EJERCICIOS RESUELTOS

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());*".

```
 #include <iostream>
#include <vector>
#include <fstream>
#include <algorithm>
using namespace std;

int main() {
	vector<string> texto;
	string palabra;
	ifstream archiLect("palabras.txt");
	if(!archiLect.is_open()){cout<<"Error";return 1;}
	while(getline(archiLect, palabra)){
		texto.push_back(palabra);
		cout<<palabra<<endl;
	}
	archiLect.close();
	ofstream archiEsc("palabras2.txt");
	if(!archiEsc.is_open()){cout<<"Error";return 1;}
	sort(texto.begin(),texto.end());
	cout<<"**********************************"<<endl;
	for(string T:texto){
		cout<<T<<endl;
		archiEsc<< T << endl;
	}
	archiEsc.close();
	return 0;
}
```

Ejercicio 2

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.

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

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

```
using namespace std;
struct ficha{
	string nombre;
	int nota1,nota2;
};
//Item a)
void modificar(ficha F, string nombArchi){
	vector<ficha> Fichero;
	ficha aux;
	ifstream archiLect(nombArchi.c_str());
	if(!archiLect.is_open()){cout<<"Error";exit;}
	while(getline(archiLect,aux.nombre)){
		archiLect >> aux.nota1;
		archiLect >> aux.nota2;
		archiLect.ignore();
		if(aux.nombre.compare(F.nombre) == 0){
			aux.nota1= F.nota1;
			aux.nota2= F.nota2;
		}
		Fichero.push_back(aux);
	}
	archiLect.close();
	ofstream archiEsc("lista2.txt");
	if(!archiEsc.is_open()){cout<<"Error";exit;}
	for(ficha FI:Fichero)
		archiEsc<< FI.nombre<<endl<<FI.nota1<<"  "<<FI.nota2<<endl;
	archiEsc.close();
}
//Item b)
void promedio(string nombArchi){
	ficha aux;
	float prom;
	ifstream archiLect(nombArchi.c_str());
	if(!archiLect.is_open()){cout<<"Error";exit;}
	ofstream archiEsc2("Promedio.txt");
	if(!archiEsc2.is_open()){cout<<"Error";exit;}
	while(getline(archiLect,aux.nombre)){
		archiLect >> aux.nota1;
		archiLect >> aux.nota2;
```

```
                archiLect.ignore();
                prom= ((aux.nota1+aux.nota2)/(2.0));
                archiEsc2<< aux.nombre<<endl<<prom<<endl;
        }
        archiLect.close();
        archiEsc2.close();
}
void mostrar(string arch){
        string cadena;
        stringstream texto;
        texto.clear();
        ifstream archivo(arch.c_str());
        if (archivo.is_open()){
                while (getline(archivo, cadena)){
                        texto<< cadena<<endl;
                }
                cout<<setw(20)<<texto.str();
        }else cout<<"ERROR"<<endl;
}

int main() {
        ficha FF;
        FF.nombre= "Garcia Ana";
        FF.nota1= 10;
        FF.nota2= 10;
        cout<<"********Lista 1**************"<<endl;
        mostrar("lista1.txt");
        modificar(FF,"lista.txt");
        cout<<"********Lista 2**************"<<endl;
        mostrar("lista2.txt");
        promedio("lista1.txt");
        cout<<"********Promedio **************"<<endl;
        mostrar("Promedio.txt");
        return 0;
}
```

