	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: *M.I. Marco Antonio Martínez Quintana*

Asignatura: *Estructura de Datos y Algoritmos 1*

Grupo: *15*

No de Práctica(s): *1*

Integrante(s): *Citlali Cuahtepitzi Cuatlapantzi*

No. de Equipo de cómputo empleado: *NO APLICA*

No. de Lista o Brigada: *NO APLICA*

Semestre: *2021-2*

Fecha de entrega: *15 de marzo del 2021*

Observaciones:

CALIFICACIÓN: _____

Guía práctica de estudio 01: Aplicaciones de arreglos

OBJETIVO:

Utilizar arreglos unidimensionales y multidimensionales para dar solución a problemas computacionales.

Actividades:
Crear arreglos unidimensionales.
Crear arreglos multidimensionales

INTRODUCCIÓN

Un **arreglo** es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse. Pueden ser unidimensionales o multidimensionales. A cada elemento (dato) del arreglo se le asocia una posición particular. Para acceder a los elementos de un arreglo es necesario utilizar un índice. En lenguaje C, el índice de cada dimensión inicia en 0 y termina en n-1, donde n es el tamaño de la dimensión.

DESARROLLO

Licencia GPL de GNU

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```
/*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program. If not, see <http://www.gnu.org/licenses/>.
*
* Author: Jorge A. Solano
*/
```

Arreglos contiguos o ligados

Un **arreglo contiguo** es aquel que se crea desde el inicio del programa y permanece estático durante toda la ejecución del mismo, es decir, no se puede redimensionar.

Un **arreglo ligado** es aquel que se declara en tiempo de ejecución y bajo demanda, por lo tanto, es posible incrementar su tamaño durante la ejecución del programa, utilizando de manera más eficiente la memoria. Para crear un arreglo ligado se debe utilizar lo que se conoce como **memoria dinámica**.

