11074 - Programación I

División Computación - Departamento de Ciencias Básicas

UNIVERSIDAD NACIONAL DE LUJÁN

OPERACIONES SOBRE ARREGLOS

Arreglo

Un tipo de dato **Arreglo** es una colección ordenada e indexada de elementos, con las siguientes características:

- Todos los elementos son del mismo tipo. Esto convierte a un arreglo en un tipo de datos **homogéneo.**

- Los elementos pueden recuperarse en cualquier orden, simplemente indicando la posición que ocupan dentro de la estructura. Esto se llama **indexación.**

- La memoria ocupada a lo largo de la ejecución del programa es fija; por esto, es una estructura de datos **estática.**

- Asimismo, un arreglo puede identificarse a partir de tres conceptos relacionados:
- **Nombre**: el arreglo posee un nombre asociado a un área de memoria fija, al igual que los otros tipos de datos vistos anteriormente
- **Índice:** que debe pertenecer a un tipo de dato ordinal, el cual permitirá acceder a cada elemento del arreglo
- **Dimensión:** Esta dimensión indica la cantidad de índices necesarios para acceder a un elemento del arreglo. Puede ser que no esté ocupado en su totalidad.

A=

33	2	55	9	30	17	21	23	88	111

El ejemplo anterior es un vector de 10 elementos enteros → longitud del vector = 10. En este caso coincide cantidad de elementos con longitud del vector. Cada elemento es referenciado por su posición dentro del vector, comenzando en 0 para el primer elemento, 1 para el segundo y así sucesivamente. Entonces si al vector anterior lo llamamos A, nos referimos a cada elemento de la siguiente manera: A[0]=33, A[1]=2, A[3]=55,...,A[9]=111

Procesar arrays en C:

Un vector se declara indicando el tipo de dato base y la cantidad de elementos almacenados, según la sintaxis siguiente:

<tipo base> <nombre de la variable arreglo>[<cant. elementos>];

Por ejemplo, una variable arreglo para almacenar 10 enteros se construye según se muestra a continuación: int arreglo_nros[10];

Cuando se trabaja con arreglos, es recomendable declarar un tipo de dato de usuario, antes de la declaración de la variable. En C, los tipos de dato de usuario se construyen utilizando la palabra reservada **typedef**:

```
/* Declaración de un tipo de dato entero */
typedef int t_arreglo[10];
/* Declaración de la variable */
t_arreglo arreglo_nros;
```

Operaciones con vectores

Las operaciones en arreglos pueden clasificarse de la siguiente forma:

- **Escritura**
- **Lectura/Asignación**
- **Actualización**
- **Ordenación**
- **Búsqueda**
- **Mezcla**

Escritura/Lectura-Asignación:

```c
#include <time.h>
#include <stdio.h>
#define N 5
int main()  {
    int i;
    int enteros[N];
/* Inicialización del arreglo con números aleatorios */
    srand(time(NULL));
    for (i=0; i<N; i++) {
        enteros[i] = rand();
        printf("enteros[%d] = %d \n", i, enteros[i]);
    }
}
```

La **inicialización** se lleva a cabo con números generados de manera aleatoria,mediante el uso de las funciones srand() y rand(). La primera permite inicializar la serie de números pseudoaleatorios a partir de una “semilla” recibida como parámetro (en este caso, se utiliza la hora del sistema como tal). Luego, cada posición del vector se inicializa con un número entero pseudoaleatorio generado con llamadas sucesivas a rand().

Para copiar los datos de un vector en otro vector **no podemos** realizar una asignación directa del tipo **v1 = v2**. Esto hay que realizarlo elemento a elemento y para ello debemos utilizar una estructura iterativa. Ejemplo:

```
#include <stdio.h>
void copiar(int size, int origen[size], int destino[size]) {
    int i;
    for(i = 0; i < size; i++)
        destino[i] = origen[i];
}
void main(){
    int i;
    int v1[5] = {3, 8, 18, 23, 33};
    int v2[5];
    copiar(5, v1, v2);
    for(i = 0; i < 5; i++)
        printf("%i ", v2[i]);
}
```

Actualización

Dentro de esta operación general se encuentran las de **modificar, insertar, agregar, eliminar** datos. Para realizar este tipo de operaciones se debe tomar en cuenta si el arreglo está o no ordenado. Ya hemos visto técnicas de búsqueda y ordenamiento en clases anteriores.

Los algoritmos de modificación, inserción, agregado y eliminación son los siguientes:

Modificar

Esta operación es exactamente igual a la asignación. Se puede modificar uno o varios elementos indicando su posición, u otro proceso similar sería modificar los elementos con un determinado valor por otro, en este último caso tendríamos que realizar una búsqueda simultánea.

