**深 圳 大 学 实 验 报 告**

**课程名称： 编译原理**

**实验项目名称： 词法分析程序设计**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 尹剑飞**

**报告人： 钟靖扬 学号：2020111051 班级： 01**

**实验时间： 2023年3月1号-2023年3月20号**

**实验报告提交时间： 2023.3.25**

**教务处制**

|  |
| --- |
| **实验目的与要求：**  **目的：**针对状态矩阵、NFA、DFA、正规式、确定化算法、最小化算法、Thompson算法等词法分析应用问题，设计并实现相应的解决方案，通过设计FA相关类族，以及实现前述几个重要的算法，既加深对抽象的词法、自动机、正规式等形式语言理论基础概念的理解与掌握，也加强对面向对象程序编写能力和计算思维的培养。  **要求:**  **第一部分： 无符号数的识别及DFA的应用**  这部分又分为2个小实验   1. 输入：一个文本文件（源代码文件）   输出：将源代码中的无符号数识别出来并输出到另一个文件中  示例：如果输入是“123\*abc+def/99.2+9.9E+c”，那么输出是：（数字, 123），（其它，\*abc+def/），（数字，99.2），（其它，+），（异常，9.9E+c）  说明：其它是非数字打头的字符串；异常是数字打头，但最后却是不符合定义的无符号数。   1. 假设：用字符“ABCDEFGHIJK”（大写）分别表示数字0..9和.、E、+、-，那么，字符串“BCLD”表示数字“12E3”=12000；   输入：一个文本文件  输出：将隐藏在文本文件中的有效无符号数识别出来。  示例：如果输入是“BCD\*abc+def/JJKC+JKJL+c”，那么输出是：（数字, 123）、（数字，99.2），无效（异常）的无符号数不输出  **第二部分：DFA/NFA的读写及确定化、最小化算法 （选做）**  这部分又分为3个小实验   1. 输入：一个文本文件（格式自定义）   输出：读入文件中的DFA/NFA，创建对应的DFA/NFA对象，再写回另一文件中。  示例：如果输入是“f(S,a)=A, f(A,b)=B, {B}” //假设默认开始状态是S，  那么输出是：K={S，A，B}；Σ={a,b}；f(S,a)=A, f(A,b)=B；S；Z={B}   1. 输入：上一实验输出的NFA文件   输出：将读入的NFA进行确定化后，输出结果，需要考虑有/无ε的情况。   1. 输入：上一实验输出的DFA文件   输出：将读入的DFA进行最小化后，输出结果。  **第三部分：正规式及其应用**  这部分又分为3个小实验   1. Thompson方法的实现   输入：一个正规式  处理：创建与正规式对应的NFA对象，再将NFA对象写回文件，格式与前述实验相同，使得能够读回；  示例：如果输入是“a\*b”，那么输出是：K={A , B , C , D , E , F}；Σ={a, b}；f(A,ε)=B, f(B,a)=C, f(C, ε)=B, f(C, ε)=D, f(A, ε)=D, f(D, ε)=E, f(E, b)=F；A；Z={F}  说明：优先级处理可用栈或递归，思路见算符优先分析法。概要来说，判断当前运算符与下一个运算符间的优先级，当前≥下一个则计算（产生NFA），否则入栈或递归。   1. 从正规式到DFA   将上述实验与前述实验连接起来，使得读入一个正规式文件，能输出一个DFA。正规式文件格式自定，通常每行一个正规式。  示例：如果正规式文件包括2行，每行分别是“a”和“b”，那么，输出的DFA实际上是“a | b”对应的DFA。   1. 字符串的识别   输入：一个正规式文件和一个字符串文件  输出：判断字符串文件中的每个字符串，能否被正规式对应的DFA所识别  其次，再给每个正规式增加一个类别，识别到给定字符串符合某个特定正规式时，输出该类别。  示例：如果输入  a\* 类A  b 类B  那么对字符串aaa输出：aaa，类A |
