שיטות הידור

פרויקט חלק 2

מגישים:

אריאל יהודה 318296365

נדב אסטרוגנו 209270875

גיל סייט 315816504

בחלק זה מימשנו מנתח תחבירי שידפיס עץ גזירה לפי חוקי הדקדוק שקיבלנו ובהתאם לקלט, בעזרת המנתח הלקסיקלי שמימשנו בחלק הקודם.

מבנה הנתונים בו השתמשנו הוא עץ שיורכב מצמתים המכילים סוג (בין אם זה אסימון מהמנתח הלקסיקלי או חלק שמאל של חוק גזירה מהמנתח התחבירי), ערך (עבור אסימונים רלוונטיים) ומצביעים לצמתים אחים/בנים.

בשביל יצירת העץ הגדרנו (בקובץ part2_helpers.h) את YYSTYPE כמצביע לצומת במקום ערך ברירת המחדל int. כך המנתח הלקסיקלי יוכל ליצור את הצמתים עבור אסימונים, והמנתח התחבירי עבור חוקים. עבור צומת השורש הצהרנו על מצביע לצומת שקראנו לו parseTree והשתמשנו בו עבור כלל הגזירה הראשון PROGRAM. בניית העץ תתבצע כך שבכל פעולת reduce ניצור צומת חדש ונחבר אותו לבנים שלו. לבסוף, הדפסת העץ תתבצע בעזרת פונקציית dumpParseTree שניתנה לנו.

בדקדוק הנתון מצאנו את הקונפליקטים הבאים:

<u>קונפליקט 1:</u>

```
CNTRL → if BEXP then STMT else STMT

CNTRL → if BEXP then STMT
```

במצב בו קיבלנו `if BEXP then STMT else` בקלט והמנתח נמצא אחרי `STMT` , ייווצר קונפליקט shift/reduce בין 2 החוקים לעיל. לפי הגדרת העדיפויות של שפת c נרצה לבצע shift. לכן כדי לפתור את הקונפליקט כתבנו את `else` לאחר `then` :

```
%precedence THEN
%precedence ELSE
```

כך תוגדר עדיפות ל `else` על פני `then` .

<u>קונפליקט 2:</u>

```
(1)   EXP → EXP addop EXP
(2)   EXP → EXP mulop EXP
(3)   EXP → ( TYPE ) EXP
```

במצב בו קיבלנו `( TYPE ) EXP addop` או `( TYPE ) EXP mulop` בקלט והמנתח נמצא אחרי הסוגריים, ייווצר קונפליקט shift/reduce בין חוק 3 לחוק 1 או 2. לפי הגדרת העדיפויות של שפת c (casting קודם לפעולות חיבור או כפל) נרצה לבצע reduce. לכן כדי לתת עדיפות לסוגריים כתבנו אותן לאחר פעולות החיבור והכפל:

```
%left ADDOP
%left MULOP
%right ( )
```

*האסוציאטיביות מוגדרת לפי הנהוג בשפת c.

את שאר האסוציאטיביות והעדיפויות הגדרנו לפי הנהוג בשפת c:

```
%right ASSIGN
%left OR
%left AND
%left RELOP
%left ADDOP
%left MULOP
%right NOT ( )
```