Taller de Programación

AGENDA

Estructuras de Datos vistas: arreglos - listas

Concepto de Ordenación

Métodos de ordenación: selección e inserción

ARREGLOS - Características

Un **arreglo** es una estructura de datos compuesta que permite acceder a cada componente por una variable índice, que da la posición de la componente dentro de la estructura de datos. La estructura arreglo se almacena en posiciones contiguas de memoria

Características: **homogénea – estática - acceso directo indexado - lineal**

Trabajan con: **dimensión física – dimensión lógica**

ARREGLOS - Características

```
Type

 nombre= array [rango] of tipo;
```

OPERACIONES

- Cargar la estructura
- Agregar un elemento
- Insertar un elemento
- Eliminar un elemento
- Recorrer la estructura
- Buscar un elemento
- **Ordenar la estructura**

LISTAS -

Una **lista** es una estructura de datos lineal compuesta por nodos. Cada nodo de la lista posee el dato que almacena la lista y la dirección del siguiente nodo. Toda lista puede recorrerse a partir de su primer elemento. Los elementos no necesariamente están en posiciones contiguas de memoria.

Para generar nuevos elementos en la lista, o eliminar alguno se deben utilizar las operaciones de new y dispose respectivamente.

Características: **homogénea – dinámica - lineal - acceso secuencial**

LISTAS -

```
Type

 lista= ^nodo;
 nodo = record
    dato:tipo;
    sig: lista;
 end;
```

OPERACIONES

- Crear una lista vacía
- Agregar un elemento adelante
- Agregar un elemento atrás
- Insertar un elemento
- Eliminar un elemento
- Recorrer la estructura
- Buscar un elemento

ARREGLOS - Ordenación

Un **algoritmo de ordenación** es un proceso por el cual un conjunto de elementos puede ser ordenado.

Cuál sería el beneficio de tener una estructura ordenada?

20	1	7	33	5	115	87	93	35	47	9	19	68	58

1	5	7	9	19	20	33	35	47	58	68	87	93	115

ARREGLOS - Ordenación

Existen **diferentes algoritmos de ordenación para vectores** donde cada uno tiene sus ventajas y desventajas en cuanto a facilidad de escritura, cantidad de memoria requerida y tiempo de ejecución

Los más conocidos: **selección –** intercambio **- inserción**

ARREGLOS – Ordenación - Selección

SELECCIÓN dado un arreglo A y una dimensión lógica (dimL), el algoritmo repite (dimL-1) veces buscar el elemento mínimo y ubicarlo en la posición correspondiente.

Es decir, en la primera vuelta busca el mínimo desde la posición 1 hasta la dimensión lógica y ubica al mínimo en la primera posición. En la vuelta i busca el mínimo desde la posición i hasta la dimensión lógica y lo ubica en la posición i.

ARREGLOS – Ordenación - Selección

dim física = 100

dim lógica=6

9	170	2	7	6	150				

Primera vuelta (busca desde 1 hasta 6)

Posminimo = 3

Intercambia los valores de las posiciones 1 y 3

2	170	9	7	6	150				

dim física = 100

dim lógica=6

2	170	9	7	6	150				

Segunda vuelta (busca desde 2 hasta 6)

Posminimo = 5

Intercambia los valores de las posiciones 2 y 5

2	6	9	7	170	150				

ARREGLOS – Ordenación - Selección

dim física = 100

dim lógica=6

2	6	9	7	170	150				

2	6	7	9	170	150				

dim física = 100

dim lógica=6

2	6	7	9	170	150				

2	6	7	9	170	150				

Tercera vuelta (busca desde 3 hasta 6)

Posminimo = 4

Intercambia los valores de las posiciones 3 y 4

Cuarta vuelta (busca desde 4 hasta 6)

Posminimo = 4

Intercambia los valores de las posiciones 4 y 4

ARREGLOS – Ordenación - Selección

dim lógica=6

2	6	7	9	170	150				

2	6	7	9	150	170				

Quinta vuelta (busca desde 5 hasta 6)

