תורת הקומפילציה

תרגיל בית 1 – בניית מנתח לקסיקלי

מתרגל אחראי: גיא סודאי

ההגשה בזוגות

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

תיקונים לתרגיל יסומנו בצהוב, חובתכם להתעדכן בהם באמצעות קובץ התרגיל.

התרגיל ייבדק בבדיקה אוטומטית. **הקפידו למלא אחר ההוראות במדויק**. הבדיקה תתבצע על שרת הקורס csComp (csComp.cs.technion.ac.il), לכן עליכם לוודא שהתרגיל שלכם פועל על השרת ובהתאם להנחיות.

הנחיות כלליות

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

הגדרות מושגים כלליים

- רווח לבן (whitespace) – אחד מבין: רווח (ספייס), טאב (התו **\t**), CR (התו **\r**), LF (התו **\n**).
- תווים ניתנים להדפסה (printable characters) – התווים שערך ה-ASCII שלהם בין **0x20** ל-**0x7E** (כולל את הקצוות), או רווחים לבנים:
  - ניתן לקרוא על תווים ניתנים להדפסה בהרחבה בוויקיפדיה בערך הבא: https://en.wikipedia.org/wiki/ASCII#Printable_characters
- רצף בריחה (escape sequence) – לוכסן אחורי (התו \) ואחריו תו או יותר שביחד מפורשים כתו אחד.
  - דוגמאות: **\n** – ירידת שורה, **\t** – טאב.
  - ניתן לקרוא על רצפי בריחה בהרחבה בוויקיפדיה בערך הבא: https://en.wikipedia.org/wiki/Escape_sequences_in_C

הגדרת אסימונים

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

COMMA	פסיק	,	,	.
LPAREN	סוגר שמאלי	(	(	[
RPAREN	סוגר ימני	)	)	]
LBRACE	סוגר מסולסל שמאלי	{	{	<
RBRACE	סוגר מסולסל ימני	}	}	>
LBRACK	סוגר מרובע שמאלי	[	[	{
RBRACK	סוגר מרובע ימני	]	]	}
ASSIGN	אופרטור השמה	=	=	==
RELOP	אופרטור רלציוני	== != <= >=	== != <= >=	<__> <>
BINOP	אופרטור בינארי	+ - * /	+ - * /	? :
COMMENT	הערת שורה	מתחילה ב-// שמופיע מחוץ למחרוזת, ואחרי שני הלוכסנים יכול לבוא כל תו מלבד ירידת שורה: LF, CR או CRLF.	// my comment	/* my comment */
ID	מזהה (Identifier)	צריך לעמוד בכללים הבאים: - יכול להכיל אותיות אנגליות קטנות וגדולות ומספרים בלבד. - על המזהה להתחיל עם אות אנגלית (קטנה או גדולה). - על המזהה להכיל תו אחד לפחות.	x max OO7	12AB 42 big_x
NUM	מספר שלם	צריך לעמוד בכללים הבאים: - אפסים מובילים אסורים (ראה דוגמא אסורה). - על המספר להכיל תו אחד לפחות.	0 102	050 5.6
NUM_B	מספר שלם עם B עוקב	צריך לעמוד בכללים הבאים: - אפסים מובילים אסורים. - על המספר להכיל תו אחד לפחות. על המספר להסתיים עם תו b.	0b 102b	050b 5.6b 50 50 b

STRING	מחרוזת	אוסף תווים בתוך מרכאות כפולות. הערות: 1. אורך המחרוזת יכול להיות בגודל אפס או יותר. 2. ניתן להניח כי אורך המחרוזת בלי המרכאות לא עולה על **1024** תווים. 3. ניתן לכלול כל תו הניתן להדפסה **פרט** לתווים הבאים: a. לוכסן אחורי: **** b. מרכאות כפולות: **"** c. תו LF: **\n** (כאשר הוא מגיע כתו בודד) d. תו CR: **\r** (כאשר הוא מגיע כתו בודד) אלא אם כן הם מגיעים כחלק מרצף בריחה תקין. רשימת רציפי בריחה תקינים: a. **** b. **\"** c. **\n** d. **\r** e. **\t** f. **\0** g. **\xDD** כאשר DD מייצג ספרה הקסדצימלית אופן הטיפול ברצפי בריחה יוסבר בהמשך, בחלק של הדפסת הלקסמות. שימו לב: כל רצף בריחה שאינו ברשימה הנ"ל מהווה קלט לא חוקי.	"simple" "also 'simple'" "escape new lines\n" "hex2 \x3A" "hi\thow\tare\tyou"	'unmatching" "unclosed "2-lined String" "ba-"-d" "bad \ escape" "hex \x10" "hex2 \x02"

הוראות התרגיל

עליכם לכתוב תכנית שתממש מנתח ותכתב בקובץ בשם **main.cpp**.

