程序代码：

实验1

# include<iostream>

using namespace std;

class Tr {

public:

Tr(int n)

{

i = n;

}

void set\_i(int n)

{

i = n;

}

int get\_i()

{

return i;

}

private:

int i;

};

void sqr\_it(Tr ob) //

{

ob.set\_i(ob.get\_i() \* ob.get\_i());

cout << "在函数sqr\_it内，形参对象ob的数据成员i的值为:" << ob.get\_i();

cout << endl;

}

int main()

{

Tr obj(10);

cout << "调用函数sqr\_it前, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

sqr\_it(obj);

cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

return 0;

}

当传递对象方式为使用对象作为函数参数时，函数表示为

void sqr\_it(Tr ob) //

{

ob.set\_i(ob.get\_i() \* ob.get\_i());

cout << "在函数sqr\_it内，形参对象ob的数据成员i的值为:" << ob.get\_i();

cout << endl;

}

当传递对象方式为使用对象指针作为函数参数时，函数表示为

void sqr\_it(Tr \*ob) //

{

Ob->set\_i(ob->get\_i() \* ob->get\_i());

cout << "在函数sqr\_it内，形参对象ob的数据成员i的值为:" << ob->get\_i();

cout << endl;

}

同时主函数更改为

int main()

{

Tr obj(10);

cout << "调用函数sqr\_it前, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

sqr\_it(&obj);

cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

return 0;

}

当传递对象方式为使用对象引用作为函数参数时，函数表示为

void sqr\_it(Tr ob) //

{

ob.set\_i(ob.get\_i() \* ob.get\_i());

cout << "在函数sqr\_it内，形参对象ob的数据成员i的值为:" << ob.get\_i();

cout << endl;

}

主函数更改回原主函数，即

int main()

{

Tr obj(10);

cout << "调用函数sqr\_it前, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

sqr\_it(obj);

cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

return 0;

}

实验2：

#include "iostream.h"  
#include <string.h>  
class TStudent  
{  
private:  
    char m\_Name[6];  
    static float m\_ClassMoney;  
public:  
    void InitStudent(char \*);  
    void ExpendMoney(float);   
    static void ShowMoney();   
};  
float TStudent::m\_ClassMoney=1000;    
void TStudent::InitStudent(char name[])  
{  
    strcpy(m\_Name,name);  
}  
void TStudent::ExpendMoney(float money)  
{  
    m\_ClassMoney-=money; //

  cout<<m\_Name<<"花费班费"<<m\_ClassMoney<<endl;  
}   
void TStudent::ShowMoney()   
{   
    cout<<"班费还剩余"<< m\_ClassMoney<<endl;   
}

void main()  
{  
    TStudent stu[3];//定义三个学生  
    stu[0].InitStudent("A");  
    stu[1].InitStudent("B");  
    stu[2].InitStudent("C");  
    stu[0].ExpendMoney(50);  
    TStudent::ShowMoney();  
     stu[1].ExpendMoney(98.5);  
    TStudent::ShowMoney();  
    stu[2].ExpendMoney(500.53);  
    TStudent::ShowMoney();  
}

程序结果：

A花费班费：950

班费还剩余：950

B花费班费：851.5

班费还剩余：851.5

C花费还剩余：350.97

班费还剩余：350.97

心得体会：

初步了解三种不同的对象传递方式，即对象 指针 引用，并通过初次使用c++中的静态成员，体会到其可以实现同一类的不同对象之间的数据共享。

在编写代码的过程中，关键的逻辑在于明确哪一变量是要被作为静态变量进行使用的，哪一个变量要同时作用于多个对象，同时被多个变量影响。