ТЕХНІЧНІ НАУКИ

Алещенко А.В.
аспирант;
Научный руководитель: **Бузовский О.В.**
доктор технических наук, профессор,
профессор кафедры вычислительной техники,
Национальный технический университет Украины
«Киевский политехнический институт»

РАЗРАБОТКА СИСТЕМЫ ГЕНЕРАЦИИ ПРОГРАММНЫХ КОДОВ НА ОСНОВЕ ГРАФИЧЕСКОЙ НОТАЦИИ СХЕМ АЛГОРИТМОВ

Рассматривается разрабатываемая система, позволяющая генерировать исполняемый код по графическому представлению алгоритма (блок-схема алгоритмов или activity diagram UML). В настоящее время существует большое количество CASE-систем, которые позволяют автоматизировать процесс разработки программных продуктов (ПП). При этом подавляющее большинство CASE-систем ориентированы на применение унифицированного языка моделирования UML. Этот язык является основным инструментом CASE-технологий при объектно-ориентированном подходе к разработке ПП и включает в себя систему различных диаграмм, на основании которых может быть построено представление о проектируемом продукте.

Программные средства, реализующие данный подход, позволяют документировать проект и построить набор диаграмм и спецификаций, целью которых является получение каркаса ПП (диаграммы классов) который описывает структуру классов, интерфейсов и отношения между ними [1, с. 181]. Эта информация достаточна для генерации кода каркаса с так называемыми «заглушками», определяющими реализацию конкретных методов, но не позволяет получить полный исполняемый код.

Следовательно, CASE-средства не позволяют полностью автоматизировать процесс создания ПП, хотя большой процент затрат в рамках проекта связан именно с кодированием тел методов. Разрабатываемая в рамках данной тематики система призвана свести ручное кодирование к минимуму, а в конечной цели – полностью отказаться от него.

Одной из причин затруднения автоматического построения кода есть проблема, связанная с описанием типов и объявлением переменных [2, с. 18]. В процессе разработки системы рассматривались следующие варианты задания соответствия между переменными и их типами: именная, динамическая и явная.

Именная типизация означает, что в зависимости от данных, которые хранит в себе переменная, модифицируется её имя. Например, используется добавление к имени переменной символа «i» и т.п. [3, с. 66]. При этом нарушается семантика имён и затрудняется описание подтипов.

Динамическая типизация означает, что предварительно тип переменной не задаётся, а определяется по мере выполнения программы (динамически), соответственно значению присваивания. Так тип одной переменной может меняться в зависимости от значений, присваиваемых ему, и операций, в которых он принимает участие. Данный способ типизации удобен при использовании графической нотации, так как не требует предварительного определения типов переменных. Однако это делает невозможным контроль типов, как на этапе компиляции, так и на этапе выполнения, а также затрудняет отладку программы и разработку самого транслятора.

По этой причине предпочтение отдаётся третьему варианту, в котором описание типа и объявление переменной выполняются явно. Это позволяет исключить ошибки, связаны с типизацией. Преимуществами явной типизации являются её наглядность, надёжность и распространённость.

На ряду с ошибками типизации, существует вторая группа ошибок, связанная с некорректным описанием граф-схемы алгоритма (ГСА). Следовательно, требуется предварительная проверка корректности задания самой ГСА.

Возможны следующие ошибки структуры ГСА:

– отсутствие терминальных вершин (начало и конец);

– множественность терминальных вершин одного класса;

– наличие бесконечных циклов (частично, в той мере, в какой оно определяется структурой блок-схемы);

– недостижимость вершины из начальной;

– наличие вершин, из которых отсутствует путь в конечную вершину.

Также возможным является выявление семантической ошибки ГСА, которая заключается в отсутствии изменений в теле цикла значений переменных входящих в состав условия завершения цикла.

Исходя из вышеизложенного, утверждается, что явная типизация и контроль корректности задания ГСА позволяют снизить количество ошибок, и, следовательно, увеличить корректность генерируемого кода.

Список использованных источников:

1. Гайнуллин Р. Ф. Разработка методов и средств анализа и контроля диаграмматики бизнес-процессов в проектировании автоматизированных систем: дис. … кандидата технических наук: 05.13.12 / Гайнуллин Ринат Фаязович. – Ульяновск, 2014. – 189 с.

2. Вирт Н. Алгоритмы + Структуры данных = Программы / Вирт Н. – М.: Мир, 1985. – 406 с.

3. Роберт У. Себеста. Основные концепции языков программирования / Роберт У. Себеста – М.: Издательский дом «Вильямс», 2001. – 672 с.