ДАЛВЯЗ 2 – ОПИСАНИЕ

ДРУЖЕЛЮБНЫЙ АЛГОРИТМИЧЕСКИЙ ВИЗУАЛЬНЫЙ ЯЗЫК, версия 2
(семейство языка ДРАКОН)

визуальное алгоритмическое программирование
с генерацией и считыванием
структурированного программного кода
для языков семейств Pascal и C/C++

Барановский Дмитрий Владимирович
главный специалист отдела ГА, сектор документирования ПО
ОАО «НИИАС»

БЛАГОДАРНОСТИ

Большое спасибо Владимиру Даниеловичу Паронджанову за его идеи в области визуального программирования, которые и побудили меня серьезно задуматься над вопросом – а для чего нужно визуальное программирование и каким оно должно быть.

Выражаю благодарность Владиславу Жаринову – во многом именно вследствие его ценных замечаний и предложений язык ДАЛВЯЗ 2 и приобрел свой нынешний облик.

Благодарю Алексея Донского и Андрея Тюгашева (TAU) за их ценные, хотя иногда и критические замечания, побудившие меня к дальнейшим размышлениям в области визуального программирования.

ИСПОЛЬЗУЕМЫЕ СОКРАЩЕНИЯ

ДРАКОН	−	Дружелюбный Русский Алгоритмический язык, Который Обеспечивает Наглядность
ЯПВУ	−	язык программирования высокого уровня
ЛСП	−	логическая структура процедуры
ПкДАЛВЯЗ	−	псевдокод языка ДАЛВЯЗ 2

НАЗНАЧЕНИЕ ЯЗЫКА ДАЛВЯЗ 2

Алгоритмический язык ДАЛВЯЗ 2 предназначен для:

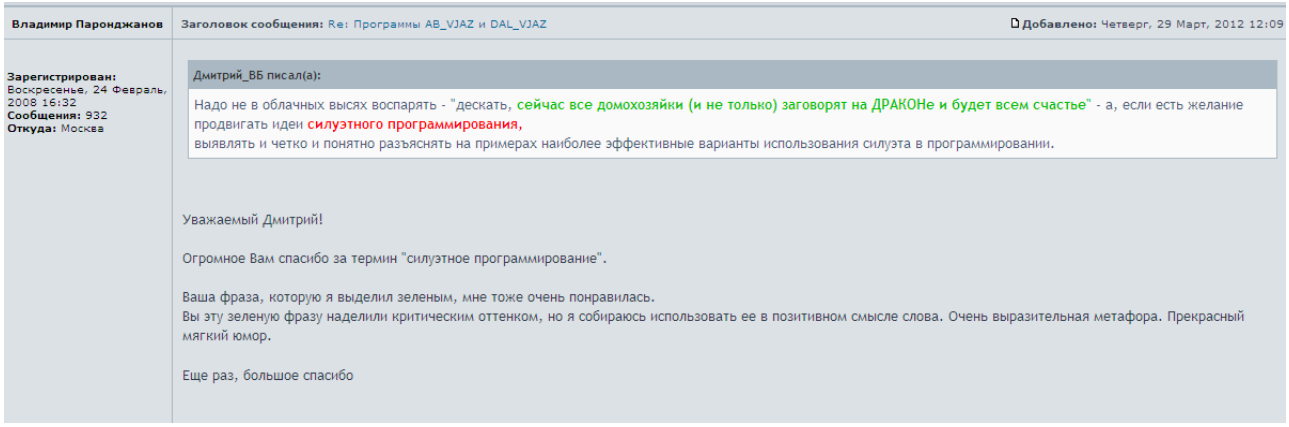
1) создания и отображения на мониторе компьютера минимизированных (компактифицированных) визуальных схем подмножества языка ДРАКОН, являющихся графической формой отображения логической структуры процедур исходного текста программы; запись текстовой формы отображения **логической структуры процедуры** (ЛСП) выполняется с клавиатуры псевдокодом ПкДАЛВЯЗ в окне ЛСП (см. ниже);
2) развития идей силуэтного программирования, впервые сформулированных В.Д. Паронджановым при разработке визуального языка ДРАКОН. Основная мысль Владимира Даниеловича:

Визуальное и эргономичное представление алгоритма в форме силуэта дает возможность охватить алгоритм одним взглядом и провести визуальную оценку правильности его структуры, что позволяет выявить часть ошибок, возможно содержащихся в этом алгоритме.

