程序一：

#include "iostream.h"

class CTest

{

public:

CTest(int ival)

{

m\_ival=ival;

cout<<"Construct Called:"<<m\_ival<<endl;

}

~CTest()

{

cout<<"Destruct Called:"<<m\_ival<<endl;

}

protected:

int m\_ival;

};

void main()

{

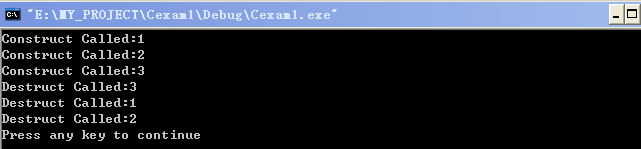
CTest t1(1);

static CTest t2(2);

CTest t3(3);

}

程序运行结果：



程序二：

#include "iostream.h"

char Increment(int ival);

void main()

{

for(int i=0;i<3;i++)

cout<<Increment(i)<<endl;

}

char Increment(int ival)

{

static char x='B';

x+=1;

return x+ival;

}

运行结果：



程序三：

#include "iostream.h"

class CBase

{ public:

CBase() //CONSTRUCT

{

X=0;Y=0;W=0;H=0;

}

CBase(int x,int y,int w,int h) //CONMSTRUCT

{

X=x;

Y=y;

W=w;

H=h;

}

virtual int Area()

{

return 0;

}

int X,Y,W,H;

};

class CChild1:public CBase

{

public:

CChild1 (int x,int y,int w,int h):CBase(x,y,w,h){};

virtual int Area()

{

return W\*H;

}

};

class CChild2:public CBase

{

public:

CChild2 (int x,int y,int w,int h):CBase(x,y,w,h){};

virtual int Area()

{

return (W-X)\*(W-Y)/2;

}

};

void DisplayVal(CBase op)

{

cout<<"Area()="<<op.Area()<<endl;

}

void DisplayRef(CBase &rf)

{

cout<<"Area()="<<rf.Area()<<endl;

}

void DisplayPointer(CBase \*ptr)

{

cout<<"Area()="<<ptr->Area()<<endl;

}

void main()

{

CBase \*ptr;

CChild1 ch1(5,5,10,10);

CChild2 ch2(10,10,30,30);

ptr=(CBase\*)&ch2;

DisplayVal(ch1);

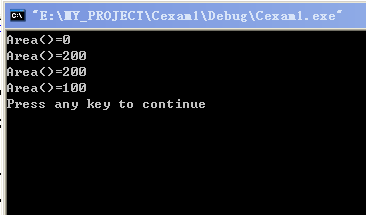
DisplayRef(ch2);

DisplayPointer(ptr);

DisplayPointer(&ch1);

}

程序运行结果：



程序四：

#include "iostream.h"

class CLOC

{

public:

CLOC()

{

X=0;

Y=0;

}

CLOC(int x,int y)

{

X=x;

Y=y;

}

~CLOC()

{

cout<<"Destructor("<<X<<","<<Y<<")"<<endl;

}

void Move(int x,int y)

{

X+=x;

Y+=y;

}

void printf()

{

cout<<"X="<<X<<",Y="<<Y<<endl;

}

private:

int X;

int Y;

};

void main()

{

CLOC \*pArr[2];

pArr[0]=new CLOC(1,1);

pArr[1]=new CLOC;

for (int i=0;i<2;i++)

pArr[i]->Move(i,2\*i);

for ( i=0;i<2;i++)

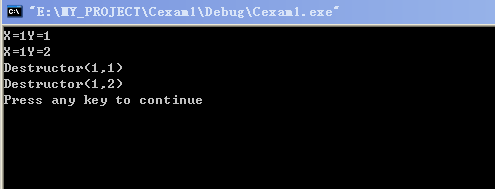
pArr[i]->printf();

for ( i=0;i<2;i++)

delete pArr[i];

}

运行结果：



程序五：

#include "iostream.h"

void main()

{

int m=1,n=2,k=3;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

cout<<"m="<<1<<"n="<<2<<"k="<<3<<endl;

{

int n=4;

m++;

k+=4;

{

int m=3,k=3;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

cout<<"m="<<3<<"n="<<4<<"k="<<3<<endl;

}

k+=5;

m-=4;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

cout<<"m="<<-2<<"n="<<4<<"k="<<12<<endl;

}

m+=2;

n++;

