Object Oriented Programming

Lecture

Operator Overloading

Some More Binary Operators

Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int x, y;
public:
    myclass(int a, int b)
    {
        x = a;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?

No, this code will not give any error:

1) The default assignment operator (operator=) performs a member-wise copy, meaning bar.x = foo.x and bar.y = foo.y

2) bar gets the values of foo, copying both x and y

Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int x, y;
public:
    myclass(int a, int b)
    {
        x = a;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?

Member-wise copy

Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

};
```

This should be:

```
*x = a;
//value of x=a
*y=b;
//value of y=b
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?

Yes, this code will give an error

1) Shallow Copy Problem
The class uses dynamic memory allocation (new), but does not define a copy assignment operator.
The default assignment operator does shallow copying, meaning:
bar = foo; // Copies the pointer addresses, not the actual data.
Now, both foo and bar point to the same memory locations.
2) There is no destructor in the class to release allocated memory (delete), leading to memory leaks.

Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?

Be careful with member-wise copy

- If member data is a pointer, the pointer address is copied
- this could be disastrous

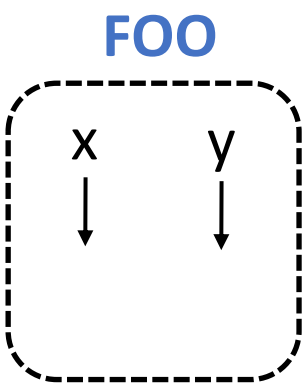
Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?



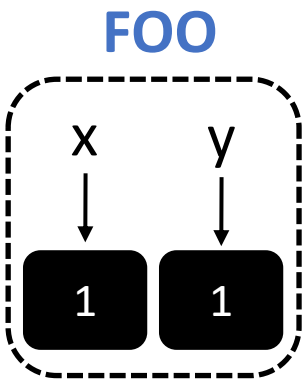
Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?



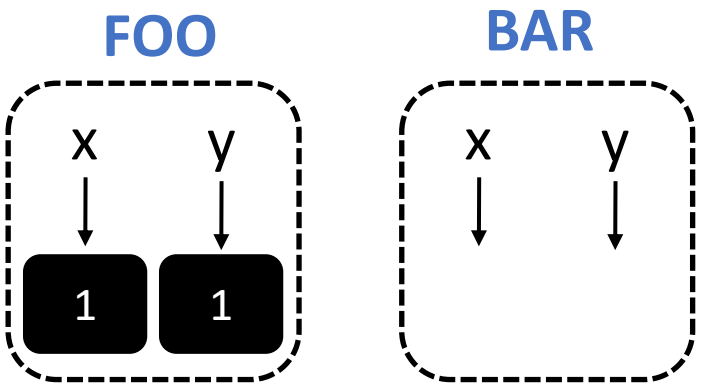
Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?



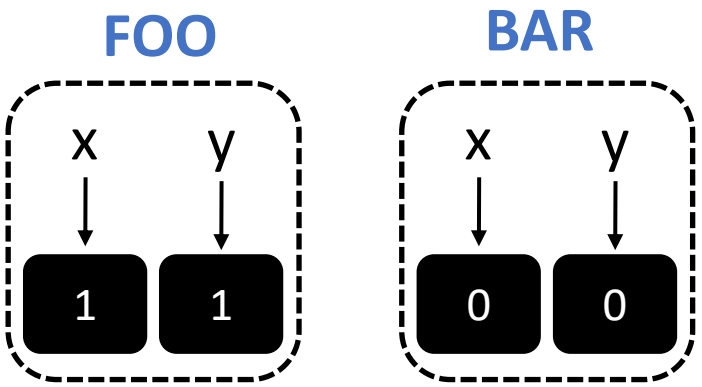
Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?



