HIGH POINT UNIVERSITY
Department of Computer Science

CSC4510 Programming Language Design and Translation

lille code generation patterns

***lille* Source** — **PAL-code**

program Id **is** B;

```
B
JMP     0       0
```

D1; ... DN; **begin** S1; ... SN **end** Id

```
D1 D2 ... DN S1 S2 ... SN
```

Id1, ... IdN : <type>

```
INC     0       N
```

Id1, ... IdN : **constant integer**:= int;

```
INC     0       N
LCI     0       int
STO     0       A 1
□
LCI     0       int
STO     0       A N
```

Do the load and store for every identifier in ident_list, where Ai is the address on the current stack frame of identifier i.

Id1, ... IdN : **constant real** := rl;

```
INC     0       N
LCR     0       rl
STO     0       A 1
□
LCR     0       rl
STO     0       A N
```

Do the load and store for every identifier in ident_list, where Ai is the address on the current stack frame of identifier i.

Id1, ... IdN : **constant string** := str;

```
INC     0       N
LCS     0       str
STO     0       A 1
□
LCS     0       str
STO     0       A N
```

Do the load and store for every identifier in ident_list, where Ai is the address on the current stack frame of identifier i.

```
procedure Id( P1; ... PN ); B          JMP    0        L
                                 Lp :  B
                                       OPR    0        0
                                 L :
```

L is the instruction after the procedure return
Lp is the label for the start of the procedure execution.
When the procedure is executed the parameters are on the new stack frame and start at address 0. Any local variables declared in block B start at address after those for the parameters.

Id1, ... IdN : **value integer** — Treat as normal variables of the current stack frame. Similarly for reals & strings.

Id1, ... IdN : **ref integer** — The addresses of ref parameters are passed to procedures so assignment to a ref parameter within the procedure must be made through the STI instruction. Similarly for reals & strings.

```
P( Idval1, ... IdvalN1, Idref1, ... IdrefN2 )     MST    L1       0
                                                  LDV    Lval1    Dval1
                                                  □
                                                  LDV    LvalN    DvalN
                                                  LDA    Lref1    Dref1
                                                  □
                                                  LDA    LrefN    DrefN
                                                  CAL    N1+N2    Lp
```

L1 is the level difference between the call and the declaration of the procedure.
Load the value of all value parameters onto tos. Load the address of all ref parameters onto tos.
Where L, D is the address of each variable.
Call procedure: N1+N2 is no of parameters, Lp is the address of the instruction which starts the procedure (shown above).

```
Id := E                                E
                                       STO    L        D
```

Where L, D is the address of a local variable Id or a value parameter Id.

```
Id := E                                E
                                       LDV    L        D
                                       STI    0        0
```

Where L, D is the address of a ref parameter Id - remembering that the value of Id is the address of the variable corresponding to the ref parameter.

```
Id := S                                LCS    0        S
                                       STO    L        D
```

Where L, D is the address of a local variable Id or a value parameter Id.

```
Id := S                                LCS    0        S
                                       LDV    L        D
                                       STI    0        0
```

Where L, D is the address of a ref parameter Id - remembering that the value of Id is the address of the variable corresponding to the ref parameter.

```
loop ... exit when C ... end loop      L1 :  □
                                             C
                                             JIF   0      L2
                                             JMP   0      L3
                                       L2 :  □
                                             JMP   0      L1
                                       L3 :
```

L3 is the address of the instruction immediately after the end of the loop.

```
for I in lb .. ub                            JMP   0      L0     Jump to code following for loop
loop                                         INC   0      3      Space for loop variable, lb, ub,
    S                                        LCI   0      lb     Load lb
end loop;                                    OPR   0      23     Duplicate top of stack
                                             STO   0      0      Store lb in loop control variable
                                             STO   0      1      Hold lb in start of loop variable
                                             LCI   0      ub     Load ub
                                             STO   0      2      Hold ub in end of loop variable
                                       L1 :  LDV   0      0      Load the for loop control variable
                                             LDV   0      2      Load the end of loop variable.
                                             OPR   0      15     Check loop var <= end of loop
                                             JIF   0      L2     Jump to L2 if false
                                             S
                                             LDV   0      0      Load loop control variable
                                             LCI   0      1      Load value 1
                                             OPR   0      3      Add values
                                             STO   0      0      Store in for loop control variable
                                             JMP   0      L1     Jump to top of loop
                                       L2 :  OPR   0      0      Exit loop
                                       L0 :  MST   0      0      Mark stack for for-loop
                                             CAL   0      20     Invoke for loop, 0 parameters
```

```
read Id                                      RDI   L      D
                                       or    RDR   L      D
```

