ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

УДК 004.4



В.В. Баранов
Российская академия народного хозяйства
и государственной службы при Президенте РФ
А.А. Звонарев
ООО «Яндекс»
К. Чжао
Московский государственный
технологический университет «СТАНКИН»

СИСТЕМА РАЗРАБОТКИ ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ВЫСОКОГО УРОВНЯ

В статье представлены результаты разработки системы прикладного программного обеспечения, использующегося при формировании высокотехнологичных производственных систем. Предложено создание механизма цепочек вызовов и синтаксических имен с целью упрощения читаемости кода. Разработана стандартная библиотека исходных кодов, имеющая разноуровневую абстракцию алгоритмов. Выполнена реализация интегрированной среды разработки АльфаКод, включающей управляемую подсветку синтаксиса и поддержку систем контроля версий.

Система прикладного программного обеспечения, язык программирования, статическая типизация, динамическая типизация, интегрированная среда разработки АльфаКод.

В условиях обострения конкуренции на рынках наукоемкой продукции актуальной задачей становится переход предприятий на принципы цифрового производства, включая информатизацию и формирование высокотехнологичных роботизированных структур. Это диктует необходимость создания качественно новой системы разработки прикладного программного обеспечения. Создаваемая система разработки программного обеспечения должна содержать инструментарий, способный помочь разработчику реализовать новый программный продукт и иметь удобный для пользователя интерфейс, библиотеку и синтаксис. В данный момент основную нишу среди прикладного программного обеспечения занимают системы, разработка в которых ведется на языках с синтаксисом, произошедшим от синтаксиса языка программирования С.

Однако использование на практике подобных систем сопряжено с рядом проблем, появление которых снижает эффективность системы. Это большой порог вхождения в язык, малое количество шаблонных решений в библиотеках, отсутствие поддержки современных методов разработки и т. д. Поэтому актуальным становится пересмотр ключевых особенностей компонентов системы разработки.

В этой связи необходим анализ современных подходов к разработке прикладного программного обеспечения, проектирование синтаксиса и разработка компилятора языка программирования. Кроме того, необходимы библиотека шаблонных решений и прототип интегрированной системы разработки.

1. Факторы, влияющие на выбор языка программирования. Основной частью большинства современных языков программирования является типизация, охватывающая несколько пересекающихся категорий. Статическая типизация предполагает, что конечные типы значений переменных, а также аргументов и возвращаемых значений функций, определяются на этапе компиляции. Подобная типизация используется в языках C++, C#, Java. Динамическая типизация предполагает, что тип значения переменной определяется на этапе выполнения программы, что характерно для языка Python [3].

Сильная (строгая) типизация запрещает смешение в выражениях различных типов данных. Кроме того, при использовании этого вида типизации не выполняется неявных преобразований, что характерно для таких языков, как C#, Java, Python. При слабой (нестрогой) типизации возможно выполнение неявных преобразований типов данных, что зачастую приводит к потере точности. Данная типизация используется в языке C++ [1]. Явная типизация, использующаяся в языках C++, C#, Java, предполагает явное указание в языке типа данных. При использовании неявной типизации, что характерно для языка Python, определение типа выполняет компилятор или интерпретатор [3].

Подход к разработке языка программирования нами был выбран на основе анализа различных типизаций. Так, например, статическая типизация, позволяя производить проверку типов один раз на этапе компиляции, ускоряет выполнение программы, осуществляет поиск потенциальных ошибок еще на этапе компиляции и обеспечивает тесную связь с интегрированной системой разработки. При потере данных произойдет ошибка компиляции или исключение, вместо тихой работы с неправильным поведением. Поэтому в такой ситуации целесообразно использовать сильную типизацию. Явная типизация дает возможность ограничить алгоритм функции по типу. При условии, что один алгоритм работает только с числами, а другой только со строками, используя явную типизацию их можно разделить. Поэтому предлагаемый нами язык программирования будет представ-

лять собой статически сильно явно типизированный язык программирования. При этом учтено, что каждый типизированный язык программирования должен иметь набор базовых типов данных, на основе которых программист может построить свои (пользовательские) типы данных.

2. Особенности разработки прикладного программного обеспечения с использованием различных языков программирования. Современное представление языков программирования характеризуется избыточным документированием кода. Ключевые слова, подчеркивающие тот или иной элемент кода, раньше выполнявшие роль разметки, в настоящее время являются лишними. Такие ключевые слова, как class, def, func, function, let, var перестали быть необходимостью при решении различных задач. Использование code style guide позволило применять его в подчеркивании синтаксических конструкций (например, имена пользовательским типам могут быть даны в PascalCase, а имена переменным и функциям – в camelCase). Подобные решения дают возможность снизить менеджерскую синтаксическую нагрузку на код [2]. В языке Python код, имея одинаковое семантическое предназначение, но без использования лишней «разметки» короче на 30% [2].

