|  |
| --- |
| 编译原理课程设计  实验报告  **指导教师：刘磊**  **年 级： 2018级**  **班 级：26班**  **小组编号： 1**  **组长学号姓名：陈建康**  **组员学号姓名：陈杰**  **组员学号姓名：王禹谦**  **2021年 4 月 14 日**  **计算机科学与技术学院** |

|  |  |  |
| --- | --- | --- |
| **完成实验内容** | | |
| 1. SNL的词法分析程序，输入SNL源程序，将单词的内部表示序列输出到stdout中和文件中。 2. SNL的递归下降分析程序，通过从第一个任务生成的token序列中读取Token序列，进而通过语法错误检查看是否有语法错误，如果有输出语法错误的错误提示信息，如果没有语法错误则打印语法树（层次文本的输出） 3. SNL的LL（1）分析程序，通过从生成的保存Token序列的链表中读取生成的token序列进行LL1分析方法的语法错误检查信息和语法树的打印 4. SNL的语义分析程序，通过利用第二个程序（第三个程序也可）所生成的语法树进行语义错误信息的输出如果没有语义错误则提示无语义错误，通过在分析语义的时候建立符号表来协助语义分析的错误检查，在扫描声明部分语法树时建立符号表在扫描语句部分语法树时查表检查语义 5. SNL语义分析完成后打印相应的符号表信息，通过逐层打印的方式输出到stdout中以验证所生成的语义分析过程中符号表的正确性。 | | |
| **小组成员任务完成情况** | | |
| 姓名 | 具体完成任务 | 工作量  百分比 |
| 陈建康 | 1、词法分析2、语法分析中递归下降3、语义分析4、符号表打印5、输出界面的设计6、报告的撰写 |  |
| 陈杰 |  |  |
| 王禹谦 |  |  |
| **小组成员协作情况** | | |
|  | | |
| **实验平台与编程语言** | | |
| **实验平台：Microsoft Visual Studio2017**  **编程语言：C语言** | | |
| **实验方案设计** | | |
| 1. 词法分析   源程序一般为字符串形式，词法分析是在单词一级上进行的，所以我们首先应该先将源程序翻译成单词序列也就是Token表，词法分析按顺序进行扫描源程序，根据源程序的此法规则输出与其等价的Token序列。  首先想要分析单词序列就必须对单词序列进行分类，一般可以分为保留字如 main，break等；标识符如变量名，常量名等；常数如整型常数，浮点型常数等；运算符如+，-，\*，/等；界限符如‘，’，‘；’，‘（）’等；保留字和表示符都是字母开头够跟若干个字母或数字构成，因此保留字是表示符的子类，故当遇到这样的情况先查保留字表，之后查表示符表，对于保留字表程序中利用WordType进行枚举。  而单词的Token形式在程序中利用一个结构体TokenType进行设置，其中包括单词所在源程序中的行号lineshow，单词的词法信息WordType（类型枚举）和单词的语义信息用输出Mean来进行表示。  DFA：  单词的正则表达式形式可以用有限自动机来进行实现，而对于识别单词的有限自动机来讲，根据不同单词的构成规则进行构造相应的状态转换图，而对于如果能识别所有单词的有限自动机来讲，首先将初始状态合并为一个唯一的初态然后花间调整状态冲突和对冲突转台重新编号。  在程序中是利用先利用一个链表进行相应存储每个单词的Token序列，这个链表是一个结构体ChainNodeType，其中属性包括单词的Token形式Token和指向链表下一个单词的指针nextToken，在利用函数getToken取得逐个单词函数来进行相应的链表节点的赋值，在这里面就用到DFA来识别单词进行链表节点Token的输入，首先定义自动机的各个状态以进行识别单词，分为开始状态（刚开始识别），完成状态（识别结束），数字状态（第一个字符为数字），标识符状态（第一个字符为字模），赋值状态（=），数组下标界限状态（.），字符标志状态（‘），注释状态（{}）。  其识别过程如下：只要不是文件结束标志即当前的单词不是ENDFILE，就一直进行确定单词Token类型加入到链表中，对于每一个单词设置一个数组tokenword用来存储当前正在识别的单词，在识别过程中设置存储标志save，如果为1则说明当前字符要存入当前的tokenword中，反之则不能存入（不需要存入的单词），对于遇到识别的每一个单词自动机状态初始为start，之后根据遇到不同字符转换不同状态来进行，当遇到不需存储的字符则直接令save为0即可，对于空格，回车等也无需存储。