| **方法、步骤：**  要完成本实验，依据实验要求进行分解，需要从数据结构设计和面向对象设计角度，思考并回答下述问题：   1. 如何识别无符号数？   如何判断一个输入的字符串中，包括无符号数字？如何判断无符号数的开始？中间字符及结束？  根据状态矩阵画出相应的DFA和状态转换表来获取数据。      得到的表如上所示，-2表示进入的是异常的状态，也就是开头数字合法，结尾却不能形成合法的数字，-1则是进入其他的状态，也就是无符号数之外的内容。  而我们需要的结果分别有三种，数字，其他和异常，因此根据我们的需要画出对应的状态表   1. 如何实现简单的字符映射加解密算法？   例如ABCD在ASCII表中是按顺序排列的，因此我们可以比较简单的直接用A～J来表示0～9用加法就可以完成一个简单的映射。  剩下的例如用K代替‘.’等等，可以用Map来实现key和value的映射，通过HashMap可以通过Key很快搜索到。   1. 如何定义DFA/NFA对象，特别的，对自动机里面的映射函数f如何定义、存储？（注意这里的“对象”，指用面向对象的程序设计思维方式（内部数据结构、消息）来定义一个DFA类、一个NFA类。）   请在这里，补充完善你的设计思路，完成后，请删去这句话。  这里用一个Automate的名称来定义自动机的类，里面首先要存的就是它的状态表，也就是上面第一问给出来的表，并且用二维int类型矩阵存起来，    如上图所示，通过数组索引的方式就很容易的可以是实现状态的转换。  而映射的关系，对于不同的输入，我们自然就有不同的映射，因此用Map可以很好地实现函数的映射.    而在判断的时候，我们会根据不同的输入，返回不同的字符串，例如输入符号什么的，就返回sign，输入数字，就返回number，那么它们就会去对应自动机的状态。   1. 如何实现确定化算法？（选做）   确定化算法需要把一个旧的状态集合作为一个新的状态，这种情况如何处理？又如何求某个状态的ε闭包？  首先要将空转移去掉，然后再去检查是否还是有不确定的状态转移。  如果有，则需要定义一个二维表格，列头表示的是各个不同的输入，例如a和b  行头表示的是不同的状态，当在已有的状态当中，发现某个输入生成了新的状态，则需要添加一行关于这个新的状态的数据  如这个NFA自动机    B状态下输入b会到达两个不同的状态，所以需要进行确定化  生成的表格如下：    将BN视为一个新的状态，即可完成确定化算法，将NFA转为DFA   1. 如何实现最小化算法？（选做）   最小化算法需要对原状态集进行等价类划分，这个划分要如何处理？  请在这里，补充完善你的设计思路，完成后，请删去这句话。   1. 汤普森方法需要处理算符的优先级，该如何处理？如何进行递归调用？   算符是给括号最高优先级，然后到或操作，到连接操作再到闭包操作，使用栈来实现运算，不需要用递归操作，当前的算符的优先级比栈顶的算符优先级高的时候，就将栈顶元素弹出并且进行运算，否则就入栈，不过这里的右括号需要稍微做一下特殊处理，因为右括号的优先级只比左括号优先级低   1. 如何根据给定DFA，判断并输出给定输入串的类别？   对于给定的DFA，有着对应的一个状态转换矩阵，可以看成一个二维数组，列就是输入的转移信息，行就是此刻处于的不同的状态。因此我们可以逐个读入输入串的字符，从起始状态开始，根据当前读入字符寻找下一个状态。 |
| **实验过程及内容：**  实验过程及内容，处理代码设计说明、代码及其注释外，特别关注编程过程。  **Part01**  第一问  我们定义identifyNumber类来实现判断  Main函数将类实例化后，直接输入路径进行判断输出就可以了。    先大致介绍一下该类下面的成员变量的作用  **成员变量**：  如下图所示，一共有七个私有的成员变量    Line就是用于存放每一行的数据，writer是文件写入流使用的类。  Numbers和Others是用于存放要输出的数字或者其他的内容，异常的内容实际上也是用Numbers来代替，因为异常最开始识别到的首先是数字。  IsException作为异常标志位，Automate是实现判断自动机状态的功能  State用来存放当前自动机识别的状态  **成员方法**：  这里只有两个成员方法。一个是用来初始化文件读写，获取文件数据的函数identify（）  这个方法很简单，功能只是将文件里的内容提取出来，并且读取每一行，然后去识别每一行的内容    最核心的功能在parseToken里面。  **ParseToken（）方法**：  在parseToken当中，用for循环去识别每行的每个字符。  根据**当前字符**以及已经构建好的**自动机**来判断下一个状态是什么。    整体的逻辑如上流程图，用了三个if-else来进行判断，主要区分的只有三种状态，也就是**异常**，**数字**和**其他**。  