Система программирования Кумир 2.х

А.Г.Кушниренко[1], М. А. Ройтберг[2], Д.В.Хачко, В. В. Яковлев[1]

1- кандидат физико-математических наук, 2 - доктор физико-математических наук.

Аннотация. Кумир 2.х – система программирования, предназначенная для поддержки курса программирования в начальной и средней школе – является развитием системы Кумир, созданной в России во второй половине 1980-х годов и использует школьный алгоритмический язык – алголоподобный язык с русской лексикой и встроенными командами управления программными исполнителями. Отличительной особенностями Кумир 2.х являются возможность использования практикумов и существенно возросшая по сравнению с предыдущими версиями скорость выполнения программ.

Ключевые слова: алгоритмический язык, система, программа.

1. Введение

В конце 1980-х годов были разработаны как система КУМИР, предназначенная для работы на всех доступных в то время архитектурах персональных и школьных ЭВМ, так и методические материалы, необходимые для поддержки курса школьной информатики. Ввиду быстрого развития компьютерной техники, к началу 2000-х готов встала задача создания новой версии системы Кумир, которая должна была работать под управлением операционных систем семейств Windows и Linux и при этом обладать всеми возможностями исходной системы Кумир. Эта задача была, в основном, решена к 2008 г., и на конференции «Свободное программное обеспечение в высшей школе» [1] была анонсирована система Кумир 1, предназначенная для работы в Windows и Linux.

В ходе разработки и последующего взаимодействия с педагогическим сообществом были выявлены направления дальнейшего совершенствования системы. Во-первых, стало ясно, что наибольшие нарекания пользователей вызывает низкая скорость выполнения КуМир-программ, обусловленная их интерпретацией во время выполнения. Во вторых, выяснилась необходимость расширения набора включенных в систему исполнителей. В третьих, оказалась успешной так называемая методика практикумов, позволяющая педагогу снабдить каждое задание на программирование средствами автоматизированной проверки правильности выполнения этого задания. С учетом накопленного опыта и возникновения новых потребностей, система Кумир была перепроектирована и переписана для решения следующих задач:

1. Возможность простого расширения функциональности системы путем реализации самодостаточных модулей.
2. Возможность создания на базе системы Кумир различных конфигураций, адаптированных под специфичные применения, например, для проведения олимпиад.
3. Радикальное увеличение скорости выполнения программ.
4. Поддержка системы практикумов с возможностью самопроверки (для учеников) и отслеживания хода работы учеников (для учителей).

Результатом работы является платформа Кумир 2.х [2], на базе которой создан комплект из компилятора-выполнителя (версия 2.0) и полнофункциональной версия обучающей системы Кумир (версия 2.1; в настоящее время комплект находится в стадии бета-тестирования).

2. Архитектура системы версий 2.х

Платформа Кумир 2.х представляет собой набор динамически загружаемых программных модулей, реализованных в виде DLL- или SO-библиотек, и небольшой программы для запуска.

Список используемых модулей и параметры их инициализации определяются конфигурацией запуска, которая может быть, как указана в явном виде, через аргументы командной строки, так и статически на этапе компиляции. На этапе компиляции системы из исходных текстов создается несколько типовых конфигураций запуска:

1. Система для поддержки учебника [3];
2. Система для поддержки учебников [4];
3. Система, предназначенная для подготовки заданий учителем;
4. Отдельные инструменты командной строки, предназначенные для работы в пакетном режиме.

Каждый модуль системы реализует один или несколько интерфейсов взаимодействия, а зависимости между модулями построены на использовании интерфейсов, а не реализаций. Это позволяет создавать экспериментальные или специфичные версии программных продуктов, основанных на платформе Кумир 2.х, с помощью добавления новых модулей и новых конфигураций запуска.

В частности, таким образом реализованы два варианта компилятора: один из них компилирует программу в выполнимый байт-код, а другой (в настоящее время имеет статус экспериментального) – в выполняемый машинный код.

Другим примером построения произвольной конфигурации является разрабатываемая среда для языка Python, которая использует ровно те же модули, что и среда для языка Кумир, но с заменой языковых моду-

лей Кумир на модуль языка Python. Аналогичным образом можно создавать среды для разработки на других языках программирования, которые будут иметь общие с системой Кумир пользовательский интерфейс и особенности работы.

