

编译原理

实验教学指导书

计算机科学与工程学院
华南理工大学

目录

1 实验简介	3
2 TINY+语言介绍	4
2.1 TINY+语言的词法定义	4
2.2 TINY+的语法定义	5
2.3 TINY+的语义定义	7
2.4 用 TINY+语言编写的示例程序	7
3 实验 1：实现 TINY+语言的词法分析器	9
3.1 实验目的	9
3.2 实验要求	10
3.3 TINY+的测试程序及词法分析器的输出	10
4 实验 2：实现 TINY+的语法分析器、语义分析器以及中间代码生成器	13
4.1 实验目的	13
4.2 实验要求	14
4.3 TINY+示例程序及其输出	14
附录：和 TINY+文法规则对应的生成三地址中间代码的属性文法	16

1 实验简介

学生在实验中，构造一个将 TINY+高级程序设计语言转换为 TINY+虚拟机上的中间代码的编译器。

整个实验包括两个部分：实验一完成 TINY+编译器的词法分析器部分；实验二完成 TINY+编译器的语法分析器部分、语义分析器部分及中间代码生成器部分。

每个同学必须独立完成自己的实验，与其他同学的讨论或合作是允许的，但必须是有限度的，可以互相交流想法和方法，但不能抄袭。学术不端将导致成绩为零。

TINY+的编译器必须用 C 语言或 C++语言实现（推荐使用 Microsoft Visual Studio）。

2 TINY+语言介绍

实验定义了一种叫 TINY+的高级程序设计语言，该语言是对 TINY 语言的一个扩充，TINY+比 TINY 增加了程序的声明部分，while 语句，字符串类型定义等等，在本节的描述中，用蓝色字体标识的是 TINY 语言原有的词法及语法规则，而用红色字体标识的是 TINY+语言扩充的词法及语法规则。本节主要是对 TINY+语言的介绍，具体包括：

- 1) TINY+语言的词法定义，包括对 TINY+语言的单词（token）的描述；
- 2) TINY+语言语法结构的 EBNF 描述；
- 3) TINY+语言主要的语义描述；
- 4) TINY+的实例程序

2.1 TINY+语言的词法定义

1. TINY+语言的关键字（keyword）包括：

or and int bool char while do

if then else end repeat until read write

所有的关键字是程序设计语言保留使用的，并且用小写字母表示，用户自己定义的标识符不能和关键字重复。

2. 特殊符号的定义如下：

> <= >= , '
 { } ; := + - * / () < =

3. 其他种类的单词包括标识符 ID，数字 NUM 以及字符串 STRING，他们的正规表达式的定义如下：

ID=letter (letter | digit)*

标识符是以字母开头，由字母和数字混合构成的符号串。

NUM=digit digit*

TINY+中对数字的定义和 TINY 相同。

STRING=' any character except ' '

一个字符串类型的单词是用单引号括起来的字符串'...'，引号内可出现除了' 以外的任何符号。一个字符串不能跨行定义。

letter=a|...|z|A|...|Z

digit=0|...|9

小写和大写字母是不同的。

4. 空白包括空格、回车以及 Tab。所有的空白在词法分析时，被当作单词 ID，NUM 以及保留字的分隔符，在词法分析之后，他们不被当作单词保留。
5. 注释是用花括号括起来的符号串 {...}，注释不能嵌套定义，但注释的定义可以跨行。

2.2 TINY+的语法定义

TINY+的语法用 EBNF 定义如下：

- 1 **program** \rightarrow declarations stmt-sequence
- 2 **declarations** \rightarrow decl ; declarations | ϵ
- 3 **decl** \rightarrow type-specifier varlist
- 4 **type-specifier** \rightarrow *int* | *bool* | *char*
- 5 **varlist** \rightarrow *identifier* { , *identifier* }
- 6 **stmt-sequence** \rightarrow statement { ; statement }
- 7 **statement** \rightarrow if-stmt | repeat-stmt | assign-stmt | read-stmt | write-stmt | while-stmt
- 8 **while-stmt** \rightarrow *while* bool-exp *do* stmt-sequence *end*
- 9 **if-stmt** \rightarrow *if* bool-exp *then* stmt-sequence [*else* stmt-sequence] *end*
- 10 **repeat-stmt** \rightarrow *repeat* stmt-sequence *until* bool-exp
- 11 **assign-stmt** \rightarrow *identifier* := exp
- 12 **read-stmt** \rightarrow *read identifier*
- 13 **write-stmt** \rightarrow *write* exp
- 14 **exp** \rightarrow arithmetic-exp | bool-exp | string-exp
- 15 **arithmetic-exp** \rightarrow term { addop term }
- 16 **addop** \rightarrow + | -
- 17 **term** \rightarrow factor { mulop factor }
- 18 **mulop** \rightarrow * | /
- 19 **factor** \rightarrow (arithmetic-exp) | *number* | *identifier*
- 20 **bool-exp** \rightarrow bterm { *or* bterm }
- 21 **bterm** \rightarrow bfactor { *and* bfactor }

