La clase de clases

En C++

Algoritmos y Estructuras de Datos I

4 de Junio de 2012

Especificación

Tipo

tipo Racional{
observador numerador(t : Racional) : $\mathbb{Z}$;
observador denominador(t : Racional) : $\mathbb{Z}$;
invariante denominador(r) > 0;
invariante mcd(denominador(r), numerador(r)) $== 1$;
}

Operaciones

problema Racional($n, d : \mathbb{Z}$) = r : Racional {
requiere : $d \neq 0$;
asegura : *formaRac*(n, d, numerador(r), denominador(r));
}

aux *formaRac*($a, b, n, m : \mathbb{Z}$) : Bool
$= n == signo(b) \times a/\mathrm{mcd}(a, b) \wedge m == |b/\mathrm{mcd}(a, b)|$
aux *signo*($x : \mathbb{Z}$) : $\mathbb{Z} = x/|x|$

Especificación

Tipo

tipo Racional{
    observador numerador(t : Racional) : $\mathbb{Z}$;
    observador denominador(t : Racional) : $\mathbb{Z}$;
    invariante denominador(r) $>$ 0;
    invariante mcd(denominador(r), numerador(r)) == 1;
}

Operaciones

problema numerador(r : Racional) = res : $\mathbb{Z}${
    asegura : res == numerador(r);
}

problema denominador(r : Racional) = res : $\mathbb{Z}${
    asegura : res == denominador(r);
}

Especificación

Tipo

tipo Racional{

observador numerador(t : Racional) : $\mathbb{Z}$;

observador denominador(t : Racional) : $\mathbb{Z}$;

invariante denominador(r) > 0;

invariante mcd(denominador(r), numerador(r)) $== 1$;

}

Operaciones

problema multiplicar(r : Racional , r_2 : Racional){

modifica r;

asegura : *formaRac*($n_1 \times n_2, d_1 \times d_2$, numerador($r$), denominador($r$));

aux $n_1 : \mathbb{Z} =$ numerador(*pre*(r));

aux $d_1 : \mathbb{Z} =$ denominador(*pre*(r));

aux $n_2 : \mathbb{Z} =$ numerador(r_2);

aux $d_2 : \mathbb{Z} =$ denominador(r_2);

}

Especificación

Tipo

tipo Racional{

observador numerador(t : Racional) : $\mathbb{Z}$;

observador denominador(t : Racional) : $\mathbb{Z}$;

invariante denominador(r) > 0;

invariante mcd(denominador(r), numerador(r)) $== 1$;

}

Operaciones

problema invertir(r : Racional){

requiere : numerador(r) $\neq 0$;

modifica r;

asegura : $formaRac(d_1, n_1,$ numerador(r), denominador(r));

aux $n_1 : \mathbb{Z} =$ numerador($pre(r)$);

aux $d_1 : \mathbb{Z} =$ denominador($pre(r)$);

}

Clases en C++

Definición

```
class Racional{




};
```

Uso

```
int main(){
   Racional r1(6, 13);
   ...
}
```

Clases en C++

Definición

```
class Racional{

      Racional(int n, int d);
      int numerador() const;
      int denominador() const;
      void multiplicar(const Racional &r);
      void invertir();



};
```

Uso

```
int main(){
   Racional r1(6, 13);
   ...
}
```

Clases en C++

Definición

```
class Racional{

      Racional(int n, int d);
      int numerador() const;
      int denominador() const;
      void multiplicar(const Racional &r);
      void invertir();

      int num;
      int den;
};
```

Uso

```
int main(){
   Racional r1(6, 13);
   ...
}
```

Clases en C++

Definición

```
class Racional{
   public:
      Racional(int n, int d);
      int numerador() const;
      int denominador() const;
      void multiplicar(const Racional &r);
      void invertir();
   private:
      int num;
      int den;
};
```

Uso

```
int main(){
   Racional r1(6, 13);
   ...
}
```

Clases en C++

Definición

```
class Racional{
   public:
      Racional(int n, int d);
      int numerador() const;
      int denominador() const;
      void multiplicar(const Racional &r);
      void invertir();
   private:
      int num;
      int den;
};
```

Uso

```
int main(){
   Racional r1(6, 13);
   ...
}
```

- Instancia de la clase

Implementación

Separación en archivos

- racional.h:

```
#ifndef racionalH
#define racionalH

class Racional{
    public:
       ...
    private:
       ...
};

#endif
```

