מכללת כפר סבא

תהליכי

דוח פרוייקט - אביב אש

סמל מוסד: 470112

מכללה: מכללת Ort design כפר סבא

שם הסטודנט: אביב אש

תז הסטודנט: 214887556

שם הפרויקט: tec – The Easy Compiler

מבנה / ארכיטקטורה של הפרויקט

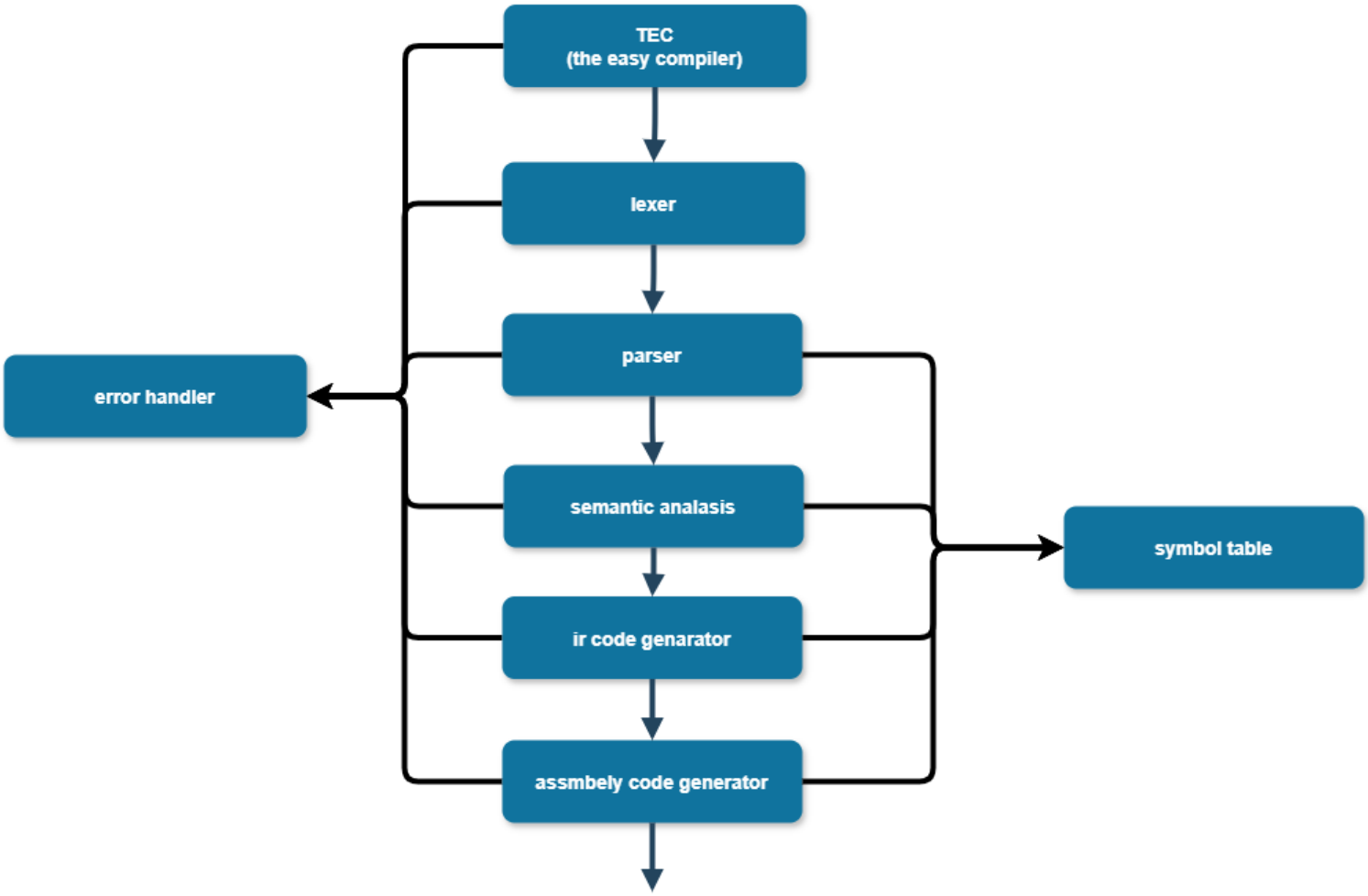
הבעיה:

מהדר הינו רכיב תוכנה בעל שלל תהליכים פנימיים מורכבים.

