ГЕНЕРАТОР БЛОК-СХЕМЫ АЛГОРИТМА НА ОСНОВЕ ИСХОДНОГО КОДА

Буринский Н.А., студент гр.458304

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Ковшер Е.И. – ассистент

В тезисе описывается программное обеспечение в виде расширения для MS Visio, предназначенное для автоматизированного построения блок-схем алгоритмов на основе исходного кода на языке C. Рассматриваются этапы преобразования кода: разбиение на токены, обработка команд, разделение на зоны видимости и генерация схемы. Анализируется работа алгоритма, включая рекурсивную обработку вложенных зон и стилизацию результата. Приводятся примеры работы программы на тестовом образце кода.

Современная разработка программного обеспечения требует наглядного представления алгоритмов для их анализа, отладки и документирования кода. Блок-схемы остаются одним из эффективных способов визуализации логики программы, однако их ручное построение является трудоемким и часто приводит к ошибкам. Вследствие этого было разработано программное решение, автоматически преобразующее исходный код в блок-схемы алгоритмов. Предложенный подход значительно упрощает процесс анализа программной логики и снижает вероятность некорректной интерпретации сложных управляющих конструкций.

Предложенное программное обеспечение выполняет генерацию блок-схемы алгоритма на основе исходного кода на языке C [1] и реализовано в качестве расширения для MS Visio [2]. Данный подход позволяет сразу после генерации перейти к редактированию результата. Процесс преобразования исходного кода алгоритма состоит из нескольких основных этапов:

1. Разбиение и классификация текстового кода на команды/токены.
2. Постобработка полученного массива команд.
3. Разбиение кода на зоны, соответствующие областям видимости, формируемым управляющими конструкциями (if, switch, while).
4. Генерация схемы зон с рекурсивной обработкой вложенных зон и созданием объектов схемы в зависимости от типа команд.
5. Постобработка и стилизация результирующей схемы.

Список типов команд содержит не только алгоритмические команды, необходимые пользователю, такие как ввод, вывод и вызов внешней функции, но и внутренние токены, которые обеспечивают корректную работу программы, например, токены открытия и закрытия зон областей видимости. Присвоение типа команде начинается со сравнения её с базой определенных команд, которая при необходимости может быть расширена пользователем. Если поиск команды не дал результата, но при этом выполняются несколько требований к команде, ей может быть присвоен тип действия или внешней функции; в противном случае она отбрасывается.

Постобработка массива команд включает:

1. Обработка необычных конструкций языка с приведением их к более стандартному виду, например, добавление токенов открытия и закрытия области видимости ветвления, если соответствующие токены явно выделены не были.
2. Объединение нескольких последовательных команд одного типа в одну, например, при последовательном вызове нескольких операций вывода текста в консоль. С точки зрения алгоритма, эти операции могут рассматриваться как один общий вывод.
3. Обработка текстового представления каждой команды, удаление пробельных символов, возможная обрезка текста.

Генерация схемы происходит через рекурсивный вызов обработчиков зон и генераторов объектов в зависимости от обрабатываемых команд. Команды считываются из списка последовательно и, при достижении команды, открывающей зону (ветвление, цикл и т.д.), в зависимости от ее типа запускается соответствующий обработчик этой зоны, Он, в свою очередь, обрабатывает особенности данной зоны, создает команды и вызывает обработчики для вложенных зон. Большинство команд и обработчиков при создании также запускает установку связей между объектами команд, вследствие которой будет также создан и настроен объект соединения.

В процессе генерации схемы также производится запись ссылок на все объекты в списки по различным параметрам, таким как тип команды, тип фигуры объекта, области видимости, в которой находится команда и т.д. Данная информация будет также использована при стилизации и постобработке результирующей схемы. Примеры результата генерации схемы, а также окна ввода программы с тестовым образцом кода представлены соответственно на рисунках 1 и 2.

