



Temas

- Ejemplo: La clase Fracción
- Funciones Friend
- Sobrecarga de operadores



Sobrecarga de operadores y
manejo de archivos de texto

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Ejemplo: La clase Fracción

Empecemos revisando un ejemplo: La clase Fracción

```
class Fracción {  
    public:  
        Fraccion(int, int);    //Inicializa con los valores que recibe  
        Fraccion();           //Inicializa con el valor 1/1  
        void Muestra();       //Muestra con el formato num/den  
    private:  
        int Num, Den;  
};
```

// CONTINUA

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Ejemplo: La clase Fracción

```
// Constructor que inicializa una fracción con el valor de 1  
Fraccion::Fraccion()  
{  
    Num = 1;  
    Den = 1;  
}  
  
// Constructor que inicializa una fracción con el valor dado por el usuario  
Fraccion::Fraccion(int N, int D)  
{  
    Num = N;  
    Den = D;  
} // CONTINUA
```

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Ejemplo: La clase Fracción

```
// Función que muestra la fracción en la forma acostumbrada
void Fraccion::Muestra()
{
    cout << Num << '/' << Den;
}
```

Se requiere un método llamado `valorDecimal` que calcule y regrese el valor decimal de la fracción dada. A continuación se muestra una versión para este método:

Dentro de la clase se declara el prototipo para el método:

```
Class Fracción {
    public:
        double Valor();    // Valor decimal de la fracción
        ...
};
```

//CONTINUA

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Ejemplo: La clase Fracción

Se define el método:

// Método que obtiene el valor decimal de la fracción

double Fracción:: Valor()

```
{  
    return Num*1.0/Den;  
}
```

Se manda llamar de la siguiente forma:

x = A.Valor();



Funciones Friend

- Una función friend es una función que no es parte de la clase, pero puede ver la parte privada de la clase.
- Se especifica poniendo la palabra friend seguida del prototipo de la función dentro de la declaración de la clase.
- Modifiquemos nuestro ejemplo para que la función Valor sea friend de la clase Fracción.
- Dentro de la clase se especifica el prototipo del método anteponiendo la palabra friend.

Funciones Friend

```
Class Fracción {  
    friend double Valor(Fraccion f); // Valor decimal de la fracción  
    ...  
};
```

Se define la función libre; es decir, no es parte de la clase:

// Función friend que obtiene el valor decimal de la fracción

```
double Valor(Fraccion f)  
{  
    return f.Num*1.0/f.Den;  
}
```

Se manda llamar de la siguiente forma

```
x = Valor(A);
```

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

Sobrecarga

- Como ya se explicó antes, el lenguaje C++, permite que se definan funciones diferentes con el mismo nombre, para dar claridad a los programas, y evitarle al programador pensar en nombres diferentes cuando la funcionalidad de un módulo es la misma.
- El único requisito es que cada función tenga diferente cantidad de parámetros, o bien, parámetros de diferente tipo.

Sobrecarga de Operadores

- Los operadores también pueden ser sobrecargados para que desempeñen funciones adicionales a las que tienen predefinidas.

Sobrecarga de operadores y manejo de archivos de texto



Sobrecarga de operadores

- Esta funcionalidad tiene sentido cuando el significado de un operador se acopla a un nuevo tipo de dato (definido a través de una clase).
- La sobrecarga se realiza al utilizar como nombre de la función la palabra operator y el símbolo del operador.
- Un operador que se sobrecarga tendrá la misma prioridad de ejecución y el mismo número de operandos que la del operador original.
- Algunos de los operadores que se pueden sobrecargar se muestran enseguida:

ARITMETICOS: + , - , * , / , % , ++ , --

RELACIONALES: < , > , <= , >= , == , !=

LOGICOS: && , || , !

OTROS: >> , <<

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

- Utilicemos ahora el operador ! para representar el valor decimal de una fracción.
- Dentro de la clase se especifica el prototipo de la sobrecarga del operador anteponiendo la palabra friend.

```
class Fraccion  
{  
    friend double operator !(Fraccion f);  
    ...  
};
```

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

- Se define la función libre; es decir, no es parte de la clase, nota que se cambia el nombre por operator !:

// Sobrecarga del operador unitario ! para obtener el valor decimal de la
// fracción

```
double operator !(Fraccion f)
```

```
{
```

```
    return f.Num*1.0/f.Den;
```

```
}
```

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

Se manda llamar de la siguiente forma:

`x = !A;`

- Nota que al hacer la sobrecarga del operador solamente cambia el nombre de la función y la forma de llamarla, todo lo demás es igual.
- Además el significado original del operador se ignora, ahora tendrá el significado que le damos con la sobrecarga del operador.

