Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра "Компьютерные технологии"

М.А. Коротков, А.П. Лукьянова, А.А. Шалыто

Система управления взаимодействием скриптов в Web-программировании

Проектная документация

Проект создан в рамках "Движения за открытую проектную документацию" http://is.ifmo.ru

Оглавление

1.	Постановка задачи					
	1.1.	Система $RemEd$	3			
	1.2.	Основные требования	7			
2.	Описание инструкций языка SIL					
	2.1.	Грамматика языка SIL	11			
3.	Функц	ционирование системы SIL	13			
4.	Примеры					
	4.1.	Пример Simple	20			
	4.2.	Пример Cycle	23			
Закл	инэгон	ee	25			
Лите	ература	a	26			
Приложения						
	Прило	жение 1. Автомат "Компилятор" (A1)	27			
	Прило	жение 2. Протокол компиляции примера Simple	49			
	-	жение 3. Протокол компиляции примера Cycle				
	-	ожение 4. Автомат "Исполнитель" (А2)				
		жение 5. Функции автоматов				
	-	жение 6. Автомат "Процессор" (A3)				
	-	жение 7. Сценарий выбора пользователей				
	-	жение 8. Протокол компиляции сценария выбора пользователей				
	-	жение 9. Протокол интерпретации примера Simple				
		жение 10. Структура базы данных				
	-	жение 11. Листинг файла mslput.php				
	Прило	жение 12. Протокол интерпретации примера Cycle	97			

1. Постановка задачи

При разработке клиент-серверных систем часто оказывается удобным, а иногда и необходимым, использовать такие языки сценариев как, например, PHP или $Perl^1$. Ниже на примере системы дистанционного обучения $Remote\ Education\ System$ показаны основные проблемы, возникающие при разработке клиент-серверных систем на базе скриптовых языков. Бо́льшая часть этих проблем решается созданием надстройки над языком PHP, описываемой в этой работе.

1.1. Система RemEd

Рассмотрим клиент-серверную систему дистанционного обучения RemEd ($Remote\ Education\ System$). Она представляет собой комплекс программ на языках PHP и Java, являющийся полноценной средой дистанционного обучения. Эта система обеспечивает все механизмы взаимодействия преподавателя и студента в рамках учебного процесса: предоставление учебных материалов, проведение тестов и опросов, контроль деятельности студентов, возможность общения в форуме и чате.

Подробное описание системы RemEd не является задачей настоящей работы. Его можно найти в материалах Всероссийской конференции IST\IMS-2003, "Технологии информационного общества – Интернет и современное общество" (http://conf.infosoc.ru/03-rDOPf03.html,[2]). Ограничимся описанием подсистемы администрирования.

Для работы с системой администрирования используется Web-интерфейс, предоставляющий, в частности, следующие возможности:

- регистрация пользователей системы (студентов, преподавателей, редакторов учебных материалов);
- управление правами доступа к учебным курсам;
- настройка средств общения;
- настройка системы учета деятельности студентов.

На рис. 1 показана страница системы администрирования.

¹Все сказанное в данной работе про язык PHP относится и к языку Perl, [1].



Рис. 1. Возможности администратора

Система RemEd позволяет администратору осуществлять выбор пользователей по некоторому признаку (например, выбор всех преподавателей данного курса). Изобразим сценарий выбора пользователей в виде схемы (рис. 2). Каждый путь в этой схеме соответствует функции выбора.

 $^{^{2}}$ Сценарием называется совокупность скриптов и механизмов взаимодействия между ними, необходимых для реализации некоторой функции Web-системы.

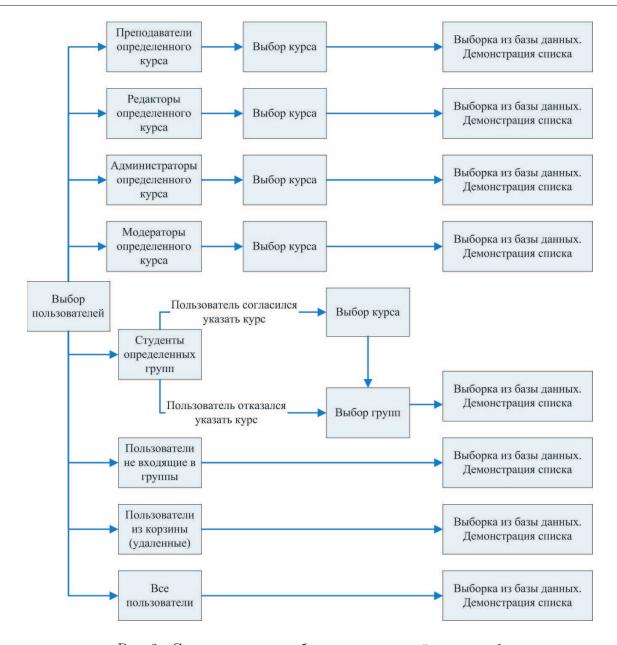


Рис. 2. Схема сценария выбора пользователей, вариант 1

Обратим внимание на то, что в этом сценарии задействовано большое количество блоков, многие из которых повторяются. Объединив на схеме дублирующиеся блоки, получим схему, представленную на рис. 3.

1.1. Система RemEd

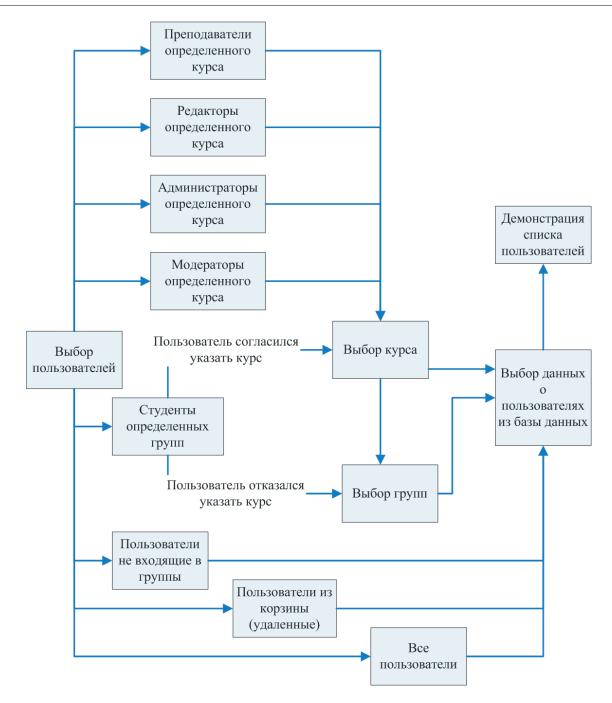


Рис. 3. Схема сценария выбора пользователей, вариант 2

Эта схема позволяет избежать дублирования кода при ее реализации (вследствие отсутствия в ней повторяющихся блоков). К сожалению, особенности языка PHP не позволяют непосредственно реализовать сценарий, соответствующий этой схеме, так как последняя не является древовидной. Это можно сделать, расширив функциональность языка PHP за счет введения некоторой надстройки. Описываемая в данной работе система $Script\ Interaction\ Language\ (SIL,\ Cucmema\ взаимодействия\ скриптов)$ и является

такой надстройкой.

Для объединения блоков система должна поддерживать следующие возможности:

- хранение некоторого количества параметров, которые могут использоваться любым блоком:
- управление переходами между блоками выбирая пользователей определенных групп можно как указывать, так и не указывать курс, среди групп которого будет производиться поиск;
- механизм вложения сценариев и обмена данными между ними. К примеру, сценарий выбора пользователей часто применяется при работе над другими задачами, в частности, над задачей регистрации нового преподавателя в системе (рис. 4). При этом указанный сценарий оказывается частью более крупного сценария.



Рис. 4. Схема сценария регистрации преподавателя

Целью настоящей работы является построение системы, обеспечивающей возможность объединения блоков сценария. Система **SIL** будет представлять собой совокупность следующих элементов:

- языка программирования **SIL**;
- компилятора языка **SIL** в байт-код;
- виртуальной машины **SIL**.

Заметим, что многие современные Web-ориентированные системы разработки, например Java([3]) и .NET([4]), состоят из того же набора структурных элементов.

1.2. Основные требования

Сформулируем требования к механизмам, поддержку которых необходимо обеспечить:

- механизмы работы с данными:
 - хранение данных;
 - передача параметров между скриптами;

- получение параметров GET (или POST) запроса³;
- механизм переходов между инструкциями программы;
- взаимодействие с *PHP*-скриптами;
- механизм подпрограмм.

Данным требованиям удовлетворяют языки инструкций (например, Assembler⁴). Язык **SIL** является простейшим языком инструкций. Его компилятор и виртуальная машина написаны на языке PHP.

Осталось выбрать механизм хранения данных программы. Рассмотрим подробнее его особенности, обусловленные использованием языка PHP и спецификой поставленной залачи:

- отсутствует потребность в типизации данных⁵;
- система должна обеспечивать только хранение и передачу данных. Их модификация будет осуществляться *PHP*-скриптами.

Наиболее распространенным механизмом доступа к данным является RAM⁶. Однако, ориентируясь на вышеперечисленные особенности, будем размещать данные в двух стеках. Это наиболее простая структура, обеспечивающая все необходимые возможности.

Язык **SIL** удовлетворяет всем сформулированным выше требованиям и, кроме того, обеспечивает:

- защиту кода программ от несанкционированной модификации;
- защиту от неавторизованного доступа к областям данных.

Похожую функциональность предоставляет механизм сессий в языке PHP. Однако он имеет ряд недостатков:

- объединение блоков в сложных сценариях крайне затруднительно;
- отсутствует механизм защиты от несанкционированного изменения кода программ и от неавторизованного доступа к данным.

Этих недостатков нет в языке **SIL**. Он предоставляет простой механизм манипуляции данными. Кроме того, программы, написанные на этом языке, компилируются в байт-код, что автоматически защищает их исходный код от несанкционированной модификации⁷ – исходный код **SIL** программ отсутствует на сервере.

³Описание этих запросов можно найти в работах [1], [5], [6].

⁴Информацию по фундаментальным идеям языков инструкций можно найти в работе [7], в части "Некоторые фундаментальные методы программирования".

 $^{^{5}}$ Язык PHP является языком с нестрогой типизацией.

 $^{^6}$ Random Access Memory – к любой ячейке памяти можно обратится по адресу, [8].

⁷Защиту можно усилить, [9], [10].

2. Описание инструкций языка SIL

Введем следующие обозначения:

```
і – целое шестнадцатибитное число;
```

b – целое восьмибитное число;

sn – номер стека (1 или 2);

esn — расширенный номер стека (0, 1 или 2);

ноль означает фиктивный пустой стек;

st - строка произвольного вида, не содержащая символов '\$' и '`;

nam — строка, соответствующая имени скрипта или метки.

Программа на языке SIL состоит из инструкций следующего вида.

- 1. Инструкции для работы с данными, хранящимися в стеках.
 - Инструкция копирования.

Краткое имя: сру

 $\Phi_{\text{OPMAT: CPY}}$ [sn'] [sn'] [i]

Описание: копировать i элементов из стека sn' в стек sn''.

• Инструкция перемещения.

Краткое имя: точ

Формат: mov [sn][esn][i]

Описание: переместить і элементов из стека sn в стек esn. Перемещение в фиктивный стек означает удаление.

• Инструкция записи.

Kраткое имя: put Формат: put [sn][st]

Описание: положить строку 8 st в стек sn.

• Инструкция очистки.

Kраткое имя: clr Формат: clr [sn]

Описание: очистить стек sn.

- 2. Инструкции логики переходов, отвечающие за порядок выполнения инструкций программы.
 - Инструкция метки.

Краткое имя: lbl Формат: lbl [nam]

Описание: поставить метку с именем nam.

