



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: TISTA GARCÍA EDGAR

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS I

Grupo: 1

No de Práctica(s): 2] Aplicaciones de apuntadores

Integrante(s): MURRIETA VILLEGAS ALFONSO

Semestre: 2018 - 2

Fecha de entrega: 26 DE FEBRERO DE 2018

Observaciones:

CALIFICACIÓN: _____

ALFONSO MURRIETA

1

APLICACIONES DE APUNTADORES

1] Objetivo de la práctica

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

2] Desarrollo

INTRODUCCIÓN:

Una de las mayores ventajas que ofrece el lenguaje C es la oportunidad de poder optimizar operaciones o procesos de manera inimaginable, es el caso del uso de apuntadores los cuales de manera directa pueden acceder a la ubicación de datos en la memoria RAM, dicho de otra forma, los apuntadores se utilizan sobre todo para dar una mayor claridad, simplicidad e incluso optimización a nuestros programas.

Un apuntador se compone de 3 pasos principales:

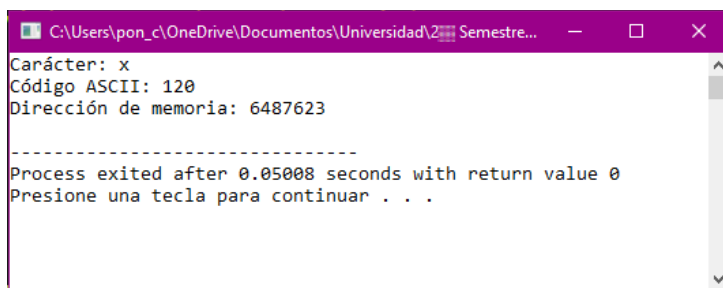
- 1) La declaración de este.
- 2) La asignación a una variable del mismo tipo.
- 3) Uso del apuntador con propósitos diversos.

NOTA: 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:

2.1] Análisis de los problemas (Propuestos)

EJERCICIO 1

Es un programa básico donde se muestra como mediante un apuntador podemos visualizar la dirección de memoria en la que se encuentra guardada nuestra variable, en el caso de la salida o de lo que fue impreso vemos primero el carácter impreso, posteriormente vemos el número al que es asignado dentro del código ASCII para ello usamos dentro del *printf* *%d* en vez de *%c* pues de esta forma imprimimos un valor numérico el cual es el valor asignado dentro del código ASCII

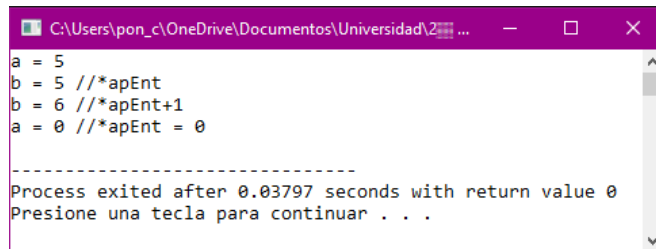


```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2º Semestre...
Carácter: x
Código ASCII: 120
Dirección de memoria: 6487623

-----
Process exited after 0.05008 seconds with return value 0
Presione una tecla para continuar . . .
```

EJERCICIO II

En el presente código se muestra un claro ejemplo de como los apuntadores pueden ser utilizados en variables del tipo *short*, además, otro aspecto relevante es que en este programa notamos como mediante el uso del apuntador **apEnt* podemos manipular de manera indirecta los valores de las variables *a* y *b*, en este caso concreto lo hicimos mediante operaciones aritméticas básicas donde usábamos el apuntador. A continuación, una captura de pantalla de la salida del programa:



```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2... -- -- X
a = 5
b = 5 /*apEnt
b = 6 /*apEnt+1
a = 0 /*apEnt = 0

-----
Process exited after 0.03797 seconds with return value 0
Presione una tecla para continuar . . .
```

EJERCICIO III

El presente ejercicio hace uso de un apuntador del tipo *short* el cual fue asignado a un arreglo del mismo tipo, la parte fundamental del programa es sin duda como existen maneras distintas en que uno puede llamar a la dirección donde se encuentra ubicado el arreglo:

