《编译原理》
项目设计指导书

黄煜廉 整理

华南师范大学计算机学院
2020.01

一、 课程基本情况

1.修读性质：必修

二、课程介绍

（1）课程简介与要求

编译程序、解释程序是所有语言软件的基本组成部分。掌握和了解这类软件的开发技术，无论是对于系统程序员，还是初级程序员，都具有极大使用价值。本项目设计课程主要是让学生充分运用编译程序构造的基本原理和方法设计一个简单有用的编译程序。经过这门课程的上机实验训练，学生应达到以下要求：

a/进一步巩固和加深对编译程序构造的基本原理和方法的理解，提高学生综合运用所学知识的能力。

b/对现有编译程序和实现编译程序具有较强的分析和设计能力。

d/进一步提高学生对中、大型软件的分析和开发能力

e/更加熟练、规范地书写软件设计的报告书。

f/进一步形成一个软件工作者所应具有的科学工作的方法和作风。

（2）实验目的及要求

编译程序是计算机系统软件的最重要组成部分之一。也是用户最需要的工具之一。编译理论与技术是计算机科学中发展最迅速、最成熟的一个分支，现已形成了一套比较系统化的理论与方法，并在编译程序自动生成方面不断取得新的进展。

如果一个人不能理解程序设计语言，它就不能真正地理解计算机。要真正理解程序设计语言，除了学会使用它，还要知道它们的设计与实现。也就是不仅要知道其然，更要知道其所以然。

本项目设计课程的开设就是为了加强理论联系实际，使学生了解编译程序的概貌和培养其实际动手能力。 进一步提高学生独立分析问题和解决问题的能力以及提高学生对中、大型软件的分析和开发能力

（3）实验方式与要求

本课程项目设计以学生独立设计或组成最多 2 人一个小组的方式来来完成设计。教师不做具体、过细的指导，只是通过参加讨论、审阅方案、测试验收等工作起把握方向和协调进度的作用。

a/首先提供一个设计范例，让学生进一步熟悉编译程序的设计理论。

b/结合设计范例，提出一个实现 Mini C 语言编译程序的构造要求

（a） 由学生在指定时间内把 Mini C 语言的编译程序进行数据结构分析、概要设计等前期工作情况提交给教师。

（b） 教师根据各同学所提交上来的分析情况进行点评批阅，并进一步指导学生下一步的工作方向。

（c） 由各同学自行根据概要设计进行详细设计、程序编制、调试，最后提交实验报告书。

三、编译器设计方案

这里定义了一个编程语言称作 Mini C,这是一种适合编译器设计方案的語言,它比 TINY 语言更复杂,包括函数和数组。本质上它是 C 的一个子集,但省去了一些重要的部分,因此得名。

这个编译器设计方案由以下几个部分组成:

第一部分，列出了语言惯用的词法,包括语言标记的描述。

第二部分， 给出了每个语句构造的 BNF 描述,同时还有相关语义的英语描述。

第三部分，给出了 Mini C 的两个示例程序作为测试用例。

第四部分，课程项目设计内容及要求。

第五部分，课程项目测试内容及要求。

第六部分，编程要求

第七部分，项目进度安排

第八部分，项目评分标准

附录：界面范例

第一部分 Mini C 惯用的词法

1.下面是语言的关键字:

```
else if  int  return void  while
```

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:

```
+ - * / <  <= > >= ==  !=  =  ; , ( ) [ ] { }  /*  */
```

