实验3.2：研究c++对象模型

1. 代码及相应注释如下：

#include<iostream>

using namespace std;

class T

{

    private:

            static int m\_mem1;//静态数据成员用来统计已经生成的对象数目

            int m\_mem2=0;

            char\*m\_mem3="the object constructed by the default constructor";//指向char的指针用来储存对象名。

    public:

    //定义无参构造函数

          T()

          {

            m\_mem1++;

            cout<<"construct"<<" "<<m\_mem3<<endl;

          }

     //定义有参构造函数

          T(int mem2,char\*mem3)

          {

            m\_mem1++;

            m\_mem2=mem2;

            m\_mem3=mem3;

            cout<<"construct"<<" "<<m\_mem3<<endl;

          }

      //定义拷贝构造函数

          T(const T&a)

          {

            m\_mem1++;

            m\_mem2=a.m\_mem2;

            m\_mem3=a.m\_mem3;

            cout<<"copy"<<" "<<m\_mem3<<endl;

          }

      //定义析构函数

          ~T()

          {

            cout<<"destruct"<<" "<<m\_mem3<<endl;

          }

      //定义静态成员函数

          static int  Getmem1()

          {

            return m\_mem1;

          }

      //输出对象的this指针、数据成员的值与地址

          void Show()

          { cout<<"the infomation of object "<<m\_mem3<<" are as follows:"<<endl;

            cout<<"this:"<<" "<<this<<endl;

            cout<<"static m\_mem1:"<<" "<<m\_mem1<<endl;

            cout<<"m\_mem2:"<<" "<<m\_mem2<<endl;

            cout<<"m\_mem3:"<<" "<<m\_mem3<<endl;

            cout<<"address of static m\_mem1:"<<"  "<<&m\_mem1<<endl;

            cout<<"address of m\_mem2:"<<" "<<&m\_mem2<<endl;

            cout<<"address of m\_mem3:"<<" "<<&m\_mem3<<endl;

          }

};

int T::m\_mem1=0;//初始化静态数据成员

T t1;

T t2(2,"t2");//定义两个全局对象

//声明一个函数，在函数内声明一个临时的对象，由传入的对象拷贝构造而成。

void Func(const T &t)

{

    cout<<"Copy an object that the constructor constructs outside of the main function"<<endl;

    T a(t);

}

int main()

{

T t3;

T t4(4,"t4");//分别使用无参构造函数与有参构造函数构造两个对象

T\*t5=new T;

T\*t6=new T(6,"t6");//动态生成两个对象

Func(t4);//调用类外的函数生成一个对象

cout<<endl;

//输出6个对象的this指针、所有数据成员的值与地址

cout<<"there are 6 'T' objects as follows:"<<endl;

cout<<"============================================"<<endl;

cout<<"t1:"<<endl;

t1.Show();

cout<<"================"<<endl;

cout<<"t2:"<<endl;

t2.Show();

cout<<"================"<<endl;

cout<<"t3:"<<endl;

t3.Show();

cout<<"================"<<endl;

cout<<"t4:"<<endl;

t4.Show();

cout<<"================"<<endl;

cout<<"t5:"<<endl;

t5->Show();

cout<<"================"<<endl;

cout<<"t6:"<<endl;

t6->Show();

cout<<"================"<<endl;

    // 输出各种函数的地址

int (\*mainPtr)() = &main;

cout << "Address of main() function: " << reinterpret\_cast<void\*>(mainPtr) << endl;

void (\*funcPtr)(const T&) =&Func;

cout << "Address of Func(const T &t) function: " << reinterpret\_cast<void\*>(funcPtr) << endl;