Read into.an integer or real variable depending on the variable's type.

```
write W : W'                                 W' load op
                                             W' conversion op
                                             W' write op
                                             OPR   0      21

W' : integer ->                              LCI   0      I      Load op
                                             OPR   0      20     Write op
W' : real ->                                 LCR   0      R      Load op
                                             OPR   0      20     Write op
W' : string ->                               LCS   0      S      Load op
                                             OPR   0      20     Write op
```

if C1 **then** SL1		C1		
elsif C2 **then** SL2		JIF	0	L1
... **elsif** CN-1 **then** SLN-1		SL1		
else SLN		JMP	0	Le
end if	L1 :	C2		
		JIF	0	L2
		SL2		
		JMP	0	Le
		□		
	LN-1 :	CN-1		
		JIF	0	LN
		SLN-1		
		JMP	0	Le
	LN :	SLN		
	Le :			
while C **loop** SL **end loop**	L1 :	C		
		JIF	0	L2
		SL		
		JMP	0	L1
	L2 :			
loop SL **end loop**	L :	SL		
		JMP	0	L
return E		E		
		OPR	0	1
odd(E)		E		
		OPR	0	9
E1=E2		E1		
		E2		
		OPR	0	10
E1/=E2		E1		
		E2		
		OPR	0	11
E1<E2		E1		
		E2		
		OPR	0	12
E1>=E2		E1		
		E2		
		OPR	0	13

E1>E2

```
      E1
      E2
      OPR   0     14
```

E1<=E2

```
      E1
      E2
      OPR   0     15
```

not C

```
      C
      OPR   0     16
```

eof

```
      OPR   0     19
```

E **in** lb .. ub

```
      E
      OPR   0     23
      lb
      OPR   0     12
      JIF   0     L1
      ub
      OPR   0     14
      JIF   0     L2
      OPR   0     17
      JMP   0     L3
L1:   OPR   0     24
L2:   OPR   0     18
L3:
```

This is the code generation sequence for a boolean expression. It is not part of a code generation sequence for a for loop.

- T

```
      T
      OPR   0     2
```

T1 + T2 - T3

```
      T1
      T2
      OPR   0     3
      T3
      OPR   0     4
```

F1 * F2 / F3

```
      F1
      F2
      OPR   0     5
      F3
      OPR   0     6
```

P1 ** P2

```
      P1
      P2
      OPR   0     7
```

Id	LDV	L	D
I	LCI	0	I
R	LCR	0	R
(E)	E		

Example Programs

PROGRAM 1

```
program prog1 is
    x, y, z : integer ;
begin
    write "Enter 2 numbers: " ;
    read x, y;
    write "Their sum is: " ;
    z := x + y ;
    writeln int2string(z) ;
end ;
```

```
JMP     0    14               (1)    Jump over the predefined functions.
LDV     0    0                (2)    Load argument.
OPR     0    25               (3)    Convert an integer to a real.
OPR     0    1                (4)    Function value return.
LDV     0    0                (5)    Load argument.
OPR     0    26               (6)    Convert a real to an integer.
OPR     0    1                (7)    Function value return.
LDV     0    0                (8)    Load argument.
OPR     0    27               (9)    Convert an integer to a string.
OPR     0    1                (10)   Function value return.
LDV     0    0                (11)   Load argument.
OPR     0    28               (12)   Convert an real to a string.
OPR     0    1                (13)   Function value return.
INC     0    3                (14)   Reserve space for local variables
JMP     0    17               (15)   Jump to start of statements or block.
INC     0    3                (16)   Reserve space for declared variables and constants.
LCS     0    'Enter 2 numbers: '          (17)     Load string value.
OPR     0    2                (18)   Write string value.
RDI     0    0                (19)   Read integer value.
RDI     0    1                (20)   Read integer value.
LCS     0    'Their sum is: '             (21)     Load string value.
OPR     0    20               (22)   Write string value.
LDV     0    0                (23)   Load variable or constant.
LDV     0    1                (24)   Load variable or constant.
OPR     0    3                (25)   Add arithmetic expressions together.
STO     0    2                (26)   Store result.
MST     1    0                (27)   Mark stack.
LDV     0    2                (28)   Load variable or constant.
CAL     1    8                (29)   Function call.
OPR     0    20               (30)   Write string value.
OPR     0    21               (31)   Terminate output to the current line.
JMP     0    0                (32)   Halt program
```

PROGRAM 2

```
program prog2 is
    number : integer;
    procedure double(n: value integer ) is
        n2: integer;
    begin
        n2 := n + n;
        write (int2string(n2));
    end;
begin
    write "Enter a number: ";
    read number;
    write "The number doubled is: ";
    double(number);
    writeln ".";
end;
```

```
JMP   0    14     (1)    Jump over the predefined functions.
LDV   0    0      (2)    Load argument.
OPR   0    25     (3)    Convert an integer to a real.
OPR   0    1      (4)    Function value return.
LDV   0    0      (5)    Load argument.
OPR   0    26     (6)    Convert an real to an integer.
OPR   0    1      (7)    Function value return.
LDV   0    0      (8)    Load argument.
OPR   0    27     (9)    Convert an integer to  a string.
OPR   0    1      (10)   Function value return.
LDV   0    0      (11)   Load argument.
OPR   0    28     (12)   Convert a real to a string.
OPR   0    1      (13)   Function value return.
INC   0    1      (14)   Reserve space for local variables
JMP   0    29     (15)   Jump to start of statements or block.
INC   0    1      (16)   Reserve space for declaredvariables and constants.
INC   0    2      (17)   Reserve space for local variables
JMP   0    20     (18)   Jump to start of statements or block.
INC   0    1      (19)   Reserve space fordeclared variables and constants.
LDV   0    0      (20)   Load value parameter.
LDV   0    0      (21)   Load value parameter.
OPR   0    3      (22)   Add arithmetic expressions together.
STO   0    1      (23)   Store result.
MST   2    0      (24)   Mark stack.
LDV   0    1      (25)   Load variable or constant.
CAL   1    8      (26)   Function call.
OPR   0    20     (27)   Write string value.
OPR   0    0      (28)   Procedure return.
LCS   0    'Enter a number: '          (29)   Load string value.
OPR   0    20     (30)   Write string value.
RDI   0    0      (31)   Read integer value.
LCS   0    'The number doubled is: '   (32)   Load string value.
OPR   0    20     (33)   Write string value.
MST   0    0      (34)   Mark stack.
LDV   0    0      (35)   Load actual value parameter
CAL   1    17     (36)   Call the procedure.
LCS   0    '.'    (37)   Load string value.
OPR   0    20     (38)   Write string value.
OPR   0    21     (39)   Terminate output to the current line.
JMP   0    0      (40)   Halt program.
```

