Гошев Владимир Андреевич,
Косовский Николай Кириллович

УДК 519.682.2

ПРОЕКТ ЯЗЫКА ПРОГРАММИРОВАНИЯ РЕФАЛ-5Е С УДОБНЫМИ РАСШИРЕНИЯМИ ПРЕПРОЦЕССОРОМ

Аннотация

Язык программирования рефал-5 позволяет решать некоторые важные задачи программирования короче и эффективнее, чем другие языки программирования. Однако существующие реализации диалектов языка программирования рефал не имеют возможности вызова программой встроенного интерпретатора и простого способа подключения внешних библиотек кода. Для устранения этих недостатков было решено разработать новый диалект языка программирования рефал – рефал-5е. Язык рефал-5е поддерживает возможность вызова из программы встроенного интерпретатора, а также добавлена возможность подключения внешних библиотек кода. Также язык программирования рефал-5е позволяет функции возвращать такие значения как «Успех» и «Неуспех», что позволяет расширить возможности использования возврата, как в языке программирования пролог.

Ключевые слова: рефал-5, трансляция, интерпретация, рефал-5е машина, библиотеки кода, встроенный интерпретатор.

ВВЕДЕНИЕ

Язык рефал (РЕкурсивных Функций АЛгоритмический) – один из старейших языков программирования, ориентированный на «символьные преобразования»: обработку термов, то есть планарно представленных древовидных структур из символьных строк (например осуществление алгебраических преобразований); перевод с одного языка (искусственного или естественного) на другой; решения задач, связанных с искусственным интеллектом. Этот язык программирования соединяет в себе математическую простоту с практической ориентацией на написание больших и сложных программ [2]. Отличительной чертой языка является использование сопоставления с параметрическим образцом и возможность переписывания термов в другие термы как основного способа определения функций. При этом переписывание термов осуществляется посредством замены простейших функциональных термов на произвольные термы.

На данный момент наиболее известны следующие диалекты языка рефал: базисный рефал, рефал-2 [8], рефал-5 [23], рефал-6 [7] и рефал+ [6]. Рефал+ является наиболее развитым из них, но в то же время рефал-5 имеет более простой синтаксис и предоставляет достаточно средств для эффективного решения большинства задач программирования.

В современном мире необходимы языки, поддерживающие эффективные технологии. Однако совершенствование вышеперечисленных диалектов языка рефал на данный момент не ведется. Как следствие, ни в одном из них нет поддержки новых современных эффективных технологий, присутствующих в компьютерах. Так, сайт наиболее современного диалекта языка рефал+ [12] обновлялся 17 декабря 2011 г., а существенных изменений не было с 31 июля 2009 г.

В языке рефал-5 отсутствуют такие конструкции языка, как вызов из программы встроенного интерпретатора для исполнения динамически создаваемого кода, анонимные функции, функции высшего порядка. Также во всех существующих реализациях диалектов языка рефал недостаточно развито взаимодействие с другими языками программирования. Возможность взаимодействия с библиотеками кода, написанных на другом языке программирования, необходимо для любого эффективного языка программирования. Наиболее распространенным способом использования уже написанного кода (тем же самым или другим программистом), в том числе и на других языках программирования, являются библиотеки кода. В разных операционных системах они называются по-разному. Например, в семействе операционных систем Microsoft Windows они называются динамически подключаемыми библиотеками, а в Unix-like системах – это «общие объекты» (англ. shared objects). Доступ к библиотекам крайне необходим, так как без него придется реализовывать множество уже доступных решений заново. По этой причине весьма полезно разработать и реализовать удобный способ простого взаимодействия с внешними библиотеками кода.

Сейчас необходимы диалекты языков, использующие новые технические возможности. Выполнение программы в несколько потоков актуально, в силу того что большинство современных компьютерных систем являются многопроцессорными. Сами принципы языка рефал способствуют построению программ так, чтобы их было возможно выполнять в несколько потоков без вмешательства разработчика, поскольку большинство функций в языке рефал не создают побочных эффектов, результат их выполнения зависит только от переданных аргументов.

