实验项目一

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);

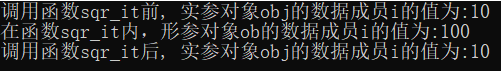
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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);

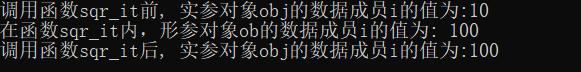
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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

函数间传递参数时使用指针或者数组能够实现对实参的修改。

实验项目二

一、程序代码

# include<iostream>

#include<string.h>

using namespace std;

class TStudent {

static float m\_ClassMoney;

char name1[3][100];

static int j;

public:

void InitStudent(char name[]);

void ExpendMoney(float money);

void ShowMoney();

};

float TStudent::m\_ClassMoney = 1000;

int TStudent::j = 0;

void TStudent::ExpendMoney(float money)

{

m\_ClassMoney = m\_ClassMoney - money;

}

void TStudent::ShowMoney()

{

cout << "班费还剩余" << m\_ClassMoney << endl;

}

int main(void)

{

TStudent A, B, C;

float money;

cout << "请输入消费金额" << endl;

cin >> money;

A.ExpendMoney(money);

A.ShowMoney();

cout << "请输入消费金额" << endl;

cin >> money;

B.ExpendMoney(money);

B.ShowMoney();

cout << "请输入消费金额" << endl;

cin >> money;

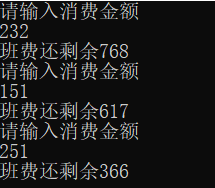
C.ExpendMoney(money);

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指针的使用，简化程序。

Copyright ©2021-2099 ShuoWang. All rights reserved