词法分析评审材料

本程序实现了所有要求的功能

【操作步骤】

首先进入 gpl-master 目录，然后输入：

make

cd sample

../gpl calc -lex

程序会在 sample 目录下输出 calc.lex 文件

其中就有需要输出的内容

1.输出单词（以二元式形式）

【运行结果】

```
(277, 0x55a8a4d469a0) //INTEGER
(59, -) //';'
(278, 0x55a8a4d46b20) //IDENTIFIER
(61, -) //'='
(278, 0x55a8a4d46b40) //IDENTIFIER
(42, -) //'*'
(277, 0x55a8a4d46b80) //INTEGER
(59, -) //';'
(278, 0x55a8a4d46d00) //IDENTIFIER
(61, -) //'='
(278, 0x55a8a4d46d20) //IDENTIFIER
(47, -) //'/'
(277, 0x55a8a4d46d60) //INTEGER
(59, -) //';'
(259, -) //KEYWORD
(40, -) //'('
(279, 0x55a8a4d46ee0) //TEXT
(44, -) //','
(278, 0x55a8a4d46fe0) //IDENTIFIER
(44, -) //','
(279, 0x55a8a4d470a0) //TEXT
(41, -) //')'
(59, -) //';'
(259, -) //KEYWORD
```

按照老师的要求给出了输出的二元式形式，并且为了方便辨别识别到

单词的具体内容，我在后面加了例如//KEYWORD //TEXT //’;’这样的注释，可以方便人阅读。

【代码简述】

主要是在 yylex 里面 按照老师给出的一些例子 然后用类似的方法写出要支持的功能

2.出错处理

【运行结果】

【用例 1】非法组合出错

error1.gpl

!?　//这是程序的内容 就一行!?

结果：

```
leizhanyao@leizhanyao-virtual-machine:~/Compiler/实验一 词法分析/gpl-master/sample$ ../gpl error1 -lex
yyerror: invalid symbol combination at line 1
```

【说明】遇到!? !! !. !*等非法组合时会报错

【用例 2】非法字符出错

error2.gpl

我爱你　//这是程序的内容 就一行中文字符“我爱你”

结果：

```
leizhanyao@leizhanyao-virtual-machine:~/Compiler/实验二 语法分析/gpl-master/sample$ ../gpl error2 -lex
yyerror: syntax error at line 1
```

【说明】遇到非法字符 如中文字符时会报错

2）部分代码与细节

【思路】在 yylex 识别单词的过程中

我主要在 yylex 中增加了以下代码处理异常：

```
if(character=='!')
    {
       concat();
       getch();
       if(character=='=')
       {
          concat();
          printf("(%d, -) //NE\n", NE);
          return NE;
       }
       else if(!digit()||!letter())
       {
           yyerror("invalid symbol combination");
          exit(1);
       }
       else
       {
          retract();
          printf("(%d, -) //'!'\n", '!');
          return '!';
       }
    }
```

在识别’!’的同时，如果识别到!? !@ !#等非法组合，可以报错

3.符号表管理

【运行结果】

```
WORDLIST:
0x5595a65814f0: main
0x5595a6582560: i
0x5595a6582600: j
0x5595a65826a0: k
0x5595a6582740: l
0x5595a65827e0: m
0x5595a6582880: i
0x5595a65828c0: 8
0x5595a6582980: j
0x5595a65829a0: i
0x5595a65829e0: 2
0x5595a6582b80: k
0x5595a6582ba0: i
0x5595a6582be0: 3
0x5595a6582d60: l
0x5595a6582d80: i
0x5595a6582dc0: 2
0x5595a6582f40: m
0x5595a6582f60: i
0x5595a6582fa0: 2
0x5595a6583120: "i="
0x5595a6583220: i
0x5595a65832e0: "\n"
0x5595a65833e0: "j="
0x5595a65834e0: j
0x5595a65835a0: "\n"
0x5595a6583660: "k="
0x5595a6583760: k
0x5595a6583840: "\n"
0x5595a6583900: "l="
0x5595a6583a00: l
0x5595a6583ae0: "\n"
```

【思路】通过创建符号表结构体，创建全局变量，在 yylex 进行识别单词时，同时把相关地址和字面量存入表中，最后再输出表即可。

【详细细节与部分代码】

结构体定义如下：

```
typedef struct wordlist
{
    int *p;
   char m[100];
}WORDLIST;
```