PROGRAM 3

```
program prog3 is
    number, newNumber: integer;
    procedure double(n1: value integer; n2: ref integer) is
    begin
        n2 := n1 + n1;
    end;
begin
    write ("Enter a number: ");
    read (number);
    double(number,newNumber);
    writeln ("Number doubled is " & int2string(newNumber));
end;
```

```
JMP   0     14        (1)     Jump over the predefined functions.
LDV   0     0         (2)     Load argument.
OPR   0     25        (3)     Convert an integer to a real.
OPR   0     1         (4)     Function value return.
LDV   0     0         (5)     Load argument.
OPR   0     26        (6)     Convert a real to an integer.
OPR   0     1         (7)     Function value return.
LDV   0     0         (8)     Load argument.
OPR   0     27        (9)     Convert an integer to a string.
OPR   0     1         (10)    Function value return.
LDV   0     0         (11)    Load argument.
OPR   0     28        (12)    Convert a real to a string.
OPR   0     1         (13)    Function value return.
INC   0     2         (14)    Reserve space for local variables
JMP   0     25        (15)    Jump to start of statements or block.
INC   0     2         (16)    Reserve space for declared variables and constants.
INC   0     2         (17)    Reserve space for local variables
JMP   0     19        (18)    Jump to start of statements or block.
LDV   0     0         (19)    Load value parameter.
LDV   0     0         (20)    Load value parameter.
OPR   0     3         (21)    Add arithmetic expressions together.
LDV   0     1         (22)    Load address of variable to store the result.
STI   0     0         (23)    Store expression.
OPR   0     0         (24)    Procedure return.
LCS   0     'Enter a number: '        (25)    Load string value.
OPR   0     20         (26)   Write string value.
RDI   0     0         (27)    Read integer value.
MST   0     0         (28)    Mark stack.
LDV   0     0         (29)    Load actual value parameter
LDA   0     1         (30)    Load actual reference parameter
CAL   2     17        (31)    Call the procedure.
LCS   0     'Number doubled is '      (32)    Load string value.
MST   1     0         (33)    Mark stack.
LDV   0     1         (34)    Load variable or constant.
CAL   1     8         (35)    Function call.
OPR   0     8         (36)    Concatenate strings.
OPR   0     20        (37)    Write string value.
OPR   0     21        (38)    Terminate output to the current line.
JMP   0     0         (39)    Halt program.
```

