11074 - Programación I

División Computación - Departamento de Ciencias Básicas

UNIVERSIDAD NACIONAL DE LUJÁN

ARREGLOS COMO PARÁMETROS DE FUNCIONES

Arreglos como parámetros de funciones

En la llamada a una función o a un procedimiento se establece una lista (opcional) de parámetros que constituyen los datos de entrada a la función. Ya vimos anteriormente, que el pasaje de parámetros se ilustró utilizando datos simples (números enteros o reales, caracteres, datos booleanos). Sin embargo, lenguajes como C también soportan el pasaje de arreglos lineales y multidimensionales como parámetros.

En el caso de **C**, el pasaje de arreglos como parámetros debe explicarse con detenimiento y se recomienda especial atención por parte del estudiante. En este lenguaje, el nombre de un arreglo es **una etiqueta para la dirección de memoria del primer elemento** del mismo. Indicar que el nombre del arreglo y la dirección de memoria del primer elemento son equivalentes e implica que, en un pasaje de parámetros, en verdad se pasa la dirección del primer elemento (un puntero) y no el contenido del arreglo. Este pasaje de referencia permite que la función altere el contenido del arreglo recibido y estos cambios se percibirán en el ámbito de quién llamó a la subrutina.

Veamos un ejemplo ilustrativo con un **arreglo unidimensional**:

```c
#include <stdio.h>
#define N 5
/* Prototipos */
void cargar_arreglo(int numeros[], int n);
float promediar(int numeros[], int n);

int main() {
    int numeros[N];
    cargar_arreglo(numeros, N);
    printf("Promedio = %.2f\n", promediar(numeros, N) );
    return 0;
}
void cargar_arreglo(int numeros[], int n) {
    int i;
    for (i=0; i<n; i++) {
        printf("Ingrese un entero: ");
        scanf("%i", &numeros[i]);
    }
}
```

```
float promediar(int numeros[], int n) {
    int i, sumatoria = 0;
    for (i=0; i<n; i++)
        sumatoria += numeros[i];
    return (float)sumatoria / n;
}
```

Este programa sencillo carga un arreglo de números enteros con valores ingresados por el usuario, para luego calcular el promedio de todos ellos. La carga de datos se lleva a cabo en la función **cargar_arreglo()** que, como primer parámetro, recibe el arreglo en cuestión (o la dirección de memoria del primer elemento). Esta **función modifica el vector**, escribiendo un nuevo dato en cada posición, y este cambio es “visible” desde la función main() debido a que el **pasaje es por referencia** (tanto main() como cargar_arreglo() acceden a las mismas posiciones en memoria principal, por estar trabajando con las mismas direcciones de memoria).

Asimismo, no se especifica el tamaño del arreglo en el listado de parámetros formales de cada función, que también es una consecuencia de lo que se acaba de explicar. **El pasaje de un arreglo como parámetro es, en la práctica, el pasaje de la dirección de memoria de su primer elemento;** entonces, la rutina recibe un dato simple, atómico, escalar: **Una dirección de memoria**. Por lo tanto, escribir **"int numeros[ ]"** es apenas una “formalidad”, y el pasaje es en sí el de un dato simple (un número entero que es una dirección de memoria) y no, como podría suponerse, el contenido de todo el arreglo. También, como otra consecuencia de lo mismo, que **es necesario indicar a la función la longitud del arreglo que se está pasando por parámetro**; de otra forma, la función no tendría medios para determinar cuáles son los límites del vector (sabe dónde comienza, porque está recibiendo la dirección del primer elemento; sin embargo, no sabe dónde termina).

