**北 京 邮 电 大 学**

**计 算 机 科 学 与 技 术 学 院**

**C＋＋程序设计**

**实验报告**

**姓名** 李剑

**学号** 2014211144

**班级** 2014211301

**实验一 简单C＋＋程序设计**

**1．源代码**

/\*一.1猜价格游戏\*/

#include<iostream>

#include<cstdlib>

#include<ctime>

using namespace std;

int main(void)

{

char input[4]={0};//保存输入的

int price,hypos;

int isleap=0,i=0;

srand(time(NULL));

price=(rand()%1000)+1;//随机生成一个价格（1-1000）

cout<<"方便测试，这是正确答案"<<price<<endl;

//提示输入

cout << "Now there is a merchandise, it's price is between 1 and 1000.please guess the price of it." << endl;

cout << "Price:";

cin >> input;

while(1)

{

isleap=0;

while(input[i]!='\0')

{

//检查输入是否合法

if(input[i]<'0' || input[i]>'9' || atoi(input)>1000)

{

isleap=1;

break;

}

i++;

}

i=0;

if(isleap==0)//输入合法

{

hypos=atoi(input);

//判断用户猜的是否正确

if(hypos==price)//猜对了

{

cout << "God damn it,you are right now!"<<endl;

break;

}

else

{

//当猜的不对的时候做出适当的提示

if(hypos<price)//猜的过小

{

cout << "You should give a bigger number.\n";

cout << "Price:";

cin >> input;

}

else//猜的过大

{

cout << "You should give a smaller number.\n";

cout << "Price:";

cin >> input;

}

}

}

else//输入不合法

{

cout << "Please input from 1 to 1000.\n Please input again:";

cin >> input;

}

}

system("pause");

return 0;

}

**2．实验总结：**这道题帮助我复习了以前c语言学的随机函数相关的知识，并且在对用户输入的合法性做出判断的时候有了更多的考虑。比如，用户输入超过1000的数，或者输入字符串，都要做出提示。

**实验二 类与对象**

**1．源代码**

/\*二.1矩形\*/

#include<iostream>

#include<cstdlib>

#include<cmath>

using namespace std;

//定义一个Point类，其属性包括点的坐标，提供计算两点之间距离的方法；

class Point{

public:

double dis(const Point & p) const{

return sqrt((xVal-p.xVal)\*(xVal-p.xVal)+(yVal-p.yVal)\*(yVal-p.yVal));

}

double getX(){ return xVal; }; //得到x坐标

double getY(){ return yVal; }; //得到y坐标

Point(double x,double y); //构造函数

~Point(); //析构函数

private:

double xVal;

double yVal;

};

Point::~Point(){}

Point::Point(double x=0,double y=0)//构造函数 默认为零

{

xVal=x;

yVal=y;

}

//定义一个矩形类，其属性包括左上角和右下角两个点，提供计算面积的方法

class recTangle{

public:

double calArea();

recTangle(Point l,Point r);

~recTangle();

private:

Point leftUpper;

Point rightLow;

};

double recTangle::calArea()

{

double dx,dy;

double area;

dy=leftUpper.getY()-rightLow.getY();

dx=rightLow.getX()-leftUpper.getX();

area=dx\*dy;

return area;

}

recTangle::recTangle(Point l,Point r)

{

leftUpper=l;

rightLow=r;

}

recTangle::~recTangle(){

}

int main(void)

{

double x,y,s,t;

int islegal(0);//检查输入是否合法

while(islegal==0)

{

cout<<"请输入左上角的坐标(用空格隔开)：";

cin>>x>>y;

cout<<"请输入右下角的坐标(用空格隔开)：";

cin>>s>>t;

if(x<s && y>t)//判断用户输入合法

islegal=1;

else

cout<<"输入不合法请重新输入\n";

}

Point l(x,y),r(s,t);

cout<<"这两点之间的距离是："<<l.dis(r)<<endl;

recTangle rec(l,r);

cout<<"矩形的面积是:"<<rec.calArea()<<endl;

system("pause");

return 0;