Язык программирования рефал-5 позволяет решать некоторые важные задачи программирования красивее и лаконичнее, чем другие языки. В то же время разработчику, для того чтобы начать разрабатывать программы на этом языке, придется потратить достаточно много времени и основательных интеллектуальных усилий для того, чтобы изучить этот язык и понять его парадигму, включающую функциональное и рекурсивно-логическое программирование. Примером функции, которую значительно проще реализовать на языке рефал – проверка, является ли строка палиндромом, то есть читается ли она слева направо так же, как и справа налево. На языке программирования рефал-5 данная функция может выглядеть следующим образом, где с символа «*» начинаются комментарии (см. листинг 1).

В то же время на языке С данная функция может выглядеть следующим образом (см. листинг 2).

Видно, что программа на языке С использует излишнюю большую детальность от программиста, чем программа на языке рефал-5, что может повлечь за собой трудно находимые ошибки.

ЦЕЛИ И ЗАДАЧИ

Для устранения описанных во введении проблем в существующих реализациях диалектов языка программирования рефал было решено разработать новый диалект – рефал-5е. В качестве диалекта для расширения был выбран диалект рефал-5. Язык рефал-5 и его синтаксис описан в работе В.Ф. Турчина [23]. Планируется, что язык рефал-5е будет удовлетворять следующим требованиям:

- Максимальная совместимость синтаксиса диалекта с синтаксисом языка программирования рефал-5 для упрощения обучения языку рефал-5е, а также упрощению переноса готовых приложений на новый диалект языка рефал.
- Реализация встроенного интерпретатора и возможность его вызова из програм-

Листинг 1

```
IsPalyndrome {
* Объявление функции
     s.A e.B s.A = <IsPalyndrome e.B>;
* Указываем, что если первый
* и последний символы совпадают,
* то откинуть их и рекурсивно вызвать себя
     s.A = True;
     = True;
* Если остался один или ноль символов,
* значит, переданная строка – палиндром
     e.A = False;
* В противном случае строка – не палиндром
};
```

мы. Это позволит выполнять динамически генерируемый код на языке рефал-5е. Наличие встроенного интерпретатора также позволит легко реализовать такие полезные конструкции, как анонимные функции (это функции особого вида, которые не имеют собственного имени и объявляются на месте их использования), замыкания (это функции, окончательное объявление которых также происходит при выполнении программы и которые ссылаются на переменные в своем контексте), динамическая загрузка модулей, обновление программы «на лету», то есть обновление кода выполняющейся программы без необходимости ее перезапуска (такая возможность очень востребована в высоко нагруженных системах).

- Более полная поддержка модулей, возможность экспорта из модуля нескольких функций, как это реализовано, например, в языках Perl [16] или Java [17]. Это позволит создавать библиотеки кода, которые программист сможет легко использовать в своих будущих программах. Все системные функции будут находиться в модулях. Такая организация позволяет легче определить, для чего необходима та или иная системная функция.
- Возможность подключать внешние библиотеки кода при помощи специально написанных модулей и использовать их в рефал-5е программе. Существует множество библиотек, которые помогают решать те или иные задачи. Например, библиотеки Qt [20] и GTK [21] позволяют строить графический интерфейс программы, библиотека OpenGL [22] позволяет работать с 2D и 3D графикой, библиотека libcurl [19] позволяет работать с сетевыми протоколами HTTP и FTP. Возможность использовать эти и другие библиотеки может упростить решение многих практических задач программирования.
- Функции дополнительно смогут возвращать значения «Успех» и «Неуспех».

Листинг 2

```
int isPalyndrome(char *str) { /* Объявление функции */
    int slen = strlen(str); /* узнаем длину строки */
    int i = 0;
    for (i = 0; i < (slen-1)/2; i++) { /* с начала и до середины строки */
        if (str[i] != str[slen-1-i]) /* проверяем, если очередной символ
не совпадает с соответствующим ему с конца, то строка – не палиндром */
            return 0;
        }
    return 1; /*Все проверки прошли удачно, значит строка –палиндром*/
}
```

Если функция вернула «Неуспех», то рефал-5е машина производит переход к следующему выражению после разделителя «;», если же таких выражений нет, то вызывающая функция также возвращает «Неуспех». Это позволяет не только более удобно управлять программой, но и повышает дружественность языка рефал-5е к пользователю, поскольку программа заканчивает свое выполнение без аварийных сообщений.

