

# תורת הקומpileציה

## תרגיל בית 1 – בניית מנתח לקסיקלי

מתרגל אחראי: נעם הר טוב

ההגשה בזוגות

עבור כל שאלה על התרגיל, יש לעין ראשית **בפיאצה** ובמידה שלא פורסמה אותה השאלה, ניתן להוסיף אותה ולקלבל מענה, אין לשЛОח מיילס בנושא תרגיל הבית כדי שנוכל לענות על השאלות שלכם ביעילות.

**תיקונים לתרגיל יסומנו בצהוב**, חובהכם להתעדכן בהם באמצעות קובץ התרגיל.

התרגיל יבדק בבדיקה אוטומטית. **הקפידו למלא** אחר ההוראות **במדוייק**. הבדיקה תבצע על שרת הקורס csComp.cs.technion.ac.il, אך עליוכם לוודא שהתרגיל שלכם פועל על השרת ובהתאם להנחיות.

### הנחיות כלליות

- בתרגיל זה תIALIZEDו מנתח לקסיקלי שיוכל לטפל בשפת C. שפה זו היא דומה לשפת C שאתם מכירים, הכוללת פעולות אריתמטיות, פונקציות, המרות ועוד.
- במנתח הלקסיקלי השתמשנו נשתמש כדי ליצור תכנית הקוראת קלט מהמשתמש ומדפיסה מידע על האסימונים שהוא מצאה.
- יש להשתמש ב- flex בלבד (ולא ב- lex).

### הגדרות מושגים כלליים

- **רווח לבן** (whitespace) – אחד מבינן: רווח (ספייס), טאב (הטו t\), CR (הטו r\), LF (הטו n\).
- **תווים ניתנים להדפסה** (printable characters) – התווים שערך ה-ASCII שלהם בין 0x20 ל-0x7E.
- (כולל את הקצתות), או רווחים לבנים:
  - ניתן לקרוא על תווים ניתנים להדפסה בהרחבה בויקיפדיה בערך הבא:  
[https://en.wikipedia.org/wiki/Printable\\_characters](https://en.wikipedia.org/wiki/Printable_characters)
- **רצף ברירה** (escape sequence) – לדוגמה אחורי (הטו \) ואחריותו או יותר שביחד מפורשים כתו אחד.
  - דוגמאות: \n – ירידת שורה, \t – טאב.
  - ניתן לקרוא על רצפי ברירה בהרחבה בויקיפדיה בערך הבא:  
[https://en.wikipedia.org/wiki/Escape\\_sequences\\_in\\_C](https://en.wikipedia.org/wiki/Escape_sequences_in_C)

## הגדרת אסימונים

שם האסימון	תיואר	ערכיהם אפשריים	דוגמאות	אנט-דוגמאות
VOID	המילה השמורה void	void	void	diov
INT	המילה השמורה לטיפוס מסוג Integer	int		long
BYTE	המילה השמורה לטיפוס מסוג Byte	byte		bit nibble
BOOL	המילה השמורה לטיפוס מסוג Boolean	bool		boolean
AND	המילה השמורה לאופרטור מסוג and (בשפת C: &&)	and		And
OR	המילה השמורה לאופרטור מסוג or (בשפת C:   )	or		Or light
NOT	המילה השמורה לאופרטור מסוג not (בשפת C: !)	not		Not
TRUE	המילה השמורה ללייטרל "אמת"	true		True 1
FALSE	המילה השמורה ללייטרל "שקר"	false		False 0
RETURN	המילה השמורה לחזרה מפונקציה	return		Return
IF	המילה השמורה ל- if עבור מבנה הבקרה של תנאי	if		If IF
ELSE	המילה השמורה ל- else עבור מבנה הבקרה של תנאי	else		Else ELSE
WHILE	המילה השמורה עבור מבנה הבקרה של while לולאת while	while		While
BREAK	המילה השמורה עבור עצירה ויציאה מולאה	break		Break BREAK
CONTINUE	המילה השמורה עבור המשך ריצת הלולאה	continue		Continue CONTINUE
SC	נקודה פסיק	;	;	

