**Universidad Nacional Autónoma de México**

Facultad de Ingeniería

**Laboratorios de computación**

**salas A y B**

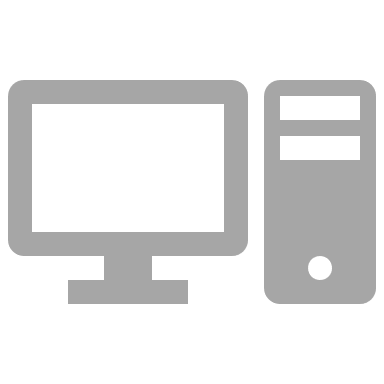
**PROFESOR:** M.I. Marco Antonio Martínez Quintana

**ASIGNATURA:** Estructura de Datos y Algoritmos I

**GRUPO:** 17

**NO DE PRÁCTICA:** 2

**NOMBRE:** Reyes Mendoza Miriam Guadalupe

**SEMESTRE:** 2020-2

**FECHA DE ENTREGA:** 15/02/2020

**OBSERVACIONES:**

**CALIFICACIÓN:**

**APLICACIONES DE APUNTADORES**

**OBJETIVO**

Utilizar apuntadores en lenguaje C para acceder a las localidades de memoria tanto de datos primitivos como de arreglos.

**INTRODUCCIÓN**

**APUNTADOR**

Variable que contiene la dirección de memoria de otra variable, es decir, hace referencia a la localidad de memoria de otra variable.

Los apuntadores se utilizan para dar claridad y simplicidad a las operaciones a nivel de memoria. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a la información almacenada.

Para declarar un apuntador se debe definir el tipo de dato y el nombre de la variable apuntador, precedida de un asterisco (\*). Una variable de tipo apuntador debe tener el mismo tipo de dato de la variable a la que va a apuntar:

**TipoDeDato \*apuntador, variable;**

Para asignarle un valor al apuntador, se debe acceder a la localidad de memoria de la variable a través de un ampersand (&):

**apuntador = &variable;**

[](https://www.google.com/url?sa=i&url=https%3A%2F%2Fcodingornot.com%2F02-apuntadores-conociendo-los-apuntadores&psig=AOvVaw2H1dhzzVLh23TqkcWvGRv7&ust=1581744613535000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCMjYy_-n0OcCFQAAAAAdAAAAABAT)**Nota:** Los apuntadores solo deben apuntar a variables del mismo tipo de dato con el que fueron declarados.

**APUNTADORES EN C**

Lenguaje C es un lenguaje de alto nivel porque permite programar a bajo nivel. La programación a bajo nivel se refiere a la manipulación de los recursos físicos de un equipo computacional. Los apuntadores permiten manipular de manera directa las localidades de memoria RAM de la computadora.

Una maquina típica tiene un arreglo de celdas de memoria numeradas o direccionadas consecutivamente, pueden manipularse individualmente o en grupo contiguos. Una situación común es que cualquier byte puede ser un char, un par de celdas de un byte pueden tratarse como un short, y cuatro bytes adyacentes forman un long. Un apuntador es un grupo de celdas (generalmente dos o cuatro) que pueden mantener una dirección. Así, si c es un char y p es un apuntador que apunta a él, podría representarse la situación de esta manera:

Imagen que contiene captura de pantalla

Descripción generada automáticamente

El operador unitario & de la dirección de un objeto, de modo que la proposición.

**p = &C;**

asigna la dirección de la variable c en la dirección de p, y se dice que p “apunta a” c. El operador & solo se aplica a objetos que están en la memoria: variables y elementos de arreglos. No puede aplicarse a expresiones, constantes o variables de tipo registro.

El operador unitario \* es el operador de indirección; cuando se aplica a un apuntador, da acceso al objeto al que señala el apuntador.

**DESARROLLO Y RESULTADOS**

1. **CÓDIGO: APUNTADOR A CARÁCTER**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa crea un apuntador con una variable de tipo carácter, lo primero que hace es declarar un apuntador de tipo carácter con el nombre ¨ap¨ y una variable del mismo tipo con el nombre de ¨c¨ donde c es la letra ¨x¨ y el apuntador accede a la localidad donde está la variable ¨c¨.

Después se empiezan a usar funciones la primera imprime el carácter de la localidad donde se está apuntando, posteriormente se imprime el código ASCII de la misma localidad y por último imprime la dirección de la memoria donde se encuentra el carácter.

1. **CÓDIGO: APUNTADOR A ENTERO**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa crea un apuntador con una variable de tipo entero, lo primero que se hace es declarar las variables de tipo entero empezando por ¨a¨ a la cual se le asigna un valor de 5 y ¨b¨ con un valor de 10 y otra que es el apuntador nombrado como ¨ap¨ y de tipo entero donde se le asigna el valor de ¨a¨ accediendo a la memoria.

