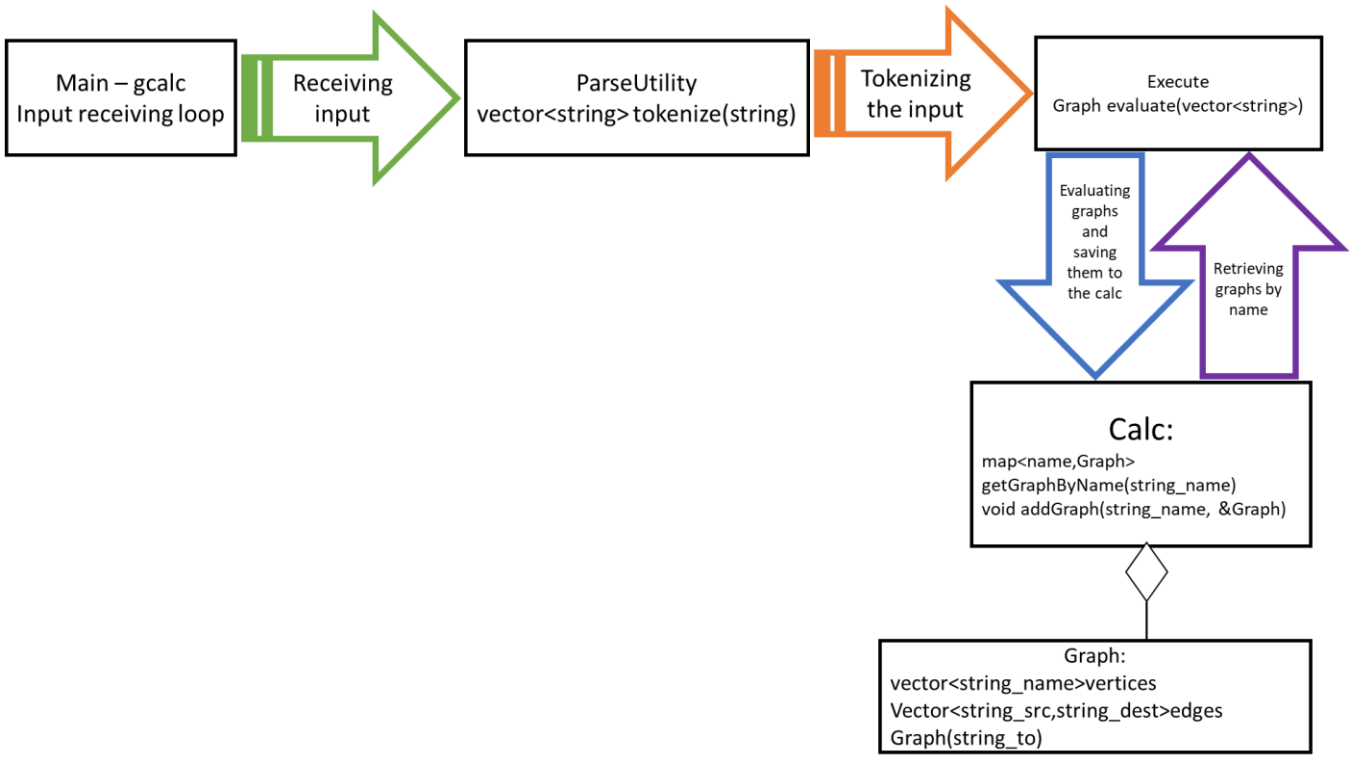

<u>סקירת המחלקות:</u>

Gcalc ה main של התוכנית, אחראית על קביעת מצב התוכנית, וקבלת הקלט שורה שורה מערוץ הקלט.

ParseUtility אחראית על חלוקת שורת המחרוזת שנקלטה למערך טוקנים כפי שמוסבר בהמשך.

Execute אחראית על הערכת מערך הטוקנים כאשר מבצעת בדיקת תקינות של כל טוקן.