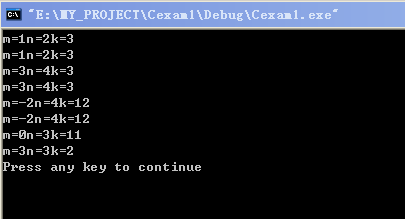
k--;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

cout<<"m="<<3<<"n="<<3<<"k="<<2<<endl;

}

运行结果：



分析：在括号内部的变量相当于不同的变量，占据不同的内存。在内部没有变量时，则还是原来的变量。

程序六：

#include "iostream.h"

class CDemo

{

public:

CDemo()

{

m\_nval=0;

}

CDemo(int nval)

{

m\_nval=nval;

cout<<"Constructor"<<m\_nval<<endl;

}

~CDemo()

{

cout<<"Destructor"<<m\_nval<<endl;

}

protected:

int m\_nval;

};

CDemo d0;

CDemo d1(1);

void main()

{

CDemo d2(2);

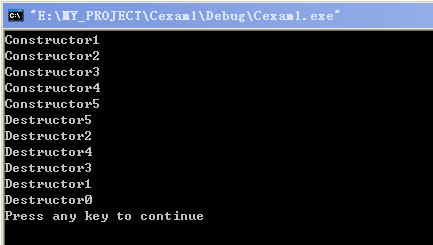
static CDemo d3(3);

static CDemo d4(4);

CDemo d5(5);

}

运行结果：



程序七：

#include "iostream.h"

class Cload

{

public:

Cload(){m\_nval=0;}

Cload(int nval){m\_nval=nval;}

void PrintVal()

{

cout<<"Member Value="<<m\_nval<<endl;

}

Cload &operator+(int i);

Cload &operator+(Cload &op);

Cload &operator-(int i);

Cload &operator-(Cload &op);

protected:

int m\_nval;

};

Cload&Cload::operator+(int i)

{

m\_nval+=i;

return \*this;

}

Cload&Cload::operator+(Cload &op)

{

op.m\_nval+=m\_nval;

return \*this;

}

Cload&Cload::operator-(int i)

{

m\_nval-=i;

return \*this;

}

Cload&Cload::operator-(Cload &op)

{

m\_nval-=op.m\_nval;

return \*this;

}

void main()

{

Cload L1(10),L2,L3(3);

L1.PrintVal();

L2=L1+L3;

L2.PrintVal();

L3=L3-7;

L3.PrintVal();

L2=L2+L1;

L2.PrintVal();

L1=L3+13;

L1.PrintVal();

L3=L2+L1;

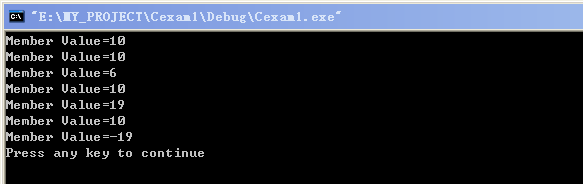
L3.PrintVal();

L3=L2-L1;

L3.PrintVal();

}

运行结果：



程序题：

1.写出下面的运行结果

#include "iostream.h"

class COperator

{

public:

COperator()

{

X=0;

Y=0;

}

COperator(int x,int y)

{

X=x;

Y=y;

}

void Printf()

{

cout<<"X="<<X<<",Y="<<Y<<endl;

}

COperator& operator+(COperator & op);

COperator& operator-(COperator & op);

COperator& operator\*(COperator & op);

COperator& operator+(int i);

private:

int X,Y;

};

COperator&COperator::operator +(COperator & op)

{

X+=op.X;

Y+=op.Y;

return \*this;

}

COperator&COperator::operator -(COperator & op)

{

X-=op.X;

Y-=op.Y;

return \*this;

}

COperator&COperator::operator \*(COperator & op)

{

X\*=op.X;

Y\*=op.Y;

return \*this;

}

COperator&COperator::operator+(int i)

{

X+=i;

return \*this;

}

void main()