הפתרון:

נפרק את הרכיב הגדול לשלל רכיבים קטנים בצורה שלכל רכיב יהיה פונקציונליות ותפקיד משל עצמו, כך הקוד יהיה יותר אחיד,רובסטי וקריא יותר.

להלן תרשים זרימה המתאר את החלקים השונים ואת התקשורת ביניהם:



היחידות בפרויקט:

המנתח הלקסיקלי (Lexical Analysis)

קלט היחידה: קוד המקור

פלט היחידה: רשימת אסימונים **למנתח התחבירי**, רשימת שגיאות אל מטפל **השגיאות**

השלב הראשון בfront end עוסק בחילוק הקוד הנתון ל**טוקנים**, אשר כל טוקן הוא יחידה בעלת משמעות כמו מילים שמורות, משתנים, מספרים, אופרטורים וכדומה.

לדוגמה, פיסת הקוד int x = 7; תוכל להיות מתורגמת לטוקנים

- Int (מילה שמורה)
- x (מזהה)
- = (אופרטור)
- 7 (מספר)
- ; (נקודה פסיק)

המנתח הלקסיקלי בשפת easy:

בחרתי לייצג את המנתח הלקסיקלי בשפתי בעזרת **אוטומט סופי דטרמניסטי** (DFA)

מצב התחלתי (מסומן ב-S) – מסמל את המצב הראשוני לסריקת אסימון לפני קבלת כל אות

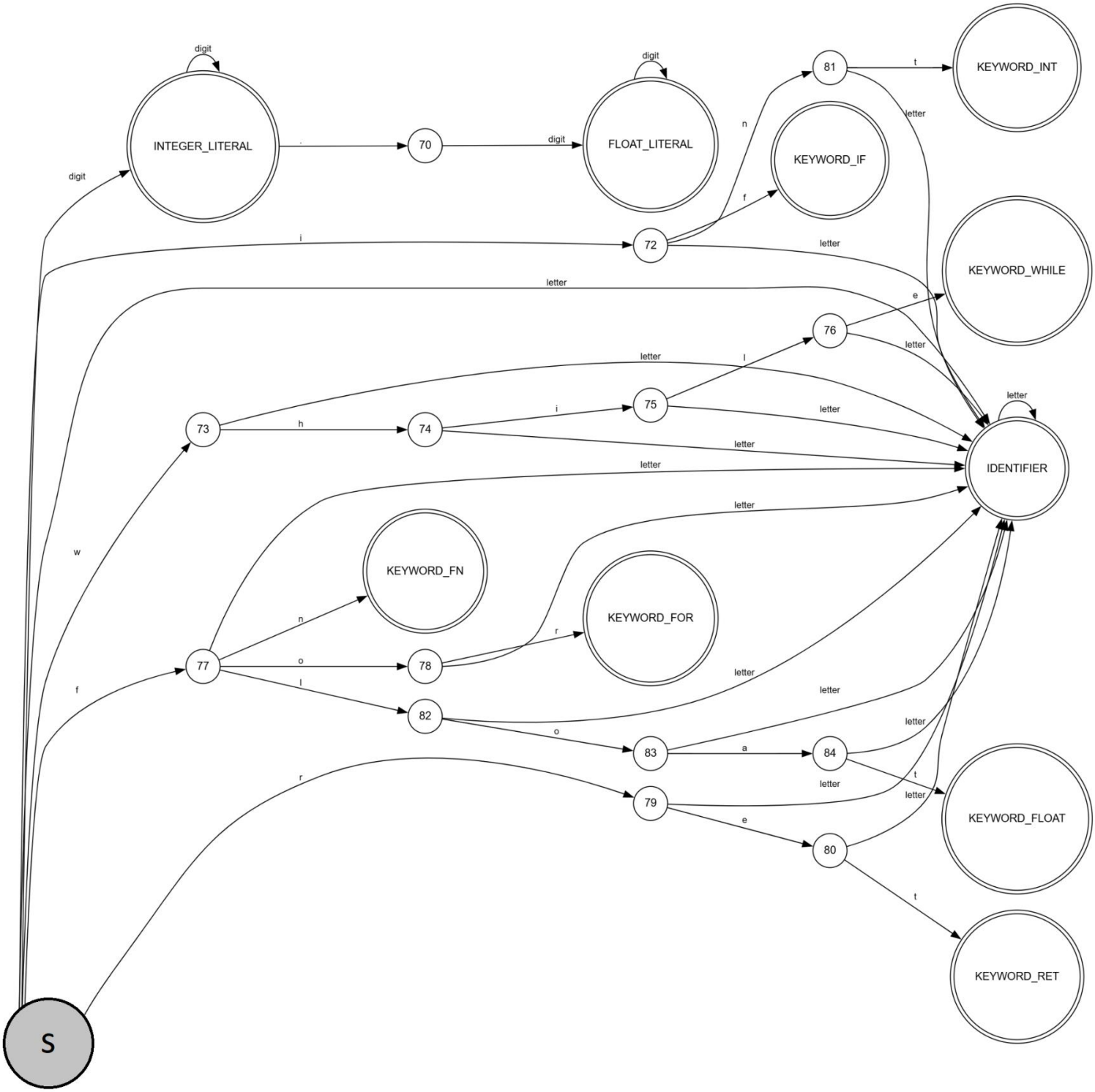
מצבי ביניים (מסומנים ב**מספרים**) – מסמלים מעבריי ביניים באוטומט. סיום הסריקה בהם תסתמן כשגיאה במהדר

