Experiment 1：Source code

|  |
| --- |
| //cpu.h  #ifndef cpu\_h  #define cpu\_h  #include<iostream>  using namespace std;  enum CPU\_Rank  {P1 = 1, P2, P3, P4, P5, P6, P7};  class CPU {  public:  CPU();  ~CPU();  CPU(int a, float b, int c);  int run();  int stop();  void show();  private:  int frequency;  float voltage;  CPU\_Rank rank;  };  #endif  //cpu.cpp  #include"cpu.h"  CPU::CPU() {  frequency = 37;  voltage = 3.0;  rank = P1;  cout << "CPU has been established." << endl;  };  CPU::~CPU(){  cout << "CPU has been destoryed." << endl; };  int CPU::run() {  cout << "CPU is running." << endl;  return 2;  };  int CPU::stop() {  cout << "CPU has stopped." << endl;  return 2;  };  CPU::CPU(int a, float b, int c) {  frequency = a;  voltage = b;  rank = static\_cast<CPU\_Rank>(c);  }  void CPU::show() {  cout << "Frequency:" << frequency << endl;  cout << "Voltage:" << voltage << endl;  cout << "Rank" << rank << endl;  }  int main()  {CPU c;  c.run();  c.stop();  c.show();  return 0;  }; |

Experiment result:

|  |
| --- |
|  |

Experiment 2: Source code

|  |
| --- |
| //cpu.h  #ifndef cpu\_h  #define cpu\_h  #include<iostream>  using namespace std;  enum CPU\_Rank  {P1 = 1, P2, P3, P4, P5, P6, P7};  class CPU {  public:  CPU();  ~CPU();  CPU(int a, float b, int c);  int run();  int stop();  void show();  private:  int frequency;  float voltage;  CPU\_Rank rank;  };  #endif  //cpu.cpp  #include"cpu.h"  CPU::CPU() {  frequency = 37;  voltage = 3.0;  rank = P1;  cout << "CPU has been established." << endl;  };  CPU::~CPU(){  cout << "CPU has been destoryed." << endl; };  int CPU::run() {  cout << "CPU is running." << endl;  return 2;  };  int CPU::stop() {  cout << "CPU has stopped." << endl;  return 2;  };  CPU::CPU(int a, float b, int c) {  frequency = a;  voltage = b;  rank = static\_cast<CPU\_Rank>(c);  }  void CPU::show() {  cout << "Frequency:" << frequency << endl;  cout << "Voltage:" << voltage << endl;  cout << "Rank" << rank << endl;  }  //RAM.h  #include<iostream>  #ifndef RAM\_H  #define RAM\_H  enum RAM\_speed  {X1 = 1, X2, X4, X8, X16};  class RAM {  public:  RAM();  RAM(int a, int b);  void show();  private:  RAM\_speed speed;  int size;  };  #endif  //RAM.cpp  #include"RAM.h"  #include<iostream>  using namespace std;  RAM::RAM()  {speed = static\_cast<RAM\_speed>(1);  size = 16;  }  RAM::RAM(int a, int b)  {speed = static\_cast<RAM\_speed>(a);  size = b;  }  void RAM::show()  {cout << "RAM's speed" << " " << speed << endl;  cout << "RAM's size" << " " << size << endl;  };  //CDROM.h  #ifndef CDROM\_H  #define CDROM\_H  class CDROM  {public:  CDROM();  CDROM(int i);  void show();  private:  int var;  };  #endif  //CDROM.cpp  #include<iostream>  #include"CDROM.h"  using namespace std;  CDROM::CDROM() {  var = 0;  }  CDROM::CDROM(int i) {  var = i;  }  void CDROM::show() {  cout << var << endl;  }  //Computer.h  #ifndef COMPUTER\_H  #define COMPUTER\_H  #include"cpu.h"  #include"CDROM.h"  #include"RAM.h"  class computer {  public:  computer();  computer(int a, float b, int c, int d, int e, int f);  void show();  void run();  void shutdown();  private:  CPU cp;  RAM ram;  CDROM cdrom;  };  #endif  //computer.cpp  #include<iostream>  #include"Computer.h"  computer::computer() {  cp = CPU();  ram = RAM();  cdrom = CDROM();  }  computer::computer  (int a, float b, int c, int d, int e, int f) :  cp(a, b, c), ram(d, e), cdrom(f) {};  void computer::show() {  cp.show();  ram.show();  cdrom.show();  }  void computer::run()  {cp.run();  cout << "The computer is running." << endl;  }  void computer::shutdown() {  cp.stop();  cout << "The computer has shut down." << endl;  cout << endl;  }  //main.cpp  #include"Computer.h"  #include<iostream>  using namespace std;  int main()  {  computer c1;  computer c2(6, 3.0, 2, 3, 16, 1);//复制构造函数的调用？  computer c3(c2);//是否需要引用？  c1.run();  c1.show();  c1.shutdown();  c2.run();  c2.show();  c2.shutdown();  c3.run();  c3.show();  c3.shutdown();  system("pause");  return 0;  } |

Experiment result:

|  |
| --- |
|  |

Experiment 3\_1: result

|  |
| --- |
|  |

Experiment 3\_2: source code

|  |
| --- |
| //Client.h  #include<iostream>  #ifndef Client\_h  #define Client\_h  class client  {public :  client(int i) ;  client operator + (client &c);  client(client & c);  client operator++(int);  void show();  private:  static int count;  int i;  };  #endif  //Client.cpp  #include "Client.h"  using namespace std;  int client::count = 0;  void client::show() {  cout<<i<<ends; }  client client::operator++(int){  i++;  return \*this;}  client client::operator +(client &c){  count += c.i;  return client (i+c.i);};  client::client(int i):i(i) {  count += i; };  client::client(client &c) :i(i) {  count += i;  cout << "CopyConsturctor" << endl; };  //函数实现的文件包含  #include"Client.h"  using namespace std;  int main ()  {client c1(8);  client c2(21);  client c3=c1+c2;  c1++;  client\* p[3] = { &c1,&c2, &c3 };  cout << "The answer should be" << endl;  cout<<9 << " " << 21 << " " << 29 << endl;  for (int i=0; i < 3; i++)  {  p[i]->show();  }  system("pause");  return 0;  } |

Experiment result:

|  |
| --- |
|  |