

【源代码】

1.文件内容：

```
const      a=10;
var       b,c;
procedure   p;
begin
    c:=b+a
end;

begin

number`201913137151
name`xb

end.
name:xb
```

2.代码内容：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

using namespace std;

//保留字和运算符的个数
#define StayNum 13
#define CaclulateNum 9

//保留字 运算符
char stayWord[StayNum][15] = {"const", "var", "procedure", "begin", "end",
"odd", "if", "then", "call", "while", "do", "read", "write"};
char caculation[CaclulateNum] = {'+', '-', '*', '/', '>', '<', '=', '#', ':'};

//判断词法类型
bool isNumber(char ch);
bool iscase(char ch);
bool iscaculationSymbol(char ch);
bool isBandSymbol(char ch);
bool issStayWord(char *str);

//接受文件里的PL语言
void getInput(char *fileName, char *str);

//词法分析
void Analysis(char *InputFileName, char *str);

int main()
{
```

```
//缓冲区
char buffer[10000];
//文件路径
char inputPath[100] = "input.txt";
//开始分析
Analysis(inputPath, buffer);
return 0;
}

//1.判断字符是否是数字
bool isNumber(char ch)
{
    return ch >= '0' && ch <= '9';
}

//2.判断字符是否是字母
bool iscase(char ch)
{
    return ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'));
}

//3.判断字符是否是运算符
bool iscaculation(char ch)
{
    int i;
    for (i = 0; i < CaclulateNum; i++)
    {
        if (ch == caculation[i])
        {
            return true;
        }
    }
    return false;
}

//4.判断字符是否是界符
bool isBoundary(char ch)
{
    return (ch == '(' || ch == ')' || ch == ',' || ch == ';' || ch == '.');
}

//5.判断字符串是否是保留字
bool isStayWord(char *str)
{
    int i;
    for (i = 0; i < StayNum; ++i)
    {
        if (strcmp(str, stayword[i]) == 0)
        {
            return true;
        }
    }
    return false;
}

//从文件中读取输入流
void getInput(char *fileName, char *str)
{
```

```
char ch;
int k = 0;
FILE *f;
//以读方式打开文件
f = fopen(fileName, "r");
while ((ch = fgetc(f)) != EOF)
{
    //以空格间隔每一行
    ch == '\n' ? str[k++] = ' ' : str[k++] = ch;
}
str[k] = '\0';
fclose(f);
}

//词法分析
void Analysis(char *filePath, char *source)
{
    //获得输入
    getInput(filePath, source);
    printf("词法分析器运行结果: \n");
    //文件流字符串长度，要分析的字符串的长度，某个词法字符串的长度
    int length = strlen(source), length1, length2;
    //中间变量
    int i = 0, j, k, t;
    //分析结果，中间变量
    char target[1000], temp[100];

    //循环读取文件流字符串
    while (i < length)
    {
        j = k = 0;
        //滤过开头空格
        while (source[i] == ' ' && i < length)
        {
            ++i;
        }

        //滤过source字符串中间的空格
        while (source[i] != ' ' && i < length)
        {
            target[j++] = source[i++];
        }
        target[j] = '\0';

        //要分析的字符串的长度
        length1 = strlen(target);
        //循环要分析的字符串
        while (k < length1)
        {
            //若字符为字母则持续读入
            if (isCase(target[k]))
            {
                t = 0;
                while (isCase(target[k]) && k < length1)
                {
                    temp[t++] = target[k++];
                }
                temp[t] = '\0';
            }
        }
    }
}
```

```
//判断是否是保留字
if (isStayword(temp))
{
    printf("%s 基本字\n", temp);
}
else
{
    printf("%s 标识符\n", temp);
}
strcpy(temp, "");

//若字符为数字则持续读入
else if (isNumber(target[k]))
{
    t = 0;
    while (isNumber(target[k]) && k < length1)
    {
        temp[t++] = target[k++];
    }
    temp[t] = '\0';
    printf("%s 数字\n", temp);

    strcpy(temp, "");
}

//判断字符是否是界符
else if (isBoundary(target[k]))
{
    printf("%c 界符\n", target[k++]);
    //判断是否结束
    if (target[k - 1] == '.')
    {
        return;
    }
}

//判断字符是否是运算符
else if (isCalculation(target[k]))
{
    if (target[k] == '<' || target[k] == '>' || target[k] == ':')
    {
        if (target[k + 1] == '=')
        {
            printf("%c%c 运算符\n", target[k], target[++k]);
        }
        else
        {
            printf("%c 运算符\n", target[k++]);
        }
    }
    else
    {
        printf("%c 运算符\n", target[k++]);
    }
}
else
{
    printf("%c 非法\n", target[k++]);
}
```

```
    strcpy(temp, "");
}
}
```

【运行截图】

```
问题 输出 终端 调试控制台 1: Code ▼ + ×
PS F:\Code_c\single_c\编译原理> cd "f:\Code_c\single_c\编译原理" ; if ($?) { g++ exec.cpp -o exec } ; if ($?) { ./exec }
词法分析器运行结果:
const 基本字
a 标识符
= 运算符
10 数字
; 界符
var 基本字
b 标识符
, 界符
c 标识符
; 界符
procedure 基本字
p 标识符
; 界符
begin 基本字
c 标识符
== 运算符
= 运算符
b 标识符
+ 运算符
a 标识符
end 基本字
; 界符
begin 基本字
number 标识符
` 非法
201913137151 数字
name 标识符
` 非法
xb 标识符
end 基本字
. 界符
PS F:\Code_c\single_c\编译原理>
```

【实验小结】

通过本次实验，我对编译的过程有了有深刻的理解，知道了高级语言是如何运行在机器上的，了解到了词法分析的过程，收获很大。