**《面向对象程序设计C++》实验指导书**

**（一）**

实验一 数据封装

1. 开发环境

所有实验均在Windows环境下开发，编程工具为Visual Studio 2010。请自行学习VS的使用。

* 注意：本次实验建造的工程必须是“控制台应用”，并且一定要选择“空项目”，否则系统将会自动生成带预编译头文件的工程，该工程的主函数是\_tmain()而非main()，并且工程的体积也很大。

1. 关键技术解析

List类用链表的方式存储对象。这产生了一个问题：链表的节点中包含的数据域的类型是一样，而我们需要存储五种不同的形体。显然，节点中的数据域的类型不能是五种形体类中的任何一个，只能用一种相对统一的方式来完成。这里给出一个简单的思路：编写一个与五种形体都很相似的类Quadrangle类，这个类至少包含了与其它五种形体类一样的数据成员：name。这样，节点的数据域的类型就是Quandrangle \*。但在遍历List类的对象时，通过name成员获得形体对象的标识，然后再通过类型强制转换获得正确的对象指针，最后在调用对象的相应成员函数完成指定的操作。

1. 示例代码
2. Quadrangle类

//bzj^\_^

//quad.h

**#pragma once**

**class** Quadrangle

{

**public**:

string name; //形体的标识

};

**typedef** Quadrangle \* QUADPTR;

//end of quad.h

1. Rect类（注意：请不要将矩形类命名为Rectangle）

//bzj^\_^

//rect.h

**#pragma once**

**class** Rect

{

**public**:

string name; //形体的标识

Rect (**int** w = 5, **int** h = 7, string nm = "Rectangle");

~Rect ();

**void** draw() **const**;

**double** area() **const**;

string what() **const**;

**int&** Width(); //获取矩形的宽。如果是梯形，还需要一个成员int& Width2()来获取另一条平行边的长度

**int&** Height();//获取矩形的高

**private**:

**int** width, height; //高和宽

};

//end of rect.h

//bzj^\_^

//rect.cpp

**#include** <iostream>

**#include** <string>

**using** **namespace** std;

**#include** "rect.h"

**//这里请自行添加构造函数和析构函数的代码**

**void** Rect::draw() **const**

{

cout << what() <<": width = " << width << ", height = " << height << endl;

}

**//请自行添加成员函数Width()、Height()、area()和what()的代码。注意函数原型要和声明是完全一致。**

1. List类

//bzj^\_^

//list.h

**#pragma once**

**#include** "quad.h"

**typedef** **void** (\*ACCESSFUN)(QUADPTR); //定义遍历时节点处理函数类型

**class** List

{

**private**:

**struct** \_Node

{

QUADPTR data;

\_Node\* next;

} \*head, \*tail; //链表的头尾指针

typedef \_Node\* \_range;

size\_t len; //链表中节点的数目，即链表的长度

**public**:

List(); //平凡构造函数

List(**const** List& l); //复制构造函数

~List();

**void** push\_back(**const** QUADPTR quad); //尾部添加

size\_t size(); //获取链表的长度

**void** traverse(ACCESSFUN f); //遍历

};

//end of list.h

//bzj^\_^

//list.cpp

**#include** <iostream>

**#include** <string>

**using** **namespace** std;

**#include** "list.h"

List::List(){**//请自行添加平凡构造函数的代码**}

List::List(**const** List& l) {**//请自行添加复制构造函数的代码。注意：要求完成深复制。**}

List::~List()

{

**//请自行添加析构函数的代码。注意：在析构时要释放整个链表**

}

//insert element into the tail of list

**void** List::push\_back(**const** QUADPTR quad)

{

**//请自行添加在链表尾部增加一个节点的代码，将quad存入节点的数据域。**

}

size\_t List::size() { **//请自行添加代码** }

**void** List::traverse(ACCESSFUN f)

{

\_range p = **//请自行添加初始化部分**

**while** (p != NULL)

{

f(p->data);

**//请自行添加指针移动部分**

}

}

1. 遍历时使用的访问函数

**void** access(QUADPTR quad)

{

**if** (quad->name == "Rectangle")

{

Rectangle \*p = **reinterpret\_cast**<Rectangle \*>(quad);

p->draw();

cout << "area = " << p->area() << endl;

}

//else if (...)

**//请自行将处理其它形体的代码补充完整**

}

1. 要求和建议
2. 五种形体的名字标识要求使用如下字符串（单词的第一个字母大写）：
   * 四边形：Quadrangle
   * 平行四边形：Parallelogram
   * 梯形：Trapezoid
   * 矩形：Rectangle
   * 菱形：Diamond
   * 正方形：Square
3. 在main函数中：
   * 不能预先定义形体对象，而采用随机的方式生成形体。这需要使用两个函数：

* 随机数发生器初始化：srand(unsigned(time(NULL))); 该函数只需调用一次；
* 随机数发生器：rand();该函数产生0-INT\_MAX之间的伪随机数。

为了使用以上函数，需要包含以下几个头文件：

cstdlib

ctime

* + 形体的所有关键数据也采用随机的方式生成
  + 程序模拟学习过程，而学习过程分为两种模式：学习模式和复习模式

1. 程序流程
   * 程序首先进入学习模式。在该模式下：
     1. 随机产生一个形体及其关键数据，并将形体加入到列表中，然后“画”出这个形体（包括显示形体的关键数据）。此后要求学习者输入面积，程序判断输入是否正确，如果不正确则要求继续输入，直到正确为止；
     2. 此后程序询问学习者是否继续。如果是，则重复上一步；
     3. 否则，程序进入复习模式。
   * 在复习模式下，程序“画”出学习者学过的每一个形体。此后程序结束。
2. 请将你所做的工程全部保存到移动介质中，以后每一次的实验都要以上一次的为基础。
   * 提示：请在保存之前先将工程做一个清理，删掉那些无用文件。
3. 实验完成后，填写实验报告。实验报告中：
   * 项目八：实验步骤，填写程序中用到的关键技术介绍、流程图、算法分析和设计等；
   * 项目九：实验数据及结果分析，粘贴程序运行的几张截图；程序结果正确与否；

提示：按ALT+RPINTSCREEN可以只截取当前活动窗口的图像而非整屏。

* + 项目十：实验结论，填写程序是否实现了实验内容的全部，是否达到预期目标；
  + 项目十一：总结及心得体会，填写你做本次实验的所学所得；
  + 附件（报告中没有，请在新页中添加）：所有程序代码。
  + **本次实验报告提交到网络学堂，不需要打印稿。**

实验二 运算符重载

对于List类重载[]说明如下：

如果有：List list;

那么list[2]返回list对象内部链表中第2个（从0开始计数）节点中quad域存储的数据，亦即类型为QUADPTR的指针。

提示：需要用到与遍历相似的算法。

建议：该函数的原型为：

QUADPTR& operator[](int index);

除此之外，本次实验无特殊要求，请按实验报告中的内容完成。实验过程和要求请参考实验一。