# 北京邮电大学软件学院

# 2015-2016学年第1学期实验报告

**课程名称： 算法与数据结构**

**实验名称： 线性表**

**实验完成人：**

**姓名：**\_\_\_刘广田\_\_\_\_**学号：**\_\_\_2018212046\_\_**成绩：**\_\_\_\_\_\_\_\_

**指导教师：**\_\_\_\_\_\_贾红娓 李璐璐\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2019 年 10 月 15 日**

1. **实验目的**

本次实验的主要目的在于熟悉线性表的基本运算在两种存储结构上的实现，其中以熟悉各种链表的操作为侧重点。同时，通过本次实验帮助学生复习高级语言的使用方法。

1. **实验内容**
2. 城市链表
3. 约瑟夫环
4. **实验环境**

Visual Studio 2019

1. **实验结果**
2. 城市链表：

实现了用户对于城市链表的各种操作，包括向已经创建好的城市链表插入城市、删除城市、查找城市坐标、打印当前链表中所有城市的信息和返回一定范围内的所有城市

1. 约瑟夫环

实现了根据用户输入的初始报数上限值、总人数以及每个人的密码模拟出约瑟夫问题的处理过程

1. **附录**

（附上实验文档，如：问题分析、设计方案、算法、设计图、程序、仿真结果、运行结果、调试心得等，具体内容根据实验要求来定）

问题分析与设计方案：

1）要求使用带有头结点单链表实现，首先创建城市类，城市类中有另外一个点类，用来表示城市的坐标，城市类中存储城市的名称，坐标，然后创建城市链表类，里面存储各个城市，并实现对链表中城市的各种操作。在main函数中首先创建一个城市链表，提示用户选择操作，使用链表的所有功能。

2）要求实现使用单向循环链表实现约瑟夫环，首先创建Person类，存储每个人的序号和密码，然后实现PersonList类，实现单向循环链表以及其中的插入删除操作并实现约瑟夫环中的算法。

算法：

1. 主要是链表的各种操作，并无特殊算法
2. Joseph算法：

使用循环链表存储每个人的序号和密码信息，由用户指定初始报数上限和每个人的序号和密码。从第一个人开始遍历循环链表，这期间i的值一直增长，直到i等于报数上限，报到报数上限的人出列即在链表中删除这个人，并输出这个人的序号信息，删除前将其手中密码当做新的报数上限，再从这个人之后开始报数，直到链表为空。

程序：

1）城市链表

Point.h

#pragma once

#include <iostream>

class Point {

private:

double x;

double y;

public:

Point();

Point(double x, double y);

double getX();

double getY();

std::string toString();

};

Point.cpp

#include "Point.h"

#include <sstream>

Point::Point() {

x = 0.0;

y = 0.0;

}

Point::Point(double x, double y) {

this->x = x;

this->y = y;

}

double Point::getX() { return x; }

double Point::getY() { return y; }

std::string Point::toString() {

std::string s;

s = "(" + std::to\_string(this->getX()) + ", " + std::to\_string(this->getY()) + ")";

return s;

}

City.h

#pragma once

#include <string>

#include <iostream>

#include <math.h>

#include "Point.h"

class City {

private:

std::string name;

Point p;

City\* next;

public:

City();

City(std::string nane, Point p);

std::string getName();

Point getPoint();

void setName(std::string name);

void setPoint(Point p);

void setNext(City\* c);

City\* getNext();

};

City.cpp

#include "City.h"

City::City() {

name = "Beijing";

p = Point();

next = nullptr;

}

City::City(std::string name, Point p) {

this->name = name;

this->p = p;

next = nullptr;

}

std::string City::getName() { return name; }

Point City::getPoint() { return p; }

void City::setName(std::string name) { this->name = name; }

void City::setPoint(Point p) { this->p = p; }

void City::setNext(City\* c) { next = c; }

City\* City::getNext() { return next; }

CityList.h

#pragma once

#include "City.h"

class CityList {

private:

City\* head;

public:

CityList();

void refresh();

void insertCity(int pos);

void deleteCity(int pos);

std::string find(std::string name);

std::string findAll(Point p, double distance);

};

CityList.cpp

#include "CityList.h"

CityList::CityList() {

head = new City;

if (nullptr == head)

std::cout << "new error!" << std::endl;

