目录

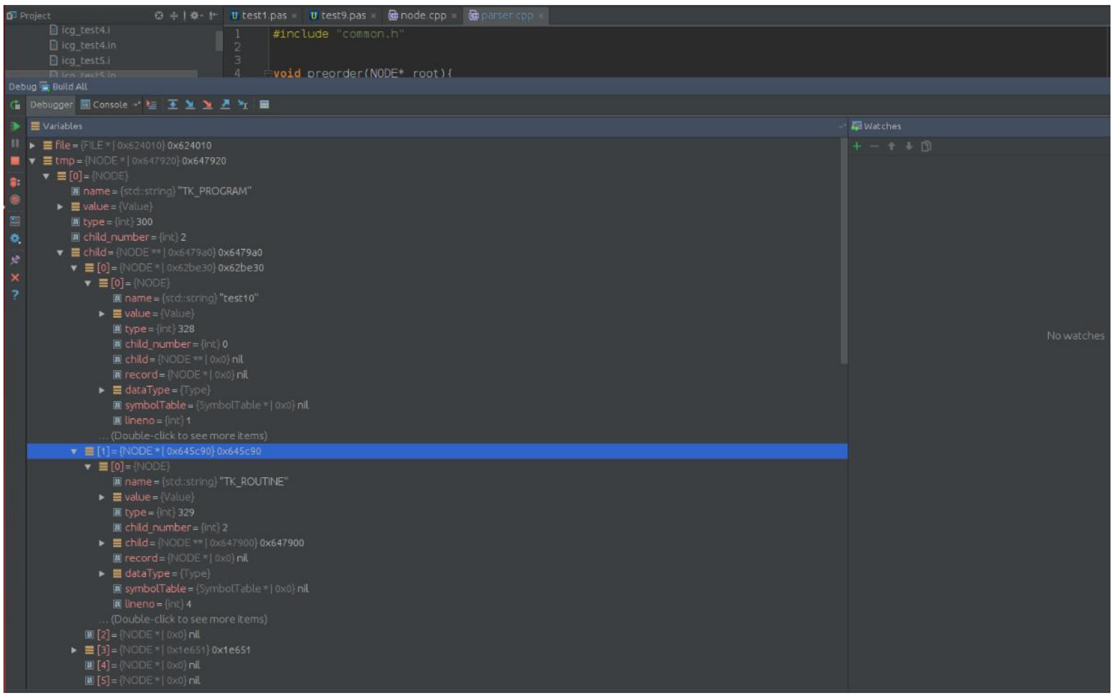
1. 语法分析测试........2
2. 语义分析与符号表测试........7
3. 代码生成测试........17
输入与输出........17
    运行结果:........18
算术运算符测试........18
    运行结果:........19
while 语句........20
    运行结果:........21
if 语句........21
case 语句........22
浮点数处理........23
函数调用........25
递归调用........26
结构体........28
常数计算优化........29
repeat........30
嵌套函数........31

1. 语法分析测试

完整测试，对于所有情况能否正确的建立语法树,测试代码如下:

```
program test10;

const
    haha = true;
    PI=3.14 ;
    fir = 1;
    sec = 10;
    LARGE = 10000202020202;

type
    arr1 = array[1..10] of integer;
     arr2 = array[-1..10] of integer;
     arr3 = array[-10..-1] of integer;
     arr4 = array[-fir..sec] of integer;
     arr5 = array[-sec..fir] of integer;
     arr6 = array[-sec..-fir] of integer;
    car = record
            number : string;
            legal : boolean;
            level : integer;
        end;

var
    a, b, c, n:integer;
    arr_var: arr1;
    hhh : integer;
    book: record
        title : integer;
        author : integer;
        content : string;
        pal : boolean;
    end;
    cc : car;
    test_str1 : string;

procedure findMin(x, y:integer; var m:integer);
procedure ptest1;
begin
end;
function ftest1:integer;
function ftest2(x1, y1:integer):real;
begin
end;
begin
end;
begin

    if x<y then
        m:=x
    else
        m:=y;
end;

procedure proc1;
begin
end;

function func1:integer;
begin
end;

begin
    read(a);
    readln(n);
    case n of
    1: writeln(0);
    2: writeln(1);
    3: begin
        a := 980*10+100;
        b := 2;
        end;
    end;

    arr_var[10]:=10;

    while true do
    begin
        a:=1;
        b:=1;
        c:=2;
    end;

    proc1;
    func1;
    book.title := 1;
    book.content := '123';

    a:=book.title;
    test_str1 := book.content;

    begin
       a:=a*2+-3-4;
       b:=(-a-b)*(3+4*5);
       c:=2;
    end;

       if (a > b) then
          c := a;
     if (a+8*9 = b) then
         c:=a;
     if (a+8*9 <> b) then
         c:=a;


     findMin(a,b,c);



       for a:= -1+2 to 100 do begin
          c:= c + 2+8*9;
     end;

     for b:= a+100*10 downto -1 do begin
          c:= c + 2+8*9;
     end;

end.
```

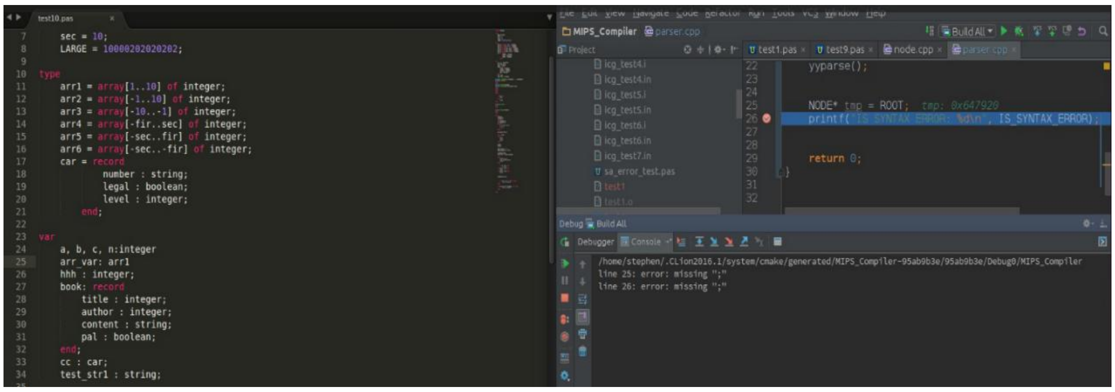
构建的语法树的节点在 IDE 的 DEBUG 模式下可以进行查看，下面是截图，检查所有的节点，语法树是合法的。



```
#include "common.h"


void preorder(NODE* root){
```

```
▶ file = {FILE * | 0x624010} 0x624010
▼ tmp = {NODE * | 0x647920} 0x647920
  ▼ [0] = {NODE}
      name = {std::string} "TK_PROGRAM"
    ▶ value = {Value}
      type = {int} 300
      child_number = {int} 2
    ▼ child = {NODE ** | 0x6479a0} 0x6479a0
      ▼ [0] = {NODE * | 0x62be30} 0x62be30
        ▼ [0] = {NODE}
            name = {std::string} "test10"
          ▶ value = {Value}
            type = {int} 328
            child_number = {int} 0
            child = {NODE ** | 0x0} nil
            record = {NODE * | 0x0} nil
          ▶ dataType = {Type}
            symbolTable = {SymbolTable * | 0x0} nil
            lineno = {int} 1
          ... (Double-click to see more items)
      ▼ [1] = {NODE * | 0x645c90} 0x645c90
        ▼ [0] = {NODE}
            name = {std::string} "TK_ROUTINE"
          ▶ value = {Value}
            type = {int} 329
            child_number = {int} 2
          ▶ child = {NODE ** | 0x647900} 0x647900
            record = {NODE * | 0x0} nil
          ▶ dataType = {Type}
            symbolTable = {SymbolTable * | 0x0} nil
            lineno = {int} 4
          ... (Double-click to see more items)
        [2] = {NODE * | 0x0} nil
      ▶ [3] = {NODE * | 0x1e651} 0x1e651
        [4] = {NODE * | 0x0} nil
        [5] = {NODE * | 0x0} nil
```

然后进行错误处理测试，可以正确检测到没有两个’；’：

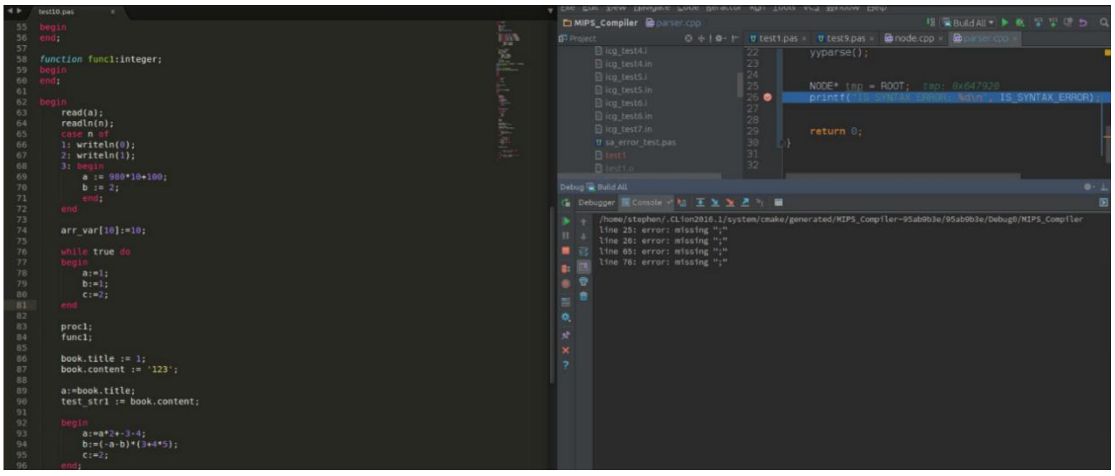


```
7       sec = 10;
8       LARGE = 10000202020202;
9
10  type
11      arr1 = array[1..10] of integer;
12      arr2 = array[-1..10] of integer;
13      arr3 = array[-10..-1] of integer;
14      arr4 = array[-fir..sec] of integer;
15      arr5 = array[-sec..fir] of integer;
16      arr6 = array[-sec..-fir] of integer;
17      car = record
18              number : string;
19              legal : boolean;
20              level : integer;
21          end;
22
23  var
24      a, b, c, n:integer
25      arr_var: arr1
26      hhh : integer;
27      book: record
28          title : integer;
29          author : integer;
30          content : string;
31          pal : boolean;
32      end;
33      cc : car;
34      test_str1 : string;
```

```
22      yyparse();
23
24
25      NODE* tmp = ROOT;   tmp: 0x647920
26      printf("IS SYNTAX ERROR: %d\n", IS_SYNTAX_ERROR);
27
28
29      return 0;
30  }
31
32
```

```
/home/stephen/.CLion2016.1/system/cmake/generated/MIPS_Compiler-95ab9b3e/95ab9b3e/Debug0/MIPS_Compiler
line 25: error: missing ";"
line 26: error: missing ";"
```

