Servicio Geológico Colombiano

Documentación Métodos Proceso20

Sistema PROVIG – Servicio Geológico Colombiano

Manizales

2014

# Introducción

En este documento se hace la descripción del funcionamiento de los métodos de ***Proceso*** el cual es uno de los programas que forma parte del ***SISTEMA PROVIG*** y tiene como función la clasificación de las señales sísmicas volcánicas y/o tectónicas, a partir de los registros continuos de las estaciones que componen las redes de monitoreo, aceptando archivos en diversos formatos de datos sísmicos como SUDS, GCF, Y-FILE y SEISAN.

Tiene integradas las utilidades necesarias para la lectura de parámetros de onda, localización espacial, análisis espectral y generación de reportes de eventos sísmicos, entre otras funciones, que permiten caracterizar las señales sísmicas e incorporar, tanto las formas de ondas como archivos de registro derivados del procesamiento a una base organizada.

La descripción de los métodos está hecha en formato de contratos y cada uno de ellos tiene vinculados los métodos con los que tiene relación directa; además fueron agrupados en sus respectivas clases y se diferencian los que son lanzados por eventos gráficos y los que no.

# Contenido

[Introducción 2](#_Toc393967730)

[Contenido 1](#_Toc393967731)

[Form1 5](#_Toc393967732)

[Form1\_Load() 5](#_Toc393967733)

[ValNugcf() 7](#_Toc393967734)

[DimensionarPanelTarjetas() 7](#_Toc393967735)

[Revisarcarpetas() 8](#_Toc393967736)

[lecturas() 8](#_Toc393967737)

[LeeMux() 9](#_Toc393967738)

[LeeDmx() 10](#_Toc393967739)

[LeeGcf() 10](#_Toc393967740)

[LeeSeisan() 11](#_Toc393967741)

[LeeYfile() 11](#_Toc393967742)

[Facnano() 12](#_Toc393967743)

[Facmilimetro() 12](#_Toc393967744)

[SeleccionarMinuto() 13](#_Toc393967745)

[volcanTarro() 14](#_Toc393967746)

[LLenaBox() 14](#_Toc393967747)

[LeerAmplitud() 15](#_Toc393967748)

[Nombre10() 15](#_Toc393967749)

[PromediosIniciales() 15](#_Toc393967750)

[Clasificar() 16](#_Toc393967751)

[YaClasificados() 16](#_Toc393967752)

[DibujoTrazas() 17](#_Toc393967753)

[fecha() 17](#_Toc393967754)

[TrazasClas() 18](#_Toc393967755)

[TrazasClasCuentas() 18](#_Toc393967756)

[Promedio() 19](#_Toc393967757)

[Subir() 20](#_Toc393967758)

[Bajar() 20](#_Toc393967759)

[Aumentar() 20](#_Toc393967760)

[Disminuir() 21](#_Toc393967761)

[Uno() 21](#_Toc393967762)

[Parametros() 21](#_Toc393967763)

[Seguir() 21](#_Toc393967764)

[Archivo() 22](#_Toc393967765)

[Reviarch() 23](#_Toc393967766)

[Cuadro\_Tremor() 23](#_Toc393967767)

[CalcularEspectro() 24](#_Toc393967768)

[Espectro() 24](#_Toc393967769)

[GraficaEspectro() 25](#_Toc393967770)

[GraficaEspectroLog() 26](#_Toc393967771)

[Dibujocoda() 26](#_Toc393967772)

[DibujoClascoda() 27](#_Toc393967773)

[CalcularEspectroCla() 28](#_Toc393967774)

[PromedioFiltrado() está pendiente 28](#_Toc393967775)

[CalculoInterpolacion() pendiente 28](#_Toc393967776)

[Saturacion() 29](#_Toc393967777)

[Dibujo() 29](#_Toc393967778)

[DibujoCodaAmp() 31](#_Toc393967779)

[DibAmpl() 31](#_Toc393967780)

[variasamplitudes() 32](#_Toc393967781)

[GrabaBase() falta vincular 32](#_Toc393967782)

[GrabaSuds() 33](#_Toc393967783)

[VerTrazasArribos() 33](#_Toc393967784)

[VerVista() 34](#_Toc393967785)

[ZoomEspectro() 35](#_Toc393967786)

[MovimientoParticula() falta terminar 35](#_Toc393967787)

[TrazaComponenteInterp()falta terminar 35](#_Toc393967788)

[TrazaComponente() falta terminar 36](#_Toc393967789)

[MoverNEZ() falta terminar 36](#_Toc393967790)

[MoverNEZinterp() falta terminar 36](#_Toc393967791)

[RevisarEstacionNeic() 37](#_Toc393967792)

[Métodos de eventos gráficos Form1 37](#_Toc393967793)

[cajon\_CheckedChanged() 37](#_Toc393967794)

[cajon\_MouseDown() 38](#_Toc393967795)

[cajon\_MouseMove() 38](#_Toc393967796)

[boGcfTar\_MouseDown() 38](#_Toc393967797)

[bMas\_Click() 39](#_Toc393967798)

[bvol\_Click() 39](#_Toc393967799)

[bcla\_Click() 39](#_Toc393967800)

[best\_MouseDown() 40](#_Toc393967801)

[listBox1\_SelectedIndexChanged() 40](#_Toc393967802)

[listBox2\_SelectedIndexChanged() 41](#_Toc393967803)

[listBox2\_MouseDown() 42](#_Toc393967804)

[panelmarca\_MouseDown() 42](#_Toc393967805)

[boclaizq\_MouseDown() 42](#_Toc393967806)

[boclader\_MouseDown() 43](#_Toc393967807)

[bocladil\_MouseDown() 43](#_Toc393967808)

[boclaenc\_MouseDown() 43](#_Toc393967809)

[boTodas\_Click() 43](#_Toc393967810)

[boEsta\_Click() 44](#_Toc393967811)

[panelEsta\_MouseDown() 44](#_Toc393967812)

[bout\_Click() 45](#_Toc393967813)

[bosuiz\_Click() 45](#_Toc393967814)

[bozoom\_Click() 45](#_Toc393967815)

[bosude\_Click() 46](#_Toc393967816)

[boinde\_Click() 46](#_Toc393967817)

[boiniz\_Click() 46](#_Toc393967818)

[boref\_Click() 47](#_Toc393967819)

[botim\_Click() 47](#_Toc393967820)

[bopep\_MouseDown() 47](#_Toc393967821)

[boPepVol\_MouseDown() 48](#_Toc393967822)

[panelcladib\_MouseDown() 48](#_Toc393967823)

[panelcladib\_MouseUp() 48](#_Toc393967824)

[panelcoda\_MouseUp() 49](#_Toc393967825)

[panel1\_MouseUp() falta vincular 50](#_Toc393967826)

[panel1a\_MouseUp() 50](#_Toc393967827)

[boHypo71\_Click() 51](#_Toc393967828)

[panelAmp\_MouseUp() falta vincular 52](#_Toc393967829)

[boPSW\_Click() 52](#_Toc393967830)

[boScilab\_Click() 53](#_Toc393967831)

[boClaSola\_Click() 54](#_Toc393967832)

[boStartMpt\_Click() 54](#_Toc393967833)

[boNeic\_MouseDown() 54](#_Toc393967834)

[textBoxNeic\_TextChanged() 55](#_Toc393967835)

[Util 55](#_Toc393967836)

[Dos() 55](#_Toc393967837)

[Leerbase() 56](#_Toc393967838)

[Letreros() 57](#_Toc393967839)

[borra() 58](#_Toc393967840)

[VerArchi() 58](#_Toc393967841)

[EscribePanelEsta() 59](#_Toc393967842)

[VerMarca() 60](#_Toc393967843)

[PonePepas() 60](#_Toc393967844)

[Contrato Base 61](#_Toc393967845)

# Form1

## Form1\_Load()

|  |  |
| --- | --- |
| **Nombre** | Form1\_Load |
| **Descripción** | Funciones que realiza:   1. Carga en memoria las variables de todo el programa. 2. Comprueba que en la configuración regional esté seleccionada la coma como separador decimal. 3. Verifica que el Octave esté instalado. 4. Cuenta el número de tarjetas que hay por cada formato y los guarda en sus respectivas variables. 5. Verifica la existencia del archivo *inicio.txt.* 6. Verifica si los datos son multiplexados o demultiplexados en el archivo inicio. 7. Verifica la existencia de la base de datos. 8. Identifica el número de volcanes y sus clasificaciones. 9. Crea los botones de los volcanes. 10. Crea los botones de las clasificaciones. 11. Verifica la existencia del archivo *estacajon.txt.* 12. Asocia estaciones por volcán a un botón determinado mediante el archivo estacajon.txt. 13. Verifica la existencia del archivo *vigilancia.txt* y manipula su contenido. |
| **Parámetros** | Object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Precondiciones** | Deben estar almacenados en la carpeta pro los archivos:  inicio.txt, estacajon.txt, vigilancia.txt. |
| **Postcondiciones** | Carga en memoria las variables utilizadas en la clase Form1. |
| **Excepciones** | Cierra el programa al no encontrar los archivos necesarios para cargar las variables de la clase form1. |
| **Proceso** | Inicia declarando variables locales que serán necesarias y comprobando que la coma sea el separador decimal por defecto.  Verifica la existencia del archivo octave.exe, al que en caso de existir mediante el método ***Dos()*** ejecuta un comando para manejar Octave; después de verificar la existencia del archivo inicio.txt cuenta cuantas tarjetas hay por formato y guarda esos valores en las variables respectivas numux, nudmx, y configuara las variables para manejar longitud local, ruta de disparo y hace visibles los botones asociados.  Verifica la existencia en inicio de la línea que permite ver todas las trazas con la misma referencia en caso de existir parametriza los componentes gráficos y las variables necesarias para mostrar las trazas de forma analógica.  Verifica que las variables nudmx > 0 numux > 0 sigcf == true && siyfi == true siseisan == true, si ninguna de las condiciones es verdadera quiere decir que en el archivo inicio.txt no existen formatos sísmicos por lo cual muestra un mensaje informando esto y cierra la aplicación, en caso contrario revisa por separado cada una de las anteriores condiciones y configura las variables para el manejo de cada tarjeta en donde la condición se cumpla.  Busca el archivo clasificación.txt si no lo encuentra muestra un aviso informando que no existe base de datos y cierra el programa, si lo encuentra lo lee y cuenta cuantos volcanes y cuantas clasificaciones hay en el archivo, en caso de que el archivo clasificación no este creado conforme al funcionamiento del programa se muestra un mensaje informando esto.  Posteriormente crea dinámicamente los botones para los volcanes y las clasificaciones.  Verifica la existencia del archivo *estacajon.txt*  el cual se utiliza para asociar las estaciones a los volcanes si este existe asocia las estaciones a su respectivo volcán.  Si el archivo vigilancia existe lo que se hace es leerlo con el fin de modificar los parámetros del rango de fechas en el que se muestran las trazas y hace el llamado a los métodos Facnano() y Facmilimetro() leer los factores para convertir cuentas en microsegundos y milímetros respectivamente para luego hacer el llamado a LlenarBox() y mostrar la información en el intervalo de fechas antes configurado.  Por ultimo termina de configurar botones que se muestran en la interfaz principal y llama el método DimensionarPanelTarjetas() para crear los CheckBox que representan las estaciones. |
| **Métodos que llama** | * Dos() (Util). * [ValNugcf()](#_ValNugcf()) (Form1). * [Revisarcarpetas()](#_Revisarcarpetas()) (Form1). * [Facnano()](#_Facnano()) (Form1). * [Facmilimetro()](#_Facmilimetro()) (Form1). * [DimensionarPanelTarjetas()](#_DimensionarPanelTarjetas()) (Form1). * [LLenaBox()](#_LLenaBox()) (Form1). * fecha() (Form1). |

## ValNugcf()

|  |  |
| --- | --- |
| **Nombre** | ValNugcf |
| **Descripción** | Rutina que lee el número de trazas en formato GCF que hay en el archivo archigcf.txt y guarda ese número en la variable nugcf. |
| **Precondiciones** | El archivo archigcf.txt debe estar creado en la base de datos. |
| **Postcondiciones** | El valor de la variable nugcf será igual al número de trazas en formato cgf en el archivo archicgf.txt. |
| **Excepciones** | Que no se encuentre el archivo archigcf.txt. |
| **Proceso** | Primero trata de abrir el archivo archigcf.txt si este existe cuenta cuantas líneas tiene y ese número se lo asigna a la variable nugcf. |

## DimensionarPanelTarjetas()

|  |  |
| --- | --- |
| **Nombre** | DimensionarPanelTarjetas |
| **Descripción** | Aquí se crean los cajones de estaciones, para que el usuario pueda escoger las estaciones a voluntad. Sobre todo útil en casos que se tenga problemas con algún computador o tarjeta y sea mejor deseleccionarla. |
| **Precondiciones** | Que los archivos inicio.txt, archifile.txt, archigcf.txt estén creados en la base de datos. |
| **Postcondiciones** | Crea los CheckBox de las estaciones. |
| **Proceso** | Crea un ciclo para cada tarjeta (mux y dmux) en el que configura los CheckBox específicos y los agrega al ***panelTar***; para las señales en formatos ***gcf,*** y ***y-file***, crea ciclos while en los que lee el archivo respectivo de cada formato de la carpeta pro y configura los CheckBox para cada tarjeta que posteriormente añade a ***panelTar***, por último para el formato ***seisan*** añade el CheckBox que representa esta tarjeta al ***panelTar.*** |
| **Excepciones** | Dejar de cargar botones de estaciones por la falta del archivo específico de configuración que tiene sus valores. |

