**例10.4 定义并测试长方形类CRect，长方形是由左上角坐标(left, top)和右下角坐标(right, bottom)组成。**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**class CRect //定义长方形类**

**{**

**private:**

**int left, top, right, bottom ;**

**public:**

**void setcoord(int, int, int, int);**

**void getcoord(int \*L, int \*T, int \*R, int \*B)**

**//注意：形参为指针变量**

**{**

**\*L = left; \*T = top;**

**\*R = right; \*B = bottom;**

**}**

**void print(void)**

**{**

**cout<<"Area = ";**

**cout<<abs(right－left)\***

**abs(bottom－top)<<endl;**

**}**

**};**

**void CRect::setcoord(int L, int T, int R, int B)**

**{**

**left=L; top=T; right=R; bottom=B;**

**}**

**int main(void)**

**{**

**CRect r, rr;**

**int a, b, c, d ;**

**r.setcoord(100, 300, 50, 200);**

**r.getcoord( &a, &b, &c, &d );**

**//用变量的指针做参数，带回多个结果**

**cout << "left=" << a << '\t';**

**cout << "top=" << b << endl;**

**cout << "right=" << c << '\t';**

**cout << "bottom=" << d << endl;**

**r.print( );**

**rr = r; //对象可整体赋值**

**rr.print( );**

**return 0;**

**}**

**运行结果：**

**left=100 top=300**

**right=50 bottom=200**

**Area = 5000**

**Area = 5000**

**返回ppt讲稿**

**例10.5 定义日期类，利用构造函数初始化数据成员。程序放在头文件date.h中，如下：**

**#include <iostream>**

**using namespace std;**

**class Date**

**{**

**int Year, Month, Day;**

**public:**

**Date( ) //重载构造函数 1**

**{**

**Year=2016; Month=5; Day=1;**

**}**

**Date(int y) //重载构造函数 2**

**{**

**Year=y; Month=5; Day=1;**

**}**

**Date(int y, int m) //重载构造函数 3**

**{**

**Year=y; Month=m; Day=1;**

**}**

**Date(int y, int m, int d) //重载构造函数 4**

**{**

**Year=y; Month=m; Day=d;**

**}**

**void ShowDate( )**

**{**

**cout <<Year<<'.'<<Month<<'.'<<Day<<endl;**

**}**

**};**

**主函数源文件为Li1005.cpp，内容如下：**

**#include "date.h"**

**int main( )**

**{**

**Date d1; //自动调用构造函数 1**

**Date d2(2016); //自动调用构造函数 2**

**Date d3(2016, 10); //自动调用构造函数 3**

**Date d4(2016, 10, 6); //自动调用构造函数 4**

**d1.ShowDate( );**

**d2.ShowDate( );**

**d3.ShowDate( );**

**d4.ShowDate( );**

**return 0;**

**}**

**运行结果是：**

**2016.5.1**

**2016.5.1**

**2016.10.1**

**2016.10.6**

当然我们可以定义带缺省值的构造函数，将上述构造函数简化，下述程序的功能与上述程序相当：

**#include <iostream>**

**using namespace std;**

**class Date**

**{**

**int Year, Month, Day;**

**public:**

**Date(int y=2016, int m=5, int d=1)**

**//带参数缺省值的构造函数**

**{**

**Year=y; Month=m; Day=d;**

**}**

**void ShowDate( )**

**{**

**cout <<Year<<'.'<<Month<<'.'<<Day<<endl;**

**}**

**};**

**int main( )**

**{**

**Date d1, d2(2016), d3(2016, 10), d4(2016, 10, 6);**

**d1.ShowDate( );**

**d2.ShowDate( );**

**d3.ShowDate( );**

**d4.ShowDate( );**

**return 0;**

**}**

**运行结果与上例一样。