
 GOBIERNO DEL ESTADO DE MÉXICO	ISO 9001:2015 MANUAL DE PRÁCTICAS FO-TESJI-11100-12	 TES TECNOLÓGICO DE ESTUDIOS SUPERIORES JILOTEPEC
---	--	---

NOMBRE DE LA PRÁCTICA	REPORTE APUNTADES			No.	15
ASIGNATURA:	MÉTODOS NÚMERICOS	CARRERA:	ISIC 3402	PLAN:	

ERIKA YAZMIN DIONISIO VELASCO
3402

I. COMPETENCIA(S) ESPECÍFICA(S):

II. MATERIAL EMPLEADO:

- Laptop
- Visual Stdio Code (Linux)

III. DESARROLLO DE LA PRÁCTICA:

¿QUÉ ES UN APUNTADES?

Un puntero es un objeto que apunta a otro objeto. Es decir, una variable cuyo valor es la dirección de memoria de otra variable.

Dirección	Etiqueta	Contenido
...		
...		
1502	x	25
1503		
1504		
...		
...		

En C no se debe indicar numéricamente la dirección de la memoria, si no que se usa una etiqueta que conocemos como variable.

- Las direcciones en memoria dependen de la arquitectura del ordenador y de la gestión que el sistema operativo haga en ella.

¿Cómo se declaran los apuntadores?

Para declarar un apuntador se especifica el tipo de dato al que apunta, el operador '*', y el nombre del apuntador.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión		
Representante de la Dirección	1		
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		

Un puntero tiene su propia dirección de memoria.

La sintaxis es la siguiente:

<tipo de dato apuntador> *<identificador del apuntador>

```
int * punt;
char * car;
float * num;
```

¿Cómo se declaran los apuntadores?

Al igual que el resto de las variables, los apuntadores se enlazan a tipos de datos específicos, de manera que a un apuntador sólo se le puede asignar direcciones de variables del tipo especificado en la declaración

```
int * punt;
char * car;
float * num;
```

Tipos de apuntadores

Hay tantos tipos de apuntadores como tipos de datos.

Se puede también declarar apuntadores a estructuras más complejas.

Funciones
Struct
Ficheros

Se pueden declarar punteros vacíos o nulos.

¿Qué es la referenciación?

La referenciación es obtener la dirección de una variable.

Se hace a través del operador '&', aplicado a la variable a la cual se desea saber su dirección

&x ; //La dirección de la variable

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró Representante de la Dirección	Versión 1		
Autorizó Director General del Tecnológico de Estudios Superiores de Jilotepec	Fecha de revisión 7 de Febrero de 2017		

¿Qué es la referenciación?

No hay que confundir una dirección de memoria con el contenido de esa dirección de memoria.

Dirección	Etiqueta	Contenido
...		
...		
1502	x	25
1503		
1504		
...		
...		

x = 25; //El contenido de la variable
&x = 1502 ; //La dirección de la variable

Fragmento de código – referenciación

```
int dato;           //variable que almacenará un carácter.

int *punt;         //declaración de puntero a carácter.

punt = &dato;      //en la variable punt guardamos la dirección
                  //de memoria de la variable dato;
                  // punt apunta a dato.
```

Dirección	Etiqueta	Contenido
...		
...		
1502	dato	/0
1503		
1504		
...		

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Fragmento de código – referenciación

```
int dato;           //variable que almacenará un carácter.

int *punt;         //declaración de puntero a carácter.

punt = &dato;      //en la variable punt guardamos la dirección
                  //de memoria de la variable dato;
                  // punt apunta a dato.
```

Dirección	Etiqueta	Contenido
...		
...		
1502	dato	/0
1503	*punt	
1504		
...		

Fragmento de código – referenciación

```
int dato;           //variable que almacenará un carácter.

int *punt;         //declaración de puntero a carácter.

punt = &dato;      //en la variable punt guardamos la dirección
                  //de memoria de la variable dato;
                  // punt apunta a dato.
```

Dirección	Etiqueta	Contenido
...		
...		
1502	dato	/0
1503	*punt	1502
1504		
...		

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

¿Qué es la desreferenciación?

Es la obtención del valor almacenado en el espacio de memoria donde apunta un apuntador.

Se hace a través del operador '*', aplicado al apuntador que contiene la dirección del valor.