## Revisarcarpetas()

|  |  |
| --- | --- |
| **Nombre** | RevisarCarpetas |
| **Descripción** | Esta rutina revisa que las carpetas de la base donde se guardan los datos, existan, de tal modo que en caso contrario, el usuario pueda revisar si se han movido. |
| **Retorna** | true si están todas las carpetas, false en caso contrario. |
| **Postcondiciones** | En caso de que no se encuentre una carpeta se muestra un mensaje informando cual es la carpeta que no existe o está mal nombrada para ser reparada, retorna un false, si todas las carpetas son encontradas retorna true. |
| **Proceso** | Va revisando una por una las carpetas de configuración cla, loc, ate, lec y sud; después entra en la carpeta lec y mediante un for que va desde 0 hasta el número de clasificaciones definido revisa que existan las carpetas con los datos de cada clasificación; este mismo proceso lo hace en la carpeta sud, al final si son encontradas todas las carpetas de configuración el método retorna un true, en caso contrario retorna false. |

## lecturas()

|  |  |
| --- | --- |
| **Nombre** | lecturas |
| **Descripción** | Rutina que se encarga de leer Todas las trazas de TODAS las tarjetas. Realmente hace llamado a las rutinas respectivas, las cuales específicamente, leen el formato sísmico correspondiente |
| **Parámetros** | bool cond: variable booleana que es utilizada para validar la inicialización de variables útiles. |
| **Retorna** | Retorna 1 si el número de trazas es mayor a 0, en caso contrario retorna un 0. |
| **Postcondiciones** | Los datos de las trazas quedan cargados en memoria en las variables cu[] [] y tim[][]. Se configura que trazas deben ser dibujadas al inverso en la vertical, queda calculado el promedio por trazas. |
| **Proceso** | Inicia declarando las matrices ***tim*** y ***cu***,la lista donde muestra los minutos y clasificaciones ***list1*** y la lista donde muestra los nombres de las estaciones ***list2***. Posteriormente va llamando los métodos que leen las trazas según su formato, cuando ya tiene las trazas en memoria verifica cual es la de menor tiempo de inicio y guarda este valor en la variable ***timin***, después convierte el tiempo en SUDS en tiempo visual c#, lee los factores de conversión de cuentas a nanómetros/segundo, así como los factores para el cálculo del Desplazamiento Reducido. Inicializa el vector promEst[ ] que guarda el promedio para cada traza, calcula que trazas deben dibujarse al inverso en la vertical y finaliza llamando el método que calcula el promedio por trazas. |
| **Métodos que llama** | LeeMux() (Form1).  LeeDmx() (Form1).  LeeGcf() (Form1).  LeeSeisan() (Form1).  LeeYfile() (Form1).  PromediosIniciales() (Form1). |

## LeeMux()

|  |  |
| --- | --- |
| **Nombre** | LeeMux |
| **Descripción** | Esta rutina lee las trazas en formato SUDS Multiplexado. Los datos de las cuentas vienen entonces por paquetes o bloques; se utiliza una variable provisional en un arreglo tridimensional (**cuu**). La primera dimensión corresponde al número de archivo; la segunda al número del bloque en el archivo y la tercera a la posición de la cuenta en el bloque. El tiempo y la rata de muestreo, se guardan en variables locales bidimensionales, ya que ambos valores vienen indicados solo al inicio de los bloques. Lo mismo para las variables **bby** (número de bytes de los datos); **dmx** (número de muestras por bloque para cada estación); **compo** (componente vertical, Este o Norte); **gana** (ganancia de la estación en la tarjeta).  Una vez guardados los valores en estas variables, se comprueba si hay huecos o traslapos entre los archivos y finalmente se asignan los valores a las variables globales (cu, tim, ra, etc.). |
| **Parámetros** | int inu: indica la tarjeta en formato MUX a leer. |
| **Precondición** | Debe existir archivos de sismos MUX en la base de datos. |
| **Postcondición** | Carga en memoria los datos de los sismos almacenados en formato MUX de una tarjeta específica, siendo determinada esta por el valor de la variable **inu.** |
| **Proceso** | Está pendiente una revisión profunda del funcionamiento de este método. |
| **Retorna** | 1 en caso de que el método se ejecute exitosamente, 0 en caso contrario. |
| **Métodos que llama** |  |

## LeeDmx()

|  |  |
| --- | --- |
| **Nombre** | LeeDmx |
| **Descripción** | Esta rutina lee los datos en formato SUDS Demultiplexado. Los datos de cuentas se guardan en una variable tridimensional (cuu), donde primeramente se asigna el número de archivo, luego el número de traza y por último los datos de cuentas. La mayoría de las demás variables son bidimensionales, donde se guarda primeramente el número de archivo y luego el valor de acuerdo al número de traza. Por último, se asignan los datos a las variables globales, estación por estación. En esencia la rutina es muy parecida a la anterior (SUDS Multiplexado), solo que aquí, los datos se desglosan completamente por estación. |
| **Precondición** | Debe existir archivos de sismos DMUX en la base de datos. |
| **Postcondición** | Carga en memoria los datos de los sismos almacenados en formato DMUX de una tarjeta específica, siendo determinada esta por el valor de la variable **inu.** |
| **Proceso** | Está pendiente una revisión profunda del funcionamiento de este método. |
| **Métodos que llama** |  |

## LeeGcf()

|  |  |
| --- | --- |
| **Nombre** | LeeGcf |
| **Descripción** | Rutina que lee el formato GCF de las Guralp. Debe tenerse en cuenta que en este formato, los datos se encuentran invertidos (little endian vs. big endian. El manual no lo dice), así que hay que invertir los datos. Normalmente el visual c# tiene la instrucción array.reverse() para invertir el arreglo. Aquí se usa la notación binaria tal y como se usaba en los programas en D.O.S. (C de la Borland, versión 3.1). (Se debe leer el manual del scream, para conocer este formato). Aquí a diferencia del formato SUDS, los datos se desglosan en archivos individuales por traza, por lo que es necesario tener un archivo adicional, donde se encuentre el nombre de las carpetas y su ruta. La variable con los datos de cuentas, es tridimensional, donde primero se le asigna el número de archivo, luego el número de bloque en el archivo y por último los datos de las cuentas. La mayoría de las demás variables son bidimensionales, donde se les asigna primero el número de archivo y el valor de acuerdo al número de bloque. |
| **Precondición** | Debe existir archivos de sismos GCF en la base de datos. |
| **Postcondición** | Carga en memoria los datos de los sismos almacenados en formato GCF. |
| **Proceso** | Está pendiente una revisión profunda del funcionamiento de este método. |
| **Métodos que llama** |  |

## LeeSeisan()

|  |  |
| --- | --- |
| **Nombre** | LeeSeisan |
| **Descripción** | Rutina que lee el formato SEISAN. La variable con los datos de cuentas, es tridimensional, donde primero se le asigna el número de archivo, luego el número de bloque en el archivo y por último los datos de las cuentas. La mayoría de las demás variables son bidimensionales, donde se les asigna primero el número de archivo y el valor de acuerdo al número de bloque. |
| **Precondición** | Deben existir sismos en formato SEISAN, además es necesario el archivo **estaseisan.txt** el cual se encuentra en la carpeta pro. |
| **Postcondición** | Carga en memoria los datos de los sismos almacenados en formato SEISAN. |
| **Proceso** | Está pendiente una revisión profunda del funcionamiento de este método. |
| **Métodos que llama** |  |

## LeeYfile()

|  |  |
| --- | --- |
| **Nombre** | LeeYfile |
| **Descripción** | Este formato corresponde a las Oriones de la Kinemetrics. Los datos se encuentran desglosados por estaciones con carpetas diferentes. No se pudo contar con suficientes datos en este formato, por lo que se sigue la costumbre de la RSNC, donde los datos se encuentran por HORAS, comenzando en la hora exacta. En versiones posteriores debe mejorarse esta rutina. |
| **Precondición** | Debe existir archivos de sismos YFILE en la base de datos. |
| **Postcondición** | Carga en memoria los datos de los sismos almacenados en formato YFILE. |
| **Proceso** | Está pendiente una revisión profunda del funcionamiento de este método. |
| **Métodos que llama** |  |

## Facnano()

|  |  |
| --- | --- |
| **Nombre** | Facnano |
| **Descripción** | Lee los factores para convertir cuentas a micrometros/segundo se leen los factores para todas las tarjetas y se colocan en una lista para usarse posteriormente en la rutina Lecturas(). |
| **Precondiciones** | Los archivos fcmsG.txt, fcmsJ.txt, fcmsT.txt, fcmsW.txt deben existir en la carpeta Proceso20\bin\Debug\pro |
| **Postcondiciones** | Se cargan los datos de factores para convertir de cuentas a micrometros/seg en el Arraylist ***fclist***. |
| **Excepciones** | Que los datos para convertir las cuentas estén incompletos.  Que no se encuentren los archivos con los datos para la conversión. |
| **Proceso** | Carga en un vector temporal de la clase FileInfo los archivos fcmsG.txt, fcmsJ.txt, fcmsT.txt, fcmsW.txt, dentro de un ciclo while anidado en un foreach va revisando línea por línea que los datos estén completos, es decir que no le falten datos y que su tamaño sea el indicado, en caso de encontrar una línea que no cumpla con los parámetros de evaluación el programa muestra un mensaje en el que indica que archivo en específico es el del error y si aún faltan archivos por revisar se salta al siguiente para revisarlo desde la primer línea; si la línea evaluada cumple con las condiciones los valores de esta son guardados en el string ***ca2*** que posteriormente es guardado en el ArrayList ***fclist*** que contiene los datos de factores para convertir de cuentas a micrometros/seg (archivos fcms?.txt). |

## Facmilimetro()

|  |  |
| --- | --- |
| **Nombre** | Facmilimetro |
| **Descripción** | Rutina que lee los factores para convertir cuentas a milímetros se leen los factores para todas las tarjetas. |
| **Precondiciones** | Los archivos fcamE.txt, fcamG.txt, fcamJ.txt, fcamK.txt, fcamW.txt deben existir en la carpeta Proceso20\bin\Debug\pro |
| **Postcondiciones** | Se cargan los datos de factores para convertir de cuentas a mm en el Arraylist ***fcmm***. |
| **Excepciones** | Que los datos para convertir las cuentas estén incompletos.  Que no se encuentren los archivos con los datos para la conversión. |
| **Proceso** | Carga en un vector temporal de la clase FileInfo los archivos fcamE.txt, fcamG.txt, fcamJ.txt, fcamK.txt, fcamW.txt, dentro de un ciclo while anidado en un foreach va revisando línea por línea que los datos estén completos, es decir que no le falten datos y que su tamaño sea el indicado, si la línea evaluada cumple con las condiciones los valores de esta son guardados en el string ***ca2*** que posteriormente es guardado en el ArrayList ***fcmm*** que contiene los datos de factores para convertir de cuentas a milimetros (archivos fcam?.txt). |

## SeleccionarMinuto()

|  |  |
| --- | --- |
| **Nombre** | SeleccionarMinuto |
| **Descripción** | Configura el tamaño del listBox2 donde se muestran las estaciones, el ítem en el que va a tener seleccionado por defecto y hacer visibles algunos botones necesarios para la clasificación de sismos. |
| **Parámetros** | bool cond: esta variable se utiliza como parámetro que se pasa al método *lecturas(bool)* cuando se hace el llamado. |
| **Postcondiciones** | Oculta los paneles que se despliegan sobre el listBox2.  Hace visibles los botones asociados a la clasificación.  Reconfigura la estructura del listBox2 y escoge el ítem que marca como seleccionado. |
| **Proceso** | Hace false el valor de visibilidad de paneles que están ubicados en posiciones sobre el ***listBox1*** (donde se muestran los rangos de tiempo) panelParti., panelFFTzoom, panelInterP, panelDesplazamiento, panelEspectros, panelBarEspInterp, boEspInterP.  Paso seguido se inicializan algunas variables necesarias para el funcionamiento de este método.  Verifica si la variable disparo es = false y de ser así lanza el método ***lecturas(cond)*** pero en caso de ser true lanza el método ***LecturaDisparo(),***  en ambos casos los métodos retornan un valor entero que se guarda en la variable ***i*** que en caso de ser = 0 (esto quiere decir que no hay trazas) hace la variable estado=false y se sale de la ejecución del método.  Si la variable ***vigilancia*** es = true revisa cada posición del listBox que almacena las estaciones comparando el texto en el listBox con los primeros 4 caracteres de la variable ***estvig*** en caso de coincidir escoge esa estación y la selecciona en el ***listBox2.***  Posteriormente configura la dimensión del listBox2 y hace visibles algunos botones necesarios para el funcionamiento de las clasificaciones. |
| **Métodos que llama** | * Reviarch() (Form1) |

## volcanTarro()

|  |  |
| --- | --- |
| **Nombre** | volcanTarro |
| **Descripción** | Rutina que sirve para escoger el volcán automáticamente cuando se escoge una estación en el panel principal. |
| **Precondiciones** | La estación escogida estar en el archivo ".\pro\estarro.txt" y estar asociada a un volcán específico. |
| **Postcondiciones** | Cambia el color del botón asociado al volcán específico de una estación. |
| **Excepciones** | Que el archivo estarro.txt no exista en la carpeta pro. |
| **Proceso** | Inicia verificando que el archivo estarro.txt exista en la carpeta pro del proyecto en caso de no existir se termina el método, si existe lee el archivo y encuentra la estación especifica que posteriormente compara su primer carácter con el primer carácter de cada uno de los volcanes guardados en el vector volcán hasta encontrar el que busca, al final cambia el color del volcán que encontró a amarillo y los colores de los demás botones de volcanes los cambia al color aguamarina. |

