תורת הקומפילציה

תרגיל בית 3

הוראות לתרגיל 3 בשפת MINI-PASCAL

עליכם להמשיך ולממש בסביבת Eclipse (Java) את הפונקציות המתאימות לפי החומר שנלמד בהרצאה ע"מ שהמהדר הנוצר יוכל לייצר קוד עבור:

1. Static arrays
2. Records
3. Functions
4. Procedures

עליכם לייצר קוד התומך בשיטות העברת פרמטרים:

1. By-Value.
2. By-Reference (called IDENTICAL in mini).

יהיה עליכם לחשב את ערכו של sep עבור כל פונקציה/פרוצדורה ע"י הפעלת הפונקציה `getMaxNeededStackSpace()` (שהגדרתם בתרגיל 2) על האובייקט `_block` של `ProcedureDeclaration` או `FunctionDeclaration`.
באופן דומה יהיה עליכם לחשב את ה ssp עבור כל פונקציה/פרוצדורה.

```
public abstract void code(PrintWriter pw);

public abstract void codeL(PrintWriter pw);

public abstract void codeR(PrintWriter pw);

public abstract void codeI(PrintWriter pw, String arrayName, int
dimension);

public abstract void codeA(PrintWriter pw, String func_name, int
param_num);
```

תוכלו להשתמש בטבלת הסמלים הקיימת תוך כדי הרחבתה לתמוך בשמות הגדרות וקריאות לפונקציות.
כמו כן תוכלו להוסיף מחלקות ופונקציות נוספות כרצונכם.
בהמשך תוכלו לראות מספר דוגמאות.

אתם מתבקשים לתמוך בפקודת ההדפסה של MINI שנקראת .WRITE שאפשר להשתמש בה בשתי דרכים:

1. WRITE("hello world!")
2. WRITE(x+y)

המימוש של הצורה הראשונה יהיה ע"י הוספת הקטע קוד הזה, בתוך הפונקציה code של מחלקה WriteStrStatement בקובץ Node.java :

```
public void code(PrintWriter pw) {
      pw.println("ldc s " + str_);
      pw.println("prnt");
}
```

המימוש של הצורה השנייה יהיה ע"י הוספת הקטע קוד הזה, בתוך הפונקציה code של מחלקה WriteVarStatement בקובץ Node.java :

```
public void code(PrintWriter pw) {
      exp_.codeR(pw);
      pw.println("prnt");
}
```

<u>בהצלחה</u>

<u>הגדרות טיפוס חדש (רשומה) בשפת MINI</u>

REC RECORD {

x FIXED;

y FIXED;

```
        p^ REC;
};
```

הגדרות משתנים בשפת MINI

משתנה פשוט – simple type

```
var1   FIXED;          ( integer type )
var2   FLOAT;          ( real type )
var3   BOOLEAN;        ( boolean type )
```

משתנה רשומה – record var

```
r1  REC;          (REC type )
p1^ REC;          (pointer to REC)
```

משתנה מסוג מערך – array type

```
name  ARRAY   [low bound : high bound ]  OF   type;
```

Examples :

```
arr    ARRAY   [0:4]  OF FIXED;
mat1   ARRAY   [1:5]  OF ARRAY  [2:5]  OF   BOOLEAN;
mat2   ARRAY   [1:5]  OF arr;
```

פנייה למשתנים

פנייה למשתנה מסוג רשומה / מצביע לרשומה

```
NEW(p1)        (initialized the pointer p1)
p1^.x          (accessing x field of record (p1^))
r1.x           (accessing x field of record r1 (not pointer))
p1^.p^.y
```

פנייה למשתנה מסוג מערך

מערך חד מימדי :

arr [expr / INTCONST]

מערך רב מימדי :

mat [expr / INTCONST][expr / INTCONST]

Examples:

Arr[1] = arr1[I + 4];

Mat [2][j + 2] = mat1 [3][5];

פונקציות ופרוצדורות בשפת MINI

כללי

פונקציה – רוטינה שמחזירה ערך.

פרוצדורה – רוטינה שלא מחזירה ערך.