Implementación

Separación en archivos

- racional.h:

```
#ifndef racionalH
#define racionalH

class Racional{
    public:
       ...
    private:
       ...
};

#endif
```

- racional.cpp

```
#include ''racional.h''
...
```

Implementación

Separación en archivos

- racional.h:
- racional.cpp

Acceso calificado

```
Racional(){
    ...
}
int denominador() const{
    ...
}
```

Implementación

Separación en archivos

- racional.h:
- racional.cpp

Acceso calificado

```
Racional::Racional(){
    ...
}
int denominador() const{
    ...
}
```

Implementación

Separación en archivos

- racional.h:
- racional.cpp

Acceso calificado

```
Racional::Racional(){
    ...
}
int Racional::denominador() const{
    ...
}
```

Especificación de métodos

Acá hay algo raro...

- problema Racional($n, d : \mathbb{Z}$) $= r$: Racional
- problema numerador(r : Racional) $= res : \mathbb{Z}$
- problema denominador(r : Racional) $= res : \mathbb{Z}$
- problema multiplicar(r : Racional , r_2 : Racional)
- problema invertir(r : Racional)

Especificación de métodos

Acá hay algo raro...

- problema Racional($n, d : \mathbb{Z}$) = r : Racional

```
Racional(int n, int d);
```

- problema numerador(r : Racional) = $res : \mathbb{Z}$

```
int numerador() const;
```

- problema denominador(r : Racional) = $res : \mathbb{Z}$

```
int denominador() const;
```

- problema multiplicar(r : Racional , r_2 : Racional)

```
void multiplicar(const Racional &r);
```

- problema invertir(r : Racional)

```
void invertir();
```

Especificación de métodos

Acá hay algo raro...

- problema Racional(this : Racional , $n, d : \mathbb{Z}$)

```
Racional(int n, int d);
```

- problema numerador(r : Racional) = $res : \mathbb{Z}$

```
int numerador() const;
```

- problema denominador(r : Racional) = $res : \mathbb{Z}$

```
int denominador() const;
```

- problema multiplicar(r : Racional , r_2 : Racional)

```
void multiplicar(const Racional &r);
```

- problema invertir(r : Racional)

```
void invertir();
```

Especificación de métodos

Acá hay algo raro...

- problema Racional(this : Racional , $n, d : \mathbb{Z}$)

```
Racional(int n, int d);
```

- problema numerador(this : Racional) = $res : \mathbb{Z}$

```
int numerador() const;
```

- problema denominador(r : Racional) = $res : \mathbb{Z}$

```
int denominador() const;
```

- problema multiplicar(r : Racional , r_2 : Racional)

```
void multiplicar(const Racional &r);
```

- problema invertir(r : Racional)

```
void invertir();
```

Especificación de métodos

Acá hay algo raro...

- problema Racional(this : Racional , $n, d : \mathbb{Z}$)

```
Racional(int n, int d);
```

- problema numerador(this : Racional) = *res* : $\mathbb{Z}$

```
int numerador() const;
```

- problema denominador(this : Racional) = *res* : $\mathbb{Z}$

```
int denominador() const;
```

- problema multiplicar(*r* : Racional , r_2 : Racional)

```
void multiplicar(const Racional &r);
```

- problema invertir(*r* : Racional)

```
void invertir();
```

Especificación de métodos

Acá hay algo raro...

- problema Racional(this : Racional , $n, d : \mathbb{Z}$)

```
Racional(int n, int d);
```

- problema numerador(this : Racional) = *res* : $\mathbb{Z}$

```
int numerador() const;
```

- problema denominador(this : Racional) = *res* : $\mathbb{Z}$

```
int denominador() const;
```

- problema multiplicar(this : Racional , *r* : Racional)

```
void multiplicar(const Racional &r);
```

- problema invertir(*r* : Racional)

```
void invertir();
```

Especificación de métodos

Acá hay algo raro...

- problema Racional(this : Racional , $n, d : \mathbb{Z}$)

```
Racional(int n, int d);
```

- problema numerador(this : Racional) = *res* : $\mathbb{Z}$

```
int numerador() const;
```

- problema denominador(this : Racional) = *res* : $\mathbb{Z}$

```
int denominador() const;
```

- problema multiplicar(this : Racional , *r* : Racional)

```
void multiplicar(const Racional &r);
```

- problema invertir(this : Racional)

```
void invertir();
```

Especificación de métodos

Acá hay algo raro...

