实验内容一

一、程序代码

# 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) // 对象ob作为函数sqr\_it的形参

{

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);

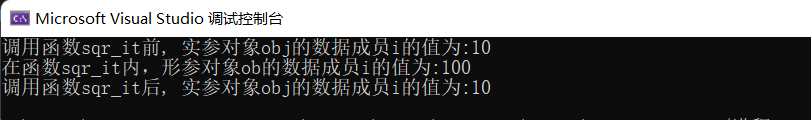
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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

return 0;

}

二、程序结果



一、程序代码

# 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) // 对象指针ob作为函数sqr\_it的形参

{

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);

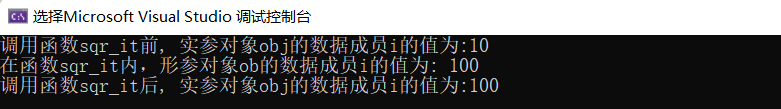
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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

return 0;

}

二、程序结果



一、程序代码

# 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);

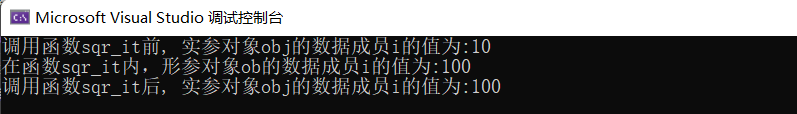
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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

return 0;

}

二、程序结果



三、感想心得

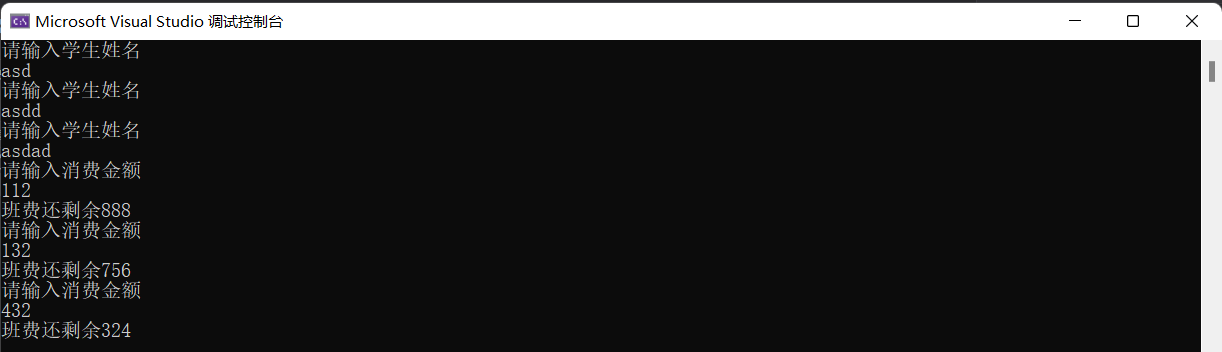
使用函数传递值的时候，需要用指针或数组等通过地址传递参数。无法使用实值传递。

实验内容二

一、程序代码

#include<iostream>  
using namespace std;  
  
class TStudent  
{  
private:  
static float m\_ClassMoney;  
char name[100];  
float money;  
public:  
  
void InitStudent()  
{  
cout << "输入学生姓名：";  
cin >> name;  
cout << "输入学生花费：";  
cin >> money;  
}  
  
void ExpendMoney()  
{  
m\_ClassMoney = m\_ClassMoney - money;  
}  
  
void ShowMoney()  
{  
cout << "剩余的钱：" << m\_ClassMoney << endl;  
}  
  
};  
float TStudent::m\_ClassMoney = 1000;  
  
int main()  
{  
TStudent A, B, C;  
  
A.InitStudent();  
A.ExpendMoney();  
A.ShowMoney();  
B.InitStudent();  
B.ExpendMoney();  
B.ShowMoney();  
C.InitStudent();  
C.ExpendMoney();  
C.ShowMoney();  
  
return 0;

}  
二、程序结果



三、感想心得

使用静态成员变量能够减少很多麻烦，特别是减少了许多指针的使用。需要注意的是静态成员变量只输出程序结束后的最终值，不显示中间结果。

储存多个人姓名时要用二维数组，如果不使用数组的话会非常的麻烦

Copyright 2021-2099 XinyuGuan. All rights reserved