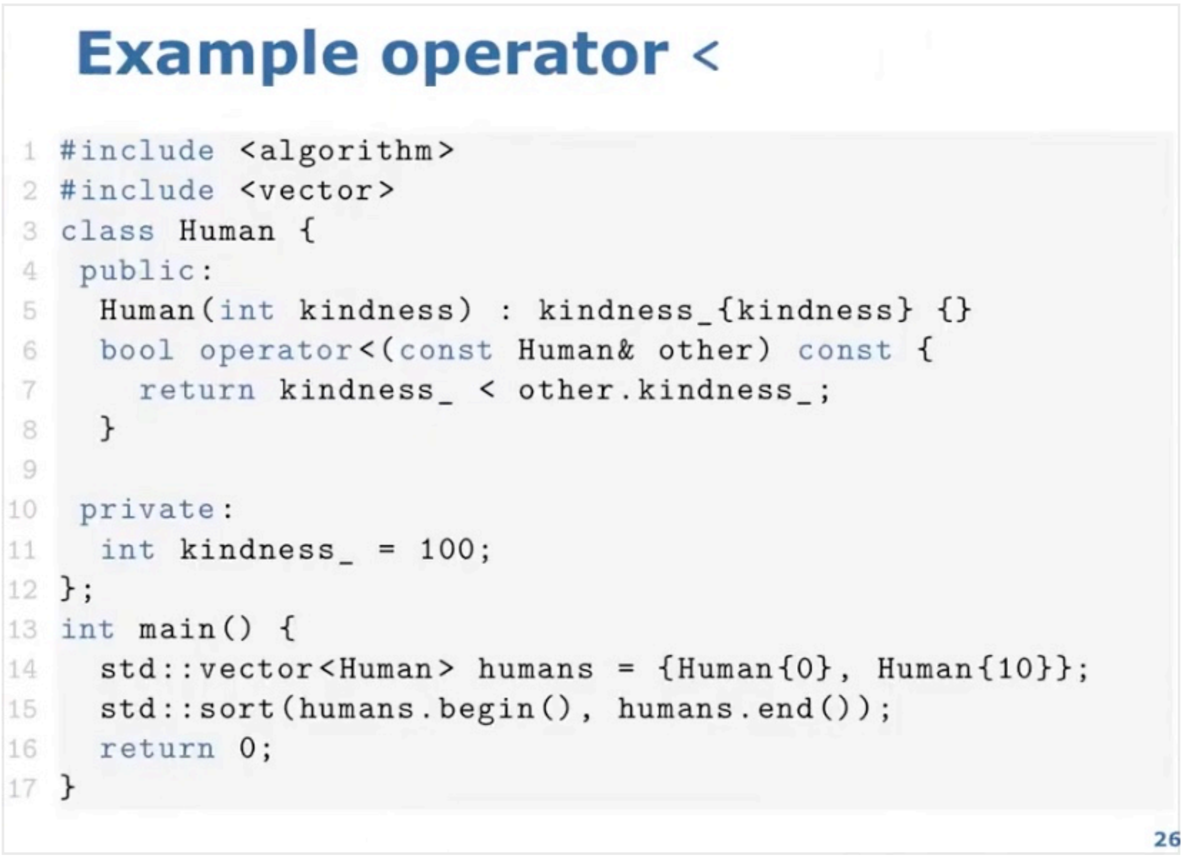
Operator Overloading

Custom operators for a class

- Operators are functions with a signature:
  <RETURN_TYPE> operator<NAME>(<PARAMS>)
- <NAME> represents target operation, ex: >, <, =, ==, <<, etc.
- Have all attributes of a function.
- Always contain word operator in name.

