```cpp
int main() {
    printf("hello, world");
return 0;
}
```

C++20coroutine_/test.cpp

```cpp
1
2    #include <coroutine>
3    #include <iostream>
4
5    struct promise {
6        struct promise_type;
7
8        struct iterator {
9            std::coroutine_handle<promise_type>& h;
10           int& operator*() {
11               return h.promise().n;
12           }
13           iterator operator++() {
14               if (!h.done())h.resume();
15               return *this;
16           }
17           bool operator!=(const iterator&)const {
18               return !h.done();
19           }
20       };
21       struct promise_type {
22           int n;
23           promise get_return_object() {
24               return { std::coroutine_handle<promise_type>::from_promise(*this) };
25           }
26           std::suspend_never initial_suspend() noexcept { return {}; }//注意返回类型 需要确保协
27           std::suspend_always final_suspend() noexcept { return {}; }
```

```cpp
        std::suspend_always yield_value(int r) { n = r; return {}; }//co_yield()需要
        void return_void() { }
        void unhandled_exception() {}
    };
    iterator begin() { return { _h }; }
    iterator end() { return { _h }; }
    std::coroutine_handle<promise_type>_h;
};

promise iota(int value) {
    std::cout << "iota\n";
    while (value) {
        co_yield value;
        --value;
    }
}

int main() {
    for (int x : iota(10)) {//范围for会执行operator!= * ++ 改变协程状态写到++中, *普通的返回值
        std::cout << x << '\n';
    }
}
```

行内代码import numpy as np