5.2对象的生存期 P76

#include<iostream.h>

void fun();

void fun2();

void main()

{

fun();

fun();

fun2();

fun();

}

void fun()

{ static int a=1;

int I=5;

a++;

I++;

cout<<"I="<<I<<",a="<<a<<endl;

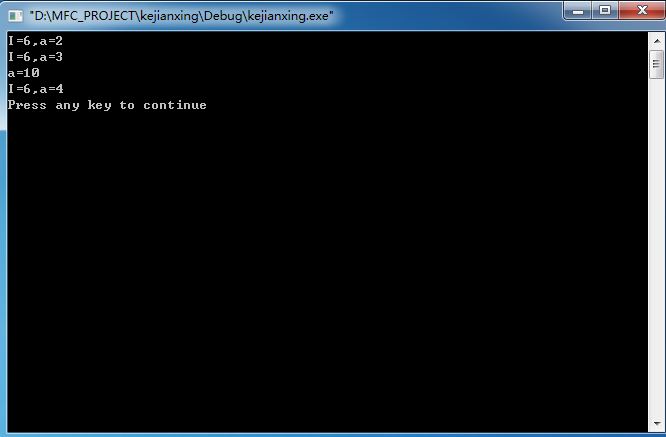
}

void fun2()

{ int a=10;

cout<<"a="<<a<<endl;

}



5.3静态成员函数和静态数据成员 P79

#include<iostream.h>

class Application

{

public:

static void f();

static void g();

private:

static int global;

};

int Application::global=0;

void Application::f()

{

global=5;

}

void Application::g()

{

cout<<global<<endl;

}

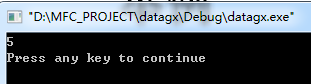
int main()

{ Application::f();

Application::g();

return 0;

}



5.3静态成员函数和静态数据成员 P81

#include<iostream.h>

class goods

{

public:

goods(int w);

~goods();

int Weight();

static int TotalWeight();

private:

int weight;

int w;

static int totalWeight;

};

goods::goods(int w)

{ weight=w;

totalWeight+=w;}

goods::~goods()

{ totalWeight-=w;

cout<<"OK"<<endl;

}

int goods::Weight()

{ return weight;}

int goods::TotalWeight()

{ return totalWeight;}

int goods::totalWeight=0;

int main()

{ int w;

cin>>w;

goods g1(w);

cin>>w;

goods g2(w);

cin>>w;

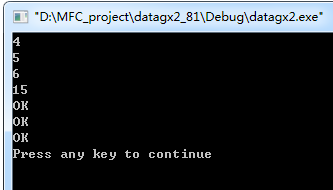
goods g3(w);

//cout<<goods::TotalWeight()<<endl;

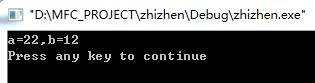
cout<<g2.TotalWeight()<<endl; //本行同上一行作用相同

return 0;

}



P85



P86-88

#if !define(\_LOCATION\_H)

#define LOCATION\_H

#include<iostream.h>

class Location{

public:

Location();

Location(int x,int y);

~Location();

void Move(int x,int y);

int GetX(){return X;}

int GetY(){return Y;}

private:

int X,Y;

};

#endif

Location::Location()

{ X=Y=0;

cout<<"Default constructor called."<<endl;

}

Location::Location(int x,int y)

{ X=x;Y=y;

cout<<"constructor called."<<X<<","<<Y<<endl;

}

Location::~Location()

{ cout<<"Destructor called."<<X<<","<<Y<<endl;

}

void Location::Move(int x,int y)

{ X=x;

Y=y;

}

int main()

{ Location A(5,10);

Location \*ptr;

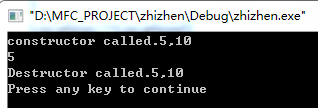
ptr=&A;

int x=ptr->GetX(); //int x=(\*ptr).GetX();

cout<<x<<endl;

return 0;

}



6.3引用 P95

#include<iostream.h>

int main()

{ int a[]={1,3,5,7,9};

int i;

cin>>i;

int &ref=a[i];

ref=55;

cout<<ref<<endl;

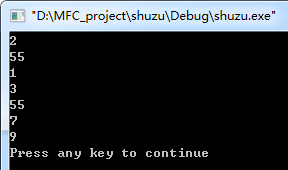
int j;

for(j=0;j!=5;j++)

cout<<a[j]<<endl;

return 0;

}



P96

#include<iostream.h>

int& f(int index,int a[])

