Методы создания специализированных языковых и процедурных инструментов для оптимальной реализации алгоритмов для прикладных предметных областей

А. А. Рубан, email: andreygrigorev0@gmail.com
Н. Е. Балакирев, email: balakirev1949@yandex.ru
М. В. Зеленова, email: businkakatilas@mail.ru
М. М. Фадеев, email: fadeev_mix@bk.ru
В. С. Родионов, email: tuukvadim@live.com

Московский авиационный институт
(национальный исследовательский университет)

Аннотация. *В данной работе представлены методы создания специализированных языковых и процедурных инструментов для оптимальной реализации алгоритмов прежде всего для прикладной области исследования анализа и синтеза информационного содержания волн, что не исключает возможность использования и в других предметных областях. При этом акцент был сделан на документированность программного продукта, так как наличие комментариев слабо отражает сущность разработанных программ, и на удобство и доступность к сущности содержания операторов при использовании предлагаемых решений в процессе конкретного написания алгоритмов. Использование макроязыка и предлагаемых методов приближают содержание текста программы к понятиям и алгоритмам той предметной области, для которой реализуется программа. Испытав на практике преимущество предлагаемых методов предлагается использовать их в других сферах создания программного продукта.*

Ключевые слова: *макроязык, оператор, алгоритмическая синонимия, самодокументирование.*

Введение

При создании современных как технических, так программных продуктов невозможно обойтись без инструментов и технологий их создания. Идея вырывается за пределы обыденности и открывает новые возможности, а инструмент ускоряет процесс материализации возникшей идеи. В технической сфере инструментарий подразделяется на исследовательско-экспериментальный и технологическо-

производственный. В программной отрасли такое подразделение сливается воедино, но это не умоляет важности самого инструментария. Правда главной особенностью программной отрасли является тенденция постепенной унификации применяемых средств, особенно если говорить о языках программирования. Но практика создания продуктов такова, что она постоянно подталкивается реальной действительностью к необходимости модификации и даже пересмотра технологий создания программных продуктов. Переход к исследованию информационного содержания волн в силу потребности получения максимальной производительности (скорости) создаваемых алгоритмов в условиях обработки большого набора данных (big data) с необходимостью привел к использованию не только общепринятых языков, но языка низкого уровня (ассемблера). Для сглаживания разности в эффективности программирования по отношению к языкам высокого уровня стал применяться МАКРОязык. Но постепенное расширение набора макроопределений привело к пониманию возможности создания инструментария в виде отдельного языка операторов. Указанный множество операторов, как набор, позволяет легко собирать алгоритм и в то же время в какой-то степени описывать на содержательном уровне «смысл» алгоритма. Использование такого инструмента для написания критичных по времени исполнения алгоритмов и оформление их в виде DLL библиотек позволило использовать эти алгоритмы для языков высокого уровня, что обеспечило комплексность подхода.

На пути обеспечения оптимальности реализации и близости к естественно-языковому описанию алгоритмов в терминах предметной области, стоит:

- отсутствие гибкости формальных языков – они строго формализованы;
- именование объектов прикладной области ведется в терминах, навязываемых языком программирования и удобных программисту, но не в терминах применительно к прикладной области;
- отсутствие возможности обеспечить локальную оптимизацию, за исключением использования неудобных вставок на языке ассемблера;
- отсутствие широких возможностей обеспечить представление в структурной форме.

Для подобного рода приложений для прикладных систем определенную гибкость предоставляет макроязык, через который можно описывать действия в виде макроинструкций или в дальнейшем мы

будем их называть термином «оператор». Этот термин употребляется для отличия его от термина «команда» для Ассемблера (процессора) или от термина «инструкция». Используя средства языка FASM, имеется возможность создать практически любой набор именованных операторов, в том числе и на русском языке. Применение шаблонов и символа продолжения строки «/» дает потенциальную возможность структурировать текст, выстраивая описание алгоритма в виде блоков. Помимо этого, простой способ переименования операторов дает возможность в этом же имени указать цель использования оператора. Таким образом, алгоритмически подобные блоки, но используемые для разных целей, могут иметь разные имена, но одинаковую сущность, такое соответствие мы обозначим понятием алгоритмическая синонимия. Это позволяет, имея по сущности один алгоритм применить его с учетом терминологии различных предметных областей.

