	Carátula para entrega de prácticas	Código	
		Versión	02
		Página	1/1
		Sección ISO	
		Fecha de emisión	25 de junio de 2014
Secretaría/División: División de Ingeniería Eléctrica		Área/Departamento: Laboratorios de computación salas A y B	

Laboratorio de computación
salas A y B

Profesor: Claudia Rodríguez Espino

Asignatura: Fundamentos de Programación

Grupo: 1102

No de Práctica(s): 11

Integrante(s): Carrasco Mendoza Jennifer

Semestre: 2018-I

Fecha de entrega: 3-Novienbre-2017

Obervaciones::

CALIFICACIÓN: _____

práctica 11:

Arreglos unidimensionales y multidimensionales

Objetivo:

Reconocer la importancia y utilidad de los arreglos, en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, así como trabajar con arreglos tanto unidimensionales como multidimensionales.

Actividades:

- Elaborar un programa en lenguaje C que emplee arreglos de una dimensión.
- Resolver un problema que requiera el uso de un arreglo de dos dimensiones, a través de un programa en lenguaje C.
- Manipular arreglos a través de índices y apuntadores.

Introducción.

ACTIVIDAD 1:

Código arreglo unidimensional while y for.

```
#include <stdio.h>

/*
Este programa genera un arreglo
unidimensional de 5 elementos y los
accede a cada elemento del arreglo a través
de un ciclo while.
*/

int main (){

#define TAMANO 5
int lista[TAMANO] = {10, 8, 5, 8, 7};

int indice = 0;

printf("\tLista\n");
while (indice < 5 ){
printf("\nCalificación del alumno %d es %d",
indice+1, lista[indice]);
indice += 1; // análogo a indice = indice + 1;
}
printf("\n");
return 0;
}
```

```
#include <stdio.h>

/*
Este programa genera un arreglo
unidimensional de 5 elementos y los
accede a cada elemento del arreglo a través
de un ciclo for.
*/

int main (){

#define TAMANO 5
int lista[TAMANO] = {10, 8, 5, 8, 7};

int indice = 0;

printf("\tLista\n");

for (int indice = 0 ; indice < 5 ; indice++){
printf("\nCalificación del alumno %d es %d",
indice+1, lista[indice]);
}

printf("\n");
return 0;
}
```

```
Lista
Calificacion del alumno 1 es 10
Calificacion del alumno 2 es 8
Calificacion del alumno 3 es 5
Calificacion del alumno 4 es 8
Calificacion del alumno 5 es 7
-----
Process exited after 0.1031 seconds with return value 0
Presione una tecla para continuar . . .
```

ACTIVIDAD 2:

Código apunadores (de tipo carácter).

```
#include <stdio.h>
/*
Este programa crea un apunador de tipo
carácter.
*/
int main () {
    char *ap, c = 'a';
    ap = &c;
    printf("Carácter: %c\n", *ap);
    printf("Código ASCII: %d\n", *ap);
    printf("Dirección de memoria: %d\n", ap);
    return 0;
}
```

```
Caracter: a
Codigo ASCII: 97
Direccion de memoria: 2293319

-----
Process exited after 7.717 seconds with return value 0
Presione una tecla para continuar . . .
```

ACTIVIDAD 3:

Código apunadores (localidades).

```
#include <stdio.h>
/*
Este programa accede a las localidades de memoria de distintas variables a
través de un apunador.
*/
int main () {
    int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
    int *apEnt;
    apEnt = &a;
    printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
    printf("apEnt = &a\n");
    b = *apEnt;
    printf("b = *apEnt \t-> b = %i\n", b);
    b = *apEnt + 1;
    printf("b = *apEnt + 1 \t-> b = %i\n", b);
    *apEnt = 0;
    printf("*apEnt = 0 \t-> a = %i\n", a);
    apEnt = &c[0];
    printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt);
    printf("\tDirección de memoria:\napEnt = &c[0] \t-> apEnt = %i\n", apEnt);
    return 0;
}
```

```
a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}
apEnt = &a
b = *apEnt      -> b = 5
b = *apEnt + 1  -> b = 6
*apEnt = 0      -> a = 0
apEnt = &c[0]    -> apEnt = 5
                Direccion de memoria:
apEnt = &c[0]    -> apEnt = 2293264

-----
Process exited after 7.556 seconds with return value 0
Presione una tecla para continuar . . .
```