{ int& r=a[index];

return r;

}

int main()

{ int a[]={1,3,5,7,9};

f(2,a)=55;

cout<<f(2,a)<<endl;

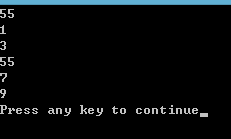
int j;

for(j=0;j!=5;j++)

cout<<a[j]<<endl;

return 0;

}



栈对象与堆对象 P98-99

#if !define(\_LOCATION\_H)

#define LOCATION\_H

#include<iostream.h>

class Location{

public:

Location();

Location(int x,int y);

~Location();

void Move(int x,int y);

int GetX(){return X;}

int GetY(){return Y;}

private:

int X,Y;

};

#endif

Location::Location()

{ X=Y=0;

cout<<"Default constructor called."<<endl;

}

Location::Location(int x,int y)

{ X=x;Y=y;

cout<<"constructor called."<<X<<","<<Y<<endl;

}

Location::~Location()

{ cout<<"Destructor called."<<X<<","<<Y<<endl;

}

void Location::Move(int x,int y)

{ X=x;

Y=y;

}

/\*int main()

{ Location A(5,10);

Location \*ptr;

ptr=&A;

int x=ptr->GetX(); //int x=(\*ptr).GetX();

cout<<x<<endl;

return 0;

}\*/

int main()

{ cout<<"Step one:"<<endl;

Location \*ptr1=new Location;

delete ptr1;

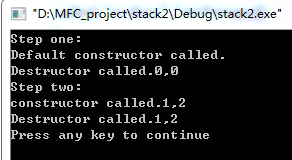
cout<<"Step two:"<<endl;

ptr1=new Location(1,2);

delete ptr1;

return 0;

}



多维数组与指针 P103

#include<iostream.h>

void f(int (\*p)[3],int n);

int main()

{ int a[2][3]={{1,2,3},{4,5,6}};

f(a,2);

return 0;

}

void f(int (\*p)[3],int n)

{ int i,j;

int \*pp=\*p+1;

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

cout<<pp[i]<<endl;

int(\*q)[3]=p+1; //指针加一即加上一行的维数

int \*pq=\*q+1;

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

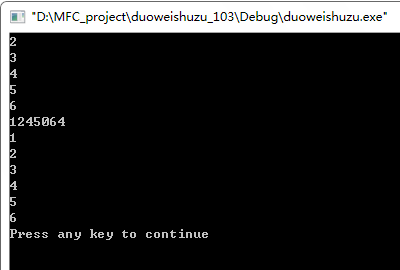
cout<<pq[i]<<endl;

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

for(j=0;j<3;j++)

cout<<p[i][j]<<endl;

}



具有子对象的类类型 P105-108

#if!defined(\_PART\_H)

#define PART\_H

#include <iostream.h>

class Part

{public:

Part();

Part(int i);

~Part();

void Print();

private:

int val;

};

#endif

//#include "part.h"

Part::Part()

{ val=0;

cout<<"Default constructor of Part"<<endl;

}

Part::Part(int i)

{ val=i;

cout<<"Constructor of Part"<<val<<endl;

}

Part::~Part()

{ cout<<"Destructor of Part"<<val<<endl;

}

void Part::Print()

{ cout<<val<<endl;

}

#if!defined(\_WHOLE\_H)

#define WHOLE\_H

//#include "part.h"

class Whole

{public:

Whole();

Whole(int I,int j,int k);

~Whole();

void Print();

private:

Part one;

Part two;

int date;

};

#endif

//#include "whole.h"

Whole::Whole()

{ date=0;

cout<<"Default constructor of Whole"<<endl;

}

Whole::Whole(int i,int j,int k):two(i),one(j),date(k)

{ cout<<"Constructor of Whole"<<endl;

}

Whole::~Whole()

{ cout<<"Destuctor of Whole"<<endl;

}

void Whole::Print()

{ one.Print();

two.Print();

cout<<date<<endl;

}

//#include "whole.h"

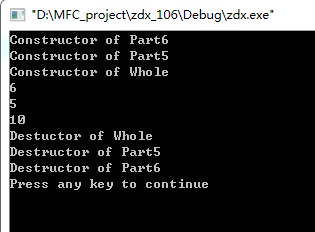
int main()

{ Whole anObject(5,6,10);

anObject.Print();

return 0;

}



（PS：课件上的程序好像有问题，自己改了半天还好用。答案课件上前两行反了）。