{

COperator op1; //X1=0;Y1=0

COperator op2(2,3); //X2=2,Y2=3; X1=0,Y1=0

op1=op1+10; //X1=10,Y1=0 X2=2,Y2=3

op1.Printf(); //X=10,Y=0 X2=2,Y2=3

op1=op1+op2; //X1=12,Y1=3 X2=2,Y2=3

op1.Printf(); //X=12,Y=3 X2=2,Y2=3

op2=op1\*op2; //X1=24,Y1=9, X2=24,Y=9

op2.Printf(); //X=24,Y=9

op1.Printf(); //X=24,Y=9

op1=op2-op1; //X1=0,Y1=0 X2=0,Y2=0

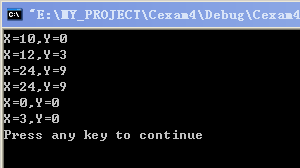
op1.Printf(); //X=0,Y=0

op2=op2+3; //X1=0,Y1=0 X2=3,Y2=0

op2.Printf(); //X2=3,Y2=0

}

运行结果：



2.写出下面的运行结果

#include "iostream.h"

#include "string.h"

class CQuiz

{

public:

CQuiz ();

CQuiz (const char \*szReason);

void ChangeName(void);

~CQuiz();

private:

char \*instName;

static int nCount;

int m\_nSerialNo;

};

int CQuiz::nCount=0;

void CQuiz::ChangeName(void)

{

int nLen;

char ch;

nLen=strlen(instName);

if(nLen%2==0)

{

for(int i=0;i<nLen/2;i++)

{

ch=instName[i];

instName[i]=instName[nLen-1-i];

instName[nLen-1-i]=ch;

}

cout<<instName<<endl;

}

}

CQuiz::CQuiz()

{

instName=new char[14];

nCount++;

m\_nSerialNo=nCount;

strcpy(instName,"Drefault");

cout<<instName<<"-NO"<<nCount<<endl;

}

CQuiz::CQuiz(const char \*szReason)

{

nCount++;

m\_nSerialNo=nCount;

if(szReason!=0&&strlen(szReason)>0)

{

instName=new char[strlen(szReason)+1];

strcpy(instName,szReason);

cout<<instName<<"-NO"<<nCount<<endl;

}

else

instName=0;

}

CQuiz::~CQuiz()

{

if(instName!=0)

{

cout<<"Release:"<<instName<<"-NO"<<m\_nSerialNo<<endl;

delete instName;

}

nCount--;

}

void main()

{

CQuiz One("One"); //One-NO1

One.ChangeName();

CQuiz Two("Twos"); //Twos-NO2

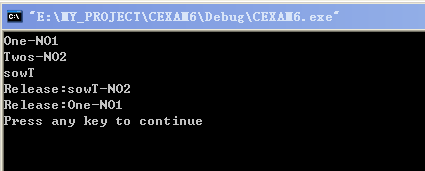
Two.ChangeName(); //sowT

//函数析构 //Release：sowT-NO2

//函数析构 //Release: One-NO1

}

运行结果：



程序一：

#include "iostream.h"

class CTest

{

public:

CTest()

{

X=0;

Y=0;

cout<<"Default construct called:"<<endl;

}

CTest(int x,int y)

{

X=x;

Y=y;

cout<<"Construct called:X="<<X<<",Y="<<Y<<endl;

}

~CTest()

{

cout<<"Destruct called:X="<<X<<",Y="<<Y<<endl;

}

private:

int X,Y;

};

void main()

{

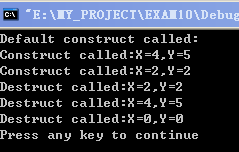
CTest t1;

CTest t2(4,5);

CTest t3(2,2);

}

运行结果：



程序二:

#include "iostream.h"

class CDemo

{

public:

CDemo()

{

m\_nVal=0;

}

CDemo(int nVal )

{

m\_nVal=nVal;

}

~CDemo()

{

cout<<"Destruct:"<<m\_nVal<<endl;

}

private:

int m\_nVal;

};

CDemo d0;

CDemo d1(1);

void main()

{

CDemo d2(2);

static CDemo d3(3);

{

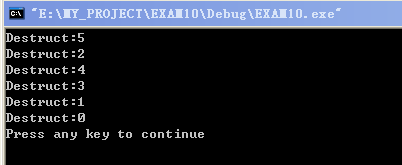
static CDemo d4(4);

CDemo d5(5);

}

}

运行结果：



程序三：

#include "iostream.h"

class CLoad

{

public:

CLoad()

{

m\_nVal=0;

}

CLoad(int nVal )

{

m\_nVal=nVal;

}

void PrintVal()

{

cout<<"Member Value="<<m\_nVal<<endl;