}

/\*二 3友元\*/

#include<iostream>

#include<cstdlib>

using namespace std;

//定义一个Boat和Car两个类，他们都具有私用属性--重量

class Car;//因为要在Boat类里用，先声明

class Boat

{

friend double

TotalWeight(Boat& b, Car& c);

public:

Boat(double wet);

private:

double weight;

};

class Car

{

friend double

TotalWeight(Boat& b, Car& c);

public:

Car(double wet);

private:

double weight;

};

//构造函数

Boat::Boat(double wet)

{

weight=wet;

}

Car::Car(double wet)

{

weight=wet;

}

//用友元函数计算总重

double TotalWeight(Boat& b, Car& c)

{

return (b.weight+c.weight);

}

int main (void)

{

double car,boat;//车重和船重

double sum;

cout << "Please input the weight of car and boat(\*\* \*\*):";

cin >> car >> boat;

Car m(car);

Boat n(boat);

sum=TotalWeight(n,m);

cout << "The totle weight is " << sum << "." << endl;

system("pause");

return 0;

}

**2．实验总结：**这次试验是首次使用类，是我从面向过程转换到面向对象的第一步，友元的使用可以给我们带来一些方便，可以在一定程度上破坏程序的封闭性。

**实验三 数组与指针**

**1．源代码**

/\*三.1矩阵1\*/

#include<iostream>

#include<stdlib.h>

//假定矩阵大小为4×5

#define row 4

#define line 5

using namespace std;

//定义矩阵初始化函数，可以从cin中输入矩阵元素

void initialMax(int matrix[][line]);

//定义矩阵输出函数，将矩阵格式化输出到cout

void disPlay(int matrix[][line]);

//定义矩阵相加的函数，实现两个矩阵相加的功能,结果保存在另一个矩阵中

void addingMax(int mat\_1[][line],int mat\_2[][line],int result[][line]);

//定义矩阵相减的函数，实现两个矩阵相减的功能,结果保存在另一个矩阵中

void distractMax(int mat\_1[][line],int mat\_2[][line],int result[][line]);

int main(void)

{

//定义三个矩阵：A1、A2、A3

int A1[row][line],A2[row][line],A3[row][line];

cout<<"We assume the size is 4\*5"<<endl;

//初始化A1、A2

initialMax(A1);

initialMax(A2);

//计算并输出：A3 = A1加A2，A3 = A1减A2

addingMax(A1,A2,A3);

cout<<"After adding,";

disPlay(A3);

distractMax(A1,A2,A3);

cout<<"After distracting,";

disPlay(A3);

system("pause");

return 0;

}

void distractMax(int mat\_1[][line],int mat\_2[][line],int result[][line])

{

int i,j;

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

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

result[i][j]=mat\_1[i][j]-mat\_2[i][j];

}

void addingMax(int mat\_1[][line],int mat\_2[][line],int result[][line])

{

int i,j;

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

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

result[i][j]=mat\_1[i][j]+mat\_2[i][j];

}

void initialMax(int matrix[][line])

{

int i,j;

cout<<"Please input a matrix:\n";

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

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

cin>>matrix[i][j];

}

void disPlay(int matrix[][line])

{

int i,j;

cout<<"the matrix is:\n";

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

{

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

cout<<matrix[i][j]<<" ";

cout<<endl;

}

}

/\*三 2矩阵二\*/

#include<iostream>

#include<stdlib.h>

//假定矩阵大小为4×5

#define row 4

#define line 5

using namespace std;

//定义矩阵初始化函数，可以从cin中输入矩阵元素

void initialMax(int \*\*matrix);

//定义矩阵输出函数，将矩阵格式化输出到cout

void disPlay(int \*\*matrix);

//定义矩阵相加的函数，实现两个矩阵相加的功能,结果保存在另一个矩阵中

void addingMax(int \*\*mat\_1,int \*\*mat\_2,int \*\*result);

//定义矩阵相减的函数，实现两个矩阵相减的功能,结果保存在另一个矩阵中

void distractMax(int \*\*mat\_1,int \*\*mat\_2,int \*\*result);

int main(void)

