

Laboratorio de Arquitectura de Redes

Vectores y cadenas en lenguaje C

Vectores y cadenas en lenguaje C

- Vectores
- Declaración de vectores
 - Vectores unidimensionales
 - Vectores multidimensionales
- Inicialización de vectores
- Cadenas
 - Funciones que utilizan cadenas
- vectores de cadenas de caracteres

Vectores

- Un *vector* es un tipo de variable especial que permite almacenar un conjunto de datos del mismo tipo
- Los elementos de un vector pueden ser referenciados de forma independiente mediante el nombre del vector seguido por los índices necesarios entre corchetes «[]»
 - El elemento que ocupa la primera posición tiene como índice cero.
 - El índice puede representarse con cualquier expresión cuyo resultado sea un número entero positivo
- Pueden ser
 - Unidimensionales
 - Multidimensionales
- Los elementos de un vector se almacenan en posiciones consecutivas de memoria (el primer elemento en la dirección más baja)
- En lenguaje C, los límites en la comprobación del tamaño y dimensiones de los vectores es responsabilidad del programador

Declaración de un vector (I)

- ❑ La declaración de un vector supone *reservar* memoria para sus elementos
- ❑ Declaración de vectores unidimensionales

```
tipodato nombrearray[tamaño];
```

 - `tipodato` representa el tipo de datos de los elementos del vector (cualquiera excepto `void`).
 - `nombrearray` es el identificador del vector y de sus elementos
 - `tamaño` representa un valor entero y constante que indica el número de elementos del vector.
 - ❑ Si el número de elementos es n , el primer elemento es el `nombrearray[0]` y el último es `nombrearray[n-1]`
 - ❑ Puede no indicarse el tamaño si se inicializan los valores en la declaración o si está declarado ya en otro punto del programa (parámetros de una función)
- ❑ La cantidad de memoria que se asigna a un vector es:
 $N^{\circ} \text{ de bytes} = \text{tamaño} * \text{sizeof}(\text{tipodato})$

Declaración de un vector (II)

□ Declaración de un vector multidimensional

```
tipodato nombreakarray[tamaño1][tamaño2]...[tamañoN];
```

- tamaño1, tamaño2, tamañoN son expresiones constantes enteras
 - El número de ellas determina el número de dimensiones del vector.
 - Cada una determinan el tamaño de cada dimensión
 - Cada dimensión necesita un índice para hacer referencia a una posición.
- Los elementos del vector se almacenan de forma consecutiva, siendo el índice más a la derecha el que más rápido cambia
- La cantidad de memoria que se reserva para el vector es:
 $N^\circ \text{ de bytes} = \text{tamaño1} * \text{tamaño2} * \dots * \text{tamañoN} * \text{sizeof}(\text{tipodato})$

Declaración de un vector (III)

- Los vectores bidimensionales se llaman también *tablas* y sus dos dimensiones se llaman **filas** y **columnas**:

```
tipodato nombrearrray[numfilas][numcolumnas];
```

- numfilas indica el número de filas

- numcolumnas indica el número de columnas

- Los elementos de una tabla se almacenan en memoria de forma consecutiva por filas.

- Ejemplos:

```
int lista[10]; /* Vector de 10 enteros */
```

```
char vocales[5]; /* Vector de 5 letras */
```

```
float matriz[6][4]; /* Tabla de 6 filas y 5  
                    columnas de números  
                    reales */
```

Inicialización de vectores (I)

- Cuando se declara un vector sólo se inicializa con cero si se trata de una variable global. En caso contrario su contenido inicial será basura
- Forma general de inicialización de un vector en su declaración

```
tipodato nombrearray[tam1]...[tamN]={listavalores};
```

- *listavalores* es una relación de constantes del tipo declarado para el vector entre llaves «{}» y separadas por comas

- Si un vector se inicializa en su declaración, se permite omitir el tamaño de su primera dimensión (la más a la izquierda: tam1):

- En los unidimensionales:

```
tipodato nombrearray[] = {listavalores};
```

- En los multidimensionales:

```
tipodato nombrearray[][tam2]..[tamN]={listavalores};
```

Inicialización de vectores (II)

- Al inicializar un vector en su declaración debe recordarse que el índice que cambia más rápido es el de la derecha
- Cuando se inicializa un vector en su declaración, se ha de inicializar completo:

```
int digit [10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int impares[] = {1, 3, 5, 7, 9};
int matriz[3][4] = {00, 01, 02, 03,
                   10, 11, 12, 13,
                   20, 21, 22, 23};
char letras[][5] = {'a', 'b', 'c', 'd', 'e',
                   'f', 'g', 'h', 'i', 'j',
                   'k', 'l', 'm', 'n', 'o',
                   'p', 'q', 'r', 's', 't',
                   'u', 'v', 'x', 'y', 'z'};
```

