Nombre de la práctica	Ejercicios en Lenguaje C			No.	4
Asignatura:	Métodos Numéricos	Carrera:	ISIC	Duración de la práctica (Hrs)	

Nombre: Rodrigo Javier Martínez **Grupo: 3041**

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):

- Aula y casa

III. Material empleado:

- Dev-C++
- Word

IV. Desarrollo de la práctica:

Esta es la continuación del curso de Lenguaje C que llevamos a cabo en Métodos Numéricos, en donde estamos realizando ejercicios en este lenguaje.
Para comenzar vamos a ver que es un apuntador, es un objeto que apunta a otro objeto. Es decir, una variable cuyo valor es la dirección de memoria de otra variable.

- ¿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.
Un puntero tiene su propia dirección de memoria.
La sintaxis es la siguiente:
<tipo de dato apuntador> *<identificador del apuntador>

- 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 diferenciación?

La diferenciación 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

- ¿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

- 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 por que ese espacio de memoria fue reservado dinámicamente.

Ejemplo:

```c
#include <stdio.h>
#include <stdlib.h>

int *p,y;
void func(){
 int x = 40;
 p = &x; //correcto
 y = *p; //corecto
 *p = 23;
}
int main(void){
 func();
 y = *p; //incorrecto
 *p = 25; //incorrecto
 printf("El valor de y es %d\n",y);
 printf("El valor de *p es %d\n",*p);
 printf("El valor de p es %p\n",p);
}
```

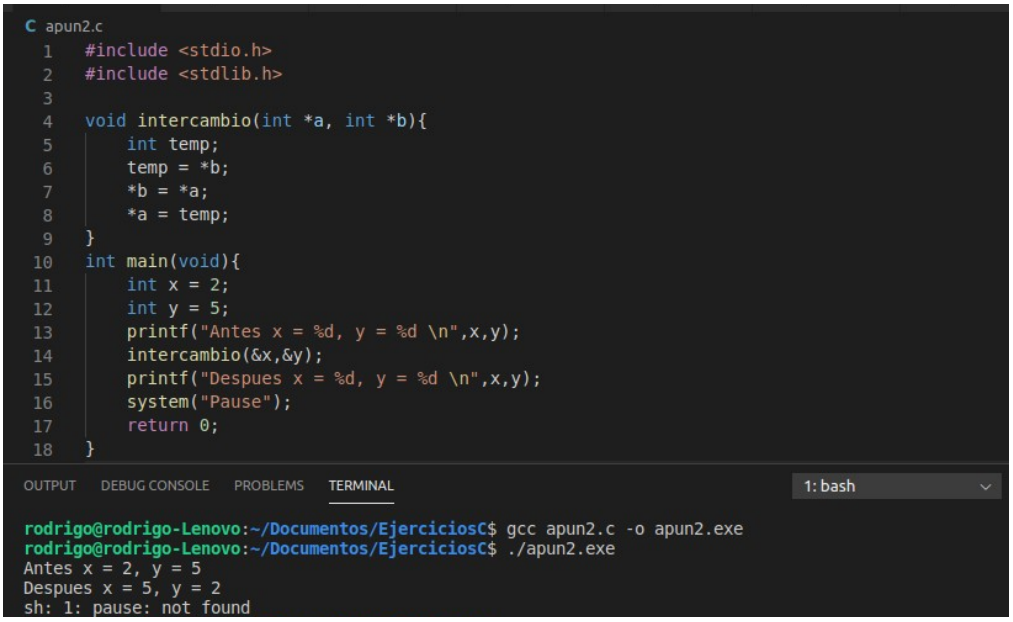
```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL

rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ gcc apun1.c -o apun1.exe
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ ./apun1.exe
El valor de y es 23
El valor de *p es 25
El valor de p es 0x7ffeed2120e4
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$
```

- 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 se cabecera como stdef.h y stdlib.h.
Se utiliza para proporcionar a un programa un medio de conocer cuándo un apuntador contiene una dirección inválida.

