Cadenas (strings) y Estructuras

Fundamentos de Programación
Fundamentos de Programación I

Operaciones básicas definidas para string

- Creación de variables:
```
string palabra, frase;
```
- Asignación:
```
frase = palabra;
frase = "hola";
```
- Acceso a los caracteres (como con vectores):
```
palabra[0]
```
- Comparación lexicográfica (==, !=, <, >):
```
frase == palabra
frase > palabra
```
- Lectura/escritura:
```
cin >> palabra;
getline (cin, frase); //lee frase de cin hasta
  encontrar el fin de línea
cout << frase << endl;
```

```
Necesita utilizar
#include <string>
```

Métodos para manipulación de string

- **Indica si la frase está vacia**
  - `vacia =palabra.empty()`
- **Devuelve la longitud de la cadena**
  - `i = palabra.length();`
- **Inserta palabra en la posición pos de frase**
  - `frase.insert(pos, palabra);`
- **concatena (une) palabra y "hola" y almacena el resultado en frase**
  - `frase = palabra + "hola";`
- **concatena (añade al final) palabra a frase**
  - `frase += palabra;`
  - `frase.append(palabra);`
- **borra num caracteres de frase desde la posición pos**
  - `frase.erase (pos,num);`

Métodos para manipulación de string

- **Sustituye (reemplaza) num caracteres de frase, empezando en la posición pos, por la cadena palabra**
  - `frase.replace (pos, num, palabra);`
- **busca palabra como una subcadena dentro de frase desde la posición pos, devuelve la posición donde la encuentra**
  - `i = frase.find(palabra,pos);`
- **devuelve la subcadena formado por num caracteres desde la posición pos de la frase**
  - `palabra = frase.substr(pos,num);`

Haz una función a la que se le pase como parámetro una cadena que almacena la fecha en formato DD/MM/AA y la convierta al formato Dia de Mes de Año. Por ejemplo la fecha 01/03/2001 se debe convertir a la cadena 01 de 03 de 2001.

02/03/1971  02 de 03 de 1971

¿Cómo la haríamos si el formato final fuera el mes en texto?
02 de marzo de 1971

¿Y si quisieramos quitar los 0 a la izquierda de los dias?
2 de marzo de 1971


```
#include <iostream>
#include <stdlib.h>
#include <string>

using namespace std;
string TransformarFecha(string fecha);
int main()
{

 string fecha1,fecha2;

 cout << "Introduce la fecha\n";
 cin>> fecha1;
 fecha2=TransformarFecha(fecha1);
 cout << fecha2;
 system("PAUSE");

 return 0;
}
string TransformarFecha(string fecha)
{
//Leo cadenas separadas por caracter '/'
string nuevafecha,dia,mes,anyo;
int inicio,pos1,pos2,tam;
inicio=0;
pos1 = fecha.find('/',inicio);
pos2 = fecha.find('/',pos1+1);
dia = fecha.substr(inicio,pos1-inicio);
inicio = pos1 + 1;
mes = fecha.substr(pos1+1,pos2-inicio);
tam = fecha.length();
inicio = pos2 + 1;
anyo = fecha.substr(pos2+1,tam-inicio);
//Formo la nueva cadena a partir de las subcadenas
nuevafecha = dia +" de " + mes + " de " + anyo;

return nuevafecha;
}
```

Manejo de cadenas en C++

Escriba un programa que lea el nombre de una persona en formato indicado:

```
Apellido, Nombre_de_pila, inicial_intermedia
```

y lo convierta al formato siguiente:

```
Lopez, Juan A.
```

Ejemplos: entradas: `Juan Antonio Lopez` … salida: `Lopez, Juan A.`
`Juan A. Lopez` … salida: `Lopez, Juan A.`
`Juan Lopez` … salida: `Lopez, Juan`

