**תיאור מבנה כללי של מימוש הקומפיילר**

הקומפיילר שבנינו בנוי משני יחידות גדולות המורכבות מתתי יחידות.

* היחידה הראשונה היא המנתח הלקסיקלי שנוצר בעזרת הכלי FLEX מתוך הקובץ scanner.lex שלנו.
* היחידה השנייה היא המנתח התחבירי שבתוכה קורה גם ייצור קוד הRISKI. המנתח התחבירי נוצר בעזרת הכלי BISON מתוך קובץ הparser.ypp שלנו.
* הוספנו לכל כלל גזירה ניתוח סמנטי לפי הדרישות שמשתמש במחלקות עזר ובמבני הנתונים שכתבנו.
* בכל כלל גזירה ניהלנו זיכרון ורגיסטרים ורשומת הפעלה לפי ABI שהחלטנו.
* לכל כלל גזירה עשינו יצירת קוד RSK תוך BACKPATCHING

ביצענו אינטגרציה בין חלק 1 של הפרויקט בו בנינו קובץ LEX ליצירת המנתח הלקסיקלי לבין חלק 2 שבו הוספנו את המנתח התחבירי לתחביר CMM והוספנו לכל כלל גזירה את הניתוח הסמנטי ויצירת קוד RSK.

כעת נתאר את מבנה מימוש הקומפיילר, מחלקות עזר ומבני הנתונים ואיך אנחנו נעזרים בהם.

**תיאור מבני הנתונים**

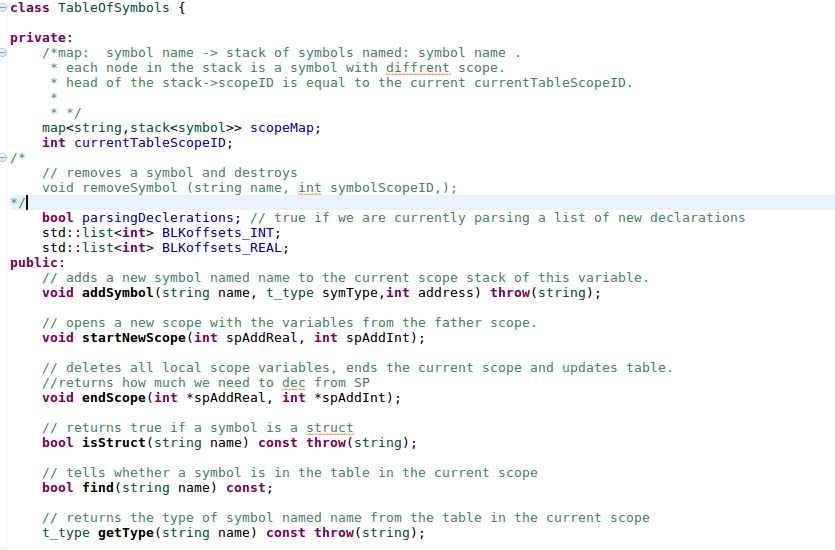
נתאר את עיקרי המימוש של מבני הנתונים ונפרט את הטיפול שלהם בנושאים שונים. במימוש מבני הנתונים השתמשנו במחלקות STL רבות.

