PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA.
FACULTAD DE CIENCIAS DE LAS INGENIERÍAS
LABORATORIO MICROPROCESADORES I.

NOMBRE: **SAMUEL PEÑA MORNOTA**

MATRICULA/ID: 20170570/10131492 **TAREA: 1**

A) ADECLARACIÓN DE VARIABLES EN C TIPOS : CHAR, INT, FLOAT.

```c
#include <stdio.h>
#include <stdlib.h>
/*Uso basico de tipos de datos simples en lenguaje C, en el presente codigo,
  se presentaran como se declaran dichos datos
  (tipo char, int, float).---> Son palabras reservadas del lenguajes C
*/
int main()
{

  /*
    Una variable no es mas que objetos que pueden cambiar
    su valor durante la ejecucion de un programa
    En una declaracion tipo char, se pueden almacenar letras, números,
    simbolos especiales, que van entre comillas
  */
  char name[50];          /*50 caracteres en esa variable*/
  char person1, person2; /*Declaracion*/

  /*
    Declarando variables de tipo int*
    (Este tipo de almacena valores enteros, 10, 20.../
    De igual forma tambien podemos inicializar
    esas variables en la misma declaracion
   */
  int num1, num2, num3;

  /*
    Declarando variables de tipo float*
    (Este tipo de almacena valores flotantes, 1.2 2.0.../
    Es decir, valores que contengan puntos decimales
    De igual forma tambien podemos inicializar esas variables en la misma declaracion
   */
  float est1, est2, est;
  float e1 = 2.3;

}
```

PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA.
FACULTAD DE CIENCIAS DE LAS INGENIERÍAS
LABORATORIO MICROPROCESADORES I.

B) CREACIÓN DE FUNCIONES, CON ENTREGA Y ESPERA DE PARÁMETROS.

```c
#include <stdio.h>
#include <stdlib.h>
/*Uso de funciones en C, con paso de parámetros Para declarar una función con parámetros,
  Primero Necesitamos saber el prototipo de la función y que Tipo de datos tiene,
  si es Void(No retorna), o si   Es int, float, etc
*/

int suma(int, int); /*Prototipo de la funcion*/
int main()
{
  int num1,num2;      /*Declaracion de num1 y num2 de tipo int (Entero)*/
  num1 = 3;           /*La variable num1, tendrá 3 como inicialización*/
  num2 = 6;           /*La variable num2, tendrá 6 como inicialización*/

    /*Llamado de la funcion suma recibe como parametros
    (num1 y num2) declaradas anteriormente
    y lo que contenga, se lo asigno a una variable auxiliar
  */
  int aux = suma(num1,num2);

  printf("Suma: [%i]",aux);
  /*Imprimo la funcion auxiliar (aux) */

  /* Input: (num1, num2)
     num1 = 3
     num6 = 6
     Outut:
     Al pasar como parametros num1 y num2
     suma(3,6);
     la funcion me retorna (3+6) = 9
  */

}
/*Creacion de funciones con entrega y espera de parametros*/
/*
    Funcion  :  Suma
    Objetivo :  Dado dos valores enteros pasados como
                Parametros, sumarlos
    Argumento:  De tipo entero num1,num2
    Retorno  :  (int) suma de num1, y  num2
  */
int suma(int num1, int num2)
{
  return num1 + num2;
}
```

PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA.
FACULTAD DE CIENCIAS DE LAS INGENIERÍAS
LABORATORIO MICROPROCESADORES I.