***p ; //El contenido de p**

Fragmento de código – referenciación----c

Bibliotecas

Método principal

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int x=17, y;
6      int *p;
7      p= &x;
8      printf("El valor de x es: %d\n", *p);
9      y=*p+3;
10     printf("El valor de y es: %d\n", y);
11 }

```

Printf, imprime el mensaje en pantalla, donde indica el valor de x.

En la variable p, guardamos lo que hasta ahora contiene el apuntador *p, y le sumamos 3, por que que si contábamos con 17 +3 =20

En la variable p guardamos la dirección en memoria de x

Ahora printf imprimira el valor de y, donde y, es el valor asignado anteriormente.

Al final del mensaje, imprime al apuntador *p, este contiene la dirección en memoria de x, pero no estamos pidiendo la dirección si no lo que contiene, así que si nos colocamos en esta dirección de x, está tiene asignado 17.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión	1	
Representante de la Dirección	Fecha de revisión	7 de Febrero de 2017	
Autorizó			
Director General del Tecnológico de Estudios Superiores de Jilotepec			

EJECUCIÓN DEL PROGRAMA

```

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS  3: bash
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~$ cd /home/erika/Documentos
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador.c -o apuntador.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador.exe
El valor de x es: 17
El valor de y es: 20
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$
Ln 11, Col 2  Spaces: 4  UTF-8  LF  C  Linux

```

Asignación de apuntadores

A un apuntador se pueden asignar direcciones de variables a través del operador de referenciación ('&') o direcciones almacenadas en otros apuntadores.

Direcciones inválidas

Un apuntador puede contener una dirección inválida por:

Cuando se declara un apuntador, posee un valor cualquiera que no se puede conocer con antelación.

Después de que ha sido inicializado, la dirección que posee puede dejar de ser válida por que la variable asociada termina su ámbito o porque ese espacio de memoria fue reservado dinámicamente.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Declaración de dos variables de tipo entero (y) y una de ellas asignada como apuntador ***p**

Declaración de nuestro método llamado **func ()**

Declaración de los atributos en nuestro método, recordemos que un método es utilizado para ser llamado más adelante junto con sus atributos en el declarado. Tenemos a una variable de tipo entero **x**, inicializada en 40

En la variable **p**, guardamos la dirección de la variable **x**
En la variable **y**, guardamos el contenido del apuntador ***p**, y como sabemos que este solo contiene la dirección de **x**, colocaremos el contenido de esa variable (**40**)
Por último, el apuntador ***p**, solo tendrá asignado el valor de 23

Bibliotecas

```
> erika > Documentos > C apuntador2.c > main(void)
#include <stdio.h>

int *p,y;

void func(){
    int x=40;
    p=&x;
    y=*p;
    *p=23;
}

int main (void){
    func();
    y=*p;
    *p=25;
    printf("El valor de y es: %d \n El valor de *p es: %d\n El valor de p es: %p\n", y,*p,p);
}
```

printf, imprimira los valores resultantes, llamando consigo a las variables correspondientes en donde se encuentra cada resultado

Una vez terminada la operación del método, retornará nuevamente a la función principal y ahora la variable **y**, cambiará por el contenido del apuntador ***p**, que como anteriormente habiamos dicho, este contiene solo la dirección en memoria de **x**, y ahora solo cambiara por lo que en ella contenga, así que este contiene **23**
Y el apuntador *p, se le asignará el valor de **25**

El método principal con un retorno de valor entero, manda a llama al método **func**, en donde realizara lo que en el contenga, por lo que cada variable realizara sus cálculos correspondientes

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión		
Representante de la Dirección	1		
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		

EJECUCIÓN DEL PROGRAMA

```

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~$ cd /home/erika/Documentos
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador2.c -o apuntador.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador2.exe
El valor de y es: 23
El valor de *p es: 25
El valor de p es: 0x7ffdf1b88ae4erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/D
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ █

```

La dirección NULL

Cuando no se desea que el apuntador apunte a algo, se le suele asignar el valor de NULL, en cuyo caso se dice que el apuntador es nulo (no apunta a nada).

NULL es una macro típicamente definida en archivos de cabecera como stddef.h y stdlib.h.

Se utiliza para proporcionar a un programa un medio de conocer cuándo un apuntador contiene una dirección inválida.

Apuntadores a apuntadores

Dado que un apuntador es una variable que apunta a otra, fácilmente se puede deducir que pueden existir apuntadores a apuntadores, y a su vez los segundos pueden apuntar a apuntadores.