3.其他标记是 ID 和 NUM ， 通过下列正则表达式定义:

```
ID= letter | letter*
NUM=digit digit *
letter= a | .. | z | A | .. | 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/* ...*/围起来。注释可以放在任何空白出现的位置(即注释不能放在标记内)上,且可以超过一行。注释不能嵌套。

第二部分 Mini C 的语法和语义

Mini C 的 BNF 语法如下:

1. program→ declaration-list
2. declaration-list→ declaration-list declaration | declaration
3. declaration→ var-declaration | fun-declaration
4. var-declaration→ type-specifier ID ; |type-specifier ID[NUM];
5. type-specifier→ int | void
6. fun-declaration→ type-specifier ID (params) | compound-stmt
7. params→params-list | void
8.param-list→param-list, param | param
9.param→ type-specifier ID | type-specifier ID[ ]
10.compound-stmt→ { local-declarations statement-list }
11.local-declarations→ local-declarations var-declaration |empty
12.statement-list→ statement-list statement | empty
13.statement→expression-stmt | compound-stmt | selection-stmt
|iteration-stmt | return-stmt
14.expression-stmt→ expression ; | ;
15.selection-stmt→ if(expression) statement
|if (expression) statement else statement
16.iteration-stmt→while(expression) statement
17.return-stmt→return ; | return expression ;
18.expression→ var=expression | simple-expression
19.var→ID |ID[expression]
20.simple-expression→additive-expression relop additive-expression
|additive-expression
21.relop→<=|<|>|>=|==|!=
22 additive-expression→additive-expression addop term | term
23.addop→+|-
24.term→term mulop factor | factor
25.mulop→*|/
26 factor→(expression)| var | call | NUM
27.call→ID(args)
28.args→arg-list | empty
29.arg-list→arg-list,expression | expression
对以上每条文法规则,给出了相关语义的简短解释。

1.program→ declaration-list
2.declaration-list→declaration-list declaration |declaration
3.declaration→var-declaration | fun-declaration

程序由声明的列表(或序列)组成,声明可以是函数或变量声明,顺序是任意的。至少必须有一个声明。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声明(这避免了向后 backpatching 引用)。程序中最后的声明必须是一个函数声明,名字为 main。注意,Mini C 缺乏原型,因此声明和定义之间没有区别(像 C 一样)。

4.var-declaration→type-specifier ID ; |type-specifier ID[NUM];
5.type-specifier→int|void

变量声明或者声明了简单的整数类型变量,或者是基类型为整数的数组变量,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和空类型。在一个变量声明中,只能使用类型指示符int。void用于函数声明(参见下面)。也要注意,每个声明只能声明一个变量。

6.fun-declaration→ type-specifier ID(params)compound-stmt
7.params→param-list | void
8.param-list→param-list, param | param
9.param→type-specifier ID | type-specifier ID [ ]

函数声明由返回类型指示符、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声明的复合语句,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(对于使用声明允许的范围)

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

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

11.local-declarations→local-declarations var-declaration | empty
12.statement-list→statement-list statement | empty

注意声明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
|selection-stmt
|iteration-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.selection-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.iteration-stmt→while (expression) statement

while 语句是 Mini C 中唯一的重复语句。它重复执行表达式,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.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。函数声明为 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)。

18.expression→var = expression | simple-expression

19.var→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 var 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。var 是简单的(整型)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。

var 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 var 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。

20.simple-expression→additive-expression relop additive-expression
|additive-expression

21.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。

22.additive-expression→ additive-expression addop term | term

23.adop→+|-

24.term→term mulop factor | factor

25.mulop→*|/

加法表达式和项表示了算术操作符的结合性和优先级。符号表示整数除;即任何余数都被截去。

26.factor→ (expression)| var | 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)。

27.call→ ID(args)
28.args→ arg-list | empty
29.arg-list→ arg-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声明,声明中参数的数目必须等于调用中参数的数目。函数声明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。

最后,上面的规则没有给出输入和输出语句。在 Mini C 的定义中必须包含这样的函数,因为与 C 不同,Mini C 没有独立的编译和链接工具;因此,考虑两个在全局环境中预定义的函数,好像它们已进行了声明:

```
int input(void){ ...}
void output(int x){ ... }
```

input 函数没有参数,从标准输入设备(通常是键盘)返回一个整数值。 output 函数接受一个整型参数,其值和一个换行符一起打印到标准输出设备(通常是屏幕)

第三部分 Mini C 的程序例子（测试用例）

下面的程序输入两个整数,计算并打印出它们的最大公因子

```
/* A program to perform Euclid's
   Algorithm to compute gcd. */

int gcd (int u, int v)
{ if (v == 0)return u;
   else return gcd(v,u-u/v*v);
  /*  u-u/v*v == u mod v */
}

void main(void)
{  int x, int y;
   x=input();
   y=input();
   output(gcd(x, y));
}
```

下面的程序输人 10 个整数的列表,对它们进行选择排序,然后再输出:

```
/* A program to perform selection sort on a 10
   element array。*/
int x[10];
int minloc(int a[], int low, int high)
{ int i; int x; int k;
```

