Задание 5

Реализуйте контроль использования приспособленцев из задания 4 с помощью «умных указателей» на основе шаблона Proxy.

Для этого разработайте класс ExpressionPointer, который используется так же, как обычный указатель Expression*. Доступ к объекту Expression должен осуществляться с помощью перегруженного оператора «->».

Модифицируйте фабрику так, чтобы функции createConstant() и createVariable() возвращали ExpressionPointer.

Добавьте в фабрику метод, который будет создавать объект класса Addition и возвращать ExpressionPointer на созданный объект. Можно использовать фабрику для хранения информации о числе использований объектов Addition. Например, можно хранить в фабрике хэш-таблицу, отображающую Expression * в число использований, и отслеживать в ней число всех контролируемых объектов, включая как приспособленцев Constant и Variable, так и обычные объекты Addition. Разница между приспособленцами и Addition будет состоять лишь в том, что Addition создаётся всегда новым, а при запросе на создание приспособленца может быть возвращён указатель на уже существующий объект.

Класс ExpressionPointer должен обеспечить корректный подсчёт ссылок при копировании и удалении умных указателей. Всё управление подсчётом числа текущих использований объектов-выражений должно осуществляться в классах ExpressionPointer и ExpressionFactory. Удаление объектов-приспособленцев должно в результате происходить по правилам, описанным в задании 4, без дополнительных действий со стороны других классов, например Addition.

Для упрощения логики удаления объектов можно реализовать иерархию умных указателей (унаследовать от ExpressionPointer классы ConstantPointer и VariablePointer) и сделать деструктор ExpressionPointer’a шаблонным методом, включающим процедуру удаления объекта из реестра фабрики. Этот приём позволит полностью отказаться от использования условной логики и dynamic_cast при удалении объектов.

Реализовав умные указатели согласно этому заданию можно разделить ответственности классов – классы Expression, Constant, Variable и Addition будут отвечать только за предметную область – проведение вычислений. Классы ExpressionPointer и ExpressionFactory будут отвечать за контроль времени жизни объектов и использование их как приспособленцев. Это лучше соответствует одному из принципов объектно-ориентированного дизайна – принципу единственной ответственности класса, чем вариант, который был реализован в задании 4.