PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJE DE PROGRAMACIÓN 1

3ra práctica (tipo b)
(Primer Semestre 2019)

Indicaciones Generales:

- Duración: 110 minutos.
- Se podrá usar como material de consulta solo sus apuntes de clase (no fotocopias ni hojas sueltas).
- No se pueden emplear **variables globales**, **clases**, **objetos (excepto cin y cout)**. tampoco las bibliotecas **cstdio, stdio.h o similares**.
- Toda función propia deberá ser implementada obligatoriamente en archivo diferentes al main.cpp, debiendo crear los archivos .h y .cpp correspondientes y estas deben ser agrupadas por temas.
- En cada archivo que implemente en los proyectos (.h y .cpp) deberá colocar un comentario en el que coloque claramente su nombre y código, de no hacerlo se **le descontará 0.5 puntos por archivo**.
- Los programas que presenten errores de sintaxis o de concepto se calificarán en base al 40% de puntaje de la pregunta. Los que no den resultados coherentes en base al 60%.
- **La presentación, la ortografía y la gramática de los trabajos influirá en la calificación.**

Puntaje total: 20 puntos

Problema

El manejo de cadenas de caracteres en lenguaje C Estándar se realiza mediante la biblioteca de funciones denominada "string.h" o "cstring.h", sin embargo las funciones definidas en ella tienen el problema que no se fijan en la integridad de los datos, es decir que no controlan errores que se puedan producir, por ejemplo, por el desbordamiento de los datos en memoria. Además la forma como se emplean difiere mucho de la forma cómo se manipulan otros tipos de datos.

Este laboratorio busca que usted desarrolle una biblioteca estática de funciones, que permita operar cadenas de caracteres de modo que su manipulación sea más segura y parecida a la forma en que se operan otros datos. Se busca que la implementación de estas funciones sea muy básica de modo que no se redunde en el uso de las funciones estándar de C, por lo que en este laboratorio ***NO PODRÁ EMPLEAR*** tampoco los elementos definidos en las bibliotecas ***string.h, cstring.h, string (de C++)*** o similares. Tampoco podrá emplear métodos como **getline** para la lectura de cadenas de caracteres.

Pregunta 1 (13 puntos)

Se solicita que desarrolle una biblioteca estática de funciones en la cual se definirá una estructura de datos que contendrá la cadena y una serie de funciones y sobrecargas como se indican a continuación.

(1) Estructura:

La **cadena** se manejará mediante una estructura (struct) denominada "**Cadena**" que contenga tres campos, uno, que es un arreglo dinámico, que maneje la cadena propiamente dicha (denominado "**cad**") mediante un puntero (**char***), otro que almacene (denominado **longitud**) que guarde la longitud de la cadena (**int**), de modo que no se tenga que contar caracteres cada vez que se le solicite y que se actualice en cada operación con la estructura y uno (denominado **capacidad**) que contará con un valor que guarde la capacidad del arreglo que guarde la cadena (**int**), la cual no será necesariamente igual que la longitud y que también se actualice con las operaciones que se realicen sobre la estructura. No podrá agregar más campos, eliminar alguno o modificar los tipos de datos.

(2) Funciones:

La biblioteca implementará las siguientes funciones:

- ✓ **`inicializa(cadena)`** → la función recibirá como parámetro una estructura de tipo **Cadena**. El puntero cad se inicializa en NULL, la longitud de la cadena será cero al igual que la capacidad.

- ✓ **inicializa(cadena, cap)** → la función recibirá como parámetro una estructura de tipo **Cadena** y un valor entero. El puntero cad se inicializa en un espacio de memoria igual al indicado por **cap**, la longitud de la cadena será cero y su capacidad igual a **cap**.
- ✓ **leer(cadena, car)** → Permite lee desde la entrada estándar uno a uno los caracteres del buffer de entrada hasta encontrar un carácter igual al contenido en car. Si la función es invocada solo con el parámetro "**cadena**" la lectura se detendrá cuando se encuentre un cambio de línea ('\n'). Si la cadena tiene capacidad para contener la cadena se le asigna sin modificar su capacidad; si no es así, la asignación de memoria debe hacerse de manera exacta.
- ✓ **imprimir(cadena)** → Permite imprimir todos los campos de la estructura, debe etiquetar cada uno de ellos. Debe realizarse de modo que cualquier otra operación de salida de datos se realice en la siguiente línea.
- ✓ **longitud(cadena)** → Permite devolver la longitud de la cadena.
- ✓ **capacidad(cadena)** → Permite devolver la capacidad de la cadena.
- ✓ **recortar(cadena)** → Permite eliminar todos los "espacios" sobrantes de la cadena (blancos, tabuladores y cambios de línea). La función eliminará todos los "espacios" al inicio y al final de la cadena y solo debe mantener entre palabra y palabra un espacio en blanco. Esto es, si la cadena fuera: " \t \t\n\t Juan \t\t Pérez \n \n\t López\t\t \n\n\t", luego de ejecutar la función quedará como: "Juan Pérez López".
- ✓ **intercambiar(cad01, cad02)** → Al ejecutar la función, todos los campos de una estructura se intercambiarán con la otra.

