**Trabajo Práctico #1 : Metodología-DP-ECP-Módulos-ED: Registros y Arreglos.**

Se requiere de un proceso que informe **diversos listados** de las ventas realizadas por los vendedores, en donde, cada uno de ellos podrá realizar más de una venta. Para ello se cuenta con un archivo de datos de texto, desordenado con el siguiente diseño:

**VentasFerreteria.Txt**: de las ventas realizadas por los vendedores, conteniendo los siguientes datos:

**a:** Código de Vendedor (999), **b:** Cantidad (999), **c:** Descripción (20 cars.), **d:** Precio Unitario (99999.99), **e:** Fecha Venta (sFec). sFec es un registro con 3 campos (dd,mm,aaaa).

La cantidad máxima de registros está indicada por la constante **MAX\_REGS = 800**, la cantidad máxima de vendedores está indicada por la constante **MAX\_VENS = 100.** Se deberán utilizar **constantes con nombres** para establecer los valores indicados anteriormente. Cada uno de los registros leídos del archivo *VentasFerreteria.Txt* **deberán ser volcados a una estructura de datos interna, la cual solo contendrá todos los datos a leer del archivo**. Para todo vendedor, éstos podrán realizar más de una venta de artículos.

En el bloque principal, solo se establecen las invocaciones a los módulos, ud. deberá declarar las variables pertenecientes a este módulo y sus tipos de datos.

ProcVentas(VentasAF,vrVendedores,cantVtas);

ListadoVtasAgrupVen(SalidaAF,vrVendedores,vrTotales);

ListadoCantTot(SalidaAF,vrTotales,cantVen);

ListadoImpTot(SalidaAF,vrTotales,cantVen);

VentasAF.close();

SalidaAF.close();

**return** 0;

**Se pide:**

1. Emitir el siguiente listado considerando que la cabecera del vendedor debe salir solo una vez por cada vendedor distinto, el diseño es:

**Listado (1) ordenado por Código de Vendedor con repetición**

**Cód. Vendedor: 999**

**#Item Fecha Cant. Descripción Pre.Unit. Tot.Item Tot. Ven.**

1 99-99-9999 999 X(20) $ 9999.99 $99999.99

2 . . . . .

. . . . . .

n 99-99-9999 999 X(20) $ 9999.99 $99999.99 $99999.99

. . .

. . .

Total General: $999999.99

Cód. Vend. mayor venta de importe: 999

1. Emitir el siguiente listado ordenado por Cantidad Total de cada vendedor, decreciente.

**Listado (2) ordenado decreciente por Cantidad Total de cada Código de Vendedor**

**Cód. Ven. Cant. Total**

999 9999

. .

1. Emitir el siguiente listado ordenado por Importe Total de cada vendedor, decreciente.

**Listado (3) ordenado decreciente por Importe Total de cada Código de Vendedor**

**Cód. Ven. Importe Total**

999 $999999.99

. .

Los listados serán destinados a un archivo en disco, con el nombre de “**ListadoAF.Txt**”, formateado con títulos y con los resultados obtenidos. Hay espacio suficiente para generar estos archivos.

**Observaciones, restricciones y recursos disponibles:**

Utilizar las siguientes funciones, invocando en donde sea necesario:

* ***LeeUnaVta(VentasAF, rVendedor)***, el cual **lee una componente de datos** del archivo y los almacena en una estructura interna. La función retorna un boolean, verdadero, si la lectura fue exitosa, caso contrario, falso.
* ***ProcVentas(VtasAF, vrVtas, cVtas)*** *//Carga c/u. de los datos leídos en vrVtas. Se debe invocar a la función LeeUnaVta. Devuelve la cantidad de registros leídos en cVtas.*
* ***OrdxBur(vrVtas, card);*** *// Ord. x código de vendedor creciente.*
* ***OrdxBur(vrTot, card);*** *//Ord. x cantidad total de cada vendedor decrec.*
* ***OrdxBur(card, vrTot);*** *//Ord. x importe total de cada vendedor decrec.*
* ***IntCmb(vrVtas[i], vrVtas[i+1]);*** *//Debe invocarse dentro de la función OrdxBur.*
* ***IntCmb(vrTot[i], vrTot[i+1]);*** *//Debe invocarse dentro de la función OrdxBur.*

