Instituto Superior Roque González

Tecnicatura Superior en Desarrollo de Software

Sistemas Operativos

Villalba Carlos

Programación en C

Trabajo Práctico Nº7

Alumnos:

Georgieff Matías

Pedrozo Octavio

Consignas

- Reescribir en C cada uno de los códigos que aparecen en las capturas de pantalla
- entregadas.
- Compilar y ejecutar cada programa utilizando gcc
- Tomar una captura de pantalla de la salida que muestre el programa funcionando
- correctamente.
- Para cada ejercicio, elaborar un documento que contenga:
    - La captura del código fuente.
    - La captura de la salida del programa.
    - Una explicación breve y clara de lo que hace el código y cómo lo hace (máximo 5 renglones por ejercicio).
- Subir todo a un repositorio en GitHub con la siguiente estructura:
    - /NombreDelRepositorio
        - ejercicio1.c
        - ejercicio1.exe
        - ejercicio2.c
        - ejercicio2.exe
        - capturas.pdf
    - README.md (opcional, pero recomendable)

- El README puede incluir un resumen del trabajo, su nombre, y los objetivos alcanzados.

<u>Recomendaciones</u>

- Usar comentarios en el código para facilitar la comprensión.
- Los binarios deben poder ejecutarse sin errores.
- El nombre del repositorio en GitHub debe ser claro, por ejemplo: tp-lenguaje-c-ejercicios.

Desarrollo

Ejercicio 1

```c
#include <stdio.h>

void cambiar (int *n){
    *n=10;
}

int main()
{
    int x;
    x=5;
    cambiar (&x);
    printf ("%d\n", x);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio1.exe
10
```

Se referenció el valor de **x** (5), pasándolo a la variable **n** de la función “cambiar”, esta al tener puntero retorna el valor 10 y se lo pasa a la variable **x**. al final se imprime el valor de **x**.

Ejercicio 2

```c
#include <stdio.h>

void cambiar (int n){
    n=10;
}

int main()
{

    int x;
    x=5;
    cambiar (x);
    printf ("%d\n", x);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio2.exe
5
```

Se referenció el valor de **x** (5), pasándolo a la variable **n** de la función “cambiar”, pero esta vez al no tener puntero no retorna el valor 10, dejando a la variable **x** con su mismo valor en la función principal (5). al final se imprime el valor de **x**.

Ejercicio 3

```c
int main(void)
{
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio3.exe
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C>
```

Estructura principal de C, donde esta la función **main** en donde va el código principal, al retornar 0 el programa no da error. En este ejemplo, al no contener nada dentro de la función no muestra nada en salida.

Ejercicio 4

```c
#include <stdio.h>
int main (void){
    printf ("Hola mundo \n");
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio4.exe
Hola mundo
```

Ejemplo de un “Hola mundo” en C, primero se incluye la libreria **#include <stdio.h>** el cual permite la entrada y salida de datos, luego con la función **printf** se puede generar una salida. Con **\n** se realiza un salto de línea.

Ejercicio 5

```c
#include <stdio.h>

int main(void)
{
    int numero = 5;
    printf ("%d", numero);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio5.exe
5
```

Ejemplo de como se puede imprimir una variable entera en C, dentro de **printf** se coloca **%d** el cual sera reemplazado por la variable que se coloque al final (en este caso, al ser un entero se coloca **%d**).

Ejercicio 6

```c
#include <stdio.h>

int numero = 10; // variable global

void mostrarnumero() {
    printf ("El numero es: %d\n", numero);
}

int main()
{
    mostrarnumero();
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio6.exe
El numero es: 10
```

Ejemplo de cómo se declara una variable global, esta misma al estar declara fuera de la función principal la vuelva **global**, existiendo para que todas las funciones las cuales la llamen. en este ejemplo se llama la en la función principal la función **mostrarnumero,** el cual muestra la variable **numero**, en este caso, 10.

Ejercicio 7

```c
#include <stdio.h>

void mostrarmensaje() {
    int numero = 5; // variable local
    printf ("El numero local dentro de la funcion es: %d\n", numero);
}

int main()
{
    mostrarmensaje(); // llama a la funcion que usa su propia variable local
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio7.exe
El numero local dentro de la funcion es: 5
```

Ejemplo de una variable local, esta misma al estar declarada dentro de una función, lo cual la hace que solo exista dentro de esa función, es decir, si se llama a la variable desde otra función, esta no existirá para esa función. El código llama en la en la función principal la función **mostrarmensaje,** el cual muestra la variable **numero**, en este caso va a ser 5

Ejercicio 8

```c
#include <stdio.h>

int main()
{
    const float PI = 3.1416;
    printf ("El valor de PI es %.4f\n", PI);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio8.exe
El valor de PI es 3.1416
```

Ejemplo de cómo se declara una constante, en este caso se declara **PI** como número flotante y constante. El número se imprime mediante **printf** y usando **%.4f** para mostrar el número (donde **.4** dice cuando decimales mostrar después de la coma y **f** es la etiqueta para mostrar un número flotante en C).

