Practica #5
Archivo Secuencial
Prof. Andrés Arcia

Objetivos Generales

- Introducción a los archivos. (Archivos Texto, Archivos de Registros, Acceso Secuencial)
- Jerarquía de las clases manejadoras de archivos (ios).
- Formato de las cadenas de salida: formato de punto flotante, formato de cadenas, enteros.
- Clase archivos secuenciales.
- Operaciones sobre archivos secuenciales.

Objetivos Específicos

- Comprender los objetos manejadores de archivos de la librería estándar STL: ifstream, ofstream, iostream.
- Hacer una agenda electrónica que garantice las operaciones de inserción y búsqueda.

La interfaz de la clase agenda es:

```
class Agenda : public fstream
{
  public:
   Agenda(char * nombre_agenda);

   void agregar_contacto(Registro & reg)

   void buscar(string & cad, Registro * arr_reg, int & c)
}
```

Observe que una Agenda ES un archivo, por lo tanto Usted puede invocar los métodos de la clase fstream sin necesidad de colocar ningún objeto. Para los puristas, se puede hacer con (*this).

```
(*this).open("nombre_archivo.dat");
```

El método "agregar_contacto" agrega, en cualquier parte del archivo, un registro de tipo Registro. Por comodidad podría hacerlo al final del archivo.

El método "buscar" se encarga de hacer una búsqueda sobre todos los campos de todos los registros del archivo. Para ello, observe que,

se tiene una cadena y que podría ser subcadena de cualquiera de los campos en el archivo. Para entender cabalmente este método, remitase a la definición de subcadena.

Suponga que Usted ha declarado un tipo Registro con los siguientes campos:

```
struct RegAgenda {
string Nombre;
string Apellido;
int edad;
string Telefono;
string email;
};
```

La búsqueda sobre los campos de tipo string puede hacerse con la función "find" de la clase “string” de la STL. Por ejemplo:

```
#include <iostream>
#include <string>
#include <cstdlib>

using namespace std;
int main()
{
  string st("Aun hay camino por recorrer");

  // intentemos buscar la palabra camino en la
  // frase anterior.

  int r = st.find("camino");

  if (r >= 0)
    {
      cout << "Encontrado!!!" << endl;
    }

  cout << "Valor del resultado: " << r << endl;

  // Note que si la palabra no es encontrada dentro
  // de la cadena, el valor de retorno es -1.

  int cedula_como_entero = 10248583;

  char cedula_como_cadena[12];


  // La funcion sprintf se encarga de escribir en una
  // cadena (char *) algun contenido segun se especifique
```

```
  // en el formato. En el ejemplo, se convierte la cedula
  // de tipo entero, a una cedula del tipo char *. Esta
  // sera pasada luego a una variable tipo string.

  sprintf(cedula_como_cadena, "%i", cedula_como_entero);

  string st_ced(cedula_como_cadena);

  // Supongamos que queremos buscar cedulas que tengan
  // la cadena 102, por supuesto el ejemplo cumple con
  // esta restriccion.

  r = st_ced.find("102");

  if (r >= 0)
    {
      cout << "Encontrado!!!" << endl;
    }

  return 0;
}
```

La funcion buscar devuelve un arreglo de registros que cumplen con la cadena solicitada. La cadena solicitada debe ser buscada exahustivamente en todo el registro.

Es posible que algunos campos dentro del registro no sean del tipo cadena. Entonces para resolver este inconveniente, se puede hacer una conversión del tipo numérico al tipo cadena tal y como se muestra en el ejemplo anterior.

Supongamos que se tienen los siguientes registros:

Andres
Arcia
21
2402992
amoret@ula.ve
Camilo
G**arcia**
1
2440303
gcamilo@gmail.com

Si suministraramos la cadena “arcia” a nuestra función de búsqueda, entonces debería devolver ambos registros, pues la subcadena se encuentra en cada uno de ellos. Asegúrese de entender bien esta restricción.