Estructuras de Datos
Práctica 3: TDA lineales
Templates

Cristina Zuheros Montes

czuheros@ugr.es

November 1, 2020

Outline

Outline

¿Qué es un template?

- Mecanismo de abstracción que permite definir tipos complejos genéricos
- Una instanciación de una plantilla es la generación de una clase plantilla para un parámetro de la plantilla. Se genera una instancia. Ej: Racional<int> racional;
- Un template permite generar un número indeterminado de instancias para distintos tipos de datos

UNIVERSIDAD DE GRANADA

Motivación - Ej. Racional

La clase Racional está definida para int

```
czuheros@czuheros-ubuntu:~/Escritorio/racional$ ./bin/usoRacional
(0/1) Racional a
(2/4) Racional d
(2/4) Racional a tras asignarle el racional d
(5,7) Racional d 5/7
(6/4) Racional a 6/4
(62/28) Racional c=a+d
Racional c=a+d irreducible (31,14)
(208/56) Racional c=c+a
Racional c=c+a irreducible (26,7)
(6,4) Racional e 6/4
(3/2) Racional f 3/2
(3/5) Racional g 3/5
f y e son iguales
f y g no son iguales
Introduzca el primer racional usando formato (r/i):
(3.8/4.2)
Introduzca el segundo racional usando formato (r/i):
La suma es (66303,176808)
```

Motivación - Ej. Racional

El template Racional permite int, float, double...

```
czuheros@czuheros-ubuntu:~/Escritorio/racional_implicito$ ./bin/usoRacional
(0/1) Racional a
(2.3/4.4) Racional d
(2.3/4.4) Racional a tras asignarle el racional d
(5,7) Racional d 5/7
(6/4) Racional a 6/4
(62/28) Racional c=a+d
Racional c=a+d irreducible (31,14)
(208/56) Racional c=c+a
Racional c=c+a irreducible (26,7)
(6,4) Racional e 6/4
(3/2) Racional f 3/2
(3/5) Racional g 3/5
f y e son iguales
f y g no son iguales
Introduzca el primer racional usando formato (r/i):
(3.4/2.3)
Introduzca el segundo racional usando formato (r/i):
(3.7/8.2)
La suma es (36.39,18.86)
```

UNIVERSIDAD DE GRANADA

Outline

Instanciación implícita

- Método simple pero perdemos la encapsulación
- Respetamos la separación en ficheros de cabecera y de implementación mediante la directiva *include* en el .h
- Incluimos en el fichero de cabecera .h las declaraciones, e indirectamente las implementaciones
- Sólo tenemos que compilar el main.cpp que incluye el .h del template
- Se suele aplicar este método

Ej. Racional - Racional.h (en include)

```
...
template <typename T>
class Racional {
    private:
        T num;
        T den;
    ...
        Racional(T n, T d);
        Racional (const Racional<T>& c);
        T numerador ();
    ...
};
#include "Racional.cpp"
#endif
```

Ej. Racional - Racional.h (en include)

Hay que prestar atención con los operadores << y >>

```
...
template <typename T> class Racional;
template <typename T> ostream& operator<<(ostream&, const Racional<T>
template <typename T> istream& operator>>(istream&, Racional<T>&);
template <typename T>
class Racional {
...
    friend ostream& operator<< <T>(ostream& os, const Racional<T> &r)
    friend istream& operator>> <T>(istream& is, Racional<T> &r);
...
};
```

Ej. Racional - Racional.cpp (en src)

```
...
template <typename T>
Racional<T>::Racional(T n, T d ){
  num = n;
  den = d;
}
template <typename T>
Racional<T>::Racional (const Racional<T>& c){
  num = c.num;
  den = c.den;
}
template <typename T>
ostream& operator<< (ostream& os, const Racional<T> &r){
  os << '(' << r.num << ',' << r.den << ')';
  return os;
}
...
```