这里着重讲一下中间这个通过创建好的自动机获取状态迁移的方法。  代码接口调用如下    看看这个方法具体是怎么实现的。    看到只有一行的代码，只是单纯的去索引一个二维数组里面的值而已。  那么为什么用一个简单的二维数组就可以实现自动机呢。  我们要先画出无符号数识别的自动机出来，就很好理解了。    这个自动机实际上也并不复杂，绿色的是可以做为终结状态，而如果在红色的状态里面停止了识别，则可以识别为异常或者其他。  通过状态表可以更清晰地分辨异常，正常和其他的状态    如果是正数，则为正常的状态，而如果是-2，则是异常的状态，如果是-1，则是“其他”的状态  这样就完成了所有的识别流程了，最后我们看看识别出来的效果。    左边是我们的测试样例，右边是则是识别出来的最后的效果。  **第二部分**：  该题目的第二部分也是比较简单的，只是将输入的一些特殊字符和数字转换成了英文，类似于一种加密，而我们只需要去做相应的解密，将字母重新转换成第一部分的正常的数字和特殊字符即可。  来看看main函数是怎么实现的。    下面那个部分其实就是调用第一部分写的类文件的代码，因此其实区别只是在上面多了一个解密的类方法而已。    为了方便对应特殊字符解密，这里用了HashMap来做一个字符的映射，而0-9直接映射到A-J即可，不需要哈希表辅助映射了。    转换代码如上图所示，如果是J前面的字符，就直接映射为数字，如果是J后面的，就用hashMap的get方法来获取对应的字符。    最终效果如上图，Decode文件是对Test02文件解密之后的结果，然后直接用第一部分的代码来识别Decode文件即可。  **Part02**  **Part03**  这题比较复杂，需要分成很多步骤去完成  首先我们要给出基本的正则式，都可以转换成什么样子的NFA，将最基本的列出来之后，就可以进行递归，一层层地将自动机构建出来。 |
| **实验结论：**   1. 测试用例的设计与说明   对每部分的实验，你分别设计了什么测试数据（测试用例）进行测试，你设计测试数据的出发点是什么，有什么目的？   1. 测试结果的说明与分析   对测试（实验）结果进行分析说明 |
| **心得体会：**  除了实验的心得体会外，  还要额外增加说明你对文法、直接推导、推导、句型、句子、语言这些概念的理解。  首先来说说对第一题的体会，才刚做完第一个实验不久，因此肯定是对第一个实验的对C语言文法分词是记忆犹新的，因此当碰到第一题要识别无符号数的时候，第一感觉就是觉得非常的相似，感觉类型似乎是一样的，于是直接套用了大概的模板，但是后面意识到本节课的内容是学到了DFA自动机，题目也是写明了，需要用到自动机的内容，因此才意识到解题的思路其实是和第一次实验的大有不同。  尽管思路上是类似的，但是这种无符号数实际上需要考虑的情况有很多，如果像之前那样使用if语句来判断，就很容易会出现非常复杂的情况，代码也难以看懂，可能会有很多if-else的语句嵌套，而在参考了DFA的思路之后，发现使用状态表来判断下一个输入处于的状态是非常的方便，只需要很少的判断逻辑，一个简单的for循环加上一个二维矩阵的索引，就可以判断出这个单词是否是一个无符号数了，而如果是判断整句的话，稍微修改一下逻辑，也是可以很快地识别出句子中所有的无符号数，比较麻烦的只不过是将一些异常情况提取出来。而第一题的第二小问本质上只是需要我们做一下字符的转换，也就是做一个简单的解密算法而已，用Map就可以很容易地实现映射。  接下来第三题才是真正比较难的题目，这题花的时间个人感觉比以往加起来的时间都要多，首先是代码的设计思路比较难懂，一开始没有想到用逆波兰式去进行转换，因此觉得要生成NFA非常的困难，在听老师讲解后，发现使用算符优先级去转换成逆波兰式后，发现思路上就容易很多了，于是开始了构造NFA的类。  到了第二部分需要转换成DFA也花费了很多时间，不仅要把空转移去掉，还要生成二维表格来表示状态转移表，该过程的代码量非常地多，即使是采用面向对象的思想，也有很多的细节需要注意，因此不仅在前期思路构想和编写代码需要花很多时间，后期还经常出现得不到想要的结果的情况，例如DFA的终止状态会有多个，又或者在去除空字符的过程中，终止状态和起始状态都有可能改变，因此往往需要进行调试，检查错误，好在最后成功实现了所有的功能，不由得感慨这次实验的难度之高。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：尹剑飞  年 月 日 |
| 备注： |