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.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 5 - Ejercicio Nº 3
```

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <vector>
#include <sstream>
using namespace std;

void crear_lista(){//Creo el archivo de inscriptos
	ofstream inscriptos("inscriptos.txt");
	if(!inscriptos.is_open()){//Verifica la apertura del archivo
		cout<<"Error en apertura de archivo.";
		getchar();exit(0);}
	char letra;
	for(int x= 0; x< rand()%200+100;x++){//Crea entre 100 y 200 registros
		letra=(char)(65+rand()%24);//Genera una letra al azar
		for(int k=0; k<10; k++){//la escribe 10 veces, una al lado de la otra
			inscriptos<<letra;
			if(k==6) inscriptos<<", ";//Separo apellido
		}
		inscriptos<<endl;//Guarda el fin de linea (\n).-
	}
	inscriptos.close();//Cierra el archivo
}
//*****************************************************
void leer(vector<string> &alumnos, string archi){//Paso vector por referencia y archivo
	ifstream archivo(archi.c_str());//Lo abro para lectura
	if(!archivo.is_open()){//Verifica la apertura del archivo
		cout<<"Error en apertura de archivo.";
		getchar();exit(0);}
	string al;
	while(getline(archivo,al))//Mientras hay palabras leo el archivo
		alumnos.push_back(al);//Agrego al vector
	archivo.close();
}
//*****************************************************
void repartir(vector<string> &alumnos){
	int comisiones= (alumnos.size()/30)+1;
	cout<<"Cantidad de comisiones: "<<comisiones<<endl;
	stringstream nombArchi;
	ofstream salida("");
	for(int x=0,comis=0;x<(int)alumnos.size();x++){
		if(x%30==0){comis++;
		if(salida.is_open())salida.close();//Cierro el archivo.
		nombArchi.str("");//Limpio el contenido
		nombArchi<<"Comision_"<<comis;//Concateno letras y numeros
		cout<<nombArchi.str()<<endl;//Muestro en pantalla
		nombArchi<<".txt";//Concateno extensión
		salida.open(nombArchi.str());//Abro el arcivo nuevo
		if(!salida.is_open()){cout<<"Error";getchar();exit(0);}//Valido
		}
```

```
                cout<<alumnos[x]<<endl;//Muestro en pantalla
                salida<<alumnos[x].c_str()<<endl;//Guardo en el archivo
        }
        salida.close();
}
//*****************************************************
int main(int argc, char *argv[]) {
        vector<string> alumnos;
        crear_lista();
        leer(alumnos,"inscriptos.txt");
        cout<<"Cantidad de Inscriptos: "<<alumnos.size()<<endl;
        for(int n=0; n<(int)alumnos.size(); n++){
                cout<<n<<" - "<<alumnos[n]<<endl;
        }
        repartir(alumnos);
        return 0;
}
```

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

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 5 - Ejercicio Nº 4
```

```
#include <iostream>
#include <vector>
#include <fstream>
#include <cstdlib>
using namespace std;
struct registro {
	string campo, valor;
	} ;
//*************************************************
class ConfigFile{
private:
	vector<registro> V;
	string nombreArchivo;
public:
	ConfigFile(string);
	string ver_campo(string campo);
	void modificar_campo(string, string);
	void guardar(string);
	void ver_vector();
};
//Implementacion ************************
ConfigFile::ConfigFile(string nombArchi){
	nombreArchivo=nombArchi;
	ifstream leer(nombArchi.c_str());
	if(!leer.is_open()){cout<<"Error";getchar();exit(0);}//Valido
	string aux,cadena;
	int pos;
	registro R;
	while(getline(leer,cadena)){
		aux= cadena;
		if(aux.substr(0,1)!="#"){//Si es distinto de #
			pos=aux.find("=");//Busco el =
			R.campo=aux.substr(0,pos);//Corto el campo
			R.valor=aux.substr(pos+1);//Corto el valor
			V.push_back(R);//Agrego R al vector
		}
	}
	leer.close();//Cierro el archivo
}
string ConfigFile::ver_campo(string campo){
if(V.empty()){return "Archivo vacio";}
int x=0;
while((x<V.size())&&(V[x].campo.compare(campo)!= 0)){
	x++;}
if(V[x].campo.compare(campo)== 0)
  return V[x].valor;
else
	return "ERROR";
}
void ConfigFile::modificar_campo(string campo, string valor2){
	if(V.empty()){return;}
```