}

void CityList::refresh() {

City\* pCity = head;

int cnt = 1;

while (pCity != nullptr) {

if (cnt == 1) {

std::cout << "列表中的城市有：" << std::endl;

std::cout << cnt << ": " << pCity->getName() << " " << "这座城市的坐标是：" << pCity->getPoint().toString() << "." << std::endl;

pCity = pCity->getNext();

}

else {

std::cout << cnt << ": " << pCity->getName() << " " << "这座城市的坐标是：" << pCity->getPoint().toString() << "." << std::endl;

pCity = pCity->getNext();

}

cnt++;

}

}

void CityList::insertCity(int pos) {

City\* pCity = head;

std::string name;

double x, y;

std::cout << "请输入城市的名称：";

std::cin >> name;

std::cout << "请输入城市的x坐标：";

std::cin >> x;

std::cout << "请输入城市的y坐标：";

std::cin >> y;

Point p(x, y);

int i = 0;

while (pCity->getNext() != nullptr && i < pos - 1) {

pCity = pCity->getNext();

i++;

}

City\* tmp = new City(name, p);

tmp->setNext(pCity->getNext());

pCity->setNext(tmp);

}

void CityList::deleteCity(int pos) {

City\* pCity = head;

int i = 1;

while (i < pos - 1) {

pCity = pCity->getNext();

i++;

}

City\* tmp = pCity->getNext();

pCity->setNext(tmp->getNext());

delete(tmp);

tmp = nullptr;

}

std::string CityList::find(std::string name) {

City\* pCity = head;

while (pCity->getName() != name) {

pCity = pCity->getNext();

if (pCity == nullptr)

return "列表中不存在这座城市";

}

return (pCity->getPoint().toString());

}

std::string CityList::findAll(Point p, double distance) {

City\* pCity = head;

std::string cities = "";

while (pCity != nullptr) {

double z = pow(pow(pCity->getPoint().getX() - p.getX(), 2) + pow((pCity->getPoint().getY() - p.getY()), 2), 1.0 / 2.0);

if (z <= distance)

cities += pCity->getName() + ", ";

pCity = pCity->getNext();

}

return cities;

}

main.cpp

#include <iostream>

#include "Point.h"

#include "City.h"

#include "CityList.h"

int main(void) {

std::cout << "正在初始化……" << std::endl;

CityList myList;

std::cout << "初始化完毕！" << std::endl;

std::cout << "请输入'Y'或'y'表示开始";

char character;

std::cin >> character;

while (character == 'Y' || character == 'y') {

std::cout << "请输入你想选择的选项：" << std::endl;

std::cout << "1.插入 2.删除 3.显示已有的城市\n"

<< "4.查看列表中一座城市的坐标 " << " 5.寻找指定范围内的所有城市" << std::endl;

int x;

std::cin >> x;

if (1 == x) {

std::cout << "请输入要填加到的位置：";

int pos;

std::cin >> pos;

myList.insertCity(pos);

}

else if (2 == x) {

std::cout << "请输入要删除的位置：";

int pos;

std::cin >> pos;

myList.deleteCity(pos);

}

else if (3 == x) {

myList.refresh();

}

else if (4 == x) {

std::cout << "请输入城市名：";

std::string newName;

std::cin >> newName;

std::cout << myList.find(newName) << std::endl;;

}

else if (5 == x) {

double x, y, distance;

std::cout << "请输入中心x坐标：";

std::cin >> x;

std::cout << "请输入中心y坐标：";

std::cin >> y;

Point point(x, y);

std::cout << "请输入距离：";

std::cin >> distance;

std::cout << myList.findAll(point, distance) << std::endl;

}

else

std::cout << "选项无效" << std::endl;

std::cout << "请输入'Y'或'y'表示继续：";

char character;

std::cin >> character;

if (character != 'y' && character != 'Y')

break;

else

continue;

}

return 0;

}

2）

Person.h

#pragma once

#include <iostream>

class Person {

private:

int id;

int password;

Person\* next;

public:

Person();

int getId();

int getPassword();

Person\* getNext();

void setPassword(int password);

void setNext(Person\* p);

void setId(int id);

};

Person.cpp

#include "Person.h"

Person::Person() {

id = 0;

password = 6;

next = nullptr;

}

int Person::getId() { return id; }

int Person::getPassword() { return password; }

Person\* Person::getNext() { return next; }

void Person::setPassword(int password) { this->password = password; }

void Person::setNext(Person\* p) { next = p; }

void Person::setId(int id) { this->id = id; }

PersonList.h

#include "PersonList.h"