Posminimo = 6

Intercambia los valores de las posiciones 5 y 6

ARREGLOS – Ordenación - Selección

```
Program ordenar;
Const  maxlen =... {máxima longitud del arreglo}

Type
      TipoElem = ... { tipo de datos del vector }
      Indice   =  0.. maxlen;
      Tvector =  Array  [ 1..maxlen] of TipoElem;
```

ARREGLOS – Ordenación - Selección

```
Procedure Ordenar ( var v: tVector; dimLog: indice );

var i, j, p: indice; item : tipoElem;

begin
 for i:=1 to dimLog-1 do begin {busca el mínimo y guarda en p la posición}
          p := i;
          for j := i+1 to dimLog do
             if v[ j ] < v[ p ] then p:=j;

          {intercambia v[i] y v[p]}
          item := v[ p ];
          v[ p ] := v[ i ];
          v[ i ] := item;
      end;
end;
```

ARREGLOS – Ordenación - Selección

- Es muy sencilla la implementación
- El tiempo de ejecución en N^2, siendo N el tamaño del arreglo
- Qué ocurre si los datos están ordenados inicialmente?

ARREGLOS – Ordenación - Inserción

INSERCIÓN (insertion sort) es una manera muy natural de ordenar un conjunto de elementos almacenados en un arreglo.

Inicialmente se considera un solo elemento, que obviamente es un conjunto ordenado.

Después, cuando hay k elementos ordenados de menor a mayor, se toma el elemento k+1 y se compara con todos los elementos ya ordenados, deteniéndose cuando se encuentra un elemento menor (todos los elementos mayores han sido desplazados una posición a la derecha) o cuando ya no se encuentran elementos (todos los elementos fueron desplazados y este es el más pequeño). En este punto se inserta el elemento k+1 debiendo desplazarse los demás elementos.

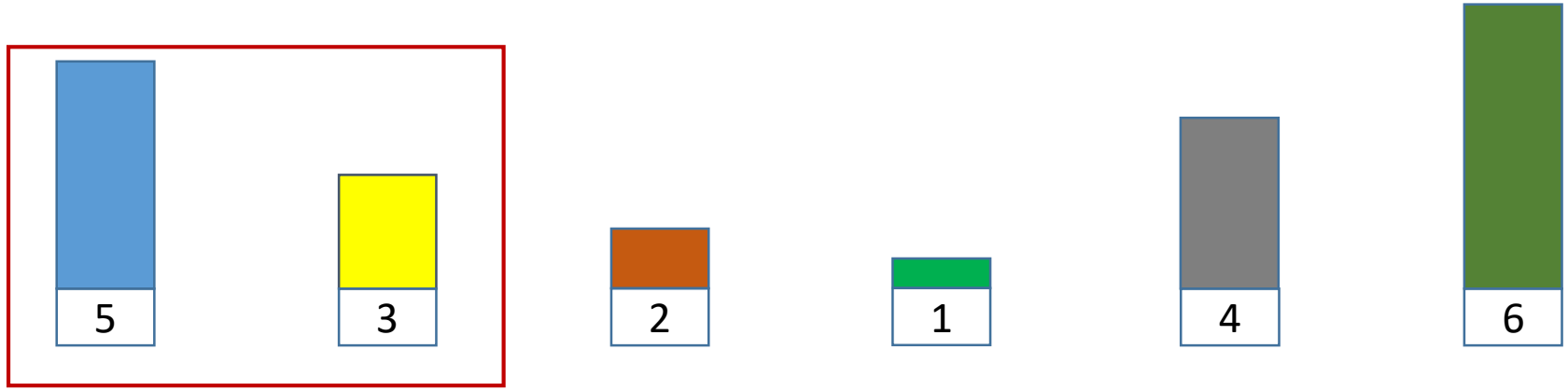
ARREGLOS – Ordenación - Inserción

dim física = 100

dim lógica=6

5	3	2	1	4	6				

Primera vuelta (toma el segundo valor y busca su lugar entre las posiciones 1 y 2)



