第一部分 课程设计任务

一、必做内容：

第一项课程设计任务要求：

（1）以文本文件的方式输入某一高级程序设计语言的所有单词对应的正则表达式，系统需要提供一个操作界面，让用户打开某一语言的所有单词对应正则表达式文本文件，该文本文件的具体格式可根据自己实际的需要进行定义。

（2）正则表达式应该可以支持命名操作，运算符号有： 转义符号(\)、连接、 选择(|)、闭包(*)、 正闭包（+）、 [ ]、 可选（?）、 括号 ( )

（3）要提供一个源程序编辑界面，让用户输入一行（一个）或多行（多个）正则表达式（可保存、打开正则表达式文件）

（4）需要提供窗口以便用户可以查看转换得到的 NFA（用状态转换表呈现即可）

（5）需要提供窗口以便用户可以查看转换得到的 DFA（用状态转换表呈现即可）

（6）需要提供窗口以便用户可以查看转换得到的最小化 DFA（用状态转换表呈现即可）

（7）将最小化得到 DFA 图转换得到的词法分析源程序（该分析程序需要用 C/C++语言描述，而且**只能采用讲稿中的转换方法一或转换方法二来生成源程序**），系统需提供窗口以便用户可以查看转换得到的词法分析源程序

（8）对要求(7)得到的源程序进行编译生成一个可执行程序，并以该高级程序设计语言的一个源程序进行测试，输出该该源程序的单词编码。需要提供窗口以便用户可以查看该单词编码。

（9）对系统进行测试：（A）先以 TINY 语言的所有单词的正则表达式作为文本来测试，生成一个 TINY 语言的词法分析源程序；（B）接着对这个词法分析源程序利用 C/C++编译器进行编译，并生成可执行程序；（C）以 sample.tny 来测试，输出该 TINY 语言源程序的单词编码文件 sample.lex。

（10）要求应用程序为窗口式图形界面

（11）书写完善的软件文档

输入格式的说明：

```
digit=[0-9]
letter=[A-Za-z]
_num100=(\+|-)?digit+
_ID101=letter(letter|digit)*
_specail200S= \+ | - | \* | / | = | < | <= | <<
```

说明：

（1）通过命名中加下划线(_)来表示该正则表达式需要生成 DFA 图。

（2）命名中的名字后的数值为对应单词的编码

（3）命名中的名字中数值后加上 S 表示后面有多个单词，但对应的单词编码从这个数值开始编码。

（4）由于整数中的正号（+），在系统已经被用作正比闭包运算符号，所以需要通过引入转义符号\来区分。同理，如算术运算符号（*），在系统已经被用作闭包运算符号，所以也需要通过引入转义符号\来区分。

第二项课程设计任务要求

（1）以文本文件的方式输入某一高级程序设计语言的所有语法对应的 BNF 文法，因此系统需要提供一个操作界面，让用户打开某一语言的所有语法对应的 BNF 文法的文本文件，该文本文件的具体格式可根据自己实际的需要进行定义。

（2）求出文法的每个非终结符号的 First 集合和 Follow 集合，并需要提供窗口以便用户可以查看该结果（可用两张表格的形式分别进行呈现）

（3）需要提供窗口以便用户可以查看文法对应的 LR(0)DFA 图。（可以用画图的方式呈现，也可用表格方式呈现该图点与边数据）

（4）判断该文法是否为 SLR(1)文法。（应提供窗口呈现判断的结果，如果不是 SLR（1）文法，需要在窗口中显示其原因）

（5）需要提供窗口以便用户可以查看文法对应的 LR(1)DFA 图。（可以用画图的方式呈现，也可用表格方式呈现该图点与边数据）

（6）需要提供窗口以便用户可以查看文法对应的 LR(1)分析表。（LR(1)分析表采用表格的形式呈现）

（7）打开一个与上述所输入 BNF 文法对应的程序设计语言的源程序（即任务一的输出文件），并采用 LR(1)语法分析方法对这个源程序中的每个语句进行语法分析，需要提供窗口以便用户可以查看对应的语法分析过程。【可以使用表格的形式逐行显示分析过程】

（8）在（6）的分析过程中同时生成相应的语法树，需要提供窗口以便用户可以查看对应的语法分析过程（语法树需要采用树状形式进行呈现）。【每个语句的语法树结构可根据实际的需要进行定义。】

