**Enunciado:**

Resuelva los siguientes ejercicios sobre funciones, cadenas de caracteres y flujos en C++. Utilice el estándar C++14 en la solución de sus problemas. No olvide compilar con los *flags* apropiados para detectar *warnings* y errores.

### Basics

1. Escriba un programa que usa `while` para la sumar los números enteros de 1 a 100.
2. Escriba un programa que imprime la suma de una secuencia con un número arbitrario de enteros que es leída usando `cin`.
3. Escriba un programa para determinar si un entero es par o impar.
4. Explique la diferencia entre el incremento prefijo (`++x`) y postfijo (`x++`).

### Functions

5. Escriba una función que determine si una letra es consonante o no.
6. Escriba una función que reciba dos meses del año (definidos como una enumeración) y retorne $-1$ si el primero es menor que el segundo, 1 si el segundo es menor que el primero, y 0 si ninguna de las anteriores opciones se satisface.
7. Escriba una función que reciba dos números enteros $n$ y $k$ (con $k < n$) y calcule, dado un conjunto de $n$ elementos, el número de conjuntos diferentes de $k$ elementos que pueden obtenerse. Este se conoce como el número de combinaciones y se calcula como
$$C(n,k) = \frac{n!}{k!(n-k)!}.$$
8. [*Binary to decimal.*] Escriba un método `int bin2dec(string s)` que convierta un número binario, en forma de `string` pedida al usuario, a su equivalente decimal y lo retorne como un número entero. La salida de su programa debe verse de la siguiente manera

```
Enter a binary string: 1011
Binary "1011" to decimal is: 11
```

El programa debe detectar si el `string` de entrada es un binario válido. Es decir, la salida del programa debe verse así, si el usuario ingresa un binario inválido,

```
Enter a binary string: 1234
ERROR: invalid binary string "1234"
```

En la implementación de su programa la función `pow(x, y)` de la librería `cmath`, que calcula $x^y$, puede serle útil.

### String

9. Para el siguiente ejercicio consulte la referencia de C++. Implemente una librería que tenga la siguiente funcionalidad:

   *a*) Una función que recibe un `string` y retorne cuántas vocales contiene.

   *b*) Una función que recibe un `string` y retorne cuántos dígitos contiene.

   *c*) Una función que recibe un `string` y retorna una copia de la misma con todas las letras en minúsculas.

   *d*) Una función que recibe un `string` y la modifica para dejar todas las letras en minúsculas.

   *e*) Una función que recibe un `string` y la modifica eliminando todos los espacios en blanco.

   *f*) Una función que recibe un `string` y retorna un acrónimo compuesto por las primeras letras de cada palabra. Por ejemplo, si el parámetro string de la función contiene la cadena "United Nations Educational, Scientific and Cultural Organization," el string retornado debe ser la cadena "UNESCO".

   Incluya esta librería en un programa de C++ que ilustre su uso.

### Streams

Un `stream` o flujo es una manera serial (un elemento tras otro) de acceder a un medio de almacenamiento. Dado que es serial, no se puede tener acceso aleatorio (*random access*) para leer o escribir en un `stream`. Adicionalmente, debido a que tiene una capacidad limitada, el contenido completo del `stream` no necesariamente está cargado en memoria en su totalidad en un momento dado. Las operaciones básicas de un `stream` son:

- Inicialización: Un `stream` se inicializa con un tipo (por ejemplo `std::string` para un `stringstream` y el nombre del archivo para un `fstream`). También se inicializa con un valor para el *modo* de acceso (por ejemplo `ios::in` para lectura, `ios::out` para escritura, etc.).
- Acceso a un punto en el `stream` mediante apuntadores `get` y `put`. Los métodos `seekg()` y `seekp()` arrastran los apuntadores `get` y `put` (respectivamente) a la posición especificada, por ejemplo al inicio (pasando el argumento `ios::beg`), al final (`ios::end`).
- Una vez localizado el apuntador, las operaciones de I/O se realizan mediante los operadores de inserción `operator<<` (*input*) y extracción `operator>>` (*output*)

Ejemplo de escritura en un archivo

```
1 #include <iostream>
2 #include <fstream>
```

```cpp
using namespace std;

int main() {
    ofstream ofs("a.txt",ios::app);
    if (ofs.good()) {
        ofs << "Hello a.txt, I'm appending this on you.\n
            ";
        for (int i = 0; i < 5; i++)
            ofs << "hello\n";
    }
    ofs.close();
    return 0;
}
```

Ejemplo de lectura de un archivo palabra por palabra y línea por línea

```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    // Word by word
    ifstream ifs("a.txt");
    string line = "";
    if (ifs.good()){
        while (!ifs.eof()){
            ifs >> line;
            cout << line << endl;
        }
    }
    ifs.close();

    // Line by line
    ifs.open("a.txt");
    if (ifs.good()){
        while (!ifs.eof()){
            getline(ifs, line);
            cout << line << endl;
        }
    }
    ifs.close();
    return 0;
}
```

10. [*Calculating* $\pi$.] Escriba un programa que calcule una aproximación al número irracional $\pi$, usando la expansión

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right).$$

El valor de $\pi$ se puede calcular como `acos(-1.)`, donde esta función esta definida en `<cmath>`. Calcule la diferencia entre su aproximación para $\pi$, para un número de términos dado en la suma, y el valor generado usando `acos(-1.)`. Genere una tabla que contenga diferentes aproximaciones de $\pi$, basados en el número de términos mantenidos en la expansión, y su respectiva diferencia o error. Es decir, cree una tabla de tres columnas que contengan el número de términos en la primera columna, el valor de la suma de arriba en la segunda columna y la diferencia con `acos(-1.)` en la tercera. Los términos que van en la primera columna deben ser de la forma $10x$, donde $x = 1, 2, 3, \ldots, 9$. Escriba esta tabla a un archivo `"compute_pi.dat"`.

11. Escriba un programa que lea un archivo donde cada línea contiene un `double` y determine el número de datos mayores a 100,0. El resultado debe mostrarse en pantalla y almacenarse en un archivo diferente.