在其中要灵活进行回退处理，例如当扫描到‘：’时自动机状态改为赋值状态则只有下一个字符为‘=’时才可令当前字符的WordType为赋值，否则回退一个字符，令当前所读的单词类别为设置为ERROR进行处理。  当state为标识符状态时识别标识符后还要调用checkbaoliu函数检查是否是保留还是自定义的标识符，按情况输出，标识符输出ID，而保留字则输出相应的KeyWords，在整数状态识别并组合数字，输出单词number，对于扫描到的=或：=如果扫描到=时向前回退一个单词进行识别如果是：则输出：=否则输出=；在注释状态略过注释内容知道扫描到}，或是EOF并不生成单词；当扫描到数组下标界限，如果下一个字符不是‘.’则为程序结束标志，否则为数组下表界限标志，在字符标志中如果读入的下一个是字母或数字则继续读下一个字符，如果下一个字符是“ ’ ”，则为字符标志其他均出错。在错误状态表示词法分析程序从源程序读入了一个不合法的字符打印错误信息ERROR，略过错误单词转开始状态继续识别和组合下一个单词符号。  在识别过程中用到了取下一个字符和向前回退一个字符函数，这两个函数在程序中分别用getnext（）和disgetnext（）来进行操作。  在输出界面中为了保存单词的行号等信息，在取字符的同时也将源程序及其行号打印在了stdout中，实现机制如下：  通过没读一个字符，调用getnext（）函数，首先定义全局变量lineno来记录当前处理的字符的行号，通过每次从源文件中读取255个字符存于line数组中进行操作，一行的最大字符数这里约定源程序每行字符数不能超过255个，在没结束前没取出255个单词就将line里的信息连同行号一并打印在stdout中，然后用哪个linelarge记录读入的行中世纪的字符个数然后每读一个返回line【Line++】即可，当Line小于linelarge时说明该行还未处理完处理该行字符，否则lineno++循环处理。  而disgetnext则是通过Line--实现单词回退。  在存取单词的时候如果自动机状态处于完成状态，则判断其是否为表示符，是的话在判断是否为保留字，通过checkbaoliu（char\*a）来实现，在立即定义的保留字中查询是否匹配若是返回保留字的单词否则返回ID表示自定义标识符，其中保留字存储在自己定义的baoliu【】结构体数组中，属性有单词和类别。之后将每个取出的单词的行号，词法和语义存入链表节点中去，知道当前所读单词为文件结束符。  这样整个程序的单词Token就存入了链表（链表头为chainHead）中了。  为了以后的语法分析能够直接从链表中读取单词的Token序列则利用了自定义函数ChainToFile（ChainNode\*chainhead）将链表中的信息存入到了文件Tokenlist中方便一以后语法分析用。  而ChainToFile（）函数实现如下：  传入Token链表首指针，然后创建新文件tokenlist将链表中的内容进行fwrite写入文件中。  为了使得在stdout中显示Token结构，利用printTokenlist（）函数进行词法分析Token序列的输出打印：通过在上一步文件中的Token序列进行扫描，打印行号，遇到保留字单词输出Key Words再打印语义信息，其他如等号小于等直接输出符号了，对于数字则输出Number，val=，字符是Inchar，char=，自定义单词是Identifier，错误就输出Error，未知单词就打印Unkonwn token，在搜索单词是每次搜寻tokenlength长度的单词，这样才可以取得每一个单词。   1. 语法分析（递归下降法）   语法分析的输入是从词法分析器输出的源程序的Token形式，在这里是存放在了ChainNode链表当中，也可在词法分析中生成的文件Tokenlist中按一个token大小的字节进行读取获得。之后根据语法的文法规则进行分析处理，如果分析有错误则输出错误信息，如果无错误则输出语法的语法树形式。  这里采用从开始符出发进行推导，递归下降分析程序，针对递归下降方法，我们是通过文法的predict集合进行推导，看源程序的文法是否是正确的文法。  在主程序分析语法时要每次读取单词的Token序列，而在程序中使用ReadNextToken来实现读取文件中的Token序列，利用fseek函数越过整数倍token大小的字节后读取相应的Token单词。  