מצבי סיום (מסומנים בעיגול כפול) – מסמלים מצביי קבלה/סיום של האוטומט. סיום הסריקה בהם תסתמן כאסימן תקין

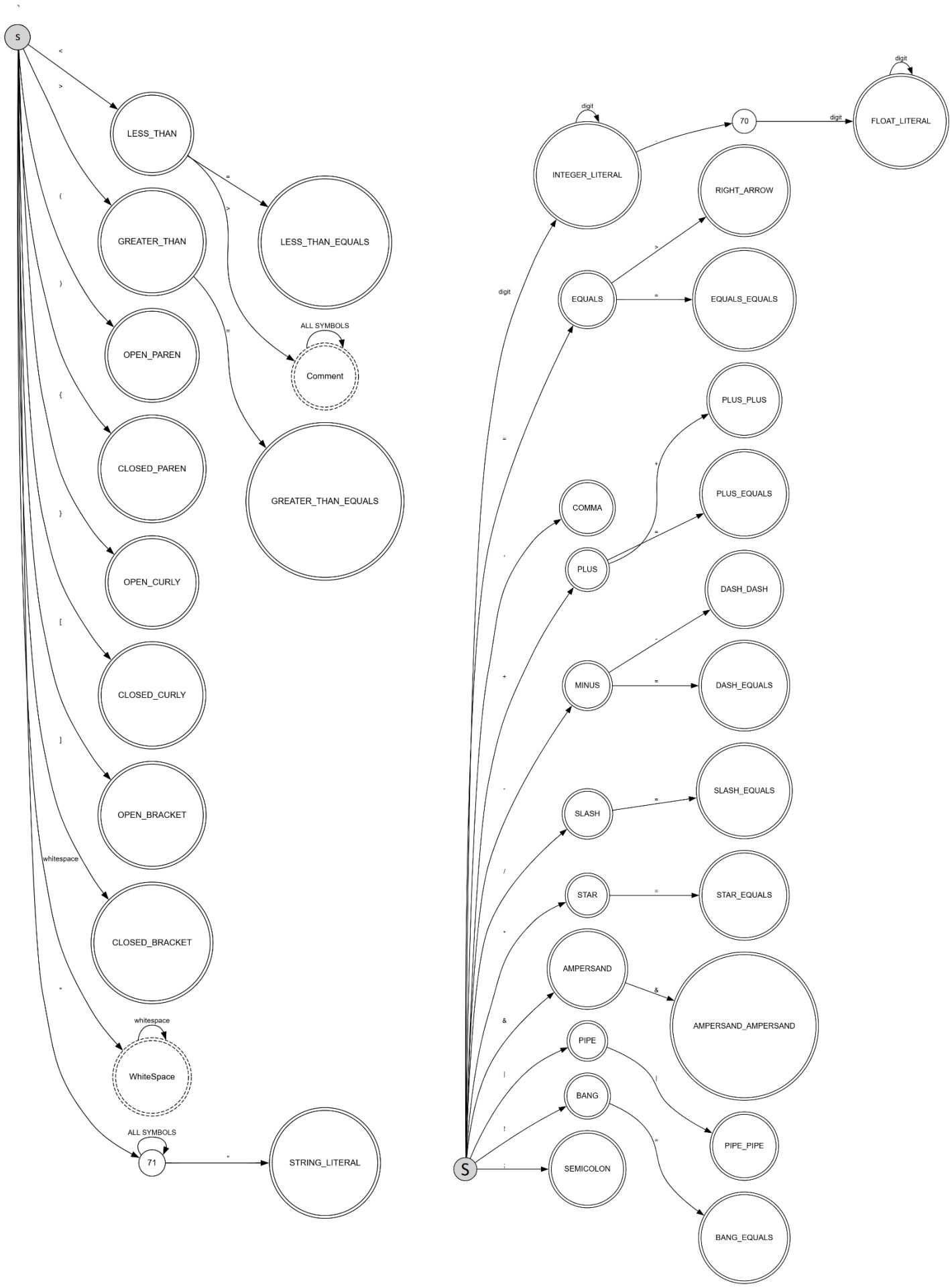
מצבי דילוג (מסומנים בעיגול מקווקו) – מסמלים מצביי דילוג של האוטומט. סיום הסריקה בהם לא תחשב ככלום במהדר