בתכנית זו תשתמשו בפונקציה **yylex()** שנוצרת ע"י flex ועליה לעמוד בדרישות הבאות:

1. המנתח יתעלם מכל הרווחים הלבנים, חוץ מבתוך מחרוזת.
2. כאשר המנתח מזהה אסימון, יש לפלוט מידע על האסימון על ידי קריאה לפונקציה printToken(lineno, token, value) המסופקת לכם בקבצי **output.hpp/.cpp**.

כאשר:

- lineno – מספר השורה בה האסימון **מסתיים**.
- token – מזהה אסימון מהקובץ **token.hpp** המסופק לכם (לפי השמות בחלק "הגדרת אסימונים" למעלה)
- value – ערך האסימון שזוהה, כלומר הלקסמה, פרט למקרה של הערות ומחרוזות, כמוסבר להלן.
- ניתן להניח שכל הערכים המספריים בתרגיל ניתנים לאחסון על ידי הטיפוס **int**.

למשל, עבור אסימון **ID** אשר זוהה עבור לקסמה **x** בשורה **12**, יש לבצע את הקריאה הבאה:

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

הדפסת הלקסמה של מחרוזות

מחרוזות יודפסו ללא המרכאות הכפולות המקיפות אותן.

נטפל ברצפי הבריחה באופן הבא:

- **\t**, **\r**, **\n** מוחלפים בסוג המתאים של רווח לבן (טאב, CR, LF)
- **** מוחלפת בלוכסן אחורי יחיד (\)
- **\"** מוחלפת במרכאות כפולות (")
- רצף בריחה של תו ASCII (**\xDD**) מוחלף בתו בעל ערך ה-ASCII אשר מייצג את הרצף ההקסדצימלי. כך למשל, עבור הרצף **\x41** יודפס התו **A**. אם הרצף מהווה ייצוג הקסדצימלי של תו אשר לא ניתן להדפסה, יש להדפיס שגיאה מתאימה (ראה סעיף טיפול בשגיאות).
  - הרצף ההקסדצימלי יכול להיות מורכב גם מאותיות קטנות וגם מאותיות גדולות. כלומר, **\x6A** ו-**\x6a** שקולים.
  - דוגמה – הקלט הבא:

```
"Hello \x57orld!\r\nThis\tis\t\x63oo\x6C, as always."
```

תודפס באופן הבא:

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

הדפסת הלקסמה של הערות

ניתן להעביר כל ערך עבור הפרמטר value, פונקציית ההדפסה תתעלם ממנו במקרה של הערה ותדפיס **//**.

קלט ופלט המנתח

המנתח יקבל את הקלט מהערוץ הסטנדרטי לקלט (stdin). flex משתמש בערוץ הקלט הסטנדרתי כברירת מחדל.

יש להיעזר בקבצי output.hpp/.cpp המצורפים לתרגיל על מנת לייצר פלט הניתן לבדיקה אוטומטית.

ניתן קבצי פלט לדוגמא. יש לבדוק כי המנתח שלכם פולט פלט זהה אליהם. הבדלים יגרמו לכישלון הבדיקות האוטומטיות.

טיפול בשגיאות

ניתן להניח כי הקלט מכיל שגיאה אחת לכל היותר.

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

המנתח נתקל בתו לא חוקי — `errorUnknownChar(character)`

שורה מסתיימת באמצע מחרוזת — `errorUnclosedString()`

מחרוזת מכילה רצף בריחה שלא מופיע בהגדרת התרגיל. למשל, עבור המחרוזת אשר מכילה את הרצף **\q** הפרמטר sequence יהיה **q**. — `errorUndefEscape(sequence)`

עבור מקרה בו הרצף **\x** מלווה בתווים שאינם מייצגים ערך הקסדצימלי או ערך הקסדצימלי מייצג תו אשר לא ניתן להדפסה או שהמחרוזת נגמרת לפני שניתן לקרוא 2 תווים לאחר ה-x (למשל עבור המחרוזת "hey \xF") או שהתו המיוצג לא ניתן להדפסה (למשל עבור המחרוזת "\xFF"), הודעת השגיאה תכיל את רצף הבריחה המלא.
כך, עבור הרצף **\xFF** ה-sequence יהיה **xFF**.