（9）以 TINY 语言的所有语法以及第一项任务的测试结果 sample.lex 作为测试，并生成对应的语法树并呈现出来。

（10）要求应用程序为窗口式图形界面

（11）书写完善的软件文档

（12）选做内容：可以生成某种中间代码【具体的中间代码形式可以自定】。

输入格式的说明：输入的文法均为 2 型文法（上下文无关文法）。文法规则为了处理上的简单，输入时均默认输入的第一个非终结符就是文法的开始符号，用#表示空串ε。因此下面的文法均为正确的输入。

例 1：

E->E+T

T->a |#

由于第一个非终结符为 E，因此 E 是文法的开始符号。

#表示空串ε

例 2：

exp ->exp addop term | term
addop -> + | -
term -> term mulop factor | factor
mulop -> * |/
factor -> (exp) | n
由于第一个非终结符为 exp，因此 exp 是文法的开始符号。

第三项课程设计任务要求

mini-c 语言作为测试

（1）以 mini-c 的词法进行测试，并以至少一个 mini-c 源程序进行词法分析的测试（该 mini-c 源程序需要自己根据 mini-c 词法和语法编写出来，类似于 sample.tny）。

（2）以 mini-c 的语法进行测试，并以测试步骤（1）的源程序所生成的单词编码文件进行语法分析，生成对应的语法树。

~~**二、选做内容：**~~

~~1.采用 LR(1)语法分析方法进行语法分析并生成相应的中间代码，中间代码可以自选（可以是四元组、三元组、伪代码等中间代码），中间代码生产结果可以在屏幕上显示或以文件的形式保存~~

第二部分 编程要求

1.系统开发编程所使用的编程语言只能是 C/C++语言，但开发平台不做要求。
2.程序应该具有良好的编程风格
3.必要的注释；（简单要求如下）
1） readme 文件对上交的实验内容文件或目录作适当的解释；
2） 每个源程序文件中注释信息至少包含以下内容：
（1）版权信息。
（2）文件名称，标识符，摘要或模块功能说明。
（3）当前版本号，作者/修改者，完成日期。
（4）版本历史信息。 // （1）--（4）部分写在文件头
（5）所有的宏定义，非局部变量都要加注释
（6）所有函数前有函数功能说明，输入输出接口信息，以及调用注意事项
（7）函数关键地方加注释

第三部分 课程设计进度安排

时间	进度安排
第 1 周	任课老师进行课程设计介绍
第 1 周~第 5 周	完成第一项课程设计任务的软件设计及实验报告书的书写要求，并以 Tiny 语言的词法进行测试及书写测试报告
第 6 周~第 10 周	完成第二项课程设计任务的软件设计及实验报告书的书写要求，并以 Tiny 语言的语法进行测试及书写测试报告
第 11 周~第 13 周	1.以 Mini-C 进行整个课程设计系统功能的测试（包含词法和语法的测试），并书写测试报告 2.完成最后课程设计实验报告的整合和自评 3.完成系统使用说明书的书写 4.课程设计报告讲解视频的录制（15 分钟）
第 14 周	完成如下内容： 1.整理好课程设计源程序、测试文本及测试源程序、测试报告、课程设计的可执行程序、使用说明书、课程设计报告书和讲解视频。 2.第 14 周周五晚上 12 点之前提交上述所有资料，在砺儒云课堂进行提交。
第 15~17 周	1.抽查、评分。 2.抽查到的同学需到学院实验室解释相关的工作。

第四部分：课程设计评分标准

主要根据学生所制作的两项课程设计任务要求的软件系统设计、运行及测试情况、课程设计实验报告书的书写情况以及课程设计报告讲解视频的情况进行评分（系统和文档各占 45%，课程设计报告讲解视频占 10%,且均需要达到及格标准以上）。

评分标准主要依据如下：

- ✓ E<60:在规定时间内上交实验源程序、实验报告书、测试数据与测试结果、测试报告、讲解视频,但必做内容的设计功能（除理论课程实验已经完成的功能外）未能完成超过 3/5。
- ✓ 60<D<70: 在规定时间内上交所有的课程设计材料,完成了必做内容要求中的大部分内容，讲解视频有完整的讲解思路，编程风格好，实验报告以及测试材料基本符合规范，设计思想基本清晰，界面基本符合要求。
- ✓ 70<C<80: 在规定时间内上交所有的课程设计材料,完成了必做内容要求中的全部内容，讲解视频讲解思路符合要求，实验报告以及测试材料等符合规范，编程风格好，设计思想基本清晰，界面美观大方，使用方便。