1. Анализ языка структурных операторов

Название языка происходит от слова «Структура», но наследованное не из понятия структурного программирования, а из желания построить и одновременно документировать на уровне смысла и терминов предметной области с помощью структурных операторов, и структур данных. Внесение элементов самодокументирования как при реализации алгоритма, так и при описании данных является, пожалуй, самой главной целью построения такого языка. Данный язык широко используется для создания программных модулей в рамках исследования распознавания речи на основе логика – лингвистического подхода [1]

Побудительным мотивом явилось то, что практически все языки программирования имеют единый стандартный набор операторов, который не позволяет при прочтении алгоритмов понять суть предметной области и содержание самого алгоритма относительно этой предметной области. Разбирая содержание алгоритма, в широком смысле этого слова, мы должны знать все операторы и нюансы, связанные с их использованием, а с другой стороны, пытаться соотнести их с предметной областью. Следует сказать, что имеется немного примеров реализаций языков, которые позволяют формализовать представление программы именно в рамках той предметной области, где мы используем этот алгоритм. К числу таких языков можно отнести язык для системы 1С, который имел целью написания алгоритмов в сфере бухгалтерских расчетов.

Таким образом, делается попытка решить вышеперечисленные проблемы через использование некого неформального языка представления алгоритмов. Внесение элементов самодокументирования,

как при реализации алгоритма, так и при описании данных является, пожалуй, самой главной целью построения такого языка.

Основой для реализации этого является наличие макроязыка для языка Ассемблера и соответствующих языковых средств, которые позволяют вводить любые наименования операторов и структур данных, практически, в любой форме. При создании такого языка придерживались следующих требований:

- Читаемость (наглядность) структуры всего алгоритма через иерархическое, наглядное, структурное представление.
- Ориентированность на предметную область. Возможность называть операторы, выполняющие одни и те же действия, но под разными именами. Операторы отражают суть предметной области.
- Широкая документированность за счет возможности вставления комментариев прямо внутри оператора. Например, комментарии, касающиеся предметной области или способа реализации данного участка алгоритма.
- Унифицированность средств реализации и приспосабливаемость под предметную область. То есть базовым является одно и то же имя оператора и его внутренний алгоритм, но этот оператор может иметь синоним близкий к предметной области и удобный интерфейс обращения.
- Уникальность представления при согласовании с предметной областью за счет возможности заводить такие синонимы. С точки зрения языка имеется единая унифицированная основа (например, оператор), а для обеспечения уникальности для конкретной предметной области привносится уникальное для неё имя, связанное с этой областью.
- Переносимость алгоритмов на нижнем уровне реализации на другие архитектурные платформы. Для каждой платформы оператор, который фактически пишется на языке системы команд компьютера, может быть реализован через новую последовательность команд. То есть мы, не меняя содержания алгоритма на языке, в рамках которого используется некоторое подмножество из набора операторов, можем пошагово перейти на новую архитектуру за счет переписывания содержимого операторов и структур, входящих в программу.
- Открытость сущности содержания реализованного алгоритма и закрытость получаемого кода, предоставленного и описанного в терминах операторов предметной области.

- Перестраиваемость структуры программы и восстанавливаемость по структуре программного продукта при наличии библиотеки описаний макроопределений, что дает возможность создания индивидуального экземпляра программы для отдельного индивидуума (индивидуализация продукта).
- Рекурсивная оптимизация реализованных алгоритмов за счет возможности переписывания предлагаемых операторов на языке низкого уровня так и на предлагаемом языке.

Это достигается через попытку систематизировать и обобщить процедуры работы с данными, акцентировать внимание на предметности через обобщенность алгоритмов обработки. Процедуры и стандартные программы, имеющие доступное структурное описание, с поддержкой информационно справочной системы дополняют и расширяют основу языка.