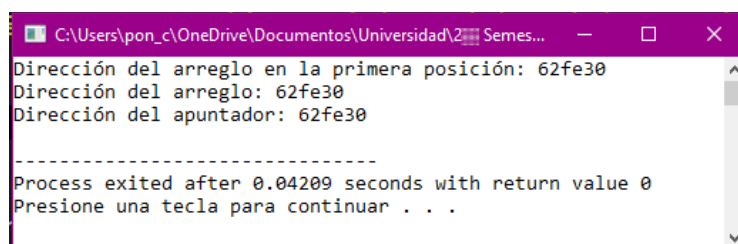
El primer caso es la manera en que se llaman a las direcciones de un arreglo o la de un elemento en particular, es el caso de *&arr[0]* donde el 0 es para referenciar la dirección del elemento 0 del arreglo

El segundo caso es la manera en la que se omite que elemento es el que se quiere saber, simplemente se quiere saber la ubicación general del arreglo y por obvias razones por ello se otorga la del elemento 0, es el caso de *&arr*

Y por último, otra forma en que se puede acceder a la dirección de memoria es la siguiente manera *apArr* o sea solamente poniendo el apuntador

NOTA: Como se puede ver en la salida las direcciones de memoria están en hexadecimal debido a la forma que fueron impresos *%x*, en caso de que se quiera escribir de manera decimal simplemente tendríamos que usar la siguiente forma de impresión *%d*

A continuación, una captura de pantalla de la salida del programa:



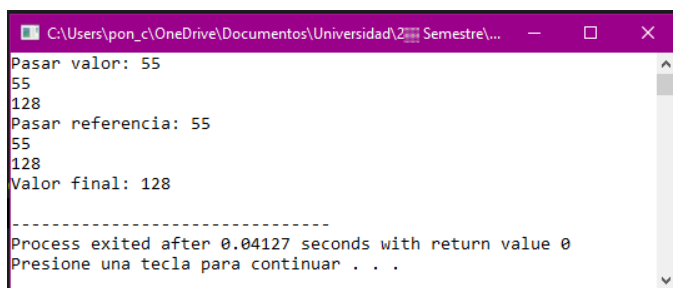
```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2... Semes... -- -- X
Dirección del arreglo en la primera posición: 62fe30
Dirección del arreglo: 62fe30
Dirección del apuntador: 62fe30

-----
Process exited after 0.04209 seconds with return value 0
Presione una tecla para continuar . . .
```

EJERCICIO IV

En el mundo de la programación la parte de argumentos o parámetros dentro de las funciones existen dos maneras en las que se pueden manipular los datos, la primera es el caso de **paso por valor** donde a los elementos principales y originales no son modificados más que en la función alternativa y esto es posible debido a que se envía una copia a esta, por otro lado, en el caso de **paso por referencia**, todo es distinto puesto a la función secundaria o alternativa se manda un apuntador el cual de manera indirecta cambia los valores de las variables originales.

Con respecto al ejercicio 4, es un claro ejemplo de que es lo que se requiere y que es lo que pasa en distintas funciones donde la que se llama *pasar valor* es el paso por valor, mientras la función que se llama *pasar por referencia* hace uso del caso de paso por referencia.

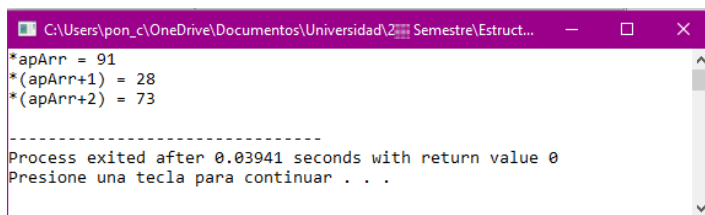


EJERCICIO V

Este ejercicio a diferencia de los anteriores hace uso del operador aritmético más básico (La suma) para poder imprimir mediante apuntadores las direcciones de variables.

NOTA: Algo relevante que destaca este código es la forma sintáctica en la que se recorre el valor que se quiere imprimir puesto que en el caso de **(apArr+1)* primero se realiza la operación contenida en los paréntesis para posteriormente direccionar el resultado mediante el operador ***.

