PRÀCTIQUES D'ESTRUCTURA DE DADES

Jordi Gonzàlez i Robert Benavente
Febrer 1999

Pràctica 1: Repàs de conceptes bàsics de C++.

1.-Objectius de la pràctica

En les dues primeres sessions de pràctiques es repassaran alguns conceptes bàsics de C++ necessaris per treballar amb la Standard Template Library (STL).

L'objectiu principal de la pràctica és treballar la definició i implementació de classes, constructors, destructors, sobrecàrrega d'operadors, streams, gestió de memòria i funcions i objectes template.

2.-Utilització del Visual C++ 5.0

La pràctica s'implementarà en l'entorn del Visual C++ 5.0. Per la realització de la pràctica s'ha de crear una aplicació basada en consola (*Console Application*) on es visualitzaran els resultats dels problemes proposats.

Per crear una aplicació basada en consola, haurem de seguir els passos següents:

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 (practica1) 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 la pràctica (vector.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 (vector.h).

Així, ja tindrem un fitxer 'vector.h' on es faran les definicions de classes, un fitxer 'vector.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.

3.-Descripció de la pràctica

1.- Definiu un nou tipus anomenat **simbol**:

typedef enum {$,w,_,x} simbol;

(Recordeu que les definicions dels tipus globals i de les classes van al fitxer '.h')

2.- Definiu una classe **VectorSimbols** que permeti definir vectors d'aquest nou tipus.

3.- Definiu un constructor per defecte, un constructor al qual se li indiqui el tamany del vector a crear i un destructor. La definició final de la classe, de moment, ha de quedar de la forma:

```
class VectorSimbols
{
....

public:
        VectorSimbols();
        VectorSimbols(unsigned int tamany);
        ~VectorSimbols();
};
```

Als constructors i al destructor afegiu el codi necessari que imprimeixi per pantalla les línies següents per tal de saber quin constructor o destructor s'ha cridat:

"S'ha cridat al constructor per defecte"
"S'ha cridat al constructor amb tamany ii"
"S'ha cridat al destructor"

(Recordeu que el codi de les funcions s'ha d'implementar al fitxer '.cpp')

4.-Definiu una funció **Size** que retorni el tamany del vector de símbols:

```
unsigned int Size ();
```

5.- Sobrecarregar els operadors següents:

[] → Per accedir als elements del vector de símbols.

```
simbol & operator [] (unsigned int posicio);
```

<< → Per enviar els elements del vector de símbols a un 'stream'.
En aquest cas, no volem que l'operador sigui un membre de la funció, sino que el definirem com una funció 'friend':

```
friend ostream& operator << (ostream & os,VectorSimbols & vec);
```

6.- A la funció ’main()’ feu la següent prova:

```
unsigned int i;
VectorSimbols a(10);

for (i=0;i<10;i++)
{
        a[i]=(simbol)’_’;
}

VectorSimbols b=a;
VectorSimbols c;
c=a;

for (i=0;i<10;i++)
{
        b[i]=(simbol)’$’;
}

for (i=0;i<10;i++)
{
        c[i]=(simbol)’x’;
}

cout<<”Vector a=”<<a<<endl;
cout<<”Vector b=”<<b<<endl;
cout<<”Vector c=”<<c<<endl;
```

A què es deu el resultat obtingut?
Per evitar-ho, definiu el constructor de còpia i l’operador s’assignació:

```
VectorSimbols (const VectorSimbols & copia);
void operator = (const VectorSimbols & vec);
```

Al codi del constructor de còpia afegiu el codi necessari per tal que cada cop que es cridi l’operador surti per pantalla la línia següent:

“S’ha cridat al constructor de còpia.”

Al codi de l’operador d’assignació afegiu el codi necessari per tal que cada cop que es cridi l’operador surti per pantalla la línia següent:

“S’ha cridat l’operador d’assignació.”

7.- A la funció *main()* del fitxer ‘main.cpp’ feu algunes operacions amb la classe que acabeu de definir i proveu el mètodes implementats. És important, que en aquestes proves incloeu tots els casos tractats (crear un objecte amb el constructor per defecte, un altre amb el constructor de còpia, etc...)

8.-Repetir tot el procés que s'ha explicat a l'apartat 2 de l'enunciat i creeu una nova '*Console Application*'. En aquest cas, anomeneu-lo 'Practica1b'. Ara, però, només s'han de crear un arxiu '.cpp' (main2.cpp) i un arxiu '.h' (vector2.h). Al fitxer vector2.h hi heu de posar la definició de la classe i la implementació de les funcions (les funcions 'template' d'una classe 'template' han d'estar al mateix fitxer que la definició de la classe).

9.-Modificar el codi de la classe **VectorSimbols** i de les seves funcions per tal d'obtenir una classe template **Vector** que permeti crear vectors d'altres tipus diferents.
La declaració de la classe ha de tenir la forma següent:

```
template <class T>
class Vector
{
...
}
```

La definició de les funcions passarà a ser:

```
template <class T> Vector<T>::Vector(unsigned int tamany)
{
...
}
```

L'operador '<<' haurà d'estar implementat dues vegades; una pel cas dels vectors de símbols i una altra pel cas general de vectors de qualsevol altre tipus.

10.- Definiu una funció **Suma** de la forma:

```
Vector<T>  Suma(const Vector<T> & vec);
```

Aquesta funció estarà implementada de dues formes diferents: una pel cas del vector de símbols i una altra per la resta de tipus. Per tant, inclourem la definició de les dues funcions en la definició de la classe.

Per tant, haureu de tenir:

```
template<class T> Vector<T> Vector<T>::Suma(const Vector<T> & vec)
{
...
}
```

i

```
Vector<simbol> Vector<simbol>::Suma(const Vector<simbol> & vec)
{
...
}
```

En aquest segon cas (pels vectors de símbols) la suma ha de quedar definida de la següent manera:

_ + _ = _
_ + simbol = simbol
simbol + _ = simbol
simbol + simbol = $

11.- A la funció *main()* del fitxer 'main2.cpp' feu algunes operacions amb la classe template que acabeu de definir i proveu el mètodes implementats. És important, que en aquestes proves treballeu amb vectors de diferents tipus per veure com funcionen els patrons (templates) en C++.

4.-Entrega

En acabar la segona sessió de la pràctica es comprovarà que tots els alumnes hagin realitzat correctament els problemes proposats. A més, s'haurà d'entregar un diskette amb el projecte Visual C++ corresponent.

En cas de no haver acabat la pràctica, s'entregarà un diskette amb la pràctica, independentment de l'estat en què es trobi. A la sessió següent, caldrà entregar la pràctica acabada.