El programa deberá funcionar colocando un punto después de la inicial intermedia, aunque la entrada no contenga dicho punto. El programa deberá contemplar usuarios que no den un nombre intermedio o inicial. En tal caso, lógicamente la salida no contendrá una inicial intermedia (ver ejemplos) Deberá producir la salida:


```
//Programa que toma el nombre de una persona y lo cambia de formato
#include <iostream.h>
#include <string>

//Prototipos de funciones

void ProcesarNombreCompleto(string todo, string &primer,string &segundo, string &apellido);


//Función principal

int main()
{
   //Declaracion de variables
   string nom_comp;
   string npri,nseg,ape;
   string formatonuevo;

   //Leo el nombre de la entrada estandar
   cout << "Introduce tu nombre completo:\n";
   getline(cin,nom_comp);

   //Analizo la cadena introducida
   //La separo en 3 partes (1 nombre, 2 nombre y apellido)
   ProcesarNombreCompleto(nom_comp, npri,nseg, ape);

   //Reconstruyo el nombre con nuevo formato a partir de
   //las partes individuales
   formato = ape + ", " + npri + " " +  nseg;

   //Escribo el resultado
   cout << formato <<  endl;

   return 0;
}
```

```
//Funcion que procesa la cadena con el nombre completo

void ProcesarNombreCompleto(string todo, string & primer, string & segundo, string & apellido)
{
   //Declaracion de variables locales
   int posini, posfinal;
   int tamanyo;

   //Extraigo el primer nombre
   posini = 0;
   posfinal = todo.find(" ");
   primer = todo.substr(posini, posfinal - posini);

   //Ahora el segundo
   posini = posfinal + 1;
   posfinal=todo.find(" ", posini);

   //Si posfinal es -1 no hay segundo nombre
   if ( posfinal > 0)
   {
      //Cuando haya segundo nombre
      segundo=todo.substr(posini,posfinal-posini);
      posini=posfinal+1;

      //Modifico el segundo nombre
      tamanyo = segundo.length();
      segundo.erase(1, tamanyo - 1 );
      segundo = segundo + ".";
   }
   else
   {
            //Cuando no hay segundo nombre
       segundo = "";
   }
    //apellido
   posfinal = todo.length();
   apellido = todo.substr(posini,posfinal-posini);

   return ;

}
```

Estructuras (registros)

Es una agrupación de elementos de tipos diferentes. También se denomina en ocasiones *registro*.

Cada elemento se denomina campo, y se representa mediante un identificador propio. Cada campo puede ser de cualquier tipo.

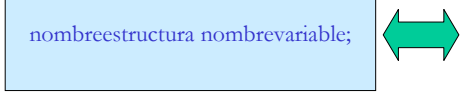
```
struct Nombre
{
Tipo Nombre_campo1;
Tipo Nombre_campo2;
...
}
```

```
struct agenda
{
  string nombre;
  string apellido;
  int numtelefono;
}
```

Uso de las estructuras:

Para declarar una variable de un tipo estructura dado:

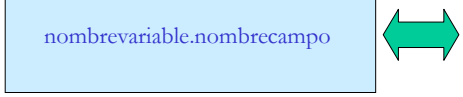
nombreestructura nombrevariable;



```
int main()
{
agenda miagenda;
…
}
```

Para acceder a cada campo individual:

nombrevariable.nombrecampo



```
int main()
{
agenda miagenda;

miagenda.nombre="Juan";

}
```


Calculo de calificaciones escolares

Escribe un programa para calcular la **nota final** de un grupo escolar que siga la siguiente política de calificación:

a) Hay 2 cuestionarios (calificados sobre 10 puntos) y 2 exámenes (parcial y final, sobre 100 puntos.)
c) Calificación final se obtiene de la siguiente forma:
El examen final supone el 50% de la nota, el parcial el 25% y los 2 cuestionarios completarán el 25% restante.

Objetivos:
1) Define una estructura para almacenar la información de cada estudiante.
2) El programa debe:
2.1) Pedir el nombre del alumno y sus notas parciales.
2.2) Calcular la nota final de cada alumno, de acuerdo a los datos anteriores.
2.3) Mostrar los datos introducidos y las calificaciones finales calculadas.