## LLenaBox()

|  |  |
| --- | --- |
| **Nombre** | Llenarbox |
| **Descripción** | Rutina que escoge el intervalo de fechas en la caja de fechas por minuto y busca sismos clasificados en la Base dentro del intervalo de tiempo seleccionado. |
| **Proceso** | Comienza limpiando el contenido del objeto listbox1, si no existe el archivo datos.txt llena el listBox1 con solo fechas que calcula dentro del método, en caso de que el archivo exista lo lee y carga los datos de las fechas y las clasificaciones de sismos desde el archivo datos.txt |
| **Postcondiciones** | Modifica el ***listBox1*** con el intervalo de fechas previamente elegido para la visualización de trazas. |
| **Métodos que llama** | * LeerAmplitud() (form1). * Leerbase() (Util). |

## LeerAmplitud()

|  |  |
| --- | --- |
| **Nombre** | LeerAmplitud |
| **Descripción** | Lee el archivo con los datos necesarios para manejar la amplitud de la base. |
| **Precondiciones** | El archivo amplis.txt debe estar creado en la base. |
| **Proceso** | Inicia contando cuantos registros de amplitud hay en el archivo amplis.txt, luego utiliza los vectores valampl, siPampl y letampl, y volamp para guardar los datos de cada lectura de amplitud que están en el archivo amplis.txt. |
| **Recomendación** | Este método tiene relaciones no explicitas con Leerbase y PonePepas de la clase Util con el que se dibujan las pepas que indican gráficamente que se clasifico un sismo en ese punto.  Podria implementarse una clase Amplitud que contenga esos 4 parámetros (valampl, siPampl y letampl, y volamp) para evitar manejarlos por separado y facilitar su utilidad. |

## Nombre10()

|  |  |
| --- | --- |
| **Nombre** | Nombre10 |
| **Descripción** | Rutina que adecua el nombre del archivo que se desea leer de la base de datos con ciertas características. |
| **Parámetros** | bool sian: si existe la carpeta del año es igual a true en caso contrario es false.  bool sime: si existe la carpeta del mes es igual a true en caso contrario es false.  bool sidi: si existe la carpeta del día es igual a true en caso contrario es false.  string nomprincipal: es el nombre principal de la traza.  string nomext: es la extensión de archive de la traza.  DateTime fech: la fecha que registra la traza. |
| **Retorna** | El nombre del archivo de la traza formateado. |

## PromediosIniciales()

|  |  |
| --- | --- |
| **Nombre** | PromediosIniciales |
| **Descripción** | Calcula el promedio inicial por trazas sumando los primeros 100 valores en la matriz cu, y guarda dichos promedios en el vector que almacena los promedios por estación. |
| **Precondiciones** | La matriz ***cu[][]*** tiene que tener valores asignados a por lo menos 100 columnas. |
| **Postcondiciones** | Guarda los promedios iniciales a todas las trazas en el vector ***promEst[]***. |
| **Excepciones** | Que el número de datos de cuentas sea menor que 100 y al buscar en una posición de la matriz se lance una excepción de valor nulo en la matriz. |
| **Proceso** | Un ciclo for que va desde 0 hasta el número de trazas registrado tiene anidado otro ciclo for que va desde 0 hasta 100 revisando cada posición de la matriz cu y sumando los valores de las cuentas que en este se encuentran guardados, al finalizar el ciclo interno calcula el promedio de esta suma y lo guarda en la posición j-esima del vector promEst[j] y continua con la ejecución del for externo; repite esta operación las veces como trazas haya. |

## Clasificar()

|  |  |
| --- | --- |
| **Nombre** | Clasificar |
| **Descripción** | Esta rutina adecua el panel de clasificación para posteriormente dibujar sobre el las trazas. |
| **Proceso** | Verifica que tosas las estaciones estén seleccionadas en caso de encontrar que una estación no está seleccionada cambia el color del botón ***boTodas*** a marrón, si todas están seleccionadas lo deja de color blanco, luego verifica la existencia del scilab si este se encuentra hace visible el botón ***boScilab***  y calcula tie1 y tie2 que corresponden a los extremos del intervalo de tiempo seleccionado en el panel principal y con respecto al cual se grafican las trazas en el panel de clasificación, por ultimo llama los métodos Dibujo trazas() y Letreros(Panel,int,int,string,Color). |
| **Métodos que llama** | DibujoTrazas() (Form1).  Letreros() (Util). |

## YaClasificados()

|  |  |
| --- | --- |
| **Nombre** | YaClasificados |
| **Descripción** | Rutina que comprueba si existen sismos ya clasificados para el inicio de la ventana seleccionada. Si existen, se lo comunica al usuario, con el fin que no exista reemplazos indeseados, de las trazas de la Base. |
| **Precondiciones** | El archivo datos.txt debe existir en la carpeta del proyecto. |
| **Postcondiciones** | Muestra un mensaje informando que el sismo ya fue clasificado anteriormente. |
| **Proceso** | Carga en memoria el archivo datos.txt y compara los tiempos de los sismos clasificados con el tiempo del sismo seleccionado actualmente si son iguales se muestra un mensaje informando que este sismo ya a sido clasificado. |

## DibujoTrazas()

|  |  |
| --- | --- |
| **Nombre** | DibujoTrazas. |
| **Descripción** | Hace el llamado a los métodos que dibujan las trazas dependiendo la forma como se quiere que se muestren, si analógicas o por cuentas. |
| **Postcondiciones** | Dibuja las trazas en el panel de clasificación panelEsta. |
| **Proceso** | Verifica que el botón ***boNano*** tenga como texto “Tra” o que la variable booleana ***anologico*** sea true, si se cumple alguna de estas condiciones lanza el método que hace el dibujo de trazas analógico en caso contrario lanza el método que las dibuja por cuentas. |
| **Métodos que llama** | TrazasClas() (Form1)  TrazasClasCuentas() (Form1) |

## fecha()

|  |  |
| --- | --- |
| **Nombre** | fecha |
| **Descripción** | Muestra el panel de dialogo para entrar la fecha inicial y final, así como la duración de la traza de la estación escogida. |
| **Proceso** | Crea la variable ***di*** y setea los valores de las fechas y el usuario esta variable, si en la ventana de dialogo se escoge el botón Ok configura la fecha inicial y final en para crear el rango de tiempo del que mostrar las trazas, luego llama el método LlenarBox().  En caso de que en el panel de dialogo se escoja la opción cancelar se configuran los valores de visibilidad de los botones que mostrará en el panel principal.  Si se asignaron fecha de inicio y fecha fin se llama el método borra(Panel, Color) y se dibuja el ***panel1***, además si el ***panelcla (donde se clasifican los sismos)*** esta visible asigna el tamaño y posición del ***panelcla*** y lanza los métodos letreros(Panel, int, int, string, Color) y DibujoTrazas(). |
| **Métodos que llama** | * LLenaBox() (Form1). * borra() (Util). * Letreros() (Util). * DibujoTrazas() (Form1). |

## TrazasClas()

|  |  |
| --- | --- |
| **Nombre** | TrazasClas |
| **Descripción** | Se encarga de dibujar las trazas en el panel de clasificación según la porción de traza seleccionada, hace el dibujo semejante a la señal analógica. |
| **Postcondiciones** | Dibuja la porción de traza seleccionada en el panel ***panelcladib*** para cada estación en la que esté presente. |
| **Proceso** | En el primer for revisa las trazas leídas que se cargaron en memoria con el fin de ver si están dentro del rango de tiempo especificado al momento de arrastrar sobre la traza de una estación, en caso de estar en ese rango de tiempo hace true la variable booleana ***siEst[]*** en la posición específica de cada traza y modifica las variables locales ***val[][]*** y ***timm[][]***, posteriormente configura la posición y color del panel donde se va a dibujar las trazas, después determina la cantidad de trazas que están dentro del rango de tiempo para usarlo como factor de tamaño del panel.  Paso seguido crea un for que va desde 0 hasta la cantidad de trazas en memoria en el verifica mediante la variable ***siesta[]*** que la traza este en el rango de tiempo específico y determina los valores máximo y mínimo de cuentas de dicho intervalo.  Luego configura las variables locales que permiten saber el tamaño y posición donde se deben dibujar las trazas, dibuja el nombre de la estación y dependiendo del valor booleano de la variable ***satu*** y si la posición especifica que se está verificando en ese momento del vector ***invertido[]*** es true o false guarda los datos de la traza de manera diferente en el vector de tipo Point ***dat[ ]*** esto con el fin de que más adelante cuando se dibujen se haga de manera invertida o no; al final verifica si la variable local error es igual a true y de ser así este método se llama de nuevo el mismo. |
| **Métodos que llama** | * TrazasClas() “Solo en caso de presentarse un error en la ejecución del mismo” |

## TrazasClasCuentas()

|  |  |
| --- | --- |
| **Nombre** | TrazasClasCuentas |
| **Descripción** | Se encarga de dibujar las trazas en el panel de clasificación según la porción de traza seleccionada, dependiendo del texto del botón boNano puede hacer el dibujo de las trazas de 2 formas distintas con base a la unidad de cuentas o en base a los nanómetros/seg. |
| **Postcondiciones** | Dibuja la porción de traza seleccionada en el panel ***panelcladib*** para cada estación en la que esté presente. |
| **Proceso** | En el primer for revisa las trazas leídas que se cargaron en memoria con el fin de ver si están dentro del rango de tiempo especificado al momento de arrastrar sobre la traza de una estación, en caso de estar en ese rango de tiempo hace true la variable booleana ***siEst[]*** en la posición específica de cada traza y modifica las variables locales ***val[][]*** y ***timm[][]***, posteriormente configura la posición y color del panel donde se va a dibujar las trazas, después determina la cantidad de trazas que están dentro del rango de tiempo para usarlo como factor de tamaño del panel.  Paso seguido crea un for que va desde 0 hasta la cantidad de trazas en memoria en el verifica mediante la variable ***siesta[]*** que la traza este en el rango de tiempo específico y determina los valores máximo y mínimo de cuentas de dicho intervalo.  Luego configura las variables locales que permiten saber el tamaño y posición donde se deben dibujar las trazas, dibuja el nombre de la estación y dependiendo del valor booleano de la variable ***satu*** y si la posición especifica que se está verificando en ese momento del vector ***invertido[]*** es true o false guarda los datos de la traza de manera diferente en el vector de tipo Point ***dat[ ]*** esto con el fin de que más adelante cuando se dibujen se haga de manera invertida o no; al final verifica si la variable local error es igual a true y de ser así este método se llama de nuevo el mismo. |

## Promedio()

|  |  |
| --- | --- |
| **Nombre** | Promedio |
| **Descripción** | Controla el estado de la variable sipro la cual indica el promedio de la señal, además cambia el color del botón boprom dependiendo dicho estado. |
| **Postcondiciones** | Cambia el valor de sipro y el color del botón bopro. |
| **Proceso** | Si el valor de sipro es 0 lo cambia a 1 y modifica el color del botón bopro por color rojo, en caso de que sipro sea 1 lo hace 0 y actualiza el color de bopro a blanco. |

## Subir()

|  |  |
| --- | --- |
| **Nombre** | Subir |
| **Descripción** | Desplaza la señal hacia arriba. |
| **Parámetros** | MouseEventArgs e: evento de click lanzado por el botón. |
| **Postcondiciones** | La señal en el panel principal se desplaza hacia arriba |
| **Proceso** | Redice en 1 el valor de la variable ***incy*** en caso de que el evento se halla lanzado con el click izquierdo, si fue lanzado con el click derecho se reduce en 15 su valor. |

## Bajar()

|  |  |
| --- | --- |
| **Nombre** | Bajar |
| **Descripción** | Desplaza la señal hacia abajo |
| **Parámetros** | MouseEventArgs e: evento de click lanzado por el botón. |
| **Postcondiciones** | La señal en el panel principal se desplaza hacia abajo. |
| **Proceso** | Aumenta en 1 el valor de la variable ***incy*** en caso de que el evento se halla lanzado con el click izquierdo, si fue lanzado con el click derecho se aumenta en 15 su valor. |

## Aumentar()

|  |  |
| --- | --- |
| **Nombre** | Aumentar |
| **Descripción** | Aumenta el valor de la variable ***ampli***  que es la encargada de controlar la amplitud de las trazas que se dibujan en el panel principal. |
| **Parámetros** | MouseEventArgs e: evento de click lanzado por el botón. |
| **Postcondiciones** | La amplitud de la señal se aumenta lo que causa una visualización de la señal más grande en términos de amplitud. |
| **Proceso** | Si el evento de click es generado por el botón izquierdo se multiplica la variable ***ampli*** por 1.2F, en caso de ser con el botón derecho se multiplica a ***ampli*** por 2.0F. |

## Disminuir()

|  |  |
| --- | --- |
| **Nombre** | Disminuir |
| **Descripción** | Disminuye el valor de la variable ***ampli***  que es la encargada de controlar la amplitud de las trazas que se dibujan en el panel principal. |
| **Parámetros** | MouseEventArgs e: evento de click lanzado por el botón. |
| **Postcondiciones** | La amplitud de la señal se disminuye lo que causa una visualización de la señal más pequeña en términos de amplitud. |
| **Proceso** | Si el evento de click es generado por el botón izquierdo se multiplica la variable ***ampli*** por 0.8F, en caso de ser con el botón derecho se multiplica a ***ampli*** por 0.5F. |