```
        int x=0;
        while((x<V.size())&&(V[x].campo.compare(campo)!= 0)){
                x++;}
        if(V[x].campo.compare(campo)== 0)
          V[x].valor= valor2;
}

void ConfigFile::guardar(string nombArchi){
        ofstream escribir(nombArchi.c_str());
        if(!escribir.is_open()){cout<<"Error";getchar();exit(0);}//Valido
        for(int x=0; x<(int)V.size(); x++){
                escribir<<V[x].campo<<"="<<V[x].valor<<endl;
        }
        escribir.close();
}
void ConfigFile::ver_vector(){
        cout<<"Tamaño del Vector:  "<<V.size()<<endl;
        for(registro R:V)
                cout<<R.campo<<"="<<R.valor<<endl;
}
//*****************************************************************************
int main(int argc, char *argv[]) {
ConfigFile F("archiE04.txt");
F.ver_vector();
cout<<"\n\nVer campo Facultad: ";
cout<<F.ver_campo("facultad");
cout<<"\n\nModificar campo nro_unidad=6: "<<endl<<endl;
F.modificar_campo("nro_unidad", "6");
F.ver_vector();
F.guardar("archiE04bis.txt");
return 0;
}
```

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/2donq10.txt) y cuente cuantas veces aparece en todo el texto la cadena *"molinos de viento"* (sin distinguir entre mayúsculas y minúsculas).

```
 //© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 5 - Ejercicio Adicional Nº 1

#include <iostream>
#include <algorithm>
#include <fstream>
using namespace std;

int cantidad(const string nombArchi, const string cadena){//Paso nombre de archivo y
cadena a buscar
	int ocurrencias=0;//La utilizo para contar las coincidencias
	ifstream leer(nombArchi.c_str());//Creo el objeto leer asociado al archivo fisico
	if(!leer.is_open()){cout<<"Error";getchar();return(0);}//Valido
	//Declaro variables a utilizar
	string cadena2, aux;
	int p;
	while(getline(leer,cadena2)){//Leo la linea del archivo
		aux= cadena2;//Asigno al string aux
		//Utilizo este recurso para pasar la linea a minusculas
		std::transform(aux.begin(), aux.end(), aux.begin(), ::tolower);
		//cout<<aux<<endl;//Descomentar para ver lectura por pantalla
		p= aux.find(cadena);//Busco la cadena en la linea con la funcion
miembro find()
		while (p !=string::npos){//Si la encontro cuenta (npos es el final del
string)
			ocurrencias++;//cuento 1 mas
			p= aux.find(cadena,p+1);//Vuelvo a buscar en la linea
		}//Termino de buscar en la linea
	}//Termino de leer el archivo */
	return ocurrencias;//Devuelvo la cantidad de ocurrencias
}
//******************************************************
int main(int argc, char *argv[]) {
	int salida;
	string archivo= "Adicionales//Quijote.txt";
	string busqueda= "molinos de viento";
	salida= cantidad(archivo , busqueda);
	cout<<"La frase: "<<busqueda<<", se encontro "<<salida<<" veces en el
archivo."<<endl;

