

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA



**Desarrollo de Software Estadístico para la Escuela Profesional de
Ingeniería Estadística e Informática – Puno 2012**

TESIS

PRESENTADA POR:

Bach. JOEL RAMOS QUISPE

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO – PERÚ

2017

**UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO
FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA**

**Desarrollo de Software Estadístico para la Escuela Profesional de
Ingeniería Estadística e Informática – Puno 2012**

TESIS

PRESENTADA POR:

Bach. JOEL RAMOS QUISPE

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

APROBADA POR:

PRESIDENTE

: 
M.Sc. ALEJANDRO APAZA TARQUI

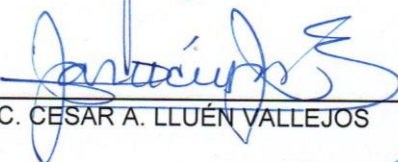
PRIMER MIEMBRO

: 
M.Sc. GODOFREDO QUISPE MAMANI

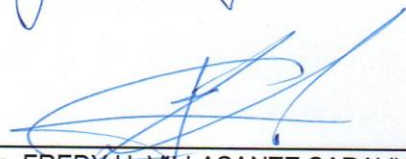
SEGUNDO MIEMBRO

: 
M.Sc. FRANCISCO CURRO PEREZ

DIRECTOR

: 
M.C. CESAR A. LLUÉN VALLEJOS

ASESOR

: 
M.Sc. FREDY H. VILLASANTE SARAVIA

ÁREA: Informática

TEMA: Ingeniería del software

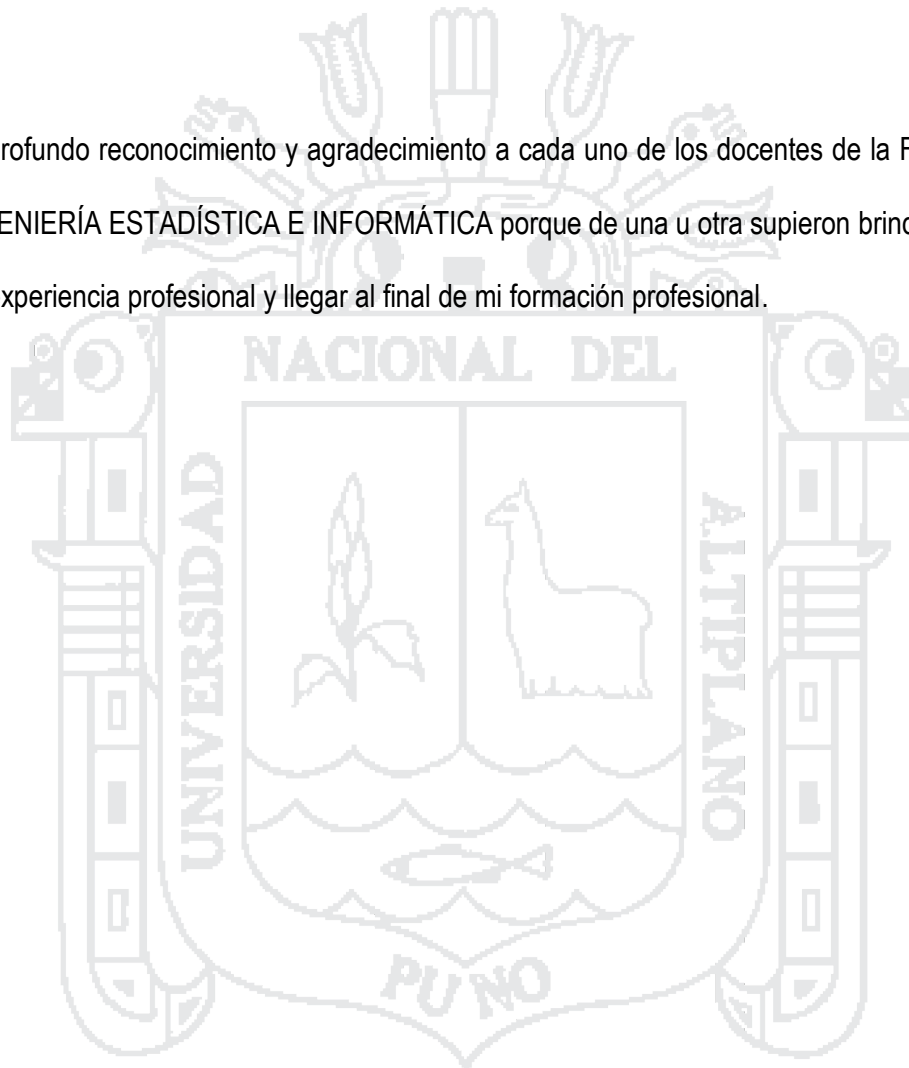
DEDICATORIAS



*A mis padres Julian y
Josefa por su apoyo
incondicional en todo momento.*

AGRADECIMIENTOS

Mi profundo reconocimiento y agradecimiento a cada uno de los docentes de la FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA porque de una u otra supieron brindarme su gama de experiencia profesional y llegar al final de mi formación profesional.



INDICE GENERAL

RESUMEN7
 ABSTRACT8
 INTRODUCCIÓN9

CAPÍTULO I

PROBLEMÁTICA DE INVESTIGACIÓN

1.1. PLANTEAMIENTO Y DEFINICIÓN DEL PROBLEMA.....12
 1.2. ANTECEDENTES DE LA INVESTIGACIÓN15
 1.3. OBJETIVO GENERAL.....17
 1.4. FORMULACIÓN DE HIPÓTESIS18

CAPÍTULO II

MARCO TEÓRICO

2.1. ANÁLISIS DESCRIPTIVO19
 2.2. BASE TEÓRICA49
 2.3. MARCO CONCEPTUAL.....51
 2.4. OPERACIONALIZACIÓN DE VARIABLES59

CAPÍTULO III

MATERIALES Y METODOS

3.1. POBLACIÓN.....60
 3.2. METODOLOGÍA.....60

CAPITULO IV

RESULTADOS Y DISCUSIONES

4.1. ALGORITMOS ESTADISTICOS68
 4.2. ESTRUCTURA DEL PROGRAMA FISTAD.....82

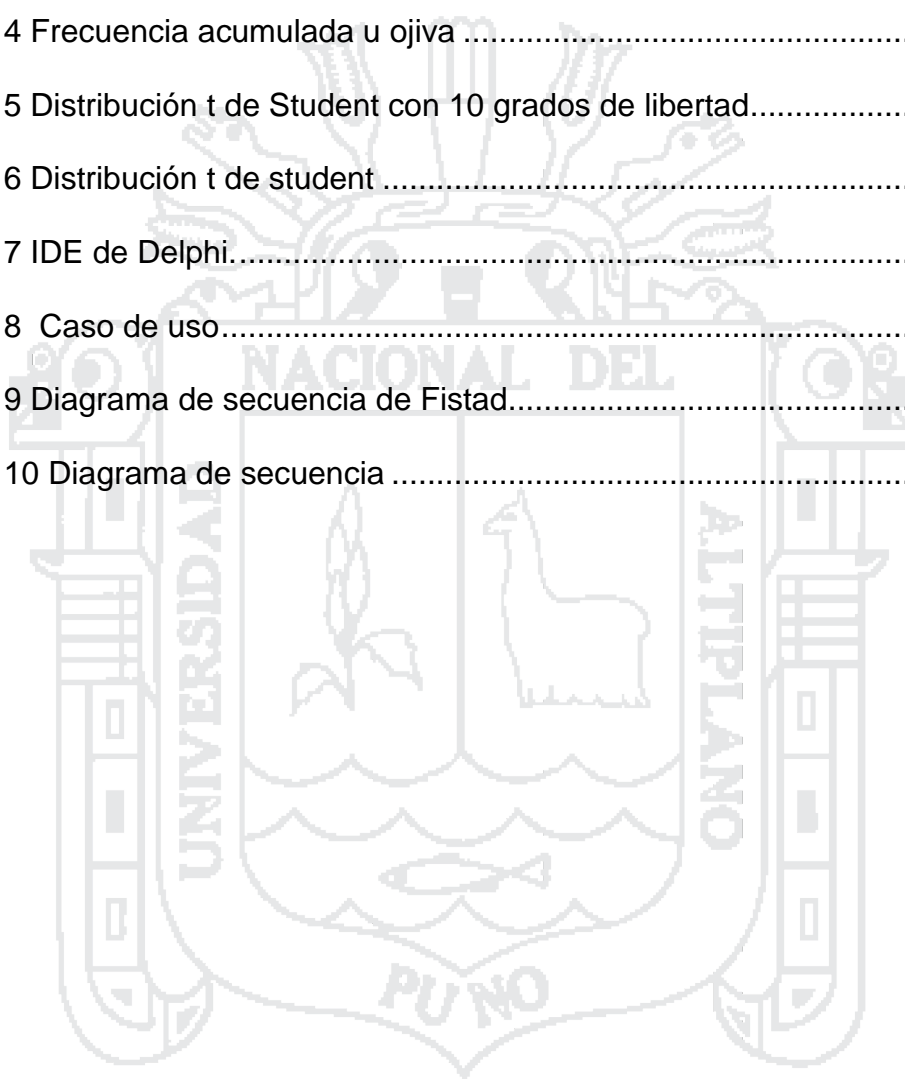
CONCLUSIONES91

RECOMENDACIONES.....92

BIBLIOGRAFÍA93

ÍNDICE DE FIGURAS

Fig. 1 Diagrama de Barras.....	23
Fig. 2 Histograma	24
Fig. 3 Histograma	24
Fig. 4 Frecuencia acumulada u ojiva	25
Fig. 5 Distribución t de Student con 10 grados de libertad.....	32
Fig. 6 Distribución t de student	33
Fig. 7 IDE de Delphi.....	48
Fig. 8 Caso de uso.....	66
Fig. 9 Diagrama de secuencia de Fistad.....	66
Fig. 10 Diagrama de secuencia	67



RESUMEN

Los estudiantes de Ingeniería Estadística e Informática se ven limitados en su quehacer académico usando software estadístico, estas herramientas se usan a nivel de usuario cuando la necesidad es entender algoritmos computacionales de las técnicas y métodos aplicados a los cursos de estadística; el presente trabajo de investigación pretende Desarrollar un software estadístico para el procesamiento de datos como guía de los alumnos, un software compacto con el código fuente entendible. Para llevar a cabo el desarrollo del software se hace uso del lenguaje UML, para modelamiento de propósito general usado para especificar, visualizar, construir y documentar los artefactos de un sistema de software. Los algoritmos estadísticos del presente trabajo cumplen la función de instruir a los usuarios en el manejo de calculos estadísticos, de los cuales el estudiante podrá tener una mejor idea al ver el algoritmo que ejecuta los procedimientos matemáticos necesarios para resolver los distintos postulados estadísticos, en el presente software se ha desarrollado algoritmos computacionales que procesan postulados de estadística básica en un lenguaje de programación popular y se ha integrado los algoritmos computacionales en un entorno de procesamiento de datos de forma similar a un software estadístico comercial, se presenta un editor básico de datos similar detallando su desarrollo.

Palabras Claves: Software Estadístico, procesamiento, datos, algoritmos, cálculos.

ABSTRACT

Students of statistical engineering and computer science are limited in their academic work using statistical software, these tools are used at the user level when the need is to understand computational algorithms of techniques and methods applied to statistics courses; The present research work aims to develop statistical software for data processing as a guide for students, a compact software with understandable source code. In order to carry out software development, UML is used for general purpose modeling used to specify, visualize, construct and document the artifacts of a software system. The statistical algorithms of the present work fulfill the function of instructing the users in the management of statistical calculations, from which the student will have a better idea when seeing the algorithm that executes the mathematical procedures necessary to solve the different statistical postulates, in the Present software has been developed computational algorithms that process basic statistics postulates in a popular programming language and has integrated computational algorithms in a data processing environment similar to commercial statistical software, a similar basic data editor is presented Detailing its development.

Keywords: Statistical software, processing, data, algorithms, calculations.

INTRODUCCIÓN

En la Escuela Profesional de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano es habitual que un estudiante se vea limitado en su quehacer académico por los programas como SPSS, Minitab, SAS, NCSS, Eviews, Stata, Statistica, Statgraphics, Einfo, etc. Estas herramientas se usan a nivel de usuario cuando la necesidad de muchos estudiantes es entender algoritmos computacionales con código fuente visible aplicados a los cursos de estadística. En las asignaturas de las ciencias estadísticas se estudian en su mayoría la teoría, conceptos, teoremas, funciones estadísticas, métodos y técnicas, cuya aplicación se hace mediante el uso de cálculos científicos de dicho análisis estadístico, en las asignaturas de informática se da una formación netamente centrada en el área, creando así un divorcio involuntario entre ambas especialidades, siendo necesario herramientas de software libre que facilite el estudio coordinado integrando las especialidades de estadística e informática.

En la actualidad las asignaturas de estadística se imparte base teórica, pero no se desarrollan algoritmos computacionales de su contenido en el área de informática ocurre de manera similar, se dictan cursos taller de programación centrada en algoritmos genéricos omitiendo algoritmos de estadística aplicada, cuando en la actualidad la estadística constituye técnicas y métodos de optimización, complejidad y eficiencia de los algoritmos computacionales, los cuales deben ser implementados en un software aplicado en las asignaturas para comprender en las técnicas y métodos, es importante generar algoritmos computacionales y mas adelante sea implementado el código fuente como parte del taller de las asignaturas de informática.

Para el estudio de algoritmos estadísticos existe en Internet código fuente limitado, superficial y disperso, que no permite ser analizado adecuadamente variando en metodología, lenguaje en el que fue escrito y la documentación de diverso tipo; por otro lado tenemos la existencia de código fuente de calidad en programas avanzados como R ó PSPP, cuyos comentarios no están en nuestro idioma, sus algoritmos son muy refinados siendo orientados para su aplicación en ambientes científicos de primer nivel lo que dificulta su entendimiento para un estudiante de pregrado, es decir están al alcance del entendimiento de muy pocos estudiantes de la escuela profesional con avanzado conocimiento de programación.

Considerando estos factores es necesario generar algoritmos computacionales sencillos y eficientes para cada una de las técnicas y métodos estadísticos estudiados en las aulas, de manera que estas sean entendidas y se puedan implementar en programas.

En el Capítulo I, Se realiza el planteamiento del problema en la Escuela Profesional de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano es habitual que un estudiante se vea limitado en su quehacer académico por los programas como SPSS, Minitab, SAS, NCSS, Eviews, Stata, Statistica, Statgraphics, Epiinfo, etc. Estas herramientas se usan a nivel de usuario cuando la necesidad de muchos estudiantes es entender algoritmos computacionales aplicados de los cursos de estadística, y su objetivo del presente proyecto fue Desarrollar un software estadístico para el procesamiento de datos guía de estudiantes de la Escuela Profesional de Ingeniería Estadística e Informática.

En el Capítulo II, Se realiza el marco teórico para el software estadístico para la escuela profesional de ingeniería estadística e informática – puno 2010” - fistad v 1.0

En el Capítulo III, se indica que materiales y métodos realizaron en el software estadístico guía de estudiante para la escuela profesional de ingeniería estadística e informática – puno 2010” - fistad v 1.0

En el Capítulo IV, se muestra los resultados y discusiones del software estadístico para la escuela profesional de ingeniería estadística e informática – puno 2010” - fistad v 1.0

En la parte final se llega a la conclusiones que los algoritmos estadísticos cumplen la función de instruir a los usuarios en el manejo de procesos estadísticos, de los cuales el estudiante podrá tener una mejor idea al ver el programa que ejecuta los procedimientos matemáticos necesarios para resolver los distintos postulados estadísticos.

CAPÍTULO I

PROBLEMÁTICA DE INVESTIGACIÓN

1.1. PLANTEAMIENTO Y DEFINICIÓN DEL PROBLEMA

1.1.1. Planteamiento del Problema

En la Escuela Profesional de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano es habitual que un estudiante se vea limitado en su quehacer académico usando software como SPSS, Minitab, SAS, NCSS, Eviews, Stata, Statistica, Statgraphics, Epiinfo, etc. Estas herramientas se usan a nivel de usuario cuando la necesidad de muchos estudiantes es entender algoritmos computacionales de las técnicas y métodos aplicados a los cursos de estadística. En las asignaturas de las ciencias estadísticas se estudian en su mayoría la teoría, conceptos, teoremas, funciones estadísticas, métodos y técnicas, cuya aplicación de dichos conocimientos se realiza en el campo de la aplicación de análisis estadístico, en las asignaturas de informática se da una formación netamente centrada en el área, creando así un divorcio involuntario entre

ambas especialidades, siendo necesario mecanismos que integren basados en software libre que facilite el estudio coordinado entre las especialidades de estadística e informática.

En la actualidad las asignaturas de estadística se imparte base teórica, pero no se desarrollan algoritmos computacionales de dicha base teórica como practica de su implementación en las asignaturas del área de informática ocurre de manera similar, se dictan cursos taller de programación centrada en algoritmos genéricos omitiendo algoritmos de estadística aplicada, cuando en la actualidad la estadística constituye técnicas y métodos de optimización, complejidad y eficiencia de los algoritmos computacionales, que muy bien deben ser implementados como software aplicado en las asignaturas para comprender la aplicabilidad de dichas técnicas y métodos, es importante generar algoritmos computacionales y mas adelante sea implementado en código fuente como parte del taller de las asignaturas de informática.

Para el estudio de algoritmos estadísticos existe en Internet código fuente limitado, superficial y disperso, que no permite ser analizado adecuadamente variando en metodología, lenguaje en el que fue escrito y la documentación de diverso tipo; por otro lado tenemos la existencia de código fuente de calidad en programas avanzados como R ó PSPP, cuyos comentarios no están en el idioma español, sus algoritmos son muy refinados siendo orientados para su aplicación en ambientes científicos de primer nivel lo que dificulta su entendimiento para un estudiante de

pregrado, es decir están al alcance del entendimiento de muy pocos estudiantes de la escuela profesional.

Considerando estos factores es necesario generar algoritmos computacionales sencillos y eficientes para cada una de las áreas de estadística estudiadas en las aulas de las diferentes asignaturas del área de estadística, de manera que estas sean implementadas y automatizadas en las asignaturas de informática, además pueda constituir la generación de software aplicado.

1.1.2. Definición del Problema

En la actualidad las asignaturas de estadística para comprender la aplicabilidad de dichas técnicas y métodos que se imparten, es importante trabajar de manera rápida con un software que facilite los procesos estadísticos.

Para el estudio de algoritmos estadísticos existe en Internet código fuente limitada, superficial y dispersa, que no permite ser analizado adecuadamente variando en metodología, lenguaje en el que fue escrito y la documentación de diverso tipo.

Considerando estos factores es necesario generar algoritmos computacionales sencillos y eficientes para cada una de las áreas de estadística estudiadas en las aulas de las diferentes asignaturas del área de estadística, de manera que estas sean implementadas y automatizadas en las asignaturas de informática, además pueda constituir la generación de software aplicado.

Por lo que se plantea siguiente pregunta: **¿De qué manera es posible desarrollar la integración de base teórica de la estadística en algoritmos computacionales coordinando áreas de Estadística e Informática como un software aplicado?**

1.2. ANTECEDENTES DE LA INVESTIGACIÓN

Jiménez, (2003) se ha pretendido impulsar el acercamiento de los profesionales e investigadores de la estadística a esta nueva manera de entender el desarrollo de software, haciéndolo además de una manera activa. Para ello, se ha trabajado en la construcción de un framework específico para el modelado estadístico con MLG (Modelo Lineal General).

Chambers, (2000) se hace hincapié precisamente en la necesidad de que un usuario de software estadístico moderno no se limite a utilizar los componentes ya implementados por otros, sino que dé un paso más y sea capaz de desarrollar modificaciones en dichos componentes o crear otros nuevos que resuelvan su problema específico.

Villasante, (1994), reporta que el diseño de prototipo es el más adecuado para su implantación y explotación de la misma durante el desarrollo de aplicaciones a través del modelo de prototipo, se sugiere a las autoridades pertinentes de la Facultad (FINESI) considerar el curso de Ingeniería de Software como parte de la currícula, se recomienda la investigación de nuevas formas de prueba o evaluación del software a través de pruebas estadísticas.

Esteban Vegas lozano (1996) la tesis de Optimización en estudios de Monte Carlo en Estadística, Aplicaciones al contraste de hipótesis hace un análisis de la importancia de la simulación, haciendo hincapié en la estadística computacional como herramienta cada vez mas empleada en el avance de muchas ciencias, su conocimiento esta en correspondencia con el crecimiento de la informática de forma que no se puede entender la primera sin considerar la de la segunda.

Javier Ibañez Gonzales (2006). La tesis se centra, fundamentalmente, en el desarrollo de métodos y la implementación de algoritmos que calculan Funciones de Matrices, y su aplicación a la resolución de Ecuaciones Diferenciales Ordinarias (EDOs) y Ecuaciones Diferenciales Matriciales de Riccati (EDMRs), con estos métodos se refiere a cualquier software de cálculo matemático requiere de algoritmos y fundamentalmente base teórica del tema.

César Augusto Cayetano Muro, Koenig Alfredo Rojas Bustillos (2007) StadCore - Herramienta Estadística Modular: Una Orientación a los Procesos de Confiabilidad y Análisis de Psicometría, Las herramientas orientadas a áreas específicas de estudio son prácticamente escasas y normalmente requieren del uso de más de una aplicación para llegar a un resultado. De esta forma, se obtienen soluciones parciales a una problemática, lo que ocasiona la necesidad de recurrir a otras herramientas para que terminen el trabajo. Un caso muy particular de este problema es el del proceso de psicometría en el cual las etapas de confiabilidad y análisis requieren del uso de varias herramientas. Además, las soluciones

obtenidas en cada etapa no son completas dado que se necesita realizar procesos no automatizados que tienen mayor probabilidad de conllevar a errores. La presencia de estos errores en un test trae consigo importantes consecuencias dado que podría obtenerse conclusiones inexactas. Para solucionar este problema se implementó una herramienta que sea lo suficientemente flexible como para permitir ampliar sus funcionalidades a diversas áreas de trabajo, debido a que las hojas de cálculo electrónicas son aplicaciones que fácilmente pueden desempeñarse en cualquier área de estudio, se utilizó esta característica como interfaz de la herramienta.

