Práctica 1

Tipos de datos simples y estructuras de control
Funciones – Identificadores

Importante: Para el desarrollo de ésta práctica y las siguientes, compile utilizando el *flag* `-Wall`. En el tutorial de Code::Blocks se explica cómo activar éste *flag* para ese entorno.

1. Al compilar un programa en C, el compilador puede notificar errores y avisos (*warnings*). ¿Qué diferencia existe entre ellos? ¿puede un programa compilar con errores? ¿y con *warnings*? Compile los siguientes programas e identifique errores y *warnings*. Luego, corríjalos para que el código compile correctamente.

a.
```
#include <stdio.h>
int main(){
    double pi= 3.14;
    int y= 5
    printf("pi= %d\n", pi);
}
```

b.
```
#include <stdio.h>
int main(){
    int x=10;
    if (x=5)
        printf("x = 5\n");
    return 0;
}
```

2. Investigue los diferentes tipos de datos del lenguaje C y luego responda:
   a. ¿Qué ocurre cuando asignamos un `char` a un `int`? ¿y al revés? Ejecute el siguiente código y analice lo que imprime. ¿Se presenta algún *warning* al compilar?
   ```
   #include <stdio.h>
   int main(){
      char c= 'a';
      int x= 64;
      printf("char c= %c\n", x);
      printf("int x= %d\n", c);
      return 0;
   }
   ```
   b. ¿Qué diferencia existe entre los tipos `float` y `double`? ¿se puede asignar sin inconvenientes?
   c. ¿Qué sucede cuando en una operación intervienen operadores diferentes?
   Por ejemplo: `x = 2*32+10%2-1;`
   d. Investigue el operador `sizeof`. ¿Para qué sirve? Luego aplíquelo y verifique el tamaño de los distintos tipos de datos en su máquina.

3. Investigue los operadores del lenguaje C. Enumérelos y describalos según las siguientes categorías: aritméticos, relacionales, lógicos, de bits, asignación y condicional.
4. Escriba un programa que imprima en pantalla todos los números enteros múltiplos de 5 comprendidos entre 1 y 50. Realice dos soluciones con las siguientes características:
   a. Utilizando el operador % para determinar cuándo un número es múltiplo de 5.
   b. Utilizando un *for* donde la variable índice se incremente en 5 unidades por cada iteración.

5. Escriba un programa que a partir de un número flotante leído desde teclado, el cual representa una distancia en kilómetros, imprima su equivalente en millas.

   Nota: 1,61 Km ≈ 1 milla.

6. El objetivo del siguiente código es leer dos caracteres de teclado e imprimirlos en pantalla. ¿El código cumple con su cometido? En caso negativo, corríjalo para que lo haga.

```
#include <stdio.h>
int main(){
    char a, b;
    printf("Ingrese el primer caracter:\n");
    scanf("%c", &a);
    printf("Se leyó el caracter: %c\n", a);
    printf("Ingrese el segundo caracter:\n");
    scanf("%c", &b);
    printf("Se leyó el caracter: %c\n", b);
    return 0;
}
```

7. Identifique y corrija los errores en cada uno de los siguientes códigos (Nota: puede haber más de uno):

a.
```
int edad;
if (edad >= 40);
    printf("Tiene %d años o más",&40);
else
   printf("Tiene menos de %d años",&40);
```

b.
```
int total, x = 1;
while (x <= 10) {
    total += x;
    ++x;
}
printf("Total = %f",total);
```

c.
```
int valor;
scanf("%d",&valor);
switch (valor % 2) {
    case 0: printf("El valor es par");
    case 1: printf("El valor es impar");
}
```

d.
```
int x,  y;
scanf("%d",x);
scanf("%d",y);
printf("x + y = %d ",x+y);
```

8. Escriba un programa que lea 2 números enteros y un carácter, el cual debe corresponderse con un operador matemático (+, -, /, *), y a partir de ellos imprima el resultado de realizar la operación pedida en pantalla. En caso de que el carácter no represente un operador válido, imprima un mensaje de error.