```
    k=low;
    x=a[low];
    i=low+1;
    while(i<high)
     { if(a[i]< x)
       { x =a[i];
         k=i;
       }
       i=i+1;
   }
   return k;
 }

void sort( int a[], int low, int high)
{ int i; int k;
  i=low;
  while(i<high-1)
   {  int t;
      k=minloc(a,i,high);
      t=a[k];
      a[k]= a[i];
      a[i]=t;
      i=i+1;
   }
}

 void main(void)
 {  int i;
    i=0;
    while(i<10)
    { x[i]=input();
      i=i+1;
      sort(x,0,10);
       i=0;
      while(i<10)
      { output(x[i]);
        i=i+1;
       }
    }
  }
```

第四部分 设计内容及要求

1.实现一个 Mini C 扫描器（词法分析器）,或者像 DFA 用手工进行,或者使用 Lex。
2.设计一个 Mini C 语法树结构,适合于用分析器产生，语法树的结果可以在屏幕上显示或以文件的形式保存。
3.实现一个 Mini C 语法分析器(这首先要完成一个 Mini C 扫描器，见设计功能 2),或者使用递归下降用手工进行,或者使用 Yacc。分析器要产生合适的语法树(见设计功能 3)。
4.实现 Mini C 的语义分析器。分析器的主要要求是,除了在符号表中收集信息外,在使用变量和函数时完成类型检查。因为没有指针或结构,并且仅有的基本类型是整型,类型检查器需要处理的类型是空类型、整型、数组和函数。