Мои добавления:

- ✓ цикл-силуэт (см. ниже) компактифицирует визуальную схему алгоритма, позволяя размещать достаточно сложный алгоритм на одном экране; а если на одном экране алгоритм все-таки не помещается, то цикл-силуэт, если он по высоте умещается в экран, дает возможность просматривать алгоритм, используя горизонтальную полосу прокрутки, а не вертикальную, что, на мой взгляд, более удобно; к тому же компактификация алгоритма уменьшает информационное давление на программиста, позволяя ему по собственному желанию просматривать текстовое содержимое выбранных им визуальных элементов схемы;

- ✓ вместо того, чтобы гнаться за полнотой реализации описания языка ДРАКОН, как оно была изложено В.Д. Пароннджановым в его книге «Как улучшить работу ума» (я считаю, что это очень трудоемкое дело), следует сосредоточить основные усилия на максимально эффективном использовании возможностей, предоставляемых программисту визуально-логической конструкцией «силуэт», т.е. на СИЛУЭТНОМ ПРОГРАММИРОВАНИИ



- ✓ отображение визуальной схемы скорректировано с целью сделать процесс создания визуальной схемы как можно более быстрым и легким

для программиста – введены понятия «основные и дополнительные маршрутные операторы» и «обобщенный цикл»;

3) формулирования и иллюстрации идей, которые могли бы быть использованы при создании систем разработки и документирования ПО для промавтоматизации (проекты средней сложности). Сейчас на господство в этой области претендуют UML-системы. Но при всех своих достоинствах они имеют и недостатки, главный из которых для России, на мой взгляд – это очень большая трудоемкость сопровождения программного проекта, созданного и поддерживаемого при помощи UML-технологий. Китай и Индия смогут потянуть разработку ПО по UML-технологиям, а России, если она хочет остаться независимой в области IT, нужно придумывать что-то свое, более простое (а то народу у нас мало).

ВИЗУАЛЬНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА ДАЛВЯЗ 2

Язык ДАЛВЯЗ 2 включает в себя следующие типы визуальных элементов, представленных на рисунке 1 визуальной схемы языка ДАЛВЯЗ 2:

1) заголовок процедуры (эл. 1);
2) условие (эл. 4);
3) дополнительное условие (эл. 6);
4) начало обобщенного цикла (эл. 2);
5) конец обобщенного цикла (эл. 21);
6) действие (эл. 5, 7, 8, 11, 15, 17, 19);
7) цикл-силуэт – программная реализация визуально-логической конструкции «силуэт» (эл. 2 – 21);
8) заголовки веток цикла-силуэта (эл. 3, 10, 13); заголовок ветки 1 реализуется с использованием элемента «условие», а заголовки веток 2 и 3 – с использованием элементов «дополнительное условие»;

