УДК 004.4'23

И. Б. Жуков

АО НТЦ РОКАД
ул. Ломаная, 11, Санкт-Петербург, 196084, Россия

ibzh@yandex.ru

МЕТОДОЛОГИЯ ЛИТЕРАТУРНОГО ПРОГРАММИРОВАНИЯ. ОПЫТ РУСИФИКАЦИИ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

Изложены принципы литературного программирования, указаны основные возможности и приведены результаты адаптации инструментальных средств литературного программирования для их использования в России. Сложность адаптации состояла в том, что коды символов свыше 126 активно используются имеющимися инструментальными средствами для внутреннего представления данных и передачи управляющих кодов между различными частями программы. В процессе адаптации расширены возможности имеющихся инструментальных средств.

Ключевые слова: литературное программирование, грамотное программирование, Дональд Кнут, TeX, WEB, паскаль, инструментальные средства.

Существенным фактором, ограничивающим производительность труда программиста и вызывающим появление ошибок, является то, что человек не в состоянии удерживать в своем сознании и отслеживать взаимосвязи между большим числом объектов. Известно, что это число составляет от 7 до 13 в зависимости от особенностей индивида, причем это относится к любой области человеческой деятельности. В программировании приходится иметь дело с гораздо бóльшим числом объектов (командами и переменными). Поэтому развитие данной отрасли всегда шло по пути снижения их числа. Исходно программирование велось в машинных кодах, когда программист должен был следить среди прочего за правильностью набранных кодов команд и адресов размещения данных. Потом появился язык ассемблера, заменивший коды команд мнемониками, а адреса – символическими именами. Программист уже мог сосредоточиться на тех действиях, которые должен выполнить процессор. Затем появились языки высокого уровня, и тогда программист мог сосредоточиться на алгоритмическом описании решаемой задачи. Переход к каждому новому уровню абстракции всегда способствовал снижению количества допускаемых ошибок, ускорению процесса разработки и повышению функциональности разрабатываемого ПО.

Литературное программирование (ЛП) дает более высокий уровень абстракции, чем обычно предоставляют современные языки высокого уровня, и позволяет вначале проработать логику программы, сделать набросок того, что она должна делать, а затем уже детализировать ее части, доводя их до реализации на конкретном языке программирования.

Довольно часто люди, не работавшие с данной системой, воспринимают ЛП как очередное средство документирования готовых программ. На самом деле процесс разработки программы и описание ее работы при использовании ЛП тесно переплетены друг с другом. По мысли Д. Е. Кнута, предложившего данный подход в начале 80-х гг. XX в. и разработавшего необходимые инструментальные средства, программа пишется человеком для человека, а машинный код является побочным продуктом этого процесса [1]. Эффективность данного

Жуков И. Б. Методология литературного программирования. Опыт русификации инструментальных средств // Вестн. НГУ. Серия: Информационные технологии. 2017. Т. 15, № 2. С. 20–26.

ISSN 1818-7900. Вестник НГУ. Серия: Информационные технологии. 2017. Том 15, № 2

подхода была доказана самим Д. Кнутом, разработавшим при помощи ЛП большое количество крупных программ, среди которых издательская система TEX и программа генерации шрифтов METAFONT. Однако из-за того, что полностью реализовать возможности ЛП можно только при подробном описании разрабатываемого кода, к чему склонны не все программисты [2], данный подход не очень широко распространен за рубежом, а в России из-за отсутствия поддержки имеющимися инструментальными средствами кириллицы он практически неизвестен.

На сегодняшний день разработаны инструментальные средства для разных языков программирования (Java, Ada, C++ [3], Matlab и др.). Автором данной статьи выполнена русификация разработанных Д. Кнутом средств для языка Паскаль. Выбор обусловлен тем, что первоначально эти средства предполагалось использовать для трансляции сделанных автором переводов программ TEX и METAFONT.

Инструментальные средства включают в себя две программы: tangle и weave, а также стилевой файл webmac.tex. Совокупность этих средств Д. Кнут назвал системой WEB и описал в работе [4]. Программа tangle используется для преобразования программы, написанной на языке WEB (web-файла), в программу на языке Паскаль (pas-файл), а программа weave предназначена для преобразования программы WEB в документ на языке TEX (tex-файл). Стилевой файл содержит команды форматирования элементов в документе для системы верстки TEX.

