11074 - Programación I

División Computación - Departamento de Ciencias Básicas

ARREGLOS DE UNA DIMENSIÓN (VECTORES)

Vectores

Tipo Abstracto de Dato (TAD)

Un **tipo abstracto de datos** (**TAD**) es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos.

Un TAD es un modelo matemático abstracto y teórico (no es un programa, por eso el TAD se llama abstracto). Para que un TAD se pueda usar en la computadora, debemos implementarlo en algún lenguaje de programación.

Clasifica los objetos de un programa y determina los valores que puede tomar y las operaciones que con ellos se puede realizar.

Ejemplo de TAD: El conjunto de los números enteros con sus operaciones (suma, resta, multiplicación, división).

TAD Vector

Conjunto de Datos: Varios elementos del mismo tipo T, posiblemente repetidos, ubicados en un cierto orden lineal.

Notación seq<T>: tipo de las secuencias cuyos elementos son de tipo T.

Colección de operaciones:

- Creación
- Asignación
- Igualdad
- Indexación
- Longitud
- Modificación de un elemento

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.
- La forma gráfica de un vector es considerablemente parecida a la de los strings dado que puede representarse a través de un conjunto de celdas o casilleros compuestos por un valor (que puede ser de uno de los tipos de datos vistos hasta el momento) y un índice determinado.

Vector

33	2	55	9	30	17	21	23	88	111

El ejemplo anterior es un vector de 10 elementos enteros, al que llamaremos A

→ longitud del vector = 10

Nos referimos a cada elemento de la siguiente manera:

A[0]=33, A[1]=2, A[3]=55,...,A[9]=111

Declaración y uso de arreglos lineales

En el caso de **C**, un vector se declara indicando el tipo de dato de 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 caracteres se construye según se muestra a continuación:

char arreglo_letras[10];

Se puede acceder a cada una de las posiciones tanto para lectura como para escritura, y para esto es necesario indicar el subíndice:

```
char arreglo_letras[10];
 char letra;
 /* Escritura de la letra 'A' en la primera posición */
 arreglo_letras[0] = 'A';/* Lectura de la primera posición
```

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 arreglo */
 typedef char t_arreglo[10];
/* Declaración de la variable */
t_arreglo arreglo_letras;
```

Cuando se declara un tipo de dato de usuario conviene anteponer "t_" al nombre del tipo (o alguna otra convención de nombres: t_arreglo, arreglo_t, tipo_arreglo, tArreglo, etc.). para favorecer la claridad del código y facilitar la distinción entre tipos y variables a lo largo del programa. Esto se aplica a la declaración de tipos en general, no sólo para arreglos.

En la práctica, los arreglos se tratan mediante **estructuras de control repetitivas**, lo que facilita en grado considerable el procesamiento de los datos que almacena, en especial cuando es necesario aplicar una misma operación a todos los elementos del vector.

En el ejemplo siguiente se inicializa un arreglo con valores enteros generados en forma aleatoria, y se buscan los valores máximo y mínimo:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 10
int main()        {
    int enteros[N];
    int max, min, i;
  /* 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]);
    }
```

```
 /* Búsqueda del máximo y el mínimo */
    max = enteros[0]; min = enteros[0];
    for (i=1; i<N; i++) {
        if (enteros[i] > max) max = enteros[i];
        if (enteros[i] < min) min = enteros[i];
    }
    printf("Valor máximo = %d\n", max); printf("Valor mínimo = %d\n", min);
    return 0;
}
```

En este ejemplo se distinguen la parte de inicialización del arreglo y la parte de procesamiento de él. 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().

"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