{

int \*\*A1,\*\*A2,\*\*A3;//（2） 矩阵空间采用new动态申请，保存在指针中；

int i;

cout<<"We assume the size is 4\*5"<<endl;

//动态申请三个矩阵：A1、A2、A3；

A1=new int\*[row];

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

A1[i]=new int[line];

A2=new int\*[row];

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

A2[i]=new int[line];

A3=new int\*[row];

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

A3[i]=new int[line];

//初始化A1、A2；

initialMax(A1);

initialMax(A2);

//计算并输出A3 = A1加A2，A3 = A1减A2；

addingMax(A1,A2,A3);

cout<<"After adding,";

disPlay(A3);

distractMax(A1,A2,A3);

cout<<"After distracting,";

disPlay(A3);

//释放内存

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

delete [] A1[i];

delete [] A1;

A1=NULL;

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

delete [] A2[i];

delete [] A2;

A2=NULL;

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

delete [] A3[i];

delete [] A3;

A3=NULL;

system("pause");

return 0;

}

void distractMax(int \*\*mat\_1,int \*\*mat\_2,int \*\*result)

{

int i,j;

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

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

result[i][j]=mat\_1[i][j]-mat\_2[i][j];

}

void addingMax(int \*\*mat\_1,int \*\*mat\_2,int \*\*result)

{

int i,j;

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

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

result[i][j]=mat\_1[i][j]+mat\_2[i][j];

}

void initialMax(int \*\*matrix)

{

int i,j;

int t;

cout<<"Please input a matrix:\n";

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

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

cin>>matrix[i][j];

}

void disPlay(int \*\*matrix)

{

int i,j;

cout<<"the matrix is:\n";

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

{

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

cout<<matrix[i][j]<<" ";

cout<<endl;

}

}

/\*三 3 矩阵三\*/

#include<iostream>

#include<cstdlib>

using namespace std;

/\* 用类来实现矩阵，定义一个矩阵的类，属性包括：

矩阵大小，用 lines, rows（行、列来表示）；

存贮矩阵的数组指针，根据矩阵大小动态申请（new）。\*/

class Matrix{

public:

Matrix(int \_row,int \_line);//构造函数

~Matrix();//析构函数

Matrix(Matrix &other);//拷贝构造函数

void Input();//输入矩阵

void Print();//输出

void Add(Matrix &A,Matrix &B);//加法

void Distract(Matrix &A,Matrix &B);//减法

private:

int rows;

int lines;

int \*\*matrix;

};

void Matrix::Add(Matrix &A,Matrix &B)//加法

{

int i,j;

for(i=0;i<this->rows;i++)

for(j=0;j<this->lines;j++)

this->matrix[i][j]=A.matrix[i][j]+B.matrix[i][j];

}

void Matrix::Distract(Matrix &A,Matrix &B)

{

int i,j;

for(i=0;i<this->rows;i++)

for(j=0;j<this->lines;j++)

this->matrix[i][j]=A.matrix[i][j]-B.matrix[i][j];

}

void Matrix::Input()

{

int i,j;

for(i=0;i<this->rows;i++)

for(j=0;j<this->lines;j++)

cin>>this->matrix[i][j];

}

void Matrix::Print()

{

int i,j;

cout<<"the matrix is:\n";

for(i=0;i<this->rows;i++)

{

for(j=0;j<this->lines;j++)

cout<<this->matrix[i][j]<<" ";

cout<<endl;

}

}

Matrix::Matrix(Matrix &other)//拷贝构造函数

{

int i,j;

rows=other.rows;

lines=other.lines;

matrix=new int\*[rows];

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

matrix[i]=new int[lines];

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

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

matrix[i][j]=other.matrix[i][j];

}

Matrix::Matrix(int \_row,int \_line)//构造函数

{

int i;

rows=\_row;

lines=\_line;

matrix=new int\*[rows];

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

matrix[i]=new int[lines];

}

//析构函数

Matrix::~Matrix()

