Операторы сравнения C++20

TL;DR Операторы сравнения C++20

	Equality	Ordering
Primary	==	<=>
Secondary	!=	<, >, <=, >=

- Первичные операторы могут быть **обращены.**
- Вторичные могут быть **переписаны** через первичные.

<=>

Официальное название: оператор трехстороннего сравнения
(The three-way comparison operator)

Неофициально: spaceship operator.

Идея трехстороннего сравнения

Единственная операция f(a, b), которая определяет отношение порядка между a и b.

Пример:

```
strcmp(a, b) < 0   // a < b
strcmp(a, b) == 0  // a == b
strcmp(a, b) > 0   // a > b
```

Three-way comparison operator

Формула: `a @ b -> (a <=> b) @ 0`

Three-way comparison operator

Формула: `a @ b -> (a <=> b) @ 0`

```
bool a = (3 <=> 4) < 0;  // 3 < 4, true
bool b = (4 <=> 4) == 0; // 4 == 4, true
bool c = (4 <=> 3) > 0;  // 4 > 3, true
```


https://godbolt.org/z/TcrE8nfK8

operator<=>

```
class Widget {
 public:
  /* comp_cat */ operator<=>(const Widget& rhs) const;

  // One of:
  std::strong_ordering operator<=>(const Widget& rhs) const;
  std::weak_ordering operator<=>(const Widget& rhs) const;
  std::partial_ordering operator<=>(const Widget& rhs) const;
};
```

std::strong_ordering

- Похоже на сравнение int’ов
- Равенство подразумевает взаимозаменяемость:
  Если a == b, то f(a) == f(b)

Допустимые значения:

- std::strong_ordering::less
- std::strong_ordering::equal (+ std::strong_ordering::equivalent)
- std::strong_ordering::greater

std::weak_ordering

- Похоже на регистронезависимое сравнение строк
- Равенство определяет класс эквивалентности:
  “hello” == “HELLO”
- Строки эквивалентны, но не равны, поэтому
  f(“hello”) ?? f(“HELLO”)

Допустимые значения:

- std::weak_ordering::less
- std::weak_ordering::equivalent
- std::weak_ordering::greater

std::partial_ordering

- Похоже на сравнение floating point
- std::weak_ordering::{less, equivalent, greater}
- std::weak_ordering::unordered

std::partial_ordering

```
const auto qnan =
std::numeric_limits<double>::quiet_NaN();

const bool a = qnan == qnan;  // false
const bool b = qnan < qnan;   // false
const bool c = qnan > qnan;   // false
```


https://godbolt.org/z/j4vGqWPY3

std::partial_ordering::unordered

```
// true
(qnan <=> qnan) == std::partial_ordering::unordered;
```


https://godbolt.org/z/j4vGqWPY3

Пример

```
struct Rect {
  std::weak_ordering operator<=>(Rect rhs) const {
    return area() <=> rhs.area();
  }

  int area() const { return width_ * height_; }

  int width_ = 0;
  int height_ = 0;
};
```


https://godbolt.org/z/1rTs95c34

Переписывание и обращение

Primary: ==, <=>

Secondary: !=, <, >, <=, >=

Primary могут быть обращены

Secondary могут быть переписаны

Пример: обращение ==

```
struct Integer {
  Int(int v) : v_(v) {}

  bool operator==(Int rhs) const {
    return v_ == rhs.v_;
  }

  int v_ = 0;
};
```

Метод, не внешняя функция!

Пример: обращение ==

```
Integer a = 1;

// C++17
bool a_eq_b = a == 1;  // Ok: a.operator==(b)
bool b_eq_a = 1 == a;  // Error: 1.operator==(a)
```

Пример: обращение ==

```
Integer a = 1;

// C++20
bool a_eq_b = a == 1;  // Ok: a.operator==(b)
bool b_eq_a = 1 == a;  // Error: 1.operator==(a)
                       // Rewrite: a.operator==(1)
```

Пример: переписывание !=

```
Integer a = 1;

// C++17
bool a_neq_b = a != 1;  // Error
bool b_neq_a = 1 != a;  // Error
                        // no match for 'operator!='
```

Пример: переписывание !=

```
Integer a = 1;

// C++20
bool a_neq_b = a != 1;  // Ok: a != 1 -> !(a == 1)

bool b_neq_a = 1 != a;  // Ok: 1 != a -> !(1 == a)
                                      -> !(a == 1)
```

Операторы для String: было

```
bool operator< (const String& lhs, const String& rhs);
bool operator<=(const String& lhs, const String& rhs);
bool operator> (const String& lhs, const String& rhs);
bool operator>=(const String& lhs, const String& rhs);
bool operator==(const String& lhs, const String& rhs);
bool operator!=(const String& lhs, const String& rhs);

bool operator< (const char* lhs, const String& rhs);
bool operator<=(const char* lhs, const String& rhs);
bool operator> (const char* lhs, const String& rhs);
bool operator>=(const char* lhs, const String& rhs);
bool operator==(const char* lhs, const String& rhs);
bool operator!=(const char* lhs, const String& rhs);

bool operator< (const String& lhs, const char* rhs);
bool operator<=(const String& lhs, const char* rhs);
bool operator> (const String& lhs, const char* rhs);
bool operator>=(const String& lhs, const char* rhs);
bool operator==(const String& lhs, const char* rhs);
bool operator!=(const String& lhs, const char* rhs);
```

Операторы для String: стало

```
class String {
 public:
  bool operator==(const String& b) const;
  bool operator==(const char* b) const;

  std::strong_ordering operator<=>(const String& b) const;

  std::strong_ordering operator<=>(const char* b) const;
};
```

Когда можно сгенерировать

```
#include <compare>

struct Int {
  Int(int v) : v_(v) {}
  int v_;
  auto operator<=>(const Int&) const = default;
};

int main() {
  Int(0) < Int(1);
}
```


https://cppinsights.io/s/17090824

Рекомендация

Каждый дип должен (одно из):

C++17:

- ==, !=
- ==, !=, <, >, <=, >=
- Не перегружать ни один из этих операторов

C++20:

- ==
- <=> = default
- <=>, ==
- Не перегружать ни один из этих операторов

Материалы

- https://brevzin.github.io/c++/2019/07/28/comparisons-cpp20/
- C++ Russia 2018: Herb Sutter, New in C++20: The spaceship operator

Q&A