相关代码如下：

```
if(letter())
   {
      while(letter() || digit())
      {
         concat();
         getch();
      }
      retract();
      num=keyword();
      if(num!=0)
      {
         printf("(%d, -) //KEYWORD\n", num);
         return num; // return keyword
```

```
        }
        else
        {
            lexeme=malloc(strlen(token+1));
            strcpy(lexeme,token);
            yylval.string=lexeme;
            wl[lenwl].p = (int*)lexeme;
            strcpy(wl[lenwl].m, token);
            lenwl++;
            printf("(%d,   %p)   //IDENTIFIER\n",   IDENTIFIER,
lexeme);
            return IDENTIFIER;
        }
    }

if(digit())
    {
        while(digit())
        {
            concat();
            getch();
        }
```

```
        retract();
        lexeme=malloc(strlen(token+1));
        strcpy(lexeme,token);
        yylval.string=lexeme;
        int *p = malloc(4);
        *p = atoi(lexeme);   // *p is an int
        wl[lenwl].p = (int*)p;
        strcpy(wl[lenwl].m, token);
        lenwl++;
        printf("(%d, %p) //INTEGER\n", INTEGER, p);
        return INTEGER;
    }



if(character=='"')
    {
        concat();
        getch();
        while(character!='"' && character!=EOF)
        {
            concat();
            getch();
```

```
        }
        if(character==EOF)
        {
           printf("lex error: \" expected\n");
           exit(1);
        }
        concat();
        lexeme=malloc(strlen(token+1));
        strcpy(lexeme,token);
        yylval.string=lexeme;
        wl[lenwl].p = (int*) lexeme;
        strcat(wl[lenwl].m, token);
        lenwl++;
        printf("(%d, %p) //TEXT\n", TEXT, lexeme);
        return TEXT;
    }
```

最后再 main 函数中输出词汇表即可：

```
if(mode==1)
    {
        yylex();
```

```
    printf("\n\n\nWORDLIST:\n");
    //printf("%p, %s", wl[0].p, wl[0].m);
    int i;
    for(i=0;i<lenwl;i++)
    {
        printf("%p: %s\n", wl[i].p, wl[i].m);
    }
    exit(0);

}
```

4.问题一：你的代码用于存放 3 种符号的值的代码位置在哪里

主要是通过（3）中提到的词汇表结构体来存储的

相关代码如下，在 gpl.y 中：

```
if(letter())
  {
    while(letter() || digit())
    {
      concat();
      getch();
```

```
        }
        retract();
        num=keyword();
        if(num!=0)
        {
           printf("(%d, -) //KEYWORD\n", num);
           return num; // return keyword
        }
        else
        {
           lexeme=malloc(strlen(token+1));
           strcpy(lexeme,token);
           yylval.string=lexeme;
           wl[lenwl].p = (int*)lexeme;
           strcpy(wl[lenwl].m, token);
           lenwl++;
           printf("(%d,   %p)   //IDENTIFIER\n",   IDENTIFIER,
lexeme);
           return IDENTIFIER;
        }
    }
```

```
if(digit())
{
   while(digit())
   {
      concat();
      getch();
   }
   retract();
   lexeme=malloc(strlen(token+1));
   strcpy(lexeme,token);
   yylval.string=lexeme;
   int *p = malloc(4);
   *p = atoi(lexeme);   // *p is an int
   wl[lenwl].p = (int*)p;
   strcpy(wl[lenwl].m, token);
   lenwl++;
   printf("(%d, %p) //INTEGER\n", INTEGER, p);
   return INTEGER;
}

if(character=='"')
{
```

```
    concat();
    getch();
    while(character!='"' && character!=EOF)
    {
       concat();
       getch();
    }
    if(character==EOF)
    {
       printf("lex error: \" expected\n");
       exit(1);
    }
    concat();
    lexeme=malloc(strlen(token+1));
    strcpy(lexeme,token);
    yylval.string=lexeme;
    wl[lenwl].p = (int*) lexeme;
    strcat(wl[lenwl].m, token);
    lenwl++;
    printf("(%d, %p) //TEXT\n", TEXT, lexeme);
    return TEXT;
}
```

（5）实现一个存放字面量的字符串

已经实现

主要是用一个全局变量 ls 实现的

打印效果如图

```
the long string is:
mainijklmijikilimi"i="i"\n""j="j"\n""k="k"\n""l="l"\n""m="m"\n"
separated:
main i j k l m i j i k i l i m i "i=" i "\n" "j=" j "\n" "k=" k "\n" "l=" l "\n" "m=" m "\n"
the positions are:
0 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 38 40 45 50 52 57 62 64 69 74 76 81 86 88

num of lexeme = 31
```

（6）实现了哈希表

主要就是使用数据结构里面哈希表相关的指示

定义哈希表的大小为 7

hash 函数为：data % 7

这样就可以尽可能多的看到哈希表的效果

打印效果如图：

```
hash_table:
0->7->21
15->29->50->57->64
9->23->86
17->31->38->45->52
11->25->74->81->88
5->19->33->40
13->27->62->69->76
```

实验一所有功能都已实现