На рис. 1 приведены возможные варианты обработки web-файла. К нему может быть применен файл изменений (chg-файл). Полученный на выходе weave tex-файл должен обрабатываться программой tex, использующей стилевые файлы plain.tex (стандартные макросы) и webmac.tex. Выданный ей dvi-файл можно просмотреть и распечатать с помощью специальных программ или преобразовать в pdf-формат программой dvipdfm.

Сейчас мы рассмотрим структуру программы на языке WEB и укажем некоторые возможности, предлагаемые это системой.

Программа на языке WEB представляет собой обычный текстовый файл, содержащий ряд однотипных разделов. Каждый раздел состоит из трех частей, любая из которых может отсутствовать:

- словесного описания на языке разметки TEX того, что должен делать следующий кусочек программы;
- макроопределений, которые обрабатывает система WEB;
- собственно, кусочка программы.

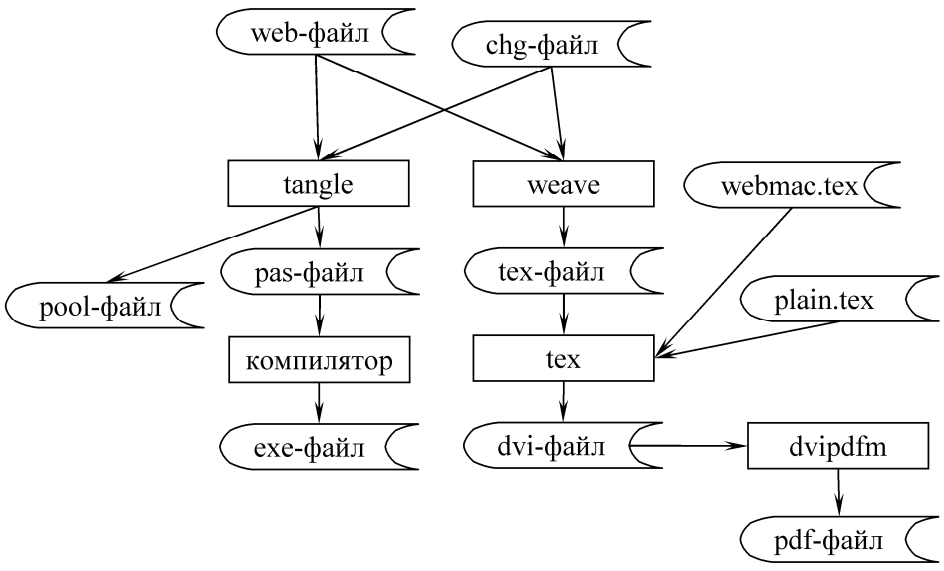

Рис. 1. Схема обработки WEB-файла

Каждый раздел начинается с символа @, за которым следует пробел или звездочка. В последнем случае название раздела (предложение до первой точки) программа weave выделяет в тексте жирным шрифтом и помещает в оглавление.

Макроопределения начинаются с команды @d, за которой следует имя макроса и его раскрытие. Форматирование имени макроса указывается командой @f.

Кусочки программы могут быть именованными и безымянными. Безымянные кусочки начинаются командой @p и вставляются программой tangle в pas-файл последовательно, в том порядке, в котором они идут в web-файле. Именованные кусочки начинаются с названия, указываемого между знаками @<…@>=, и вставляются только там, где на них есть ссылки. Ссылка представляет собой название кусочка, заключенное между знаками @<…@> (без знака равенства). Каждый именованный кусочек может вставляться в нескольких местах программы, в том числе и в других именованных кусочках. Именованный кусочек может описываться в нескольких разделах, тогда все эти описания последовательно соединяются.