	return 0;
}
```

Ejercicio 2

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.

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

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.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 5 - Ejercicio Adicional Nº 2

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <vector>
#include <algorithm>
using namespace std;
//*******************************************************
void creaLista(){
        ofstream lista("Adicionales//agenda.txt");
        if (!lista.is_open()){cout<<"ERROR ";getchar();exit(0);}
        lista<<"Lopez Javier"<<endl;
        lista<<"0342-15619022"<<endl;
        lista<<"Garcia Ana"<<endl;
        lista<<"0342-15677381"<<endl;
        lista<<"Farias Daniel"<<endl;
        lista<<"0342-15604602"<<endl;
        lista.close();//cerramos el archivo
}
//-------------------------------------------------------------
struct ficha{
        string nombre, telefono;
};
//Clase Agenda, Interface--------------------------------------
class Agenda{
private:
        vector<ficha> V;
public:
        bool cargar(string nombArch);
        ficha buscar(string dato);
```

```
        void ver_agenda();
};
//Clase Agenda, Implementacion --------------------------------
bool Agenda::cargar(string nombArch){
        ifstream archivo(nombArch.c_str());//Lo abro para lectura
        if(!archivo.is_open()){//Verifica la apertura del archivo
                cout<<"Error en apertura de archivo.";
                getchar();return false;}
        string a;
        ficha aux;
        while(getline(archivo,a)){//Mientras hay datos leo el archivo
                aux.nombre= a;
                getline(archivo,a);//Leo Telefono
                aux.telefono= a;
                V.push_back(aux);//Agrego al vector
        }
        archivo.close();
        return true;
}
ficha Agenda::buscar(string dato){
        int p;
        ficha aux; aux.nombre="E R R O R";
        string may;
        for(int n=0; n<(int)V.size(); n++){
                may=V[n].nombre;//Utilizo este recurso para pasar las cadenas a
mayusculas
                std::transform(may.begin(), may.end(), may.begin(), ::toupper);
                std::transform(dato.begin(), dato.end(), dato.begin(), ::toupper);
                p= may.find(dato);;//Busco la cadena en la linea con la funcion miembro
find()
                if (p !=string::npos){//Si la encontro (npos es el final del string)
                        aux= V[n];//Copio el struct
                        return aux;}//Devuelvo el struct y salgo del metodo
                }
        return aux;
}

void Agenda::ver_agenda(){
        cout<<"Cantidad de Fichas: "<<V.size()<<endl;
        for(int n=0; n<(int)V.size(); n++){
                cout<<n<<" - "<<V[n].nombre<<" - "<<V[n].telefono<<endl;
        }
}

//*****************************************************************************
int main(int argc, char *argv[]) {
        creaLista();//Creo el archivo
        Agenda A;//Declaro el objeto tipo Agenda
        A.cargar("Adicionales//agenda.txt");//Cargo las fichas desde el archivo
        A.ver_agenda();//Muestro el vector de fichas en pantalla
        char dato[50];
```

```
        cout<<"Ingrese Apellido y/o nombre a buscar: ";
        cin.getline(dato,50);//Ingreso el dato a buscar
        ficha f1= A.buscar(dato);//Busco el Apellido y/o nombre en el arreglo dinamico
        cout<<f1.nombre<<" - "<<f1.telefono<<endl;//Muestro resultado
        return 0;
}
```

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.

```
class Agenda{
private:
        vector<ficha> V;
        string nombreArchivo;
public:
        bool cargar(string nombArch);
        ficha buscar(string dato);
        void ver_agenda();
        void agregar_contacto(ficha f);
};
```

```
  void Agenda::agregar_contacto(ficha f){
        V.push_back(f);
        ofstream salida(nombreArchivo.c_str());
        //Verifica la apertura del archivo
        if(!salida.is_open()){cout<<"Error";getchar();exit(0);}//Valido
        for(int x=0;x<(int)V.size();x++){
                salida<<V[x].nombre.c_str()<<endl;//Guardo en el archivo
                salida<<V[x].telefono.c_str()<<endl;//Guardo en el archivo
        }
        salida.close();
}
```

