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

Se requiere de un proceso que asigne turnos por servicios médicos según especialidad, día y turno (mañana, tarde, noche), de acuerdo a la solicitud de quien requiere el servicio. Para ello, se cuenta con los siguientes archivos de datos:

**Especialidades.Txt**: desordenado, de las Especialidades médicas, conteniendo en cada línea el nombre de una especialidad (str20). **Cantidad máxima 20**.

**Medicos.Txt:** sin orden, **cantidad máxima 100,** cada línea representa un médico conteniendo:

|  |  |  |  |
| --- | --- | --- | --- |
| Apellido Nombre (str20) | Nro. Matricula (6 dig.) | Nombre Especialidad (str20) | Turno (char)  (M, T, N) |

Para todo médico un solo turno y todos los días. Franjas horarias (8 a 12, 12 a 16, 16 a 20).

**TurnosDiaHora.Txt**: sin orden, cada una de las líneas contiene los siguientes datos:

|  |  |  |
| --- | --- | --- |
| Nombre Especialidad | Dia (2 díg.) | Hora (2 díg.) |
| Min (2 díg.) | Obra Social (str15) | Nro. Credencial (6 díg.) |

En este archivo están las solicitudes de los días previos al actual.

**SolicitudTurnos.Txt**: sin orden, cada línea contiene los siguientes datos por cada paciente:

|  |  |  |  |
| --- | --- | --- | --- |
| Apellido Nombre (str20) | Edad (2 díg.) | Obra Social (str15) | Nro. Matricula (8 díg.) |
| Nombre Especialidad | Dia | Turno (char: M, T, N) |  |

En este archivo están todas las solicitudes que se realizan al dia actual.

Cada uno de estos archivos deberán ser volcados en estructuras de datos internas estáticas una por cada archivo, las cuales solo contendrán los datos a leer del archivo**.**

**Cantidad de Días 31**, **Cantidad de Turnos 24** (desde las 08:00 a 19:30, con intervalos de 30 mins.)

**Se pide:**

1. Listado de las solicitudes previas al proceso, ordenado por Especialidades, Días y Turnos todos ascendentes. El diseño del listado obedece al siguiente formato de salida

**Listado de turnos INICIAL orden Espec.+Día+Turno**

**Especialidad Día Horario Obr. Soc. Nro. Cred.**

X(20) 99 99:99 X(15) 9(6).99

1. Listado de las solicitudes actuales al proceso, ordenado cronológico de las transacciones.

**Listado de Solicitudes Actuales orden Cronológico**

**Apellido Nombre Nom.Obr. Soc. Nro. Cred. Especialidad Dia Turno Medico/a**

X(20) X(15) 9(9.99 X(20) 99 99:99 X(20)

1. Listado de las solicitudes actualizadas una vez finalizado el proceso de las solicitudes actuales, ordenado por Espacialidades, Días y Turnos todos ascendentes.

**Listado de turnos ACTUALIZADO orden Espec.+Día+Turno**

**Especialidad Día Horario Obr. Soc. Nro. Cred.**

X(20) 99 99:99 X(15) 9(6).99

**Nota**: Cada uno de los listados deberán ser destinados a un único archivo de texto con el siguiente nombre “**Listadosx3.Txt**”, formateado con títulos y con los resultados obtenidos. Hay espacio suficiente para generar estos archivos.

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.

main() {

Declarar las variables utilizadas en el bloque main().

Abrir todos los archivos

ProcMedicos( ); // Descarga archivo Medicos.

ProcEspecialidad( ); // Descarga archivo Especialidades.

ProcTurnos( ); // Descarga archivo TurnosDiaHora.

LstTurnos( ); // Lista ord. x Especialidades, Días y Turnos del vuelco del archivo TurnosDiasHora.

ProcTurnos( ); // Descarga archivo SolicitudesTurnos y lista líneas de las solicitudes.

LstTurnos( ); // Lista ord. x Especialidades, Días y Turnos Actualizado con las SolicitudesTurnos.

Cerrar todos los archivos

return 0;

}

**Observaciones, restricciones y recursos disponibles:**

Utilizar las siguientes funciones, invocando en donde sea necesario:

* *bool* LeeSuf(*ifstream* &id, *sid* &id), 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; *sid* indica un tipo de estructura de datos. Crear una función de lectura para cada archivo que se lee. Suf luego de Lee en el nombre de la función es un sufijo para el archivo que se lee, el cual debe ser reemplazado Suf por el nombre apropiado, p.e. *LeeSlc*, *LeeMed*…
* *void* GrabaSlctd(*ofstream* &id, *tid* id), graba línea solicitudes actuales orden cronológico.
* *tid* BusBin(*tid* tbl, *tid* clv, *tid* ult)***,*** en donde ***tid*** indica el tipo de identificador a utilizar. Crear una función de búsqueda para clave de ***tid*** diferente.
* *void* OrdxBur(*tid* tbl, *tid* card)*, ordena tbl PerMed por NomEsp+Turno.*
* *void* OrdxBur(*tid* tbl, *tid* card)*, ordena tbl Especialidades.*
* *void* IntCmb(*id* &elem1, *id* &elem2)*, intercambia ambos elementos.* Crear una función de búsqueda para clave de ***tid*** diferente.
* *void* InicEspTur(*tid* tbl)*, que inicializa tbl de turnos nomObrSoc y nroCred campos del registro con ‘\*’ y 0 respectivamente.*
* *short* cnvHhMm(*short* hhmm) retorna un valor entero entre 0 y 23 para cada caso del selector hhmm que será un valor entre 800 y 1930 con paso de 30, p.e. 800 retorna 0, 830 retorna 1…
* *char \**cnvChr2Cad(*char* car)*; convierte car a cadena.*
* *char* \* cnvPosHhMm(*short* n) retorna una cadena entre “08:00” y “19:30” para cada caso del selector n que será un valor entre 0 y 23, p.e. 0 retorna “08:00”, 1 retorna “08:30”…
* *char \*space(short n), retorna una cadena con n espacios.*
* *string* replicate(*char* car, *unsigned* n), retorna una cadena con n veces car.

**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.

**Paradigma de Programación**: Solo se aceptará el **Paradigma Imperativo**, **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. 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 indicados anteriormente como así también el archivo de salida todos ubicados en la carpeta raíz del disco.

**Cada grupo debe crear su propia muestra de datos para los archivos.**

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)**:

(**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, 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:
   1. *El diseño de los registros de cada uno de los archivos.*
   2. *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. Construir una muestra de datos, para los 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 los archivos de Datos, 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, 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).

**Formato del archivo de datos TurnosDiasHoras.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:

Cada columna representa de izquierda a derecha lo siguiente:

**Nom.Especialidad Dia Horario Nom.Obr.Soc. Nro.Cred.**

X(20) 99 00:99 X(15) 9(6)

**Ejemplos:**

Fonoaudiologia 23 10:30 OSECAC 235401

Pediatria 7 9:00 OSTEL 62296

Radiologia 15 17:30 OSPSA 192357

Para los ingresos de los datos de cada archivo de texto, utilizar el editor Code-Blocks.

Se utilizará este formato semejante a los demás archivos.

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.).**