

【C++】 Day14(2)

▼ Class	C++
📅 Date	@December 2, 2021
🔗 Material	
# Series Number	
☰ Summary	Jump Statement

【Ch5】 Jump Statement

5.5 Jump Statement

Jump statements interrupt the flow of execution. C++ offers four jumps: `break`, `continue`, `goto`, and `return`.

5.5.1 The break Statement

A `break` statement terminates the nearest enclosing while, do while, for, or switch statement. Execution resumes at the statement immediately following the terminated statement.

A break can appear only within an iteration statement or switch statement(including inside statements or blocks nested inside such loops). A break affects only the nearest enclosing loop or switch:

```
string buf;
while (cin >> buf && !buf.empty()) {
    switch(buf[0]) {
        case '-':
            //process up to the first blank
            for(auto it = buf.begin() + 1; it != buf.end(); ++it) {
                if (*it == ' ')
                    break; // #1, leaves the for loop
            }
            //break #1 transfers control here.
```

```

    //remaining '-' processing
    break; // #2, leaves the switch statement
case '+':
    //...
}
//end of switch break
}

```

The break **labeled #1** terminates the for loop that follows the hyphen case label. It **does not terminate the enclosing switch statement and in fact does not even terminate the processing for the current case.**

5.5.2 The continue Statement

A **continue statement** terminates the current iteration of the nearest enclosing loop and immediately begins the next iteration.

A continue can **appear only inside a for, while, or do while loop**, including inside statements or blocks nested inside such loops.

A continue **interrupts the current iteration**; execution stays inside the loop.

5.5.3 The goto Statement

A **goto statement** provides **an unconditional jump from the goto to another statement** in the same function.

Best Practices: Programs should not use goto. goto make programs hard to understand and hard to modify.

The syntactic form of a goto statement is

```
goto label;
```

where **label** is **an identifier that identifies a statement.**

A goto **cannot transfer control from a point where an initialized variable is out of scope to a point where that variable is in scope:**

```
goto end;  
int ix = 10; //error: goto bypasses an initialized variable definition  
end:  
//error: code here could use ix but the goto bypassed its declaration  
ix = 42;
```