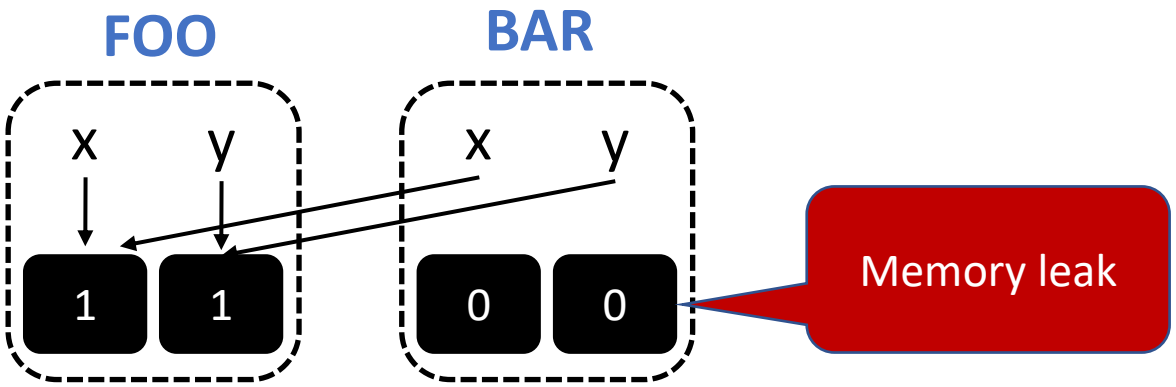
Assignment (=) Operator

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

};
```

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Will it give me any error?



Solution = Assignment Operator Overloading

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

friend myclass& operator= (myclass&
param1, myclass& param2);

};
```

```
myclass& operator= (myclass& param1, myclass&
param2)
{
    *(param1.x) = *(param2.x);
    *(param1.y) = *(param2.y);
    return param1;
}


int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Solution = Assignment Operator Overloading

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

friend myclass& operator= (myclass&
param1, myclass& param2);

};
```

```
myclass& operator= (myclass& param1, myclass&
param2)
{
    *(param1.x) = *(param2.x);
    *(param1.y) = *(param2.y);
    return param1;
}


int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

If your object has a pointer to memory that was dynamically allocated previously, e.g., in the constructor, you will need an overloaded assignment operator

Solution = Assignment Operator Overloading

- **Consider this example:**

Why taking reference parameters?

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

friend myclass& operator+ (myclass&
param1, myclass& param2);

};
```

```
myclass& operator+ (myclass& param1, myclass&
param2)
{
    *(param1.x) = *(param2.x);
    *(param1.y) = *(param2.y);
    return param1;
}


int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Solution = Assignment Operator Overloading

- **Consider this example:**

Why taking reference parameters?

Avoid **Copy Constructor**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

friend myclass& operator+ (myclass&
param1, myclass& param2);

};
```

```
myclass& operator+ (myclass& param1, myclass&
param2)
{
    *(param1.x) = *(param2.x);
    *(param1.y) = *(param2.y);
    return param1;
}


int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Solution = Assignment Operator Overloading

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

friend myclass& operator+ (myclass&
param1, myclass& param2);

};
```

```
myclass& operator+ (myclass& param1, myclass&
param2)
{
    *(param1.x) = *(param2.x);
    *(param1.y) = *(param2.y);
    return param1;
}
```

Why returning **param1**?

```
int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Solution = Assignment Operator Overloading

- **Consider this example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int a, int b)
    {
        x= new int
        x = a;
        y= new int;
        y = b;
    }

friend myclass& operator+ (myclass&
param1, myclass& param2);

};
```

```
myclass& operator+ (myclass& param1, myclass&
param2)
{
    *(param1.x) = *(param2.x);
    *(param1.y) = *(param2.y);
    return param1;
}

int main() {
    myclass foo(1, 1);
    myclass bar(0, 0);
    bar = foo;
    return 0;
}
```

Why returning **param1**?

