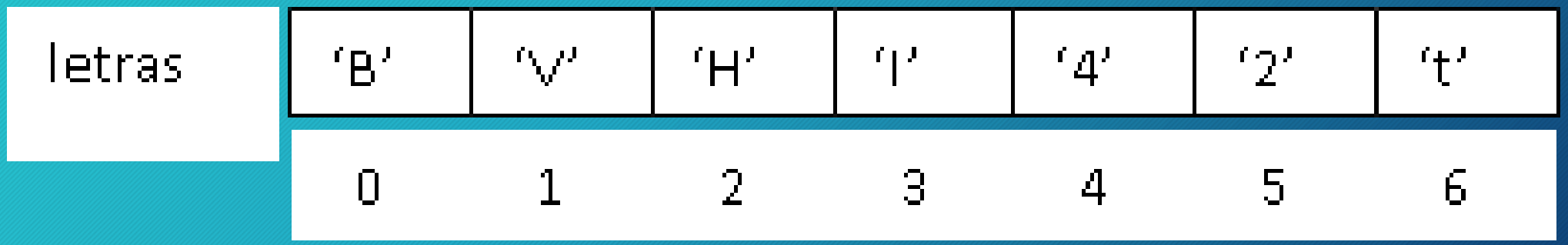
Arreglos y Estructuras en C

PROGRAMACIÓN I
Otoño 2022
NRC: 10844

Dr. Mario Rossainz López
Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla

Arreglos en C

- **Arreglos lineales, unidimensionales o vectores:**
- Estructura de datos en donde podemos almacenar un conjunto de valores o datos y no necesariamente uno sólo, todos del mismo tipo, identificados mediante un índice representado con un valor entero.



Arreglos en C

- Todo arreglo en principio es finito y tiene un tamaño, es decir, tiene un límite de almacenamiento.
- Si un arreglo almacenará datos de tipo entero entonces tendrá que ser declarado y definido como un arreglo de enteros.
- Si por el contrario el arreglo almacenará caracteres, entonces tendrá que ser definido como un arreglo de caracteres, etc.
- No podemos tener, por tanto, un arreglo que almacene en algunas posiciones de él números enteros y en otros caracteres porque tendremos un problema de incompatibilidad de tipos pues el arreglo es declarado como de tipo entero o de tipo carácter pero no de ambos.

Arreglos en C

CARACTERÍSTICAS:

- **Es una estructura homogénea:** Es decir, todos los elementos almacenados en el arreglo son del mismo tipo de datos.

- **Es una estructura lineal de acceso directo:** Significa que podemos acceder a los datos almacenados en el arreglo de manera directa a través de su posición mediante el índice.

- **Es una estructura estática:** El arreglo es finito, es decir, tiene un tamaño y es de un determinado tipo y se define en tiempo de compilación. Una vez creado el arreglo éste no cambia de tamaño durante la ejecución del programa debido a que en el momento de compilar el programa, el compilador reserva antes de la ejecución del mismo las casillas necesarias de memoria que el arreglo requiere para almacenar los datos.

Arreglos en C

DECLARACIÓN Y USO:

```
<tipo_de_dato_base> nombreArreglo [<tamaño>];
```

Ejemplo:

```
double sueldos_maestros[10];
double salario= 1500.00;
sueldos_maestros[4]=salario;  //Escritura
sueldos_maestros[0]=126.34;
double valor= sueldos_maestros[4]; //Lectura
```

Arreglos en C

DECLARACIÓN DE TIPOS:

```
typedef double t_miArreglo[10];

t_miArreglo sueldos_maestros;
```

Arreglos en C

En la práctica, los arreglos se trabajan mediante ciclos, lo cual facilita el procesamiento de los datos almacenados en él, especialmente cuando se aplica una misma operación a todos los elementos del arreglo.

