APRENDIENDO A USAR VECTORES EN:

Pascal

VECTOR O ARREGLO

Un arreglo (ARRAY) es una estructura de datos compuesta que permite acceder a cada componente por una variable indice, que da la posicion del componente dentro de la estructura de datos.

Todos los elementos del arreglo son del mismo tipo.

Declaracion:

```
type
    nombreTipoArreglo = array[rango] of tipoElemento;
```

Donde rango es `valorInicial .. valorFinal`

> valorInicial y valorFinal tienen que ser ordinales siendo el 1° menor al 2°

Donde tipo es cualquier tipo de dato declarado anteriormente o nativo del lenguaje.

> Es buena practica usar al menos el valorFinal como una constante

IMPORTANTE: antes de acceder a un elemento del arreglo se tiene que validar si el indice a utilizar es uno valido para el rango del arreglo si no se tiene la certeza de que es uno valido.

Acceso

El acceso a un array se hace a traves del indice correspondiente al elemento a acceder.

```
type
    diezEnteros = array[1..10] of integer;
var
    valores : diezEnteros;
begin
    { La primera posicion es 1, al arrancar en 1 el rango del tipo }
        readln(valores[1]);
    { La ultima posicion es 10, al terminar en 10 el rango del tipo }
        readln(valores[10]);
end;
```

Como los indices del arreglo son ordinales se puede utilizar alguna estructura de control iterativa para facilitar el acceso a los elementos

```
{ i es del mismo tipo que el rango del array }
for i:=1 to cantElementos do
    { cualquier operacion que se realice con el elemento }
    arreglo[i]

i:=0;
while( i < cantElementos ) do begin
    i:= i + 1;
    arreglo[i]
end;
```

Dimension fisica y logica

Fisica

Se especifica en el momento de la declaracion y determina su ocupacion maxima de memoria.

La cantidad de memoria total reservada no variara durante la ejecucion del programa.

Logica

Se determina cuando se cargan contenidos a los elementos del arreglo.

Indican la cantidad de posiciones de memoria ocupadas con contenido real. Nunca puede superar la dimension Fisica.

Busqueda

Buscar un elemento en un array consiste en ver si se encuentra en alguna posicion y de ser asi devolver el elemento o posicion.

El array puede estar ordenado o no, dependiendo de esto se puede hacer una busqueda mas eficiente.

> Si esta desordenado si o si hay que verificar cada posicion valida ya que puede ser que exista en cualquiera

```
function busqueda(a:numeros; dimL:integer; num:integer): boolean;
var
    pos:integer;
begin
    pos:=1;
    while((pos <= dimL) and (a[pos] <> num)) do
        pos:= pos + 1;
    buscar:= pos <= dimL;
end;
```

> Si esta ordenado se puede hacer una busqueda hasta encontrarl o que se deje de cumplir que la posicion sea valida ya que el criterio de orden establece que no se puede encontrar mas alla.

```
function busqueda(a:numeros; dimL:integer; num:integer): boolean;
var
    pos:integer;
begin
    pos:=1;
    while((pos <= dimL) and (a[pos] <> num)) do
        pos:= pos + 1;
    buscar:= (pos <= dimL) and (a[pos] = num);
end;
```

> Si esta ordenado se puede usar un algoritmo mas eficiente llamado busqueda dicotomica, el cual consiste en ir partiendo a mitades de las cuales se elige la que deberia contener el valor buscado hasta encontrar el valor si es que existe.

```
function busquedaBinaria(vec:numeros; dimL:integer; buscado:integer): boolean;
var
    ini, fin, medio:integer;
begin
    ini:= 1;
    fin:= dimL;
    medio:= (ini + fin) div 2;
    while (ini <= fin) and (buscado <> vec[medio]) do begin
        if (buscado < vec[medio]) then
            fin:= medio - 1;
        else
            ini:= medio + 1;
        medio:= (ini + fin) div 2;
    end;
    busquedaBinaria:= (ini <= fin) and (buscado = vec[medio]);
end;
```

Eliminacion

Para eliminar un elemento del array hay que hacer el corrimiento de los elementos que lo precedan de manera que el vector solo quede con elementos validos y en el orden que tenian.

```
procedure borrar(var a:numeros; var dimL:integer; var exito:boolean; pos:integer);
begin
    exito:= false;
    if ((pos >= 1) and (pos <= dimL)) then begin
        dimL:= dimL + 1;
        for i:=pos to dimL do
            a[i]:= a[i+1];
        exito:=true;
    end;
end;
```