**Enunciado del problema a resolver.** (Puntaje Total 7 (siete) sin errores. Tiempo máximo total 1h30m).

Se evalúan los temas de: Uso de una Metodología para resolver problemas, Datos Primitivos, Estructuras de Control de Programas, Diseño Modular, Estructuras de Datos (Registro, Arreglos y Archivo de Texto).

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 de datos:

* **Aeropuertos.Txt**: sin orden, con **57 líneas**, conteniendo:

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

* **Vuelos.Txt:** desordenado, **máximo 20.000 líneas**, que representan los vuelos de un día.

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

* **Consultas.Txt**: sin orden, conteniendo nro. de vuelo (*str9*) cada línea 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* ( );

VolcarAeropuertos ( );

GenIndVuelos ( );

ConsultasVuelos ( );

ListVueAeropSld ( );

*Cerrar* ( );

**return** 0;

} // main

**Tabla de diseño de invocaciones y prototipos (interfaz) de funciones**

|  |
| --- |
| **Constantes a utilizar:** |
| **const** *CANT\_AEROP = 57, MAX\_VUELOS = 20000, TAM\_LINEA\_VUELOS = 60;* |
| **Completar los struct y los typedef (\*)** |
| **struct** *sAerop,* ***struct*** *sVue*, **struct** *sIndVue,*  **typedef** *\_\_\_\_ tvrAerop* [ \_\_\_\_\_ ]; **typedef** \_\_\_\_\_ *tvrIndVue*[ \_\_\_\_ ]; **typedef** \_\_\_ *str30*[ \_\_\_\_ ]; |
| **Prototipos de funciones cuyos módulos deberán desarrollarse con los argumentos indicados:** |
| ***void*** VolcarAeropuertos (*ifstream* &Aerop, *tvrAerop* vrAerop); |
| ***void*** GenIndVuelos (*ifstream* &Vuelos, *tvrIndVue* vrIndVuelos, ***short*** &cantVue); |
| ***void*** ConsultasVuelos (*ifstream* &Cons, *ifstream* &Vue, *tvrIndVue* vrIndVue, *tvrAerop* vrAerop, ***short*** cantVue, ***short*** &diaHoy); |
| ***void*** ListVueAeropSld (*ifstream* &Vue, *tvrIndVue* vrIndVue, *tvrAerop* vrAerop, ***short*** cantVue, ***int*** diaAct); |
| **Prototipos de funciones que NO deberán desarrollarse pero si llamarlas con los argumentos adecuados:** |
| ***bool*** LeerAerop(*ifstream* &Aerop, *sAerop* &rAerop); |
| ***bool*** LeerVuelos(*ifstream* &Vuelos, *sVue* &rVuelo); |
| ***bool*** LeerCons(*ifstream* &Cons, *str9* nroVue); |
| ***void*** OrdxBur(*tvrAerop* vrA, ***short*** card); |
| ***void*** InsertarEnOrden(*tvrIndVue* vE, *sIndVue* elem, ***short*** card); |
| ***int*** BusBinVec(*tvrIndVue* vrI, ***char*** clv[], ***short*** ult); |
| ***int*** BusBinVec(*tvrAerop* vrA, **c*h*ar** clv[], ***short*** ult); |
| ***void*** EmiteLinDet(*sVue* rVue,***char*** nomAerop[], ***char*** ciudad[], ***char*** nomAeropD[],***char*** nomCiudadD[], ***int*** nDiaHoy); |
| ***char*** \* SubCad(***char*** cad[], ***short*** dsd, short hst); |

**Se pide:**

1. **(1 punto).** Dibujar y codificar: Las estructuras de datos indicadas en la tabla de más arriba (\*).