继续删除’；’，发现仍然可以检测到:

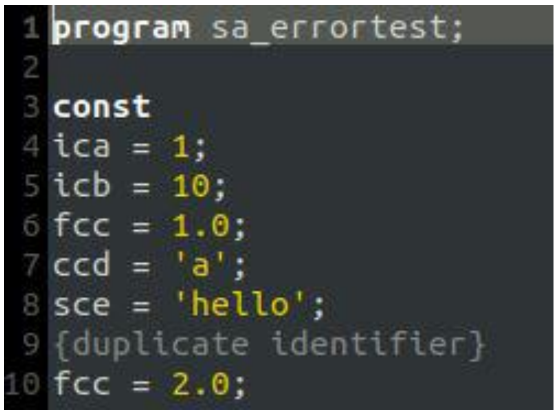


```
55  begin
56  end;
57
58  function func1:integer;
59  begin
60  end;
61
62  begin
63      read(a);
64      readln(n);
65      case n of
66      1: writeln(0);
67      2: writeln(1);
68      3: begin
69          a := 980*10+100;
70          b := 2;
71          end;
72      end
73
74      arr_var[10]:=10;
75
76      while true do
77      begin
78          a:=1;
79          b:=1;
80          c:=2;
81      end
82
83      proc1;
84      func1;
85
86      book.title := 1;
87      book.content := '123';
88
89      a:=book.title;
90      test_str1 := book.content;
91
92      begin
93          a:=a*2+-3-4;
94          b:=(-a-b)*(3+4*5);
95          c:=2;
96      end;
```

```
/home/stephen/.CLion2016.1/system/cmake/generated/MIPS_Compiler-95ab9b3e/95ab9b3e/Debug0/MIPS_Compiler
line 25: error: missing ";"
line 26: error: missing ";"
line 65: error: missing ";"
line 76: error: missing ";"
```

2. 语义分析与符号表测试

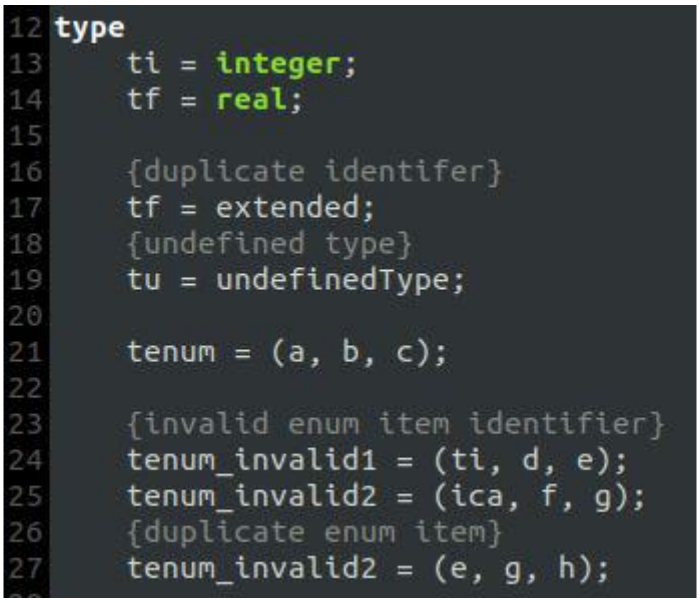
在源文件目录的 testcase 文件夹下有一个 sa_error_test.pas，如果将该源文件解析出的 AST 送入 semanticAnalysis 函数，可以输出所有可以检查到的错误，下面我们对该文件进行分析:



```
program sa_errortest;

const
ica = 1;
icb = 10;
fcc = 1.0;
ccd = 'a';
sce = 'hello';
{duplicate identifier}
fcc = 2.0;
```

首先是一系列常量定义，当发现常量名与之前重复时，报 duplicate identifier 错误。

然后是类型定义:

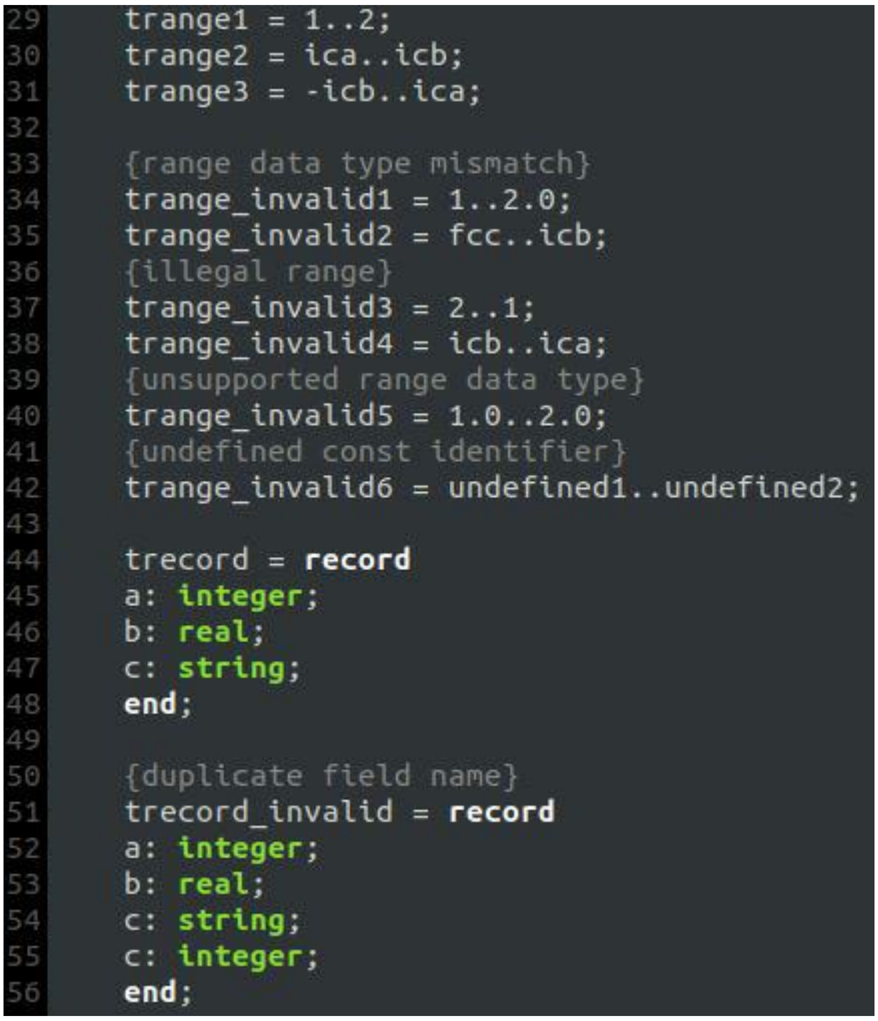


```
type
    ti = integer;
    tf = real;

    {duplicate identifer}
    tf = extended;
    {undefined type}
    tu = undefinedType;

    tenum = (a, b, c);

    {invalid enum item identifier}
    tenum_invalid1 = (ti, d, e);
    tenum_invalid2 = (ica, f, g);
    {duplicate enum item}
    tenum_invalid2 = (e, g, h);
```

与常量定义类似，如果发现类型名与之前的标识符重复，也报 duplicate identifier 错误。当等号右侧不是原生类型时，如果符号表中也找不到该标识符所代表的类型，则报 undefined type 错误。

当定义枚举类型时，要求等号右侧的枚举量标识符不能已被使用（如已被定义为常量或者类型），否则报 invalid enum item identifier 错误

当枚举量标识符已在之前的枚举类型中被定义，则报 duplicate enum item 错误。

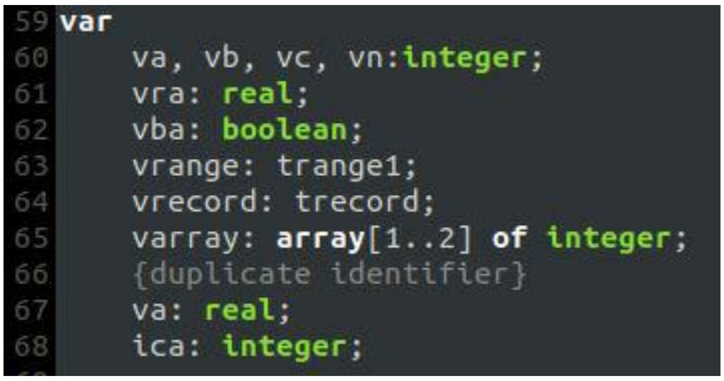


```
    trange1 = 1..2;
    trange2 = ica..icb;
    trange3 = -icb..ica;

    {range data type mismatch}
    trange_invalid1 = 1..2.0;
    trange_invalid2 = fcc..icb;
    {illegal range}
    trange_invalid3 = 2..1;
    trange_invalid4 = icb..ica;
    {unsupported range data type}
    trange_invalid5 = 1.0..2.0;
    {undefined const identifier}
    trange_invalid6 = undefined1..undefined2;

    trecord = record
    a: integer;
    b: real;
    c: string;
    end;

    {duplicate field name}
    trecord_invalid = record
    a: integer;
    b: real;
    c: string;
    c: integer;
    end;
```

在定义范围类型时，要求其范围一致且为整数类型，且左侧小于等于右侧，不一致时报 range data type mismatch，左侧大于右侧时报 illegal range，不为整数类型时报 unsupported range data type，当找不到该常量标识符时报 undefined const identifier。

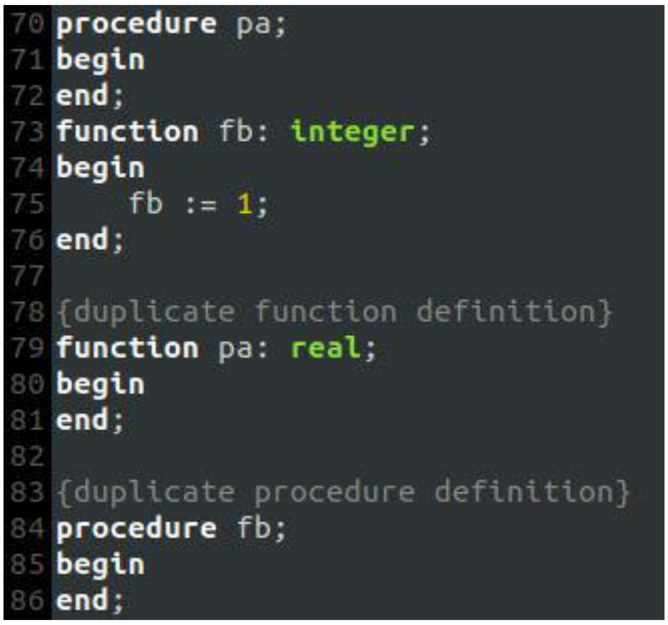
定义 record 时，如果域名有重复，则报 duplicate field name 错。

