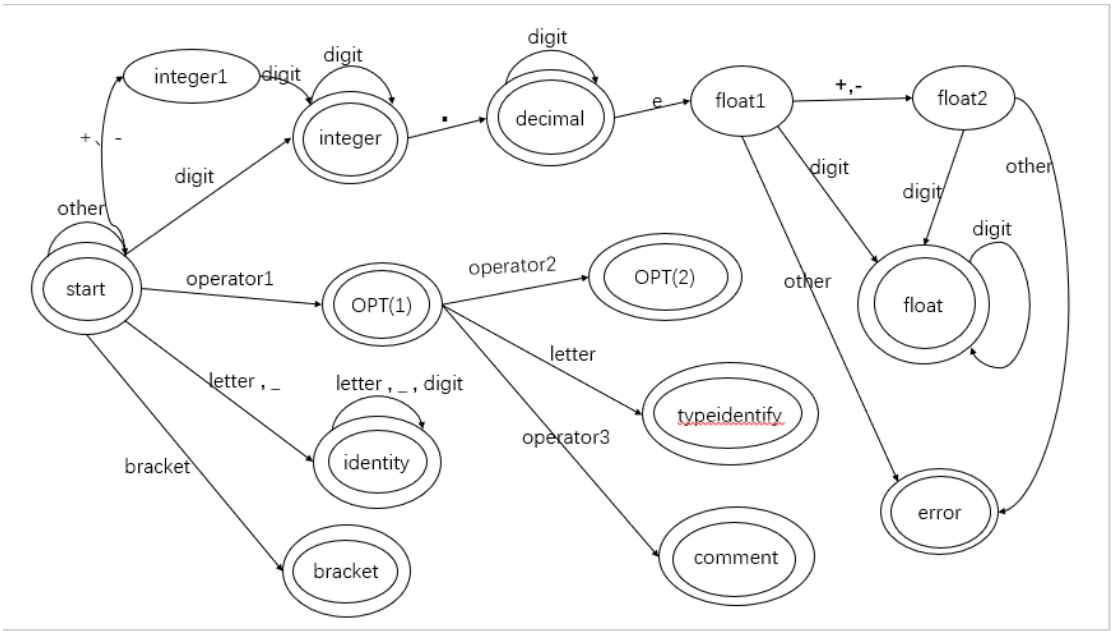
DFA:



小组成员信息：

何元梅　1033180311

毛萍兰　1033180310

陈智瑞　1033180309

孙程程　1033180308

分工：

资料收集：何元梅

代码编辑：何元梅、毛萍兰，陈智瑞，孙程程

DFA 编辑：何元梅、毛萍兰、陈智瑞、孙程程

总结心得：何元梅，毛萍兰，陈智瑞，孙程程

总结心得：

1. 何元梅

实验一开始是有点懵，粗略阅读了一下题目，了解到第一题的用 flex 做，第二题用简单 C++来完成，感觉第二题用之前的学过相关的 C++或者 C 的基础知识就能完成，只要考虑的足够详细。令人抓狂的第一题，首先我不清楚该咋哪儿写代码，因为下载的 flex 就是一块类似于 Dos 命令提示符的大黑块，网上还有教程安装 flex 得添加路径，然而我尝试多次并无用。其次不知道咋运行，初步猜测是直接将写完的文件在大黑块上输入命令运行，最后这个东西的语法该咋写，初步觉得写正则表达式。通过网上搜索，学习了几篇博客，终于搞清楚 flex 相关结构和语法，主要分为三个部分组成：定义、规则和用

户代码。定义位于%和%之间,运行 flex 文件时原样拷贝到 lex.yy.c 文件中；规则部分是由模式 pattern 和动作 action 组成,pattern 用正则表达式表示，含义是需要匹配得词的规则，action 用 C 代码表示，含义是成功匹配该词后执行得动作；所有的用户代码也会被拷贝到文件 lex.yy.c 中，在这里可以定义一些辅助函数和代码，供扫描器 yylex()调用，或者是调用扫描器（一般来说就是 main（）了），这部分是可有可无的，如果没有的话，Flex 文件中的第二个%%是可以省略的。搞清楚了这些基本知识，高高兴兴把代码按照自己的思路敲上，但是 debug 过程真的过于艰难，记事本写出的代码没法测试，一开始连.c 文件都没有生成，没法定位到底是哪段代码出现问题，只能把不需要测试的代码全给删了，就这样一条条地排除，可真是过于艰难。终于写完之后发现我的代码逻辑出现了错误，因为第三个测试样例没有通过，直接是对于输入的每个词语，匹配分析后之后输出，没进行存储，导致即使某个 float 类型的此输入是错误的，之前格式对的词仍然会被输出结果。于是重新设计函数把所有的结果都存储到一个字符串中，输入格式全对的话，最后再输出全部的结果，若是输入有格式错误就直接报错，不对字符串中的内容进行输出。在此探究过程中，需要将行数存入数组中，我一开始用的是 itoa 函数，在我电脑的 Visual Studio 平台上能正常运行，结果没错，倒是上传到 CG 平台上时 itoa 这个函数报错，通过网络搜索发现是这个函数的移植性较差，很多平台不支持 itoa 方法并建议用 sprintf 取代。