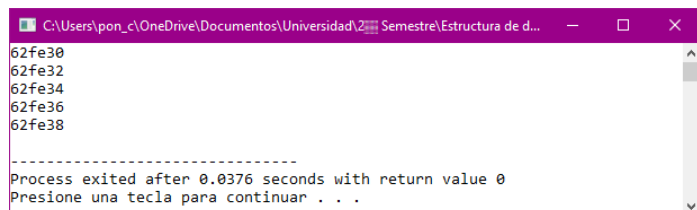
De esta forma obtenemos la siguiente salida:



EJERCICIO VI

Con respecto a este código, mediante un *for* lo que hacemos es imprimir la dirección de memoria de cada elemento de un arreglo declarado, al igual que en los códigos anteriores podemos notar como la forma en que se imprime las direcciones es en su expresión hexadecimal

A continuación, una captura de pantalla de la salida del programa:



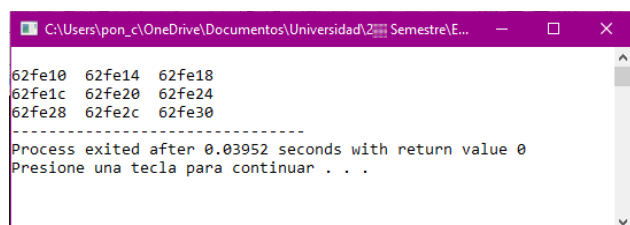
```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2º Semestre\Estructura de d...
62fe30
62fe32
62fe34
62fe36
62fe38
-----
Process exited after 0.0376 seconds with return value 0
Presione una tecla para continuar . . .
```

EJERCICIO VII

En este ejercicio lo que podemos notar es que como mediante el uso de un puntero ubicamos la localización de todos los elementos de un arreglo bidimensional o también llamado matriz, obviamente para poder dar como salida cada una de las ubicaciones hacemos uso de un ciclo *for*.

NOTA: Al igual que en los anteriores códigos nuevamente al usar `%x` la manera en que se expresa la dirección de memoria es en hexadecimal

A continuación, una captura de pantalla de la salida del programa:



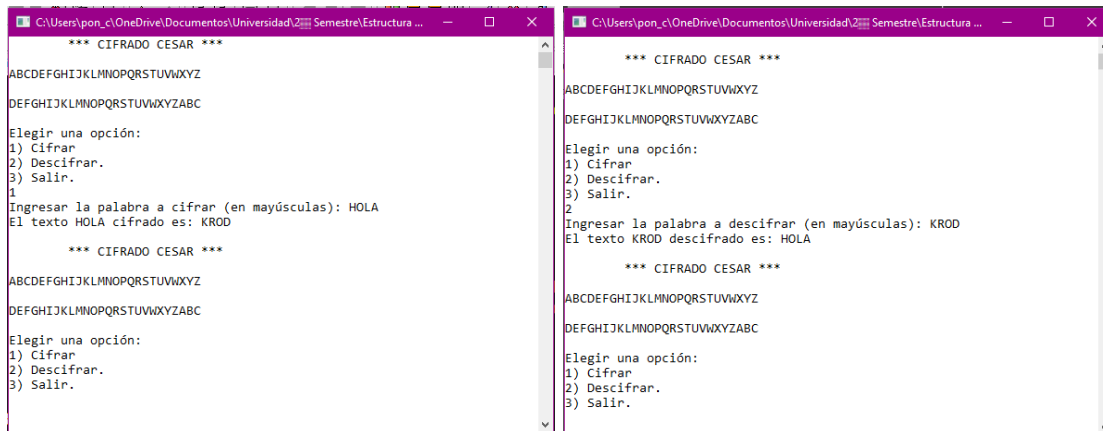
```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2º Semestre\E...
62fe10 62fe14 62fe18
62fe1c 62fe20 62fe24
62fe28 62fe2c 62fe30
-----
Process exited after 0.03952 seconds with return value 0
Presione una tecla para continuar . . .
```

EJERCICIO VIII

Este código es sin duda un claro ejemplo de cómo se pueden implementar los apuntadores, en general lo que hace este programa es cifrar un mensaje mediante un simple algoritmo el cual recorre la posición de las letras para finalmente entregar un mensaje ya cifrado.