להלן סרטוט האוטומט:

חלק האוטומט הקולט מזהים (שמות משתנים ופונקציות),מחרוזות,מספרים שלמים, מספרים עשרוניים ומילות מפתח:



חלק האוטומט הקולט את שלל סימני השפה, רווחים והערות:



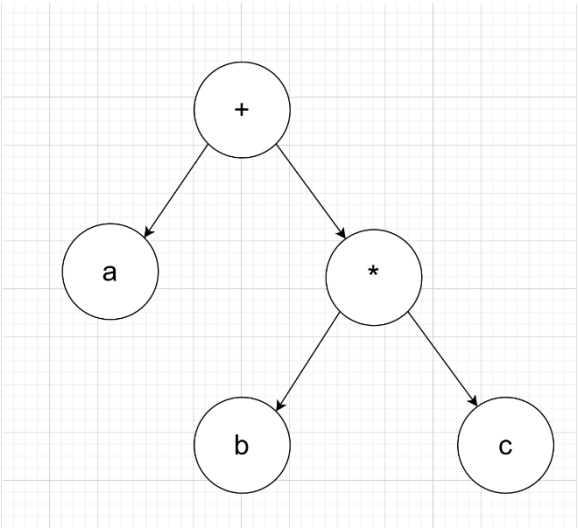
המנתח תחבירי (Syntax Analysis)

קלט היחידה: רשימת אסימונים **מהמנתח הלקסיקלי**

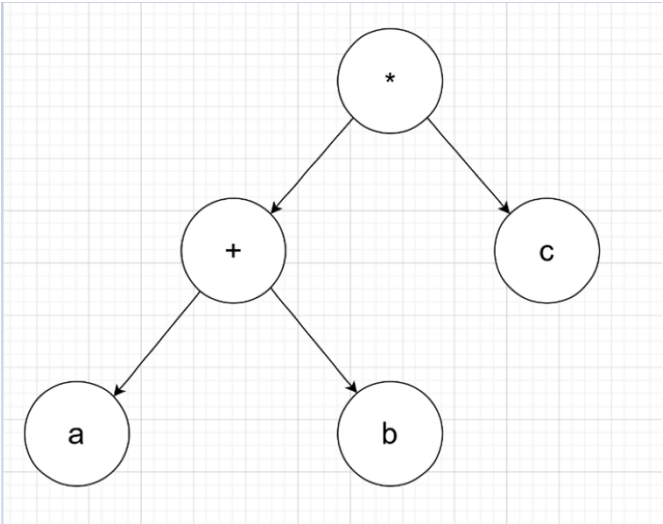
פלט היחידה: עץ ניתוח **למנתח הסמנטי**, כתיבה אל **טבלת הסימנים**, רשימת שגיאות אל מטפל **השגיאות**

מטרה: השלב הבא הוא לבדוק אם רצף הטוקנים שנוצר בתהליך הקודם תואם לחוקי התחביר של השפה (הדקדוק שלה). השלב הזה יוצר את ה-עץ תחבירי או ה-עץ סמנטי (abstract syntax tree - AST), שמייצג את מבנה הקוד לפי כללי השפה.