输出的话是输出与源程序结构相同的语法树，为了呈现的比较清晰，又加入了类型声明，变量声明，函数声明，语句序列等标志节点。  而语法树的数据结构如下：  语法树的节点类型程序中使用TreeTypeKind来表示，其中分为两类，其一是标志节点，有树根的Pro，程序头ProHead，子结点都是类型声明节点Type，子结点都是变量声明节点Var，函数声明节点FuncDec，语句序列节点StmL，这些只表示节点的类型没有其他的具体内容，而另一个为具体节点，有语句声明节点Stmt，表达式节点Exp，声明节点Dec。其中声明节点DeclareKind可分为数组Array，字符Char，整数Integer等，语句类型节点SentenceKind枚举有If，While,Assign等，表达式类型节点ExceptionKind枚举有Op操作符，常数Const以及标识符变量类型VarType。在语法树中记录节点其他属性，其中分为数组，过程以及表达式，而在过程总要记录过程的参数类型ParamType枚举为值参Val和变参Varpa，在表达式属性中有变量的类别VarKind枚举为标识符Identifier，数组成员ArrayMember和域成员类型FieldMember以及表达式的结果类型ExpKind枚举为空Void，整形INTEger，布尔类型Boolean。  而语法树节点的数据结构如下：treechild【0..2】指向子语法树，brother指向兄弟语法树指针，noline记录源代码的行号，nodekind为节点类型，域类型kind记录语法树节点的具体类型，samenum记录相同类型的变量个数，name标识符的名称，table为指针数组成员为各个标志符在符号表中的入口，属性结构体Figuer中由数组属性包括数组上下界和数组子类型，只有当kind中的声明类型为Array时有效，过程属性中的过程参数类型，，表达式属性包括表达式操作符，表达式的值以及变量的类别和检查类型。在进行语法分析时语法分析程序将根据SNL的文法产生式，为相应的非终极符创建一个语法树节点并为之赋值。  递归下降法：即对每个非终极符按其产生式结构构造相应语法分析子程序，其中终极符产生匹配命令，即在程序中调用match函数，而对于非终极符则产生过程调用命令，其中match函数检查当前输入符是否是产生式所要求的单词，如果是则指向下一个单词，如果不是则出错，输出错误信息提示。其中match函数将终极符单词作为函数参数，当前在文件中索取单词与函数参数比较如果一致则取下一个单词，否则打印未匹配的错误信息，继续读下一个单词。  在语法分析中在main函数中利用语法分析函数parse函数来进行递归调用，parse函数读入单词调用总程序处理分析函数返回语法树的根节点，当递归回溯完成后如果当前的token不是文件结束符则说明文件提前结束输出错误信息，否则返回树根t为后面的打印语法树提供语法树的树根TreeNode\*t；  而总程序处理分析函数Program内主要生成三个儿子节点treechild[0..2],依次是程序头分析程序ProgramHead（），程序声明部分分析函数DeclarePart（），以及函数体部分处理函数ProcBody（），然后匹配程序结束标志“.”，若成功返回树根结点，否则返回空。  （1）程序头ProgramHead主要是新建程序头结点，匹配program，然后记录程序名于程序头结点，匹配ID，如果成功返回节点否则返回空。声明部分处理函数因为声明语句的文法有类型声明节点，变量声明节点和函数声明节点，并且彼此之间都是作为brother节点的关系，所以先处理类型声明，新建类型节点，调用类型声明处理分析函数TypeDec（），变量声明处理分析函数VarDec()，函数声明部分处理分析函数ProDec（），其中TypeDec生成的接待为类型声明的第一个子结点，VarDec（）为变量声明的第一个子结点。如果没有类型声明节点则令声明部分指针指向变量声明节点依次类推，如果都没有函数返回NULL；函数体分析函数则作为树根的第三个儿子节点，其调用程序体部分分析函数ProgramBody，根据其相应的文法生成相应的递归函数，新建语句标志类型节点，匹配begin，然后调用语句序列函数StmList（）作为语句标志类型节点的第一个儿子节点。  最后匹配END。  关于新建节点函数虽然写了很多函数但是内部都是大同小异，只拿其中一个来讲，如创建函数类型语法树节点函数newProcNode（），其作用是为语法树创建一个新的函数类型节点，并将语法树成员初始化：首先申请树节点，对于树节点的三个儿子节点初始化为空，兄弟节点也为空，节点类型为函数类型FuncDec，源代码行号为linelinshi，这个是没匹配一个Token时将此Token的行号赋值为linelinshi，相同变量个数为0，标识符名称为空，符号表地址为空。