C) 1.1 FOR LOOP

```c
#include <stdlib.h>
#include <stdio.h>
void lectura(int[], int);   /*Prototipo de la funcion*/
float suma(float[], float); /*Prototipo de la funcion*/
float prom(float[], float); /*Prototipo de la funcion*/
int main()
{
  int cant; /*Declaracion de int, almacena la cantidad de   valores a promediar*/
  printf("Digite la cantidad de valores a promediar: ");
  scanf("%i", &cant); //Escaneo el valor int

  int num[cant]; //Cantidad lo almaceno en un arreglo de numeros (Num)
  printf("Indique [%i] numeros ", cant);
  lectura(num, cant); /*Llamada de la funcion lectura, paso parametros
                        los numeros ingresados, y la acantidad de números */
  printf("Promedio de [%i] ingresados: [%i]", cant, prom(num, cant));
}
/*  Funcion  :  lectura
    Objetivo :  leer un arreglo de E elementos de tipo entero.
    Argumento:  Arreglo de tipo enter y una cantidad de elementos E.
    Retorno  :  No retorna valor (vlid)
*/
void lectura(int A[], int E)
{
  int i;
  for (i = 0; i < E; i++)
  {
    printf("\nValor [%i]: ", i + 1);
    scanf("%i", &A[i]);
  }
}
/*  Funcion  :  suma
    Objetivo :  Sumar la cantidad de elemntos (E) dado un arreglo.
    Argumento:  Arreglo de tipo flontante y (E) elementos
    Retorno  :  suma de los elementos del arreglo
*/
float suma(float A[], float E)
{
  /*  Declaracion de suma inicializada en (0)
      y un indice, que es el que rrecorrera
      cada uno de los valores dede cero
      hasta T elementos, en el For loop, podemos
```

```
      saber el numero de iteraciones que tendra.
  */
  int suma = 0, i;
  for (i = 0; i < E; i++)
    suma += A[i];
  return suma; //Retorno la suma
}
/*
    Funcion  :  promedio
    Objetivo :  Promediar A (arreglo), dado una cantidad
                de elementos E
    Argumento:  Arreglo de tipo flontante Y (E) elementos
    Retorno  :  Al llamar la fuuncion suma, este retorna el
                primedio de (E)
*/
float prom(float A[], float E)
{
  return suma(A, E) / E; //Retorno el promedio
}
```

1.2) DO-WHILE LOOP

```
#include <stdlib.h>
#include <stdio.h>
/*
   Do-while loop Las instrucciones se evaluan al menos una vez, por lo que la condicion
   se evalua al final
*/
int main()
{
  int num1; //Declaracion de num1;
  do
  {
    printf("\nDigite un numero: ");
    scanf("%i", &num1); //Lo ingresado lo almacno en num1
    if (num1 <= 0)
    {
      /*Si el numero es menor o igual que o Se desplegara este
                  print hasta que se cumpla la condicion
                */
      printf("\nIngrese un numero mayor ");
    }
    /*
```

```
      Las instrucciones se evaluan almenos una vez
      hasta que se cumpla esta condicion,
    */
  } while (num1 <= 0);
}
```

1.3) IF

```
#include <stdlib.h>
#include <stdio.h>
void probar(int, int); /*Prototipo de la funcion*/
int main()
{
  printf(probar(7, 6));
}
/*
   Funcion  :  Probar
   Objetivo :  Verificar si el primer numero es mayor que el segungo
   Argumento:  2 numeros de tipo entero
   retorno  : No retorna valor (void)
*/
void probar(int num1, int num2)
{
  /* (if) evalua una codicion dada, si se cumple la condicion
        se ejecuta el bloque de codigo que lo contiene, si no
        se cumple la condicion evalua otras sentencias
    */
  if (num1 > num2)
  {
    printf("EL pimer numero es mayor que el segundo");
  }
}
```

