PRÁCTICA 0

1. ESTRUCTURAS DE CONTROL: CONDICIONALESS Y CICLOS DE REPETICIÓN

1) Escribir un programa que lea dos números y visualice el mayor, utillizar el operador ternario `? :`.
2) Escribir y ejecutar un programa que lea números enteros hasta leer el número -1 y muestre su máximo y mínimo. Además imprimir la cantidad de números leídos y el promedio.
3) A partir del código siguiente, determinar qué hace y escribir el mismo código utilizando la sentencia switch case:

Algorithm 1: Segundo Ejemplo.

```
1   if ((car == 'a') || (car == 'A')
2       printf ( "%c es es una vocal ", car);
3   else if ((car == 'e') || (car == 'E')
4               printf ( "%c es una vocal ", car);
5           else if ((car == 'i') || (car == 'I')
6                   printf ( "%c es una vocal ", car);
7               else if ((car == 'o') || (car == 'O')
8                       printf ( "%c es una vocal ", car);
9                                   else if ((car == 'u') || (car == 'U')
10                                      printf ( "%c es una vocal ", car);
11                                  else
12                                      printf("%c no es una vocal", car);
```

4) Definir qué realiza el código siguiente. Programar y ejecutar un programa similar pero utilizando otra estructura de control para realizar el ciclo de iteración. Tenga en cuenta que el código ASCII de la letra 'a' es 97.

Algorithm 2: Tercer Ejemplo.

```
1   char car = 'a';
2   do
3   {
4       printf("%d", car);
5       car ++;
6   }while (car <= 'z');
```

2. EVALUACIÓN EN CORTOCIRCUITO DE EXPRESIONES LÓGICAS

1) Si `x` es negativo, la expresión `(x >= 0) && (y >1)` se evalúa en cortocirucuito. ¿Qué significa esto? ¿Cómo se evalúa la expresión si `x = 0`?
2) Si se usa el operador || y la primer expresión es verdadera, ¿cómo es toda la expresión? ¿Se evalúa la expresión completa?
3) Si `x` es cero, la condición:

```
if ((x != 0.0) && (y/x >7.5))
```

es falsa, ya que `(x != 0.0)` es falsa. Por consiguiente, no hay necesidad de evaluar la condición `(y/x >7.5)` cuando x sea cero. ¿Qué pasaría si se utilizaría evaluación completa?
4) Dada la situación de 3) ¿Qué sucede si se altera el orden de las condiciones y se escribe:

```
if ((y/x >7.5) && (x != 0.0))
```

¿Es crítico el orden de las expresiones con operadores && y ||? Justifique.

3. ESTRUCTURAS (REGISTROS)

1) Escribir y ejecutar un programa que maneje números complejos (parte real e imaginaria asumirla como enteros). Implemente funciones para: leer un número complejo, mostrar por pantalla un número complejo, sumar 2 complejos, restar 2 números complejos y multiplicar dos números complejos.

4. RECURSIÓN

1) Determine si el siguiente código resuelve el cálculo del factorial:

Algorithm 3: Factorial.

```
1  #include <stdio.h>
2  int factorial(int num)
3  {
4      if (num == 1)
5          return 1;
6      else
7          return (num * factorial(num-1));
8  }
9  int main()
10 {
11     int n=4, resultado;
12     resultado = factorial(n) ;
13     printf("%d ", resultado);
14     return 0;
15 }
```

Responda las siguientes preguntas:
a. El código es correcto?
b. Es correcto que una función se llame a si misma?
c. Cuántas veces es llamada la función factorial?
d. Se puede reemplazar las lineas 12 y 13 por

```
printf("%d", factorial(n));
```

e. Qué pasa si se inicializa `n = 0`?

2) Basandose en el código anterior, escriba un programa recursivo que calcule los números de fibonacci. Para un valor n, calcular $fibonacci_n$, siendo $f_n = f_{n-1} + f_{n-2}$, con $f_0 = 0$ y $f_1 = 1$.

Nota: recuerde que para compilar un programa utilizamos:

```
gcc <nombreArchivo.c> -o <nombreEjecutable>
```