Los **arreglos unidimensionales** están constituidos por localidades de memoria (ya sea contiguas o ligadas) ordenadas bajo un mismo nombre y sobre un solo nivel (una dimensión).

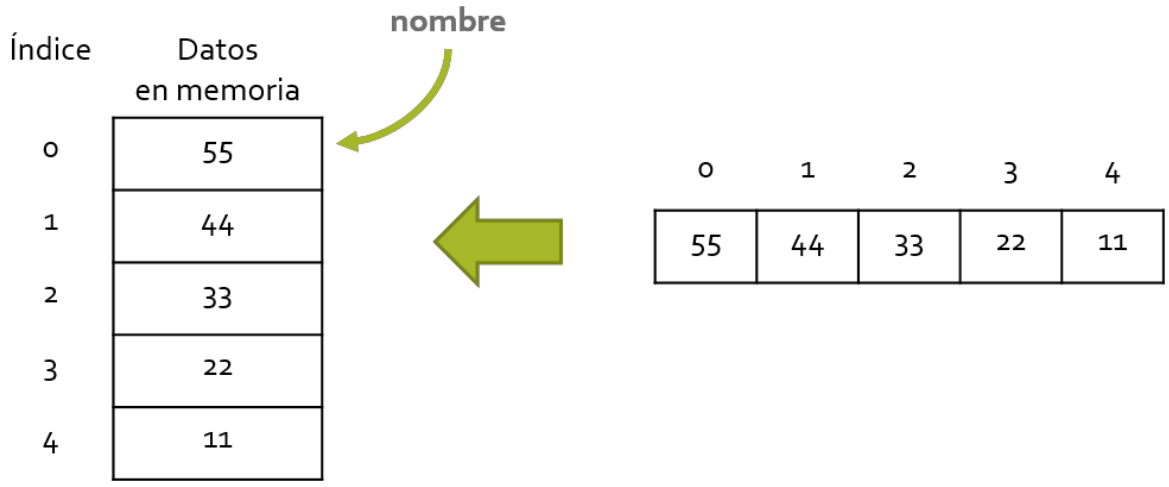


Figura 1. Representación en memoria de un arreglo unidimensional.

Los **arreglos multidimensionales** están constituidos por localidades de memoria (ya sea contiguas o ligadas) ordenadas bajo un mismo nombre y que pueden tener varios niveles (varias dimensiones) que van desde el plano (2 dimensiones) hasta la enésima dimensión.

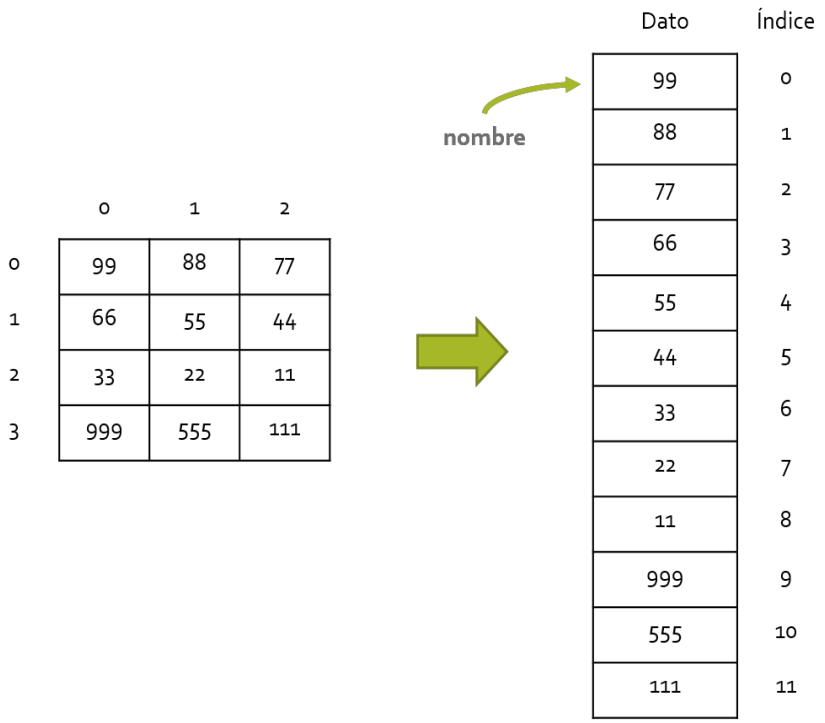


Figura 2. Representación en memoria de un arreglo bidimensional.

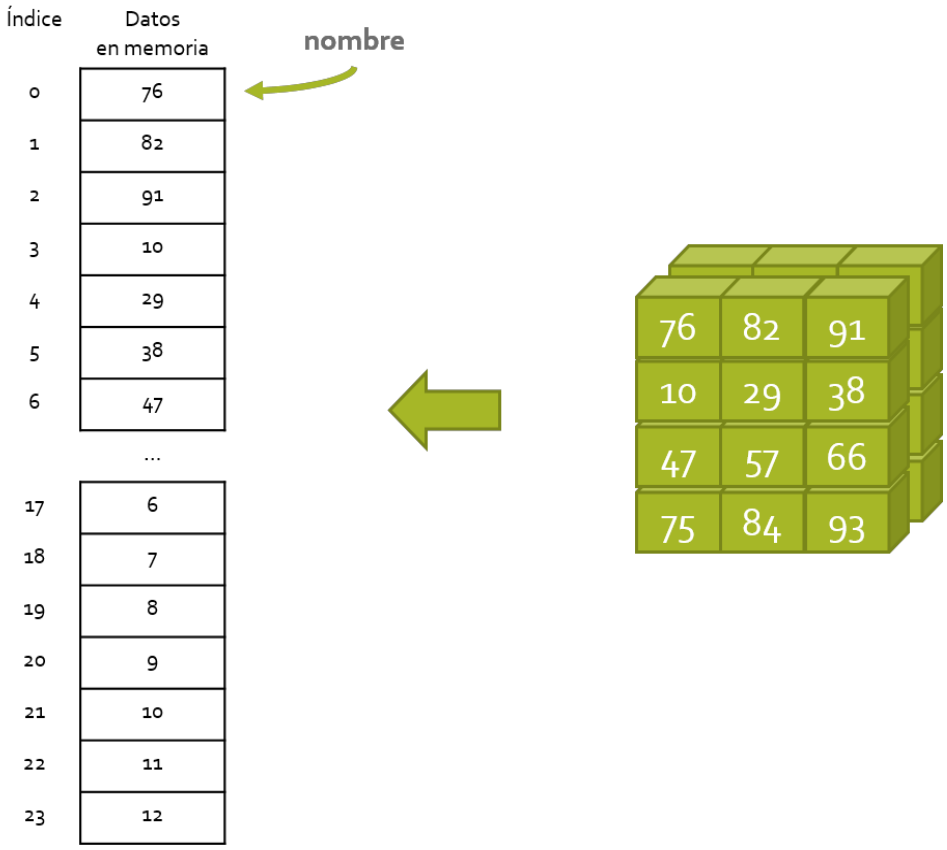


Figura 3. Representación en memoria de un arreglo tridimensional.

Aplicaciones de arreglos

Los arreglos son una herramienta indispensable a la hora de realizar **aplicaciones computacionales**. Si se quiere programar un juego de mesa (como ajedrez o scrabble), llevar el control de calificaciones de un grupo de alumnos, implementar estructuras de datos, optimizar operaciones matemáticas, etc., se utilizan necesariamente arreglos.

La escítala espartana

Uno de los primeros métodos criptográficos conocidos proviene de Esparta, Grecia. El método consiste en enrollar una tira de escritura a lo largo de un palo llamado escítala y escribir sobre la tira una vez enrollada. Al desenrollar el mensaje resulta ininteligible a menos que se posea una escítala similar a la que se usó para crear el mensaje.

Figura 4. Forma de la escítala espartana y la tira de escritura.

Código (la escítala espartana)

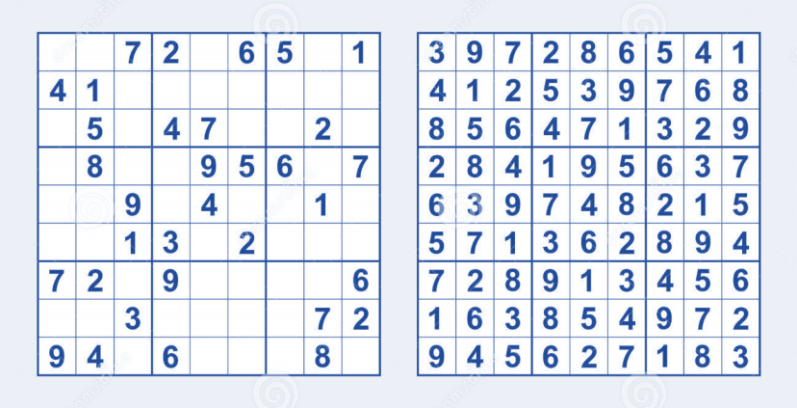
```
#include<stdio.h>// Archivos de cabecera (Librerías)
/*Programa que realiza la implementación de la escítala espartana para cifrar y descifrar.*/