.	,	,	,	פסיק	COMMA
[	(	)	(	סגור שמאלי	LPAREN
]	)	)	)	סגור ימני	RPAREN
<	{		{	סגור מסולסל שמאלי	LBRACE
>	}		}	סגור מסולסל ימני	RBRACE
{	[		[	סגור מרובע שמאלי	LBRACK
}	]		]	סגור מרובע ימני	RBRACK
==	=		=	אופרטור השמה	ASSIGN
>_< <>	== != < > <= >=		== != < > <= >=	אופרטור רצינו	RELOP
?	+		+	אופרטור בינארי	BINOP
:	- * /		- * /		
/* my comment */	// my comment	מתחילה ב-// שמופיע מחוץ למחוזת, ואחריו שני הילוקסנים יכול לבוא כל תו מלבד ירידת שורה: LF, .CRLF או CR	הערה שורה		COMMENT
12AB	x	צריך לעמוד ב כלליים הבאים: - יכול להכיל אותיות אנגליות קטנות וגדלות ומספרים בלבד.	מזהה (Identifier)		ID
42	max	על המזהה להתחיל עם אות אנגלית (קטנה או גדולה). על המזהה להכיל תו אחד לפחות.			
big_x	007	על המזהה לסיים עם אחד אם המספר מכיל תו אחד לפחות.			
050	0	צריך לעמוד ב כלליים הבאים: - אפסים מובילים אסורים (ראה דוגמא אסורה). על המספר להכיל תו אחד לפחות.	מספר שלם		NUM
5 . 6	102	אם המספר מכיל תו בודד הוא יכול להיות '0'.			
050b	0b	צריך לעמוד ב כלליים הבאים: - אפסים מובילים אסורים. על המספר להכיל תו אחד לפחות.	מספר שלם עם B עוקב		NUM_B
5 . 6b	102b	אם המספר מכיל תו בודד הוא יכול להיות '0'.			
50		על המספר להסתיים עם תו b.			
50 b					

'unmatching"	"simple"			
"unclosed	"also 'simple'"	אוסף תווים בתוך מרכאות כפולות. הערות: 1. אורך המחרוזת יכול להיות בגודל אפס או יותר.		מחרוזת
"2-lined String"	"escape new lines\n"	2. ניתן להניח כי אורך המחרוזת ב <li>המרכזות לא עולה על <b>1024</b> תווים.</li>		
"ba--d"	"hex2 \x3A"	3. ניתן לכלול כל <a href="#">תו הניתן לדפסה</a>		
"bad \ escape"	"hi\thow\tare\tyou"	<b>פרט</b> לתווים הבאים: a. לוכסן אחריו: \ b. מרכאות כפולות: " c. תו LF: \n (כאשר הוא מגיע כתו בודד) d. תו CR: \r (כאשר הוא מגיע כתו בודד) אליא אם כן הם מגיעים חלק <a href="#">מרצף בריחה</a> תקין. רשימת <a href="#">רצפי בריחה</a> תקינים: a. \\ b. \" c. \n d. \r e. \t f. \0 g. \DD\x כאשר DD מייצג ספירה הקסדצימלית אופן הטיפול <a href="#">רצפי בריחה</a> יסביר בהמשך, בחלק של <a href="#">הדפסת הליקסמה</a> .		STRING
"hex \x10"				
"hex2 \x02"		שימוש לב: כל <a href="#">רצף בריחה</a> שאינו בראשימה הנ"ל מהו <a href="#">קלט לא חוקי</a> .		

## הוראות התרגיל

עליכם לכתוב תוכנית שתשתמש מנתח ותכתב בקובץ בשם `main.cpp`.

בתכנית זו תשתמשו בפונקציה (`lexyy` שנוצרת ע"י `flex` ועליה לעמוד בדרישות הבאות:

- הmanınיח יתעלם מכל [הרווחים הלבנים](#), חוץ מבתור מחרוזת.
- כאשר המנתח מזהה אסימון, יש לפולט מידע על האסימון על ידי קריאה לפונקציה `output.hpp` המספקת `printToken lineno, token, value`.

כאשר:

- `lineno` – מספר השורה בה האסימון [מסתיים](#).
- `token` – מזהה אסימון מהקובץ `token.hpp` המספק לכם (לפי השמות בחלק [הגדרת אסימונים](#) "למעלה")
- `value` – ערך האסימון שזוהה, כולם הליקסמה, פרט למועדן של [הערות](#) [ומחרוזות](#), כמו שבסר להלן.
- ניתן להניח שככל הערכים המספריים בתרגיל ניתנים לאחסן על ידי הטיפוס `int`.

למשל, עבר אסימון `ID` אשר זואה עברו לקסמה `x` בשורה 12, יש לבצע את הקריאה הבא:

```
output::printToken(12, ID, "x");
```

הדף הלקסמה של מחרוזות  
מחרוזות יודפסו ללא המרคาות הכפולות המקיפות אותן.