**

**返回ppt讲稿**

**例10.6 定义学生类，利用构造函数初始化数据成员，利用析构函数做清理工作。**

**#include <iostream>**

**#include <cstring>**

**using namespace std;**

**class Student**

**{**

**char Num[10]; //学号，注意：用数组实现**

**char \*Name; //姓名，注意：用指针实现**

**int Score; //成绩**

**public:**

**Student(char \*nump, char \*namep, int score)**

**{**

**if(nump) //在构造函数中，**

**{ //不需要动态申请Num成员的空间**

**strcpy(Num, nump);**

**}**

**else**

**strcpy(Num, "");**

**if(namep) //在构造函数中，**

**{ //需动态申请Name成员的空间**

**Name=new char[strlen(namep)+1];**

**strcpy(Name, namep);**

**}**

**else Name=** **NULL;**

**Score=score;**

**cout<<"Constructor Called!\n";**

**}**

**~Student( ) //在析构函数中，**

**{ //需释放Name成员的空间**

**if(Name) delete [ ] Name;**

**cout<<"Desturctor Called!\n";**

**}**

**void Show( )**

**{**

**cout << Num << '\t';**

**cout << Name << '\t';**

**cout << Score << endl;**

**}**

**};**

**int main( )**

**{**

**Student a("040120518", "George", 80);**

**a.Show( );**

**return 0;**

**}**

**此程序运行结果是：**

**Constructor Called! //调用构造函数时的输出**

**040120518 George 80**

**Desturctor Called! //调用析构函数时的输出**

**返回ppt讲稿**

**例10.7 调用构造函数和析构函数的时机**

**#include <iostream>**

**using namespace std;**

**class Date**

**{**

**int Year, Month, Day;**

**public:**

**Date(int y = 2000, int m = 1, int d = 1)//A，所有参数都有默认值**

**{**

**Year = y; Month = m; Day = d;**

**cout << "Constructor: ";**

**ShowDate();**

**}**

**void ShowDate()**

**{**

**cout << Year << '.' << Month << '.' << Day << endl;**

**}**

**~Date()**

**{**

**cout << "Destructor: ";**

**ShowDate();**

**}**

**};**

**Date d4(2016, 4, 4); //全局对象（静态的）**

**void fun()**

**{**

**cout << "进入 fun()函数!\n";**

**static Date d2(2016, 2, 2); //局部静态对象**

**Date d3(2016, 3, 3); //局部动态对象**

**cout << "退出 fun()函数!\n";**

**}**

**int main()**

**{**

**cout << "进入 main()函数!\n";**

**Date d1(2016, 1, 1); //局部动态对象**

**fun();**

**fun();**

**cout << "退出 main()函数!\n";**

**return 0;**

**}**

**此程序运行结果是：**

**Constructor: 2016.4.4 //调用构造函数，产生d4对象**

**进入 main( )函数!**

**Constructor: 2016.1.1 // 调用构造函数，产生d1对象**

**进入 fun( )函数!**

**// 第1次进入fun( )函数，产生下述 d2, d3对象**

**Constructor: 2016.2.2**

**Constructor: 2016.3.3**

**退出 fun( )函数!**

**// 退出fun( )函数，撤消d3对象，不撤消d2对象**

**Destructor: 2016.3.3**

**进入 fun( )函数!**

**// 第2次进入fun( )函数，再次产生d3对象**

**Constructor: 2016.3.3**

**退出 fun( )函数!**

**Destructor: 2016.3.3 // 退出fun( )函数，撤消d3对象**

**退出 main( )函数!**

**// 退出main ( )函数，撤消d1, d2, d4对象**

**Destructor: 2016.1.1**

**Destructor: 2016.2.2**

**Destructor: 2016.4.4**

**返回ppt讲稿**

**例10.9 定义一个“平面坐标点”类，测试拷贝构造函数的调用。**

**//头文件 "point.h"**

**class Point**

**{**

**int x, y;**

public:

