|  |  |
| --- | --- |
| **Ex12-05a.cpp** | |
| **Line#** | **Code** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | /\* ---- Notes ----------  -shared\_ptr is a smart pointer which can share the ownership of object.  -Several shared\_ptr can point to the same object.  -It keep a reference count to maintain how many shared\_ptr are pointing to the same object.  -Few ways the shared object will be deleted:  1) The last shared\_ptr go out of scope  2) Initialize shared\_ptr with some other shared\_ptr  3) Reset shared\_ptr  -Refrence count doesn't work when use reference or pointer to shared\_ptr  \*/  #include <iostream>  #include <memory>  using namespace std;  #include "Student.h"  int main() {  shared\_ptr<Student> pStudent1(new Student("Ali", 21, 3.14F));  cout << "Name:" << pStudent1->Name << " Counter:" << pStudent1.use\_count() << endl;  shared\_ptr<Student> pStudent2 = pStudent1;  cout << "Name:" << pStudent2->Name << " Counter:" << pStudent2.use\_count() << endl;  pStudent1.reset();  cout << "Name:" << pStudent2->Name << " Counter:" << pStudent2.use\_count() << endl;  //pStudent2.reset();  cout << "main() is ending..." << endl;  return 0;  } |

|  |  |
| --- | --- |
|  | |
| **Ex12-05b.cpp** | |
| **Line#** | **Code** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #include <iostream>  #include <memory>  using namespace std;  #include "Student.h"  int main() {  shared\_ptr<Student> pStudent1(new Student("Ali", 21, 3.14F));  shared\_ptr<Student> pStudent2(new Student("Abu", 21, 3.14F));  pStudent1 = pStudent2;  cout << "main() is ending..." << endl;  return 0;  } |

|  |  |
| --- | --- |
|  | |
| **Ex12-05c.cpp** | |
| **Line#** | **Code** |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #include <iostream>  #include <memory>  using namespace std;  #include "Student.h"  int main() {  shared\_ptr<Student> pStudent1(new Student("Ali", 21, 3.14F));  cout << "Name:" << pStudent1->Name << " Counter:" << pStudent1.use\_count() << endl;  shared\_ptr<Student>& pStudent2 = pStudent1;  cout << "Name:" << pStudent2->Name << " Counter:" << pStudent1.use\_count() << endl;  shared\_ptr<Student>\* pStudent3 = &pStudent1;  cout << "Name:" << (\*pStudent3)->Name << " Counter:" << pStudent1.use\_count() << endl;  cout << "main() is ending..." << endl;  return 0;  } |