{

for(int i=0;i<this->rows;i++)

delete [] this->matrix[i];

delete [] this->matrix;

}

int main(void)

{

int row,line;

cout<<"Please input the row and line:";

cin>>row>>line;

//定义三个矩阵：A1、A2、A3

Matrix A(row,line),B(row,line),C(row,line);

cout<<"Please input matrix A:\n";

A.Input();

cout<<"Please input matrix B:\n";

B.Input();

//计算并输出A3 = A1加A2，A3=A1减A2

C.Add(A,B);

cout<<"After adding A and B,";

C.Print();

C.Distract(A,B);

cout<<"After distracting A and B,";

C.Print();

//用new动态创建三个矩阵类的对象：pA1、pA1、pA3

Matrix \*pA1 = new Matrix(row,line);

Matrix \*pA2 = new Matrix(row,line);

Matrix \*pA3 = new Matrix(row,line);

//初始化pA1、pA2

cout << "Please input matrix A1:" << endl;

(\*pA1).Input();

cout << "Please input matrix A2:" << endl;

(\*pA2).Input();

//计算并输出pA3=pA1加pA2，pA3=pA1减pA2

cout << "After adding A1 and A2,";

(\*pA3).Add(\*pA1, \*pA2);

(\*pA3).Print();

cout << "After distracting A1 and A2,";

(\*pA3).Distract(\*pA1, \*pA2);

(\*pA3).Print();

//释放pA1、pA1、pA3

delete pA1;

delete pA2;

delete pA3;

system("pause");

return 0;

}

**2．实验总结：**矩阵（一）让我复习了以前学习的矩阵的实现与操作，不同的是用 c++ 的语法来写代码，矩阵（二）是在（一）的基础上改为用动态申请数组的方式来实现，也就是 new 和 delete 的使用。矩阵（三）则更进一步，使用类来实现矩阵，这也是最不同的地方，最需要掌握的地方。这次实验还是很有帮助的，尤其是学到了用 new 动态申请类的对象。

**实验四 继承与派生**

**1．源代码**

/\*四 1 形状1\*/

#include<iostream>

#include<stdlib.h>

#define PI 3.14159

using namespace std;

//声明一个基类Shape，其中包含一个方法来计算面积

class Shape{

public:

double Area()

{

return x\*y;

};

Shape(double m,double n);

~Shape(){

cout<<endl<<"这是基类的析构函数\n";

};

protected:

double x,y;

};

Shape::Shape(double m,double n)

{

x=m;

y=n;

cout<<endl<<"这是基类的构造函数\n";

}

//从Shape派生两个类：矩形类和圆形类

class Rec:public Shape{

public:

double Area(){

return x\*y;

}

Rec(double m,double n):Shape(m,n){

cout<<endl<<"这是矩形的构造函数\n";

};

~Rec(){

cout<<endl<<"这是矩形的析构函数"<<endl;

};

};

class Circle:public Shape{

public:

double ShowArea(){

return x\*x\*PI;

};

Circle(double m,double n):Shape(m,n){

cout<<endl<<"这是圆形的构造函数\n";

};

~Circle(){

cout<<endl<<"这是圆的析构函数"<<endl;

};

};

//从矩形类派生正方形类

class Square:public Rec{

public:

Square(double ch,double ku):Rec(ch,ku){

cout<<endl<<"这是正方形的构造函数"<<endl;

};

~Square();

};

Square::~Square(){

cout<<endl<<"这是正方形的析构函数"<<endl;

}

int main(void)

{

//对不同对象计算面积

double chang,kuan;

cout<<"请输入矩形的长和宽：";

cin>>chang>>kuan;

Rec a(chang,kuan);

cout<<"矩形的面积是："<<a.Area()<<endl;

cout<<"请输入圆的半径和正方形的边长：";

cin>>chang>>kuan;//用chang表示圆的半径 kuan表示正方形的边长

Circle b(chang,chang);

Square c(kuan,kuan);

cout<<"圆的面积是："<<b.ShowArea()<<endl;

cout<<"正方形的面积是："<<c.Area()<<endl;

system("pause");

return 0;