Inicialización de vectores (III)

- La inicialización de un vector después de su declaración (en tiempo de ejecución) requiere la programación de un bucle por cada una de las dimensiones del vector.
- Para inicializar una matriz de FILxCOL enteros el código necesario será similar la siguiente:

```
int matriz[FIL][COL], f, c; /*Declaraciones */
for(f=0 ; f<FIL ; f++)
    for(c=0 ; c<COL ; c++)
    {
        printf("Introduce el dato [%d][%d]",
                f, c);
        scanf("%d", &matriz[f][c]);
    }
```

Cadenas (I)

- Una *cadena de caracteres*, *string* o **cadena** es un vector unidimensional en el que todos sus elementos son de tipo `char` y el último elemento es el carácter nulo «`'\0'`»

```
char nombrecadena[longcad];
```

- `nombrecadena` es un identificador válido para la cadena completa
- `longcad` es una constante entera que indica el número de elementos de la cadena *incluido el nulo final*.

Cadenas (II)

❑ Inicializaciones en la declaración

```
char nombrecadena[longcad]= "cadena";
```

```
char nombrecadena[longcad]= {listarcaracteres};
```

- ❑ `nombrecadena` es el identificador de la cadena
- ❑ `longcad` es una constante entera que determina el número de caracteres de la cadena **incluido el carácter nulo** (si es menor producirá error y si es mayor se rellenarán con "nulos").
- ❑ `listacaracteres` representa un conjunto de constantes de carácter (entre comillas simples «' '» y separados por comas) que deberá incluir el nulo `'\0'` al final.

Cadenas (III)

- ❑ El carácter nulo final es un limitador acordado
- ❑ Los caracteres de una cadena pueden ser accedidos como elementos de un vector ordinario
- ❑ Ejemplos de inicialización en la declaración:

```
char nombre[6]= "Pedro";
```

```
char apellido[]={ 'R', 'u', 'a', 'n', 'o', '\0' };
```

Cadenas (IV)

- Las cadenas no son un tipo de dato
 - Su inicialización fuera de la declaración requeriría un bucle
 - Existen muchas funciones que facilitan el trabajo con las cadenas o “strings”
- La mayoría de las funciones que facilitan las operaciones con cadenas se encuentran declaradas en
 - `STDIO.H`
 - `STDLIB.H`
 - `STRING.H`

Cadenas. Funciones que utilizan las cadenas de caracteres (I)

- **scanf()** permite leer una cadena de caracteres desde el teclado, con las siguientes condiciones
 - `scanf("%[^\\n]s", cadena);` nos permite leer una cadena completa hasta pulsar return ('\\n')
 - `cadena`, sin corchetes y sin que vaya precedido por el operador «&», es el identificador de la cadena.
 - Es conveniente utilizar `fflush(stdin)` previamente para limpiar el buffer de entrada para eliminar los caracteres no recogidos
- **printf()** permite, utilizando el especificador de formato `%s`, imprimir cadenas de caracteres.
- **gets()** lee una cadena completa y sustituye el return por el caracter nulo al almacenarla.
- **puts()** escribe una cadena completa, sustituyendo previamente el caracter nulo por el return.
- **strcat(cadena1, cadena2)** concatena `cadena2` al final de `cadena1`.

Cadenas. Funciones que utilizan las cadenas de caracteres (II)

- ❑ **strcpy(cadena1, cadena2);** copia cadena2 en cadena1
- ❑ **strcmp(cadena1, cadena2);** compara cadena1 y cadena2
- ❑ **strlen(cadena);** devuelve la longitud de la cadena
- ❑ **strlwr(cadena);** convierte los caracteres de cadena a minúsculas
- ❑ **strupr(cadena);** convierte los caracteres de cadena a mayúsculas
- ❑ **atof(cadena);** convierte la cadena a un decimal en doble precisión equivalente al representado por los dígitos que cadena contiene
- ❑ **atoi(cadena);** convierte la cadena a un entero
- ❑ **atol(cadena);** convierte la cadena a un entero largo
- ❑ **fcvt();** convierte un número real a una cadena de caracteres.

Vectores de cadenas de caracteres

□ Un **vector de cadenas** es un vector bidimensional en el que el índice izquierdo señala el número de cadena y el índice derecho la longitud máxima de las cadenas

```
char nombrearray[numcad][longcad];
```

□ Ejemplo

```
char frases[3][80]= {"Error de lectura",  
                    "Error de escritura",  
                    "Error de acceso" };  
  
/* frases[0] representa la cadena "Error de  
lectura" y puede mostrarse en pantalla  
escribiendo puts(frases[0]; */  
  
/* La longitud de 80 caracteres nos asegura que  
quepan todas las frases, aunque en algunas se  
desaproveche la memoria */
```