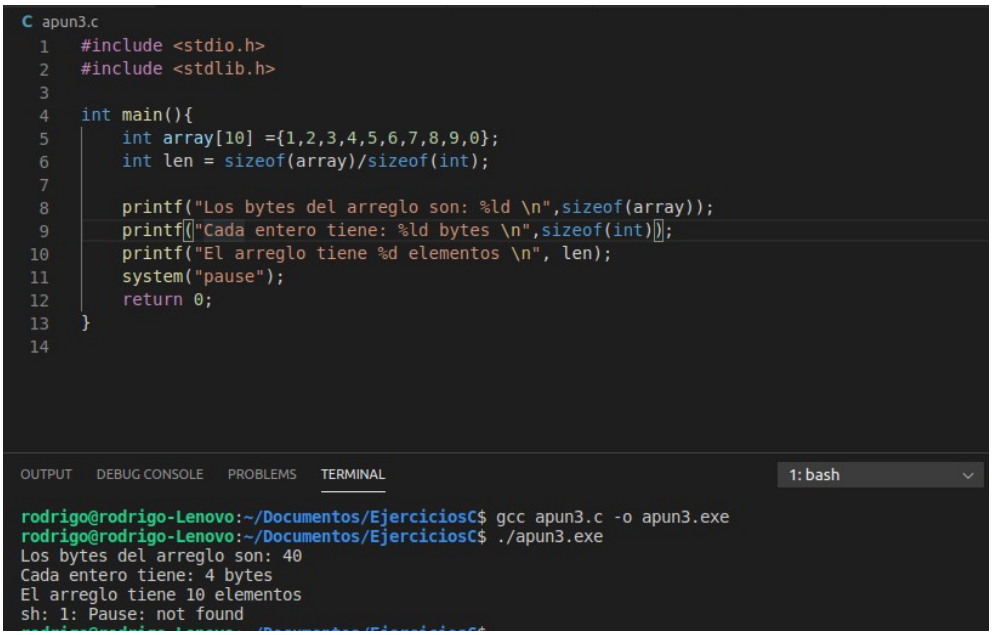
Paso de parámetros por referencia
Ejemplo:



- Función sizeof

Devuelve el tamaño en Bytes que ocupa un tipo de variable en memoria.

Ejemplo:



- Asignación dinámica de memoria

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

- free()

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

void free (void* ptr);

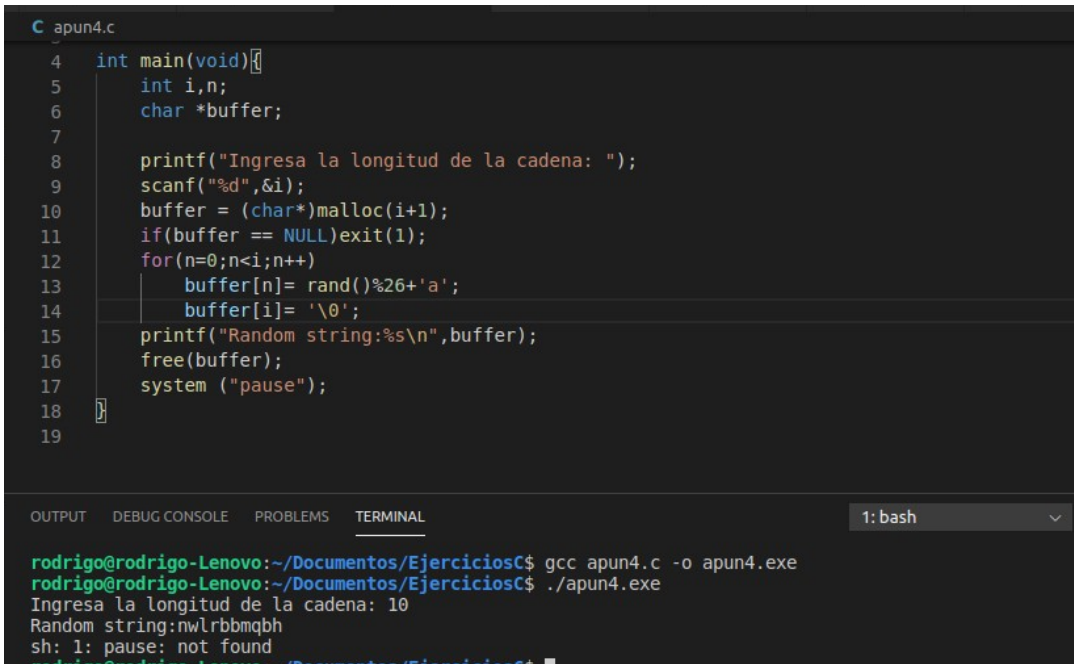
ptr es un puntero de cualquier tipo que apunta a un área de memoria reservada previamente con malloc.