变量定义:



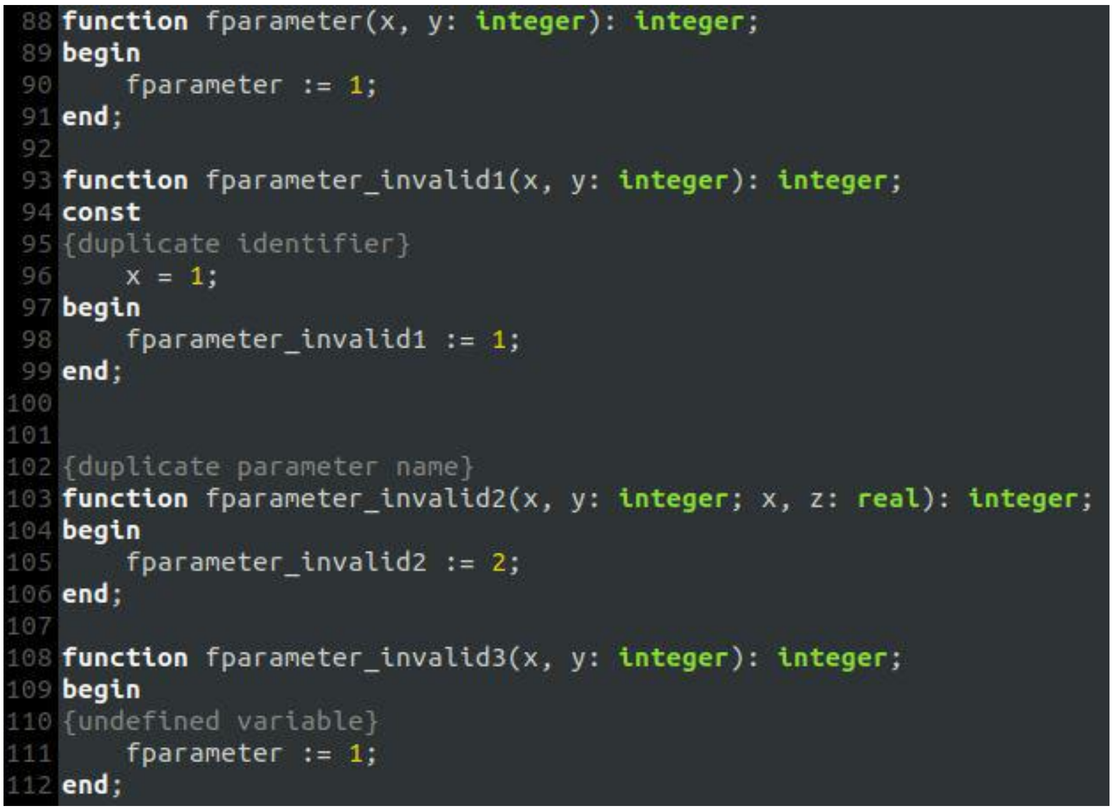
```
var
    va, vb, vc, vn:integer;
    vra: real;
    vba: boolean;
    vrange: trange1;
    vrecord: trecord;
    varray: array[1..2] of integer;
    {duplicate identifier}
    va: real;
    ica: integer;
```

由于变量定义与类型定义类似，类型定义部分所报的错误变量定义都会报，同时变量部分检查是否定义了重名变量。



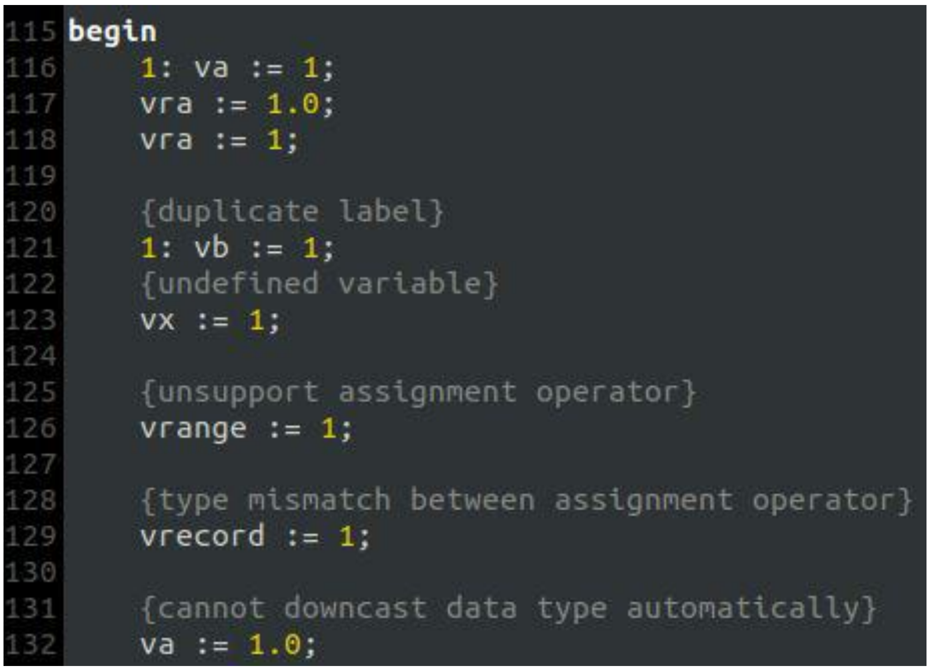
```
procedure pa;
begin
end;
function fb: integer;
begin
    fb := 1;
end;

{duplicate function definition}
function pa: real;
begin
end;

{duplicate procedure definition}
procedure fb;
begin
end;
```

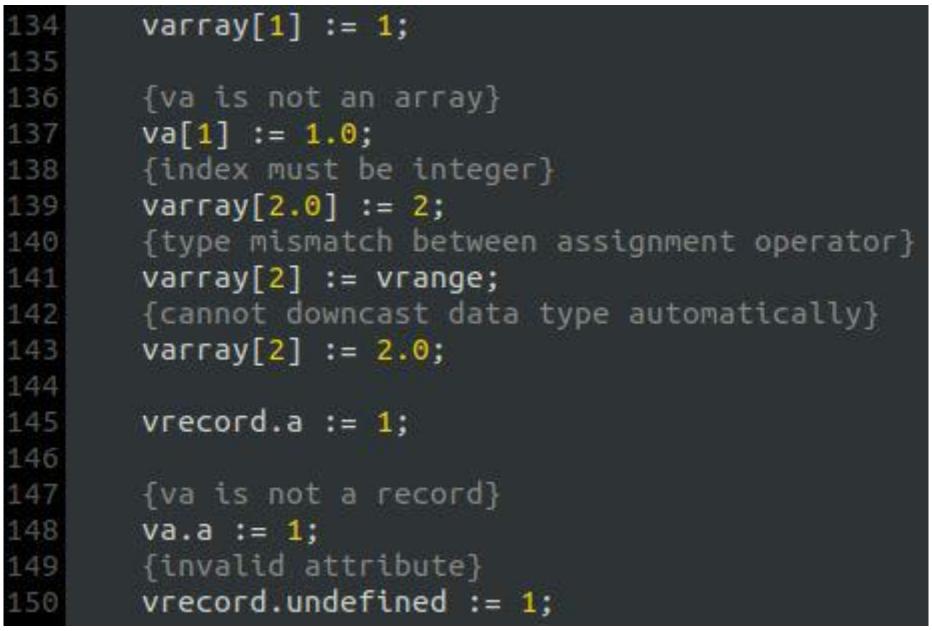
函数与过程根据函数名与其类型构成函数签名作为判断是否重名的依据。上图中 pa 虽然一个是过程，一个是类型，但仍判定下面的 pa 与 fb 重名。



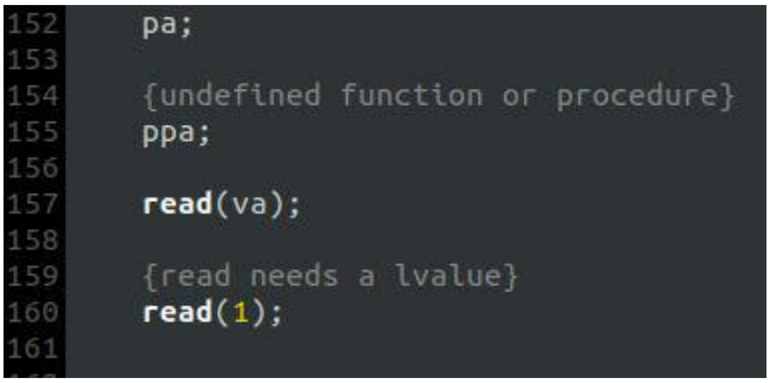
```
function fparameter(x, y: integer): integer;
begin
    fparameter := 1;
end;

function fparameter_invalid1(x, y: integer): integer;
const
{duplicate identifier}
    x = 1;
begin
    fparameter_invalid1 := 1;
end;


{duplicate parameter name}
function fparameter_invalid2(x, y: integer; x, z: real): integer;
begin
    fparameter_invalid2 := 2;
end;

function fparameter_invalid3(x, y: integer): integer;
begin
{undefined variable}
    fparameter := 1;
end;
```

在函数中，不能定义与参数重名的变量/常量/类型，否则会报 duplicate identifier 错误。函数参数本身也不能重名，否则报 duplicate parameter name 错误。函数内部支持对同名变量进行赋值，该变量没有显式定义，但其他的未定义变量仍旧会报 undefined variable 错误。


```
begin
    1: va := 1;
    vra := 1.0;
    vra := 1;

    {duplicate label}
    1: vb := 1;
    {undefined variable}
    vx := 1;

    {unsupport assignment operator}
    vrange := 1;

    {type mismatch between assignment operator}
    vrecord := 1;

    {cannot downcast data type automatically}
    va := 1.0;
```
当定义了重复的 goto label 时，程序会报 duplicate label 错误。对于不支持赋值操作的变量，会报 unsupport assignment operator 错误，而如果试图将 integer 赋值给 record 则会报 type mismatch between assignment operator 错误。本编译器支持类型的自动提升，但不支持类型的隐式降低，如将 real 赋值给 integer，此时会报 cannot downcast data type automatically 错误。