- problema Racional(this : Racional , $n, d : \mathbb{Z}$)

  ```
  Racional(int n, int d);
  ```
  $\longrightarrow$ constructor
  - `Racional r1(6,13);`

- problema numerador(this : Racional) = $res : \mathbb{Z}$

  ```
  int numerador() const;
  ```

- problema denominador(this : Racional) = $res : \mathbb{Z}$

  ```
  int denominador() const;
  ```

- problema multiplicar(this : Racional , r : Racional)

  ```
  void multiplicar(const Racional &r);
  ```

- problema invertir(this : Racional)

  ```
  void invertir();
  ```

El parámetro implícito

Especificación

- problema Racional(this : Racional , $n, d : \mathbb{Z}$){
  requiere : $d \neq 0$;
  modifica this;
  asegura : *formaRac*(n, d, numerador(this), denominador(this));
  }

- problema numerador(this : Racional) = $n : \mathbb{Z}${
  asegura : $n ==$ numerador(this);
  }

- problema invertir(this : Racional){
  requiere : numerador(this) $\neq 0$;
  modifica this;
  asegura : numerador(this) $==$ denominador(*pre*(this));
  asegura : denominador(this) $==$ numerador(*pre*(this));
  }

El parámetro implícito

Especificación

- problema Racional(this : Racional , $n, d : \mathbb{Z}$){
  requiere : $d \neq 0$;
  modifica this;
  asegura : *formaRac*(n, d, numerador(this), denominador(this));
  }

- problema numerador(this : Racional) = $n : \mathbb{Z}${
  asegura : $n ==$ numerador(this);
  }

Uso

- *dot notation*

```
r1.multiplicar(r2);
```

Ejemplo

```
int main(){
   Racional r1(6,13);
   Racional r2(1,3);
   Racional r3(1,2);
```

(construimos tres instancias de `Racional`)

Ejemplo

```
int main(){
   Racional r1(6,13);
   Racional r2(1,3);
   Racional r3(1,2);

   r1.multiplicar(r2);
   r1.multiplicar(r3);


   int num = r1.numerador();
   int den = r1.denominador();
```

(construimos tres instancias de `Racional`)

(`r1` es el parámetro implícito de este y los próximos llamados)

¿Qué valor toma `num` aquí?
¿Y `den`?

Ejemplo

```
int main(){
   Racional r1(6,13);
   Racional r2(1,3);
   Racional r3(1,2);

   r1.multiplicar(r2);
   r1.multiplicar(r3);


   int num = r1.numerador();
   int den = r1.denominador();

   r1.invertir();

   num = r1.numerador();
   den = r1.denominador();
   return 0;
}
```

(construimos tres instancias de `Racional`)

(`r1` es el parámetro implícito de este y los próximos llamados)

¿Qué valor toma `num` aquí?
¿Y `den`?

¿Qué valor toma `num` aquí?
¿Y `den`?

Más sobre constructores

Constructor definido

```
Racional r1(6, 13);
    Racional(int n, int d);
```

Más sobre constructores

Constructor definido

```
Racional r1(6, 13);
    Racional(int n, int d);
```

Constructores importantes

- Constructor por defecto
  - `Racional r1;`

Más sobre constructores

Constructor definido

```
Racional r1(6, 13);
    Racional(int n, int d);
```

Constructores importantes

- Constructor por defecto
  - `Racional r1;`
  - Construcción implícita

Más sobre constructores

Constructor definido

```
Racional r1(6, 13);
    Racional(int n, int d);
```

Constructores importantes

- Constructor por defecto
  - `Racional r1;`
  - Construcción implícita

```
Racional();
```

Más sobre constructores

Definición de la clase

```
class MiClase {
public:
    MiClase(int n1, int d1, int n2, int d2);
   ...
private:
    Racional a;
    Racional b;
};
```

Inicialización de los atributos

```
MiClase::MiClase(int n1, int d1, int n2, int d2)
{
    a= Racional(n1, d1);
    b= Racional(n2, d2);
    ...
}
```

Más sobre constructores

Definición de la clase

```
class MiClase {
public:
    MiClase(int n1, int d1, int n2, int d2);
   ...
private:
    Racional a;
    Racional b;
};
```

Inicialización de los atributos

```
MiClase::MiClase(int n1, int d1, int n2, int d2)
 : a(n1, d1), b(n2, d2)
{
    ...
    ...
}
```