```

char c = 'z';
char *pc = &c;
char **ppc = &pc;
char ***pppc = &ppc;

***pppc = 'm'

```

Dirección	Etiqueta	Contenido
...		
1502	c	z
1503	pc	1502
1504	ppc	1503
1505	pppc	1504
...		
...		

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión		
Representante de la Dirección	1		
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		

Apuntadores constantes

Es posible declarar apuntadores a constantes. De esta manera, no se permite la modificación de la dirección almacenada en el apuntador, pero si se permite la modificación del valor al que apunta.

```
int x = 5, y = 7;
int *const p = &x;
*p=3;
p=&y;
```

Dirección	Etiqueta	Contenido
...		
1502	x	5
1503	y	7
1504	*p	1502
1505		
...		
...		

Paso de parámetros por referencia

En este tipo de llamadas los argumentos contienen direcciones de variables.

Dentro de la función la dirección se utiliza para acceder al argumento real.

En las llamadas por referencia cualquier cambio en la función tiene efecto sobre la variable cuya dirección se pasó como argumento.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Bibliotecas

printf, Imprime el mensaje en pantalla, únicamente con los valores asignados en un inicio a cada variable

Llamará al método **intercambio** junto con sus argumentos y por ende se desplazará a donde se encuentre esta instrucción

Declaración del método, junto con sus argumentos

Método principal con retorno de tipo entero

Declaración de 2 variables, inicializadas con algún valor numérico

```

C a.c      C apuntador.c      C apuntador2.c      C apuntador3.c X
home > erika > Documentos > C apuntador3.c > intercambio(int *,int *)
1  #include <stdio.h>
2  void intercambio (int *a,int *b);
3
4  int main(void){
5      int x=2;
6      int y=5;
7
8      printf("Antes x= %d, y = %d\n", x,y);
9      intercambio (&x,&y);
10     printf("Después x= %d, y = %d\n", x,y);
11
12 }
13
14 void intercambio (int *a,int *b){
15     int temp;
16     temp= *b;
17     *b=*a;
18     *a=temp;
19 }
  
```

Cuando termine la ejecución del método, ahora solo imprimirá el intercambio de variables

Cuando el método es llamado, entonces ejecutará las instrucciones que se encuentran en él, junto con sus argumentos, que en este caso se declararan como **apuntadores** pero estaremos consientes que estos argumentos toman la ubicación en memoria de las variables que se asignaron en la parte de arriba.

Inicializaremos una variable de tipo entero **temp**, donde guardaremos el contenido del apuntador ***b**, en donde **b** tiene la ubicación en memoria, pero en este caso guardará lo que se encuentre en aquella ubicación que en este caso es la ubicación de **y** asignada en la parte de arriba (5).

El apuntador ***b**, guardará el contenido de **a** cuerdo a la ubicación en memoria que en ella este indicado, por lo que en este caso es en la variable **y** y asignará **2**.

Ahora el apuntador ***a**, cambiara de posición por lo que este guardado en la variable **temp**, que en este caso toma el valor de **x 5**.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión	1	
Representante de la Dirección	Fecha de revisión	7 de Febrero de 2017	
Autorizó			
Director General del Tecnológico de Estudios Superiores de Jilotepec			

EJECUCIÓN DEL PROGRAMA

```
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~$ cd /home/erika/Documentos
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador3.c -o apuntador3.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador.exe
Antes x= 2, y = 5
Después x= 5, y = 2
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$
```

La función sizeof() __codigo

Devuelve el tamaño en bytes que ocupa un tipo o variable en memoria.

Bibliotecas

Método principal

Declaración de un arreglo de tipo **char**, con 10 posiciones.

printf, imprime el mensaje en pantalla, seguido del especificador de conversión de acuerdo al tamaño en memoria en donde se almacena cada carácter utilizado

```

1  #include <stdio.h>
2
3  int main(){
4      char cadena[10];
5
6      printf("Un int ocupa %ld bytes\n", sizeof(int));
7      printf("Un char ocupa %ld bytes\n", sizeof(char));
8      printf("Un float ocupa %ld bytes\n", sizeof(float));
9      printf("Un double ocupa %ld bytes\n", sizeof(double));
10     printf("Una cadena ocupa %ld bytes\n", sizeof(cadena));
11 }
      
```

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

EJECUCIÓN DEL PROGRAMA

