



Unidad IV Arreglos y estructuras

M.C. Juan Carlos Olivares Rojas



Agenda

- 4.1 Concepto de arreglo
- 4.2 Manejo de cadenas
- 4.3 Concepto de estructuras
- 4.4 Concepto de unión
- 4.5 Empleo de apuntadores

4.1 Concepto de arreglo

- Un arreglo es una colección de objetos del mismo tipo.
- Los arreglos dependiendo de sus dimensiones pueden ser unidimensionales (vectores), bidimensionales (matriz o tabla) y multidimensionales.
- Los arreglos pueden ser de cualquier tipo básico o creado por el usuario.

Arreglos

- Los arreglos se definen así: tipo nombre[tamaño];
- Por ejemplo: `int arreglo[10];` indica un arreglo de diez elementos enteros. Para acceder a cada posición se utiliza un índice.
- Los índices van desde 0 hasta N-1.

Arreglos

- Por ejemplo: $a[3] = 5$; indica que se le asigna el valor de 5 al cuarto elemento del arreglo.
- En el ejemplo la posición mínima es 0 y la máxima 9.
- Matrices: $\text{float tam}[14][5]$; indica una matriz de $14 \times 5 = 70$ elementos.

Arreglos

- `printf(“%f”, tam[3][2]);` imprime el valor de la cuarta fila y tercera columna.
- El nombre de un arreglo sólo es un puntero al inicio de los datos. Si a esa dirección base se le suma un desplazamiento se pueden recorrer varios elementos.
- `int a[5]; a[3] = 10 → a + 4*sizeof(int);`

4.2 Manejo de cadenas

- Las cadenas son un arreglo de caracteres con un terminador especial (`\0`).
- En C no existe un tipo de datos básico para cadenas especial como en otros lenguajes.
- En java `String` cadena; en C# `string`
- En C/C++ `char cadena[20];`

Cadenas

- Declara una cadena de 20 elementos, de los cuales el último es = \0.
- Por ejemplo la cadena “casa” se representa `cadena[0]='c'`; `cadena[1]='a'`; `cadena[2]='s'`; `cadena[3]='a'`; `cadena[4]='\0'`; por lo que la longitud (tamaño) de la cadena es 5.
- Existen muchas funciones auxiliares para el manejo de cadenas.

Funciones de cadena

- `#include <string.h>`
- `strlen(cad)`; devuelve el tamaño de la cadena
- `strcmp(cad1, cad2)`; compara dos cadenas y determina si son iguales (0), `cad1` es mayor que `cad2` (>0) o `cad2` es mayor que `cad1` (<0)
- `strcpy(cad1, cad2)`; copia el contenido de `cad2` en `cad1` (similar a `cad1=cad2`);

Cadenas

- Las cadenas se pueden definir como punteros. Ejemplo `char *cadena="hola";` sólo que en algunos compiladores marca advertencia.
- Otras funciones como `strtok()` permiten encontrar subcadenas. En muchas ocasiones las cadenas son utilizadas para crear buffers en procesos auxiliares

4.3 Concepto de estructuras

- Una estructura es un tipo de datos que puede contener otros tipos de datos, los cuales se manejan como si fuera una unidad.
- El tamaño de una estructura depende del tamaño de todos los atributos.
- Una estructura sirve para almacenar y organizar la información de mejor manera.

Estructura

- Las estructuras se declaran de la siguiente forma

```
struct pacientes
{
    char nombre[40];
    unsigned short edad;
    float consulta;
} paciente;
```

Estructuras

- La cual declara un tipo de datos pacientes la cual tiene tres elementos nombre del paciente de tipo carácter, edad de tipo entero corto sin signo y finalmente el precio de la consulta en tipo flotante. Posteriormente se define una variable paciente de ese tipo.
- O bien se pueden declarar dentro de un bloque de instrucciones como: `struct pacientes paciente;`

Estructuras

- Para acceder a los elementos de una estructura se utiliza el operador '.' Por ejemplo:
- `printf(“%s”, paciente.nombre);`
- `paciente.edad = 10;`
- `paciente.consulta = 120;`
- Las estructuras se pasan por valor.

Estructuras

- Se pueden utilizar punteros a estructuras para el mejor manejo de información. Por ejemplo `struct pacientes *pac`
- Para acceder a los elementos de una estructura a través de un puntero se utiliza el operador flecha `-->`
- `pac-->edad = 11;`

Estructuras

- También se pueden declarar arreglos de estructuras y tener elementos estructuras dentro de una estructura.
- Las estructuras son muy utilizadas en programación.
- El operador typedef abrevia tipos de datos. Por ejemplo typedef pacientes P. Permitiria declarar P paciente;

4.4 Concepto de unión

- La unión es un tipo de datos compuesto muy parecido a la estructura. La única diferencia es que la unión sólo puede conetener un elemento a la vez pero permite cambiar de elemento cuando se requiera.
- Las uniones se utilizar para guardar algunos valores temporales y luego convertirlos

Unión

- union pacientes
 - {
 - char nombre[40];
 - unsigned short edad;
 - float consulta;
 - } paciente;
-
- El tamaño de una unión es el tamaño del atributo más grande. `sizeof(paciente)=40`

4.5 Empleo de apuntadores

- Los apuntadores se utilizan para manejar datos directamente desde la memoria (controladores, juegos, aplicaciones de alto rendimiento).
- Se suelen utilizar para pasar datos por referencia, pero en general aplican para cualquier acción que se quiera realizar.

Empleo de apuntadores

- Los apuntadores también se emplean cuando se utiliza memoria dinámica.
- Otro uso de los apuntadores consiste en maneja información que de alguna forma no se conoce por completo su estructura o tipo de datos.
- Un puntero void puede hacer referencia a cualquier tipo de dato. Ejemplo: `void *p = (void *) f;`

¿Preguntas?