## Uno()

|  |  |
| --- | --- |
| **Nombre** | Uno |
| **Descripción** | Regresa el valor de la variable ampli y el tamaño del dibujo de la señal a su condición inicial. |
| **Postcondiciones** | Regresa el valor de la variable ampli y el tamaño del dibujo de la señal a su condición inicial. |

## Parametros()

|  |  |
| --- | --- |
| **Nombre** | Parametros |
| **Descripción** | Método encargado de modificar los valores duración en minutos de una línea, duración en segundos de la ventana, separación en pixeles de las líneas y tamaño en pixeles de las pepas. |
| **Postcondiciones** | Se actualizan los nuevos valores de las variables ***totven, tam, dur, esp*** y actualiza los cambios en la visualización. |
| **Proceso** | Despliega un panel de dialogo en el cual captura los valores para las variables ***totven, tam, dur, esp*** que en el caso de que el usuario acepte aplicar dando click al botón Ok del panel de dialogo se encarga de actualizar los valores de las variables y refrzcar la pantalla. |

## Seguir()

|  |  |
| --- | --- |
| **Nombre** | Seguir |
| **Descripción** | Rutina que permite visualizar la continuación en el tiempo de la traza de la estación seleccionada, cambia el índice del listBox de las estaciones. |
| **Parámetros** | boo cond: Si es true hace el cálculo de la estación siguiente, en caso sontrario usa el parámetro ii.  int ii: índice de la estación siguiente a la estación a la que se le está leyendo la traza. |
| **Postcondiciones** | Cambia el índice del listBox de las estaciones. |
| **Proceso** | Si la variable ***cond*** es true calcula el índice nuevo para el listBox de las estaciones en caso de que sea false le asigna la variables ***ii***  a dicho índice. |

## Archivo()

|  |  |
| --- | --- |
| **Nombre** | Archivo |
| **Descripción** | Permite visualizar la extensión de tiempo de la traza de los archivos ya clasificados. |
| **Precondición** | El botón ***butarch*** debe estar visible en el panel. |
| **Parámetros** | MouseEventArgs e: evento de mouse que lanza el botón ***butarch***. |
| **Postcondiciones** | Dibuja los recuadros que indican el rango de tiempo de una traza si ya se a clasificado, estos recuadros son de diferentes colores. |
| **Proceso** | Si el parámetro e es null cambia el valor de la variables ***siArch***  a true y llama en este orden los métodos ***LeerBase()*** y ***VerArchi().***  En caso de que ***siArch*** == true valida que el evento del click haya sido lanzado con el click derecho y de ser así lanza en este orden los métodos ***LeerBase() sihayclas()*** y ***VerArchi()*** si es false hace el valor de ***siArch*** = a false y cambia el color del botón ***butarch*** a blanco.  Ahora bien si ***siArch*** == false hace que cambie su valor a true, cambia el color del botón ***butArch*** a dorado, verifica si el evento del click fue lanzado por el botón derecho del mouse y de ser asi llama los métodos ***LeerBase()*** y ***sihayclas()***.  Por ultimo lanza el método ***VerArchi()*** que es para dibijar los rectángulos que muestran la extensión de la tiempo clasificado de las trazas y hace visible el botón ***boEliClas***  visible si la variable ***sihayclas*** es true. |
| **Métodos que llama** | Leerbase() (Util).  VerArchi() (Util).  Reviarch() (Form1) |

## Reviarch()

|  |  |
| --- | --- |
| **Nombre** | Reviarch |
| **Descripción** | Rutina que revisa los archivos de clasificación con el fin de averiguar los intervalos de tiempo que se encuentran clasificados. En Datos.txt se encuentran los tiempos en formato visual c#. |
| **Precondición** | El archivo datos.txt debe estar en la base de datos. |
| **Postcondiciones** | Configura los vectores que guardan la información de las trazas que ya han sido clasificadas, |
| **Proceso** | Si el archivo datos.txt no existe hace la variable ***siArch*** = false y retorna un false; si el archivo existe configura las variables locales t1 y t2 que representan el tiempo inicial y final respectivamente de la traza de la estación activa.  Paso seguido dentro de un while revisa el archivo datos.txt y cuenta cuantos sismos hay clasificados para la traza activa, en caso de encontrar sismos guarda los tiempos iniciales, duración y nombre de los mismos en los vectores double ***tiar,*** string ***duar*** ystring ***nomar*** respectivamente, estos vectores son configurados seguidamente en otro while que lee de nuevo el archivo datos.txt.  Finaliza cambiando el color ***butarch*** a blanco y retornando un true***.*** |

## Cuadro\_Tremor()

|  |  |
| --- | --- |
| **Nombre** | Cuadro\_Tremor |
| **Descripción** | Rutina que presenta un rectángulo indicando la duración de los archivos de "tremor", con el fin que el usuario realice las lecturas de amplitud, dentro de dicho rectángulo, el cual se va desplazando consecutivamente y automáticamente, una vez se realicen dichas lecturas. |
| **Precondición** | La variable booleana ***tremor*** debe ser ture. |
| **Postcondiciones** | Queda representado gráficamente el rectángulo que muestra la duración del tremor. |
| **Proceso** | Calcula el delta entre el tiempo inicial y final del tremor y configura las dimensiones del rectángulo utilizando el tiempo inicial en que se registró una traza y dibuja el rectángulo. |

## CalcularEspectro()

|  |  |
| --- | --- |
| **Nombre** | CalcularEspectro |
| **Descripción** | Se encarga de calcular los valores de las variables para el posterior cálculo del espectro y posicionar la gráfica del mismo. |
| **Precondición** |  |
| **Parámetros** | Panel pan: panel principal donde se están mostrando las trazas, puede ser el panel principal panel1 o el panel secundario panel1a.  Panel panelBar: panel donde se grafica el espectro.  ushort id: el id de la estación de la traza a la que se desea calcular el espectro.  MouseEventArgs e: evento de mouse que se genera al dar click sobre el panel. |
| **Proceso** | Comienza inicializando las variables xf y yf con las que determina el ancho y alto del panel donde se va a calcular el espectro, y la variable jb que contiene la última muestra de la traza actual, con estas variables se calcula el ***t1esp*** que sirva como tiempo inicial desde donde se calcula el espectro, además de asignar valores a las variables ***yesp***  y ***xesp*** que posteriormente servirán para posicionar el panel donde se dibuja el espectro. |
| **Métodos que llama** | Espectro() (Form1) |

## Espectro()

|  |  |
| --- | --- |
| **Nombre** | Espectro |
| **Descripción** | Se encarga de calcular el espectro de una porción de traza y preparar el panel donde se muestra el resultado. |
| **Postcondición** | Queda guardado el cálculo del espectro para una porción de traza especificada en la variable ***vaesp[]***, además despliega la gráfica de dicho espectro. |
| **Parámetros** | Panel pan: panel en el que se está clasificando la traza actual, panel1 en caso de ser el panel principal, y panel1a en caso de ser el panel secundario.  Panel panelBar: se usa para dibujar una barra de ubicación.  ushort id: id de la traza que se está clasificando.  bool cond: se usa para verificar si se modifica o no la localización del panel ***panelFFTzoom,***  si cond es true se cambia la posición del panel, si es false no. |
| **Proceso** | Si la variable VerEspectro es false se sale del método mediante un return, sino determina la cantidad de cuentas que hay en la traza y determina el valor de la variable ***denom*** calculando el delta de tiempo entre el tiempo final registrado en la traza con el tiempo mínimo que se registró una traza y dividiendo este resultado entre la duración del intervalo de tiempo en que se seleccionó para calcularle el espectro.  Ahora hace los cálculos para asignar la localización gráfica del panel ***panelFFTzoom***, posteriormente hace el cálculo del espectro mediante el método ***panelFFTzoom*** el cual almacena en el vector de tipo double ***vaesp[]***, a los valores obtenidos del cálculo del espectro dependiendo si la variable ***silog*** es true les saca logaritmo en base 10 solo si el valor es mayor que 0 en caso de que sea menor que 0 les asigna -1000000.0 directamente.  Configura la localización y tamaño del panel que utiliza como barra de ubicación (***panelBar***) y por último si la variable ***silog*** == false llama el método GraficaEspectro() para graficar el espectro, en caso de que sea true llama el método ***GraficaEspectroLog().*** |
| **Métodos que llama** | TrazasClas() (Form1)  RealFFTAmpli() (Fourier)  GraficaEspectro() (Form1)  GraficaEspectroLog() (Form1) |

## GraficaEspectro()

|  |  |
| --- | --- |
| **Nombre** | GraficaEspectro |
| **Descripción** | Realiza el gráfico del espectro de una porción de traza, y lo despliega en el panel panelFFTzoom. |
| **Postcondición** | Despliega el panel panelFFTzoom mostrando gráficamente el espectro de la porción de la traza que se seleccionó. |
| **Parámetros** | int id: id de la traza que se está clasificando.  double[] va: los valores del espectro.  bool cond: indica que el checkBoxFFT1 está seleccionado, por ende que se aplique la FFT (transformada rápida de Fourier) a la señal.  bool vacioesp: true si hay vacios en el espectro, false si el cálculo fue continuo. |
| **Proceso** | En un for que recorre el arreglo ***va[]*** que almacena los valores del espectro determina cual es el menor y el mayor valor de espectro, posteriormente calcula el promedio de la suma de estos y usa este valor para acomodar la gráfica de los puntos del espectro en el panel ***panelFFTzoom,*** estos puntos son almacenados en el arreglo ***dat[]*** de la clase Point |

## GraficaEspectroLog()

|  |  |
| --- | --- |
| **Nombre** | GraficaEspectroLog |
| **Descripción** | Realiza el gráfico del espectro de una porción de traza, y lo despliega en el panel panelFFTzoom, anexa un factor al cual le aplica logaritmo en base 10 y con este modifica la gráfica del espectro antes calculado. |
| **Postcondición** | Despliega el panel panelFFTzoom mostrando gráficamente el espectro de la porción de la traza que se seleccionó, aplicando logaritmo en base 10 a un factor que modifica el gráfico del espectro antes calculado. |
| **Parámetros** | int id: id de la traza que se está clasificando.  double[] va: los valores del espectro.  bool vacioesp: true si hay vacios en el espectro, false si el cálculo fue continuo. |
| **Proceso** | En un for que recorre el arreglo ***va[]*** que almacena los valores del espectro determina cual es el menor y el mayor valor de espectro, posteriormente calcula el promedio de la suma de estos y usa este valor para acomodar la gráfica de los puntos del espectro en el panel ***panelFFTzoom,*** estos puntos son almacenados en el arreglo ***dat[]*** de la clase Point. |

## Dibujocoda()

|  |  |
| --- | --- |
| **Nombre** | Dibujocoda |
| **Descripción** | Dibuja la traza de la estación seleccionada, en el panel de Coda. |
| **Parámetros** | int[] va: vector con las cuentas que forman una traza especifica.  short idd: el id de la estación a la que se va asignar los valores P, S, C. |
| **Precondición** | La estación seleccionada debe tener una traza. |
| **Postcondición** | Se dibuja la traza de la estación seleccionada en el panelcoda y lo despliega. |
| **Proceso** | Dibuja un rectángulo blanco en el panelcoda, determina la posición en la matriz ***tim[][]*** donde inicia el minuto desde donde se hizo el arrastre para clasificar el sismo, igualmente la posición del minuto donde se terminó el arrastre, de la misma forma determina el menor y el mayor valor de cuenta en esta porción de tiempo.  Realiza verificaciones de estado de variables booleanas y dependiendo de su estado hace el dibujo de la traza de forma distinta, las variable que verifica son:  analogcoda, si es true dibuja la traza en forma analógica.  satu, si es true dibuja la traza más pequeña ya que viene saturada.  invertido[idd], para verificar si se dibuja la traza invertida o no. |
| **Métodos que llama** | borra() (Util)  DibujoClascoda() (Util) |

## DibujoClascoda()

|  |  |
| --- | --- |
| **Nombre** | DibujoClascoda |
| **Descripción** | Dibuja en color naranja, en el panel de clasificación, el sector de coda seleccionado por el usuario. |
| **Postcondición** | Se pinta de color naranja la porción de traza seleccionada en el panel de clasificación. |
| **Proceso** | Determina en que estación se está leyendo la traza y cuantas trazas hay en las estaciones activas (numtotra), luego cuanto es el tiempo que se seleccionó arrastrando sobre la traza.  Determina la posición en el vector de tiempos de las muestras en las que se encuentran tanto el tie1 (cuando se empezó el arrastre sobre la traza) como el tiempo tie2 (momento cuando se termina el arrastre), la posición de tie1 la guarda en nm1 y la de tie2 en nm2.  Determina en el lapso de muestras de cuentas del vector ***cu[]*** comprendido entre las posiciones nm1 y nm2 la muestra con menor valor y la de mayor valor.  Por último se encarga de asignar los valores de los puntos que se van a pintar de color naranja guardándolos en el arreglo de tipo Point ***dat[].*** |

