Principios y Herramientas de Programación

Dra. Jessica Andrea Carballido

jac@cs.uns.edu.ar

Dpto. de Ciencias e Ingeniería de la Computación

UNIVERSIDAD NACIONAL DEL SUR

Matrices

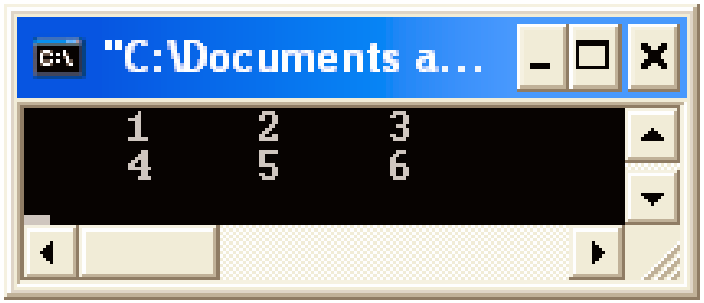
Una matriz es un arreglo bidimensional homogéneo.
Los elementos se acceden por fila y por columna.

```
int m[2][3];

m[0][0] = 1;
m[0][1] = 2;
m[0][2] = 3;
m[1][0] = 4;
m[1][1] = 5;
m[1][2] = 6;
```

Declaración de la variable m como una matriz de 2x3 (2 filas, 3 columnas)

Matrices

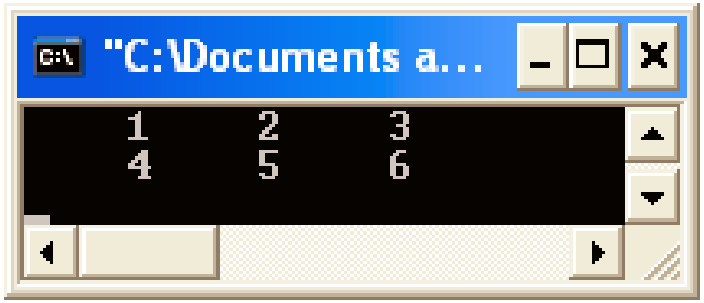
```
#include <stdio.h>
int main()
{
int i, j;  int m[2][3];

m[0][0] = 1; m[0][1] = 2; m[0][2] = 3; m[1][0] = 4; m[1][1] = 5;m[1][2] = 6;
for(i=0; i<2; i++) // i va desde 0 hasta cantidad de filas -1
{
    for(j=0; j<3; j++) //  j va desde 0 hasta cantidad de columnas -1
            printf( "%i  ", m[i][j]);
    printf("\n" );
}
return 0;
}
```



Matrices

```
#include <stdio.h>
#define filas 2
#define cols 3
typedef int tMatriz[filas][cols];

int main()
{
int i, j;  tMatriz m;

m[0][0] = 1; m[0][1] = 2; m[0][2] = 3; m[1][0] = 4; m[1][1] = 5;m[1][2] = 6;
for(i=0; i<2; i++) // i va desde 0 hasta cantidad de filas -1
{
    for(j=0; j<3; j++) //  j va desde 0 hasta cantidad de columnas -1
          printf( "%i  ", m[i][j]);
    printf("\n" );
}
return 0;
}
```

Cuidado!
C es sensible
a las
mayúsculas!



Mostrar matriz

```
#define filas 2
#define cols 3
typedef int tMatriz[filas][cols];
```

COMO Función (primitiva)

```
void imprimir(tMatriz a)
{
    int i, j;
    for( i=0; i<filas; i++)
    {
        for( j=0; j<cols; j++)
            printf( "%i", a[i][j]);
        printf( "\n" );
    }
}
```

Invocación

```
int main(void)
{
    tMatriz m = {{1,2,3}, {4,5,6}};
    imprimir(m);
    return 0;
}
```

Matrices

Problema: Sumar todos los elementos de una matriz

DE: matriz (tmatriz) DS: suma (entero)

```
int sumaElemMat(tMatriz a)
{
    int i, j, suma=0;

    for( i=0; i<filas; i++)
    {
        for( j=0; j<cols; j++)
            suma=suma+a[i][j];
    }
    return(suma);
}
```

```
#define filas 2
#define cols 3
typedef int tMatriz[filas][cols];
```