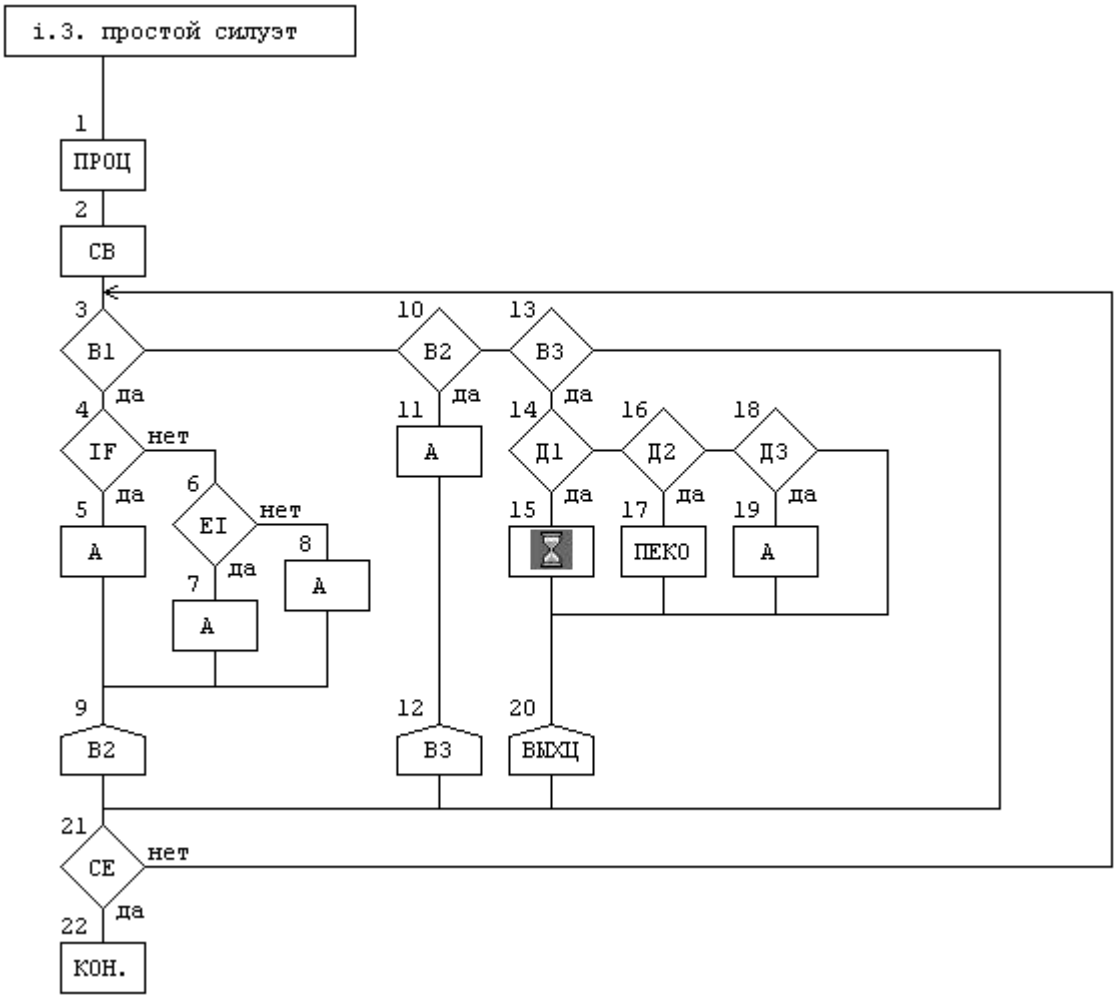


Рисунок 1 – Визуальная схема языка ДАЛВЯЗ 2

9) переход к ветке цикла-силуэта (эл. 9, 12); реализуются с использованием элемента «действие»;
10) выход из цикла-силуэта (эл. 20); реализуется с использованием элемента «действие»;
11) заголовки действий переключателя (эл. 14, 16, 18); заголовок действия 1 переключателя реализуется с использованием элемента «условие», а заголовки действий 2 и 3 переключателя – с использованием элементов «дополнительное условие»;
12) окончание процедуры (эл. 22).

ЧТО ТАКОЕ ЛСП

Основными строительными блоками программы являются действие (присвоение или вызов процедуры), цикл и сложное условие. Это постулат структурного программирования.

В ДАЛВЯЗ 2 **действие** (присвоение или вызов процедуры или группа каких-нибудь операторов языка программирования), **обобщенный цикл и сложное условие являются основными маршрутными операторами, полноценно отображаемыми на визуальной схеме** в виде блоков действий, связанных между собой линиями логических переходов, **и задаваемыми псевдокодом ПкДАЛВЯЗ** в окне ЛСП **при создании** и редактировании **визуальной схемы.**

Логическая структура процедуры – это последовательность основных маршрутных операторов, укрупненно описывающая логику процедуры.

Поэтому можно сказать, что визуальные схемы языка ДАЛАЯЗ 2 являются одной из разновидностей структурных блок-схем: чем более структурно написан текст процедуры, тем больший процент маршрутных операторов процедуры будет явно отображен на визуальной схеме.

Дополнительные маршрутные операторы и символьные сокращения для них, заменяющие номера их пиктограмм в определителе рисунков и сокращений:

- выход из цикла по условию (b?);
- продолжение цикла по условию (c?);
- переход к обработке ошибки по условию (e?);
- обработка ошибки (er);
- выход из процедуры по условию (r?);
- переход goto по условию (u? , mu - вверх и метка перехода вверх;

d? , md - вниз и метка перехода вниз)

отображаются в виде специальных значков (пиктограмм) внутри визуальных элементов схемы. Этот прием заимствован из программирования, только условный переход для наглядности обозначается не ключевым словом, а пиктограммой.

Кроме этого, значки действий (а не форма блоков действий, как в языке ДРАКОН) могут быть использованы для документирования программы.

ПСЕВДОКОД ПкДАЛВЯЗ

Псевдокод ПкДАЛВЯЗ для описания структурного текстового представления визуальной схемы включает в себя минимальный набор операторов, необходимый для создания полноценного исходного кода программы. Все операторы ПкДАЛВЯЗ однозначно соответствуют аналогичным операторам условий и циклов ЯПВУ семейств Паскаль и C/C++, а потому задание в файле конфигурации конкретного символьного представления операторов ПкДАЛВЯЗ позволяет считывать и генерировать текст программы для ЯПВУ семейств Паскаль и C/C++, а также для псевдокода. Для справки в нижеследующем описании операторов ПкДАЛВЯЗ приводятся и соответствующие им операторы языка Паскаль.