הערות נוספות על התרגיל

- בתרגיל זה תדרשו לכתוב קובץ **lex** יחיד בשם **scanner.lex**. שימרו עליו פשוט, וממשו את הלוגיקה הרצויה בקבצי ה-cpp.
- באופן דיפולטי, הפונקציה **yylex()** מחזירה טיפוס **int**, וחוזרת למשתמש כאשר קיימת פקודת **return** ב-action של האסימון. (ראו שקף 23 בתרגול על המנתח הלקסיקלי).
- לתרגיל מצורף קובץ בשם **tokens.hpp** המכיל טיפוס enum הכולל בתוכו את כל האסימונים.
  ביצוע include לקובץ זה הן בקובץ **scanner.lex** והן בקבצי ה- **cpp** מאפשר "תקשורת" בין המנתח ש-flex יוצר לבין התכנית שתכתבו. כלומר, התכנית שתכתבו תדע להבין אילו אסימונים המנתח מחזיר.
  לדוגמא, נניח כי יש לנו אסימון בשם **FOR**, לכן נוכל לכתוב בקובץ ה-**scanner.lex** ב-rules section:
  ```
  for   return FOR
  ```
  ואילו בקובץ ה- **cpp**:
  ```
  if (yylex() == FOR) {...}
  ```
- בנוסף, קובץ ה-**tokens.hpp** מכיל הגדרות שיאפשרו לכם להשתמש בפונקציה **yylex()** ובמשתנים **yylineno**, **yytext**, **yyleng**.
- לתרגיל מצורף קובץ טמפלייט **main.cpp** המכיל את לולאת הקריאה ל-**yylex()**. העזרו בו.
- מומלץ להיוועץ ב-manual של flex לצורך ביצוע התרגיל. קל יותר לבצע אותו על ידי שימוש ביכולות מתקדמות של flex שלא נלמדו בתרגולים כגון regex patterns מתקדמים ו-debug mode.
- **טיפ**: תוכלו להשתמש באתר https://regex101.com שעוזר בהבנה ובבנייה של תבניות regex מורכבות.
- **טיפ**: כעקרון, לא תבדקו על דליפות זיכרון, איכות קוד, וכדומה. ועדיין, מומלץ לבדוק עם valgrind, לקמפל עם -Wall -Wextra -Wmissing-declarations, ולשנות את הקוד כדי לצמצם דליפות ואזהרות.

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 מכוונים את ה-Encoding של הקובץ ל-ASCII או ANSI, או מבצעים save as כ-ASCII.

לנוחותכם, וכדי למנוע בעיות בהעתקה בין קבצים, להלן מפתח של התווים המוזכרים בתרגיל וערכי ה-ASCII שלהם:

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

קבצי הטסט זמינים בקובץ zip ומומלץ תמיד להוריד ולהעביר אותם כ-zip על מנת למנוע שינוי אוטומטי של ירידות השורה על ידי תוכנות להעברת קבצים.

הוראות הגשה

מסופק לכם קובץ Makefile שאיתו תקומפל ההגשה שלכם. שימו לב כי קובץ ה-Makefile מאפשר שימוש ב-STL. אין לשנות את ה-Makefile.

יש להגיש קובץ אחד בשם ID1-ID2.zip, עם מספרי ת"ז של שתי המגישות. 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 יכיל** את:

- קובץ ההרצה.
- קבצי הפלט של flex.
- קובץ Makefile שסופק כחלק מהתרגיל.

יש לוודא כי בביצוע unzip לא נוצרת תיקיה 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 פורמט הפלט זהה לפורמט הפלט של הדוגמאות הנתונות. כלומר, 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
```

ייצור את קובץ ההרצה בתיקיה הנוכחית ללא שגיאות קומפילציה, יריץ אותו, ו-diff יחזיר 0.

הגשות שלא יעמדו בדרישות לעיל יקבלו ציון 0 ללא אפשרות לבדיקה חוזרת.

בדקו היטב שההגשה שלכן עומדת בדרישות הבסיסיות הללו לפני ההגשה עצמה.

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

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

בהצלחה! ☺