Ayuda: Comenzar con un programa que calcule las notas de un solo alumno

Nota: Tabla de Calificaciones
Sobresaliente = nota entre [90,100].
Notable = nota entre [70,90].
Aprobado = … [50,70].
Suspendido < 50.

```
//Programa que calcula calificaciones de alumnos
#include <iostream.h>
#include <stdlib.h>
#include <string>

const int MAXIMO = 100 ;
//Definición de estructuras
struct Alumno
{
  string n_alumno;
  int test1,test2;
  int parcial, final;
  string notaglobal;
};
//Definicion de tipos
typedef Alumno Curso[MAXIMO];

//Prototipos de funciones
void LeerRegistro(Alumno & alu);
void MostrarRegistro(Alumno alu);
void CalcularNotaRegistro(Alumno & alu);
```

```
 int main()
{

 Alumno elalu;

LeerRegistro(elalu);

CalcularNotaRegistro(elalu);

MostrarNotaRegistro(elalu);

return 0;
}
```

```
void LeerRegistro(Alumno &alu)
{
    //Leo cada elemento de la estructura de forma
    //independiente

    cout << "Nombre: " ;
    getline(cin,alu.n_alumno);
    cout << endl;
    cout << "Introduce las calificaciones\n";
    cout << "Test 1: ";
    cin >> alu.test1;
    cout << endl;
    cout << "Test 2: " ;
    cin >> alu.test2;
    cout << endl;
    cout << "Parcial: ";
    cin >> alu.parcial;
    cout << endl;
    cout << "Final: ";
    cin >> alu.final;

    return;

}
//Escribo los datos almacenados en el registro por pantalla

void MostrarRegistro(Alumno alu)
{
     cout << "Nombre: ";
     cout << alu.n_alumno << endl;

     cout << "\t\tCalificaciones: ";
     cout << "\t" << alu.test1 << " " << alu.test2 <<" " << alu.parcial << " " << alu.final << endl;

     cout << "\t\tCalificacion global: ";
     cout << "\t" << alu.notaglobal<< endl;

     return
}
```

```
//Calculo la nota final

void CalcularNotaRegistro(Alumno & alu)
{
     float notanumerica;

     notanumerica = (alu.test1 + alu.test2) * 25 / 20 + (alu.final / 2 )  + (alu.parcial / 4);

     if (notanumerica > 90.0)
        alu.notaglobal= "Sobresaliente";

     else if (notanumerica > 70.0)
        alu.notaglobal= "Notable";
        else if (notanumerica > 50.0 )
                alu.notaglobal="Aprobado";
                else
                        alu.notaglobal="Suspendido";

     return;

}
```



```
 int main()
{
   //Declaro un vector para almacenar la información de
   //los alumnos
    Curso micurso;
    int numero;
    bool seguir;
    int i, elementos; //Numero de alumnos introducidos

    seguir = true;
    i = 0;
    do{
         //Lee datos
         LeerRegistro(micurso[i]);

         //Calcula nota global
         CalcularNotaRegistro(micurso[i]);

         //Muestro el resultado
         MostrarRegistro(micurso[i]);

       //Compruebo condicion de salida
         if (i > MAXIMO)
            seguir = false;
         cout << "Para terminar pulse 0. Para continuar
                  1\n";
         cin >> numero;
         if (numero == 0)
            seguir = false;
         i++;

    }while (seguir);

    elementos = i;

    cout << "Notas almacenadas en esta sesion:" << endl;

    for (i = 0; i < elementos ; i++)
        MostrarRegistro(micurso[i]);

    return 0;
}
```

Manejo de Cadenas: Cifrado de datos

Una técnica de cifrado elemental consiste en lo siguiente: se dispone de una frase clave que llamamos llave. A cada carácter leído del mensaje que queremos cifrar, le aplicamos la operación XOR bit a bit con un carácter de llave y el resultado es el carácter cifrado. El carácter de llave aplicado es el siguiente en número de orden siguiendo un ciclo. Es decir si llave tiene 4 caracteres, cifraremos el primer carácter leído con llave[0], el segundo con llave[1], ... el quinto con llave[0] etc.

Este método tiene la propiedad de que cifrando el texto ya cifrado con la misma llave vuelve a aparecer el texto original.

Implementa una función que reciba como argumento una llave, una cadena y la devuelva la cadena encriptada.

Realiza una función que pida al usuario una frase, la encripte, muestre la cadena encriptada y después la desencripte con la misma función y muestre el resultado por pantalla.

NOTA:

En C++ puedes aplicar la operación lógica XOR sobre dos variables de tipo carácter de la siguiente manera:

```
char a='A';
char b='B';
char resul;
resul = a ^ b;   // el signo '^' es el XOR en C++.
```

