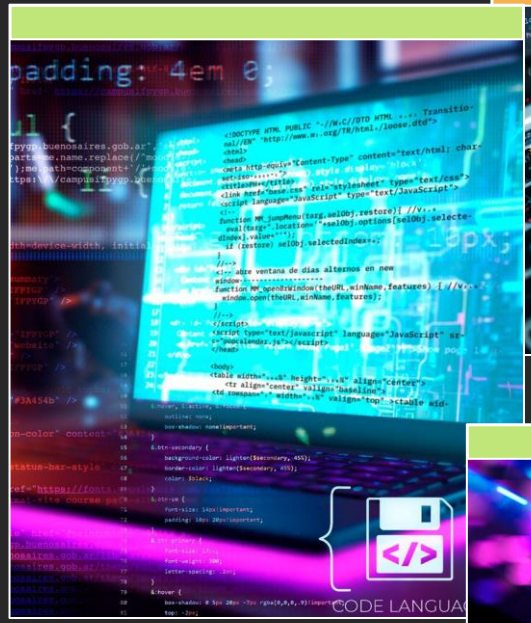
# Programacion II [ PRACTICA ]


1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0

1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0

# INTERFAZ GRAFICA DE USUARIOS (G.U.I)

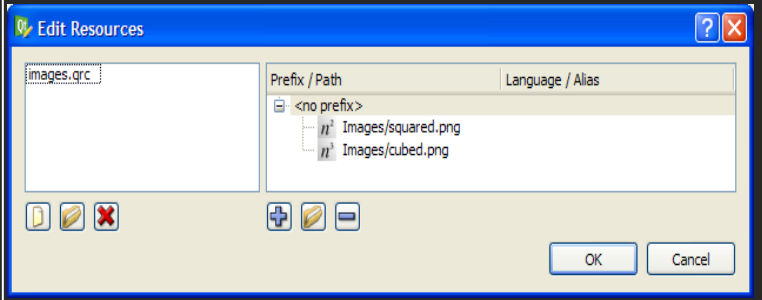




1 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0

1 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0

# Qt Resource System




QR Resource se refiere a un sistema en Qt que permite almacenar archivos dentro del binario de la aplicación. Los archivos se empaquetan en un archivo de recursos (.qrc), que luego se puede acceder mediante rutas de recursos dentro del código.

## Ventajas de usar QR Resource

1. **Portabilidad:** *Los recursos forman parte del binario, lo que significa que la aplicación no necesita buscar archivos externos durante la ejecución.*
2. **Simplicidad:** *Simplifica la distribución, ya que todos los recursos están incluidos dentro del ejecutable.*
3. **Protección:** *Dado que los archivos están incrustados en el ejecutable, pueden estar más protegidos contra modificaciones o acceso indebido.*
4. **Optimización de acceso:** *El sistema de recursos está optimizado para acceder a archivos de manera rápida y eficiente.*

# Resource Collection Files – Como Crearlos [.qrc]

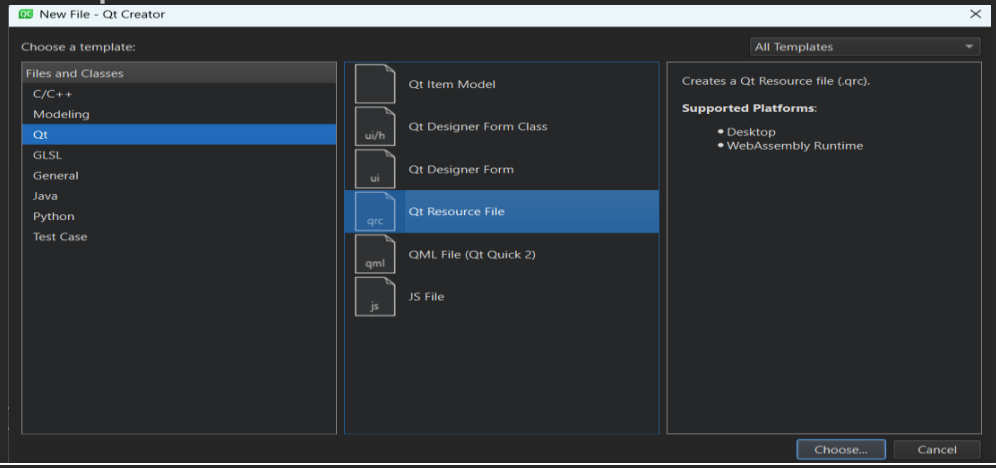
Cómo crear y usar archivos .qrc en QtCreator:

1. **Creación del archivo .qrc:**

En QtCreator, haz clic derecho en el proyecto y selecciona:
"Add New..." > "Qt" > "Qt Resource File".
Esto crea un archivo .qrc.

2. **Agregar archivos a QR Resource:**

Abre el archivo .qrc y agrega recursos arrastrando archivos (como imágenes o sonidos) directamente desde el explorador de archivos o añadiéndolos manualmente en el archivo XML.



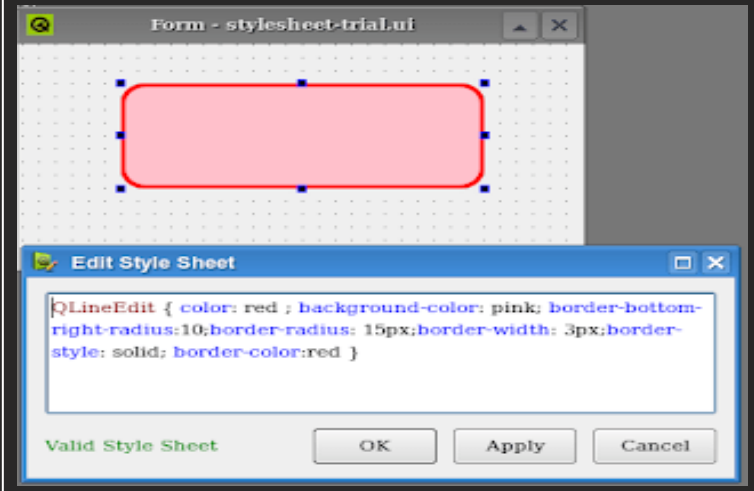
# Limitaciones del sistema QR Resource

1. **Tamaño del Ejecutable:**
Incluir muchos recursos puede aumentar significativamente el tamaño del ejecutable.

2. **No es Adecuado para Archivos Grandes:**
Recursos como videos o archivos de gran tamaño no deberían incluirse dentro del ejecutable, ya que esto podría hacer que el programa sea difícil de manejar.

3. **Complejidad en Mantenimiento:**
En proyectos grandes, gestionar los archivos dentro del .qrc puede volverse complejo si no se estructura adecuadamente.

# QStyleSheets




Los QStyleSheets en Qt son una poderosa herramienta que permite personalizar la apariencia de los widgets de manera similar a cómo se utilizan las hojas de estilo en cascada (CSS) para personalizar elementos HTML en las páginas web. Qt utiliza una sintaxis similar a CSS, lo que hace que los desarrolladores puedan cambiar rápidamente los colores, fuentes, bordes, tamaños, y otros aspectos visuales de los widgets, sin necesidad de modificar el código C++ directamente.

## ¿Qué es un QStyleSheet?

Un QStyleSheet es una cadena de texto que contiene reglas de estilo aplicables a los widgets de una aplicación Qt. Estas reglas permiten cambiar propiedades como el color de fondo, color del texto, bordes, márgenes, relleno y más. Se aplican de manera jerárquica, es decir, puedes definir estilos para toda la aplicación o de manera específica para un widget en particular.