## CalcularEspectroCla()

|  |  |
| --- | --- |
| **Nombre** | CalcularEspectroCla |
| **Descripción** | Determina la posición inicial del espectro y el tiempo desde donde se empieza a calcular. |
| **Parámetros** | Panel panelBar: se usa para pasar como parámetro del método Espectro() que llama al final del método.  ushort idc: es el id de la traza escogida del panel de clasificación usada para el espectro.  MouseEventArgs e: es el evento que desencadena la ejecución del método, este parámetro es usado para determinar la posición donde se desencadeno el evento. |
| **Proceso** | Verifica que se cumplan las precondiciones necesarias para la ejecución del método, siendo estas que VerEspectro = true, que e.x >= 40 y que estado = true en caso de que alguna no se cumpla sale de la ejecución del método.  Determina la posición en la fila representada por la variable ***idc*** en la matriz de tiempos donde el tiempo es mayor que 0, posteriormente verifica que esa posición sea > 2 en caso de no serlo sale del método, si se cumple modifica los valores de las variables globales xesp, yesp y t1esp siendo posición inicial en x del espectro, posición inicial en y del espectro y tiempo inicial del espectro, al final llama a el método ***Espectro(panelcladib, panelBar, idc, true)***. |
| **Métodos que llama** | Espectro() (Form1). |

## PromedioFiltrado() está pendiente

|  |  |
| --- | --- |
| **Nombre** | PromedioFiltrado |
| **Descripción** |  |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** | PasaAltos() (Util) |

## CalculoInterpolacion() pendiente

|  |  |
| --- | --- |
| **Nombre** | CalculoInterpolacion |
| **Descripción** |  |
| **Parámetros** | int idd: indica el id de la estación que se está clasificando. |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** | Mensaje() (Util)  Dos() (Util)  PasaAltos() (Util) |

## Saturacion()

|  |  |
| --- | --- |
| **Nombre** | Saturacion |
| **Descripción** | Regula el valor de la variable satur,con la cual se determina si se debe limitar o no el espacio de dibujo de las trazas. |
| **Parámetros** | MouseEventArgs e: evento que se usa para determinar con que botón del mouse se dio click. |
| **Proceso** | Si el click que se dio sobre el botón boSatu fue con el botón izquierdo la variable booleana ***satu*** es false, cambia el valor de la variable ***satur*** a 0 para limitar el espacio de dibujo de la señal.  Ahora en el caso de que el click se haya dado con el botón derecho del mouse lo que hace es aumentar en 1 el valor de ***satur*** y cambiar a ***satu*** a true. |

## Dibujo()

|  |  |
| --- | --- |
| **Nombre** | Dibujo |
| **Descripción** | Rutina que dibuja toda la traza de la estación seleccionada y la cual sirve de referencia para la clasificación. |
| **Parámetros** | Panel pan: panel donde se dibuja la traza, generalmente son los paneles **panel1** y **panel1a**.  ushort id: corresponde al número de traza de la estación seleccionada.  int[] va: contiene los datos de las cuentas. |
| **Precondición** | Deben existir trazas, en caso de que no haya se termina la ejecución del método. |
| **Postcondición** | Grafica en el panel que se pasó como parámetro la traza de la estación seleccionada. |
| **Proceso** | Inicia comprobando que se cumplan las condiciones necesarias para su ejecución y preparando el panel para graficar sobre él.  Con base a la duración de la traza calcula la forma de como dibujarla y el tamaño.  En caso de que se haya buscado el promedio de una porción de esa traza (sipro=2) determina el valor mínimo y máximo de cuenta de ese sector y los guarda en **mxp** y **mnp** respectivamente.  Si al variable **analogico** es falsa deja el espaciamiento entre líneas por defecto, pero en caso de ser true realiza un cálculo para determinar dicho espaciamiento.  Ahora determina los valores mínimo y máximo de cuenta de toda la traza y los guarda en **min** y **max** respectivamente.  Dibuja en el panel la fecha, hora, total de minutos a graficar, el espaciamiento entre líneas y el nombre de usuario.  Posteriormente dentro de un for que va de 0 hasta el valor de la variable **denom** (variable que guarda el número de líneas a graficar) anidado en un try, teniendo en cuenta si la señal está saturada (en caso de que la traza sea muy grande y se tenga que reducir su amplitud) y si se desea que se grafique la traza de forma invertida (esto lo controla el vector **invertido[]**) determina los valores de los puntos que formaran la gráfica de la traza y los guarda en el vector **dat[]** de la clase Point, al finalizar de calcular los puntos realiza la gráfica de la traza.  Finalmente si están activas las pepas para la visualización las dibuja. |
| **Métodos que llama** | borra() (Util)  MarcaTiempo() (Util) Calcula a que distancia se deben pintar las líneas zapotes que se pintan como marcas de tiempo y pinta las líneas.  PonePepas() (Util)  VerArchi() (Util) |

## DibujoCodaAmp()

|  |  |
| --- | --- |
| **Nombre** | DibujoCodaAmp |
| **Descripción** | Dibuja en el panel de coda, el sector correspondiente al periodo y amplitud leído. |
| **Precondición** | Debe existir una lectura de periodo y amplitud para el sector de traza que se está clasificando en el panelcoda. |
| **Postcondición** | Dibuja sector al que se le escogió amplitud. |
| **Proceso** | Determina los límites del panelcoda y los valore min y max de cuentas del sector de traza seleccionado para graficar la amplitud.  Determina los puntos a graficar de la traza y los guarda en el vector de la clase Point **dat;**  estos puntos son graficados de maneras diferentes dependiendo de si esta activada la opción analógico y la opción saturación. |
| **Métodos que llama** |  |

## DibAmpl()

|  |  |
| --- | --- |
| **Nombre** | DibAmpl |
| **Descripción** | Rutina que dibuja el sector de traza escogido en el panel de coda y donde se efectúa la lectura de la amplitud y el periodo. |
| **Precondición** | Se debe seleccionar una porción de traza a la cual clasificarle la amplitud. |
| **Postcondición** | Grafica en el panelAmp la porción de traza a la que se le asigna valor de amplitud |
| **Proceso** | Inicialmente prepara el panelAmp que es donde se dibuja el sector de traza que se seleccionó al cual se le hará la lectura de la amplitud, luego determina la posición en la que está la muestra seleccionada en la matriz **tim** (la que guarda los tiempos de las cuentas).  Ahora llena el vector dat de la clase Point con los valores de las cuentas a dibujar, para esta tarea tienen en cuenta las variables **satu** e **invertido[]** y dependiendo de sus valores de verdad cambia la forma de visualización de la traza, posteriormente dibuja el sector de traza seleccionado. |
| **Métodos que llama** | borra() (Util)  DibujoCodaAmp() (Form1) |

## variasamplitudes()

|  |  |
| --- | --- |
| **Nombre** | variasamplitudes |
| **Descripción** | Rutina que lee el archivo donde se guardan los valores de las lecturas cuando se realizan lecturas para varios sismos, y guarda dichas lecturas en la base. |
| **Parámetros** | string clasi: clasificación con la que se guarda el sismo.  string clas: clasificación que se le asigna al sismo.  string ss: fecha formateada de la traza. ej: 20140318122045 año-mes-día-hora-minuto-segundo. |
| **Proceso** | Carga en memoria los datos de la amplitud esto lo hace del archivo **amplivarias.txt** y con estos valores formatea el nombre del archivo que va a guardar cada amplitud por separado. |

## GrabaBase() falta vincular

|  |  |
| --- | --- |
| **Nombre** | GrabaBase |
| **Descripción** | Rutina que Graba Todos los datos del sismo en la Base en su propio archivo y además graba en el archivo cla.txt el sismo como ya clasificado. |
| **Postcondición** | Graba en la base de datos de sismos y en el archivo de sismos clasificados el sismo a clasificar. |
| **Proceso** | Comienza verificando que la porción de traza seleccionada a clasificar tenga una duración mayor de 10 segundos de lo contrario no graba nada y muestra un mensaje informando este hecho.  Ahora ya sea que no se haya dado valor a Pti, Ati, periodo, o clas[1] sea = “\_” muestra un mensaje informando específicamente que valor no se asignó y pregunta si se desea salir del proceso de guardado, en caso de contestar que si se termina la ejecución del método y retorna un -1, en caso contrario continua.  Verifica que el valor que determina el final de tremor este asignado en caso de no encontrar este valor lo indica la usuario con un mensaje en pantalla.  Verifica si Pti = Ati = periodo = 0 en caso se ser true hace la variable booleanan gralec = false lo que indica que esa lectura no se grabaria.  Posteriormente formatea el nombre del archivo en el que se guardara el sismo *("D:\\base\\sud\\LP\\14\\03181020.MLP")* si este archivo ya existe muestra un mensaje informándolo si no exite lo crea y guarda el sismo, además registra que se clasificacó este sismo en el archivo cla.tx que está en la base de datos en la carpeta cla y allí se guardan las clasificaciones por mes; al final si el proceso de grabación del sismo en la base de datos es exitoso retorna un 1. |
| **Métodos que llama** | Mensaje() (Util)  variasamplitudes() (Form1)  GrabaSuds() (Form1)  LLenaBox() (Form1)  RevisaGrabacion() (Form1) //Revisa que se hayan grabado las lecturas y la traza en la Base. |

## GrabaSuds()

|  |  |
| --- | --- |
| **Nombre** | GrabaSuds |
| **Descripción** | Rutina que graba las trazas en la Base en formato SUDS Demultiplexado. |
| **Parámetros** | short bas: en la ejecución del programa se lo invoca con 3 parámetros diferentes (0, 1 y 2).  Cuando se graba con el parámetro = 0 es el formato por defecto.  Cuando se graba con el parámetro = 1 es para ser leída por el PSW.  Cuando se graba con el parámetro = 2 es para ser leída por el SCILAB. |
| **Proceso** | Está pendiente una revisión profunda del funcionamiento de este método. |

## VerTrazasArribos()

|  |  |
| --- | --- |
| **Nombre** | VerTrazasArribos |
| **Descripción** | Rutina que permite ver los arribos secuencialmente, aunque debe calibrase para aquellos sismos de interés. Esta parte realiza la búsqueda de los primeros arribos en las estaciones. |
| **Parámetros** | int[] idcan: Indice de las estaciones en la vista.  double tii: Representa el momento en punto en la traza donde se inicio el arrastre para la clasificación -1.  double[][] time: Tiempo para la visualización de arribos.  bool cond: Condición que indica si en el archivo /oct/estaML.txt hay datos de meridiano y longitud para las estaciones.  bool[] siML: Condición que indica si el nombre de la estación esta en los archivos /oct/estaML.txt y /pro/estavista.txt. |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** | Ordena los arribos según el tiempo de llegada que se almacena en la matriz time[][]  Necesita una revisión a fondo de su funcionamiento. |

## VerVista()

|  |  |
| --- | --- |
| **Nombre** | VerVista |
| **Descripción** | Grafica los primeros arribos en la modalidad de mapa. |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** | Inicia comprobando la existencia del archivo ".\\pro\\estavista.txt" si este no existe termina la ejecución del método, continua verificando si el archivo ".\\oct\\estaML.txt"  Necesita una revisión a fondo de su funcionamiento. |
| **Métodos que llama** | TopoMapaArribos() (Util)  EstacionesArribos (Util)  Borra()(Util)  VerTrazasArribos()(Form1)  Circulos() (Form1)  UnaEstacionArribo() (Util)  Dos()(Util) |

## ZoomEspectro()

|  |  |
| --- | --- |
| **Nombre** | ZoomEspectro |
| **Descripción** | Determina los valores mayor y menor del espectro y con estos calcula cual debe ser el alto del panel, después tomando los valores del espectro calcula los puntos x,y en que debe dibujar cada punto y los dibuja, esto genera una gráfica. |
| **Parámetros** | MouseEventArgs e: Se usa para determinar la posición en donde está ubicado el mouse. |

## MovimientoParticula() falta terminar

|  |  |
| --- | --- |
| **Nombre** | MovimientoParticula |
| **Descripción** |  |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** | VerMapa() (util)  TrazaComponente() (Form1)  TrazaComponenteInterp() (Form1) |

## TrazaComponenteInterp()falta terminar

|  |  |
| --- | --- |
| **Nombre** | TrazaComponenteInterp |
| **Descripción** |  |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** | borra() (util)  CalculoInterpolacion() (Form1) |

## TrazaComponente() falta terminar

|  |  |
| --- | --- |
| **Nombre** | TrazaComponente |
| **Descripción** |  |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** | borra() (util)  CalculoInterpolacion() (Form1) |

## MoverNEZ() falta terminar

|  |  |
| --- | --- |
| **Nombre** | MoverNEZ |
| **Descripción** |  |
| **Parámetros** | int[] vN:  int[] vE:  int[] vZ:  DoWorkEventArgs e: |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** |  |

## MoverNEZinterp() falta terminar

|  |  |
| --- | --- |
| **Nombre** | MoverNEZinterp |
| **Descripción** |  |
| **Parámetros** |  |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** |  |

## RevisarEstacionNeic()

|  |  |
| --- | --- |
| **Nombre** | boNeic\_MouseDown |
| **Descripción** | Verifica la existencia del archivo /pro/tab/estaciones.dat. |
| **Retorna** | Retorna true en caso de encontrarlo, false en caso contrario. |
| **Proceso** | Verifica que exista el archivo \tab\\estaciones.dat y dependiendo si existe o no retorna true o false. |

## VerRespuestaFiltro()

|  |  |
| --- | --- |
| **Nombre** | VerRespuestaFiltro |
| **Descripción** | Dependiendo del valor de cfilx (1,2,3) determina qué tipo de filtro aplicar a la traza del panel1a, paso seguido determina los puntos a aplicar el filtro y grafica dichos puntos en el panel1a. |
| **Métodos que llama** | HBajo() (Util)  HAlto() (Util)  HBand() (Util)  RealFFTAmpli() (Fourier) // Devuelve solamente la amplitud del espectro como valores reales. |