ПкДАЛВЯЗ включает в себя следующие операторы:

1) Оператор начала процедуры	HD
Номера типов визуальных элементов:	1
(см. пояснения к рис.1)	
Клавиша окна ЛСП:	нет

Начало процедуры / функции в языке Паскаль:	procedure / function
2) Оператор начала блока условия	IF
Номера типов визуальных элементов:	2, 8, 11
Клавиша окна ЛСП:	F2
Начало блока условия в языке Паскаль:	if (usl) then begin
3) Оператор дополнительного блока условия	EI
Номера типов визуальных элементов:	3, 8, 11
Клавиша окна ЛСП:	F3
Дополнительный блок условия в языке Паскаль:	end else if (usl) then begin
4) Оператор альтернативного блока условия	EB
Номера типов визуальных элементов:	нет
Клавиша окна ЛСП:	F4
Альтернативный блок условия в языке Паскаль:	end else begin
5) Оператор конца блока условия	E
Номера типов визуальных элементов:	нет
Клавиша окна ЛСП:	F5
Конец блока условия в языке Паскаль:	end;
6) Оператор начала обобщенного цикла	CB
Номера типов визуальных элементов:	4
Клавиша окна ЛСП:	F6
Начало цикла в языке Паскаль:	for, while, repeat
7) Оператор конца обобщенного цикла	CE
Номера типов визуальных элементов:	5

Клавиша окна ЛСП: F7

Конец цикла в языке Паскаль: «;», end, until

8) Оператор действия A

Номера типов визуальных элементов: 6

Клавиша окна ЛСП: F8

Оператор действия в языке Паскаль: любой исходный код на языке Паскаль, например: a := b+c;

9) Оператор окончания процедуры / функции R

Номера типов визуальных элементов: 12

Клавиша окна ЛСП: нет

Конец блока условия в языке Паскаль: end;

Я считаю, что после получения некоторой практики ввода визуальной схемы с клавиатуры можно добиться скорости ввода схемы как минимум не меньшей, а возможно и большей, чем скорость ввода такой же визуальной схемы в графическом редакторе при помощи мыши.

Вопрос – писать ли полноценный графический редактор для ввода визуальной схемы – решат для себя самостоятельно будущие разработчики визуальной среды для ДАЛВЯЗ 2 (если таковые найдутся).

ФАЙЛ КОНФИГУРАЦИИ

```
;
;  ФАЙЛ КОНФИГУРАЦИИ ЗАДАН ДЛЯ ЯЗЫКА ПАСКАЛЬ
;
;                          " " или "значение"
;
__шаг_уровня_структуры   "3"    ; число пробелов слева (или шаг
```

```
;                                  табуляции)
;
__количество_пиктограмм  "22"   ; до 40 рисунков p_XX.bmp, см. picts.txt
;
__толщина_линии          "1"    ; для схемы и структуры, 1 или 2
;
;--------------------------
;
; определения для записей
;
__загол_прог_записи      "(* i."   ; заголовок программной записи
__загол_текст_записи     "(* t."   ; заголовок текстовой записи
__опред_рис_сокр         "(*рис"   ; определитель рисунка и сокращения
;
;--------------------------
;
; задание ключевых слов и символов для программной процедуры / структуры
; для языка Паскаль
;
__строка_док_IF          "(*i*)  if"
__строка_док_EI          "(*i*)  end else if"
__строка_док_EB          "(*i*)  end else begin"
__строка_док_END         "(*i*)  end;"
__строка_док_CB          "(*i CB *)"
__строка_док_CE          "(*i CE *)"
__строка_док_A           "(*i*)"
__строка_док_RET         "(*i ВЫХОД *)"
__строка_док_STOP        "(*i КОНЕЦ *)"  ; следует после конца процедуры
__проверка_ветки         "if (wetka = "
__переход_к_ветке        "wetka := "
;
;-------------------------------------------------------------------------
;
; рабочие каталоги (до 30):
;
__каталог01          " "
__каталог02          "d:\wrk_comp\dima_f\delphi\dalvjaz2"
```

```
;
__каталог_чт        ".\folder_r"
;
; для сохранения текущих версий файлов, загруженных в программу
; при считывании новых версий файлов извне
;
__рабочий_каталог    ".\rk"
; каталог, где находятся файлы сокращений и где сохраняются
; bmp-файл схемы и файл текстового описания схемы
;
;
; модули (до 99):
;
; в именах модулей не должно быть пробелов
; в именах должна быть только 1 точка
; длина имен не больше 20 символов
; имена модулей должны быть уникальными
;
; N модуля     N     ссылочный  имя       загрузка       N файла
;            ката-   N модуля   файла     в программу   сокращений
;            лога    м.б.>100             ДОК/КОД/---   sokrN.txt
;
__модуль01   "01>      1      u_test1.txt    ДОК          1"
__модуль02   "01>      2      u_test3.pas    ДОК          1"
__модуль03   "02>      3      u_test2.txt    КОД          1"
;
;                                    в программу можно
;                                   загружать до 10 файлов
;
; загрузка в программу: ДОК - загрузка модуля в режиме документирования
;                       КОД - загрузка модуля в режиме кодогенерации
;                       --- - нет загрузки модуля
;
;
; htm-файлы, на которые можно ссылаться (до 10):
;
__htm01     "02>              dalvj2.htm"
```

```
;
;
__браузер     "D:\Program Files\Internet Explorer\IEXPLORE.EXE"
;
;__браузер    "<полное имя браузера для просмотра htm>"
;
; ссылка на записи других модулей из программы:
; _>4.12  - это ссылка на запись 12 модуля 4 (ссылочный N модуля)
; _>.17   - это ссылка на запись 17 текущего модуля
;
; при ссылке на незагруженный в программу модуль появится
; сообщение:  "модуль N не загружен"
;
; ссылка на записи htm-файлов (здесь N модуля - это N htm-файла):
;
; _>2#10  - это ссылка на запись с именем "i.10" 2-го htm-файла
;
;--------------------------
.
```