# QStyleSheets - Propiedades comunes en QStyleSheets

- **background-color:** Cambia el color de fondo del widget.
- **color:** Cambia el color del texto del widget.
- **font-family:** Cambia la familia de fuentes (e.g., Arial, Verdana).
- **font-size:** Establece el tamaño de la fuente.
- **border:** Define el tipo, color y grosor del borde.
- **border-radius:** 10px; redondea las esquinas del borde.
- **padding:** Establece el espacio interno entre el contenido del widget y su borde.
- **margin:** Define el espacio exterior entre el widget y otros elementos adyacentes.
- **min-width, min-height:** Establece el tamaño mínimo de un widget.
- **max-width, max-height:** Establece el tamaño máximo de un widget.

# QStyleSheets - Estructura

La estructura de QStyleSheets en Qt sigue un formato similar al de las hojas de estilo en cascada (CSS) que se utilizan en la web, con selectores que permiten aplicar estilos a widgets específicos, propiedades que definen la apariencia de estos widgets, y valores que determinan el comportamiento visual de cada propiedad.

```
Selector {
    propiedad1: valor1;
    propiedad2: valor2;
    ...
}
```

**Componentes Principales:**

- **Selector:** Define a qué widget o conjunto de widgets se aplica el estilo.
- **Propiedad:** Especifica qué aspecto del widget se va a modificar, como el color, el borde, el tamaño de la fuente, etc.
- **Valor:** Es el valor que tomará la propiedad (por ejemplo, un color específico, un tamaño de borde, etc.).

MAS INFORMACION AL RESPECTO ESCANEAR EL CODIGO QR

# QStyleSheets – Ventajas & Limitaciones

## Ventajas

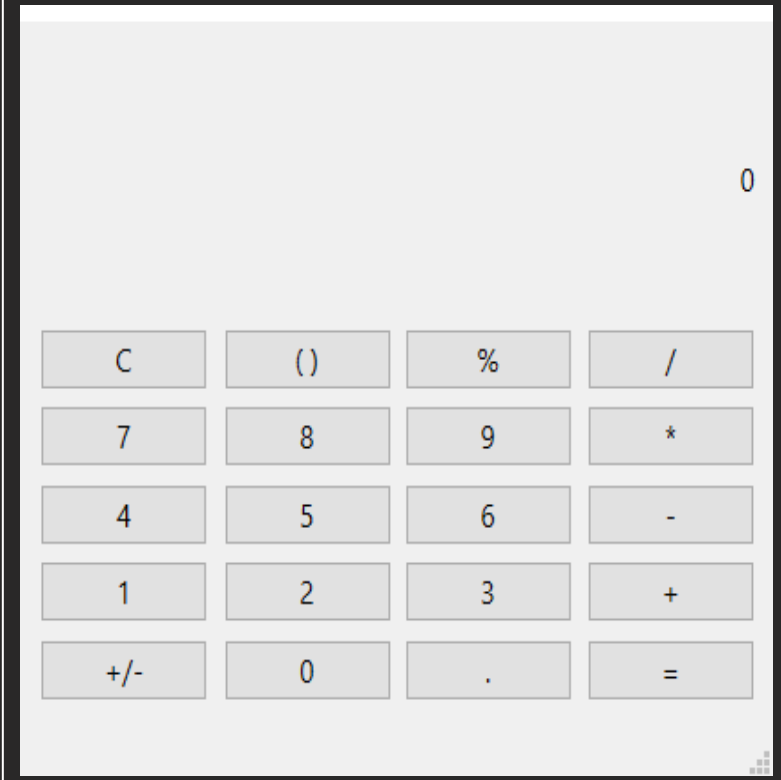
1. **Fácil de usar:** No necesitas modificar el código fuente C++, lo que hace que las personalizaciones sean rápidas y simples.
2. **Separación de la lógica y la apariencia:** Mantiene la personalización visual separada de la lógica del programa.
3. **Flexible:** Permite cambios rápidos sin recompilar la aplicación.
4. **Reutilizable:** Puedes copiar y reutilizar estilos entre diferentes proyectos de Qt.