}

/\*四 2 形状2.1\*/

#include<iostream>

#include<stdlib.h>

#define PI 3.14159

using namespace std;

//声明一个基类Shape，其中包含一个方法来计算面积

class Shape{

public:

virtual double Area()

{

return x\*y;

};

Shape(double m,double n);

~Shape(){

cout<<endl<<"这是基类的析构函数\n";

};

protected:

double x,y;

};

Shape::Shape(double m,double n)

{

x=m;

y=n;

cout<<endl<<"这是基类的构造函数\n";

}

class Rec:public Shape{

public:

double Area(){

return x\*y;

}

Rec(double m,double n):Shape(m,n){

cout<<endl<<"这是矩形的构造函数\n";

};

~Rec(){

cout<<endl<<"这是矩形的析构函数"<<endl;

};

};

class Circle:public Shape{

public:

double Area(){

return x\*x\*PI;

};

Circle(double m,double n):Shape(m,n){

cout<<endl<<"这是圆形的构造函数\n";

};

~Circle(){

cout<<endl<<"这是圆的析构函数"<<endl;

};

};

class Square:public Rec{

public:

Square(double ch,double ku):Rec(ch,ku){

cout<<endl<<"这是正方形的构造函数"<<endl;

};

~Square();

double Area(){

return x\*x;

}

};

Square::~Square(){

cout<<endl<<"这是正方形的析构函数"<<endl;

}

int main(void)

{

double chang,kuan;

cout<<"请输入矩形的长和宽：";

cin>>chang>>kuan;

Rec a(chang,kuan);

cout<<"矩形的面积是："<<a.Area()<<endl;

cout<<"请输入圆的半径和正方形的边长：";

cin>>chang>>kuan;//用chang表示圆的半径 kuan表示正方形的边长

Circle b(chang,chang);

Square c(kuan,kuan);

cout<<"圆的面积是："<<b.Area()<<endl;

cout<<"正方形的面积是："<<c.Area()<<endl;

system("pause");

return 0;

}

/\*四 2 形状2.2\*/

#include<iostream>

#include<stdlib.h>

#define PI 3.14159

using namespace std;

//声明一个抽象基类Shape，其中包含一个方法来计算面积

class Shape{

public:

virtual double Area()=0;

Shape(double m,double n);

~Shape(){

cout<<endl<<"这是基类的析构函数\n";

};

protected:

double x,y;

};

Shape::Shape(double m,double n)

{

x=m;

y=n;

cout<<endl<<"这是基类的构造函数\n";

}

class Rec:public Shape{

public:

double Area(){

return x\*y;

}

Rec(double m,double n):Shape(m,n){

cout<<endl<<"这是矩形的构造函数\n";

};

~Rec(){

cout<<endl<<"这是矩形的析构函数"<<endl;

};

};

class Circle:public Shape{

public:

double Area(){

return x\*x\*PI;

};

Circle(double m,double n):Shape(m,n){

cout<<endl<<"这是圆形的构造函数\n";

};

~Circle(){

cout<<endl<<"这是圆的析构函数"<<endl;

};

};

class Square:public Rec{

public:

Square(double ch,double ku):Rec(ch,ku){

cout<<endl<<"这是正方形的构造函数"<<endl;

};

~Square();

double Area(){

return x\*x;

}

};

Square::~Square(){

cout<<endl<<"这是正方形的析构函数"<<endl;

}

int main(void)

{

double chang,kuan;

cout<<"请输入矩形的长和宽：";

cin>>chang>>kuan;

Rec a(chang,kuan);

cout<<"矩形的面积是："<<a.Area()<<endl;

cout<<"请输入圆的半径和正方形的边长：";

cin>>chang>>kuan;//用chang表示圆的半径 kuan表示正方形的边长

Circle b(chang,chang);

Square c(kuan,kuan);

cout<<"圆的面积是："<<b.Area()<<endl;

cout<<"正方形的面积是："<<c.Area()<<endl;

system("pause");

return 0;