UNIVERSIDAD DE GRANADA

Ej. Racional

usoRacional.cpp

```
#include <iostream>
#include "Racional.h"
using namespace std;

int main(void){
   Racional<float> a,b,c,e,f,g,w,z;
   Racional<float> d(2.3,4.4);
   ...
}
```

Makefile

```
...
$(BIN)/usoRacional : $(OBJ)/usoRacional.o
        $(CXX) -o $(BIN)/usoRacional $(OBJ)/usoRacional.o

$(OBJ)/usoRacional.o : $(SRC)/usoRacional.cpp
        $(CXX) $(CPPFLAGS) -o $(OBJ)/usoRacional.o $(SRC)/usoRacional.cpp -I$(INCLUDE)
```

UNIVERSIDAD DE GRANADA

Outline

UNIVERSIDAD DE GRANADA

Instanciación explícita

- Es necesario conocer los usos que tendrá el template
- Respetamos la separación en ficheros de cabecera y de implementación de forma habitual
- Tenemos que compilar el main.cpp y el .cpp del template

Ej. Racional - Racional.h (en include)

```
...
template <typename T>
class Racional {
    private:
        T num;
        T den;
    ...
        Racional(T n, T d);
        Racional (const Racional<T>& c);
        T numerador ();
    ...
};
#endif
```

Ej. Racional - Racional.h (en include)

Hay que prestar atención con los operadores << y >>

```
...
template <typename T> class Racional;
template <typename T> ostream& operator<<(ostream&, const Racional<T>
template <typename T> istream& operator>>(istream&, Racional<T>&);
template <typename T>
class Racional {
...
    friend ostream& operator<< <T>(ostream& os, const Racional<T> &r)
    friend istream& operator>> <T>(istream& is, Racional<T> &r);
...
};
```

UNIVERSIDAD DE GRANADA

Ej. Racional - Racional.cpp (en src)

```
#include "Racional.h"
...
template <typename T>
Racional<T>::Racional(T n, T d ){
  num = n;
  den = d;
}
...
template class Racional<float>;
template class Racional<int>;
...
template ostream& operator<<(ostream&, const Racional<float>&);
template ostream& operator<<(ostream&, const Racional<int>&);
...
template istream& operator>>(istream&, Racional<float>&);
template istream& operator>>(istream&, Racional<int>&);
```

Ej. Racional

usoRacional.cpp

```
#include <iostream>
#include "Racional.h"
using namespace std;

int main(void){
   Racional<float> a,b,c,e,f,g,w,z;
   Racional<float> d(2.3,4.4);
   ...
}
```

Makefile

```
...
$(BIN)/usoRacional : $(OBJ)/usoRacional.o $(OBJ)/Racional.o
        $(CXX) -o $(BIN)/usoRacional $(OBJ)/usoRacional.o $(OBJ)/Racional.o

$(OBJ)/Racional.o : $(SRC)/Racional.cpp
        $(CXX) $(CPPFLAGS) -o $(OBJ)/Racional.o $(SRC)/Racional.cpp -I$(INCLUDE)

$(OBJ)/usoRacional.o : $(SRC)/usoRacional.cpp
        $(CXX) $(CPPFLAGS) -o $(OBJ)/usoRacional.o $(SRC)/usoRacional.cpp -I$(INCLUDE)
```

UNIVERSIDAD DE GRANADA

Actividad opcional (sin calificación)

Recuperar el TDA Racional trabajado en la P2 y pasarlo a Template mediante:

- Instanciación implícita
- Instanciación explícita

UNIVERSIDAD DE GRANADA

Actividad obligatoria

Diseñar el TDA Pila_Max como un Template

- Pila_max_Cola: usar cola proporcionada con templates
- Pila_max_VD: usar **preferentemente** la clase vector de la STL con templates
- Mediante instanciación implícita o explícita (a elegir una)

UNIVERSIDAD DE GRANADA