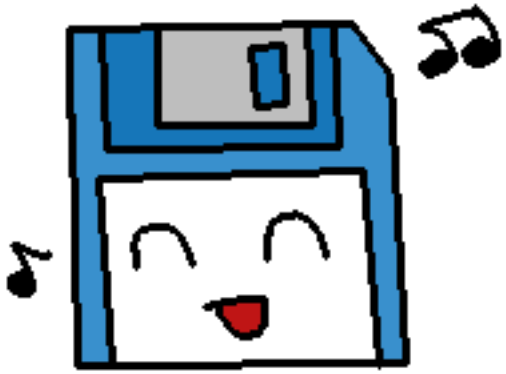
Operator Overloading

Operator Overloading

Operator Overloading would be used within a **class**, to specify how your custom object handles math (+, -, *, /), streaming (<<, >>), subscripts([ ]) and other things.



Operator Overloading

Why overload operators?

Writing a custom class and adding the functionality

Example: A **fraction** class with a numerator/denominator

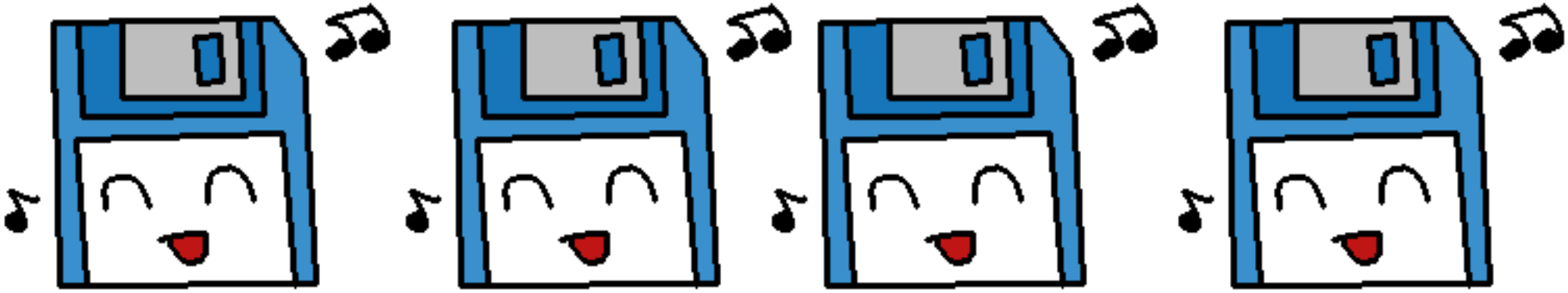
Would implement addition, subtraction, multiplication, and division

Example: Your own **list** class

Want to overload subscript [ ] operators to get a specific element.

Operator Overloading

Here is a sample of some of the common types of operator overloading, for reference:



Overloading Arithmetic

Declaration

In MyClass.h

```
class MyClass
{
    public:
    friend MyClass operator+( const MyClass& item1, const MyClass& item2 );
    friend MyClass operator-( const MyClass& item1, const MyClass& item2 );
    friend MyClass operator*( const MyClass& item1, const MyClass& item2 );
    friend MyClass operator/( const MyClass& item1, const MyClass& item2 );
};
```

Implementation

In MyClass.cpp

```
MyClass operator+( const MyClass& item1, const MyClass& item2 )
{
    // ...
}
```

Notice how this function definition is not part of MyClass. It is just a normal function.

Overloading Comparison

Declaration

In MyClass.h

```
class MyClass
{
    public:
    friend bool operator==( MyClass& item1, MyClass& item2 );
    friend bool operator!=( MyClass& item1, MyClass& item2 );
    friend bool operator<( MyClass& item1, MyClass& item2 );
    friend bool operator>( MyClass& item1, MyClass& item2 );
    friend bool operator<=( MyClass& item1, MyClass& item2 );
    friend bool operator>=( MyClass& item1, MyClass& item2 );
};
```

Implementation

In MyClass.cpp

```
bool operator==( MyClass& item1, MyClass& item2 )
{
    // ...
}
```

Relational Operators are also just friend functions; not members of the class itself.

Overloading Streams

Declaration

In MyClass.h

```
#include <fstream>
class MyClass
{
    public:
    friend ostream& operator<<( ostream& out, MyClass& item );
    friend istream& operator>>( istream& in, MyClass& item );
};
```

Implementation

In MyClass.cpp

```
ostream& operator<<( ostream& out, MyClass& item )
{
    out << item.textField << endl;
    return out;
}

istream& operator>>( istream& in, MyClass& item )
{
    in >> item.loadedField;
}
```

Stream Operators are also just friend functions

Overloading Subscript

Declaration
In MyClass.h

```
class MyClass
{
    public:
    int& operator[]( const int index );
    private:
    int m_data[];
};
```

Implementation
In MyClass.cpp

```
int& MyClass::operator[]( const int index )
{
    return m_data[ index ];
}
```

Note:
For Arithmetic, Stream, and Comparison, they <u>did not</u> belong to the **MyClass** class and were only <u>friends</u> of the **MyClass** class.

Here, the operator[ ] <u>is part of **MyClass**</u>.

Secondly:
The operator[ ] <u>does not need to return an int</u>, it will return whatever data-type you're wanting to access.

Overloading Assignment

Declaration
In MyClass.h

```
class MyClass
{
    public:
    MyClass& operator=( const MyClass& rhs );
    private:
    int variable1;
    float variable2;
};
```

Note:
“rhs” stands for
“Right-Hand Side”

Implementation
In MyClass.cpp

```
MyClass& MyClass::operator=( const MyClass& rhs )
{
    if ( this == &rhs )
        return *this;

    variable1 = rhs.variable1;
    variable2 = rhs.variable2;

    return *this;
}
```

MUST check for self-assignment!

There will be problems if the user tries to set an object to itself (without this check!)

Operator Overloading examples