 $^{^{8}}$ Как уже упоминалось выше, язык PHP является языком с нестрогой типизацией, поэтому под типом данных "строка" подразумевается обобщенный тип данных.

• Инструкция перехода на метку.

Краткое имя: jmp Формат: jmp [nam]

Описание: перейти на метку с именем nam.

• Инструкция перехода на метку из стека.

Kраткое имя: jss Формат: jss [sn]

Описание: извлечь из стека **sn** имя метки и перейти на нее.

• Инструкция перехода на инструкцию.

Kраткое имя: jsn Формат: jsn [sn]

Описание: извлечь номер инструкции из стека **sn** и перейти на нее.

• Инструкция пропуска.

Краткое имя: skp Формат: skp [sn]

Описание: пропустить взятое из стека sn число инструкций.

• Инструкция возврата.

Краткое имя: bck Формат: bck [sn]

Описание: вернуться на взятое из стека sn число инструкций назад.

• Инструкция "нет операции".

Краткое имя: nop Формат: nop

Описание: нет операции.

• Инструкция завершения.

Краткое имя: end Формат: end

Описание: выход из SIL-программы в вызвавшую программу или в систему.

- 3. Инструкции взаимодействия с PHP-скриптами. Эти инструкции позволяют запускать скрипты, передавать им данные и получать данные от них.
 - Инструкция получения параметра.

Kраткое имя: get Формат: get [sn] [nam]

Описание: получить параметр скрипта с именем **nam** и положить его в стек **sn**.

• Инструкция запуска.

Краткое имя: run

Φορмат: run [sn][b][esn][script_name]

Описание: запустить скрипт с именем script_name с передачей ему b параметров из стека sn и направлением вывода в стек esn.

• Инструкция запуска с выходом.

Краткое имя: off

Φορмат: off [b] [script_name]

Описание: запуск скрипта с именем script_name с передачей ему b параметров в формате GET запроса. Именами параметров являются переменные из первого стека, а их значениями – переменные из второго стека. SIL-программа при этом завершается.

• Инструкция обработки.

Краткое имя: ргс

Формат: prc [script_name]

Описание: запуск скрипта без передачи ему параметров. Обычно используется для обработки директив препроцессора, описанных ниже.

4. Директивы препроцессора. Обработка директив осуществляется следующим образом: после объявления имени программы может следовать произвольное число директив в формате

$$=$$
трехбуквенное имя $>$ [st],

где st — директива (произвольная строка, не содержащая символов \$). Значения директив возвращаются функцией

2.1. Грамматика языка SIL

Опишем грамматику языка **SIL** в несколько модифицированной форме Бэкуса-Наура. В стандартной нотации квадратные скобки обозначают необязательный элемент в формуле. Здесь же они являются символами алфавита.

```
< sil program >
                           ::= < name \ instruction > < newline >
                           < directive \ block > < instruction \ block >
< name instruction >
                           := nam[ < name \ value > ]
< directive \ block >
                           ::= \{ \langle directive \rangle \langle newline \rangle \}
< directive >
                           := $ < letter >< letter > [ < free value > ]
< instruction \ block >
                           ::= \{ < qet \ instruction > < newline > \}
                           {< loader \ command > < newline >} end
                           := get[ < sn \ value > ][ < name \ value > ]
< qet instruction >
< loader \ command >
                           ::= < clr instruction > | < cpy instruction > | < mov instruction > |
                           < put \ instruction > | < nop \ instruction > | < skp \ instruction > |
                           < bck | instruction > | < jsn | instruction > | < run | instruction > |
                           < off instruction > | < prc instruction > | < lbl instruction > |
                           < jmp\_instruction > | < jss\_instruction > | < cll instruction > |
< clr instruction >
                           ::= clr[ < sn \ value > ]
                           ::= cpy[ < sn\_value > ][ < sn\_value > ][ < number\_value > ]
< cpy\_instruction >
< mov instruction >
                           := mov[ < sn \ value > ][ < esn \ value > ][ < number \ value > ]
                           ::= put[ < sn\_value > ][ < free \ value > ]
< put instruction >
< nop instruction >
                           := nop
< skp instruction >
                           ::= skp[ < sn \ value > ]
                           ::= bck[ < sn \ value > ]
< bck instruction >
                           := jsn[ < sn \ value > ]
< jsn instruction >
                           ::= run[ < sn \ value > ][ < number \ value > ]
< run instruction >
                           [ < esn\_value > ][ < name\_value > ]
< off instruction >
                           := off[ < number \ value > ][ < name \ value > ]
< prc\_instruction >
                           ::= prc[ < name \ value >
< lbl instruction >
                           ::= lb1[ < name \ value > ]
                           ::= jmp[ < name \ value > ]
< jmp instruction >
< iss instruction >
                           := iss[ < sn \ value > ]
                           ::= cll[ < name | value > ]
< cll instruction >
\langle sn \ value \rangle
                           ::= 1|2
< esn \quad value >
                           ::= 0 | < sn \quad value >
< number value >
                           ::= < decimal > \{ < decimal > \}
                           ::= \{ \langle decimal \rangle \mid \langle letter \rangle \mid \langle symbol \rangle \}
< free value >
< name \quad value >
                           ::= \{ \langle decimal \rangle \mid \langle letter \rangle \mid \mid : \mid \setminus \}
< decimal >
                           ::= 0|1|2|3|4|5|6|7|8|9
< letter >
                           ::= a|b| \dots |y|z|A|B| \dots |Y|Z
\langle symbol \rangle
                           ::=, |.|(|)|; |:||!|$|%|*|-|_|+|=|?|
< newline >
                           ::= newline \ symbol
```

3. Функционирование системы SIL

Жизненный цикл SIL-программы разбивается на четыре шага.

- 1. Написание **SIL**-программы.
- 2. Компиляция.
- 3. Загрузка в базу данных.
- 4. Исполнение программы.

Схема жизненного цикла программы приведена на рис. 5.

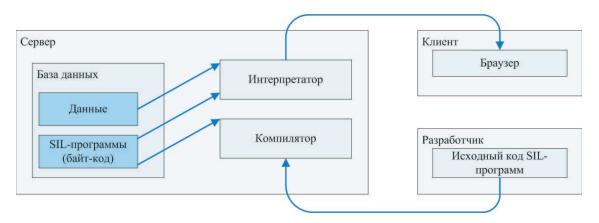


Рис. 5. Жизненный цикл программы

Опишем его более подробно.

Шаг первый. Написание SIL-программы.

Шаг второй. Компиляция. SIL-программа компилируется в байт-код. Компиляция осуществляется скриптом mslexec.php (приложение 1), который написан в соответствии с парадигмой автоматного программирования с явным выделением состояний ([11]). Этот скрипт реализует автомат "Компилятор" (A1). Его схема связей и граф переходов представлены на рис. 6, 7.

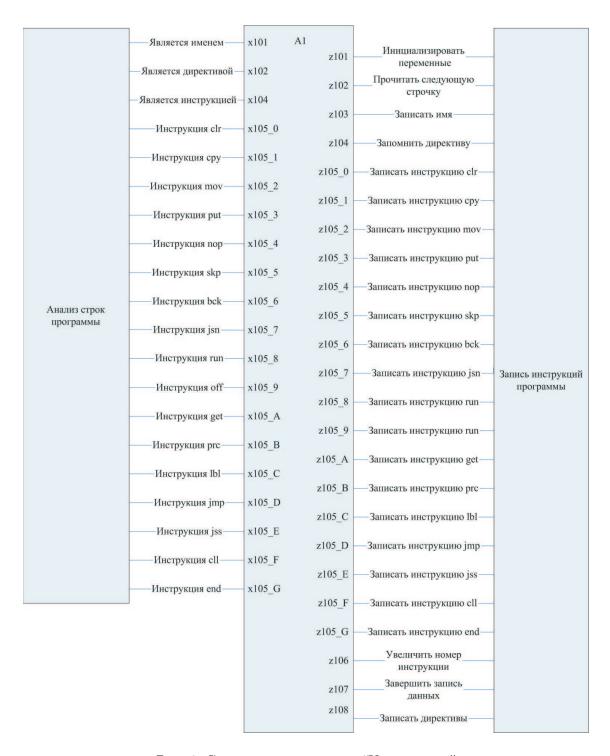


Рис. 6. Схема связей автомата "Компилятор"

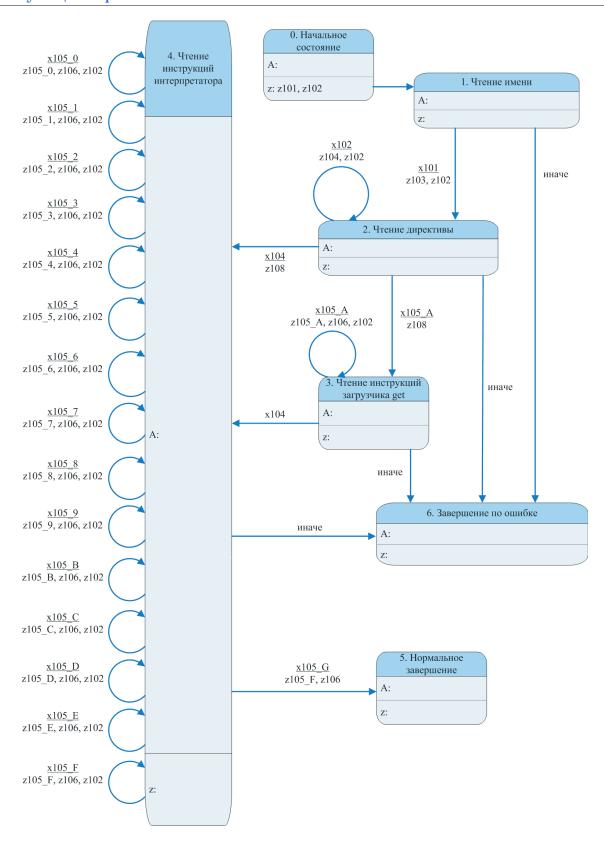


Рис. 7. Граф переходов автомата "Компилятор"

Компилятор выдает следующие результаты.

- 1. Файл (*.bin) с кодом всех **SIL**-программ, из которого они впоследствии будут загружены в базу данных на сервере.
- 2. На экран выводится набор кратких сообщений о проделанных операциях и о возникших во время компиляции ошибках.
- 3. В отдельном файле сохраняется протокол работы автомата "Компилятор" (A1) (приложения 2,3).

Шаг третий. Загрузка в базу данных. Байт-код программы загружается в базу данных.

Шаг четвертый. Исполнение программы. Система загружает программу из базы, создает область данных и выполняет инструкции программы. Каждой паре "пользователь-программа" сопоставляется область данных, идентифицируемая уникальным ключом. Он генерируется при каждом действии, что исключает возможность работы со страницами, содержащими устаревшие данные⁹.

Заметим, что у описанного механизма есть недостаток, заключающийся в потребности самостоятельно обрабатывать функцию возврата на предыдущую страницу, обычно обеспечиваемую браузером 10 .

Для того, чтобы запустить программу необходимо вызвать скрипт executor.php (приложения 4,5), передав ему ее имя. Этот скрипт, также как и компилятор, написан с использованием автоматной технологии. Он реализует автомат "Исполнитель" (A2), схема связей и граф переходов которого приведены на рис. 8, 9. Основная функция этого автомата заключается в формировании области данных SIL-программы. Кроме того, он реализует функции препроцессора языка – разбор инструкций получения параметра (get).

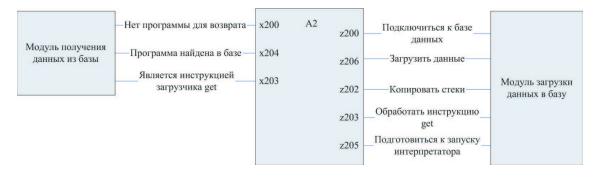


Рис. 8. Схема связей автомата "Исполнитель"

⁹Сервер не примет данные из формы, сохраненной, например, в кеше браузера, в случае, если они устарели.

