פרויקט – חלק 1

כפי שבוודאי כבר סיפרו לכם, עד סוף הסמסטר תממשו קומפיילר בסיסי. בתרגיל זה נתחיל בבניית היסודות עליהם יעמוד הקומפיילר.

שפת הקלט לקומפיילר שלכם היא שפת --C הידועה ☺, ובתרגיל זה עליכם לבנות מנתח לקסיקלי אשר מקבל קלט של תוכנית בשפה, מחלק אותו לאסימונים, ומדפיס כל אסימון שזיהה.

אסימוני השפה --C:

מילים שמורות:

int float void write read va_arg
while do if then else return

סימנים:

( ) { }
, ; : ...

אסימונים מורכבים:

האסימון	אוסף הלקסימות המתאימות
id	מתחיל באות (לטינית גדולה או קטנה) וממשיך באותיות או ספרות או קו-תחתי (_)
integernum	מספר שלם (אין צורך לתמוך במספרים שליליים)
realnum	מספר בעל נקודה עשרונית (אין צורך לתמוך במספרים שליליים)
str	מחרוזת מתחילה בגרשיים ומסתיימת בגרשיים ומכילה את כל התווים שביניהם. • מחרוזת לא יכולה להתפצל בין מספר שורות. אם מסתיימת השורה והמחרוזת לא הסתיימה (חסר גרשיים סוגרים) יש לטפל בשגיאה כמפורט בעמוד הבא. התו גרשיים עצמו יכול להופיע באמצע המחרוזת אם לפניו יש תו לוכסן '\' (escaped). לדוגמה: "This is a \"test\"" . • ה- escaped characters המותרים בתוך המחרוזת הם: \t, \n, \"
פעולות אריתמטיות/לוגיות: הסימן "|" מסמן "או", למעט בהגדרה של האסימון or	
relop	== | <> | < | <= | > | >=
addop	+ | -
mulop	* | /
assign	=
and	&&
or	||
not	!

הערות:

הערות בשפה מתחילות בסימן # בכל מקום בשורה, ומסתיימות בסוף השורה. לפני סימן תחילת ההערה יכולים להופיע בשורה אסימונים מסוגים אחרים. מתחילת ההערה ועד סוף השורה אין לפענח אסימונים של השפה, אלא כל התווים של ההערה נכללים בלקסמה של ההערה, כלומר החל בתו # (כולל) ועד סוף השורה.

ההערה אינה כוללת את סימן מעבר לשורה חדשה שבסוף השורה. סימן # בתוך מחרוזת (כלומר, בין גרשיים, כמוגדר לעיל) אינו מתחיל הערה אלא נכלל בתוכן המחרוזת כמו תווים רגילים.

רווחים:

בין אסימונים יכולים להופיע תווי רווח - whitespace (רווח רגיל, טאב, שורה חדשה) – אבל לא חייבים להיות.

שימו-לב, שורה חדשה יכולה להיות בסגנון UNIX (LF) או בסגנון DOS (CR+LF) וודאו שבדקתם קלטים עם סוף שורה משני הסוגים. ניתן להיעזר ביישום dos2unix/unix2dos על מנת לייצר קלטים מתאימים.

המשימה:

- כתבו מנתח לקסיקוגרפי (קובץ Lex עבור הכלי Flex) אשר מקבל תוכנית בשפת --C, מזהה את האסימונים שבה ומדפיס אותם בצורה הבאה (בהתאם לחלוקה לעיל של מבנה השפה לקבוצות):
  1. עבור מילה שמורה יודפס <*reserved_word*> (כלומר, המילה השמורה בסוגרים), למשל: <int>
  2. עבור **integernum** יודפס <integernum,*integernum_lexeme*>, למשל: <integernum,3>.
  3. עבור **realnum** יודפס <realnum,real*num_lexeme*>, למשל: <realnum,3.5>.
  4. עבור **id** יודפס <id,*id_lexeme*>, למשל: <id,foo>.
  5. עבור פעולות אריתמטיות/לוגיות יודפס שם הקבוצה של הפעולה ושם הפעולה: <op_type,*op_lexeme*>, למשל: <relop,==>
  6. עבור str יודפס <`str`,*string lexeme*> כאשר הלקסמה - תוכן המחרוזת – מודפסת ללא הגרשיים בהתחלה ובסוף. במידה ויש שימוש ב-escaped characters , כלומר תווים שיש לוכסן לפניהם, יש להדפיס את המחרוזת בצורה המקורית שלה, כלומר שיראו את תווי הלוכסן, ולא את הפרשנות שלהם. למשל, עבור הקלט:
     `"This is a \"test\"\n123\n"` יש להדפיס:
     `<str,This is a \"test\"\n123\n>`