5.实现 Mini C 的代码产生器，其代码指令与参考资料中的虚拟机一致。代码生产结果可以在屏幕上显示或以文件的形式保存。
6.配套修改参考资料中虚拟机程序以实现代码指令的解释执行，并执行得出相应的结果。
7.书写实验报告以及系统的使用说明书。

第五部分 测试内容及要求

1.组织相应的测试数据来测试 Mini C 的所有词法。
2.组织相应的测试数据来测试 Mini C 的所有语法。
3.组织相应的测试数据来测试 Mini C 的所有语义问题。
4.使用第三部分的测试用例来测试程序，并显示或保存相应的语法树。
5.使用第三部分的测试用例来测试程序，并显示或保存相应的虚拟机目标代码。
并使用虚拟机解释执行得出程序的执行结果。

第六部分 编程要求

1.编程所使用的编程语言以及平台不做要求。
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

第七部分 项目进度安排

时间	进度安排
第 1 周	课程项目介绍
第 2 周	分组。或独立设计或最多 2 人一个小组，名单发给学习委员
第 3 周	各班学习委员整理分组名单发给老师
第 3 周~第 9 周	完成如下内容： 1. 选择好实用的开发平台以及编译工具 2. 完成词法分析和语法分析的设计及测试 3. 第九周周五晚上 12 点之前提交中期检查报告。由各分组组长汇报项目的完成情况，发给各班的学习委员，最后整理好发给老师
第 10 周~第 15 周	完成如下内容： 1. 完成语义分析、代码生成功能及测试 2. 完成虚拟机解释功能的修改及测试 3. 完成实验报告的书写和自评 4. 完成系统使用说明书的书写
第 16 周	完成如下内容： 1.整理好源程序、测试用例、执行程序、使用说明书以及项目设计报告书。 2. 第 16 周周五晚上 12 点之前提交上述所有资料，由各分组组长发给各班的学习委员，最后整理好发给老师
第 17、18 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。