Para poder llevar acabo esto el programa hace uso de 3 funciones, la función principal, la función cifrar y la función descifrar. Cabe destacar que en el programa se hacen uso de dos abecedarios guardados en variables de tipo *char*, donde uno es el abecedario normal y el otro está recorrido (Es el que se usa para poder cifrar los mensajes).

A continuación, se muestran capturas de pantalla donde vemos como se cifra y descifra un mensaje el cual solo contiene a la palabra HOLA:



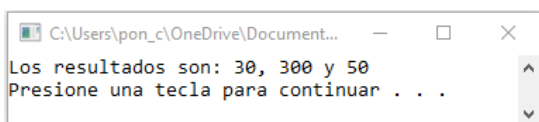
2.2] Análisis de los problemas (Práctica)

EJERCICIO I

El presente ejercicio es un código realizado por algún programador el cual contiene bastantes errores relacionados con el uso de apuntadores, entre los errores más comunes se encuentran el asignar de manera incorrecta los operadores de dirección e indirección de apuntadores, además de asignar operadores a variables que no son apuntadores.

Recordemos que siempre al usar apuntadores debemos declararlos, asignarlos a alguna variable del mismo tipo y por último utilizarlos de manera correcta.

A continuación, se muestra la salida final del programa una vez corregido:



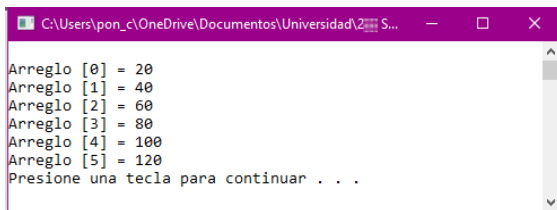
NOTA: No pude agregar una captura de pantalla del programa erróneo en tiempo de ejecución debido a que este no se puede compilar.

EJERCICIO II

Este ejercicio al igual que el anterior implementa apuntadores para poder realizar una suma básica de los elementos de un arreglo ya declarado, sin embargo, nuevamente se ha declarado mal el apartado de asignación del apuntador a la variable puesto recordemos que para asignar un puntero a una variable se debe escribir de la siguiente forma:

Nombredelapuntador = nombredel arreglo;

Donde el nombre del apuntador no debe llevar ningún operador de indirección o dirección
A continuación, se muestra la salida final del programa una vez corregido:



```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2do S...
Arreglo [0] = 20
Arreglo [1] = 40
Arreglo [2] = 60
Arreglo [3] = 80
Arreglo [4] = 100
Arreglo [5] = 120
Presione una tecla para continuar . . .
```

NOTA: No pude agregar una captura de pantalla del programa erróneo en tiempo de ejecución debido a que este no se puede compilar

EJERCICIO III

Este programa debe solicitar al usuario 2 valores los cuales se guardarán en variables independientes y además con estas únicas dos variables debe realizarse el producto entre ambos valores, además de que como segundo resultado debe nuevamente multiplicarse el producto anterior con la segunda variable. Como principal conflicto que tenemos en este programa es el hecho de que no disponemos de otras variables para poder guardar los valores de los resultados, por lo tanto y como es lógico debemos usar las mismas variables para poder realizar las operaciones anteriormente mencionadas.

Para ello lo que he idea es usar paso por referencia ya que de esta forma podemos utilizar las mismas variables donde se llevaran a cabo las operaciones, para ello usé una función extra la cual mediante apuntadores hice esto posible.

```
void funcionalterna(int *a, int *b){
    *a= (*a) * (*b);
```

El anterior pedazo de código es la parte que se encarga de guardar dentro de la misma variable a el resultado del producto de a y b quienes son las variables que da el usuario, cabe destacar que para hacer esto posible debemos utilizar los apuntadores de las respectivas variables.