## Limitaciones

1. **Rendimiento:** Si se utilizan QStyleSheets complejos en widgets muy dinámicos, pueden impactar en el rendimiento de la aplicación.
2. **Menos control que QStyles:** Aunque son fáciles de usar, los QStyleSheets no permiten la misma personalización detallada a nivel de control y comportamiento de los widgets que ofrece QStyles.

# Retos de Programacion

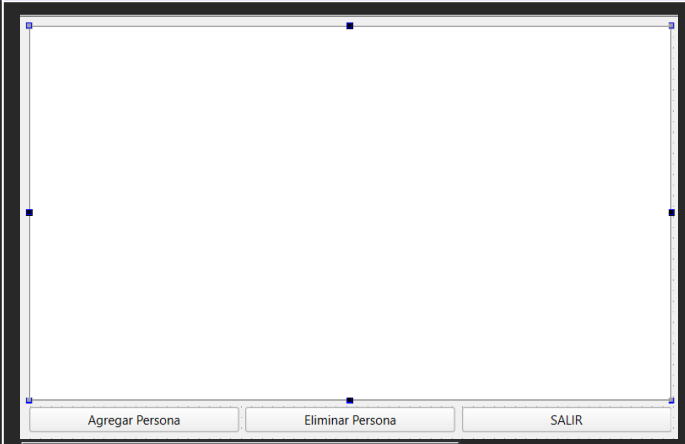
# 01 Calculadora (Front)

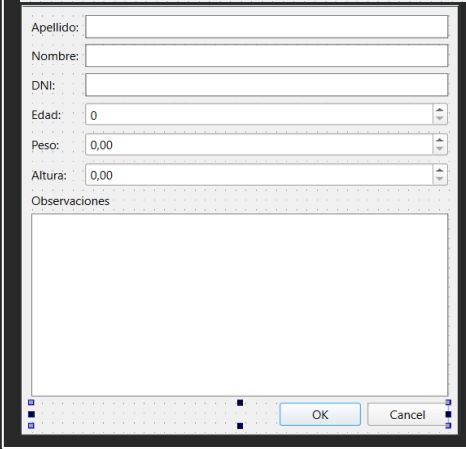



Al ejercicio realizado de la calculadora cargar un stylo ya creado.

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

# 02 Formularios




Al siguiente programa. Generar un Stylo propio repasando lo visto.
El programa debera tener:

- QMainWindow
  - 01 Qlabel
  - 01 QLCDNumber
- Qwiget:
  - QTable
  - Botón Agregar
  - Botón Eliminar
  - Botón SALIR
- Qdialog:
  - 07 QLabel
  - 03 QLineEdit
  - 01 QSpinBox
  - 02 QDouble SpinBox
  - 01 QPlainTextEdit
  - 02 botones (OK/CANCEL)

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

# 03 VARIOS

1. Crea una aplicación en QMainWindow que permita a los usuarios cambiar el tema de la aplicación usando QStyleSheets. Debe incluir:

- Un botón para aplicar un estilo oscuro y otro para aplicar un estilo claro.
- Utiliza QStyleSheets para cambiar los colores de fondo, texto, y los botones.
- Crea un archivo .css para cada tema (oscuro y claro), y permite cargarlos dinámicamente.
- Usa componentes como QPushButton, QLabel, y QLineEdit para observar los cambios de estilo.

2. Desarrolla una aplicación que use el Qt Resource System para gestionar archivos de imágenes e iconos. La aplicación debe:

- Incluir un QPushButton con un icono de imagen cargado desde el archivo .qrc (Resource System).
- Mostrar una imagen en un QLabel que también provenga del Resource System.
- Crear un menú en la barra de herramientas con opciones que usen iconos, cargados desde recursos.
- El archivo .qrc debe incluir las imágenes y los iconos.