דוגמה: אם הקוד הוא a + b * c, האנליזר התחבירי ייצור עץ תחבירי שמייצג את ההצהרה על המשתנה וההקצאה.דוגמא לעץ סמנטי תקין:



דוגמא לעץ סמנטי לא תקין:



המנתח סמנטי (Semantic Analysis)

קלט היחידה: עץ הניתוח **מהמנתח התחבירי**, **טבלת הסימנים**

פלט היחידה: כתיבה אל **טבלת הסימנים**, רשימת שגיאות אל מטפל **השגיאות**

שלב הניתוח הסמנטי הוא שלב שמטרתו לעבור על ה**עץ הסמנטי** ולוודא שאין בקוד שגיאות סמנטיות כגון, בדיקת טיפוסים, וידוא התאמה של פרמטרים לפונקציות ועוד....

מכולל קוד הביניים

קלט היחידה: עץ הניתוח **מהמנתח התחבירי, טבלת הסימנים**

פלט היחידה: כתיבת **קוד הביניים**, רשימת שגיאות אל מטפל **השגיאות**

תפקיד היחידה הינו עבירה על העץ הסמנטי ותרגומו לקוד ביניים הישמש לשלבים הבאים ב-beck end

יצירת קוד הסף

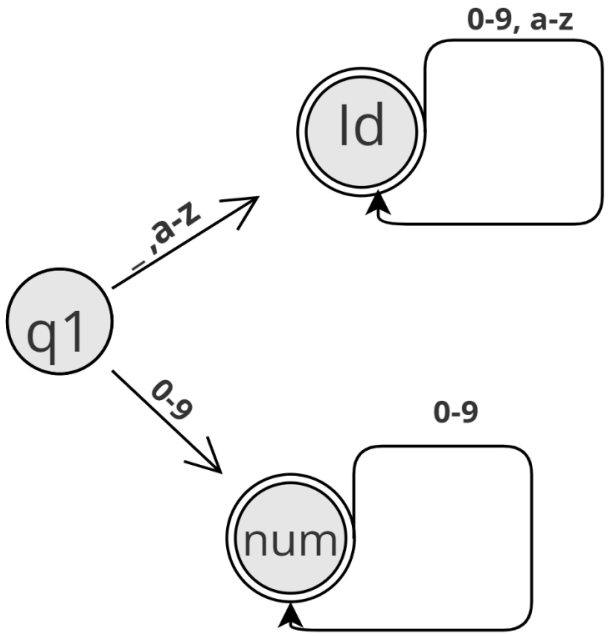
קלט היחידה: עץ הניתוח **קוד הביניים** , **פלט היחידה**: כתיבת **קוד הסף**

תפקיד היחידה הינו לקיחת קוד הביניים שעבר אופטימיזציה ולהמיר אותו לקוד סף

מבנה נתונים

בפרויקט נעשה שימוש בשלל מבני נתונים. אחד ממבני הנתונים המרכזיים בפרויקט שלי הינו, **מטריצה דו ממדית**

במהלך פיתוח המהדר כשנרצה להשתמש **באוטומט דטרמיניסטי סופי** נצטרך לייצג אותו בקוד. לדוגמא, נקח את האוטומט הפשוט הבא:



כפי שאנו רואים, האוטומט יצליח לזהות מספרים ומזהים . כלומר: אם ניקח לדוגמה את המזהה sum נראה כי האוטומט יסיים בצומת Id וכך נדע שהוא מזהה. ציין כי עם קיבלנו קלט שהוא לא אחד מהקלטים האפשריים בצומת הנוכחית, נשלח לצומת מלכודת (Trap state) המציינת טוקן לא תקין

אחת הדרכים לייצגו בקוד הינה **מטריצה דו ממדית** בעזרתה נוכל את האוטומט שלנו בצורה הבאה:

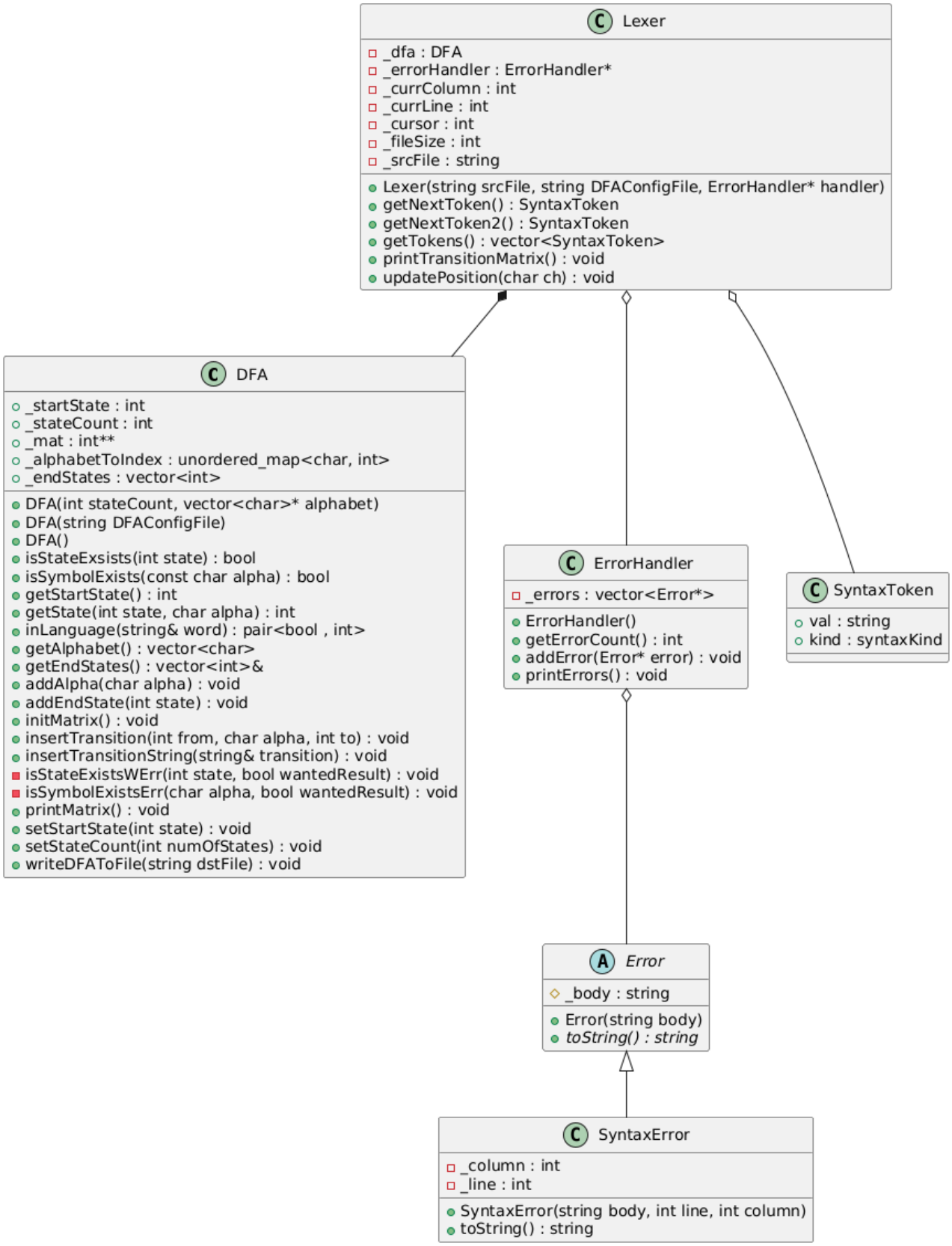
STATE	NUMBER	LETTER	_
q1	**num**	**Id**	**Id**
num	**num**	**TRAP**	**TRAP**
id	**Id**	**Id**	**TRAP**

למה מטריצה?