Esto realiza la operación XOR sobre cada uno de los bits de a y b


```
// Programa que encripta una frase introducida por teclado según una clave tambien introducida por
//teclado

#include <iostream.h>
#include <string>

//Prototipos de funciones
string encriptar(string frase, string clave);


int main()
{

   string frase;
   string clave;
   string res,res2;

   cout << "Este programa encripta una frase introducida
             por teclado \n segun una clave tambien
             introducida por teclado\n";
   cout << "Introduce una frase para encriptar\n";
   //Leo la frase
   getline(cin,frase);

   //Leo la clave
   cout << "clave:";
   cin >> clave;
   res = frase;
   res2 = frase;

   res = encriptar(frase, clave);

   cout << " La frase encriptada es:\n";
   cout << res;

   cout << "Proceso inverso (desencripto)\n";
   res2 = encriptar(res,clave),
   cout << "La frase desencriptada es:\n");
   cout << res2;
   return 0;
}
```

```
//Funcion que encriptar una frase pasada como parametro
/7segun la clave tambien pasada como parametro
//devuelve la frase encriptada

string encriptar(string frase, string clave)
{
   string res;
   int a,b,c;
   int i,j;
   int longi;

   res = frase;
   //Recorro la cadena para obtener la frase encriptada
   for(i = 0; i < frase.length() ; i++)
   {
      a = int (frase[i]);

      //Calculo el indice de la clave
      j = i % clave.length();
      b = int (clave[j]);

      //Operación xor
      c = a ^ b;

      //Guardo el caracter encriptado
      res[i] = char( c );

   }


   return res;

}
```


Compañías aéreas

Haz un programa que almacene información sobre vuelos de compañías aéreas. Los datos
almacenados son: el número de vuelo, la compañía, la ciudad origen, la ciudad destino, el
número de plazas totales y el número de plazas libres. Un ejemplo puede ser:

1 Iberia Valencia Madrid 250 10
2 Iberia Madrid Buenos-Aires 300 50
3 AirFrance Valencia Paris 250 70

El programa deberá permitir realizar desde menú las siguientes operaciones:

a. Añadir vuelo
b. Borrar vuelo
c. Modificar el número de plazas libres.
d. Ver todos vuelos
e. Ver vuelos con plazas libres a un cierto destino.

```
#include <iostream>
#include <stdlib.h>
#include <string>

using namespace std;
struct vuelo{
 int id;
 string nombre;
 string origen;
 string destino;
 int totales;
 int libres;
} ;
const int MAX=200;

typedef vuelo VectorVuelos[MAX];

void InsertarVuelo(struct vuelo &miv);

void EscribirRegistro(struct vuelo miv);

void MostrarTodosVuelos(VectorVuelos v, int tam);

void BorrarVuelo(VectorVuelos v,int &tam,int idvuelo);

void MostrarVuelosDestino(VectorVuelos v, int tam, string destino);

void ModificarPlazasLibres(VectorVuelos v,int tam,int idvuelo,int libres);
```



```
int main()
{
      VectorVuelos losvuelos;
      int numvuelos = 0;
      int opcion, idvuelo, libres;
      bool seguir = true;
      string destino;
      do{
          cout  << "1. Insertar vuelo." <<endl;
          cout  <<  "2. Borrar vuelo." << endl;
          cout  << "3. Mostrar vuelo." << endl;
          cout  << "4. Modificar numero de plazas libres para un vuelo." << endl;
          cout  << "5. Mostrar vuelos con plazas libres para un destino." <<endl;
          cout  << "6.Salir" <<endl;
          cin >> opcion;
          switch(opcion)
          {
            case 1: InsertarVuelo(losvuelos[numvuelos]);
                    numvuelos ++;
                    break;
            case 2:
                    cout << " Introduce el identificador del vuelo\n";
                    cin >> idvuelo;
                    BorrarVuelo(losvuelos,numvuelos,idvuelo);
                    break;
            case 3: MostrarTodosVuelos(losvuelos, numvuelos);
                    break;
            case 4: cout << "Indica el numero de vuelo y de plazas libres\n";
                    cin >> idvuelo >> libres;
                        ModificarPlazasLibres(losvuelos,numvuelos,idvuelo,libres);
                    break;
            case 5: cout << "Consultar plazas vuelo con destino: ";
                    cin >> destino;
                    MostrarVuelosDestino(losvuelos, numvuelos, destino);
                    break;
            case 6: seguir = false;
          }
      }while(seguir);
      system("PAUSE");
      return 0;
}
```