1.3. OBJETIVO GENERAL

Desarrollar un software estadístico guía de procesamiento de datos para estudiantes de la Escuela Profesional de Ingeniería Estadística e Informática

1.3.1 OBJETIVOS ESPECÍFICOS

- ✓ Diseñar algoritmos computacionales para los métodos y técnicas estadísticas más usados.
- ✓ Construir módulos de código abierto de los algoritmos generados en un lenguaje de programación.
- ✓ Integrar diferentes módulos con características de fácil uso y programación amigable.

1.4. FORMULACIÓN DE HIPÓTESIS

El software estadístico con algoritmos de código abierto mejorará en el tratamiento de datos usando las técnicas y métodos estadísticos de las diferentes asignaturas en la escuela profesional de Ingeniería Estadística e Informática.



CAPÍTULO II

MARCO TEÓRICO

2.1. ANÁLISIS DESCRIPTIVO

Según Galán M. (2003), existen dos tipos de fenómenos en estadística descriptiva que son los fenómenos deterministas y los no deterministas ó aleatorios.

a) Fenómeno aleatorio. La experiencias que al ser repetidas en condiciones que estimamos similares, nos llevan a resultados diferentes constituyen los llamados fenómenos aleatorios.

El lanzamiento de un dado.

Número de usuarios que acuden a un banco en un determinado horario.

Las consecuencias de una determinada medicación.

b) Fenómeno determinista. Se denomina fenómeno determinista toda experiencia que, al ser repetida en condiciones que nosotros

consideramos similares produce siempre el mismo resultado, dentro de unos límites razonables de precisión.

La mayoría de los fenómenos macroscópicos estudiados por las ciencias de la naturaleza y de la vida.

2.1.1. Tabulación de frecuencias

Según Galán M. Es una distribución organizada de los datos recogidos en un estudio. Contiene un listado de distintas modalidades del fenómeno considerado, con la frecuencia absoluta, relativa y acumulada de cada una. Cuando el número de modalidades es demasiado grande (eso ocurre siempre con modalidades continuas) se agrupan en clases.

Reglas prácticas.

- Las clases han de ser excluyentes.
- Los límites de cada clase deben tener más precisión que las medidas realizadas.
- Aunque no tiene que ser necesariamente así, es conveniente que la amplitud de los intervalos sea constante.
- Todos los datos de una clase quedan representados por la marca de clase, que es el valor medio de intervalo que forma la clase. De esta manera todos los cálculos se realizan como si en lugar de tener N valores distintos en una clase, tuviéramos N veces la marca de clase.

Según Mitacc. M. la regla general para la construcción de una distribución de frecuencias de una variable agrupada en intervalos es:

1. Determinar el rango. Es conveniente determinar el rango de los datos obtenidos, los cuales se han definido por:

$$R = \max\{X\} - \min\{X\}, \text{ para construir intervalos traslapados. Ó}$$

$$R = \max\{X\} - \min\{X\} + \text{unidad de medida para construir intervalos no traslapados}$$

2. Determinación del número de intervalos de clases. Consiste en dividir el rango en un número conveniente de intervalos de clase, generalmente del mismo tamaño, es conveniente tener entre 5 y 20 intervalos de clase. Entre mas datos se tengan más intervalos de clase deben considerarse. No hay una fórmula exacta para calcular el número de intervalos de clase.

Una solución de referencia podría ser.

- a. Número de clases = $k = 5$, si n tamaño de la muestra y $k = \sqrt{n}$, si $n = 25$.

- b. Fórmula de Sturges: $k = 1 + 3.22 \log_{10}(n)$

Cuando los resultados para obtener k en a) y b) son números reales con decimales, entonces se redondará al entero inmediato mayor.

3. Determinación del tamaño de intervalos. Como regla general para encontrar la longitud de los intervalos y último intervalo. Esto es para permitir uniformidad en las comparaciones de frecuencias de clase.
4. Determinación de los límites de clase. Se debe tomar los resultados numéricos mas bajos de los datos originales como el límite inferior del primer intervalo de clase. Agregar C para obtener el límite superior de dicha clase, esto es, cuando se usa

Rango $= R = \max \{x\} - \min \{x\}$ (agregar 0-1 para obtener el límite superior de clase, caso que se usa intervalos no traslapados). Se añade los demás intervalos.

5. Determinación de la frecuencia de clase. Consiste en determinar el número de observaciones que caen en cada intervalo de clase.

2.1.2. Representación gráfica de variables cuantitativas. Las más usadas son:

1. **Diagrama de barras.** Esta forma de distribución gráfica es propia de las distribuciones que tienen muchas observaciones pero pocos valores distintos de una variable.

Dicho diagrama se elabora colocando en el eje de las abscisas los distintos valores de la variable y sobre cada una de ellas se levanta una línea perpendicular, cuya altura es la frecuencia (absoluta o relativa) de dicho valor.

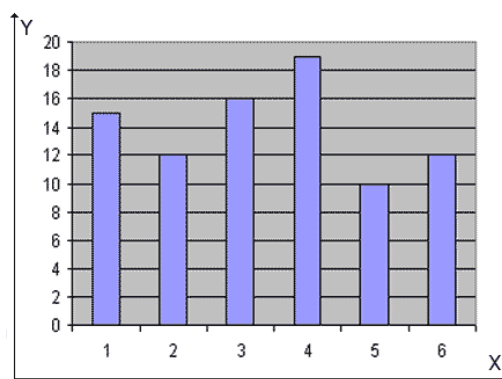


Fig. 1 Diagrama de Barras

2. Histograma. Es una representación gráfica de frecuencias agrupadas en intervalos de clase, mediante una serie de rectángulos contiguos que tienen:

- Sus bases sobre un eje horizontal (eje de las X) con centros en las marcas de clase y longitud igual al tamaño de los intervalos de clase.
- Las alturas proporcionales a la frecuencia (absoluta o relativa) tomamos sobre el eje de las Y.

A veces conviene más graficar en el histograma las frecuencias relativas en lugar de las frecuencias absolutas. En este caso, la altura correspondiente a cada rectángulo que habrá que levantar sobre el eje de las ordenadas será el cociente entre la frecuencia relativa del mismo y la amplitud del intervalo. El único cuidado que debe tenerse es que el área total del histograma sea igual a la correspondiente suma total de áreas de cada rectángulo.

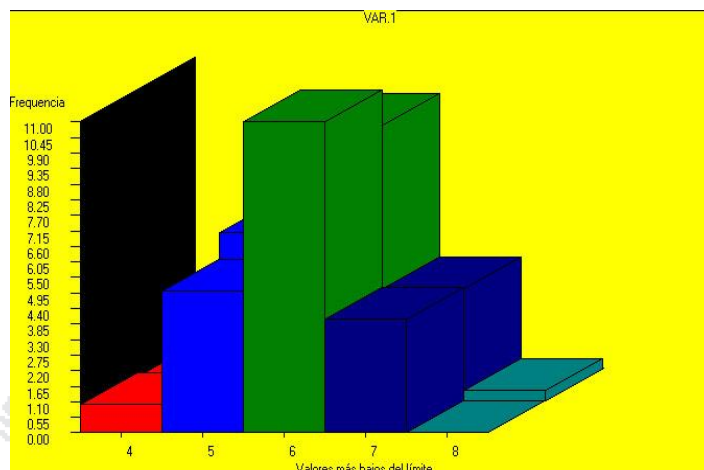


Fig. 2 Histograma

3. Polígono de frecuencias. Si la variable está agrupada en intervalos de clase, el polígono de frecuencias se obtiene uniendo los puntos medios de las bases superiores del rectángulo del histograma.

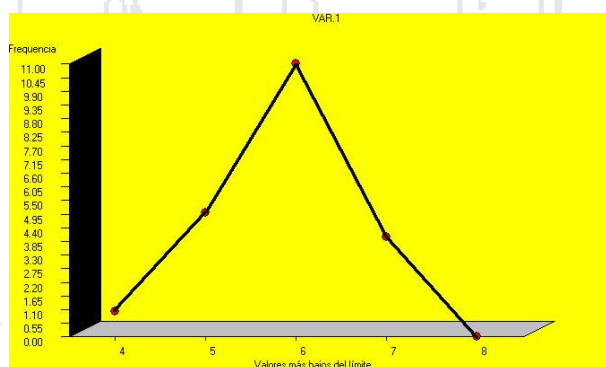


Fig. 3 Histograma

4. Polígono de frecuencia acumulada u ojiva. Esta representación es válida para variables estadísticas agrupadas en intervalos de clase.

En el eje de las abscisas representamos los distintos intervalos de clase que han de estar naturalmente traslapados. En el extremo superior de cada intervalo se levanta una vertical con altura igual a la frecuencia (absoluta o relativa) acumulada, luego se unen los extremos superiores de las

verticales con segmentos rectilíneos. Así el polígono de frecuencias absoluta acumuladas alcanzara su máxima altura en el último intervalo.

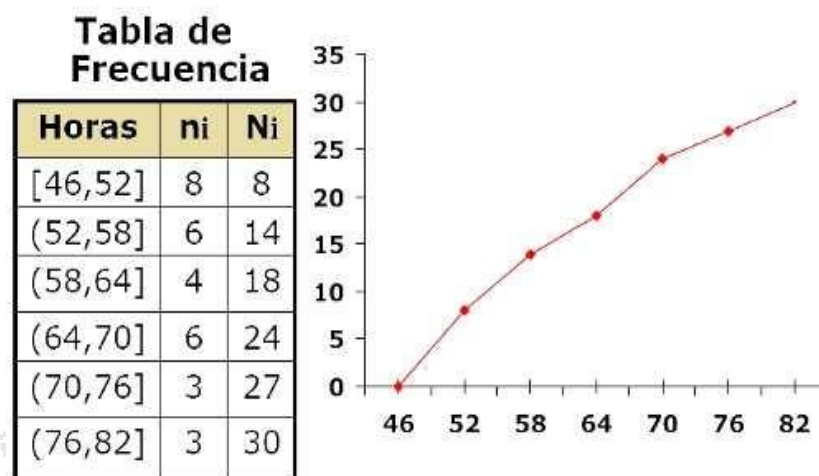


Fig. 4 Frecuencia acumulada u ojiva

2.1.3. Medidas de la tendencia central

Mittac M. (1996), indica que una medida de tendencia central es un índice de localización central empleado en la descripción de las distribuciones de frecuencias. También sirve como una base para medir y evaluar valores anormalmente altos o anormalmente bajos (o valores extremos).

Características del valor central.

1. Debe estar definido en forma objetiva.
2. Debe depender de toda la información obtenida en lo posible.
3. Debe ser fácil de comprender (no debe tener un carácter de abstracción) y de interpretar.
4. Debe ser fácil de calcular.

5. Debe ser estable (no debe ser sensible a fluctuaciones).
6. Debe ser adecuado a cálculos algebraicos posteriores.

Los promedios para datos agrupados más usuales son:

- a. **La Media aritmética o media.** Según Mitacc (1996). Sean x_1, x_2, \dots, x_k valores de la variable X ponderamos por sus respectivas frecuencias absolutas f_1, \dots, f_k . La media de la variable X es dada por.

$$M(X) = \bar{X} = \frac{\sum_{i=1}^k f_i x_i}{n} \quad \text{ó} \quad \bar{X} = \sum_{i=1}^k h_i x_i \quad \text{Ecuación (1)}$$

Donde: $n = \sum_{i=1}^k f_i$ y h_1, \dots, h_k son las frecuencias relativas respectivas.

- b. **La mediana.** Es un valor que divide a un conjunto de observaciones ordenadas en forma ascendente o descendente en dos grupos de igual número de observaciones.

La fórmula de la mediana para datos agrupados es:

$$\tilde{X} = \ell_{med} + \left(\frac{\frac{n}{2} - F_{k-1}}{F_k - F_{k-1}} \right) \cdot C_{med} \quad \text{Ecuación (2)}$$

Donde:

ℓ_{med} : Límite inferior de la clase que contiene a la mediana.

n : Tamaño de muestra

C_{med} : Amplitud de clase que contiene a la mediana.

F_k : Frecuencia acumulada de la clase que contiene a la mediana.

F_{k-1} : Frecuencia acumulada de la clase inmediatamente anterior a la clase que contiene la mediana.

2.1.4. Medidas de Dispersión o Concentración

Según Mitacc M. Las medidas de dispersión son los que cuantifican el grado de concentración o de dispersión de los valores de la variable en torno de un promedio o valor central de la distribución. Las medidas de dispersión se necesitan para dos propósitos básicos:

- Para verificar la confiabilidad de los promedios.
- Para que sirva como base para el control de la variación misma.

También podemos decir que los términos *concentración* y *dispersión* pueden ser utilizados indistintamente, pues se da la relación.

Alta dispersión \leftrightarrow Baja concentración

Baja dispersión Alta concentración.

Las medidas de dispersión que se utilizan con mayor frecuencia son:

1. Recorrido o rango.

El recorrido de una variable estadística es simplemente la diferencia entre su valor máximo y su valor mínimo y se denota por:

$$R = \max .\{X\} - \min \{X\}$$

2. Recorrido intercuantílico.

Como los cuartiles son tres puntos (valores) que dividen un ordenamiento de datos o una distribución de frecuencias en 4 grupos aproximadamente. Entonces la medida dada por.

$$Q_1 = Q_3 - Q_1$$

Llamada recorrido o amplitud intercuantílico, incluye la mitad central de los valores.

3. Recorrido semi-intercuantílico.

Esta medida se basa en la posición ocupada por el 50% de los valores centrales de la distribución, es dada por:

$$Q_{s_1} = \frac{Q_3 - Q_1}{2}$$

4. Desviación media.

La desviación media o promedio, es simplemente la media aritmética de los valores absolutos de las desviaciones de todos los valores en relación con algún punto central, tal como la media o la mediana. Formalmente tenemos la siguiente definición.

Sean x_1, x_2, \dots, x_k valores de la variable x , con frecuencias absolutas f_1, \dots, f_k , respectivamente. La desviación media de X con respecto a un promedio p es dado por:

$$D_M(p) = \frac{\sum_{i=1}^k f_i |x_i - p|}{n} \quad \text{donde } n = \sum_{i=1}^k f_i \quad \text{Ecuación (3)}$$

La fórmula para la desviación con respecto a la media aritmética es:

$$D_M(\bar{X}) = \frac{\sum_{i=1}^k f_i |x_i - \bar{X}|}{n} \quad \text{ó} \quad D_M(\bar{X}) = \sum_{i=1}^k h_i |x_i - \bar{X}| \quad \text{Ecuación (4)}$$

Y con respecto a la mediana es

$$D_M(\tilde{X}) = \frac{\sum_{i=1}^k f_i |x_i - \tilde{X}|}{n} \quad \text{ó} \quad D_M(\tilde{X}) = \sum_{i=1}^k h_i |x_i - \tilde{X}| \quad \text{Ecuación (5)}$$

Donde h_1, h_2, \dots, h_k son las frecuencias relativas.

5. Varianza y desviación estándar.

Sean x_1, x_2, \dots, x_k valores de la variable X , con frecuencias absolutas f_1, \dots, f_k , respectivamente la varianza muestral de X es dada por:

$$Var[X] = S^2 = \frac{\sum_{i=1}^k f_i (x_i - \bar{X})^2}{n-1} = \frac{n}{n-1} \sum h_i (x_i - \bar{X})^2 \quad \text{Ecuación (6)}$$

Donde: $n = \sum_{i=1}^k f_i$ y h_1, \dots, h_k son las frecuencias relativas.

La varianza poblacional se define en términos de la media poblacional μ , esto es.

$$\sigma^2 = \frac{\sum_{i=1}^k f_i (x_i - \mu)^2}{N} \quad \text{Ecuación (7)}$$

Donde $N = \sum_{i=1}^k f_i$ es el tamaño de la población.

2.1.5. Análisis comparativo

A la teoría de pequeñas muestras también se le llama teoría exacta del muestreo, ya que también la podemos utilizar con muestras aleatorias de tamaño grande.

Veremos un nuevo concepto necesario para poder entender la distribución t Student. este concepto es "grados de libertad".

Para definir grados de libertad se hará referencia a la varianza muestral.

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1} \quad \text{Ecuación (9)}$$

Esta fórmula está basada en n-1 grados de libertad.

Esta fórmula está basada en $n-1$ grados de libertad. Esta terminología resulta del hecho de que si bien s^2 está basada en n ,

$$x_1 - \bar{x}, x_2 - \bar{x}, \dots, x_n - \bar{x}$$

éstas suman cero, así que especificar los valores de cualquier $n-1$ de las cantidades determina el valor restante.

Por ejemplo: si $n=4$ y $x_1 - \bar{x} = 8$; $x_2 - \bar{x} = -6$ y $x_4 - \bar{x} = -4$, entonces automáticamente tenemos $x_3 - \bar{x} = 2$ así que sólo tres de las cuatro medidas de x_i están libremente determinadas, la otra debe tomar el valor que haga esta suma cero; es por esto que solo tenemos 3 grados de libertad.
 grados de libertad = número de mediciones - 1

Distribución de probabilidad t-Student

Una variable aleatoria se distribuye según el modelo de probabilidad t o T de Student con k grados de libertad, donde k es un entero positivo, si su función de densidades la siguiente:

$$h_k(t) = \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{k}{2}\right)\sqrt{\pi k}} \left(1 + \frac{t^2}{k}\right)^{-\frac{(k+1)}{2}} \quad -\infty < t < \infty, \quad \text{donde} \quad \Gamma(p) = \int_0^{\infty} e^{-x} x^{p-1} dx$$

Ecuación (10)

La gráfica de esta función de densidad es simétrica, respecto del eje de ordenadas, con independencia del valor de k , y de forma algo semejante a la de una distribución normal:

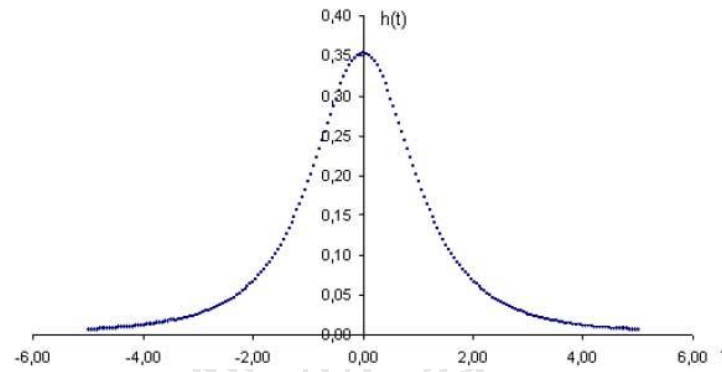


Fig. 5 Distribución t de Student con 10 grados de libertad

Su valor medio y varianza son:

$$E(T) = \mu = \int_{-\infty}^{\infty} t \cdot h_k(t) \cdot dt = \int_{-\infty}^{\infty} t \cdot \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{k}{2}\right)\sqrt{\pi k}} \left(1 + \frac{t^2}{k}\right)^{-\frac{(k+1)}{2}} dt = \dots = 0$$

Ecuación (11)

Si $k > 3$

$$\text{Var}(T) = \sigma^2 = E((T - \mu)^2) = \int_{-\infty}^{\infty} (t - \mu)^2 \cdot h_k(t) \cdot dt = \int_{-\infty}^{\infty} t^2 \cdot \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{k}{2}\right)\sqrt{\pi k}} \left(1 + \frac{t^2}{k}\right)^{-\frac{(k+1)}{2}} dt = \dots = \frac{k}{k-2}$$

Ecuación (12)

La siguiente figura presenta la gráfica de varias distribuciones t. La apariencia general de la distribución t es similar a la de la distribución normal estándar: ambas son simétricas y unimodales, y el valor máximo de la ordenada se alcanza en la media $\mu = 0$. Sin embargo, la distribución t tiene colas más amplias que la normal; esto es, la probabilidad de las colas es mayor que en la distribución normal. A medida que el número de grados de libertad tiende a infinito, la forma límite de la distribución t es la distribución normal estándar.

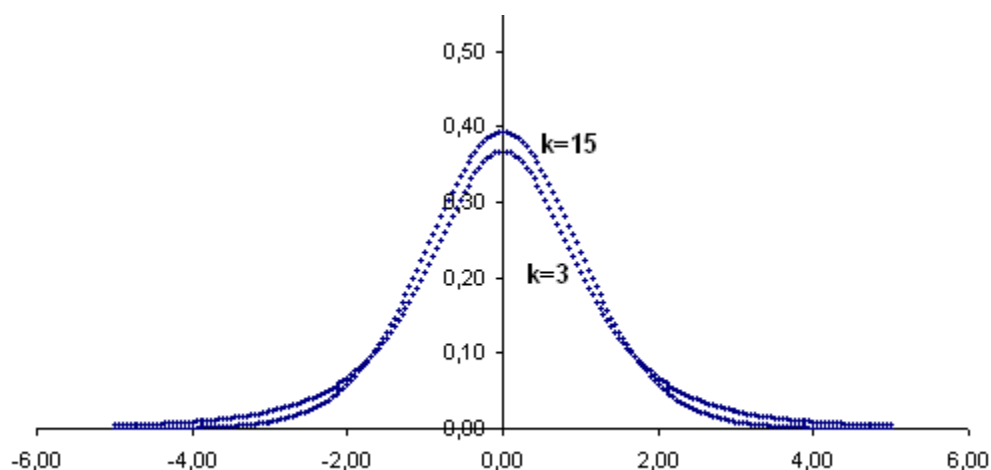


Fig. 6 Distribución t de student

Propiedades de las distribuciones t

1. Cada curva t tiene forma de campana con centro en 0.
2. Cada curva t, está más dispersa que la curva normal estándar.
3. A medida que k aumenta, la dispersión de la curva t correspondiente disminuye.
4. A medida que $k \rightarrow \infty$, la secuencia de curvas t se aproxima a la curva normal estándar

La distribución de probabilidad de t se publicó por primera vez en 1908 en un artículo de W. S. Gosset. En esa época, Gosset era empleado de una cervecería irlandesa que desaprobaba la publicación de investigaciones de sus empleados. Para evadir esta prohibición, publicó su trabajo en secreto bajo el nombre de "Student". En consecuencia, la distribución t

normalmente se llama distribución t de Student, o simplemente distribución t.

