**实 验 报 告**

**课程名称 数据结构（c）（英文）**

**实验项目** 串的应用

**实验仪器** 计算机

**系 别** 计算机学院学院

**专 业** 计算机大类

**班级/学号** 计算机类1801 2018011137

**学生姓名**  武文斌

**实验日期**  2019.12.01

**成 绩**

**指导教师**  蔡英

实验四（2）

由于对此实验非常感兴趣，便在老师没有要求的情况下另做了此实验，算作实验四的部分内容进行呈现。

1. 实验目的
2. 了解和运用KMP算法以及next函数
3. 实验内容

文学助手

实验内容描述：

文学研究人员需要统计某篇英文文章中某些词汇的出现次数和位置。试写一个实现这一目标的文字统计系统，称“文学研究助手”。

【基本要求】

将文章存于一个文本文件中。待统计的词汇集合要一次输入完毕，即统计工作必须在程序的一次运行之后就全部完成。程序的输出结果是每个词的出现次数和出现位置所在的行号，格式自行设计。

1. 设计思路

// 利用模式串T的next函数求T在主串S中第pos个字符之后的位置的KMP算法。  
 // 其中,T非空,1≤pos≤StrLength(S)  
 //Next(char \*p,int n[])函数计算模式串各值的部分匹配串长度。  
 //根据Next函数，用KMP\_match(char \*T,char \*P)函数计算匹配位置。

1. 源代码

#include<iostream>  
#include<cstdlib>  
#include<fstream>  
#include <c++/cstring>  
  
using namespace std;  
  
char \*FileRead(char ch[])  
//读文件函数  
{  
 char \*c = (char \*)calloc(2000, sizeof(char));  
 //calloc分配出来的内存，返回值也是void \* 但是分配的内存是被清空过的，更安全  
 int i = 0;  
 ifstream in(ch);  
 if (!in) { cout << "Unable to open file" << ch << endl; return 0; }  
 in >> noskipws; //读文件不跳过空格  
 while (!in.eof()) //eof 判断是否读到文件尾  
 {  
 in >> c[i++];  
 }  
 in.close();  
 return c;  
}  
  
void GetNext(char t[], int next[])  
{// 求模式串T的next函数值并存入数组next  
 int j = 0, k = -1;  
 int n = strlen(t);  
 next[j] = -1;  
 while (j<n)  
 {  
 if (k == -1 || t[j] == t[k])  
 {  
 j++; k++; next[j] = k;  
 }  
 else k = next[k];  
 }  
}  
  
int IndexKMP(char s[], char t[], int next[])  
{// 利用模式串T的next函数求T在主串S中第pos个字符之后的位置的KMP算法。  
 // 其中,T非空,1≤pos≤StrLength(S)  
 //Next(char \*p,int n[])函数计算模式串各值的部分匹配串长度。  
 //根据Next函数，用KMP\_match(char \*T,char \*P)函数计算匹配位置。  
 int i, j;  
 i =0; j = 0;  
 int count = 0; //计数器  
 int m = strlen(s), n = strlen(t);  
 while (i<m && j<n)  
 {  
 if (j == -1 || s[i] == t[j])  
 {  
 i++; j++;  
 } // 继续比较后继字符  
 else j = next[j];// 模式串向右移动  
 if (j >= n)  
 {  
 count++;  
 cout << "The "<<count<<" time a word appears " << (i - j + 1) / 84 + 1 << " line " << (i - j + 1) << " character start " << endl;  
 j = 0; //J必须重新赋值为零，  
 }// 匹配成功  
 }  
 return count;  
}  
  
void ShowMenu()  
{  
 cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  
 cout << "\*\*\*\*\*\* Literary research assistant \*\*\*\*\*\*" << endl;//文学研究助手  
 cout << "\*\*\*\*\*\* 0.Safe exit system \*\*\*\*\*\*" << endl;//安全退出系统  
 cout << "\*\*\*\*\*\*1.The document is read into the novel\*\*\*\*\*\*" << endl;//文件读入小说  
 cout << "\*\*\*\*\*\* 2.Output novel text \*\*\*\*\*\*" << endl;//输出小说文本  
 cout << "\*\*\*\*\*\* 3.Search for novel keywords \*\*\*\*\*\*" << endl;//查询小说关键字  
 cout << "\nPlease select:";  
}  
  
int main()  
{  
 char index[10] = { 0 }, novel\_path[10] = { 0 };  
 //赋初值为零，分配内存时初始化为零，  
 char \*ch;  
 int \*next, index\_length, n, x = 100;  
 cout << "\t\t\tLog on to the literary research assistant...\n\n\n"<< endl;  
 while (x != 0)  
 {  
 system("pause");  
 system("cls"); //清屏  
 ShowMenu();  
 cin >> x;  
 switch (x)  
 {  
 case 0:  
 exit(0);  
 break;  
 case 1:  
 cout << "Please enter the novel file path:";  
 //输入小说路径  
 cin >> novel\_path;  
 ch = FileRead(novel\_path);  
 cout << "File read successfully!" << endl;  
 break;  
 case 2:  
 cout << ch << endl;  
 break;  
 case 3:  
 {  
 cout << ch << endl;  
 cout << "Please enter the word you want to query:";  
 //请输入想要查询的单词  
 getchar();  
 cin.get(index, 20);  
 index\_length = strlen(index);  
 next = new int[index\_length];  
 GetNext(index, next);  
 n = IndexKMP(ch, index, next);  
 cout << index << "In the novel total appear " << n << " times" << endl;  
 //在小说中总共出现过多少次  
 }  
 break;  
 default:  
 break;  
 }  
 }  
 system("pause");  
 return 0;  
}

1. 实验心得

了解和运用KMP算法以及next函数

1. 进一步的改进和设想

进一步实现中文文章检索