PROGRAM 4

```
program prog4 is
    number, count, total, average : integer;
    procedure newNumber(n: value integer) is
    begin
        count := count + 1;
        total := total + n;
    end;
begin
    count := 0;
    total := 0;
    loop
        write "Enter a number, 0 to exit: ";
        read number ;
        exit when number = 0;
        newNumber(number);
    end loop;
    writeln "Count is " & int2string(count);
    writeln "Total is " & int2string(total);
    if count <> 0 then
        average := total / count;
        writeln "Average is " & int2string(average);
    end if;
end;
```

```
JMP   0     14      (1)     Jump over the predefined functions.
LDV   0     0       (2)     Load argument.
OPR   0     25      (3)     Convert an integer to a real.
OPR   0     1       (4)     Function value return.
LDV   0     0       (5)     Load argument.
OPR   0     26      (6)     Convert a real to an integer.
OPR   0     1       (7)     Function value return.
LDV   0     0       (8)     Load argument.
OPR   0     27      (9)     Convert an integer to a string.
OPR   0     1       (10)    Function value return.
LDV   0     0       (11)    Load argument.
OPR   0     28      (12)    Convert a real to a string.
OPR   0     1       (13)    Function value return.
INC   0     4       (14)    Reserve space for local variables
JMP   0     28      (15)    Jump to start of statements or block.
INC   0     4       (16)    Reserve space for declared variables and constants.
INC   0     1       (17)    Reserve space for local variables
JMP   0     19      (18)    Jump to start of statements or block.
LDV   1     1       (19)    Load variable or constant.
LCI   0     1       (20)    Load integer value.
OPR   0     3       (21)    Add arithmetic expressions together.
STO   1     1       (22)    Store result.
LDV   1     2       (23)    Load variable or constant.
LDV   0     0       (24)    Load value parameter.
OPR   0     3       (25)    Add arithmetic expressions together.
STO   1     2       (26)    Store result.
OPR   0     0       (27)    Procedure return.
LCI   0     0       (28)    Load integer value.
STO   0     1       (29)    Store result.
LCI   0     0       (30)    Load integer value.
STO   0     2       (31)    Store result.
LCS   0     'Enter a number, 0 to exit: '       (32)    Load string value.
OPR   0     20      (33)    Write string value.
RDI   0     0       (34)    Read integer value.
LDV   0     0       (35)    Load variable or constant.
LCI   0     0       (36)    Load integer value.
OPR   0     10      (37)    Compare expressions.
JIF   0     40      (38)    Do not exit loop.
JMP   0     44      (39)    Unconditional jump.
MST   0     0       (40)    Mark stack.
LDV   0     0       (41)    Load actual value parameter
CAL   1     17      (42)    Call the procedure.
JMP   0     32      (43)    Jump to start of loop.
LCS   0     'Count is '     (44)    Load string value.
MST   1     0       (45)    Mark stack.
LDV   0     1       (46)    Load variable or constant.
CAL   1     8       (47)    Function call.
OPR   0     8       (48)    Concatenate strings.
OPR   0     20      (49)    Write string value.
OPR   0     21      (50)    Terminate output to the current line.
LCS   0     'Total is '     (51)    Load string value.
MST   1     0       (52)    Mark stack.
LDV   0     2       (53)    Load variable or constant.
CAL   1     8       (54)    Function call.
OPR   0     8       (55)    Concatenate strings.
OPR   0     20      (56)    Write string value.
OPR   0   21              (57)   Terminate output to the current line.
LDV   0   1               (58)   Load variable or constant.
LCI   0   0               (59)   Load integer value.
OPR   0   11              (60)   Compare expressions.
JIF   0   74              (61)   Jump if false.
LDV   0   2               (62)   Load variable or constant.
LDV   0   1               (63)   Load variable or constant.
OPR   0   6               (64)   Divide arithmetic expression at tos-1 by expression at tos.
STO   0   3               (65)   Store result.
LCS   0   'Average is '   (66)   Load string value.
MST   1   0               (67)   Mark stack.
LDV   0   3               (68)   Load variable or constant.
CAL   1   8               (69)   Function call.
OPR   0   8               (70)   Concatenate strings.
OPR   0   20              (71)   Write string value.
OPR   0   21              (72)   Terminate output to the current line.
JMP   0   74              (73)   Unconditional jump.
JMP   0   0               (74)   Halt program.
```

PROGRAM 5

```
program prog5 is
    number, dNumber, qNumber: integer;

    procedure double(n1: value integer; n2: ref integer) is
    begin
        n2 := n1 + n1;
    end double;

    procedure quadruple(n1: value integer; n2: ref integer) is
        n: integer;
    begin
        double(n1, n);
        double(n, n);
        n2 := n;
    end quadruple;

begin
    write ("Enter a number: ");
    read (number);
    double(number, dNumber);
    quadruple(number, qNumber);
    writeln; -- Leave an empty line
    writeln (int2string(number), "  ", int2string(dnumber), "  ", int2string(qNumber));
end prog5;
```

```
JMP   0   14                   (1)    Jump over the predefined functions.
LDV   0   0                    (2)    Load argument.
OPR   0   25                   (3)    Convert an integer to a real.
OPR   0   1                    (4)    Function value return.
LDV   0   0                    (5)    Load argument.
OPR   0   26                   (6)    Convert a real to an integer.
OPR   0   1                    (7)    Function value return.
LDV   0   0                    (8)    Load argument.
OPR   0   27                   (9)    Convert an integer to a string.
OPR   0   1                    (10)   Function value return.
LDV   0   0                    (11)   Load argument.
OPR   0   28                   (12)   Convert a real to a string.
OPR   0   1                    (13)   Function value return.
INC   0   3                    (14)   Reserve space for local variables
JMP   0   40                   (15)   Jump to start of statements or block.
INC   0   3                    (16)   Reserve space for declared variables and constants.
INC   0   2                    (17)   Reserve space for local variables
JMP   0   19                   (18)   Jump to start of statements or block.
LDV   0   0                    (19)   Load value parameter.
LDV   0   0                    (20)   Load value parameter.
OPR   0   3                    (21)   Add arithmetic expressions together.
LDV   0   1                    (22)   Load address of variable to store the result.
STI   0   0                    (23)   Store expression.
OPR   0   0                    (24)   Procedure return.
INC   0   3                    (25)   Reserve space for local variables
JMP   0   28                   (26)   Jump to start of statements or block.
INC   0   1                    (27)   Reserve space for declared variables and constants.
MST   1   0                    (28)   Mark stack.
LDV   0   0                    (29)   Load actual value parameter
LDA   0   2                    (30)   Load actual reference parameter
CAL   2   17                   (31)   Call the procedure.
MST   1   0                    (32)   Mark stack.
LDV   0   2                    (33)   Load actual value parameter
LDA   0   2                    (34)   Load actual reference parameter
CAL   2   17                   (35)   Call the procedure.
LDV   0   2                    (36)   Load variable or constant.
LDV   0   1                    (37)   Load address of variable to store the result.
STI   0   0                    (38)   Store expression.
OPR   0   0                    (39)   Procedure return.
LCS   0   'Enter a number: ' (40)     Load string value.
OPR   0   20                   (41)   Write string value.
RDI   0   0                    (42)   Read integer value.
MST   0   0                    (43)   Mark stack.
LDV   0   0                    (44)   Load actual value parameter
LDA   0   1                    (45)   Load actual reference parameter
CAL   2   17                   (46)   Call the procedure.
MST   0   0                    (47)   Mark stack.
LDV   0   0                    (48)   Load actual value parameter
LDA   0   2                    (49)   Load actual reference parameter
CAL   2   25                   (50)   Call the procedure.
OPR   0   21                   (51)   Terminate output to the current line.
MST   1   0                    (52)   Mark stack.
LDV   0   0                    (53)   Load variable or constant.
CAL   1   8                    (54)   Function call.
OPR   0   20                   (55)   Write string value.
LCS   0   ' '                  (56)   Load string value.
```

```
OPR   0   20    (57)   Write string value.
MST   1   0     (58)   Mark stack.
LDV   0   1     (59)   Load variable or constant.
CAL   1   8     (60)   Function call.
OPR   0   20    (61)   Write string value.
LCS   0   ' '   (62)   Load string value.
OPR   0   20    (63)   Write string value.
MST   1   0     (64)   Mark stack.
LDV   0   2     (65)   Load variable or constant.
CAL   1   8     (66)   Function call.
OPR   0   20    (67)   Write string value.
OPR   0   21    (68)   Terminate output to the current line.
JMP   0   0     (69)   Halt program.
```

