**暨南大学本科实验报告专用纸**

课程名称 C++程序设计 成绩评定

实验项目名称 文本查询编程实验 指导教师 王勇杰

实验项目编号 8 实验项目类型 编程型 实验地点 C105

学生姓名 李炜鹏 学号 2017052544

学院 智能科学与工程学院 系 计算机 专业 信息安全

实验时间2019年12月10日 午～12月13日 午 温度 ℃湿度

1. **实验目的**
2. 理解并实现文本查询程序（教材P430-P435）
3. 熟悉标准库的使用
4. 巩固前面几个实验所学的知识：面向对象编程、智能指针
5. 实验要求：阅读课本第12章的“12.3 使用标准库：文本查询程序”内容， 试调试运行实验八中的源代码
6. 实验报告要求：
   1. 实验目的
   2. 实验原理
   3. 注释必要的代码
   4. 实验结果的截图
   5. 实验评估
7. **实验原理**
8. TextQuery类的设计：
   1. 数据成员：一个指向动态分配的vector（保存输入文件）的shared\_ptr和一个string到shared\_ptr<set>的map。Map将文件中每个单词关联到一个动态分配的set上，而此set保存了该单词所出现的行号
   2. TextQuery构造函数：读取输入文件并建立单词到行号的映射（详见教材P433）
   3. query函数：作为最重要的函数之一，query接收一个string参数，即查询单词，query用它来在map中定位对应的行号set。如果找到了这个string，query函数构造一个QueryResult，保存给定string、TextQuery的file成员以及从wm提取的set。
   4. 本代码新增的函数：

display\_map：调试例程，将打印查找映射的内容

cleanup\_str(const std::string&)：规范化文本，删除标点符号并使所有内容变为小写

1. QueryResult类：
   1. 数据成员：一个string，保存查询单词；一个shared\_ptr，指向保存输入文件的vector；一个shared\_ptr，指向保存单词出现行号的set。
   2. 构造函数：初始化三个数据成员
   3. 本代码新增的函数：

std::set<line\_no>::size\_type size() const { return lines->size(); }

line\_it begin() const { return lines->begin(); }

line\_it end() const { return lines->end(); }

std::shared\_ptr<std::vector<std::string> > get\_file() { return file; }

1. 类外函数print（QueryResult类的友元函数）：打印结果

类外函数make\_plural：如果单词数大于1，则返回复数形式（加上s/es之类的）

1. **主要仪器设备**

**仪器：**计算机

**实验环境：**Visual Studio 2017或Dev-C++

1. **源程序**

**make\_plural.h**

#include <cstddef>

using std::size\_t;

#include <string>

using std::string;

#include <iostream>

using std::cout; using std::endl;

#ifndef MAKE\_PLURAL\_H

#define MAKE\_PLURAL\_H

//如果ctr大于1，则返回单词的复数形式

inline string make\_plural(size\_t ctr, const string &word, const string &ending)

{

return (ctr > 1) ? word + ending : word;

}

#endif

**TextQuery.h**

#ifndef TEXTQUERY\_H

#define TEXTQUERY\_H

#include <memory>

#include <string>

#include <vector>

#include <map>

#include <set>

#include <fstream>

#include "QueryResult.h"

class QueryResult; //为了定义函数query的返回类型，这里必须提前声明QueryResult类

class TextQuery {

public:

typedef std::vector<std::string>::size\_type line\_no;//为std::vector<std::string>::size\_type声明别名line\_no

TextQuery(std::ifstream&);

QueryResult query(const std::string&) const;

void display\_map();//调试例程，将打印查找映射的内容

private:

std::shared\_ptr<std::vector<std::string> > file; //输入文件

std::map<std::string, std::shared\_ptr<std::set<line\_no> > > wm; //每个单词到它所在的行号的集合的映射

static std::string cleanup\_str(const std::string&);//规范化文本：删除标点符号并使所有内容变为小写

};

#endif

**QueryResult.h**

#ifndef QUERYRESULT\_H

#define QUERYRESULT\_H

#include <memory>

#include <string>

#include <vector>

#include <set>

#include <iostream>

class QueryResult {

friend std::ostream& print(std::ostream&, const QueryResult&);//print作为此类的友元函数，用于打印结果

public:

typedef std::vector<std::string>::size\_type line\_no;

typedef std::set<line\_no>::const\_iterator line\_it;

QueryResult(std::string s, std::shared\_ptr<std::set<line\_no> > p, std::shared\_ptr<std::vector<std::string> > f) :sought(s), lines(p), file(f) { }//构造函数

std::set<line\_no>::size\_type size() const { return lines->size(); }

line\_it begin() const { return lines->begin(); }

line\_it end() const { return lines->end(); }

std::shared\_ptr<std::vector<std::string> > get\_file() { return file; }

private:

std::string sought; //查询的单词

std::shared\_ptr<std::set<line\_no> > lines; //出现的行号

std::shared\_ptr<std::vector<std::string> > file; //输入文件

};

std::ostream &print(std::ostream&, const QueryResult&);