## CalculoInterpolacion()

|  |  |
| --- | --- |
| **Nombre** | CalculoInterpolacion |
| **Descripción** | Realiza una interpolación en los datos de la traza identificada por el valor del parámetro idd utilizando el octave. |
| **Parámetros** | int idd: Indica el id de la estación que se está clasificando. |
| **Precondición** | Debe estar instalado el programa Octave. |
| **Retorna** | true en caso de que la interpolación se realice exitosamente, false en caso contrario. |
| **Postcondición** | Guarda en memoria los datos de la traza interpolada y realiza la gráfica de dicho proceso. |
| **Proceso** | Determina los datos a interpolar a partir de la traza seleccionada, posteriormente mediante el método Dos() de la clase Util llama al octave para el cálculo de la interpolación el cual guarda sus resultados en el archivo spl.txt ubicado en la carpeta Debug del proyecto, este archivo luego se lee para cargar en memoria los datos de la interpolación en el vector de enteros spl[], y en el vector timspl se guardan los tiempos de los datos de la interpolados. |
| **Métodos que llama** | Dos() (Util)  Mensaje() (Util) //Dibuja un mensaje en pantalla. |

## GrabaAscii()

|  |  |
| --- | --- |
| **Nombre** | GrabaAscii |
| **Descripción** | Graba en la base los datos de la interpolación en formato ascii. |
| **Parámetros** | string nom: nombre con el que se desea guardar el archivo.  bool cond: Se utiliza para verificar si además de guardar los datos de la interpolación de velocidad de cuentas también se guardan los datos de desplazamiento de cuentas, esto último en caso de que cond sea true. |
| **Postcondición** | Almacena los datos de la interpolación en archivos y los guarda en la carpeta asc. |
| **Proceso** | Crea un archivo con el nombre suministrado como parámetro en el método en la carpeta asc y en el guarda los datos de la interpolación, en caso de que la variable con sea false guarda los datos que están en la variable spl[], en caso de que sea true guarda los datos de la variable dzp[] |

## GrabaSudsInterpol()

|  |  |
| --- | --- |
| **Nombre** | GrabaSudsInterpol |
| **Descripción** | Graba en la base los datos de la interpolación en formato suds extensión .dmx. |
| **Parámetros** | string nom: nombre con el que se desea guardar el archivo.  bool cond: Se utiliza para verificar si además de guardar los datos de la interpolación de velocidad de cuentas también se guardan los datos de desplazamiento de cuentas, esto último en caso de que cond sea true. |
| **Postcondición** | Almacena los datos de la interpolación en archivos y los guarda en la carpeta sud. |
| **Proceso** | Crea un archivo con el nombre suministrado como parámetro en el método en la carpeta sud y en el guarda los datos de la interpolación, en caso de que la variable con sea false guarda los datos que están en la variable spl[], en caso de que sea true guarda los datos de la variable dzp[] |

## GrabaSeisan()

|  |  |
| --- | --- |
| **Nombre** | GrabaSeisan |
| **Descripción** | Graba en la base los datos de la interpolación en formato seisan. |
| **Parámetros** | string nom: nombre con el que se desea guardar el archivo.  bool cond: Se utiliza para verificar si además de guardar los datos de la interpolación de velocidad de cuentas también se guardan los datos de desplazamiento de cuentas, esto último en caso de que cond sea true. |
| **Postcondición** | Almacena los datos de la interpolación en archivos y los guarda en la carpeta sei. |
| **Proceso** | Crea un archivo con el nombre suministrado como parámetro en el método en la carpeta sei y en el guarda los datos de la interpolación, en caso de que la variable con sea false guarda los datos que están en la variable spl[], en caso de que sea true guarda los datos de la variable dzp[] |

# Métodos de eventos gráficos Form1

## cajon\_CheckedChanged()

|  |  |
| --- | --- |
| **Nombre** | cajon\_CheckedChanged |
| **Descripción** | Marca o desmarca los checkBox que representan las tarjetas, y modifica el valor booleano del arreglo que representa la tarjeta específica del checkBox. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondiciones** | En el panel de tarjetas se hace visible el cajón marcado o desmarcado en caso de que el click fuera para este fin, además el arreglo que controla los valores de verdad de los cajones cambia en el índice del cajón modificado dependiendo el valor de verdad anterior al click. |
| **Proceso** | Cuando se genera un evento de click sobre alguno de los checkBox que representa las tarjetas se revisa que chekBox en específico fue el que se escogió y dependiendo si estaba o no marcado antes del click lo marca o desmarca, además modifica el valor de verdad de ese chekBox en el arreglo que controla los cajones para cada formato de traza (MUX, DMUX GCF, YFILE o SEISAN).  Ejemplo: En el panel de tarjetas se tiene desmarcado el checkBox que representa la tarjeta MUX1, se da click en este checkBox y visualmente cambia de estado, en el código el arreglo cajmux[0] será = true. |

## cajon\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | cajon\_MouseDown |
| **Descripción** | Cuando de hace click derecho sobre la variable cajón cambia el valor boooleano que tenga asignado la variable CajonGcf. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondiciones** | Si el valor de CajonGcf es true lo cambia a false y viceversa. |

## cajon\_MouseMove()

|  |  |
| --- | --- |
| **Nombre** | cajon\_MouseMove |
| **Descripción** | Cambia el valor del vector booleano cajgcf en una posición dependiente de la casilla donde se realizó el movimiento del mouse y el estado de verdad de la variable CajonGcf. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondiciones** | Cambia el valor booleano del vector cajgcf en una posición especifica |

## boGcfTar\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | boGcfTar\_MouseDown |
| **Descripción** | Controla el botón inferior que permite deseleccionar o seleccionar estaciones en formato GCF. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondiciones** | Activa o desactiva la función que permite seleccionar estaciones en formato GCF |

## bMas\_Click()

|  |  |
| --- | --- |
| **Nombre** | bMas\_Click |
| **Descripción** | Este botón permite visualizar o esconder los botones de volcanes cuando su número sea mayor a 14. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Si el objeto panel3 tiene el atributo Visible como false lo cambia a true, en el caso de que el atributo Visible sea true lo cambia a false y llama al método DibujoTrazas(). |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1). |

## bvol\_Click()

|  |  |
| --- | --- |
| **Nombre** | bvol\_Click |
| **Descripción** | Comprueba si se ha seleccionado un volcán y |e cambia el color al botón especifico de ese volcán por amarillo. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondiciones** | Cambia el color del botón del volcán. |
| **Proceso** | Se recorre el vector de botones hasta encontrar el botón especifico y es a ese que se le cambia el color del background a amarillo. |

## bcla\_Click()

|  |  |
| --- | --- |
| **Nombre** | bcla\_Click |
| **Descripción** | Comprueba si se ha seleccionado una clasificación y llama a la rutina que graba los datos. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Llama el método que graba los datos de la clasificación GrabarBase después hace invisibles los paneles usados para la clasificación del sismo y queda visible el panel donde se ven todos los sismos; si GrabarBase retorna 1 hace visibles los botones boHypo71 y boAten. |
| **Métodos que llama** | * GrabaBase() (Form1) |

## best\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | best\_MouseDown |
| **Descripción** | Comprueba la activación de los botones de cajones asociados con las estaciones y volcanes. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Si ***totestaloc*** es menor o igual a 0 sale del método.  Se asegura que si está presente el código del tiempo “IRG”, se tenga en cuenta para ser dibujado en el panel junto con las estaciones, paso seguido verifica que dependiendo el botón seleccionado se asocien las estaciones correspondientes a ese volcán para posteriormente dibujarlas, además de cambiar el color al botón del volcán seleccionado haciendo que este sea amarillo y dibuja las trazas nuevamente.  Si no están todas las estaciones presentes (seleccionadas), se le avisa al usuario, pintando el botón TODAS, en color rosado, y por ultimo pinta de color rojo el botón que se seleccionó. |
| **Postcondiciones** | Dibuja solo las estaciones asociadas al volcán que se escogió.  Cambia el color del botón del volcán a amarillo.  Pinta el botón al que se le dio click de color rojo. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1). * EscribePanelEsta() (Util) * DibujoClascoda() (Form1) |

## listBox1\_SelectedIndexChanged()

|  |  |
| --- | --- |
| **Nombre** | listBox1\_SelectedIndexChanged |
| **Descripción** | ListBox1, contiene las fechas por minuto, que el usuario puede seleccionar para visualizar la traza; este método se lanza cuando ocurre un cambio de selección en uno de los ítems del listBox1. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Revisa por cada tipo de tarjeta (mux, dmux y gcf) los vectores booleanos ***yamux[],yadmx[], y yagcf[],*** en un ciclo for para cada uno y hace false todos sus valores; después lanza el método SeleccionarMinuto(). |
| **Métodos que llama** | * [SeleccionarMinuto() este método necesito que me lo explique Alvaro Pablo](#_SeleccionarMinuto(bool_cond)_este) (Form1) |

## listBox2\_SelectedIndexChanged()

|  |  |
| --- | --- |
| **Nombre** | listBox2\_SelectedIndexChanged |
| **Descripción** | ListBox2, contiene las estaciones, este método se lanza cuando ocurre un cambio de selección en uno de los ítems del listBox2 y esta encargado de configurar las variables necesarias para la clasificación de una traza al cambiar de estación. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Empieza ocultando los paneles panelParti, panelFFTzoom, si la variable siNeic es true muestra el botón boNeic.  Inicializa todos los campos del arreglo siEst en true.  Si nuhueco es > 0 limpia la lista huecolist y hace nuhueco = 0.  Verifica si esta activa la lectura de tremor continua en varios archivos mediante la variable seguir en caso de ser false pinta el botón boTremor de color blanco y hace false la variable tremor, y a tinitremor=0.  Verifica si la estación tiene un valor de UMBRAL registrado en caso de tenerlo hace visible el botón boUmbral que al activarlo muestra en color rojo sobre traza, el sector que sobrepasa el valor de Umbral. |
| **Métodos que llama** | * TrazasClas() (Form1). * [volcanTarro()](#_volcanTarro()) (Form1). |

## listBox2\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | listBox2\_MouseDown |
| **Descripción** | Se lanza cuando ocurre un evento de click derecho sobre el listBox; Con esta rutina se puede mostrar en pantalla otra estación adicional en un panel inferior. En este caso se puede filtrar la señal sin utilizar el Octave. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Precondiciones** | Para que funcione es necesario activar el nombre de la Estación con el botón derecho del ratón. |
| **Proceso** | Si el evento de click se generó con la tecla derecha del mouse configura los tamaños de borde, posición, y tamaño del panel ***panel1a*** y el tamaño del panel ***panel1***  y dibuja en pantalla el panel ***panel1a.*** |

## panelmarca\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | panelmarca\_MouseDown |
| **Descripción** | Panel donde aparecen en 8 columnas, los caracteres que sirven de marca a los sismos, marca es la variable que guarda los 8 caracteres de marcación de un sismo. Cuando el sismo no está marcado, todos los caracteres son asteriscos. |
| **Parámetros** | Object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Precondiciones** | El panel marca debe estar visible en pantalla. |
| **Métodos que llama** | * VerMarca() (Util). |

## boclaizq\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | boclaizq\_MouseDown |
| **Descripción** | Desplaza el gráfico de las trazas en la ventana de clasificación hacia la izquierda. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1) * EscribePanelEsta() (Util) |

## boclader\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | boclader\_MouseDown |
| **Descripción** | Desplaza el gráfico de las trazas en la ventana de clasificación hacia la derecha. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1) * EscribePanelEsta() (Util) |

## bocladil\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | bocladil\_MouseDown |
| **Descripción** | Disminuye el tiempo en Clasificar de las trazas o si se quiere 'estira' la señal. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1). * EscribePanelEsta() (Util) |

## boclaenc\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | boclaenc\_MouseDown |
| **Descripción** | Aumenta el tiempo en Clasificar de las trazas o si se quiere 'comprime' la señal. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1). * EscribePanelEsta() (Util) |

## boTodas\_Click()

|  |  |
| --- | --- |
| **Nombre** | boTodas\_Click |
| **Descripción** | Selecciona todas las estaciones para clasificar el sismo. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Asigna todas las posiciones del vector siEst[] como true, luego cambia el color de todos los botones best[] por color Peru (marron), por ultimo cambia a blanco el botón boTodas y lanza el método DibujoTrazas() que pinta de nuevo todas las estaciones. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1). * DibujoClascoda() (Form1). |

## boEsta\_Click()

|  |  |
| --- | --- |
| **Nombre** | boEsta\_Click |
| **Descripción** | Despliega o osconde el panel donde se escojen las trazas activas para la clasificación del sismo. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Si el panel panelEsta esta invisible lo hace visible y muestra que eestaciones hay seleccionadas en caso de que alguna no este seleccionada pinta el botón boTodas de color rojo, si el panel panelEsta esta visible lo hace invisible y dibuja las trazas. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1). * DibujoClascoda()(Form1) |

