Sobrecarga: operador producto (*)

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}
```

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,[illegible]), h;
  h = f * g;
  f*=g*=h;
}
```

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction() : m_numerator(0), m_denominator(1)
{
}
```

h

m_numerator: ?
m_denominator: ?

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction() m_numerator(0), m_denominator(1)
{
}
```

h

m_numerator: 0
m_denominator: ?

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction() : m_numerator(0), m_denominator(1)
{
}
```

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction() : m_numerator(0), m_denominator(1)
{
}
```

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}
```

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,[illegible]), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

g

m_numerator: ?
m_denominator: ?

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,[illegible]), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

g

m_numerator: 3
m_denominator: ?

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,1), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,[illegible]), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,[illegible]), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,[illegible]), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fracti f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

f

m_numerator: ?
m_denominator: ?

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fracti f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

f

m_numerator: 1
m_denominator: ?

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fracti f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(nu m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

f

m_numerator: 1
m_denominator: 2

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fracti f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

f

m_numerator: 1
m_denominator: 2

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fracti f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

// fraction.cpp
Fraction::Fraction(const int num, const int den) :
    m_numerator(num), m_denominator(den)
{
    if(m_denominator == 0)
    {
        cout << “error: 0 passed in as denominator” << endl;
        exit(1);
    }
}
```

f

m_numerator: 1
m_denominator: 2

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

Operador producto: ejemplo

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}
```

f

m_numerator: 1
m_denominator: 2

g

m_numerator: 3
m_denominator: 4

h

m_numerator: 0
m_denominator: 1

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    return (result*=rhs);
}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
  Fraction result(lhs);
  return (result*=rhs);
}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*

m_numerator: ?
m_denominator: ?

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
  Fraction result(lhs);
    return (result*=rhs);
}

Fraction::Fraction(const Fraction & source) :
    m_numerator(source.m_numerator), m_denominator(source.m_denominator) {}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*

m_numerator: 1
m_denominator: ?

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
  Fraction result(lhs);
  return (result*=rhs);
}

Fraction::Fraction(const Fraction & source) :
  m_numerator(source.m_numerator), m_denominator(source.m_denominator) {}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*

m_numerator: 1
m_denominator: 2

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
  Fraction result(lhs);
    return (result*=rhs);
}

Fraction::Fraction(const Fraction & source) :
    m_numerator(source.m_numerator), m_denominator(source.m_denominator) {}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    retu(result*=rhs);
}
```

result en operator*

m_numerator: 1
m_denominator: 2

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*
rhs en operator*=

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*
this en operator=

m_numerator: 1
m_denominator: 2

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    return (result*=rhs);
}
Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*
rhs en operator*=

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*
this en operator=

m_numerator: 1
m_denominator: 2

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    return (result*=rhs);
}
Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*
rhs en operator*=

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*
this en operator=

m_numerator: 3
m_denominator: 2

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    retu(result*=rhs);
}
Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*
rhs en operator*=

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*
this en operator=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    retu(result*=rhs);
}
Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*
rhs en operator*=

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result en operator*
this en operator=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    retu(result*=rhs);
}
Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    retu(result*=rhs);
}
```

result en operator*

m_numerator: 3
m_denominator: 8

f en main
lhs en operator*

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}

Fraction operator* (const Fraction & lhs, const Fraction & rhs)
{
    Fraction result(lhs);
    return (result*=rhs);
}
```

result en operator*

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  f * g;
  f*=g*=h;
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 0
m_denominator: 1

result from operator*

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 3
m_denominator: 8

result from operator*

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main

m_numerator: 3
m_denominator: 4

h en main

m_numerator: 3
m_denominator: 8

f en main

m_numerator: 1
m_denominator: 2

g en main
this en operator=

m_numerator: 3
m_denominator: 4

h en main
rhs en operator*=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main
this en operator=

m_numerator: 3
m_denominator: 4

h en main
rhs en operator*=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main
this en operator=

m_numerator: 9
m_denominator: 4

h en main
rhs en operator*=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main
this en operator=

m_numerator: 9
m_denominator: 4

h en main
rhs en operator*=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main
this en operator=

m_numerator: 9
m_denominator: 32

h en main
rhs en operator*=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main
this en operator=

m_numerator: 9
m_denominator: 32

h en main
rhs en operator*=

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}
```

f en main

m_numerator: 1
m_denominator: 2

g en main

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8

f en main
this en operator=

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*=

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
this en operator=

m_numerator: 1
m_denominator: 2

g en main
rhs en operator*=

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
this en operator=

m_numerator: 9
m_denominator: 2

g en main
rhs en operator*=

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
this en operator=

m_numerator: 9
m_denominator: 64

g en main
rhs en operator*=

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

f en main
this en operator=

m_numerator: 9
m_denominator: 64

g en main
rhs en operator*=

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}

Fraction& Fraction::operator*= (const Fraction & rhs)
{
    m_numerator*=rhs.m_numerator;
    m_denominator*=rhs.m_denominator;
    return (*this);
}
```

```
//main.cpp
...
int main()
{
  Fraction f(1,2), g(3,4), h;
  h = f * g;
  f*=g*=h;
}
```

f en main

m_numerator: 9
m_denominator: 64

g en main

m_numerator: 9
m_denominator: 32

h en main

m_numerator: 3
m_denominator: 8