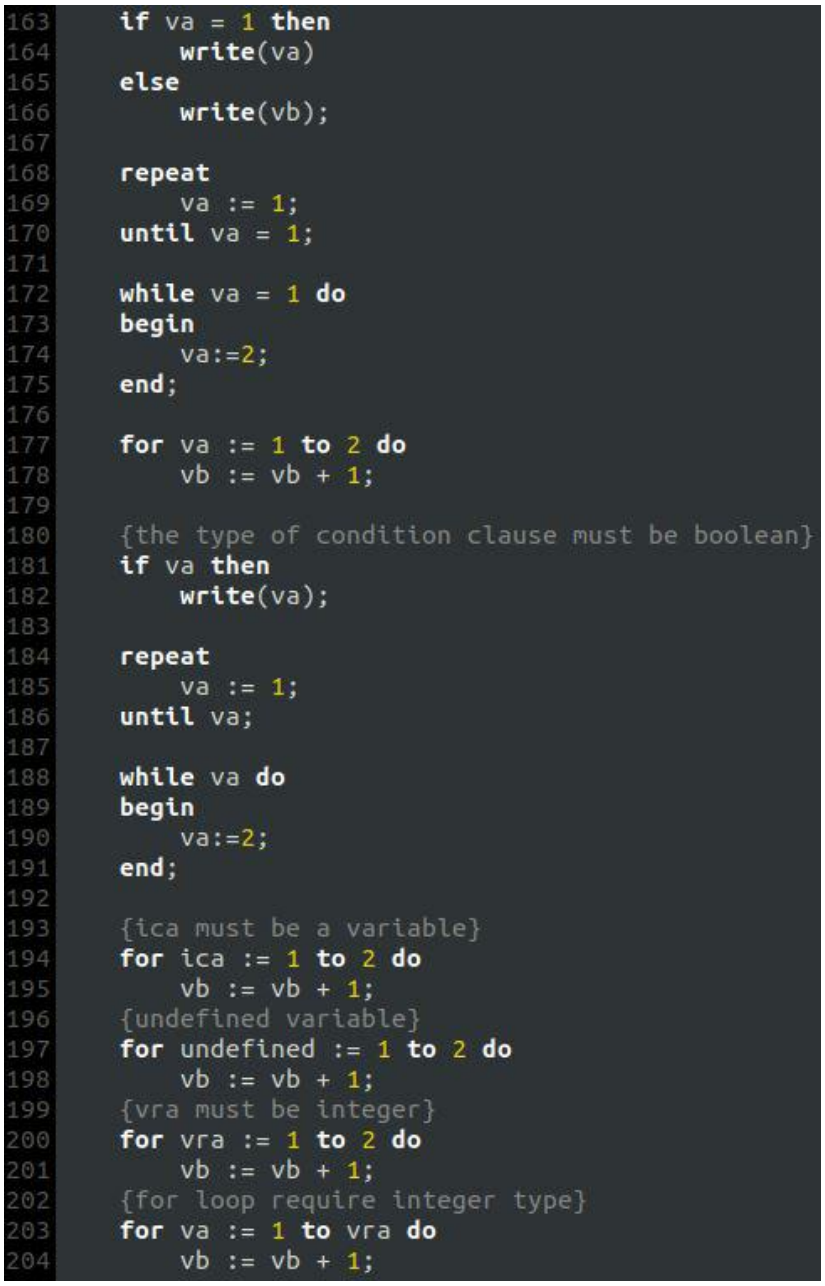

```
    varray[1] := 1;

    {va is not an array}
    va[1] := 1.0;
    {index must be integer}
    varray[2.0] := 2;
    {type mismatch between assignment operator}
    varray[2] := vrange;
    {cannot downcast data type automatically}
    varray[2] := 2.0;

    vrecord.a := 1;

    {va is not a record}
    va.a := 1;
    {invalid attribute}
    vrecord.undefined := 1;
```
当试图对非数组进行数组成员访问时，将会报 xx is not an array 错误。如果下标类型不是 integer，则会报 index must be integer 错误。

试图对非 record 进行 record 域成员访问是，会报 xx is not a record 错误，试图访问一个 record 中不存在的域成员时，会报 invalid attribute 错误。



```
    pa;

    {undefined function or procedure}
    ppa;

    read(va);

    {read needs a lvalue}
    read(1);

```

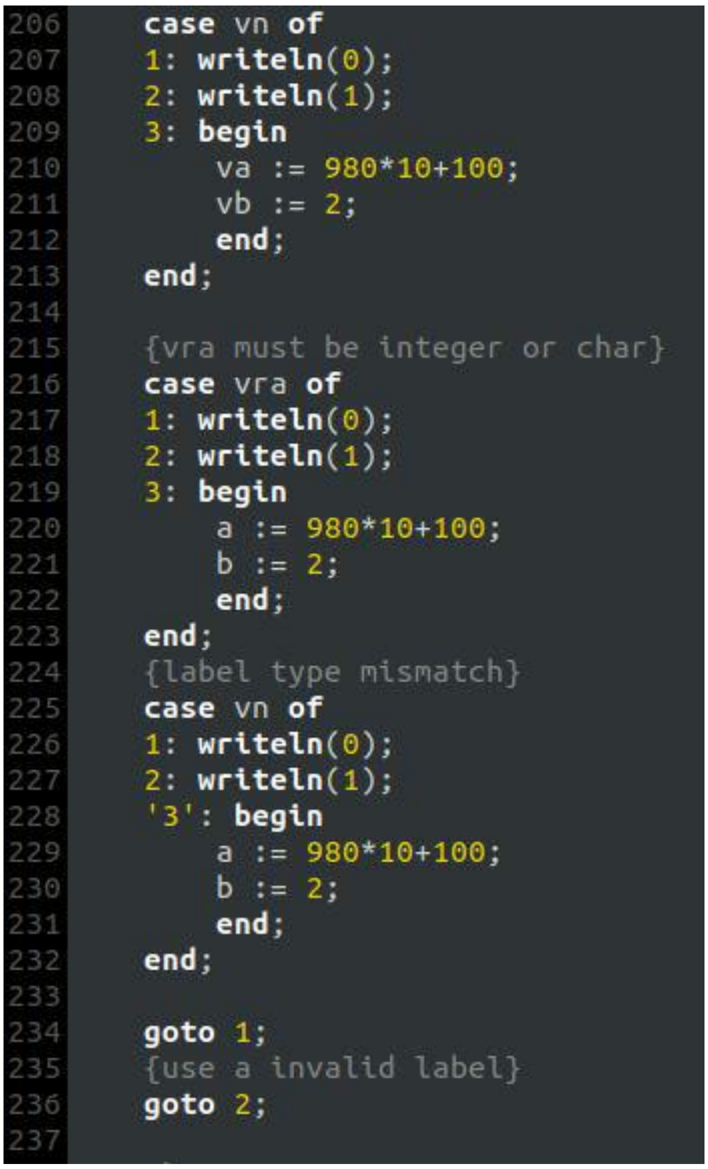
试图调用一个不存在的函数/过程时，会报 undefined function or procedure 错误，如果试图传递一个非左值给需要参数为左值的函数，会报 xx needs a lvalue 错误。



```
    if va = 1 then
        write(va)
    else
        write(vb);

    repeat
        va := 1;
    until va = 1;

    while va = 1 do
    begin
        va:=2;
    end;

    for va := 1 to 2 do
        vb := vb + 1;

    {the type of condition clause must be boolean}
    if va then
        write(va);

    repeat
        va := 1;
    until va;

    while va do
    begin
        va:=2;
    end;

    {ica must be a variable}
    for ica := 1 to 2 do
        vb := vb + 1;
    {undefined variable}
    for undefined := 1 to 2 do
        vb := vb + 1;
    {vra must be integer}
    for vra := 1 to 2 do
        vb := vb + 1;
    {for loop require integer type}
    for va := 1 to vra do
        vb := vb + 1;
```

if，repeat-until，while 的条件语句要求类型为 boolean，当类型非 boolean 时报 the type of condition clause must be boolean 错误。

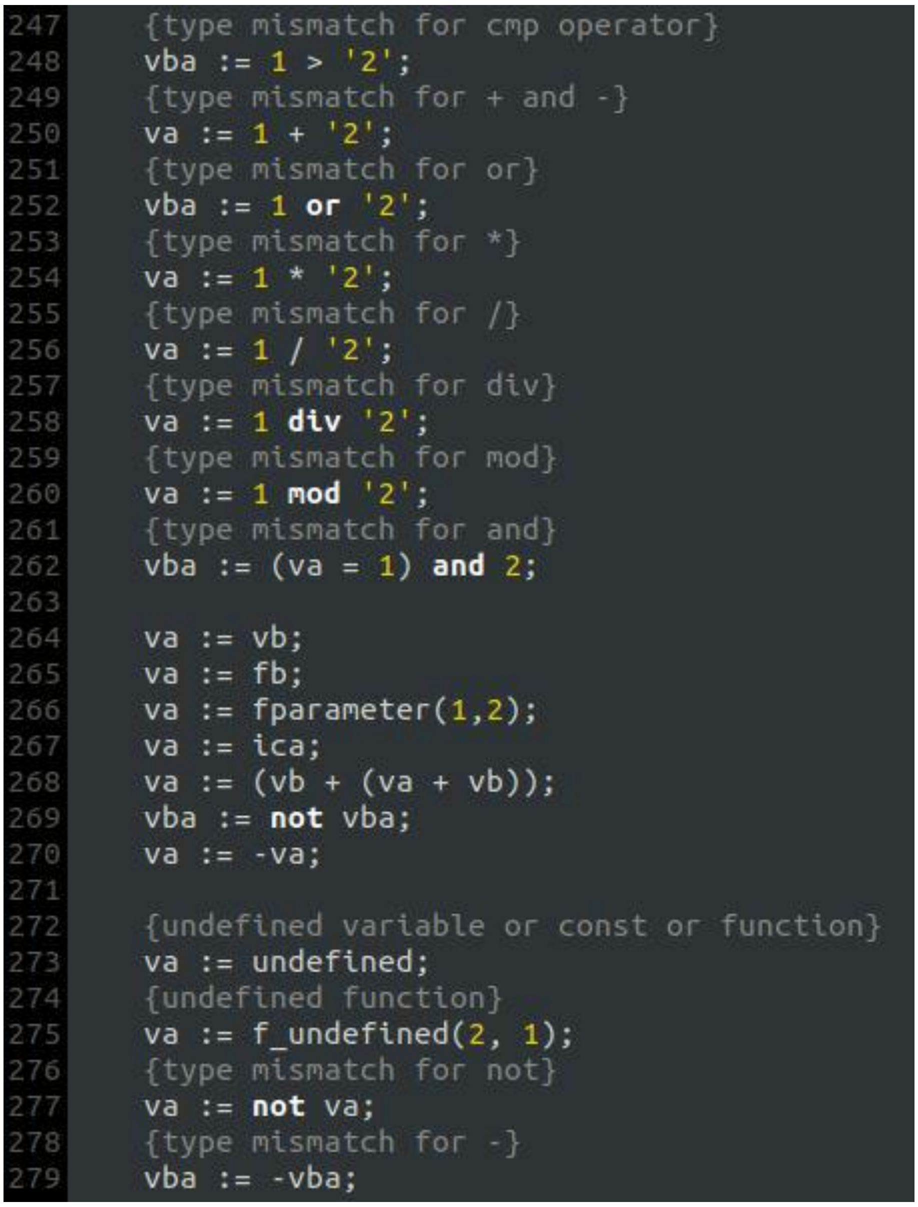
当 for 的循环变量不是变量或未定义时，报 xx must be a variable/undefined variable 错误。当其不是 integer 类型时，报 xx must be integer 错误。当其循环边界不是 integer 类型时，报 for loop require integer type 错误。



```
    case vn of
    1: writeln(0);
    2: writeln(1);
    3: begin
        va := 980*10+100;
        vb := 2;
        end;
    end;

    {vra must be integer or char}
    case vra of
    1: writeln(0);
    2: writeln(1);
    3: begin
        a := 980*10+100;
        b := 2;
        end;
    end;
    {label type mismatch}
    case vn of
    1: writeln(0);
    2: writeln(1);
    '3': begin
        a := 980*10+100;
        b := 2;
        end;
    end;

    goto 1;
    {use a invalid label}
    goto 2;