2.1.6. Asociación y correlación

El concepto de Asociación o Correlación Estadística entre dos variables es, en general, otro de los temas de Estadística Descriptiva que no está correctamente presentado en los textos básicos de Estadística. Por ello es frecuente encontrar falsas y absurdas conclusiones supuestamente basadas en el estudio de correlaciones estadísticas como ejemplos para desprestigiar los métodos estadísticos (ver “La Correlación no Implica Relación Causa Efecto”) Insistimos que esta situación sólo refleja la incultura estadística de quienes proponen tales conclusiones y que dicha incultura es de responsabilidad de los que estamos a cargo de la enseñanza de esta disciplina.

Una Conceptualización de Correlación Estadística

Una adecuada interpretación del concepto de asociación o correlación en Estadística es fundamental independizar su definición de cualquier sugerencia sobre la relación causa-efecto.

Para ello se debe precisar en qué sentido se dice que dos variables cuantitativas están asociadas o correlacionadas estadísticamente. Al respecto se sugiere utilizar la siguiente caracterización:

En Estadística Descriptiva se dice que dos variables cuantitativas “están asociadas”, “son dependientes”, o “están correlacionadas” si cuando se

aumentan los valores de una variable, los valores de la otra tienden a: i) o bien a aumentar (y se dice que la asociación o dependencia es directa o que la correlación es positiva)

ii) o bien a disminuir (y se dice que la asociación o dependencia es inversa o que la correlación es negativa)

Cuando no se presenta esta tendencia se dice que las variables no están asociadas o no son dependientes o no están correlacionadas.

Para completar la conceptualización de la Correlación Estadística es necesario destacar que la “tendencia” se refiere a la seguridad o grado de certeza respecto del hecho de que cuando una variable aumenta, la otra cumple o bien i) o bien ii). Si se tiene una alta expectativa sobre esta ocurrencia, la Correlación es “alta”; si la expectativa de ocurrencia es “baja” la Correlación es baja.

No se debe confundir la calificación de “alto” o “bajo” con la magnitud del incremento o disminución. Los siguientes dibujos ilustran mejor esta situación.

2.1.7. Modelado Orientado a Objetos

2.1.7.1. Introducción al concepto de Modelado

Según Booch et al. (1999), se puede decir que el modelado es una parte central de todas las actividades que conducen a la programación de buen software. Se construyen modelos para comunicar la estructura deseada y el comportamiento del sistema.

Estos mismos autores (Booch et al., 1999) plantean que el software de calidad se centra en optimizar al máximo el código, imaginar cómo escribir menos software y conseguir el mejor rendimiento sin que disminuya la eficacia.

2.1.7.2. Paradigma Orientado a Objetos en el Desarrollo de Software

La génesis de las ideas básicas de la OO se produce a principios de los años 60 y se atribuye al trabajo del Dr. Nygaard y su equipo de la Universidad de Noruega. Estos investigadores realizaban trabajos en sistemas informáticos para simular sistemas físicos, como el funcionamiento de motores. La dificultad con la que se encontraban era doble: los programas eran muy complejos y forzosamente tenían que ser modificables. Este segundo punto era especialmente problemático, ya que eran muchas las ocasiones en las que se requería probar la viabilidad y el rendimiento de estructuras alternativas.

La solución que idearon fue diseñar el programa con una estructura paralela a la del sistema físico. Es decir, si el sistema físico estaba compuesto por cien componentes, el programa también tenía cien módulos, uno por cada pieza. Este enfoque resolvió los dos problemas planteados.

2.1.7.3. Programación Orientada a Objetos

Existen cuatro conceptos o principios básicos de la OO:

- Encapsulado y ocultación de la información.
- Clasificación, tipos abstractos de datos

- Herencia.
- Polimorfismo.

2.1.7.4. Requisitos del sistema

El esfuerzo principal en la fase de requisitos es desarrollar un modelo del sistema que se va a construir, y la utilización de los casos de uso es una forma adecuada de crear ese modelo. Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales del sistema. Mediante la utilización de los casos de uso, los analistas se ven obligados a pensar en términos de quiénes son los usuarios y qué necesidades u objetivos se deben cumplir. Además, los casos de uso dirigen todo el proceso de desarrollo, puesto que la mayoría de actividades como el análisis, diseño y prueba se llevan a cabo partiendo de los casos de uso. Esas dos razones, la captura de los requisitos del sistema y la dirección del proceso de desarrollo, son precisamente los motivos por los cuales los casos de uso se han hecho populares y se han adoptado universalmente (Jacobson y Christerson, 1995).

2.1.7.5. Introducción Al Concepto De Modelado

Siguiendo a Booch et al. (1999), se puede decir que el modelado es una parte central de todas las actividades que conducen a la programación de buen software. Se construyen modelos para comunicar la estructura deseada y el comportamiento del sistema.

A través del modelado, se consiguen cuatro objetivos:

1. Ayudar a visualizar cómo es o debería ser un sistema.
2. Especificar la estructura o el comportamiento de un sistema.
3. Proporcionar plantillas que guíen la construcción de un sistema.
4. Documentar las decisiones adoptadas.

2.1.7.6. Ingeniería del Software

La informática es la rama de la ciencia de la técnica que se ocupa de los ordenadores, tanto en la vertiente de arquitectura de los sistemas informáticos, como en la vertiente de diseño de técnicas para el tratamiento de la información.

2.1.7.7. Fases Del Modelado Orientado a Objetos

En el proceso de modelado orientado a objetos (OO) podemos distinguir las tres fases tradicionales de elaboración de un sistema informático (Pressman, 1993):

- 1) Análisis: se ocupa del qué, de entender el dominio del problema.
- 2) Diseño: responde al cómo, y se centra en el espacio de la solución.
- 3) Implementación: fase en la que se adapta la solución a un entorno de programación específico.

Como señalan McGregor y Korson (1990), durante el análisis se abordan los objetos del dominio del problema, mientras que en el diseño se especifican los objetos adicionales necesarios para obtener la solución

implementada en el ordenador. Precisamente la no consideración de ambas fases ha conducido al desastre a no pocos proyectos de software.

2.1.7.8. Análisis

Siguiendo a Booch (1996), el análisis orientado a objetos (AOO) se puede definir como:

«Un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema».

Este mismo autor, examina varios enfoques contrastados para el análisis que son de relevancia para los sistemas orientados a objetos: enfoques clásicos, enfoques de análisis del comportamiento, enfoques de análisis del dominio, análisis de casos de uso, fichas CRC y descripción informal en lenguaje natural.

2.1.7.9. Diseño

El objetivo de la fase de diseño es la generación de una descripción sobre cómo sintetizar los objetos extraídos del dominio de la solución al problema, cuyo comportamiento esté de acuerdo con el modelo de análisis y con el resto de requisitos del sistema asociados a la fase de implementación de dichos objetos como componentes de software.

De forma más abstracta, Booch (1996) define el diseño orientado a objetos como «un método de diseño que abarca el proceso de descomposición orientada a objetos y una notación para describir los modelos lógico y

físico, así como los modelos estático y dinámico del sistema que se diseña». PU junto con los métodos OOSE (Jacobson et al., 1992) y OMT (Rumbaugh et al., 1991). El método de Booch (1996) recomienda el uso de cuatro modelos: lógico, físico, estático y dinámico. Concretamente, en el diseño del sistema considera la dimensión estática y dinámica mediante dos niveles para cada una de ellas: el nivel lógico y el nivel físico. De ahí los cuatro modelos enunciados.

1. El modelo lógico contempla las clases y objetos existentes y las relaciones que se establecen entre ellas.
2. El modelo físico se refiere a la arquitectura del sistema, organizada en módulos y en procesos.
3. El modelo estático, tanto lógico como físico, contempla los aspectos estructurales – estructura de clases y de módulos– y las relaciones – entre clases (herencia, asociación, agregación) y entre módulos (dependencias).
4. El modelo dinámico se refiere al comportamiento del sistema, tanto a nivel lógico (escenarios de utilización), como físico (asignación de procesos a módulos y procesadores).

La expresión de estos modelos se lleva a cabo mediante diversos diagramas:

1. Diagrama de clases: muestra la existencia de clases y las relaciones entre ellas en la perspectiva lógica y estática del sistema.

2. Diagrama de objetos: muestra la existencia de objetos y las relaciones entre ellos en la perspectiva lógica y dinámica (puesto que permiten reseguir la ejecución de un escenario) del sistema.
3. Diagramas de transición de estados: muestra el comportamiento dinámico de las clases describiendo los sucesivos estados en que pueden existir. Se refieren, por tanto, a la perspectiva lógica y dinámica del sistema.
4. Diagramas de módulos: muestra la asignación de clases a módulos en la perspectiva física y estática del sistema.
5. Diagramas de procesos: muestra la asignación de procesos a los procesadores en la perspectiva física y dinámica del sistema.

2.1.7.10. Implementación

La fase de implementación de la solución descrita en el diseño está condicionada por las características propias del lenguaje de programación OO elegido para la codificación. En la implementación codificamos el sistema en términos de componentes, es decir, ficheros de código fuente, ficheros de código binario, ejecutables y similares. Afortunadamente, la mayor parte de la arquitectura del sistema ha sido capturada durante el diseño, siendo el propósito principal de la implementación el desarrollar la arquitectura y el sistema como un todo a partir de la programación orientada a objetos (POO).

Booch (1996) define la POO como «un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son, todas ellas, miembros de una jerarquía de clases unidas mediante relaciones de herencia».

En la definición anterior hay tres partes importantes:

1. La programación orientada a objetos utiliza objetos, no algoritmos, como sus bloques lógicos de construcción fundamentales; desde el punto de vista físico, el bloque de construcción de los lenguajes orientados a objetos es una colección lógica de clases y objetos, en lugar de los subprogramas de la programación estructurada o algorítmica (véase apartado 2.3.2).
2. Cada objeto es una instancia de alguna clase.
3. Las clases están relacionadas con otras clases por medio de relaciones de herencia (la jerarquía de clases de la que se habló en el apartado 2.3.4).

Para el desarrollo del software es necesario realizar el análisis, diseño e implementación de FisTad, se puede adecuar al estándar conocido como Proceso Unificado de Desarrollo de Software (Jacobson et al., 2000), generalmente referenciado como Proceso Unificado (UP). El objetivo es construir un producto software.

2.1.7.11. El Proceso Unificado de Desarrollo de Software

La relación entre estas tres etapas y la forma de operativizar dicha relación se puede adecuar al estándar conocido como Proceso Unificado de Desarrollo de Software (Jacobson et al., 2000), generalmente referenciado como Proceso Unificado (UP).

Los autores del UP crean un modelo de proceso enfocado desde la perspectiva de desarrollo de software en el campo industrial y empresarial. En ese sentido, el proceso involucra tanto a las personas (sus habilidades o roles) pertenecientes a los diferentes niveles de organización de las empresas, como a las tecnologías –lenguajes de programación, sistemas operativos, ordenadores, entornos de desarrollo, estructuras de red, etc.– disponibles en el momento en que se va a emplear el proceso, y a las herramientas de software que se utilizan para automatizar las actividades definidas en el proceso (por ejemplo, una herramienta CASE de modelado para UML).

El UP es el producto final de tres décadas de desarrollo y uso práctico en el que han estado involucrados otros productos de proceso anteriores y que ha recibido aportaciones de muchas otras fuentes (Jacobson et al., 2000).

El UP se define como un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Está basado en componentes (i.e., el sistema software en construcción está formado por componentes software) y utiliza el nuevo estándar de modelado visual UML para describir los

distintos modelos generados durante el desarrollo del proceso, es decir, para preparar todos los esquemas de un sistema software.

Se diferencian cuatro fases en el proceso de desarrollo de software, atendiendo al momento en que se realizan: inicio, elaboración, construcción y transición.

En la fase de inicio se establece una visión del proyecto y su alcance, es cuando la idea inicial para el desarrollo se lleva al punto de estar suficientemente fundamentada para garantizar la entrada en la fase de elaboración. En esta fase de inicio se evalúa la viabilidad del proyecto, sobre todo cuando está en juego una gran inversión de recursos humanos y económicos.

La elaboración es la segunda fase del proceso, cuando se definen la visión del producto y su arquitectura. En esta fase se expresan con claridad los requisitos del sistema, proporcionando una arquitectura estable para guiar el sistema a lo largo del ciclo de vida.

En la última iteración dentro de la fase de elaboración se observa cómo empieza a adquirir relevancia la implementación del diseño. En cada iteración se identifican e implementan unos cuantos casos de uso. Cada iteración, excepto quizás la primera de todas de un proyecto, se dirige por los casos de uso a través de todos los flujos de trabajo, de los requisitos al diseño y a la prueba, añadiéndose un incremento más en el desarrollo del sistema.

Durante la fase de construcción es cuando se desarrolla, también de forma iterativa e incremental, un producto completo que está preparado para la transición hacia la comunidad de usuarios. Esto significa describir los requisitos restantes, refinando el diseño y completando la implementación y las pruebas de software.

Finalmente, durante la fase de transición, el software se despliega en la comunidad de usuarios. Una vez que el sistema ha sido puesto en manos de los usuarios finales, a menudo aparecen cuestiones que requieren un desarrollo adicional para ajustar el sistema, corregir algunos problemas no detectados o finalizar algunas características que habían sido pospuestas. Esta fase comienza normalmente con una versión beta del sistema, que luego será reemplazada por la versión definitiva del producto.

2.1.7.12. Requisitos del Sistema

Hay diferentes puntos de partida para la captura de requisitos del sistema. Cuando el software a desarrollar da soporte directamente al negocio de una empresa, se puede acudir al llamado modelo de negocio, como abstracción o esquema del modelo de casos de uso. El modelo de negocio permite comprender los procesos de negocio de la organización, en términos de casos de uso del negocio y actores del negocio; permite entender el contexto del que se van a extraer los casos de uso para el sistema software.

2.1.7.13. Análisis de los Requisitos

Llevando a cabo el análisis se consigue una separación de intereses que prepara y simplifica las subsiguientes actividades de diseño e

implementación, delimitando los temas que deben resolverse y las decisiones que deben tomarse en esas actividades. En este sentido, el diseño y la implementación son mucho más que el análisis. En el diseño, se debe moldear el sistema y encontrar su forma: una forma que dé vida a todos los requisitos incorporados en el sistema; una forma que incluya componentes de código que se compilan e integran en versiones ejecutables del sistema durante la implementación; una forma que se mantenga firme bajo las presiones del tiempo, los cambios y la evolución; una forma con integridad (Jacobson et al., 2000).

2.1.7.14. Diseño de la Solución

De igual forma que en el modelo de análisis, el modelo de diseño también define clasificadores –en este caso, se definen clases, subsistemas e interfaces–, relaciones entre estos clasificadores, y colaboraciones que llevan a cabo los casos de uso (las realizaciones de los casos de uso). Sin embargo, los elementos definidos en el modelo de análisis son más conceptuales, mientras que el modelo de diseño es más “físico” por naturaleza, sus elementos se adaptan al entorno de la implementación.

Puede hacerse la traza de una realización de caso de uso en el modelo de análisis a partir de una realización de caso de uso en el modelo de diseño

2.1.8. Instrumentos de Desarrollo

Cuando se utiliza una plataforma como MS-Windows, se programa mediante el API de Windows, como se sabe, los primeros Lenguajes en acceder a las API de Windows fueron Ms C++ 7.0, Borland C++ 3.1, pero

la programación en estos lenguajes era bastante difícil, ya que trabajan mediante Ms DOS, y la programación Visual tomaba mucho tiempo.

Otros lenguajes como Ms Visual Fox Pro y Ms Visual Basic no son orientado a objetos, aunque den esa apariencia, lo que permite una programación estructurada, e incluso híbrida, en su entorno visual.

El Lenguaje de programación Object Pascal trae ventajas como la de programar a través de un IDE, Librerías Active X de Windows, y sus Librerías VCL, propias de Embarcadero, la empresa que actualmente desarrolla Delphi.

La programación en Delphi es fácil de adaptar para programadores en C++, permitiendo manejar objetos de manera adecuada, accediendo a las propiedades, métodos y eventos de cada uno de los objetos, accediendo a la Programación Orientada a Eventos, se puede compartir un mismo evento para diversos objetos y/o clases

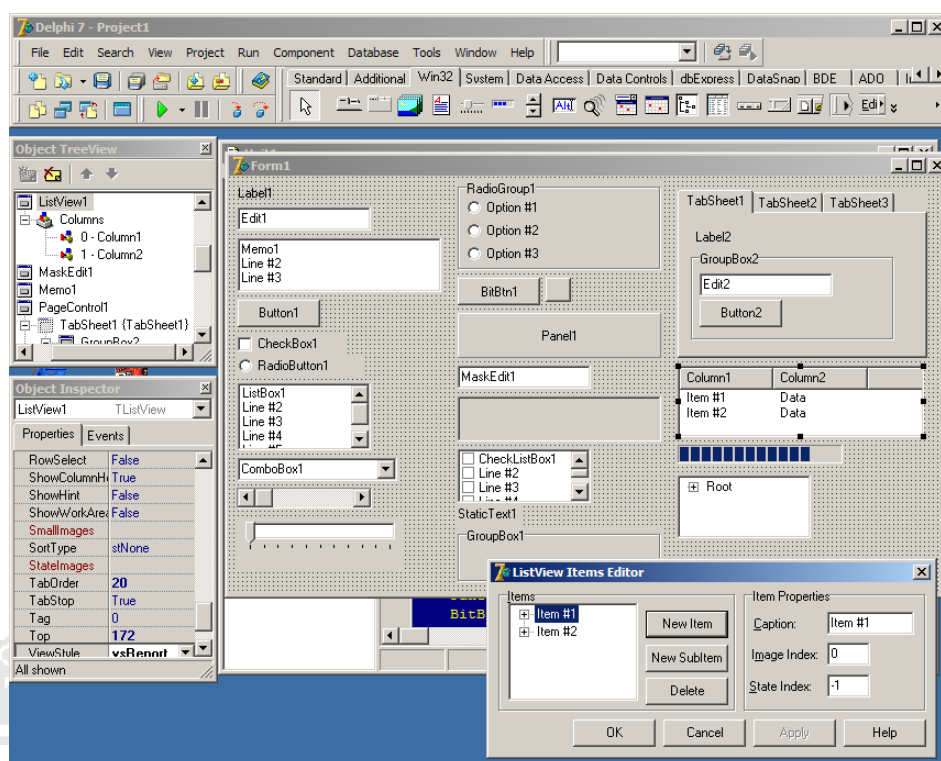


Fig. 7 IDE de Delphi.

CARACTERÍSTICAS DE DELPHI

- El IDE
- Proyectos
- El VCL (Componentes, Propiedades y Eventos)
- Información en tiempo de ejecución RTTI (Run Time Type Information)
- Incluye mejoras al Object Pascal, introduce muchas clases de manejo de clases especiales. Tiene constructores, plantillas, reglas de precedencia, el GDI y más.
- Es 10 veces más rápido que lenguajes como Visual Basic, Java o Power Builder.

Permite programar con los estándares como STL.

2.2. BASE TEÓRICA

2.2.1. Concepto de Población y muestra

- Población o universo. Conjunto de elementos que poseen una propiedad común, y que constituye la totalidad de individuos de interés para nuestro estudio.
- Muestra. Cualquier subconjunto de la población sobre el que se realizan los estudios para obtener conclusiones acerca de las características de la población.
- Individuo. Cada uno de los elementos de la muestra o de la población. No tienen porque ser objetos físicos.

2.2.2. Fases o niveles del método estadístico

- Estadística descriptiva. Recopilación y análisis de datos. Pueden ser datos referentes a toda la población o solo a una muestra, pero en este último caso no se pretende sacar conclusiones acerca de la población a la que pertenece la muestra.
- Teoría de la probabilidad. Es la teoría matemática en que se basa la posibilidad de realizar inferencias acerca de las propiedades de la población partiendo de la información contenida en la muestra.

- Estadística inferencial o inductiva. Utiliza la información que se desprende del análisis de una muestra para realizar una estimación de las propiedades de la población de la que se extrajo.

2.2.3. Organización de datos.

a. Organización de Fístad.

Modo de vista de datos. Muestra los valores de datos reales de valor definidas. En vista cada columna es una variable única y cada fila un caso individual, una observación.

Vista de variables. Muestra la información de definición de cada variable, que incluye las etiquetas de la variable definida (explicación de la variable), el tipo de datos escala a medida y los valores perdidos definidos por el usuario

En la ventana en la que muestra todos los resultados de las operaciones estadísticas realizadas. E posible editar los resultados para emplearlos posteriormente.

Las barras de menús contienen una serie de menús desplegable que permiten controlar la mayor parte de las acciones de FisTad. Está ubicado en la parte superior del programa.

Barras de herramientas. Una barra de herramientas es un conjunto de botones-íconos que permiten ejecutar muchas de las funciones de Fístad de forma sencilla.

Barras de estado. Las barras de estado están situadas en la parte inferior de las ventanas de FisTad. Pueden ocultarse/mostrarse según la opción que se elija.

Procedimiento general de resolución de un problema estadístico con FisTad.

- Recoger información.
- Grabar esa información en un archivo de datos.
- Sobre el archivo de datos usando los procedimientos establecidos seleccionar las opciones
- Mostrar los resultados en un visor de resultados
- Interpretar los resultados y extraer de los mismos las conclusiones que le parecen relevantes, y con eso se cierra el ciclo.

Trabajo con archivo de datos. Los datos deben de soportar variables cuantitativas como cualitativas, es decir variables discretas como continuas.

La aplicación debe de tener en cuenta la definición de variables

2.3. MARCO CONCEPTUAL

2.3.1. Modelamiento

Consiste en la habilidad para analizar un problema, resumir sus características esenciales, seleccionar y modificar las suposiciones básicas

que caracterizan al sistema, y luego enriquecer y elaborar el modelo hasta obtener una aproximación útil.

2.3.2. Lenguaje De Programación

Vocabulario y conjunto de reglas gramaticales con el fin de indicarle a un computador como realizar tareas específicas. Cada lenguaje tiene un conjunto único de vocablos (palabras con semántica) y reglas de sintaxis.

2.3.3. Programa

Conjunto de instrucciones escritas en un lenguaje de programación que, traducidas a lenguaje de máquina, desempeñan una tarea determinada.