22 bfactor → comparison-exp

23 comparison-exp → arithmetic-exp comparison-op arithmetic-exp

24 comparison-op → < | = | > | >= | <=

25 string-exp → string

2.3 TINY+的语义定义

- 一个用 TINY+语言编写的程序包括变量的声明和语句序列两个部分。变量声明部分可以为空，但一个 TINY+程序至少要包含一条语句。
- 所有的变量在使用之前必须声明，并且每个变量只能被声明一次。
- 变量以及表达式的类型可以是整型 int，布尔类型 bool 或者字符串类型 char，必须对变量的使用和表达式进行类型检查。

2.4 用 TINY+语言编写的示例程序

```
char str;
```

```
int x, fact;
```

```
str:= 'sample program in TINY+ language- computes factorial' ;
```

```
read x;
```

```
if x>0 and x<100 then {don't compute if x<=0}
```

```
    fact:=1;
```

```
    while x>0 do
```

fact:=fact*x;

x:=x-1

end;

write fact

end

3 实验 1：实现 TINY+语言的词法分析器

3.1 实验目的

学生在该实验中构造 TINY+语言的词法分析程序。具体要求如下：

- 1 词法分析器的输入是源程序文件，输出是单词（token）流。
- 2 构造的词法分析器必须遵循最长子串原则，例如字符串‘:=’ 应该被识别为一个代表赋值符号的单词，而不是识别为两个单词‘:’和‘=’。
- 3 词法分析器识别出的单词应该被表示成二元组的形式（Kind, Value），其中 Kind 表示单词的种类，Value 表示单词的实际值。

要求用如下符号表示不同种类的单词：

KEY 表示关键字；

SYM 表示特殊符号；

ID 表示标识符；

NUM 表述数字常量

STR 表示字符串常量

- 4 词法分析器的任务除了完成对单词的识别之外，还要检查程序中的词法错误，给出错误的信息以及错误在源程序中出现的位置（所在的行号）。词法错误的种类包括：
 - 非法符号，词法分析器可能识别到一个 TINY+程序的字母表中不允许的符号，例如，词法分析器在一个源程序中识别到

\$, 应报告一个词法错误, 发现了一个非法符号;

- 字符串类型的单词 **STRING**, 单引号不匹配。例如, 源程序中出现' **scanner**, 词法分析器应报告错误“字符串缺少右引号”;

注释的括号不匹配。例如源程序中出现{**this is an example**, 词法分析器应报告错误“注释缺少右括号”。

3.2 实验要求

- 1 用 C 语言或 C++语言构造词法分析器
- 2 实验学时为 4 学时。学生必须提交实验报告和词法分析器程序的源代码。

3.3 TINY+的测试程序及词法分析器的输出

Test1

```
or    and    int    bool    char
while do     if     then   else
end    repeat until  read    write
,      ;      :=     +      -
*      /      (      )      <
=      >      <=     >=     a2c
```

123 'EFG'

词法分析器应产生如下输出:

(KEY, or) (KEY, and) (KEY, int) (KEY, bool)

(KEY, char) (KEY, while) (KEY, do) (KEY, if)
 (KEY, then) (KEY, else) (KEY, end) (KEY, repeat)
 (KEY, until) (KEY, read) (KEY, write) (SYM, ,)
 (SYM, ;) (SYM, :=) (SYM, +) (SYM, -)
 (SYM, *) (SYM, /) (SYM, () (SYM,))
 (SYM, <) (SYM, =) (SYM, >) (SYM, <=)
 (SYM, >=) (ID, a2c) (NUM, 123) (STR, EFG)

Test2

{this is an example}

int A,B;

bool C1, C2, C3;

char D;

D:= 'scanner';

while A<=B do

A:=A*2

end

词法分析器应产生如下输出：

(KEY, int) (ID, A) (SYM, ,) (ID, B)

(SYM, ;) (KEY, bool) (ID, C1) (SYM, ,)

(ID, C2) (SYM, ,) (ID, C3) (SYM, ;)

