**Entregable 02:**

* ***Función genérica de Listado*** <T>
* Tiene un vector de tipo de dato <T> (T \* v = new T[n])
* Agregamos al vector tipo de datos int

**Entregable 03**

Seguimos usando la ***Función genérica de Listado*** y aplicamos las funciones:

* borrar\_del\_final( int cuantos ) y borrar\_del\_principio( int cuantos )
* más los básicos como pop\_back(), erase(), insert(), clear()

**Entregable 04:**

Aplicamos **Herencia con clases genéricas** y **Sobrecarga de operadores**

* Creamos la clase Poste y Cliente
* Cliente:
  + **operator+, -** (más explicación de la sobrecarga de operadores).
* Poste:
  + **operator +** (dos operators +)
* Creamos los operadores float **operator+**( Poste poste ) y Poste **operator+**( float altura )

**Entregable 05**

Aplicamos **Sobrecarga de operadores en la clase genérica Listado**, (que creamos anteriormente)

* Sobrecargamos ***void operator+( int cuantasNuevasCeldas )***
* Sobrecargamos **Listado< T > operator+( const Listado< T > otro );**

**Entregable 06**

Vimos **Diversas maneras de usar la Herencia con clases genéricas** y usamos un ejemplo de Herencia con las clases Persona y Cliente

* Persona con ID genérico. Se hacen Personas con ID que sean un struct.
* Sobrecarga al ***operator++***, para aumentar la edad de Persona

**Entregable 07**

Vimos **Python, cuadernos de código y herramientas** y no aprendimos un pingo

**Entregable 08**

Acá tampoco aprendimos un pingo, más **Python.**

**Entregable 09**

Acá vimos **OpenCV** y aplicamos ponerle nuestro nombre en medio de la pantalla (usamos entornos virtuales).

**Entregable 10**

Acá vimos de nuevo la bosta del cuaderno de Google y no aprendimos un pingo, no sirve de nada esa bosta de código.

**Entregable 11**

Vimos **Miembros estáticos,** para el entregable resumimos todos los métodos estáticos de las materias pasadas, como **QString::number( int ).**

**Entregable 12**

Vimos al **Constructor copia** y al **Operador de asignación** al mismo lo aplicamos en un QVector de Ventanas:

* Ventana & operator=( const Ventana & ventana ) ; <= Operador Asignación
* Ventana( const Ventana & ventana ) ; <= Constructor Copia

**Entregable 13**

Clase LineaDeTexto que hereda de QLineEdit:

* Hacer línea = l1 + l2 (que línea tenga el texto que tiene l1 + l2)
* Usamos:
  + Cosntructor copia
  + Operator=
  + Operator +

**Entregable 14**

Hicimos la clase **MiVector** que hereda de **QVector< T >**  e hicimos 3 objetos MiVector de Tipo **Persona, Poste, Cliente** les hicimos unos Appends y a los elementos los ordenamos con el método bubuja. Sobrecargando el operator< en las clases mencionadas. Y mostramos el resultado en un **QTextEdit.**

**Entregable 15**

Utilizamos el mismo código del Entregable 13 y le agregamos la opción para que **aparezca las sugerencias** a medida que se escribe en el LineEdit a partir de un QStringList.

**Entregable 16**

Hacer un Login donde los LineEdit estén promovidos a la clase creada en el entregable 15.

Las sugerencias se pueden hacer mediante **Google** o mediante una lista hardcodeada.

**Entregable 17**

Primer QMainWindow donde hicimos el SublimeText con los iconos

**Entregable 18**

Clases Singletons con clase Manager que controle las clases Login, Formulario y Ventana.

**Entregable 19:**

* Grupo 1: Crear objetos comparables: Persona (sin herencia), Login (hereda de QWidget), Imagen (hereda de QImage)
* Grupo 2: AdminDB con todas las funcionalidades y singleton
* Grupo 3: Clase Archivador como singleton que almacene Logs en un archivo de texto.
* Grupo 4: LineaDeTexto listo para promiocionar y con opción de sugerencia de Google y lista fija. Manera de setear la segunda columna.
* Grupo 5: QMainWindow que tenga programado un método que reciba una enumeración en su constructor (SUBLIME, DARK, GALERIA, CLASSIC) que permita combinar (por ejemplo, DARK|SUBLIME o DARK|GALERIA)