2.3.4. Memoria

Son áreas de almacenamiento interno en el computador. Se puede pensar en la memoria como un arreglo de casillas, donde cada una puede almacenar información.

2.3.5. IDE

Entorno de Desarrollo Integrado, se usa este concepto para describir a un software que tiene un entorno de desarrollo donde se facilita la programación visual con el mouse liberando al programador del tedioso trabajo del diseño de interfaces desde cero.

2.3.6. Algoritmo

Conjunto de pasos ordenados para resolver un problema, como una fórmula matemática o las instrucciones de un programa.

2.3.7. Analista

Persona que realiza el análisis de un problema. Tras el análisis del problema el programador puede confeccionar el programa para computadora a partir de las indicaciones del análisis.

2.3.8. Objetos

Un objeto es como una "caja negra" el cual recibe y envía mensajes. Una caja negra contiene código y datos. Una regla principal de la programación Orientada a Objeto es: al usar un objeto nunca necesitamos ver lo que hay dentro de la caja.

¿Porqué no necesitamos ver lo que hay dentro de un objeto?. La respuesta es simple y se debe a que todas las comunicaciones se dan vía mensajes, es decir llamado de métodos los cuales reciben los datos, estos los procesan y regresan un resultado. El objeto que recibe un mensaje es llamado el receptor del mensaje y el que envía el mensaje es llamado el emisor del mensaje. Cualquier objeto puede ser representada por estos mensajes. Así que por ello no necesitamos conocer nada acerca de lo que hay en esa "caja negra".

Entonces podemos decir que un objeto es, sencillamente, algo que tiene sentido en el contexto de la aplicación, más claro, el objeto tendrá un

significado dependiendo en que aplicación se esté desarrollando y los resultados que este nos arroje. Se definirá un objeto como un concepto, abstracción o cosa con límites bien definidos y con significado a efectos del problema que se tenga entre manos. Los objetos tienen dos propósitos: promover la comprensión del mundo real y proporcionar una base práctica para la implementación por computadora. La descomposición de un problema en objetos depende del juicio y la naturaleza del problema. Todos los objetos poseen su propia identidad y se pueden distinguir entre sí. El término identidad significa que los objetos se distinguen por su existencia inherente y no por las propiedades descriptivas que pueda tener.

2.3.9. Clases

Un objeto es definido vía clases, las cuales determinan las propiedades de los objetos. Una clase de objetos describe un grupo de objetos con propiedades (atributos) similares, con relaciones comunes con otros y con una semántica común. Persona, compañía, animal, proceso y ventana son todos ellos clases de objetos. Cada persona tiene una edad, un CI1 y puede efectuar ciertos trabajos. Todo proceso tiene un poseedor, una prioridad y una lista de recursos requeridos. Los objetos y sus clases suelen parecer como sustantivos en las descripciones de problemas.

Es frecuente utilizar la abreviatura clase en lugar de decir clase de objetos. Los objetos de una clase tienen los mismos atributos y los mismos patrones de comportamiento. Casi todos los objetos derivan su individualidad de diferencias en los valores de sus atributos y en sus relaciones con otros

objetos. Sin embargo, son posibles, objetos que tengan unos valores de atributos idénticos y también las mismas relaciones entre sí.

2.3.10. Atributos

Un atributo es una característica que describe los objetos de una clase. Nombre, edad y peso son atributos de los objetos del tipo Persona. Color, peso y año del modelo son atributos del objeto del tipo coche. Cada atributo tiene un valor para cada instancia del objeto.

Los atributos deberían ser valores puros de datos y no objetos. A diferencia de los objetos, los valores puros de datos no poseen identidad.

2.3.11. Operaciones Y Métodos

Una operación es una función o transformación que se puede aplicar a los objetos de una clase. Mover, Seleccionar y Pintar son operaciones de la clase Figura. Abrir, Cerrar y Mostrar son operaciones de la clase Ventana.

Cada operación tiene un objeto blanco como argumento implícito. El comportamiento de la operación depende de la clase de su blanco. Todo objeto "conoce" su clase y, por tanto, la implementación correcta de la operación.

Una misma operación puede aplicarse a muchas clases distintas. Tal operación será polimórfica; esto es, una misma operación adopta distintas formas en distintas clases. Un método es la implementación de una operación una clase. Por ejemplo, la clase puede tener una operación imprimir. Se podrían implementar distintos métodos para imprimir archivos

ASCII, archivos binarios, y archivos de imágenes digitalizadas. Todos estos métodos efectúan una misma tarea lógica: imprimir un archivo, por tanto es posible aludir a ellos mediante la operación genérica imprimir. Si embargo, cada método puede estar implementado mediante un segmento de código fuente.

2.3.12. Especialización

La agregación no es el único camino en el cual dos objetos pueden ser relacionados. Un objeto puede ser una especialización de otro objeto.

Especialización puede ser:

El proceso de definir un nuevo objeto que se basa en una (típica) definición más limitada de un objeto existente, un objeto que está directamente relacionado, y más estrictamente definida, con otro objeto.

La especialización es usualmente asociada con clases. Esto es lo que llamamos "classless" de los sistemas orientados a objeto.

2.3.13. Superclase y Subclase

Al crear una clase nueva, en lugar de escribir variables atributos y métodos totalmente nuevos, el programador puede indicar que la nueva clase debe heredar los atributos y los métodos de una superclase previamente definida. Decimos que la nueva clase es una subclase. Cada subclase es una candidata para convertirse en superclase de alguna subclase futura.

La superclase directa de una subclase es la superclase de la que la subclase hereda explícitamente. Una superclase indirecta hereda desde dos o más niveles hacia arriba en la jerarquía de clases.

Una subclase normalmente agrega sus propios atributos y métodos, de modo que una subclase generalmente es mayor que su superclase. Una subclase es más específica que su superclase y representa un grupo más pequeño de objetos. Con la herencia única, la subclase inicialmente es prácticamente igual a su superclase. El verdadero valor de la herencia radica en la capacidad de definir en la subclase adiciones a las características heredadas de la superclase o sustituciones de éstas.

Generalización

La generalización es la relación entre una clase y una o más versiones refinadas de esa misma clase. La que se está refinando se denomina la superclase y cada versión refinada se denomina una subclase.

Existen dos caminos por los cuales la generalización puede ser afectada:

- Propiedades comunes y funciones de un grupo de objeto similares son agrupadas junto a una nueva generación de tipos de datos.
- Subtipos de un tipo de objeto dado pueden ser definidos usando predicados, forzando los valores de los atributos.

La generalización es una construcción útil tanto para el modelado conceptual como para la implementación. La generalización facilita el

modelo estructurando las clases, capturando de forma concisa lo que es similar y lo que es distinto con respecto a las otras clases.

2.3.14. Matrices

Una matriz es un arreglo bidimensional o tabla bidimensional de números consistente en cantidades abstractas que pueden sumarse y multiplicarse entre sí.

Es una disposición de valores numéricos y/o variables (representadas por letras), en columnas y filas, de forma rectangular.

Una matriz es una tabla cuadrada o rectangular de datos (llamados elementos o entradas de la matriz) ordenados en filas y columnas, donde una fila es cada una de las líneas horizontales de la matriz y una columna es cada una de las líneas verticales de la matriz. A una matriz con m filas y n columnas se le denomina matriz $m \times n$; y a m y n se les denomina dimensiones de la matriz.

Las dimensiones de la matriz siempre se dan con el número de fila primero y el número de columnas.

Por lo general se trabaja con matrices formadas por números reales. Las matrices se usan generalmente para describir sistemas de ecuaciones lineales, sistemas de ecuaciones diferenciales o representar una aplicación lineal (dada una base).

2.4. OPERACIONALIZACIÓN DE VARIABLES

VARIABLES	DIMENSIONES	INDICADORES
Independiente. 1. Métodos y técnicas estadísticas.	1.1. Estadística Descriptiva. 1.2. Estadística Comparativa. 1.3. Asociación y Correlación.	1.1.1. Medidas de tendencia Central Media, Mediana, Moda, Media Ponderada, Media Geométrica. 1.1.2. Medidas de Dispersión varianza y desviación estándar. 1.1.3. Tabulación de frecuencias y representación gráfica de datos.
2. Algoritmos	2.1. Algoritmos estadísticos	1.2.1. t de student 1.3.1. Auto correlación. 2.1.1 Programación de algoritmos

CAPÍTULO III

MATERIALES Y METODOS

3.1. POBLACIÓN

La población está constituida por: Los alumnos de la escuela profesional de Ing. Estadística e Informática que están cursando actualmente el sexto semestre hasta el décimo semestre, que haciende a un total de 125 estudiantes con asistencia regular.

3.2. METODOLOGÍA

En la construcción del software estadístico haremos uso de la metodología UML.

3.2.1. Análisis de requerimientos

Antes de iniciar el desarrollo de un instrumento informático es preciso especifica claramente cuáles son las características que deberá poseer. Este análisis permitirá guiar el estudio de la viabilidad del proyecto,

estableciendo un marco preciso para evaluar y comparar las diferentes plataformas existentes en la actualidad, con el objetivo de decidir cuál de ellas es la más adecuada para implementar la herramienta final (Hawryskiewicz, 1990; Bell et al., 1992).

Nuestra primera consideración para el desarrollo de un software estadístico es que este software debe ser manejado mediante un sistema de menús con opciones para cada técnica o método estadístico ofreciendo distintas opciones a través de la selección de menús y cuadros de diálogos, a estos sistemas se denominan sistemas guiados, la facilidad de uso de los sistemas guiados por menús es una característica deseada en cualquier herramienta informática aunque implique menor libertad para usuarios avanzados, compensando en alguna manera tener acceso al código fuente para dichos usuarios que pueden personalizar sus herramientas.

Al desarrollar el software en un lenguaje de programación de alto nivel el investigador tiene pleno control sobre el desarrollo de las interfaces, cuadros de dialogo y algoritmos computacionales óptimos.

El entorno debe proporcionar funciones para la visualización grafica de los resultados, fabulación de datos y mostrar resultados obtenidos de forma concisa.

Las principales características que debería cumplir la herramienta informática en la opinión del investigador es:

- Debe tratarse de un programa cerrado basado en interface gráfica.

- Las interfaces deben estar diseñadas de forma específica para el análisis estadístico
- El programa debe proporcionar funciones y documentación necesaria para construir subprogramas que funciones de forma independiente en el entorno.
- El programa debe incorporar funciones para la visualización gráfica.

3.2.2. Estudio de la Viabilidad

a. Plataforma de desarrollo

Se desarrolló el Delphi que es un lenguaje de programación orientado a objetos, basado en pascal.

El desarrollo de programas escritos en uno de los lenguajes de programación de alto nivel estándar como fortran, pascal, c o basic entre otros, ofrecen una ventaja de ser muy adaptables y amplios permitiendo, a su vez, resolver cualquier problema de simulación estadística (Fishman, 1978; Gordon, 1980; Ripley, 1987; Bratley et al, 1987; Lewis y Orav, 1989).

El esfuerzo y la cantidad de tiempo requerido para construir los programas es bastante elevado (Naylor et al., 1982; thisted, 1988; Kleignen y Groenendaal, 1992).

b. Delphi como entorno de desarrollo.

Delphi está basado en una versión de Pascal denominada Object Pascal.

Object Pascal expande las funcionalidades del Pascal estándar:

Encapsulación: declarando partes privadas, protegidas, públicas y publicadas de las clases, Propiedades: concepto nuevo que luego han adaptado muchos otros lenguajes. Las propiedades permiten usar la sintaxis de asignación para setters y getters y simplificación de la sintaxis de referencias a clases y punteros.

Soporte para manejo estructurado de excepciones, mejorando sensiblemente el control de errores de usuario y del sistema.

Programación activada por eventos (event-driven), posible gracias a la técnica de delegación de eventos. Esta técnica permite asignar el método de un objeto para responder a un evento lanzado sobre otro objeto. Fue adoptada por Niklaus Wirth, autor del Pascal Original.

Delphi es una Two-Way-Tool, es decir una herramienta de dos direcciones, porque permite crear herramientas de dos formas: una de forma visual en la pantalla por medio de la función de arrastrar y colocar (Drag & Drop), la otra es a través de la programación convencional, escribiendo el código. Ambas técnicas pueden utilizarse de forma alternativa o simultánea.

b.1. Desarrollo de módulos estadísticos del Fistad.

Delphi soporta la programación orientada a objetos y posee un abanico de componentes VCL necesarios para el desarrollo de interfaces de usuario, cuadros de dialogo y librerías matemáticas estándares para el desarrollo de algoritmos estadísticos sin entrar en programación de bajo nivel.

b.2. Elementos de Programación.

En este apartado veremos el panorama software estadístico que facilite el entendimiento del código del software estadístico mostrando una visión general de la estructura de los algoritmos y su estructura.

b.2.1. Manejo de matrices

A parte de algoritmos computacionales expuestos en el apartado, en delphi es posible diseñar de forma nativa aplicaciones completas que funciones bajo Windows con interfaces de usuario gráfica, con un sistema de menús, íconos y cuadros de diálogo. (Joyanes, 2003).

3.2.3. Especificación de Diseño de Módulos.

El Software FisTad 1.0

En este apartado describiremos los distintos módulos que comprenden el software estadístico FisTad 1.0. La ejecución de los módulos se basa en los algoritmos explicados en el capítulo anterior. La exposición de cada módulo se realiza en dos fases: presentación del algoritmo creado a partir de las técnicas y métodos estadísticos y por último se brindará una explicación del papel de dicho algoritmo en el software estadístico.

DIAGRAMAS UML

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones. En concreto, un diagrama ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece

una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Uso para modelar los procesos 'business'.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

3.2.3.1. Casos de Uso.

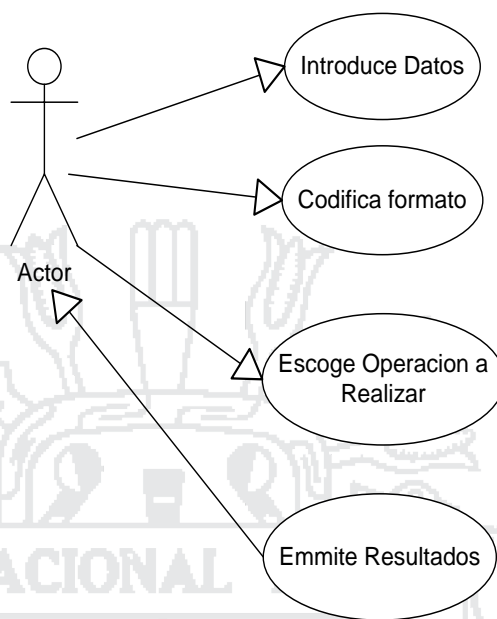


Fig. 8 Caso de uso

3.2.3.2. Diagramas de Clases

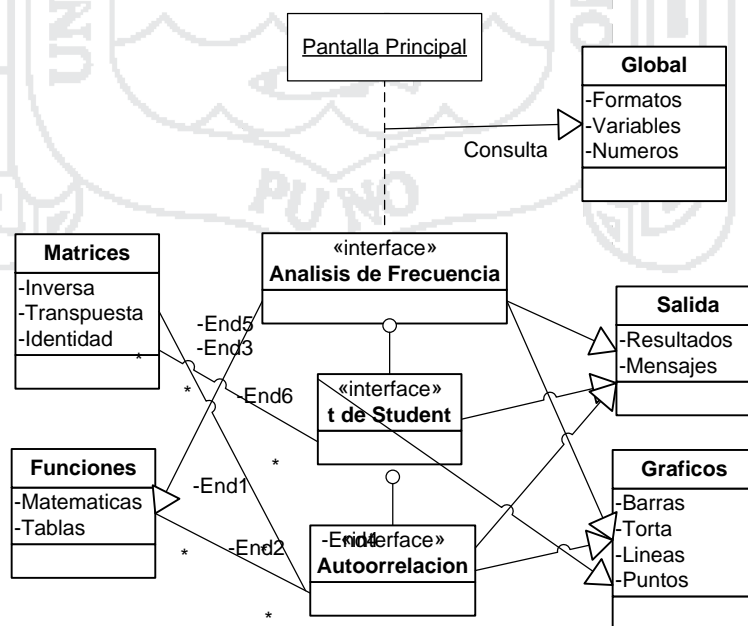


Fig. 9 Diagrama de secuencia de Fistad

3.2.3.3. Diagrama de Secuencia

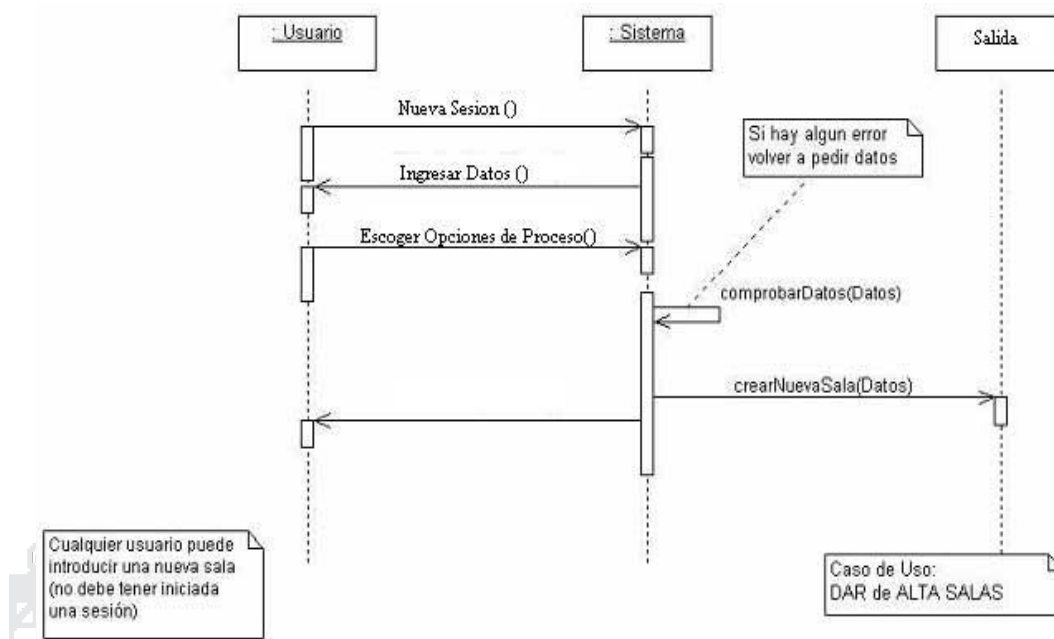
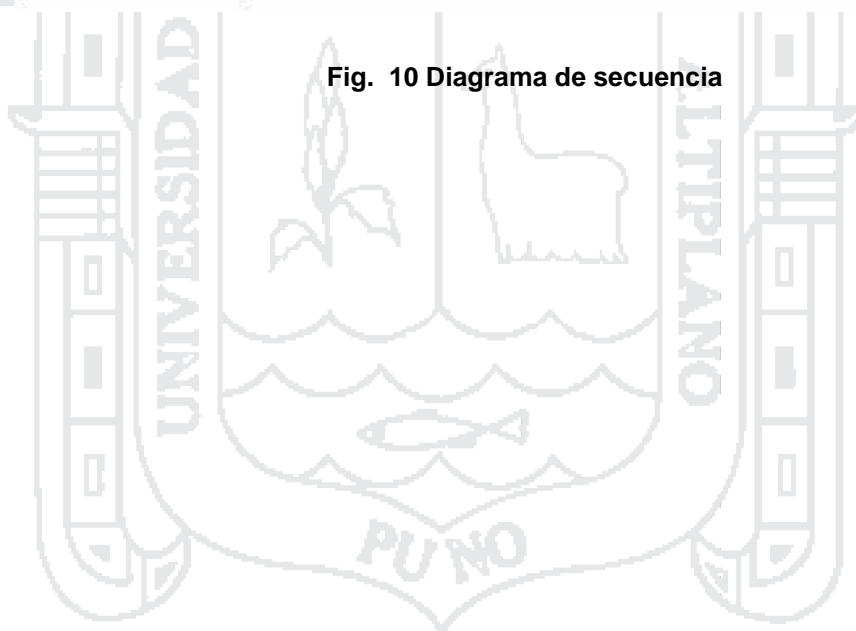


Fig. 10 Diagrama de secuencia



CAPITULO IV

RESULTADOS Y DISCUSIONES

Un software estadístico de reducidas dimensiones con el código fuente ayudará a entender los métodos y técnicas estadísticas, los estadísticos necesitan no solo conocer los métodos y técnicas estadísticas sino la algoritmia de dichos métodos para optimizar sus procesos de forma más eficiente que se haría con un software comercial, esto debido a los conocimientos de programación de los que disponen los estudiantes de la Escuela profesional de Ing. Estadística e Informática.

