Sección Teórica

1. Un template es una herramienta que nos permite llevar un manejo de funciones/métodos y clases, que nos permite trabajarlos como “plantillas”, es decir, cómo una resolución genérica para cierta tarea independientemente del tipo de dato que se le entregue. Esto permite, principalmente, evitar la necesidad de reutilizar código (tener que hacer sobrecargas de funciones por cada tipo de dato).

   Desde la versión 20 de C++, se puede utilizar la palabra clave “concept”, como una forma de crear constraints para los templates, de esta forma limitando qué tipos de datos serán válidos para el uso en nuestra función.

   Otra característica de templates, es la posibilidad de crear lo que se denomina “variadic template”, este tipo de template nos permite crear una plantilla que podrá utilizar una cantidad de parámetros “indefinida”.

2. La ventaja del uso de sobrecarga de operadores, es el no tener que depender de un método para operaciones básicas. El ejemplo más rápido podría ser con el manejo de una clase Vector3 custom, en lugar de tener un método “Add” que reciba como parámetro otro vector, podríamos sobrecargar el operador “+”, para que este realice la operación y devuelva un nuevo Vector3 que represente la suma entre estos. En esencia representan lo mismo, ya que ambos necesitan parámetros y devuelven un tipo de dato, pero, los operadores resultan mucho más legibles que el uso de constantes llamadas a métodos.

3. Tipos de clasificación para el manejo de archivos:

   - Por tipo de flujo: Es decir, por la manera en la que se operará el archivo.
     - Input: Flujo de entrada, supone la lectura del archivo para su uso en la ejecución del programa (Persistente >> Volátil), fallará si el archivo no se encuentra o es nulo.

       En C++, utilizamos el tipo de variable ifstream, la cual permitirá abrir archivos con este tipo de flujo.

     - Output: Al contrario que la entrada, este flujo supone la escritura, (creación, sobrescritura, adición), de un archivo, permitiendo pasar variables volátiles a un variables que perduren en la memoria ROM (Persistente << Volátil).

En C++, utilizamos el tipo de variable ifstream, la cual permitirá abrir archivos con este tipo de flujo.

- Por tipo de archivo:
  - De texto: Los archivos de texto, son aquellos que operan con caracteres, son secuenciales y contienen un dato EOF (end of file), que nos permite saber cuando se llegó al final del archivo.

    Este tipo de archivo tiene como ventaja y desventaja, el ser fácilmente legible y modificable. Permitiendo a cualquiera que logre acceder a él, modificarlo de manera sencilla.

  - Binario: Un archivo de tipo Binario, por otro lado, se maneja con bytes y se encuentra “encriptado”, si se quiere, evitando que sea fácilmente legible por alguien externo. Ya que se maneja por variables dependientes del tamaño en bytes que estas ocupan.

- Por tipo de acceso:
  - Secuencial: Comienza al principio, y secuencialmente avanza por el archivo hasta alcanzar el final de este, supone una forma de manejo lenta y consumidora de tiempo.

  - Aleatorio: Permite moverse por el archivo “saltando” y accediendo directamente a porciones, lo que supone una forma de manejo mucho más rápida.

4. Un archivo de texto, es un tipo de archivo externo a los archivos del programa, es decir, no forma parte de los archivos que se utilizan en el proceso de la compilación, en el que se pueden almacenar y luego leer cadenas de caracteres.

   Se diferencia de los archivos binarios en la forma en la que manejan la información, ya que el archivo binario se maneja con asignaciones de memoria, mientras que el archivo de texto se maneja con un flujo que recibe cadenas de texto, a su vez. Un archivo de texto, a diferencia de uno binario, puede ser fácilmente manipulado mediante un editor de texto. Ya que todo su contenido es visible.