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.

```
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 5 - Ejercicio Adicional Nº 4

#include <iostream>
#include <string>
#include <cstdlib>
#include <fstream>
using namespace std;
class AnalizaTexto {
	string frase;
	string::size_type largo;
public:
	AnalizaTexto(string archivo);
	void contar(int &vocales, int &consonantes, int &espacios, int &parrafos);
	string::size_type ver_largo();
	string ver_frase(){return frase;}
};
AnalizaTexto::AnalizaTexto(string narchivo){
	frase.clear();
	ifstream archivo(narchivo.c_str());//Lo abro para lectura
	if(!archivo.is_open()){cout<<"Error en apertura de archivo.";
	getchar();exit(0);}
	string  a;
	while(getline(archivo,a)){//Mientras hay datos leo el archivo
		frase+=a;//Concateno
		frase+="\n";//Enter
	}
	largo = frase.length();
}
void AnalizaTexto::contar(int &vocales, int &consonantes, int &espacios, int
&parrafos){
	int i;
	vocales=0; consonantes=0; espacios= 0; parrafos=1;
	int p= frase.find_first_of("aeiou");//Busca vocales
	while (p !=string::npos){//Si encontro alguna cuenta
		vocales++;
		p= frase.find_first_of("aeiou",p+1);//Vuelve a buscar
	}
	for (char k='a'; k < 'z'; k++){//Busca letras
		int p= frase.find(k);
		while (p !=string::npos){//Si encontro alguna, cuenta
			consonantes++;
			p= frase.find(k,p+1);//Vuelve a buscar
		}
	}
	consonantes -= vocales;//Calcula las consonantes (letras - vocales)
	p= frase.find_first_of(" ");//Busca espacios
	while (p !=string::npos){//Si encontro alguna, cuenta
		espacios++;
		p= frase.find_first_of(" ",p+1);//Vuelve a buscar
```

```
        }
        p= frase.find(".\n");//Busca espacios punto y aparte
        while (p !=string::npos){//Si encontro alguna, cuenta
                parrafos++;
                p= frase.find(".\n",p+1);//Vuelve a buscar
        }
}
string::size_type AnalizaTexto::ver_largo(){
        return largo;
}
//*************************************************************************
int main(){
        AnalizaTexto T("Adicionales\\agenda.txt");
        int voc, cons, esp, par;
        cout<<" FICH - P.O.O."<<endl;
        cout<<"frase: "<<endl;
        cout<<T.ver_frase();
        T.contar(voc,cons,esp,par);
        cout<<"\n";
        cout<<"\nTotal de letras: "<<cons+voc<<endl;
        cout<<"\nTotal de Vocales: "<<voc<<endl;
        cout<<"\nTotal de Consonantes: "<<cons<<endl;
        cout<<"\nTotal de Espacios: "<<esp<<endl;
        cout<<"\nTotal de Parrafos: "<<par<<endl;
}
```

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.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 5 - Ejercicio Adicional Nº 6

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <algorithm>
#include <sstream>
using namespace std;
int reemplaza(string arch_origen, string arch_destino, string cad1, string cad2){
	int ocurrencias=0;//La utilizo para contar los reemplazos
	ifstream lectura(arch_origen.c_str());//Creo el objeto para leer asociado al archivo fisico
	if(!lectura.is_open()){cout<<"Error";getchar();exit(0);}//Valido
	ofstream escritura(arch_destino.c_str());//Creo el objeto para escribir asociado al archivo fisico
	if(!escritura.is_open()){cout<<"Error";getchar();exit(0);}//Valido
	//Declaro variables a utilizar
	string cadena2, aux;
	stringstream aux2;
	int p_ini,p_fin,longitud= cad2.length();
	while(getline(lectura,cadena2)){//Leo la linea del archivo
		aux= cadena2;//Asigno al string aux
		aux2.str("");//Linpio aux2
		//Utilizo este recurso para pasar la linea a minusculas
		std::transform(aux.begin(), aux.end(), aux.begin(), ::tolower);
		//cout<<aux<<endl;//Descomentar para ver lectura por pantalla
```

