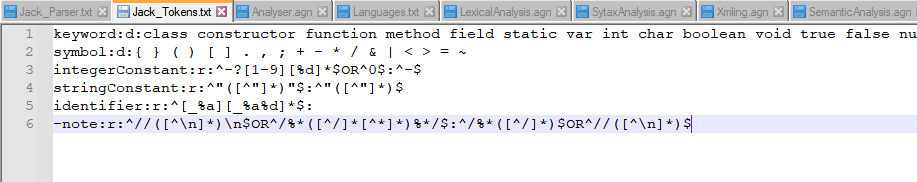
מדריך קבצים – נריה מחפוד

**קובץ טוקנים**



כל שורה מכילה טוקן שונה, שמורכבת מארבעה חלקים שמופרדים ב ':'.

* **שם הטוקן.**

אם שם הטוקן מתחיל ב '-', הקומפיילר פשוט מתעלם מהטוקן (כי זו הערה).

* **הדרך לקרוא את הטוקן.**

שתי אפשרויות:

* + d – dictionary – מילון. אוסף סופי של מילים שאם המחרוזת מרכיבה אחת מהן אז הטוקן הוא אמת.
  + r – regular expression – ביטוי רגולרי. בעזרת הפונקציה strings.match ניתן לבדוק אם המחרוזת מתאימה לביטוי.
* **תכולת הטוקן.**

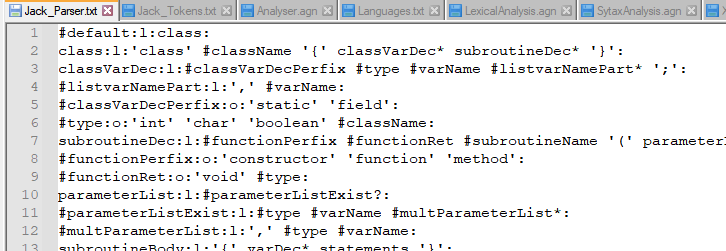
הביטוי\המילון.

* **אופציונלי.**

כדי לדעת מתי צריך לעצור את בדיקת הטוקנים, צריך לדעת באיזה שלב אין מצב זה חלק מהטוקן.

* + עבור מילון זה לא קיים. המנתח פשוט יודע לבדוק אם המחרוזת היא תחילית של אחת המילים במילון.
  + עבור ביטוי רגולרי זה ביטוי רגולרי שמבטא את ההתחלה של הטוקן.

**קובץ האוטומט**



כל שורה מכילה מצב באוטומט שבונה את עץ הטוקנים. כל שורה מורכבת בארבע חלקים:

* **שם המצב.**

אם שם המצב מתחיל ב '#', הממיר של הקובץ לXML מתעלם מהעיגול הזה בעץ ועובר ישר לילדים שלו.

* **הדרך לקרוא את המצב.**

שלוש אפשרויות:

* + l – list – רשימה. רשימה של מילים\מצבים (מופרדים ברווח) שכל אחד מהם חייב להיות ילד של המצב הנוכחי בעץ הטוקנים.

אם המילה מוקפת ב ', אז זה לא מצביע למצב אחר, אלא פשוט המילה. היינו ילד בעץ הטוקנים, שמתאים לאחד מהמצבים המילוניים.

אם סוף המילה הוא ? אז הקומפיילר בודק אם הטוקן הנוכחי מתאים להתחיל את המילה\המצב שהמילה מפנה אליו. אם כן, אז הוא יוסיף למצב את המילה. אם לא הוא ימשיך הלאה.

אם סוף המילה זה \*, אז הקומפיילר יבצע את אותה הבדיקה, שוב ושוב עד שהטוקן לא מתאים.

* + o – or – אפשרויות. רשימה של מילים, שאחת מהן היא הילד של המצב הנוכחי.
  + t – token – טוקן. פשוט אומר שהמצב הוא טוקן, וזה עלה בעץ הטוקנים.
* **תכולת המצב.**

פשוט הרשימה הנזכרת לעיל.

* **ריק.**

חייבים שיהיה מצב default שמצביע למצב הראשון – שורש העץ. משם הקומפיילר מתחיל לבנות את העץ.

**קובץ המעבר**

זה הקובץ הכי מסובך. הוא לא בנוי נכון ויש הרבה עקרונות שמופרים, וכן הוא מסורבל ומבאס, אבל הוא עובד מעולה. וכל פעם שצריך עוד משהו יחסית קל להוסיף אותו (בגבולות הסביר, וצריך להימלט משגיאות, אבל אחרי שהמערכת קיימת כבר אפשר להבין מה לא עובד בה).

אז ככה: Graphical user interface, text, application

Description automatically generated

לקובץ הזה יש 226 שורות, וזה חלוקתם:

יש שלושה סוגים של קידומות:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **קידומת** | **הסבר** | | | | | | | |
| **פשוט** | **תוכן** | | | **הסבר** | | **טעינה** | **פריקה** |
| **h** | שורה בheader. ההדר הוא טבלה שעוזרת לי לנהל את העניינים. | Source | | | מקור. מחולק ברווח בין שם השפה לסיומת השפה. | |  |  |
| Destination | | | מטרה. כנ"ל. | |  |  |
| Dictionary | | | המילון. מטרתו לשמור כל מיני נתונים לאורך התוכנית. לדוגמא, מה שם המחלקה (חשוב לדעת בשביל קריאה לפונקציות), או this כדי לדעת כמה שדות יש למחלקה, ו-Label, שצריך לכור באיזה לייבל אנחנו שלא יהיו קפיצות לא הגיוניות. | | ad | %  %'ClassName  זה גישה לשם המחלקה וכו'. |
| Current | | | טבלת הנוכחיים. זה מתפקד בדיוק כמו המילון, אבל לתפקידים אחרים, יותר משתנים. | | ac | &  כנ"ל. |
| Temp | | | זמני. זה חשוב. זה מאפשר לנו תקשורת לרוב העץ בין האחים, בלי שהבנים יוכלו לשנות אותו. זה חשוב מלולאות מקוננות וכד', כשרוצים לשמור את המספר של הלייבל. | | at | $ |
| Arg | | | ארגומנט. זה מאפשר לנו לשלוח משתנה בגובה העץ. | | aa | ; |
| Scoper | | | טבלת הסמלים של הסקופר, היינו של הפונקציות. | | as |  |
| SymbolTable | | | טבלת הסמלים | | i |  |
| Define | | | כל המילים שפשוט צריך להחליף וזה יעשה לנו חיים קלים פי כמה (בפועל יש לנו רק אחד כזה, אבל בגדול יכול להיות יותר). הפונקציה שמפענחת את ערך הטוקנים פשוט מחליפה את המילה אם היא נמצאת בטבלה הזו. | | def |  |
| **s** | שורה מנחת טוקן.  העניין הוא שצריך לדעת מה עושים עבור כל טוקן שאנחנו מקבלים. אז אחרי שעברנו את כל הכותרות, עוברים על שאר הקובץ, וכל שורה שמסומנת כך, סימן שכאשר אנחנו מקבלים את הטוקן שלה הפקודות עד הs הבא שייכות אליה. שורות s מחולקות ל3. הs, שם הטוקן, והדרך לקרוא את רצף הפקודות. | | | l | | List: רשימה פשוטה. פשוט צריך לעבור על הפקודות ולבצע כל אחת ואחת מהן. | | |
| ol | | Or-List: כאשר לטוקן יכול להיות רק בן אחד, אבל אנחנו לא יודעים איזה בן זה, ועבור כל בן יש סט שונה של פקודות שצריך לבצע, אז כאן יש ol, שעובד ככה:  s:tok\_father:ol  tok\_san\_1:ad:'ClassName 'TokSan\_1  tok\_san\_1:c:'hello  tok\_san\_2:ad:'ClassName 'TokSan\_2 | | |
| val | | Value: פשוט הערך של הטוקן. לכותרת מהסוג הזה אין רשימה של בנים, אלא היא פשוט מחזירה את הערך של הטוקן. | | |
| f | | For: תמיד הערך של טוקן יהיה מחרוזת. אז f עובר על כל תו בערך של הטוקן ומבצע את שורת הפקודות. גישה לתו הנוכחי מתבצעת ע"י @. | | |
| o | | Or. לכותרת כזו יש בן יחיד, המסומן ב oo. זה סימן שלטוקן אפשר שיהיה רק בן אחד, וצריך לבצע עבורו את הבן הנ"ל. | | |
| st | | אין הבדל בין זה לval. במקור היה, אבל שיניתי את זה אז עכשיו פשוט יש שתי שמות לפקודת הגישה ל value. | | |
| **כל השאר** | כל השאר זה הפקודות שאפשר לתת. רובן מוצגות לעיל כגישה לטבלאות, אך כמה מהן לא: | c | Code: הפקודה הזו היא פשוט להוסיף תוכן הפקודה לשורה בקוד. התוכן מחולק ברווחים, כאשר לכל מילה יש כמה אפשרויות:  **מחרוזת**, מתחילה ב', וזה פשוט להוסיף את המילה לקוד.  **גישה לזיכרון,** מתחיל באחד הסימנים לעיל.  **טבלת סמלים**. כאן טעיתי בעיצוב, אבל הייתי כבר לחוץ בזמן וזה היה לקראת הסוף אז לא משנה. אבל ! משמעותו גישה לטבלת הסמלים, וזה מוסיף לקוד את הסוג (local, this וכד') ואת המספר.  **כל השאר**, אם זה לא אחד מאלה אז זה טוקן, וצריך להיכנס אליו ולבצע את מה שכתוב שם.  **?**, מופיע בסוף השורה. אם אחד הטוקנים שמוצגים לא קיים, אז לא מבצעים את השורה אלא את אופציה ב. | | | | | |
| c\* | Multy-Code: זה גישה חוזרת לטוקן, היינו לבצע טוקן כלשהו את כל הפעמים שהוא כתוב. | | | | | |
| cs | Counter-Set: זה עקרונית יכול להיכלל בגישה לזיכרון אבל תסלחו לי על זה. אחד מהארגומנטים של הפונקציה הרקורסיבית הוא קאונטר, וזה מאוד חשוב. כי אם נניח בהינתן טוקן מסוים, אנחנו רוצים לבצע חלק מהבן שלו בשלב אחד וחלק בשלב שני. אז אצל הבן לפני הפקודות מופיעים מספרים, היינו באיזה קריאה לבצע אותם.  אז בד"כ הספירה מתבצעת לבד, אבל לעמים אתה רוצה לבצע שונה מהספירה, ולכן אתה יכול לשים ערך בקאונטר. | | | | | |
| if | תנאי: זה מחולק לשלושה חלקים: תנאי, תוצאה, ואחרת. | | | | | |

יש עוד הרבה נקודות שאפשר לגעת בהן, אבל אני כבר צריך להגיש.