ТЕКСТОВАЯ И ПРОГРАММНАЯ ЗАПИСИ

ДАЛВЯЗ 2 дает программистам возможность создавать визуальные схемы из уже существующих процедур и загружать уже существующие проекты из нескольких модулей на языках семейств Pascal и C/C++. Для этого один или несколько модулей проекта разбиваются на текстовые записи заголовками вида:

(* t. имя записи *) - для семейства Pascal

/* t. имя записи */ - для семейства C/C++

после чего модуль может быть считан в программу, реализующую работу с языком ДАЛВЯЗ 2.

Текстовые записи отображаются только как текст, у них активны ссылки на другие записи (например: _>4.12 _>.17 _>2#10 – активные ссылки

выделены синим цветом, для перехода по ссылке нужно щелкнуть на ней левой кнопкой мыши), текстовые записи можно редактировать, если модуль загружен в режиме КОД.

Программные записи могут отображаться как текст или как схема, в зависимости от того, взведен или сброшен флажок «схема».

Силуэт с рисунка 1 имеет следующий вид в текстовом формате:

Сгенерированный текст процедуры:

```
(* i.3. простой силуэт *)
   (*i CB *)
      (*i*)  if (wetka = 1)    (*рис . .0001 *)
         (*i*)  if
            (*i*)
         (*i*)  end else if
            (*i*)
         (*i*)  end else begin
            (*i*)
         (*i*)  end;
         (*i*)  wetka := 2;    (*рис . .0004 *)
      (*i*)  end else if (wetka = 2)
         (*i*)
         (*i*)  wetka := 3;
      (*i*)  end else if (wetka = 3)
         (*i*)  if    (*рис . . Д1 *)
            (*i*)    (*рис .14 . *)
         (*i*)  end else if  (*рис . . Д2 *)
            (*i*)    (*рис . .ПЕКО *)
         (*i*)  end else if  (*рис . . Д3 *)
            (*i*)
         (*i*)  end;
         (*i*)  wetka := 0;
      (*i*)  end;
   (*i CE *)
   (*i ВЫХОД *)
(*i КОНЕЦ *)
```

ЛСП:

```
HD
CB
  IF
    IF
      A
    EI
      A
    EB
      A
    E
    A
  EI
    A
    A
  EI
    IF
      A
    EI
      A
    EI
      A
    E
    A
  E
CE
R
```

Для преобразования текстовой записи в программную нужно:

1) скопировать требуемое число раз основной служебный комментарий задания элемента визуальной схемы (*i*) (или /*i*/ для семейства C/C++), а затем в некоторых местах его подкорректировать - для циклов, выхода и окончания процедуры;
2) поменять в заголовке записи t на i ;
3) взвести флажок «схема» – после этого запись отобразится как схема (см. рис.1).

СОВМЕСТНАЯ РАБОТА СО СРЕДАМИ ПРОГРАММИРОВАНИЯ

После сохранения программой новой версии файла исходного кода при переходе к окну среды программирования там появится вопрос: "Файл XXX изменился. Загрузить новую версию?". Ответ – «Да».

Программа по таймеру постоянно проверяет времена создания файлов исходного кода и файла сокращений и при их изменении автоматически считывает новые версии этих файлов, получаемые при нажатии на кнопку "Сохранить файл" в среде программирования или в текстовом редакторе.

СПЕЦИФИКА РАБОТЫ СО СРЕДОЙ БЛЭКБОКС

Хотя в справочной системе ББ говорится, что файлы исходного кода должны создаваться с расширением .odc, но я попробовал и выяснилось, что для создания программного модуля подходит и файл .txt . Таким образом была устранена несовместимость в направлении ББ –> ДАЛВЯЗ 2.

