**UTN.BA Materia** : Algoritmos y Estructuras de Datos **T.P.#2:** Consultas de Vuelos.

**Trabajo Práctico #2 :** Se evalúan los temas de: Estructuras de Datos: (Archivos Binarios, Estructuras Dinámicas de Datos, Técnicas recursivas, Templates, más temas del TP1).

Se requiere realizar un proceso que "**permita hacer Consultas Remotas de los Vuelos de un día del mes actual**"; contando para ello con los siguientes archivos binarios de datos:

- **Aeropuertos.Dat**: sin orden, con **57 registros**, conteniendo:

| | | | | |
|---|---|---|---|---|
| **a)** provincia (*str20* ) | **b)** ciudad (*str25*) | **c)** nom. Aerop. (*str30*) | d) codOACI (*str4*) | e) codIATA (*str3*) |

- **Vuelos.Dat:** desordenado, que representan los vuelos de un día, cada registro contiene:

| | | | |
|---|---|---|---|
| **a)** nro.Vuelo (*str9*) | **b)** dist.Km (*short* 4 díg.) | **c)** vel.Crucero (*short* 4 díg.) | d) cant.Pasajeros. (*short*) |
| d) empresa (*str8*) | **e)** marca Aeronave **(***str11***)** | f) fecha Sale (***int*** aaaammdd) | g) hora Sale (***short*** hhmm) |

- **Consultas.Dat**: sin orden, conteniendo cada componente nro. de vuelo (*str9*), para, simular las consultas remotas, cuyo formato consta de: 3 letras que representa el código IATA, seguido de 3 caracteres dígitos, seguido de 3 letras que representa otro código IATA. El **código IATA** representa un aeropuerto, por lo que el código IATA de la izquierda representa el aeropuerto de Salida, mientras que el código IATA de la extrema derecha representa el aeropuerto de Llegada de la aeronave. **Ej: MDQ*739*AEP** indica el nro.Vuelo 739 que proviene del aeropuerto Astor Piazzolla de Mar del Plata, para arrivar al aeroparque Jorge Newbury de CABA. **Ej. empresa**: FlyBondy, JetSmart, Aero.Arg., LADE. **Ej. Marcas**: BOING 737, 787, AIRBUS A320, A370, A380.

**Se deberá considerar el siguiente bloque principal:**

```
main() {   // Definir los tipos, variables, inicialización y argumentos a incluir en cada invocación a las funciones.
   Abrir ( lista de argumentos );
   ProcAeropuertos ( lista de argumentos );
   ProcVuelos ( lista de argumentos );
   ConsultasVuelos ( lista de argumentos );
   ListVueAeropSld ( lista de argumentos );
   Cerrar ( lista de argumentos  );
   return 0;
} // main
```

| **Constantes a utilizar:** |
|---|
| **const** *CANT_AEROP = 57;* |
| **Completar los struct y los typedef (*)** |
| **struct sAerop; struct sVue; struct** *tinfo,* ***struct*** *sNodo,* **struct** *sTblAerop,* **typedef** ____ *tvrAerop* [ _____ ]; **typedef** struct sNodo* _____; |

**Se pide:**

1. **ProcAeropuertos** para volcar el **codIATA, su ubicación y NULL**, a la tabla vrAerop, que debe quedar ordenada por el campo **codIATA**.
2. **ProcVuelos** que obtiene el **codIATA** del aeropuerto de origen del campo nroVuelo, para localizarlo en el **vrAerop** que retorna la **posición** encontrada, asigna al rInfo el **nro. de vuelo** y su **posición** del archivo de Vuelos y genera un nodo insertado en la Lista de vuelos ordenado por nro. de Vuelo colgado del codIATA del aeropuerto de salida.
3. **ConsultasVuelos** para emitir los siguientes datos, cuyo diseño de salida se indica a continuación:

**Consultas de vuelos del 99 de mesStr de 9999**

| NroVue. | Ciu.Orig. | Nom.Aerop.Orig. | Empresa | Marca | Ciu.Dest. | Nom.Aerop.Dest. | Estado | dia | hhAct | hhSa | t.V. | hhLl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X(9) | X(16) | X(16) | X(8) | X(11) | X(16) | X(16) | X(15) | 99 | 99:99 | 99:99 | 99:99 | 99:99 |

4. **ListVueAeropSld** para obtener un listado ordenado por código Aeropuerto de Salida con repetición, recorriendo la tabla **vrAerop**. Utilizar la técnica de Corte de Control. El diseño del listado:

**UTN.BA Materia** : Algoritmos y Estructuras de Datos **T.P.#2:** Consultas de Vuelos.

**Listado Salidas Aerop. Origen a otros Aerop. Llegada, del dia 99**

**Aerop. origen : XXX X(30) Ciudad: X(25)**

| **NroVue.** | **Empresa** | **Marca** | **Ciu.Dest.** | **Nom.Aerop.Dest.** | **Estado** | **dia** | **hhAct** | **hhSa** | **t.V.** | **hhLl** |
|---|---|---|---|---|---|---|---|---|---|---|
| X(9) | X(8) | X(11) | X(16) | X(16) | X(15) | 99 | 99:99 | 99:99 | 99:99 | 99:99 |