- malloc()

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

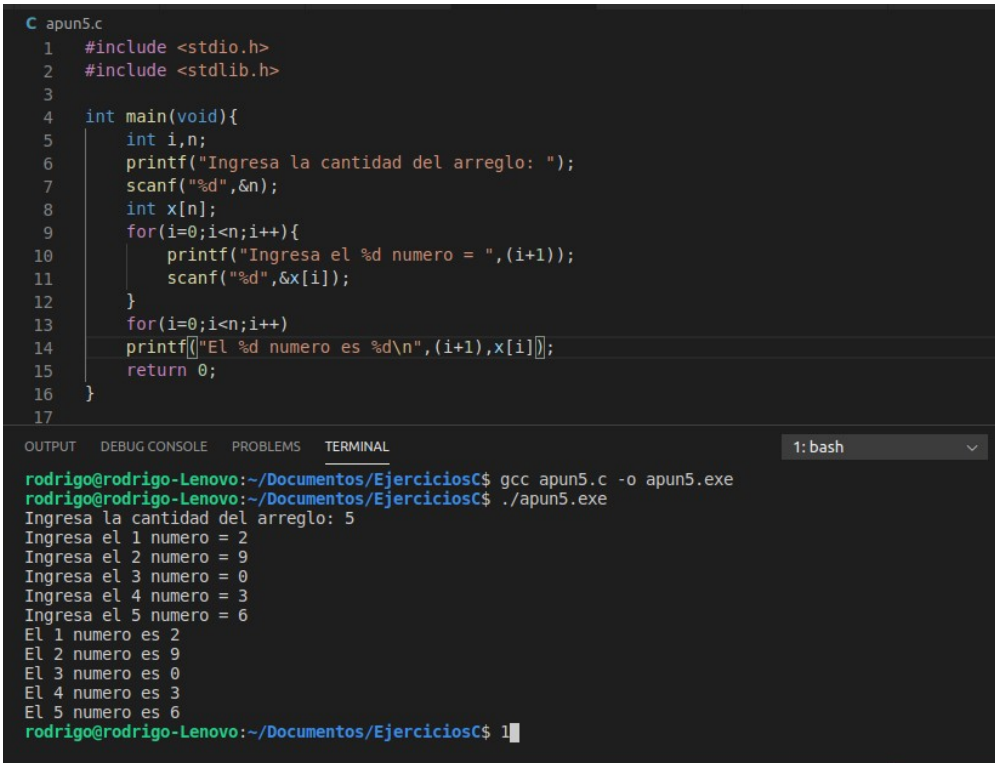
Void *malloc(size_t tam_bloque)