**Point(int a=0, int b=0) //缺省构造函数**

**{**

**x=a; y=b;**

**}**

**Point(Point &p); //拷贝构造函数原型说明**

**~Point( ) //析构函数**

**{**

**cout<<x<<','<<y<<" Destructor Called.\n" ;**

**}**

**void Show( )**

**{**

**cout<<"Point: "<<x<<','<<y<<endl;**

**}**

**int Getx( )**

**{ return x; }**

**int Gety( )**

**{ return y; }**

**};**

**Point::Point(Point &p) //定义拷贝构造函数**

**{**

**x=p.x; y=p.y;**

**cout<<x<<','<<y<<" Copy-initialization Constructor Called.\n";**

**}**

**//文件 Li1009.cpp**

**#include <iostream>**

**using namespace std;**

**#include "point.h"**

**int main( )**

**{**

**Point p1(6, 8), p2(4, 7);**

**Point p3(p1); // A调用拷贝构造函数**

**Point p4=p2; // B调用拷贝构造函数**

**p1.Show( );**

**p3.Show( );**

**p2.Show( );**

**p4.Show( );**

**return 0;**

**}**

**此程序运行结果是：**

**6, 8 Copy-initialization Constructor Called.**

**4, 7 Copy-initialization Constructor Called.**

**Point: 6, 8**

**Point: 6, 8**

**Point: 4, 7**

**Point: 4, 7**

**4, 7 Destructor Called. // 撤销 P4**

**6, 8 Destructor Called. // 撤销 P3**

**4, 7 Destructor Called. // 撤销 P2**

**6, 8 Destructor Called. // 撤销 P1**

**析构函数与构造函数的调用顺序相反**

**返回ppt讲稿**

**例10.10 不定义拷贝构造函数时，运行出错。**

**//文件 Li1010.cpp**

**#include <iostream>**

**#include <cstring>**

**using namespace std;**

**class Student**

**{**

**char \*Name; //姓名，注意：用指针实现**

**int Age; //年龄**

public:

**Student(char \*namep, int age) //构造函数**

**{**

**Age=age;**

**if(namep) //在构造函数中，需动态申请空间**

**{**

**Name=new char[strlen(namep)+1];**

**strcpy(Name, namep);**

**}**

**else Name=NULL;**

**}**

**~Student( )//因在构造函数中动态申请了空间，**

**{ //则在析构函数中，需释放空间**

**if(Name) delete [ ] Name;**

**}**

**void Show( )**

**{**

**cout << Name << ',' << Age << endl;**

**}**

**};**

**int main()**

**{**

**Student a("George", 20);**

**Student b=a; // A**

**return 0;**

**}**

此程序运行时出错，原因是: 没有定义类的拷贝构造函数。系统自动产生的拷贝构造函数如下：

**Student::Student(const Student &s)**

**{**

**Name = s.Name; // 注意：地址值直接赋值**

**Age = s.Age;**

**}**

**Name:**

**1000**

**Age:**

**20**

**a**

**'G' 'e' 'o' 'r' 'g' 'e' '\0'**

**Name:**

**1000**

**Age:**

**20**

**b**

**正确的做法是，定义如下拷贝构造函数：**

**Student::Student(const Student &s)**

**{**

**Age=s.Age;**

**if(s.Name)**

**{**

**Name = new char[strlen(s.Name)+1]; //C**

**strcpy(Name, s.Name);**

**}**

**else Name=NULL;**

**}**

**Name:**

**1000**

**Age:**

**20**

**a**

**'G' 'e' 'o' 'r' 'g' 'e' '\0'**

**Name:**

**2000**

**Age:**

**20**

**b**

**'G' 'e' 'o' 'r' 'g' 'e' '\0'**

**返回ppt讲稿**

**例10.11 在本例中，使用例10.9中“平面坐标点”类的头文件point.h，测试用对象做函数参数及函数返回值时拷贝构造函数的使用。**

**#include <iostream>**

**using namespace std;**

**#include "point.h" //普通函数，不是类的成员函数**

**Point move(Point p, int xoffset, int yoffset)**

**{**

**int x = p.Getx( )+xoffset, y = p.Gety( )+yoffset;**

**Point t(x, y);**

**Point p=p1 //参数传递时**

**Point 内存临时对象=t。//返回对象时**

**return t;**

**}**

**int main( )**

**{**

**Point p1(6, 8), p2;**

**p2=move(p1, 2, 4);**

**return 0;**

**}**

**此程序运行结果是：**

**6, 8 Copy-initialization Constructor Called. //A**

**8, 12 Copy-initialization Constructor Called. // B**

**8, 12 Destructor Called. // 撤消对象t**

**6, 8 Destructor Called. // 撤消对象p**

**8, 12 Destructor Called. // 撤消内存临时对象**

**8, 12 Destructor Called. // 撤消对象p2**

**6, 8 Destructor Called. // 撤消对象p1**

**返回ppt讲稿**

**例10.13 利用构造函数完成类型转换**

**//文件 Li1013.cpp**

**#include <iostream>**

**using namespace std;**

**class Complex**

**{**

**double Real, Image;**

**public:**

**Complex(double x=0, double y=0)**

**{**

**Real=x; Image=y;**

**Show( );**

**cout << "调用了构造函数\n";**

**}**

**~Complex( )**

**{**

**Show( );**

**cout << "调用了析构函数\n";**

**}**

**void Show( )**

**{**

**cout<<'('<<Real<<','<<Image<<')';**

**}**

**};**

**int main( )**

**{**

**Complex c1(3, 5), c2; //A**

**c1 = 8.0; //B 等价于c1 = Complex(8.0);**

**c2 = Complex(9.0, 9.0); //C**

**return 0;**

**}**

**此程序运行结果是：**

**(3, 5)调用了构造函数 //在A行创建c1对象时，**

**//调用构造函数**

**(0, 0)调用了构造函数 //在A行创建c2对象时，**

**//调用构造函数**

**(8, 0)调用了构造函数**

**//创建、撤消临时对象，B行**

**(8, 0)调用了析构函数**

**(9, 9)调用了构造函数**

**//创建、撤消临时对象，C行**

**(9, 9)调用了析构函数**

**(9, 9)调用了析构函数 //在程序结束，**

**//撤消c2对象时，调用析构函数**

**(8, 0)调用了析构函数 //在程序结束，**

**//撤消c1对象时，调用析构函数**

**返回ppt讲稿**

**例10.15 处理线性表**

**#include <iostream>**

**using namespace std;**

**class ListClass**

**{**

**int \*ListPtr; // 指向线性表的指针**

**int nLen; // 线性表的长度**

**int nElem; // 线性表中当前元素的个数**

**public:**

**ListClass(int n=10) // 构造函数，初始化线性表,**

**最大长度的缺省值为10**

**{**

**nElem=0;**

**nLen=n;**

**if(n)**

**ListPtr = new int[n];**

**else**

**ListPtr=0;**

**}**

**~ListClass(void) // 析构函数**

**{ delete [nLen] ListPtr; }**

**int Elem(int); //重载函数①,在线性表尾增加一个元素**

**int &Elem(unsigned n) // 重载函数②,**

**返回线性表中第n个元素的引用**

**{ return ListPtr[n]; }**

**int Elem(void) //重载函数③,**

**返回线性表中当前元素的个数**

**{ return nElem; }**

**int Len(void) // 返回线性表的长度**

**{ return nLen; }**

**int GetElem(int i) // 返回线性表第i个元素的值**

**{**

**if((i>=0) && (i<nElem))**

**return