1. Включение таких возможностей не исключают некоторых недостатков и издержек:
2. Требуется новая технология создания программ освоение новой парадигмы.
3. Отказ от использования формальных привычных языков высокого уровня и переход на не формальные конструктивные элементы.
4. Акцентирование внимания на предметной области и самодокументировании реализуемого алгоритма.
5. Увеличение количества наименований данных и операторов
6. Незначительное увеличение объема программного продукта по отношению к написанию напрямую через использование Ассемблера.

2. Причины и мотивы акцентирования внимания на структурах

Реализуя алгоритмы из какой–либо конкретной области, приходится оперировать с собственными характерными структурами и строить подсистемы, работающие на основе некоторых базовых алгоритмов, которые и ориентированы на эти структуры. Но, с другой стороны, имеется множество необходимых языковых конструкций и структур, не имеющих отношения к создаваемой программе, но необходимых для обеспечения интерфейса с другими системами, включая ОС (среда исполнения). Безусловно, содержательно необходимо сделать акцент на самом алгоритме, а среду обеспечения «экранировать» при реализации алгоритма.

Структуризация инструментария может основываться как на создании библиотеки алгоритмов, так и макроинструментарии, преимуществом которого может стать наглядность, близкая к

естественному языку и языку математики. Именно на макроконструкции делается упор в данном языке. Фактически, языковый набор средств выступает основой, как для сборки алгоритмов, так и для их описания.

В рамках свободы выбора имен операторов и идентификаторов при их большом разнообразии должна выстраиваться как модульная схема сборки, так и определенные соглашения именования объектов, которые устанавливаются пользователями языка. При использовании языка в конструировании блоков анализа и синтеза речи был апробирован определенный порядок конструирования идентификаторов (меток), которые по возможности учитывает особенности обработки информации на уровне аппаратуры, языка транслятора, но, в первую очередь, сущность и предназначение реализуемого алгоритма.

Что касается «экранирования» среды исполнения при реализации алгоритмов, то для их программирования имеются шаблоны конструкций (программная платформа), которые учитывают необходимые «стандартные» соглашения при обращении к ОС или оформлении в виде определенного ресурса [2-**Ошибка! Источник ссылки не найден.**]. Внося лишь некоторый минимальный набор параметров в такой шаблон (например, указание точки входа в программу), на его основе из большого набора операторов собирается программа, реализующая алгоритм. При этом множество системных объявлений и включений экранировано одним оператором, который текстуально минимизирует объем текста и является по факту именем некоторого шаблона с параметрами. Объем информации, необходимый программе при использовании системных ресурсов и отражаемый в шаблоне, может быть оптимизирован с учетом востребованности. Если ресурс требуется в программе, то путем указания одного символа «+» в шаблоне, данный ресурс подгружается, в противном случае ставится или стоит символ «-». При этом в экранируемом оператором шаблоне перечислены все возможные и доступные ресурсы, со ссылкой на их описание и назначение. По такой же схеме подключается набор операторов языка, который необходим в программе. Такая схема экранирования при возможности обращения к «первоисточнику» позволяет не захламлять программу не нужными «деталями» и экономить объем памяти при работе транслятора.

3. Некоторые фрагменты возможностей языка

И так целевой вектор предложений был направлен на приближение описания программы к более «человечному» пониманию, то есть приближение к первому этапу разработки (касающегося идеи, модели и алгоритма) и возможности выразить интерпретацию алгоритма программистом человеческим языком. Использование языка FASM,

показавшего свои несомненные преимущества перед другими подобными языками низкого уровня, позволило реализовать выше сказанное как в английской, так и в русской нотации. Любой параметр в обращении к шаблону системных вызовов и соглашений сопровождается ключевым словом, которому присваивается конкретное значение из множества вариантов, смотри рис.1.

```
include 'C:\FASM\MZERO.inc'
НАЧАЛО_ПРОГРАММЫ Форма=PE,Вывод=Console,Вход=start
СЕКЦИЯ_ДАННЫХ Имя=data Статус_чтения=+ Статус_записи=-
```

Рис. 1. Пример начала программы

Для обеспечения обозримости и наглядности были реализованы операторы, которые имеют (условно) многоуровневую нотацию с использованием знака «\». Первым идет имя оператора, являющегося ключом к макроопределению и отражающего смысл и назначение. Второй строкой указываются переменные, относительно которых устанавливаются отношения, а в третей строке указаны метки перехода с учетом этих отношений.