- ✓ 80<B<90: 在规定时间内上交所有的课程设计材料,完成了必做内容要求中的全部内容，并完成选做内容中的部分要求，讲解视频讲解思路清晰，实验报告以及测试材料等规范清晰，编程风格好，设计思想清晰，界面美观大方,使用方便。
- ✓ 90<A<100: 在规定时间内上交所有的课程设计材料,完成了必做内容和选做内容的全部内容，且功能完善，讲解视频讲解思路十分清晰，实验报告以及测试材料等十分规范清晰，设计思想十分清晰，编程风格好，界面美观大方，使用方便。

第五部分 测试内容及要求

1.在课程设计的第一个任务和第二个任务的实现过程中，需要分别以 Tiny 语言的词法和语法进行测试。

2.在课程设计的第一个任务和第二个任务都全部完成后，再以 Mini-c 语言进行新一轮的词法和语法进行测试。

第二组的测试数据（课程设计的第一个任务和第二个任务全部完成后进行的测试）：

1.Mini C 的单词

1).下面是语言的关键字:
else if int float real return void while do
所有的关键字都是保留字,并且必须是小写。

2).下面是专用符号:
_ + - * / % ^ . < <= > >= == != = ; , ( ) [ ] { } //行注解

3).其他标记是 ID 和 NUM ，通过下列正则表达式定义:
ID= (_|letter)(_|letter|digit)*
NUM=digit+(.digit+)?
letter= [a-zA-Z]
digit=[0-9]
小写和大写字母是有区别的。

4).空格由空白、换行符和制表符组成。空格通常被忽略,除了它必须分开 ID、NUM 关键字。

5).注释用通常的 C 语言符号//是行注解且不可以超过一行。注释不能嵌套。

2.Mini C 的语法

Mini C 的BNF语法如下:

1. program-> definition-list
2. definition-list-> definition-list definition | definition
3. definition-> variable-definition | function-definition
4. variable-definition-> type-indicator ID ; |type-indicator ID[NUM];
5. type-indicator-> int | float | real| void
6. function-definition-> type-indicator ID（parameters）compound-stmt
7. parameters->parameter-list | void
8. parameter-list->parameter-list, parameter | parameter
9. parameter-> type-indicator ID | type-indicator ID[ ]
10. compound-stmt-> { local-definitions statement-list }
11. local-definitions-> local-definitions variable-definition |empty
12. statement-list-> statement-list statement | empty
13. statement->expression-stmt | compound-stmt | condition-stmt|while-stmt |dowhile-stmt | return-stmt
14. expression-stmt-> expression ; | ;
15. condition-stmt-> if(expression) statement |if（expression）statement else statement
16. while-stmt-> while (expression) statement;
17. dowhile-stmt-> do statement while(expression) ;
18. return-stmt->return ; | return expression ;
19. expression-> variable=expression | simple-expression
20. variable->ID |ID[expression]
21. simple-expression->additive-expression relop additive-expression |additive-expression
22. relop-><=|<|>|>=|==|!=
23 additive-expression->additive-expression addop term | term
24. addop->+|-
25. term->term mulop factor | factor
26. mulop->*|/|% |^
27. factor->(expression)| variable | call | NUM
28. call->ID(arguments)
29. arguments->argument-list | empty
30. argument-list->argument-list,expression | expression

为了更方便构造Mini C各语句的语法树结构或中间代码，你可以根据下面对每个文法规则的语义解释说明进行组织。

对以上每条文法规则,给出了相关语义的简短解释

0.上述文法规则中出现的empty即为空串ε，输入时可以用#替代。

1.program→ definition-list
2.definition-list→definition-list definition |definition
3.definition→variable-definition | function-definition

程序由定义的列表(或序列)组成,定义可以是函数或变量定义,顺序是任意的。至少必须有一个定义。接下来是语义限制(这些在 C 中不会出现)。所有的变量和函数在使用前必须定义(这避免了向后 backpatching 引用)。程序中最后的定义必须是一个函数定义,名字为 main。注意,Mini C 缺乏原型,因此定义和声明之间没有区别(像 C 一样)。

