1、写出下面程序的运行结果。

#include<iostream>

using namespace std;

class toy

{

public:

toy(int q, int p)

{quan= q;

price=p;}

int get\_quan()

{ return quan;}

int get\_price ()

{ return price;}

private:

int quan, price;

};

int main()

{

toy op[3][2]={toy(10,20),toy(30,48),

toy(50,68),toy(70,80),

toy(90,16),toy(11,120)};

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

{ cout<<op[i][0].get\_quan()<<",";

cout<<op[i][0] .get\_price()<< "\n";

cout<<op[i][1].get\_quan()<<",";

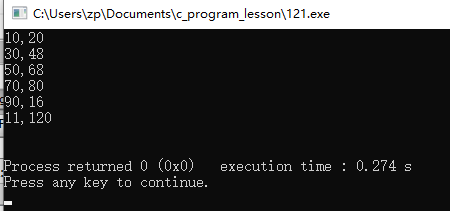
cout<<op[i][1].get\_price()<< "\n";}

cout<< endl ;

return 0;

}

【参考答案】本程序的运行结果如下:



**2、**写出下面程序的运行结果。

class example

{ public:

example(int n)

{ i=n;

cout<< "Constructing\n ";

}

~ example()

{ cout << "Destructing\n";

}

int get\_i()

{ return i;}

private:

int i;

};

int sqr\_it(example o)

{ return o.get\_i()\*o.get\_i();}

int main()

{

example x(10);

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

cout << sqr\_it(x) <<endl;

return 0;

}

【参考答案】本程序的运行结果如下：

Constructing

10

Destructing

100

Destructing

说明：一般认为，本程序执行时构造函数只被调用了一次，但是，析构函数却被调用了两次。构造函数的调用发生在生成对象X的时候；析构函数的调用，第1次出现在执行函数sqr\_it结束时，第2次出现在整个程序运行结束时。

在调用函数sqr\_it,进行实参和形参结合时，调用了拷贝构造函数(由于没有显式定义拷贝构造函数，这时调用的是系统自动生成的默认拷贝构造函数)，对形参对象o进行初始化。当函数sqr\_it执行终止，对象o被撤销的时候，第2次调用了析构函数。

3、写出下面程序的运行结果。

#include<iostream>

using namespace std;

class aClass

{

public:

aClass()

{total++;}

~aClass()

{total--;}

int gettotal()

{return total;}

private:

static int total;

};

int aClass::total=0;

int main()

{

aClass o1,o2,o3;

cout<<o1.gettotal()<<"objects in existence\n";

aClass\*p;

p=new aClass;

if(!p)

{cout<<"Allocation error\n";

return 1;}

cout<<o1.gettotal();

cout<<"objects in existence after allocation\n";

delete p;

cout<<o1.gettotal();

cout<<"objects in existence after deletion\n";

return 0;

}

【参考答案】本程序的运行结果如下：

3 objects in existence

4 objects in existence after allocation

3 objects in existence after deletion

说明：这个程序使用静态数据成员追踪记载创建对象的个数。完成这一工作的方法就是每创建一个对象就调用构造函数一次。每调用构造函数一次，静态数据成员total就增加1。每撤销一个对象就调用析构函数一次。每调用析构函数一次，静态数据成员total就减少1。

**4、**写出下面程序的运行结果。

#include<iostream>

using namespace std;

class test

{

public:

test();

~test() { };

private:

int i;

};

test::test()

{

i=25;

for(int ctr=0; ctr< 10; ctr++)

{ cout<< "Counting at "<<ctr<< "\n";}

}

test anObject;

int main ()

{ return 0;

}

【参考答案】本程序的运行结果如下：

Counting at 0

Counting at 1

Counting at 2

Counting at 3

Counting at 4

Counting at 5

Counting at 6

Counting at 7

Counting at 8

Counting at 9

说明：在本程序中主函数main只包括了一个return语句，但竟然有内容输出！什么时候调用了构造函数？我们知道，构造函数在对象被定义时调用。那么对象anObject是何时 被调用的呢？在主函数main之前，语句“test anObject”处。

5、写出下面程序的运行结果。

# include< iostream>

using namespace std;

class A{

int a, b;

public:

A ()

{ a=0;

b=0;

cout<<"Default constructor called.\n";

}

A{int i, int j }

{ a=i;

b=j;

cout<< "Constructor: a= "<< a << ", b= "<< b<<end1;

}

)；

int main ()

{ A a[3];

A b [3] = {A (1,2), A (3, 4), A (5, 6) }；

return 0;

}

【参考答案】本程序的运行结果如下：

Default constructor called.