其他新建类型标志节点，新建变量声明标志节点等都一样，只是将里面的nodekind设置为相应的标志节点类型即可。注意在新建语句节点，表达式节点时除了赋值nodekind为相应的类型，还要将具体类型赋值。  （2）之后则从语法树从左往右进行分析，对于声明标志节点中的类型声明处理函数TypeDec根据读入的下一个单词选择调用typeDeclaration（）函数，如果读到的是Type则调用，否则不做动作根据文法而定。  而对于typeDeclaration其根据文法产生式匹配Type调用TypeDecList（）函数，根据TypeDecList函数的文法产生式，创建新的声明类型节点。调用TypeId()，匹配保留字EQ（=），调用TypeName（），匹配SEMI（；），之后调用TypeDecMore（），返回值给声明类型节点作为其兄弟节点。  其中TypeId（TreeNode\*t）是根据读入的单词判断其是否在标识符中，将该字符的语义拷贝到参数t的标识符名字中，之后匹配单词ID。  TypeName（TreeNode\*t）根据文法产生式和predict集进行选择调用。当匹配到整型和字符型调用基础类型函数BaseType，当单词为数组或记录类型则调用结构体类型函数StructureType进行赋值，BaseType根据文法产生式进行匹配赋值。  而StructureType则是选择数组类型还是记录类型，前者对读入单词匹配，记录数组上下界数值再匹配保留字最后调用基本函数BaseType记录数组子类型。记录类型RecType则匹配调用与函数FieldDecList（），最后对读入单词匹配FieldDecList根据文法和predict集进行调用函数，判断记录里的单词的类型进行相应的调用，在域中成员互为兄弟节点。  对于TypeDecMore则根据读入的单词或和文法产生式以及predice集进行调用相应的函数。并将该返回的节点作为TypeDecList的兄弟节点。最后FiledDecMore函数是读入单词判断若为END则不做处理否则若为基本类型或数组则递归处理FiledDecList，否则读入下一个token。域中在调用完基本类型等后有IdList函数其是匹配标识符记录标识符名称的，之后递归到用IdMore处理相应函数。  之后是变量声明标志节点VarDec（），根据产生式和读入单词的Predict集调用相应的函数，其实与上面的类型声明类似，如果相应的predict集对应的文法不为空则调用相应的处理函数，如果为Var时调动VarDeclaration（），匹配Var，调用VarDecList（）：创建爱你新的声明节点，根据文法产生式调用typeName，varIdList，VarDecMore，之后根基读到的不同的单词调用varIdList（），之后匹配“；”，之后创建节点VarDecMore（）将其作为新声明节点的兄弟节点。  其中varIdList（）匹配标识符名记录标识符名称如果当前单词为ID，则将当前的标识符名赋值到参数t的成员t.name中匹配id，调用VarIdMore（），根据文法产生式及predict集确定执行哪个分支（是否还有其他变量），队规调用varIdList（）。  最后是过程声明标志节点ProDec（），根据产生式对当前单词进行predict集的判断若为begin则不做动作，若为procedure则调用ProcDeclaration（t），否则读入下一个单词。  对于ProcDeclaration（），创建新的过程节点，匹配Program，匹配标识符名字，之后匹配左括号，调用参数处理函数ParamList（t），匹配右括号，匹配“；”，之后反复调用类型和过程体节点作为其第二个和第三个儿子节点，实现函数的嵌套。  其中参数声明的处理函数ParamList（）根据文法产生式和predict集当匹配为右括号则直接返回，否则调用ParamDecList()并且将该函数返回的节点作为参数处理节点的儿子节点。  对于ParamDecList（）根据文法产生式调用Param和Parammore函数返回节点作为Param的兄弟节点：对于Parmmore（）根据文法产生式根据当前读入的单词如果为右括号则结束否则当前单词是否为“；”，若是则匹配调用ParamDecList（）函数递归调用以看是否有其他的参数声明。  