编译原理课程词法分析实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **专业班级** | | **计科5班** | **姓名** | | **李林轩** | **学号** | **20184320** |
| **实验题目** | 词法分析程序的设计与实现 | | | | | | |
| **实验时间** | **2021/4/10** | | | **实验地点** | **DS1421** | | |
| **实验成绩** |  | | | **实验性质** | **□验证性 □设计性 ■综合性** | | |
| 教师评价：  □算法/实验过程正确；□源程序/实验内容提交 ☑程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | | |
| 一、实验目的  🡪了解编译器中词法分析环节的作用和意义  🡪掌握词法分析基础  🡪实现实验要求中C语言子集的词法分析器程序 | | | | | | | |
| 二、实验项目内容   1. 实验内容   根据某个现有编程语言的词法规则（或设计一套新的词法规则），建立词法分析器展开识别，具体功能要求包括：   1. 支持标识符、常量（含十进制与十六进制的整数、浮点数）、关键字、分界符、运算符等词法类型； 2. 输出每个词的类型和单词属性； 3. 检查源程序中存在的词法错误，并报告错误所在的位置； 4. 查填符号表，例如，对于变量标识符，需在符号表存入标识符名字、类型等信息。 5. 实验要求： 6. 自选程序设计语言（java，python，c/c++）作为实现语言，手工编写词法分析程序。 7. 提交实验报告及源代码。实验报告需严格遵循学校文档规范，内容包含对应文法、词类编码表、词法分析测试用例。 | | | | | | | |
| 三、实验过程或算法内容要点，供参考：   1. 程序实现内容介绍 2. 语言说明：词法分析器所包含C语言子集相应的文法/正规式/状态转换图 3. 词法编码表及说明 4. 符号表/符号表接口及说明 5. 错误处理说明 6. 程序说明：可选择其中重点对象进行说明，不需要在这里列出全部源代码   一、程序实现内容介绍  本程序实现了以C语言的词法规则子集为基础，在此基础上进行一些修改而成的词法分析器。该词法分析器可识别标识符、常量（含十进制与十六进制的整数、浮点数）、关键字、分界符、运算符等词法类型，可以进行分词并输出每个词的编码，类型与单词属性，可以检查源程序中存在的词法错误，并报告错误所在的行数与出错的单词，并且将输出自定义的错误类型编码。可以通过符号表，常数表，错误表以及相应的入口确定需要查看的单词。  二、语言说明  源程序中的记号，表达式，C语言子集文法：   1. 标识符：按照C语言的规则，标识符的正则表达式为： ^[\_A-Za-z][\_A-Za-z0-9]]+$ 2. 运算符：本词法分析器可识别的运算符包括：   +、-、\*、/、%、+=、-=、\*=、/=、==、!=、>、<、<=、>=、&、&&、|、||   1. 关键字：本词法分析器可识别的关键字是C语言的子集，可能有少量关键字不包括在内。关键字的单词属性用数字表示，一共分为了：   声明变量类： （int,float,struct等）  限定修饰符类： （const,unsigned,static等）  循环分支语句类：（for,while,break等）  输入输出类： （scanf,printf）  函数特有类： （void,return,volatile）  关键字及关键字对应的单词属性存储在vector<pair<string, int>> keyword   1. 数字：数字分为十进制整数，十六进制整数，十进制浮点数，十六进制浮点数四种，会分别进行类别的识别与输出。同时，词法分析器允许以小数点开头的浮点数，视为整数部分为0。 2. 分界符：分界符包括(、)、{、}、[、]、#、,、;、’、” 3. 词法编码表及说明  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **单词** | **编码号** | **单词** | **编码号** | **单词** | **编码号** | | **auto** | **1** | **break** | **26** | **/=** | **51** | | **int** | **2** | **do** | **27** | **==** | **52** | | **double** | **3** | **default** | **28** | **>=** | **53** | | **float** | **4** | **scanf** | **29** | **<=** | **54** | | **char** | **5** | **printf** | **30** | **!