ECE 150 Comparison operators

Douglas Harder

December 2023

1 Comparison operators for numeric types

Integers and floating-point numbers can be compared using any of six binary operators: `<`, `<=`, `==`, `!=`, `>=` and `>`. It is important to write `<=` and not `=<`, so always write it the way you say it: $x \leq y$ is read x is *less than* or *equal to* y, or `<=`. You never say that x is equal to or less than y.

Given two integers or two finite floating-point numbers (not $\pm\infty$) `a` and `b`, exactly one and only one of these three comparisons returns true:

1. `a < b`,
2. `a == b`, or
3. `a > b`.

The comparison `a == b` returns true if and only if `(a - b) == 0`. Thus, we also have the following relationships:

1. The result of `!(a < b)` must be the same as `a >= b`.
2. The result of `!(a >= b)` must be the same as `a < b`.
3. The result of `!(a == b)` must be the same as `a != b`.
4. The result of `!(a != b)` must be the same as `a == b`.
5. The result of `!(a > b)` must be the same as `a <= b`.
6. The result of `!(a <= b)` must be the same as `a > b`.

Also, `(a <= b) && (a >= b)` is true if and only if `a == b`.

2 Comparison operators for addresses

You can compare two addresses (pointers), as well, so you can see if two addresses are the same, or if they are different. For example, is a pointer equal to the null pointer, or is that pointer not equal to the null pointer? As another example, is the address of an argument passed by reference equal to `this`?

```
Linked_list &operator=( Linked_list const &rhs ) {
    if ( this == &rhs ) {
        // We are assigning this list to itself,
        //     so do nothing...
        return *this;
    }

    // Other code to assign the right-hand side
    // linked list entries to this linked list...

    return *this;
}
```

The only place where it makes sense to use the other comparison operators on addresses is if you know those addresses all come from the same array:

- The address of one entry is less than or equal to the address of another if an only if the first entry appears in the array either in the same location or before the second entry.
- The address of one entry is greater than the address of another if and only if the first entry appears after the second in the array.

Not on the examination

Please remember, comparing addresses of nodes in a linked list says nothing, because where the operating system assigns its memory has nothing to do with what is being stored in those nodes. Thus, the address of the first node allocated may be very large (for example, `0x54b80`), while the next may be small (for example, `0x1ca8`) and the third may be in between (for example, `0x31d40`).

3 Using the result of comparison operations

While comparison operations often appear in the condition of conditional statements, for-loops and while loops, you must remember that all of these operations simply result in a Boolean value of either `true` or `false`. Consequently, the result can be assigned to a local variable of type `bool`, or can also be returned from a function that has a return type of `bool`. For example,

```
bool is_above{ (x*x - 4.0)*x > x*x - 5.7 };
```

or

```
bool Linked_list::empty() const {
    return p_list_head_ == nullptr;
}
```