## panelEsta\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | panelEsta\_MouseDown |
| **Descripción** | Selecciona una o todas las trazas del paneEsta dependiendo si el click que se dio sobre el panel fue izquierdo o derecho. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Verifica que con que botón del maouse si hizo el click en caso de ser el derecho solo deja seleccionada la traza donde se hizo click esta aparecerá de color negro y las demás de color rojo, y esa traza es la única que se dibuja en el panel de clasificación del sismo, en caso de ser click izquierdo cambia el color de la traza seleccionada a rojo indicando que no se tienen en cuenta en la clasificación y no la dibuja en el panel. |
| **Métodos que llama** | * [DibujoTrazas()](#_DibujoTrazas()) (Form1). * EscribePanelEsta() (Util) |

## bout\_Click()

|  |  |
| --- | --- |
| **Nombre** | bout\_Click |
| **Descripción** | Rutina que inicializa variables, cuando se cierra el panel de clasificación. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Cuando se registra el evento que cierra el panel de clasificación se modifica las variables de visibilidad de los paneles y botones utilizados para la clasificación de sismos, de esta manera oculta los paneles. |

## bosuiz\_Click()

|  |  |
| --- | --- |
| **Nombre** | bosuiz\_Click |
| **Descripción** | Ubica el panel de clasificación en la parte superior izquierda de la pantalla. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Cambia los valores de localización del panel ***panelcla*** de tal forma que este queda ubicado en la esquina superior izquierda de la pantalla. |

## bozoom\_Click()

|  |  |
| --- | --- |
| **Nombre** | bozoom\_Click |
| **Descripción** | Rutina que amplía el panel de clasificación a toda la pantalla. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Verifica si el tamaño del panel de clasificación es menor al tamaño de la ventana en caso de ser menor cambia la posición del ***panelcla*** a la esquina superior izquierda y asigna las dimensiones de la pantalla principal al ***panelcla.*** |
| **Métodos que llama** | * [Clasificar()](#_Clasificar()) (Form1). * EscribePanelEsta(). |

## bosude\_Click()

|  |  |
| --- | --- |
| **Nombre** | bosude\_Click |
| **Descripción** | Ubica el panel de clasificación en la parte superior derecha de la pantalla. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Cambia los valores de localización del panel ***panelcla*** de tal forma que este queda ubicado en la esquina superior derecha de la pantalla. |

## boinde\_Click()

|  |  |
| --- | --- |
| **Nombre** | boinde\_Click |
| **Descripción** | Coloca el panel de clasificación en la parte inferior derecha de la pantalla |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Cambia los valores de localización del panel ***panelcla*** de tal forma que este queda ubicado en la esquina inferior derecha de la pantalla. |

## boiniz\_Click()

|  |  |
| --- | --- |
| **Nombre** | boiniz\_Click |
| **Descripción** | Coloca el panel de clasificación en la parte inferior izquierda de la pantalla |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Cambia los valores de localización del panel ***panelcla*** de tal forma que este queda ubicado en la esquina inferior izquierda de la pantalla. |

## boref\_Click()

|  |  |
| --- | --- |
| **Nombre** | boref\_Click |
| **Descripción** | Cambia el valor booleano de la variable ***refe*** si es true a false y viceversa. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Métodos que llama** | BotRefe() (Form1) // Colorea los botones que escogen entre referencia o no, para que el usuario sepa en que estado se encuentran. |

## botim\_Click()

|  |  |
| --- | --- |
| **Nombre** | botim\_Click |
| **Descripción** | Oculta o hace visibles las marcas de tiempo en las trazas. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Si la variable booleana ***marcati*** es true la hace false y cambia el color del botón ***botim***  a blanco; en caso de que ***marcati*** sea false la hace = a true y cambia el color de ***botim***  a naranja y llama el método MarcaTiempo()de la clase Util. |
| **Métodos que llama** | MarcaTiempo() (Util). // Calcula a que distancia se deben pintar las líneas zapotes que se pintan como marcas de tiempo y pinta las lineas. |

## bopep\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | bopep\_MouseDown |
| **Descripción** | Oculta o hace visibles las pepas que indican donde se clasificó un sismo. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Verifica si el evento de click que lanzó el botón ***bopep*** fue lanzado con el click izquierdo del mouse en caso de ser así si la variable ***pepas***  es true la hace false esto para ocultar las pepas y hace de color blanco al botón ***bopep,***  pero si ***pepas*** es false la hace true y cambia el color de ***bopep*** a amarillo verdoso.  En caso de que el evento fuera lanzado con el click derecho del mouse lanza el método ***LeerAmplitud()*** y al método ***PonePepas().*** |
| **Métodos que llama** | LeerAmplitud() (Form1)  PonePepas() sin terminar (Util). |

## boPepVol\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | boPepVol\_MouseDown |
| **Descripción** | Hace solo visibles las pepas de las trazas clasificadas que pertenezcan al volcán activo en el momento. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Revisa si la variable ***pepasvol*** es false, de ser así cambia su valor a true y el valor de la variable ***pepas*** a false, en caso de que ***pepasvol*** sea true la hace false y a ***pepas*** la hace true. |

## panelcladib\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | panelcladib\_MouseDown |
| **Descripción** | Método que se ejecuta cuando se da click con el mouse sobre el panel de clasificación para desplegar el panel donde se muestra el espectro de una porción de la traza. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondición** | Se despliega el panel donde se muestra el cálculo del espectro de una porción de la traza. |
| **Proceso** | Si se da click con el botón izquierdo del mouse y la variable ***VerEspectro*** es true se realiza el cálculo del espectro mediante el método ***CalcularEspectroCla()*** y se despliega el panel donde muestra el espectro. |
| **Métodos que llama** | CalcularEspectroCla() (Form1) |

## panelcladib\_MouseUp()

|  |  |
| --- | --- |
| **Nombre** | panelcladib\_MouseUp |
| **Descripción** | Se busca saber si se ha arrastrado el mouse en el panel de clasificación, si se hace con el botón derecho (información sobre el tiempo o duración) o izquierdo (entra en el panel de coda). |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | En caso de que haya dibujada un línea guía en el panel1 o panel1a (paneles principales); además verifica que estaciones están activas.  Si el click se dio con el botón derecho formatea una cadena para que indique el intervalo de tiempo al que es equivalente la porción que se arrastró.  Si el click se da con el botón izquierdo configura las variables necesarias para la visualización del panelcoda y lo despliega; si la variable ***moverparti*** es true calcula el movimiento de partículas. |
| **Métodos que llama** | MovimientoParticula() (Form1)  DibujoTrazas() (Form1)  BuscaCompCoda() (Form1)  TrazasClas() (Form1)  ChequeoFactormm() (Form1) |

## panelcoda\_MouseUp()

|  |  |
| --- | --- |
| **Nombre** | panelcoda\_MouseUp |
| **Descripción** | Busca saber si se arrastra o no el mouse en el panel de coda. Si se arrastra el mouse se entra en el panel para la lectura de la amplitud y frecuencia. Si no se arrastra y se pulsa el botón izquierdo, se hace las lecturas secuenciales de P, S y Coda. Si es con el botón derecho, siempre se lee la coda. Si el botón HUECO existe y está activado, las lecturas corresponden a los huecos para los cálculos de energía, de acuerdo al Observatorio de Pasto. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Este método aún no está completamente documentado |
| **Métodos que llama** | BotonesCoda() (Form1) // rutina que muestra el botón activo, esta descrita en la documentación del código. |

## panel1\_MouseUp() falta vincular

|  |  |
| --- | --- |
| **Nombre** | panel1\_MouseUp |
| **Descripción** | (panel1) Aquí se busca saber si hay arrastre del mouse sobre la traza. Si se activa el botón izquierdo y se desplaza el mouse, se entra al panel de clasificación (si no se arrastra, no se hace nada por el momento). Con el botón derecho, se indica el tiempo, absoluto si no hay arrastre o el intervalo correspondiente al arrastre. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Está pendiente una revisión minuciosa de su funcionamiento. |
| **Métodos que llama** | PromedioFiltrado() (Form1)  Cuadro\_Tremor() (Form1)  CalculoInterpolacion() (Form1)  YaClasificados() (Form1)  Clasificar() (Form1)  BuscaCompCoda() (Form1)  Crea una instancia del Form2 y lo despliega.  Mensaje() (Util)  EliminaClasificacion() (Util)  Leerbase() (Util)  LeerAmplitud() (Form1)  Reviarch() (Form1) |

## panel1a\_MouseUp()

|  |  |
| --- | --- |
| **Nombre** | panel1a\_MouseUp |
| **Descripción** | (panel1a) Aquí se busca saber si hay arrastre del mouse sobre la traza. Si se activa el botón izquierdo y se desplaza el mouse, se entra al panel de clasificación (si no se arrastra, no se hace nada por el momento). Con el botón derecho, se indica el tiempo, absoluto si no hay arrastre o el intervalo correspondiente al arrastre. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Determina el punto donde se soltó el mouse y la posición en el vector de tiempo tim[ida][] del último dato de tiempo de la traza activa en el panel1a.  Configura el espaciamiento entre líneas para dibujar la traza.  Determina el valor del momento en el que se inició el arrastre del mouse y el del momento en que se terminó el arrastre, guarda estos valores en ***tie1*** y ***tie2*** respectivamente.  Si el valor de sipro es 1 determina el periodo1 y periodo2 del sector de traza comprendido entre ***tie1*** y ***tie2.***  Si el tie1 es diferente al tie2  Verifica que en la base de datos se encuentre el archivo ***datos.txt*** y de encontrarlo lanza el método.  En caso de que el delta de tiempo seleccionado para la clasificación sea menor de 10 (esto indica que se quiere clasificar solo 10 segundos) se despliega un mensaje que avisa que no se guardan clasificaciones menores de 10 segundos.  Guarda en el vector Cladat[] de la clase Point en la posición 0 los valores respectivos a el punto donde se inició el arrastre de la traza, y en la posición 1 guarda los valores del punto donde se terminó el arrastre.  Finaliza llamando el método Clasificar().  En caso de que tie1 y tie2 sean iguales grafica una línea zapote en el panel1a y en el panel1 que indica el instante de tiempo donde tie1 y tie2 son iguales, igualmente grafica una línea roja en cada panel para mostrar en que traza se esta clasificando. |
| **Métodos que llama** | YaClasificados() (Form1)  Clasificar() (Form1) |

## boHypo71\_Click()

|  |  |
| --- | --- |
| **Nombre** | boHypo71\_Click |
| **Descripción** | Hace llamado al formulario2, el cual corresponde a las lecturas de arribos para la localización con el Hypo71. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Crea un objeto de tipo Form2 y lo despliega en pantalla. |
| **Métodos que llama** | Form2(Form1 ventana) (Form2) |

## panelAmp\_MouseUp() falta vincular

|  |  |
| --- | --- |
| **Nombre** | panelAmp\_MouseUp |
| **Descripción** | Fin del intervalo de lectura del panel de amplitud. Dicho intervalo equivale al Periodo. Se busca además el valor máximo de cuentas pico a pico. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Determina los valores de tiempo en donde se inició el arrastre como en donde se terminó, seguido verifica que la lectura de la amplitud este en los parámetros adecuados (que no empiece antes que donde se ubicó la lectura de P ni después de la lectura de Coda), ahora busca en la matriz **tim** las posiciones de los valores de tiempo antes determinados **ti1** y **ti2** y losasigna a variables locales para determinar el intervalo de tiempo seleccionado.  Por ultimo calcula **ampp** y llama los métodosCalculoInterpolacion() y IntegracionSpl(). |
| **Métodos que llama** | CalculoInterpolacion() (Form1)  IntegracionSpl() (Form1) |

## boPSW\_Click()

|  |  |
| --- | --- |
| **Nombre** | boPSW\_Click |
| **Descripción** | Hace llamado al programa PSW del sismologo Jaime Raigosa.  Graba la última lectura en formato Demultiplexado en el archivo psw.dmx y la abre en el PSW. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondición** | Genera el archivo psw.dmx y lo abre en el programa PSW. |
| **Proceso** | Resisa si el archivo ***psw.dmx*** exite, en caso afirmativo lo elimina.  Posteriormente llama al método GrabaSuds(1) con lo que genera el archivo psw.dmx y llama al método Dos() con el que lanza el PSW, al final repinta la trazas mediante DibujoTrazas(). |
| **Métodos que llama** | GrabaSuds() (Form1)  Dos() (Util)  DibujoTrazas() (Form1) |

## boScilab\_Click()

|  |  |
| --- | --- |
| **Nombre** | boScilab\_Click |
| **Descripción** | Guarda la traza actual en el archivo demux.dmx y lo abre en el programa libre SCILAB. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondición** | Genera el archivo demux.dmx y lo abre en el programa SCILAB. |
| **Proceso** | Resisa si el archivo ***psw.dmx*** exite, en caso afirmativo lo elimina.  Posteriormente llama al método GrabaSuds(2) con lo que genera el archivo demux.dmx.  Guarda en el archivo fecha.txt la fecha del primer registro que se tenga en la matriz tim[id][0] de la estación activa.  Mediante el método Dos() abre el SCILAB y al final repinta la trazas mediante DibujoTrazas(). |
| **Métodos que llama** | GrabaSuds() (Form1)  Dos() (Util)  DibujoTrazas() (Form1) |

## boClaSola\_Click()

|  |  |
| --- | --- |
| **Nombre** | boClaSola\_Click |
| **Descripción** | Botón que aparece en la parte inferior derecha del panel de Amplitud y al activarlo, graba los datos de lecturas en la Base. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Métodos que llama** | GrabaBase() (Form1)  Seguir() (Form1)  Cuadro\_Tremor() (Form1) |