כלומר, באופן כללי: <token_type[,*value*]> כאשר token_type הוא סוג האסימון הנגזר ו-*value* הוא הלקסמה של האסימון הנגזר, בסוגי האסימון הרלוונטיים. שימו לב: אין להוסיף בפלט רווחים בין סימני '<' או '>' או ',' ובין הערכים המודפסים.

חשוב: ניתוח הקלט וסיווג הלקסמות לאסימונים יעשה באמצעות מנגנון זיהוי האסימונים של Flex בלבד! עליכם להגדיר ביטויים רגולריים מתאימים לכל סוג אסימון ולשייך לכל ביטוי שכזה פעולה מתאימה, כפי שנלמד. אין להשתמש בקוד C++/C בפונקציות לעיבוד והשוואת מחרוזות ואין לממש השוואת מחרוזות בעצמכם ב-C++/C לצורך ניתוח הקלט וסיווג הקלט לאסימונים!

הוראות נוספות:

1. יש להגדיר בקובץ ה-Lex מקרואים עבור הביטויים הרגולריים של id, str ו- num בחלק ה-Definitions ולהשתמש בהם בחלק ה-Rules כנ"ל.
שמות המקרואים יהיו כשמות האסימונים (id, integernum, realnum, str). הגדרות אלו ייבדקו בנפרד בנוסף לבדיקת תפקוד המנתח.
2. המנתח הלקסיקלי צריך להתעלם מאסימוני הערות (כלומר, הלקסמות המתאימות לאסימון מסוג הערה לא תופענה בפלט).
3. כל אסימון מקבוצת ה"סימנים" יועבר לפלט ללא שינוי.
4. כל תו מסוג רווח (רווח רגיל, טאב, שורה חדשה) יועבר לפלט ללא שינוי.

- מצורפת דוגמה לפלט מצופה עבור תכנית דוגמה: $example.cmm \rightarrow example.tokens$
- הקלט של המנתח מגיע מהקלט הסטנדרטי (`stdin`) והפלט צריך להיכתב לפלט הסטנדרטי. אין צורך לפתוח קבצים לקלט או לפלט. ניתן להעביר למנתח קובץ באמצעות redirection, לדוגמה:
$\$ ./part1 < example.cmm$

טיפול בשגיאות:

אם זוהה סימן שאיננו אסימון חוקי בשפה, יש להדפיס **<u>בשורה חדשה</u>** הודעת שגיאה בתבנית הבאה ולצאת מיידית מהתוכנית עם קוד שגיאה 1:

Lexical error: ’<lexeme>’ in line number <line_number>

כאשר <lexeme> הינה הלקסמה (הסימן) הלא חוקית שזוהתה ו-<line_number> הוא מספר השורה של אותה לקסמה. לדוגמא:

Lexical error: ’@’ in line number 2

כלומר, בפלט יופיע הניתוח של כל מה שהופיע לפני השגיאה ואז תופיע הודעת השגיאה בשורה נפרדת, והניתוח ייפסק.

