老师好，我是来自2022级计算机科学与技术3班的林江荣，学号是20212821028，接下来将由我来进行编译原理课程项目的汇报。

(下一页PPT)

我将从项目任务一、项目任务二、项目任务三以及最后的项目总结四个方面进行汇报。

(下一页PPT)

首先是项目任务一

(下一页PPT)

内容主要包括以下6个部分，分别是正则表达式的预处理、正则表达式转NFA、NFA转DFA、DFA最小化、词法分析源程序生成以及最后的词法分析。

(下一页PPT)

首先是正则表达式的预处理部分，我的预处理步骤主要有中括号运算符的替换、正则表达式替换以及添加连接运算符这3个步骤，下面以一个例子来讲解预处理操作(点一下PPT)

对于这两条正则表达式，第一步中括号运算符会被小括号和选择运算符替换，即[0-9]会被展开，如PPT所示，[1-9]同样道理；第二步会进行正则表达式的替换，即查看是否有某个正则表达式的内容存在另一个正则表达式的命名，有则替换，在本例中，number正则表达式内容存在digit，因此会将digit替换为其内容；最后一步是添加省略的连接运算符，比如本例中number正则表达式内容括号与括号之间的连接符被省略，会在这一步预处理中添加。以上就是正则表达式预处理的步骤。

(下一页PPT)

接着对于正则表达式转NFA，采用的是Thompson方法，该方法可以通过类似计算算术表达式的方式，“计算”出正则表达式的NFA结果。Thompson方法对于不同运算的操作如图所示：对于基本的正则表达式r直接构建一个新的初态和终态，设置初态经过r转移到终态，这就是NFA的一个基本单元；而对于连接运算符只需要添加前一个NFA基本单元的终态到后一个基本单元的初态的ε转移即可，对于选择、闭包、正闭包以及可选运算的具体操作该图也清晰展示了；（点一下PPT）

之后为了实现简便地计算出正则表达式的NFA结果，我采用了先将正则表达式转为对应后缀表达式的形式，之后按照后缀表达式的计算方法结合Thompson最终“计算”出每个正则表达式对应的NFA结果。

(下一页PPT)

下一步将NFA转为DFA，下面将以左侧这个NFA为例子讲解如何实现到DFA的转换：首先求出NFA初态的ε闭包，（点一下PPT）即求出NFA初态经过ε能到达哪些状态，这些状态的集合就构成DFA的初态（点一下PPT），之后找DFA当前未被处理的状态，查看该状态通过各个转移能到达的NFA状态集合，比如对于该DFA状态，经过r转移能转移到的NFA状态集合为3（点一下PPT），但此时还需要求该状态集合的ε闭包才能得到完整的DFA状态，最终为{3,5}（点两下PPT），同理可以得到DFA初态经过s转移的NFA状态集合（点一下PPT），同样要求对应的ε闭包（点一下PPT）。之后对未处理过的DFA状态重复上述转移构造操作，直至所有DFA状态都被处理过为止，即整个DFA稳定，说明NFA转DFA完毕。（点一下PPT）这是NFA转DFA的具体流程图。

(下一页PPT)

接下来是实现各个DFA的最小化，首先要对DFA状态按照是否为终态进行划分，之后将非空的初始划分集合放入队列，记录初始队列长度length；接着进入划分集合的循环，每次循环先读取队头集合，依据各操作数的转移来划分该集合，能划分出多个集合时就取出队头集合并置入各划分集合到队尾，否则取出再重新置入队尾；每次处理完一个队头集合用计数器记录，当计数器达到length即一轮划分结束，此时比较本轮划分前后队列长度，相等则划分集合循环结束，不等就继续循环划分。循环结束后队列中各个集合即为对应的最小化DFA状态，通过原始DFA状态转移，构造最小化DFA的转移。

(下一页PPT)