4.1. ALGORITMOS ESTADISTICOS

4.1.1. Chi Cuadrado

```
ChiCuadrado
var
  factor : doble; // factor con multiplies sumas de series
  g      : doble; // lngamma(k1+1)
  k1     : doble; // ajuste con grados de libertad
  sum    : doble; // Almacenamiento parcial de sumas
  term   : doble; // termino de series
  x1     : doble; // argumento de función ajustada
  chi2prob : doble; // Probabilidad chi-cuadrada
```

```

inicio
  si (X < 0.01) or (X > 1000.0) entonces
    inicio
      si X < 0.01 entonces chi2prob := 0.0001
      sino chi2prob := 0.999;
    fin
  sino
    inicio
      x1 := 0.5 * X;
      k1 := 0.5 * k;
      g := gammln(k1 + 1);
      factor := exp(k1 * ln(x1) - g - x1);
      sum := 0.0;
      si factor > 0 entonces
        inicio
          term := 1.0;
          sum := 1.0;
          mientras ((term / sum) > 0.000001) do
            inicio
              k1 := k1 + 1;
              term := term * (x1 / k1);
              sum := sum + term;
            fin;
          fin;
        chi2prob := sum * factor;
      fin; //fin si .. sino
    Result := chi2prob;
  fin;

```

4.1.2. Algoritmo de Matriz inversa

```

matinv (VAR a, vtimesw, v, w: hacerbleMat; n: entero);
//Descomposicion de matrices
LABEL 1,2,3;

VAR
  ainversa : array de array de hacerble;
  m,mp,np,nm,l,k,j,its,i: entero;
  z,y,x,scalar,s,h,g,f,c,anorm: hacerble;
  rv1: array de hacerble;

inicio
  setlength(rv1,n);
  setlength(ainversa,n,n);
  m := n;
  mp := n;
  np := n;
  g := 0.0;
  scalar := 0.0;
  anorm := 0.0;
  desde i := 0 hasta n-1 hacer
    inicio
      l := i+1;
      rv1[i] := scalar*g;
      g := 0.0;
      s := 0.0;

```

```

scalar := 0.0;
si (i <= m-1) entonces
inicio
  desde k := i hasta m-1 hacer
    inicio
      scalar := scalar+abs(a[k,i])
    fin;
si (scalar <> 0.0) entonces
  inicio
    desde k := i hasta m-1 hacer
      inicio
        a[k,i] := a[k,i]/scalar;
        s := s+a[k,i]*a[k,i]
      fin;
      f := a[l,i];
      g := -sign(sqrt(s),f);
      h := f*g-s;
      a[l,i] := f-g;
      si (l <> n-1) entonces
        inicio
          desde j := l hasta n-1 hacer
            inicio
              s := 0.0;
              desde k := i hasta m-1 hacer
                inicio
                  s := s+a[k,i]*a[k,j]
                fin;
              f := s/h;
              desde k := i hasta m-1 hacer
                inicio
                  a[k,j] := a[k,j]+
                    f*a[k,i]
                fin
              fin
            fin;
          desde k := i hasta m-1 hacer
            inicio
              a[k,i] := scalar*a[k,i]
            fin
          fin
        fin;
      w[l,i] := scalar*g;
      g := 0.0;
      s := 0.0;
      scalar := 0.0;
      si ((i <= m-1) y (i <> n-1)) entonces
        inicio
          desde k := l hasta n-1 hacer
            inicio
              scalar := scalar+abs(a[i,k])
            fin;
          si (scalar <> 0.0) entonces
            inicio
              desde k := l hasta n-1 hacer
                inicio
                  a[l,k] := a[l,k]/scalar;
                  s := s+a[l,k]*a[l,k]
                fin;
            fin;
          fin;
        fin;
      fin;
    fin;
  fin;

```

```

f := a[l,l];
g := -sign(sqrt(s),f);
h := f*g-s;
a[i,l] := f-g;
desde k := l hasta n-1 hacer
    inicio
    rv1[k] := a[l,k]/h
fin;
si (i <> m-1) entonces inicio
    desde j := l hasta m-1 hacer
        inicio
        s := 0.0;
        desde k := l hasta n-1 hacer
            inicio
            s := s+a[j,k]*a[l,k]
        fin;
        desde k := l hasta n-1 hacer
            inicio
            a[j,k] := a[j,k]
            +s*rv1[k]
        fin
    fin
fin;
desde k := l hasta n-1 hacer
    inicio
    a[i,k] := scalar*a[i,k]
fin
fin
fin;
anorm := max(anorm,(abs(w[l,i])+abs(rv1[i])))
fin;
desde i := n-1 hasta 0 hacer
    inicio
    si (i < n-1) entonces
        inicio
        si (g <> 0.0) entonces
            inicio
            desde j := l hasta n-1 hacer
                inicio
                v[j,i] := (a[l,j]/a[l,l])/g
            fin;
            desde j := l hasta n-1 hacer
                inicio
                s := 0.0;
                desde k := l hasta n-1 hacer
                    inicio
                    s := s+a[l,k]*v[k,j]
                fin;
                desde k := l hasta n-1 hacer
                    inicio
                    v[k,j] := v[k,j]+s*v[k,i]
                fin
            fin
        fin
    fin
fin;
desde j := l hasta n-1 hacer
    inicio
    v[l,j] := 0.0;
    v[j,i] := 0.0

```

```

    fin
  fin;
  v[l,i] := 1.0;
  g := rv1[i];
  l := i
fin;
desde i := n-1 hacerWNhasta 0 hacer
inicio
  l := i+1;
  g := w[l,i];
  si (i < n-1) entonces
  inicio
    desde j := l hasta n-1 hacer
    inicio
      a[j,i] := 0.0
    fin
  fin;
  si (g <> 0.0) entonces
  inicio
    g := 1.0/g;
    si (i <> n-1) entonces
    inicio
      desde j := l hasta n-1 hacer
      inicio
        s := 0.0;
        desde k := l hasta m-1 hacer
        inicio
          s := s+a[k,i]*a[k,j]
        fin;
        f := (s/a[i,i])*g;
        desde k := i hasta m-1 hacer
        inicio
          a[k,j] := a[k,j]+f*a[k,i]
        fin
      fin
    fin;
    desde j := i hasta m-1 hacer
    inicio
      a[j,i] := a[j,i]*g
    fin
  fin sino inicio
    desde j := i hasta m-1 hacer
    inicio
      a[j,i] := 0.0
    fin
  fin;
  a[i,i] := a[i,i]+1.0
fin;
desde k := n-1 hacerWNhasta 0 hacer
inicio
  desde its := 1 hasta 30 hacer
  inicio
    desde l := k hacerWNhasta 0 hacer
    inicio
      nm := l-1;
      si ((abs(rv1[l])+anorm) = anorm) entonces GOTO 2;
      si ((abs(w[nm,nm])+anorm) = anorm) entonces GOTO 1
    fin;

```



```

1:
  s := 1.0;
  desde i := l hasta k hacer
    inicio
      f := s*rv1[i];
      si ((abs(f)+anorm) <> anorm) entonces
        inicio
          g := w[l,i];
          h := sqrt(f*f+g*g);
          w[l,i] := h;
          h := 1.0/h;
          c := (g*h);
          s := -(f*h);
          desde j := 0 hasta m-1 hacer
            inicio
              y := a[j,nm];
              z := a[j,i];
              a[j,nm] := (y*c)+(z*s);
              a[j,i] := -(y*s)+(z*c)
            fin
          fin
        fin;
2:
  z := w[k,k];
  si (l = k) entonces
    inicio
      si (z < 0.0) entonces
        inicio
          w[k,k] := -z;
          desde j := 0 hasta n-1 hacer
            inicio
              v[j,k] := -v[j,k]
            fin
          fin;
  GOTO 3
  fin;
  si (its = 30) entonces
    inicio
      showMessage('No converge en 30 iteraciones');
      exit;
  fin;
  x := w[l,l];
  nm := k-1;
  y := w[nm,nm];
  g := rv1[nm];
  h := rv1[k];
  f := ((y-z)*(y+z)+(g-h)*(g+h))/(2.0*h*y);
  g := sqrt(f*f+1.0);
  f := ((x-z)*(x+z)+h*((y/(f+sign(g,f))-h))/x;
  c := 1.0;
  s := 1.0;
  desde j := l hasta nm hacer
    inicio
      l := j+1;
      g := rv1[i];
      y := w[l,i];
      h := s*g;
      g := c*g;
      z := sqrt(f*f+h*h);

```

```

rv1[j] := z;
c := f/z;
s := h/z;
f := (x*c)+(g*s);
g := -(x*s)+(g*c);
h := y*s;
y := y*c;
desde nm := 0 hasta n-1 hacer
    inicio
        x := v[nm,j];
        z := v[nm,i];
        v[nm,j] := (x*c)+(z*s);
        v[nm,i] := -(x*s)+(z*c)
    fin;
z := sqrt(f*f+h*h);
w[j,j] := z;
si (z <> 0.0) entonces
    inicio
        z := 1.0/z;
        c := f*z;
        s := h*z
    fin;
f := (c*g)+(s*y);
x := -(s*g)+(c*y);
desde nm := 0 hasta m-1 hacer
    inicio
        y := a[nm,j];
        z := a[nm,i];
        a[nm,j] := (y*c)+(z*s);
        a[nm,i] := -(y*s)+(z*c)
    fin
fin;
rv1[l] := 0.0;
rv1[k] := f;
w[k,k] := x
3: fin;
desde i := 0 hasta n-1 hacer
    desde j := 0 hasta n-1 hacer
        inicio
            si w[j,j] < 1.0e-6 entonces vtimesw[i,j] := 0
            sino vtimesw[i,j] := v[i,j] * (1.0 / w[j,j] );
        fin;
desde i := 0 hasta m-1 hacer
    desde j := 0 hasta n-1 hacer
        inicio
            ainversa[i,j] := 0.0;
            desde k := 0 hasta m-1 hacer
                inicio
                    ainversa[i,j] := ainversa[i,j] + vtimesw[i,k] * a[j,k]
                fin;
            fin;
desde i := 0 hasta n-1 hacer
    desde j := 0 hasta n-1 hacer
        a[i,j] := ainversa[i,j];
ainversa := nil;
rv1 := nil;
fin;

```

4.1.3. Algoritmo de Integral de Simsom

```

simpsonintegral(a,b : real) : real;
const error = 1.0e-4 ;

var h : real; (* longitud actual de intervalo *)
    i : integer; (* contador *)
    integral : real; (* aproximación actual a integral *)
    lastint : real; (* aproximacion previa*)
    n : integer; (* no.de intervalos *)
    sum1,sum2,sum4 : real; (* sumas de valores de la función *)

inicio
    n := 2 ; h := 0.5 * (b - a);
    sum1 := h * (zdensity(a) + zdensity(b) );
    sum2 := 0;
    sum4 := zdensity( 0.5 * (a + b));
    integral := h * (sum1 + 4 * sum4);
    repetir
        75asting := integral; n := n + n; h := 0.5*h;
        sum2 := sum2 + sum4;
        sum4 := 0; I := 1;
        repetir
            sum4 := sum4 + zdensity(a + i*h);
            I := I + 2
        hasta I > n;
        integral := h * (sum1 + 2*sum2 + 4*sum4);
        hasta abs(integral - lastint) < error;
        simpsonintegral := integral/3
    fin;

```

4.1.4. Algoritmo de Chi Inversa

```

inversachi(p : doble; k : ientero)
var
    a1, w, z : double;
begin
    z := inversez(p);
    a1 := 2.0 / ( 9.0 * k);
    w := 1.0 - a1 + z * sqrt(a1);
    Result := (k * w * w * w);
end;

```

4.1.5. Algoritmo de T de Student

```

funcion STUDENT(q,v,r : real) : real;

{ Rinde la probabilidad de un valor de la muestra de Q o mayo de una población
con medios y r grados de libertad para el erro medio cuadrado de v.}
var
    probq : real;
    ifault : entero;

```

```

funcion alnorm(x: real; upper: boleano): real;
var
  ltone, utzero, zero, half, one, con, z, y : real;
  up : boleano;
  altemp: real;
inicio
  ltone := 7.0;
  utzero := 18.66;
  zero := 0.0;
  half := 0.5;
  one := 1.0;
  con := 1.28;
  up := upper;
  z := x;
  si z < zero entonces
  inicio
    up := not up;
    z := -z;
  fin
  si (z <= ltone) o (up) y (z <= utzero) entonces
  inicio
    y := half * z * z;
    si z > con entonces
    inicio
      altemp := 0.398942280385 * exp(-y) /
        (z - 3.8052e-8 + 1.00000615302 /
        (z + 3.98064794e-4 + 1.98615381364 /
        (z - 0.151679116635 + 5.29330324926 /
        (z + 4.8385912808 - 15.1508972451 /
        (z + 0.742380924027 + 30.789933034 /
        (z + 3.99019417011))))));
    fin
      sino altemp := half - z * (0.398942280444 - 0.399903438504 * y /
        (y + 5.75885480458 - 29.8213557808 /
        (y + 2.62433121679 + 48.6959930692 /
        (y + 5.92885724438))));
    fin
      sino altemp := zero;
    si not up entonces altemp := one - altemp;
    alnorm := altemp;
  fin
fin

```

4.1.6. Algoritmo Logaritmo Gamma

```

funcion lngamma(w : real) : real;
constante a = 0.57236494;
var sum:real;
inicio
  sum := 0;
  w := w-1;
  mientras w > 0.0 do
  inicio
    sum := sum + ln(w);
    w := w - 1
  fin
  si w < 0.0

```

```

    entonces lngamma := sum + a
    sino lngamma := sum
fin

```

4.1.7. Algoritmo de ordenada

```

funcion ordinate(z : double) : double;
var pi : double;
inicio
    pi := 3.14159;
    Result := (1.0 / sqrt(2.0 * pi)) * (1.0 / exp(z * z / 2.0));
fin // Fin de funcion
//-----

procedure Rank(v1col : entero; VAR Values : DbldyneVec);
var
    pctiles, CatValues : DbldyneVec;
    freq : IntDyneVec;
    i, j, nocats : entero;
    Temp, cumfreq, upper, lower : double;
inicio
    SetLength(freq, NoCases);
    SetLength(pctiles, NoCases);
    SetLength(CatValues, NoCases);

    // obtener valores que se clasifican en los valores del vector
    fo i := 1 to NoCases do
        Values[i-1] := StrToFloat(PrincipalFrm.DataGrid.Cells[v1col,i]);

    // ordenar los valores
    fo i := 1 to NoCases - 1 do //ordenar del mas alto al mas bejo
        inicio
            fo j := i + 1 to NoCases do
                inicio
                    si (Values[i-1] < Values[j-1]) entonces // swap
                        inicio
                            Temp := Values[i-1];
                            Values[i-1] := Values[j-1];
                            Values[j-1] := Temp;
                        fin
                    fin
                fin
            fin
        fin

    // ahora obtener ninguna. valores de únicos y frecuencia
    nocats := 1;
    fo i := 1 to NoCases do freq[i-1] := 0;
    Temp := Values[0];
    CatValues[0] := Temp;
    fo i := 1 to NoCases do
        inicio
            si (Temp = Values[i-1]) entonces freq[nocats-1] := freq[nocats-1] + 1
            sino // nnuevo valores
                inicio
                    nocats := nocats + 1;
                    freq[nocats-1] := freq[nocats-1] + 1;
                fin
            fin
        fin
    fin

```

```

        Temp := Values[i-1];
        CatValues[nocats-1] := Temp;
    fin
fin

// obtener rangos
cumfreq := 0.0;
fo i := 1 to nocats do
inicio
    upper := NoCases-cumfreq;
    cumfreq := cumfreq + freq[i-1];
    lower := NoCases - cumfreq + 1;
    pcntiles[i-1] := (upper - lower) / 2.0 + lower;
fin
// convertir valores originales correspondiente al rango
fo i := 1 to NoCases do
inicio
    Temp := StrToFloat(PrincipalFrm.DataGrid.Cells[v1col,i]);
    fo j := 1 to nocats do
inicio
        si (Temp = CatValues[j-1]) entonces Values[i-1] := pcntiles[j-1];
    fin
fin
// limpiar
CatValues := nil;
pcntiles := nil;
freq := nil;
fin

```

4.1.8. Algoritmo punto de rango

```

procedure PRank(v1col : entero; VAR Values : DbldyneVec);
// calcula los rangos percentiles de los valores almacenados en la tabla de
// datos en la columna v1col
var
    pcntiles, cumfm, CatValues : DbldyneVec;
    freq, cumf : IntDyneVec;
    Temp : double;
    i, j, nocats : entero;
inicio
    SetLength(freq, NoCases);
    SetLength(pcntiles, NoCases);
    SetLength(cumf, NoCases);
    SetLength(cumfm, NoCases);
    SetLength(CatValues, NoCases);

    // obtener valores que se clasifican en los valores del vector
    fo i := 1 to NoCases do
        Values[i-1] := StrToFloat(PrincipalFrm.DataGrid.Cells[v1col,i]);

    // ordenar los valores
    fo i := 1 to NoCases - 1 do //orden de mayo a menor
    inicio
        fo j := i + 1 to NoCases do
            inicio

```

```

    si (Values[i-1] < Values[j-1]) entonces // swap
    inicio
        Temp := Values[i-1];
        Values[i-1] := Values[j-1];
        Values[j-1] := Temp;
    fin
fin
fin
nocats := 1;
fo i := 1 to NoCases do freq[i-1] := 0;
Temp := Values[0];
CatValues[0] := Temp;
fo i := 1 to NoCases do
inicio
    si (Temp = Values[i-1]) entonces freq[nocats-1] := freq[nocats-1] + 1
    sino // nuevo valo
    inicio
        nocats := nocats + 1;
        freq[nocats-1] := freq[nocats-1] + 1;
        Temp := Values[i-1];
        CatValues[nocats-1] := Temp;
    fin
fin
// encontrar nueva frecuencia acumulada
cumf[nocats-1] := freq[nocats-1];
fo i := nocats - 2 downto 1 do cumf[i-1] := freq[i-1] + cumf[i];
// conseguir frecuencia acumulada de puntos medios y rangos percentiles
cumfm[nocats-1] := freq[nocats-1];
pcntiles[nocats-1] := (cumf[nocats-1] / 2.0) / NoCases;
fo i := nocats - 2 downto 1 do
inicio
    cumfm[i-1] := (freq[i-1] / 2.0) + cumf[i];
    pcntiles[i-1] := cumfm[i-1] / NoCases;
fin
// convertir los valores originales a sus rangos percentiles correspondientes
fo i := 1 to NoCases do
inicio
    Temp := StrToFloat(PrincipalFrm.DataGrid.Cells[v1col,i]);
    fo j := 1 to nocats do
    inicio
        si (Temp = CatValues[j-1]) entonces Values[i-1] := pcntiles[j-1];
    fin
fin
fin

//limpiar el montón
CatValues := nil;
cumfm := nil;
cumf := nil;
pcntiles := nil;
freq := nil;
fin

```

4.1.9. Algoritmo de estadística básica

funcion UniStats(N : entero; VAR X : DblDyneVec; VAR z : DblDyneVec;

```

VAR Mean : double; VAR varianza : double; VAR SD : double;
VAR Skew : double; VAR Kurtosis : double; VAR SEmean : double;
VAR SESkew : double; VAR SEKurtosis : double; VAR min : double;
VAR max : double; VAR Range : double; VAR MissValue : string) :
entero;

```

VAR

```

NoGood : entero; // N° de buenos casos devuelto po la función
i : entero; // Índice de bucles
num, den, cases, sum, M2, M3, M4, deviation, devsqr : double;
valustr : string;

```

inicio

```

Mean := 0.0;
varianza := 0.0;
SD := 0.0;
Skew := 0.0;
Kurtosis := 0.0;
SEmean := 0.0;
SESkew := 0.0;
SEKurtosis := 0.0;
min := 1.0e20;
max := -1.0e20;
range := 0.0;
NoGood := 0;
sum := 0.0;
M2 := 0.0;
M3 := 0.0;
M4 := 0.0;

```

```
fo i := 0 to N-1 do
```

inicio

```

ValueStr := FloatToStr(X[i]);
si Trim(MissValue) = ValueStr entonces continue;
NoGood := NoGood + 1;
sum := sum + X[i];
varianza := varianza + (X[i] * X[i]);
si X[i] < min entonces min := X[i];
si X[i] > max entonces max := X[i];

```

fin

```
si NoGood > 0 entonces
```

inicio

```

Mean := sum / NoGood;
range := max - min;

```

fin

```
si NoGood > 1 entonces
```

inicio

```

varianza := varianza - (sum * sum) / NoGood;
varianza := varianza / (NoGood - 1);
SD := sqrt(varianza);
SEmean := sqrt(varianza / NoGood);

```

```
fo i := 0 to N-1 do
```

inicio

```

ValueStr := FloatToStr(X[i]);
si Trim(MissValue) = ValueStr entonces continue;
deviation := X[i] - Mean;
z[i] := deviation / SD;

```



```

        devsqr := deviation * deviation;
        M2 := M2 + devsqr;
        M3 := M3 + (deviation * devsqr);
        M4 := M4 + (devsqr * devsqr);
    fin
fin
si NoGood > 3 entonces
inicio
    Skew := (NoGood * M3) / ((NoGood - 1) * (NoGood - 2) * SD * varianza);
    num := 6.0 * NoGood * (NoGood - 1);
    den := (NoGood - 2) * (NoGood + 1) * (NoGood + 3);
    SESkew := sqrt(num / den);
    Kurtosis := (NoGood * (NoGood + 1) * M4) - (3.0 * M2 * M2 * (NoGood - 1));
    Kurtosis := Kurtosis / ((NoGood - 1) * (NoGood - 2) * (NoGood - 3) *
        (varianza * varianza));
    SeKurtosis := sqrt((4.0 * (NoGood * NoGood - 1) * (SESkew * SESkew)) /
        ((NoGood - 3) * (NoGood + 5)));
fin
Result := NoGood;
fin
//-----
fin.

