SIMULACRO EXAMEN FINAL

# ENUNCIADO

Desarrollar una aplicación que permita convertir una temperatura dada en cierta unidad de medida a otras unidades de medida.

La aplicación debe desarrollar una clase donde implemente toda la lógica de la solución y una ventana para la interface de usuario.

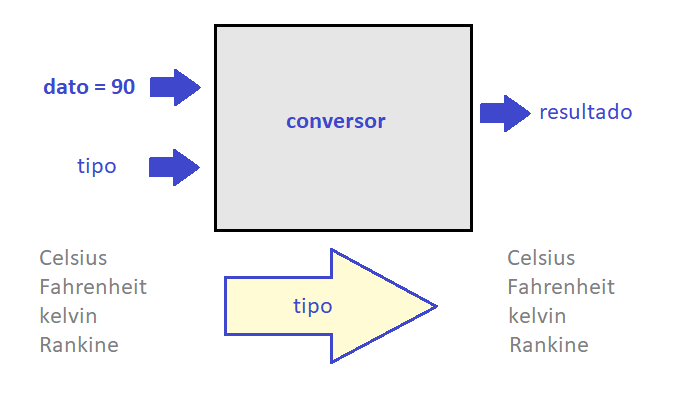
Para seleccionar la unidad origen se debe utilizar radio buttons o combo box.

Se solicita:

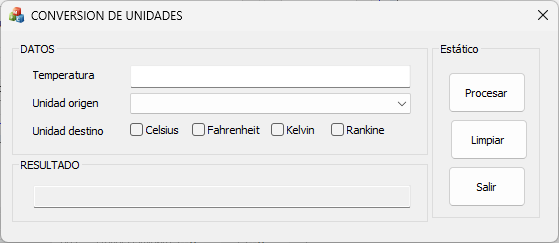
1. Diseño de la interfaz de usuario.
2. El identificador de los controles y sus respectivas variables asociadas.
3. La clase de la lógica de la solución, de nombre **CLogica**. Los métodos pueden ser de instancia o de clase.
4. La programación de los botones.

# SOLUCIÓN

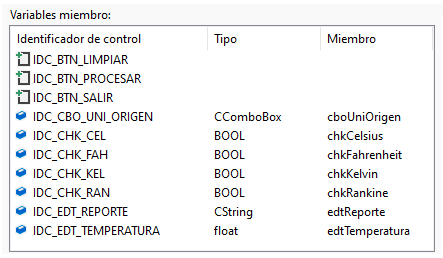
## Análisis



## Interfaz de Usuario



## Identificador de los controles



## Archivo: CLogica.h

enum TConvert

{

CTOF, CTOK, CTOR,

FTOC, FTOK, FTOR,

KTOC, KTOF, KTOR,

RTOC, RTOF, RTOK,

};

class CLogica {

private:

// Constructor privado

CLogica(){}

public:

// Método de clase

static float convert(TConvert tipo, float dato) {

float resultado = 0;

switch (tipo) {

case CTOF: resultado = 1; break;

case CTOK: resultado = 2; break;

case CTOR: resultado = 3; break;

case FTOC: resultado = 4; break;

case FTOK: resultado = 5; break;

case FTOR: resultado = 6; break;

case KTOC: resultado = 7; break;

case KTOF: resultado = 8; break;

case KTOR: resultado = 9; break;

case RTOC: resultado = 10; break;

case RTOF: resultado = 11; break;

case RTOK: resultado = 12; break;

}

return resultado;

}

};

## Programación de los botones

**Botón: IDC\_BTN\_PROCESAR**

UpdateData(true);

edtReporte.Empty();

CString repoParcial;

float resultado = 0;

// Punto 1: Destino es Celsius

if (chkCelsius) {

switch (cboUniOrigen.GetCurSel()) {

case 0: resultado = edtTemperatura; break;

case 1: resultado = CLogica::convert(FTOC, edtTemperatura); break;

case 2: resultado = CLogica::convert(KTOC, edtTemperatura); break;

case 3: resultado = CLogica::convert(RTOC, edtTemperatura); break;

}

repoParcial.Format(\_T(" %.2f C"), resultado);

edtReporte.Append(repoParcial);

}

// Punto 2: Destino es Fahrenheit

if (chkFahrenheit) {

switch (cboUniOrigen.GetCurSel()) {

case 0: resultado = CLogica::convert(CTOF, edtTemperatura); break;

case 1: resultado = edtTemperatura; break;

case 2: resultado = CLogica::convert(KTOF, edtTemperatura); break;

case 3: resultado = CLogica::convert(RTOF, edtTemperatura); break;

}

repoParcial.Format(\_T(" %.2f F"), resultado);

edtReporte.Append(repoParcial);

}

// Punto 3: Destino es Kelvin

if (chkKelvin) {

switch (cboUniOrigen.GetCurSel()) {

case 0: resultado = CLogica::convert(CTOK, edtTemperatura); break;

case 1: resultado = CLogica::convert(FTOK, edtTemperatura); break;

case 2: resultado = edtTemperatura; break;

case 3: resultado = CLogica::convert(RTOK, edtTemperatura); break;

}

repoParcial.Format(\_T(" %.2f K"), resultado);

edtReporte.Append(repoParcial);

}

// Punto 3: Destino es Rankine

if (chkRankine) {

switch (cboUniOrigen.GetCurSel()) {

case 0: resultado = CLogica::convert(CTOR, edtTemperatura); break;

case 1: resultado = CLogica::convert(FTOR, edtTemperatura); break;

case 2: resultado = CLogica::convert(KTOR, edtTemperatura); break;

case 3: resultado = edtTemperatura; break;

}

repoParcial.Format(\_T(" %.2f R"), resultado);

edtReporte.Append(repoParcial);

}

UpdateData(false);

**Botón: IDC\_BTN\_LIMPIAR**

UpdateData(true);

edtTemperatura = 0;

edtReporte.Empty();

cboUniOrigen.SetCurSel(-1);

chkCelsius = false;

chkFahrenheit = false;

chkKelvin = false;

chkRankine = false;

UpdateData(false);

**botón: IDC\_BTN\_SALIR**

this->OnCancel();