第五题：

源代码：

// main.cpp

// 第五章5

// Created by mac on 16/4/8.

// Copyright © 2016年 mac. All rights reserved.

#include <iostream>

using namespace std;

class X{

private:

int \*p;

public:

X(){p=new int[2];cout<<"X(). ";}

~X(){delete [] p;cout<<"~X(). \n";}

};

class Y:public X{

private:

int \*q;

public:

Y(){q=new int [1023];cout<<"Y():T::q="<<q<<". ";}

~Y(){delete [] q;cout<<"~Y().";}

};

int main(int argc, const char \* argv[]) {

// insert code here...

for(int i=0;i<8;i++){

Y \*t=new Y;

X \*r=t;

delete r;

}

return 0;

}

运行结果：

**X(). Y():T::q=0x100803600. ~X().**

**X(). Y():T::q=0x100804600. ~X().**

**X(). Y():T::q=0x100805600. ~X().**

**X(). Y():T::q=0x100806600. ~X().**

**X(). Y():T::q=0x100807600. ~X().**

**X(). Y():T::q=0x100808600. ~X().**

**X(). Y():T::q=0x100809600. ~X().**

**X(). Y():T::q=0x10080a600. ~X().**

**Program ended with exit code: 0**

**修改后的源代码：**

#include <iostream>

using namespace std;

class X{

private:

int \*p;

public:

X(){p=new int[2];cout<<"X(). ";}

virtual ~X(){delete [] p;cout<<"~X(). \n";}

};

class Y:public X{

private:

int \*q;

public:

Y(){q=new int [1023];cout<<"Y():T::q="<<q<<". ";}

~Y(){delete [] q;cout<<"~Y().";}

};

int main(int argc, const char \* argv[]) {

for(int i=0;i<8;i++){

Y \*t=new Y;

X \*r=t;

delete r;

}

return 0;

}

在基类中声明虚析构函数，用来释放积累对象指针指向派生类对象的的内存

运行结果：

**X(). Y():T::q=0x100803600. ~Y().~X().**

**X(). Y():T::q=0x100803600. ~Y().~X().**

**X(). Y():T::q=0x100803600. ~Y().~X().**

**X(). Y():T::q=0x100803600. ~Y().~X().**

**X(). Y():T::q=0x100803600. ~Y().~X().**

**X(). Y():T::q=0x100803600. ~Y().~X().**

**X(). Y():T::q=0x100803600. ~Y().~X().**

**X(). Y():T::q=0x100803600. ~Y().~X().**

**Program ended with exit code: 0**

**第六题：**

**源代码：**

#include <iostream>

#include <cstring>

#include <cstdlib>

using namespace std;

class strtype{

char \*p;

public:

strtype(char \*s);

~strtype(){delete [] p;}

char \*get(){return p;}

};

strtype::strtype(char \*s){

int l;

l=strlen(s)+1;

p=new char[l];

if(!p){

cout<<"Allocation error"<<endl;

exit(1);

}

strcpy(p, s);

}

void show(strtype x){

char \*S;

S=x.get();

cout<<S<<endl;

}

int main(int argc, const char \* argv[]) {

strtype a("Hello"),b("There");

show(a);

show(b);

return 0;

}

**运行结果：**

**Hello**

**There**

**第五章7(10127,0x100082000) malloc: \*\*\* error for object 0x100101c30: pointer being freed was not allocated**

**\*\*\* set a breakpoint in malloc\_error\_break to debug**

**分析原因：**

在外部函数show()中，局部形式参数x与实参对象匹配时，系统创建一份实参对象的拷贝，此时型参对象x的成员＊p也指向实参对象＊p所指向的内存单元，在函数生命期结束的时候，系统将调用析构函数回收堆内的内存资源。此时已将实参＊p成员所指向的内存单元释放。在主程序结束时再次调用析构释放堆中内存系统将会报错。

修改后源代码：

#include <iostream>

#include <cstring>

#include <cstdlib>

using namespace std;

class strtype{

char \*p;

public:

strtype(char \*s);

~strtype(){delete [] p;}

char \*get(){return p;}

};

strtype::strtype(char \*s){

int l;

l=strlen(s)+1;

p=new char[l];

if(!p){

cout<<"Allocation error"<<endl;

exit(1);

}

strcpy(p, s);

}

void show(strtype &x){

char \*S;

S=x.get();

cout<<S<<endl;

}

int main(int argc, const char \* argv[]) {

strtype a("Hello"),b("There");

show(a);

show(b);

return 0;

}

正确运行结果：

**Hello**

**There**

**Program ended with exit code: 0**

**第七题：**

**源代码：**

#include <iostream>

using namespace std;

class myclass{

public:

myclass();

myclass(const myclass &o);

myclass f();

};

myclass::myclass(){

cout<<"Constructing normally\n";

}

myclass::myclass(const myclass &o){

cout<<"Constructing copy\n";

}

myclass myclass:: f(){

myclass temp;

return temp;

}

int main(int argc, const char \* argv[]) {

myclass obj;

obj=obj.f();

return 0;

}

运行结果：

**Constructing normally**

**Constructing normally**

**Constructing copy**

**Program ended with exit code: 0**

**答：**

**在调用obj对象f()方法时返回了一个myclass 类的实例对象，此时在用obj对象接收是将调用myclass类的copy构造函数进行创建临时对象，并用临时对象的值为obj实例赋值，但在这里有可能调用c++语言默认的拷贝构造函数，在某些IDE程序运行结果有可能与本例程运行结果不同。**