ACTIVIDAD 4:
Código apuntadores (aritmética).

```
#include <stdio.h>
/*
Este programa trabaja con aritmética
de apuntadores para acceder a los
valores de un arreglo.
*/
int main () {
    int arr[] = {5, 4, 3, 2, 1};
    int *apArr;
    apArr = arr;
    printf("int arr[] = {5, 4, 3, 2, 1};\n");
    printf("apArr = &arr[0]\n");
    int x = *apArr;
    printf("x = *apArr \t -> x = %d\n", x);
    x = *(apArr+1);
    printf("x = *(apArr+1) \t -> x = %d\n", x);
    x = *(apArr+2);
    printf("x = *(apArr+1) \t -> x = %d\n", x);
    return 0;
}
```

```
int arr[] = {5, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr      -> x = 5
x = *(apArr+1)  -> x = 4
x = *(apArr+1)  -> x = 3

-----
Process exited after 0.1058 seconds with return value 0
Presione una tecla para continuar . . .
```

ACTIVIDAD 5:
Código apuntadores en cadenas.

```
#include <stdio.h>
#include <string.h>
/*
Este programa muestra el manejo de cadenas en lenguaje C.
*/
int main(){
    char palabra[20];
    int i=0;
    printf("Ingrese una palabra: ");
    scanf("%s", palabra);
    printf("La palabra ingresada es: %s\n", palabra);
    for (i = 0 ; i < strlen(palabra) ; i++){
        printf("%c\n", palabra[i]);
    }
    return 0;
}
```

ACTIVIDAD 6:

Código arreglos multidimensionales con punteros.

```
#include <stdio.h>

/* Este programa genera un arreglo de
dos dimensiones (arreglo
multidimensional) y accede a sus
elementos a través de un puntero utilizando
un ciclo for.
*/

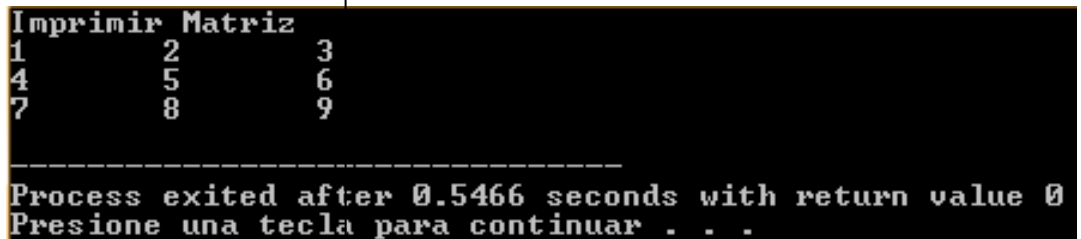
int main(){
    int matriz[3][3] = {1,2,3,4,5,6,7,8,9};
    int i, cont=0, *ap;
    ap = **matriz;

    printf("Imprimir Matriz\n");

    for (i=0 ; i<9 ; i++){
        if (cont == 3){
            printf("\n");
            cont = 0;
        }

        printf("%d\t",*(ap+i));
        cont++;
    }

    printf("\n");
    return 0;
}
```



```
Imprimir Matriz
1      2      3
4      5      6
7      8      9

-----
Process exited after 0.5466 seconds with return value 0
Presione una tecla para continuar . . .
```

ACTIVIDADES

Programa 1:
Suma de matrices.

```
#include <stdio.h>
int matriz1[5][5],a,b, matriz2[5][5], matriz3[5][5];

main()
{
    puts("\t\t\t\t Suma de matrices\n");
    puts("Matriz 1:");
    for(a=1;a<3;a++)
    {
        for(b=1;b<3;b++)
        {
            printf("Dame el elemento [%d,%d]: ",a,b);
            scanf("%d",&matriz1[a][b]);
        }
    }
}
```

```

puts("\nMatriz 2:");
for(a=1;a<3;a++)
{
    for(b=1;b<3;b++)
    {
        printf("Dame el elemento [%d,%d]: ",a,b);
        scanf("%d",&matriz2[a][b]);
    }
}
puts("\nSuma de la matriz 1 con la matriz 2:");
for(a=1;a<3;a++)
{
    for(b=1;b<3;b++)
    {
        matriz3[a][b]=matriz1[a][b]+matriz2[a][b];
        printf("Elemento [%d,%d]: %d\n",a,b,matriz3[a][b]);
    }
}
}