对于param函数则根据产生式选择调用分值函数，令当前节点的参数成员赋值为参数类型，调用TypeName（t）和FormList（t），否则如果当前单词为Var则匹配单词var，t参数成员赋值为参数类型调用调用TypeName（t）和FormList（t）；对于FormList（t），匹配id，执行FidMore（t）函数，FidMore根据产生式要么不做动作要么递归调用FormList（）；  （3）之后是过程提部分处理函数ProcBody（），调用ProgramBody（）主程序部分处理函数，建立新的句子标志节点，匹配begin，之后令语句部分处理函数StmList（）作为新的句子标志结点的儿子节点，最后匹配end。  对于语句部分处理函数StmList（），根据产生式调用stm和stmmore处理函数并将后者作为前者的兄弟节点，对于stm根据不同的predict集选择不同的分支，如若当前单词为IF，则调用条件语句处理函数ConditonalStm（），赋值或调用语句特殊，如果当前单词为ID，则应先匹配ID，在调用函数AssCall。  赋值语句和函数调用语句的开始部分都是标识符，所以该函数根据读入的单词选择调用相应的处理程序：当前单词为ASSIGN则调用AssignmentRest（）函数，当前单词为LPAREN则调用CallStmRest（）函数。  针对AssignmentRest（）来讲根据文法生成式创建新的赋值语句类型语法树节点，赋值语句节点的第一个儿子节点记录赋值语句的左侧变量名，第二个儿子节点记录赋值语句的右侧表达式，令第一个儿子节点为变量表达式节点，行号赋值为当前所读Token的行号，将当前所读单词的名字赋值为节点的标识符名字，然后调用变量处理函数VariMore（t），判断下一个单词如果是左中括号则匹配左中括号，表达式结点的第一个儿子节点调用表达式函数Exception（），并将根节点的表达式属性的变量书信该设置为Identifier，匹配有中括号结束，如果当前单词为DOT，则匹配DOT处理域变量函数fieldvar（），让第一个儿子指向域成员节点，令根的表达式属性的变量类型设置为域成员类型，域成员变量节点的表达式属性的变量类型设置为Identifier；否则读下一个TOKEN。  对于域变量处理函数FiledVar（），根据产生式先建立新的变量表达式节点，然后匹配id，赋值标识符的名字，递归调用FieldVarMore（t），该函数根据产生式调用相应的递归处理域变量为数组类型的情况，如果匹配到左括号，则t的第一个儿子节点等于Exception（），然后属性的表达式类型属性为数组类型之后匹配右中括号。否则读取下一个单词。对于表达式处理函数Exception（），调用简单表达式处理函数simple\_exp()，返回语法树节点t，当前单词为小于或等于，则创建一个新的运算表达式类型语法树节点p，将简单表达式处理函数的返回值t赋给新语法树节点p的第一个儿子节点作为运算表达式的做运算简式，将当前单词赋给新语法树节点p成员op，作为运算表达式的运算符，并将新的语法树节点p赋给t作为当前语法树节点。匹配小于或等于，调用simple\_exp()将返回的值赋给当前语法树二儿子，作为运算表达式的右运算简式。  而simple\_exp()根据产生式调用相应递归处理函数，调用原处理函数term（）返回t，如果当前单词token为加法运算符或减法则创建新的操作符表达式类型语法树节点p初始化第一个子结点p的op为当前所读单词，将函数返回值t赋成语法树节点p，当前单词token与指定加法运算单词匹配，调用原处理函数term（）返回语法树节点指针给t的第二子节点。  其中用到的项处理函数term（），调用因子处理函数factor（），返回给t，如果当前单词是乘或除，创建新的操作符表达式类型语法树节点，赋值为p，与上面加减处理一样，之后将匹配乘法运算符单词，第二个孩子节点也为factor（）；  则factor（）因子处理函数，读当前单词，如果为常整型，则创建新的常量表达式语法树节点赋值为t，当前单词为数字，则将当前单词名装换为整数并赋给语法树节点t的数值成员，匹配常整型。如果当前单词token为表示符单词ID，则创建新的标识符表达式类型语法树节点t=variable（），如果当前单词为左括号，则匹配左括号，调用表达式处理函数Exception（），犯规语法树节点指针给t，匹配右表达式。否则读取下一个单词。  