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



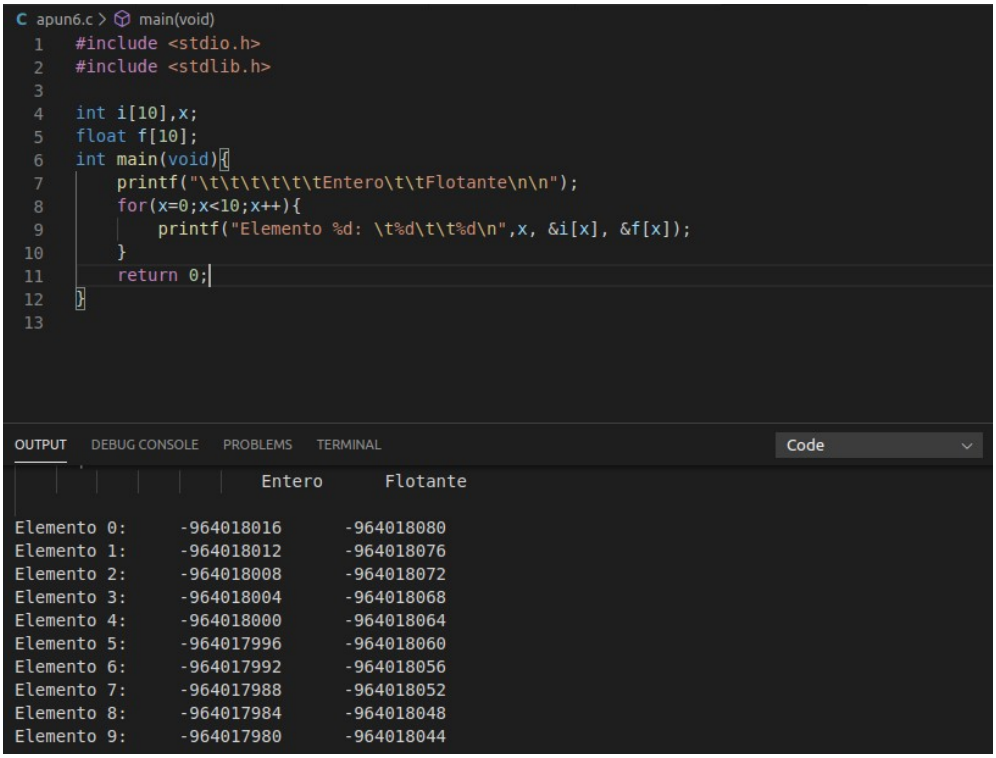
Ejercicio:

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



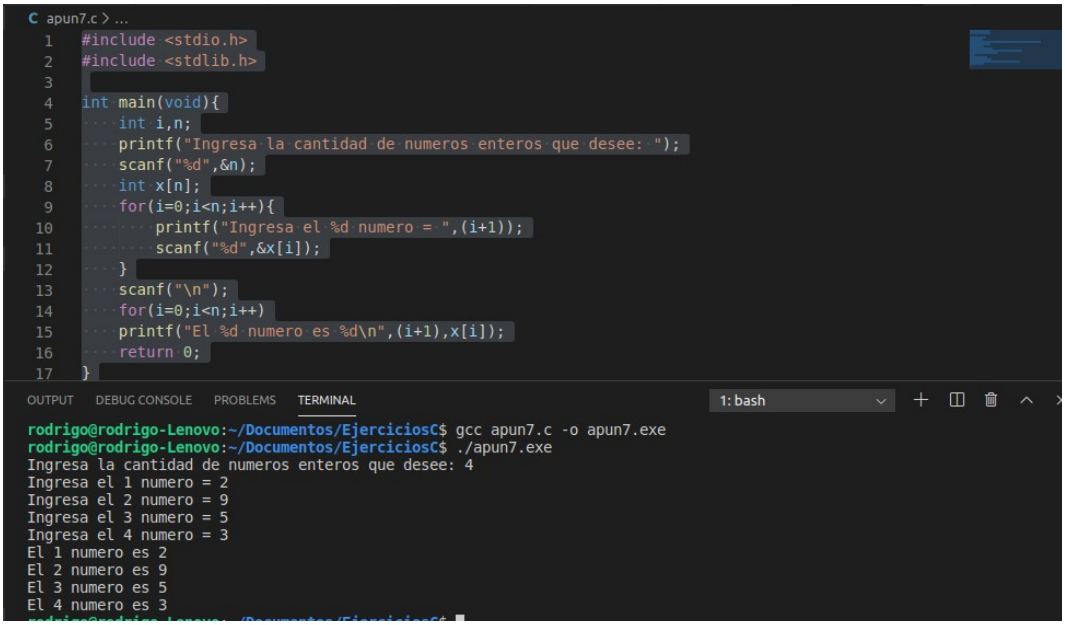
- Apuntadores a arreglos

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



Ejercicio:

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.



```c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    int i,n;
    printf("Ingresa la cantidad de numeros enteros que desee: ");
    scanf("%d",&n);
    int x[n];
    for(i=0;i<n;i++){
        printf("Ingresa el %d numero = ",(i+1));
        scanf("%d",&x[i]);
    }
    scanf("\n");
    for(i=0;i<n;i++)
    printf("El %d numero es %d\n",(i+1),x[i]);
    return 0;
}
```

```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL                    1: bash

rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ gcc apun7.c -o apun7.exe
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ ./apun7.exe
Ingresa la cantidad de numeros enteros que desee: 4
Ingresa el 1 numero = 2
Ingresa el 2 numero = 9
Ingresa el 3 numero = 5
Ingresa el 4 numero = 3
El 1 numero es 2
El 2 numero es 9
El 3 numero es 5
El 4 numero es 3
```

Ejemplo:

apun8.c

```c
#include <stdio.h>
#include <stdlib.h>

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

    puts("Ingresa la longitud del arreglo");
    scanf("%d",&n);
    buffer = (int*) malloc((n)* sizeof(int));
    if(buffer==NULL)exit(1);
    p_buffer = buffer;

    for(i=0;i<n;i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d",p_buffer);
    }
    p_buffer = buffer;
    printf("\nLos valores son\n");
    for(n=0;n<i;n++){
        printf("arreglo[%d] = %d \n", n,*p_buffer);
    }
    free(buffer);
    system("pause");
}
```

```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ gcc apun8.c -o apun8.exe
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ ./apun8.exe
Ingresa la longitud del arreglo
4
Ingresa el valor 0
5
Ingresa el valor 1
7
Ingresa el valor 2
1
Ingresa el valor 3
3

Los valores son
arreglo[0] = 3
arreglo[1] = 3
arreglo[2] = 3
arreglo[3] = 3
sh: 1: pause: not found
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$
```

Ejercicio:
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.

apun9.c

```c
#include <stdio.h>
#include <stdlib.h>

int main (void){
    int i,n,letra;
    char *buffer, *p_buffer;
    printf("Ingresa la cantidad de letras: ");
    scanf("%d",&n);
    buffer=(char*)malloc((n)* sizeof(char));
    if(buffer==NULL){
        exit(1);
    }
    p_buffer=buffer;
    for(i=0;i<n;i++){
        puts("    ");
        scanf("%c",p_buffer);
        printf("Ingresa una letra %d\n",i+1);
        scanf("%c",p_buffer++);
    }
    p_buffer=buffer;
    printf("\nLa palabra es: %s\n",p_buffer);
    puts("La palabra invertida es: ");
    for(i=(n-1);i>=0;i--){
        printf("%c",p_buffer[i]);
    }
    puts("\n");
    free(buffer);
    return 0;
}

```

```
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ gcc apun9.c -o apun9.exe
rodrigo@rodrigo-Lenovo:~/Documentos/EjerciciosC$ ./apun9.exe
Ingresa la cantidad de letras: 4

Ingresa una letra 1
h

Ingresa una letra 2
o

Ingresa una letra 3
l

Ingresa una letra 4
a

La palabra es: hola
La palabra invertida es:
aloh
```

V. Conclusiones:

Para concluir podemos decir que el lenguaje de programación C es uno de los más completos y complejos de utilizar ya que cuenta con muchas funciones las cuales nos pueden ayudar a resolver distintos problemas ya que los puedes implementar en la programación y sacarle provecho a esta. También agregar que el lenguaje C es la base casi toda la programación que su estructura es muy similar a las de los demás lenguajes.