```

Documentos$ gcc apuntador4.c -o apuntador4.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador4.c -o ap
untador4.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador4.exe
Un int ocupa 4 bytes
Un char ocupa 1 bytes
Un float ocupa 4 bytes
Un double ocupa 8 bytes
Una cadena ocupa 10 bytes
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$

```

DEMOSTRACIÓN

Dirección	Contenido
1000	int
1001	
1002	
1003	
1004	char
1005	float
1006	
1007	
1008	
1009	double
1010	
1011	
1012	
1013	
1014	
1015	
1016	

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Ejemplo:

1000	
1001	miArreglo[0]
1002	
1003	
1004	
1005	miArreglo[1]
1006	
1007	
1008	
1009	miArreglo[2]
1010	
1011	
1012	
1013	miArreglo[3]
1014	
1015	
1016	

Int miArreglo[4]={1,2,3,4}

Sizeof(miArreglo)->16

Declaración de un arreglo con 10 posiciones en las cuales son asignadas

Devuelve el tamaño en bytes que ocupa un tipo o variable en memoria.

Bibliotecas

Método principal

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int array [10]= {1,2,3,4,5,6,7,8,9,0};
6      int len = sizeof (array)/sizeof(int);
7      printf("Los bytes del arreglo son: %ld\n", sizeof(array));
8      printf("Cada entero tiene: %ld bytes\n", sizeof(int));
9      printf("El arreglo tiene %d elemento\n", len);
10
11

```

En la variable **len**, guardará el tamaño en bytes del arreglo y este se dividirá entre el tamaño en bytes de un entero

Sizeof(array), imprimirá los bytes que ocupa el arreglo

len, es la variable en donde se realizó la división de los bytes

Sizeof(int), imprimirá los bytes que ocupa un entero

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión 1		
Representante de la Dirección			
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		

EJECUCIÓN DEL PROGRAMA

```
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~$ cd /home/erika/Documentos
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador5.c -o apuntador5.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador5.exe
Los bytes del arreglo son: 40
Cada entero tiene: 4 bytes
El arreglo tiene 10 elemento
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$
```

Asignación dinámica de memoria

Los programas pueden crear variables globales o locales.

Las variables declaradas globales en sus programas se almacenan en posiciones fijas de memoria (segmento de datos) y todas las funciones pueden utilizar estas variables.



Las variables locales se almacenan en la pila (stack) y existen solo mientras están activas las funciones donde están declaradas.

En ambos casos el espacio de almacenamiento se reserva en el momento de compilación del programa.

Para asignar memoria dinámicamente se utilizan las funciones malloc() y free(), definidas típicamente en el archivo stdlib.h.



LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión 1		
Representante de la Dirección			
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		

 GOBIERNO DEL ESTADO DE MÉXICO	ISO 9001:2015 MANUAL DE PRÁCTICAS FO-TESJI-11100-12	 TESJI TECNOLÓGICO DE ESTUDIOS SUPERIORES JILOTEPEC
---	--	---

free()

La función free() permite liberar la memoria reservada a través de un apuntador.

void free (void* ptr);

ptr es un puntero de cualquier tipo que apunta a un área de memoria reservada previamente con malloc.

malloc()

La función malloc() reserva memoria y retorna su dirección, o retorna NULL en caso de no haber conseguido suficiente memoria.

Void *malloc(size_t tam_bloque)

malloc() reserva memoria sin importar el tipo de datos que almacenará en ella.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Ejemplo:

Bibliotecas

Declaración de dos variables de tipo entero **i,n**

Declaración de un apuntador de tipo char llamado ***buffer**

printf, imprime el mensaje en pantalla, en donde introducirá la longitud de la cadena y se guardará en la variable **i**

En la variable **buffer** indica que almacenará valores de tipo **char**, con **malloc** reservará valores en memoria y **i+1** es para incrementar a 1 ya que en el arreglo inicia en 0

```

e > erika > Documentos > C apuntador6.c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    int i,n;
    char *buffer;

    printf("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer=(char*)malloc(i+1);
    if(buffer==NULL) exit(1);

    for(n=0;n<i;n++){
        buffer[n]=rand()%26+'a';
        buffer[i]='\0';

        printf("Random string: %s\n", buffer);
        free (buffer);
    }

```

Método principal con retorno de tipo entero

En la variable **buffer** almacenará la posición del arreglo resultante en donde **rand** tiene 26 elementos, comenzando de la letra **a**, y así hasta que nuestro arreglo se llene

Condicionamos **si** la variable **buffer** es igual a null (si no tiene algún valor en memoria), terminará el programa.

En el ciclo **for**, inicializamos la variable **n** en 0 y si **n<i**(longitud de la cadena), entonces incrementará a 1, hasta llegar al valor de **i**.