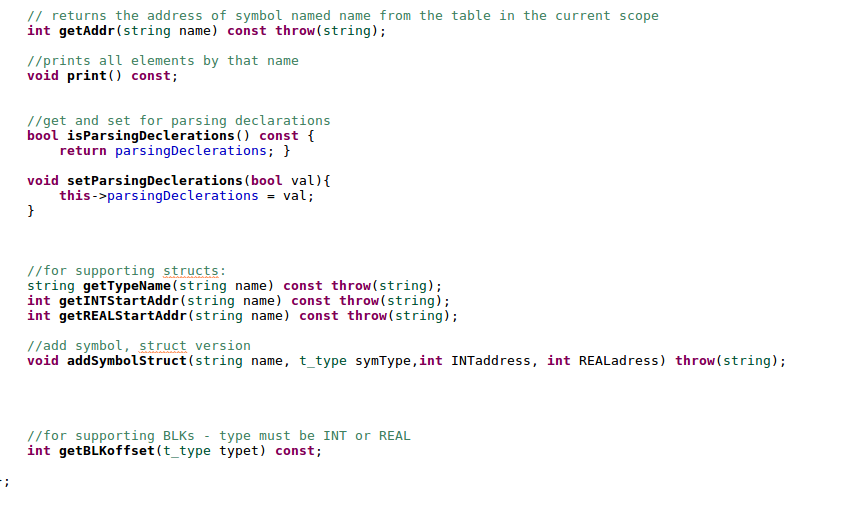
**מבני הנתונים:**

**TableOfSymbols**:

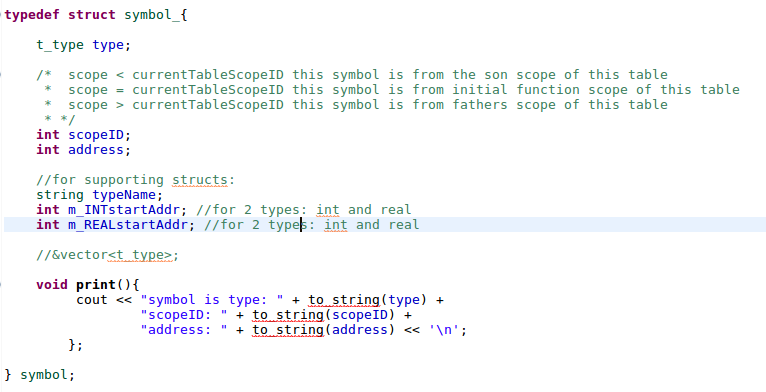
זהו מבנה הנתונים לניהול הסמלים בכל SCOPE כולל הסמלים בקפיצות לפונקציות.

הסמלים שמורים במבנה הבא: **map<string,stack<symbol>> scopeMap**;

כך שכל סמל מזוהה ע"י השם שלו והסמל הזה והמופעים השונים שלו בכל SCOPE מנוהלים במחסנית שיש לכל סמל.



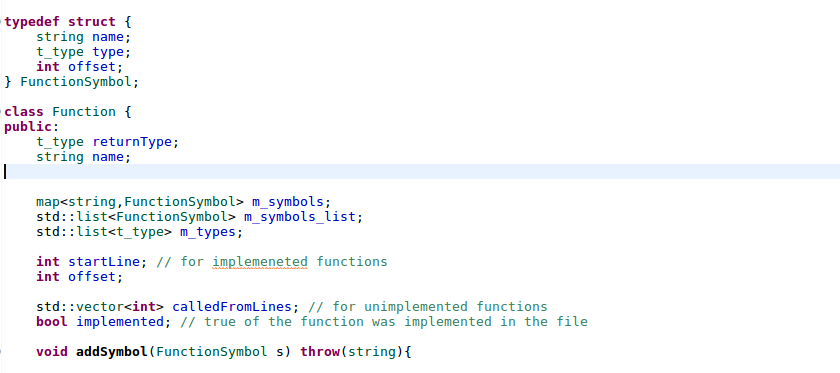
סמל בודד נראה כך:



ניהול הStatic scoping נעשה בעזרת מתודות **startNewScope** **endScope** והמשתנה **currentTableScopeID.** המשתנה מאותחל ל 0 ובMAIN נפתח SCOPE חדש והמשתנה יורד ל1- , כמו כן בכל תת בלוק המשתנה יורד בעוד יחידה. בהתאמה בסגירת SCOPE המשתנים המקומיים נמחקים והSCOPE עולה באחד.

בגלובלי לטובת הPARSER אנחנו שומרים מחסנית של אובייקטים**TableOfSymbols .**לכל פונקציה יש תבלה משלה.

**מבנה הנתונים Function:**

מבנה זה אחראי לנהל פונקציות עזר ואת הסמלים שלהם. לכל פונקציה בקוד המשתמש יש אובייקט של מבנה נתונים זה. מחלקה זאת עוזרת לנו לנהל את הקריאות לפונקציות ולשמור את המידע שלהם ושל הסמלים שלה. באופן כללי המחלקה שומרת את הנתונים הבאים: טיפוס החזרה של הפונקציה,שם הפונקציה, רשימת הסמלים שלה, רשימת הטיפוסים שלהם, מספר השורה במימוש האסמבלי. 