 $^{^{10}}$ Кроме того, в такой системе скорее всего придется запретить кеширование на стороне клиента.

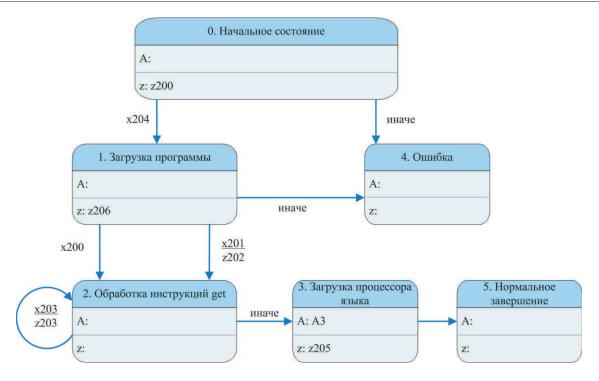


Рис. 9. Граф переходов автомата "Исполнитель"

После завершения работы исполнителя вызывается процессор языка.

Автомат "Процессор" (A3), реализуемый скриптом loader.php (приложения 5,6), производит разбор остальных инструкций SIL-программы и их исполнение. Его схема связей и граф переходов представлены на рис. 10, 11. Автомат реализует достаточно своеобразное поведение, поддерживая следующие возможности:

- вызов вложенного автомата "Исполнитель" (инструкция cll). Обратим внимание на то, что в скриптовом программировании, в отличие от объектного, вызов автомата подразумевает включение кода, а не вызов функции. Классический механизм вызова подпрограмм (стек вызовов) здесь не является удобным, поскольку усложняет структуру базы данных, не обеспечивая при этом существенных преимуществ;
- сохранение состояния автомата в базе данных;
- загрузка состояния автомата из базы данных. Внешний скрипт, используя идентификатор области данных программы, может запустить автомат "Процессор" (A3) так, чтобы тот возобновил работу с состояния, на котором его исполнение было приостановлено.

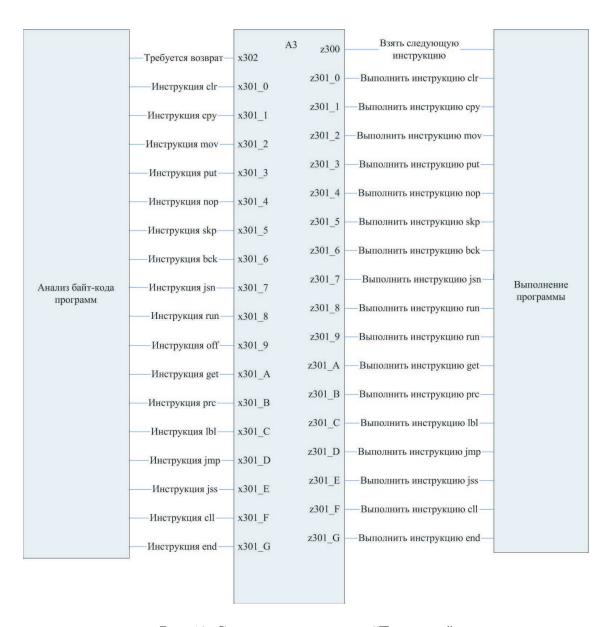


Рис. 10. Схема связей автомата "Процессор"

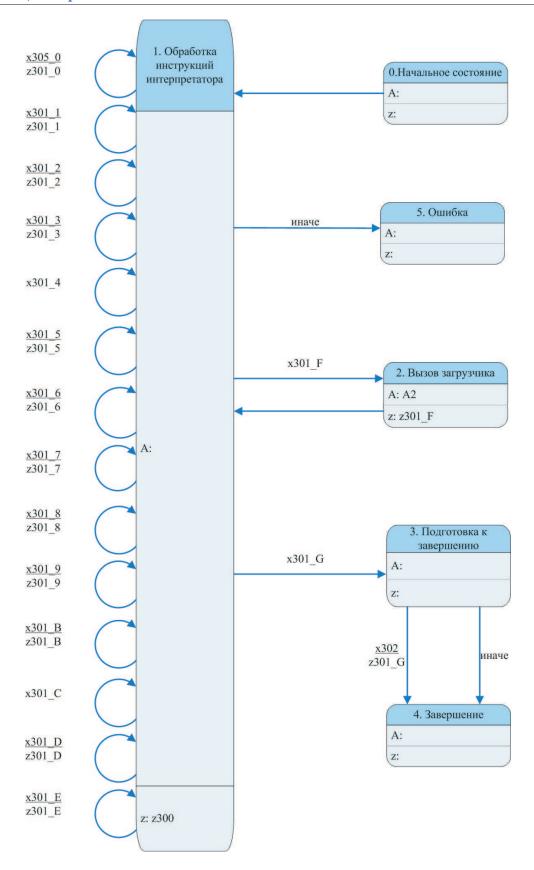


Рис. 11. Граф переходов автомата "Процессор"

4. Примеры 20

4. Примеры

Приведенные в этом разделе примеры предназначены исключительно для демонстрации основных принципов работы системы. Более содержательный пример (реализация сценария выбора пользователей системы RemEd) приведен в приложении 7, а протокол его компиляции - в приложении 8.

4.1. Π pumep Simple

Шаг первый. Написание SIL-программы. SIL-программа пишется и сохраняется в файл simple.msl.

```
nam[simple]
put[1][Сумма этих чисел равна:]
put[1][25]
put[1][30]
run[1][3][0][script0.php]
end
```

Прокомментируем эту программу.

Первая инструкция

```
nam[simple]
```

объявляет имя программы.

Следующие три инструкции

```
put[1][Сумма этих чисел равна:]
put[1][25]
put[1][30]
```

помещают в первый стек три элемента – строку 'Сумма этих чисел равна: 'и два числа: '25' и '30'.

Пятая инструкция

```
run[1][3][0][script0.php]
```

запускает PHP-скрипт script0.php, предварительно поместив в массив sinp три верхних элемента из первого стека (элементы из стека удаляются).

Рассмотрим код скрипта script0.php:

```
<?php
global $inp;
echo "<h3>Первое число: ".$inp[0]."</h3>";
echo "<h3>Второе число: ".$inp[1]."</h3>";
echo "<h3>".$inp[2]." ".($inp[0]+$inp[1])."</h3>";
?>
```

Данные, передаваемые скрипту, находятся в массиве \$inp. Первым его элементом будет последнее помещенное в стек число ('30'). Вторым станет число '25'. И последним является строка 'Сумма этих чисел равна:'.

Этот скрипт покажет в окне браузера следующий результат (рис. 12).

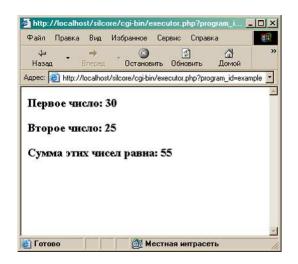


Рис. 12. Пример Simple. Скриншот 1

Поясним текст скрипта для тех, кто не знаком с языком PHP. Скрипт формирует html-страницу, которую затем показывает браузер. Оператор **echo** записывает строку в html-страницу. Строки конкатенируются точками. Имя переменной всегда начинается с символа '\$'.

Шаг второй. Компиляция SIL-программы. Текст SIL-программы, сохраненный в файле simple.msl, компилируется в байт-код¹¹ с помощью скрипта mslexec.php. Полученный байт-код сохраняется в файл scenario.bin.

Протокол работы компилятора над данным сценарием приведен в приложении 9.

Шаг третий. Загрузка в базу данных. Все скомпилированные на предыдущем шаге сценарии (в данном случае — только один, под названием simple), загружаются из файла scenario.bin в базу данных (приложение 10) скриптом mslput.php (приложение 11). Они хранятся в таблице program_texts следующего формата (рис. 13):

 $^{^{11}{}m B}$ данной документации мы не будем останавливаться подробно на структуре байт-кода, так как для понимания принципов работы системы ${
m SIL}$ этот вопрос не имеет сколько-нибудь принципиального значения.

program_name	program	program_num	program_directives	program_labels
I				I
			!	

Рис. 13. Схема данных

Каждому сценарию соответствует запись в таблице program_texts, содержащая следующую информацию:

```
program_name–имя программы,program–байт-код программы (кроме меток и директив),program_num–строка адресов инструкций в байт-коде,program_directives–строка директив,program labels–строка меток.
```

Шаг четвертый. Исполнение. Чтобы запустить **SIL**-программу, необходима стартовая html-страница:

```
<a href="executor.php?program_id=simple">Запустить сценарий Simple</a>
```

Эта страница содержит единственную ссылку с текстом 'Запустить сценарий Simple'. При переходе по ней запускается скрипт executor.php с параметром program_id = simple, задающим имя SIL-программы.

Скрипт executor.php (автомат "Исполнитель" (A2)) находит в базе данных в таблице program_texts запись, соответствующую программе с именем simple, получает все необходимые для ее исполнения данные (байт-код, строку меток, строку директив и т. д.), и запускает процессор языка (автомат "Процессор" (A3)), который последовательно выполняет все инструкции SIL-программы.

В данном случае это означает, что скрипт записывает в массив \$inp три элемента – строку 'Сумма этих чисел равна:' и два числа '25' и '30'. После этого подключается скрипт script0.php, который выводит три строки в html-страницу, используя данные из массива \$inp, и демонстрирует ее в окне браузера (рис. 14).

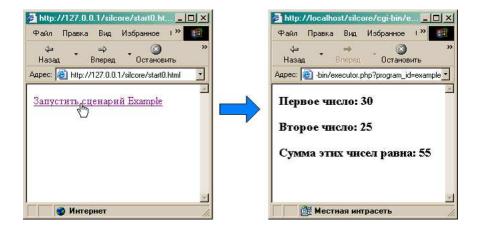


Рис. 14. Пример Simple. Скриншот 2

4.2. Пример Cycle

Описанная выше **SIL**-программа **simple** была приведена с единственной целью – продемонстрировать на максимально простом примере работу системы. Она использует лишь небольшую часть инструкций языка **SIL**.

Теперь рассмотрим пример более сложной программы:

```
nam[cycle]
put[2][-1]
lbl[start]
put[1][Hello world!]
run[1][1][0][printString.php]
run[2][1][2][increment.php]
cpy[2][2][1]
skp[2]
jmp[start]
put[1][Good bye!]
run[1][1][0][printString.php]
end
```

Прокоментируем эту программу. Первая инструкция

```
nam[cycle]
```

объявляет имя програмы: cycle.

В начале программы помещаем во второй стек число '-1'.

```
put [2] [-1]
```

В третьей строке стоит метка с именем 'start'.

```
lbl[start]
```

Далее следует уже описанная в предыдущем примере конструкция:

```
put[1][Hello world!]
run[1][1][0][printString.php]
```

В первый стек добавляется строка 'Hello world!'. После этого вызывается скрипт printString.php, выводящий эту строку в html-страницу:

```
<?php
   global $inp;
   echo $inp[0]."<br>";
?>
```

Следующая инструкция **SIL**-программы вызывает другой скрипт – increment.php, передавая ему один параметр из второго стека, и давая доступ к записи во второй стек

```
run [2] [1] [2] [increment.php]
```

Скрипт прибавляет единицу к переданному ему параметру и помещает полученное число в стек:

```
<?php
  global $inp;
  global $output_type;
  mslutils_stack_put($inp[0]+1);
  echo "В стек номер ".$output_type." добавлен элемент: ".($inp
      [0]+1)."<br>";
?>
```

Процедура mslutils_stack_put(x) добавляет элемент x в стек вывода скрипта (в данном случае – во второй стек).

