function and operator overloading
friend classes
Topics

● function and operator overloading
● friend classes
● command line args
● classes vs. struct
Polymorphism

“the condition of occurring in several different forms.”

In C++, we have two forms that we will discuss here:

- Function overloading
- Operator overloading
Function Overloading

MyArray();

MyArray(int count);

Notice that these two function declarations share a name but have different parameter signatures. This is something that you may have seen in Java already. We see it commonly in constructors but its use is not restricted to that case.

Example: Array2D constructor
Operator Overloading - aka ad-hoc polymorphism

is a form of polymorphism in which some or all operators are treated as polymorphic functions.

C++ allows you to specify more than one definition for an operator (like +, = or ==) in the same scope.

It can easily be emulated using function calls and is considered “syntactic sugar”. That is, it makes source code easier to understand but does not add functionality that can not be accomplished in other ways.
Reference

We can look at a list of operators that can be overloaded here. You are NOT bound to make the functionality match that of other cases of their use. However, you should avoid using them in ways that confuse those reading your code (including yourself).

Assignment Operator - “=”

What does this mean?

Obj1 = Obj2;

The assignment operator (operator=) has special properties. There are two important cases:

- copy assignment
- move assignment

We will look at these in some detail on Wednesday.
These Can NOT Be Overloaded

- `?:` (conditional)
- `.` (member selection)
- `.*` (member selection with pointer-to-member)
- `::` (scope resolution)
- `sizeof` (object size information)
- `typeid` (object type information)
- `static_cast` (casting operator)
- `const_cast` (casting operator)
- `reinterpret_cast` (casting operator)
- `dynamic_cast` (casting operator)
When Would One Overload an Operator?

What does this mean?

\[ \text{Obj3} = \text{Obj1} + \text{Obj2}; \]

It has no meaning until it has been defined.
When Would One Overload an Operator?

What does this mean?

\[ \text{Obj3} = \text{Obj1} + \text{Obj2}; \]

If the objects represent quantities then we understand this pretty well. If they are more complicated, then we have to define what it means.
When Would One Overload an Operator?

What does this mean?

```
Obj3 = Obj1 + Obj2;
```

string Obj1 = “Hello”;
string Obj2 = “World”;
Operator Overloading - "<<"

A useful operator that you may want to overload is "<<".
What is a Friend Class?

- prototypes for friend functions appear in the class declaration
- friends are not member functions
- defined outside that class' scope
- can access all private and protected members of the class.
appear in the class definition

class MyArray {
public:

    MyArray();
    MyArray(int count);
    ~MyArray();

    int at(int location);
    int size();

    friend std::ostream& operator << (std::ostream& os, MyArray& arr);
};
IMPLEMENTED outside that class' scope (MyArray:: not present)

```cpp
std::ostream& operator << (std::ostream& os, MyArray& arr)
{
    for (int j=0; j<arr.count; j++){
        os<<arr.at(j)<<"  
    }
    os<<endl;
    os<<endl;
    return os;
}
```
Operator Overloading - “<<” (not friend - size() is public)

```cpp
std::ostream& operator << (std::ostream& os, MyArray& arr)
{
    for (int j=0; j<arr.size(); j++) {
        os << arr.at(j) << "  " << endl;
    }
    os << endl;
    return os;
}
```
C++ Command Line Arguments

main (int argc, char ** argv)
Links

Operator Overloading:

https://www.tutorialspoint.com/cplusplus/cpp_overloading.htm

Friend Class:

https://www.tutorialspoint.com/cplusplus/cpp_friend_functions.htm