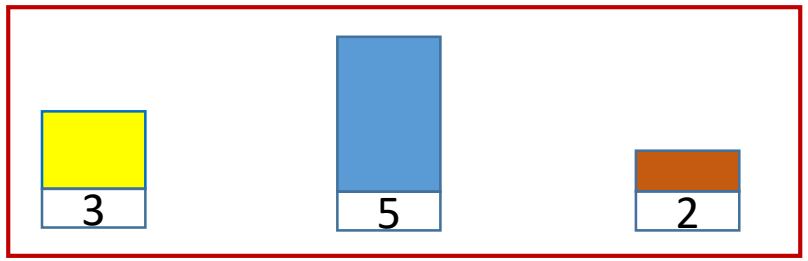
3	5	2	1	4	6				

ARREGLOS – Ordenación - Inserción

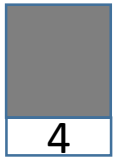
dim física = 100

dim lógica=6

3	5	2	1	4	6				



Segunda vuelta (toma el tercer valor y busca su lugar entre las posiciones 1 y 3)



2	3	5	1	4	6				

ARREGLOS – Ordenación - Inserción

dim física = 100

dim lógica=6

1	2	3	5	4	6				

Tercera vuelta (toma el cuarto valor y busca su lugar entre las posiciones 1 y 4)

1	2	3	5	4	6				

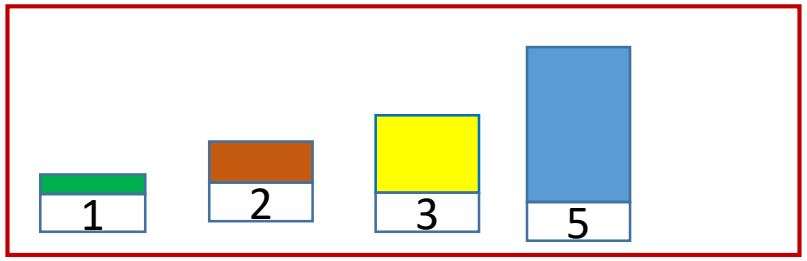
ARREGLOS – Ordenación - Inserción

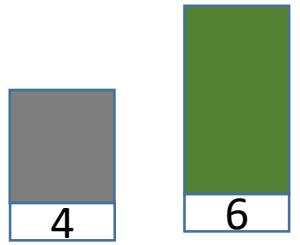
dim física = 100

dim lógica=6

1	2	3	5	4	6				

Cuarta vuelta (toma el quinto valor y busca su lugar entre las posiciones 1 y 5)





1	2	3	4	5	6				

ARREGLOS – Ordenación - Inserción

dim física = 100

dim lógica=6

1	2	3	5	4	6				

Quinta vuelta (toma el sexto valor y busca su lugar entre las posiciones 1 y 6)

1	2	3	4	5	6				

ARREGLOS – Ordenación - Inserción

```
Program ordenar;
Const  maxlen =... {máxima longitud del arreglo}

Type
      TipoElem = ... { tipo de datos del vector }
      Indice   =  0.. maxlen;
      Tvector =  Array  [ 1..maxlen] of TipoElem;
```

ARREGLOS – Ordenación - Inserción

```
Procedure Ordenar ( var v: tVector; dimLog: indice );
var i, j: indice; actual: tipoElem;

begin
 for i:=2 to dimLog do begin
     actual:= v[i];
     j:= i-1;
     while (j > 0) y (v[j] > actual) do
       begin
         v[j+1]:= v[j];
         j:= j - 1;
       end;
     v[j+1]:= actual;
 end;
end;
```

ARREGLOS – Ordenación - Inserción

- No es tan sencillo de implementar.
- El tiempo de ejecución en N^2, siendo N el tamaño del arreglo
- Qué ocurre si los datos están ordenados inicialmente?

Si los datos están ordenados de menor a mayor el algoritmo solo hace comparaciones, por lo tanto es de orden (n).

Si los datos están ordenados de mayor a menor el algoritmo hace todas las comparaciones y todos los intercambios, por lo tanto es de orden (N^2). comparaciones.