Пример фрагмента программы и его форматирования приведен на рис. 2. Здесь есть один безымянный кусочек, в котором содержатся программа *a* и ссылки на два именованных кусочка <Some data input> и <Result output>. Указанные в конце названий кусочков цифры в форматированном фрагменте означают, что первый кусочек начинает определяться во втором разделе, а второй кусочек – в четвертом. Если именованный кусочек определен более чем в одном разделе, то в документе после первого раздела, где он начал определяться, будет приведен перечень номеров разделов, где этот кусочек определен (текст «See also section…»). Также указывается, в каких разделах этот кусочек используется (текст «This code is used in section…»).

Имя кусочка может содержать до четырех строк и, по сути, должно описывать то, что кусочек делает. Полностью имя нужно указывать только один раз. Далее к нему можно обращаться по его первой части, заменяя остаток многоточием. Если к первой части имени подходит больше одного кусочка, то будет выдано сообщение об ошибке.

```
@* Первый раздел программы.
Описание следующего кусочка
программы.
@d incr(#) = =  #:=#+1
@p
program a;
var x, y:integer;
begin
@<Some data input@>;
if x=1 then incr(y);
@<Result output@>;
end.
@ Ввод переменной |x|.

@<Some data input@>=
readln(x);



@ Ввод переменной |y|.

@<Some…@>=
readln(y);
```

1. Первый раздел программы. Описание следующего кусочка программы.

```
  define incr(#) ≡  # ← # + 1

program a;
  var x, y:integer;
  begin  <Some data input 2>;
       if  x = 1 then incr(y);
       <Result output 4>;
  end.
```

2. Ввод переменной *x*.

```
<Some data input 2>≡
readln(x);
```

See also section 3.

This code is used in section 1.

3. Ввод переменной *y*.

```
<Some data input 2>+≡
readln(y);
```

Рис. 2. Фрагмент WEB-файла слева и его форматирование в dvi-файле справа

ЛП позволяет сначала набросать общий план программы и затем постепенно уточнять отдельные его пункты, реализуя тем самым подход «сверху вниз». Иногда, наоборот, удобнее использовать поход «снизу вверх», когда пишутся сначала небольшие кусочки программы, а затем они объединяются в более крупные, что также возможно с помощью ЛП. Некоторые процедуры, например инициализация начальных значений переменных, могут постепенно дописываться на протяжении всей программы. Д. Кнут в своих работах использует все указанные подходы одновременно. Таким образом, ЛП позволяет описывать программу в том порядке, в котором человек мыслит, давая возможность концентрироваться на логике работы относительно небольшого числа объектов, что существенно упрощает процесс разработки.

Чтобы облегчить анализ чужих программ, weave при формировании документа создает перечень идентификаторов (имен переменных и функций) с указанием разделов, где они упоминаются и объявляются. В последнем случае номера подчеркиваются. Также составляется указатель именованных кусочков с указанием разделов, куда они включаются, и разделов, где они определяются.

Для того чтобы можно было легко вносить изменения в программу, не трогая исходный web-файл, программы tangle и weave используют файл изменений (с расширением chg). В этом файле между командами @x и @y указываются искомые строки, а между @y и @z строки, на которые нужно их заменить. Программа weave, собирая документ, помечает номера измененных разделов звездочкой.

Разработанные Д. Е. Кнутом инструментальные средства непосредственно применяться в России не могут, так как не позволяют использовать кириллицу. Это обусловлено тем, что вводимые из файлов данные фильтруются (символы с кодами больше 127 заменяются пробелами) и преобразуются в упакованные списки лексем. В программе tangle лексемы хранятся в массиве tok_mem, в котором коды меньше 127 представляют буквы латинского алфавита, а также особые знаки. Например, код 32 (знак пробела) представляет две точки «..». Коды от 128 до 167 указывают на идентификатор, коды от 168 до 200 – на номер модуля, и т. д. Коды свыше 127 также используются для внутреннего представления управляющих кодов, передаваемых наравне с кодами символов. В программе weave коды выше 127 также используются в памяти лексем и в выходном буфере для команд форматирования, ссылок на идентификаторы, управляющих кодов и др.

