

Code Generation Assignment on Ven

Name: kareem gamal Mahmoud Mohamed

Computer science level 4

Advanced compiler

The assignment contains :

- Scanner

- Parser

- EBNF syntax

- Semantic Analyzer

- Code Generation

- Test File

- Abstract Syntax

EBNF syntax for ven language :

$\langle \text{parse_ven} \rangle ::= \langle \text{program} \rangle \text{ EOF}$

$\langle \text{program} \rangle ::= \langle \text{block} \rangle$

$\langle \text{block} \rangle ::= \text{begin } \langle \text{declaration-seq} \rangle \langle \text{command-seq} \rangle \text{ end}$

$\langle \text{declaration-seq} \rangle ::= \langle \text{declaration} \rangle \{ \langle \text{declaration} \rangle \}$

$\langle \text{declaration} \rangle ::= \langle \text{type} \rangle \langle \text{name-list} \rangle \mid \text{proc } \langle \text{name} \rangle [(\langle \text{parameter-list} \rangle)] = \langle \text{command} \rangle$

$\langle \text{type} \rangle ::= \text{integer} \mid \text{Boolean}$

$\langle \text{parameter-list} \rangle ::= \langle \text{type} \rangle \langle \text{name-list} \rangle \{ ; \langle \text{type} \rangle \langle \text{name-list} \rangle \}$

$\langle \text{name-list} \rangle ::= \langle \text{name} \rangle \{ , \langle \text{name} \rangle \}$

$\langle \text{command-seq} \rangle ::= \langle \text{command} \rangle \{ ; \langle \text{command} \rangle \} \langle \text{command} \rangle ::= \langle \text{name} \rangle :=$

$\langle \text{expr} \rangle \mid \text{read } \langle \text{name} \rangle \mid \text{write } \langle \text{expr} \rangle \mid \text{if } \langle \text{expr} \rangle \text{ then}$

$\langle \text{command-seq} \rangle [\text{else } \langle \text{command-seq} \rangle] \text{ end if} \mid$

$\text{while } \langle \text{expr} \rangle \text{ do } \langle \text{command-seq} \rangle \text{ end while} \mid \text{call } \langle \text{name} \rangle [(\langle \text{name-list} \rangle)]$

$\langle \text{expr} \rangle ::= \langle \text{expr1} \rangle \{ \text{or } \langle \text{expr1} \rangle \}$

$\langle \text{expr1} \rangle ::= \langle \text{expr2} \rangle \{ \text{and } \langle \text{expr2} \rangle \}$

$\langle \text{expr2} \rangle ::= \langle \text{expr3} \rangle \mid \text{not } \langle \text{expr} \rangle$

$\langle \text{expr3} \rangle ::= \langle \text{expr4} \rangle \{ \langle \text{relation} \rangle \langle \text{expr4} \rangle \}$

$\langle \text{expr4} \rangle ::= \langle \text{term} \rangle \{ \langle \text{weak op} \rangle \text{term} \}$

$\langle \text{term} \rangle ::= \langle \text{element} \rangle \{ \langle \text{strong op} \rangle \langle \text{element} \rangle \}$

$\langle \text{element} \rangle ::= [-] \langle \text{elem} \rangle$

$\langle \text{elem} \rangle ::= \langle \text{numeral} \rangle \mid \langle \text{name} \rangle \mid (\langle \text{expr} \rangle)$

$\langle \text{relation} \rangle ::= < \mid <= \mid <> \mid = \mid > \mid >=$

$\langle \text{weak op} \rangle ::= + \mid -$

$\langle \text{strong op} \rangle ::= * \mid /$

$\langle \text{name} \rangle ::= [\langle \text{ident} \rangle] \langle \text{letter} \rangle \mid \langle \text{ident} \rangle \langle \text{digit} \rangle$

$\langle \text{letter} \rangle ::= a \mid b \mid c \mid d \mid e \mid f \mid g \mid h \mid i \mid j \mid k \mid l \mid m \mid n \mid o \mid p \mid q \mid r \mid s \mid t \mid u \mid v \mid w \mid x \mid y \mid z$

$\langle \text{numeral} \rangle ::= \langle \text{digit} \rangle \{ \langle \text{digit} \rangle \}$

$\langle \text{digit} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Abstract Syntax for ven:

Program \leftarrow Program(Block)

Block \leftarrow Block(DecSeq , ComSeq)

DecSeq \leftarrow Declaration+ SB e

Declaration \leftarrow var(type , Ids) | proc(Id , Paras? , command)

type \leftarrow integer | Boolean

Paras \leftarrow Paras (type , Ids)+ SB ;

Ids \leftarrow Id+ SB ,

ComSeq \leftarrow command+ SB ;

Command \leftarrow Assign(Id , Expr) | Read(Id) | Write(Id) |

IfThenElse (Expr , ComSeq , ComSeq?)

| While(Expr , ComSeq) | Call(Id , Ids?)