Часто при разработке программы приходится работать с коллекциями элементов. Это диктует необходимость писать функции обработки элемента коллекции и в дополнение создавать функцию обработки самой коллекции. Подобная функция может быть представлена как простой цикл. Вследствие этого возникает необходимость разработки концепции синтаксических имен.

Язык Python имеет синтаксическую конструкцию «генератор списков», что упрощает выполнение данной задачи. Язык C++ не обладает такими возможностями. Поэтому, применяя этот язык, необходимо использовать прямой метод. Язык Лего имеет синтаксическую конструкцию «синтаксические имена», что минимизирует использование семантически идентичных кусков кода, которые существенно замедляют чтение чужого кода.

В ряде языков, например в языке C++, у переменных и типов аргументов нет указания типов, а у функций – возвращаемых значений. Это связано с тем, что все функции по умолчанию рассматриваются как шаблонные, выводящие типы аргументов и возвращаемого значения на этапе компиляции. Поэтому, исходя из опыта использования динамически типизированных языков программирования, можно рекомендовать создание алгоритма, который будет инвариантен по отношению к типу данных. В статически типизированных языках к использованию шаблонов прибегают лишь в исключительных случаях. Использование шаблонов накладывает существенные ограничения на доработку кода в будущем, поскольку шаблонная функция может принимать всегда один и тот же тип, а нешаблонная функция всегда относится к определенному типу и никогда не может быть функцией другого типа.

Указать тип аргументов функции можно двумя способами. Во-первых, указав полный тип (Int, Shrt, Str, Vctr<Int>), а во-вторых, – частичный тип (Vctr, Container, Integer, String). В первом случае указывается конкретный тип значения аргумента или возвращаемого значения. Во втором случае указывается лишь часть информации о типе. Это может быть либо семейство Integer (целые, т.е. Int, UInt, Shrt, Long, BInt и т. д.), либо не специализированный шаблонный тип (например, Vctr, List).

В случае использования шаблонных типов иногда появляется необходимость доступа к косвенному имени типа. Например, для создания временной переменной внутри тела функции того же типа, что и аргумент функции, возможно использовать специальные имена шаблонных типов. Эти имена характеризуются теми же правилами составления, что и обычные типы в языке. Особенностью является то, что подобные имена должны начинаться с *T_*. Это повышает читаемость кода. Отсутствие указания типа у переменных объясняется автоматическим выводом по выражению справа. Если в этом выражении присутствует целое число, то Int, если присутствует дробное значение, то Dbl и т. д. Если необходимо заменить Int на Shrt, то используется конструктор num = Shrt(0). Тогда типом значения переменной num будет Shrt. Переменные с момента их объявления и до момента уничтожения характеризуются одним типом.

3. Разработка языка программирования. Одной из важнейших задач, решаемых в процессе разработки программного обеспечения, является разработка компилятора языка программирования. Компилятор необходим для преобразования исходного текста программы, написанного на языке программирования, в представление понятное исполнителю программы (компьютеру или виртуальной машине). Для достижения подобного синтаксиса может быть использован базовый синтаксический анализатор, который позволяет типизировать токены на этапе лексинга [5].

Для достижения синтаксиса «синтаксических имен» на этапе формирования модели целесообразно использовать процесс генерации исходного кода. Суть генерации заключается в поиске «синтаксических имен» и автоматическом генерированием кода функций. Поиск «синтаксических имен» представляет собой поиск суффиксов у имен вызываемых пользовательских функций.

Если в области видимости существует функция, определенная пользователем, с именем foo, и обеспечивается вызов функции с именем fooEach, во время построения модели будет обнаружено, что функция с именем fooEach не существует. В этом случае начнется поиск суффикса в названии функции. После завершения этого процесса будет найден суффикс, и на основе подготовленного для этого суффикса шаблона будет сгенерирована функция с именем fooEach.

Созданная система прикладного программного обеспечения охватывает стандартную библиотеку типов и функций, а также интегрированную среду разработки АльфаКод. В отличие от стандартной библиотеки языка C++, которая построена по принципу максимальной абстракции, в языке программирования Лего стандартная библиотека имеет разноуровневую абстракцию алгоритмов.

