C++ Introduction

Operators

Contents

List of some Operators

```
1  R& K::operator =(S b);                /* basic assignment */
2  R operator +(K a, S b);               /* addition */
3  R operator -(K a, S b);               /* subtraction */
4  R operator +(K a);                    /* unary plus (integer
       promotion) */
5  R operator -(K a);                    /* unary minus (additive
       inverse) */
6  R operator *(K a, S b);               /* multiplication */
7  R operator /(K a, S b);               /* division */
8  R operator %(K a, S b);               /* modulo (integer remainder
       ) */
9  R& operator ++(K& a);                 /* prefix increment */
10 R operator ++(K& a, int);             /* postfix increment */
11 R& operator --(K& a);                 /* prefix decrement */
12 R operator --(K& a, int);             /* postfix decrement */
13 //...
```

Operators demystified

Mathematical Definitions:

- Unary Operator
  $operator! : bool \rightarrow bool : !(t1) \mapsto \neg t1$
- Binary Operator
  $operator+ : T1 \times T1 \rightarrow T1 : +(t1, t2) \mapsto t1 + t2$

Example

```
1  #include <iostream>
2
3  struct complex {
4    double real;
5    double imaginary;
6    complex(double real, double imaginary) {
7      this->real = real;
8      this->imaginary = imaginary;
9    }
10 };
11
12 complex operator +(complex const & a, complex const & b) {
13   return complex(a.real + b.real, a.imaginary + b.imaginary);
14 }
15
16 int main() {
17   complex c1(1, 3), c2(2, 4);
18   complex c3 = c1 + c2;
19   std::cout << c3.real << ", " << c3.imaginary << std::endl;
20   return 0;
21 }
```

Operators as interface

- C++ exposes operators as interfaces to invoke common operations on objects
- $\rightarrow$ compare objects, add objects, ...

Example

```
#include <iostream>

struct complex {
  double real;
  double imaginary;

  complex(double real, double imaginary) {
    this->real = real;
    this->imaginary = imaginary;
  }
};

bool operator==(complex const & lhs, complex const & rhs) {
  return (lhs.real == rhs.real && lhs.imaginary == lhs.imaginary);
}

int main() {
  complex c1(1, 3), c2(1, 3);
  if (c1 == c2)
    std::cout << "true";
  else
    std::cout << "false";
  return 0;
}
```