Programación Orientada a Objetos / Programación II - UNL-FICH - Parcial 2 - 08/11/24 - Tema A

Ej1 (30pts) a) Implemente una función *to_list* que reciba un *string* conteniendo una lista de datos entre llaves y separados por coma, y retorne una *std::list<string>* con los datos individuales. Un ejemplo del *string* que recibe podría ser "{FuPro,POO,AED,Ing Soft}"; y la *std::list* resultante debería contener 4 elementos: "FuPro", "POO", "AED", e "Ing Soft". **b)** Implemente una función *to_string* que haga lo contrario: reciba una *std::list<string>* y la convierta en un solo *string* con el formato del ejemplo (agregue las llaves y las comas para armar un único *string*). **c)** Implemente una función *reemplazar* que reutilice las dos anteriores para reemplazar en una lista que recibe como un único *string,* un elemento dado por otro. La lista y ambos elementos son argumentos de la función: Ej: *reemplazar("{FuPro,POO,AED}", "POO", "Prog II")* cambiaría la lista por *"{FuPro,Prog II,AED}".* La función debe retornar *true* si realizó el reemplazo, *false* si el elemento a reemplazar no estaba en la lista.

Ej2 (30pts) Un editor de imágenes guarda en un archivo de texto llamado "historial.txt" la lista de archivos recientes, uno por línea. Es decir que la primera línea tiene la información del último archivo de imagen que sobre el que el usuario trabajó, la segunda línea la ruta del ante-último, y así sucesivamente. Cada línea consta de un entero fecha-hora (en formado AAMMDDHHMM) y luego la ruta, separada por un espacio (la ruta puede tener más espacios!). Ej: "2411080955 Parcial Poo.jpg" indicaría que el archivo "Parcial Poo.jpg" se usó a las 9:55 del 8/11/2024. Dado el *struct Entrada { string nombre; unsigned int fecha_hora; }* para representar una lista como un *vector<Entrada>*: **a)** Escriba una función *combinar* que reciba dos vectores con 2 historiales diferentes, y un entero N. La función debe generar un nuevo vector combinando la información de ambas listas originales en una nueva, asegurándose de no incluir elementos repetidos (si una entrada estaba en las dos listas originales, debe estar una sola vez en la nueva) y de no tener más de N líneas (si entre las dos listas hay más de N entradas diferentes, se debe quedar con las N más recientes). **b)** Complete el programa implementando el *main* y cualquier otra función necesaria para que combine dos listas desde los archivos “lista1.txt” y “lista2.txt” en una tercera “lista_comb.txt” de no más de 20 entradas.

Ej3 (25pts) a) Implemente una función genérica que reciba el nombre de un archivo binario que guarda muchos datos de un cierto tipo, y dos valores A y B de ese mismo tipo. La función debe buscar en el archivo al valor A, y reemplazarlo por B (en todas las ocurrencias, puede estar más de una vez); y retornar el nro de reemplazos realizados. La función debe operar directamente sobre el archivo (no usar un arreglo, std::vector, std::list o cualquier otro contenedor). **b)** ¿Qué requisitos debe cumplir un tipo de dato para poder ser utilizado con esta función?

Ej4 (15pts) a) Mencione un ejemplo de una situación/programa donde convendría utilizar un archivo de texto para guardar la información, y otro donde convendría un archivo binario. Justifique. **b)** ¿Qué significa que un contenedor tenga acceso aleatorio? Mencione un contenedor que provea acceso aleatorio y uno que no. **c)** ¿Qué es un *c-string*? Mencione una ventaja de utilizar un *c-string* por sobre la clase *std::string.*

Programación Orientada a Objetos / Programación II - UNL-FICH - Parcial 2 - 08/11/24 - Tema B

Ej1 (30pts) Una dirección web URL (como por ej “http**://**unl.edu.ar**/**ii/parciales**?**materia=poo") tiene las siguientes partes: primero el protocolo (lo que va antes del "://"), luego el dominio (lo que sigue, hasta la primera "/"), luego el directorio (lo que resta hasta encontrar "?" o hasta el final) y finalmente los parámetros (desde el "?" en adelante). En el primer ejemplo, " el protocolo es "http", el dominio "unl.edu.ar", el directorio "ii/parciales" y los parámetros "materia=poo". El directorio podría estar vacío y los parámetros a veces pueden directamente no estar (ej: "https**://**unl.edu.ar**/**"). **a)** Escriba una función que dado un string con una url la separe en las partes mencionadas y retorne un *struct url_s { string protocolo, dominio, directorio, parámetros; }*. **b)** Escriba una función que reciba una *std::list* de strings que contengan urls y utilizando la función anterior genere una nueva *std::list* conteniendo sólo los dominios y sin repetir (ya que dos o más urls diferentes podrían tener el mismo dominio).

Ej2 (30pts) Una plataforma de streaming guarda para cada usuario un archivo binario con la lista de películas que el usuario vió y calificó. La calificación es un puntaje de 1 a 5 para cada película. El archivo guarda por cada una: título, fecha en que la terminó de ver, y calificación que le asignó. **a)** Proponga una posible definición de un *struct Pelicula* que se use para guardar y leer esos datos en el archivo binario. **b)** Implemente una función que reciba un *std::vector<Pelicula>* con información de todas las películas que vio y calificó un usuario, y un entero *N*. La función debe filtrar la lista de la siguiente forma: eliminar todas las películas con calificación menor a 4; ordenar la lista por fecha de más reciente a más antigua, y si la lista tiene más de *N* elementos, quedarse solo con los *N* más recientes. **c)** Complete el programa (implementando el *main* y cualquier otra función necesaria) para que lea una lista de un archivo “vistas.dat”, la filtre con la función de a) utilizando *N*=20, y actualice el archivo dejando solamente los elementos de la lista filtrada.

Ej3 (25pts) a) Implemente una función genérica que reciba el nombre de un archivo de texto y dos valores A y B de cierto tipo. El archivo guarda una lista de valores de ese tipo, uno por linea. La función debe modificar el archivo reemplazando ese al valor A por B (en todas las ocurrencias, puede estar más de una vez en la lista); y retornar el nro de reemplazos realizados. **b)** ¿Qué requisitos debe cumplir un tipo de dato para poder ser utilizado con esta función?

Ej4 (15pts) a) Al utilizar una plantilla de clase o de función, ¿en qué casos debo explicitar el tipo de dato con el que especializarla? ¿y en qué casos puedo omitirlo? **b)** ¿Qué ventajas y desventajas presenta el uso de *std::list* en lugar de *std::vector*? **c)** ¿Qué es un *iterator*? ¿Por qué los algoritmos de la STL utilizan *iterator*s en lugar de índices?