פונקציות ופרוצדורות בשפת MINY, מוגדרות בתוך התוכנית הראשית, או בתוך פונקציות אחרות לאחר שלב ההגדרות ולפני הבלוק של הקטע הביצועי בתוכנית.
לפיכך קיימת אפשרות לקינון פונקציות ופרוצדורות בשפה. טווח ההכרה של הפונקציות, זהה לטווח ההכרה של המשתנים בתוכנית, כלומר ניתן להפעיל אותם מתוך התוכנית שהגדירה אותם, מתוך עצמם או מתוך תת-פונקציות שלהם או של התוכנית שהגדירה אותם.

<u>מיקום ההגדרה</u>

לדוגמא :

```
PROGRAM example
        a   FIXED;
        b   FIXED;

      PROCEDURE sort ( ……..)
      {
      .
      .
      }
{
.
.
}
```

<u>הגדרת פרוצדורה</u>

מילה שמורה שם by Value פרמטרים by reference פרמטר

```
PROCEDURE   proc   ( a   FIXED,  b   FIXED,  c   FIXED IDENTICAL )
        d   FIXED;
{
}
```

<u>הגדרת פונקציה</u>

מילה שמורה שם by Value פרמטר by reference פרמטר ערך מוחזר

```
FUNCTION    func   (  a   FIXED,   c   FIXED   IDENTICAL )   :   FIXED
        d   FIXED;
{
```

func = d ; (החזרת ערך (השמת הערך המוחזר לתוך שם הפונקציה

}

בסוף הגדרת הפונקציה יש להוסיף ":" ואחריהם את הטיפוס של הערך המוחזר מהפונקציה.

החזרת הערך מתבצעת בסוף הפונקציה כאשר מבצעים השמה של הערך המוחזר לתוך שם הפונקציה, כפי שמופיע בדוגמא. למעשה כאילו מבצעים השמה למשתנה ששמו זהה לשם הפונקציה ואין צורך להגדירו.

<u>פרמטרים</u>

קיימות שתי צורות להעברת פרמטרים :

1. <u>by Value</u> - העתקת <u>הערך</u> של המשתנה למחסנית (כלומר ערכו של הפרמטר לא משתנה ביציאה מהפונקציה, גם אם שינו אותו בתוך הפונקציה).

הגדרת פרמטר כזה : טיפוס הפרמטר שם הפרמטר.

דוגמא : FIXED param

2. <u>by Reference</u> - העתקת <u>הכתובת</u> של המשתנה למחסנית (כלומר שינוי הפרמטר בתוך הפונקציה משנה את ערכו גם ביציאה מהפונקציה).

כדי לזהות פרמטר שמועבר בצורה זו משתמשים במילה השמורה – IDENTICAL בסוף ההצהרה על הפרמטר.

הגדרת פרמטר כזה : IDENTICAL טיפוס הפרמטר שם הפרמטר.

דוגמא : IDENTICAL param FIXED

<u>קריאה לפונקציה/פרוצדורה</u>

הפעלת פונקציות ופרוצדורות זהה לשפות אחרות. הקריאה מתבצעת ע"י כתיבת שם הרוטינה ואחריה הפרמטרים המועברים.

קריאה לפרוצדורה היא פקודה בשפה , לדוגמא :

swap(a, b)

בעוד שפונקציה יכולה להחזיר ערך ולכן ניתן להציב את הערך המוחזר לתוך משתנה ואז הקריאה לפונקציה תהיה כחלק מביטוי השמה בתוכנית, לדוגמא :

ReturnValue = func (a,b);

מבנה עץ התוכנית

הגדרת פרוצדורה

Class ProcudureDeclaration
String name_ (field holds Procedure name)

אם אין פרמטרים :

formal_list_ (which should be instance of class ParameterList) is null
procedure's block_ :
declarations
sub-proc/func declarations
code .

אם יש פרמטרים :

formal_list_ is an instance of class ParameterList – (details later)
And the same for Block:
procedure's block_ :
declarations
sub-proc/func declarations
code .

הגדרת פונקציה

Class FunctionDeclaration
String name_ (field holds Procedure name)

ההבדל היחיד הוא שיש שדה type אשר מכיל את טיפוס הערך המוחזר.

הגדרת פרמטרים

class ParameterList
parameter_list is defined like any other list ontains the list of parameters of current function/procedure. Each parameter in the list can be byValue, or byRef as defined below

הגדרת פרמטר - byValue

Class ByValueParameter
string_ holds parameter's name.
type_ could be any type. (The parameter type).

