

# Язык программирования E<sub>q</sub>

## — Аннотация —

Зайченков П.О.<sup>1</sup> и Шинкаров А.Ю.<sup>2</sup>

<sup>1</sup> Московский физико-технический институт, Кафедра информатики и  
вычислительной техники

<sup>2</sup> University of Hertfordshire, Hatfield, Hertfordshire, AL10 9AB, United Kingdom

Большинство естественнонаучных проблем сопряжено с огромным количеством вычислений, которые варьируются от решения систем уравнений до обработки данных, полученных с телескопа. Большинство таких вычислений реализовано на языках программирования подобных Фортрану. На то имеется ряд причин: во-первых, обратная совместимость – метеорологические программы насчитывают миллионы строк кода, а первые версии появились в тот момент, когда Фортран был одним из самых передовых языков программирования; во-вторых, производительность – языки высокого уровня, такие как MatLab, Python и Java, предоставляют высокий уровень абстракций, но не могут состязаться в скорости с языками низкого уровня; и, наконец, сложность разработки – Фортран оказался удачным компромиссом между скоростью разработки и скоростью исполнения программы. На сегодняшний день тенденции в производстве компьютеров сменились с гонки за тактовой частотой на увеличение количества ядер процессора. Именно этим фактом обусловлен рост интереса к параллельному программированию. Немаловажную роль играет появление на рынке графических ускорителей GPGPU, предоставляющих еще большие возможности для увеличения скорости программ, однако требующие серьезного изменения парадигмы программирования.

Для того, чтобы выполнить программу параллельно, необходимо выявить участки кода без временных зависимостей, т.е. набор операций, который можно исполнить в произвольном порядке. Сделать это можно двумя способами: либо указать на такие участки явным образом (такой подход, к примеру, используется в библиотеках MPI и OpenMP), либо предоставить возможность компилятору – посредством некоторого анализа, выявить такие участки самостоятельно. В первом случае, программист имеет больше контроля над тем, как именно будет исполняться программа, но сложность разработки и отладки такой программы увеличивается на порядок. Во втором случае, возрастает сложность компилятора, однако уменьшается время, необходимое на разработку.

В данной статье мы представляем язык программирования E<sub>q</sub>, который использует комбинацию двух подходов и с помощью своего синтаксиса значительно упрощает анализ параллельности кода. В языке E<sub>q</sub> имеется две принципиально важных конструкции: параллельное и рекуррентное выражение. Рекуррентное выражение – это аналог нити исполнения в операционной системе, когда одно действие должно строго следовать за другим. Внут-

ри параллельного выражения все действия могут быть исполнены в любом порядке. Мы утверждаем, что двух этих конструкций, ветвления и атомарных операций достаточно, чтобы записать любую программу на Фортране и принять оптимальное решение о параллельности того или иного участка кода. В качестве примера рассмотрим гнездо циклов в некоторой программе на Фортране. Предположим, что некоторая часть операций внутри этих циклов может быть исполнена параллельно, а значит, может быть записана с помощью параллельного выражения. В таком случае все оставшиеся операции группируются в одно рекуррентное выражение, где формула рекуррентности заменит шаг изменения счетчиков внутри циклов.

Основой синтаксиса для E<sub>q</sub> является текстовый процессор L<sup>A</sup>T<sub>E</sub>X, являющийся стандартом для верстки научных публикаций. В итоге программа, записанная на E<sub>q</sub>, понимается текстовым процессором и имеет стандартный графический оттиск; с другой стороны, эта же программа компилируется на большинство современных архитектур. Язык E<sub>q</sub> позволяет разделить усилия ученого, заинтересованного исключительно в результате определенных вычислений, и программиста, разрабатывающего компилятор, соответствующий ситуации на современном рынке компьютеров.