Кроме того, данные программы не позволяют использовать многие возможности современных компиляторов. В частности, free pascal допускает задавать строковые константы в виде «#1#2#3#4», а tangle рассматривает знак # как место возможного разрыва строки, что приводит к необходимости ручной правки кода. В именах идентификаторов допускается использовать знак «_», а tangle его отбрасывает, об этом не стоило бы упоминать, если бы некоторые библиотеки для free pascal не использовали этот знак в именах своих функций. Кроме того, tangle обрезает идентификаторы до первых семи знаков, что также ограничивает возможности его использования. Все это указывает на необходимость модификации разработанных Д. Кнутом программ.

Исходными файлами для создания русифицированных версий послужили tangle.web [5] и weave.web [6]. Модификация кода в них производилась с помощью программы tangle.exe из дистрибутива dostp22 путем применения файла изменений. В качестве среды программирования использовался свободно распространяемый компилятор free pascal. Обход «узких мест» при передаче символов кириллицы в программе выполнялся с помощью специально введенной для этого автором статьи булевой переменной fcyr_sign, указывающей, что передается не управляющий код, а знак кириллицы. В массиве tok_mem, хранящем упакованные лексемы, перед знаком кириллицы вставляется код cyr_switch=1, указывающий, что следующий за ним код должен трактоваться как символ, а не как ссылка на идентификатор или модуль.

Русифицированным версиям программ tangle и weave даны имена rutangle и ruweave соответственно. (Д. Кнут настаивает на том, чтобы любые модификации его программ имели собственные имена.) Они предназначены для работы в консольном режиме Windows. Входные (web-) и выходные (pas- и tex-) файлы должны иметь кодировку CP866. В этих програм-

мах переведены на русский язык все выводимые ими сообщения. Имена входных и выходных файлов могут быть русскими.

После завершения работы программы возвращают операционной системе значение переменной окружения errorlevel, которое можно использовать для пакетной обработки в командных файлах. (Исходные версии программ tangle и weave переменную errorlevel не устанавливали.) Если программа отработала успешно, возвращается 0, если найдены мелкие синтаксические ошибки – 1, если произошли серьезные ошибки, или задан несуществующий chg-файл – 2, значение 3 возвращается при фатальной ошибке.

Введена новая команда @s. Если первые символы строки @s или @S, то данная строка отбрасывается – как ruweave, так и rutangle. Эту команду можно использовать для комментариев или для исключения части кода из обработки. Перед @s и @S пробелов быть не должно.

Увеличены предельная длина строки символов и максимальная длина идентификатора.

Программа rutangle в отличие от исходного tangle в именах идентификаторов символ подчеркивания «_» не отбрасывает. (Иначе невозможно будет использовать некоторые библиотеки free pascal.) Также она не преобразует строчные буквы в прописные в именах идентификаторов. Программа rutangle не разрывает строки на знаке #. (В исходной версии этот знак считался удобным местом для разрыва строки в программе, что затрудняло запись строк кодами их символов.)

Добавлена возможность записывать числа в формате, принятом в ассемблере (в виде 200h). Free pascal в отличие от чистого языка Паскаль допускает запись чисел в таком виде. Исходный tangle, встречая знаки 200h, обязательно вставлял пробел между 200 и h, из-за чего free pascal не мог корректно обрабатывать эти числа. Программа rutangle вставляет пробел между десятичным числом и буквой, только если он стоит и в исходном web-файле.

В программе rutangle передача русских букв из web-файла в pas-файл зависит от контекста. В строках, заключенных в кавычки ("…" и '…') и при дословной передаче текста (текст между символами @=…@>), знаки с кодами выше 128 передаются как есть. В других местах web-программы (в идентификаторах) они транслитерируются и сравниваются с имеющимися латинскими именами, если найдено совпадение, то выдается предупреждение о несогласованности имен.

Русифицированной версии стилевого файла webmac.tex [7] дано название rwebmac.tex. В ней подключены русские шрифты, переведены названия элементов указателей, знаки ≤ и ≥ заменены принятыми в России ⩽ и ⩾. Данный стилевой файл должен использоваться совместно с пакетом cyrplain, с LaTeX он не совместим. Для случая, когда dvi-файл обрабатывается программой dvipdfm, добавлена поддержка гиперссылок для всех номеров разделов, указателей идентификаторов и имен модулей, а также оглавления. Сделаны закладки (bookmarks) для разделов, начинающихся командой @*. Гиперссылки и закладки добавляются с помощью команд \special, описанных в руководстве к dvipdfm, поэтому данный стилевой файл не совместим с программой pdftex.