Calc מבנה הנתונים בו נשתמש על מנת לשמור את משתני הגרף של המחשבון- מפה בה המפתח הוא שם הגרף ואילו הערך הוא הגרף עצמו. ממשת את פעולות save and load הבינאריות. **הסבר לload בסוף הקובץ.**

Graph מגדירה את האובייקט גרף, מעמיסה את האופרטורים המגדירים את הפעולות המותרות בין גרפים.

<u>תיאור תהליך ריצת התוכנית:</u>

<u>בקצרה: מקבלים קלט, מחלקים אותו לטוקנים, מוודאים את תקינות כל טוקן, מפעילים פעולות סדרתיות על מנת לשמור על סדר הפעולות התקין של המחשבון עד שמקבלים גרף שאותו שומרים למפה של Calc .</u>

<u>טיפה פחות בקצרה:</u>

אציין כי ייתכן וסדר הדברים בפועל במימוש בתוכנית טיפה שונה על מנת שאוכל לפסול מקרים שגויים מראש ולייעל את התוכנית. בסוף ההסבר אני נותן דוגמא לכך**.

<u>שלב 1: קביעת מצב התוכנית וקבלת קלט : gcalc</u>

מצב ריצת התוכנית נעשה ב main כאשר אם מסופקים לתוכנית קובץ פלט וקובץ קלט היא תעבוד במצב אוטומטי ואם לא תעבוד במצב אינטרקטיבי.

הקבלה שלה האינפוט נעשית באמצעות לולאה הקוראת שורה שורה. עשיתי שימוש בספרייה rdbuf על מנת לעשות רידיירקט ל stdout לתוך קובץ במידה אני במצב האוטומטי. כך שכל מה שהודפס במהלך ריצת התוכנית יכתב לקובץ הפלט.

שלב 2: החלוקה לטוקנים: ParseUtility

מקבלים קלט של שורה ומחלקים אותה למערך של טוקנים, על ידי הטוקנים הבאים: "(…)load", "words", "{…}" , כל תו אחר הוא טוקן בודד. אנו נפתרים מרווחים למעט בטוקן "(…)load " שם יש חשיבות לרווחים בתוך שם הקובץ. (בנוגע ל" words" כמובן שאם היה רווח בתוך מילה אז המילה תחולק לשני טוקנים נפרדים) בשלב זה איננו בודקים תקינות של אף טוקן.

שלב ההערכת השורה: Execute

נשים לב כי ניתן לאפיין את פעולות המחשבון על ידי שני מצבים שונים. הראשון הוא ביצוע פעולה המיוצגת על ידי מילה שמורה כמו print. והשני, הוא ביצוע פעולת השמה, כאשר אגף ימין מייצג שם תקין של משתנה גרף ואילו בצד שמאל קיים ביטוי בעל ערך. בשני המצבים לקלט יש מבנה מסויים שניתן לנצל. עבור הראשון לכל מילה שמורה יש מבנה משלה לדוגמא עבור: (…)print אני יודעים כי חייבים להופיע הטוקנים הבאים:

‘print’ , ‘(’ , ‘…’ , … , ’… ’, ‘)’

כלומר הטוקן הראשון חייב להיות המילה השמורה הדפס, השני סוגריים פותחים, והאחרון סוגריים סוגרים.

כעת, נותר להעריך את הביטוי שנמצא בתוך הסוגריים. שאותו נעביר לפעולת הevaluate.

כעת נעבור למצב השני, לאחר שסיימנו עם כל המילים השמורות, מבנה השורה חייב להיות מהצורה:

‘graph_name’ , ‘=’ , ‘…’ , … , ’… ’

כעת נותר להעריך את הביטוי שנמצא באגף שמאל שאותו נעביר לפעולה evaluate. וגם לוודא ששם הגרף מימין תקין.

הפעולה evaluate:

הפעולה ראשית עוברת ומחליפה את כול הטוקנים של "(…)load" במחרוזת של קבוע גרף המתאר את הגרף הנטען.

כעת הפעולה משנה את סדר הטוקנים בהתאם למופעי הסוגריים.