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

Ejemplo completo:

```
// Ejemplo de la clase Fraccion, es para mostrar el uso de la sobrecarga  
// de operadores que son friend o que no son friend pero no necesitan  
// los datos de la clase (como es el caso del operador ++).
```

```
#include <iostream.h>
```

```
class Fraccion
```

```
{
```

```
    friend Fraccion operator + (Fraccion, Fraccion); // Suma
```

```
    friend Fraccion operator - (Fraccion, Fraccion); // Resta
```

```
    friend double Valor(Fraccion f); // Valor decimal de la fraccion
```

```
    friend int operator > (Fraccion, Fraccion); // Comparacion por >
```

```
    friend double operator !(Fraccion f); // Valor decimal de la fraccion
```

```
public:
```

```
    Fraccion(int, int);
```

```
    Fraccion();
```

```
    void Muestra();
```

```
private:
```

```
    int Num, Den;
```

```
}; // CONTINUA
```


Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

// Constructor que inicializa una fraccion con el valor de 1

```
Fraccion::Fraccion()  
{  
    Num = 1;  
    Den = 1;  
}
```

// Constructor que inicializa una fraccion con el valor dado por el usuario

```
Fraccion::Fraccion(int N, int D)  
{  
    Num = N;  
    Den = D;  
}
```

// Funcion que muestra la fraccion en la forma acostumbrada

```
void Fraccion::Muestra()  
{  
    cout << Num << '/' << Den << endl;  
}
```

// CONTINUA

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

// Sobrecarga del operador + binario (suma de fracciones)

Fraccion operator + (Fraccion f1, Fraccion f2)

{

Fraccion res;

res.Num = (f1.Num*f2.Den + f2.Num*f1.Den);

res.Den = f1.Den*f2.Den;

return res;

}

// Sobrecarga del operador - binario (resta de fracciones)

Fraccion operator - (Fraccion f1, Fraccion f2)

{

Fraccion res;

res.Num = (f1.Num*f2.Den - f2.Num*f1.Den);

res.Den = f1.Den*f2.Den;

return res;

}

// CONTINUA

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

```
// Sobrecarga del operador ++ (sumar 1 unidad a la fraccion)
// Esta sobrecarga NO esta definida como friend y NO es parte de la clase
Fraccion operator ++ (Fraccion f)
{
    Fraccion res, uno;

    res = f + uno;

    return res;
}
// Funcion friend que obtiene el valor decimal de la fraccion
double Valor(Fraccion f)
{
    return f.Num*1.0/f.Den;
}
// Sobrecarga del operador unitario ! para obtener el valor decimal de la fracc
double operator !(Fraccion f)
{
    return f.Num*1.0/f.Den;
} // CONTINUA
```

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

```
// Sobrecarga del operador > (mayor que) regresa verdadero o falso
int operator > (Fraccion A, Fraccion B)
{
    if (!A > !B)
        return 1;
    else
        return 0;
}
int main()
{
    int x, y;
    cout << "Teclea el valor de A (2,5)";
    cin >> x >> y;
    Fraccion A(x,y), B(3, 4), C;

    cout << "A ";
    A.Muestra();
    cout << "B ";
    B.Muestra();
    // CONTINUA
```

Sobrecarga de operadores y manejo de archivos de texto



TECNOLÓGICO
DE MONTERREY



Sobrecarga de operadores

```
C = A + B;  
cout<< "Suma ";  
C.Muestra();  
C = A - B;  
cout << "Resta ";  
C.Muestra();  
C = ++B;  
cout << "Incremento ";  
C.Muestra();  
cout<<"Valor de la fraccion ";  
cout<<Valor(A);  
cout<<"\n Valor de la fraccion ";  
cout<< ! A;  
if (A > B)  
    cout<<"\n A es mayor que B"<<endl;  
else  
    cout<<"\n A no es mayor que B"<<endl;  
  
return 0;  
}
```