Инструкция

```
cpy[2][2][1]
```

копирует один элемент из второго стека в него же. Таким образом, если в вершине стека находился элемент st, то после выполнения этой инструкции там будет находиться пара элементов st, st.

Следующая инструкция

```
skp[2]
```

Интерпретатор пропускает взятое из второго стека число инструкций.

Инструкция

```
jmp[start]
```

означает безусловный переход на метку с именем 'start' (третья инструкция этой программы).

Оставшиеся три строки SIL-программы завершают ее работу, выводя в html-страницу строку 'Good bye!' с использованием уже описанного скрипта printString.php:

```
put[1][Good bye!]
run[1][1][0][printString.php]
end
```

Для запуска **SIL**-программы **cycle** будем использовать *html*-страницу, аналогичную описанной в предыдущем примере. По выполнении программы браузер клиента отобразит страницу, показанную на рис. 15.



Рис. 15. Пример Cycle. Скриншот

Протокол работы компилятора над данным сценарием приведен в приложении 12.

Заключение

В настоящей работе описан язык **SIL** версии 2.0. Предыдущая версия была применена при построении клиент-серверной системы дистанционного обучения *Remote Education System*.

В свою очередь, новая версия языка предоставляет расширенные возможности по работе с директивами препроцессора, более эффективный интерпретатор и компилятор с полной проверкой синтаксиса. Это позволяет говорить о системе SIL как об удобном и эффективном механизме построения сценариев взаимодействия скриптов в Webприложениях и клиент-серверных системах. Использование автоматного подхода позволило повысить устойчивость системы 12 , а также предоставить разработчику удобный механизм отладки, что особенно важно в связи с *отсутствием* таких механизмов в языке PHP.

 $^{^{12}}$ Например, в случае получения на вход некорректных данных система **SIL** выведет в протокол сообщение об ошибке.

Литература

- [1] Ливингстон Д., Белью К., Браун М. Perl 5. BHV, 2001.
- [2] Солдаткин В. И. Российский портал открытого образования. Московский государственный институт управления, 2003.
- [3] *Ноутон П., Шилдт Л.* Java 2. СПб.: БХВ-Петербург, 2001.
- [4] Esposito D. Building Web Solutions with ASP.NET. Microsoft Press, 2002.
- [5] Швендимен Б. РНР 4. Руководство разработчика. Вильямс, 2002.
- [6] Геншвинде Э., Шениг Г. Разработка Web-приложений. ДиаСофт, 2003.
- [7] $\mathit{Khym}\ \mathcal{A}$. Искусство программирования. Том 1. Основные алгоритмы. Вильямс, 2000.
- [8] $Axo~A.,~Xonкpoфm~\mathcal{A}.,~Ульман~\mathcal{A}$ эк. Структуры данных и алгоритмы. Вильямс, 2001.
- [9] Чмора А. Л. Современная прикладная криптография. М.: Гелиос АРВ, 1986.
- [10] Введение в криптографию. Под общ. ред. В. В. Ященко. М.: МЦНМО, 1999.
- [11] *Шалыто А. А.* Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.

Приложения

В этом разделе приведены листинги наиболее важных модулей системы, протоколы и листинг базы данных. Дистрибутив системы можно найти на сайте http://is.ifmo.ru.

Приложение 1. Автомат "Компилятор" (А1)