```

4.1.10. Algoritmo que calcula los rangos percentiles

```

funcion PRank(v1col : entero; VAR Values : DblDyneVec);
// calcula los rangos percentiles de los valores almacenados en la tabla de
// datos en la columna v1col
var
    pctiles, cumfm, CatValues : DblDyneVec;
    freq, cumf : IntDyneVec;
    Temp : haceruble;
    i, j, nocats : entero;
inicio
    SetLength(freq, Nocasos);
    SetLength(pctiles, Nocasos);
    SetLength(cumf, Nocasos);
    SetLength(cumfm, Nocasos);
    SetLength(CatValues, Nocasos);
    // obtener valores que se clasifican en los valores del vector
    para i := 1 a Nocasos hacer
        Values[i-1] := StrToFloat(PrincipalFrm.DataGrid.Cells[v1col,i]);
    // ordenar los valores
    para i := 1 a Nocasos - 1 hacer //orden de mayo a menor
    inicio
        para j := i + 1 a Nocasos hacer
        inicio
            si (Values[i-1] < Values[j-1]) entonces // swap
            inicio
                Temp := Values[i-1];
                Values[i-1] := Values[j-1];
                Values[j-1] := Temp;
            fin
        fin
    fin
fin

```

```

nocats := 1;
para i := 1 a Nocasos hacer freq[i-1] := 0;
Temp := Values[0];
CatValues[0] := Temp;
para i := 1 a Nocasos hacer
inicio
  si (Temp = Values[i-1]) entonces freq[nocats-1] := freq[nocats-1] + 1
  sino // nuevo valo
  inicio
    nocats := nocats + 1;
    freq[nocats-1] := freq[nocats-1] + 1;
    Temp := Values[i-1];
    CatValues[nocats-1] := Temp;
  fin
fin
// encontrar nueva frecuencia acumulada
cumf[nocats-1] := freq[nocats-1];
para i := nocats - 2 hacerwnto 1 hacer cumf[i-1] := freq[i-1] + cumf[i];
// conseguir frecuencia acumulada de puntos medios y rangos percentiles
cumfm[nocats-1] := freq[nocats-1];
pcntiles[nocats-1] := (cumf[nocats-1] / 2.0) / Nocasos;
para i := nocats - 2 hacerwnto 1 hacer
inicio
  cumfm[i-1] := (freq[i-1] / 2.0) + cumf[i];
  pcntiles[i-1] := cumfm[i-1] / Nocasos;
fin
// convertir los valores originales a sus rangos percentiles correspondientes
para i := 1 a Nocasos hacer
inicio
  Temp := StrToFloat(PrincipalFrm.DataGrid.Cells[v1col,i]);
  para j := 1 a nocats hacer
  inicio
    si (Temp = CatValues[j-1]) entonces Values[i-1] := pcntiles[j-1];
  fin
fin
//limpiar el montón
CatValues := nil;
cumfm := nil;
cumf := nil;
pcntiles := nil;
freq := nil;
fin

```

4.2. Estructura del Programa FISTAD

4.2.1. Algoritmo del módulo principal

```

inicio
for i := 1 to noseleccionar hacer
inicio
  cellCadena := ListaC1.Items.Cadenas[i-1];
  for j := 1 to NoVariables hacer
    if cellCadena = FistadPrincipal.DataGrid.Cells[j,0] entonces seleccionar[i-1] := j;
  fin
fin
for j := 1 to noseleccionar hacer

```

```

inicio
  if Opcion1.Checked entonces
    inicio
    fin
  for i := 1 to NoCases hacer
    inicio
      if CaseChk.Checked entonces
        inicio
          if not ValidValue(i,seleccionar[j-1]) entonces continue;
        fin
        else if not GrabarOk(i,noseleccionar,seleccionar) entonces continue;
        if (value < min) entonces min := value;
        if (value > max) entonces max := value;
      fin
    if numcasos > 0 entonces
      inicio
        mean := sum / numcasos;
        range := max - min;
      fin
    if numcasos > 1 entonces
      inicio
        varianza := varianza - (sum * sum) / numcasos;
        varianza := varianza / (numcasos - 1);
        stddev := sqrt(varianza);
        semean := sqrt(varianza / numcasos);
      fin
    if numcasos > 3 entonces
      inicio
        for i := 1 to NoCases hacer
          inicio
            if CaseChk.Checked entonces
              inicio
                if not ValidValue(i,seleccionar[j-1]) entonces continue;
              fin
              else if not GrabarOk(i,noseleccionar,seleccionar) entonces continue;
              value := StrToFloat(FistadPrincipal.DataGrid.Cells[k,i]);
              if stddev > 0.0 entonces
                inicio
                  desviacion := value - mean;
                  devsqr := desviacion * desviacion;
                  M2 := M2 + devsqr;
                  M3 := M3 + (desviacion * devsqr);
                  M4 := M4 + (devsqr * devsqr);
                  z := (value - mean) / stddev;
                  if Opcion1.Checked entonces
                    inicio
                      cellCadena := format('%8.5f',[z]);
                      FistadPrincipal.DataGrid.Cells[NoVariables,i] := cellCadena;
                    fin
                  fin
                fin
              fin
            if numcasos > 2 entonces
              inicio
                skew := (numcasos * M3) /
                  ((numcasos - 1) * (numcasos - 2) * stddev * varianza);
                cases := numcasos;
                num := 6.0 * cases * (cases - 1.0);
                den := (cases - 2.0) * (cases + 1.0) * (cases + 3.0);
              fin
            fin
          fin
        fin
      fin
    fin
  fin

```

```

        seskew := sqrt(num / den);
    fin
    if numcasos > 3 entonces
    inicio
        kurtosis := (numcasos * (numcasos + 1) * M4) -
            (3 * M2 * M2 * (numcasos - 1));
        kurtosis := kurtosis /
            ((numcasos - 1) * (numcasos - 2) * (numcasos - 3) * (varianza * varianza));
        sekurtosis := sqrt( (4.0 * (numcasos * numcasos - 1) * (seskew * seskew) ) /
            ((numcasos - 3) * (numcasos + 5) ));
    fin
    fin
    fin

```

4.2.2. Algoritmo principal de Fistad

```

Inicio Frecuencias
Funcion Resetear
Funcion Cancelar
Funcion Entrada
Funcion Salida
Funcion ahacer
Funcion ahacerBien
var
i : entero;
inicio
    VarLista.Borrar;
    Lista1.Borrar;
    for i := 1 a NoVariables hacer
    inicio
        VarLista.Items.Añadir(FPrincipal.DataCeldas.Cells[i,0]);
    fin;
    Opciones2.ItemIndex := -1;
    BEntrada.Habilitar := true;
    BSalida.Habilitar := false;
    Opciones1.ItemIndex := -1;
    NormalP.Checked := false;
fin;
//-----
Funcion CFrecuencia.Cancelar
inicio
    FFrecuencia1.Hide;
fin;
//-----
Funcion CFrecuencia.Entrada
var
    index, i : entero;
inicio
    index := VarLista.Items.Count;
    i := 0;
    mientras i < index hacer
    inicio
        si (VarLista.Selected[i]) entonces
        inicio
            Lista1.Items.Añadir(VarLista.Items.Cadena[i]);
            VarLista.Items.Borrar(i);

```

```

        index := index - 1;
        i := 0;
    fin
    sino i := i + 1;
    fin;
    BSalida.Habilitar := true;
fin;
//-----
Funcion CFrecuencia.Salida
var
    index: entero;
inicio
    index := Lista1.ItemIndex;
    VarLista.Items.Añadir(Lista1.Items.Cadena[index]);
    Lista1.Items.Borrar(index);
    BEntrada.Habilitar := true;
    si Lista1.Items.Count = 0 entonces BSalida.Habilitar := false;
fin;
//-----
Funcion CFrecuencia.ahacer
var
    count, index : entero;
inicio
    count := VarLista.Items.Count;
    for index := 0 a count-1 hacer
        inicio
            Lista1.Items.Añadir(VarLista.Items.Cadena[index]);
        fin;
        VarLista.Borrar;
    fin;
//-----
Funcion CFrecuencia.ahacerBien
label again, cleanup;
var
    i, j, k : entero;
    freq : hacerblevecar;
    pcnt : hacerblevecar;
    cumpcnt : hacerblevecar;
    pcntilerank : hacerblevecar;
    cumfreq : hacerblevecar;
    XValor : hacerblevecar;
    Valor : hacerble;
    NoVars : entero;
    cellval : string;
    col : entero;
    min, max : hacerble;
    rango : hacerble;
    incrTam : hacerble;
    nointervals : hacerble;
    nints : entero;
    Salidalinea : string;
    NoSelected : entero;
    NormDist : boolean;
    Histograma : boolean;
    Sumx, Sumx2, media, Varianza, StdDev, zlow, zhi : hacerble;
    X, zproplow, zprophi, zfreq : hacerble;
    Ncases : entero;
inicio

```

```

si Opciones1.ItemIndex = 1 entonces Histograma := true sino Histograma := false;
si NormalP.Checked = true entonces NormDist := true sino NormDist := false;
ConjuntoLen(freq,NoCasos);
ConjuntoLen(pcnt,NoCasos);
ConjuntoLen(cumpcnt,NoCasos);
ConjuntoLen(pcntilerank,NoCasos);
ConjuntoLen(cumfreq,NoCasos);
ConjuntoLen(XValor,NoCasos);

FSalida.TextoEdit.Borrar;
FSalida.TextoEdit.lineas.Añadir('ANALISIS DE FRECUENCIA');
FSalida.TextoEdit.lineas.Añadir("");
FSalida.TextoEdit.ParaGraph.Alignment := taLeftJustsiy;
NoVars := Lista1.Items.Count;
for i := 1 a NoVars hacer
inicio
col := 1;
cellval := Lista1.Items.Cadena[i-1];
for j := 1 a NoVariables hacer
inicio
si FPrincipal.DataCeldas.Cells[j,0] = cellval entonces
inicio
col := j;
Salidalinea := format('Análisis de Frecuencia para %s',cellval);
FSalida.TextoEdit.lineas.Añadir(Salidalinea);
break;
fin;
fin;
NoSelected := 1;
min := 1.0e32;
max := -1.0e32;
for j := 1 a NoCasos hacer
inicio
si Not ValidValor(j,col) entonces continue;
Valor := StraFloat(FPrincipal.DataCeldas.Cells[col,j]);
si Valor > max entonces max := Valor;
si Valor < min entonces min := Valor;
fin;
rango := max - min + 1.0;
incrTam := 1.0;
si rango > 200.0 entonces incrTam := rango / 15;
nointervals := rango / incrTam;
nints := round(nointervals);
FFrecuenciaesperada.VarName.Text := cellval;
FFrecuenciaesperada.Minimum.Text := FloataStr(min);
FFrecuenciaesperada.Maximum.Text := FloataStr(max);
FFrecuenciaesperada.rango.Text := FloataStr(rango);
FFrecuenciaesperada.IntTam.Text := FloataStr(incrTam);
FFrecuenciaesperada.NoInts.Text := IntaStr(nints);
again: FFrecuenciaesperada.MostrarMohacer;
incrTam := StraFloat(FFrecuenciaesperada.IntTam.Text);
nointervals := StraFloat(FFrecuenciaesperada.NoInts.Text);
nints := round(nointervals);
si nints+1 > NoCasos entonces
inicio
ShowMessage('ERROR! N ° de intervalos que no puede ser > q no. de los casos!');
goa again;
fin;

```

```

si nints > 200 entonces
inicio
  nints := 200;
  Application.MessageBox('Max. incremenas ajusta hacers a 200','Maximo
Excedihacer!',MB_OK);
fin;
{Ahora, obtener la frecuencia de casos en cada intervalo }
for j := 1 a nints+1 hacer freq[j-1] := 0;
Ncases := 0;
for j := 1 a NoCasos hacer
inicio
  si Not ValidValor(j,col) entonces continue;
  Ncases := Ncases + 1;
  Valor := StraFloat(FPrincipal.DataCeldas.Cells[col,j]);
  for k := 1 a nints hacer
  inicio
    si (Valor >= min + ((k-1) * incrTam)) y
      (Valor < min + (k * incrTam)) entonces freq[k-1] := freq[k-1] + 1;
  fin;
fin;
for j := 1 a nints+1 hacer XValor[j-1] := min + (j-1) * incrTam;
{ obtener frecuencias acumuladas y porcentajes a punas medios}
cumfreq[0] := freq[0];
pcnt[0] := freq[0] / Ncases;
cumpcnt[0] := cumfreq[0] / Ncases;
pcntilerank[0] := (freq[0] / 2.0) / Ncases;
for k := 2 a nints hacer
inicio
  cumfreq[k-1] := cumfreq[k-2] + freq[k-1];
  pcnt[k-1] := freq[k-1] / Ncases;
  cumpcnt[k-1] := cumfreq[k-1] / Ncases;
  pcntilerank[k-1] := (cumfreq[k-2] + freq[k-1] / 2.0) / Ncases;
fin;
{ Imprimir resultahacers }
FSalida.TextoEdit.lineas.Añadir('DE A FREQ. PCNT CUM.FREQ. CUM.PCNT. %ILE
RANG');
FSalida.TextoEdit.lineas.Añadir('');
for k := 1 a nints hacer
inicio
  Salidalineaa := format('%8.2f%8.2f%8.0f%8.2f %8.2f %8.2f %8.2f',
  [min+(k-1)*incrTam, // de
  min+k*incrTam, // a
  freq[k-1], // freq
  pcnt[k-1], // pcnt
  cumfreq[k-1], // cum.freq.
  cumpcnt[k-1], // cum.pcnt.
  pcntilerank[k-1]]); // %ile rango
  FSalida.TextoEdit.lineas.Añadir(Salidalineaa);
fin;
FSalida.MostrarMohacer;
FSalida.TextoEdit.Borrar;
{ la trama Valores como se indica en la lista de opciones }
si Histograma = true entonces FGrafico.Barprop := 1.0 sino
  FGrafico.Barprop := 0.5;
si NormDist = true entonces FGrafico.nosets := 2 sino FGrafico.nosets := 1;
FGrafico.nbarras := nints+1;
FGrafico.Titulo := cellval;
FGrafico.XTitle := 'Valores más bajos del límite';

```



```

FGrafico.YTitle := 'Frecuencia';
si NormDist = false entonces
  ConjuntoLen(FGrafico.Ypoints,1,nints+1)
sino ConjuntoLen(FGrafico.Ypoints,2,nints+1);
ConjuntoLen(FGrafico.Xpoints,1,nints+1);
for k := 1 a nints+1 hacer
  inicio
    FGrafico.Ypoints[0,k-1] := freq[k-1];
    FGrafico.Xpoints[0,k-1] := XValor[k-1];
  fin;
si NormDist = true entonces
  inicio
    FSalida.TextoEdit.lineas.Añadir('Intervalos ND Freq. ');
    // Sólo utilizar parcelas 3D cuanhacer curva normal funciona
    Opciones2.ItemIndex := 3;
    //conseguir media y desviación estándar de xValor, de la curva normal y z
    sumx := 0.0;
    sumx2 := 0.0;
    for k := 1 a nints hacer
      inicio
        sumx := sumx + (XValor[k-1] * freq[k-1]);
        sumx2 := sumx2 + ((XValor[k-1] * XValor[k-1]) * freq[k-1]);
      fin;
    media := sumx / Ncases;
    Varianza := sumx2 - ((sumx * sumx) / Ncases);
    Varianza := Varianza / (Ncases - 1);
    StdDev := sqrt(Varianza);
    for k := 1 a nints+1 hacer
      inicio
        X := XValor[k-1] - (incrTam / 2.0);
        si StdDev > 0.0 entonces zlow := (X - media) / StdDev
        sino zlow := 0.0;
        X := XValor[k-1] + (incrTam / 2.0);
        si StdDev > 0.0 entonces zhi := (X - media) / StdDev
        sino zhi := 0.0;
        // cuanhacer consigue la z en frecuencia
        zproplow := probz(zlow);
        zprophi := probz(zhi);
        zfreq := NoCasos * abs(zprophi - zproplow);
        FGrafico.Ypoints[1,k-1] := zfreq;
        Salidalinea := format(' %2d %6.2f',[k,FGrafico.Ypoints[1,k-1]]);
        FSalida.TextoEdit.lineas.Añadir(Salidalinea);
      fin;
    FSalida.MostrarMohacer;
    FSalida.TextoEdit.Borrar;
  fin;
si Opciones2.ItemIndex = 2 entonces // 2D vertical barras
  inicio
    { introducir parámetros para grafico de barras 2d }
    FGrafico.GraficoTipo := 1; // 2d barras
    FGrafico.AuaEscala := true;
    FGrafico.ShowLeftWall := true;
    FGrafico.ShowRightWall := true;
    FGrafico.ShowBotamWall := true;
    FGrafico.MostrarVolverMuro := true;
    FGrafico.ColorNegro := clYellow;
    FGrafico.WallColor := clBlack;
    FGrafico.MostrarMohacer;
  fin;

```

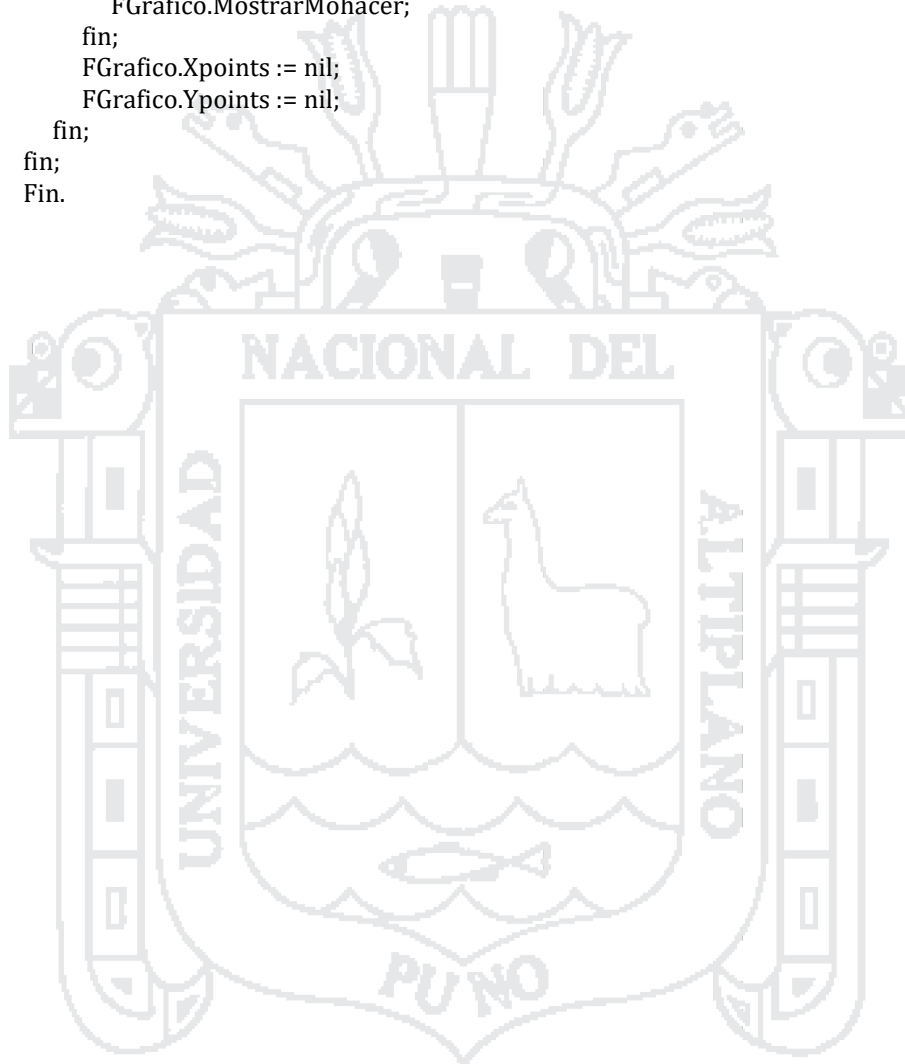


```

fin;
si Opciones2.ItemIndex = 6 entonces // 2D linea arta
inicio
  FGrafico.AuaEscala := true;
  FGrafico.GraficoTipo := 5; // 2d lineas
  FGrafico.ColorNegro := clYellow;
  FGrafico.MostrarVolverMuro := true;
  FGrafico.MostrarMohacer;
fin;
si Opciones2.ItemIndex = 7 entonces // 3D linea Graph
inicio
  FGrafico.AuaEscala := true;
  FGrafico.GraficoTipo := 6; // 3d lineas
  FGrafico.ColorNegro := clYellow;
  FGrafico.MostrarVolverMuro := true;
  FGrafico.MostrarMohacer;
fin;
si Opciones2.ItemIndex = 8 entonces
inicio
  FGrafico.AuaEscala := true;
  FGrafico.GraficoTipo := 7; // 2d points
  FGrafico.ColorNegro := clYellow;
  FGrafico.MostrarVolverMuro := true;
  FGrafico.MostrarMohacer;
fin;
si Opciones2.ItemIndex = 4 entonces
inicio
  FGrafico.AuaEscala := true;
  FGrafico.GraficoTipo := 3; // 2d pie arta
  FGrafico.ColorNegro := clYellow;
  FGrafico.MostrarVolverMuro := true;
  FGrafico.MostrarMohacer;
fin;
si Opciones2.ItemIndex = 5 entonces
inicio
  FGrafico.AuaEscala := true;
  FGrafico.GraficoTipo := 4; // despiece arta
  FGrafico.ColorNegro := clYellow;
  FGrafico.MostrarVolverMuro := true;
  FGrafico.MostrarMohacer;
fin;
si Opciones2.ItemIndex = 0 entonces
inicio
  FGrafico.AuaEscala := true;
  FGrafico.GraficoTipo := 9; // 2d
  FGrafico.ColorNegro := clYellow;
  FGrafico.MostrarVolverMuro := true;
  FGrafico.MostrarMohacer;
fin;
si Opciones2.ItemIndex = 3 entonces
inicio
  FGrafico.AuaEscala := true;
  FGrafico.GraficoTipo := 2; // 3d
  FGrafico.ColorNegro := clYellow;
  FGrafico.WallColor := clBlack;
  FGrafico.FloorColor := clLtGray;
  FGrafico.MostrarVolverMuro := true;
  FGrafico.MostrarMohacer;

```

```
fin;  
si Opciones2.ItemIndex = 1 entonces  
inicio  
    FGrafico.AuaEscala := true;  
    FGrafico.GraficoTipo := 10; // 3d Barras  
    FGrafico.ColorNegro := clYellow;  
    FGrafico.WallColor := clBlack;  
    FGrafico.FloorColor := clLtGray;  
    FGrafico.MostrarVolverMuro := true;  
    FGrafico.MostrarMohacer;  
fin;  
FGrafico.Xpoints := nil;  
FGrafico.Ypoints := nil;  
fin;  
fin;  
Fin.
```



CONCLUSIONES

PRIMERA.- Los algoritmos estadísticos cumplen la función de instruir a los usuarios en el manejo de procesos estadísticos, de los cuales el estudiante podrá tener una mejor idea al ver el programa que ejecuta los procedimientos matemáticos necesarios para resolver los distintos postulados estadísticos.

SEGUNDA.- En el presente software se ha desarrollado algoritmos computacionales que procesan postulados de estadística básica en un lenguaje de programación.

TERCERA.- Se ha integrado los algoritmos computacionales en un entorno de procesamiento de datos de forma similar a un software estadístico comercial, ya que actualmente se requiere el uso de computadores para la formación de los estudiantes de la Escuela Profesional de Ing. Estadística e Informática no solo desde el punto de vista de usuario de software estadístico, sino debe de estar en capacidad de desarrollar módulos adecuados a sus necesidades usando un lenguaje de programación.

CUARTA.- Se presenta un software con funciones básicas para ser analizado por los interesados y puedan implementar mas funciones, mejorar los algoritmos, etc.