```

```

Suma de matrices
Matriz 1:
Dame el elemento [1,1]: 5
Dame el elemento [1,2]: 6
Dame el elemento [2,1]: 7
Dame el elemento [2,2]: 8
Matriz 2:
Dame el elemento [1,1]: 9
Dame el elemento [1,2]: 10
Dame el elemento [2,1]: 11
Dame el elemento [2,2]: 4
Suma de la matriz 1 con la matriz 2:
Elemento [1,1]: 14
Elemento [1,2]: 16
Elemento [2,1]: 18
Elemento [2,2]: 12
-----
Process exited after 17.6 seconds with return value 0
Presione una tecla para continuar . . .

```

Programa 2:
Matriz por un escalar

```

#include <stdio.h>
#define u 163

int matrizA[5][5], n, a, b, matrizB[5][5];

main()
{
    puts("\t\t\t Matriz de 3x3 por escalar");
    for(a=1;a<4;a++)
    {
        for(b=1; b<4;b++)
        {

```

```

        printf("Dame el elemento [%d,%d] de la matriz A: ",a,b);
        scanf("%d",&matrizA[a][b]);
    }
}
printf("\nIngrese el número que quiera multiplicar la matriz A: ",u);
scanf("%d",&n);

puts("\n\t Matriz por escalar");
for(a=1;a<4;a++)
{
    for(b=1; b<4;b++)
    {
        matrizB[a][b]=(matrizA[a][b]*n);
        printf("Elemento [%d,%d]: %d\n",a,b,matrizB[a][b]);
    }
}
}

```

```

                                Matriz de 3x3 por escalar
Dame el elemento [1,1] de la matriz A: 3
Dame el elemento [1,2] de la matriz A: 4
Dame el elemento [1,3] de la matriz A: 5
Dame el elemento [2,1] de la matriz A: 6
Dame el elemento [2,2] de la matriz A: 7
Dame el elemento [2,3] de la matriz A: 8
Dame el elemento [3,1] de la matriz A: 1
Dame el elemento [3,2] de la matriz A: 2
Dame el elemento [3,3] de la matriz A: 4

Ingrese el número que quiera multiplicar la matriz A: 5

                                Matriz por escalar
Elemento [1,1]: 15
Elemento [1,2]: 20
Elemento [1,3]: 25
Elemento [2,1]: 30
Elemento [2,2]: 35
Elemento [2,3]: 40
Elemento [3,1]: 5
Elemento [3,2]: 10
Elemento [3,3]: 20

```

Programa 3:
Gastos mensuales con promedio anual

```

#include <stdio.h>

int contador;
float mes[15],suma=0,promedio;
const char *meses[15] = {"\0", "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio",
"Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"};
main()
{
    for(contador=1;contador<13; contador++)
    {
        printf("Gastos del mes de %s: $",meses[contador]);
        scanf("%f",&mes[contador]);
        suma=suma+mes[contador];
    }
}

```

```

promedio=suma/12;
printf("\nLos gastos del mes de %s fueron: $%.2f\n",meses[6],mes[6]);
printf("Los gastos del mes de %s fueron: $%.2f\n",meses[12],mes[12]);
printf("\nEl promedio anual de los gastos son: $%.2f",promedio);
}

```

```

Gastos del mes de Enero: $400.50
Gastos del mes de Febrero: $230.90
Gastos del mes de Marzo: $456.87
Gastos del mes de Abril: $435.90
Gastos del mes de Mayo: $567.80
Gastos del mes de Junio: $453.20
Gastos del mes de Julio: $600.50
Gastos del mes de Agosto: $190.89
Gastos del mes de Septiembre: $456.78
Gastos del mes de Octubre: $345.67
Gastos del mes de Noviembre: $900
Gastos del mes de Diciembre: $390.87

Los gastos del mes de Junio fueron: $453.20
Los gastos del mes de Diciembre fueron: $390.87

El promedio anual de los gastos son: $452.49
-----
Process exited after 57.9 seconds with return value 0
Presione una tecla para continuar . . .

```

Conclusiones:

El uso de arreglos permite reducir considerablemente el manejo de distintas variables para un solo tipo de dato; incluso, permite estructurar la información en una sola variable. Trabajar con arreglos, tanto unidimensionales como multidimensionales, conlleva a la comprensión del ordenamiento de los datos.