Al termino del programa solo libera lo que estaba dentro de la memoria

Ahora solo imprimirá lo que está dentro de la variable **buffer** (lo que está dentro del arreglo)

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

EJECUCIÓN DE PROGRAMA

```
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~$ cd /home/erika/Documentos
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador6.c -o ap
untador6.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador6.exe
Teclea la longitud de la cadena5
Random string: nwlrb
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$
```

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

**Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.
Llena todos los elementos del arreglo con datos ingresados por el usuario.
Muestra los valores**

Bibliotecas

Declaración de 3 variables
de tipo entero **x,i,val**
Declaración del apuntador
***p y *buffer**

puts, manda el mensaje
en pantalla para ingresar el
tamaño del arreglo y lo
guarda en la variable **x**

Método
principal

En el ciclo **for**,
incrementará cada
posición del arreglo
hasta que este llegue a
los indicados en un
principio, para después
almacenarlo en la
variable **val**.
Ahora en la variable **p**,
almacenará el valor de
val.
En la variable **buffer**,
almacenará la posición
del arreglo y el valor
almacenado

```

C apuntador5.c  C apuntador6.c  C apuntador7.c x  C apuntador8.c
ne > erika > Documentos > C apuntador7.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int x,i,val;
6     int *p;
7     int *buffer;
8
9     puts("Ingresa el tamaño del arreglo: ");
10    scanf("%d", &x);
11
12    buffer = (int*) malloc ((x+1)*sizeof (int));
13    printf("Ingrese los %d elementos del arreglo\n", x);
14
15    for(i=0;i<x;i++){
16        scanf("%d", &val);
17        p= &val;
18        buffer[i]=*p;
19    }
20    printf("***VALORES DEL ARREGLO***\n");
21    for(i=0;i<x;i++){
22        printf("%d ", buffer[i]);
23    }
24    printf("\n");
25    return 0;
26
27

```

En la variable **buffer**
indica que almacenará
valores de tipo **int**, con
malloc reservará valores
en memoria y **i+1** es para
incrementar a 1 ya que en
el arreglo inicia en 0,
seguido de **Sizeof(int)**,
que imprimirá los bytes
que ocupa un entero

Mandaré el mensaje en
pantalla para ingresar los
elementos de acuerdo a
los espacios
almacenados en
memoria y los
almacenará en **x**

Ahora imprimirá los valores
del arreglo, para ello
declararemos el ciclo **for**,
en donde imprimirá la
posición de acuerdo al
tamaño del arreglo, y
seguido de lo que se
almaceno de acuerdo a lo
escrito por el usuario

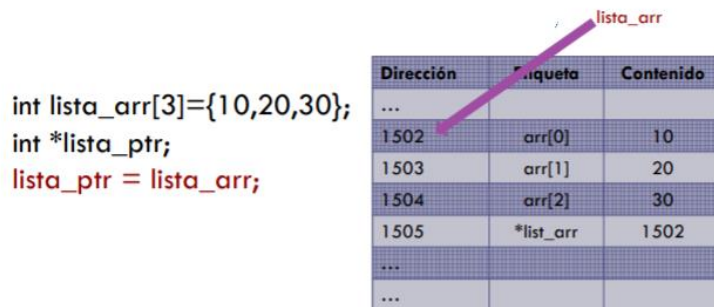
LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión	1	
Representante de la Dirección	Fecha de revisión	7 de Febrero de 2017	
Autorizó			
Director General del Tecnológico de Estudios Superiores de Jilotepec			

EJECUCIÓN DEL PROGRAMA

```
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador7.c -o apuntador7.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador7.exe
Ingresa el tamaño del arreglo:
7
Ingresa los 7 elementos del arreglo
5
6
7
8
2
3
4
***VALORES DEL ARREGLO***
5 6 7 8 2 3 4
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$
```

Apuntadores a arreglos

El nombre de un arreglo es simplemente un apuntador constante al inicio del arreglo



 Se crea el apuntador lista_ptr para poder ir modificando la dirección a donde apunta

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Arreglos

- `miArreglo = 1000`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`

1000		
1001		
1002	miArreglo[0]	
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016	variable	

Direcciones del arreglo

Bibliotecas

Declaración de 2 arreglos uno de tipo **entero** con 10 posiciones y otro de tipo **float** de igual manera con 10 posiciones