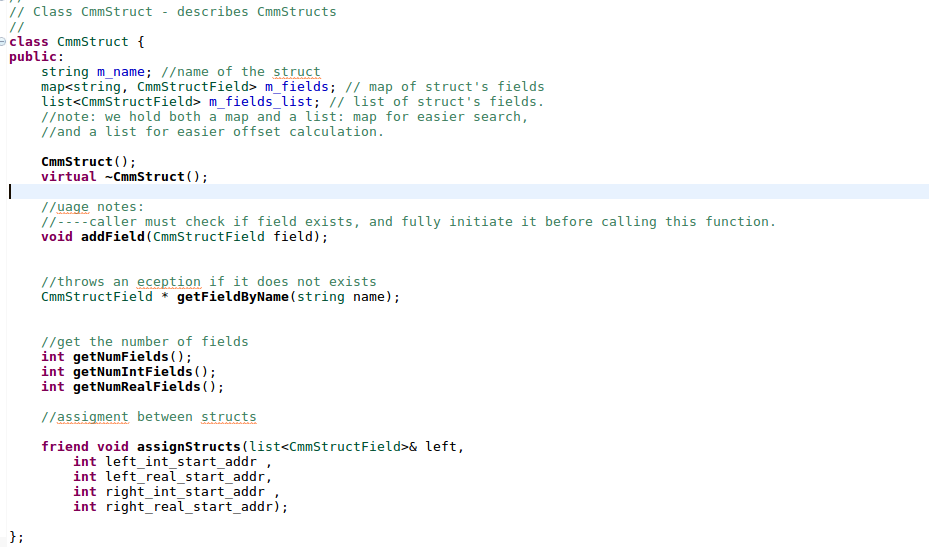
אנו משתמשים במחלקה הזאת ומחזיקים מילון פונקציות גלובלי לטובת הPARSER לפי שמם.

**מבנה נתונים CmmStruct**

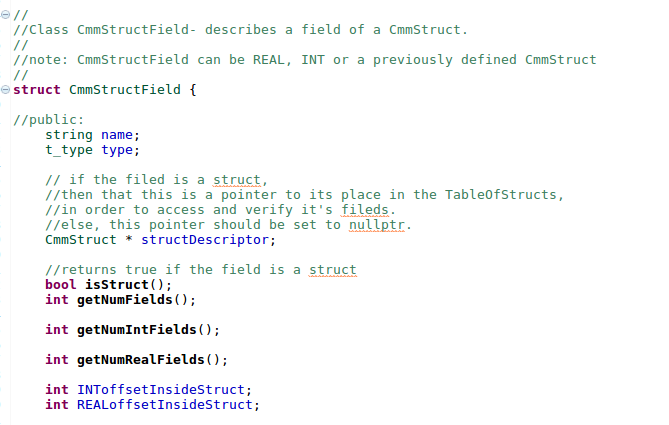
מבנה זה מתחזק את מבנה הנתונים של מבנים בשימוש במחלקת עזר cmmStructField

מתאר את השדות של המבנה ומחזיק את שמו והטיפוס של השדה כמו כן את המידעcmmStructField.

האם השדה הוא מבנה בעצמו. מבנה השדה מתחזק את הOFFSET של המשתנים שלו. **CmmStruct** מחזיק ומתחזק נתונים כמו השם שלו, מילון האש של השדות שלו שממפים שם של שדה לcmmStructField שלו. כמו כן מתודה חשובה של מבנה נתונים זה הוא אשמה בין מבנים.



שדה אחד מוחזק באמצעות המחלקה הבאה, הקטע העיקרי בה הוא הטיפול בשדה שגם הוא טיפוס מורכב.