PROGRAM 6

```
program prog6 is
begin
    writeln "In ascending order: ";
    for i in 1..5
    loop
        write int2string(i) & " ";
    end loop;
    writeln;
    writeln "In descending order: ";
    for i in reverse 1..5
    loop
        write int2string(i) & " ";
    end loop;
    writeln;
end prog6;
```

```
JMP   0   14     (1)    Jump over the predefined functions.
LDV   0   0      (2)    Load argument.
OPR   0   25     (3)    Convert an integer to a real.
OPR   0   1      (4)    Function value return.
LDV   0   0      (5)    Load argument.
OPR   0   26     (6)    Convert an real to an integer.
OPR   0   1      (7)    Function value return.
LDV   0   0      (8)    Load argument.
OPR   0   27     (9)    Convert an integer to a string.
OPR   0   1      (10)   Function value return.
LDV   0   0      (11)   Load argument.
OPR   0   28     (12)   Convert an real to a string.
OPR   0   1      (13)   Function value return.
INC   0   0      (14)   Reserve space for local variables
JMP   0   16     (15)   Jump to start of statements or block.
LCS   0   'In ascending order: '     (16)   Load string value.
OPR   0   20     (17)   Write string value.
OPR   0   21     (18)   Terminate output to the current line.
JMP   0   43     (19)   Unconditional jump.
INC   0   3      (20)   Reserve space for for-loop control variable, lb and ub values.
LCI   0   1      (21)   Load integer value.
OPR   0   23     (22)   Duplicate top of stack
STO   0   0      (23)   Store the lower bound of the range as initial value of for loop parameter.
STO   0   1      (24)   Store the lower bound of the range as the start value of the loop.
LCI   0   5      (25)   Load integer value.
STO   0   2      (26)   Store the upper bound of the range as the end value of the loop.
LDV   0   0      (27)   Load the value of the for loop parameter.
LDV   0   2      (28)   Load the end value of the for loop.
OPR   0   15     (29)   Check if loop parameter <= end value.
JIF   0   42     (30)   Jump if false.
MST   2   0      (31)   Mark stack.
LDV   0   0      (32)   Load For loop parameter.
CAL   1   8      (33)   Function call.
LCS   0   ' '    (34)   Load string value.
OPR   0   8      (35)   Concatenate strings.
OPR   0   20     (36)   Write string value.
LDV   0   0      (37)   Load the value of the for loop parameter.
LCI   0   1      (38)   Load the value 1 onto the stack.
OPR   0   3      (39)   Add values.
STO   0   0      (40)   Store value of for loop parameter.
JMP   0   27     (41)   Jump to beginning of for loop for next iteration.
OPR   0   0      (42)   Return from for loop.
MST   0   0      (43)   Mark stack for for loop.
CAL   0   20     (44)   Effectively call the for loop - 0 parameters.
OPR   0   21     (45)   Terminate output to the current line.
LCS   0   'In descending order: '    (46)   Load string value.
OPR   0   20     (47)   Write string value.
OPR   0   21     (48)   Terminate output to the current line.
JMP   0   73     (49)   Unconditional jump.
INC   0   3      (50)   Reserve space for for-loop control variable, lb and ub values.
LCI   0   1      (51)   Load integer value.
STO   0   2      (52)   Store the lower bound of the range as the end value of the loop.
LCI   0   5      (53)   Load integer value.
OPR   0   23     (54)   Duplicate top of stack
STO   0   0      (55)   Store the upper bound of the range as initial value of for loop parameter.
STO   0   1      (56)   Store the upper bound of the range as the start value of the loop.
```