- Функции будут иметь возможность создавать исключительные ситуации и обрабатывать их. В языке будет единый интерфейс работы с ошибками, например такими, как деление на ноль, неверные данные пользователя и другие.
- Управляемая поддержка многопоточности программ. Современные компьютеры, как правило, имеют не одно, а несколько ядер, что позволяет многопоточным программам эффективнее использовать ресурсы компьютера.
- Поддержка наиболее распространенных архитектур: Intel i386, Intel X86_64 [18], ARM [1]. Поддержка этих архитектур позволит запускать рефал-5е-машину, а как следствие, и рефал-5е программы на большинстве современных пользовательских компьютеров и серверов.
- Поддержка операционных систем Windows [10], Linux [14], FreeBSD [9].
- Транслятор планируется распространять под открытой лицензией. Наличие открытой лицензии позволит заинтересованным людям вносить свои изменения и исправления в транслятор и тем самым улучшать его. Это может ускорить рост сообщества пользователей языка.
- Планируется разработка утилиты для автоматической конвертации файлов исходного кода на языке рефал-5 в файлы исходного кода на языке рефал-5е, как это сделано для трансляции программ на языке программирования рефал-5 в рефал+ [4].

ОПИСАНИЕ НОВЫХ КОНСТРУКЦИЙ ЯЗЫКА

В языке рефал-5е программа всегда будет начинаться с функции Main без параметров. Имя функции **`Main`** вместо **`Go`** для языка рефал-5 выбрано для того, чтобы разработчикам, использовавшим такие языки как, например, C, C++ [11], было легче найти главную функцию программы, так как в этих языках главная функция называется **`Main`**. Для того чтобы указать транслятору, что функция должна быть экспортирована из модуля, то есть видна и другим модулям, используется ключевое слово **`$export`**. По причинам указанным выше, в языке рефал-5е нет необходимости в ключевом слове **`$ENTRY`**, поэтому оно не будет использоваться.

В языке программирования рефал-5е было решено использовать комментарии в стиле языка C, так как это наиболее распространенный формат комментариев, и для разработчика, плохо знакомого с языком рефал-5е, будет легко отличить комментарий от кода программы. Так, две косые черты: «**`//`**» указывают, что все от них и до конца строки является комментарием. Такие комментарии называются строчными. Для того, чтобы закомментировать блок кода необходимо перед ним поставить «**`/*`**», а после – «**`*/`**» – такой комментарий называется блочным.

Как было указано выше, в языке программирования рефал-5е планируется поддержка дополнительной возможности возврата функцией специального значения «Неуспех». «Неуспех» можно использовать для организации бэктрекинга и как значение «Ложь» булевой функции. Для того чтобы функция вернула «Неуспех», необходимо, чтобы ни одно из сопоставлений с образцом не прошло успешно, или же были не выполнены условия (where-конструкция в понятиях языка рефал-5). В языке программирования рефал-5 при выполнении таких условий транслятор завершал выполнение программы с ошибкой «REFAL ERROR: RECOGNITION IMPOSSIBLE». Если вызываемая функция вернула «Неуспех», то рефал-5е-машина производит переход к обработке следующего выражения после разделителя «;» (при этом надо помнить, что при выполнении правила рефал-5е машина не может вернуться левее, знака «**`=`**» в нем). Если же таких выражений нет, то вызывающая функция возвращает «Неуспех». Примеры использования возвращения значения

«Неуспех» продемонстрированы в следующем разделе.

Планируется также разрешить программисту выбирать способ сравнения объектного выражения с образцом, а именно указывать, в каком направлении будут последовательно увеличиваться длины переменных для выражений в образце: слева-направо, справа-налево или же в лексикографическом порядке. Это позволит программисту, разрабатывающему программу на языке рефал-5е, указывать наиболее оптимальный способ организации удлинения переменных e-типа. Это может повысить скорость работы программы, а в некоторых случаях позволит избежать ошибок в программе при сопоставлении с образцом.

МОДУЛИ

Модуль – это файл с программой на языке рефал-5е, который может содержать функцию **`Main`**. Каждая функция, которую необходимо экспортировать для использования в других модулях должна предваряться ключевым словом **`$export`**. Пример модуля приведен в листинге 3.

Модуль из этого примера содержит одну функцию – **`IsPalyndrome`** и экспортирует ее, то есть данную функцию можно будет использовать и в другом модуле, импортирующем этот модуль. Данная функция почти эквивалентна одноименной функции на языке рефал-5 из введения к этой статье. В данном примере отсутствует строка «**`e.A = False`**», так что в случае, если переданная строка не является палиндромом, то функция **`IsPalyndrome`** вернет «неуспех».