PersonList::PersonList() {

head = new Person;

head->setNext(head);

}

PersonList::PersonList(int a[], int n) {

head = new Person;

head->setNext(head);

Person\* pre = head;

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

Person\* p = new Person;

p->setPassword(a[i]);

p->setId(i + 1);

pre->setNext(p);

pre = p;

}

pre->setNext(head);

}

void PersonList::show(int n) {

Person\* p = head->getNext();

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

std::cout << "序号：" << p->getId() << "手中的密码为：" << p->getPassword() << std::endl;

p = p->getNext();

}

}

void PersonList::Joeph(int n) {

int m = n;

int order = 0, cnt = 1;

Person\* pre = head;

Person\* p = head->getNext();

while (pre != p->getNext()) {

if (cnt == m && p != head) {

Person\* x = new Person;

x = p;

pre->setNext(p->getNext());

p = p->getNext();

order++;

std::cout << "第" << order << "个人出列，序号为" << x->getId() << std::endl;

m = x->getPassword();

delete(x);

cnt = 1;

}

else {

if (p != head)

cnt++;

pre = p;

p = p->getNext();

}

}

std::cout << "第" << ++order << "个人出列，序号为" << p->getId() << std::endl;

delete(p);

delete(pre);

}

main.cpp

#include "PersonList.h"

int main(void) {

std::cout << "请输入初始报数上限值：";

int m;

std::cin >> m;

std::cout << "请输入总人数（小于等于30）：";

int num;

std::cin >> num;

int\* p = new int[num];

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

std::cout << "请输入第" << (i + 1) << "个人的密码：";

int x;

std::cin >> x;

p[i] = x;

}

PersonList list(p, num);

list.Joeph(m);

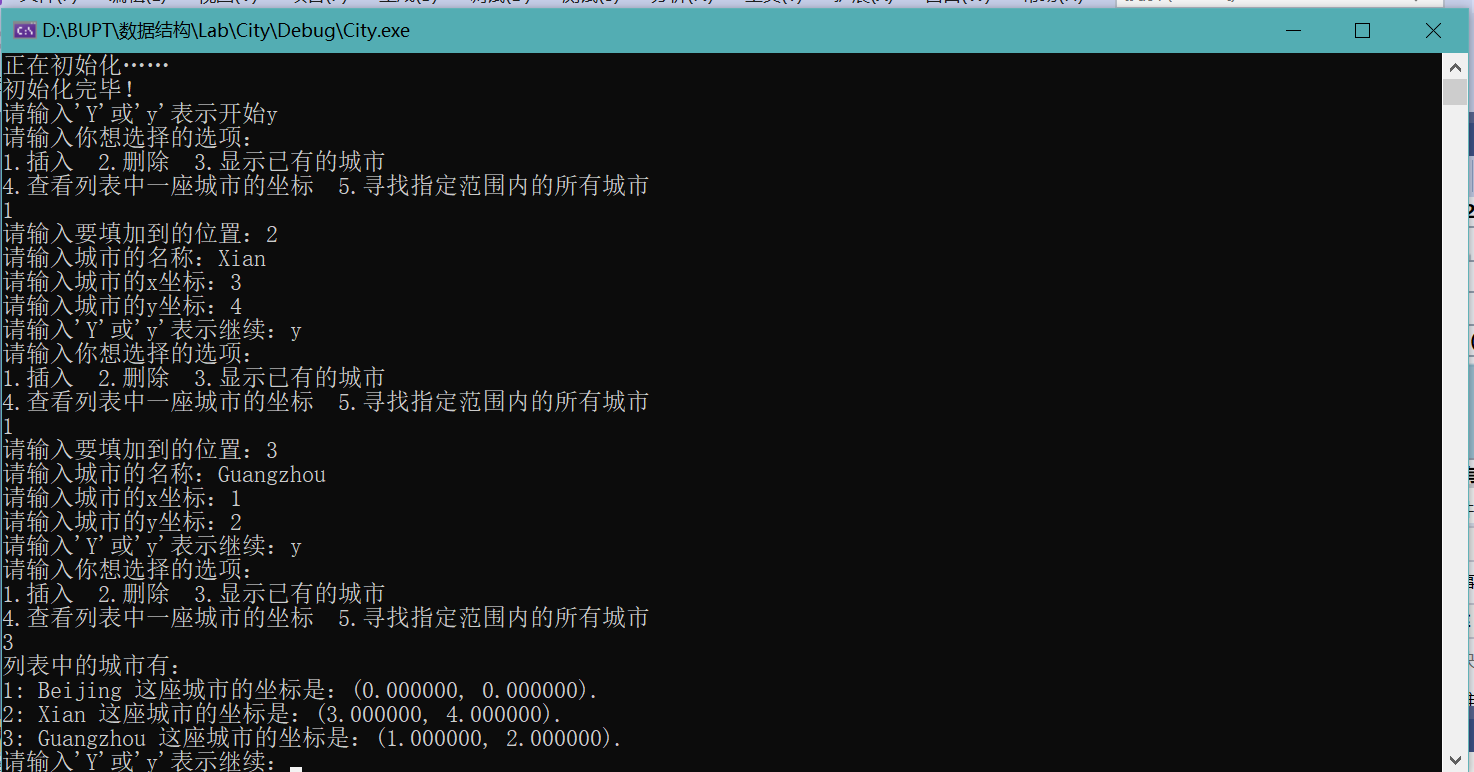
delete[] p;