3. Быстродействие программ

Платформа Кумир 2.x содержит два варианта реализации поддержки школьного алгоритмического языка:

1. Анализатор программ, генератор компактного байт-кода, и выполнитель полученного байт-кода. Этот вариант реализован в конфигурациях с пользовательским интерфейсом, и позволяет получить полный контроль над выполнением программы.

2. Анализатор программ, точно такой же, как в варианте использования байт-кода, и генератор биткода LLVM [5], который затем транслируется в машинный код для непосредственного выполнения на процессоре. Этот вариант в настоящее время имеет реализацию, разработанную, в первую очередь, для системы Linux. Имеется экспериментальная поддержка версии для Windows версии не ниже Vista, поскольку для использования LLVM требуется компилятор не старее Visual Studio 2012.

3.1. Выполнение с использованием байт-кода

В отличии от предыдущих (1.x) версий системы Кумир, которые интерпретировали каждую выполняемую строку кода, в новой версии выполняется трансляция исходной программы в набор простейших команд – байт-код. Размер каждой команды – фиксированный, равный 4-м байтам, что позволяет реализовать эффективный стековый выполнитель.

Набор команд байт-кода покрывает необходимый и достаточный функциональный минимум для выполнения на виртуальной машине, состоящей из арифметико-логического блока, стековой памяти и памяти-кучи.

Помимо базового функционального набора команд, предусмотрены команды для управления работой самой виртуальной машины. Они используются при создании «отладочных» версий программ, которые сообщают номера строк, осуществляют вывод значений на поля и т. д. Таким образом, при выполнении программы без показа на полях, и при выполнении программы по шагам, выполняется разный байт-код; в первом случае, выполняется более короткая программа, которая не содержит инструкций, предназначенных для управления самой виртуальной машиной.

3.2. Генерация биткода LLVM

Более радикального ускорения программ можно добиться за счет компиляции в машинный код либо перед выполнением программы, либо во время ее выполнения.

Оба механизма можно реализовать с помощью набора библиотек LLVM, которые также используются для генерации машинного кода в реализации CLang компилятора Си/Си++, и эталонной реализации языка Rust.

Поставка Кумир 2.1.0 для Linux и экспериментальная сборка MSVC2012 для Windows включаются в себя консольный компилятор kumir2-llvmc, который принимает на вход программу на школьном алгоритмическом языке, а результатом его работы является самая обычная программа для запуска без использования виртуальной машины.

Ранее мы уже экспериментировали [7] с генерацией машинного кода, используя язык Си в качестве промежуточного представления. Тогда мы столкнулись с проблемой медленной работы компилятора GCC (а особенно – MinGW, его реализации для Windows). Для ускорения процесса компиляции было принято решение отказаться от использования GCC в пользу CLang для генерации кода, который во-первых компилирует программы заметно быстрее, а во-вторых может быть использован в виде библиотеки, а не отдельно запускаемого процесса. Поскольку программный интерфейс CLang оказался тесно связанным с LLVM, то оказалось более удобным использовать в качестве промежуточного языка при генерации машинного кода именно LLVM, а не Си.

3.3. Эквивалентность выполнения байт-кода и программ в машинном коде

Имея две различные реализации выполнения: байт-код в конфигурации на компьютере пользователя, и биткод LLVM, например, в системе автоматизированного тестирования программ, необходимо гарантировать эквивалентность результатов выполнения одних и тех же исходных программ. Для решения этой задачи, помимо стандартной библиотеки алгоритмического языка, реализованной на Си++, в отдельный модуль были вынесены некоторые части виртуальной машины, выполняющей байт-код. Все это оформлено в виде программного модуля, который один раз компилируется в биткод LLVM, и в дальнейшем связывается с результатом компиляции пользовательских программ. Это позволяет гарантировать одинаковое выполнение программ, что особенно критично при выполнении учебных программ, содержащих ошибки выполнения.

Недостатком такого подхода является нерациональное использование системных ресурсов: для большинства простейших операций приходится вызывать через стек внешние процедуры, изначально предназначенные для реализации виртуальной машины. Эти процедуры не только выполняют требуемые операции, но и осуществляют контроль корректности их выполнения.