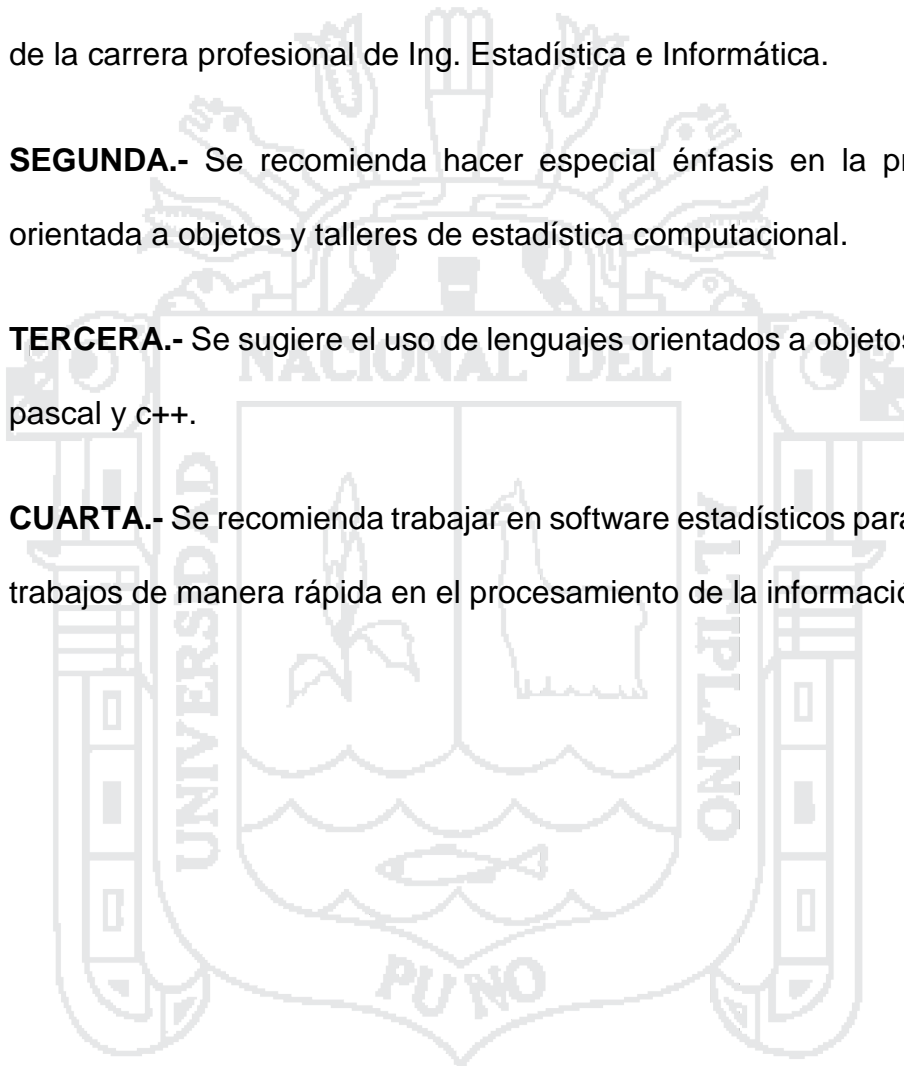
RECOMENDACIONES

PRIMERA.- Se sugiere incluir el dictado de cursos de algoritmia estadística en los cursos de programación, así como incluir programación en los cursos de estadística para poder lograr una cohesión entre ambas especialidades de la carrera profesional de Ing. Estadística e Informática.

SEGUNDA.- Se recomienda hacer especial énfasis en la programación orientada a objetos y talleres de estadística computacional.

TERCERA.- Se sugiere el uso de lenguajes orientados a objetos como java, pascal y c++.

CUARTA.- Se recomienda trabajar en software estadísticos para realizar así trabajos de manera rápida en el procesamiento de la información.



BIBLIOGRAFÍA

- Allen L. Webster (2000), *Estadística Aplicada a los Negocios y a la Economía*, Editorial McGraw-Hill
- Burda, Faerber (1996), *El gran libro de Delphi*, Marcombo, S.A., 1996.
- David C Lay, Jesús Murrieta Murrieta (2007), *Álgebra lineal y sus aplicaciones*, (Tercera Edición) Pearson Educacion
- David Moore (2006), *Estadística Aplicada Basica*, Antoni Bosch editor.
- Dirk Louis, José Luis Cortés (2000), *Delphi 5*, Alfaomega Marcombo.
- Francisco Charte Ojeda (2003), *DELPHI 7 GUIA PRÁCTICA*, Anaya Multimedia.
- Gary Cornell (1996), *Delphi para programadores*, McGraw-Hill/Interamericana de Espana, S.A.
- Ian Marteens (2003), *La cara oculta de Delphi 6*, Danysoft Internacional.
- Ibañez Quispe Vladimiro (1998), *Modelos Lineales*, Universidad Nacional del Altiplano – Facultad de Ingeniería Estadística e Informática.
- J.M. Juran, R.S. Bingham, Frank M. Gryna, José María Vallhonrat (1983), *Manual de control de la calidad*, Editorial Reverté.
- Jaime Serret Moreno-Gil (1995), *Manual de Estadística Universitaria*, ESIC Editorial.
- Jeff Sutherland (2007), *Pretty Good Scrum: Secret Sauce For Distributed Teams*, Scrum Trained Institute.

Julián de la Horra Navarro (2003), *Estadística Aplicada*, Ediciones Díaz de Santos.

Ludovic Dubois (2000), *Delphi: Desarrollo rápido de aplicaciones bajo Windows*, Gestión 2000.

M. Otero, José María Otero Moreno (1993), *Series temporales y predicción*, Editorial AC.

Marco Cantu, Eduardo Gómez Melguizo (2005), *La Biblia de Delphi 7*, Anaya Multimedia.

Martínez Ciro Bergamino (1996), *Estadística y Muestreo*. Editorial Medellín.

Mitacc Meza Máximo (1998), *Tópicos de Estadística Descriptiva y Probabilidades*.

Palomino Quispe, Platón (1996), *Diseños y Técnicas de Investigación* (1er edición), Titicaca FCEDUR-UNA-Puno.

Naresh K. Malhotra, José Francisco Javier Dávila Martínez (2004), *Investigación de Mercados*, Pearson Educación.

Rising Linda, Janoff Norman S. (2000) *The Scrum Software Development Process for Small Teams*, AG Communication Systems.

David S Rubin , Levin Richard I. (1998), *Statistics For Management*, Pearson Educación.

Pressman Roger S. (1998) *Ingeniería del software: Un enfoque práctico*. Mc Graw Hill, 4ta. Edición.

Sharon L. Lohr, scar Alfredo (trad.) Palmas Velasco, Carlos (rev.) Martínez Reyes (2000), *Muestreo: Diseño y análisis*, International Thomson.

Sixto Rios (1992), *Iniciación estadística*, Editorial Paraninfo.

Sutherland Jeff y Schwaber Ken (2010), *The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework*, Scrum Training Institute.

Zea Flores Wilfredo, (s.f.) *Estadística y Diseños Experimentales*, Universidad Nacional del Altiplano – Facultad de Ciencias Agrarias.

WEBGRAFIA

Algoritmos. Wikipedia, la enciclopedia libre, disponible en web.

<http://es.wikipedia.org/wiki/Algoritmos>

Codigo Fuente. Wikipedia, la enciclopedia libre, disponible en web.

<http://es.wikipedia.org/wiki/Algoritmos>.

Métricas, Estimación y Planificación en Proyectos de Software. Universidad de Guadalajara [ref. de 16 de agosto 2004]. Disponible en Web:

http://www.willydev.net/descargas/willydev_planeasoftware.pdf.

Metodología Métrica Versión 3, Ministerio de Administraciones Públicas (MAP), 2001, Disponible en web:

<http://www.map.es/csi/metrica3>.

Minitab. Wikipedia, la enciclopedia libre, disponible en web.

<http://es.wikipedia.org/wiki/Minitab>

Moto-Oka y Kitsuregawa. *Algoritmia*. Universidad de Tier. [ref. Octubre 1986].

Disponible en Web:

<http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/k/Kitsuregawa:Masaru.html>

PSPP. Fundación para el software libre (free software fundation), disponible en web.

<http://www.gnu.org/software/pspp/pspp.html>.

SAS. Wikipedia, la enciclopedia libre, disponible en web.

<http://es.wikipedia.org/wiki/SAS>

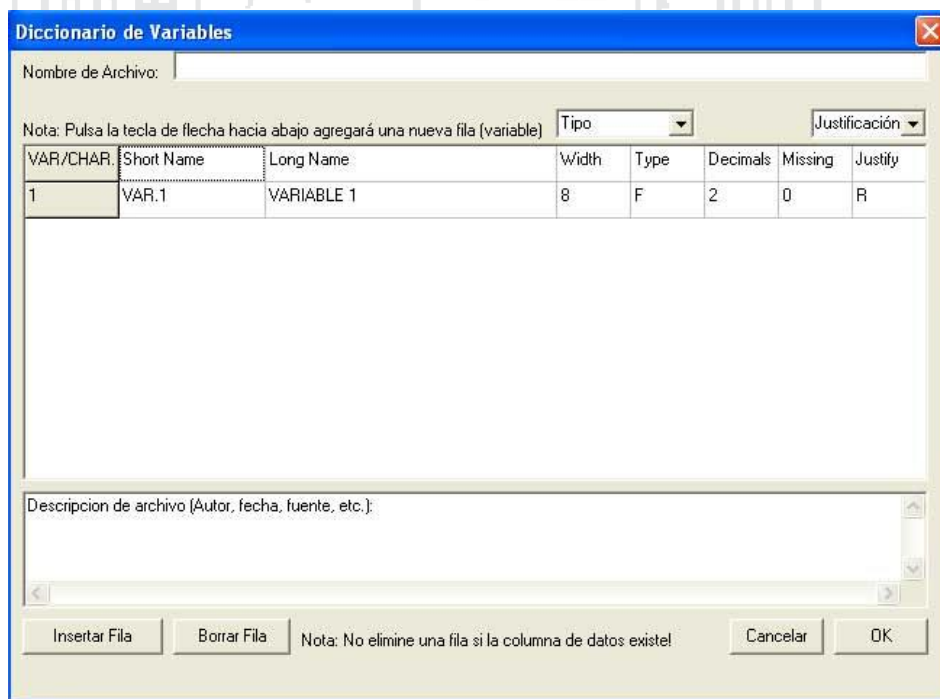
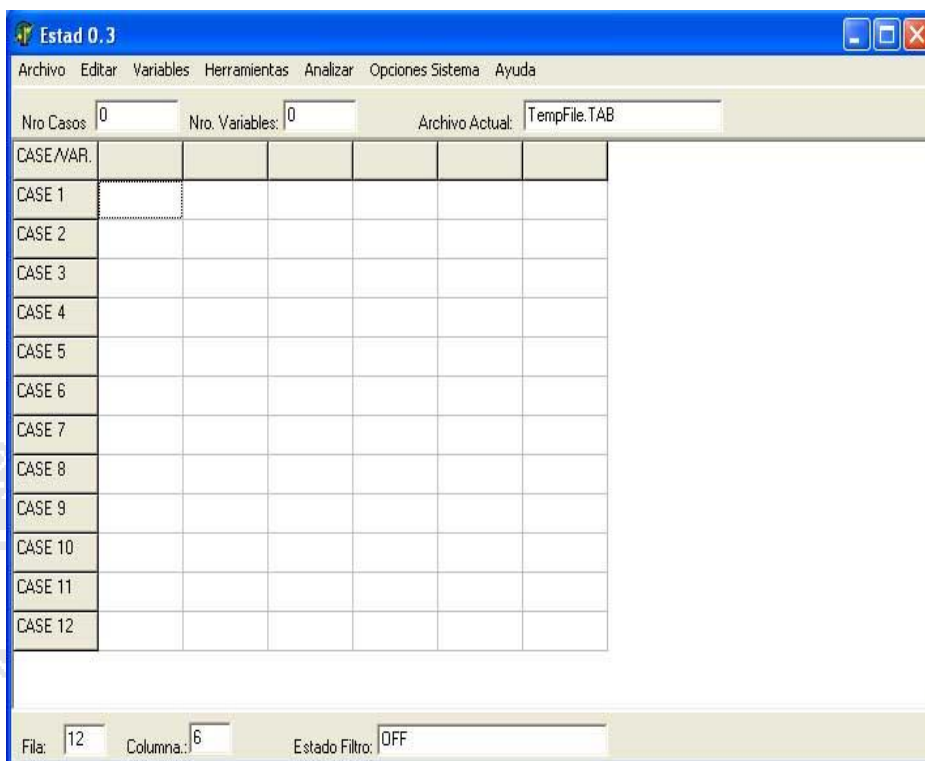
SPSS. Wikipedia, la enciclopedia libre, disponible en web.

<http://es.wikipedia.org/wiki/SPSS>



ANEXOS Nº 01

Imágenes del software



Analisis de Frecuencia

Variables Disponibles: VAR.2

Variables a Analizar: VAR.1

Opciones Gráficas:

- Barras Horizontales 2D
- Barras Horizontales 3D
- Barras Verticales 2D
- Barras Verticales 3D
- Gráfico de Torta 2D
- Gráfico de Torta Separado
- Línea Gráfica 2D
- Línea Gráfica 3D
- Puntos

Tipo de Gráfico:

- Bar Chart
- Histogram

Grat. Dist. Norm. Reseteo Cancel OK

CASE 7	43.00	7.00
CASE 8	8.00	8.00
CASE 9	56.00	9.00
CASE 10	57.00	8.00
CASE 11	38.00	7.00
CASE 12	89.00	3.00
CASE 13	45.00	6.00
CASE 14	67.00	7.00

Fila: 12 Columna: 1 Estado Filtro: OFF

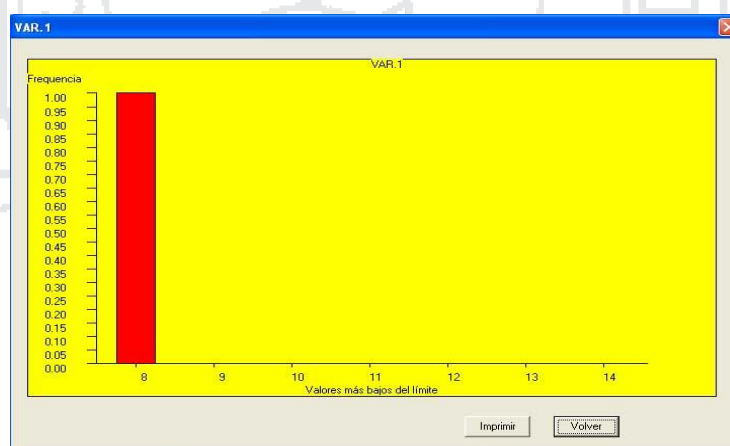
Salida de Archivos

ANALISIS DE FRECUENCIA

Analisis de Frecuencia para VAR.1

DE	A	FREQ.	PCNT	CUM.FREQ.	CUM.PCNT.	%ILE RANG
8.00	9.00	1	0.07	1.00	0.07	0.04
9.00	10.00	0	0.00	1.00	0.07	0.07
10.00	11.00	0	0.00	1.00	0.07	0.07
11.00	12.00	0	0.00	1.00	0.07	0.07
12.00	13.00	0	0.00	1.00	0.07	0.07
13.00	14.00	0	0.00	1.00	0.07	0.07

Imprimir Volver



Muestreo Simple

Estadística de Interés:

- Muestra de Media
- Muestra de Proporciones
- Muestra de Correlación
- Muestra de Varianza

Muestra Estadística:

Parámetros de Pob.:

Tamaño de Muestra:

Nivel de Confianza (%): 95.0

Muestra Std. Dev.:

Resetear Cancelar Ok

Mostrar variables.

Diccionario de Variables

Nombre de Archivo:

Nota: Pula la tecla de flecha hacia abajo agregará una nueva fila (variable) Tipo Justificación

VAR/CHAR.	Short Name	Long Name	Width	Type	Decimals	Missing	Justify
1	VAR.1	VARIABLE 1	8	F	2	0	R
2	VAR.2	VARIABLE 2	8	F	2	0	R

File Description (Author, Date, source, etc.):

Insertar Fila Borrar Fila Nota: No elimine una fila si la columna de datos existe! Cancelar Ok

Tenemos la posibilidad de configurar los puntos decimales.

Opciones de Sistema

Para Fracciones Decimales Use:

- Estandar Norteamericano (punto por defecto)
- Estandar Europeo (coma)

Localización de Archivo por Defecto:

Valores Perdidos por Defecto:

- (blanco)
- . (periodo)
- 0 (cero)

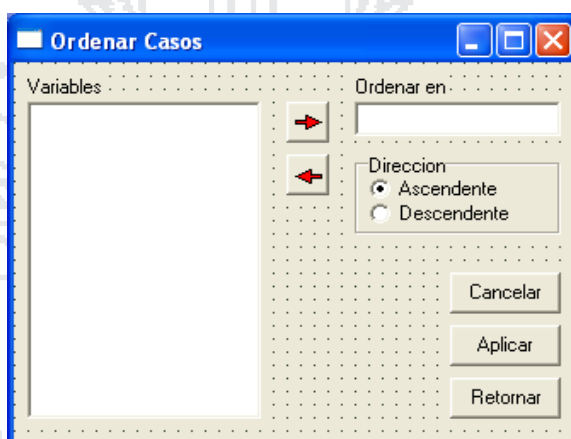
Justificación de Celdas:

- Izquierda
- Centro
- Derecha

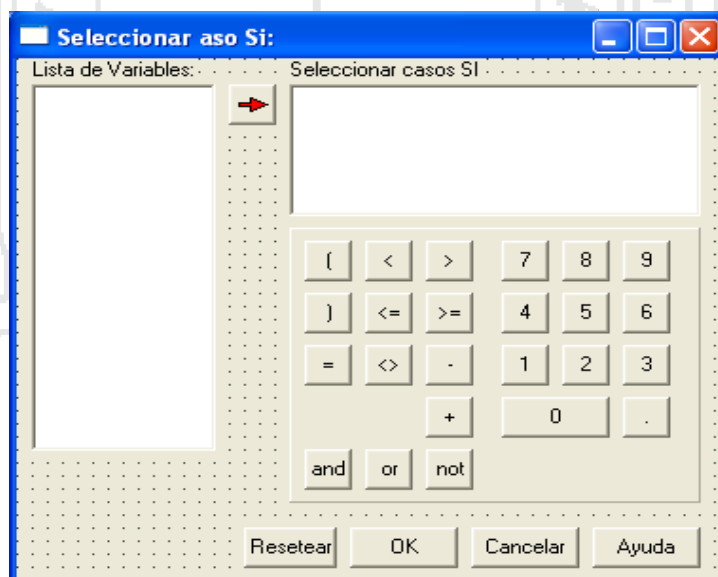
Cancelar Ok

Editar celdas, desde fistad tenemos la posibilidad de editar las celdas, copiar, cortar y pegar los datos existentes en las celdas, asi como almacenar nuestros datos para posteriores trabajos.

Ordenar celdas. Para ordenar celdas nos vamos al menú Herramientas → Ordenar casos.



Podemos ordenar celdas, incrementar un número determinado a una celda y hacer demás operaciones con celdas de forma similar a un programa estadístico comercial.



ANEXO Nº 02

Ejemplos con FisTad.

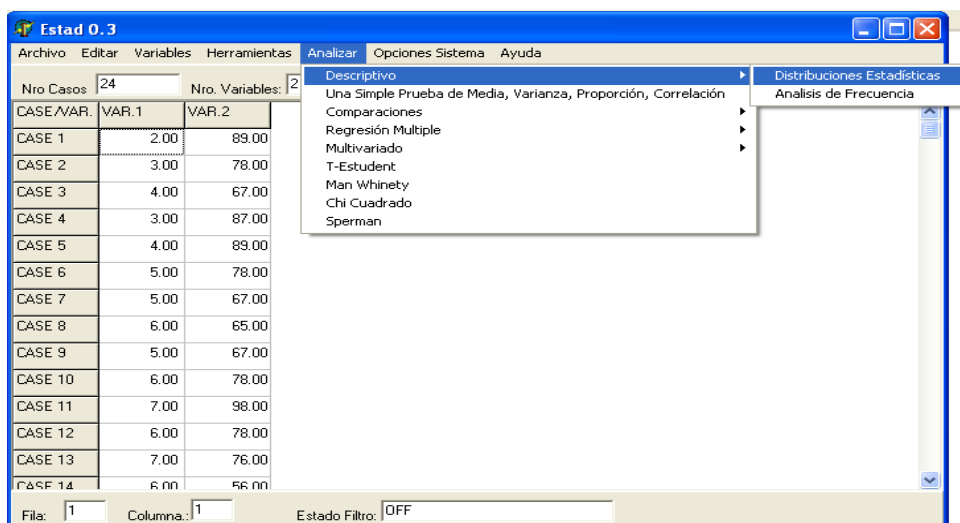
ANÁLISIS DESCRIPTIVO

MEDIDAS DE TENDENCIA CENTRAL, DE VARIABILIDAD Y DE DISPERSIÓN

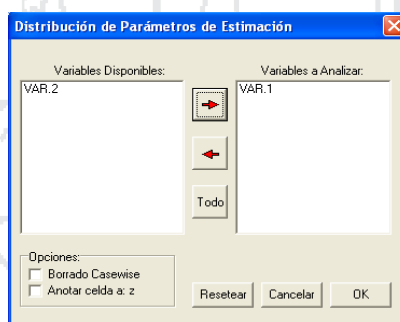
Una vez que la base de datos se ha importado conforme lo establecido, y además de ello las variables fueron definidas según las indicaciones del apartado anterior, se puede comenzar un análisis estadístico de los datos a nivel descriptivo. Para dicho fin se usará como ejemplo la base de datos “Autoestima y ansiedad”.

Nos ubicamos en **Analisis → Descriptivo → Distribuciones estadísticas**. En el cuadro de diálogo Estadística Descriptiva, hay las opciones para hallar, según la variable observada, la suma de sus valores (**Suma simple, Suma**), Media (**Media**), Varianza (**Var.**) y Desviación Estándar (**Std. Dev.**), Error estándar de la media (**Std. Error de media**), Intervalo de confianza (**Confidencialidad intervalica**), entre otras. Entonces según al análisis que pretendamos hacer seleccionamos las variables, así como los estadísticos que vamos a estudiar.

