



Fundamentals of Programming: Operators

Abdul Haseeb

```
for object to mirror  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.name))  
mirror_ob.select = 0  
bpy.context.selected_objects  
data.objects[one.name].select  
print("please select the object")  
  
-- OPERATOR CLASSES --  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"
```

Agenda

- Operators
- Arithmetic Operators
- Incremental or Decremental Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Shift Operator
- Assignment Operator
- Ternary Operator

Introduction

- ▶ Operators perform operations on operands
- ▶ Operands are variables on which operation is performed

```
int c = a + b;
```

Here, '+' is the addition operator. 'a' and 'b' are the operands that are being 'added'.

Introduction

4

Operators in C++

Unary operator

Binary operator

Ternary operator

Operator	Type
+, -, ++, --	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, , !	Logical operator
&, , <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator





Arithmetic Operators

- ▶ Perform arithmetic or mathematical operations on the operands
- ▶ Arithmetic operators can be classified into two categories:
 - ▶ Unary {Increment operator(++), Decrement Operator (--)}
 - ▶ Binary(+, -, *, /, %)

String Concatenation

String concatenation is the act of combining two strings together. This is done with the + operator.

```
string a = "This is an ";  
string b = "example string";  
string c = a + b;  
cout << c << endl;
```

challenge

What happens if you:

- Concatenate two strings without an extra space (e.g. remove the space after an in `string a = "This is an";`)?
- Use the += operator instead of the + operator (e.g. `a+=b` instead of `a + b`)?
- Add 3 to a string (e.g. `string c = a + b + 3;`)?
- Add "3" to a string (e.g. `string c = a + b + "3";`)?

String Concatenation