Файл mslexec.php

```
<?php
/**
  UNIT..... mslexec.php
  PROJECT . . . : SIL
  AUTHOR....: М. Коротков, А. Лукьянова
  CREATED . . . . : 13.06.2003
  PURPOSE . . . : Компилятор SIL программ
  FUNCTIONS . . :
                mslexec_log_error
                mslexec_log_condition
                mslexec_log_action
                mslexec_x101
                mslexec_x102
                mslexec_x104
                mslexec_x105_0
                mslexec_x105_1
                mslexec_x105_2
                mslexec_x105_3
                mslexec_x105_4
                mslexec_x105_5
                mslexec_x105_6
                mslexec_x105_7
                mslexec_x105_8
                mslexec_x105_9
                mslexec_x105_A
                mslexec_x105_B
                mslexec_x105_C
                mslexec_x105_D
                mslexec_x105_E
                mslexec_x105_F
                mslexec_x105_G
                mslexec_z101
                mslexec_z102
                mslexec_z103
```

```
mslexec_z104
             mslexec_z105_0
             mslexec_z105_1
             mslexec_z105_2
             mslexec_z105_3
             mslexec_z105_4
             mslexec_z105_5
             mslexec_z105_6
             mslexec_z105_7
             mslexec_z105_8
             mslexec_z105_9
             mslexec_z105_A
             mslexec_z105_B
             mslexec_z105_C
             mslexec_z105_D
             mslexec_z105_E
             mslexec_z105_F
             mslexec_z105_G
             mslexec_z106
             mslexec_z107
             mslexec_z108
PARAMS....: $msl -- получаемый на вход makefile
             $outfile -- файл для сохранения сценариев
NOTES....: данный модуль получает на вход
               makefile со списком программ
               на языке SIL;
             компилирует каждую программу,
               помещая байт-код всех программ
               в указанный бинарный файл;
             ведет лог компиляции каждой
               программы в файле
               log_<имя программы>.txt в
               подкаталоге \logs
               рабочего каталога.
HISTORY . . . : DATE
                          NOTES
             _____
                          Создан
             14.06.2003
             22.11.2003 Перемещен в проект SIL
             25.11.2003 Полноценное использование
```

```
автоматного подхода
*/
if (!function_exists("paths_escape")) require_once("paths.php")
require_once(CGI_PATH.SN_DB); // модуль работы с БД
require_once(CGI_PATH.SN_MSLUTILS); // утилиты работы с MSL
require_once(CGI_PATH.SN_CONSTS); // константы
echo "<head><link rel=\"stylesheet\" type=\"text/css\" href=\"
  main.css\"/></head><body>";
/* Открытие входного файла */
global $msl;
if ( !isset($msl) ) {
    echo "No file sent!";
    exit;
}
db_connect(); // подключение к БД
  echo "<h1>Begin Log</h1>";
  /* Открытие makefile 'a */
  $makefile = fopen($msl, "r") or
    exit("Cannot open file: \"$msl\"");
  /* Создание файла для компилированного msl кода */
  $msl_binary_file = fopen($outfile, "w") or
    exit("Cannot open file: ".$outfile);
  while ( !feof($makefile) )
    $file = trim(fgets($makefile)); // читаем имя программы
    if ( $file != "" )
      echo "<h2>Begin: ".$file."</h2>"; // лог работы
                                         // в окне браузера
      $mslf = dirname($msl)."/".$file;
                                        // полный путь
                                         //к коду программы
      /* Открытие файла с кодом программы */
      $msl_code_file = fopen($mslf, "r");
```

```
/* Создание файла для ведения лога компиляции */
global $log_file;
global $log_file_txt;
global $spaces;
mslexec_log_open(dirname($msl)."/logs/log_".substr($file
  , 0, strlen($file)-4));
/* Строчка, содержащая номера символов в коде,
  соответствующие
   каждой иструкции (по 2 байта на номер)
                                                   */
$num_line = "";
/* Бинарник - код программы, помещенный в строчку */
$bin_line = "":
/* Строка директив программы в формате
   $имя_директивы$значение_директивы$...
   имя директивы обязательно имеет длину 3 буквы
   */
$dir_line = "";
/* Строка меток программы в формате
   $имя_метки$значение_метки$...
   значение метки - неупакованное число,
   соответствующее номеру инструкции,
   на которой встретилась метка
   */
$1b1_line = "";
/* Номер текущей инструкции */
curr = 0;
/* Номер символа, соответствующего
   началу текущей инструкции */
curr_num = 0;
/* Строка с прочитанным кодом очередной инструкции*/
$line = "";
$line_count = 0;
$program_id = $file;
st = 0;
```

```
mslexec_log_start("A1");
АВТОМАТ КОМПИЛЯТОР
while ($st <> 8){
      switch ($st) {
        case 0:
           mslexec_log_state($st, "Начальное состояние");
           mslexec_z101(); mslexec_z102();
            st = 1:
          break;
        case 1:
           mslexec_log_state($st, "Чтение имени");
           if (mslexec_x101()) {
             mslexec_z103(); mslexec_z102();
             st = 2;
           } else {
             mslexec_log_error("Отсутствие имени у программы
             st = 6;
          break;
        case 2:
           mslexec_log_state($st, "Чтение директив");
           if (mslexec_x102()) {
             mslexec_z104(); mslexec_z102();
             st = 2;
           }
            elseif (mslexec_x105_A()) {
             mslexec_z108();
             st = 3;
           }
            elseif (mslexec_x104()) {
             mslexec_z108();
             st = 4;
           }
            else {
             mslexec_log_error("Ошибка на стадии разбора
                директив");
```

```
st = 6;
    }
  break;
case 3:
    mslexec_log_state($st, "Чтение инструкций
       загрузчика get");
    if (mslexec_x105_A()) {
      mslexec_z105_A(); mslexec_z106(); mslexec_z102
         ();
      st = 3;
    elseif (mslexec_x104()) {
      st = 4;
    else {
      mslexec_log_error("Ошибка на стадии разбора
        инструкций get");
      st = 6;
    }
  break;
case 4:
    mslexec_log_state($st, "Чтение инструкций
      интерпретатора");
    if (mslexec_x105_0()) {
      mslexec_z105_0();
      mslexec_z106();
      mslexec_z102();
      st = 4;
    }
    elseif (mslexec_x105_1()) {
      mslexec_z105_1();
      mslexec_z106();
      mslexec_z102();
      st = 4;
    }
    elseif (mslexec_x105_2()) {
      mslexec_z105_2();
      mslexec_z106();
      mslexec_z102();
      st = 4;
```

```
}
elseif (mslexec_x105_3()) {
  mslexec_z105_3();
  mslexec_z106();
  mslexec_z102();
  st = 4;
elseif (mslexec_x105_4()) {
  mslexec_z105_4();
  mslexec_z106();
  mslexec_z102();
  st = 4;
}
elseif (mslexec_x105_5()) {
  mslexec_z105_5();
  mslexec_z106();
  mslexec_z102();
  st = 4;
}
elseif (mslexec_x105_6()) {
  mslexec_z105_6();
  mslexec_z106();
  mslexec_z102();
  st = 4;
}
elseif (mslexec_x105_7()) {
  mslexec_z105_7();
  mslexec_z106();
  mslexec_z102();
  st = 4;
elseif (mslexec_x105_8()) {
  mslexec_z105_8();
  mslexec_z106();
  mslexec_z102();
  st = 4;
}
elseif (mslexec_x105_9()) {
  mslexec_z105_9();
  mslexec_z106();
  mslexec_z102();
  st = 4;
```

```
}
  elseif (mslexec_x105_B()) {
    mslexec_z105_B();
    mslexec_z106();
    mslexec_z102();
    st = 4;
  }
  elseif (mslexec_x105_C()) {
    mslexec_z105_C();
    mslexec_z106();
    mslexec_z102();
    st = 4;
 }
  elseif (mslexec_x105_D()) {
    mslexec_z105_D();
    mslexec_z106();
   mslexec_z102();
    st = 4;
  }
  elseif (mslexec_x105_E()) {
   mslexec_z105_E();
    mslexec_z106();
    mslexec_z102();
    st = 4;
  }
  elseif (mslexec_x105_F()) {
    mslexec_z105_F();
    mslexec_z106();
   mslexec_z102();
    st = 4;
  }
  elseif (mslexec_x105_G()) {
    mslexec_z105_G();
    mslexec_z106();
    st = 5;
  } else {
    mslexec_log_error("Неверная инструкция
      интерпретатора");
    st = 6;
  }
break;
```

```
case 5:
            mslexec_log_state($st, "Нормальное завершение.");
            mslexec_z107();
            st = 8;
          break;
        case 6:
            mslexec_log_state($st, "Завершение по ошибке.");
            st = 8;
          break;
      }
     }
     /* Закрытие файлов */
     mslexec_log_finish("A1");
     fclose($msl_code_file);
     mslexec_log_close();
     /* Лог успешного окончания
        компиляции программы в браузер */
     echo "<h2>End: ".$file."</h2>";
   }
 }
 fclose($makefile);
 fwrite($msl_binary_file,"0");
 fclose($msl_binary_file);
     /* Лог успешного окончания
        работы в браузер */
 echo "<h1>End Log</h1></body>";
ФУНКЦИИ, РЕАЛИЗУЮЩИЕ ВХОДНЫЕ ПЕРЕМЕННЫЕ
function mslexec_x101()
 global $line;
 sec = ereg("nam \setminus [[a-zA-Z0-9_]* \setminus ]", sline);
 mslexec_log_condition("x101 - является ли именем", $res);
```

```
return $res;
}
function mslexec_x102()
  global $line;
  $res = (($line[0] == '$') &&
          (ereg("[a-zA-Z0-9_]{3}\[[a-zA-Z0-9_\-]*\]", substr(
             $line, 1, strlen($line)-1)));
 mslexec_log_condition("x102 - является ли директивой", $res);
 return $res;
}
function mslexec_x104()
{
  global $line;
  sec = (ereg("^[a-z]{3}", sline));
 mslexec_log_condition("x104 - является ли инструкцией
    интерпретатора", $res);
 return $res;
}
function mslexec_x105_0()
  global $line;
  sec = (ereg("clr\[[12]\]", sline));
 mslexec_log_condition("x105_0 - является ли инструкцией
    интерпретатора clr", $res);
 return $res;
}
function mslexec_x105_1()
  global $line;
  sec{$res = (ereg("cpy\[[12]\]\[[12]\], $line));}
  mslexec_log_condition("x105_1 - является ли инструкцией
    интерпретатора сру", $res);
 return $res;
}
function mslexec_x105_2()
{
```

```
global $line;
  sec = (ereg("mov\[[12]\]\[[012]\]\[[0-9]*\]", $line));
  mslexec_log_condition("x105_2 - является ли инструкцией
    интерпретатора mov", $res);
 return $res;
}
function mslexec_x105_3()
 global $line;
  sec = (ereg("put\[[12]\]\[[-\/\a-sA-Sa-zA-Z0-9,!?:;_
    ]*\]", $line));
  mslexec_log_condition("x105_3 - является ли инструкцией
    интерпретатора put", $res);
 return $res;
}
function mslexec_x105_4()
  global $line;
  $res = (ereg("^nop", $line));
 mslexec_log_condition("x105_4 - является ли инструкцией
    интерпретатора nop", $res);
 return $res;
}
function mslexec_x105_5()
{
  global $line;
  sec = (ereg("skp\[[12]\]", sline));
 mslexec_log_condition("x105_5 - является ли инструкцией
    интерпретатора skp", $res);
  return $res;
}
function mslexec_x105_6()
  global $line;
  sec{sec} = (ereg("bck\[[12]\]", $line));
 mslexec_log_condition("x105_6 - является ли инструкцией
    интерпретатора bck", $res);
  return $res;
```

```
}
function mslexec_x105_7()
  global $line;
  sec = (ereg("jsn\[[12]\]", sline));
  mslexec_log_condition("x105_7 - является ли инструкцией
     интерпретатора jsn", $res);
  return $res;
}
function mslexec_x105_8()
  global $line;
  sec = (ereg("run \setminus [[12] \setminus ] \setminus [[012] \setminus ] \setminus [[a-zA-Z0-9._]])
     :\/\\]*\]", $line));
  mslexec_log_condition("x105_8 - является ли инструкцией
     интерпретатора run", $res);
  return $res;
}
function mslexec_x105_9()
  global $line;
  sec = (ereg("off \setminus [[0-9]* \setminus ] \setminus [[a-zA-Z0-9_: \setminus ] * \setminus ]", sline));
  mslexec_log_condition("x105_9 - является ли инструкцией
     интерпретатора off", $res);
  return $res;
}
function mslexec_x105_A()
{
  global $line;
  sec = (ereg("get\[[12]\]\[[a-zA-Z0-9_:\]*\]", $line));
  mslexec_log_condition("x105_A - является ли инструкцией get
    ", $res);
  return $res;
}
function mslexec_x105_B()
{
  global $line;
```

```
sec = (ereg("prc\[[a-zA-Z0-9_:\]*]", $line));
  mslexec_log_condition("x105_B - является ли инструкцией
     интерпретатора prc", $res);
  return $res;
}
function mslexec_x105_C()
  global $line;
  sec = (ereg("lbl\[[a-zA-Z0-9_]*\]", sline));
  mslexec_log_condition("x105_C - является ли инструкцией
     интерпретатора lbl", $res);
  return $res;
}
function mslexec_x105_D()
  global $line;
  sec{1}{res} = (ereg("jmp \setminus [[a-zA-Z0-9_]* \setminus ]", $line));
  mslexec_log_condition("x105_D - является ли инструкцией
     интерпретатора jmp", $res);
  return $res;
}
function mslexec_x105_E()
  global $line;
  sec = (ereg("jss \setminus [[12]* \setminus ]", sline));
  mslexec_log_condition("x105_E - является ли инструкцией
     интерпретатора jss", $res);
  return $res;
}
function mslexec_x105_F()
  global $line;
  sec = (ereg("cll\[[a-zA-Z0-9_]*\]", $line));
  mslexec_log_condition("x105_F - является ли инструкцией
     интерпретатора cll", $res);
  return $res;
}
```

```
function mslexec_x105_G()
{
 global $line;
 $res = (ereg("^end", $line));
 mslexec_log_condition("x105_G - является ли инструкцией
    интерпретатора end", $res);
 return $res;
}
ФУНКЦИИ, РЕАЛИЗУЮЩИЕ ВЫХОДНЫЕ ВОЗДЕЙСТВИЯ
function mslexec_z101()
 mslexec_log_action("z101 - инициализируем переменные");
 global $dir_line;
 global $lbl_line;
 $dir_line = "$";
 $1b1_line = "$";
}
function mslexec_z102()
 global $line_count;
 mslexec_log_action("z102 - читаем следующую строчку программы
    , ".($line_count+1)."");
 global $msl_code_file;
 global $line;
 $line_count++;
 $line = fgets( $msl_code_file );
}
function mslexec z103()
 mslexec_log_action("z103 - записываем имя");
 global $line;
 global $msl_binary_file;
 $pr_name = mysql_real_escape_string(substr($line, 4, strpos(
    $line, "]", 4)-4));
 fwrite($msl_binary_file, $pr_name."\n");
```

```
}
function mslexec_z104()
 mslexec_log_action("z104 - записываем директиву");
  global $dir_line;
  global $line;
  $dir_line .= substr($line, 1, 3)."$".substr($line, 5, strpos(
    $line, "]", 5)-5)."$";
}
function mslexec_z105_0()
 global $curr;
  global $curr_num;
  global $line;
 mslexec_log_action("z105_0 - компилируем инструкцию clr");
  $num_data = pack("n", $curr_num);
  $bin_data = "0".pack("nc", $curr, substr($line, 4,1));
  $curr_num+=4;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_1()
 global $curr;
  global $curr_num;
  global $line;
 mslexec_log_action("z105_1 - компилируем инструкцию сру");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 10, strpos($line, "]", 10)-10);
  $bin_data = "1".pack("nccn", $curr, substr($line, 4,1),
    substr($line, 7,1), $tmp_data);
  curr_num += 7;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
```

```
$bin_line.=$bin_data;
}
function mslexec_z105_2()
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_2 - компилируем инструкцию mov");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 10, strpos($line, "]", 10)-10);
  $bin_data = "2".pack("nccn", $curr, substr($line, 4,1),
    substr($line, 7,1), $tmp_data);
  curr_num += 7;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_3()
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_3 - компилируем инструкцию put");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 7, strpos($line, "]", 7)-7);
  $bin_data = "3".pack("ncn", $curr, substr($line, 4,1),
    strlen($tmp_data)).$tmp_data;
  $curr_num+=6+strlen($tmp_data);
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_4()
{
```

```
global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_4 - компилируем инструкцию nop");
  $num_data = pack("n", $curr_num);
  $bin_data = "4".pack("n", $curr);
  $curr_num+=3;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_5()
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_5 - компилируем инструкцию skp");
  $num_data = pack("n", $curr_num);
  $bin_data = "5".pack("nc", $curr, substr($line, 4,1));
  curr_num += 4;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_6()
{
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_6 - компилируем инструкцию bck");
  $num_data = pack("n", $curr_num);
  $bin_data = "6".pack("nc", $curr, substr($line, 4,1));
  $curr_num+=4;
  global $num_line;
  global $bin_line;
```

```
$num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_7()
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_7 - компилируем инструкцию jsn");
  $num_data = pack("n", $curr_num);
  $bin_data = "7".pack("nc", $curr, substr($line, 4,1));
  $curr_num+=4;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_8()
{
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_8 - компилируем инструкцию run");
  $num_data = pack("n", $curr_num);
  $tmp_data_01 = substr($line, 7, strpos($line, "]", 7)-7);
  tmp_pos_01 = 6 + strlen(tmp_data_01) + 3;
  $tmp_data_02 = substr($line, $tmp_pos_01, strpos($line, "]",
    $tmp_pos_01)-$tmp_pos_01);
  tmp_pos_02 = 6 + strlen(tmp_data_01) + 3 - 1 + strlen(tmp_data_02)
    +3:
  $tmp_data_03 = substr($line, $tmp_pos_02, strpos($line, "]",
     $tmp_pos_02)-$tmp_pos_02);
  $bin_data = "8".pack("ncccn", $curr,
                                 substr($line, 4,1),
                                 $tmp_data_01,
                                 $tmp_data_02,
                                 strlen($tmp_data_03))
                                 .$tmp_data_03;
```

```
$curr_num+=8+strlen($tmp_data_03);
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_9()
{
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_9 - компилируем инструкцию off");
  $num_data = pack("n", $curr_num);
  $tmp_data_01 = substr($line, 4, strpos($line, "]", 4)-4);
  tmp_pos_01 = 3+strlen(tmp_data_01)+3;
  $tmp_data_02 = substr($line, $tmp_pos_01, strpos($line, "]",
    $tmp_pos_01)-$tmp_pos_01);
  tmp_pos_02 = 3+strlen(tmp_data_01)+3-1+strlen(tmp_data_02)
    +3:
  $bin_data = "9".pack("ncn", $curr, $tmp_data_01, strlen(
     $tmp_data_02)).$tmp_data_02;
  $curr_num+=6+strlen($tmp_data_02);
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_A()
{
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_A - компилируем инструкцию get");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 7, strpos($line, "]", 7)-7);
  $bin_data = "A".pack("ncn", $curr, substr($line, 4,1), strlen
    ($tmp_data)).$tmp_data;
  $curr_num+=6+strlen($tmp_data);
```

```
global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_B()
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_B - компилируем инструкцию prc");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 4, strpos($line, "]", 4)-4);
  $bin_data = "B".pack("nn", $curr, strlen($tmp_data)).
    $tmp_data;
  $curr_num+=5+strlen($tmp_data);
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
function mslexec_z105_C()
  global $curr;
  global $curr_num;
  global $lbl_line;
  global $line;
  mslexec_log_action("z105_C - компилируем инструкцию lbl");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 4, strpos($line, "]", 4)-4);
  $lbl_line .= $tmp_data."$".$curr."$";
  $bin_data = "C".pack("n", $curr);
  $curr_num+=3;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
```

```
function mslexec_z105_D()
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_D - компилируем инструкцию jmp");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 4, strpos($line, "]", 4)-4);
  $bin_data = "D".pack("nn", $curr, strlen($tmp_data)).
    $tmp_data;
  $curr_num+=5+strlen($tmp_data);
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_E()
{
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_E - компилируем инструкцию jss");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 4, strpos($line, "]", 4)-4);
  $bin_data = "E".pack("nc", $curr, $tmp_data);
  $curr_num+=4;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_F()
{
  global $curr;
  global $curr_num;
  global $line;
```

```
mslexec_log_action("z105_F - компилируем инструкцию cll");
  $num_data = pack("n", $curr_num);
  $tmp_data = substr($line, 4, strpos($line, "]", 4)-4);
  $bin_data = "F".pack("nn", $curr, strlen($tmp_data)).
    $tmp_data;
  $curr_num+=5+strlen($tmp_data);
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z105_G()
  global $curr;
  global $curr_num;
  global $line;
  mslexec_log_action("z105_G - компилируем инструкцию end");
  $num_data = pack("n", $curr_num);
  $bin_data = "G".pack("n", $curr);
  $curr_num+=3;
  global $num_line;
  global $bin_line;
  $num_line.=$num_data;
  $bin_line.=$bin_data;
}
function mslexec_z106()
  mslexec_log_action("z106 - увеличиваем номер текущей
    инструкции");
  global $curr;
  $curr++;
}
function mslexec_z107()
  mslexec_log_action("z107 - завершаем запись");
  global $curr;
  global $curr_num;
  global $num_line;
```

```
global $bin_line;
  global $lbl_line;
  global $msl_binary_file;
  $num_data = pack("n", $curr_num);
  $num_line .= $num_data;
  $num_data = pack("n", $curr);
  $num_line .= $num_data;
  $num_line = mysql_real_escape_string($num_line);
  $bin_line = mysql_real_escape_string($bin_line);
  fwrite($msl_binary_file, $lbl_line."\n".$num_line."\n".
    $bin_line."\n");
  echo "Normal compilation"; // сообщение в браузер
                             // об успешной компиляции
}
function mslexec_z108()
 mslexec_log_action("z108 - пишем строку директив");
  global $msl_binary_file;
 global $dir_line;
 fwrite($msl_binary_file, $dir_line."\n");
?>
```