Método principal con retorno de tipo entero

```

C apuntador5.c  C apuntador6.c  C apuntador7.c  C apuntador8.c
e > erika > Documentos > C apuntador8.c
#include <stdio.h>
#include <stdlib.h>

int i[10],x;
float f[10];

int main(void){
    printf("\t\tEntero\t\tFlotante\n\n");
    for(x=0;x<10;x++){
        printf("Elemento %d:\t%p\t%p\n", x, &i[x],&f[x]);
    }
}

```

printf, imprime el mensaje en pantalla en donde indica el valor en memoria de tipo **entero** y tipo **flotante**

Ahora declaramos al ciclo **for**, en donde indica que si la variable **x=0** y menor que cada uno de los arreglos declarados, entonces imprimirá las 10 posiciones, seguido del valor asignado en memoria para cada arreglo.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión 1		
Representante de la Dirección			
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		

EJECUCIÓN DEL PROGRAMA

```
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador8.c -o ap
untador8.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador8.exe
Entero          Flotante

Elemento 0:      0x55c583b100a0      0x55c583b10060
Elemento 1:      0x55c583b100a4      0x55c583b10064
Elemento 2:      0x55c583b100a8      0x55c583b10068
Elemento 3:      0x55c583b100ac      0x55c583b1006c
Elemento 4:      0x55c583b100b0      0x55c583b10070
Elemento 5:      0x55c583b100b4      0x55c583b10074
Elemento 6:      0x55c583b100b8      0x55c583b10078
Elemento 7:      0x55c583b100bc      0x55c583b1007c
Elemento 8:      0x55c583b100c0      0x55c583b10080
Elemento 9:      0x55c583b100c4      0x55c583b10084
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$
```

ARITMÉTICA DE OPERADORES

Incremento de operadores

Cuando se incrementa un apuntador se está incrementando su valor.

Si incrementamos en 1 el valor del apuntador, C sabe el tipo de dato al que apunta e incrementa la dirección guardada en el apuntador en el tamaño del tipo de dato.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Arreglos

- ☐ `miArreglo = 1000`
- ☐ `p_miArreglo = miArreglo`
- ☐ `&miArreglo[0] = 1000`
- ☐ `&miArreglo[1] = 1004`

1000		
1001	miArreglo[0]	
1002		
1003		
1004	miArreglo[1]	
1005		
1006		
1007	miArreglo[2]	
1008		
1009		
1010	miArreglo[3]	
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016	p_miArreglo	1000

`miArreglo = 1000`
`p_miArreglo = miArreglo`
`&miArreglo[0] = 1000`
`&miArreglo[1] = 1004`
`p_miArreglo ++;`

1000		
1001	miArreglo[0]	
1002		
1003		
1004	miArreglo[1]	
1005		
1006		
1007	miArreglo[2]	
1008		
1009		
1010	miArreglo[3]	
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016	p_miArreglo	1004

¿Ahora que valor
tiene p_miArreglo?



LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión		
Representante de la Dirección	1		
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo ++;`

1000		
1001		
1002	miArreglo[0]	
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016	p_miArreglo	1004

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo += 2;`

1000		
1001		
1002	miArreglo[0]	
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016	p_miArreglo	1004

¿Y ahora?



- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo += 2;`

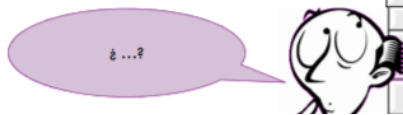
1000		
1001		
1002	miArreglo[0]	
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016	p_miArreglo	1004

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión	1	
Representante de la Dirección	Fecha de revisión	7 de Febrero de 2017	
Autorizó			
Director General del Tecnológico de Estudios Superiores de Jilotepec			

```

1 miArreglo = 1000
2 p_miArreglo = miArreglo
3 &miArreglo[0] = 1000
4 &miArreglo[1] = 1004
5 p_miArreglo --;