При создании закладок rwebmac.tex делает русские буквы активными, чтобы заменять их соответствующим представлением в unicode. Поэтому он не совместим с пакетом inputenc, также делающим русские буквы активными.

При создании pdf-файла заглавные буквы Ш и Щ в закладках идут в сочетании с символом, изображаемым как квадрат. Дело в том, что программа dvipdfm, преобразующая dvi-файл в pdf, следит за равновесием открывающих и закрывающих скобок. Буквы Ш и Щ в unicode представляются двухбайтовыми последовательностями, второй байт которых содержит открывающую и закрывающую круглые скобки. Чтобы dvipdfm не завершал работу с ошибкой, после буквы Ш вставляется закрывающая, а перед буквой Щ открывающая круглые скобки. Чтобы они не отображались скобками в закладках, приняты неопределенные в unicode символы с кодами 0ff28h и 0ff29h, которые и отображаются квадратами. На вид печатного документа закладки никак не влияют, а читаемость закладок при таком решении сильно не ухудшается.

Работоспособность русифицированных программ проверялась тестовыми примерами на предмет корректности обработки ошибок, а также правильностью обработки крупных

программ TEX и METAFONT. На рис. 3 приведен снимок экрана с документом в формате pdf, полученным русифицированными средствами из перевода файла tex.web.

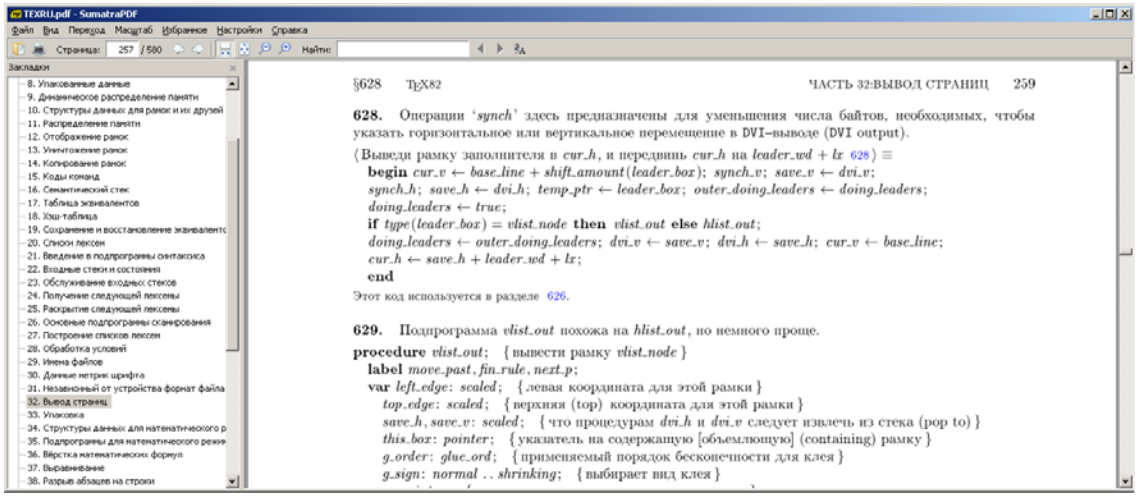

Рис. 3. Снимок экрана, отображающего отформатированный web-документ

Заключение

Разработанные Д. Кнутом методика и средства литературного программирования позволяют очень быстро писать собственные программы, разбираться в чужих, а также вносить в них необходимые изменения. Последнее обусловлено тем, что можно сосредоточиться на поиске и исправлении только нужных частей кода. Механизм файлов изменений позволяет не портить исходный текст чужой программы, благодаря чему, легко отменить ошибочные изменения.

Поскольку указанные средства не позволяют использовать кириллицу, то автором данной статьи в эти программы были внесены изменения, адаптирующие их для использования в России, а также составлено описание системы литературного программирования на русском языке.

Данные средства позволяют определять макросы с русскими именами, заменяемые при обработке rutangle ключевыми словами стандартного языка Паскаль, и, таким образом, осуществлять программирование полностью на русском языке.