Для того чтобы импортировать модуль, используется ключевое слово **`$import`**. Использование функций импортированного модуля должно предваряться названием модуля и двумя двоеточиями. Для примера импорта модуля, являющегося и примером использования его функций, приведем программу, использующую вышеописанный модуль (листинг 4).

Листинг 3

```
$export IsPalyndrome {
     s.A e.B s.A = <IsPalyndrome e.B>;
     s.A = ;
     = ;
};
```

Листинг 4

```
$import Test; /* Импорт вышеописанного модуля */
$import SystemIO; /* Импорт стандартного модуля рефал-5е ввода-вывода*/
Main { /* Функция-точка входа в программу */
     = <PrintIsPaly <SystemIO::Arg 1>>;
     /* При запуске программы вызываем функцию PrintIsPaly и передаем
ей в качестве аргумента первый аргумент командной строки,
переданный пргорамме */
};

PrintIsPaly {
     e.A = {
          <Test::IsPalyndrome e.A> <SystemIO::PrintLn e.A 'is a palyndrome'>;
          /* Если функция Test::IsPalyndrome вернула успех, то сообщаем
об этом */
          <SystemIO::PrintLn e.A 'is not a palyndrome'>;
          /* В противном случае сообщаем. что переданная строка не
является палиндромом */
     };
};
```

ВСТРОЕННАЯ УНИВЕРСАЛЬНАЯ ФУНКЦИЯ КАК ВОЗМОЖНОСТЬ МНОГОКРАТНОЙ ПРЕПРОЦЕССИИ

Универсальная функция (как и универсальная машина Тьюринга) – это функция, которая может заменить собой вычисление любой функции и в качестве входных данных использует вод и все аргументы исполняемой функции. Получив на вход код функции и входные данные (аргументы функции), она возвращает результат вычисления по входным данным функции, код которой был дан на вход. По существу универсальная функция является транслятором интерпретирующего типа. Так как язык рефал является обобщением нормальных алгоритмов Маркова, то такую функцию можно реализовать и средствами языка, в работе [3] указан проект реализации такой функции для языка рефал-5 средствами самого языка. Но стоит заметить, что реализация универсальной функции средствами самого языка будет работать значительно медленнее, чем реализация средствами рефал-5е машины, так как рефал-5е машина будет компилировать переданный ей код и потом выполнять его как любую другую функцию. Реализация универсальной функции средствами языка заставляет рефал-5е машину выполнять код, который будет вычислять значение рефал-функции, являющейся аргументом. В этом случае происходит двойная интерпретация кода.

В языке программирования рефал-5е для универсальной функции планируется отдельный модуль – **`Eval`**. Модуль **`Eval`** экспортирует 2 функции:

- Функция **`Compile`** компилирует переданные ей аргументы в код функции рефал-5е машины и возвращает сгенерированное для нее имя. В результате такую функцию можно вызывать в программе неограниченное количество раз без необходимости заново производить трансляцию ее кода в код рефал-машины.
- Функция **`Run`** компилирует функцию из кода, переданного первым аргументом (обособленным структурными скобками) в код рефал-5е машины, и интерпретирует ее, передав скомпилированной функции аргументы для нее.

Наличие встроенного интерпретатора, позволяет также использовать в языке рефал-5е функции высшего порядка, анонимные функции и замыкания и использовать более привычный и свой для каждого программиста синтаксис, если написать программу претрансляции для такого синтаксиса с помощью встроенного интерпретатора.

ФУНКЦИИ ВЫСШЕГО ПОРЯДКА, АНОНИМНЫЕ ФУНКЦИИ И ЗАМЫКАНИЯ

Существует реализация рефала с функциями высшего порядка, которая названа ее авторами как рефал-7 [13]. Транслятор языка рефал-5е будет предполагать следующий более эффективный метод реализации функций высшего порядка. Имя функции является рефал-словом, поэтому для передачи функции как аргумента или возвращения функции как результата некоторой функции используется при помощи передачи имени функции.

Для того чтобы создать анонимную функцию, код этой функции необходимо поместить внутрь квадратных скобок, после чего рефал-5е машина присвоит вновь созданной функции уникальное имя для его дальнейшего использования, далее это имя функции можно записать в переменную, возвратить из функции или сразу выполнить именованную функцию. В листинге 5 приведен пример программы, которая создает и использует анонимную функцию.