ListPtr[i];**

**else**

**{**

**cout << "下标越界" << endl;**

**return(－1);**

**}**

**}**

**void Print(void); // 输出线性表中的所有元素，**

**在类体外实现**

**};**

**int ListClass::Elem(int elem) // 重载函数①，**

**在线性表尾增加一个元素**

**{**

**if(nElem==nLen) // 线性表已满**

**{**

**int \*newptr;**

**newptr = new int[nLen+10]; // A行，**

**申请新线性表空间**

**for(int i=0; i<nElem; i++)**

**newptr[i] = ListPtr[i]; // 将原线性表中的元素**

**拷贝到新线性表中**

**delete[nLen]ListPtr; // 释放原线性表空间**

**nLen += 10;**

**ListPtr = newptr; // 让指针指向新线性表空间**

**}**

**ListPtr[nElem++] = elem; // 添加元素**

**return(nElem); // 返回新线性表的元素个数**

**}**

**void ListClass::Print(void) // 输出线性表中全体元素**

**{**

**for(int i=0; i<nElem; i++)**

**cout << ListPtr[i] << '\t';**

**cout << endl;**

**}**

**int main(void)**

**{**

**ListClass list(6);**

**for(int i=0; i<5; i++)**

**list.Elem(i); // 调用重载函数①**

**cout<<"线性表的长度为："<<list.Len( )<<endl;**

**cout<<"线性表的元素个数为：";**

**cout<<list.Elem( )<<endl; // 调用重载函数③**

**cout<<"线性表的元素为：";**

**list.Print( );**

**list.Elem(3u) = 100; // 调用重载函数②**

**cout<<"线性表下标为3的元素的值为: ";**

**cout<<list.Elem(3u)<<endl; // 调用重载函数②**

**list.Elem(20); // 调用重载函数①**

**list.Elem(200); // 调用重载函数①**

**cout<<"现在线性表的长度为："<<list.Len( )<<endl;**

**cout << "现在线性表中的元素个数为: ";**

**cout <<list.Elem( )<<endl; // 调用重载函数③**

**cout<<"线性表的元素为：";**

**list.Print( );**

**cout << "线性表的最后一个元素为: ";**

**cout<<list.GetElem(list.Elem( )－1)<<endl;**

**// 调用重载函数③**

**return 0;**

**}**

**程序的运行结果：**

**线性表的长度为：6**

**线性表的元素个数为：5**

**线性表的元素为：0 1 2 3 4**

**线性表下标为3的元素的值为: 100**

**现在线性表的长度为：16**

**现在线性表中的元素个数为: 7**

**线性表的元素为：0 1 2 100 4 20 200**

**线性表的最后一个元素为: 200**

**返回ppt讲稿**

**例10.16 初始化对象成员。**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**class Point //定义“点”类**

**{**

**int x, y;**

public:

**Point(int a=0, int b=0)**

**{**

**x=a; y=b;**

**cout<<x<<','<<y<<" 构造 Point\n";**

**}**

**int Getx( )**

**{ return x; }**

**int Gety( )**

**{ return y; }**

**~Point( )**

**{ cout<<x<<','<<y<<" 析构 Point\n"; }**

**};**

**class Line //定义“线”类，两点决定一条线**

**{**

**int width, color; //指定“线”的宽度、颜色**

**Point p1, p2; //指定“线”的两个端点**

public:

**Line(int x1, int y1, int x2, int y2, int w, int c) : p1(x1,y1), p2(x2,y2) // A**

**{**

**width=w; color=c;**

**cout<<width<<','<<color<<"构造 Line\n";**

**}**

**double LineLen( )**

**{**

**double len;**

**int x1, y1, x2, y2;**

**x1=p1.Getx( ); y1=p1.Gety( );**

**x2=p2.Getx( ); y2=p2.Gety( );**

**len=sqrt((x1－x2)\*(x1－x2)+(y1－y2)\*(y1－y2));**

**return(len);**

**}**

**~Line( )**

**{**

**cout<<width<<','<<color<<"析构 Line\n";**

**}**

**};**

**int main( )**

**{**

**Line Li(0, 0, 1, 1, 3, 6);**

**cout<<"长度="<<Li.LineLen( )<<endl;**

**return 0;**

**}**

**此程序运行结果是：**

**0, 0 构造 Point**

**1, 1 构造 Point**

**3, 6构造 Line**

**长度=1.41421**

**3, 6析构 Line**

**1, 1 析构 Point**

**0, 0 析构 Point**

**返回ppt讲稿**

**例 10.17 说明this指针的使用。**

**#include <iostream>**

**using namespace std;**

**class Sample**

**{**

**int x, y;**

**public:**

**Sample( int a=0, int b=0)**

**{**

**this－>x = a; // 通过this指针访问数据成员**

**this－>y = b; // 通过this指针访问数据成员**

**}**

**void Print( )**

**{**

**cout << this << '\t'; // E**

**cout << this－>x <<'\t'; // 通过this指针访问数据成员**

**cout << this－>y <<'\n'; // 通过this指针访问数据成员**

**}**

**};**

**int main( )**

**{**

**Sample c1(1,4), c2(3,7);**

**cout << &c1 <<'\n'; // A**

**c1.Print( ); // B**

**cout << &c2 <<'\n'; // C**

**c2.Print( ); // D**

**return 0;**

**}**

**程序的运行结果是：**

**0x0012FF78 //输出对象c1的地址**

**0x0012FF78 1 4 // E行输出this指针值**

**0x0012FF70 //输出对象c2的地址**

**0x0012FF70 3 7 // E行输出this指针值**

**返回ppt讲稿**