有了各个正则表达式的最小化DFA后，即可生成对应的词法分词源程序。首先生成必要的全局变量，其中scr\_file代表待分析的源程序输入文件流对象，res\_file代表保存词法分析结果的输出文件流对象，read\_pos代表输入文件流的文件指针，还有其他必要的变量；（点一下PPT）接着生成一个跳过空白字符的函数；（点一下PPT）之后通过讲稿中的第二种方法为每一个最小化DFA生成一个匹配函数便于在main函数中调用；对于main函数的生成，通过循环读取文件流的字符，每轮循环开始先清空上轮匹配成功的临时结果变量tok\_suc和buf\_suc（首轮也清），调用各个match函数匹配（先关键字DFA匹配，如果有的话），失败则给buf\_err赋值当前buffer内容，成功且长度大于本轮此前最大匹配成功长度则更新buf\_suc和tok\_suc，每次调用match匹配完清空buffer，重置文件流读取位置到匹配前的位置；若本轮匹配最终未匹配到任何内容，输出错误信息到buf\_err并退出程序；否则输出匹配成功的类型标识及字符内容，更新 read\_pos，进入下一轮循环。

(下一页PPT)

项目任务一的最后需要利用上述生成的源程序进行词法分析，对此我的步骤是先即时编译上述生成的源程序得到可执行文件——分词器，之后运行该可执行文件读取用户上传的源程序进行分析，最后保存分词结果到文件并展示给用户。

到此项目任务一的相关功能实现就汇报完毕，接下来我将利用tiny语言的正则表达式对项目任务一的功能进行演示。

（演示项目任务一）

（从项目任务二的标题页开始）

接下来将对项目任务二的内容进行汇报。

(下一页PPT)

内容主要包括以下6个部分，分别是文法规则预处理、求解非终结符的FIRST/FOLLOW集合，构建LR(1)DFA、LALR(1)DFA、LALR(1)分析表以及最后的源程序语法分析。

(下一页PPT)

首先是文法规则的预处理，主要有以下四个步骤，其中主要的就是如果输入的第一条文法规则存在或运算符则需要进行文法扩充，另外对于文法规则的存储结构如第3条步骤最后所示，（点一下PPT）这是具体的示意图。

(下一页PPT)

接着对于求解FIRST集合和FOLLOW集合，我将分开阐述，首先是求解FIRST集合的方法，由于讲稿中的方法不能处理左递归情况下的FIRST集合，因此这里采用迭代法来求解非终结符的FIRST集合，这是求解FIRST集合算法的伪代码：首先初始化所有非终结符对应的FIRST集合为空，之后只要有非终结符的first 集合变化就循环：即对文法规则所有产生式，获其左侧非终结符及右侧符号列表，遍历列表，依其是终结符或非终结符确定对应的first集合，将该集合去掉ε后加入左侧非终结符的 first 集合，若该集合无ε则结束内层循环，若能够遍历完列表且最后符号的FIRST集合存在ε就给左侧非终结符的first集合加ε。

(下一页PPT)

对于FOLLOW集合的求解按照讲稿中的迭代算法执行，这里就不过多赘述。

(下一页PPT)

接下来是依据输入的文法规则构建LR(1)分析的DFA，为了构造LR(1)的DFA需要用到两个算法，其一是闭包构造算法，这个算法用于用于扩展一个项目集，直到它包含所有可能从当前状态推导出来的项目，即通过给定的包含初始Item的 State，构建出完整的State，这是算法的伪代码描述；其二是转移构造算法，这个算法用于确定当前项目集I遇到特定输入符号X时，应该转移到的项目集状态，下面同样是其伪代码描述。通过这两个算法即可构建出LR(1)的DFA，具体流程如图所示（点一下PPT）。首先构造DFA初态，先构建代表首个文法规则的 Item，将其向前看字符集合设为{$}，再将此Item放入State形成初态，并调用CLOSURE函数完善初态；然后利用DFS构建完整的LR (1)的DFA，定义接受状态参数的匿名深度优先搜索函数，在函数里收集父状态的移进项，基于移进项构建子状态并处理：对每个移进项，先通过GOTO函数构造新状态；若状态为空则跳过，若已存在于LR (1)的DFA中则记录转移关系，若为新状态则添加到DFA并记录关系后以其为新父状态递归调用DFS函数继续探索。通过从DFA初态开始调用DFS函数进行深度优先搜索以完成新状态构建、状态信息完善及状态间转移关系确定，最终构建出完整的LR (1)的DFA。

(下一页PPT)

下一块内容是构建出LALR(1)的DFA，主要是基于并查集的思想，首先进行一定的初始化，之后遍历LR(1)的DFA所有状态对判断是否为同心集以进行并查集的合并操作，最后基于并查集的结构构建LALR(1)的DFA状态，并依据LR(1)DFA的转移构建LALR(1)的转移，即可构建出LALR(1)的DFA。