Вышеописанная программа выведет следующие 2 строки.

```
12
33
```

ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ

Бывают ситуации, в которых дальнейшая корректная работа программы не продумана. В качестве примера можно привести выполнение деления на ноль.

Для того, чтобы сообщить об исключительной ситуации, используется ключевое слово **`$error`**. Если исключительную ситуацию не обработать должным образом, то

Листинг 5

```
$import SystemIO; /* Импортируем модель ввода-вывода */
$import Math; /* Импорт модуля математических функций */

MakeAddXFunc {
/* Данная функция принимает в качестве аргумента один символ – число
и создает функцию, которая увеличивает переданный ей аргумент на это
число */
     s.X = [ s.Y = <Math::Add s.X s.Y> ];
};

Main { /* Точка входа в программу */
     , <MakeAddXFunc 11>: s.Func = /* Вызов вышеописанной функции и
присвоение ее результата, созданной функции-переменной */
     <SystemIO::PrintLn <s.Func 1>> /* Вызов созданной функции 2 раза
для различных аргументов */
<SystemIO::PrintLn <s.Func 22>> ;
};
```

программа завершается с сообщением об ошибке. Для перехвата и обработки исключительных ситуаций используется конструкция `$try S1 $catch :: S2, S3,`. Здесь `S1` – код, который может создать исключительную ситуацию, `S2` – описание переменных, которым присваивается информация исключительной ситуации, а `S3` – код, который выполняется, если при выполнении `S1` произошла исключительная ситуация. Пусть рефал-5е машина выполняет код `S1`. Если при его выполнении возникла исключительная ситуация, то она присваивает информацию исключительной ситуации выражению `S2` по правилам, как это делается для локальных переменных и выполняет код `S3`. Приведем пример (листинг 6).

ИСПОЛЬЗОВАНИЕ ВНЕШНИХ БИБЛИОТЕК КОДА

Одно из наиболее важных свойств современного эффективного языка программирования – это возможность взаимодействия программы с другими программами и библиотеками, в том числе и написанными на других языках программирования. Для взаимодействия с другими программами система может предоставлять множество способов, например, именованные каналы или механизм общей памяти (англ. shared memory), однако для их использования необходимо использовать библиотечные функции.

Библиотека – это общепринятый способ повторного использования кода. Библиотеки бывают двух видов: статические и динамические. Статические подключаются к приложению в момент компиляции, динамические же подключаются в момент выполнения программы и, как следствие, должны присутствовать не только на компьютере разработчика, но также и на компьютере пользователей программного обеспечения. Так как транслятор языка рефал-5е на последней стадии работы будет являться интерпретатором, то нас будут интересовать толь-

Листинг 6

```
MakeAndCatchError {
     e.A = $try $error "Some error: " e.A,
          $catch :: e.Error,
          <SystemIO::PrintLn "Error: " e.Error>;
};
```

ко динамические библиотеки и взаимодействие с ними.

Динамические библиотеки экспортируют функции, которые принимают ноль или более типизированные аргументы и возвращают типизированный результат. Язык рефал не имеет дополнительной типизации аргументов. Все функции языка рефал принимают аргумент только одного типа, впрочем, программист может моделировать любое количество аргументов при помощи структурных скобок. Из-за различий в интерфейсах функций языка рефал-5е и динамических библиотек необходимо разработать и реализовать механизм их взаимодействия. В качестве основы для данного механизма можно взять один из двух следующих вариантов:

- За основу взять интерфейс динамической библиотеки, а в языке рефал-5е при импорте указывать сигнатуру функции. В этом варианте конвертацией между представлением данных в языке рефал-5е и в библиотеке будет заниматься интерпретатор языка рефал-5е.
- За основу взять интерфейс языка рефал-5е, тогда будет возможно импортировать только специально написанные функции. В этом варианте конвертацией представления данных при необходимости будут заниматься сами функции.

У каждого из этих подходов есть свои положительные и отрицательные стороны. Рассмотрим их подробнее. При использовании интерфейса библиотеки в качестве плюса можно выделить отсутствие необходимости писать обертки для большинства существующих библиотек. Из минусов можно выделить следующее:

- Необходимо разработать и реализовать механизм описания и разбора сигнатуры импортируемой функции, что сильно усложнит систему.
- В силу того что количество и типы параметров для каждой функции различны и, более того, становятся известны только на этапе работы интерпретатора, передавать аргументы и вызывать функцию придется вручную на низком уровне, используя ассемблер. Способ передачи параметров сильно зависит от системы (Windows [10], Linux [14], …) и архитектуры процессора (i386, X86-64 [18], ARM [1], …). Таким образом, для каждой пары операционная система-архитектура процессора будет необходимо реализовывать механизм вызова библиотечных функций заново, что сильно усложнит перенос системы на новые платформы.
- Библиотечные функции часто используют указатели в качестве аргументов и выходных параметров, а так же возвращают некоторые данные в аргументах. В языке рефал-5е нет указателей, а также невозможно изменить аргументы функции.