3.4. Сравнение производительности Кумир-LLVM с другими языками программирования

Методика оценки. Были реализованы три алгоритма, которые могут встречаться в рамках олимпиадных заданий:

1. Алгоритм Флойда-Уоршелла для оценки скорости работы с массивами;

2. Алгоритм сортировки слиянием для оценки скорости вызова подпрограмм;

3. Вычисление коэффициентов ряда Фурье для оценки скорости выполнения простых программ и операций с плавающей точкой.

Для языка Паскаль использовалась реализация

FreePascal, опции компилятора '-Cr -Co -Ct'. Для языка Си использовался компилятор CLang с с опцией '-O0'. Python – стандартная реализация версии 2.7.

За единицу измерения было взято время выполнения программы на языке Си, скомпилированной без оптимизации компилятором, поскольку машинный (точнее – ассемблерный) код именно этой реализации является легко интерпретируемым для дальнейшего анализа производительности.

Результаты приведены на рис.1. Как видно, в задачах с частым обращением к элементам массива, программы на языке Кумир на порядок медленнее, чем на Си или Паскале. Это связано, в первую очередь, с проверками на определенность элемента массива, которой нет в других компилируемых языках программирования. Отставание Кумир от Си и Паскаля в задаче на рекурсивный вызов алгоритма связан с использованием «вариантных» типов переменных, заимствованных из реализации виртуальной машины (причины использования см. выше). Аналогичная ситуация наблюдается и в простых вычислительных задах.

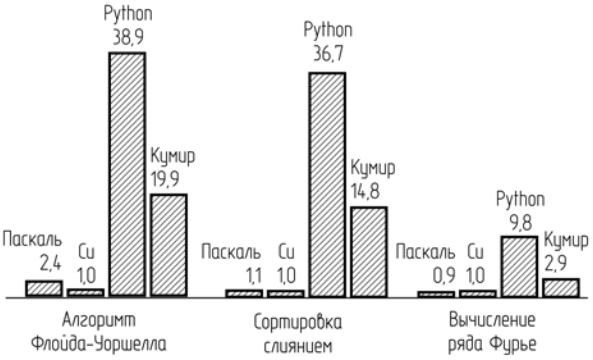

Рис. 1. Сравнение времени выполнения программ на различных языках программирования. За единицу измерения принято время выполнения эквивалентной программы на языке Си

Теоретически возможная скорость работы программ – на уровне языка Паскаль, так что, при необходимости, скорость работы может быть улучшена. В настоящее время это не является для нас первоочередной задачей, поскольку допустимым языком программирования при проведении олимпиад или при выполнении вычислительных задач, как правило может быть и Python. Как видно, реализация Кумир-LLVM является, несмотря на отсутствие каких-либо оптимизаций, в два раза более высокопроизводительной.

4. Система с точки зрения пользователя

4.1. Пользовательский интерфейс

Пользовательский интерфейс системы, в целом, остался очень похожим на интерфейс предыдущих версий системы «Кумир»: центральное место занимает редактор с полями для отображения ошибок и промежуточных значений переменных, а также область ввода-вывода.

В процессе взаимодействия с сообществом пользователей, а также по результатам опытных эксплуатаций, были проведены улучшения интерфейса для улучшения эргономики и адаптации к реальным условиям.

В частности, более гибкой стала система управления вспомогательными окнами: в Кумир 2.1 появилась возможность прикрепления (см. рис. 2) окна исполнителя, или любого другого вспомогательного окна, в фиксированную область главного окна без перекрытия. Это оказалось необходимым для работы в компьютерных классах с ноутбуками, у которых диагональ экрана не превышает 15 дюймов.

В то же время, остается возможность произвольного размещения вспомогательных окон, если используется большой экран, или экран совместно с проектором. Переключение режима отображения окна осуществляется одной кнопкой в заголовке. Для реализации этой возможности пришлось отказаться от использования стандартного оформления окон операционной системой, и реализовать немного не стандартный вид заголовка и кнопок.