}

CLoad&operator+(int i);

CLoad&operator+(CLoad& op);

CLoad&operator-(int i);

CLoad&operator-(CLoad& op);

private:

int m\_nVal;

};

CLoad&CLoad::operator+(int i)

{

m\_nVal+=i;

return \*this;

}

CLoad&CLoad::operator+(CLoad& op)

{

op.m\_nVal+=m\_nVal;

return \*this;

}

CLoad&CLoad::operator-(int i)

{

m\_nVal-=i;

return \*this;

}

CLoad&CLoad::operator-(CLoad& op)

{

m\_nVal-=op.m\_nVal;

return \*this;

}

void main()

{

CLoad L1(10),L2,L3(3);

L1.PrintVal();

L2=L1+L3;

L2.PrintVal();

L3=L3-7;

L3.PrintVal();

L2=L2+L1;

L1.PrintVal();

L1=L3+13;

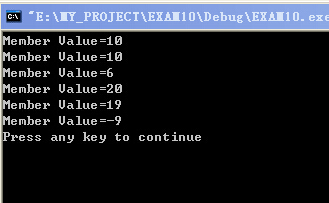
L1.PrintVal();

L3=L2-L1;

L2.PrintVal();

}

运行结果：



程序四：

#include "iostream.h"

class CBase

{ public:

CBase() //CONSTRUCT

{

X=0;Y=0;W=0;H=0;

}

CBase(int x,int y,int w,int h) //CONMSTRUCT

{

X=x;

Y=y;

W=w;

H=h;

}

virtual int Area()

{

return 0;

}

int X,Y,W,H;

};

class CChild1:public CBase

{

public:

CChild1 (int x,int y,int w,int h):CBase(x,y,w,h){};

virtual int Area()

{

return W\*H;

}

};

class CChild2:public CBase

{

public:

CChild2 (int x,int y,int w,int h):CBase(x,y,w,h){};

virtual int Area()

{

return (W-X)\*(W-Y)/2;

}

};

void DisplayVal(CBase op)

{

cout<<"Area()="<<op.Area()<<endl;

}

void DisplayRef(CBase &rf)

{

cout<<"Area()="<<rf.Area()<<endl;

}

void DisplayPointer(CBase \*ptr)

{

cout<<"Area()="<<ptr->Area()<<endl;

}

void main()

{

CChild1 ch1(5,5,10,10);

CChild2 ch2(10,10,30,30);

CBase \*ptr;

DisplayVal(ch1);

DisplayRef(ch2);

ptr=(CBase\*)&ch2;

DisplayPointer(ptr);

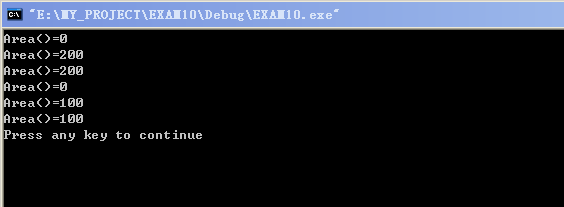
cout<<"Area()="<<((CBase)ch1).Area()<<endl;

DisplayPointer(&ch1);

cout<<"Area()="<<((CBase\*)&ch1)->Area()<<endl;

}

运行结果：



程序五：

#include "iostream.h"

void main()

{

int m=1,n=2,k=3;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

{

int n=4;

m++;

k+=4;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

{

k--;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

int m=3,k=3;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

}

k+=5;

m-=4;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

}

m+=2;

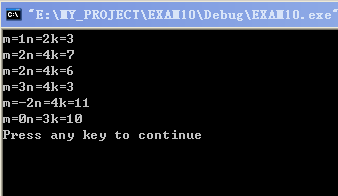
n++;

k--;

cout<<"m="<<m<<"n="<<n<<"k="<<k<<endl;

}

运行结果：



#include "iostream.h"

class Base

{

public:

virtual void fn()

{

cout<<"In base class"<<endl;

}

};

class SubClass:public Base

{

public:

virtual void fn()

{

cout<<"In SubClas"<<endl;

}

};

void Test(Base &b)

{

b.fn();

}

void main()

{

Base bc;

SubClass sc;

cout<<"calling test(bc):"<<endl;

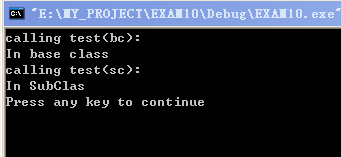
Test(bc);

cout<<"calling test(sc):"<<endl;