При использовании в качестве основы интерфейса функций языка рефал-5е можно выделить следующие положительные стороны:

- Все импортируемые функции имеют один и тот же интерфейс, поэтому нет необходимости описания их сигнатур, а также дополнительной ручной низкоуровневой передачи аргументов, что позволяет без дополнительных изменений переносить рассматриваемую систему на другую платформу.
- Отсутствие необходимости конвертировать параметры из одного представления в другое в некоторых случаях позволит значительно повысить быстродействие.
- Возможность более тесной интеграции импортируемых функций в программу. Так, например, можно из импортированных функций в качестве результата возвращать ошибку.
- Возможность поддержки разных входных/выходных данных в импортируемой функции.

Отрицательной стороной такого решения будет необходимость делать функции-обертки для всех желаемых библиотек.

Исходя из вышеуказанных плюсов и минусов, было решено использовать в качестве основы интерфейс функций языка рефал-5е, так как этот способ менее трудоёмок и потенциально предоставляет больше возможностей и большее быстродействие.

В рефал-5е программе для импорта функции из библиотеки будет использоваться директива $import_func, в качестве аргумен-

та ей будет передаваться строка, состоящая из имени функции и имени библиотеки, разделенных символом «:». В качестве имени библиотеки может выступать как имя файла библиотеки без расширения (для отсутствия необходимости писать разный код под разные системы), так и полный путь к библиотеке, в случае если она не находится в месте, в котором система не производит поиск динамических библиотек. Приведем пример использования.

```
$import_func 'PrintLn:StdIO';
```

Данная команда указывает интерпретатору языка рефал-5е, что необходимо импортировать функцию `PrintLn` из библиотеки `StdIO` (стандартной библиотеки ввода-вывода языка).

Для того чтобы загрузить функцию из библиотеки, интерпретатор должен сделать 2 действия:

1. Загрузить нужную библиотеку в память.

2. Найти нужную функцию в загруженной библиотеке.

В случае, если оба действия завершились успешно, адрес найденной функции сохраняется для дальнейшего использования. Код функции, выполняющей все эти действия на языке С, выглядит следующим образом (см. листинг 7).

При вызове функции из рефал-5е програмы интерпретатор первым делом будет определять, какая вызвана функция: внутренняя рефал-5е функция или внешняя, импортированная из какой-либо библиотеки. Это можно сделать, проверив указатель на функцию, хранящийся в дополнительной информации о вызываемой функции. Если он не равен NULL, то это импортируемая функция. Если функция является импортируемой, то ее нужно вызвать по указанному указателю, а в противном случае нужно передать выполнение функции вызова рефал-5е функций. На языке С код выглядит следующим образом (см. листинг 8).

Листинг 7

```
bool_t import_function(const char *lib_name, const char *func_name){
  void* library = dlopen(lib_name, RTLD_LAZY);
  if (library == NULL) {
    throw_runtime_error(RE_IMPORT_FUNCTION);
    return FALSE;
                                                                    }
  void* func = dlsym(library, func_name);
  if (func == NULL) {
    dlclose(library);
    throw_runtime_error(RE_IMPORT_FUNCTION);
    return FALSE;
  } else {
    return add_ext_function(func_name, func);
  }
```

Листинг 8

```
REFAL_pattern *call_function(func_info *function, REFAL_pattern *arg)
{
  if (function->ptr == NULL) {
    return call_REFAL_function(function, arg);
  } else {
    return function->ptr(arg);
  }
}
```

Литература

1. Архитектура и система команд RISC-процессоров семейства ARM / http://www.gaw.ru/html.cgi/txt/doc/micros/arm/index.htm (дата обращения: 20.12.2013).

2. *Бабаев И.О., Герасимов М.А., Косовский Н.К., Соловьев И.П.* Интеллектуальное программирование. Турбо Пролог и Рефал-5 на персональных компьютерах. Л.: Изд-во ЛГУ, 1992.