}

**2．实验总结：**此次实验是对于 c++ 类的派生与继承的进一步理解，值得注意的是构造函数，析构函数的调用顺序，在一般情况下，调用析构函数的次序正好与调用构造函数的次序相反：最先被调用的构造函数，其对应的(同一对象中的)析构函数最后被调用，而最后被调用的构造函数，其对应的析构函数最先被调用。虚函数与抽象类也是很重要的。虚函数可以实现多态性，通过指向派生类的基类指针，访问派生类中的同名覆盖函数，而且虚函数必须是基类的非静态成员函数，由于虚函数还提供了实现，派生类可以选择改写或者不该写。抽象类包含了纯虚函数，在基类中没有必要或者无法实现，需要在派生类里实现。

**实验五 多态性**

**1．源代码**

/\*五 1 运算符重载\*/

#include<iostream>

#include<stdlib.h>

using namespace std;

//Point类的属性包括点的坐标（x，y）

class Point{

public:

void showPoint(void)

{

cout<<"("<<x<<' '<<y<<")"<<endl;

};

Point(int dx,int dy);

~Point();

//对Point类重载＋＋和――运算符

void operator++()

{

x++; y++;

}

void operator--()

{

x--; y--;

}

void operator++(int)

{

x++; y++;

}

void operator--(int)

{

x--; y--;

}

private:

int x;

int y;

};

Point::Point(int dx,int dy)

{

x=dx;

y=dy;

}

Point::~Point(){

}

int main(void)

{

int n,m;

cout<<"Please input the coordinate a(\* \*):";

cin>>n>>m;

Point yu(n,m);

--yu;

cout<<"--a=";

yu.showPoint();

yu++;

cout<<"a++=";

yu.showPoint();

yu--;

cout<<"a--=";

yu.showPoint();

++yu;

cout<<"++a=";

yu.showPoint();

system("pause");

return 0;

}

**2．实验总结：**c++的多态性也是一大特色，对于运算符的重载极大的方便我们的代码，需要注意的是对于前置++/--和后置++/--他们重载时在参数上有一点不同。

**实验六 流式IO**

**1．源代码**

/\*六 1 流式IO（一）\*/

#include<stdlib.h>

#include<iomanip>

#include<iostream>

#include<fstream>

using namespace std;

int main(void)

{

ofstream MyFile;

int a=-225;//整数

unsigned int b=324;//无符号整型

char c[]="Hello world!";//字符串

long int d=124342212;//长整型

float e=993.34;//浮点型

MyFile.open("1.txt");//使用ofstream 向一个文本文件中输出各种类型的数据

if(MyFile.fail())

cout<<"文件打开失败。\n";

MyFile<<"整数"<<setw(10)<<a<<endl;

MyFile<<"无符号整型"<<setw(10)<<b<<endl;

MyFile<<"字符串"<<setw(10)<<c<<endl;

MyFile<<"长整型"<<setw(10)<<d<<endl;

MyFile<<"浮点型"<<setw(10)<<e<<endl;

MyFile<<"十进制 ："<<dec<<b<<endl;

MyFile<<"八进制 ："<<oct<<b<<endl;

MyFile<<"十六进制："<<hex<<b<<endl;

MyFile<<e<<"保留四位有效数字"<<setprecision(4)<<e;//控制精度

MyFile.close();

system("pause");

return 0;

}

/\*六 3 流式IO（三）\*/

#include<iostream>

#include<fstream>

#include<string.h>

#define SIZE 20

using namespace std;

int main(void)

{

char s1[SIZE],s2[SIZE],temp;

ifstream file\_1;

ofstream file\_2;

int i=1,j;

cout<<"请输入一个文本文件名:";//输入一个文本文件名

cin>>s1;

strcat(s1,".txt");

file\_1.open(s1,ios::binary);//打开文件

if(file\_1.fail())

cout<<"文件打开失败。\n";

cout<<"请输入一个目标文件名:";

cin>>s2;

strcat(s2,".txt");

file\_2.open(s2,ios::binary);