Ejercicio 9

```c
#include <stdio.h>

#define PI 3.1416

int main()
{
    printf ("El valor de PI es: %.4f\n", PI);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio9.exe
El valor de PI es: 3.1416
```

Otro ejemplo de cómo se declara una constante, esta vez utilizando **#define** su diferencia con **const** es que esta función realiza un reemplazo de texto durante el preprocesamiento, antes de la compilación, y no guarda la constante en memoria. El resto del código hace lo mismo que el anterior ejemplo.

Ejercicio 10

```c
#include <stdio.h>

int cuadrado(int v){
    return (v*v);
}

int main(void)
{
    int x=3;
    int y=0;

    y=cuadrado(x);
    printf ("%d", y);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio10.exe
9
```

Ejemplo de cómo utilizar funciones, en la función principal se declaran la variables **x** e **y** , luego se pasa el valor de **x** (3) a la función **cuadrado**, el cual pasa momentáneamente su valor a **v** *(pasando de 0 a 3)* y retorna su cuadrado (*3x3*=**9**). Ell valor de ese retorno se guarda en **y** para luego imprimirlo al usuario.

Ejercicio 11

```c
#include <stdio.h>

int main(void)
{
    int x=30; float y=7.3; char caracter='A';

    printf ("Ejemplo de Printf!\n");
    printf ("El valor de x es %d y su equivalente octal es %o \n", x, x);
    printf ("El valor de y es %f,\n..., caracter vale %c. \n", y, caracter);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio11.exe
Ejemplo de Printf!
El valor de x es 30 y su equivalente octal es 36
El valor de y es 7.300000,
..., caracter vale A.
```

Ejemplo de cómo se utilizan los diferentes especificadores de formato. se declaran varias variables y se muestran con su diferente especificador de formato. **%d** es para mostrar variables enteras, **%o** para mostrar variables en formato octal (el mismo programa lo pasa a octal), **%f** para mostrar variables con números flotantes y **%c** para mostrar caracter.

Ejercicio 12

```c
#include <stdio.h>

int main(void)
{
    int x, y;

    printf("Ingrese numero: ");
    scanf("%d", &x);
    printf ("Ingrese otro numero: ");
    scanf("%d", &y);

    printf ("La division entre los valores es: %.2f\n", (float) x/y);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio12.exe
Ingrese numero: 10
Ingrese otro numero: 2
La division entre los valores es: 5.00
```

Ejemplo de cómo se utiliza **scanf**, este comando permite leer alguna entrada que se haga y guardarlo en una variable. Entre comillas va el especificador de forma del valor que se ingresará y seguido la variable con su **cursor** para dirigir el valor ingresado a esta misma. El programa termina haciendo la división de los números ingresados y mostrando el resultado.

Ejercicio 13

```c
#include <stdio.h>

int main(void)
{
    char ca;
    printf ("Eliga una opcion\n\n");
    printf ("1 - Opcion 1\n");
    printf ("2 - Opcion 2\n");
    printf ("3 - Salir\n\n");

    ca=getchar();
    printf ("\n La opcion elegida es: ");
    putchar (ca);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio13.exe
Eliga una opcion

1 - Opcion 1
2 - Opcion 2
3 - Salir

2

 La opcion elegida es: 2
```

El algoritmo escribe una serie de opciones con números, luego, mediante el comando **getchar** recupera el input del teclado que se haya hecho, registrando la tecla que se haya tocado y guardandolo en la variable **ca**, luego, con el comando **putchar** lo que ha guardado la variable **ca**, en este caso, el carácter **2**.

Ejercicio 14

```c
#include <stdio.h>

int main(void)
{
    int a=0;

    printf("Ingrese un valor: ");
    scanf ("%d", &a);

    if (a%2)
    {
        printf ("el valor ingresado es impar.\n");
    }else{
        printf ("El valor ingresado es par.\n");
    }
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio14.exe
Ingrese un valor: 2
El valor ingresado es par.
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio14.exe
Ingrese un valor: 7
el valor ingresado es impar.
```

El algoritmo pide que se ingrese un valor entero y lo guarda en la variable **a**, luego mediante un condicional IF (sabiendo que 1 es **VERDADERO** y 0 es **FALSO**) hace el módulo de la variable **a**. Si el resultado de la condición de resto 1, imprime que el número ingresado es impar. Si el resultado de la condición de resto 0, imprime que el número ingresado es par.

Ejercicio 15

```c
#include <stdio.h>

int main(void)
{
    int valor = 1;

    while (valor<5)
    {
        printf ("El doble %d es %d\n", valor, valor * 2);
        valor++;
    }
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio15.exe
El doble 1 es 2
El doble 2 es 4
El doble 3 es 6
El doble 4 es 8
```

Ejemplo de cómo se utiliza el bucle While, el cual entra en la función cuando su condición de verdadera. Este buble empieza comprobando que la variable **valor** sea **menor a 5,** cada vez que entra al bucle imprime el valor de la variable **valor,** el doble de la variable **valor** y finaliza aumentando **valor** en +1. Esto se repite hasta que la condición del bucle sea falsa.