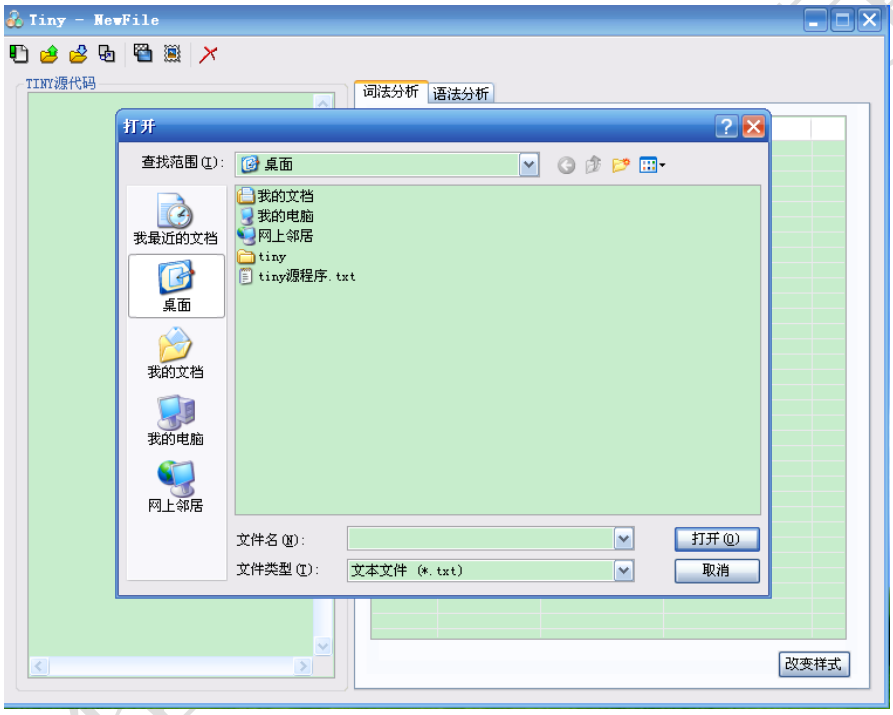
第八部分：项目评分标准

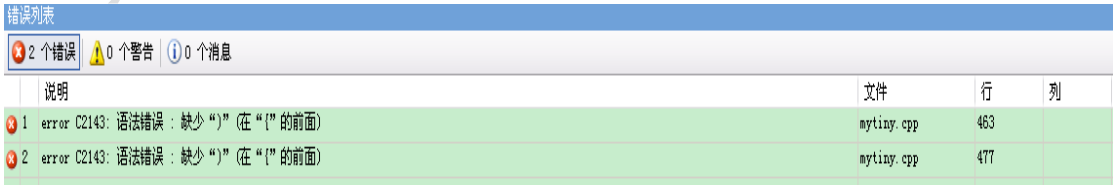
主要根据学生所制作的编译程序的情况及书写设计实验报告书的情况进行评分。评分标准主要依据如下：

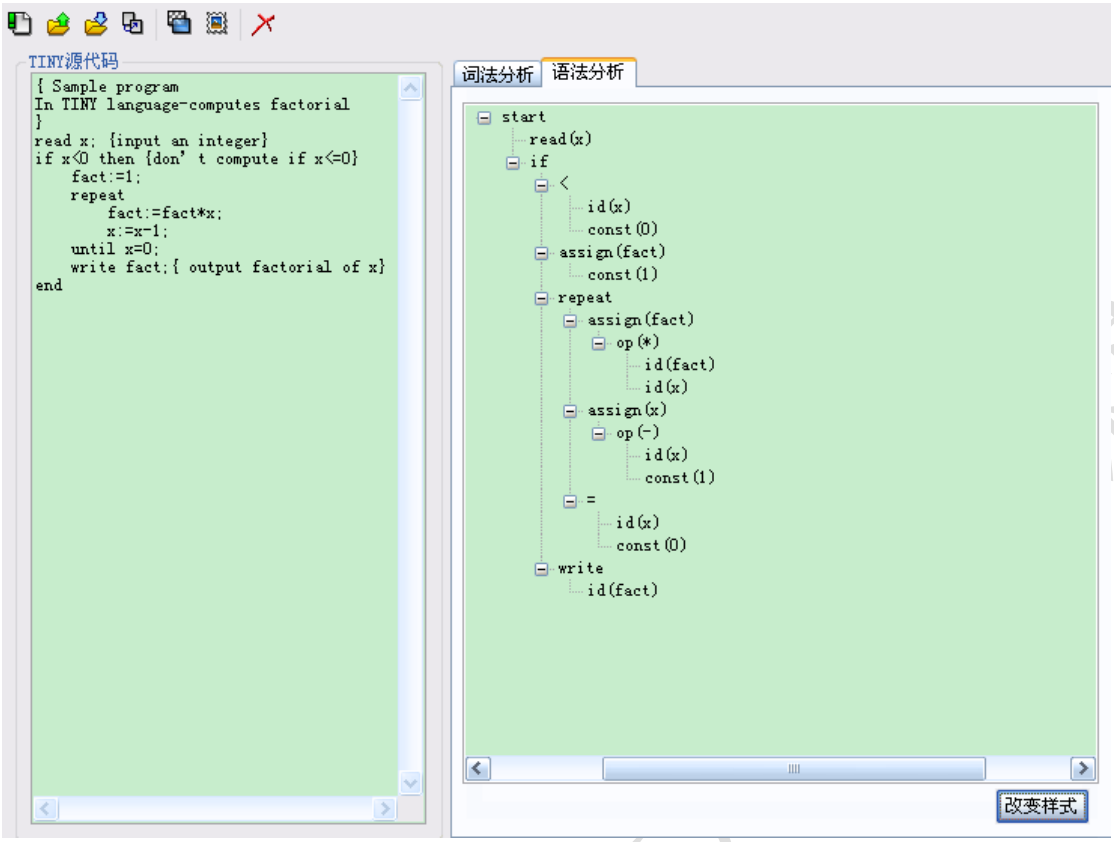
- ✓ E<60: 在规定时间内上交实验程序及文档,基本要求中的大部分内容未完成。
- ✓ 60<D<70: 在规定时间内上交实验程序及文档,完成了基本要求中的大部分内容，编程风格好，文档基本符合规范，设计思想基本清晰，界面基本符合要求。

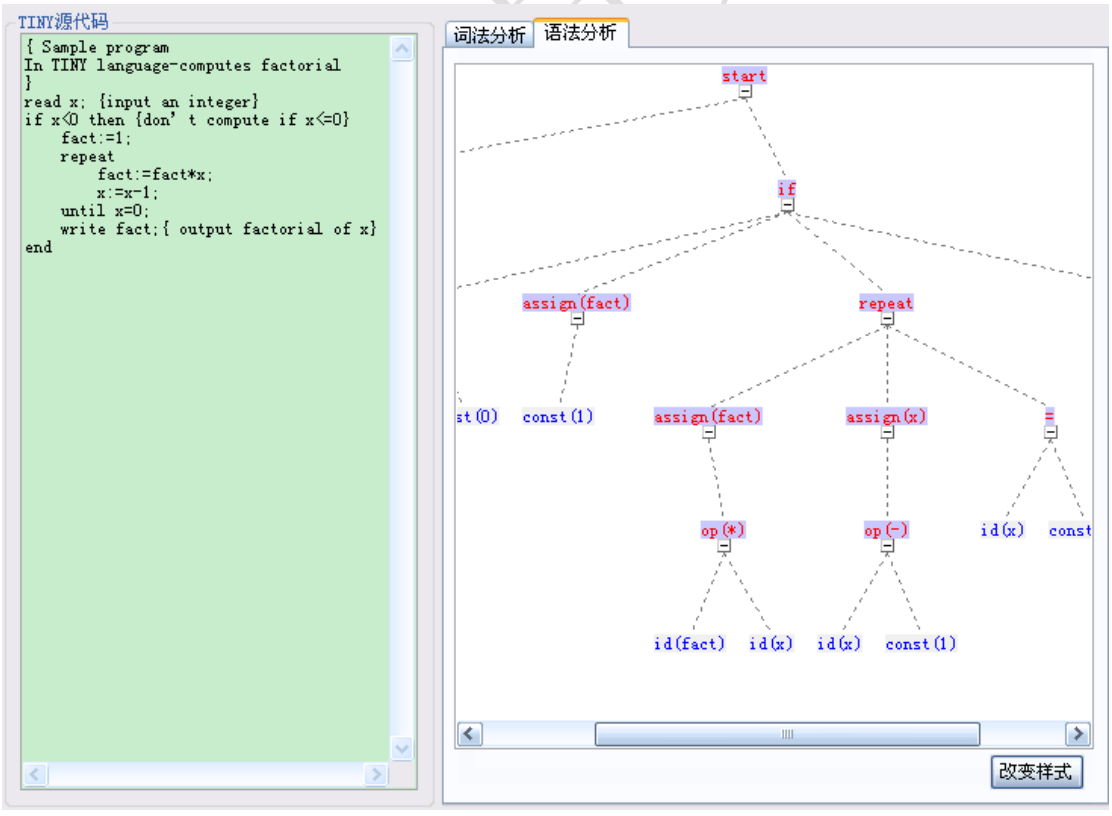
- ✓ 70<C<80：在规定时间内上交实验程序及文档，完成了基本要求中的全部内容，文档规范，编程风格好，设计思想基本清晰，界面美观大方，使用方便。
- ✓ 80<B<90：在规定时间内上交实验程序及文档，完成了基本要求中的全部内容，并完成选做内容中的部分要求，文档规范清晰，编程风格好，设计思想清晰，界面美观大方，使用方便。
- ✓ 90<A<100：在规定时间内上交实验程序及文档，完成了基本要求和选做内容的全部内容，且功能完善，文档规范清晰，设计思想十分清晰，编程风格好，界面美观大方，使用方便。

附录：界面要求范例






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
read(x)
if
<
id(x)
const(0)
assign(fact)
const(1)
repeat
assign(fact)
op(*)
id(fact)
id(x)
assign(x)
op(-)
id(x)
const(1)
=
id(x)
const(0)
write
id(fact)
改变样式


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