baz=bar=foo; **???**

Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size2];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    // Memberwise copy ///
    return p1;
}
int main()
{
    myclass foo(10, 10);
    myclass bar(5, 5);
    bar = foo;
    return 0;
}
```

Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

Self Assignment

Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

Self Assignment

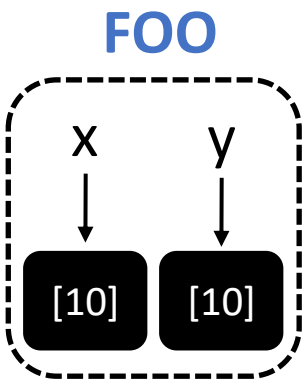
Any Problem???

Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

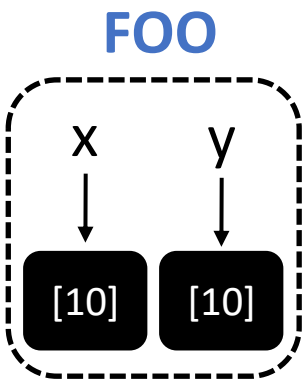


Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

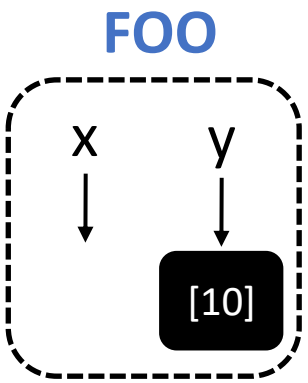


Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

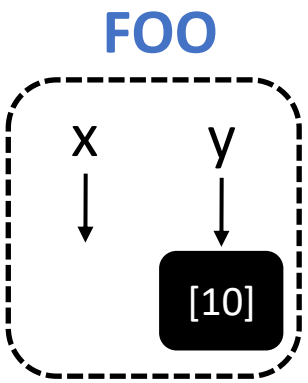


Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

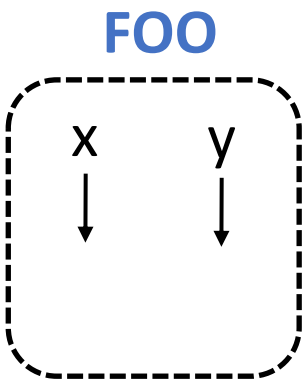


Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

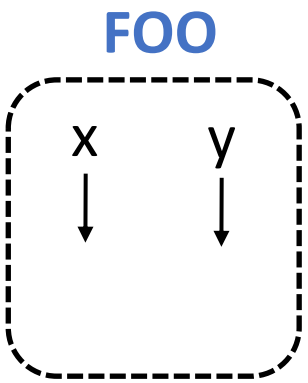


Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

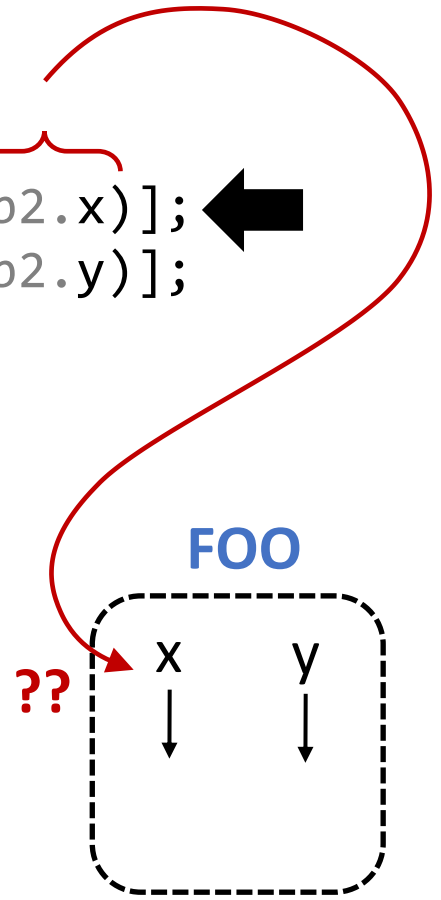


Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    delete [] p1.x;
    delete [] p1.y;
    p1.x= new int [sizeof(p2.x)];
    p1.y= new int [sizeof(p2.y)];

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```



Solution = Assignment Operator Overloading

- **Another complex example:**

```
class myclass {
    int *x, *y;
public:
    myclass(int size1, int size2)
    {
        x= new int [size1]
        y= new int [size1];
    }

friend myclass operator+ (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+ (myclass& p1, myclass& p2)
{
    if(&p1 =! &p2)
    {
        delete [] p1.x;
        delete [] p1.y;
        p1.x= new int [sizeof(p2.x)];
        p1.y= new int [sizeof(p2.y)];
    }
    return p1;
}
int main()
{
    myclass foo(10, 10);
    foo = foo;
    return 0;
}
```

Overloading "+=" operator

- **Example:**

```
class myclass {
    int x, y;
public:
    myclass(int a, int b)
    {
        x= a;
        y= b;
    }

friend myclass operator+= (myclass& p1,
myclass& p2);

};
```

```
myclass& operator+= (myclass& p1, myclass&
p2)
{
    p1.x= p1.x + p2.x;
    p1.y= p1.y + p2.y;

    return p1;
}

int main()
{
    myclass foo(10, 10);
    foo += foo;
    return 0;
}
```

Other binary operators

- **The operators**

  `-=, /=, *=, |=, %=, &=, ^=, <<=, >>=, !=`

  can be overloaded in a very similar fashion

Thanks a lot