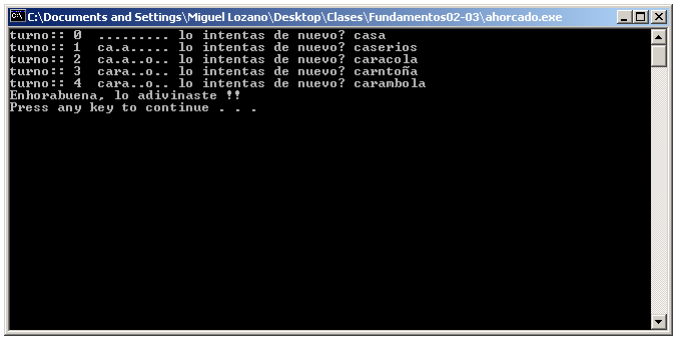
```
void InsertarVuelo(struct vuelo &miv)
{
  cout  <<"Id: ";
  cin >> miv.id;
  cout  <<"Nombre: ";
  cin >> miv.nombre;
   cout  <<"Origen: ";
  cin >> miv.origen;
  cout  <<"Destino: ";
  cin >> miv.destino;
  cout  <<"Numero de plazas totales: ";
  cin >> miv.totales;
  miv.libres = miv.totales;
 }
 void EscribirRegistro(struct vuelo miv)
 {
   cout << miv.id <<" ";
   cout << miv.nombre<<" ";
   cout << miv.origen<<" ";

   cout << miv.destino<<" ";
   cout << miv.totales<<" ";
   cout << miv.libres <<endl;

 }
 void MostrarTodosVuelos(VectorVuelos v, int tam)
 {
   int i;
   for (i = 0 ; i < tam ; i++)
       EscribirRegistro(v[i]);
   return;

 }
```

```
void BorrarVuelo(VectorVuelos v,int &tam,int idvuelo)
{
   int i= 0;
   int elemento;
   while (v[i].id != idvuelo && i < tam) //Buscar vuelo
   {
       i++;
    }
   elemento = i;
   if ( elemento < tam) //Lo he encontrado
   {
      for ( i = elemento ; i < tam -1 ; i++)
        v[i] = v[i + 1];
    tam --;
   }else
       cout << "No existe ese vuelo\n";
   return;
}
void ModificarPlazasLibres(VectorVuelos v,int tam,int idvuelo,int libres)
{
    int i, elemento;
    i = 0;

    while (v[i].id != idvuelo && i < tam)
    {
       i++;
    }
    elemento = i;
    if ( elemento < tam) //lo he encontrado
    {
        v[elemento].libres = libres;
        cout << "Modificación realizada\n";
    }
    return;
}
```

```
void MostrarVuelosDestino(VectorVuelos v, int tam, string destino)
 {
   int i;
   for (i = 0; i < tam ; i++)
   {
      if (destino == v[i].destino && v[i].libres > 0)
         EscribirRegistro(v[i]);

   }
 }
```


Juego del ahorcado:

a) Introducir palabra secreta (ej: carambola)
b) Permitir (MAX_INTENTOS) para adivinarla
Tras cada intento (turno), se mostrará el estado actual del proceso (ver ejemplo)



```
// Juego del ahorcado
#include <string>
const int MAX_INTENTOS = 10;
string turno(int n, string sol_act, string solucion);

int main (int argc ,char *argv[])
{
 string solucion, sol_act;
 int n_intentos=0, i;
 bool adivinada = false;

 cout << "Introduce la palabra a adivinar: ";
 cin >> solucion;

 for(i=0; i< solucion.length(); i++)
   sol_act.append(".");

 system("CLS"); // Borrar antes de empezar
 do
 {
    n_intentos++;
    cout << "Intento: " << n_intentos;
    sol_act = turno(sol_act, solucion);
    if(sol_act == solucion)
      adivinada = true;
 }while(!adivinada && n_intentos < MAX_INTENTOS);
```

```
if(adivinada)
    cout << "Enhorabuena, lo adivinaste !! " <<
endl;
 else
    cout << "Lástima ... quizás a la próxima " <<
endl;


  system("PAUSE");
}  // fin del programa ppal.
```

```
string turno(string sol_act, string solucion)
{
  string intento;
  int i;

  cin >> intento;

  for(i = 0; i< solucion.length(); i++)
   if (sol_act[i] == '.' && intento[i] == solucion[i])
     sol_act[i] = solucion[i];
   // cambio '-' por la letra recien acertada ...

  return sol_act;
}
```


Calculo de calificaciones escolares

Escribe un programa para calcular la nota de un grupo escolar que siga la siguiente política de calificación:
Hay 2 cuestionarios, cada uno de ellos se califica sobre 10 puntos.
Hay un examen parcial y un examen final, cada uno de los cuales se califica sobre 100 puntos.
El examen final representa el 50% de la nota, el parcial el 25% y los 2 cuestionarios juntos el 25% restante.