Si el pasaje de un arreglo como parámetro de una función es, en verdad, el pasaje de una dirección de memoria, entonces el fragmento de código

```
void cargar_arreglo(int numeros[], int n)
```

puede reescribirse como se muestra a continuación:

```
void cargar_arreglo(int* numeros, int n)
```

En el segundo caso se utiliza nomenclatura de punteros; el signo **"*"** significa que el parámetro numeros es un **puntero** a un entero (**una variable A es un puntero cuando almacena la dirección de memoria de un dato B; entonces, se indica que la variable A "apunta" a B**). Ambas formas (usando **"[ ]" o "*"**) **son equivalentes** . Por extensión puede deducirse el pasaje de un **arreglo multidimensional** como parámetro de una rutina. Examinemos el ejemplo siguiente, donde la función **cargar_matriz()** permite cargar una matriz bidimensional con números:

```
#include <stdio.h>
int const M=3;
int const N=3;
void cargar_matriz(int matriz[][N], int filas, int columnas);
int main() {
    int matriz_numeros[M][N];
    cargar_matriz(matriz_numeros, M, N);
    return 0;
}
```

```
void cargar_matriz(int matriz[][N], int filas, int columnas) {
    int i,j;
    for (i=0; i<filas; i++) {
        for (j=0; j<columnas; j++) {
            printf("\n Ingrese un numero: ");
            scanf("%d", &matriz[i][j]);
        }
    }
}
```

La construcción **"int matriz[ ][N]"** indica que el parámetro matriz es un arreglo de dos dimensiones cuyos elementos son enteros. Tal vez se esperaba "**int matriz[ ][ ]**". Esta cuestión puede aclararse si se interpreta la matriz bidimensional como si se tratase de un arreglo lineal donde cada elemento es otro vector de N caracteres. Cuando se pasa un arreglo como parámetro, **se exige especificar el tipo de dato base** (o sea, el tipo de dato de cada elemento del arreglo)

Cadenas de caracteres

Un arreglo de caracteres es el recurso que ofrece **C** para trabajar con cadenas desde un programa. Cuando se declara:

char cadena[30];

se entiende que cadena es un arreglo lineal de caracteres de 30 bytes de tamaño (30 x 1 byte, siendo 1 byte el tamaño de cada carácter) capaz de alojar 29 caracteres y el carácter especial **'\0'** de “fin de cadena”. Por ejemplo, si se declara "char palabra[5]", podrían almacenarse hasta 4 caracteres además de '\0':

‘H’	**‘O’**	**‘L’**	**‘A’**	**‘\0’**

Así, en el lenguaje **C**, una cadena es un arreglo de caracteres sin particularidad especial alguna, salvo la existencia del carácter **'\0'**, que permite aplicar sobre estos arreglos un conjunto de funciones para manipulación de cadenas, que se encuentran en la unidad de biblioteca string.h.

Para que estas funciones puedan aplicarse, es fundamental delimitar el fin de una cadena con el carácter **'\0'**. De otra forma, los resultados obtenidos serían indefinidos.

A continuación se muestra un ejemplo simple de uso de la **función strlen()** (retorna la longitud (cantidad de caracteres) del arreglo (incluido \0'))

```
char palabra[] = {'H','o','l','a','\0'};

printf("%d\n", strlen(palabra)); /* Devuelve 5 */
```

Ya que una cadena es un arreglo lineal de caracteres, es posible acceder a ellos mediante

la sintaxis de arreglos:

```
palabra[0] = 'H';

palabra[1] = 'o';

...
```

Enumeraciones

El tipo enumerado permite construir datos con propiedades ordinales. La característica de este tipo de dato es que sólo puede tomar los valores de su enumeración. A continuación se presenta un ejemplo de declaración y uso de un tipo enumerado en **C**:

```c
#include <stdio.h>
typedef enum {bicicleta, moto, coche} t_transporte;
int main() {
    t_transporte vehiculo;
    vehiculo = moto;
    switch (vehiculo) {
        case bicicleta:
        printf("El vehículo es bicicleta.\n"); break;
        case moto:
        printf("El vehículo es moto.\n"); break;
        case coche:
        printf("El vehículo es coche.\n"); break;
        default:
        printf("El tipo de vehículo no existe!\n");
        }
    return 0;
}
```

"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