Приложение 2. Протокол компиляции примера Simple

```
Запущен автомат: А1 с программой simple.msl

Состояние 0: Начальное состояние

z101 - инициализируем переменные

z102 - читаем следующую строчку программы, 1

Состояние 1: Чтение имени

x101 - является ли именем - Да

z103 - записываем имя

z102 - читаем следующую строчку программы, 2

Состояние 2: Чтение директив

x102 - является ли директивой - Нет

x105_А - является ли инструкцией get - Нет

x104 - является ли инструкцией интерпретатора - Да

z108 - пишем строку директив
```

```
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 3
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 4
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 5
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
 х105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 6
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Нет
```

```
x105_4 - является ли инструкцией интерпретатора nop - Her
    х105_5 - является ли инструкцией интерпретатора skp - Нет
    x105_6 - является ли инструкцией интерпретатора bck - Нет
    x105_7 - является ли инструкцией интерпретатора jsn - Her
    x105_8 - является ли инструкцией интерпретатора run - Нет
    x105_9 - является ли инструкцией интерпретатора off - Her
    х105_В - является ли инструкцией интерпретатора prc - Нет
    x105_C - является ли инструкцией интерпретатора lbl - Нет
    х105_D - является ли инструкцией интерпретатора jmp - Нет
    x105_E - является ли инструкцией интерпретатора jss - Нет
    x105_F - является ли инструкцией интерпретатора cll - Нет
    x105_G - является ли инструкцией интерпретатора end - Да
    z105_G - компилируем инструкцию end
    z106 - увеличиваем номер текущей инструкции
  Состояние 5: Нормальное завершение.
    z107 - завершаем запись
Завершен автомат: А1
```

Приложение 3. Протокол компиляции примера Cycle

```
Запущен автомат: A1 с программой cycle.msl
  Состояние 0: Начальное состояние
    z101 - инициализируем переменные
    z102 - читаем следующую строчку программы, 1
  Состояние 1: Чтение имени
    х101 - является ли именем - Да
    z103 - записываем имя
    z102 - читаем следующую строчку программы, 2
  Состояние 2: Чтение директив
    х102 - является ли директивой - Нет
    х105_А - является ли инструкцией get - Нет
    х104 - является ли инструкцией интерпретатора - Да
    z108 - пишем строку директив
  Состояние 4: Чтение инструкций интерпретатора
    x105_0 - является ли инструкцией интерпретатора clr - Нет
    х105_1 - является ли инструкцией интерпретатора сру - Нет
    x105_2 - является ли инструкцией интерпретатора mov - Нет
    x105_3 - является ли инструкцией интерпретатора put - Да
    z105_3 - компилируем инструкцию put
    z106 - увеличиваем номер текущей инструкции
    z102 - читаем следующую строчку программы, 3
```

```
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
 х105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
 x105_8 - является ли инструкцией интерпретатора run - Нет
 x105_9 - является ли инструкцией интерпретатора off - Heт
 х105_В - является ли инструкцией интерпретатора prc - Нет
 x105_C - является ли инструкцией интерпретатора lbl - Да
  z105_C - компилируем инструкцию lbl
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 4
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 5
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
 x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 6
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Нет
```

```
x105_4 - является ли инструкцией интерпретатора nop - Her
 х105_5 - является ли инструкцией интерпретатора skp - Нет
  x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Нет
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 7
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Да
  z105_1 - компилируем инструкцию сру
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 8
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Да
  z105_5 - компилируем инструкцию skp
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 9
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
 x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
 x105_8 - является ли инструкцией интерпретатора run - Нет
  x105_9 - является ли инструкцией интерпретатора off - Her
 х105_В - является ли инструкцией интерпретатора prc - Нет
  x105_C - является ли инструкцией интерпретатора lbl - Her
 х105_D - является ли инструкцией интерпретатора jmp - Да
 z105_D - компилируем инструкцию jmp
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 10
Состояние 4: Чтение инструкций интерпретатора
```

```
x105_0 - является ли инструкцией интерпретатора clr - Нет
   х105_1 - является ли инструкцией интерпретатора сру - Нет
    x105_2 - является ли инструкцией интерпретатора mov - Her
   x105_3 - является ли инструкцией интерпретатора put - Да
    z105_3 - компилируем инструкцию put
    z106 - увеличиваем номер текущей инструкции
    z102 - читаем следующую строчку программы, 11
 Состояние 4: Чтение инструкций интерпретатора
    x105_0 - является ли инструкцией интерпретатора clr - Нет
   х105_1 - является ли инструкцией интерпретатора сру - Нет
   x105_2 - является ли инструкцией интерпретатора mov - Heт
   x105_3 - является ли инструкцией интерпретатора put - Нет
   х105_4 - является ли инструкцией интерпретатора пор - Нет
   х105_5 - является ли инструкцией интерпретатора skp - Нет
   х105_6 - является ли инструкцией интерпретатора bck - Нет
    x105_7 - является ли инструкцией интерпретатора jsn - Her
   x105_8 - является ли инструкцией интерпретатора run - Да
    z105_8 - компилируем инструкцию run
    z106 - увеличиваем номер текущей инструкции
    z102 - читаем следующую строчку программы, 12
 Состояние 4: Чтение инструкций интерпретатора
    x105_0 - является ли инструкцией интерпретатора clr - Нет
   х105_1 - является ли инструкцией интерпретатора сру - Нет
   x105_2 - является ли инструкцией интерпретатора mov - Нет
   x105_3 - является ли инструкцией интерпретатора put - Нет
   х105_4 - является ли инструкцией интерпретатора пор - Нет
   х105_5 - является ли инструкцией интерпретатора skp - Нет
   х105_6 - является ли инструкцией интерпретатора bck - Нет
   x105_7 - является ли инструкцией интерпретатора jsn - Her
   x105_8 - является ли инструкцией интерпретатора run - Нет
   x105_9 - является ли инструкцией интерпретатора off - Her
    x105_В - является ли инструкцией интерпретатора prc - Нет
   x105_C - является ли инструкцией интерпретатора lbl - Her
   х105_D - является ли инструкцией интерпретатора jmp - Нет
    x105_E - является ли инструкцией интерпретатора jss - Her
   x105_F - является ли инструкцией интерпретатора cll - Нет
    x105_G - является ли инструкцией интерпретатора end - Да
    z105_G - компилируем инструкцию end
    z106 - увеличиваем номер текущей инструкции
 Состояние 5: Нормальное завершение.
    z107 - завершаем запись
Завершен автомат: А1
```

Приложение 4. Автомат "Исполнитель" (А2)

Φ айл executor.php

```
<?php
/**
  UNIT.... executor.php
  PROJECT...: SIL
  AUTHOR....: М. Коротков, А. Лукьянова
  CREATED . . . . : 14.06.2003
  PURPOSE . . . . : Исполнитель программ
  FUNCTIONS . . :
  PARAMS . . . . :
  NOTES . . . . :
  HISTORY . . . : DATE
                           NOTES
               12.06.2003 Создан
               22.11.2003 Перемещен в проект SIL,
                           внесены изменения
*/
/* Includes */
if (!function_exists("paths_escape"))
 require_once("paths.php");
require_once(CGI_PATH.SN_DB); // работа с базой данных
require_once(CGI_PATH.SN_CONSTS); // константы
require_once(CGI_PATH.SN_MSLUTILS); // утилиты
require_once(CGI_PATH.SN_AUTOMATE_FUNCTIONS);
                                     // функции автомата
$logs = true;
/* Globals */
global $program_id;
global $ret_id;
if ($logs) {
  global $log_file;
  global $log_file_txt;
  global $spaces;
```

```
if (!isset($log_file))
 {
     mslexec_log_open(CGI_PATH."logs/log_".$program_id);
 }
}
st = 0;
$current_i = 0;
global $result;
global $program_name;
global $bin_line;
global $num_line;
global $i_num;
global $wiz_id;
global $data;
global $load;
global $stack;
global $output_type;
global $max_i;
mslexec_log_start("A2");
АВТОМАТ ИСПОЛНИТЕЛЬ
while ($st <> 6) {
 switch ($st) {
   case 0:
     mslexec_log_state($st, "Начальное состояние");
     executor_z200();
     if (executor_x204()) {
      st = 1;
     } else {
      mslexec_log_error("Программа отсутствует в базе");
      st = 4;
     }
   break;
   case 1:
    mslexec_log_state($st, "Загрузка программы");
     executor_z206();
     if (executor_x200()) {
```

```
st = 2;
      }
      else {
       executor_z202();
       st = 2;
    break;
    case 2:
      mslexec_log_state($st, "Обработка директив загрузчика get
      if (executor_x203()) {
       executor_z203();
        st = 2;
      } else {
        st = 3;
    break;
    case 3:
     mslexec_log_state($st, "Запуск процессора языка");
     executor_z205();
     require(CGI_PATH.SN_LOADER);
      st = 5;
    break;
    case 4:
     mslexec_log_state($st, "Аварийное завершение");
      st = 6;
    break;
    case 5:
     mslexec_log_state($st, "Нормальное завершение");
      st = 6;
    break;
 }
}
mslexec_log_finish("A2");
?>
```