Test(sc);

}

运行结果：



当成员函数为虚拟时，C++会根据实参对象是基类还是子类对象来决定调用的是基函数还是派生类的函数。

编程说明题

编制一个C++的类，表示一个仓库的管理记录，每定义一个该类的对象，代表一批货物入，每析构一个该类的对象，代表一批货物出。

要求：

1. 有代表单批货物重量和当前仓库货物总重量的数据成员
2. 析构函数
3. 构造函数
4. 输出当前货物总重量和某批货物重量的成员函数
5. 编写一个main（）函数，使用该类定义的对象并调用各个数据成员

程序：

#include "iostream.h"

class GOODS

{

public:

static int AllWeight;

GOODS(int weight);

~GOODS();

ShowM\_Weight();

ShowAllWeight();

private:

int m\_Weight;

};

int GOODS::AllWeight=0;

GOODS::GOODS(int weight)

{

cout<<"有一批货物进入"<<endl;

m\_Weight=weight;

AllWeight+=m\_Weight;

}

GOODS::~GOODS()

{

cout<<"有一批货物输出"<<endl;

AllWeight-=m\_Weight;

cout<<"The Weight of all goods is :"<<AllWeight<<endl;

}

GOODS::ShowM\_Weight()

{

cout<<"The Weight of input is :"<<m\_Weight<<endl;

}

GOODS::ShowAllWeight()

{

cout<<"The Weight of all goods is :"<<AllWeight<<endl;

}

void main()

{

GOODS gd1(100);

gd1.ShowM\_Weight();

gd1.ShowAllWeight();

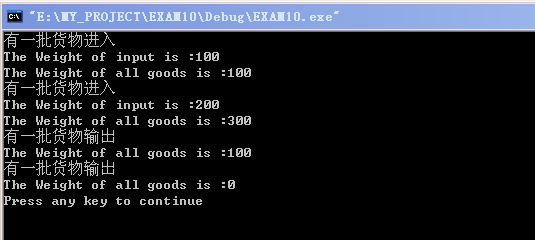
GOODS gd2(200);

gd2.ShowM\_Weight();

gd2.ShowAllWeight();

}

运行结果：



编程说明题

1. 编制一个C++的类，表示一个学生记录，每定义一个该类的对象，代表一个新生入学，每析构一个函数，地表一个学生毕业

要求：（1）有代表学生姓名，年龄，学分（浮点型），以及当前注册学生总数的数据

（2）构造函数

（3） 析构函数

（4）输出学生姓名的函数，输出学生年龄的函数，输出某学生学分的函数和学生的总数函数

（5）编写一个main（）函数，使用该类定义对象并调用各个成员函数

程序：

#include "iostream.h"

#include "string.h"

class STUDENT

{

public:

static int TotalNumber;

STUDENT(char \*name ,int age,float score);

~STUDENT();

ShowName();

ShowAge();

ShowScore();

ShowTotalNumber();

private:

int m\_age;

float m\_score;

char m\_name[30];

};

int STUDENT::TotalNumber=0;

STUDENT::STUDENT(char \*name ,int age,float score)

{ strcpy(m\_name,name);

m\_age=age;

m\_score=score;

cout<<"有一个新生入学"<<endl;

TotalNumber++;

}

STUDENT::~STUDENT()

{

cout<<"有一个学生毕业"<<endl;

TotalNumber--;

cout<<"The total of all students arre :"<<TotalNumber<<endl;

}

STUDENT::ShowName()

{

cout<<"Name:"<<m\_name<<endl;

}

STUDENT:: ShowAge()

{

cout<<"age:"<<m\_age<<endl;

}

STUDENT::ShowScore()

{

cout<<"score:"<<m\_score<<endl;

}

STUDENT :: ShowTotalNumber()

{

cout<<"The total of all students are :"<<TotalNumber<<endl;

}

void main()

{

STUDENT ST1("ZHANG WEI",26,99);

ST1.ShowAge();

ST1.ShowName();

ST1.ShowScore();

ST1.ShowTotalNumber();

STUDENT ST2("QI YOU CHENG ",26,99);

ST2.ShowAge();

ST2.ShowName();

ST2.ShowScore();

ST2.ShowTotalNumber();

}

运行结果：