Для устранения несовместимости ДАЛВЯЗ 2 –> ББ нужно параллельно с ББ открыть текстовый файл исходного кода в каком-нибудь текстовом редакторе, реагирующем на изменение времени создания редактируемого

файла. После считывания новой версии файла исходного кода текстовым редактором следует скопировать измененную в ДАЛВЯЗ 2 процедуру в межзадачный буфер текстового обмена, а затем вставить содержимое буфера в нужное место загруженного в ББ файла исходного кода.

КОДОГЕНЕРАЦИЯ И ДОКУМЕНТИРОВАНИЕ

В режиме КОД (кодогенерация) выполняется полноценная кодогенерация исходного кода модуля из программы, реализующей работу с ДАЛВЯЗ 2, по нажатию кнопки "Сохранить файл" или после подтверждения сохранения файла при выходе из программы.

В режиме ДОК (документирование) окна редактирования текстовой записи и элемента схемы открываются как только для чтения, можно менять лишь значок пиктограммы и сокращение для элемента схемы. По нажатию клавиши «Запомнить изменения для элемента» программа выполнит сохранение текущей версии файла исходного кода со служебным расширением, а на его место запишет новую версию файла исходного кода со сделанными для элемента исправлениями.

Если все модули загружены в режиме ДОК и не планируется менять пиктограммы и сокращения для элементов схемы, то программа будет работать в режиме просмотрщика с активными ссылками на другие записи модулей проекта (в том числе и модулей текстового описания программы) и htm-файлов описания проекта.

ОПРЕДЕЛИТЕЛЬ РИСУНКА И СОКРАЩЕНИЯ

В программной записи визуальной схемы (см. выше) в первой строке описания элемента может находиться определитель рисунка (пиктограммы) и сокращения для элемента, например: `if (wetka = 1)  (*рис . .0001 *) .`

По умолчанию внутри визуальных элементов (исключения: начало и конец процедуры, заголовок ветки, переход к другой ветке, выход из цикла-силуэта, заголовок действия переключателя) выводятся типы этих элементов.

При редактировании текстового содержимого визуального элемента (вход в редактирование – нажатием левой кнопки мыши на элементе) для элемента можно задать пиктограмму и (или) сокращение (до 4-х символов).

Пиктограмма выбирается из списка пиктограмм, а сокращение – из списка сокращений, задаваемых в файле сокращений. Сокращения вводятся в файл сокращений в алфавитном порядке или, если сокращение состоит из 4-х цифр – в порядке следования номеров сокращений.

Если заданы и пиктограмма и сокращение, то внутри элемента будет выводиться пиктограмма, а при наведении указателя мыши на элемент во всплывающем окне для элемента будут выводиться тексты для пиктограммы и для сокращения (если они есть), а также текстовое содержимое элемента, например: `(*i*) wetka := 2; (*рис . .0004 *) .`

На рис. 1 для действия 15 задана пиктограмма «ожидание по времени» и определитель для этого действия имеет вид: `(*рис .14 . *).` Для действия 17 задано сокращение «пересчет координат схемы», а определитель для этого действия имеет вид: `(*рис . .ПЕКО *).`

Предлагаемый набор пиктограмм приведен на рис. 2 и может быть изменен пользователем, тексты описания пиктограмм задаются в текстовом файле описаний пиктограмм.