```
                p_ini= 0;
                p_fin= aux.find(cad1);//Busco la cadena en la linea con la funcion miembro
find()
                while (p_fin !=string::npos){//Si la encontro cuenta (npos es el final del string)
                        ocurrencias++;//cuento 1 mas
                        aux2<<aux.substr(p_ini,p_fin)+cad2+aux.substr(p_fin+longitud);//Creo
aux2 con el reemplazo
                        aux= aux2.str();//Guardo la nueva cadena en aux
                        p_ini=p_fin+1;//Actualizo p_ini
                        p_fin= aux.find(cad1,p_ini);//Vuelvo a buscar en la linea
                }//Termino de buscar en la linea
                escritura<<aux<<endl;//Guardo la nueva linea en escritura
        }//Termino de leer el archivo */
        lectura.close();
        escritura.close();
        return ocurrencias;
}
//*************************************************************************
int main(int argc, char *argv[]) {
        int salida;
        string archivo_orig= "Adicionales//Quijote.txt";
        string archivo_dest= "Adicionales//Quijote_Trucho.txt";
        string busqueda= "molinos de viento";
        string cambia="turbina de agua";
        int r= reemplaza(archivo_orig, archivo_dest,busqueda,cambia);
        cout<<"Se ralizaron "<<r<<" reemplazos"<<endl;
        return 0;
}
```

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.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
```

```
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 5 - Ejercicio Adicional Nº 7

#include <iostream>
#include <cstdio>
#include <fstream>
#include <cstdlib>
#include <algorithm>
using namespace std;
void corregir_formato(string arch_origen, string arch_destino){
ifstream lectura(arch_origen.c_str());//Creo el objeto para leer asociado al archivo fisico
if(!lectura.is_open()){cout<<"Error";getchar();exit(0);}//Valido
ofstream escritura(arch_destino.c_str());//Creo el objeto para escribir asociado al archivo fisico
if(!escritura.is_open()){cout<<"Error";getchar();exit(0);}//Valido
//Declaro variables a utilizar
string cadena2, aux, car1, aux2;
int p_ini,p_fin;
while(getline(lectura, cadena2)){//Leo la linea del archivo
        aux+= cadena2;//Concateno
        aux+= "\n";//Enter, salto de linea
}
lectura.close();//Termino la lectura del archivo
//cout<<aux<<endl;//Descomentar para ver en pantalla
//Tengo el archivo en aux
//Utilizo este recurso para pasar la linea a minusculas
std::transform(aux.begin(), aux.end(), aux.begin(), ::tolower);
//Paso primer letra a mayusculas.
aux[0]= toupper(aux[0]);
//Punto y aparte
p_ini= 0;
p_fin= aux.find("\n");//Busco la cadena en la linea con la funcion miembro find()
while (p_fin != string::npos){//Si la encontro cuenta (npos es el final del string)
        car1 = aux[p_fin-1];
        if("." == car1){//Si el caracter anterior al <enter> es <punto>
                aux[p_fin+1]= toupper(aux[p_fin+1]);//Guardo la letra mayuscula
        }
        p_ini=p_fin+2;//Actualizo p_ini
        p_fin= aux.find("\n",p_ini);//Vuelvo a buscar en la linea
}//Termino busqueda en aux
//Punto y seguido
p_ini= 0;
p_fin= aux.find(". ");//Busco la cadena en la linea con la funcion miembro find()
while (p_fin !=string::npos){//Si la encontro cuenta (npos es el final del string)
        if(p_fin+2<aux.length()){
                aux[p_fin+2]= toupper(aux[p_fin+2]);//Guardo la letra mayuscula
                p_ini=p_fin+3;//Actualizo p_ini
                p_fin= aux.find(". ",p_ini);//Vuelvo a buscar en la linea
                }
        }//Termino de buscar en aux
```

```
escritura<<aux<<endl;//Guardo la nueva linea en escritura
//Termino de utilizar el archivo */
escritura.close();
//cout<<endl<<aux<<endl;//Descomentar para ver en pantalla
}
//*****************************************************************************
int main(int argc, char *argv[]) {
string archivo_orig= "Adicionales//Texto.txt";
string archivo_dest= "Adicionales//Texto2.txt";
corregir_formato(archivo_orig,archivo_dest);
return 0;
}
```