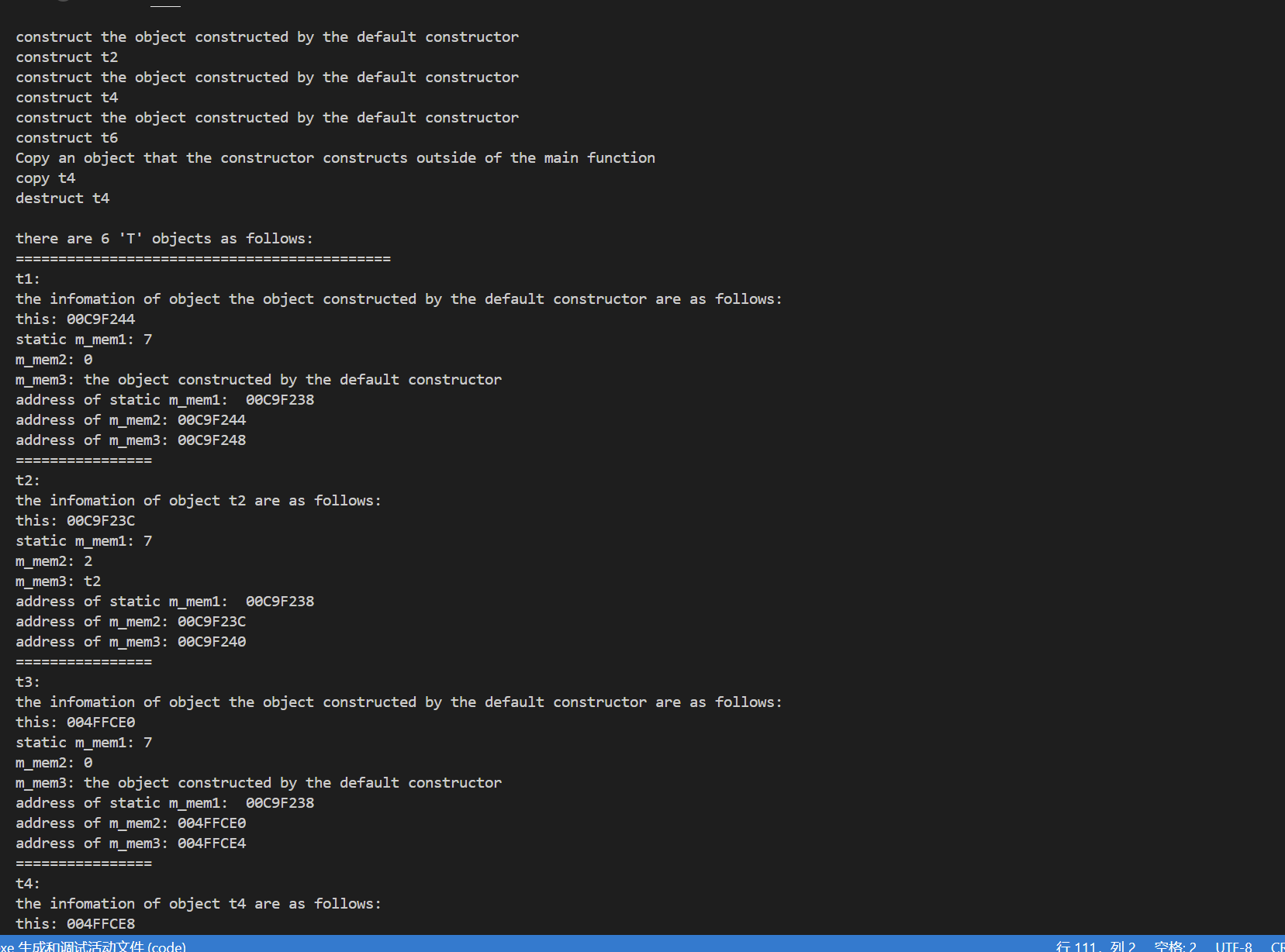
int (\*GetPtr)() = &T::Getmem1;