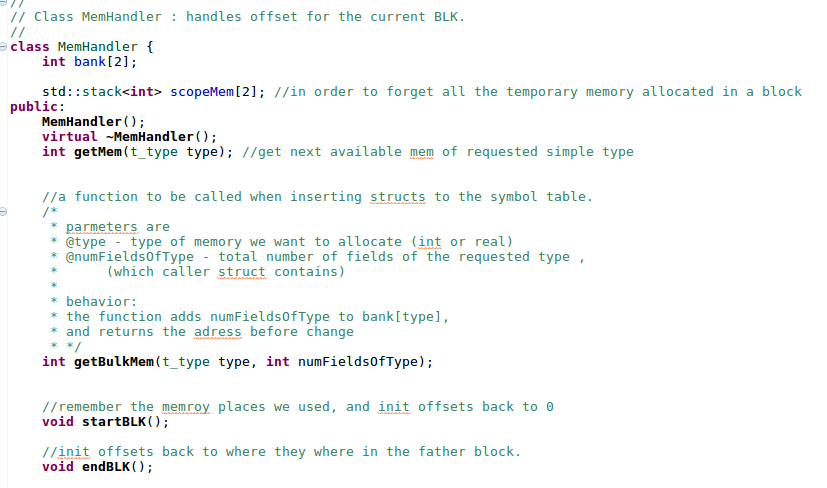
גלובלית לטובת הPARSER אנחנו מחזיקים מבנה נתונים של מילון האש שממפה את שם המבנה לאובייקט **CmmStruct** שלו.

**תיאור BACKPATCHING**

**המודלים השונים בקוד**

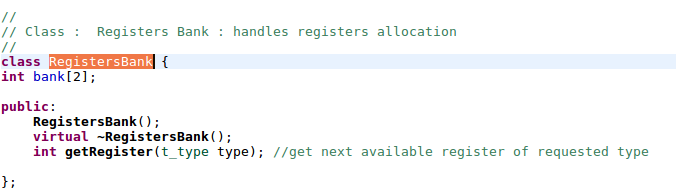
**memHandler**

אנו משתמשים בו לניהול הזיכרון. מבנה המחלקה:



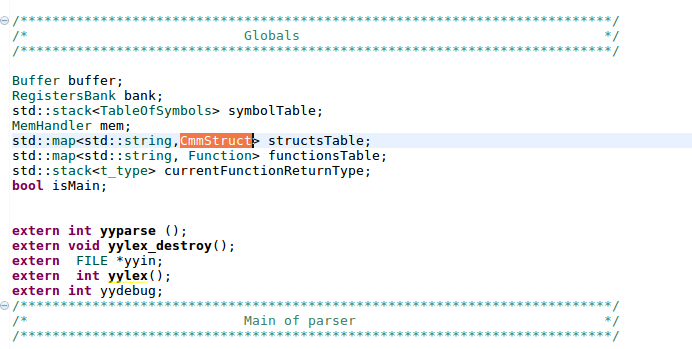
**RegistersBank**

אנו משתמשים בו לניהול הרגיסטרים

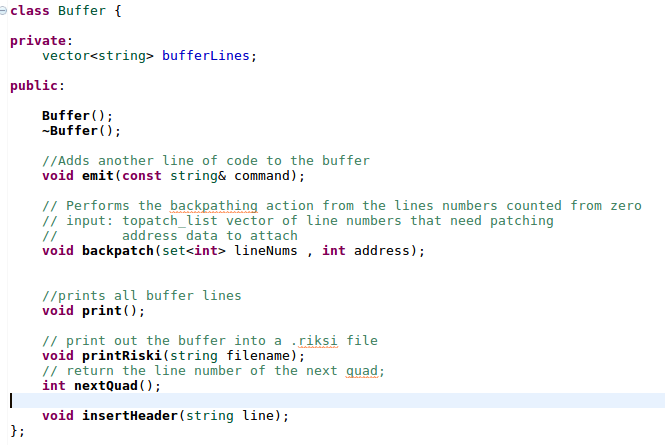


**Common**

מחזיק את כל המשתנים ומבני הנתונים הגלובליים שמשותפים למספר קבצים



**Buffer**

שמו מתאר טוב את תפקידו כבאפר ששומר את הפקודות מכונה שלנו ובעזרתו מיישמים את שיטת ה EMIT. כמו כן יש לנו מתודה שמדפיסה אותו לתוך קובץ .RSK . 

**תיאור כללי של רשומת ההפעלה- ABI**

**אופן הקצאת רגיסטרים מיוחדים**

**הסבר כללי על מימוש טיפוסים מורכבים והשמה ביניהם**