## boStartMpt\_Click()

|  |  |
| --- | --- |
| **Nombre** | boStartMpt\_Click |
| **Descripción** | Es el que marca el inicio del cálculo de movimiento de partículas y desencadena la animación del mismo. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Métodos que llama** | VerMapa() (Form1)  TrazaComponente() (Form1)  TrazaComponenteInterp() (Form1) |

## boNeic\_MouseDown()

|  |  |
| --- | --- |
| **Nombre** | boNeic\_MouseDown |
| **Descripción** | Este botón hace visible el panelMapaMundo en caso de que haya estaciones NEIC (que exista el archivo /pro/tab/estaciones.dat) y abre en el navegador web predeterminado la página http://earthquake.usgs.gov/earthquakes/recenteqsww/Quakes/quakes\_all.php. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Precondición** | La variable nomWeb debe estar configurada con la url http://earthquake.usgs.gov/earthquakes/recenteqsww/Quakes/quakes\_all.php. |
| **Postcondición** | Despliega el panelMapaMundo. |
| **Métodos que llama** | RevisarEstacionNeic() (Form1). |

## textBoxNeic\_TextChanged()

|  |  |
| --- | --- |
| **Nombre** | textBoxNeic\_TextChanged |
| **Descripción** | El texbox textBoxNeic es el que aparece en el panelMapaMundo y es donde se ingresa los datos del sismo distante, con este dato se determina el mapa a dibujar en el panelMapaMundo, con base a los datos de la página http://earthquake.usgs.gov/earthquakes/recenteqsww/Quakes/quakes\_all.php. |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Proceso** | Está pendiente una revisión de tallada de su funcionamiento. |
| **Métodos que llama** | CrearCajonesTeoricos() (Form1) ///Crea los checkbox que representan los archivos.txt alojados en la carpeta .\bin\Debug\tab, se usan para presentar los tipos de arribo a ese sismo.  ChequearCajones() (Form1). /// Asigna los colores del texto y fondo de los checkBox cbx, además de virificar cuales están checked = true.  PonerTeorico() (Form1). ///Se lanza cuando se cambia de estado a cualquiera de los checkBox cbx, y determina el checkbox especifico que tuvo este evento, si el inidice de dicho checkBox es diferente de -1 llama al método PonerTeorico(). |

## boGraInterpol\_Click()

|  |  |
| --- | --- |
| **Nombre** | boGraInterpol\_Click |
| **Descripción** | Llama los métodos para grabar la interpolación en la base en cada formato diferente (Seisan, ascii, Suds). |
| **Parámetros** | object sender: objeto que lanza el evento.  EventArgs e: evento. |
| **Postcondición** | Guarda los datos de la interpolación de la traza seleccionada en un archivo. |
| **Proceso** | Determina la el momento desde el que se va a guardar la interpolación, y para cada formato diferente verifica que la variable booleana respectiva asociada a dicho formato sea true, en caso de ser true guarda los datos de la interpolación para la velocidad de cuentas, y en caso de que se haya clasificado el desplazamiento de cuentas también guarda dichos datos. |
| **Métodos que llama** | GrabaSeisan() (Util)  GrabaAscii() (Util)  GrabaSudsInterpol() (Util) |

# Util

## Dos()

|  |  |
| --- | --- |
| **Nombre** | Dos |
| **Descripción** | Crea un proceso cmd para ejecutar la línea que inicia el Octave. |
| **Parámetros** | string strCmdLine: línea a ejecutar en la consola cmd.  bool cond: valor que se utiliza para decidir si se despliega la ventana del proceso creado, true si no quiere mostrar la ventana, false si quiere desplegar la ventana. |
| **Precondición** | El sistema Octave debe estar instalado. |
| **Proceso** | Crea un proceso mediante el cual hace llamado al cmd en el que ejecuta la línea de comando parámetro y establece el período de tiempo que se debe esperar hasta que termine el proceso asociado y bloquea el subproceso que se está ejecutando en ese momento hasta que el tiempo de espera se haya agotado o el proceso haya terminado. |

## Leerbase()

|  |  |
| --- | --- |
| **Nombre** | Leerbase |
| **Descripción** | Revisa si a determinado minuto corresponde sismos clasificados y si tienen lectura de amplitud y de coda, en otras palabras chequea y cuenta los minutos que correspondan a sismos clasificados en la base, escribe en los archivos datos.txt y amplis.txt los datos respectivos en caso de que existan sismos clasificados. |
| **Parámetros** | Panel panel: panel utilizado para informar que se está leyendo la base.  string rutbas: cadena que especifica la ruta de la base de datos.  long ll1: tiempo en formato C# del momento desde donde se deben mostrar las trazas (fecha inicial).  long ll2: tiempo en formato C# del momento hasta donde se deben mostrar las trazas (fecha final).  string[] cl: contiene el tipo de clasificaciones que hay guardados para los volcanes, VT, LP, TL etc.  string[] volcán: contiene los volcanes que se encuentran en la base, RUIZ, MACHIN etc. |
| **Retorna** | int contados: la cantidad de sismos que se encuentran clasificados. |
| **Proceso** | Verifica la existencia de los archivos leedatos2.txt, leedatos.txt, amplis2.txt y amplis.txt, en caso de encontrar alguno de estos archivos lo borra.  Inicializa un objeto StreamWriter ***le***  para escribir en el archivo amplis2.txt.  Después en un ciclo for que va desde la fecha inicial ***ll1*** + 864000000000 (corresponde a un día en centenares de nanosegundos, que es la unidad de tiempo que maneja la variable ticks del visual C#) hasta la fecha final ***ll2***, inicializa un objeto DateTime ***fech*** pasando como parámetro ***ll1*** y dándole formato YY:MM:DD, paso siguiente en un for anidado que va desde 0 hasta el número de clasificaciones en ***cl[]*** verifica el tercer carácter del arreglo ***cl,* SI** este es un + en otro for anidado que va desde 0 hasta el tamaño del arreglo ***volcan*** -1 inicializa un string con la ruta tentativa que contendría el archivo a leer (D:\\base\\lec\\VT\\RT140318.txt) esta se construye con la ruta de la base y en ella la carpeta lec, la clasificación del sismo y el archivo que se construye así:   * La primera letra es la inicial del volcán. * La segunda letra es el segundo carácter de la clasificación del volcán. * Por último la fecha.   Ahora si dicho archivo existe lo copia en un archivo temporal llamado ***cla.txt*** el cual lee en un while y va guardando cada línea leída en un string que si su carácter 82 es = “.” (Punto) escribe en el objeto ***le*** la línea que leyó y cuenta el número de amplitudes registradas y lo va agregando a la variable ***contampl***; esto lo hace para cada archivo en las carpetas de clasificación en la carpeta lec de la base.  Ahora en caso de que el tercer carácter del arreglo ***cl*** **NO** sea + solo cambia la forma de construir la ruta del archivo asi:   * La primera letra es una X. * La segunda letra es el segundo carácter de la clasificación del volcán. * Por último la fecha.   Si la variable ***contampl*** > 0 hace un llamado a el método Dos(string,bool).  Luego nuevamente en un for desde ***ll1*** + 864000000000 hasta ***ll2*** encierra un while en el que quenta cuantos sismos hay clasificados y los guarda en el archivo datos2.txt, si este número > llama al método Dos(string,bool) y termina retornando el número de sismos clasificados. |
| **Métodos que llama** | * Dos()(Util). |

## Letreros()

|  |  |
| --- | --- |
| **Nombre** | Letreros |
| **Descripción** | Pinta el nombre de usuario que está trabajando en la clasificación de trazas. |
| **Parámetros** | Panel panel: panel donde se dibuja el letrero.  int x : coordenada x del panel.  int y: coordenada y del panel.  string let: texto que se dibuja en el panel.  Color col: color del texto que se dibuja. |
| **Proceso** | Dibuja en el panel que se pasa como parámetro el string indicado. |

## borra()

|  |  |
| --- | --- |
| **Nombre** | borra |
| **Descripción** | Pinta el interior del panel del color que se pasa como parámetro. |
| **Parámetros** | Panel panel: panel a pintar.  Color col: Color con el que se desea rellenar el panel. |
| **Proceso** | Obtiene el objeto grafico del panel y los valores de ancho y alto del mismo, para luego rellenar el panel del color que se proporciona como parámetro. |

## VerArchi()

|  |  |
| --- | --- |
| **Nombre** | VerArchi |
| **Descripción** | Dibuja un rectángulo sin relleno sobre la traza en el intervalo de tiempo correspondiente a un sismo clasificado. |
| **Postcondiciones** | Dibuja sobre la traza rectángulos que indican el lapso de tiempo que se clasificó para ese sismo. |
| **Parámetros** | Panel panel: panel donde se dibuja el rectángulo.  double timin: tiempo mínimo del en que se registró una traza, o dicho de otra forma la lectura que empezó primero.  double[] tim: vector que almacena los valores de los tiempos de las cuentas de una traza especifica.  double[] tiar: tiempo inicial de la duración del archivo clasificado.  ushort[] duar: duración del archivo clasificado.  ushort esp: valor del espectro.  float dur: duración del tiempo del intervalo de traza sobre el cual se arrastró el mouse.  ushort contarch: almacena la cantidad de sismos clasificados en un lapso de tiempo seleccionado. |
| **Proceso** | Configura a variable lap[] del tipo Pen con tres colores diferentes para alternar el color de los rectángulos en caso de que se hayan clasificado sismos solapados para poder diferenciarlos, en un ciclo for que va desde 0 hasta la cantidad de sismos clasificados se configura el tamaño de cada rectángulo y se dibuja, cada vez que dibuja un rectángulo cambia el índice de la variable que apunta a el color almacenado en lap[], al finalizar libera los recursos utilizados en el método. |

## EscribePanelEsta()

|  |  |
| --- | --- |
| **Nombre** | EscribePanelEsta |
| **Descripción** | Dibuja los nombres de las estaciones en el panel panelEsta, si la estación esta activa en la clasificación que se está realizando esta se pinta de color negro, sino se pinta de color rojo para indicar que no se tiene en cuenta en la clasificación de ese sismo. |
| **Postcondiciones** | Se hace visible el panel panelEsta y se muestran los nombres de las estaciones, en color negro si esta activa o roja sino. |
| **Parámetros** | Panel panelEsta: panel donde se grafican los nombres de las estaciones activas, este panel está dentro del panel de clasificación.  ushort nutra: es la cantidad de trazas leídas.  string[] est: contiene los nombres de las estaciones.  bool[] siEst: indica que estaciones están seleccionadas como activas. |
| **Retorna** | -1 en caso de que todas las estaciones estén activas.  1 en caso de que por lo menos 1 de las estaciones no este activa. |
| **Proceso** | Mediante ciclos for recorre los vectores verifica si las estaciones están activas en el momento de la clasificación para ese sismo en específico, y en caso de estar activa dibuja su nombre en el panelEsta de color negro, sino lo dibuja de color rojo; si alguna de las estaciones no está activa retorna 1 pero en caso de que todas estén activas retorna -1. |

## VerMarca()

|  |  |
| --- | --- |
| **Nombre** | VarMarca |
| **Descripción** | Configura y dibuja el panel donde se puede generar la marca que se le asigna a un sismo para posteriores búsquedas. |
| **Postcondiciones** | Despliega el panel ***panelmarca.*** |
| **Parámetros** | Panel panel: panel donde se muestran los caracteres para generar la marca.  string marca: etiqueta que se le da al sismo para futuras búsquedas. |
| **Proceso** | Inicialmente dibuja el valor del parámetro marca en la parte superior del panel, luego para cada carácter dibuja debajo suyo los 43 caracteres con los cuales puede diseñar la marca propia para el sismo. |

## PonePepas()

|  |  |
| --- | --- |
| **Nombre** | PonePepas |
| **Descripción** | Dibuja las pepas que indican en donde se ha clasificado un sismo. |
| **Parámetros** | Panel panel: el panel donde se pintan las pepas.  double timin: tiempo mínimo del que se registró una traza.  double[] tim: representa el tiempo de cada valor de cuenta para una traza especifica.  ushort esp: es el espaciamiento entre líneas.  float dur: duración de tiempo del intervalo donde se arrastra el mouse.  ushort contampl: cantidad de lecturas de amplitud de la base.  double[] valampl:  byte[] clR: representa la intensidad del color rojo.  byte[] clG: representa la intensidad del color verde.  byte[] clB: representa la intensidad del color azul.  char[] letampl:  string[] cl: tipos de clasificaciones que se pueden asignar a un sismo.  float tam: tamaño de las pepas.  short nucla: cantidad de clasificaciones disponibles.  bool[] siPampl:  char[] volampl:  char voll: primera inicial del volcán.  bool cond: visualización de pepas en el volcán activo.  bool nolec: no hay lecturas en la base.  double tigrabacion:  char leclec:  int denom: guarda el número de líneas a grafiacar. |
| **Precondición** | Deben existir sismos clasificados en la base de datos. |
| **Postcondición** | Se dibujan las pepas en los lugares donde se ha clasificado sismos. |
| **Proceso** | Mediante la verificación conjunta de los valores de las variables:   * double[] valampl * bool[] siPampl * char[] letampl * char[] volampl   Determina en que punto de la traza ya se ha clasificado un sismo, que tipo de clasificación tiene el sismo en específico y de qué color debe graficar la pepa. |

# Contrato Base

|  |  |
| --- | --- |
| **Nombre** |  |
| **Descripción** |  |
| **Parámetros** |  |
| **Precondición** |  |
| **Postcondición** |  |
| **Proceso** |  |
| **Métodos que llama** |  |