#endif

**TextQuery.cpp**

#include "TextQuery.h"

#include "make\_plural.h"

#include <cstddef>

#include <memory>

#include <sstream>

#include <string>

#include <vector>

#include <map>

#include <set>

#include <iostream>

#include <fstream>

#include <cctype>

#include <cstring>

#include <utility>

using std::size\_t;

using std::shared\_ptr;

using std::istringstream;

using std::string;

using std::getline;

using std::vector;

using std::map;

using std::set;

using std::cerr;

using std::cout;

using std::cin;

using std::ostream;

using std::endl;

using std::ifstream;

using std::ispunct;

using std::tolower;

using std::strlen;

using std::pair;

//因为我们不能使用auto关键字，所以我们将使用typedef来定义别名简化我们的代码

//TextQuery对象中查找映射的类型wmType

typedef map<string, shared\_ptr<set<TextQuery::line\_no> > > wmType;

//wmType的迭代器wmIter

typedef wmType::const\_iterator wmIter;

//包含行号的集合的共享智能指针类型lineType

typedef shared\_ptr<set<TextQuery::line\_no> > lineType;

//set<TextQuery::line\_no>的迭代器lineIter

typedef set<TextQuery::line\_no>::const\_iterator lineIter;

//读取输入文件并建立单词到行号的映射

TextQuery::TextQuery(ifstream &is) : file(new vector<string>)

{

string text;

while (getline(is, text)) { //文件中的每一行读取到text中

file->push\_back(text); //保存此文本到共享智能指针file指向的内存中

int n = file->size() - 1; //当前行号

istringstream line(text); //将行文本分解成单个单词

string word;

while (line >> word) { //行中的每个单词逐个输入到word中

word = cleanup\_str(word);

//如果单词不在wm中，以之为下标在wm中添加一项

lineType &lines = wm[word]; //lines是一个shared\_ptr

if (!lines) //第一次遇到这个单词时，此指针为空

lines.reset(new set<line\_no>); //分配一个新的set

lines->insert(n); //将此行号插入到set中

}

}

}

QueryResult TextQuery::query(const string &sought) const

{

//如果未找到sought，返回一个执行此set的指针

static lineType nodata(new set<line\_no>);

//使用find而不是下标运算符来查找单词，避免将单词添加到wm中

wmIter loc = wm.find(cleanup\_str(sought));

if (loc == wm.end())

return QueryResult(sought, nodata, file); //未找到

else

return QueryResult(sought, loc->second, file);

}

ostream &print(ostream & os, const QueryResult &qr)

{

//如果找到了单词，打印出现次数和所有出现的位置

os << qr.sought << " occurs " << qr.lines->size() << " " << make\_plural(qr.lines->size(), "time", "s") << endl;

//打印单词出现的每一行

for (lineIter num = qr.lines->begin(); num != qr.lines->end(); ++num)

//避免行号从0开始给用户带困惑

os << "\t(line " << \*num + 1 << ") " << \*(qr.file->begin() + \*num) << endl;

return os;

}

//本书未涵盖cleanup\_str：删除标点符号并将所有文本转换为小写，以便查询以不区分大小写的方式运行

string TextQuery::cleanup\_str(const string &word)

{

string ret;

for (string::const\_iterator it = word.begin(); it != word.end(); ++it) {

if (!ispunct(\*it))

ret += tolower(\*it);

}

return ret;

}

//调试例程，将打印查找映射的内容

void TextQuery::display\_map()

{

wmIter iter = wm.begin(), iter\_end = wm.end();

//遍历整个map

for (; iter != iter\_end; ++iter)

{

cout << "word: " << iter->first << " {";

//提取位置向量作为const引用，以防止对它进行了修改（事实上并没有使用const？这里标注出错了？）

lineType text\_locs = iter->second;

lineIter loc\_iter = text\_locs->begin(), loc\_iter\_end = text\_locs->end();

//打印该单词的所有行号

while (loc\_iter != loc\_iter\_end)

{

cout << \*loc\_iter;

if (++loc\_iter != loc\_iter\_end)

cout << ", ";

}

cout << "}\n"; // end list of output this word

}

cout << endl; //完成对整个map的输出

}

**querymain.cpp**

#include <string>

using std::string;

#include <fstream>

using std::ifstream;

#include <iostream>

using std::cin; using std::cout; using std::cerr;

using std::endl;

#include <cstdlib> // EXIT\_FAILURE 的头文件

#include "TextQuery.h"

#include "make\_plural.h"

void runQueries(ifstream &infile)

{

//infile是一个ifstream，它是我们要查询的文件

TextQuery tq(infile); // 存储文件并创建map

//与用户进行交互：提示输入单词以查找并打印结果

while (true) {

cout << "enter word to look for, or q to quit: ";

string s;

//如果检测到文件结尾标识符或输入了“ q”，则停止

if (!(cin >> s) || s == "q") break;

//运行查询并打印结果

print(cout, tq.query(s)) << endl;