```

case 语句要求检查的变量为 integer 与 char，当变量不是上述两种类型时，报 xx must be integer or char 错误。变量类型与下面的分支类型不匹配时，报 label type mismatch 错误。

当试图 goto 一个不存在的 label 时，报 use a invalid label 错误。

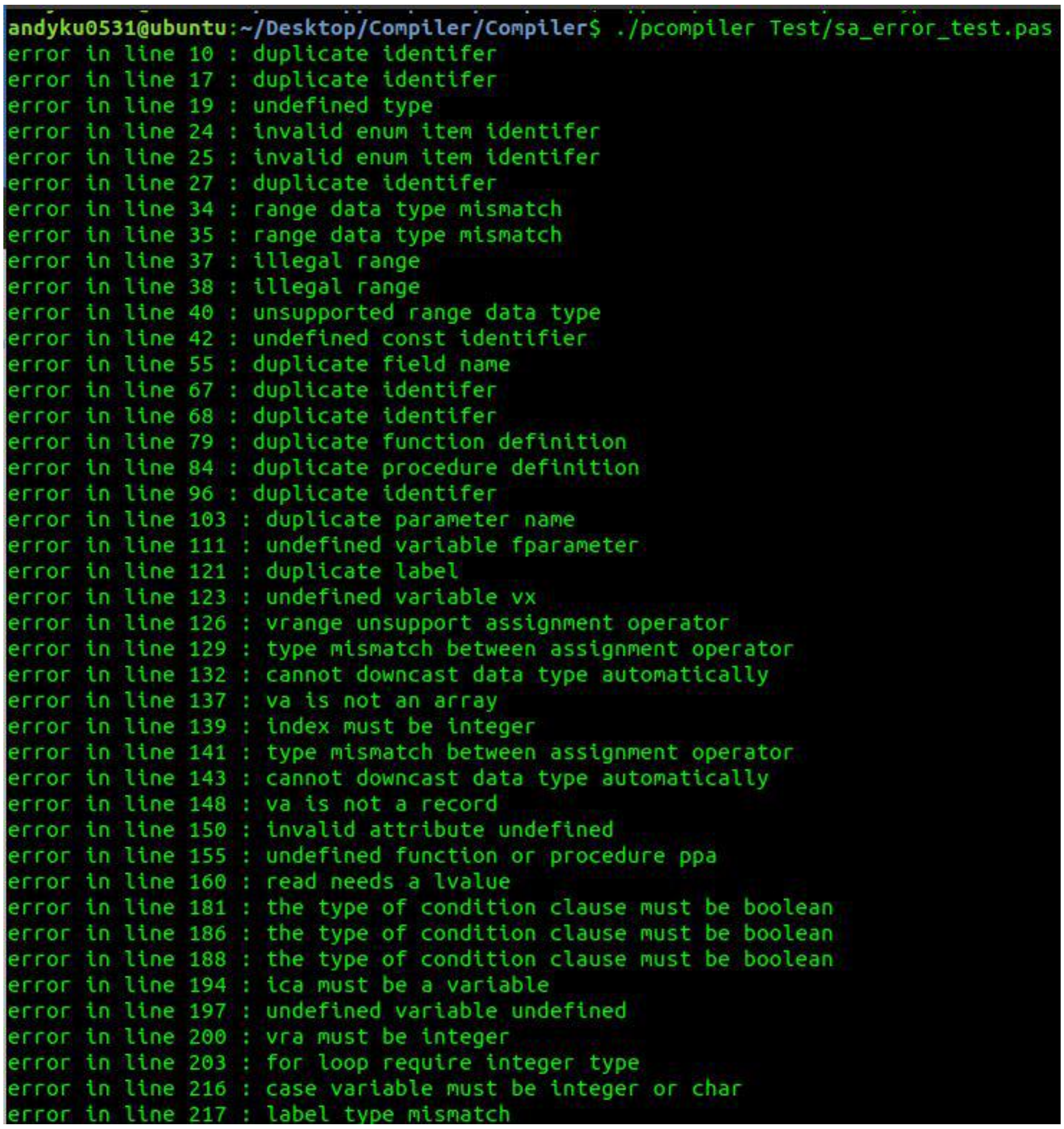


```
    {type mismatch for cmp operator}
    vba := 1 > '2';
    {type mismatch for + and -}
    va := 1 + '2';
    {type mismatch for or}
    vba := 1 or '2';
    {type mismatch for *}
    va := 1 * '2';
    {type mismatch for /}
    va := 1 / '2';
    {type mismatch for div}
    va := 1 div '2';
    {type mismatch for mod}
    va := 1 mod '2';
    {type mismatch for and}
    vba := (va = 1) and 2;

    va := vb;
    va := fb;
    va := fparameter(1,2);
    va := ica;
    va := (vb + (va + vb));
    vba := not vba;
    va := -va;

    {undefined variable or const or function}
    va := undefined;
    {undefined function}
    va := f_undefined(2, 1);
    {type mismatch for not}
    va := not va;
    {type mismatch for -}
    vba := -vba;
```

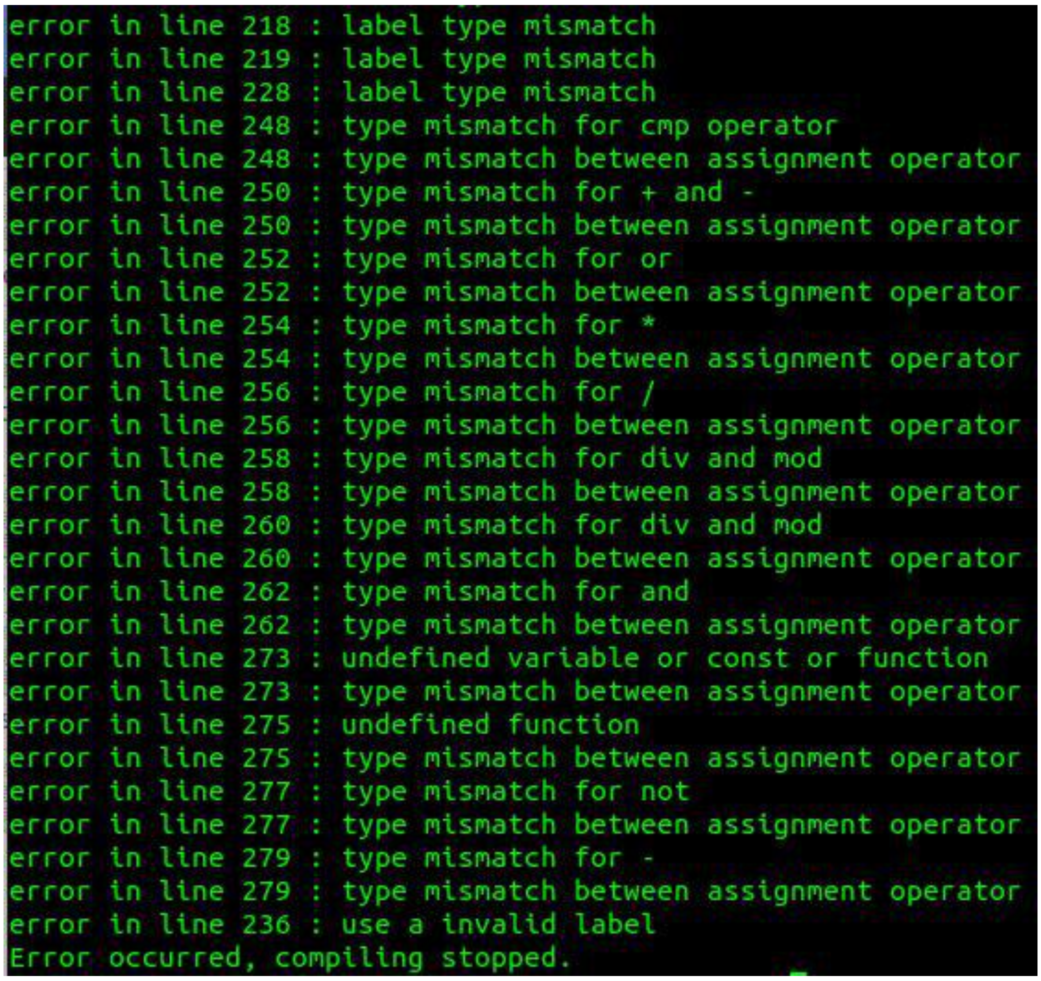
上述代码检查 expression 中的各种类型不一致，包括比较运算，+, -, *, /, div, mod, 以及逻辑运算。

当赋值符号的右边是标识符时，会检查 constSymbolTable, varSymbolTable, funcSymbolTable 三个符号表，都找不到时提示 undefined variable or const or function。如果根据语法确定其是一个函数时，则提示 undefined function。