1.4) WHILE LOOP

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
/*
   While se utiliza cuando no sabemos el numero de veces que se ejecuta un ciclo
*/
void eleva(int);
int main()
{
  int auxNum1;
  printf("Ingrese un numero entero positivo: ");
  scanf("%i", &auxNum1);
  if (auxNum1 < 0)
  {
    printf("El numero ingresado debe ser mayor que cero: ");
  }
  else
  {
    eleva(auxNum1);
  }
}
/*  Funcion  :  elevar
    Objetivo :  Elevar al cubo un numero ingresado y sumarlos al salir del Loop (cuando no se
                cumpla la condicion de que auxNum1 sea que mayor que 0
    Argumento: (num1) de tipo entero
    retorno:   (void) no retorna valor
*/
void eleva(int num1)
{
  int suma = 0, cua = 0; //Variable de tipo entero, (suma) y cuadraddo
  while (num1)
  {
    /*Pow, de la libreria math.h se le da el numero
          Y el numero que se va a elevar
        */
    cua = pow(num1, 3);
    printf("[%i] elvado cubo es [%i]", num1, cua);
    suma += cua;
    printf("\nIngrese un numero: ");
    scanf("%i", &num1);
  }
  /*  Si el numero ingresado es 0, con lo que se evalua
      en la condicion del Main (auxNum1), el while no va a entrar
      y se imprimira este print
```

```
  */
  printf("La suma de los cuadrados es [%i]", suma);
}
```

1.5) SWITCH

```
#include <stdio.h>
#include <stdlib.h>
/* Estructura selectiva Switch, se utliza mayormente
   para agilizar una serie de decisiones con con un mismo
   selector, esto por ejemplo ahorra un una gran cantidad de
   if con un unico selector donde se cumplan diferentes
   condiciones
*/
int main(){
    char calif;
    printf("Ingrese calificacion del alumno: ");
    scanf("%s",&calif);

        switch(calif){ //Selector calificacion
        //Cuando se combrueba el primer caso, rompe y evalua segundo de form sucesiva
        case 'A':  printf("Calificacion Excelente"); break;
        case 'B':  printf("Calificacion buena");     break;
        case 'C':  printf("Aceptable");              break;
        case 'D':  printf("Promovido");              break;
        case 'F':  printf("Reprobado");              break;
        //Minusculas
        case 'a':  printf("Calificacion Excelente"); break;
        case 'b':  printf("Calificacion buena");     break;
        case 'c':  printf("Aceptable");              break;
        case 'd':  printf("Promovido");              break;
        case 'f':  printf("Reprobado");              break;
        default: printf("Error al ingresar calificacion: ");break;
        /* Si no se cumple niguno de los casos
           por deafult ponemos poner un error al ingresar la calificacion
        */

    }

}
```

PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA.
FACULTAD DE CIENCIAS DE LAS INGENIERÍAS
LABORATORIO MICROPROCESADORES I.

C) OPERADORES LÓGICOS AND Y OR

```c
#include <stdio.h>
#include <stdlib.h>
/*Los operadores logicos And (&&) y Or (||)*/
int main()
{
  int num1, num2;
  printf("Ingresa numero 1: ");
  scanf("%i", &num1);
  printf("Ingresa numero 1: ");
  scanf("%i", &num2);
  /*Una operacion con  And (&&) resulta verdadera
      si y solo si se cumplen las codiciones
      Es decir, si al usar && tenemos dos valores verdaderos (1 y 1)
      La operacion se va a ejecutar, de lo contrario si tenemos
      un valor verdadero y otro falso (1 y 0) lo que se obtendra sera
      lo falso en la operacion, si tenemos dos valores falso su resultado
      sera un valor falso, en resumidas cuentas solo obtendremos verdadero
      cuando sus dos operandos sean verdaderos
    */
  if (num1 == num2 && num2 == num1)
  {
    printf("El [%i] y [%i]: son iguales", num1, num2);
  }
  else
  {
    printf("El [%i] y [%i]: no son iguales", num1, num2);
  }
  /*En el Or(||) obtendremos un valor verdadero si sus dos operandos son
      derdaderos, cualquiera valor que tome un valor verdadero en sus operandos
      obtendremos un valor verdadero (1 || 1) == Verdader
      (1 || 0) == verdadero, (0 || 1) == verdader0, (0 || 0) === falso
    */
  if (num1 == num2 || num2 == num1)
  {
    printf("\nUno de los numeros son iguales");
  }
}
```

PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA.
FACULTAD DE CIENCIAS DE LAS INGENIERÍAS
LABORATORIO MICROPROCESADORES I.

INSTALACION Y PRINT DE KEIL