```



1000		
1001	miArreglo[0]	
1002		
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016		
	p_miArreglo	1000

```

miArreglo = 1000
p_miArreglo = miArreglo
&miArreglo[0] = 1000
&miArreglo[1] = 1004
p_miArreglo --;

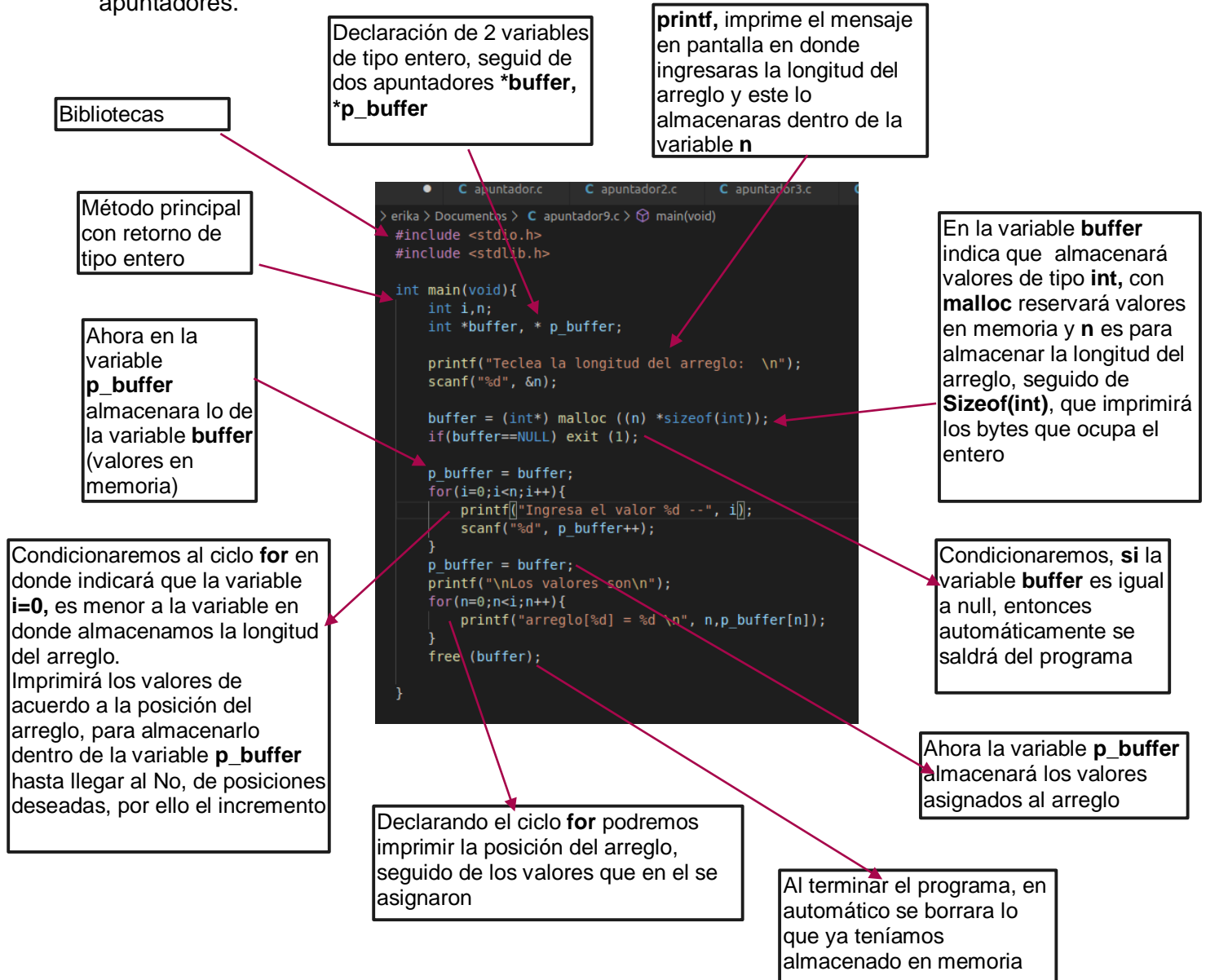
```

1000		
1001	miArreglo[0]	
1002		
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016		
	p_miArreglo	1000

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión	1	
Representante de la Dirección			
Autorizó	Fecha de revisión	7 de Febrero de 2017	
Director General del Tecnológico de Estudios Superiores de Jilotepec			

Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.

Llena todos los elementos del arreglo con datos ingresados por el usuario usando apuntadores.



LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión 1		
Representante de la Dirección			
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jirotepec		7 de Febrero de 2017	

EJECUCIÓN DEL PROGRAMA

```

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ gcc apuntador9.c -o apuntador9.exe
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$ ./apuntador9.exe
Teclea la longitud del arreglo:
5
Ingresa el valor 0 --6
Ingresa el valor 1 --8
Ingresa el valor 2 --9
Ingresa el valor 3 --2
Ingresa el valor 4 --5

Los valores son
arreglo[0] = 6
arreglo[1] = 8
arreglo[2] = 9
arreglo[3] = 2
arreglo[4] = 5
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$

```

Código

Crea un arreglo de tipo char de tamaño x, en donde x es ingresado por teclado.

Llena elemento por elemento del arreglo con letras ingresados por el usuario.