LDV	0	0	(57)	Load the value of the for loop parameter.
LDV	0	2	(58)	Load the end value of the for loop.
OPR	0	13	(59)	Check if loop parameter >= end value.
JIF	0	72	(60)	Jump if false.
MST	2	0	(61)	Mark stack.
LDV	0	0	(62)	Load For loop parameter.
CAL	1	8	(63)	Function call.
LCS	0	' '	(64)	Load string value.
OPR	0	8	(65)	Concatenate strings.
OPR	0	20	(66)	Write string value.
LDV	0	0	(67)	Load the value of the for loop parameter.
LCI	0	1	(68)	Load the value 1 onto the stack.
OPR	0	4	(69)	Subtract values.
STO	0	0	(70)	Store value of for loop parameter.
JMP	0	57	(71)	Jump to beginning of for loop for next iteration.
OPR	0	0	(72)	Return from for loop.
MST	0	0	(73)	Mark stack for for loop.
CAL	0	50	(74)	Effectively call the for loop - 0 parameters.
OPR	0	21	(75)	Terminate output to the current line.
JMP	0	0	(76)	Halt program.

PROGRAM 7

```
program prog7 is
    i: integer;

begin
    i := 15;
    writeln "Before the first for loop, i = " & int2string(i);
    writeln "In ascending order: ";
    for i in 1..5
    loop
        write int2string(i) & " ";
    end loop;
    writeln;
    writeln "Between the for loops, i = " & int2string(i);
    writeln "In descending order: ";
    for i in reverse 1..5
    loop
        write int2string(i) & " ";
    end loop;
    writeln;
    writeln "After the second for loops, i = " & int2string(i);
end prog7;
```

```
JMP   0   14        (1)    Jump over the predefined functions.
LDV   0   0         (2)    Load argument.
OPR   0   25        (3)    Convert an integer to a real.
OPR   0   1         (4)    Function value return.
LDV   0   0         (5)    Load argument.
OPR   0   26        (6)    Convert a real to an integer.
OPR   0   1         (7)    Function value return.
LDV   0   0         (8)    Load argument.
OPR   0   27        (9)    Convert an integer to a string.
OPR   0   1         (10)   Function value return.
LDV   0   0         (11)   Load argument.
OPR   0   28        (12)   Convert a real to a string.
OPR   0   1         (13)   Function value return.
INC   0   1         (14)   Reserve space for local variables
JMP   0   17        (15)   Jump to start of statements or block.
INC   0   1         (16)   Reserve space for declared variables and constants.
LCI   0   15        (17)   Load integer value.
STO   0   0         (18)   Store result.
LCS   0   'Before the first for loop, i = '      (19)   Load string value.
MST   1   0         (20)   Mark stack.
LDV   0   0         (21)   Load variable or constant.
CAL   1   8         (22)   Function call.
OPR   0   8         (23)   Concatenate strings.
OPR   0   20        (24)   Write string value.
OPR   0   21        (25)   Terminate output to the current line.
LCS   0   'In ascending order: '      (26)   Load string value.
OPR   0   20        (27)   Write string value.
OPR   0   21        (28)   Terminate output to the current line.
JMP   0   53        (29)   Unconditional jump.
INC   0   3         (30)   Reserve space for for-loop control variable, lb and ub values.
LCI   0   1         (31)   Load integer value.
OPR   0   23        (32)   Duplicate top of stack
STO   0   0         (33)   Store the lower bound of the range as initial value of for loop parameter.
STO   0   1         (34)   Store the lower bound of the range as the start value of the loop.
LCI   0   5         (35)   Load integer value.
STO   0   2         (36)   Store the upper bound of the range as the end value of the loop.
LDV   0   0         (37)   Load the value of the for loop parameter.
LDV   0   2         (38)   Load the end value of the for loop.
OPR   0   15        (39)   Check if loop parameter <= end value.
JIF   0   52        (40)   Jump if false.
MST   2   0         (41)   Mark stack.
LDV   0   0         (42)   Load For loop parameter.
CAL   1   8         (43)   Function call.
LCS   0   ' '       (44)   Load string value.
OPR   0   8         (45)   Concatenate strings.
OPR   0   20        (46)   Write string value.
LDV   0   0         (47)   Load the value of the for loop parameter.
LCI   0   1         (48)   Load the value 1 onto the stack.
OPR   0   3         (49)   Add values.
STO   0   0         (50)   Store value of for loop parameter.
JMP   0   37        (51)   Jump to beginning of for loop for next iteration.
OPR   0   0         (52)   Return from for loop.
MST   0   0         (53)   Mark stack for for loop.
CAL   0   30        (54)   Effectively call the for loop - 0 parameters.
OPR   0   21        (55)   Terminate output to the current line.
LCS   0   'Between the for loops, i = '      (56)   Load string value.
```

```
MST   1    0        (57)    Mark stack.
LDV   0    0        (58)    Load variable or constant.
CAL   1    8        (59)    Function call.
OPR   0    8        (60)    Concatenate strings.
OPR   0    20       (61)    Write string value.
OPR   0    21       (62)    Terminate output to the current line.
LCS   0    'In descending order: '    (63)    Load string value.
OPR   0    20       (64)    Write string value.
OPR   0    21       (65)    Terminate output to the current line.
JMP   0    90       (66)    Unconditional jump.
INC   0    3        (67)    Reserve space for for-loop control variable, lb and ub values.
LCI   0    1        (68)    Load integer value.
STO   0    2        (69)    Store the lower bound of the range as the end value of the loop.
LCI   0    5        (70)    Load integer value.
OPR   0    23       (71)    Duplicate top of stack
STO   0    0        (72)    Store the upper bound of the range as initial value of for loop parameter.
STO   0    1        (73)    Store the upper bound of the range as the start value of the loop.
LDV   0    0        (74)    Load the value of the for loop parameter.
LDV   0    2        (75)    Load the end value of the for loop.
OPR   0    13       (76)    Check if loop parameter >= end value.
JIF   0    89       (77)    Jump if false.
MST   2    0        (78)    Mark stack.
LDV   0    0        (79)    Load For loop parameter.
CAL   1    8        (80)    Function call.
LCS   0    ' '      (81)    Load string value.
OPR   0    8        (82)    Concatenate strings.
OPR   0    20       (83)    Write string value.
LDV   0    0        (84)    Load the value of the for loop parameter.
LCI   0    1        (85)    Load the value 1 onto the stack.
OPR   0    4        (86)    Subtract values.
STO   0    0        (87)    Store value of for loop parameter.
JMP   0    74       (88)    Jump to beginning of for loop for next iteration.
OPR   0    0        (89)    Return from for loop.
MST   0    0        (90)    Mark stack for for loop.
CAL   0    67       (91)    Effectively call the for loop - 0 parameters.
OPR   0    21       (92)    Terminate output to the current line.
LCS   0    'After the second for loops, i = '    (93)    Load string value.
MST   1    0        (94)    Mark stack.
LDV   0    0        (95)    Load variable or constant.
CAL   1    8        (96)    Function call.
OPR   0    8        (97)    Concatenate strings.
OPR   0    20       (98)    Write string value.
OPR   0    21       (99)    Terminate output to the current line.
JMP   0    0        (100)   Halt program.
```