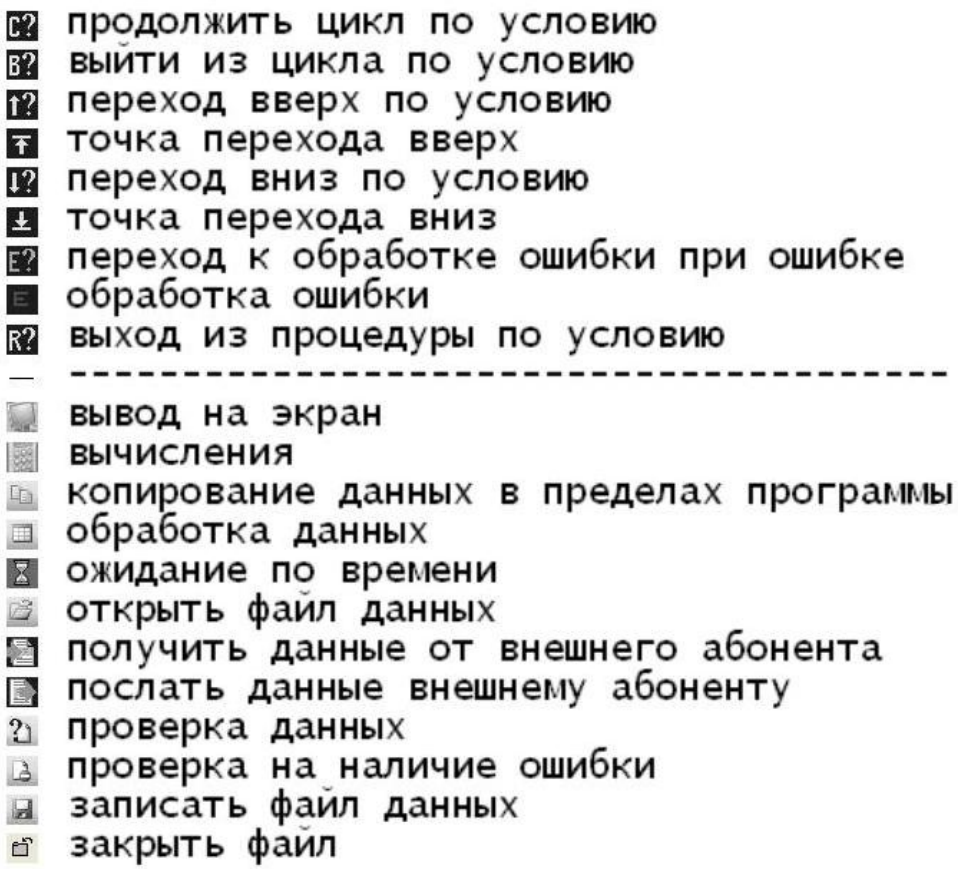


Рисунок 2 – Предлагаемый набор пиктограмм для визуальных элементов

ЦИКЛ-СИЛУЭТ

Пример цикла-силуэта приведен на рис.1.

Для создания цикла-силуэта нужно:

1) задать в окне ЛСП начало и конец цикла;
2) ввести в процедуре локальную переменную wetka и первым оператором процедуры сделать присвоение wetka := 1;
3) между началом и концом цикла ввести сложное условие, у которого число дополнительных блоков условия было бы равно числу веток силуэта – 1;
4) для каждого условия отредактировать его содержимое, задав:

if (wetka = N) then begin

5) задать нужные переходы к другим веткам, отредактировав соответствующие действия: wetka := N; (wetka := 0; - для выхода из цикла-силуэта);
6) для начала цикла силуэта задать содержимое: while (wetka <> 0) do begin ;
7) для конца цикла силуэта задать содержимое: end; .

БЛОК ПЕРЕКЛЮЧАТЕЛЕЙ

Пример блока переключателей приведен на рис.1.

Для задания блока переключателей нужно:

1) задать сложное условие, у которого число дополнительных блоков условия было бы равно числу действий переключателя – 1;
2) для каждого условия переключателя отредактировать его содержимое, задав для него определитель рисунка и сокращения вида `(*рис . . ДN *)` .

ОБОБЩЕННЫЙ ЦИКЛ

Внешний вид у обобщенного цикла такой же, как и у традиционного для блок-схем цикла "повторять до", а вот реальное содержание может соответствовать любому из типов циклов, в зависимости от контекста.

Ниже приведены визуальная схема силуэта с циклами и ее тестовое представление, при знакомстве с которыми станет понятно, что досрочный выход из цикла в языке ДАЛВЯЗ 2 может быть структурным (если он организован при помощи основного маршрутного оператора) или неструктурным (если он организован при помощи дополнительного маршрутного оператора).