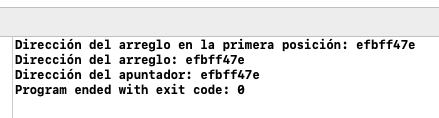
Se empieza usando la función printf para imprimir el valor de ¨a¨ después se le asigna a ¨b¨ el valor que esta en la localidad a la que apunta para después imprimirlo en pantalla. Posteriormente lo que se hace es sumarle 1 al valor que esta en la localidad a la cual se esta apuntando eso nos daría 6 que es justamente lo que imprime, lo que sucede después es que se la da el valor de 0 a la variable que apunta ¨ap¨ que en este caso es ¨a¨ y es la que aparece en pantalla.

1. **CÓDIGO: APUNTADOR A ARREGLO**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**RESULTADOS**

****

**EXPLICACION**

Este programa crea un apuntador con una variable de tipo entero pero que en este caso es un arreglo y un apuntador al cual se le asigna la posición o del arreglo. En la primera instrucción se imprime la dirección en la memoria de la posición 0, en la segunda la dirección del arreglo y el ultimo la dirección del apuntador, podemos ver que son las misma puesto que todas están en la misma posición debido a que nos estamos refiriendo a la misma variable.

1. **CÓDIGO: PASO DE VARIABLES POR VALOR Y POR REFENCIA**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa tiene dos funciones que van a regresar valores enteros y las cuales son llamadas en la función principal y la cual empieza definiendo un arreglo y un apuntador al cual se le asigna el valor del arreglo, la primera función imprime el valor de la localidad que apunta que es el elemento 0 del arreglo. Después se llama a la primera función la cual imprime el valor de la variable entera ¨equis¨ el cual es 55 pero después le asigna el valor de 128 y lo imprime, se sigue la función principal y ahora se vuelve a imprimir el valor de la localidad a la que apunta para después llamar a la siguiente función donde se apunta a la localidad donde esta ¨equis¨ la cual estaba en 55 pero después se le da otro valor y ahora apunta a la 128 que es el valor que imprime la última instrucción de la función principal.

1. **CÓDIGO: ARITMETICA DE DIRECICONES**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa crea un arreglo y un apuntador para moverse en el mismo esto se hace dándole a la variable ¨ap¨ el valor del arreglo y el cual será su primera posición, esto lo podemos ver porque al imprimir el valor de la localidad a la que está apuntando da el valor que está en la posición 0 de arreglo, después para moverse en el arreglo a la localidad se le suma 1, es decir, avanza una posición en la memoria lo cual hace que vaya a la posición 1 del arreglo e imprimir su valor, lo mismo pasa en la última instrucción puesto que se le suma dos localidades los cual hace que desde la posición 0 se dirija a la posición 2 del arreglo e imprima su valor.

En pocas palabras podemos sumar a la localidad del apuntador para que este se desplace sobre las diferentes localidades de memoria y en caso de arreglos para que avance a la siguiente posición.

1. **CÓDIGO: ARREGLO UNIDIMENCIONAL**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa muestra la dirección en la que se encuentran los diferentes elementos de un arreglo unidimensional donde el apuntador va a ser la localidad donde están sus elementos. Mediante un ciclo de repetición for va recorriendo el arreglo y se va imprimiendo la localidad de memoria donde se encuentran las distintas posiciones del arreglo, podemos ver que solo cambian los dos últimos dígitos y que estos van de dos en dos.

1. **CÓDIGO: ARREGLO BIDIMENSIONAL**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa muestra la dirección en la que se encuentran los diferentes elementos de un arreglo bidimensional donde el apuntador va a ser la localidad donde está cada uno de sus elementos. Mediante un ciclo de repetición for va recorriendo el arreglo y se va imprimiendo la localidad de memoria donde se encuentran sus distintas posiciones.

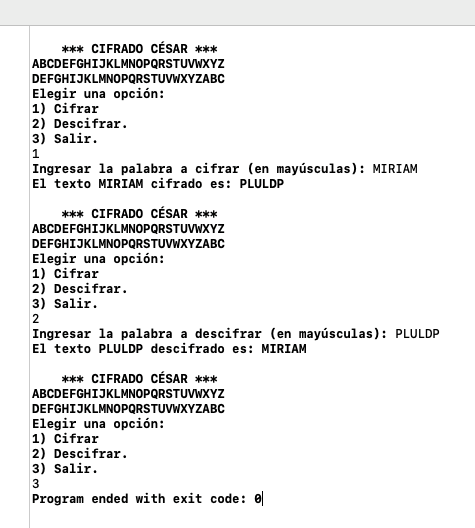
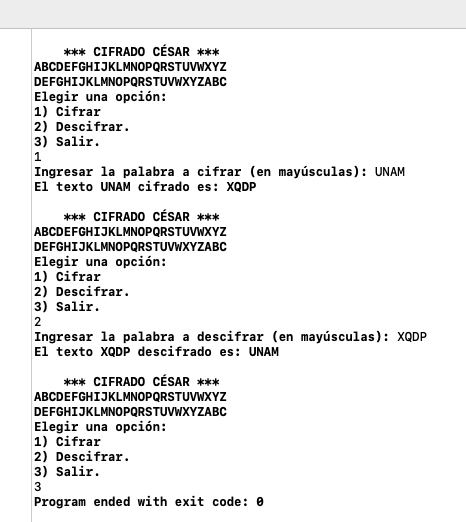
1. **CÓDIGO: MI CIFRADO CESAR**

**Imagen que contiene captura de pantalla

Descripción generada automáticamenteImagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

****

**EXPLICACION**

Este programa crea un cifrado de César primero se define el tamaño de la palabra y después el tamaño del abecedario también mediante un arreglo de tipo carácter se definen las letras a usar y en otro cómo se van a cifrar (el nuevo orden que se le da al abecedario) sustituyendo las letras del texto original por las otras, después se declaran las dos funciones que apuntan a la dirección de los respectivas palabras y que servirán para cifrar y descifrar el texto usado en la función principal.