9. Indique qué valores de la variable de control *i* se imprimen para cada uno de las siguientes estructuras de control *for*. Ejecute cada código y analice por qué ocurre cada caso:

a.
```
for (i = 0; i <= 11; i++)
        printf("%d\n",i);
```

b.
```
for (i = 0; i <= 11; ++i)
        printf("%d\n",i);
```

c.
```
for (i = 2; i <= 11; i+=2)
        printf("%d\n",i);
```

d.
```
for (i = 1; i <= 11; i*=2)
        printf("%d\n",i);
```

e.
```
for (i = 11; i >= 1; i-=2)
        printf("%d\n",i);
```

f.
```
for (i = 11; i > 1; i=+1)
        printf("%d\n",i);
```

10. Escriba un programa que imprima la raíz cuadrada, el cuadrado y el cubo de los números enteros comprendidos en el rango 1..10.

    Nota: para calcular la raíz cuadrada y la potencia de un número investigue las funciones `sqrt()` y `pow()`, ambas de la librería `<math.h>`.

11. Para todo número natural *n*, se llama *factorial de n* al producto de todos los naturales desde 1 hasta *n*:

$$n! = 1 \times 2 \times 3 \times 4 \times ... \times (n-1) \times n$$

    Escriba una función que reciba como parámetro un número entero y retorne como resultado el número factorial del mismo.

    a. Resolviéndolo de forma iterativa.
    b. Resolviéndolo de forma recursiva.

12. Un número primo es un número natural mayor que 1, que tiene dos divisores distintos únicamente: él mismo y el 1.

    a. Escriba una función que reciba como parámetro un número entero *n* y determine si el mismo es primo o no.
    b. Escriba un programa que lea números enteros desde teclado hasta que la cantidad de números primos leídos sea 5.

13. Escriba una función que reciba un número entero y retorne el número resultante de invertir la posición de todos sus dígitos. Por ejemplo, si se pasa como argumento el número 1234 la función deberá retorna el número 4321.

14. Analice y ejecute el siguiente código:

```
void swap(int a, int b);

int main()
{
    int x, y;
    x = 10;
    y = 20;
    printf("x=%d\ty=%d\n",x,y);
    swap(x, y);
    printf("x=%d\ty=%d\n",x,y);
    return 0;
}

void swap(int a, int b)
{
    int tmp;

    tmp = a; // guarda el valor de a
    a = b;   // almacena b en a
    b = tmp; // almacena a en b
}
```

    Se supone que la función `swap` intercambia el valor de sus dos parámetros enteros, ¿funciona correctamente? Si no es así, modifique la función `swap` para que lo haga.

15. Escriba una función que reciba como argumento un número entero `n` y que dentro de ella lea `n` números flotantes desde teclado. La función debe retornar el mínimo y el máximo número leído.

16. Escriba un programa que imprima 15 números generados pseudo-aleatoriamente. ¿La secuencia generada es siempre la misma? ¿Por qué? ¿Qué se puede hacer para generar secuencias pseudo-aleatorias diferentes?

    Nota: para generar números en forma pseudo-aleatoria investigue las funciones `rand()` y `srand()`, las cuales se encuentran definidas en la biblioteca `<stdlib.h>`.

17. Una escuela primaria desea evaluar si sus alumnos son capaces de resolver operaciones matemáticas de suma de números enteros. Para ello se pide realizar un programa que muestre al alumno 4 operaciones de suma y le pida que ingrese sus correspondientes resultados. Los operandos de las operaciones a resolver deben ser generados aleatoriamente y deben estar en el rango de 0 a 100. Cada operación respondida de forma correcta suma 1 punto. Al finalizar informar la nota obtenida por el alumno de acuerdo a la siguiente tabla:

Puntaje	4	3	2	1	0
Nota	A	B	C	D	E

    Nota: para asegurar que un número se encuentre en el rango de 0 a N, puede utilizar el operador % con (N+1) como segundo operando.

18. Desarrolle una función llamada `damePar()`, la cual debe retornar un número entero. `damePar()` devuelve 0 en el primer llamado, 2 en el segundo, 4 en el tercero, y así sucesivamente. Luego, escriba un programa que lea un número entero positivo `n` desde teclado y utilizando `damePar()` imprima los primeros `n` números pares. No utilice variables globales.