* מצורפת דוגמה לפלט מצופה עבור תכנית עם שגיאה:

C—source file: `example-err.cmm` error output: `example-err.tokens`

כללי:

- התרגיל נבדק באופן חצי אוטומטי. יש להקפיד על מבנה הפלט כמפורט ולא להוסיף או לגרוע רווחים כדי שיתאים לפלט המצופה במדויק.
- סביבת הבדיקה הרשמית הינה המכונה הווירטואלית של לינוקס המסופקת לכם. ניתן להוריד את קובץ המכונה ליבוא לסביבת **VirtualBox** באמצעות הקישור המסופק באתר הקורס. באותו מקום ניתנים גם מספר קישורים המתעדים התקנת **VirtualBox** במחשבכם האישי. יש להשתמש בגרסה עדכנית של **VirtualBox** לתפקוד מיטבי של המכונה (רצוי גרסה 5.1.x ומעלה).
- ניתן לפתח במחשב אחר מהמכונה הווירטואלית, אולם חובה לוודא שהתרגיל המוגש נבנה ורץ היטב במכונה הווירטואלית, לפני ההגשה.

  *המלצה: אם אתם מפתחים במכונה שונה מהמכונה הרשמית, בִּדקו באופן שוטף במהלך העבודה על התרגיל את תאימות המימוש שלכם במכונת היעד הרשמית ואל תידחו זאת לרגעים האחרונים לפני ההגשה. לא תהיה התחשבות בהגשה שעובדת במכונה שלכם אבל לא במכונה הרשמית לבדיקה. לא תאושר דחיה בהגשה לצורך התאמה למכונת היעד הרשמית.
- לא מומלץ לפתח בסביבות מבוססות ווינדוס. כלי ה**Flex** עלול להתנהג באופן שונה בסביבת ווינדוס לעומת סביבת המטרה (לינוקס). מעבר בין הסביבות עלול לחשוף שגיאות שקשה לאתר בגלל מבנה קבצים שונה וחוסר תאימות מלא לגרסאות התוכנה בשרתים.
- משאבים נוספים לגבי השימוש בכלי ה-Flex (תיעוד ודוגמאות) תוכלו למצוא במצגת התרגול הראשון ובאתר הקורס (תחת "חומר עזר לקורס ולפרויקט").
- שימו-לב למדיניות בנוגע לאיחורים בהגשה המפורסמת באתר הקורס. במקרה של נסיבות המצדיקות איחור, יש לפנות מראש לצוות הקורס לתיאום דחיית מועד ההגשה.
- הקפידו לוודא כי העליתם את הגרסה של ההגשה אותה התכוונתם להגיש. לא יתקבלו טענות על אי התאמה בין הקובץ שנמצא ב-Moodle לבין הגרסה ש"התכוונתם" להגיש ולא יתקבלו הגשות מאוחרות במקרים כאלו.

הוראות להגשה:

- מועד אחרון להגשה: יום א 01/12/2024 בשעה 23:55.
- ההגשה בזוגות. הגשה בבודדים תתקבל רק באישור מראש מצוות הקורס.
- יש להגיש בצורה מקוונת באמצעות אתר ה-Moodle של הקורס, מחשבונו של אחד מהסטודנטים.
- יש להגיש קובץ ארכיב מסוג Bzipped2-TAR בשם מהצורה (שרשור מספרי ת.ז – 9 ספרות):

  `proj-part1-<student1_id>-<student2_id>.tar.bz2`

  לדוגמה: `proj-part1-012345678-345678901.tar.bz2`
- בקובץ הארכיב יש לכלול את הקבצים הבאים:
  - את קובץ ה LEX בשם `part1.lex`
  - קובץ `makefile` היוצר את המנתח הלקסיקלי - שם קובץ ההרצה של המנתח הנוצר צריך להיות **`part1`**
- קובץ הארכיב צריך להיות "שטוח" (כלומר, שלא ייצור ספריות משנה בעת הפתיחה אלא הקבצים ייווצרו בספריה הנוכחית).
- דוגמה לפקודה ליצירת הארכיב (בלינוקס) – נקראת מהספרייה בה נמצאים קבצי ההגשה:

```
$ tar cjf proj-part1-012345678-345678901.tar.bz2 part1.lex makefile
```

- על התרגיל להתקמפל ולרוץ בהצלחה במכונה הווירטואלית של לינוקס המסופקת לכם.
- <u>תרגיל שלא יצליח להתקמפל יקבל 0.</u>

בהצלחה!