Las notas se representan de la siguiente forma:
Sobresaliente corresponde a una nota entre [90,100].
Notable corresponde a una nota promedio entre [70,90].
Aprobado entre [50,70].
Suspendido una nota promedio inferior a 50.

Define y emplea una estructura para almacenar la información de cada estudiante.
El programa leera los datos almacenados en un fichero y el resultado lo escribirá en otro fichero con el mismo formato.

Ejemplo de fichero de entrada con las notas:

```
Pedro Lopez
10 10 100 100
Juan Martinez
9 9 90 90
Elena Ibañez
8 8 80 80
```

Ejemplo de fichero salida con la nota global calculada:

```
Pedro Lopez
10 10 100 100 Sobresaliente
Juan Martinez
9 9 90 90 Sobresaliente
Elena Ibañez
8 8 80 80 Notable
```

```
//Programa que calcula calificaciones de alumnos
#include <iostream.h>
#include <stdlib.h>
#include <string>
#include <fstream.h>

const int MAXIMO = 100 ;
//Definición de estructuras
//Definicion de tipos

struct Alumno
{

  string n_alumno;
  int test1,test2;
  int parcial, final;
  string notaglobal;

};

typedef Alumno Curso[MAXIMO];

//Prototipos de funciones
bool LeerRegistro(ifstream& in_f, Alumno & alu);
void MostrarRegistro(ofstream& out_f, Alumno alu);
void CalcularNotaRegistro(Alumno & alu);
```



```
int main()
{
    //Declaro un vector para almacenar la información de los alumnos
    Curso micurso;
    int i, elementos;
    ifstream ent_f;
    ofstream sal_f;
    string nombre_in, nombre_out;

    cout << "Introduce mi nombre de fichero con los
             datos\n";
    //Leo nombre y almaceno en una variable tipo 'string'
    cin >> nombre_in;
    cout << nombre_in;
    ent_f.open( nombre_in.c_str() );
    if (!ent_f)
        cout << "Error abriendo fichero de datos."<< endl;
    else
    {
        i = 0;
        //Leo registros almacenados en el fichero
        while(LeerRegistro(ent_f, micurso[i]))
        {
           CalcularNotaRegistro(micurso[i]);
           i++;
        }
        elementos = i;
        ent_f.close(); //Cierro el fichero
        //Fichero donde guardar los resultados
        cout << "Introduce el nombre del fichero donde
                  guardar el resultado\n";
        cin >> nombre_out;
        //Convierto a cadena de caracteres tipo C
        sal_f.open( nombre_out.c_str() );
        if ( !sal_f)
                cout << "Error abriendo fichero\n";
        else
        {for (i = 0; i < elementos ; i++)
                MostrarRegistro(sal_f, micurso[i]);
            sal_f.close(); //Cierro el fichero
        }
    }
    return 0;
}
```

```
bool LeerRegistro(ifstream& in_f, Alumno &alu)
{
    //Leo cada elemento de la estructura de forma independiente

    //Leo el nombre
    getline(in_f,alu.n_alumno);
     //Leo las notas
    in_f >> alu.test1;
    in_f >> alu.test2;
    in_f >> alu.parcial;
    in_f >> alu.final;
    //Para saltar el salto de linea que queda en el buffer
    in_f.ignore();
    //Devuelvo true si todo ha ido bien, false en caso contrario
    return !(in_f.eof()) ;
}
//Escribo los datos almacenados en el registro a un archivo
void MostrarRegistro(ofstream& out_f, Alumno alu)
{
     //Escribo el resultado en el fichero asociado a out_f
     out_f << alu.n_alumno << endl;
     out_f << alu.test1 << " " << alu.test2 << " " << alu.parcial << " " << alu.final << " "
           << alu.notaglobal<< endl;
     return;
}
//Calculo la nota final
void CalcularNotaRegistro(Alumno & alu)
{
     float notanumerica;
     notanumerica = (alu.test1 + alu.test2) * 25 / 20 +
                    (alu.final / 2 )  + (alu.parcial / 4);
     if (notanumerica > 90.0)
        alu.notaglobal= "Sobresaliente";
     else if (notanumerica > 70.0)
        alu.notaglobal= "Notable";
        else if (notanumerica > 50.0 )
               alu.notaglobal="Aprobado";
             else
                 alu.notaglobal="Suspendido";
     return;
}
```