Таблица

Результаты реализации функций «сортировка выбором» (F1) и «умножение матриц» (F2) на различных языках программирования

Параметр		Язык программирования					
		C++		Python		Лего	
		F 1	F 2	F 1	F 2	F 1	F 2
1.	Количество символов	302	449	248	272	271	299
2.	Время работы, с	92,77	24,44	1218,6	185,1	93,8	25,6
3.	Используемый объем памяти	1,4 Гб	12,9 Мб	9,8 Гб	101 Мб	1,7 Гб	13,0 Мб

4. Формирование интегрированной среды разработки. Для формирования интегрированной среды разработки нами был выбран фреймворк Qt [4], на основе которого реализована совокупность мер поддержки, охватывающих, во-первых, управляемую подсветку синтаксиса, а во-вторых, поддержку систем контроля версий. Интегрированная среда разработки, охватывающая комплекс программных средств разработки прикладного программного обеспечения, в предложенной нами системе функционирует при работе с проектом как с обычной папкой, предлагая возможность сохранить настройки как в папке проекта, так и в папке пользователя. В предложенной нами интегрированной среде разработки имеется возможность создания и открытия различных проектов.

В системе реализована возможность добавления в проект ссылочных объектов. Данный функционал полезен при разработке модулей к существующим приложениям. Это дает возможность использовать статический анализ кода и осуществлять построение деревьев зависимостей с учетом стороннего кода, а также видеть его в папке проекта. Реализована функция предупреждения об удалении с выбором в диалоговом окне, а также закрытии несохраненных изменений с выбором и просмотром изменений в формате diff. Достоинствами предложенных нами решений являются возможности интегрированной среды разработки в сфере реализации управления системой контроля версий из интерфейса среды, а также возможности работы с несколькими вкладками, просмотра структуры проекта и файла.

5. Сравнение предложенных решений с существующими разработками. Сравнение по параметру функциональности языков проводилось методом тестирования предложенных решений с существующими разработками. Результаты реализации функций «сортировка выбором» (F1) и «умножение матриц» (F2) на различных языках программирования приведены в таблице.

Анализ результатов, приведенных в таблице, показывает, что предложенный язык позволяет по сравнению с существующими языками повысить эффективность процесса разработки программного обеспечения. Причем это достигается без кардинальных изменений парадигмы разработки.

В статье предложена система разработки прикладного программного обеспечения, состоящая из языка программирования Лего, транслятора из языка Лего в язык C++, стандартной библиотеки и интегрированной среды разработки АльфаКод. Система отличается легкостью восприятия, низким порогом вхождения, не загруженным избыточной семантикой синтаксисом. Рекомендуемый для создания системы язык позволяет без кардинальных изменений парадигмы разработки повысить эффективность процесса создания программного обеспечения по сравнению с существующими языками.

Уникальность подхода к представлению синтаксиса обеспечивает высокий уровень компактификации файлов, уменьшая размер файла и увеличивая скорость разработки решений. Стандартная библиотека представлена в виде дерева абстракции, а новая модель типизации обеспечивает баланс между статической и динамической типизацией. Проведенное тестирование системы показало высокую эффективность ее применения. Предложенный подход к представлению синтаксиса, позволяет достичь высокого уровня (15–69%) компактификации файлов. Это уменьшает размер файла, вследствие чего происходит увеличение скорости разработки решений.

Литература

1. Страуструп, Б. Дизайн и эволюция языка C++ / Б. Страуструп. – Москва: ДМК Пресс, 2014. – 448 с.

2. Таненбаум, Э. Современные операционные системы / Э. Таненбаум. – 4-е изд. – Санкт-Петербург: Питер, 2015. – 1120 с.

3. Бердоносов, В. Д. Применение ТРИЗ-эволюционного подхода к исследованию объектно-ориентированных языков программирования / В. Д. Бердоносов, А. А. Животова. – Комсомольск-на-Амуре: КнАГТУ, 2014. – 116 с.

4. Вартанов, С. П. Применение статической инструментации байт-кода языка Java для динамического анализа программ / С. П. Вартанов, М. К. Ермаков // Труды ИСП РАН. – 2015. – Т. 1. – С. 25–38.

5. Рассел, С. Искусственный интеллект. Современный подход / С. Рассел, П. Норвиг. – Москва: Вильямс, 2007. – 1408 с.

V.V. Baranov
Russian Academy of National Economy and Public Administration
under the President of the Russian Federation
A.A. Zvonarev
Yandex LLC
K. Chzhao
Moscow State Technological University "STANKIN"

DEVELOPMENT SYSTEM OF APPLIED SOFTWARE USING A HIGH-LEVEL LANGUAGE

The article presents the results of the development of an applied software system used in the formation of high-tech production systems. It is proposed to create a mechanism for call chains and syntactic names in order to simplify the readability of the code. A standard source code library has been developed; it has a multi-level abstraction of algorithms. The AlphaCode integrated development environment including controlled syntax highlighting and support for version control systems has been implemented.

Applied software system; programming language; static typing; dynamic typing; integrated development environment AlfaCode.