Автор использовал русифицированные средства в своей профессиональной деятельности для разработки генераторов отчетов на WinAPI с использованием OLE-автоматизации.

Заинтересовавшиеся могут скачать программы rutangle, ruweave, стилевой файл rwebmacpdf и руководство по web на русском с сайта http://ibzh.novhost.cf/tex/literate.htm или http://ibzh.16mb.com/tex/literate.htm.

Список литературы

1. *Knuth D. E.* Literate Programming. Stanford, California: Centre for the Study of Language and Information – CSLI Lecture Notes, no. 27. 1992. 368 с.

2. *Шалыто А. А.* Новая инициатива в программировании. Движение за открытую проектную документацию // Информационно-управляющие системы. 2003. № 4. С. 52–56.

3. *Knuth D. E.*, *Levy S.* The CWEB System of Structured Documentation. Reading Massachusetts: Addison-Wesley, 1993. iv +277 p.

4. *Knuth D. E.* The WEB System of Structured Documentation. Department of Computer Science Stanford University. September 1983. Report № STAN-CS-83-980.

5. *Knuth D. E.* The TANGLE processor. Version 4.5 // Comprehensive TEX Archive Network. URL: ftp://ftp.dante.de/pub/tex/systems/knuth/dist/web/tangle.web (дата обращения 01.03.2015)
6. *Knuth D. E.* The WEAVE processor. Version 4.4 // Comprehensive TEX Archive Network. URL: ftp://ftp.dante.de/pub/tex/systems/knuth/dist/web/weave.web (дата обращения 01.03.2015)
7. *Knuth D. E.* Standard macros for WEB listings // Comprehensive TEX Archive Network. URL: ftp://ftp.dante.de/pub/tex/systems/knuth/dist/lib/webmac.tex (дата обращения 01.03.2015)

Материал поступил в редколлегию 20.04.2017

I. B. Zhukov

ROKAD
11 Lomanaya Str., St. Petersburg, 196084, Russian Federation

ibzh@yandex.ru

LITERATE PROGRAMMING METHODOLOGY. EXPERIENCE OF RUSSIFICATION OF DEVELOPMENT TOOLS

Approaches of literate programming are stated, basic capabilities are shown, and results of adopting development tools of literate programming for using in Russia are given. Complication of adopting is that symbol codes above 126 are actively used for data representing and control code transmitting between different parts of programs of existing development tools. During adaptation some new features were shipped.

Keywords: literate programming, Donald E. Knuth, TeX, WEB, pascal, development tools.

References

1. *Knuth, D. E.* Literate Programming / D. E. Knuth (Stanford, California: Centre for the Study of Language and Information – CSLI Lecture Notes, no. 27), 1992. 368 c.
2. *Shalyto A. A.* Novaia initciativa v programmirovanii. Dvizhenie za otqrytuiu proeqtnuiu doqumentatciiu [New initiative in programming. The movement for open project documentation] // Informatsionno-upravliaiushchie sistemy [Information and Control Systems]. 2003, № 4, pp. 52–56.
3. *Knuth, D. E., Levy, S.* The CWEB System of Structured Documentation. Reading Massachusetts: Addison-Wesley, 1993, iv +277 pp.
4. *Knuth, D. E.* The WEB System of Structured Documentation / Report № STAN-CS-83-980. Department of Computer Science Stanford University. September 1983.
5. *Knuth, D. E.* The TANGLE processor. Version 4.5 // Comprehensive TEX Archive Network. [Date of update 2002]. URL: ftp://ftp.dante.de/pub/tex/systems/knuth/dist/web/tangle.web (Date of access: 01.03.2015)
6. *Knuth, D. E.* The WEAVE processor. Version 4.4 // Comprehensive TEX Archive Network. [Date of update 1992]. URL: ftp://ftp.dante.de/pub/tex/systems/knuth/dist/web/weave.web (Date of access: 01.03.2015)
7. *Knuth, D. E.* Standard macros for WEB listings// Comprehensive TEX Archive Network. URL: ftp://ftp.dante.de/pub/tex/systems/knuth/dist/lib/webmac.tex (Date of access: 01.03.2015)