A continuación, se muestra la salida del programa:

```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2º Semestre\Estructura de datos y al...
Ingrese el valor de su primer dato: 2
Ingrese el valor de su segundo dato: 4
El producto entre los números es 8
El producto entre el producto anterior y el segundo dato es 32
-----
Process exited after 1.688 seconds with return value 0
Presione una tecla para continuar . . .
```

EJERCICIO IV

Para llevar a cabo este programa lo primero que se tiene que hacer es considerar que para poder intercambiar los valores asignados a 3 variables se puede hacer de distintas formas, sin embargo, considerando que debido a que el programa menciona que se debe hacer a través de *paso por referencia* por lo tanto bien sabemos que necesitaremos de apuntadores para así poder implementar algún algoritmo el cual se encargará en el intercambio de valores.

Para ello me he basado en el programa que el profesor hizo en clase puesto a pesar de que solo cambió los valores asignados de 2 variables, haciendo algunas modificaciones podemos implementar este algoritmo, pero para 3 variables.

A continuación, se muestra una parte del código del programa, la cual se encarga de cambiar los valores asignados de las variables mediante el uso de operaciones aritméticas de apuntadores.

```
void mifuncion(int *k,int *m){
    /*Esta parte simplemente lo que hace es cambiar el valor de cada variable que entra en la función
    esto es posible debido a las operaciones aritméticas que se pueden usar con los apuntadores*/
    *k = *k - *m;
    *m = *m + *k;
    *k = *m - *k;
```

NOTA: Como es obvio, para poder cambiar los valores de 3 variables necesitamos al menos realizar 2 veces este algoritmo debido a que solamente cambiamos por cada vez los valores de 2 variables. Por último, a continuación se muestra la salida del programa ya terminado.

```
C:\Users\pon_c\OneDrive\Documentos\Universidad\2º Semestre\Estructura de datos y algoritmos\Pr...
Ingrese el valor de i: 2
Ingrese el valor de j: 10
Ingrese el valor de k: 7

Los valores actuales de cada variable son:

    i: 10 | j: 7 | k: 2
-----
Process exited after 7.881 seconds with return value 24
Presione una tecla para continuar . . .
```

2.3] Relación entre los ejercicios y el tema

Todos los ejercicios del manual de prácticas excepto el último son solamente para introducción y uso general de apuntadores, sin embargo, el último es un claro ejemplo de cómo al implementar apuntadores dentro de nuestros códigos podemos crear cosas asombrosas y es que el poder codificar mensajes de una manera tan simple es sin duda increíble, cabe destacar que el algoritmo de encriptación es relativamente sencillo, sin embargo, el punto principal es como los apuntadores nos ayudan de una manera enorme para poder llegar a esto.

Por otro lado, los ejercicios de la práctica (Los propuestos por el profesor) en el caso particular de los dos primeros son para poder ver de manera general que mediante apuntadores podemos también realizar operaciones aritméticas de manera indirecta, además, los últimos dos ejercicios son un claro ejemplo de cómo mediante la abstracción del problema y a través del razonamiento podemos optimizar nuestros códigos y encontrar otras maneras de operar nuestras variables.

3] Conclusiones

El uso de apuntadores dentro del mundo de la programación y del lenguaje C es sin duda ilimitado, esta práctica es un claro ejemplo que podemos utilizar apuntadores para prácticamente cualquier problema que se nos ponga enfrente, además el uso de apuntadores tiene como uno de sus muchos motivos optimizar y demostrar que C es un lenguaje que ofrece ventajas al momento de manipular los datos relacionados con la dirección y espacio de memoria.

Algo que también debo destacar en esta práctica es que lamentablemente el tiempo al igual que en la anterior se me hizo muy poco, además de que los nervios y las prisas no me beneficiaron nada, entre los errores que pude corregir con respecto al ejercicio 3 se encuentra un apuntador mal usado y sobre todo la parte de haber confundido la salida de la *funcionalterna* que utilicé en el programa.

Por último, también cabe destacar que en esta práctica pude notar y aprender el uso de apuntadores desde variables primitivas como fue el caso de enteros (Los primeros ejercicios) hasta el uso de estos para casos más complejos como arreglos o cadenas de caracteres.

4] Referencias

- 1) Estructuras de datos en C. Luis Jayanes A, Matilde Fernández A., primera edición, ESPAÑA, McGrawHill 2005.
- 2) El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.
- 3) Curso de Programación C7 C++. Javier Ceballos, tercera edición. USA, Ra – Ma. Educación 2007