Pràctiques Estructura de Dades
Exemple d'utilització de les llibreries STL.

Robert Benavente i Jordi Gonzàlez, Març 99

Tenim una taula de notes d'alumnes en un fitxer de texte. Per cada alumne, tenim una fila en la que hi figura el seu DNI, Cognom, nom i dues notes.

```
45276523   Radford    Neal 3.5 5.5
73256255   Carpenter  Gail 7.8 6.0
45246578   Hinton      D   5.6 7.9
12351256   Sejnowsky   D   1.0 1.0
43623635   Grossberg   J   2.3 5.6
32151357   Sanger      R   7.9 5.4
34254556   Rossenfeld  D   8.2 1.2
24534523   Pappert     B   4.3 4.9
33215566   Friedmad    R 4.99 4.99
15826578   Falconer    K   9.4 5.0
```

Volem fer un programa, utilitzant les STL, que ens tregui una llista dels alumnes, ordenats per la nota mitjana que han obtingut.En aquesta llista volem que apareguin el DNI de l'alumne, la nota mitjana, i si estan aprovats o suspesos.

Per fer-ho, seguim els seguents passos:

1.- Triar l'opció **New** del menú **File** de l'entorn de treball del Visual C++.

2.- Seleccionar l'opció **Project**, donar un nom al projecte (exemple) i seleccionar l'opció **Win32 Console Application.**

3.- Tornar a triar l'opció **New** del menú **File**.

4.-Seleccionar l'opció **Files**, triar **C++ Source File**, marcar la casella **Add to Project** i donar un nom al fitxer (main.cpp)

5.-Repetir el pas 4 per crear el fitxer on hi haurà la implementació de les funcions de l'exemple (exemple.cpp).

6.-Repetir el pas 4 , però triant en aquest cas l'opció **C/C++ Header File**, per crear el fitxer de capçalera on hi haurà la definició de la classe que definirem (exemple.h).

Així, ja tindrem un fitxer 'exemple.h' on es faran les definicions de classes, un fitxer 'exemple.cpp' on hi haurà la implementació de les funcions membre de la classe i un fitxer 'main.cpp' on definirem una funcio main() i des d'on es realitzaran operacions amb la classe i les funcions que s'han d'implementar.

Ara ja podem codificar el nostre programa com si ho féssim desde el compilador normal de C per DOS. Haurem d'assegurar-nos però de fer 1 cosa a main.cpp:
1.-Posar en la primera línea:

#include "exemple.h"

using namespace std;

A l'hora d'utilitzar llistes formades per les nostres pròpies estructures, ens haurem d'assegurar que tinguin sobrecarregats el constructor de còpia, i els operadors == i < (el significat d'aquest operador, el definirem en funció del criteri d'ordenació que volguem seguir) . De la mateixa forma, si volem treballar amb stream_iterators, haurem de sobrecarregar els operadors << i >>.

Així el nostre fitxer **exemple.h** quedaria així:

```
#include "operatorsbase.h"
#include <vector>
#include <algorithm>
#include <numeric>
#include <iostream>
#include <fstream>
#include <list>
#include <iterator>
#include <string>

#ifndef __EXEMPLE_H
#define __EXEMPLE_H

using namespace std;

class CNotaFinal
{
public:
	string DNI_alumne;
	float valor;

	CNotaFinal (const CNotaFinal & in) {DNI_alumne=in.DNI_alumne;valor=in.valor;};
	CNotaFinal (string dni, float nota_in) {DNI_alumne=dni;valor=nota_in;};
	CNotaFinal () {};

	friend ostream& operator<< (ostream& out,const CNotaFinal & nota);
	friend istream& operator>>  (istream& in,CNotaFinal & nota);

	bool operator ==(const  CNotaFinal & a) const;
	bool operator <( const  CNotaFinal & a) const;
};


class CNotaEntrada
{
public:
	string DNI;
	string Nom;
	string Cognom;
	float Nota1;
	float Nota2;


	CNotaEntrada () {};	//s'ha de definir el constructor de còpia:
	CNotaEntrada ( const CNotaEntrada & nota)
{DNI=nota.DNI;Nom=nota.Nom;Cognom=nota.Cognom;Nota1=nota.Nota1;Nota2=nota.Nota2;};

	friend ostream& operator<< (ostream& out,CNotaEntrada & nota);
	friend istream& operator>> (istream& in, CNotaEntrada & nota);
	bool operator< ( const CNotaEntrada & a) const;
	bool operator== ( const CNotaEntrada & a) const;
};
```

```
class treu_DNI_i_mitja
{
public:
	CNotaFinal operator () (const CNotaEntrada & entrada)
	{
		CNotaFinal nova_nota (entrada.DNI,(entrada.Nota1+entrada.Nota2)/2.0);
		return nova_nota;
	}
};


#endif
```

