עבודה עם bison

- סדרת הפעולות הנדרשת כדי לקמפל את המתרגם כולו היא:
  1. bison –defines source.ypp
  2. lex source.lex
  3. g++ source.tab.cpp lex.yy.c

  שימו לב ששלב (1) חייב לבוא לפני (2) ולא להיפך, כדי ששינויים שביצעתם בהגדרת האסימונים בקובץ source.ypp ישתקפו במנתח הלקסיקלי (שמשתמש בקובץ שמיוצר ע"י bison, source.tab.hpp).
  מומלץ להכין סקריפט שמבצע את שלושת הפעולות הנ"ל.
- מומלץ להגדיר את כל הטיפוסים של התכונות ואת YYSTYPE בקובץ h. נפרד, ולעשות לו include מתוך הגדרות המנתח הלקסיקלי (source.lex) והתחבירי (source.ypp), כדי ששני המנתחים "יכירו" את הטיפוסים.
- בקובץ ההגדרות ל- bison (*.ypp), בחלק האחרון (פונקציות C), **חובה** לממש את הפונקציות הבאות:
  - main: פונקציה זו צריכה לבצע פעולות אתחול (טבלת הסמלים וכו') ואז לקרוא ל- ()yyparse, שמבצעת את הקוד של המנתח. (בלי פונקציית main שקוראת ל- yyparse, כשתריצו את המנתח לא יקרה כלום.)
    השלד של הפונקציה main נראה כך:

```
main()
{
        yyparse();
}
```

  - yyerror: פונקציה זו נקראת כשהמנתח נתקל בשגיאה תחבירית, והיא אמורה לדווח על השגיאה ולטפל בה. מימוש אפשרי:

```
int yyerror(char * s)
{
        printf("Parse error: %s! Bye now\n");
        exit(1);
        return 0;
}
```

- אם רוצים להשתמש במשתנה של lex (yytext, yylineno, yyleng) מתוך המנתח התחבירי, צריך להכריז עליו כמשתנה חיצוני:

```
extern int yylineno;
```

  בחלק הראשון (הכרזות C/C++) של קובץ ה- bison (*.ypp).

- המנתח התחבירי שנבנה ע"י lex "ממחזר" את המשתנים שלו, ובפרט yytext. לכן אם רוצים להחזיר את הלקסמה שזוהתה למנתח התחבירי, צריך להעתיק את הלקסמה למקום אחר (למשל בעזרת strcpy), כי בפעם הבאה שהמנתח הלקסיקלי יזהה אסימון הוא ישחרר את הזכרון שהוקצה ללקסמה הקודמת.

איך מדבגים את המנתח שנוצר ע"י bison

- כדי לקבל הודעות שגיאה קצת יותר מפורטות מ- "parse error", אפשר להוסיף בחלק הכרזות ה- C/C++:

```
#define YYERROR_VERBOSE 1
```

- ל- bison יש יכולת להדפיס תוך כדי הניתוח מידע מפורט על מצבו הנוכחי: המצב באוטומט, האסימונים אותם הוא מצפה לקבל, תוכן המחסנית והפעולה הנבחרת בכל צעד. ע"מ לקבל את ההודעות האלו בזמן ריצה צריך להגדיר:

```
#define YYDEBUG 1
```

הגדרה זו גורמת ל- bison להוסיף את יכולות ה- trace למנתח, אבל עדיין לא מפעילה אותן. כדי להפעיל אותן צריך להוסיף במקום כלשהו ב - main:

```
yydebug = 1;
```

טיפול בקונפליקטים

אחרי ש- bison מסיים לבנות את המנתח, הוא מודיע אם נמצאו קונפליקטים כלשהם, אבל קונפליקטים לא ייגרמו לו להיכשל (bison פשוט ייבחר פעולה כלשהי). אם נמצאו קונפליקטים, יש לטפל בהם ע"י הגדרה מפורשת של אסוציאטיביות ועדיפות (בעזרת %left ו- %right בסדר הנכון, לפי סדר העדיפויות של האסימונים; ובעזרת %prec כדי לתת עדיפות לכללים).

כדי למצוא קונפליקטים, או אם רוצים לוודא שהאוטומט וטבלת הניתוח נכונים, bison יכול לייצר קובץ שמתאר את האוטומט וטבלת הניתוח שבנה. כדי לקבל את הקובץ הזה מריצים את bison עם האופציה verbose--. הפלט יתקבל בקובץ source.output.

שימו לב: bison כותב בכל מצב רק את הגרעין (kernel); ה- closure אינו נכתב לקובץ ה- output., למרות ש- bison מחשב אותו כרגיל.
על חלק ממחשבי ה- Linux בחווה מותקנת גרסה מתקדמת יותר של bison, אשר מאפשרת לכתוב את המצב המלא (כולל closure). זוהי גרסה 1.85, וכדי לקבל את ה- closure בעזרתה יש להריץ:

```
bison --verbose --report=itemset <file>
```

(פקודה זו לא תעבוד על csl1.)

מחשבים בחווה עליהם מותקנת גרסה 1.85

מחשבים אליהם יש להתחבר עם סיסמת csl1:

eas47
eas28
eas42
eas52
ibm25
ibm21
eas29

מחשבים אליהם יש להתחבר עם סיסמת Windows (td-csf):

ibm70
ibm14
ibm17