cpp Documentation

Release v1.0.0

jxm

Jan 23, 2021

CONTENTS:

CHAPTER

ONE

通用工具

1.1 Pair 和 TUple

1.1.1 Pair

1.1.2 TUple

1.2 Smart Pointer

1.2.1 Class shared_ptr

- std::shared_ptr

托管 Pointer 的最后一个 shared_ptr 象在 reference 被删除是时, 会自动执行 delete 释放指针。

```
#include <iostream>
#include <memory>

using namespace std;
class A
{
public:
    int i;
    A(int n):i(n) { };
    ~A() { cout << i << " " << "destructed" << endl; }
};
int main()
{
    shared_ptr<A> sp1(new A(2)); /* A(2) 由 sp1 托管,  */
    shared_ptr<A> sp2(sp1);      /* A(2) 同时交由 sp2 托管 */
    shared_ptr<A> sp3;
```

(continues on next page)

(continued from previous page)

```
    sp3 = sp2;                    /* A(2) 同时交由 sp3 托管 */
    cout << "use_count : " << sp3.use_count() << endl;
    cout << sp1->i << "," << sp2->i <<"," << sp3->i << endl;

    A * p = sp3.get();           /* get 返回托管的指针, p 指向 A(2) */
    cout << p->i << endl;        /* 输出 2 */

    sp1.reset(new A(3));         /* reset 导致托管新的指针, 此时 sp1 托管 A(3) */
    cout << "sp1.reset(A(3)), sp3.use_count : " << sp3.use_count() << endl;

    sp2.reset(new A(4));         /* sp2 托管 A(4) */
    cout << sp1->i << endl;      /* 输出 3 */
    cout << "sp2.reset(A(4)), sp3.use_count : " << sp3.use_count() << endl;

    sp3.reset(new A(5));        /* sp3 托管 A(5),A(2) 无人托管, 被 delete*/
    cout << "sp3.reset(A(5)), sp3.use_count : " << sp3.use_count() << endl;


    sp3.reset();                /* 复位,A(5) 无人托管, 被 delete*/
    cout << "sp3.reset(); sp3.use_count : " << sp3.use_count() << endl;

    cout << "end" << endl;
    return 0;
}
```

```
use_count : 3
2,2,2
2
sp1.reset(A(3)), sp3.use_count : 2
3
sp2.reset(A(4)), sp3.use_count : 1
2 destructed
sp3.reset(A(5)), sp3.use_count : 1
5 destructed
sp3.reset(); sp3.use_count : 0
end
4 destructed
3 destructed
```

1.2.2 Class unique_ptr

- std::unique_ptr

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search