PROGRAM 8

```
program prog8 is
    i, j: integer;

begin
    i := 0;
    while i<5
    loop
        j := 0;
        while j<5
        loop
            write "(", int2string(i) & ", " & int2string(j) & ")   ";
            exit when j >= 3;
            j := j + 1;
        end loop;
        writeln;
        exit when i >= 3;
        i := i + 1;
    end loop;
    writeln;
end prog8;
```

```
JMP   0   14      (1)    Jump over the predefined functions.
LDV   0   0       (2)    Load argument.
OPR   0   25      (3)    Convert an integer to a real.
OPR   0   1       (4)    Function value return.
LDV   0   0       (5)    Load argument.
OPR   0   26      (6)    Convert an real to an integer.
OPR   0   1       (7)    Function value return.
LDV   0   0       (8)    Load argument.
OPR   0   27      (9)    Convert an integer to a string.
OPR   0   1       (10)   Function value return.
LDV   0   0       (11)   Load argument.
OPR   0   28      (12)   Convert an real to a string.
OPR   0   1       (13)   Function value return.
INC   0   2       (14)   Reserve space for local variables
JMP   0   17      (15)   Jump to start of statements or block.
INC   0   2       (16)   Reserve space for declared variables and constants.
LCI   0   0       (17)   Load integer value.
STO   0   0       (18)   Store result.
LDV   0   0       (19)   Load variable or constant.
LCI   0   5       (20)   Load integer value.
OPR   0   12      (21)   Compare expressions.
JIF   0   64      (22)   Jump if false.
LCI   0   0       (23)   Load integer value.
STO   0   1       (24)   Store result.
LDV   0   1       (25)   Load variable or constant.
LCI   0   5       (26)   Load integer value.
OPR   0   12      (27)   Compare expressions.
JIF   0   53      (28)   Jump if false.
LCS   0   '(‘.    (29)   Load string value.
OPR   0   20      (30)   Write string value.
MST   1   0       (31)   Mark stack.
LDV   0   0       (32)   Load variable or constant.
CAL   1   8       (33)   Function call.
LCS   0   ', ‘.   (34)   Load string value.
OPR   0   8       (35)   Concatenate strings.
MST   1   0       (36)   Mark stack.
LDV   0   1       (37)   Load variable or constant.
CAL   1   8       (38)   Function call.
OPR   0   8       (39)   Concatenate strings.
LCS   0   ').  ‘. (40)   Load string value.
OPR   0   8       (41)   Concatenate strings.
OPR   0   20      (42)   Write string value.
LDV   0   1       (43)   Load variable or constant.
LCI   0   3       (44)   Load integer value.
OPR   0   13      (45)   Compare expressions.
JIF   0   48      (46)   Do not exit loop.
JMP   0   53      (47)   Unconditional jump.
LDV   0   1       (48)   Load variable or constant.
LCI   0   1       (49)   Load integer value.
OPR   0   3       (50)   Add arithmetic expressions together.
STO   0   1       (51)   Store result.
JMP   0   25      (52)   Jump to start of loop.
OPR   0   21      (53)   Terminate output to the current line.
LDV   0   0       (54)   Load variable or constant.
LCI   0   3       (55)   Load integer value.
OPR   0   13      (56)   Compare expressions.
```

JIF	0	59	(57)	Do not exit loop.
JMP	0	64	(58)	Unconditional jump.
LDV	0	0	(59)	Load variable or constant.
LCI	0	1	(60)	Load integer value.
OPR	0	3	(61)	Add arithmetic expressions together.
STO	0	0	(62)	Store result.
JMP	0	19	(63)	Jump to start of loop.
OPR	0	21	(64)	Terminate output to the current line.
JMP	0	0	(65)	Halt program.