void crearMensaje();//Prototipo de la función
void descifrarMensaje();//Prototipo de la función

int main(){//Programa principal
    short opcion=0;//Declaración de variable tipo entero corto, inicializada en 0.
    while (1){ // Intrucción de repetición
        printf("\n\t*** ESCITALA ESPARTANA ***\n");//Imprime en pantalla
        printf("\n\tQue desea realizar?\n");//\t, tabulador
        printf("\n1) Crear mensaje cifrado.\n");// \n añade una línea
        printf("2) Descifrar mensaje.\n");
        printf("3) Salir.\n");
        scanf("%d", &opcion);// Lee y almacena la variable "opción"
        switch(opcion){//Instrucción de selección, según "opcion"
            case 1:// Si "opcion" 1
                crearMensaje();// Llama a la función
                break;//Termina el caso
            case 2:// Si "opcion" 2
                descifrarMensaje();// Llama a la función
                break;//Termina el caso
            case 3:// Si "opciones" 3
                return 0;//Finaliza la ejecución
            default://En caso de no ser ninguna de las anteriores
                printf("\n\tOpcion no valida.\n");//Imprime en pantalla
        }     }
    return 0;//Regrita la función main
```

```
}
void crearMensaje(){//Función, se ejecutará cuando sea llamada
    int ren, col, i, j, k=0;//Declaración de variables enteras
    printf("\n\tIngresar el tamaño de la escitala:\n");//Imprime en pantalla
    printf("\nRenglones:");// \n añade una línea
    scanf("%i",&ren);//Lee y almacena "ren" un entero decimal,hexadecimal u octal.
    printf("\nColumnas:");
    scanf("\t%i",&col);//Lee y almacena "col" un entero decimal,hexadecimal u octal.
     char escitala[ren][col];//Declaración de una matriz variable como tipo de dato "carácter"
     char texto[ren*col];//Declaración de una matriz variable como tipo de dato "carácter"
     //Duda en la razón de multiplicar renglones por columnas
    printf("\nEscriba el texto a cifrar:\n");
    scanf("%s", texto);//Lee y almacena "texto" una cadena
    //Lee los elementos de cada uno de los espacios de nuestra matriz
     for (i=0 ; i<ren ; i++)//for para leer cada columna
        //Utiliza el valor actual de i en la expresión en la cual esté i y después se incrementa en 1 hasta ren
        for (j=0 ; j<col ; j++)//for para leer cada renglón
        //Utiliza el valor actual de k en la expresión en la cual esté k y después se incrementa en 1 hasta col
            escitala[i][j] = texto[k++];//Tengo duda en las razones de esta igualación
    //Imprime los elementos de cada uno de los espacios de nuestra matriz
    printf("\nEl texto en la tira queda de la siguiente manera:\n\n");
    for (i=0 ; i<col ; i++)//for para imprimir cada columna
        for (j=0 ; j<ren ; j++)//for para imprimir cada renglón
            printf("%c", escitala[j][i]);//Imprime la matriz de acuerdo a for
    printf("\n");//Imprime en pantalla, \n añade una línea
}
//Tiene los elementos de la función anterior
 void descifrarMensaje(){//Función, se ejecutará cuando sea llamada
    int ren, col, i, j, k=0;//Declaración de variables enteras
    printf("\n\tIngresar el tamaño de la escitala\n");//Imprime en pantalla
    printf("\nRenglones:");
    scanf("%i",&ren);
    printf("\nColumnas:");
    scanf("%i",&col);
     char escitala[ren][col];
     char texto[ren*col];
     printf("\nEscriba el texto a descifrar:\n\n");
     scanf("%s", texto);
     for (i=0 ; i<col ; i++)
        for (j=0 ; j<ren ; j++)
            escitala[j][i] = texto[k++];
    printf("\n\tEl texto descifrado es:\n\n");
    for (i=0 ; i<ren ; i++)
        for (j=0 ; j<col ; j++)
            printf("%c", escitala[i][j]);
    printf("\n");//Imprime en pantalla, \n añade una línea
}
```

SUDOKU



		7	2		6	5		1
4	1							
	5		4	7			2	
	8			9	5	6		7
		9		4			1	
		1	3		2			
7	2		9					6
		3					7	2
9	4		6				8	

3	9	7	2	8	6	5	4	1
4	1	2	5	3	9	7	6	8
8	5	6	4	7	1	3	2	9
2	8	4	1	9	5	6	3	7
6	3	9	7	4	8	2	1	5
5	7	1	3	6	2	8	9	4
7	2	8	9	1	3	4	5	6
1	6	3	8	5	4	9	7	2
9	4	5	6	2	7	1	8	3

