תכנות מתקדם

תרגיל 2

סורק - scanner

מועד אחרון להגשה: 30.4.2021

בתרגיל זה נבנה scanner שמחזיר tokens ומבצע cross reference.

- ההגשה ביחיד לכתובת:

http://submit.org.il/ariel

לבחור קורס cpp38510

- **צריך להעלות שלושה קבצים:**

`token.cpp , symtab.cpp , scanner.cpp`

(בדפדפן, אפשר לבחור את שלושתם)

הקבצים צריכים לממש את המחלקות שבקבצי הכותרת.

שאר הקבצים כבר נמצאים בשרת הבדיקות ואי אפשר להעלותם.

הקובץ **`C.l`** הוא קובץ **`lex`** שנועד להגדיר את דרישות התרגיל.

כדי לפשט השמטנו וצמצמנו חלק מההגדרות.

(הערה: העלאת קוד שמדפיס את הפלט תדווח לוועדת משמעת)

תאור ה- scanner

ה- scanner מבוסס על שלוש מחלקות:

1. Token (מילה)

המחלקה מכילה את סוג המילה והטכסט שלה.

סוג המילה הוא enum שמוגדר בקובץ tokentype.h.

מתודות של המחלקה מחזירות את הסוג והמילה.

המתודה printToken() מדפיסה את ה- Token בצורה הבאה:

cout << סוג המילה << טאב << טכסט המילה << endl;

לדוגמה:

INT int

IDENTIFIER counter

= =

CONSTANT 10

; ;

ניתן להיעזר בקובץ tokenstrings.h להדפסת ה- enum.

(ראו תשובה 30 https://stackoverflow.com/questions/3342726/c-print-out-enum-value-as-text)

varToken

היא מחלקה יורשת של **Token** ומשמשת עבור מילה שהיא משתנה.

במילה שהיא משתנה שומרים ב- set את מספרי השורות בהן היא הופיעה.

ישנן מתודות להוספת מספר שורה ולהחזרת מספרי השורות.

2. SymbolTable (טבלת סמלים)

המחלקה מכילה map מטכסט של מילה ל- Token של אותה מילה.

המפה מכילה רק משתנים ומילים שמורות.

המפה מאותחלת עם המילים השמורות (מצורף קובץ reserced.txt שיכול לשמש לאתחול הטבלה).

כשה- scanner עובר על קובץ הקלט, אם הוא פוגש במילה מסוג משתנה הוא מחפש (lookupToken) את המילה בטבלת הסמלים.

אם המילה לא נמצאת אזי הוא מוסיף אותה (insertToken).

אם המילה היא מסוג משתנה, בין אם המילה כבר היתה ובין אם הוסיף אותה, הוא מוסיף את מספר השורה באמצעות add_line() של Token.

xref() מדפיס את המשתנים שבטבלת הסמלים בצורה הבאה:

cout << טכסט של המשתנה << טאב << מספר שורה << רווח << מספר שורה << רווח << … << endl;

(אחרי כל מספר שורה, כולל האחרון, יש רווח)

אם זו מילה שמורה הוא לא מדפיס.

3. **Scanner (סורק)**

המחלקה מכילה את קובץ הקלט, טבלת הסמלים, מספר השורה הנוכחית והתוו הבא.

המתודה nextToken() מממשת את הסורק.

המתודה nextChar() קוראת את התוו הבא.

מבנה המתודה nextToken():

קרא תוו.

- אם התוו הוא רווח או התחלה של הערה, דלג על כל תווי הרווח **וההערות** וקרא תוו.
- אם התוו הוא מילה שאינה התחלה של מילה אחרת:

```
switch (ch) {    // each character represents itself
case ';' : case '{' : case '}' : case ',' : case ':' : \
case '(' : case ')' : case '[' : case ']' : case '~' : \
case '*' : case '%' : case '^' : case '?' : case '/':
```

```
        return shared_ptr<Token>
          (new Token(static_cast<tokenType>(ch),string(1,ch)));
      break;
```

- אם התוו יכול להיות מילה או התחלה של מילה (ראו קובץ `C.l`), צריך לקרוא את התוו הבא.

  אם לא היה צורך בתוו הבא, אפשר להחזיר אותו עם הפקודה:

```
      inputFile.unget();
```

- אם התוו שקראנו הוא התחלה של מספר אזי צריך לקרוא תווים עד לתוו שאינו אחד מהתווים ספרה, נקודה, `E` או `e`.

  ואז לבדוק אם המחרוזת שהתקבלה מתאימה לאחד מהביטויים הרגולריים של מספר (ראו קובץ `C.l`).

  אם כן אז להחזיר מילה מסוג `CONSTANT` אחרת מילה מסוג `ERROR`.

- אם התוו שקראנו הוא התחלה של משתנה אזי צריך לקרוא תווים עד לתוו שאינו תוו של משתנה.

  כעת צריך לחפש את המחרוזת בטבלת הסמלים.

  אם היא נמצאת לבדוק אם היא מסוג משתנה או לא.

    אם היא לא מסוג משתנה, להחזיר את המילה (השמורה).

    אם היא מסוג משתנה להוסיף את מספר השורה הנוכחית.

  אם היא לא נמצאת צריך להוסיף אותה לטבלת הסמלים ולהוסיף את מספר השורה הנוכחית.

- אם התוו שקראנו הוא ' (גרש), אם כן זו הגדרה של char.

  להחזיר את התוו הבא אם התוו שאחרי הבא הוא גרש, אחרת להחזיר שגיאה.

- אם התוו שקראנו הוא " (גרשיים), אם כן זו הגדרה של מחרוזת.

  לקרוא תווים עד לתוו שאינו " (גרשיים) ולהחזיר את המחרוזת שביניהם (ללא הגרשיים)

  (בגרש וגרשיים פשטנו את ההגדרה, ראו קובץ `C.l`).

בהצלחה