4.variable-definition→type-indicator ID ; |type-indicator ID[NUM];
5.type-indicator→int|float|real|void

变量定义或者定义了简单的整数或浮点数类型变量,或者是基类型为整数或浮点数的数组变量,索引范围从 0 到 NUM-1。注意,在 Mini C 中仅有的基本类型是整型、浮点数(分单精度 float 和双精度 real)和空类型。在一个变量定义中,只能使用类型指示符 int、float 或 real。void 用于函数定义(参见下面)。也要注意,每个定义只能定义一个变量。

6.function-definition→ type-indicator ID(parameters)compound-stmt
7.parameters→parameter-list | void
8.parameter-list→parameter-list, parameter | parameter
9.parameter→type-indicator ID | type-indicator ID [ ]

函数定义由返回类型指示符、标识符以及在圆括号内的用逗号分开的参数列表组成,后面跟着一个复合语句,是函数的代码。如果函数的返回类型是 void,那么函数不返回任何值(即是一个过程)。函数的参数可以是 void(即没有参数),或者一列描述函数的参数。参数后面跟着方括号是数组参数,其大小是可变的。简单的整型或浮点数参数由值传递。数组参数由引用来传递(也就是指针),在调用时必须通过数组变量来匹配。注意,类型“函数”没有参数。一个函数参数的作用域等于函数定义的复合语句,函数的每次请求都有一个独立的参数集。函数可以是递归的(对于使用定义允许的范围)

10.compound-stmt →{local-definitions statement-list }

复合语句由用花括号围起来的一组定义和语句组成。复合语句通过用给定的顺序执行语句序列来执行。局部定义/声明的作用域等于复合语句的语句列表,并代替任何全局定义/声明。

11.local-definitions→local-definitions variable-definition | empty
12.statement-list→statement-list statement | empty
注意定义和语句列表都可以是空的(非终结符 empty 表示空字符串,有时写作ε。)

13.statement→ expression-stmt
|compound-stmt
|condition-stmt
|dowhile-stmt
|return-stmt

14.expression-stmt→expression; |;

表达式语句有一个可选的且后面跟着分号的表达式。这样的表达式通常求出它们一方的结果。因此,这个语句用于赋值和函数调用。

15.condition-stmt→if(expression) statement
|if(expression)statement else statement

if 语句有通常的语义:表达式进行计算;非 0 值引起第一条语句的执行;0 值引起第二条语句的执行,如果它存在的话。这个规则导致了典型的悬挂 else 二义性,可以用一种标准的方法解决:else 部分通常作为当前 if 的一个子结构立即分析(“最近嵌套”非二义性规则)。

16.while-stmt→while (expression) statement

while 语句是 Mini C 中的重复语句表示之一。它重复执行语句,并且如果表达式的求值为非 0，则执行语句，当表达式的值为 0 时结束。

17.dowhile-stmt→do statement while (expression)

do while 语句是 Mini C 中的重复语句表示之二。它重复执行语句,直到表达式的求值为 0。

18.return-stmt→ return ; | return expression ;

返回语句可以返回一个值也可无值返回。函数没有说明为 void 就必须返回一个值。函数定义为 void 就没有返回值。 return 引起控制返回调用者(如果它在 main 中,则程序结束)。

19.expression→variable = expression | simple-expression
20.variable→ID |ID[expression]

表达式是一个变量引用,后面跟着赋值符号(等号)和一个表达式,或者就是一个简单的表达式。赋值有通常的存储语义:找到由 variable 表示的变量的地址,然后由赋值符右边的子表达式进行求值,子表达式的值存储到给定的地址。这个值也作为整个表达式的值返回。variable 是整型变量、浮点数变量或下标数组变量。负的下标将引起程序停止(与 C 不同)。然而,不对其进行下标越界检查。

variable 表示 Mini C 比 C 有进一步的限制。在 C 中赋值的目标必须是左值（l-value),左值是可以由许多操作获得的地址。在 Mini C 中唯一的左值是由 variable 语法给定的,因此这个种类按照句法进行检查,代替像 C 中那样的类型检查。故在 Mini C 中指针运算是禁止的。

21.simple-expression→additive-expression relop additive-expression
|additive-expression
22.relop→<=|<|>|>=|==|!=