}

}

//程序采用单个参数指定要查询的文件

int main(int argc, char \*\*argv)

{

//打开用户要查询单词的文件

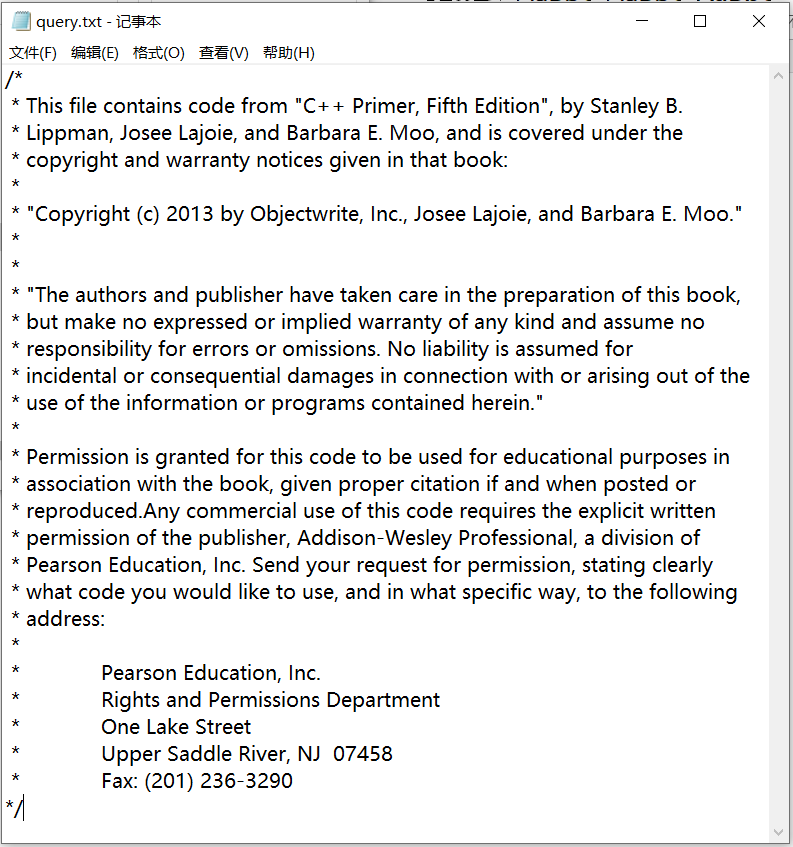
ifstream infile("query.txt");

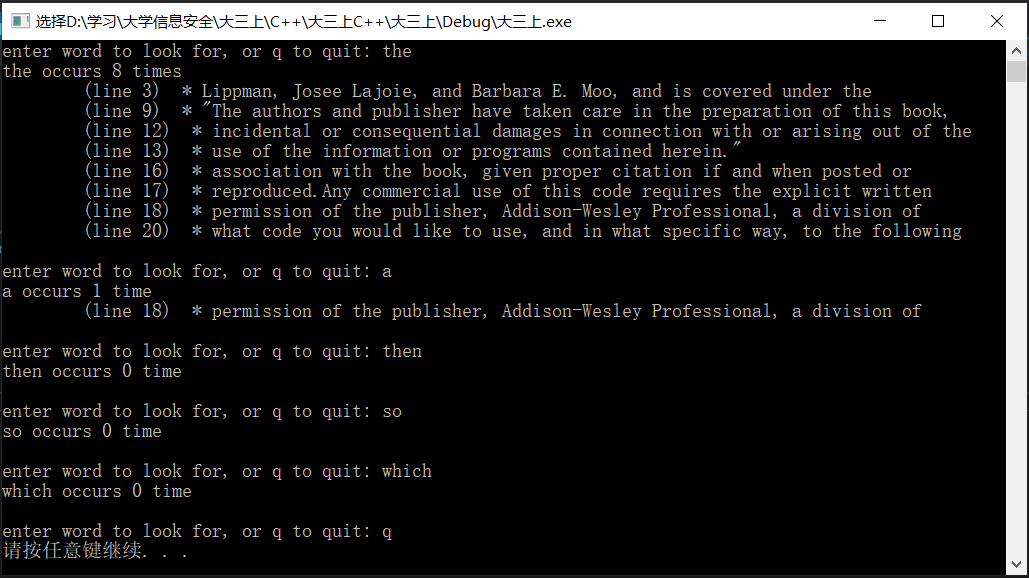
runQueries(infile);

system("pause");

return 0;

}





先后输入the、a、then、so、which、q，运行结果符合预期

1. **实验评估**

本实验是教材第12章（动态内存）的综合性实验，涉及的知识点比较多：C++面向对象编程、C++标准模板库、动态内存分配与智能指针，文件的输入输出，难度较大。通过本实验，我理解了文本查询程序这个小项目，并且上机实现了项目的运行，经过思考，我领会到类的设计应该是怎样构思与实现的，类之间是如何通过共享智能指针共享数据的，各模块之间是如何设计接口实现连接的。本实验仅仅是一个新知识领域的开端，洗完通过以后的练习进一步加强。