El nostre fitxer **exemple.cpp** quedaria:

```
#include "operatorsbase.h"
#include "exemple.h"

// Implementació dels operadors

bool CNotaFinal::operator < ( const CNotaFinal & a) const
{
	return (valor>a.valor); //ens interessa ordenar els alumnes per nota de major a menor
								// per tant, implementem l'operador a
l'inrevés (donant-li el
								// sentit que convé a la nostra aplicació)
}

bool CNotaFinal::operator ==(  const CNotaFinal & a) const
{
	return DNI_alumne==a.DNI_alumne;
};


ostream& operator<< (ostream& cout,const CNotaFinal & nota)
{
	cout<<nota.DNI_alumne<<"
"<<nota.valor<<"("<<((nota.valor>=5)?"Aprovat)":"Suspes)")<<endl;
	return cout;
};

bool CNotaEntrada::operator< ( const CNotaEntrada & a) const
{
	return DNI<a.DNI;
};

bool CNotaEntrada::operator== ( const CNotaEntrada & a) const
{
	return DNI==a.DNI;
};
ostream& operator<< (ostream& cout,CNotaEntrada & nota)
{
	return cout;
}; // no l'utilitzem
```

```
istream& operator>> (istream& cin, CNotaEntrada & nota)
{
	cin>>nota.DNI;cin>>nota.Cognom;cin>>nota.Nom;cin>>nota.Nota1;cin>>nota.Nota2;
	return cin;
};
```

I el nostre programa **main.cpp** podria ser, per exemple:

```
#include <vector>
#include <algorithm>
#include <numeric>
#include <iostream>
#include <fstream>
#include <list>
#include <iterator>
#include <string>
#include "operatorsbase.h"

#include "exemple.h"

using namespace std;



int main(void)
{
	list <CNotaEntrada> llista_entrada; // crea la llista de notes (buida)
	list <CNotaFinal> sortida;	// crea la llista de sortida (buida)
	treu_DNI_i_mitja TDIM;

	ifstream fitxer ("notes.txt");	//obre els fitxers d'entrada


	istream_iterator <CNotaEntrada> entrada (fitxer); // els hi associa un stream iterator
	istream_iterator <CNotaEntrada> final_fitxer;	//iterador a final de fitxer

	// Llegeix la taula de entrada i amb els seus valors els inserta a llista_entrada
	copy (entrada,final_fitxer,inserter (llista_entrada,llista_entrada.end ()));

	// crear una nova llista per posari DNI+nota mitja
	sortida.insert (sortida.begin (),llista_entrada.size(),CNotaFinal("",0));
	//s'ha insertat a la llista tants elements com notes s'han llegit del fitxer de entrada
	//tots els elements de la nova llista tenen per DNI el del primer de la llista de entrada
	transform (llista_entrada.begin(),llista_entrada.end(),sortida.begin(),TDIM);

	// als algorismes STL se'ls hi pot passar tant un object function com una funcio normal com
	//"treu_DNI_i_mitja"
	// ara a la llista sortida hi tenim els DNI's i les notes mitjanes que tenen associats
	// també se'ls podria passar una funció normal.

	sortida.sort(); //orddenem la llista per el numero de DNI

	//treiem les notes per pantalla
	cout<<endl<<"   DNI     Nota Mitja "<<"\n\r\n\r";

	copy (sortida.begin (),sortida.end (),ostream_iterator <CNotaFinal> (cout,"\n\r"));
```

```
	return 0;
}
```

I el fitxer **operatorsbase.h** és de suport per als operadors de comparació i conté:

```
#if !defined(__OPERATORSBASE_H)
#define __OPERATORSBASE_H


template <class T>
inline bool operator!=(const T& x, const T& y){return !(x==y);}

template <class T>
inline bool operator>(const T& x, const T& y){return y<x;}

template <class T>
inline bool operator<=(const T& x, const T& y){return !(y<x);}

template <class T>
inline bool operator>=(const T& x, const T& y){return !(x<y);}

#endif
```

Com a resultat, el programa, treuria per pantalla:

```
    DNI     Nota Mitja
 15826578  7.2(Aprovat)
 73256255  6.9(Aprovat)
 45246578  6.75(Aprovat)
 32151357  6.65(Aprovat)
 33215566  4.99(Suspes)
  34254556  4.7(Suspes)
  24534523  4.6(Suspes)
  45276523  4.5(Suspes)
 43623635  3.95(Suspes)
   12351256  1(Suspes)
```