Problema: Buscar un elemento en una matriz

DE: matriz (tmatriz), elemento (entero) DS: encontre (lógico)

```
bool buscar(tmatriz matriz, int elemento)
{
int i, j; bool encontre;

      encontre=false;
      i=0;
      while (i<filas)&&(encontre==false)
          {
          j=0;
          while (j<cols)&&(encontre==false)
            {
            if (matriz[i][j]==elemento)
                        encontre=true;
            else j++;
            }
          i++;
          }
      return (encontre);
}
```

```
#define filas 2
#define cols 3
typedef int tMatriz[filas][cols];
```

```
int sumaElemMat(tMatriz a)
{
    ...
}
```

```
void imprimir(tMatriz a)
{
    ...
}
```

```
int main(void)
{
Int sumaTodos;
tMatriz m = {{1,2,3}, {4,5,6}};

    imprimir(m);
    sumaTodos = sumaElemMat(m);
    printf("La suma de los elementos de la matriz da %i", sumaTodos);
    return 0;
}
```

Ejercicios

Lenguaje C

- Agregar al programa anterior una primitiva para cargar la matriz por consola (en lugar de inicializarla al momento de declararla con valores constantes).
- Validar el ingreso de cada elemento de la matriz para que sea un entero MAYOR o IGUAL a 0.

Strings

Lenguaje C

- Un *string* es un **arreglo** de caracteres.
- Los *strings* se terminan internamente con un caracter null ➔ “\0”.
- Ejemplo: La cadena “Elefante” se representa en memoria como:

E	l	e	f	a	n	t	e	\0

'H'	'O'	'L'	'A'	' '	'M'	'U'	'N'	'D'	'O'	'\0'

Strings

- Declaración e inicialización

```
char s1[] = "Hello, world!";

/* En este ejemplo no necesito poner la dimensión porque el
/* compilador la calcula a partir de la constante con la que se
/* inicializa.
```

```
char s2[100];

/* Si no la inicializamos, como en este caso,  debemos poner la
/* longitud máxima esperada

printf("Ingrese la cadena:\n");

scanf("%s", s2);

printf("La cadena es %s \n", s2);
```

scanf

La entrada se termina con un **espacio en blanco** y un caracter *null* es guardado al final de la cadena de caracteres.

Strings

- Declaración e inicialización

```
char s2[100];

printf("Ingrese la cadena:\n");
fgets(s2, 100, stdin);
printf("La cadena es %s\n", s2);
```

fgets

La entrada se termina con un **enter** que se almacena junto con la cadena. **stdin** significa que lo lee de teclado.

Strings

- Declaración e inicialización

Definimos un *tCadena*

```
typedef  char tCadena[100];
```

Declaramos variables de este tipo:

```
tCadena string1, string2;
```

'H'	'O'	'L'	'A'	' '	'M'	'U'	'N'	'D'	'O'	'\0'

Strings

- Para cualquier operación que vayamos a llevar a cabo sobre cadenas usaremos las funciones en las LIBRERIAS de C.
  En este caso: **string.h**

No podemos asignar un *string* a otro usando el operador de asignación.

```
#include <string.h>

typedef char tCadena[100];

...

tCadena string1 = "Hello, world! ", string2;

strcpy(string2, string1); /* a string2 le asignamos string1
```

'H'	'O'	'L'	'A'	' '	'M'	'U'	'N'	'D'	'O'	'\0'

Strings

No podemos comparar dos *strings* directamente con los operadores de comparación.

```
tCadena string3 = "this is", string4 = "a test";
if (strcmp(string3, string4) == 0)
        printf("strings are equal\n");
else
        printf("strings are different\n");
```

strcmp retorna 0 si son idénticas, un número negativo si la 1ra es alfabéticamente menor que la 2da,
o un número positivo si la 1ra es mayor que la 2da.