最后程序的输出截图如下:



```
andyku0531@ubuntu:~/Desktop/Compiler/Compiler$ ./pcompiler Test/sa_error_test.pas
error in line 10 : duplicate identifer
error in line 17 : duplicate identifer
error in line 19 : undefined type
error in line 24 : invalid enum item identifer
error in line 25 : invalid enum item identifer
error in line 27 : duplicate identifer
error in line 34 : range data type mismatch
error in line 35 : range data type mismatch
error in line 37 : illegal range
error in line 38 : illegal range
error in line 40 : unsupported range data type
error in line 42 : undefined const identifier
error in line 55 : duplicate field name
error in line 67 : duplicate identifer
error in line 68 : duplicate identifer
error in line 79 : duplicate function definition
error in line 84 : duplicate procedure definition
error in line 96 : duplicate identifer
error in line 103 : duplicate parameter name
error in line 111 : undefined variable fparameter
error in line 121 : duplicate label
error in line 123 : undefined variable vx
error in line 126 : vrange unsupport assignment operator
error in line 129 : type mismatch between assignment operator
error in line 132 : cannot downcast data type automatically
error in line 137 : va is not an array
error in line 139 : index must be integer
error in line 141 : type mismatch between assignment operator
error in line 143 : cannot downcast data type automatically
error in line 148 : va is not a record
error in line 150 : invalid attribute undefined
error in line 155 : undefined function or procedure ppa
error in line 160 : read needs a lvalue
error in line 181 : the type of condition clause must be boolean
error in line 186 : the type of condition clause must be boolean
error in line 188 : the type of condition clause must be boolean
error in line 194 : ica must be a variable
error in line 197 : undefined variable undefined
error in line 200 : vra must be integer
error in line 203 : for loop require integer type
error in line 216 : case variable must be integer or char
error in line 217 : label type mismatch
```



```
error in line 218 : label type mismatch
error in line 219 : label type mismatch
error in line 228 : label type mismatch
error in line 248 : type mismatch for cmp operator
error in line 248 : type mismatch between assignment operator
error in line 250 : type mismatch for + and -
error in line 250 : type mismatch between assignment operator
error in line 252 : type mismatch for or
error in line 252 : type mismatch between assignment operator
error in line 254 : type mismatch for *
error in line 254 : type mismatch between assignment operator
error in line 256 : type mismatch for /
error in line 256 : type mismatch between assignment operator
error in line 258 : type mismatch for div and mod
error in line 258 : type mismatch between assignment operator
error in line 260 : type mismatch for div and mod
error in line 260 : type mismatch between assignment operator
error in line 262 : type mismatch for and
error in line 262 : type mismatch between assignment operator
error in line 273 : undefined variable or const or function
error in line 273 : type mismatch between assignment operator
error in line 275 : undefined function
error in line 275 : type mismatch between assignment operator
error in line 277 : type mismatch for not
error in line 277 : type mismatch between assignment operator
error in line 279 : type mismatch for -
error in line 279 : type mismatch between assignment operator
error in line 236 : use a invalid label
Error occurred, compiling stopped.
```

3. 代码生成测试

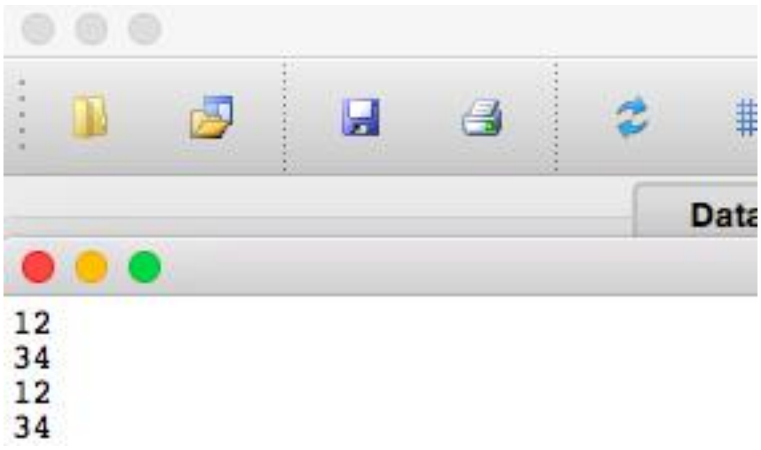
输入与输出

Pascal 源码	三地址码	汇编代码

program test1;
var
    n,m:integer;
begin
    readln(n,m);
    writeln(n,m);
end.
	entry main
	sp = sp - 8
	point int t0
	t0 = bp - 4
	read t0
	point int t0
	t0 = bp - 8
	readln t0
	point int t0
	t0 = bp - 4
	var int t1
	t1 = *t0
	println t1
		(由于汇编代码较长，放在文档里会显得很乱，故在测试报告里没有放汇编代码。老师若是想查看汇编代码，可以在工程目录下的 Assemble 文件下自行查看)
		
		汇编代码可在 Assemble/1.asm 中查看

	point int t0
	t0 = bp - 8
	var int t1
	t1 = *t0
	println t1
	sp = sp + 8	

运行结果：



```
12
34
12
34
```

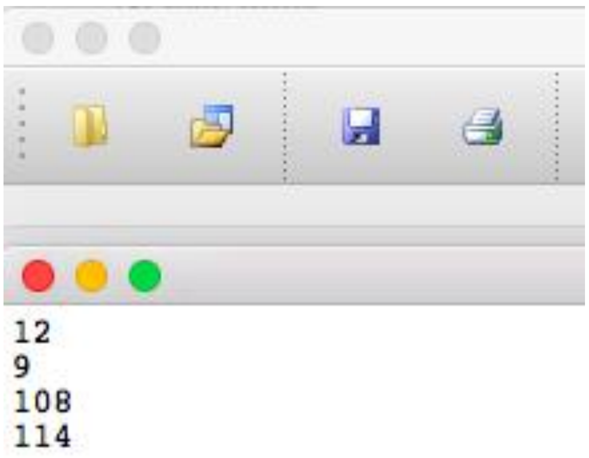
算术运算符测试

Pascal 源码	三地址码	汇编代码

program test2;
var
    a,b,c,d:longint;
begin
    a:=10;
    a:=a+2;
    b:=a-3;
    c:=a*b;
    d:=12345 div c;
    writeln(a);
    writeln(b);
    writeln(c);
    writeln(d);
end.
	entry main
	sp = sp - 16
	point int t0
	t0 = bp - 4
	*t0 = 10
	point int t0
	t0 = bp - 4
	point int t1
	t1 = bp - 4
	var int t2
	t2 = *t1 + 2
	*t0 = t2
	point int t0
	t0 = bp - 8
	point int t1
		汇编代码可在 Assemble/2.asm 中查看

```
t1 = bp - 4
var int t2
t2 = *t1 - 3
*t0 = t2
point int t0
t0 = bp - 12
point int t1
t1 = bp - 4
point int t2
t2 = bp - 8
var int t3
t3 = *t1 * *t2
*t0 = t3
point int t0
t0 = bp - 16
point int t1
t1 = bp - 12
var int t2
t2 = 12345 DIV *t1
*t0 = t2
point int t0
t0 = bp - 4
var int t1
t1 = *t0
println t1
point int t0
t0 = bp - 8
var int t1
t1 = *t0
println t1
point int t0
t0 = bp - 12
var int t1
t1 = *t0
println t1
point int t0
t0 = bp - 16
var int t1
t1 = *t0
println t1
sp = sp + 16
```

运行结果:



```
12
9
108
114
```

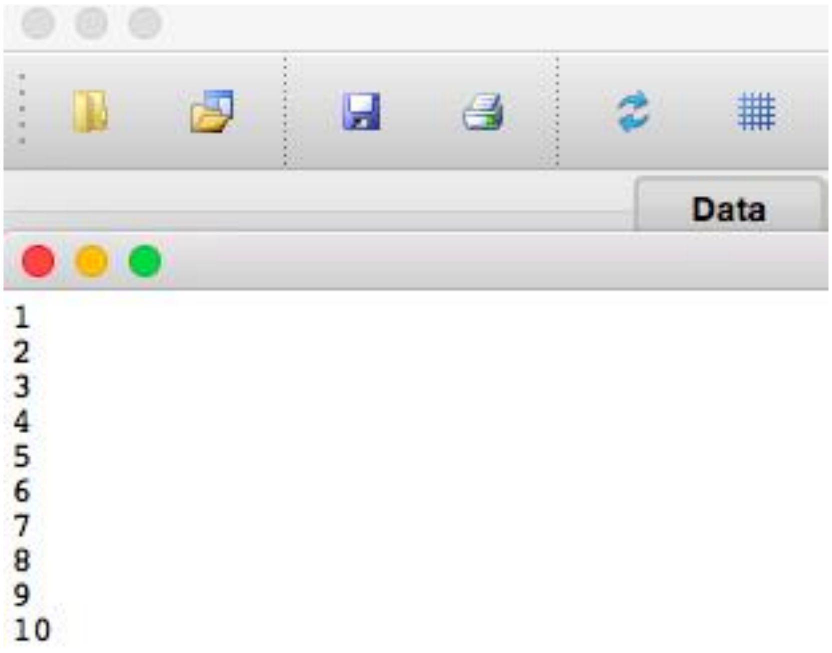
while 语句

Pascal 源码	三地址码	汇编代码

program test3;
var
    i:integer;
begin
    i:=0;
    while (i<10) do begin
        i:=i+1;
        writeln(i);
    end;
end.
	entry main
	sp = sp - 4
	point int t0
	t0 = bp - 4
	*t0 = 0
	label L0
	point int t0
	t0 = bp - 4
	var boolean t1
	t1 = *t0 < 10
	if_false t1 goto L1
	point int t0
	t0 = bp - 4
	point int t2
	t2 = bp - 4
	var int t3
	t3 = *t2 + 1
	*t0 = t3
	point int t0
	t0 = bp - 4
	var int t2
	t2 = *t0
	println t2
	goto L0
	label L1
		汇 编 代 码 可 在 Assemble/3.asm 中查看

	sp = sp + 4	

运行结果:



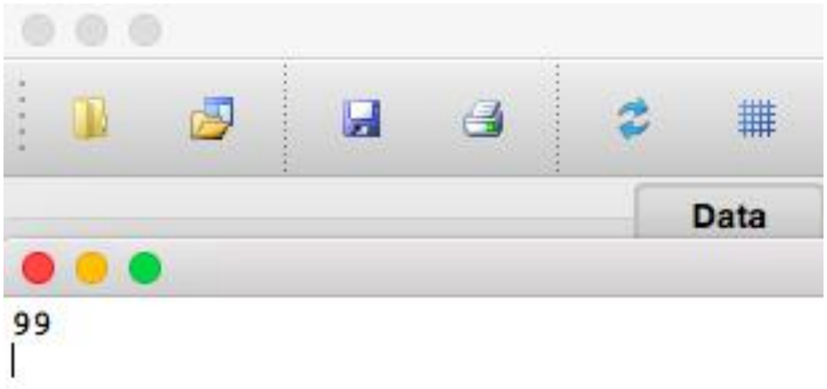
```
1
2
3
4
5
6
7
8
9
10
```

if 语句

Pascal 源码	三地址码	汇编代码

program test4;
var
    a,b:integer;
begin
    a := 123;
    b := 345;
   if (a>b) then
        writeln(100)
   else
       writeln(99);
end.
	entry main
	sp = sp - 8
	point int t0
	t0 = bp - 4
	*t0 = 123
	point int t0
	t0 = bp - 8
	*t0 = 345
	point int t0
	t0 = bp - 4
	point int t1
	t1 = bp - 8
	var boolean t2
	t2 = *t0 > *t1
	if_false t2 goto L0
	println 100
	goto L1
		汇编代码可在 Assemble/4.asm 中查看

	label L0 println 99 label L1 sp = sp + 8	



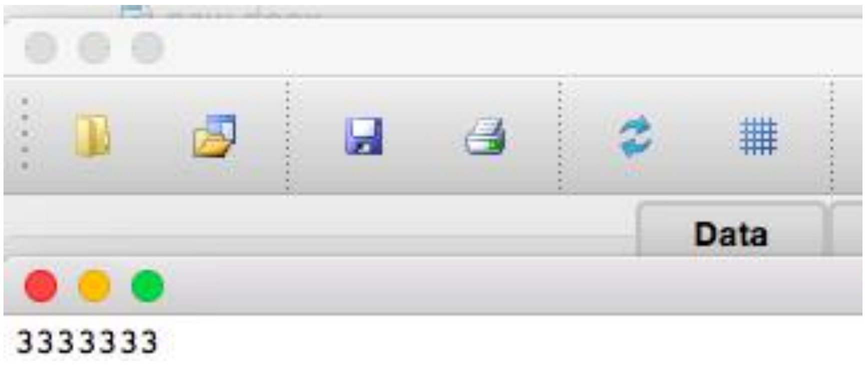
```
99
```

case 语句

Pascal 源码	三地址码	汇编代码

program test5;
var
    c:integer;
begin
    c:=3;
    case (c) of
 	
1:writeln(1111111);
 	