(3) <u>Sobrecarga de operadores</u>:

La biblioteca implementará las siguientes sobrecargas:

- ✓ **Asignación (cad01 = cad02;)** → Sobrecargando el operador =. Se asignarán todos los campos de la estructura de la derecha en la de la izquierda. Las estructuras no deben compartir los mismos espacios de memoria.
- ✓ **Agregación (cad01 += cad02;)** Sobrecargando operador +=. Agrega la cadena de caracteres contenida en la estructura **cad02** al final de la cadena de la estructura **cad01**. Si la cadena tiene capacidad para contener las dos cadenas se le asigna sin modificar su capacidad; si no es así, la asignación de memoria debe hacerse de manera exacta.
- ✓ **Concatenación (cad01 = cad02 + cad03;)** Sobrecargando operador +. La función debe devolver una estructura **Cadena** con la concatenación de las cadenas **cad02** y **cad03**, de manera similar al del operador += pero en este caso no se deben modificar las cadenas **cad02** ni **cad03**.
- ✓ **Comparación (cad01 == cad02, cad01 > cad02** y **cad01 < cad02)** → Sobrecargando los operadores ==, > y <. Se comparan dos operandos de tipo **Cadena.** Se devuelve true si se cumple la condición y false si no.
- ✓ **Lectura (cin>>cadena;)** Sobrecargando operador >>. Permite lee desde la entrada estándar uno a uno los caracteres del buffer de entrada hasta encontrar el carácter de cambio de línea.
- ✓ **<u>Impresión</u> (cout<<cadena;)** Sobrecargando operador <<. Permite imprimir la cadena contenida en la estructura.

***<u>Consideraciones</u>*:**

- La solución debe contemplar la elaboración de: un proyecto de implementación y prueba de las funciones (deben probarse todas las funciones y sobrecargas), un proyecto que genere la biblioteca estática y un proyecto donde se pruebe la biblioteca ya compilada. La prueba de las funciones debe ser hecha lo más simple posible pero que muestre claramente los resultados correctos. Los proyectos se denominarán: "**ManejoDeCadenasImplementacionYPrueba**", "**ManejoDeCadenas**", "**ManejoDeCadenasPruebaCompilada**" respectivamente. Los tres proyectos deberán colocarse en una carpeta denominada "**Pregunta01**"

PARTE 2: Prueba final (7 puntos)

Se solicita que desarrolle un proyecto denominado "**Pregunta02"** en el cual se utilizará la biblioteca estática "**ManejoDeCadenas**" (compilada). El proyecto solucionará el problema planteado a continuación:

Se tiene un archivo de textos que contiene un registro de todas las citas médicas realizadas en un centro médico. El archivo es similar al que se muestra a continuación:

```
Pérez Carpio José # Traumatología # Yep Valderrama Rony
Roncal Neyra Ana Cecilia # Medicina Interna # Calixto Castillo Erick Paul
Guzmán Chávez Carlos Alberto # Cardiología # Inga Palomares Erika
Quispe Huamán Luisa Naomi # Cardiología # Cueva Vásquez Ángel Fausto
Fernández Millán Valentina # Cirugía plástica # Cerna Cerna Cinthia
Zavala Jiménez Rodrigo Pascual # Cardiología # Inga Palomares Erika
…
```

En cada línea se encuentra el nombre de un paciente, la especialidad en que se atendió y el nombre del doctor que lo atendió. Cada uno de estos campos estaré separado por el carácter '#'. Cada palabra que conforman los nombres o especialidades estará separada por uno o más espacios en blanco o tabuladores.

Empleando las funciones y sobrecargas de la biblioteca estática, deberá leer los datos de cada línea y colocarlos en tres arreglos, uno de ellos contendrá solo los nombres de los pacientes, el otro las especialidades y el último los doctores. Luego empleando estos tres arreglos deberá emitir el siguiente reporte:

```
     ESPECIALIDAD                DOCTOR                          PACIENTE
=============================================================================================
  …  …                           …                               …
 23) Cardiología                 Cueva Vásquez Ángel Fausto      Quispe Huamán Luisa Naomi
  …  …                           …                               …
 35) Cardiología                 Inga Palomares Erika            Guzmán Chávez Carlos Alberto
 36) Cardiología                 Inga Palomares Erika            Zavala Jiménez Rodrigo Pascual
  …  …                           …                               …
 77) Cirugía plástica            Cerna Cerna Cinthia             Fernández Millán Valentina
  …  …                           …                               …
123) Medicina Interna            Calixto Castillo Erick Paul     Roncal Neyra Ana Cecilia
  …  …                           …                               …
481) Traumatología               Yep Valderrama Rony             Pérez Carpio José
  …
```

El reporte debe estar perfectamente tabulado (sin usar el carácter '\t'). Los datos deben aparecer ordenados ascendentemente por especialidad, dentro de las especialidades por doctor, y dentro de cada doctor por paciente. La ordenación debe usando QuickSort o algún otro algoritmo con similar eficiencia (n log n). Las palabras de cada campo solo estarán separadas solo por un espacio en blanco.

Anotaciones finales

Al finalizar el laboratorio, comprima[1] las dos carpetas creadas (Pregunta01 y Pregunta02) en un archivo con nombre <código del alumno con 8 dígitos>.zip y súbalo a la intranet del curso, en el enlace Documentos, en la carpeta \Laboratorio3\<código del horario>\<aula>.

Profesores del curso: Miguel Guanira E.

San Miguel, 26 de abril del 2019.

[1] Para evitar problemas en la corrección de la prueba, utilice el programa de compresión que viene por defecto en el Windows (Zip) no 7z.