(KEY, char)	(ID, D)	(SYM, ;)	(ID, D)
(SYM, :=)	(STR, scanner)	(SYM, ;)	(KEY, while)
(ID, A)	(SYM, <=)	(ID, B)	(KEY, do)
(ID, A)	(SYM, :=)	(ID, A)	(SYM, *)
(NUM, 2)	(KEY, end)		

4 实验 2：实现 TINY+的语法分析器、语义分析器以及中间代码生成器

4.1 实验目的

学生在该实验中构造 TINY+语言的语法分析器、语义分析器以及中间代码生成器，具体要求如下：

- 1 为 TINY+构造一个递归下降语法分析器。该语法分析器对词法分析器生成的单词序列进行语法分析，产生一棵抽象语法树作为语法分析器的输出。
- 2 构造 TINY+的语义分析器，该语义分析器构建符号表，然后检查程序中的语义错误。
- 3 构造 TINY+的中间代码生成系，该中间代码生成器将 TINY+程序翻译为三地址中间代码。
- 4 要求能检查程序中语法和语义错误，具体包括：
 - a) 语法错误：
 - 语法结构的开始符号以及跟随符号错误；
 - 标识符错误，例如在程序的变量声明中，关键字 *int* 后没有跟随标识符；
 - 括号不匹配的的错误，例如左括号(和右括号)不匹配。
 - 符号错误，例如赋值语句中要求使用的正确符号是‘:=’，而在关系比较表达式要求使用的正确符号是‘=’。

b) 语义错误:

- 一个标识符没有声明就使用, 以及一个标识符被声明不止一次。
- 一个条件表达式的类型不是布尔类型 *bool*。
- 一个二元操作符的两个操作数的类型不相等。
- 赋值语句左右部的类型不相等。

4.2 实验要求

- 1 用 C 语言或 C++语言实现
- 2 实验学时为 12 学时。学生必须提交实验报告和程序源代码。

4.3 TINY+示例程序及其输出

Test1

```
int  A,B,C,D;

while A<C and B>D do

    if A=1 then  A:= B*C+37

                else    repeat A:=A*2

                    until A+C<=B+D

                end

end
```

实验应产生如下输出:

说明: 假定 `stmt-sequence.next=L0`, 即假定 `L0` 是和语句序列 `stmt-sequence` (语法树的树根) 的代码结束后要执行的第一条三地址指

令对应标号。

1) Label L1	9) goto L5	17) t3:=A*2
2) if A<C goto L3	10) Label L4	18) A:=t3
3) goto L0	11) t1:=B*C	19) t4:=A+C
4) Label L3	12) t2:=t1+37	20) t5:=B+D
5) if B>D goto L2	13) A:=t2	21) if t4<=t5 goto L1
6) goto L0	14) goto L1	22) goto L6
7) Label L2	15) Label L5	23) Label L0
8) if A=1 goto L4	16) Label L6	

Test 2

int x,fact;

read x;

if x>0 and x<100 then {don't compute if x<=0}

fact:=1;

while x>0 do

fact:=fact*x;

x:=x-1

end;

write fact

end

实验应产生如下输出：

1) read x;	8) Label L2	16) fact:=t1
2) Label L1	9) fact:=1	17) Label L8
3) if x>0 goto L3	10) Label L4	18) t2:=x-1
4) goto L0	11) Label L6	19) x:=t2
5) Label L3	12) if x>0 goto L7	20) goto L6
6) if x<100 goto L2	13) goto L5	21)Label L5
7) goto L0	14) label L7	22) write fact
	15) t1:=fact*x	23) Label L0

附录：和 TINY+文法规则对应的生成三地址中间代码的属性文法

1 Seq->S ; Seq1

{S.next=newlabel; Seq1.next=Seq.next;

Seq.code=S.code || Label S.next || Seq1.code}

2 Seq->S

{S.next=Seq.next; Seq.code=S.code}

3 S->repeat Seq until E

{S.begin =newlabel; Seq.next=newlabel;

E.true=S.next; E.false=S.begin;

S.code=Label S.begin || Seq.code || Label Seq.next || E.code}

4 S->while E do Seq end

{S.begin=newlabel Seq.next=S.begin

E.true=newlabel E.false=S.next

S.code=Label S.begin || E.code || Label E.true || Seq.code || goto S.begin }

5 S->if E then Seq end

{E.true=newlabel; E.false=S.next; Seq.next=S.next

S.code=E.code || Label E.true || Seq.code}

6 S->if E then Seq1 else Seq2 end

{E.true=newlabel; E.false=newlabel;

Seq1.next=S.next; Seq2.next=S.next;

S.code=E.code || Label E.true || Seq1.code || goto S.next || Label E.false ||

Seq2.code }