Заключение

В приводимом материале затронут лишь небольшой фрагмент конкретных возможностей в развивающемся языке СТРУКТОП. Но как пример практичности и удобства языка следует отметить опыт использования для реализации алгоритмов по извлечению информационного содержания языка. При реализации одного из алгоритмов, фрагмент показан на рис. 2, потребовалось использовать несколько десятков операторов «СучетомОтношений!Перейти», которые определяли структуру участка волны по древовидной схеме, и заполняли объявленную структуру по мере продвижения по ветвям дерева. При наличии таких и подобных операторов потребовалось менее недели на написание и отладку, при этом вначале было отлажена программа по управлению (правильный обход дерева с учетом входных данных), а затем были добавлены операторы заполнения введенной структуры хранения. При этом, используя оператор «Записали», который позволяет описать «одновременное» присвоение значений для структуры, состоящей из 8 байт, удалось проследить весь путь изменений в соответствующих полях записываемой информации. Обозримость текста алгоритма позволяла быстро находить ошибки и недочеты. Авторы надеются, что подобные методы будут применимы и для других предметных областей с возможным пополнением

необходимых операторов с учетом специфики обработки информации в данной области.

```
                СучетомОтношений!Перейти
                        A ? B ? C \
        =>P1 =>P2 =>P3  =>P4 =>P5=>P6=>P7 =>P9 =>P10 =>P11  =>P_12  =>P_13 =>P_14
;*****************************************************************

;******************************************** БЫЛО  для P1 P2 P3
;  ([RSa_b]=al)       ([RotA_B]=1)      ([RSb]=bl)      ([RotB_C]=1)      ([RSc]=cl)      ([RPr]=0)
([RP7]$пусто)   ([RP8]$пусто)
;        1              2         3          4               5        6         7
8
;******СТАЛО |*****************************************************
P1:                                 ;Фиксировать    первоначальный    тип    структуры
только  \
                Записали\
      ([RSa]$Тож)      ([RotA_B]:+1)    ([RSb]=cl)     ([RotB_C]=0)      ([RSc]=0)       ([RPr]=1)
([RP7]$пусто)   ([RP8]$пусто)
;Стало1            2          3          4               5          6        7
8
        ПерейтиВ      INCMN                 ;jmp    INCMN
;*****************************************************************
P2:                                 ;Фиксировать    первоначальный    тип    структуры
только  /
                Записали\
     ([RSa]$Тож)      ([RotA_B]:+1)    ([RSb]=cl)     ([RotB_C]=0)      ([RSc]=0)       ([RPr]=2)
([RP7]$пусто)   ([RP8]$пусто)      ;Стало1       2          3          4                5
6             7               8
        ПерейтиВ      INCMN                          ;jmp    INCMN
;*****************************************************************
```

Рис. 2. Пример операторов языка СтрукТоп

Список литературы

1. Балакирев Н. Е. Логико-лингвистический подход по распознаванию содержания физических волн / Н.Е. Балакирев // Материалы XV Международной конференции «Информатика: проблемы, методология, технологии». (Воронеж, 12-13 февраля 2015 г.). – Воронеж, 2019. – Т. 1. – С. 31-36.
2. Competent object model [Электронный ресурс] : информационная статья. – Режим доступа: http://techn.sstu.ru/kafedri/%D0%BF%D0%BE%D0%B4%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F/1/MetMat/murashev/com/lec/cl01.htm
3. Технологии программирования [Электронный ресурс] : информационная статья. – Режим доступа : http://bourabai.ru/alg/com/gl09.htm