Визуальные конструкторы Telegram ботов позволяют пользователям без глубоких знаний в программировании строить ботов, с помощью набора блоков (компонентов). Каждый компонент обычно выполняет некоторую простейшую функцию, например «отправить картинку». Однако функциональные возможности при таком подходе к построению ботов ограничены набором доступных компонентов, что не позволяет создавать проекты со сложной специфичной логикой работы. Чтобы более гибко настроить логику работы бота, в систему можно интегрировать предметно-ориентированный язык программирования, направленный на расширение функциональных возможностей визуального конструктора. Так как речь идет о узкоспециализированной предметной области, язык должен быть простой и иметь небольшое количество возможных команд и операций, а внедрение какого-либо из языков общего назначения может оказаться избыточным решением.

Пример того, что можно реализовать с использования языка приведен на слайде. Это простая корзина товаров с возможностью многоразового выбора, просмотра списка выбранных товаров и их суммарной стоимости.

Цель выпускной квалификационной работы – расширение функциональных возможностей визуального конструктора Telegram ботов за счет создания предметно-ориентированного языка программирования.

Для выполнения требуемых функций достаточно, чтобы язык имел базовый набор типов данных и минимальный набор возможных операций, которые представлены на слайде.

В качестве метода трансляции кода была выбрана интерпретация, а не компиляция, так как предполагается, что язык будет использоваться для написания небольших функций, не предполагающих высоких требований к производительности выполнения, а также портативность, простота интеграции с серверной частью конструктора, меньшее время на разработку.

Для достижения поставленной цели необходимо описать грамматику предметно-ориентированного языка, разработать алгоритмы функционирования модуля интерпретации для него и выполнить программную реализацию.

В процессе разработки грамматики был выполнен обзор способов задания языков, разработана и описана с помощью расширенной формы Бэкуса-Наура формальная грамматика языка.

Модуль интерпретации состоит из: лексического анализатора (лексера), синтаксического анализатора (парсера), семантического анализатора и непосредственно исполнителя.

Процесс лексического анализа является первым шагом в трансляции исходного кода программы и формирует основу для следующих этапов, таких как синтаксический анализ и построение абстрактного синтаксического дерева.

После проектирования лексического анализатора выполнена разработка алгоритмов функционирования синтаксического анализатора. Результатом работы парсера является абстрактное синтаксическое дерево (AST), которое отражает синтаксическую структуру входной программы и содержит всю необходимую информацию для дальнейших этапов работы транслятора. В данном проекте реализован нисходящий анализатор, работающий по методу рекурсивного спуска, известный как парсер Пратта.

Семантический анализ выполняется после построения абстрактного синтаксического дерева синтаксическим анализатором. В данном проекте под семантическим анализом понимается процесс, в ходе которого выполняются такие операции, как проверка типов данных, правильность использования переменных, функций, выражений, а также обнаружение смысловых ошибок в коде. Результат анализа формируется в виде внутреннего представления, так называемой объектной системы. Её можно представить в виде диаграммы классов.

Последним этапом в процессе обработки исходного кода является его исполнение. На данном этапе выполняется вычисление выражений, представленных в виде объектов внутреннего представления. Результат исполнения формируются в виде объектов внутреннего представления.

Программная реализация выполнена с использованием языка программирования Go. Go – компилируемый многопоточный язык. Также использование данного языка упрощает интеграцию модуля интерпретации с серверной частью конструктора. Интеграция выполняется с помощь импорта в бэкенде конструктора пакета с модулем интерпретации, который предоставляет API в виде функции, принимающей на вход исходный код на предметно-ориентированном языке с необходимыми переменными окружения конструктора, и возвращает результат интерпретации.

Выполнено модульное тестирование основных компонентов программного продукта, а также приведен пример использования.

Результат работы - разработанный предметно-ориентированный язык и модуль интерпретации для него, расширяющий функциональные возможности визуального конструктора Telegram ботов.