# CONVERSION OPERATORS

Changing a value's data type

Delroy A. Brinkerhoff

# FUNDAMENTAL TYPE CONVERSIONS

- **Type promotions**
- Typecasting
- Functions
  - C-strings to numbers
  - Numbers to C-strings
  - string objects to numbers
  - Numbers to string objects

- `10.0 + 5`
- `double d = 2;`
- `void function(double d);`
  - `function(5);`
- `double average(…) { …; return 10; }`

# FUNDAMENTAL TYPE CONVERSIONS

- Type promotions
- **Typecasting**
- Functions
  - C-strings to numbers
  - Numbers to C-strings
  - string objects to numbers
  - Numbers to string objects

- `(a + b) * c`

- `(double)2 / 3`
- `double(2) / 3`

- `Shape S;`
- `Circle C : public Shape;`
- `Shape S2 = (Shape)C;`

# FUNDAMENTAL TYPE CONVERSIONS

- Type promotions
- Typecasting
- **Functions**
  - C-strings to numbers
  - Numbers to C-strings
  - string objects to numbers
  - Numbers to string objects

- `atoi("123)`
- `itoa(123)`

- `string s("123")`
- `stoi(s)`
- `to_string(123)`

# CONVERSION CONSTRUCTORS

```
Time::Time(int s)
{
    hours = s / 3600;
    s %= 3600;
    minutes = s / 60;
    seconds = s % 60;
}
```

```
fraction(int n = 0, int d = 1)
: numerator(n), denominator(d) {}
```

- `fraction f1;`
- `fraction f2(5);`
- `fraction f3(2, 3);`

# CONVERSION OPERATORS

- `operator int() { return hours * 3600 + minutes * 60 + seconds; }`

- `Time T(….);`
- `(int)T`
- `int(T)`

# CONVERSION OPERATORS

- `operator double() { return (double)numerator / denominator; }`

- `fraction F(…);`
- `double d = (double)F;`
- `double d = double(F);`

# INCOMPATIBLE CONVERSIONS

- Typically, a class may not have a conversion constructor & conversion operator
- `Time T(…);`
  - `T + 30`
- `Fraction F(…);`
  - `F + 5`