递归子程序法语法分析程序的设计与实现

学号：20201202075 姓名：myp 班级：20计科1

#### 一、实验题目

设计、编制、调试一个典型的语法分析程序，实现对如下文法

的递归子程序语法分析，进一步掌握常用的语法分析方法。

#### 实验目的和要求

(1）文法：G[E]:

E→eBaA

A→a|bAcB

B→dEd|aC

C→e|dC

(2）程序的输入是任意符号串，输出是本次输入的符号串是否是

该文法的句子的结论。

#### 三、程序的框架结构

程序读入一个字符串，自左向右开始分析，当识别到一个可推导文法，则调用对应的分析程序进行分析，如果匹配，则返回0，否则返回1。

#### 四、实验结果

**1．使用说明：**

**运行程序，输入待分析的字符串，程序输出“符合文法G” 或 “不符合文法G”**

1. **其他测试数据：**

**输入：Eadddddddddddddddddddddddddddddddded**

输出：

**输入：eaeaa**

输出：

**输入：edeaeaadabacae**

输出：

#### 六、源代码

#ifndef SUBMAT\_G\_H  
#define SUBMAT\_G\_H  
#include <iostream>  
// 文法 G 的语法分析程序  
class G {  
private:  
 std::string content; // 分析的句子  
 char c; // 当前的字符  
 int c\_index;  
 int errcode; // 0 代表无错误  
 std::string errMsg;  
 int line;  
 int pos;  
  
private:  
 int igetch();  
 int ungetch();  
  
public:  
 G(std::string content);  
 int E();  
 int A();  
 int B();  
 int C();  
 int start();  
 int getResult();  
};  
  
  
#endif //SUBMAT\_G\_H

#include "G.h"  
#include <String>  
#include <iostream>  
  
G::G(std::string content){  
 this->content = content;  
 this->c = '#';  
 this->c\_index = -1;  
 this->errcode = 0;  
 this->errMsg = "success";  
 this->line = 1;  
 this->pos = 0;  
}  
  
int G::igetch() {  
 if (this->c\_index < int(this->content.length()) - 1){  
 this->c\_index++;  
 this->c = this->content[this->c\_index];  
 if (this->c == '\n')  
 {  
 this->line++;  
 this->pos = 0;  
 }  
 return this->c;  
 }  
 return 0;  
}  
int G::ungetch() {  
 if (this->c\_index > -1){  
 this->c\_index--;  
 this->c = this->content[this->c\_index];  
 if (this->c == '\n')  
 {  
 this->line--;  
 this->pos = 0; // 这里还有错，暂时不改  
 }  
 return this->c;  
 }  
 return 0;  
}  
  
int G::E(){  
 if (igetch() != 'e')  
 {  
 errcode = -1;  
 std::cout << R"++(parase error on line: )++" << this->line << "\n";  
 return -1;  
 } else if (B() == 0)  
 {  
 return 0;  
 } else{  
 return -1;  
 }  
}  
int G::A(){  
 if (igetch() == 'a')  
 {  
 return 0;  
 } else if (this->c == 'b' && A() == 0)  
 {  
 if (igetch() == 'c' && B() == 0)  
 {  
 return 0;  
 }  
 return -1;  
 }  
 return -1;  
}  
int G::B(){  
 if (igetch() == 'd')  
 {  
 if (E() == 0 && igetch() == 'd')  
 {  
 return 0;  
 }  
 return -1;  
 } else if (this->c == 'a')  
 {  
 return C() == 0 ? 0 : -1;  
 }  
 return -1;  
}  
int G::C(){  
 if (igetch() == 'e')  
 {  
 return 0;  
 } else if (this->c == 'd' && C())  
 {  
 return 0;  
 }  
 return -1;  
}  
int G::start(){  
 E();  
  
}  
int G::getResult(){  
 if (E() == 0 && igetch() == 0)  
 {  
 std::cout << "符合文法G"<<std::endl;  
 return 0;  
 }  
 std::cout << "不符合文法G";  
 return -1;  
}

int main(){

string str = "eadddddddddddddddddddddddddddddddded";

G g = G(str);

g.getResult();

return 0;

}