总之，通过这次学习了很多知识，提高自学以及探索的能力，学会 C 语言中的有些以前不太熟悉的函数的使用，比如 strcat，itoa，sprintf 等，通过书写正则表达式加深了对正则文法的理解。收获了很多知识，提高了自己的编程能力。

2. 毛萍兰

本次大作业题目整体来说还是比较简单的，老师给的两周时间是和够用的，但是做作业的过程还是又那么些许曲折。比如在自己电脑上 vs 能够正常运行，但在 cg 平台上出现编译错误导致得分为 0、某些测试样例一直通不过等问题，最终通过不断修改代码，和队员们一起沟通交流，上网查阅资料等方法最终还是把两个题目都写出来并且测试样例全部通过了。

这次作业最开始的任务其实是统计老师给出的测试样例中的所有字符、字符串里的所有类型集合，这个过程挺简单的就是有点废眼，统计完成后进行下一步的设计 DFA 了。

构造 DFA 这个部分我们就不同类型的数、字符、类型符等什么时候开始什么时候结束，经过什么样的过程等问题进行了比较久的分析讨论。一开始的时候想的是所有类型判断完毕以后共同走向一个 end 终态，但后续在理逻

辑的时候发现只要判断完单个字符或字符串的类型，那么这一过程就已经结束了，开始新一轮判断了，并且如果类型节点不为终态的话，那么走向 end 终态的过程应该是空，可是 DFA 不该有空闭包的出现，要确定化，所以最终所有的类型状态就作为终态了。DFA 构造过程中另一个讨论的点就是 float 类型的判断，e 后如果既没有正负号又没有数字的话格式就不正确了，直接报错，如果 e 后有正负号，但是正负号后面不接数字的话那么也会出现格式错误也该报错。根据这个逻辑设置了两个 float1、float2 节点，代码内也将这部分内容的逻辑充分体现了。单个字符表示运算符和两个字符表示一个运算符这部分内容也是挺纠结的，单个符号可以直接是 OPT 作为终态；单个符号加上一个符号并且二者组合起来在双个字符运算符集里的情况也可以作为 OPT 终态；单个字符加上字母，那么一定是%d&c 这种引导类型符；单个符号加另一符号且二者组合起来为注释集内的字符，则为注释集终态。上述是构造 DFA 过程中遇到的争议性问题，解决以后，根据 DFA 编写词法分析函数就变得比较简单了。

为了能够更方便后续的引用，我们编辑了一个元素是否在集合内的一个 isInGroup 函数，最开始是用 sizeof 函数来获得每个数组的大小的，但是测试样例通不过，sizeof 函数面对 char 型数组直接读的是 char 类型所占的字节数，无法读出数组内元素个数。strline 的逻辑也不正确，读的是字符串的长度，读不了字符数组的元素个数。最终选择了比较土的办法，直接数不同集合的元素个数，用常量传参获得数组大小，for 循环来判断彼时的字符或者字符串是否在集合内。另一个最初无法通过的测试样例是报错样例，题目要求的是促使出错以后所有的词法分析结果就不能给出了，只有错误信息内容，为了能够及时的终止进程，我们想到了在函数中只要 return 了一个值，那么函数就能够成功结束。因为错误信息只在 float 类型里有出现，所以我们把判断 float 数据类别的函数设置为布尔型，一旦报错及时 return 回到布尔型的 start 函数，这样就成功解决了报错信息的输出问题。运用 flex 构造词法分析器也遇到了这个问题，和手工构造词法分析器的解决方法不一样，我们用到了 sprintf 这个函数，把之前的内容覆盖，得到只输出报错信息的结果。

总的来说，经过这次构造词法分析器的大作业，我对编译器进行词法分析的过程有了更深刻的了解，对教材上的内容也加深了理解和体会，这次编程作业还是学到了很多东西的。其中给我最让我感受深刻的一点，不管做什么事情，一定要逻辑思维过程清晰准确，才能完成后续一系列的操作，思路理清以后所有事情才能有条不紊的进行下去。另外一个，细节决定成败，哪怕很小的一个格式错误也能一直无法通过测试样例，所以，在今后学习中，