PROGRAM 9

```
program prog9 is

begin
    for i in 1..5
    loop
        exit when i > 3;
        for j in reverse 1..5
        loop
            write "(", i, ", ", j, ")   ";
            exit when j < 3;
        end loop;
        writeln;
    end loop;
    writeln;
end prog9;
```

```
JMP   0    14     (1)    Jump over the predefined functions.
LDV   0    0      (2)    Load argument.
OPR   0    25     (3)    Convert an integer to a real.
OPR   0    1      (4)    Function value return.
LDV   0    0      (5)    Load argument.
OPR   0    26     (6)    Convert an real to an integer.
OPR   0    1      (7)    Function value return.
LDV   0    0      (8)    Load argument.
OPR   0    27     (9)    Convert an integer to a string.
OPR   0    1      (10)   Function value return.
LDV   0    0      (11)   Load argument.
OPR   0    28     (12)   Convert an real to a string.
OPR   0    1      (13)   Function value return.
INC   0    0      (14)   Reserve space for local variables
JMP   0    16     (15)   Jump to start of statements or block.
JMP   0    75     (16)   Unconditional jump.
INC   0    3      (17)   Reserve space for for-loop control variable, lb and ub values.
LCI   0    1      (18)   Load integer value.
OPR   0    23     (19)   Duplicate top of stack
STO   0    0      (20)   Store the lower bound of the range as initial value of for loop parameter.
STO   0    1      (21)   Store the lower bound of the range as the start value of the loop.
LCI   0    5      (22)   Load integer value.
STO   0    2      (23)   Store the upper bound of the range as the end value of the loop.
LDV   0    0      (24)   Load the value of the for loop parameter.
LDV   0    2      (25)   Load the end value of the for loop.
OPR   0    15     (26)   Check if loop parameter <= end value.
JIF   0    74     (27)   Jump if false.
LDV   0    0      (28)   Load For loop parameter.
LCI   0    3      (29)   Load integer value.
OPR   0    14     (30)   Compare expressions.
JIF   0    33     (31)   Do not exit loop.
JMP   0    74     (32)   Unconditional jump.
JMP   0    66     (33)   Unconditional jump.
INC   0    3      (34)   Reserve space for for-loop control variable, lb and ub values.
LCI   0    1      (35)   Load integer value.
STO   0    2      (36)   Store the lower bound of the range as the end value of the loop.
LCI   0    5      (37)   Load integer value.
OPR   0    23     (38)   Duplicate top of stack
STO   0    0      (39)   Store the upper bound of the range as initial value of for loop parameter.
STO   0    1      (40)   Store the upper bound of the range as the start value of the loop.
LDV   0    0      (41)   Load the value of the for loop parameter.
LDV   0    2      (42)   Load the end value of the for loop.
OPR   0    13     (43)   Check if loop parameter >= end value.
JIF   0    65     (44)   Jump if false.
LCS   0    '('    (45)   Load string value.
OPR   0    20     (46)   Write string value.
LDV   1    0      (47)   Load For loop parameter.
OPR   0    20     (48)   Write integer or real value.
LCS   0    ', '   (49)   Load string value.
OPR   0    20     (50)   Write string value.
LDV   0    0      (51)   Load For loop parameter.
OPR   0    20     (52)   Write integer or real value.
LCS   0    ').  ' (53)   Load string value.
OPR   0    20     (54)   Write string value.
LDV   0    0      (55)   Load For loop parameter.
LCI   0    3      (56)   Load integer value.
```

```
OPR   0    12    (57)    Compare expressions.
JIF   0    60    (58)    Do not exit loop.
JMP   0    65    (59)    Unconditional jump.
LDV   0    0     (60)    Load the value of the for loop parameter.
LCI   0    1     (61)    Load the value 1 onto the stack.
OPR   0    4     (62)    Subtract values.
STO   0    0     (63)    Store value of for loop parameter.
JMP   0    41    (64)    Jump to beginning of for loop for next iteration.
OPR   0    0     (65)    Return from for loop.
MST   0    0     (66)    Mark stack for for loop.
CAL   0    34    (67)    Effectively call the for loop - 0 parameters.
OPR   0    21    (68)    Terminate output to the current line.
LDV   0    0     (69)    Load the value of the for loop parameter.
LCI   0    1     (70)    Load the value 1 onto the stack.
OPR   0    3     (71)    Add values.
STO   0    0     (72)    Store value of for loop parameter.
JMP   0    24    (73)    Jump to beginning of for loop for next iteration.
OPR   0    0     (74)    Return from for loop.
MST   0    0     (75)    Mark stack for for loop.
CAL   0    17    (76)    Effectively call the for loop - 0 parameters.
OPR   0    21    (77)    Terminate output to the current line.
JMP   0    0     (78)    Halt program.
```

Michael Oudshoorn
August 13, 2023