Muestra el arreglo impreso en forma inversa.

Todo debe ser manejado con apuntadores.

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):		Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró		Versión		
Representante de la Dirección		1		
Autorizó		Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec		7 de Febrero de 2017		

Bibliotecas

Método principal
con retorno de
tipo entero

Ahora en la
variable
p_buffer
almacenara lo de
la variable **buffer**
(valores en
memoria)

Condicionaremos al ciclo **for** en
donde indicará que la variable
i=0, es menor a la variable en
donde almacenamos la longitud
del arreglo **n**.
Imprimirá los valores de
acuerdo a la posición del
arreglo, para almacenarlo
dentro de la variable **p_buffer**
hasta llegar al No. de posiciones
deseadas, por ello el incremento

Declaración de 2 variables
de tipo entero, seguid de
dos apuntadores ***buffer**,
***p_buffer**

printf, imprime el mensaje
en pantalla en donde
ingresaras la longitud del
arreglo y este lo
almacenaras dentro de la
variable **n**

```

erika > Documentos > C apuntador10.c > main(void)
#include<stdio.h>
#include<stdlib.h>

int main(void){
    int i,n;
    char *buffer, *p_buffer;

    printf("Teclea la longitud del arreglo: ");
    scanf("%d", &n);

    buffer = (char*) malloc ((n) *sizeof(char));
    if(buffer==NULL) exit(1);

    p_buffer=buffer;
    for(i=0;i<n;i++){
        printf("Ingresa el valor %d: ", i);
        scanf("%s", p_buffer++);
    }
    p_buffer=buffer;
    printf("\n***LOS VALORES SON***\n");
    for(n=i-1;n>=0;n--){
        printf("Arreglo[%d] = %c\n", n,p_buffer[n]);
    }
    free (buffer);
}

```

En la variable **buffer**
indica que almacenará
valores de tipo **int**, con
malloc reservará valores
en memoria y **n** es para
almacenar la longitud del
arreglo, seguido de
Sizeof(int), que imprimirá
los bytes que ocupa el
entero

Condicionaremos, **si** la
variable **buffer** es igual
a null, entonces
automáticamente se
saldrá del programa

Ahora la variable **p_buffer**
almacenará los valores
asignados al arreglo

Ahora imprimiremos los resultados
obtenidos, por lo que declarando el
ciclo **for** en donde podremos imprimir
la posición del arreglo, seguido de los
valores que en el se asignaron, pero
en este caso de forma descendente

Al terminar el programa, en
automático se borrara lo
que ya teníamos
almacenado en memoria

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión	1	
Representante de la Dirección	Fecha de revisión	7 de Febrero de 2017	
Autorizó			
Director General del Tecnológico de Estudios Superiores de Jilotepec			

EJECUCIÓN DEL PROGRAMA

```

Teclea la longitud del arreglo: 10
Ingresa el valor 0: E
Ingresa el valor 1: R
Ingresa el valor 2: I
Ingresa el valor 3: K
Ingresa el valor 4: A
Ingresa el valor 5: Y
Ingresa el valor 6: A
Ingresa el valor 7: Z
Ingresa el valor 8: M
Ingresa el valor 9: I

***LOS VALORES SON***
Arreglo[9] = I
Arreglo[8] = M
Arreglo[7] = Z
Arreglo[6] = A
Arreglo[5] = Y
Arreglo[4] = A
Arreglo[3] = K
Arreglo[2] = I
Arreglo[1] = R
Arreglo[0] = E
erika@erika-Lenovo-ideapad-Y700-Touch-15ISK:~/Documentos$

```

IV. CONCLUSIONES:

Como conclusión puedo argumentar que los apuntadores son de suma importancia, ya que gracias a ellos es cómo podemos saber que parte de nuestra memoria es donde se está almacenando cada valor, es como si nos introdujéramos más a fondo de nuestra máquina y así poder disminuir o aumentar la forma en la que codificamos cada programa, así que si queremos saber la dirección de memoria que ocupa cada variable declarada lo único que tenemos que hacer es utilizar un apuntador

LUGAR DE REALIZACIÓN DE LA PRÁCTICA (LABORATORIO/TALLER/AULA):	Salón de clases y hogar	DURACIÓN DE LA PRÁCTICA (HRS):	4
Elaboró	Versión 1		
Representante de la Dirección			
Autorizó	Fecha de revisión		
Director General del Tecnológico de Estudios Superiores de Jilotepec	7 de Febrero de 2017		