=** | **55** | | **string** | **6** | **void** | **31** | **(** | **56** | | **long** | **7** | **return** | **32** | **)** | **57** | | **struct** | **8** | **volatile** | **33** | **[** | **58** | | **register** | **9** | **+** | **34** | **]** | **59** | | **typedef** | **10** | **-** | **35** | **{** | **60** | | **short** | **11** | **\*** | **36** | **}** | **61** | | **union** | **12** | **/** | **37** | **#** | **62** | | **const** | **13** | **=** | **38** | **,** | **63** | | **unsigned** | **14** | **>** | **39** | **;** | **64** | | **signed** | **15** | **<** | **40** | **‘** | **65** | | **static** | **16** | **&** | **41** | **“** | **66** | | **extern** | **17** | **|** | **42** | **:** | **67** | | **for** | **18** | **!** | **43** | **整数** | **68** | | **while** | **19** | **%** | **44** | **浮点数** | **69** | | **switch** | **20** | **++** | **45** | **布尔常量** | **70** | | **case** | **21** | **--** | **46** | **标识符** | **71** | | **goto** | **22** | **\*\*** | **47** |  |  | | **if** | **23** | **+=** | **48** |  |  | | **else** | **24** | **-=** | **49** |  |  | | **continue** | **25** | **\*=** | **50** |  |  |   在上述编码中，每一个关键字都有自己的编码，关键字的编码按照类别顺序编码，对于编码1-12的关键字，其定义为声明变量类，所有词类编码如下列表格所示：   |  |  | | --- | --- | | **关键字编码号** | **所属类别** | | 1-12 | 声明变量类 | | 13-17 | 限定、修饰符类 | | 18-28 | 循环，分支语句类 | | 29,30 | 输入输出类 | | 31,33 | 函数类 | | 34-55 | 运算符类 | | 56-67 | 分界符类 | | 68-70 | 常量类 | | 71 | 标识符类 |   常量一共分为5种：十进制和十六进制的整数和浮点数，以及布尔常数。 注意，“true”和“false”是关键字，即为布尔常量。本词法分析器中，将“true” 和“false”的词类编码设置为同一个，表示布尔常量。   1. 符号表/符号表接口及说明   本词法分析器将符号表分为三部分：标识符符号表，常数表以及错误表，他们的规则都是通过在词法分析序列中的符号表入口以及对应的符号表来确定想要查找的单词以及编码。  符号表接口：通过该函数可将符号表与符号表入口对应，写入symbol\_table中 IMG_256  具体测试结果见 第四部分：实验测试结果   1. 错误处理说明   1. 错误输出：  错误发生的行号与错误类型将输出在terminal，错误的类型编码与错误表入口将输出在result.txt，具体单词与错误类型将输出在error.txt文件中，可通过result.txt中的入口确定错误。  2. 错误定义：  1 数字错误：  当一串字符串以数字开头时，将默认其为数字，此时如果后面的字符不全为数字，则判定为数字错误。当然也有可能存在是以数字开头而造成的标识符命名错误，在本词法分析器中统一按照数字错误处理。该类型错误编码为-1 。  2 浮点数字错误  当数字中出现了小数点时，状态则转为浮点数，浮点数出现错误与数字错误相似。该类型错误编码为-2 。  3 运算符错误 当出现了不存在于编码表中的运算符时，按照错误运算符处理。该类型错误编码为-3 。  七、程序说明及状态转换图说明  1. 程序输入形式  以文件输入，输入时需要指明文件路径，或输入default使用同级目录下的文件  2. 程序输出形式  将词法分析的结果输出到文件中，末尾会输出对该程序的词法分析统计结果，包括每种词法的数量，字符总数，行数等；在屏幕上将输出错误，包括错误类型，错误所在行数，错误的单词。  3. 程序执行流程  对文件内容逐行输入，对每一行逐个字符进行分析。根据当前读入字符进行状态转移，在实现的过程中没有明显的将状态表现出来（比如case xxx：），而是隐式的判断后进行状态转换。  