Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Ingeniería Informática

PROGRAMACIÓN ORIENTADA A OBJETOS

UNIDAD 05

Flujos de Entrada/Salida

Guía de trabajos prácticos
2018

UNIDAD 05
Flujos de Entrada/Salida

Ejercicio 1

Escriba un programa que cargue en un vector de strings una lista de palabras desde un archivo de texto (que contendrá una palabra por línea), muestre en pantalla la cantidad de palabras leídas, las ordene en el vector alfabéticamente, y reescriba el archivo original con la lista ordenada.
Ayuda: para ordenar un vector *v* de principio a fin puede utilizar la sentencia "*sort(v.begin(),v.end());*".

Ejercicio 2

```
Lopez Javier
56 90
Garcia Ana
77 81
Farias Daniel
60 62
```

En un archivo de texto llamado *lista.txt*, como el que se muestra a la derecha, se encuentran los nombres y notas de los alumnos de una comisión de Programación Orientada a Objetos.

a) Escriba una función modificar que reciba el nombre de un alumno y dos notas, y modifique el archivo reemplazando las dos notas de ese alumno.
b) Escriba una función que lea la lista del archivo y genere otro archivo de texto *promedios.txt* con una tabla donde cada línea posea el nombre, el promedio, y la condición final de cada uno de los alumnos.

Ayuda: utilice manipuladores de flujo (*setw*, *right*, *left*, *fixed*, *setprecision*) para dar formato a la tabla del archivo que se genera en b).

Ejercicio 3

Se tiene un archivo "inscriptos.txt" con una lista de nombres de alumnos inscriptos al cursado de Fundamentos de Programación. Se desea distribuir los estudiantes en comisiones de práctica de no más de 30 alumnos. Escriba un programa que determine cuantas comisiones deberían formarse de acuerdo a la cantidad de inscriptos y reparta los alumnos en comisiones de igual tamaño, guardando la lista de alumnos de cada comisión en archivo de texto "comision1.txt", "comision2.txt", … "comisionN.txt".
Ayuda: puede utilizar la clase stringstream como auxiliar para concatenar en un string texto y números para formar los nombres de los archivos.

Ejercicio 4

Un conjunto de archivos de texto contiene información guardada en el formato que se muestra en el recuadro, donde cada línea contiene el nombre de un campo y su respectivo valor separados por el signo igual (=). Las líneas que comienzan con el carácter '#' deben ser ignoradas.

```
#ejemplo de archivo de configuración
materia=Programacion Orientada a Objetos
carrera=Ingeniería en Informática
facultad=Facultad de Ingeniería y Ciencias Hídricas
universidad=Universidad Nacional del Litoral
#esta linea hay que ignorarla, igual que las 3
últimas
nro_unidad=5
nombre_unidad=Flujos de entrada/salida
otro_campo=otro_valor
otro_campo_mas=otro_valor_mas
#campo_que_no_se_lee_1=basura
#campo_que_no_se_lee_2=basura
#campo_que_no_se_lee_3=basura
```

Escriba una clase llamada ConfigFile que permita interpretar el contenido de estos archivos. La clase debe poseer:

a) un constructor que reciba el nombre del archivo y cargue sus datos en un vector de structs (cada elemento es struct con dos strings: campo y valor).
b) un método para consultar el valor asociado a un campo
c) un método para modificar el valor asociado a un campo
d) un método para guardar el archivo con los datos actualizados

Ejercicios Adicionales

Ejercicio 1

Escriba un programa que abra un archivo con el texto de *Don Quijote* de Miguel de Cervantes (puede obtener el archivo desde la dirección: http://www.gutenberg.org/files/2000/old/2dong10.txt) y cuente cuantas veces aparece en todo el texto la cadena *"molinos de viento"* (sin distinguir entre mayúsculas y minúsculas).

Ejercicio 2

```
Lopez Javier
342569085
Garcia Ana
342778180
Farias Daniel
342606234
```

Escriba un archivo de texto desde un editor de textos cualquiera (por ejemplo: el Bloc de Notas de Windows). Dicho archivo debe contener los nombres y números telefónicos de *N* personas con el formato mostrado en el recuadro. Guarde el archivo con el nombre *agenda.txt* u otro nombre de su elección.

a) Diseñe una clase llamada *Agenda* con un arreglo de structs que permita almacenar nombres y números telefónicos de un conjunto de personas.
b) Implemente un método *Cargar* que reciba un objeto de tipo *std::string* con el nombre del archivo que contiene los datos. El mismo debe abrir el archivo y cargar los datos de las personas en retornando un bool que indique si la lectura se realizó con éxito (true) o si ocurrieron errores durante la apertura del archivo (false).
c) Implemente un método *Buscar* que debe recibir un *std::string* con parte del nombre o del apellido de un contacto y devuelva los datos de la primer ocurrencia que coincida con dicha cadena.
d) Codifique un programa cliente que cargue los datos del archivo, informa la cantidad de personas cargadas y permita realizar un búsqueda.

Ejercicio 3

Agregue a la clase desarrollada en el ejercicio anterior un método *AgregarContacto(...)* que reciba los datos de un nuevo contacto, y lo agregue al arreglo; y agregue un método *Guardar* que actualice el archivo. Utilice la función desde un programa cliente para agregar un nuevo contacto y abra nuevamente el archivo desde un editor de texto para comprobar que los datos fueron agregados correctamente.

Ejercicio 4

Escriba una clase **AnalizaTexto** con métodos para analizar un archivo de texto (que contendrá varios párrafos) y obtener los siguientes resultados:

A. cantidad de caracteres en el texto (longitud del mismo)
B. cantidad párrafos (nota: recuerde ignorar líneas en blanco)
C. cantidad de letras, cantidad de espacios en blanco, y de otros caracteres (como signos depuntuación) por separado.

Proponga un programa cliente que la utilice.

Ejercicio 5

Investigue el uso de la clase *stringstream* para entender los siguientes ejemplos:

1) Ingresar datos en un stream y obtener el string resultante:

```
#include <iostream>
#include <sstream>
using namespace std;

int main(int argc, char *argv[]) {
	int n1,n2;
	cout<<"Ingrese dos enteros para sumar:";
	cin>>n1>>n2;
	int r=n1+n2;
	stringstream ss;
	ss<<r;
	string str=ss.str();
	cout<<"El resultado tiene ";
	cout<<str.size()<<" digitos."<<endl;
	return 0;
}
```

2) Convertir el contenido de un string en un stream para extraer datos:

```
#include <iostream>
#include <sstream>
using namespace std;

int main(int argc, char *argv[]) {
	cout<<"Ingrese dos numeros separados por un espacio: ";
	string str;
	getline(cin,str);
	stringstream ss(str);
	double n1,n2;
	ss>>n1>>n2;
	cerr<<(n1+n2);
	return 0;
}
```

Ejercicio 6

Escriba una función *reemplaza* que permita modificar un archivo de texto reemplazando todas las ocurrencias de una palabra o frase por otra. Por ejemplo, si se la invoca con *reemplazar("texto.txt","open source","código abierto");* debe buscar

en el archivo “texto.txt” la frase “open source” y reemplazarla por “código abierto” todas las veces que la encuentre.

Ejercicio 7

Escriba una función para corregir mayúsculas y minúsculas de un texto almacenado en un archivo: las mayúsculas solo van al comienzo de una oración, es decir, en la primer letra y luego de un punto; el resto debe estar en minúsculas.