En la función principal se empiezan por declarar las variables a usar y las cuales son 2 la primera es opción que se inicializa con 0 sirviendo para que el usuario pueda seleccionar una opción de acuerdo con lo que desee hacer y la variable contador que se va a utilizar para mostrar (moverse) dentro del arreglo con el abecedario que se hizo anteriormente para escribir nuestro texto y el otro que cambia el orden las letras de acuerdo a como se definió es su respectivo arreglo, después se declara la variable de tipo carácter llamada palabra que usa el arreglo del abecedario para que el usuario pueda escribir su palabra.

Mediante una estructura de repetición (while) se imprime en pantalla el menú y se realizan todas las instrucciones dentro del mismo, estando definido de la siguiente manera:

Primero se imprime el titulo del programa, posteriormente mediante un ciclo de repetición for de imprime el arreglo de caracteres ordenados definido con anterioridad y después se imprime igualmente con un ciclo for el abecedario cifrado (las letras en diferente orden y que sustituirán a las del texto original) esto es donde se hace apuntando a la dirección de las localidades del arreglo de los abecedarios. Al terminar se imprime el menú y se lee la opción que el usuario escribió, para que el programa haga lo de la opción que se seleccionó se define una estructura de selección (switch) donde en cada caso dependiendo de la opción se realizan diferentes tareas la primera opción hace lo siguiente:

Pide al usuario que Ingrese la palabra que desea cifrar, se lee y se llama a la función cifrar () y dentro de esta el texto con el que va a trabajar:

La función cifrar() lo que hace es leer el texto ordenado apuntando a las localidades donde están los caracteres del texto ordenado, lo primero que se hace es que se imprime en pantalla el texto que se cifro posteriormente para cifrarlo se definen tres variables de la cuales dos son contador solo que uno es de la palabra y otro del abecedario mediante dos ciclo de repetición (for) va recorriendo cada una de la letras de la palabra y del abecedario para que con una estructura if donde si se cumple al condición se va imprimiendo el cifrado de cada una de las letras, esto se hace con cada letra (se lee la primera letra en el texto ordenado se busca en el abecedario cifrado que posición estaba y se cambia por la que esta en esa misma posición para después imprimirla y así con todas las letras de la palabra).

La segunda opción realiza las siguientes instrucciones pues realmente hace casi lo mismo solo que ahora lee el texto cifrado y recorre el original sustituyendo las letras por el abecedario ordenado. Para esto primero se pide ingresar la palabra y se lee, después se dirige a la función descifrar ():

Está primero escribe el texto que se va a descifrar mediante un apuntador dirigido a las localidades de cada uno de los caracteres del texto cifrado; después se declaran las variables que se van a utilizar y mediante un ciclo de repetición (for) y un if se recorren se va recorriendo el arreglo del abecedario Cifrado y se cambia la letra que esta en la misma posición pero del abecedario en claro y así con todas las letras de la palabra hasta terminar mientras las imprime .

En caso de escoger la opción 3 se regresa un 0 y se termina el programa y en caso de escoger la opción que no esta definida dentro del menú se imprime en pantalla que no es una opción válida.

**CONCLUSIONES**

En el último programa podemos ver la aplicación de apuntadores y realmente es muy interesante ver cómo podemos ir optimizando el código con este tipo de cosas e igualmente volveremos a ver el uso de arreglos, lo que tiene el último programa es que podemos ver una gran optimización de código que a final de cuentas nos va a distinguir y que nivel de programación tenemos.

Con esta práctica pude aprender y dominar el concepto de apuntador pues nos permite muchas veces manejar los datos o la información de una manera más sencilla y ubicando de mejor manera la información, sin embargo, al copiar el programa lo hacia mal, esto me hizo darme cuenta y no solo en este si no también en los programas anteriores donde no se utilizaba alguna variable, que muchas veces es muy difícil encontrar el error dentro de muchas líneas de código y el programa no compila o simplemente no hace nada pero muchas veces es por acomodar mal las llaves o no poner punto y coma, a final de cunetas yo creo que esto es algo en lo que nosotros tenemos que tener cuidado pero también desarrollar esa habilidad para reconocer el error fácilmente.

**BIBLIOGRAFÍA**

* Brian W. Kernighan, Dennis M. Ritchie. (1991). *El lenguaje de programación C* (2ª. Ed.). México: Pearson Educación.