简单表达式由无结合的关系操作符组成(即无括号的表达式仅有一个关系操作符)。简单表达式在它不包含关系操作符时,其值是加法表达式的值,或者如果关系算式求值为 true,其值为 1,求值为 false 时值为 0。

23.additive-expression→ additive-expression addop term | term
24.adop→+|-
25.term→term mulop factor | factor
26.mulop→*|/|%

加法表达式和项表示了算术操作符的结合性和优先级。符号/对于整数则表示整数除，即任何余数都被截去，对于浮点数则表示除法；符号%表示整数求除，浮点数不支持%运算。

27.factor→ (expression)| variable | call | NUM

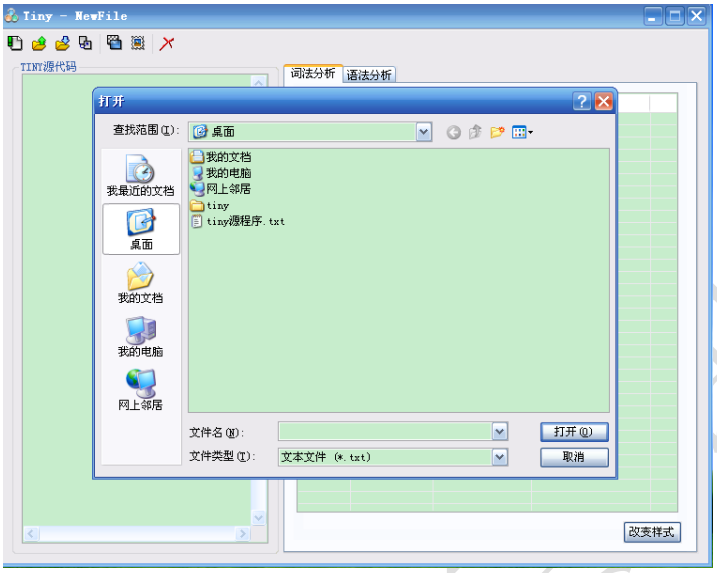
因子是围在括号内的表达式;或一个变量,求出其变量的值;或者一个函数调用,求出函数的返回值;或者一个 NUM,其值由扫描器计算。数组变量必须是下标变量,除非表达式由单个 ID 组成,并且以数组为参数在函数调用中使用(如下所示)。

29.call→ ID(arguments)
29.arguments→ argument-list | empty
30.argument-list→ argument-list, expression | expression

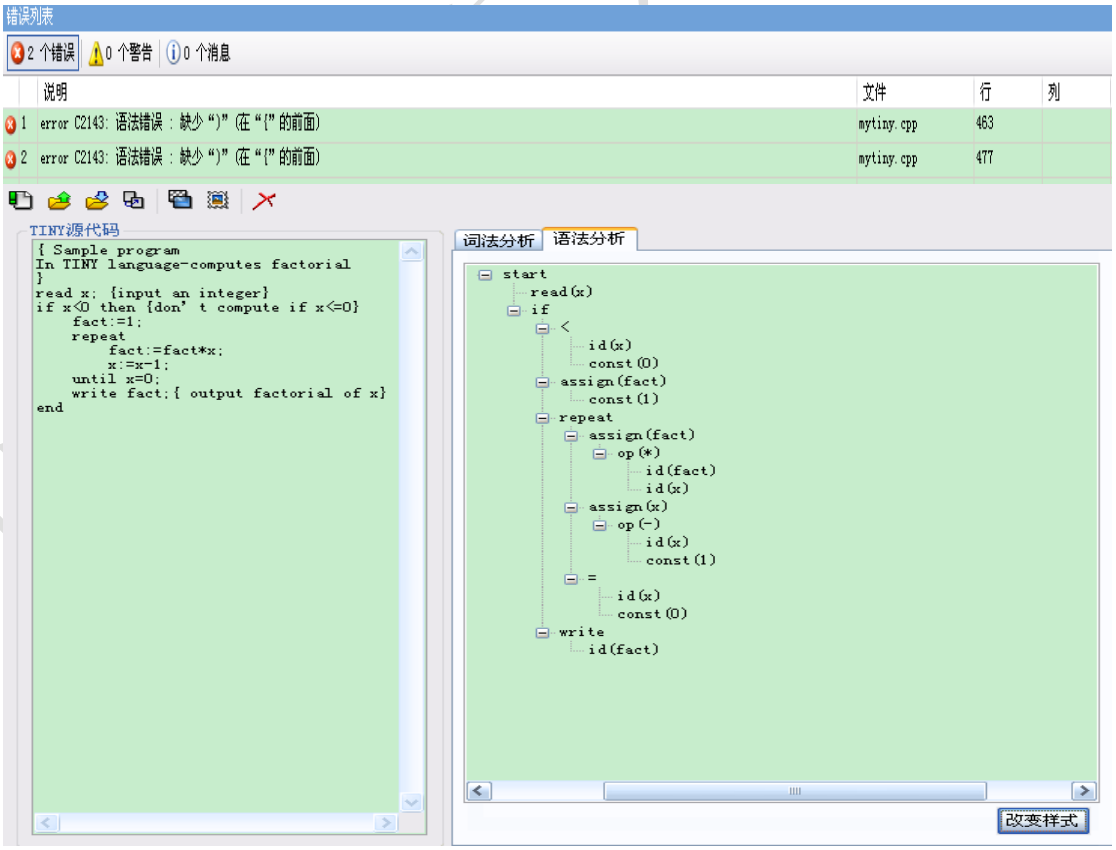
函数调用的组成是一个 ID(函数名),后面是用括号围起来的参数。参数或者为空,或者由逗号分割的表达式列表组成,表示在一次调用期间分配的参数的值。函数在调用之前必须定义,定义中参数的数目必须等于调用中参数的数目。函数定义中的数组参数必须和一个表达式匹配,这个表达式由一个标识符组成表示一个数组变量。