El dibujo debe ser**: COMPLETO, PROLIJO, SIMPLE, CLARO, y con RÓTULOS APROPIADOS.**

1. (**6 puntos**). Desarrollar las siguientes funciones:
2. (**1 punto**). Diagramar la función **VolcarAeropuertos** invocando a **LeerAerop** para volcar todos sus datos, a una tabla, invocar a la función **OrdxBur** que ordena los datos por el campo **codIATA**.
3. (**1,5 punto**). Codificar la función **GenIndVuelos** invocando a **LeerVuelos,** asignar el **nro. de vuelo** y su **posición** a otra tabla invocando a la función **InsertarEnOrden** ordenado por **nro. de vuelo**.
4. (**2 puntos**). Codificar la función **ConsultasVuelos** que invoca a: **GetDate**, **EmiteTitCab(1)**, luego, invocando a **LeerCons** que debe emitir una linea en respuesta al **nro. de vuelo** consultado remotamente (simulado con el archivo **Consultas.Txt**). Procesando los datos de este archivo se debe invocar a: **BusBinVec** que busca el **nroVue** en la tabla **vrIndVue**, para ubicar el puntero al archivo invocando a **LeeVuelos**, luego invocar a **SubCad**, para dividir (split) el código del aerop. de origen y de destino, indicado en el **nro. de vuelo**, luego acceder con **BusBinVec** dos veces a la tabla de aeropuertos para obtener las posiciones de AeropOrigen y AeropDest, por último, invocar a la función **EmiteLinDet**, para emitir los siguientes datos, cuyo diseño de salida se indica a continuación:

**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

1. (**1,5 punto**). Diagramar la función **ListVueAeropSld** para obtener un listado ordenado por código Aeropuerto de Salida con repetición, recorriendo la tabla **vrIndVue**. Se deberá obtener la **SubCad** del aeropuerto de origen y hacer una **BusBinVec** en la tabla de aerpuertos y la posición del aeropuerto de origen, emitir titulos de la cabecera del aeropuerto de origen. Luego si el aeropuerto de origen es el mismo hacer; obtener la **subCad** para el aeropuerto de destino, mover el puntero al archivo vuelos según la tabla vrIndVue, **LeerVuelos**, obtener la **subCad** el aeropuerto de destino, hacer la **BusBinVec** para obtener la posición y finalmente invocar a la función **EmiteLinDet** e incrementar el índice para la tabla de vrIndVue. El diseño del listado:

**Listado Salidas Aerop. Origen a otros Aerop. Llegada del dia 31**

**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**

1. El enunciado se complementa con explicaciones adicionales y gráficos indicados en la pizarra, de ser necesarios.
2. **IMPORTANTE**: Resolver primero el punto 1; tiempo estimado 15 minutos.
3. **Cada una de las preguntas respondidas deben estar rotuladas a qué pregunta responde: ( 1:, 2.a:, 2.b:, 2.c:, 2.d: )**.
4. Establecer una muestra de datos con 15 nros. de Vuelos del archivo “**Vuelos.Txt**”, considerando los siguientes cód. IATA: AEP, MDQ, NQN, USH, MDZ, COR. La muestra debe contemplar repeticiones de aeropuertos de origen, en 3 aeropuertos distintos.
5. Dado que el **uso de los ciclos afecta el rendimiento del proceso**, se considerará oportuno optimizar estas situaciones de las Estructuras de Control de Programas (ECP) de la **Repetición**, como así también de la **Selección** y la **Concatenación**.

**S.E.u O.**

**Afirmaciones** (Tiempo máximo 5 min. puntaje máximo 3 (tres) puntos)

**Marcar con un círculo** la afirmación que ud. considera correcta **V** o **F**, si responde bien, suma 1, y si NO responde, o responde mal, suma 0. **NO escribir en la columna Pntj.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Afirmación** | **Marcar** | | **Pntj.** |
| 1 | Dada la siguiente afirmación:  Los parámetros pasados por referencia, sus correspondientes argumentos pueden ser constantes, expresiones o funciones. | V | **F** |  |
| 2 | Dado el siguiente código:  ***char*** \* SubCad(***char*** cad[], ***short*** dsd, ***short*** hst) {  **static** ***char*** sCad[4];  strcpy(sCad,"");  **for** (***short*** i = dsd, j = 0; i <= hst; i++,j++)  sCad[j] = cad[i];  **return** sCad;  } // Retorna la sub-cadena de la cadena cad, desde la posición dsd hasta la posición hst. Ej.: si cad es MDQ519AEP y dsd es 0 y hst es 2, retorna MDQ. | **V** | F |  |
| 3 | Dado el siguiente código:  **short** horaSale = 1735; // representa formato de la hora militar.  short minSale;  minSale = horaSale % 100;  horaSale = horaSale / 100;  cout << horaSale << ‘:’ << minSale;  // Emite: 17:35  // (que representa la hora compuesta en dos componenes hora y minutos). | V | F |  |

**Respuestas a las Afirmaciones**

**Afirmación Pregunta 1 Parcial 1 (agosto 2025)**

Dada la siguiente afirmación:

**FALSO**. Los argumentos pasados por referencia solo deben ser variables, ya que las variables están ubicadas en una posición en la memoria RAM.