Приложение 5. Функции автоматов

Скрипты excutor.php и loader.php подключают свои функции из фала automate functions.php.

```
<?php
ФУНКЦИИ ИСПОЛНИТЕЛЯ
function executor_x200()
 global $ret_id;
 $res = ((!isset($ret_id)));
 mslexec_log_condition("x200 - проверка отсутствия программы
    для возврата", $res);
 return $res;
}
function executor_x204()
 global $result;
 global $program_id;
 $res = ((mysql_num_rows($result)<>0)&&(isset($program_id)));
 mslexec_log_condition("x204 - проверка наличия программы в
   базе", $res);
 return $res;
}
function executor_x203()
 global $current_i;
 global $data;
 $data = mslutils_getnext($current_i);
 $res = ($data["instr"] == "A");
 mslexec_log_condition("x203 - является ли инструкцией
    загрузчика get", $res);
 return $res;
}
function executor_z200()
{
```

```
global $program_id;
  global $result;
  mslexec_log_action("z200 - подключение к базе");
  db_connect();
  db_query("max_remove_wizard_sessions_by_time", TTL);
  $result = db_query("max_get_program", $program_id);
}
function executor_z206()
  global $program_id;
  global $result;
  global $program_name;
  global $bin_line;
  global $num_line;
  global $i_num;
  global $wiz_id;
  mslexec_log_action("z206 - загрузка данных из базы");
  $program_name = $program_id;
  $arr = mysql_fetch_array($result, MYSQL_BOTH);
  $bin_line = $arr["program"];
  $num_line = $arr["program_num"];
  $num_line = mslutils_unreal_string($num_line);
  $bin_line = mslutils_unreal_string($bin_line);
  \sin_n = -2 + (strlen(\sin_n \sin_n )/2);
  mt srand():
  do
  {
    $wiz_id = sprintf("%08x", rand(1, 0xFFFFFFF));
  } while (mysql_num_rows(db_query("max_get_wizard_session",
    $wiz_id))!=0);
  db_query("max_create_wizard_session", $program_name, $i_num,
    $wiz_id);
  db_query("max_set_current",0,$wiz_id);
function executor_z202()
  global $wiz_id;
  global $ret_id;
```

```
mslexec_log_action("z202 - копирование стеков");
  $result = db_query("max_get_wizard_session", $ret_id);
  $arr = mysql_fetch_array($result, MYSQL_ASSOC);
  db_query("max_set_wizard_session", $arr["stack1"], $arr["
    stack2"], 0, $wiz_id);
  db_query("max_set_wizard_ret_id", $ret_id, $wiz_id) or die("
    Ret id setting failed");
}
function executor_z203()
  global $wiz_id;
  global $data;
  global $current_i;
  mslexec_log_action("z203 - обработка инструкции загрузчика
    get");
  $perem = $HTTP_POST_VARS[$data["text"]];
  $perem_num = $data["stack"];
  switch ($perem_num)
  {
  case "1":
    db_query("max_concat_to_stack1", $perem."^", $wiz_id)
            ."<br>":
    break;
  case "2":
    db_query("max_concat_to_stack2", $perem."^", $wiz_id)
            ."<br>";
    break:
  default:
    break:
  }
  $current_i++;
}
function executor_z205()
{
  global $load;
  global $program_name;
  global $stack;
  global $output_type;
```

```
global $current_i;
 global $max_i;
 global $wiz_id;
 global $ret_id;
 mslexec_log_action("z203
        - подготовка к запуску автомата АЗ");
 db_query("max_set_current", $current_i, $wiz_id);
 10ad = 1;
 $result = db_query("max_get_wizard_session", $wiz_id);
 $arr = mysql_fetch_array($result, MYSQL_BOTH);
 $program_name = $arr["program_name"];
 $current_i = $arr["cur_instruction"];
 $max_i = $arr["max_instruction"];
 $stack["1"] = $arr["stack1"];
 $stack["2"] = $arr["stack2"];
 $output_type = $arr["output_type"];
 $ret_id = sprintf("%08x", $arr["ret_id"]);
}
ФУНКЦИИ ПРОЦЕССОРА
function loader_x301_0()
 global $data;
 $res = ($data["instr"] == "0");
 mslexec_log_condition("x301_0 - является ли инструкция
    инструкцией clr", $res);
 return $res;
function loader_x301_1()
 global $data;
 $res = ($data["instr"] == "1");
 mslexec_log_condition("x301_1 - является ли инструкция
    инструкцией сру", $res);
 return $res;
```

```
function loader_x301_2()
 global $data;
  $res = ($data["instr"] == "2");
 mslexec_log_condition("x301_2 - является ли инструкция
    инструкцией mov", $res);
 return $res;
}
function loader_x301_3()
 global $data;
 $res = ($data["instr"] == "3");
 mslexec_log_condition("x301_3 - является ли инструкция
    инструкцией put", $res);
 return $res;
}
function loader_x301_4()
{
  global $data;
  $res = ($data["instr"] == "4");
 mslexec_log_condition("x301_4 - является ли инструкция
    инструкцией nop", $res);
 return $res;
}
function loader_x301_5()
  global $data;
 $res = ($data["instr"] == "5");
 mslexec_log_condition("x301_5 - является ли инструкция
    инструкцией skp", $res);
 return $res;
}
function loader_x301_6()
 global $data;
  $res = ($data["instr"] == "6");
```

```
mslexec_log_condition("x301_6 - является ли инструкция
    инструкцией bck", $res);
  return $res;
function loader_x301_7()
  global $data;
  $res = ($data["instr"] == "7");
  mslexec_log_condition("x301_7 - является ли инструкция
    инструкцией jsn", $res);
  return $res;
}
function loader_x301_8()
  global $data;
  $res = ($data["instr"] == "8");
  mslexec_log_condition("x301_8 - является ли инструкция
    инструкцией run", $res);
 return $res;
}
function loader_x301_9()
  global $data;
  $res = ($data["instr"] == "9");
  mslexec_log_condition("x301_9 - является ли инструкция
    инструкцией off", $res);
 return $res;
}
function loader_x301_B()
{
  global $data;
  $res = ($data["instr"] == "B");
  mslexec_log_condition("x301_В - является ли инструкция
    инструкцией prc", $res);
  return $res;
}
function loader_x301_C()
```

```
{
  global $data;
  $res = ($data["instr"] == "C");
  mslexec_log_condition("x301_C - является ли инструкция
    инструкцией lbl", $res);
  return $res;
}
function loader_x301_D()
  global $data;
  $res = ($data["instr"] == "D");
  mslexec_log_condition("x301_D - является ли инструкция
    инструкцией jmp", $res);
 return $res;
}
function loader_x301_E()
  global $data;
  $res = ($data["instr"] == "E");
  mslexec_log_condition("x301_E - является ли инструкция
    инструкцией jss", $res);
 return $res;
}
function loader_x301_F()
{
  global $data;
  $res = ($data["instr"] == "F");
  mslexec_log_condition("x301_F - является ли инструкция
    инструкцией cll", $res);
  return $res;
}
function loader_x301_G()
{
  global $data;
  $res = ($data["instr"] == "G");
  mslexec_log_condition("x301_G - является ли инструкция
    инструкцией end", $res);
  return $res;
```

```
}
function loader_x302()
  global $ret_id;
  $res = ($ret_id <> "00000000");
  mslexec_log_condition("x302 - требуется ли возврат во внешний
      автомат A2", $res);
  return $res;
}
function loader_z300()
  global $data;
  global $current_i;
  mslexec_log_action("z300 - взятие следующей инструкции, ".(
    $current_i+1)."");
  $data = mslutils_getnext($current_i);
  $current_i++;
}
function loader_z301_0()
  global $stack;
  global $data;
  mslexec_log_action("z301_0 - выполнение инструкции clr");
  $stack[$data["stack"]] = "";
}
function loader_z301_1()
{
  global $stack;
  global $data;
  mslexec_log_action("z301_1 - выполнение инструкции сру");
  $temp_str = "";
  now_char = 0;
  for ($i = 0; $i < $data["number"]; $i++)
    $next_char = strpos($stack[$data["stack1"]],"^",$now_char);
    $temp_str .= substr($stack[$data["stack1"]], $now_char,
       $next_char - $now_char + 1);
```

```
$now_char = $next_char+1;
  }
  $stack[$data["stack2"]] = $temp_str.$stack[$data["stack2"]];
}
function loader_z301_2()
  global $stack;
  global $data;
  mslexec_log_action("z301_2 - выполнение инструкции mov");
  $temp_str = "";
  now_char = 0;
  for ($i = 0; $i < $data["number"]; $i++)
    $next_char = strpos($stack[$data["stack1"]],"^",$now_char);
    $temp_str .= substr($stack[$data["stack1"]], $now_char,
       $next_char - $now_char + 1);
    $now_char = $next_char+1;
  }
  $stack[$data["stack1"]] = substr($stack[$data["stack1"]],
    $now_char);
  if ($data["stack2"]!="0")
    $stack[$data["stack2"]] = $temp_str.$stack[$data["stack2
       "]];
}
function loader_z301_3()
  global $stack;
  global $data;
  mslexec_log_action("z301_3 - выполнение инструкции put");
  $stack[$data["stack"]] = $data["text"]."^".$stack[$data["
    stack"]];
}
function loader_z301_5()
{
```

```
global $stack;
  global $data;
  global $current_i;
  mslexec_log_action("z301_5 - выполнение инструкции skp");
  $next_char = strpos($stack[$data["stack"]],"^",0);
  $temp_str = substr($stack[$data["stack"]], 0, $next_char);
  $stack[$data["stack"]] = substr($stack[$data["stack"]],
    $next char+1):
  $current_i+=$temp_str;
}
function loader_z301_6()
  global $stack;
  global $data;
  global $current_i;
  mslexec_log_action("z301_6 - выполнение инструкции bck");
  $next_char = strpos($stack[$data["stack"]],"^",0);
  $temp_str = substr($stack[$data["stack"]], 0, $next_char);
  $stack[$data["stack"]] = substr($stack[$data["stack"]],
    $next_char+1);
  $current_i -=1+$temp_str;
function loader_z301_7()
  global $stack;
  global $data;
  global $current_i;
  mslexec_log_action("z301_7 - выполнение инструкции jsn");
  $next_char = strpos($stack[$data["stack"]],"^",0);
  $temp_str = substr($stack[$data["stack"]], 0, $next_char);
  $stack[$data["stack"]] = substr($stack[$data["stack"]],
    $next_char+1);
  $current_i = -1 + $temp_str;
function loader_z301_8()
  global $stack;
  global $data;
  global $inp;
```

```
global $output_type;
  mslexec_log_action("z301_8 - выполнение инструкции run");
  srun = 1:
  $stackn = $data["stack"];
  $paramn = $data["params"];
  sinp[0] = 0;
  $output_type = $data["type"];
 for ($i = 0; $i < paramn; $i++)
  {
    $inp[$i] = mslutils_stack_get($stack[$stackn]);
  }
 require(mslutils_format_constant($data["text"], CGI_PATH));
function loader_z301_9()
  global $stack;
  global $data;
  global $current_i;
 mslexec_log_action("z301_9 - выполнение инструкции off");
  now_char1 = 0;
  $temp_str_1 = "";
  now_char2 = 0;
  $current_i++;
  for ($i = 0; $i < $data["params"]; $i++)</pre>
    $next_char1 = strpos($stack["1"],"^",$now_char);
    $next_char2 = strpos($stack["2"],"^",$now_char);
    $temp_str_1 .= "&".substr($stack["1"], $now_char,
       $next_char - $now_char)."=".substr($stack["2"], $now_char,
       $next_char - $now_char);
    $now_char1 = $next_char1+1;
    $now_char2 = $next_char2+1;
  }
  $stack["1"] = substr($stack["1"], $now_char1);
  $stack["2"] = substr($stack["2"], $now_char2);
```

```
mslutils_put_current($current_i, $wiz_id);
  $wiz_id = mslutils_put_other($stack, 0, $wiz_id);
  header("Location: ".CGI_HOST.$data["text"]."?wiz_id=".$wiz_id
     .$temp_str_1);
 exit;
}
function loader_z301_B()
  global $data;
 mslexec_log_action("z301_B - выполнение инструкции prc");
 require(mslutils_format_constant($data["text"], CGI_PATH));
}
function loader_z301_D()
  global $data;
  global $current_i;
 mslexec_log_action("z301_D - выполнение инструкции jmp");
  $current_i = mslutils_get_label($data["text"]);
}
function loader_z301_E()
  global $stack;
  global $data;
  global $current_i;
  mslexec_log_action("z301_E - выполнение инструкции jss");
  $next_char = strpos($stack[$data["stack"]],"^",0);
  $temp_str = substr($stack[$data["stack"]], 0, $next_char);
  $stack[$data["stack"]] = substr($stack[$data["stack"]],
    $next_char+1);
  $current_i = mslutils_get_label($temp_str);
}
function loader_z301_F()
{
  global $stack;
  global $data;
  global $current_i;
  global $wiz_id;
```

```
global $ret_id;
  global $program_id;
  mslexec_log_action("z301_F - выполнение инструкции cll");
  mslutils_put_current($current_i, $wiz_id);
  $wiz_id = mslutils_put_other($stack, 0, $wiz_id);
  $program_id = $data["text"];
  $ret_id = $wiz_id;
}
function loader_z301_G()
  global $stack;
  global $data;
  global $wiz_id;
  global $ret_id;
  global $program_id;
  global $program_name;
  global $current_i;
  global $max_i;
  global $stack;
  global $output_type;
  global $bin_line;
  global $num_line;
  mslexec_log_action("z301_G - выполнение инструкции end");
  $wiz_id = $ret_id;
  db_query("max_set_wizard_session", $stack["1"], $stack
    ["2"], 0, $wiz_id);
  $data = mslutils_wizard_get($wiz_id);
  $program_name = $data["name"];
  $program_id = $data["name"];
  $current_i = $data["current_i"];
  $max_i = $data["max_i"];
  $stack = $data["stack"];
  $output_type = $data["output_type"];
  $ret_id = sprintf("%08x", $data["ret_id"]);
  $bin_line = $data["bin_line"];
  $num_line = $data["num_line"];
}
?>
```