(下一页PPT)

之后依据LALR(1)的DFA即可构建出LALR(1)的分析表，但是由于输入文法可能不是LALR(1)文法，即LALR(1)的DFA图可能存在冲突，因此构建分析表之前需要判断LALR(1)的DFA存在冲突。（点一下PPT）这是具体的两种冲突判断方法，如果存在归约-规约冲突，说明输入文法不是LALR(1)文法，无法构建分析表；如果存在移进-归约冲突，虽然文法也不是LALR(1)文法，但是可按照最长串匹配原则取移进操作来构建分析表。（点一下PPT）对于如图所示分析表的存储，我采用了图下所示的存储结构，其中的Cell单元格有三种类型，分别代表移进、归约和接受输入操作。（点一下PPT）基于该存储结构即可按照这三个步骤构建分析表，其中先处理归约后处理移进目的是使用解决可能存在移进-规约冲突的情况。

(下一页PPT)

有了LALR(1)的分析表即可对对应文法的源程序进行语法分析。源程序语法分析要先基于词法分析获取单词编码文件，然后据此及LALR (1)分析表展开。首先进行初始化准备，建立分析栈并压入DFA初态编号，读取词法分析结果作输入流且末尾添终止符$。接着进入循环匹配过程，每次循环读取输入流首单词与分析栈顶状态编号并查看分析表对应单元格，若无内容先考虑EPSILON转移，仍无则表明有语法错误、匹配结束；若有内容，单元格类型为移进就push移进状态编号进分析栈进入下一循环（EPSILON转移时要重放读取的输入到输入流），单元格类型为规约则找规约规则、从分析栈按规定弹出元素，不足则有语法错误，再依栈顶状态经规约符号的转移重复判断对应操作，若单元格类型为接受输入则语法分析结束且源程序无语法错误，每次循环结束若有规约还需将本次循环读取的输入放回输入流中。

(下一页PPT)

最后一块内容是关于源程序语法树的构建，其主要基于前文语法分析过程。因不同类型语句语法树子树构建及不同语言语句类型均有差异，为方便构建，语法树构建前需先对源程序对应的各条文法规则编码。（点一下PPT）以tiny的两条文法规则为例，编码文件中每条文法规则下一行是其规则编码，编码数字个数与该文法规则右部符号个数相同且一一对应（点三下PPT）。具体的编码含义如PPT所示，比如对于stmt-sequence文法规则的右部，stmt-sequence编码为1说明对于这条文法规则的语法树来说，stmt-sequence是根节点（点一下PPT），而statement编码为3说明其是语法树根节点的兄弟节点（点一下PPT）；又比如repeat-stmt右部规则的exp编码为2说明其是repeat子句语法树根节点的子节点（点三下PPT），以此类推，按照上述规则即可完成语法树编码。

(下一页PPT)

完成文法规则的语法树编码后即可按照PPT所示步骤构建源程序的语法树，首先初始化节点栈以及语法树根节点，之后按照语法分析过程中遇到的移进、归约、接受输入行为执行对应操作，最后在语法分析结束后，源程序的语法树即构建完毕。

以上就是项目任务二的相关内容汇报，接下来我将以tiny的文法规则进行系统项目任务二功能的演示。

（演示项目任务二）

（从项目任务三的标题页开始）

对于项目任务三主要是对于本系统在mini-c语言上的可用性进行测试，下面我将以mini-c的单词正则表达式以及文法规则进行系统的可用性测试。

（演示项目任务三）

（从项目总结的标题页开始）

到此三个项目任务的汇报完毕，接下来我将对项目进行一定的总结

(下一页PPT)

首先项目任务一完成从正则表达式分析到生成词法分析源程序，实现自动分词，并用 tiny 的单词正则表达式测试；任务二完成文法规则分析及源程序语法分析（含过程展示与语法树构建），用 tiny 的文法规则测试，其中选做部分由于时间问题仅在文档中进行一定描述并未有代码实现，因此PPT暂未演示；最后任务三针对mini-c语言的相关内容进行上述功能测试，确保系统重用性。

到此我的课程项目讲解完毕，谢谢老师观看。