'H'	'O'	'L'	'A'	' '	'M'	'U'	'N'	'D'	'O'	'\0'

Strings

Para concatenar dos *strings*:

```
tCadena string5 = "Hello, ", string6 = "world!";
printf("%s\n", string5);
strcat(string5, string6);
printf("%s\n", string5);
```

Se pegará el contenido de string6 al final de string5.

```
tCadena string7 = "abc";
len = strlen(string7);   /* para obtener la longitud
printf("%d\n", len);                /* muestra 3
```

'H'	'O'	'L'	'A'	' '	'M'	'U'	'N'	'D'	'O'	'\0'

¡No confundir strings (cadenas de caracteres) con caracteres!

```
#include <stdio.h>

int main()
{
char str[] = "Hola alumnos de PHP!";

char uncaracter = 'A';

   printf(" El char es: %c\n El string es: %s\n\n", uncaracter, str);

   uncaracter= "?";  // ES UN ERROR!! Convierte a entero y hace cualquier cosa. NO DA ERROR EL COMPILADOR
   str= "CHAU"; // ES UN ERROR; no se puede asignar CON "="un dato de tipo str DA ERROR EL COMPILADOR

   printf(" El char es: %c\n El string es: %s\n", uncaracter, str);

   return 0;
}
```

'H'	'O'	'L'	'A'	' '	'M'	'U'	'N'	'D'	'O'	'\0'

Estructuras

Lenguaje C

- Una estructura es una colección de una o más variables, de tipos posiblemente diferentes, agrupadas bajo un solo nombre.
- Cada una de esas variables se llama campo.
- Cada campo se accede con el operador “.”.

Definición de Tipos STRUCT

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

Luego puedo definir las siguientes variables:

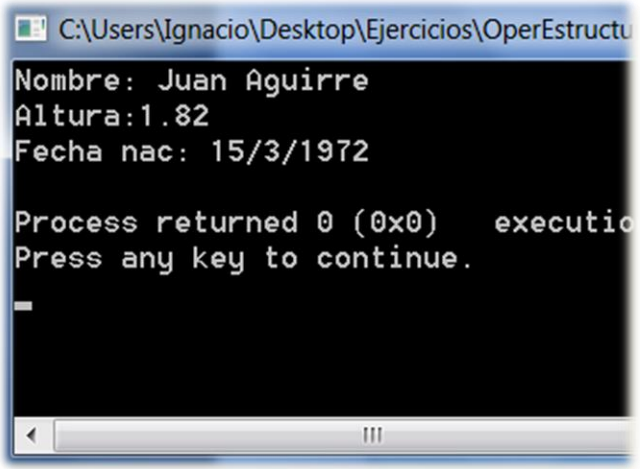
```
tFecha fecha1;
tFecha fecha2;
```

Uso de estructuras


```
#include <stdio.h>
#include <string.h>
typedef struct
{
    int dia;
    int mes;
    int anio;
} tfecha;

typedef struct
{
    char nombre[20];
    float altura;
    tfecha fechaNacimiento;
} tpersona;
```

Uso de estructuras

```
int main()
{
    tpersona jb;
    strcpy( jb.nombre, "Juan Aguirre" );
    jb.altura = 1.82;
    jb.fechaNacimiento.dia = 15;
    jb.fechaNacimiento.mes = 3;
    jb.fechaNacimiento.anio = 1972;
    printf( "Nombre: %s\n Altura:%3.2f\nFecha nac: %i/%i/%i\n",
            jb.nombre, jb.altura, jb.fechaNacimiento.dia,
            jb.fechaNacimiento.mes, jb.fechaNacimiento.anio );
    return 0;
}
```



Ejemplo de productos

- Declarar el tipo *tproducto* (nombre y precio) y *tlista* (lista de productos en un arreglo)
- Leer los datos de un producto
- Armar una lista de productos
- Mostrar el precio de un producto, dado su nombre
- Mostrar toda la lista de productos

```
typedef struct {
    char producto[20];
    float precio;
} tproducto;

typedef tprod tlista[100];
```

Dra. Jessica Andrea Carballido
CONICET - DCIC (UNS)

que tengas un lindo día