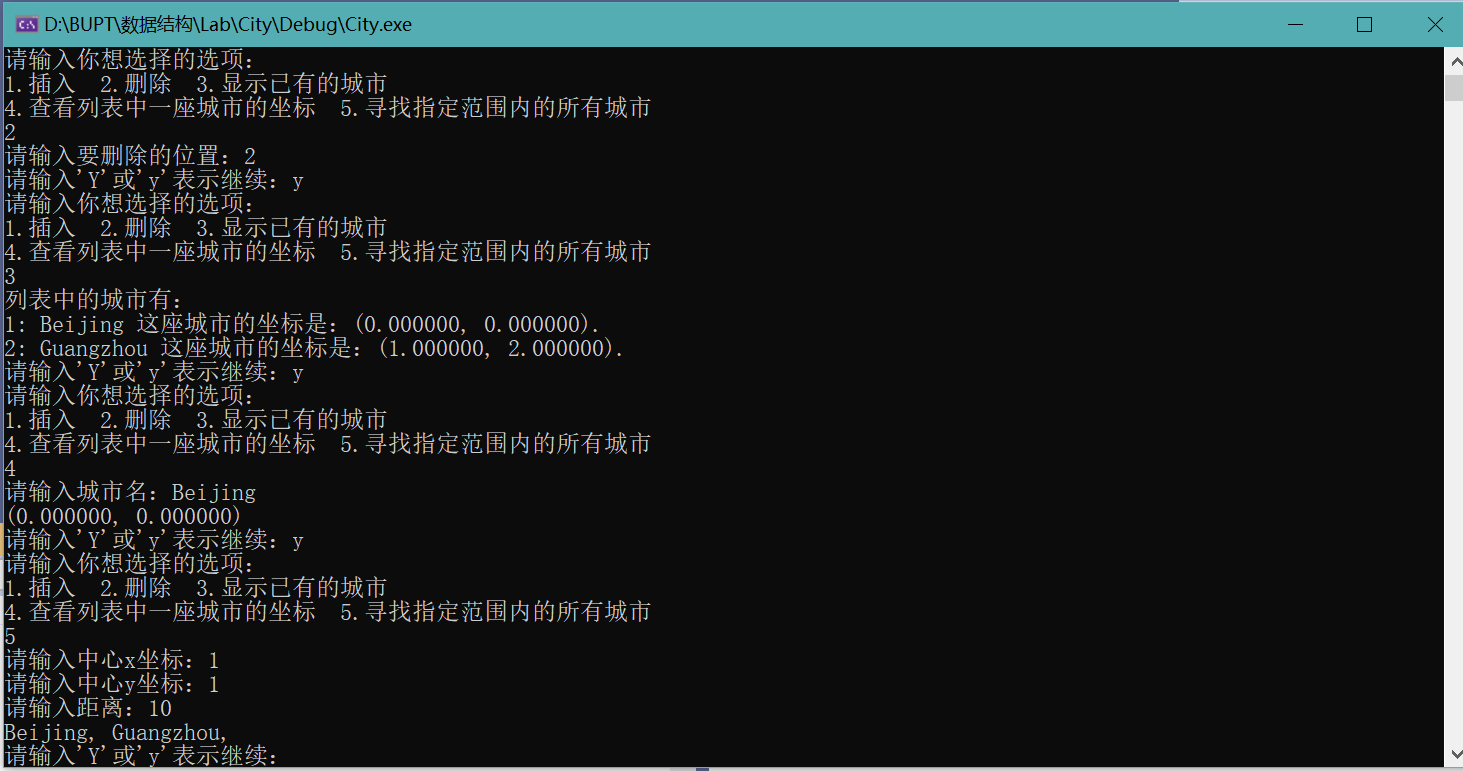
return 0;