Ejercicio 16

```c
#include <stdio.h>

int main(void)
{
    int valor;
    do
    {
        printf ("Ingrese un valor entero (0 para salir)\n");
        scanf ("%d", &valor);
        printf ("El valor ingresado es: %d\n", valor);
    } while (valor!=0);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio16.exe
Ingrese un valor entero (0 para salir)
10
El valor ingresado es: 10
Ingrese un valor entero (0 para salir)
3
El valor ingresado es: 3
Ingrese un valor entero (0 para salir)
0
El valor ingresado es: 0
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C>
```

Ejemplo de como se utiliza el bucle **Do While**, este funciona de la misma manera que el bucle While con la diferencia de que este **al menos se ejecuta una vez antes de comprobar la condición**. Este algoritmo pide que el usuario ingrese un número, imprimiendo el número que se ingresa. se sale del bucle Do While cuando el valor ingresado es **0.**

Ejercicio 17

```c
#include <stdio.h>

int main(void)
{
    int i=1;

    for (i = 1; i < 10; i++)
    {
        printf ("Linea numero: %d\n", i);
    }
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio17.exe
Linea numero: 1
Linea numero: 2
Linea numero: 3
Linea numero: 4
Linea numero: 5
Linea numero: 6
Linea numero: 7
Linea numero: 8
Linea numero: 9
```

Ejemplo de bucle for, parecido al While, con la diferencia de que este tiene un propio contador que se repite la cantidad de veces que cumplan la condición. En el algoritmo del ejemplo la variable **i** se imprime cada vez que se repite el bucle, mostrando en qué línea va. Esto lo hace hasta que la variable **i** sea **menor a 10**.

Ejercicio 18

```c
#include <stdio.h>

int main(void)
{
    int a;

    printf ("Ingrese el dia dia de la semana: ");
    scanf ("%d", &a);

    switch (a)
    {
    case 5:
        printf ("Dia laboral.\n");
        break;
    case 7:
        printf ("Dia no laboral.\n");
        break;
    default:
        printf ("Error.\n");
        break;
    }
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio18.exe
Ingrese el dia dia de la semana: 5
Dia laboral.
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio18.exe
Ingrese el dia dia de la semana: 7
Dia no laboral.
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio18.exe
Ingrese el dia dia de la semana: 2
Error.
```

Ejemplo de cómo usar un **Switch**, este funciona analizando la variable y **viendo en qué bloque de código entrar dependiendo de su valor**. En este ejemplo se pide que el usuario ingrese un día de la semana, el Switch comprueba la variable y dependiendo de su valor imprime los mensajes. Si es valor que se ingresa no está contemplado en las variables ejecute el bloque **default**

Ejercicio 19

```c
#include <stdio.h>

void reset (int *a, int b)
{
    *a=0;
    b=0;
}

int main(void)
{
    int x=1;
    int y=1;

    reset (&x,y);
    printf ("%d %d\n", x, y);
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio19.exe
0 1
```

En este ejemplo se utilizan cursores, es similar al primer ejemplo. Las variables **x** e **y** inicializan en la función principal con el valor **1**, luego se llama a la función **reset** y se pasan los valores de **x** e **y** a las variables **a** y **b**. Estas mismas toman los valores de 0, pero, al tener el puntero (*) en la variable **a**, solo se pasa a la variable **x** el valor 0. dando como salida **x=0** y **y=1**

Ejercicio 20

```c
#include <stdio.h>

int main(void)
{
    int i, vector[10];

    for (i = 0; i < 10; i++)
    {
        vector[i]= i;
        printf ("%d\n", vector[i]);
    }
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio20.exe
0
1
2
3
4
5
6
7
8
9
```

Ejemplo de arreglos en C, Se declara una variable **i** y un arreglo **vector** de 10 espacios como enteros, al entrar en el **for** el valor de **i** se guarda en el arreglo **vector** en la posición número **i**. Al terminar, imprime el valor que se encuentra en la posición **i** de **vector**. Esto se repite hasta que **i sea menor que 10**.

Ejercicio 21

```c
#include <stdio.h>

int main(void)
{
    struct persona
    {
        int legajo;
        int edad;
        char nombre [20];
        char apellido [50];
    };

    struct persona miembro_ieee1;
    struct persona miembro_ieee2 = {2, 30, "Jose", "PEREIRA"};

    printf ("Legajo: %d\n Edad:%d\n Nombre:%s\n Apellido:%s\n",
        miembro_ieee2.legajo,
        miembro_ieee2.edad,
        miembro_ieee2.nombre,
        miembro_ieee2.apellido
    );
    return 0;
}
```

```
PS C:\xampp\htdocs\EjerciciosDesarrolloSoftware\Sistema Operativo\TP programacion C> .\Ejercicio21.exe
Legajo: 2
Edad:30
Nombre:Jose
Apellido:PEREIRA
```

Ejemplo de como usar Estructuras, se declara la estructura **persona** el cual contendrá las variables de **legajo, edad, nombre** y **apellido**. Primero se crea una estructura con el nombre **miembro_ieee1** que estará vacío, luego crea otra estructura con el nombre **miembro_ieee2** en donde se le cargaran los datos. Por último imprime los datos guardados en las variables de la estructura **miembro_ieee2**.