**华南师范大学本科生实验报告**

姓名＿ 何子浩 学号 20192133046

院系＿计算机学院＿ 专业＿ 人工智能 ＿

年级 2019 班级＿6班＿

实验时间 ＿2021＿年＿9＿月＿16＿日

实验名称＿ C++单词拼装器 ＿

课程名称＿＿ 编译原理＿＿＿＿

华南师范大学教务处编印

|  |
| --- |
| 实验课程：编译原理  实验名称：单词拼装器 |
| **一、实验内容**   * **基本内容**   **1.把C++源代码中的各类单词（记号）进行拼装分类。C++语言中包含了几种类型的单词（记号）：标识符，关键字，数（包括整数、浮点数），字符串，注释，特殊符号（分解符）和运算符号等。**  **2.要求应用程序应为windows界面。**  **3.打开一个C++源文件，列出所有可以拼装的单词（记号）。**  **4.应该书写完善的软件设计文档。**   * **选做内容**   **预编译系统的实现----打造具有个人风格的XC++语言（单词替换）**  **1.描述具有风格的XC++的单词有哪些，分别对应原C++的是哪些单词。**  **2.实现这个单词替换方案。**  **3.需要按上述1，2的内容书写相应的设计文档** |
| **二、实验目的**   * + 对编译器运作原理形成初步的了解   + 为后续实现编译器的词法分析等打下基础   + 简单回顾数据结构中的一些基本结构的特性及使用 |
| **三、实验文档：**  **（1）设计思路：**  **单词拼装的过程大致可分为：代码读取->单词切分->单词识别。**  **代码读取：每次读取一行，存放在一个字符数组line中，后续的操作都基于该数组。**  **单词切分：本程序采用较简单从左到右依次判断拼装，根据单词开头进入不同的switch分支。由于C++单词较多，为提高选择效率，设置了一个int型分支判别数组的辅助结构，数组大小为200+256。初始化时需要读取给定特殊符号的文件，该文件每行为一个C++运算符或其它特殊符号，每次读取一行，根据第一个字符的ASCII码对数组对应的单元按以下规则顺序定义：**  **1. 下划线、字母**  **2. 数字**  **5. 特殊字符**  **3. 单引号、双引号**  **4. 斜杠，为注释**  **6. 分解符：大、中、小括号，分号，逗号**  **7. 井号**  **其余字符默认为0**  **特别说明，分支判别辅助数组大小为200+256，在switch语句选择中需要对转换后的ASCII码加上200才为实际对应单元，该操作可以合并某些特殊字符，如遇中文时，一个汉字由3个负的ASCII（GBK编码）组成，此时这些字符在数组中的单元只能为0~199。**  **在对行操作时，定义了一个拼装后的单词字符数组word，标识该行是否为预处理器命令的布尔变量prepp且默认为假，当读入起始字符为井号令其为真，此时部分分支需要作出相应修改。**  **以下为各switch分支说明：**  **case 1：从左往右依次扫描，若为下划线或字母则拼装进word数组，否则停止扫描。然后在关键字字典中查找，根据结果输出该词为关键字还是标识符。**  **case 2：先判断起始的数字是否为0，若为0且后一位字符为x，扫描后续符合十六进制的字符进行拼装；若0后一位字符为小数点，依次判断是否为指数形式（e和E），扫描拼装数字后再判断l（L）结尾；若0后一位是字母或下划线，不符合格式，提示语法错误，分词终止；其余情况则拼装成八进制数。若起始字符不为0，则限定为十进制数，依次检查’u’（U）、’l’（L），若还包含小数点，再检查指数形式’e’（E），单精度’f’（f），高精度’l’（L）。**  **case 3：单引号或双引号开头，在该行中往右扫描直至找到匹配的引号（同时判断前一个字符是否为反斜杠），将引号中的内容拼装成串。**  **case 4：斜杠开头，若后一个字符为’\*’，则后续内容为段注释，依次扫描每一行，直至找到匹配的”\*/”，然后将该内容拼装成串，若找不到则提示未能找到配对的注释号，分词终止；若后一个字符为’/’，则后续内容为行注释，对该行剩余内容拼装成串输出；若皆不是星号或斜杠，则跳转到下一个case；**  **case 5：由于运算符最多由3个符号组成，扫描并最多拼装3个特殊符号，在运算符字典查找当前拼装的单词，若查找不到则回退一个字符继续查找，如此循环最多3次，其余情况提示语法错误，分词终止。**  **case 6：分界符直接输出**  **case 7：井号标记当前行为预处理器命令，考虑预处理器关键字较少，采用字符串数组的线性存储结构存储关键字。若识别出的单词为include，对后面出现的包括尖括号’<’，’>’内容拼装成一个整体，输出为头文件；同时case 5增加判断prepp为真且当前单词为“##”时，输出特殊符号。**  **单词识别：定义了关键字字典和特殊符号字典，字典采用拉链方式的散列表实现。散列表类中包含一个结点结构体数组，哈希方法为除留余数法，每个单元是带表头结点的单链表。