המטריצה נותנת לנו עבודה נוחה ומהירה של האוטומט וייתרן רציני על צורות ייצוג אחרות בכך ששליפה מהמטריצה (מציאת המצב הבא) הינה – O(1)

תיאור מערכת התוכנה

תיאור המחלקות (UML):



תיאור המחלקות (טבלה):

המחלקה DFA:

מחלקה המייצגת אוטומט דטרמיניסטי סופי (DFA)

כותרת הפונקציה	הסבר
DFA(int stateCount, vector<char> *alphabet)	בנאי האוטומט, המאתחל אותו בעזרת מספר המצבים והשפה של האוטומט
DFA(string DFAConfigFile)	בנאי המאתחל את האוטומט לפי קובץ
DFA()	בנאי הבונה אוטומט ריק
void addAlpha(char alpha)	מוסיף סימן לאלף בית של האוטומט
void setStateCount(int numOfStates)	מגדיר את כמות המצבים של האוטומט
void setStartState(int state)	מגדיר את המצב ההתחלתי של האוטומט
int getStartState() const	מחזיר את המצב ההתחלתי של האוטומט
void addEndState(int state)	מגדיר מצב סופי חדש לאוטומט
const vector<int> & getEndStates() const	מחזיר רשימה של המצבים הסופיים של האוטומט
vector<char> getAlphabet() const	מחזיר רשימה של סימני האוטומט
int getState(int state, char alpha) const	מחזיר את המצב הבא של האוטומט לפי מצב נוכחי וסימן
void initMatrix()	מאתחל אל מטריצת המצבים של האוטומט
void insertTransition(int from, char alpha, int to)	מוסיף מעבר למטריצת המצבים לפי מצב התחלתי סימן ומצב הבא
void insertTransitionString(string &transition)	מוסיף מעבר למטריצת המצבים לפי מחרוזת

bool isStateExsists(int state) const	מחזיר האם המצב קיים באוטומט
bool isSymbolExists(const char alpha) const	מחזיר האם הסימן קיים באוטומט
pair<bool, int> inLanguage(string &word) const	מחזיר האם מילה קיימת בשפת האוטומט ובאיזה מצב סופי היא הסתיימה
void writeDFAToFile(string dstFile)	כתיבת האוטומט לקובץ
void printMatrix() const	הדפסת מטריצת המעברים של האוטומט

<u>המחלקה Lexer:</u>

המחלקה היא רכיב **המנתח הלקסיקלי** במהדר, היא אחראית על ייצור האסימונים אל **המנתח הסמנטי** ודיווח השגיאות אל **מטפל השגיאות**

כותרת הפונקציה	הסבר
Lexer(string srcFile, string DFAConfigFile, ErrorHandler *handler)	בנאי המנתח הלקסיקלי לפי קובץ מקור, קובץ קונפיגורציה לDFA וכתובת של מטפל השגיאות
vector<SyntaxToken> getTokens()	מחזיר את **רשימת האסימונים**
SyntaxToken getNextToken()	מחזיר את האסימון הבא לפי קובץ המקור
void updatePosition(char ch)	מעדכן את פוזיציית הרחיב לפי תו
void printTransitionMatrix() const	מדפיס את מטריצת המצבים של אוטומט המנתח

המחלקה Error handler:

המחלקה היא רכיב **מטפל השגיאות**, של המכולל והוא אחראי לשמור את השגיאות הנוצרות בתהליך הקומפילציה

כותרת הפונקציה	הסבר
ErrorHandler()	בנאי המחלקה, המאתחל את המטפל אל רשימת שגיאות ריקה
int getErrorCount() const	מחזיר את כמות השגיאות הקיימות במטפל
void addError(Error *error)	מוסיף שגיאה לפי כתובת
void printErrors()	מדפיס את השגיאות

המחלקה Error:

מחלקה אבסטרקטית המייצגת שגיאת מהדר

כותרת הפונקציה	הסבר
Error(string body)	בנאי המחלקה, המאתחל שגיאה לפי הודעת שגיאה
virtual string toString() const	החזרת שגיאה כמחרוזת

המחלקה SyntaxError:

מחלקה המייצגת שגיאת מהדר בשלב **הניתוח הלקסיקלי**

חתימות המחלקה והפונקציונליות שלהן זהות לאלו במחלקה **Error**

מבנים נוספים:

syntaxKind – enum המייצג **סוג האסימון**

syntaxToken – struct המייצג **אסימון לקסיקלי**