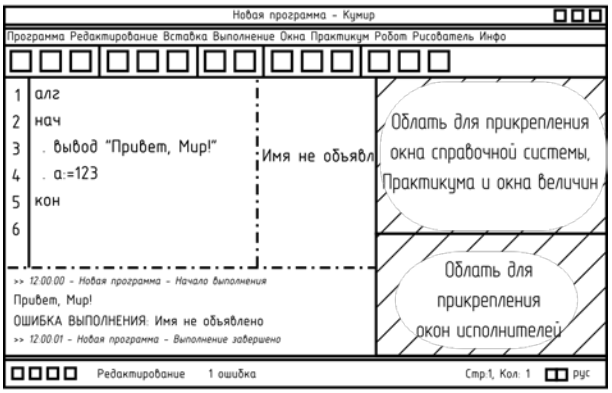

Рис. 2. Пользовательский интерфейс системы, адаптированный для экранов ноутбуков

Не обошли стороной и пользователей с ограниченными физическими возможностями. Система Кумир следует настройкам операционной системы в случае использования высококонтрастных тем: она может инвертировать цвета редактора при использовании темного фона. Значки на панели инструментов реализованы монохромными, и их цвет подстраивается во время запуска под настройки операционной системы.

Помимо удобства использования системы слабовидящими пользователями, мы стараемся сделать интерфейс Кумира доступным и для людей с нарушениями моторики: работать с системой, в большинстве ситуаций, можно и без мыши, используя только клавиатуру.

4.2. Практикум

Модуль поддержки практикумов в системе Кумир 2.x наследует все возможности аналогичного модуля, реализованного в системе Кумир 1.9. Практикум содержит набор заданий, имеющий древовидную структуру (разбитый на разделы, подразделы и т.д.); в каж-

дом задании ученик должен написать программу. Задание представляет собой заготовку, которая включает заголовок нужной программы, необходимые комментарии и, возможно, отдельные элементы программы; ученик не может изменять строки заготовки, он может только дополнить их своими строками.

Подготовив программу – решение задания, ученик может проверить ее, используя встроенную систему тестов; по результатам проверки за задание выставляется оценка. Все программы, отправлявшиеся на проверку, сохраняются и (при желании) доступны для последующей проверки учителем.

Для подготовки заданий практикумов используется специальная учительская конфигурация системы; создание своего практикума и редактирование уже готовых практикумов, как показывает опыт, доступно квалифицированному учителю [8].

4.3. Инструменты для пакетной обработки

Важной особенностью новой платформы является наличие конфигураций, предназначенных для выполнения на сервере без использования графического интерфейса пользователя. Реализованы консольные инструменты: компилятор в байт-код, соответствующий ему выполнитель байт-кода, а также компилятор в машинный код. Эти инструменты являются необходимыми для организации автоматизированного тестирования программ. В отличии от ранней реализации Кумир для КТС-ЕГЭ, данные инструменты не являются специально подготовленной версией, а используют ровно те же модули системы, что и версия с пользовательским интерфейсом.

5. Дальнейшие направления разработки

5.1. Упрощение разработки новых исполнителей

Сообщество пользователей системы Кумир – это не только учителя информатики, но и разработчики оригинальных методик и собственных программных разработок [9]. Мы хотим дать им возможность самостоятельной реализации необходимых им исполнителей. Исходные тексты платформы Кумир 2.x открыты; они, в частности, содержат вспомогательный скрипт и документацию разработчика, облегчающие создание новых исполнителей. В дальнейшем планируется создание специализированной поставки (SDK), ориентированной на разработчиков дополнений.

5.2. Использование платформы для реализации сред на других языках программирования

Модульная архитектура платформы Кумир 2.x позволяет создавать решения на базе существующих компонент. В частности, мы разрабатываем аналогичную систему для языка программирования Python 3.x. Как и система Кумир для школьного алгоритмического языка, среда для Python на базе Кумир ориентирована на обучение программированию, а не на профессиональное программирование. Эта среда использует ряд компонентов системы Кумир: модуль пользовательского интерфейса, систему поддержки практикумов, и может даже использовать «исполнители» системы Кумир (правда, в случае с Python, это не для всех исполнителей имеет смысл из-за отличающейся методики обучения).