נטפל ברצף הבריחה באופן הבא:

- \t, \n, \r, \f מוחלפים בסוג המתאים של רווח לבן (טאב, CR, LF)
  - \\ מוחלפת בולוכן אחורי יחיד (\)
  - \" מוחלפת במרคาות כפולות ("")
- רץ בבריחה של תוו ASCII (xx) מוחלף בתו בעל ערך ה-xx אשר מייצג את הרץ הקסדצימלי.
- כך למשל, עבור הרץ **41x1** יודפס התו A. אם הרץ מהוות ייצוג הקסדצימלי של תוו אשר לא ניתן להדפסה, יש להדפיס שגיאה מתאימה (ראה סעיף טיפול בשגיאות).
- הרץ הקסדצימלי יכול להיות מורכב גם מאותיות קטנות וגם מאותיות גדולות. כמו, **A6x1** ו-**e6x1** שחוקים.
  - דוגמה – הקלט הבא:
- ```
"Hello \x57orl!\\r\\nThis\\tis\\t\x63oo\x6C, as always."
```

תודפס באופן הבא:

```
1 STRING Hello World!  
This is cool, as always.
```

הדף הלקסמה של הערוות  
ניתן להעביר כל ערך עבור הparameter `value`, פונקציית ההדפסה תתעלם ממנו במקרה של הערת ותדפיס //.

### קלט ופלט המנתה

המנתיח מקבל את הקלט מהערך הסטנדרטי לקלט (stdin). `flex` משתמש בערזץ הקלט הסטנדרטי כברירת מחדל.

יש להיעזר בקובץ `output.hpp` המצורפים לתרגום על מנת לייצר פלט הנitin לבדיקה אוטומטית.  
ניתן לקבץ פלט לדוגמא. יש לבדוק כי המנתה שלכם פולט פלט זהה אליהם. הבדלים יגרמו לכישלון הבדיקות האוטומטיות.

### טיפול בשגיאות

ניתן להניח כי הקלט מכיל שגיאה אחת לכל היותר.

על מנת לדוח על שגיאות יש להשתמש בפונקציות הנתונות בקובץ **output.hpp**:

```
errorUnknownChar (character)                                     המנתה נתקל בתו לא חוקי
```

```
errorUnclosedString ()   שורה מסתינמת באמצעות מחרוזת
```

```
errorUndefEscape (sequence)                                     מחרוזת מכילה רצף בריחה שלא מופיע בהגדרת התרגום.  
למשל, עבור המחרוזת אשר מכילה את הרץ \q הparameter  
sequence יהיה sequence.
```

עבור מקרה בו הרץ \q מלווה בתווים שאינם מייצגים ערך  
הקסדצימלי או ערך הקסדצימלי מייצג תוו אשר לא ניתן להפදסה  
או שהמחרוזת נגמרה לפני שנitin לקרו 2 תוים לאחר ה-**x**  
(למשל עבור המחרוזת "F\x" ("hey")) או שהתו המיוצג לא ניתן  
להדפסה (למשל עבור המחרוזת "FF\x" ("FFFF")), הודעת השגיאה תכיל  
את רצף הבריחה המלא.  
כך, עבור הרץ **FF\x** ה-**sequence** יהיה **FFFF**.

## הערות נוספות על התרגילים

- בתרגיל זה תדרשו לכתוב קובץ **lex** יחיד בשם **scanner.lex**. שימרו עליו פשוט, ומשמו את הלוגיקה הרצiosa בקובצי **scanner.cpp**.
- באופן דיפולטי, הפונקציה **(yylex()** מחזירה טיפוס **int**, וחוזרת למשתמש כאשר קיימת פקודת **return** ב-**action** של האסימונ. (ראו שוף 23 בתרגול על המנתה הלקסיקלי).
- לתרגיל מצורף קובץ בשם **tokens.hpp** המכיל טיפוס וענוה הכלול בתוכו את כל האסימונים.
- ביצוע **include** לקובץ זה הינו בקובץ **scanner.lex** והן בקובץ ה- **scanner.cpp** מאפשר "תקשורות" בין המנתה ש-**flex** יוצר לבין התכנית שתכתבו. כמובן, התכנית שתכתבו תדע להבין אילו אסימונים המנתה מוחזיר.
- לדוגמה, נניח כי יש לנו אסימון בשם **FOR**, אך יכול להיכל בקובץ ה- **scanner.lex** ב-**section**:

```
rules section scanner.lex
for    return FOR
```

ואילו בקובץ ה- **scanner.cpp**:
- בנוספ, קובץ ה-**scanner.cpp** מכיל הגדרות שיאפשרו לכם להשתמש בפונקציה **(yylex()** ובמשתנים **yylineno**, **yytext**.
- לתרגיל מצורף קובץ טמפליט **main.cpp** המכיל את LOLAט הקריאה ל-**(yylex()**. העזרו בו.
- מומלץ להיעזר ב-**manual** של **flex** לצורך ביצוע התרגילים. כל יתר לבצע אותו על ידי שימוש ביכולות **match**.
- מתקדמות של **flex** שלא נלמדו בתרגולים כגון **regex patterns** מתקדמים ו-**debug mode**.
- טיפ: תוכלו להשתמש באתר <https://regex101.com> שעוזר בהבנה ובבנייה של מבניות **regex** מורכבות.
- טיפ: כאמור, לא תבדקו על דליות זיכרון, איקות קוד, וכדומה. ועדין, מומלץ לבדוק עם **valgrind**, **Wextra**, **-Wall**, **-Wmissing-declarations** לkiemפל עם

## הערות נוספות על תווים בקובץ

ניתן להניח כי קבצי הדוגמאות הם קבצי ASCII בלבד (כלומר: אינם UTF-8 או UTF-16). בהכינכם קבצי בדיקה, וודאו כי אתם מכוננים את ה-Encoding של הקובץ ל-ASCII או ANSI, או מבצעים as save C-ASCII.