**Espacio en disco**: Solo para generar los archivos de los listados en formato texto.

**Espacio para arreglos y registros**: Lo necesario que requiera este proyecto.

**Espacio en memoria dinámica**: 0 bytes.

**Accesos a los archivos**: un solo recorrido secuencial, para leer cada uno de los datos.

**Bloque Principal**: sólo invocaciones a módulos, según lo establecido anteriormente.

**Optimización**: dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos.

Utilizar nombres significativos para los identificadores, dibujos para las estructuras de datos a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Preparar una muestra de datos para el archivo de datos e imprimirla, para ejecutar el programa. En el disco solo contendrán los archivos “**TP1V\_K1\_ \_ \_G\_-Apellido Nombre.cpp**”, y los archivos de datos “**VentasFerreteria.Txt**” (*de texto*), y el archivo de texto de la salida de los listados “**ListadoAF.Txt**” ubicados en el directorio raíz de la unidad.

**Cada grupo debe crear su propia muestra de datos para el archivo.**

La cantidad máxima de registros está establecida al inicio de este documento.

Se deben utilizar constantes con nombres para indicar estas cantidades.

La muestra de datos para VentasFerreteria.Txt debe ser: no menos de 10 códigos de vendedores distintos, además debe registrarse por cada renglón los datos de un registro de un vendedor, sin títulos, solo debe tener los datos, recordando que un mismo vendedor puede realizar varias ventas, considerando que el archivo está desordenado, los items están indicados en la sección de archivos.

El Trabajo Práctico deberá ser entregado de acuerdo a las pautas indicadas más abajo, el cual se aprobará si reúne los requerimientos solicitados en **tiempo y forma**:

(**Se debe respetar el orden indicado a continuación**)

1. Entregar en carpeta tamaño A4 de tapa transparente y con sujetador de gancho para las hojas lo siguiente:
2. Carátula con los datos de los integrantes del TP1, entre 7 y 10 alumnos por grupo del mismo curso.
3. Esta misma hoja que establece el enunciado del problema a resolver. Cada grupo elegirá un líder del proyecto, que será el responsable de realizar las entregas del TP. Si un líder abandona la cursada, se deberá elegir otro líder.
4. Diseñar las estructuras de Datos graficándolas indicando con rótulos apropiados, cada elemento, su tamaño en bytes y las variables utilizadas. Las estructuras de datos a graficar son:
   1. *El diseño de los datos a leer del archivoVentasFerreteria.Txt.*
   2. *Otras estructuras de datos que considere necesarias para poder realizar el proceso solicitado.*
5. Graficar el Bloque Principal.
6. Graficar cada uno de los módulos –funciones- a utilizar, cabecera y cuerpo.
7. Construir una muestra de datos, para el archivo de datos, la cual se la utilizará para probar el Algoritmo. En la hoja impresa a entregar debe haber rótulos apropiados, pero, NO en el archivo de Datos, “VentasFerreteria.Txt”, el cual contendrán solamente, los datos. Ver detalle del formato más abajo.
8. Emitir según la muestra establecida, los resultados esperados, siempre acompañada de los rótulos apropiados.
9. Codificación del Algoritmo completo en el Lenguaje C++, emitiendo números de líneas. Usar Code-Blocks. Las primeras líneas serán de comentario indicando: Nombre del programa, fecha entrega, Nro. versión, breve comentario del objetivo del programa, datos del curso, nombre del día, turno, nro. del grupo e integrantes (Apellido, Nombre).
10. Además de la documentación requerida entregada en carpeta, se deberá entregar acompañando a la misma en pen drive o tarjeta de memoria el código fuente escrito en C++ para compilarlo con la versión de Borland 5.5. También incluir el archivo de datos.

**Formato del archivo de datos VentasFerreteria.Txt**

Se deberá ajustar la escritura de los datos respetando a raja tabla este formato en donde cada columna representa un dato y su tipo de dato y ancho:

999 9(3) X(20) 9(5).99 99 99 9999

Cada columna representa de izquierda a derecha lo siguiente:

**Cód.Ven Cant. Descrip. Precio Unit. Fecha**

(3 díg.) (3 díg.) (20 caracteres) (99999.99) (99 99 9999)

**Ejemplos:**

352 12 thinner 1 ltr. 154.52 15 5 2022

7 307 destornillador phili 702.19 7 5 2022

83 271 papel de lija fina 18.60 12 5 2022

352 5 pinza punta fina 1009.08 14 5 2022

Cada nueva entrega además del nombre indicado para el archivo del código en C++ irá acompañado de la versión entregada, iniciando la primera entrega con el sufijo V1, luego la segunda entrega V2, y así sucesivamente. **(S.E. ú O.).**