Помимо реализации среды для Python, мы экспериментировали с созданием аналогичной среды для языка Паскаль.

5.3. Поддержка курса робототехники

У нас есть опыт в программировании с помощью системы Кумир 1.9 обучающего робота Lego NXT [10]. Эксплуатация этого робота подразумевает использование канала радиосвязи Bluetooth для телеметрии, при этом сама программа выполняется на компьютере. Опыт показал, что такое взаимодействие накладывает ограничения на возможности использования из-за временных задержек радиоканала. Например, реализация алгоритма обхода лабиринта по правилу правой руки надежно работает только в том случае если робот двигается медленно и осторожно, иначе он просто не успевает обмениваться данными с компьютером и допускает неточности при поворотах.

Решением проблемы с задержками в радиосвязи является перенос выполнения программы со стороны компьютера на сторону микроконтроллера, установленного в роботе. Недавнее обновление линейки Lego NXT – модель Lego EV3 обладает достаточными ресурсами для выполнения интерпретатора байт-кода системы Кумир 2.x, а сам байт-код является более компактным, чем выполняемый код, поэтому его проще передавать по радиоканалу.

Не имея возможности эксплуатации Lego EV3, мы обеспечили работоспособность интерпретатора байт-кода Кумир на одноплатном модуле Raspberry Pi, который основан на процессоре с ARM-архитектурой и обладает скромными техническими характеристиками.

В дальнейшем мы планируем портировать интерпретатор для выполнения на процессоре Lego EV3 и реализовать связь интерпретатора с компьютером.

5.4. Перемещение в «облако»

Современные тенденции в программном обеспечении ведут к тому, что скоро на персональных компьютерах останется только одна программа – браузер, а привычные нам приложения станут Web-приложениями. Сегодня в виде Web-приложений существуют не только офисные приложения (Google Docs, Microsoft Office 365 и др.), но даже графические редакторы и CAD-системы. Целесообразно создать Web-версию системы Кумир. Это достаточно трудоемкий, но необходимый процесс, который предстоит нам в недалеком будущем.

System KUMIR 2.X

A.G. Kushnirenko, M.A. Roytberg, D.V. Hachko, V.V. Yakovlev

Abstract: KUMIR 2.x – is a programming environment designed to support the course of programming in primary and secondary schools. It is a reincarnation and development of KUMIR, created in the second half of the 1980s and uses the so-called russian algorithmic language – Algol-like language with Russian vocabulary. Otherness KUMIR 2.x is the support of automatic check of students programm correctness (both syntactic and semantic) and significantly increased (compared to previous versions) speed of execution.

Keywords: Kumir, algorithmic language, system, program.

Литература

1. А. Г. Кушниренко, А. Г. Леонов, М. А. Ройтберг, Д. В. Хачко, В. В. Яковлев, А. В. Карпов, Н. М. Субоч. «Система программирования КУМИР – интегрированная поддержка начальных курсов информатики и программирования». III Конференция «Свободное программное обеспечение в высшей школе», М.: AltLinux, 2008.
2. https://gitlab.com/niisi/kumir2
3. Информатика: 7–9 кл.: Учеб. для общеобразоват. учр. А. Г. Кушниренко, Г. В. Лебедев, Я. Н. Зайдельман. М.: Дрофа, 2003. 335 с.
4. Информатика. 10 класс. Углубленный уровень. В 2 частях (комплект из 2 книг). К. Ю. Поляков, Е. А. Еремин. М.: Бином, 2014. 648 с.
5. "Introduction to the LLVM Compiler System" Chris Lattner Plenary Talk, ACAT 2008: Advanced Computing and Analysis Techniques in Physics Research, Erice, Sicily, Italy, Nov. 2008.
7. М. А. Ройтберг, В. В. Яковлев. «Конвертор КУМИР – C++: поддержка перехода от учебных к профессиональным системам программирования». III Конференция «Свободное программное обеспечение в высшей школе», М.: AltLinux, 2008
8. http://kpolyakov.spb.ru/school/kumir.htm
9. http://www.vinforika.ru/index.php/mnu-umki
10. http://www.lego.com/ru-ru/mindstorms/