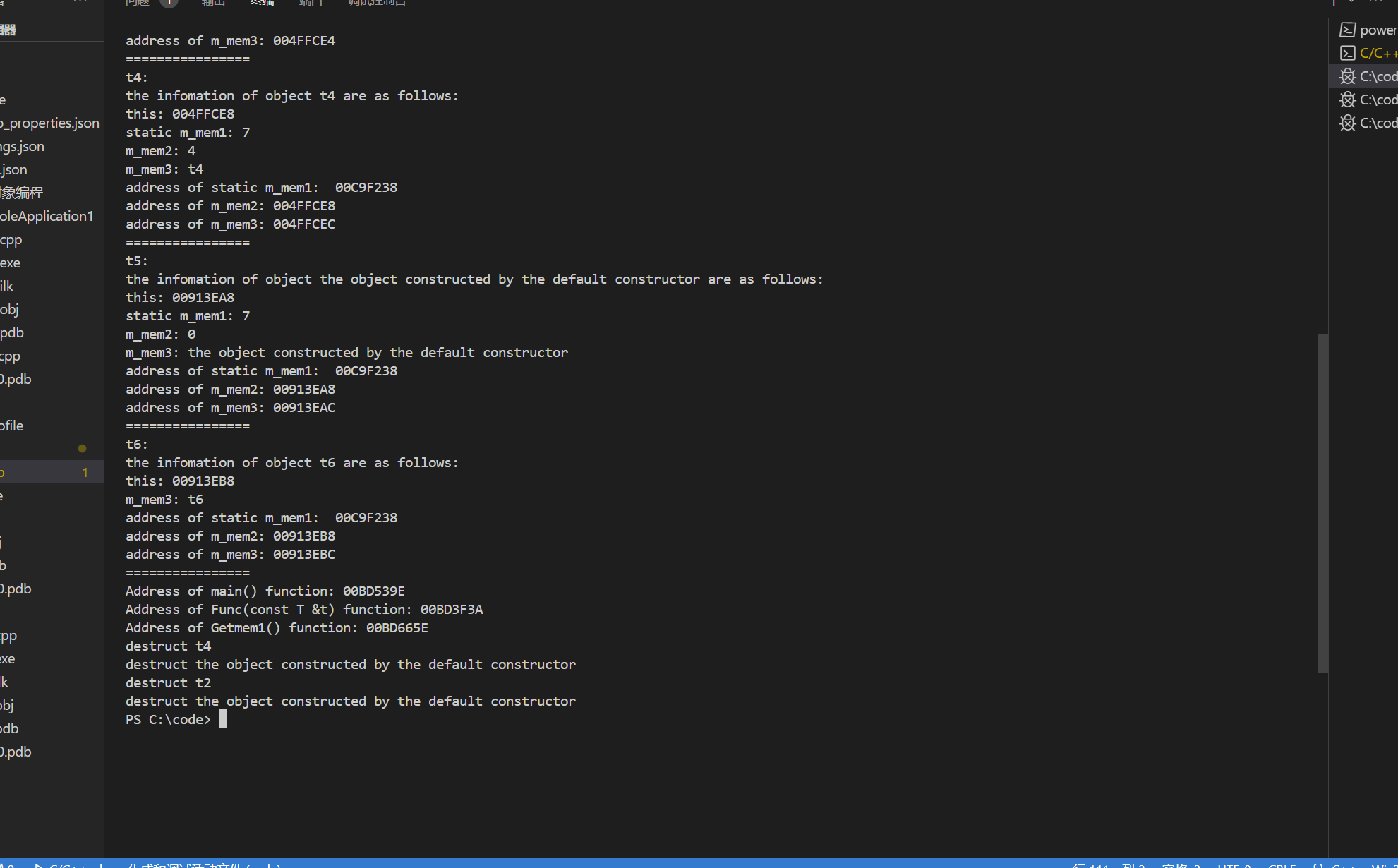
cout << "Address of Getmem1() function: " << reinterpret\_cast<void\*>(GetPtr) << endl;

return 0;

}

（2）代码运行结果如下：





（3）结果分析：

代码创建了六个T类对象，分别是在全局、main函数内、外部函数Func()内创建无参构造函数与有参构造函数。每个对象都有一个int型成员、一个静态数据成员与一个用来指向保存对象名的char指针，对象的Show函数用来输出每个对象哥哥数据成员的值与地址还有对象的this指针指向。

1. 各种对象地址：由代码运行结果来看同一区域创建的对象（如t1与t2，t3与t4，t5

与t6），其内存是连续的，不同区域创建的对象内存不连续（如t1与t3）。且最先生成的对象最后析构，是FILO的栈的结构。可见静态数据成员是本类对象共享一份拷贝等问题。同时，可以得出类对象是在内存上分配一个连续的存储对象的非静态数据成员的空间·。

1. 数据成员的值与地址：非静态数据成员的地址在一起（连续），且都在所在的对象的地址之后，而静态数据成员的地址固定不变。同时int型数据成员起始地址总是4的倍速，float 数据成员起始地址总是8的倍数，指向char的指针起始地址也是4的倍数，对齐现象存在。
2. 各种函数地址：由代码运行结果可得main()函数、Func()函数、静态成员函数的地址是随运行不断变化的，但三种函数的相对位置并未发生变化。