其中的变量处理函数variable（）：根据产生式处理变量，建立新的变量表达式节点返回给t，然后将当前单词的语义赋值为t的标识符名字，之后匹配id，递归调用VariMore函数（）；这是所有关于赋值和调用的函数处理。  之后是循环语句部分的处理函数LoopStm（）：根据文法产生式，新建while语句类型节点，匹配WHILE关键字，令节点第一个儿子节点指向表达式Exception（），匹配DO，之后第二个儿子节点指向语句处理函数StmList（），匹配ENDWH关键字，结束返回while语句节点。  条件语句部分处理函数ConditionalStm（），新建IF语句类型节点，匹配IF，小儿子指向Exception（），匹配THEN，之后二儿子指向then之后的语句序列StmList（），如果下一个单词为else则匹配else之后令第三个儿子节点指向else后的语句序列StmList（），否则直接匹配FI返回IF语句节点t；  函数调用语句部分的处理函数CallStmRest（），建立Call类型语法树节点给t，子结点指向实参，匹配保留字LPAREN，函数名的节点用变量表达式的节点，将当前所读的函数名给t的第一个儿子节点的name，调用实参处理函数ActParamList（），将实参处理函数的返回值给t的第二个儿子节点，匹配RPAREN。  赋予实参处理函数根据文法生成以及相应的predict集，判断当前的单词，如果为右括号则break，如果为ID或为常整数则调用表达式处理函数，如果有的话则调用更多实参处理函数ActParamMore（），返回给t的brother成员，否则读取下一个单词。  对于更多的实参处理函数ActParamMore（），则读取当前的单词，如果为右括号则直接返回空，否则如果为“，”则递归调用实参处理函数ActParamList（）。  输入语句部分处理函数InputStm（），建立新的Read语句语法树节点，匹配read和左括号，将下一个单词的语义等赋值给当前节点的name成员，作为read语句的标识符名，匹配ID，匹配右括号。  输出语句部分处理函数OutputStm（），建立WRITE语句语法树节点，匹配write，左括号，调用表达式处理函数返回值复位新语法树节点成员的第一个儿子节点作为write语句的表达式，匹配右括号，结束。  返回语句部分处理函数ReturnStm（），建立新的RETURN语句语法树节点，匹配return，返回。  在调用parse函数后语法树已经形成了，其中的根节点为root，则打印语法分析数函数printTree（TreeNode\*tree），为了使得语法树看上去美观明了，这里是先打印根节点，之后对三个儿子节点递归打印，儿子都打印完成后再打印兄弟节点，每次进入一个节点的时候回缩进固定数目的空格，跳出后悔回退缩进的空格数目，会使得同一层的兄弟在同一列上，在没输出一行之前先在行首打印行号，之后根据行的数字进行相应的补充空格填写，如果是两位数，则输出行号后向后打印2个空格，如果是三位数打印一个，如果是移位数打印三个空格，这样在后面统一打印空格时才会使得同兄弟在同一列上。之后对数的nodekind进行判断，如果是Pro则为程序开始，ProHead则为程序头，打印程序头并且打印节点的name即为程序名，如果是Dec则为声明类型节点，对节点的过程参数进行判断打印，形实参数的信息打印，之后判断节点的具体类型，如果是数组Array则打印上下界，对数组的子类型进行判断输出，如果是字符整型记录标识符等则直接打印，如果是标识符的话则要把表示符名打印出来。对相同类型的变量进行依次打印，如果samenum为0的话说明没有变量声明错误！！如果节点类型为Type则打印Type，如果为Var则打印Var，如果是函数声明则打印FuncDec，打印函数的名字，如果是语句序列标志节点，则打印StmL，如果是语句类型则打印Stmt，对语句类型的具体类型进行判断，如果是If，While，Assign等直接打印，如果是Read的话则打印Read，将标识符名字打印出来。如果是表达式类型，则打印Exp，对具体类型，如果是操作符，则打印Op，对表达式属性的操作符进行判断，打印相应的符号。如果是数组成员，则打印ArrayMember，接着打印树组成员的语义信息。如果是常量，则打印Const，对表达式的变量类型进行判断，输出相应的类型和语义信息。如果是百纳领，则输出VarType，之后和Const一样处理。  语法分析（递归下降法）结束！！  三、语法分析（LL1分析法） | | |
| **程序界面及运行截图** | | |
|  | | |
| **源程序核心代码** | | |
|  | | |