- Contamos con un arreglo de n elementos, n < = longitud del arreglo y tenemos que reemplazar todos los elementos con **contenido igual** a x por **contenido igual** a k.

```
for(i = 0; i < n; i++) {
    if A[i]=x
        A[i]=k;
}
```

- Contamos con un arreglo de n elementos, n < = longitud del arreglo y tenemos que reemplazar todos **los elementos de posición** par por contenido igual a k.

```
for(i = 0; i < n; i=i+2)
    A[i]=k;
```

Insertar

Contamos con un arreglo de n elementos, n < = longitud del arreglo y tenemos que insertar un elemento x en la posición j. Validando los datos con los que contamos, iniciamos el proceso:

```
#define MAX 100
int A[MAX], n, x;
if (n + 1 <= MAX) {
    for (int i = n; i>j; i--)
        A[i]=A[i-1];
    A[j]=x;
    n=n+1;
}
```

Agregar

Contamos con un arreglo de n elementos, n < = longitud del arreglo y tenemos que agregar nuevo un elemento x. Validando los datos con los que contamos, iniciamos el proceso:

```
#define MAX 100
int A[MAX], n, x;
if (n + 1 <= MAX) {
    A[n]=x;
    n=n+1;
}
else {
    printf("No hay espacio suficiente para agregar el elemento" );
 }
```

Eliminar

Contamos con un arreglo de n elementos, n < = longitud del arreglo y tenemos que eliminar el elemento de la posición x. Entonces el elemento a su derecha debe correrse hacia la izquierda para ocupar ese lugar, al igual que el resto de los elementos hasta el último del vector y disminuye en 1 la cantidad de elementos ocupados:

```
#define MAX 100
int A[MAX], n, x;
for (i=x+1,i< n, i++)
    A[i-1]=A[i];
n=n-1;
```

Mezcla

Vamos a introducir otro concepto, **Mezcla**, que se utiliza para tomar **dos vectores ordenados**, del mismo tipo pero no de la misma longitud necesariamente, y generar un **tercero también ordenado** a partir de ellos.
El procedimiento consiste en tomar el elemento más pequeño entre los 2 vectores y colocarlo en el nuevo, luego el siguiente y así sucesivamente hasta que se agotan los elementos de uno de los vectores, en ese momento se copia el resto del otro vector en el nuevo.
Por ejemplo:

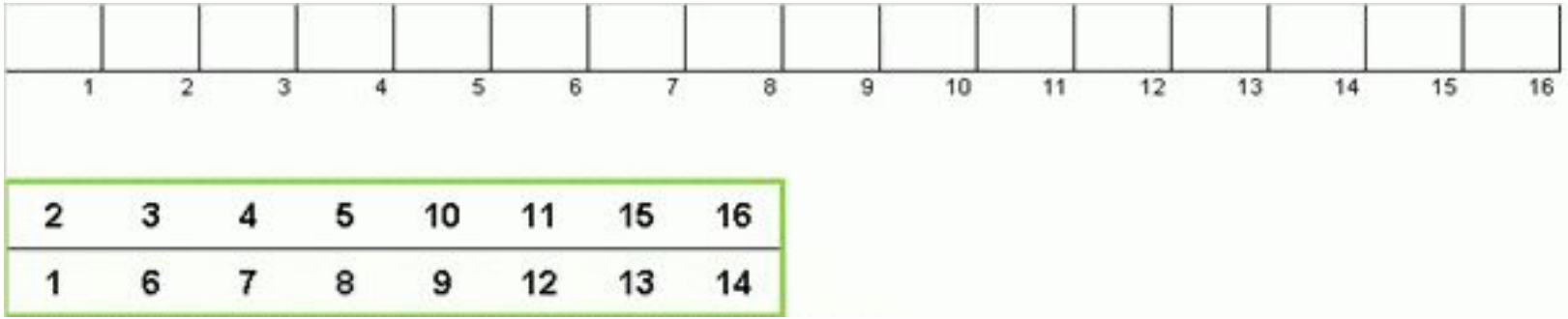
Procedimiento de mezcla

A
2
7
8
9
11
12

AB
2
4
6
7
8
9
10
11
11
12
18

B
4
6
10
11
18

Procedimiento de mezcla



```c
#include <stdio.h>
#define M 4
#define N 5
int main() {
    int arreglo1[] = {2, 3, 5, 8};
    int arreglo2[] = {1, 2, 6, 9, 10};
    int destino[M+N];
    int i, j, k;
    i = 0;
    j = 0;
    k = 0;
```

```
/* Mezcla */
    while ( i < M && j < N ) {
        if ( arreglo1[i] <= arreglo2[j] ) {
            destino[k] = arreglo1[i];
            i++;
        }
        else {
            destino[k] = arreglo2[j];
            j++;
        }
        k++;
    }
}
```

```
    if (i == M)
  /* 'arreglo1' no tiene más elementos, copiar 'arreglo2' */
        for (; j < N; j++) {
            destino[k] = arreglo2[j];
            k++;
        }
    else
        if (j == N)
 /* 'arreglo2' no tiene más elementos, copiar 'arreglo1' */
            for (; i < M; i++)   {
                destino[k] = arreglo1[i];
                k++;
            }
    for (i = 0; i < M+N; i++)
           printf("%i \n", destino[i]);
}
```

"Enseñar no es transferir conocimientos, sino crear las posibilidades de su construcción; quien enseña aprende al enseñar y quien aprende enseña al aprender"

Paulo Freire