Arreglos en C

El siguiente programa inicializa un arreglo con valores enteros generados de manera aleatoria y a continuación se buscan los valores máximo y mínimo en él.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define TAM 10

int main()
   {
    int enteros[TAM];
    int max, min, i;

    srand(time(NULL));
    for(i=0;i<TAM;i++)
      {
        enteros[i]=rand();
        printf("enteros[%d]= %d\n",i,enteros[i]);
      }

    min=enteros[0];
    max=enteros[0];

    for(i=1;i<TAM;i++)
      {
       if(enteros[i]>max)
           max= enteros[i];

       if(enteros[i]<min)
           min=enteros[i];
      }

    printf("Valor maximo = %d\n",max);
    printf("Valor minimo = %d\n",min);

    return 0;
   }
```

Arreglos en C [cadenas de caracteres]

MANEJO DE CADENAS:

Para trabajar con cadenas desde un programa en C, éste proporciona como recurso los arreglos de caracteres. Cuando hacemos la siguiente declaración:

char cadena[30];

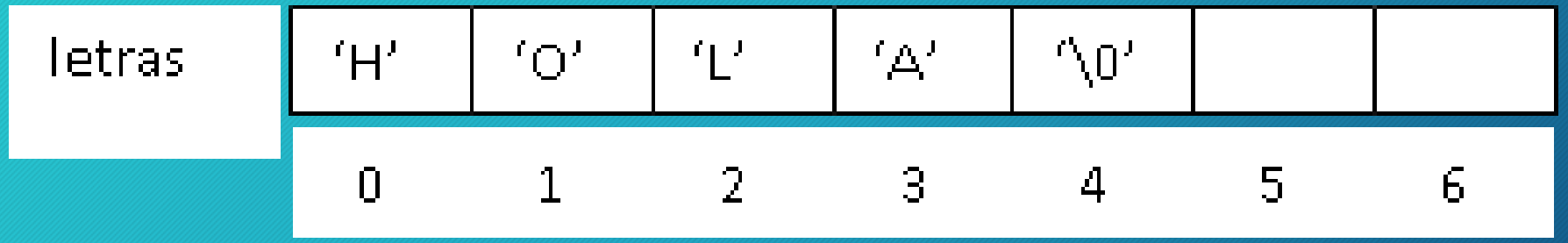
entendemos que estamos declarando un arreglo unidimensional de caracteres de 30 bytes (30 casillas X 1 byte), siendo 1 byte el tamaño de cada caracter.

El arreglo alojará entonces 29 caracteres, más el carácter especial **'\0'** que representa el **"fin de la cadena"**.

Arreglos en C [cadenas de caracteres]

MANEJO DE CADENAS:

Por ejemplo: El siguiente es un arreglo de caracteres de 7 casillas que tiene almacenada la cadena “hola” constituida de 4 caracteres más el carácter de fin de cadena.



Arreglos en C [cadenas de caracteres]

FUNCIONES PARA MANEJO DE CADENAS (<string.h>):

FUNCIÓN	DESCRIPCIÓN
`char* strcpy(char *s1, char *s2)`	Copia s2 en s1 y retorna s1
`char* strncpy(char *s1, char *s2, int n)`	Copia hasta n caracteres de s2 en s1 y retorna s1
`char* strcat(char *s1, char *s2)`	Concatena s2 a s1 y regresa s1
`char* strncat(char *s1, char *s2, int n)`	Concatena hasta n caracteres de s2 a s1 y regresa s1
`int strcmp(char *s1, char *s2)`	Compara s1 y s2. Regresa 0 si son iguales, un entero negativo si s1 es menor o un entero positive si s1 es mayor
`char* strncmp(char *s1, char *s2, int n)`	Lo mismo que la anterior pero sólo se consideran los primeros n caracteres de ambas cadenas
`char* strchr(char *s, int c)`	Busca el character c en s y regresa un apuntador a él si se encuentra, o NULL en caso contrario
`Char* strstr(char *s1, char *s2)`	Busca la cadena s2 en s1 y regresa un apuntador a ella si se encuentra, o NULL en caso contrario
`int strlen(char *s)`	Regresa la longitud (cantidad de caracteres) del arreglo s sin contar '\0'

Arreglos en C [cadenas de caracteres]

```
#include <stdio.h>
#include <string.h>

int main()
   {
    char cadena1[50]={'H','o','l','a','\0'};
    char cadena2[50]={'M','U','N','D','O','\0'};
    int valor;

    printf("La cadena %s tiene %d caracteres...\n",
            cadena1,strlen(cadena1));
    printf("La cadena %s tiene %d caracteres...\n",
            cadena2,strlen(cadena2));
    strcat(cadena1,cadena2);
    printf("La cadena %s tiene %d caracteres...\n",
            cadena1,strlen(cadena1));

    valor=strcmp(cadena1,cadena2);

    if (valor==0)
       printf("%d: Las cadenas %s y %s son iguales...\n",
               valor,cadena1,cadena2);
    else  if (valor>0)
                  printf("%d: Las cadena %s es mayor que la cadena
                          %s...\n",valor,cadena1,cadena2);
          else printf("%d Las cadena %s es menor que la cadena
                       %s...\n",valor,cadena1,cadena2);