2:writeln(2222222);
 	
3:writeln(3333333);
 	
4:writeln(4444444);
    end;
end.
	entry main
	sp = sp - 4
	point int t0
	t0 = bp - 4
	*t0 = 3
	point int t0
	t0 = bp - 4
	goto L0
	Label L2
	println 1111111
	goto L1
	Label L3
	println 2222222
	goto L1
	Label L4
	println 3333333
	goto L1
	Label L5
	println 4444444
	goto L1
	label L0
	if *t0 == 1 then goto L2
		汇 编 代 码 可 在 Assemble/5.asm 中查看

	if *t0 == 2 then goto L3
	if *t0 == 3 then goto L4
	if *t0 == 4 then goto L5
	label L1
	sp = sp + 4	



```
3333333
```

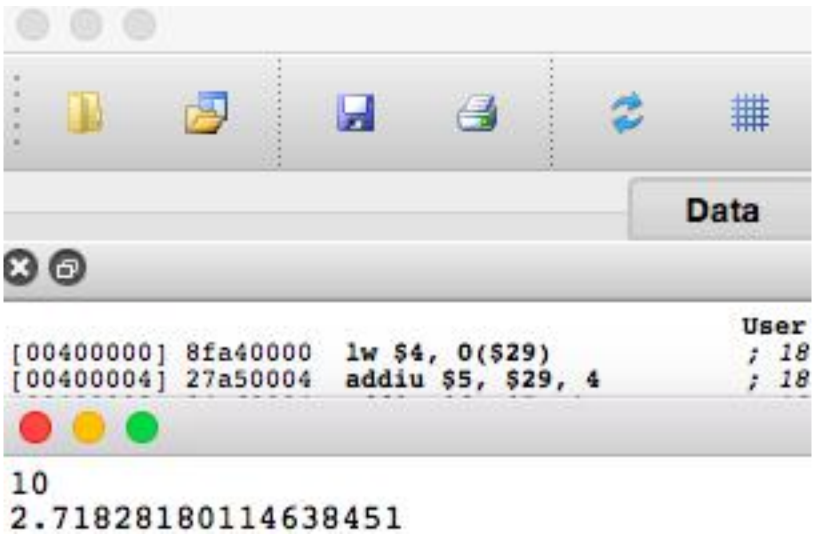
浮点数处理

Pascal 源码	三地址码	汇编代码

program test6;
var
    n,fact,i:longint;
    a:real;
begin
    read(n);
    fact:=1;
    a:=1.0;
    for i:=1 to n do
        begin
 	
fact:=fact*i;
 	
a:=a+1/fact;
        end;
    writeln(a);
end.
	entry main
	sp = sp - 20
	point int t0
	t0 = bp - 4
	read t0
	point int t0
	t0 = bp - 8
	*t0 = 1
	point double t0
	t0 = bp - 20
	*t0 = 1
	point int t0
	t0 = bp - 12
	point int t1
	t1 = bp - 4
	*t0 = 1
	var int t2
	t2 = -1
	label L0
	var int t3
	t3 = *t0 != *t1
	if_false t3 goto L1
		汇编代码可在 Assemble/6.asm 中查看

```
point int t4
t4 = bp - 8
point int t5
t5 = bp - 8
point int t6
t6 = bp - 12
var int t7
t7 = *t5 * *t6
*t4 = t7
point double t4
t4 = bp - 20
point double t5
t5 = bp - 20
point int t6
t6 = bp - 8
var double t7
t7 = 1 / *t6
var double t6
t6 = *t5 + t7
*t4 = t6
*t0 = *t0 + t2
goto L0
label L1
label L2
var int t3
t3 = *t0 == *t1
if_false t3 goto L3
point int t4
t4 = bp - 8
point int t5
t5 = bp - 8
point int t6
t6 = bp - 12
var int t7
t7 = *t5 * *t6
*t4 = t7
point double t4
t4 = bp - 20
point double t5
t5 = bp - 20
point int t6
t6 = bp - 8
var double t7
t7 = 1 / *t6
```

	var double t6 t6 = *t5 + t7 *t4 = t6 *t0 = *t0 + t2 goto L2 label L3 point double t1 t1 = bp - 20 var double t3 t3 = *t1 println t3 sp = sp + 20	



```
[00400000] 8fa40000  lw $4, 0($29)       ; 18
[00400004] 27a50004  addiu $5, $29, 4    ; 18
```

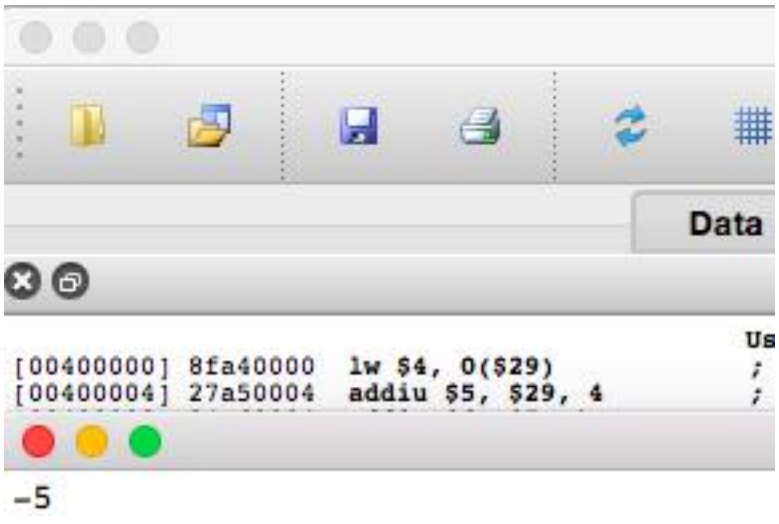
```
10
2.718281801146384 51
```

函数调用

Pascal 源码	三地址码	汇编代码

program test7;
var
    n,sum:integer;
function
nabs(x:integer):intege
r;
begin
    nabs:=0-x;
end;
begin
    n:=5;
	entry _nabs
	sp = sp - 4
	point int t0
	t0 = bp - 4
	point int t1
	t1 = bp + 8
	var int t2
	t2 = 0 - *t1
	*t0 = t2
	sp = sp + 4
	ret
		汇 编 代 码 可 在 Assemble/7.asm 中查看

sum:=nabs(n);
    writeln(sum);
end.
	entry main
	sp = sp - 8
	point int t0
	t0 = bp - 4
	*t0 = 5
	point int t0
	t0 = bp - 8
	point int t1
	t1 = bp - 4
	arg *t1
	call _nabs
	point int t1
	t1 = sp - 12
	var int t2
	t2 = *t1
	*t0 = t2
	point int t0
	t0 = bp - 8
	var int t1
	t1 = *t0
	println t1
	sp = sp + 8	



```
[00400000] 8fa40000  lw $4, 0($29)          ;
[00400004] 27a50004  addiu $5, $29, 4       ;
```

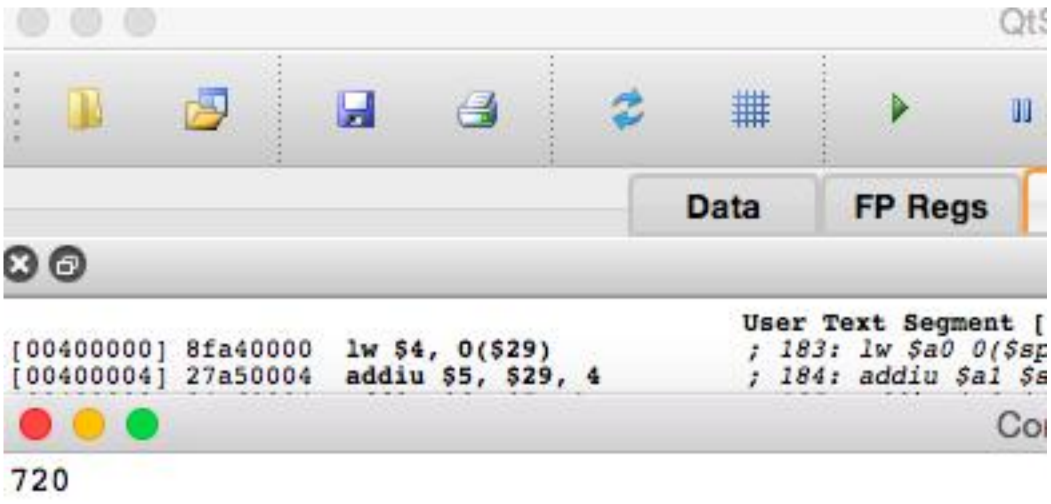
```
-5
```

递归调用

Pascal 源码	三地址码	汇编代码
program test8;	entry _fact	汇　编　代　码　可　在

var
    n:longint;
function fact(x:longint):longint;
begin
    if (x=0) then fact:=1 else fact := x * fact(x-1);
end;
begin
    n:=3;
    n:=fact(fact(n));
    writeln(n);
end.
	sp = sp - 4
	point int t0
	t0 = bp + 8
	var boolean t1
	t1 = *t0 == 0
	if_false t1 goto L0
	point int t0
	t0 = bp - 4
	*t0 = 1
	goto L1
	label L0
	point int t0
	t0 = bp - 4
	point int t1
	t1 = bp + 8
	point int t2
	t2 = bp + 8
	var int t3
	t3 = *t2 - 1
	arg t3
	call _fact
	point int t2
	t2 = sp - 12
	var int t3
	t3 = *t2
	var int t2
	t2 = *t1 * t3
	*t0 = t2
	label L1
	sp = sp + 4
	ret
	entry main
	sp = sp - 4
	point int t0
	t0 = bp - 4
	*t0 = 3
	point int t0
	t0 = bp - 4
	point int t1
	t1 = bp - 4
	arg *t1
	call _fact
	point int t1
	t1 = sp - 12
		Assemble/8.asm 中查看

	var int t2 t2 = *t1 arg t2 call _fact point int t1 t1 = sp - 12 var int t2 t2 = *t1 *t0 = t2 point int t0 t0 = bp - 4 var int t1 t1 = *t0 println t1 sp = sp + 4	



```
[00400000] 8fa40000  lw $4, 0($29)        User Text Segment [
[00400004] 27a50004  addiu $5, $29, 4     ; 183: lw $a0 0($sp
                                          ; 184: addiu $a1 $s
```

