lista.datos[i].nombre
           << setw(12)
           << fixed
           << setprecision(2)
           << lista.datos[i].sueldo
           << endl;
   }
}

bool operator>(tDato opIzq, tDato opDer) {
   return (opIzq.nombre > opDer.nombre);
}
```

```
Antes de ordenar:
     10000              Sergei   100000.00
     10000            Hernández  150000.00
     11111             Benítez   100000.00
     11111              Urpiano   90000.00
     11111               Pérez    90000.00
     11111               Durán   120000.00
     12345              Álvarez  120000.00
     12345               Gómez   100000.00
     12345             Sánchez    90000.00
     12345             Turégano  100000.00
     21112            Domínguez   90000.00
     21112              Jiménez  100000.00
     22222            Fernández  120000.00
     33333             Tarazona  120000.00
Después de ordenar:
     11111             Benítez   100000.00
     21112            Domínguez   90000.00
     11111               Durán   120000.00
     22222            Fernández  120000.00
     12345               Gómez   100000.00
     10000            Hernández  150000.00
     21112              Jiménez  100000.00
     11111               Pérez    90000.00
     10000              Sergei   100000.00
     12345             Sánchez    90000.00
     33333             Tarazona  120000.00
     12345             Turégano  100000.00
     11111              Urpiano   90000.00
     12345              Álvarez  120000.00
```

Cambia a `codigo` o `sueldo` para ordenar por otros campos

Luis Hernández Yáñez

Fundamentos de la programación

Estabilidad de la ordenación

Luis Hernández Yáñez