更要养成细心的好习惯。

3. 陈智瑞

本次上机实验主要强调的是个人的逻辑能力和编写代码能力，通过两种不同的方式来实现简单的词法分析过程，这是我第一次尝试了解编译过程中编程语言针对输入的不同类型的字符串的辨别和分析，也使我更加充分并且更加深刻的理解了计算机代码的识别和编译过程。

通过参与手动构造词法分析器的过程，我遇见了很多难题以及很多意料之外的错误，攻克它们的经历也使我学会且巩固了很多知识，以下是一些汇总。

首先，是关于求数组长度的函数的使用问题。我们的思路是将不同类型的输入字符或字符串分类后分别存于一个数组中，方便后续针对键盘输入的文本中的字符进行对比和分类，而对比的过程将会用到 for 循环，循环的结束条件需要知道所对比数组的长度。我们纠结与 sizeof 函数和 strlen 函数的具体使用方法和选择，之后我们决定用 strlen 函数来求所有字符数组的长度，经过多次调试后最终得到理想的输出结果。但遗憾的是 CG 平台并不支持该函数的使用，我们只能退而求其次的增加了函数参数，将每个数组的长度直接传参比较。

再者，就是将 NFA 转变为 DFA 的过程以及具体的词法分析过程。我们分别设置了针对于不同类型输入字符的判断函数和具体处理过程，每次只对一个输入的字符进行处理。首先进入 start 函数，在 start 中分情况判断，然后分别传入到对应的函数中进行输出或者是否出错的处理，并且在分类过程中还需要引用到递归算法来保证在总体输入完毕后再判断字符串究竟属于哪一类，最大程度上做到识别准确。特别需要注意的是，根据测试样例的要求，如果出现错误，则需要终止输出，但是要输出具体的错误位置和错误字符串。为了达到能够终止输出的目的，我们采用了 bool 类型的函数来判断，通过其返回值为 false 来终止针对输入字符的下一次判断处理，再通过之前设定好的错误输出格式输出具体错误位置和错误信息。

最后就是代码初步完成后的不断优化和改正过程，我们通过对比绘制的 DFA 的逻辑思路逐个对每个函数进行检查和调试，本着简化代码的原则删除了一些不重要函数的引用，使整个代码看起来更加简洁明了，同时也提高了运行速度，实现了代码优化。

以上就是我本人针对本次编程实验的具体总结和心得收获，日后我将尽可能的将所学的知识应用到具体实践中，提高自身能力，取得更大的进步。

4. 孙程程

这次实验刚开始很懵，不知道每个题目代码如何下手，经过很多网上学

习和尝试最终得到了实验结果。

第一题用 flex 实现词法分析器，我首先先去了解了 flex 的工作原理，flex 的功能是根据 flex 源文件构造一个词法分析器。 Flex 源文件内容包括定义和辅助函数、识别规则三部分。了解之后发现 flex 用法也很简单，我们采取直接长按鼠标左键将用 flex 文法编写的程序拖进 flex.exe，通过 flex 编译后会生成一个 lex.yy.c 文件可以直接用 C 语言编写工具打开编译运行。代码成功写完后，还是遇到了很多问题，因为从记事本写代码直接放到 flex 运行无法得知到底是哪块代码出错，我们采取了最笨的方法将代码一行一行分别注释掉最终发现是因为多打了一个空格导致无法编译成功，这其中还包括几个单词拼写错误，这说明我们在写代码的时候还是非常粗心的，导致了很多意料之外的错误，浪费很多时间，以后一定要注意。遇到最大的错误是第三个测试样例始终无法通过，在识别出 float 型数据出现词法分析错误的时候，输出错误之前把之前每个词法分析正确的语句也输出出来了，这个地方卡了很久，我们尝试采取建立一个字符串存储输出的语句，如果报错只输出报错信息，不报错则输出全部输出语句。在我们电脑上可以成功运行，结果在 cg 平台不能通过编译。最终经过网上参考采取 sprintf 方法给动态数组赋值成功输出正确结果。第二道题手工构造词法分析器，我们采取比较熟悉的 C++环境，首先建立 DFA 分别考虑标识符、数据等字符串的各种组成形式，然后根据测试样例我们建立相应的字符及字符串数组储存各种符号，采取调用函数嵌套与递归的方法去完成对输入数据中每个字符串的识别及词法分析。这个题目我们同样遇到了第三个测试案例输出错误的情况，经过第一个题目我们很快找到原因是词法分析报错时没有让函数返回错误终止函数，修改后得到了正确输出。

这次的实验让我更加深刻的了解了 NFA 转化 DFA 的过程，及正则文法与 DFA、NFA 的关系。希望以后的实验也能帮我对编译原理的理论知识有更深的理解。