הגדרת פרמטר - byReference

Class ByReferenceParameter
string_ holds parameter's name.
type_ could be any type. (The parameter type).

קריאה לפונקציה/פרוצדורה

Class ProcedureStatement
string_ holds procedure/function name.

אם אין פרמטרים :

expr_list_ is null

אם יש פרמטרים :

expr_list_ is an instance of class ExprList which is a list of expressions

It basically contains the list of parameters of current function/procedure.
Each parameter in the list is an expr (var , expression etc).

דוגמאות לקוד הנוצר

דוגמא 1: עבור התוכנית הנתונה

```
PROGRAM main
        x FIXED;
        y FIXED;
        i FIXED;
        arr ARRAY [1:5] OF FIXED;
        PROCEDURE swap(a FIXED IDENTICAL, b FIXED IDENTICAL(
                temp FIXED;
```

```
        }
                temp = a;
                a = b;
                b = temp;
        {
    FUNCTION factorial(c FIXED): FIXED
        }
                IF(c == 0)THEN
                        factorial = 1;
                ELSE
                        factorial = c * factorial(c-1(;
                FI
        {
    PROCEDURE p(d ARRAY[1:5] OF FIXED(
        }
                d[2] = 20;
        {
}
    x = 4;
    y = 5;
    swap(x, y);
    i = factorial(x);
    p(arr);
    WRITE("i=");
    WRITE(i);
{
```

הקוד הנוצר

ssp 13	ind i	ssp 10	lda 0 5
sep 10	ldc i 0	sep 2	ind i
ujp L1	equ i	lda 0 5	cup 1 L5
L4	fjp L2	ldc i 2	sto i
ssp 8	lda 0 0	chk 1 5	mst 0
sep 2	ldc i 1	ixa 1	lda 0 8
lda 0 7	sto i	dec a 1	movs 5
lod a 0 5	ujp L3	ldc i 20	cup 5 L6

ind i	L2	sto i	ldc s i=
sto i	lda 0 0	retp	prnt
lod a 0 5	lda 0 5	L1	lda 0 7
lod a 0 6	ind i	lda 0 5	ind i
ind i	mst 1	ldc i 4	prnt
sto i	lda 0 5	sto i	stp
lod a 0 6	ind i	lda 0 6	
lda 0 7	ldc i 1	ldc i 5	
ind i	sub i	sto i	
sto i	cup 1 L5	mst 0	
retp	mul i	lda 0 5	
L5	sto i	lda 0 6	
ssp 6	L3	cup 2 L4	
sep 8	retf	lda 0 7	
lda 0 5	L6	mst 0	

דוגמא 2: עבור התוכנית הנתונה

```
PROGRAM main
        n FIXED;
        i FIXED;

        FUNCTION fact(n FIXED) : FIXED
        {
                IF n == 0 THEN
                        fact = 1;
```

```
        ELSE
                fact = fact(n-1)*n;
        FI
    }

{
    i=0;
    WHILE (i < 7)
    {
        n = fact(i);
        WRITE(i);
        WRITE("!=");
        WRITE(n);
        i = i+1;
    }
}
```

הקוד הנוצר

ssp 7	lda 0 0	lda 0 6	prnt
sep 7	mst 1	ind i	lda 0 6
ujp l1	lda 0 5	ldc i 7	lda 0 6
l2	ind i	les i	ind i
ssp 6	ldc i 1	fjp l8	ldc i 1
sep 8	sub i	lda 0 5	add i
ujp l3	cup 1 l2	mst 0	sto i
l3	lda 0 5	lda 0 6	ujp l7
lda 0 5	ind i	ind i	l8
ind i	mul i	cup 1 l2	stp
ldc i 0	sto i	sto i	
equ i	l5	lda 0 6	
fjp l4	retf	ind i	
lda 0 0	l1	prnt	
ldc i 1	lda 0 6	ldc s !=	
sto i	ldc i 0	prnt	
ujp l5	sto i	lda 0 5	
l4	l7	ind i	

אתם יכולים לבדוק את הקוד הנוצר ע"י Pmachine emulator המצורף:

1. open command prompt
2. change directory to the one containing Pemulator.jar
3. issue this command: java –jar Pemulator.jar code.txt 100 > out.txt

The Pemulator will execute your p-machine code 'code.txt' and the result will be printed into 'out.txt'