**Afirmación Pregunta 2 Parcial 1 (agosto 2025)**

Dado el siguiente código:

**VERDADERO**. Retorna los tres primeros caracteres de la cadena cad, siendo estos caracteres MDQ.

**Afirmación Pregunta 3 Parcial 1 (agosto 2025)**

Dado el siguiente código:

**VERDADERO.** Desgloza la hora militar representado por hhmm en sus dos componentes hh:mm que representa la hora hh y los minutos mm.

**Folleto explicativo del problema a resolver**

**ESTRUCTURAS DE DATOS**

**typedef** *sAerop* *tvrIndVue* [MAX\_VUELOS];

**Tabla Aeropuertos. const** CANT\_AEROP = 57, TAM\_LINEA\_VUELOS = 60, MAX\_VUELOS = 20000;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *sAerop*  **struct** sAerop {  *str20* pcia;  *str25* ciudad;  *str30* nomAerop;  *str4* codOACI,  codIATA;  ***float*** preUni;  }; | | | | |
| *str20*  pcia  21b. | *str25*  ciudad  26b. | *str30*  nomAerop  4b. | *str4*  codOACI  5b. | *str3*  codIATA  5b. |
| 0 |  |  |  |  |  |
| *1* |  |  |  |  |  |
| *2* |  |  |  |  |  |
| BB |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  | BB |
| *CANT\_AEROP-1* |  |  |  |  |  |
|  | sAerop vrAerop [CANT\_AEROP]  tvrAerop vrAerop; | | | | |

**Vuelos.Txt** Ej.: en la posición 8 entre otros campos también está el **nro. de vuelo** COR375AED, luego acceder a la tabla vrAerop con la sub-cadena AeropOrig y con la sub-cadena AeropDest con BB.

0 1 2 3 4 5 6 7 8 9 10 … m - 1

*typedef* *sIndVue* *tvrIndVue* [MAX\_VUELOS];

|  |  |  |
| --- | --- | --- |
|  | *sIndVue* | |
| T.R.: 12 bytes. | ***str9***  nroVue  10 b. | ***short***  posVue  2 b. |
| 0 |  |  |
| 1 |  | DD |
| 2 |  |  |
| 3 |  |  |
|  | COR375AED | 8 |
| cantVuelos-1  BB |  |  |
|  |  |  |
| MAX\_VUELOS - 1 |  |  |
| T.TBL.: 240000 bytes | *tvrIndVue* vrIndVuelos;  *sIndVue vrIndVuelos* [MAX\_VUELOS]; | |

**struct** *sIndVue* {

*str9* nroVue;

***short*** posVue;

};

**typedef** *sIndVue*  *tvrIndVue* [MAX\_VUELOS];

*sIndVue* vrIndVuelos[MAX\_VUELOS];

o

*tvrIndVue* vrIndVuelos;

**Consultas.Txt**: contiene solamente **nroVuelo** desordenado, una línea por cada consulta realizada. Ej.: posición cero el nro. de vuelo es **COR375AED**

0 1 2 3 4 … n - 1

Rec.Sec. **Consultas.Txt** por c/u de las líneas leídas acceder a la tabla **vrIndVuelos** con BB. con el **nroVue** leído, desde este lugar con el campo **posVue** acceder con DD. a **Vuelos.Txt** desde este otro lugar acceder con BB. con el campo **nroVue** de la sub-cadena del codIATA AeropOrig a la tabla **vrAerop** y también con el mismo campo **nroVue** de la sub-cadena del codIATA AeropDest a la tabla **vrAerop**.