לנוחותכם, וכך למנוע בעיות בהעתקה בין קבצים, להלן מפתח של התווים המזוכרים בתרגיל וערчи ה-ASCII שלהם:

| שם                 | סימן | ערך ASCII (hex) |
|--------------------|------|-----------------|
| סגור מרובע שמאלית  | [    | 5B              |
| סגור מרובע ימני    | ]    | 5D              |
| סגור מסולסל שמאלית | {    | 7B              |
| סגור מסולסל ימני   | }    | 7D              |
| נקודותים           | :    | 3A              |
| שווה               | =    | 3D              |
| סימן קראיה         | !    | 21              |
| לוכסן אחורי        | \    | 5C              |
| סולמית             | #    | 23              |
| נקודה פסיק         | ;    | 3B              |
| מינוס / מינוס      | -    | 2D              |
| פלוס               | +    | 2B              |
| פסיק               | ,    | 2C              |
| קו תחתון           | _    | 5F              |
| נקודה              | .    | 2E              |
| גרש                | '    | 27              |
| מרקאות כפולות      | "    | 22              |
| Carriage return    | CR   | 0D              |
| Line feed          | LF   | 0A              |
| רווח               |      | 20              |
| טאוב               |      | 09              |
| שטרודל             | @    | 40              |
| סגור משולש ימני    | >    | 3E              |
| טילדה              | ~    | 7E              |
| כוכבית             | *    | 2A              |
| לוכסן (סלש)        | /    | 2F              |

קובצי הטסט זמינים בקובץ koz ומומלץ תמייד להוריד ולהעביר אותם כ-koz על מנת למנוע שינוי אוטומטי של ירידות השורה על ידי תוכנות להעברת קבצים.

## הוראות הגשה

מסופק לכם קובץ Makefile שאיתו תקומפל הגשה שלכם. שימוש לב Ci קובץ ה-Makefile מאפשר שימוש ב-`STL`. אין לשנות את ה-`Makefile`.

יש להגיש קובץ אחד בשם `zip2-ID1D2`, עם מספרי ת"ז של שני המגנים. על הקובץ להכיל:

- קובץ `flex` בשם `scanner.lex` המכיל את כללי הניתוח הלקסיקלי.
- את כל הקבצים הנדרשים לבניית המנתח, כולל קבצים שסופקו כחלק מהתרגיל אם בחרתם להשתמש בהם.

בנוסף, יש להקפיד שהקובץ לא יכול את:

- קובץ ההרצתה.
- קבצי הפלט של `flex`.
- קובץ `Makefile` נוסף כחלק מהתרגיל.

יש לוודא כי ביצוע `zip2` לא נוצרת תייקה נפרדת. על המנתח להיבנות על השרת `csComp` ללא שגיאות באמצעות קובץ `Makefile` שסופק עם התרגיל. באתר הקורס מופיע קובץ `zip` המכיל קבצי בדיקה לדוגמה. יש לוודא כי פורתט הפלט זהה לפורתט הפלט של הדוגמאות הנתונות. למשל, ביצוע הפקודות הבאות:

```
unzip id1-id2.zip  
cp path-to/Makefile .  
cp path-to/hw1-tests.zip .  
unzip hw1-tests.zip  
make  
. /hw1 < t1.in 2>&1 > t1.res  
diff t1.res path-to/t1.out
```

יצור את קובץ ההרצתה בתיקיה הנוכחיות ללא שגיאות קומpileציה, ירייז אותו, ו-`diff` יחזיר 0.

**הגשות שלא יעדמו בדרישות לעיל יקבלו ציון 0 ללא אפשרות לבדיקה חזרת.**

בדקו היטב שהגשה שלכם עומדת בדרישות הבסיסיות הללו לפני הגשה עצמה.

שימוש לב Ci באתר מופיע `script` לבדיקה עצמית לפני הגשה בשם `selfcheck`. תוכלו להשתמש בו על מנת לוודא כי ההגשה שלכם תקינה.

בתרגיל זה (כמו בתרגילים אחרים בקורס) **יבדקו העתקות**. אנא כתבו את הקוד שלכם בעצמכם.

בהצלחה! ☺