//该文件的每一行前面加上一个行号，保存在另外一个文本文件中

if(file\_2.fail())

cout<<"文件打开失败。\n";

file\_2<<i<<'.';

i++;

temp=file\_1.get();

while(temp!=EOF)

{

if(temp=='\n')

{

file\_2<<temp;

file\_2<<i<<'.';

i++;

}

else

file\_2<<temp;

temp=file\_1.get();

}

file\_1.close();

file\_2.close();

system("pause");

return 0;

}

**2．实验总结：**关于文件的操作c++ 与c有很大不同之处，ofstream ifstream和fstream的不同使用，也主要是一些对文件格式控制语法的熟悉

**实验七 C＋＋程序设计应用**

**1．源代码**

**/\*电话本\*/**

**#include<iostream>**

**#include<stdlib.h>**

**#include<string.h>**

**#include<fstream>**

**#include<iomanip>**

**#define SIZE 100**

**using namespace std;**

**struct Contact{**

**char name[10];//姓名**

**char num[12];//电话**

**};**

**typedef struct Contact CON;**

**void WelcomeInterface(void);//界面**

**void addNumber(void);//增加号码**

**int seekNum(string na,int ifp);//查找**

**void Showall(void);//打印所有的**

**int deleteNum();//删除联系人**

**void GuanLi(void);//电话本可以保存在指定文件中，电话可被从指定文件中读入到内存**

**int main(void)**

**{**

**char choice;//保存用户的选项**

**string na;//想用户需要查找的名字**

**while(1)**

**{**

**system("cls");**

**WelcomeInterface();**

**cin>>choice;**

**if(choice=='0')//"0"的时候退出程序**

**break;**

**switch(choice)**

**{**

**//查找**

**case '1':**

**system("cls");**

**cout<<"Please input the name:";**

**cin>>na;**

**if(seekNum(na,1)!=1)**

**cout<<"There is no such a person.\n";**

**break;**

**//添加**

**case '2':**

**addNumber();**

**cout<<"Added successfully.\n";**

**break;**

**case '3'://删除**

**if(deleteNum()==1)**

**cout<<"It has been done.\n";**

**else**

**cout<<"There may not exists the person.\n";**

**break;**

**case '4'://打印所有存在的**

**Showall();**

**break;**

**case '5'://清空所有记录**

**{**

**fstream file;**

**file.open("data.dat",ios::binary | ios::out );**

**file.close();**

**if(file.fail())**

**cout<<"文件打开失败。\n";**

**}**

**break;**

**case '6'://电话本可以保存在指定文件中，电话可被从指定文件中读入到内存**

**GuanLi();**

**break;**

**default:**

**cout<<"Error input.Please choose from 0-6.\n";**

**fflush(stdin);**

**}**

**system("pause");**

**}**

**system("cls");**

**cout<<"Hope to serve you next time.\n";**

**system("pause");**

**return 0;**

**}**

**int deleteNum()**

**{**

**fstream file;**

**CON person,blankP={"",""};**

**string na;**

**CON people[SIZE];**

**int i=0,j=0;**

**system("cls");**

**cout<<"Please input the person you want to delete:";**

**cin>>na;**

**if(seekNum(na,0)==0)//先查找一下**

**return 0;**

**file.open("data.dat",ios::binary | ios::in);**

**if(file.fail())**

**cout<<"文件打开失败。\n";**

**else**

**{**

**while(file.peek()!=EOF)**

**{**

**file.read((char\*)&person,sizeof(CON));**

**if(string(na)!=string(person.name))**

**{**

**strcpy(people[i].name,person.name);**

**strcpy(people[i++].num,person.num);**

**}**

**}**

**file.close();**

**file.open("data.dat",ios::binary | ios::out);**

**if(file.fail())**

**cout<<"文件打开失败。\n";**

**else**

**{**

**while(j<i)**

**file.write((char\*)&people[j++],sizeof(CON));**

**file.close();**

**return 1;**

**}**

**return 0;**

**}**

**return 0;**

**}**

**int seekNum(string na,int ifp)**

**{**

**CON person;**

**ifstream file;**

**file.open("data.dat",ios::binary);**

**if(file.fail())**

**cout<<"文件打开失败。