МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

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

Студент 1-го курса Магистратуры:
Рубан Андрей Алексеевич
Научный руководитель: Балакирев Н.Е.

Цель

- Цель работы: Разработка методов создания специализированных языковых и процедурных инструментов для оптимальной реализации алгоритмов, с акцентом на само документированность программного продукта.

Побудительные мотивы

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). Вместе с ассемблером широко **применялся МАКРОязык**. **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 и для языков высокого уровня.

Причины отказа от других языков программирования.

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.

Языка структурных операторов.

Название «**СтрукТОП**» 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 **само документирования** как при реализации алгоритма, так и при описании данных является, пожалуй, **самой главной целью построения** такого **языка**. Стоит отметить, что данный язык широко используется для создания программных модулей в рамках исследования распознавания речи на основе логика – лингвистического подхода.

Язык ассемблера FASM и его широкие возможности.

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 обозначили понятием **алгоритмическая синонимия**. Это позволяет, имея по сущности один алгоритм применить его с учетом терминологии различных предметных областей.

Требования к языку

:

- **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.

Продолжение

- **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.

Недостатки и издержки

- Требуется новая технология создания программ освоение новой парадигмы.
- Отказ от использования формальных привычных языков высокого уровня и переход на не формальные конструктивные элементы.
- Акцентирование внимания на предметной области и само документировании реализуемого алгоритма.
- Увеличение количества наименований данных и операторов
- Незначительное увеличение объема программного продукта по отношению к написанию напрямую через использование Ассемблера.

Некоторые фрагменты возможностей языка

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:

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

Многоуровневая нотация

```
;********************************************************************** БЫЛО для P1 P2 P3
                 СучетомОтношений!Перейти
                       A ? B ? C \
        =>P1 =>P2 =>P3  =>P4 =>P5=>P6=>P7 =>P9 =>P10 =>P11  =>P_12  =>P_13 =>P_14
;**********************************************************************
; ([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:              Записали\
     ([RSa]$Тож)  ([RotA_B]:+1)  ([RSb]=cl)  ([RotB_C]=0)   ([RSc]=0)   ([RPr]=1)   ([RP7]$пусто)   ([RP8]$пусто)
;Стало  1            2              3            4            5           6            7                8
        ПерейтиВ        INCMN                    ;jmp      INCMN
;**********************************************************************
P2:
                 Записали\
    ([RSa]$Тож)  ([RotA_B]:+1)  ([RSb]=cl)  ([RotB_C]=0)   ([RSc]=0)   ([RPr]=2)   ([RP7]$пусто)   ([RP8]$пусто)
;Стало  1            2              3            4            5          6            7                 8
        ПерейтиВ        INCMN                    ;jmp      INCMN
;**********************************************************************

        ....
;**********************************************************************
```

Еще один пример

СучетомОтношений!Получить!Перейти\

AX =? BX =? CX =? \

rg=КодОтвета \

(PRA=> P1) (PRE=> P2) (PRO => P3) (PRX => P4) (PRY=> P5) (PRI =>P6) (PRJ=>P7) (PRH=>P8) (PRG=>P9) (PRK=>P10) (PRC=>P11) (PRS=>P12) (PRZ=>P13)

Еще один пример

```
Перейти_по_Значению_в_списке \
?:=EAX \
=>STAGE1 =>STAGE2 =>STAGE3 => STAGE4 \
=> Exit
```

Еще один пример

```
Если_это_Символ_то_Перейти \

?:=AL \

(‘+’  =>PLUS ) (‘>’  =>GREAT ) ( ‘<’  =>LESS ) ( 03Bh  => LESS)\

=> ExiPloxo
```

Еще один пример

```
УстановитьОтношенийБайтовыхМножеств \

al  =?bl \

=> P1 => P2 => P3 => P4 => P5 => P6 => P7 => P8 => P9 => P10 => P11 => P12 => P13 => P14
```

Заключение

В представленном материале затронут лишь небольшой фрагмент конкретных возможностей развивающегося языке СТРУКТОП.

СПАСИБО ЗА ВНИМАНИЕ