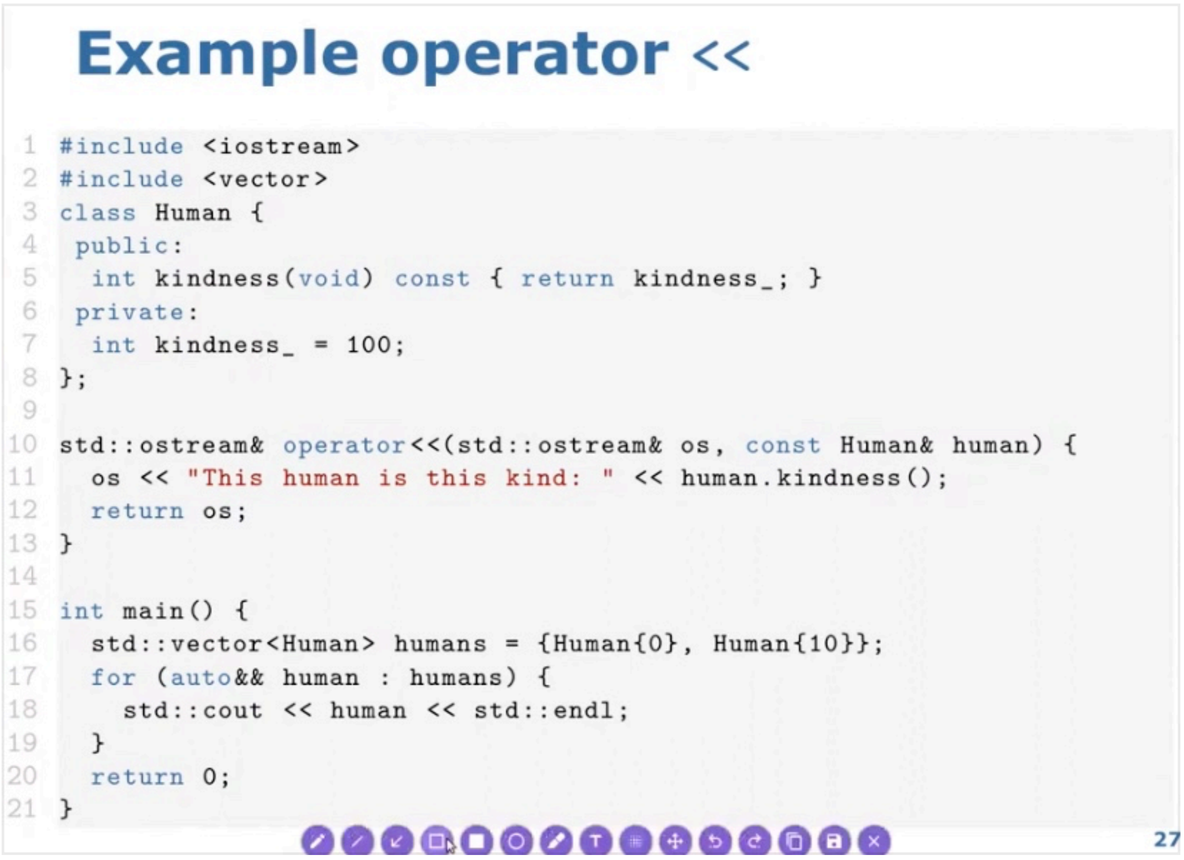


Example operator <

```cpp
#include <algorithm>
#include <vector>
class Human {
 public:
  Human(int kindness) : kindness_{kindness} {}
  bool operator<(const Human& other) const {
    return kindness_ < other.kindness_;
  }

 private:
  int kindness_ = 100;
};
int main() {
  std::vector<Human> humans = {Human{0}, Human{10}};
  std::sort(humans.begin(), humans.end());
  return 0;
}
```

26



Example operator <<

```cpp
#include <iostream>
#include <vector>
class Human {
 public:
  int kindness(void) const { return kindness_; }
 private:
  int kindness_ = 100;
};

std::ostream& operator<<(std::ostream& os, const Human& human) {
  os << "This human is this kind: " << human.kindness();
  return os;
}

int main() {
  std::vector<Human> humans = {Human{0}, Human{10}};
  for (auto&& human : humans) {
    std::cout << human << std::endl;
  }
  return 0;
}
```