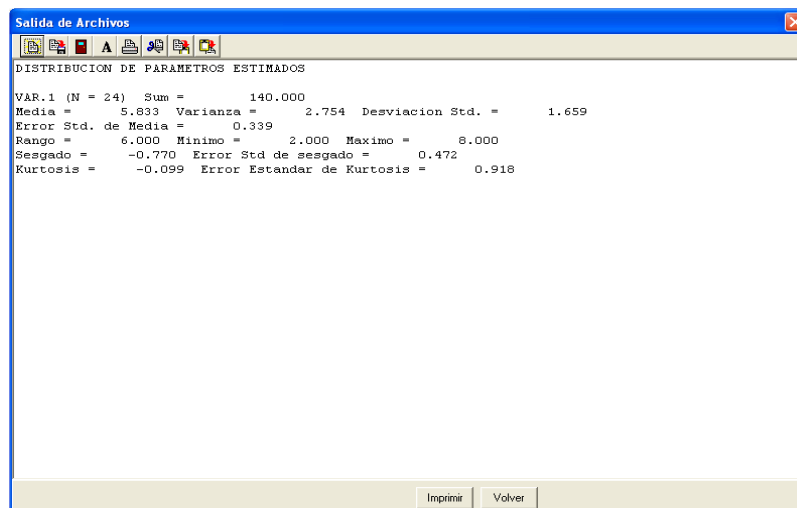
Ingresamos los datos a procesar en forma secuencial.



Una vez escogida la variable a la que debemos analizar pulsamos OK para proceder con el procesamiento de las operaciones.



Procedemos a visualizar los resultados de nuestra operación.



Anexo N° 03.

```

Codigo fuente.
unit FisTadMainUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, ExtCtrls, Menus, Globals, ActnList, StdCtrls;

type
  TFisTadMainFrm = class(TForm)
    MainMenu1: TMainMenu;
    Files1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
    Panel1: TPanel;
    Panel2: TPanel;
    DataGrid: TStringGrid;
    ActionList1: TActionList;
    FileNameEdit: TEdit;
    Label1: TLabel;
    SaveDialog1: TSaveDialog;
    OpenFileDialog1: TOpenDialog;
    Label2: TLabel;
    NoCasesEdit: TEdit;
    Label3: TLabel;
    NoVarsEdit: TEdit;
    Label4: TLabel;
    RowEdit: TEdit;
    Label5: TLabel;
    ColEdit: TEdit;
    Label6: TLabel;
    FilterEdit: TEdit;
    OpenFisFile1: TMenuItem;
    SaveFisFileAS1: TMenuItem;
    SaveFisFile1: TMenuItem;
    ImportFileofType1: TMenuItem;
    TabSeparatedValues1: TMenuItem;
    CommaSeparatedValues1: TMenuItem;
    SpaceSeparatedValues1: TMenuItem;
    N2: TMenuItem;
    ExportGridFileAS1: TMenuItem;
    TabSeparatedValuesFile1: TMenuItem;
    CommaSeparatedValuesFile1: TMenuItem;
    SpaceSeparatedValuesFile1: TMenuItem;
    CloseFile1: TMenuItem;
    N3: TMenuItem;
    Variables1: TMenuItem;
    Define1: TMenuItem;
    PrintDefinitions1: TMenuItem;
    TransformValues1: TMenuItem;
    RecodeValues1: TMenuItem;
    Tools: TMenuItem;
    FormatGridCells1: TMenuItem;
    SortCases1: TMenuItem;
  end;

```

```

PrintGridFile1: TMenuItem;
SelectCases1: TMenuItem;
LoadaSubFile: TMenuItem;
Edit: TMenuItem;
InsertNewColumn1: TMenuItem;
CopyColumn1: TMenuItem;
CutColumn1: TMenuItem;
PasteColumn1: TMenuItem;
InsertNewRow1: TMenuItem;
CopyRow1: TMenuItem;
CutRow1: TMenuItem;
PasteRow1: TMenuItem;
N4: TMenuItem;
Analyses1: TMenuItem;
Options1: TMenuItem;
N5: TMenuItem;
ReOpen1: TMenuItem;
Action1: TAction;
Action2: TAction;
Action3: TAction;
Action4: TAction;
Action5: TAction;
Action6: TAction;
Action7: TAction;
Action8: TAction;
Descriptive1: TMenuItem;
DistributionStatistics1: TMenuItem;
FrequencyAnalysis1: TMenuItem;
OneSampleTests: TMenuItem;
Comparisons1: TMenuItem;
GeneralLinearModelGLM1: TMenuItem;
Correlation1: TMenuItem;
Autocorrelation1: TMenuItem;
MultipleRegression1: TMenuItem;
ForwardStepwise1: TMenuItem;
BackwardStepwise1: TMenuItem;
Multivariate1: TMenuItem;
GeneralLinearModel1: TMenuItem;
CrFissClassification1: TMenuItem;
AxBLogLinear1: TMenuItem;
NonparametricStatistics1: TMenuItem;
ContingencyChiSquare1: TMenuItem;
KruskalWallisOneWayANOVA1: TMenuItem;
SignTest1: TMenuItem;
Help1: TMenuItem;
About1: TMenuItem;
HelpContents1: TMenuItem;
FisTad2Fis2File1: TMenuItem;
procedure Exit1Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure DataGridKeyPress(Sender: TObject; var Key: Char);
procedure DataGridKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure OpenFisFile1Click(Sender: TObject);
procedure SaveFisFileAS1Click(Sender: TObject);
procedure SaveFisFile1Click(Sender: TObject);
procedure TabSeparatedValues1Click(Sender: TObject);
procedure CommaSeparatedValues1Click(Sender: TObject);

```

```

procedure SpaceSeparatedValues1Click(Sender: TObject);
procedure TabSeparatedValuesFile1Click(Sender: TObject);
procedure CommaSeparatedValuesFile1Click(Sender: TObject);
procedure SpaceSeparatedValuesFile1Click(Sender: TObject);
procedure CloseFile1Click(Sender: TObject);
procedure Define1Click(Sender: TObject);
procedure FormatGridCells1Click(Sender: TObject);
procedure CopyColumn1Click(Sender: TObject);
procedure CutColumn1Click(Sender: TObject);
procedure InsertNewColumn1Click(Sender: TObject);
procedure PasteColumn1Click(Sender: TObject);
procedure CopyRow1Click(Sender: TObject);
procedure CutRow1Click(Sender: TObject);
procedure PasteRow1Click(Sender: TObject);
procedure PrintGridFile1Click(Sender: TObject);
procedure PrintDefinitions1Click(Sender: TObject);
// procedure TransformValues1Click(Sender: TObject);
procedure Action1Execute(Sender: TObject);
procedure Action2Execute(Sender: TObject);
procedure Action3Execute(Sender: TObject);
procedure Action4Execute(Sender: TObject);
procedure Action5Execute(Sender: TObject);
procedure Action6Execute(Sender: TObject);
procedure Action7Execute(Sender: TObject);
procedure Action8Execute(Sender: TObject);
procedure DistributionStatistics1Click(Sender: TObject);
procedure FrequencyAnalysis1Click(Sender: TObject);
procedure SortCases1Click(Sender: TObject);
procedure SelectCases1Click(Sender: TObject);
procedure RecodeValues1Click(Sender: TObject);
procedure LoadaSubFileClick(Sender: TObject);
procedure InsertNewRow1Click(Sender: TObject);
procedure ThreeDimensionVariableRotation1Click(Sender: TObject);
procedure PlotXversusY1Click(Sender: TObject);
procedure Options1Click(Sender: TObject);
procedure OneSampleTestsClick(Sender: TObject);
procedure N12or3WayAnalysisofVariance1Click(Sender: TObject);
procedure GeneralLinearModelGLM1Click(Sender: TObject);
procedure Autocorrelation1Click(Sender: TObject);
procedure ForwardStepwise1Click(Sender: TObject);
procedure BackwardStepwise1Click(Sender: TObject);
procedure GeneralLinearModel1Click(Sender: TObject);
procedure AxBLogLinear1Click(Sender: TObject);
procedure KuderRichardson21Reliability1Click(Sender: TObject);
procedure SpearmanBrownProphecyReliability1Click(Sender: TObject);
procedure ContingencyChiSquare1Click(Sender: TObject);
procedure KruskalWallisOneWayANOVA1Click(Sender: TObject);
procedure SignTest1Click(Sender: TObject);
procedure FriedmanTwoWayANOVA1Click(Sender: TObject);
procedure About1Click(Sender: TObject);
procedure HelpContents1Click(Sender: TObject);
procedure FisTad2Fis2File1Click(Sender: TObject);

private
{ Private declarations }
public
{ Public declarations }
end;

```

```

var
  FisTadMainFrm: TFisTadMainFrm;
  PrevRow : integer;
  PrevCol : integer;
implementation
uses OptionsUnit, DictionaryUnit, DataProcs, DescriptiveUnit, FreqUnit, OutPutUnit,
  PlotXYUnit, OneSampUnit, BlkAnovaUnit, GLMUnit, AutoCorUnit, StepFwdMr, BackRegUnit,
  TwoWayLogLinUnit, ChiSqrUnit, KWanova, SignTestUnit, RChartUnit, SigmaChartUnit,
  pChartUnit, cChartUnit, RecodeUnit, SortUnit, SelectUnit,
  FileExtractUnit;
{$R *.DFM}
procedure TFisTadMainFrm.Exit1Click(Sender: TObject);
begin
  OptionsFrm.SaveBtnClick(Self);
  TempStream.Free;
  TempVarItm.Free;
  Close;
end;
//-----
procedure TFisTadMainFrm.FormShow(Sender: TObject);
var i : integer;
begin
  OpenStatPath := GetCurrentDir;
  LoggedOn := false;
  OptionsFrm.InitOptions(Self);
  NoVariables := 0; // global variable for no. of variables (columns)
  NoCases := 0; // global variable for no. of cases (rows)
  TempStream := TMemoryStream.Create; // global variable (simulate clipboard)
  TempVarItm := TMemoryStream.Create; // global var. for dictionary clips
  FilterOn := false; // global variable = true when a filter variable selected
  DictLoaded := false; // global variable = true when a dictionary file read
  RowEdit.Text := '1';
  ColEdit.Text := '1';
  FileNameEdit.Text := 'TempFile.TAB';
  FilterEdit.Text := 'OFF';
  FisTadMainFrm.DataGrid.RowCount := 2;
  FisTadMainFrm.DataGrid.ColCount := 2;
  FisTadMainFrm.DataGrid.Cells[0,0] := 'CASE/VAR.';
  FisTadMainFrm.DataGrid.Cells[0,1] := 'CASE ' + IntTFistr(1);
  FisTadMainFrm.DataGrid.Cells[1,1] := '';
  PrevRow := 1;
  PrevCol := 1;
  NoCasesEdit.Text := '0';
  NoVarsEdit.Text := '0';
  DictionaryFrm.DictGrid.RowCount := 0;
  FisTadMainFrm.DataGrid.SetFocus;
end;
//-----
procedure TFisTadMainFrm.DataGridKeyPress(Sender: TObject; var Key: Char);
var
  row, col : integer;
begin
  if ord(Key) = 13 then exit;
  row := FisTadMainFrm.DataGrid.Row;
  if StrToInt(NoCasesEdit.Text) < row then
  begin
    NoCasesEdit.Text := IntTFistr(row);
    NoCases := row;
  end;
end;

```

```

end;
col := FisTadMainFrm.DataGrid.Col;
if StrToInt(NoVarsEdit.Text) < col then
begin
  NoVarsEdit.Text := IntTFistr(col);
  NoVariables := col;
  if VarDefined[col] = false then
  begin
    DictionaryFrm.NewVar(col);
  end;
end;
if FisTadMainFrm.DataGrid.Cells[0,row] = '' then
begin
  NoCases := row;
  FisTadMainFrm.DataGrid.Cells[0,row] := 'CASE ' + IntTFistr(row);
end;

if NoVariables < col then
begin
  NoVariables := col;
end;
if ((PrevCol <> col) or (PrevRow <> row)) then
  if FisTadMainFrm.DataGrid.Cells[PrevCol,PrevRow] <> '' then
FormatCell(PrevCol,PrevRow);
  PrevCol := col;
  PrevRow := row;
end;
//-----
procedure TFisTadMainFrm.DataGridKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
  x, y, v : integer;
begin
  x := FisTadMainFrm.DataGrid.Row;
  y := FisTadMainFrm.DataGrid.Col;
  v := ord(Key);

  case v of
    13 : begin // return key
      if y = FisTadMainFrm.DataGrid.ColCount - 1 then
      begin
        FisTadMainFrm.DataGrid.ColCount := FisTadMainFrm.DataGrid.ColCount + 1;
        FisTadMainFrm.DataGrid.Col := y + 1;
      end;
    end;
    40 : begin // arrow down
      if x = FisTadMainFrm.DataGrid.RowCount - 1 then
      begin
        FisTadMainFrm.DataGrid.RowCount := FisTadMainFrm.DataGrid.RowCount + 1;
        FisTadMainFrm.DataGrid.Cells[0,x+1] := 'CASE ' + IntTFistr(x+1);
      end;
    end;
  end;
  RowEdit.Text := IntTFistr(FisTadMainFrm.DataGrid.RowCount - 1);
  ColEdit.Text := IntTFistr(FisTadMainFrm.DataGrid.ColCount - 1);
  if ((PrevCol <> y) or (PrevRow <> x)) then
  if FisTadMainFrm.DataGrid.Cells[PrevCol,PrevRow] <> '' then
FormatCell(PrevCol,PrevRow);

```

```

end;
//-----
procedure TFisTadMainFrm.OpenFisFile1Click(Sender: TObject);
var
  i : integer;
  filename : string;
begin
  OpenFis2File;
  filename := FileNameEdit.Text;
  // move all down 1 and add new one at the top
  for i := 8 downto 1 do
  begin
    MainMenu1.Items[0].Items[11].Items[i].Caption :=
      MainMenu1.Items[0].Items[11].Items[i-1].Caption;
    MainMenu1.Items[0].Items[11].Items[i-1].Caption := ' ';
  end;
  MainMenu1.Items[0].Items[11].Items[0].Caption := filename;
  OptionsFrm.SaveBtnClick(Self);
end;
//-----
procedure TFisTadMainFrm.SaveFisFileAS1Click(Sender: TObject);
var
  i : integer;
  filename : string;
begin
  SaveFis2File;
  filename := FileNameEdit.Text;
  // move all down 1 and add new one at the top
  for i := 8 downto 1 do
  begin
    MainMenu1.Items[0].Items[11].Items[i].Caption :=
      MainMenu1.Items[0].Items[11].Items[i-1].Caption;
    MainMenu1.Items[0].Items[11].Items[i-1].Caption := ' ';
  end;
  MainMenu1.Items[0].Items[11].Items[0].Caption := filename;
  OptionsFrm.SaveBtnClick(Self);
end;
//-----
procedure TFisTadMainFrm.SaveFisFile1Click(Sender: TObject);
begin
  SaveFis2File;
end;
//-----

procedure TFisTadMainFrm.TabSeparatedValues1Click(Sender: TObject);
begin
  OpenTabFile;
end;
//-----

procedure TFisTadMainFrm.CommaSeparatedValues1Click(Sender: TObject);
begin
  OpenCommaFile;
end;
//-----
procedure TFisTadMainFrm.SpaceSeparatedValues1Click(Sender: TObject);
begin
  OpenSpaceFile;

```



```

end;
//-----

procedure TFisTadMainFrm.TabSeparatedValuesFile1Click(Sender: TObject);
begin
    SaveTabFile;
end;
//-----
procedure TFisTadMainFrm.CommaSeparatedValuesFile1Click(Sender: TObject);
begin
    SaveCommaFile;
end;
//-----
procedure TFisTadMainFrm.SpaceSeparatedValuesFile1Click(Sender: TObject);
begin
    SaveSpaceFile;
end;
//-----

procedure TFisTadMainFrm.CloseFile1Click(Sender: TObject);
var
    i, j : integer;
begin
    for i := 0 to NoCases do
        for j := 0 to NoVariables do FisTadMainFrm.DataGrid.Cells[j,i] := '';
        FormShow(self);
    end;
end;
//-----

procedure TFisTadMainFrm.Define1Click(Sender: TObject);
begin
    DictionaryFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.FormatGridCells1Click(Sender: TObject);
begin
    FormatGrid;
end;
//-----

procedure TFisTadMainFrm.CopyColumn1Click(Sender: TObject);
begin
    CopyColumn;
end;
//-----

procedure TFisTadMainFrm.CutColumn1Click(Sender: TObject);
begin
    DeleteCol;
end;
//-----

procedure TFisTadMainFrm.InsertNewColumn1Click(Sender: TObject);
begin
    InsertCol;
end;
//-----

```

```

procedure TFisTadMainFrm.PasteColumn1Click(Sender: TObject);
begin
    PasteColumn;
end;
//-----

procedure TFisTadMainFrm.CopyRow1Click(Sender: TObject);
begin
    CopyRow;
end;
//-----

procedure TFisTadMainFrm.CutRow1Click(Sender: TObject);
begin
    CutRow;
end;
//-----

procedure TFisTadMainFrm.PasteRow1Click(Sender: TObject);
begin
    PasteRow;
end;
//-----

procedure TFisTadMainFrm.PrintGridFile1Click(Sender: TObject);
begin
    PrintData;
end;
//-----

procedure TFisTadMainFrm.PrintDefinitions1Click(Sender: TObject);
begin
    PrintDict;
end;
//-----

procedure TFisTadMainFrm.Action1Execute(Sender: TObject);
var
    filename : string;
    atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[0].Caption;
    atpFis := PFis('&',filename);
    if atpFis > 0 then Delete(filename,atpFis,1);
    if FileExists(filename) then ReOpen(filename);
end;
//-----

procedure TFisTadMainFrm.Action2Execute(Sender: TObject);
var
    filename : string;
    atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[1].Caption;
    atpFis := PFis('&',filename);

```



```

        if atpFis > 0 then Delete(filename,atpFis,1);
        if FileExists(filename) then ReOpen(filename);
    end;
//-----
procedure TFisTadMainFrm.Action3Execute(Sender: TObject);
var
    filename : string;
    atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[2].Caption;
    atpFis := PFis('&',filename);
    if atpFis > 0 then Delete(filename,atpFis,1);
    if FileExists(filename) then ReOpen(filename);
end;
//-----

procedure TFisTadMainFrm.Action4Execute(Sender: TObject);
var
    filename : string;
    atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[3].Caption;
    atpFis := PFis('&',filename);
    if atpFis > 0 then Delete(filename,atpFis,1);
    if FileExists(filename) then ReOpen(filename);
end;
//-----

procedure TFisTadMainFrm.Action5Execute(Sender: TObject);
var
    filename : string;
    atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[4].Caption;
    atpFis := PFis('&',filename);
    if atpFis > 0 then Delete(filename,atpFis,1);
    if FileExists(filename) then ReOpen(filename);
end;
//-----

procedure TFisTadMainFrm.Action6Execute(Sender: TObject);
var
    filename : string;
    atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[5].Caption;
    atpFis := PFis('&',filename);
    if atpFis > 0 then Delete(filename,atpFis,1);
    if FileExists(filename) then ReOpen(filename);
end;
//-----

procedure TFisTadMainFrm.Action7Execute(Sender: TObject);
var

```

```

filename : string;
atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[6].Caption;
    atpFis := PFis('&',filename);
    if atpFis > 0 then Delete(filename,atpFis,1);
    if FileExists(filename) then ReOpen(filename);
end;
//-----

procedure TFisTadMainFrm.Action8Execute(Sender: TObject);
var
    filename : string;
    atpFis : integer;
begin
    DictLoaded := false;
    filename := MainMenu1.Items[0].Items[11].Items[7].Caption;
    atpFis := PFis('&',filename);
    if atpFis > 0 then Delete(filename,atpFis,1);
    if FileExists(filename) then ReOpen(filename);
end;
//-----

procedure TFisTadMainFrm.DistributionStatistics1Click(Sender: TObject);
begin
    DescriptiveFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.FrequencyAnalysis1Click(Sender: TObject);
begin
    FreqFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.SortCases1Click(Sender: TObject);
begin
    SortFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.SelectCases1Click(Sender: TObject);
begin
    SelectFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.RecodeValues1Click(Sender: TObject);
begin
    RecodeFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.LoadaSubFileClick(Sender: TObject);
begin
    FileExtractFrm.ShowModal;
end;

```

```

end;
//-----

procedure TFisTadMainFrm.InsertNewRow1Click(Sender: TObject);
begin
    InsertRow;
end;
//-----

procedure TFisTadMainFrm.ThreeDimensionVariableRotation1Click(
    Sender: TObject);
begin
    //Rot3DFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.PlotXversusY1Click(Sender: TObject);
begin
    PlotXYFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.Options1Click(Sender: TObject);
begin
    OptionsFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.OneSampleTestsClick(Sender: TObject);
begin
    OneSampFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.N12or3WayAnalysisofVariance1Click(Sender: TObject);
begin
    BlksAnovaFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.GeneralLinearModelGLM1Click(Sender: TObject);
begin
    GLMFrm.ShowModal;
end;

//-----

procedure TFisTadMainFrm.Autocorrelation1Click(Sender: TObject);
begin
    AutoCorFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.ForwardStepwise1Click(Sender: TObject);
begin
    StepFwdFrm.ShowModal;
end;

```

```

//-----

procedure TFisTadMainFrm.BackwardStepwise1Click(Sender: TObject);
begin
    BackRegFrm.ShowModal;
end;

//-----

procedure TFisTadMainFrm.GeneralLinearModel1Click(Sender: TObject);
begin
    GLMFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.AxBLogLinear1Click(Sender: TObject);
begin
    TwoWayLogLinFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.KuderRichardson21Reliability1Click(Sender: TObject);
begin
    //KR21Frm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.SpearmanBrownProphecyReliability1Click(
    Sender: TObject);
begin
    //SpBrFrm.ShowModal;
end;

//-----

procedure TFisTadMainFrm.ContingencyChiSquare1Click(Sender: TObject);
begin
    ChiSqrFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.SignTest1Click(Sender: TObject);
begin
    SignTestFrm.ShowModal;
end;
//-----

procedure TFisTadMainFrm.HelpContents1Click(Sender: TObject);
begin
    Application.HelpContext(100);
end;
//-----

procedure TFisTadMainFrm.FisTad2Fis2File1Click(Sender: TObject);
begin
    OpenFisData;
end;
//-----

```