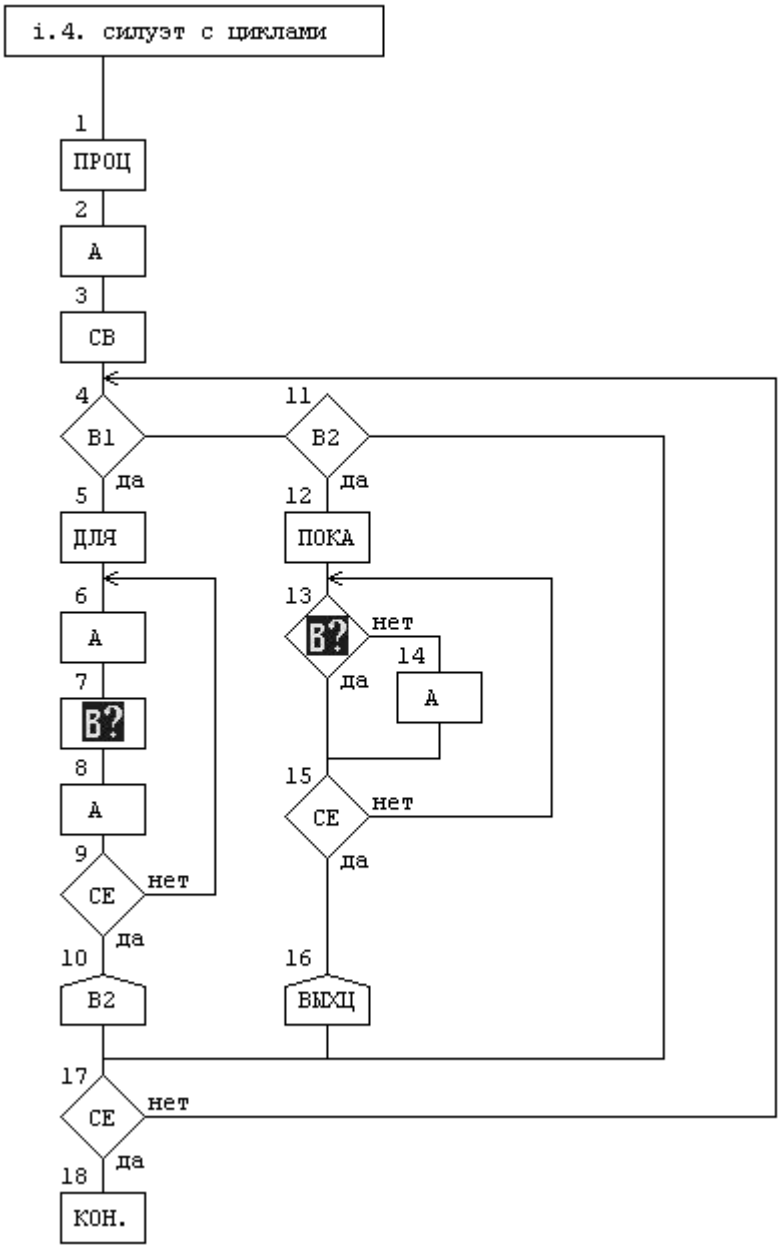


Рисунок 3 – Силуэт с циклами

```
(* i.4. силуэт с циклами *)
procedure  proc4;
var i, N, wetka:integer;
   (*i*)
   N := 10;   somevar := true;   usl_wyhoda := true;
   (*i CB *)  wetka := 1;
```

```
   while (wetka <> 0)  do begin
      (*i*)  if (wetka = 1)
         (*i CB *)   (*рис . .ДЛЯ *)
         for i:=1  to  N  do begin
            (*i*)
            (*i*)   (*рис .b? . *)
            if (usl_wyhoda)  then  break;
            (*i*)
         (*i CE *)  end;
         (*i*) wetka := 2;
      (*i*)  end else if (wetka = 2)
         (*i CB *)   (*рис . .ПОКА *)
         while (somevar = true)  do begin
            (*i*)  if (usl_wyhoda)   (*рис .b? . *)
                      then begin break;
            (*i*)  end else begin
               (*i*)  somevar := false;
            (*i*)  end;
         (*i CE *)  end;
         (*i*)  wetka := 0;
      (*i*)  end;
   (*i CE *)  end;
   (*i ВЫХОД *)
   end;
(*i КОНЕЦ *)
```

Для первой ветки зададим цикл for, а для второй – цикл while.

У первого цикла будет неструктурный досрочный выход:

```
            (*i*)   (*рис .b? . *)
            if (usl_wyhoda)  then  break;
```

У второго цикла досрочный выход будет структурным:

```
            (*i*)  if (usl_wyhoda)   (*рис .b? . *)
                      then begin break;
```

Как говорится, почувствуйте разницу.

Впрочем, у новичка одно восприятие (ему лучше, чтобы все логические связи были показаны явно – так проще разобраться) а у программиста со стажем работы 20 с лишним лет – совсем другое.

Но вопрос заключается в том, что если логика цикла достаточна сложна, а досрочный выход выполняется откуда-то из середины (имеется в виду из одной из центральных вертикалей цикла), то досрочный структурный выход из цикла вообще становится проблематичным. В таком случае не стоит мучить себя во имя излишней структурности – используйте неструктурный досрочный выход из цикла (или, если у вас есть избыток времени, перепишите процедуру, избавив содержимое цикла от излишних подробностей).

ПОЛНАЯ ФОРМА ОТОБРАЖЕНИЯ ВИЗУАЛЬНОЙ СХЕМЫ

Вопрос – писать ли полную форму отображения визуальной схемы – решат для себя самостоятельно будущие разработчики визуальной среды для ДАЛВЯЗ 2 (если таковые найдутся).

ЛИТЕРАТУРА И ССЫЛКИ В СЕТИ ИНТЕРНЕТ

1. В.Д. Паронджанов «Как улучшить работу ума: Алгоритмы без программистов – это очень просто !» Москва, Издательство «Дело», 2001

2. В.Д. Паронджанов «ЯЗЫК ДРАКОН краткое описание»

http://store.oberoncore.ru/lib/book/DrakonDescription.pdf

3. В.В. Фаронов «Основы Турбо Паскаля» Москва, Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992

4. http://forum.oberoncore.ru