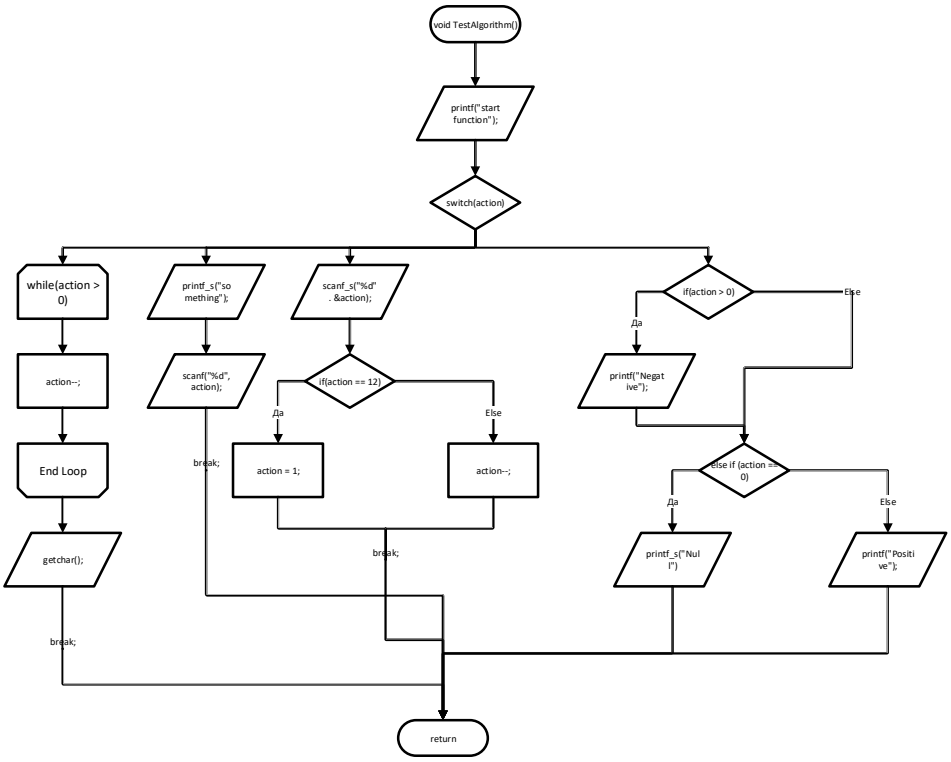

Рисунок 1 – Пример результата генерации

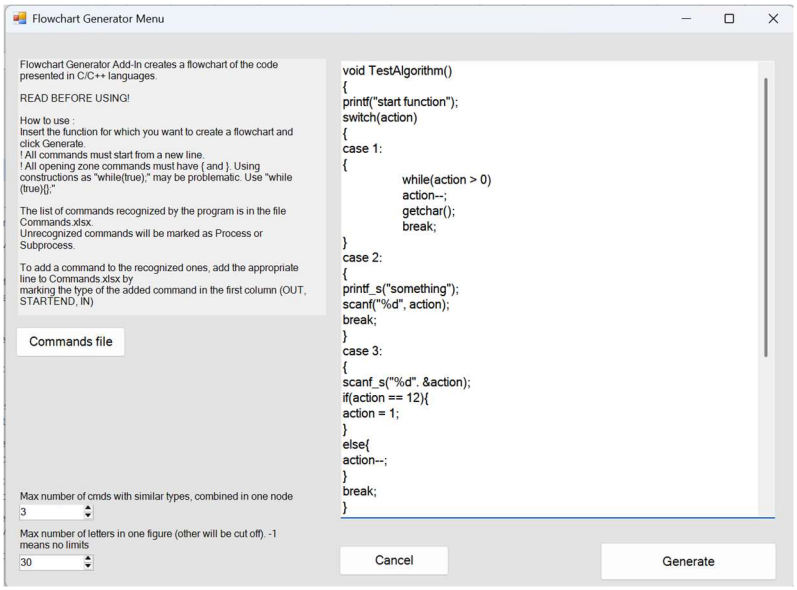

Рисунок 2 – Пример окна ввода программы

Текстовое поле содержит код, на основе которого будет сгенерирована схема. В левой части окна располагаются поля для настройки некоторых параметров стилизации, а также кнопка открытия файла, содержащего информацию об известных программе командах.

Разработанное программное обеспечение значительно ускоряет процесс визуализации алгоритмов в виде блок-схем, что особенно важно в условиях высокой трудоемкости их ручного проектирования. Предлагаемый инструмент обладает большим потенциалом для применения в образовательной сфере и может быть эффективно использован для анализа и рефакторинга программного кода.

Список использованных источников

1. C/C++ [Электронный ресурс] // Microsoft Docs. – 2024. – Режим доступа: https://learn.microsoft.com/ru-ru/cpp/?view=msvc-170 (дата доступа: 28.10.2024).

2. Надстройки VSTO для MS Visio [Электронный ресурс] // Microsoft Docs. – 2024. – Режим доступа:https://learn.microsoft.com/ru-ru/visualstudio/vsto/visio-solutions?view=vs-2022 (дата доступа: 24.11.2024).