Default constructor called.

Default constructor called.

Constructor: a=1, b=2

Constructor: a=3, b=4

Constructor: a=5, b=6

6、写出下面程序的运行结果。

#include<iostream>

using namespace std;

class N {

private:

int A;

static int B;

public:

N(int a)

{A=a;

B+= a;}

static void f1 (N m);

};

void N::f1(N m)

{

cout<< "A="<<m. A<<endl;

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

}

int N::B=0;

int main()

{

N P(5),Q(9);

N::f1(P);

N::f1(Q);

return 0;

}

【参考答案】本程序的运行结果如下：

A=5

B=14

A=9

B=14

7、写出下面程序的运行结果。

#include<iostream>

using namespace std;

class M {

int x, y;

public:

M()

{ x=y=0;}

M(int i, int j)

{ x=i;y=j;}

void copy(M \* m);

void setxy(int i, int j)

{x= i;

y=j;}

void print()

{cout<< x<< ","<< y<< endl;}

};

void M::copy(M \*m)

{ x=m->x;y=m->y;}

void fun(M m1,M \*m2)

{ m1.setxy(12,15);

m2->setxy(22,25);

}

int main()

{

M p(5,7),q;

q.copy(&p);

fun(p,&q);

p.print();

q.print();

return 0;

}

【参考答案】本程序的运行结果如下：

5, 7

22, 25

8、构建一个类Stock,含字符数组stockcode[]及整型数据成员quan、双精度型数据成员price。构造函数含3个参数:字符数组na[]及q、p。当定义Stock的类对象时，将对象的第1个字符串参数赋给数据成员stockcode,第2和第3个参数分别赋给quan、price。未设置第2和第3个参数时,quan的值为1000,price的值为8. 98。成员函数print

没有形参,需使用this指针，显示对象数据成员的内容。假设类Stock第1个对象的3个参数分别为: "600001"、3000和5.67 ,第2个对象的第1个数据成员的值是" 600001",第2和第3个数据成员的值取默认值。要求编写程序分别显示这两个对象数据成员的值。

【参考答案】实现本题功能的程序如下:

#include<iostream>

using namespace std;

#include <string.h>

const int SIZE=80;

class Stock{

public:

Stock()

{strcpy(stockcode," ");}

Stock(char code[],int q=1000,double p=8.98)

{strcpy(stockcode, code);

quan=q;

price=p;}

void print(void)

{cout<< this-> stockcode;

cout<<" "<<this->quan<<" "<< this-> price<< endl;}

private:

char stockcode[SIZE];

int quan;

double price;

};

int main()

{

Stock st1("600001",3000,5.67);

st1.print();

Stock st2("600002");

st2.print();

return 0;

}

本程序的运行结果如下:

600001 3000 5.67

600002 1000 8. 98

**[程序设计题]**

9、编写一个程序,已有若干学生的数据,包括学号、姓名、成绩，要求输出这些学生的数据并计算出学生人数和平均成绩(要求将学生人数和总成绩用静态数据成员表示)。

【参考答案】实现本题功能的程序如下:

#include<iostream>

#include<string>

using namespace std;

class Student{

public:

Student(int n,string na, double d)

{ no=n;

deg=d;

name= na;

sum+=d;

num++ ;}

static double avg()

{ return sum/num;}

static int total()

{ return num;}

void disp()

{cout<<no<<" "<< name<<" "<< deg<< endl;}

private:

int no; //学号

string name; //姓名

double deg; //成绩

static double sum; //总成绩

static int num; //学生人数

};

double Student::sum= 0;

int Student::num=0;

int main()

{

Student s1(1001,"Zhou",97),s2(1002,"Zhan",65),s3(1003,"Chen",88);

cout<<"学号 姓名 成绩\n";

s1.disp();

s2.disp();

s3.disp();

cout<<"学生人数= "<<Student::total()<<endl;

cout<< "平均成绩= "<<Student::avg();

return 0;

}

本程序的运行结果如下:

学号 姓名 成绩

1001 Zhou 97

1002 Zhan 65

1003 Chen 88

学生人数=3

平均成绩= 83.3333