(ממומש באמצעות מחסנית כאשר אנו דוחפים את האינדקס של סוגריים פותחים וכאשר מקבלים טוקן של סוגריים סוגרים אני מחשבים את הביטוי ומחליפים אותו בטוקן אחד המייצג את הגרף כמחרוזת. בשלב זה נבדקת התקינות של הסוגריים באגף שמאל של המשוואה. יש לשים לב כי אם הופיע אופרטור ! לפני הביטוי יש לקחת זאת בחשבון בחישוב קבוע הגרף ולהסיר גם אותו.)

לאחר שעשינו זאת קיבלנו שורה שיש בה שמות של משתני גרף, קבועי גרף, ואופרטורים. מאחר ולאופרטור ! יש קדימות על האופרטורים האחרים אנו עוברים ומחליפים את כל המופעים שלו עם קבוע גרף מתאים. לאחר מכן נשאר עם מערך טוקנים שלכל האופרטורים בו יש אותה הקדימות אז כל שנותר הוא לרוץ משמאל לימין ולחשב את הביטוי.

למחלקת גרף יש בנאי המקבל מחרוזת שאותה הוא מחלק אותה לטוקנים, מבצע את כל הבדיקות הדרושות ומחזיר אובייקט של גרף אסביר איך ניתן **לפענח הגדרת גרף תקינה בסוף הקובץ**. למחלקה גרף הועמסו האופרטורים של פעולות המחשבון (עבור פעולת אופרטור בין שני גרפים). לכן כאשר אנו רוצים לחשב את הביטוי מימן לשמאל כל שנותר הוא ליצור גרף חדש מהטוקן הראשון שיהיה גרף התוצאה ואז להפעיל את האופרטור המתאים על גרף התוצאה עם הטוקן הבא במערך כאשר אם מדובר קבוע גרף מתבצעת קריאה לבנאי של גרף המחזיר אובייקט גרף עבור המחרוזת ואילו אם מדובר בשם משתנה ניגשים למפה במחשבון אשר ממפה כל גרף על ידי השם שלו כמפתח ואובייקט הגרף כערך.

(כמובן שהפעולה הזו לא עושה את כל הפעולות האלו בעצמה אלה קוראת לפעולות עזר כגון- evaluteBeforeRemovingBrackets ו replaceAllLoad).

**דוגמא לשוני אפשרי בתוכנית הוא כאן כאשר את הבדיקה של אגף ימין (שם המשתנה גרף) נעשה לפני שבכלל נקרא לפעולה evaluate כלומר רק אחרי שנבדוק שאגף ימין הוא טוקן המייצג שם גרף תקין.

שלב שמירת הגרף

לאחר שביצענו את כל ההערכה ניתן לשמור את הגרף שקיבלנו במפה שנמצאת במחלקת מחשבון כפי שהזכרתי.

באמצעות הפעולה addGraph.

איך מתרגמים גרף:

את המחרוזת גם נחלק למערך של טוקנים כאשר טוקן אחד הוא שם חוקי של קודקוד ואילו כל תו אחר הוא טוקן באורך אחד. גם כאן נפתרים מכל הרווחים. שוב אדגיש שאם היה רווח באמצע שם של קודקוד אז הוא יופרד לשני טוקנים שונים. כעת נותר לקרוא עד קו מפריד | (שהוא טוקן בודד) שמות של קודקודים. בין כל קודקוד וקודקוד חייב להופיע הוטקן פסיק . לאחר שהגענו לקו המפריד נקרא שמות של קשתות. כאשר חייב להופיע תוקן > , אחריו טוקן שם קודקוד חוקי, טוקן פסיק, טוקן שם קודקוד חוקי , טוקן <, בין כל קשת וקשת חייב להופיע טוקן פסיק גם כן.

אם הגרף לא עומד במבנה הזה נדפיס הודעת שגיאה מתאימה.

איך טוענים גרף מקובץ בינאי:

מייצרים מחרוזת מתורגמת של התוכן הבינארי ומעבירים אותה לבנאי של מחלקת גרף שיודע לקחת מחרוזת וליצור ממנה אובייקט גרף.