    system("pause");
    return 1;
   }
```

Arreglos en C [cadenas de caracteres]

ARCHIVO DE ENCABEZADO <ctype.h>:

`int isupper(char c)`	c es una letra mayúscula
`int islower(char c)`	c es una letra minúscula
`int isalpha(char c)`	Isupper(c) o islower(c)
`int isdigit(char c)`	c es un dígito
`int isalnum(char c)`	c es alfanumérico
`int isprint(char c)`	c es un carácter imprimible
`int iscontrol(char c)`	c es carácter de control
`int ispunct(char c)`	c es carácter de puntuación

Arreglos en C [cadenas de caracteres]

ARCHIVO DE ENCABEZADO <stdlib.h>:

`double atof(char *s)`	Convierte la cadena s en un flotante
`int atoi(char *s)`	Convierte la cadena s en un entero
`long atol(char *s)`	Convierte la cadena s en un entero long

Arreglos en C [cadenas de caracteres]

LECTURA Y ESCRITURA DE CADENAS:

```
Lectura de un caracter y de una cadena
3XYZ
caracter= 3
cadena= XYZ
--------------------------------------------------
Lectura de un caracter y de una cadena
4    RTRTR
caracter= 4
cadena= RTRTR
--------------------------------------------------
Lectura de un caracter y de una cadena
345 67 8989898
caracter= 3
cadena= 45
```

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
  {
   char car;
   char cadena[10];

   printf("Lectura de un caracter y de una cadena\n");
   scanf("%c%s",&car,cadena);

   printf("caracter= %c\n",car);
   printf("cadena= %s\n",cadena);
   system("pause");
   exit(0);
  }
```

Arreglos en C [cadenas de caracteres]

ARCHIVO DE ENCABEZADO <stdio.h>:

- int sscanf(char *s, char *formato, elementos…);
- int sprintf(char *s, char *formato, elementos…);

char getchar()	Lee y regresa un carácter del flujo estándar de entrada
char putchar(char c)	Que escribe el carácter c en el flujo estándar de salida
char *gets(char *s, int n)	Que lee una cadena de caracteres s, desde el flujo de entrada hasta que se encuentra una línea nueva o que se hayan leido hasta n caracteres. Al final de la cadena se coloca el carácter '\0'
char puts(char *s)	Que escribe la cadena s y el carácter de fin de línea '\n'

Arreglos en C [cadenas de caracteres]

```
int main(int argc, char *argv[])
  {
   char car;
   char cadena[10];

   printf("Lectura de un caracter y de una cadena\n");
   car=getchar();
   gets(cadena);

   printf("\n");
   putchar(car);
   printf("\n");
   puts(cadena);

   printf("\ncaracter= %c\n",car);
   printf("cadena= %s\n",cadena);
   system("pause");
   exit(0);
  }
```

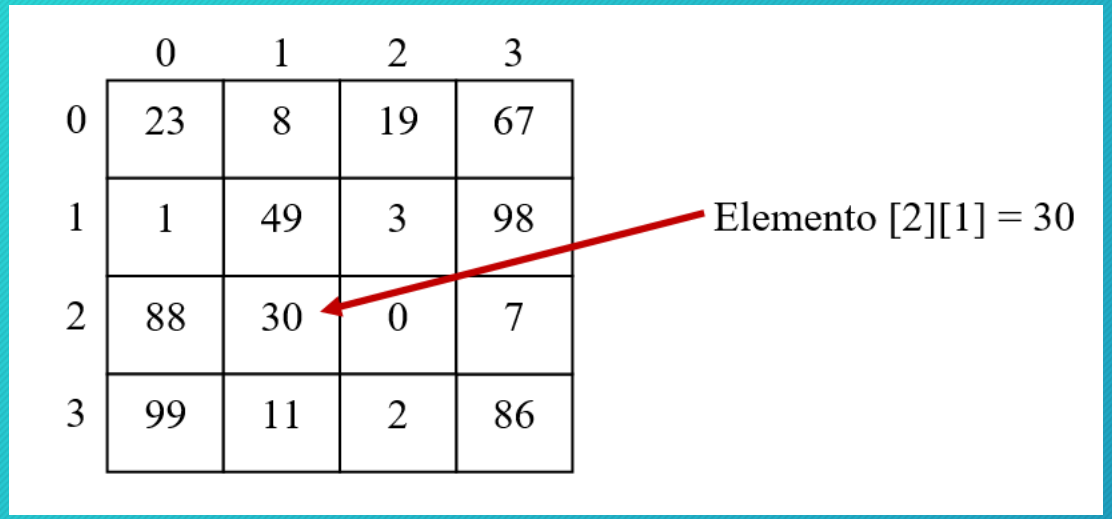
Una salida del programa es:

Lectura de un caracter y de una cadena
345 67 8989898

3
45 67 8989898

caracter= 3
cadena= 45 67 8989898

Arreglos Bidimensionales [Matrices]




```
int enteros[M][N];
int i,j;
for(i=0;i<M;i++)
    {
      for(j=0;j<N;j++)
          {
            Enteros[i][j]=0;
          }
    }
```

Arreglos Bidimensionales [Matrices]

```
int main()
   {
    int i,j;
    int matriz[FILAS][COLUMNAS];

    printf("Llenado de una matriz de %dx%d con valores
            enteros\n",FILAS,COLUMNAS);
    printf("_____________________________________________\n\n");

    for(i=0;i<FILAS;i++)
       for(j=0;j<COLUMNAS;j++)
          {
           printf("Dame un numero entero para almacenarlo en la posicion
                   [%i,%i]: ",i,j);
           scanf("%i",&matriz[i][j]);
          }
    printf("\n");
    for(i=0;i<FILAS;i++)
       {
        for(j=0;j<COLUMNAS;j++)
           printf("%i ",matriz[i][j]);
        printf("\n");
       }
    system("pause");
    return 0;
   }
```

Estructuras [structs]

- Una estructura o **struct** en C es un registro heterogeneo que es capaz de alojar datos de diferentes tipos, por ejemplo nombre y apellido de una persona, su edad, número de cuenta bancaria y domicilio. Cada porción de información en un struct se denomina campo y se puede acceder a el por su nombre.

- La implementación en C de una estructura o registro se lleva a cabo utilizando el tipo de dato ***struct*** el cual agrupa un conjunto de campos los cuales pueden ser variables de tipos básicos o definidos por el usuario.

Estructuras [structs]

```
typedef struct
  {
   char titulo[20];
   char autor[30];
   float precio;
   int edición;
  } t_libro;
. . .
t_libro libro1, libro2;
```

```
struct t_libro
  {
   char titulo[20];
   char autor[30];
   float precio;
   int edición;
  };
. . .
struct t_libro libro1, libro2;
```

Estructuras [structs]

```
typedef struct
  {
   char titulo[20];
   char autor[30];
   float precio;
   int edición;
  } t_libro;
. . .
t_libro libro1, libro2;
```

```
libro1.titulo=”Rayuela”;
strcpy(libro1.autor,”Julio Cortazar”);
libro1.precio=45.00;
libro1.edicion=1;

libro2=libro1;
```

Estructuras [structs] Jerárquicas

```
struct asignatura curso;

curso.nombre=”Programacion I”;
curso.profesor.nombre=”Juan Morales”;
curso.fecha_examen_ord.dia=10;
curso.fecha_examen_ord.mes=5;
curso.fecha_examen_ord.anio=2013;
```

```
typedef struct
   {
    int dia;
    int mes;
    int anio;
   } t_fecha;

struct persona
   {
    char nombre[50];
    char apellido[50];
   } ;

struct asignatura
   {
    char nombre[50];
    int num_hrs;
    struct persona profesor;
    struct persona ayudante;
    char salón[50];
    t_fecha fecha_inicio;
    t_fecha fecha_examen_ord
    t_fecha fecha_asignatura[40];
   };
```

Tablas [Arreglos de Estructuras]

	CAMPOS				
	Nombre	**Apellido**	**Direccion**	**Telefono**	**Edad**
Registro 0	Juan	Pérez	Avda. San Manuel	222-2-11-11-11	38
Registro 1					
Registro 2					

```
directorio[0].nombre = “Juan”;
directorio[0].apellido = “Perez”;
directorio[0].direccion = “San Manuel”;
directorio[0].telefono = “222-2-11-11-11”;
Directorio[0].edad=38;
```

```
typedef struct
  {
   char nombre[20];
   char apellido[30];
   char direccion[40];
   char telefono[15];
   int edad;
  } t_persona;
. . .
t_persona directorio[50];
```