变量和函数：    IMG_256  4. 不同类别单词状态转换图及程序实现   * 运算符类   状态转换图：    代码实现：  当前读入字符在运算符的字符串中寻找到了，将状态置为-1.意味着本次读取结束，不需要后续状态转移：由于运算符最多为两个字符，最多再读一个字符即可，因此在下方判断后一个字符是否也为运算符。    随后进行状态机转移判断，如果满足转移条件，则将对应的操作符以及词类编码输出到文件中，如果不满足转移条件，无法到达结束状态，则进行错误处理  IMG_256  框1：正常情况下，在最终的词法单元序列中输出词法编码 框2：如果无法进行状态转移，未寻找到字符，则进行错误处理。   * 分界符类   分界符类较为简单，读入一个字符判断即可，无后续状态转移，读入一个字符直接到达终结状态。  状态转移图：    代码实现：对读入符号进行判断，将状态转为-1，之后在文件里进行输出对应词法编码。  IMG_256   * 常量类   布尔常量尽管属于常量类，但是在实现上放入存储关键字的vector<pair<string,int>>keyword更方便查找，因此常量类的状态转移图只包括十进制和十六进制整数和浮点数  状态转换图：    代码实现：  当数字开头为’0x’时，即识别出当前数字为16进制，接着进行对应的状态转换即可。如果出现错误将会返回对应类型的错误，交予judge函数进行错误处理。     * 其余关键字类别与标识符类   由于其余关键字均由英文小写字母组成，因此代码对其余关键字类别的识别实现是通过对读入的字符串进行匹配，读入的字符串按照标识符的状态转移。  状态转移图：    代码实现： 首先需要将在接受范围内的字符读入，即a-z,A-Z,\_和 . 。由于在读入后判断是否为数字常量或关键字或标识符，因此统一读入，读入完毕后交给judge函数判断。  IMG_256 | | | | | | | |
| 四、实验测试结果   1. 测试程序一：将实验二基于的程序稍作更改：将get，put改为   # include<stdio.h> int main(){  int num1,num2,op,ans;  scanf("%d%d%d",num1,num2,op);  if(op==0)  {  ans = num1 + num2;  ans ++;  };  if(op==1)  {  ans = num1 - num2;  ans --;  };  if(op==2)  {  ans = num1 & num2;  ans %= 10;  };  if(op==3)  {  ans = num1 | num2;  ans |= 10;  ans &= 15;  ans = ans\*\*2;  };  double f = 0.001;  double f\_err = 0.0.1;  double 9\_errID = 0x12.34abc;  printf("result = %d",ans); }  控制台输出： IMG_256  result.txt文件内容（由于文件内容较长，此处放部分截图）  IMG_256  对于 |=，&=这类错误的运算符，该词法分析器可以进行识别并将错误类型输出在result.txt中，对于\*\*等双字符的运算符也能够正确识别,但由于在词法单元序列中以编码的方式进行存储，在此不好展示。  对于0.0.1这种不合法的浮点数与9\_errID这种不合法的标识符/数字，能够进行错误类型以及出错单词的输出  IMG_256  在result.txt文件末尾将输出对于词法分析测试程序的统计结果，包括程序所有单词，字符以及错误的数量，以及对于每个单词的数量统计。  IMG_256   1. 测试程序二   #include <stdio.h> int main() {  printf("Hello World");  int \_dawdwaf;  int a = 11;  int normal\_digit = 0x12312;  int error\_digit = 0123fsdffe;  int error\_float\_digit = 0.21.e;  int >= &=;  return 0; }  控制台输出IMG_256  result.txt部分内容:  IMG_256  symbol\_table.txt内容：从上往下以此为常数表，符号表以及错误表IMG_256  可通过result.txt中符号表入口与符号表中的序号进行查询确定。  当没有错误存在时，会输出Perfect!，不在此演示 | | | | | | | |
| 五、实验总结  1、实验过程中所遇到的问题及解决办法  · 目前该词法分析器还有能够优化的部分，比如说在读入文件中未涉及到的字符时（比如说中文的标点符号），会因为找不到下一个状态而进入死循环，后续可以引入新状态用于处理这些字符，将这部分进行优化。  · 代码部分存在大量冗余，状态转移也采用的是隐式的状态转移，从程序的结构上没有很好的体现，后续可以考虑进行代码重构。 | | | | | | | |