Ю. В. Устимчук, Е. М. Березовская
(ГГУ им. Ф.Скорины, Гомель)

РЕАЛИЗАЦИЯ ЭЛЕМЕНТОВ РОДОВОГО ПРОГРАММИРОВАНИЯ

До настоящего момента предметом изучения была инженерия предметной области, средство разработки и реализации моделей семейств систем – доменных моделей. В то же время возможность разработки доменных моделей – это лишь первый шаг в направлении порождающего программирования. В контексте порождающего программирования родовое программирование выполняет важную функцию структурирования пространства решений порождающей доменной модели.

Назначение родового программирования заключается в том, чтобы алгоритмы и структуры данных можно было выражать в свободно адаптируемой форме, предполагающей их способность к взаимодействию и непосредственное применение в ходе конструирования программных средств.

Суть родового программирования, которое обязано своему распространению стандартной библиотеке шаблонов C++, состоящей из структур-контейнеров и алгоритмов, заключается в том, что оно позволяет представлять предметные области в виде коллекций абстрактных компонентов очень общего характера, из многочисленных сочетаний которых производятся крайне эффективные конкретные программы. Поддержка родового программирования со стороны объектно-ориентированных средств моделирования и программирования довольно скудна. Большинство инструментов просмотра, существующих в рамках интегрированных объектно-ориентированных сред разработки, основываются на иерархиях наследования и с точки зрения организации родовых библиотек оказываются практически бесполезными.

В работе рассматривались принципы и методики родового программирования – в частности, параметризация типов, разного рода полиморфизм, понятие параметризированных компонентов, принципы родового анализа и проектирования.

Практика применения полиморфизма подтипов в сочетании с *переопределением* является средством обеспечения нескольких реализаций одного и того же виртуального метода в различных классах. Среди языков со статическим контролем типов перегрузка довольно распространена (она присутствует, в частности, в C++ и Java, однако ее нет в C и Pascal). Следует иметь в виду, что для дифференциации перегруженных реализаций при перегрузке тип результата не используется, зато задействуются типы всех аргументов. Почти повсеместное отсутствие поддержки перегрузки (переопределения) типа результата связано с возможностью вызова функции «исключительно ради побочных эффектов».

Рассмотренные концепции были проиллюстрированы фрагментами кода на языке C++ и Java.