**  **本程序分词的主要函数为：**  **HashTable<String, int> \*createDict(string path): 创建字典**  **int \*createBranch: 创建分支判别辅助数组**  **bool complie(): 切分单词**  **（2）程序结构体/类：**   |  |  | | --- | --- | | **HashTable(散列表结构体)** | | | **属性** | **方法** | | int divisor: 哈希的除数 | bool search(const K key, E &elem)  根据搜索成功与否返回true/false | | int TableSize: 桶的数量 | bool insert(const K key, E &elem)  新元素插入散列表 | | ChainNode<E, K> \*ht 散列表定义 |  |  |  |  | | --- | --- | | **ChainNode(散列表结点结构体)** | | | **属性** | **方法** | | E data: 元素 |  | | ChainNode<E,K> \*link: 链指针 |  |  |  |  | | --- | --- | | **分支判别数组(字符数组)** | | | **单元（字符的ASCII码值+200）** | **值** | | 下划线、字母 | 1 | | 数字 | 2 | | 特殊符号 | 5 | | 单引号、双引号 | 3 | | 斜杠 | 4 | | 分界符：各种括号，逗号，分号 | 6 | | 井号 | 7 |   **（3）图形界面设计：**  设计一个主窗口，主界面为两个文本框，左边为读入的文件，右边为解析结果。菜单栏包含两个选项：打开解析文件、开始解析  除了根据QT类型修改部分C++源文件的部分数据类型，以及修改文件读写方式，其他内容嵌入QT项目中；  **（4）程序不足之处：**  ①程序对大部分不符合C++语法格式的单词做了初步判断，但存在极个别疏漏；  ②程序使用了较多的if语句，代码的复用率和可读性较低；  ③目前程序只支持GBK编码的cpp，否则其中的非英文字符会出现乱码；  ④考虑到cpp文件每行的代码字符一般不会太长（书写习惯），故行读入的字符数组line设定为1000。但如果超过1000个，后面的字符会出现截断，且可能引发意外的语法错误提示。  （5）**系统的整体性能分析：**  ①关键字、特殊字符字典创建：由于词的数量固定，且采用哈希方式，时间复杂度为O(1)；  ②创建分支判别辅助数组：只对其中部分单元修改，且修改次数固定，时间复杂度为O(1)；若已经创建则直接读入，时间复杂度为O(1)。  ③单词切分：根据字符的ASCII码直接定位分支判别数组中的单元，根据单元中的数进入对应的switch分支，由于各分支执行语句不同一且多为if语句，根据代码文件的行数n决定，则时间复杂度为O(n)。  ④单词识别：选择合适的除数，则散列表的查询复杂度为O(1)  **针对本程序代码进行测试：**   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 第一组 | 第二组 | 第三组 | 第四组 | | 行数 | 300+ | 1100+ | 9500+ | 95000+ | | 时间/s | 0.05s | 0.24s | 0.90s | 7.99s |   （7）**图形界面**  **主界面：** |
| **四、实验心得：**  本次实验内容比较简单，只要基本学过C++语言也能按从左到右依次切分拼装的简单思路实现。但实际编写发现由于C++语言语法、符号太丰富，编码过于繁琐冗余，且容易考虑不周，某些情况未能处理或错误处理。  分词的基本思路是每次读入一个字符，根据单词的起始字符进入不同switch分支。如果每一种情况都单独设定一个case，不仅代码量增多，且增加了大量无用的判断，降低分词效率。虽然查阅资料了解，编译器会根据switch分支条件有3中情况的查找优化，但为了进一步提高判断效率且考虑到数组是查找效率最高的数据结构，定义一个分支判别数组，通过字符的ASCII码直接定位数组单元，根据对应单元存储的值进入对应分支，由于C++单词的分类较少，可以极大减少case分支，提高判断效率。  在分词方法确定的情况下，如何提高单词识别效率是另一大问题。C++关键词较多，采用线性查找方式在处理代码量中等或大量的时候分词效率低下，且未对其进行词法分析和优化。这里程序采用了拉链散列表的方式解决，侧面说明了基本数据结构虽然不多且较为简单，却能有效通用解决大部分应用问题。  在实际分词速度测试上，针对本程序在百、千、万、十万级别代码量进行分词，在小规模代码量的分词速度可以接受，且基本呈线性增长。更进一步的分词效率/词法分析在往后进一步提高。 |
| **五、参考文献：**  《数据结构》教材  《c++程序设计》  C++switch语句性能分析：  <https://blog.csdn.net/weixin_44363885/article/details/96025797>  部分预处理器关键字：  <https://www.cnblogs.com/hehehaha/p/6332162.html>  C++运算符：  <https://www.runoob.com/cplusplus/cpp-operators.html>  C++常量：  <https://www.runoob.com/cplusplus/cpp-constants-literals.html>  C++注释：  <https://www.runoob.com/cplusplus/cpp-comments.html>  C++关键字：  <https://www.runoob.com/w3cnote/cpp-keyword-intro.html>  C++memset函数用法：<https://blog.csdn.net/qq_39707351/article/details/83958024>  QT char\*、string和QString转换：<http://c.biancheng.net/view/1843.html>  QT TextBrowser用法： <https://blog.csdn.net/qq_41767945/article/details/99687416>  QT QFileDialog用法： <https://blog.csdn.net/xdw_it/article/details/80283036>  QT release版本下进行调试：<https://blog.csdn.net/qq_26374395/article/details/90313312> |