$\text{Expr} \leftarrow \text{Expr1} + \text{SB or}$

$\text{Expr1} \leftarrow \text{Expr2} + \text{SB and}$

$\text{Expr2} \leftarrow \text{Expr3} \mid \text{Not}(\text{Expr})$

$\text{Expr3} \leftarrow \text{Expr4} + \text{SB relation}$

$\text{Expr4} \leftarrow \text{Term} + \text{SB w_op}$

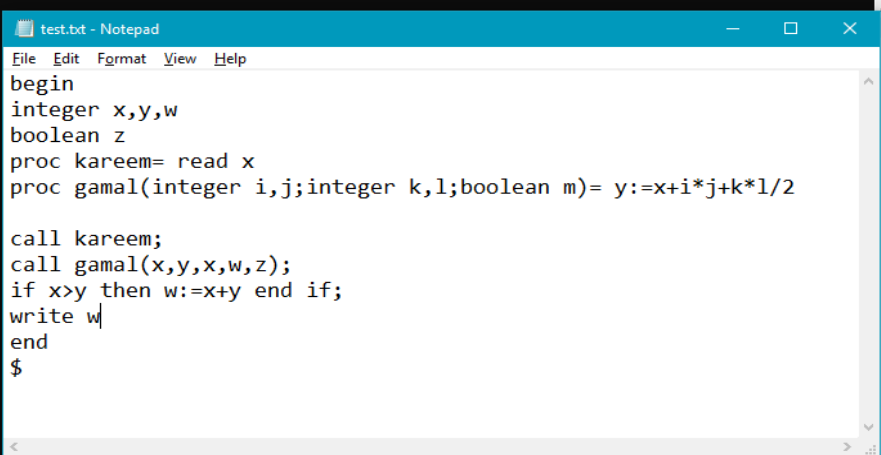
$\text{Element} \leftarrow \text{Number} \mid \text{Id} \mid \text{Expr} \mid \text{-(Element)}$

$\langle \text{relation} \rangle ::= < \mid <= \mid <> \mid = \mid > \mid >=$

$\langle \text{weak op} \rangle ::= + \mid -$

$\langle \text{strong op} \rangle ::= * \mid /$

Run the test file :

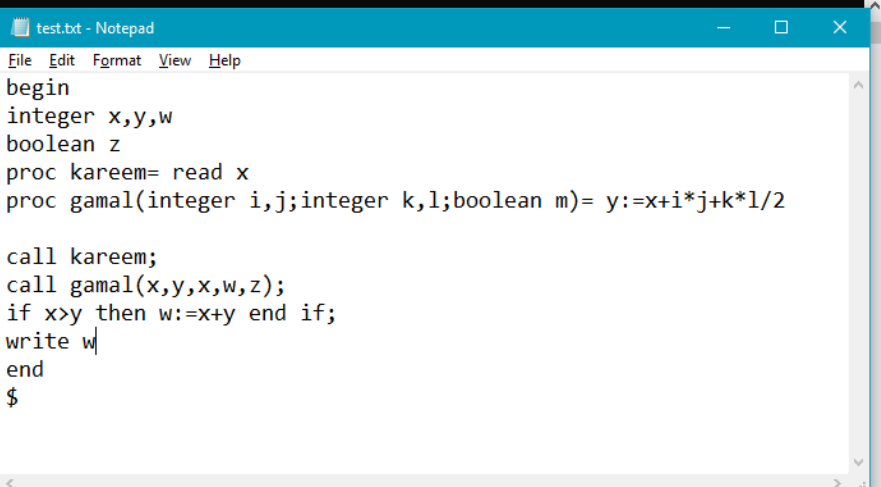


```
begin matched ttype
integer matched ttype
x (Identifier) matched ttype
, matched ttype
y (Identifier) matched ttype
, matched ttype
w (Identifier) matched ttype
boolean matched ttype
z (Identifier) matched ttype
proc matched ttype
kareem (Identifier) matched ttype
= matched ttype
read matched ttype
x (Identifier) matched ttype
proc matched ttype
gamal (Identifier) matched ttype
( matched ttype
integer matched ttype
i (Identifier) matched ttype
, matched ttype
j (Identifier) matched ttype
; matched ttype
integer matched ttype
k (Identifier) matched ttype
, matched ttype
l (Identifier) matched ttype
; matched ttype
boolean matched ttype
m (Identifier) matched ttype
) matched ttype

test.txt - Notepad
File Edit Format View Help
begin
integer x,y,w
boolean z
proc kareem= read x
proc gamal(integer i,j;integer k,l;boolean m)= y:=x+i*j+k*l/2

call kareem;
call gamal(x,y,x,w,z);
if x>y then w:=x+y end if;
write w
end
$
```

This is a part of parser and check semantic with two passes.



```
Display code generation :
CALL kareem
CALL gamal 5
GT x y T1
IFN T1 GOTOL1
ADD x y T2
ASSIGN T2 w
WRITE w
Press any key to continue . . .

test.txt - Notepad
File Edit Format View Help
begin
integer x,y,w
boolean z
proc kareem= read x
proc gamal(integer i,j;integer k,l;boolean m)= y:=x+i*j+k*l/2

call kareem;
call gamal(x,y,x,w,z);
if x>y then w:=x+y end if;
write w
end
$
```

This is the code generation for this program.