3. *Гошев В.А.* Схема универсальной функции для языка рефал-5 / Материалы всероссийской научной конференции по проблемам информатики. 23–26 апреля 2013 г. Санкт-Петербург, СПб.: Изд-во ВВМ, 2013.

4. *Гошев В.А., Косовский Н.К.* Разработка и реализация транслятора основных конструкций языка Рефал-5 и Рефал-Плюс / Сборник тезисов конференции «Технологии Microsoft в теории и практике программирования». СПб.: Изд-во Политехн. Ун-та, 2012.

5. *Гошев В.А., Косовский Н.К.* Методика трансляции Пролога в рефал+ / https://sunx.me/uploads/Prolog-To-Refal-plus.pdf (дата обращения: 20.12.2013).

6. *Гурин Р.Ф., Романенко С.А.* Язык программирования Рефал плюс. М.: ИНТЕРЕХ, 1991.

7. *Климов Р.Ф., Романенко С.А.* Система программирования Рефал-2 для ЕС ЭВМ. Описание библиотеки функций. М.: ИПМ им. М.В. Келдыша ФН СССР, 1986, препринт № 200.

8. *Климов Р.Ф.* Программирование на языке Рефал. 2004 / http://refal.net/~arklimov/refal6/manual.htm (дата обращения: 20.12.13).

9. *Колисниченко Д.* FreeBSD. От новичка к профессионалу. СПб.: БХВ-Петербург, 2012.

10. *Линн С.* Администрирование Microsoft Windows Server 2012. СПб.: Питер, 2013.

11. *Павловская Т.* C/C++. Программирование на языке высокого уровня. СПб.: Питер, 2013.

12. Сайт, посвященный развитию языка Рефал / http://wiki.botik.ru/Refaldevel/WebHome (дата обращения: 20.12.2013).

13. *Скоробогатов С.Ю., Чеповский А.М.* Язык Refal с функциями высшего порядка // Информационные технологии, 2006. № 9 / http://iu9.bmstu.ru/science/refal.pdf (дата обращения 20.12.2013).

14. *Собел М.Г.* Lunix. Администрирование и системное программирование. СПб.: Питер, 2011.

15. Тестирование производительности Рефал-5 и Рефал+ / http://rfp.botik.ru/cgrp/test (дата обращения: 20.12.2013).

16. *Уолл Л., Кристиансен Т., Орвант Д.* Программирование на Perl. СПб.: Символ-Плюс, 2006.

17. *Шилдт Г.* Java. Полное руководство. М.: Вильямс, 2012.

18. Intel® 64 and IA-32 Architectures Software Developer Manuals / http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html (дата обращения: 20.12.2013).

19. Libcurl – the multiprotocol file transfer library / http://curl.haxx.se/libcurl/ (дата обращения: 20.12.2013).

20. Qt – Cross-platform application and UI development framework / http://qt.digia.com/ (дата обращения: 20.12.2013).

21. The GTK+ Project / http://www.gtk.org/ (дата обращения: 20.12.2013).

22. The Industry's Foundation for High Performance Graphics / https://www.opengl.org/ (дата обращения: 20.12.2013).

23. Turchin V.F. Refal-5. Programming Guide and Reference Manual. New England Publishing Co., Holyoke, 1989.

PROJECT OF PROGRAMMING LANGUAGE REFAL-5E WITH CONVENIENT EXTENSIONS BY MEANS OF PREPROCESSOR

Abstract

Refal-5 language gives an ability to perform important tasks shorter and more effectively than another programming languages But existing implementations of Refal programming language dialects have no possibility to call a built-in interpreter from a program or to use external code libraries. To overcome these demerits a new programming language Refal-5e is offered. Refal-5e programming language provides an opportunity to call a built-in interpreter from a program and allows to use external code libraries. Also Refal-5e language allows a

function to return such result as “Success” or “Failure”, which permits to use backtracking like in Prolog language.

Keywords: REFAL-5, translation, compilation, interpretation, DLL, REFAL-5 machine.

Гошев Владимир Андреевич,
аспирант кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета,
sunx@sunx.cc,

Косовский Николай Кириллович,
доктор физико-математических наук, профессор, заведующий кафедрой информатики математико-механического факультета Санкт-Петербургского государственного университета,
kosov@NK1022.spb.edu.