**Observaciones, restricciones y recursos disponibles:**

Utilizar las siguientes funciones, invocando en donde sea necesario:

- *long* **GetTime**(*int* hora,*int* min, *int* seg). *La función retorna la hora larga, como un solo número en el formato hhmmss. Además en sus parámetros devuelve la hora, los min. y los segundos*.
- *long* **GetDate***(int* year,*int* mes,*int* dia,*int* diaSem*). La función retorna la fecha larga, como un solo número en el formato aaaammdd. Además en sus parámetros devuelve el año, el mes, el día y el día de la semana.* Esta función se encuentra en OBTENER LA FECHA Y HORA DEL SISTEMA del apunte del prof. Hugo Cuello Teoría y Práctica del Lenguaje C/C++ *ANEXOS*
- *Desde el bloque principal se mencionan las invocaciones a los siguientes módulos: Abrir, ProcAeropuertos, ProcVuelos, ConsultasVuelos, ListVueAeropSld y Cerrar.*
- *Como orientación para resolver este TP2, podrán consultar la solución enviada correspondiente al* ***Recup1 del Parcial1*** *que fuera enviado al aula virtual.* ***Acá encontrarán otros módulos que podrán ser utilizados****: replicate, SubCad, OrdxBur, IntCmb, InsertarEnOrden, BusBinVec, HoraLlega, FormatoHoraMin, VerifEstado, SepararHoraMin, CortarCadenas, EmiteTitCab, EmiteLinDet.* ***Además de estos módulos se deberán utilizar los módulos para las estructuras dinámicas de datos****: InsertaNodo, InsertaInicio, InsertaEnMedio, InsertaEnLugar, BuscarClvNodo, SacarPrimerNodo.*

**Espacio en disco**: Solo para generar el archivo de salida **Listado Consulta Vuelos.Txt** en formato texto.
**Espacio para arrays y registros**: Lo necesario que requiera este proyecto.
**Espacio en memoria dinámica**: Un nodo por cada vuelo, con tamaño del nodo de acuerdo a ProcVuelos, colgado del codIATA del aeropuerto de salida y ordenado por el nroVuelo.
**Accesos a los archivos**: un solo recorrido secuencial, para leer en *Aeropuertos*, *Vuelos* y en *Consultas*. Además acceso al azar para leer en *Aeropuertos*, acceso al azar para leer en Vuelos, en las instancias que correspondan.
**Bloque Principal**: sólo invocaciones a módulos, según lo establecido anteriormente.
**Paradigma de Programación**: Solo se aceptará el Paradigma Imperativo Procedural, Programación Estructurada y Modular.
**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. **Se entregan las muestra de datos de los tres archivos** que deberán imprimirlos, estos deberán ser convertiods a archivos binarios, para ello se deben crear los programas de conversión de texto a binario que NO deberán ser entregados. En el disco solo contendrán los archivos "**TP2V_K1_ _ _G_-Apellido Nombre.cpp**", y los archivos de datos indicados anteriormente como así también el archivo de salida todos ubicados en la carpeta raíz del disco. Ejemplo: TP2V**1**_K1**023**G**3**_**PEREZ JUAN**.CPP

**Cada grupo debe crear su propia muestra de datos para los archivos, mencionados. A partir de los archivos de texto por cada uno de ellos hacer un programa que lee el archivo de texto y lo grabe en binario. Estos archivos NO deberán ser entregados.**

Se deben utilizar constantes con nombres para indicar cantidades.

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 (tres fechas máximas)**: A: Aprobado, N: No Aprobado. (**Se debe respetar el orden indicado a continuación**):

1. Entregar en carpeta tamaño A4 de tapa transparente y con sujetador de gancho perfectamente alineadas para las hojas lo siguiente: **(no se aceptan hojas sueltas ni otro tipo de carpeta)**
2. Carátula con los datos de los integrantes del TP, la cantidad de alumnos por grupo del mismo curso se determinará en clase no superando 5 grupos como máximo.
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:
    a. *El diseño de los registros de cada uno de los archivos.*
    b. *Otras estructuras de datos que considere necesarias para poder realizar el proceso solicitado. Algunas serán explicadas en clase, como complemento a este documento.*
5. Graficar el Bloque Principal.
6. Graficar cada uno de los módulos –funciones- a utilizar, cabecera y cuerpo.
7. Como fuera indicado anteriormente se entregan los tres archivos de texto de datos, que deberán ser converitos a binario, la cual se la utilizará para probar el Algoritmo. En la hoja impresa a entregar debe haber rótulos apropiados, pero, NO en los archivos de Datos, el cual contendrán solamente, los datos en binario. Ver detalle del formato más abajo.
8. Emitir según la muestra establecida, los resultados esperados, siempre acompañada de los rótulos apropiados, según formato de salida indicados anteriormente.
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). Nombre del compilador: Borland C++ V.5.5**

La salida de los resultados debe estar dirigida a un archivo de texto con el nombre **Listado Consulta Vuelos.Txt** se debe utilizar la sentencia **freopen**, para redirigir la salida de la pantalla a archivo de texto al utilizar **cout**.
La cantidad de datos de muestra para cada archivo deberán ser adecuados contemplando las distintas posibilidades del proceso.
Dar nombres de identificadores representativos a su uso, es decir, con significado.
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.ù O.**