```c
#include<stdio.h>
int main()
{
    int i,j,A[9][9],B[9][9];
    printf("\n\n\t SUDOKU \n");
    printf("\n\n\t Introduce los numeros del sudoku \n");
    printf("\n\n\t Para introducir la respuesta, ingresa un 0 \n");
    for (i=0;i<9;i++)
        for (j=0;j<9;j++)
        {
            printf("\n\n\t SUDOKU [%d][%d] = ",i+1,j+1);
            scanf("%d",&A[i][j]);
            B[i][j]=A[i][j];
                if (A[i][j]==0)
                {
                printf("\n\n\t Comienza a llenar! \n");
                scanf("%d",&A[i][j]);
                }
        }
        printf("\n\n\n\tSUDOKU\n\n ");
        for (i=0;i<9;i++)
            {
                for (j=0;j<9;j++)
                    printf("\t%d",B[i][j]);
                    printf("\n");
            }
        printf("\n\n\n\tSOLUCION\n\n ");
        for (i=0;i<9;i++)
            {
                for (j=0;j<9;j++)
                    printf("\t%d",A[i][j]);
                    printf("\n");
            }
}
```

```
        SUDOKU

        Introduce los numeros del sudoku

        Para introducir la respuesta, ingresa un 0

        SUDOKU [1][1] = 0

        Comienza a llenar!
3

        SUDOKU [1][2] = 0

        Comienza a llenar!
9

        SUDOKU [1][3] = 7

        SUDOKU [1][4] = 2

        Comienza a llenar!
7

        SUDOKU [9][7] = 0

        Comienza a llenar!
1

        SUDOKU [9][8] = 8

        SUDOKU [9][9] = 0

        Comienza a llenar!
3


       SUDOKU
       0       0       7       2       0       6       5       0       1
       4       1       0       0       0       0       0       0       0
       0       5       0       4       7       0       0       2       0
       0       8       0       0       9       5       6       0       7
       0       0       9       0       4       0       0       1       0
       0       0       1       3       0       2       0       0       0
       7       2       0       9       0       0       0       0       6
       0       0       3       0       0       0       0       7       2
       9       4       0       6       0       0       0       8       0


       SOLUCION
       3       9       7       2       8       6       5       4       1
       4       1       2       5       3       9       7       6       8
       8       5       6       4       7       1       3       2       9
       2       8       4       1       9       5       6       3       7
       6       3       9       7       4       8       2       1       5
       5       7       1       3       6       2       8       9       4
       7       2       8       9       1       3       4       5       6
       1       6       3       8       5       4       9       7       2
       9       4       5       6       2       7       1       8       3
```

CONCLUSIÓN

Los arreglos unidimensionales y multidimensionales tienen diversas aplicaciones para la resolución de problemas computacionales, es por eso de su importancia, su mejor funcionamiento depende de su buen uso y correcta aplicación, con ello podemos lograr una optimización a códigos que con otro tipo de estructura serían largos, complejos, o imposibles de elaborar.

BIBLIOGRAFÍA O REFERENCIAS:

El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.