\n";**

**else**

**{**

**if(file)**

**{**

**while(file.peek()!=EOF)**

**{**

**file.read((char\*)&person,sizeof(CON));**

**if(string(na)==string(person.name))**

**{**

**if(ifp==1)**

**cout<<"The phone number is "<<person.num<<endl;**

**file.close();**

**return 1;**

**}**

**}**

**}**

**}**

**fflush(stdin);**

**return 0;**

**}**

**void addNumber(void)**

**{**

**CON person={"",""},temp;**

**fstream file;**

**int n;//本次添加的人数**

**system("cls");**

**file.open("data.dat",ios::binary | ios::out | ios::app);**

**if(file.fail())**

**cout<<"文件打开失败。\n";**

**else**

**{**

**cout<<"How many people you want to add:";**

**cin>>n;**

**while(n)**

**{**

**cout<<"Please input the name:";**

**cin>>person.name;**

**if(seekNum(person.name,0)!=1)//先查找若不存在则添加**

**{**

**n--;**

**cout<<"Please input the number:";**

**cin>>person.num;**

**file.write((char\*)&person,sizeof(CON));**

**}**

**else//提示已存在**

**cout<<"The people already exists!\n";**

**}**

**file.close();**

**}**

**}**

**void WelcomeInterface(void)**

**{**

**cout<<" Welcome to our phone book system~\n";**

**cout<<" ---------------------------------------------\n";**

**cout<<" | 1.Look up a number |\n";**

**cout<<" | 2.Add a number |\n";**

**cout<<" | 3.Delete a number |\n";**

**cout<<" | 4.Print all the number |\n";**

**cout<<" | 5.Empty the file(Carefully) |\n";**

**cout<<" | 6.Management |\n";**

**cout<<" | 0.Exit |\n";**

**cout<<" ---------------------------------------------\n";**

**cout<<"Please input your choice:";**

**}**

**void Showall(void)//打印当前所有的信息**

**{**

**CON person;**

**ifstream file;**

**system("cls");**

**file.open("data.dat",ios::binary);//打开文件**

**if(file.fail())**

**cout<<"文件打开失败。\n";**

**else**

**{**

**cout<<setw(6)<<"name"<<setw(15)<<"number"<<endl;**

**//输出所有记录**

**while(file.peek()!=EOF)**

**{**

**file.read((char\*)&person,sizeof(CON));**

**cout<<setw(6)<<person.name<<setw(15)<<person.num<<endl;**

**}**

**file.close();**

**}**

**}**

**void GuanLi(void)**

**{**

**ofstream f;**

**fstream file;**

**CON people[SIZE];**

**char filename[SIZE];**

**int i=0,j=0;**

**system("cls");**

**//将文件里的记录读入内存中**

**file.open("data.dat",ios::binary | ios::in);**

**if(file.fail())**

**cout<<"文件打开失败。\n";**

**else**

**{**

**while(file.peek()!=EOF)**

**file.read((char\*)&people[i++],sizeof(CON));**

**file.close();**

**//将记录输出到指定文件**

**cout<<"Now the information is saved in the internal storage memory.\n";**

**cout<<"Input the destination file's name:";**

**cin>> filename; strcat(filename,".txt");//读入文指定件名**

**fflush(stdin);**

**f.open(filename);**

**while(j<i)**

**{**

**f.write((char\*)&people[j],sizeof(CON));**

**f.write("\n",1);**

**j++;**

**}**

**f.close();**

**cout<<"Now the file is saved.\n";**

**}**

**}**

**2．实验总结：**电话本的实验主要训练了文件的打开、写入、输出，还有字符串的比较等操作。我在写的时候设计了一个界面，每次一个操作进行完之后清屏，每次读入用户的输入进行合法性的检测，在增加和删除时先查找原来的记录中是否存在。另外我增加了一个打印所有记录的功能，方便测试。