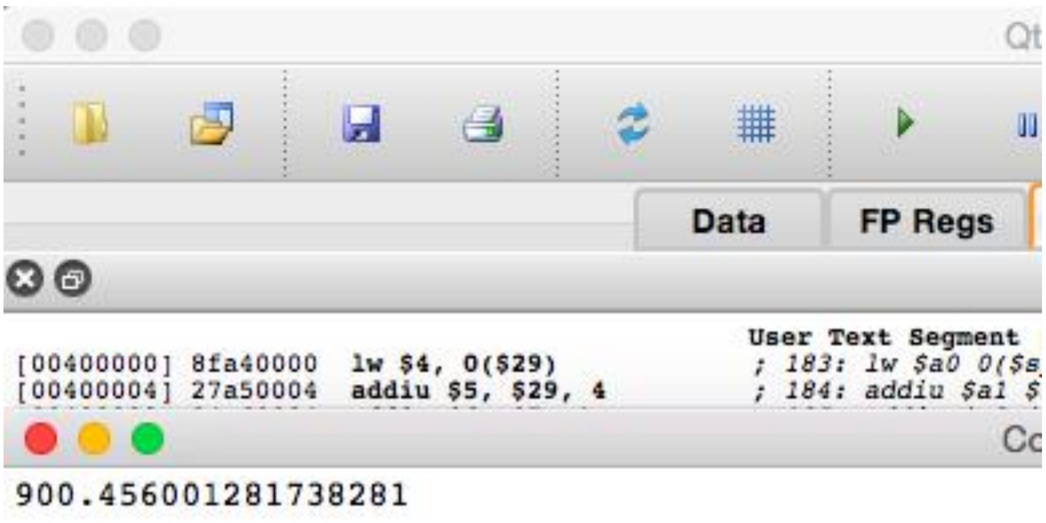
```
720
```

结构体

Pascal 源码	三地址码	汇编代码

program test9;
var
    a:record
        a:real;
        b:integer;
    end;
 	
    sum:real;
	entry main
	sp = sp - 20
	point record t0
	t0 = bp - 12
	point double t1
	t1 = t0 + 0
	*t1 = 123.456
	point record t0
	t0 = bp - 12
		汇编代码可在 Assemble/9.asm 中查看

begin
    a.a := 123.456;
    a.b := 777;
    sum := a.a + a.b;
    writeln(sum);
end.
	point int t1
	t1 = t0 + 8
	*t1 = 777
	point double t0
	t0 = bp - 20
	point record t1
	t1 = bp - 12
	point double t2
	t2 = t1 + 0
	point record t1
	t1 = bp - 12
	point int t3
	t3 = t1 + 8
	var double t1
	t1 = *t2 + *t3
	*t0 = t1
	point double t0
	t0 = bp - 20
	var double t1
	t1 = *t0
	println t1
	sp = sp + 20	



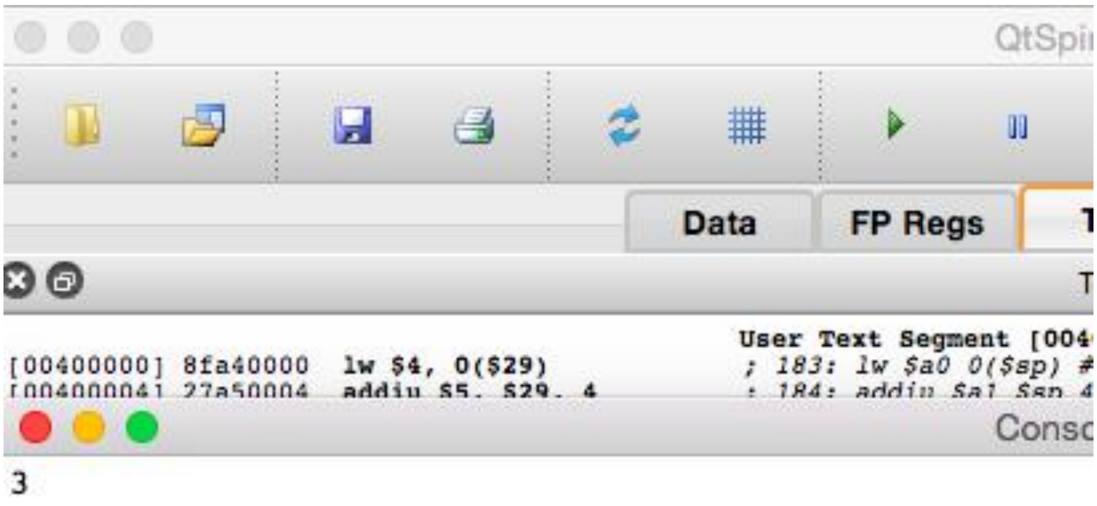
```
                                                User Text Segment
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($s
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $
```

```
900.456001281738281
```

常数计算优化

Pascal 源码	三地址码	汇编代码
program test10;	entry main	汇 编 代 码 可 在

var
    a:integer;
begin
    a:=1 + (2*3 div 4 +5 * 6) div 7-2;
    writeln(a);
end.
	sp = sp - 4
	point int t0
	t0 = bp - 4
	*t0 = 3
	point int t0
	t0 = bp - 4
	var int t1
	t1 = *t0
	println t1
	sp = sp + 4
		Assemble/10.asm 中查看



```
                                          User Text Segment [004
[00400000] 8fa40000  lw $4, 0($29)        ; 183: lw $a0 0($sp) #
[00400004] 27a50004  addiu $5, $29, 4     ; 184: addiu $a1 $sp 4
```

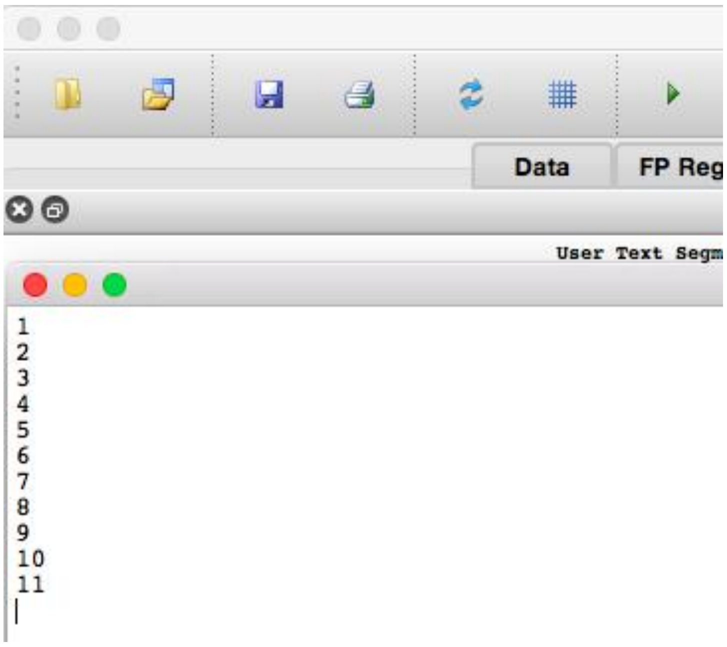
```
3
```

repeat

Pascal 源码	三地址码	汇编代码

program test11;
var
    n,i:integer;
begin
    n:=10;
    i:=0;
    repeat
        i:=i+1;
        writeln(i);
    until i>n;
end.
	entry main
	sp = sp - 8
	point int t0
	t0 = bp - 4
	*t0 = 10
	point int t0
	t0 = bp - 8
	*t0 = 0
	label L0
	point int t0
	t0 = bp - 8
	point int t1
	t1 = bp - 8
	var int t2
	t2 = *t1 + 1
		汇编代码可在 Assemble/11.asm 中查看

	*t0 = t2 point int t0 t0 = bp - 8 var int t1 t1 = *t0 println t1 point int t0 t0 = bp - 8 point int t1 t1 = bp - 4 var boolean t2 t2 = *t0 > *t1 if_false t2 goto L0 sp = sp + 8	



```
1
2
3
4
5
6
7
8
9
10
11
```

嵌套函数

Pascal 源码	三地址码	汇编代码

program test12;
var
    n,m:integer;
function
minsquare(x,y:integer)
	entry _minsquare
	sp = sp - 8
	point int t0
	t0 = bp - 8
	point int t1
		汇编代码可在 Assemble/12.asm 中查看

:integer;	t1 = bp + 12	
var	arg *t1	
temp:integer;	point int t1	
	t1 = bp + 8	
function	arg *t1	
min(x,y:integer):integ	call _min	
er;	point int t1	
begin	t1 = sp - 12	
if (x<y) then	var int t2	
min:=x else min:=y;	t2 = *t1	
end;	*t0 = t2	
	point int t0	
begin	t0 = bp - 4	
temp:=min(x,y);	point int t1	
	t1 = bp - 8	
minsquare:=temp*temp;	point int t2	
end;	t2 = bp - 8	
begin	var int t3	
n:=3;	t3 = *t1 * *t2	
m:=4;	*t0 = t3	
n:=minsquare(n,m);	sp = sp + 8	
writeln(n);	ret	
end.	entry _min	
	sp = sp - 4	
	point int t0	
	t0 = bp + 12	
	point int t1	
	t1 = bp + 8	
	var boolean t2	
	t2 = *t0 < *t1	
	if_false t2 goto L0	
	point int t0	
	t0 = bp - 4	
	point int t1	
	t1 = bp + 12	
	*t0 = *t1	
	goto L1	
	label L0	
	point int t0	
	t0 = bp - 4	
	point int t1	
	t1 = bp + 8	
	*t0 = *t1	
	label L1	

	sp = sp + 4	
	ret	
	entry main	
	sp = sp - 8	
	point int t0	
	t0 = bp - 4	
	*t0 = 3	
	point int t0	
	t0 = bp - 8	
	*t0 = 4	
	point int t0	
	t0 = bp - 4	
	point int t1	
	t1 = bp - 4	
	arg *t1	
	point int t1	
	t1 = bp - 8	
	arg *t1	
	call _minsquare	
	point int t1	
	t1 = sp - 12	
	var int t2	
	t2 = *t1	
	*t0 = t2	
	point int t0	
	t0 = bp - 4	
	var int t1	
	t1 = *t0	
	println t1	
	sp = sp + 8	