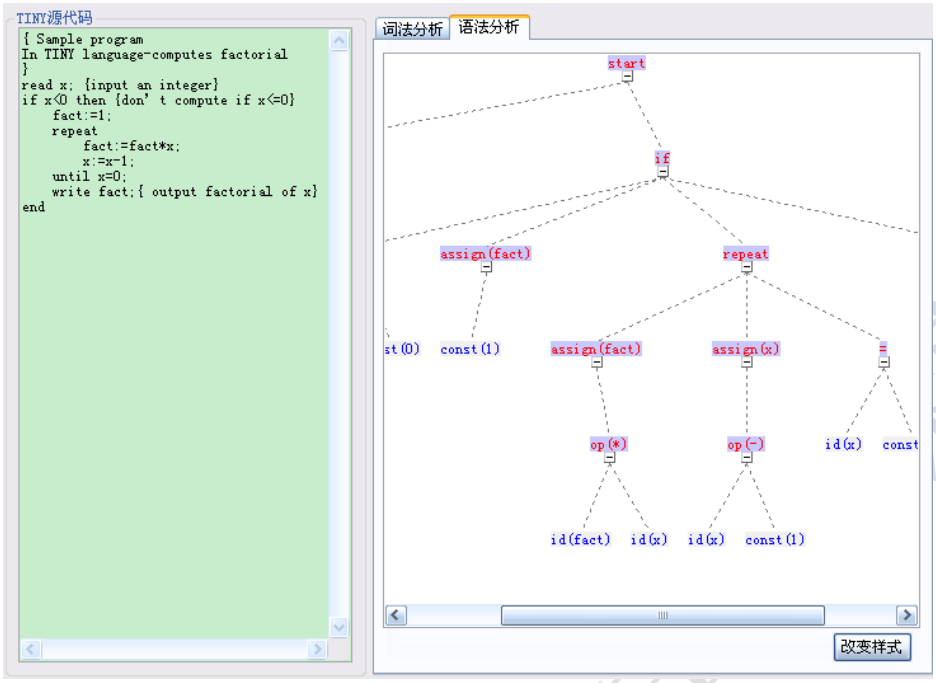
附录：

1. 文件打开的界面要求范例



2. 语法树呈现的界面要求规范




TINY源代码
{ Sample program
In TINY language-computes factorial
}
read x; {input an integer}
if x<0 then {don' t compute if x<=0}
fact:=1;
repeat
fact:=fact*x;
x:=x-1;
until x=0;
write fact;{ output factorial of x}
end
词法分析
语法分析
start
if
assign(fact)
repeat
st(0)
const(1)
assign(fact)
assign(x)
=
op(*)
op(-)
id(x)
const
id(fact)
id(x)
id(x)
const(1)
改变样式