}

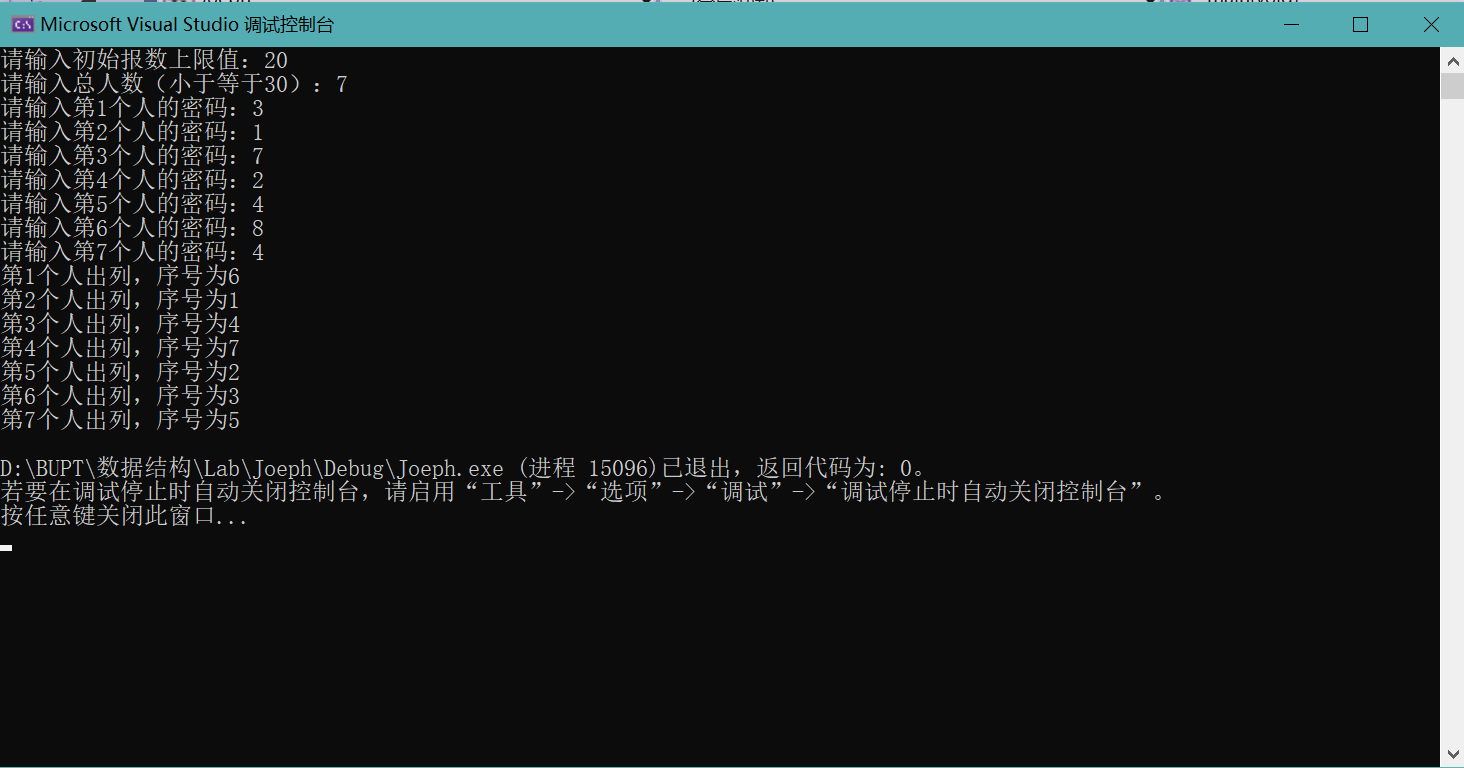
运行结果：

1）





2）



调试心得：

1. 在对链表的操作中，next指针很关键，错误常发生在next指针为NULL的情况下。
2. 在链表中要删除某个节点并要输出该节点的相关信息时，要先将该节点的信息暂存到一个新节点中，输出完毕再将该新节点删除。