Вводится код в верхнее окошко с текстом, затем нажав на кнопку - вызывается OnCompileButtonPressed и в нем вызывается метод Compile();

В Compile мы получаем текст с текстбокса (т.е. наш код) и кидаем его в функцию AnalyzeSyntax

AnalyzeSyntax - проверяем каждую строчку кода на ошибки: на пустой файл, на символы ; (есть ли он вообще, если их 2 выдадим ошибку что в одной строке может быть только одна строка кода), в случае если все ок - идем в AnalyzeSemantic и кидаем туда опять наш код

AnalyzeSemantic - каждую строчку кода проверяем на соответствие паттернам через ф-ю Match, в случае если все ок - начинаем парсить

ф-я Match - проверяет заданную строчку кода по всем имеющимся шаблонам , выдаст ошибку если какая нибудь строчка кода не совпалане с одним из шаблонов, ф-я Match использует ф-ю GetTypeOfWord - она принимает кусок(слово) из кода и смотрит к какому типу комманд относится и возвращает этот ти, в случае когда в коде идет имя или значение переменной (а они уникальны), то мы кидаем еще и текущее слово из шаблона currentPatternWord и кидаем назад соответствующий тип.

Когда вф-ии Analyzesemantic все проверилось и все ок, то начинаем парсить и кидаем код в ф-ю Parse - в ней очищаем список переменных(values), там где храним переменные нашего языка, затем бежим по каждой строчке кода и кидаем ее в метод DoAction()

DoAction - по своей сути рекурсивно разбирает строчку кода в зависимости от команд:

-если идет ВЫВОД(он относится к command) то сразу кидаем в наш логгер вывод и ждем возврата для вывода через рекурсию

-если ПЕРЕМЕННАЯ(special) ,то здесь рекурсивно ждем переменную для добавления в val,базово чекаем на добавление в список, а также проверяем, чтобы такой переменной уже не было, иначе кидаем ошибку, если все ок добавлеям

-если идет любая операция (СУМ - тип operation), через свитч смотрим- если арифметика то получаем значение (если их не возможно кастовать к инт - то выдадим ошибку что это неинт), получив значение возвращаем сумму,разность или умножение,если идет длина строки то рекурсивно получаем строку в strret, проверяем есть ли у нее одинарные кавычки(таков синтаксис ввода строки) и возвращаем ее длину, если идет подстрока (ПОДСТР) - получаем опять же значение строки и проверяем есть такая полстрока в строке, возвращаем подстроку если есть,иначе что ее нет

-если идет TYPE тип переменной, то в нем возвращаем для special переменную с типом,именем и значением, получив ее рекурсивно

-если идет name - название переменной, то смотрим, если длина массива, на который мы разбили строку по пробелам, больше 1 -это значит что щас мы создаем переменную и возвращаем имя и значения для type,иначе идет процесс получения переменной: если список пуст – возвращаем ошибку, что переменная не определена, если не пустой, бежим по списку values и сравниваем их имена,если есть возвращаем значение, если нет после foreach выдаст ошибку в нашу консоль, что переменной нет

Это весь цикл

отмечу что список values хранит тип стринги туда наши переменные с нашего языка пишутся в формате: type,name,value -> ЧИСЛО,А,10, затем когда мы что сравнинаем,просто сплитим эту строчку по запятой и можем получить имя тип и значение