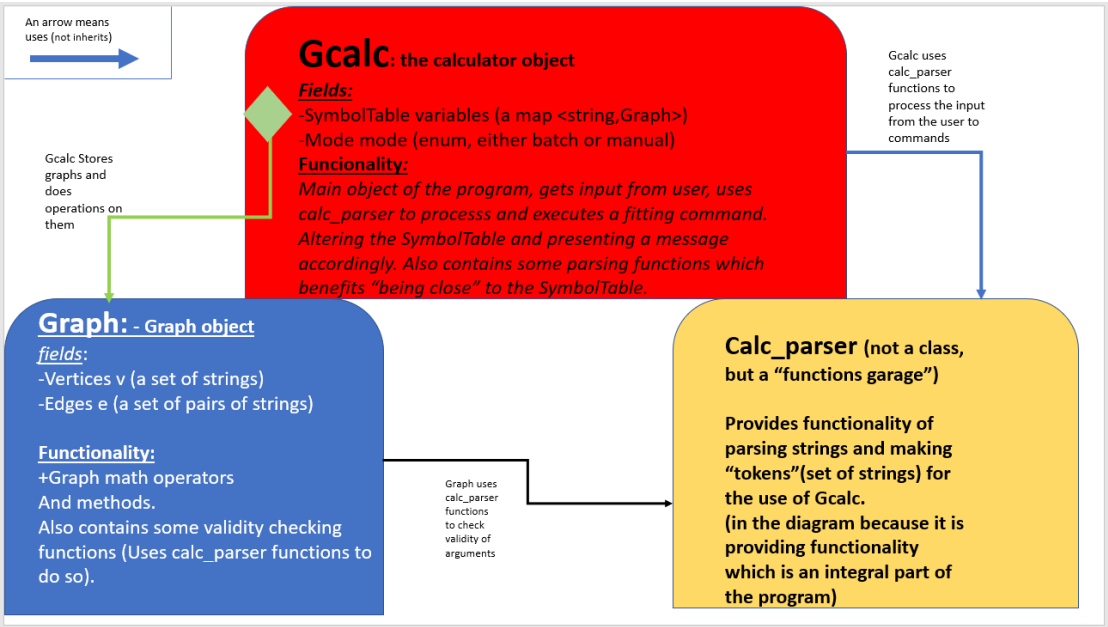
פרויקט סיום מת"ם



תיאור מהלך התוכנית:

פונקציית המain בקובץ main.cpp יוצרת אוביקט Gcalc מאותחל להיות ללא משתנים ובmode שנקבע לפי מספר הארגומנטים שנקראו ביחד את התוכנית. (2 ארגומנטים נוספים לקובץ למצב batch , ללא ארגומנטים נוספים, מצב manual).

פעולת ()start של Gcalc נקראת ומתחיל לקלוט קלט מהשתמש קורה אחר שורה.

השורה עוברת תהליך Tokenization באמצעות פעולות מ calc_parser, "ספריה" של פונקציות לעיבוד של הקלט.

Tokens הוא וקטור של מחרוזות, המתאר את הפעולה על חלקיה, הוא מחולק למחרוזות לפי סימנים מיוחדים בקלט (אופרטורים, סימני סוגריים וכו').

מרגע שנוצר הTokens המאתר את הביטוי שהתקבל, מתבצע תהליך זיהוי של תבנית הפקודה, מה היא כוונת הכותב. תנאים להתאמת הTokens עוברים דרך פעולות בGcalc וcalc_parser לעיבוד כוונת הפקודה וביצועה אם היא מוצלחת. פקודה לא מוצלחת תזרוק שגיאה מתאימה, לפי מעין "ניחוש" כוונה. לאחר שהביטויים בפקודה מובנים, נוצרים גרפים (Graph) ומבוצעות ביניהם הפעולות המתאימות.

ברגע שפקודה הסתיימה, נקראת הפקודה הבאה וכך חוזר חלילה .

כך מתבצע עד לסוף הקלט או קבלת קלט המבקש סיום של התוכנית. אז מופסקת הלולאה הקוראת פקודות מפקודת ה()start והתוכנית מסתיימת.

מחלקות:

1) **Gcalc- אויבקט המתאר את המחשבון.** *שדות: טבלת משתנים, מוד פעולה, דגל הפעלה.*

טבלת המשתנים היא map<std::string,Graph>.

מטרתו היא לאחסן מידע הרלוונטי על משתני המחשבון ומצבו.

פקודות המשתמש נקלטות , מועברות בשרשרת פענוח (פענוח סוג פקודה -> פענוח מבנה

תקין של הפקודה עצמה -> חישוב ביטוי אם צריך ..) ומשתמש לשם כך בפונקציות מ calc_parser.h המכילה פונקציות פענוח.
בחישוב ביטוי המורכב ממספר פעולות מבוצעת בצורה רקורסיבית קריאה של תתי ביטויים לפי הצורך והפעלת האופרטורים על הגרפים בהתאם ללוגיקה המתבקשת.

2) **MyGraph – אויבקט המתאר גרף.** ***<u>שדות:</u> צמתים וקשתות***
צמתים מאוחסנים כset (std) של צמתים(המתוארים ע"י מחרוזות) , קשתות מאוחסנות כset של זוגות צמתים.
מכילה פונקציות יצירה , הוספת צמתים, קשתות. פונקצית הדפסה ואופרטורים חשבוניים.
בנוסף, פעולות לוידוא תקינות של גרף, עיבודו לקובץ בינארי ועוד פעולות שהנושא העיקרי שלהן הוא גרף ותכונותיו.

ספריות עזר:

1) **Calc_parser – ספריה לניתוח מחרוזות וביטויים.**
מהווה מאגר של פונקציות המשמשות לניתוח של ביטויים מורכבים, זיהוי תבניות וחלוקה שלהם לחלקים המקלים על הפענוח.
הערה: במהלך כתיבת התוכנית התגבש רעיון למימוש של מחלקה גנרית לביצוע ניתוחים של מחרוזות וביטויים- מעין מכונת מצבים ששדותיה יהיו "חוקי שפה" ,החלוקה לtokens תהיה בא מפורטת יותר, וכך שפה מורכבת יותר תוכל להיות מפוענחת. אך רעיון זה לא יצא לפועל בגלל מחסור בזמן ומתן עדיפות לתוכנית שכבר עובדת. אין לי ספק שמימוש מחלקה מהסוג הזה היא דרך נכונה יותר לפתור בעיותparsing מורכבות יותר.

2) **myutils –** *מכילה פונקציות עזר כלליות ששימושם חוזר במחלקות השונות.*

3) **Exeptions** – *מכילה את מחלקות השגיאות השונות, רובן יורשות ממחלקת ParserException , מתחילה כל פקודת who של מחלקות יורשות ממני ב :Error ומקבלת ביטוי לשרשר לאחר מכן.*

4) **Graph_for_python** – *ממשיקה את Graph לפייתון .*