Приложение 6. Автомат "Процессор" (А3)

Файл loader.php

```
<?php
/**
  UNIT ....: loader.php
  PROJECT . . . : SIL
  AUTHOR....: М. Коротков, А. Лукьянова
  CREATED . . . . : 13.06.2003
  PURPOSE....: Процессор SIL-программ
  FUNCTIONS . . :
  PARAMS....: все перечислены ниже как глобальные
  NOTES . . . . . :
  HISTORY . . . : DATE
                         NOTES
              12.06.2003 Создан
              22.11.2003 Перемещен
                          в проект SIL
*/
/* Includes */
if (!function_exists("paths_escape")) require_once("paths.php")
require_once(CGI_PATH.SN_DB); // работа с базой данных
require_once(CGI_PATH.SN_MSLUTILS); // утилиты
/* Globals */
st = 0;
                  // ключ образа данных программы
global $wiz_id;
                   // байт-код
global $bin_line;
global $num_line;
                    // номера инструкций
                    // номер программы для возврата
global $ret_id;
global $stack; // cTek
global $program_name; // имя программы
global $program_id; // уникальный идентификатор
global $current_i; // текущая инструкция
global $data;
                    // образ данных в памяти
global $max_i; // последняя инструкция
global $output_type; // номер выходного стека
```

```
global $load;
global $logs;
global $spaces;
global $log_file;
mslexec_log_start("A3");
/*
               АВТОМАТ ПРОЦЕССОР
                                                    */
while ($st <> 6) {
 switch ($st) {
   case 0:
     mslexec_log_state($st, "Начальное состояние");
     st = 1;
   break;
   case 1:
     mslexec_log_state($st, "Обработка инструкций");
     loader_z300();
     if (loader_x301_0()) {
      loader_z301_0();
      st = 1:
     } elseif (loader_x301_1()) {
      loader_z301_1();
      st = 1;
     } elseif (loader_x301_2()) {
      loader_z301_2();
      st = 1:
     } elseif (loader_x301_3()) {
      loader_z301_3();
      st = 1;
     } elseif (loader_x301_4()) {
      st = 1:
     } elseif (loader_x301_5()) {
      loader_z301_5();
      st = 1;
     } elseif (loader_x301_6()) {
      loader_z301_6();
      st = 1:
     } elseif (loader_x301_7()) {
       loader_z301_7();
```

```
st = 1;
 } elseif (loader_x301_8()) {
   loader_z301_8();
    st = 1;
 } elseif (loader_x301_9()) {
    loader_z301_9();
   st = 1;
 } elseif (loader_x301_B()) {
   loader_z301_B();
   st = 1;
 } elseif (loader_x301_C()) {
    st = 1;
 } elseif (loader_x301_D()) {
   loader_z301_D();
   st = 1;
 } elseif (loader_x301_E()) {
   loader_z301_E();
   st = 1;
 } elseif (loader_x301_F()) {
    st = 2;
 } elseif (loader_x301_G()) {
   st = 3;
 } else {
   mslexec_log_error("Heизвестная инструкция");
    st = 5;
 }
break;
case 2:
 mslexec_log_state($st, "Вызов нового загрузчика");
 loader_z301_F();
 require(CGI_PATH.SN_EXECUTOR);
  st = 1;
break;
case 3:
 mslexec_log_state($st, "Подготовка к завершению работы
    автомата");
 if (loader_x302()) {
   loader_z301_G();
   st = 4;
 } else {
```

```
$st = 4;
}
break;

case 4:
    mslexec_log_state($st, "Нормальное завершение");
    $st = 6;
break;

case 5:
    mslexec_log_state($st, "Аварийное завершение");
    $st = 6;
break;
}

mslexec_log_finish("A3");
?>
```

Приложение 7. Сценарий выбора пользователей

```
nam[user_master]
$dsc[msg-seluser]
$acc[RAT]
$blk[]
$upd[]
put [1] [SN_SITEADM_USERS]
run [1] [1] [1] [SN_RUNNER_USER_MASTER]
skp[1]
run [2] [0] [2] [SN_COURSE_SELECTOR]
put [2] [3]
put [2] [2]
put [2] [SN_PUTTER_LISTER]
put [2] [SN_SITEADM_USERS]
run[2][5][2][SN_RUNNER_LISTER]
mov[2][1][1]
mov[1][2][2]
mov[2][1][1]
mov[1][2][2]
put [1] [-1]
```

```
mov[2][1][1]
put [1] [-1]
mov [2] [1] [1]
mov[2][1][1]
put [1] [39]
skp[1]
put [1] [-1]
put [1] [SN_SITEADM_USERS]
run [1] [1] [SN_RUNNER_GROUP_MASTER]
skp[1]
run [2] [0] [2] [SN_COURSE_SELECTOR]
put [2] [3]
put [2] [3]
put [2] [SN_PUTTER_LISTER]
put [2] [SN_SITEADM_GROUPS]
run [2] [5] [2] [SN_RUNNER_LISTER]
mov[1][2][1]
mov[2][1][2]
put [1] [0]
skp[1]
run [1] [3] [1] [SN_GROUP_SELECTOR]
put [1] [2]
put [1] [3]
put [1] [SN_PUTTER_LISTER]
put [1] [SN_SITEADM_GROUPS]
run[1][5][2][SN_RUNNER_LISTER]
mov [1] [2] [1]
mov[1][2][1]
mov [2] [1] [2]
mov [2] [1] [1]
mov [1] [2] [3]
mov [2] [1] [1]
put [1] [-1]
put [1] [-1]
mov [2] [1] [1]
mov [2] [1] [1]
put [1] [7]
skp[1]
mov[1][2][1]
mov [1] [2] [1]
put [1] [-1]
put [1] [-1]
```

```
put [1] [-1]
mov [2] [1] [1]
mov [2] [1] [1]
run [1] [5] [1] [SN_USER_SELECTOR]
put [1] [1]
put [1] [1]
put [1] [SN_EMPTY]
put [1] [SN_SITEADM_USERS]
run [1] [6] [1] [SN_RUNNER_LISTER]
end
```

Приложение 8. Протокол компиляции сценария выбора пользователей

```
Запущен автомат: A1 с программой user_master.msl
  Состояние 0: Начальное состояние
    z101 - инициализируем переменные
    z102 - читаем следующую строчку программы, 1
  Состояние 1: Чтение имени
    х101 - является ли именем - Да
    z103 - записываем имя
    z102 - читаем следующую строчку программы, 2
  Состояние 2: Чтение директив
    х102 - является ли директивой - Да
    z104 - записываем директиву
    z102 - читаем следующую строчку программы, 3
  Состояние 2: Чтение директив
    х102 - является ли директивой - Да
    z104 - записываем директиву
    z102 - читаем следующую строчку программы, 4
  Состояние 2: Чтение директив
    х102 - является ли директивой - Да
    z104 - записываем директиву
    z102 - читаем следующую строчку программы, 5
  Состояние 2: Чтение директив
    х102 - является ли директивой - Да
    z104 - записываем директиву
    z102 - читаем следующую строчку программы, 6
  Состояние 2: Чтение директив
    х102 - является ли директивой - Нет
    x105_A - является ли инструкцией get - Нет
```

```
х104 - является ли инструкцией интерпретатора - Да
  z108 - пишем строку директив
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 7
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
  x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Heт
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 8
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 x105_5 - является ли инструкцией интерпретатора skp - Да
  z105_5 - компилируем инструкцию skp
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 9
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
  x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
```

```
x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 10
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 11
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 12
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 13
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 14
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Heт
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
```

```
x105_5 - является ли инструкцией интерпретатора skp - Нет
 х105_6 - является ли инструкцией интерпретатора bck - Нет
 х105_7 - является ли инструкцией интерпретатора jsn - Heт
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 15
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 16
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 17
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 18
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 19
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
```

```
z102 - читаем следующую строчку программы, 20
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 21
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 22
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 23
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
 z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 24
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 25
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
```

```
x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105 5 - является ли инструкцией интерпретатора skp - Да
  z105_5 - компилируем инструкцию skp
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 26
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 27
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Да
 z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 28
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
 x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 29
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 x105_5 - является ли инструкцией интерпретатора skp - Да
```

```
z105_5 - компилируем инструкцию skp
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 30
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 x105_5 - является ли инструкцией интерпретатора skp - Нет
 x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 31
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 32
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 33
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
  х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 34
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
```

```
х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
  x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 35
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 x105_5 - является ли инструкцией интерпретатора skp - Нет
 х105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Her
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 36
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
 z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 37
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 38
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Да
 z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 39
Состояние 4: Чтение инструкций интерпретатора
```

```
x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 x105_5 - является ли инструкцией интерпретатора skp - Да
  z105_5 - компилируем инструкцию skp
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 40
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 х105_5 - является ли инструкцией интерпретатора skp - Нет
 х105_6 - является ли инструкцией интерпретатора bck - Нет
  x105_7 - является ли инструкцией интерпретатора jsn - Нет
 x105_8 - является ли инструкцией интерпретатора run - Да
 z105_8 - компилируем инструкцию run
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 41
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Да
 z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 42
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 43
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
```

```
x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 44
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 45
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Нет
  х105_4 - является ли инструкцией интерпретатора пор - Нет
 x105_5 - является ли инструкцией интерпретатора skp - Нет
 х105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Heт
  x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 46
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 47
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
  х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 48
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
```

```
x105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 49
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
 z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 50
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 51
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 52
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 53
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Да
 z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 54
Состояние 4: Чтение инструкций интерпретатора
```

```
x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 55
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
 z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 56
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
  х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
  x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 57
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Нет
 х105_4 - является ли инструкцией интерпретатора пор - Нет
 x105_5 - является ли инструкцией интерпретатора skp - Да
  z105_5 - компилируем инструкцию skp
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 58
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 59
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
```

```
z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 60
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Да
 z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 61
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
 z105_3 - компилируем инструкцию put
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 62
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
 z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 63
Состояние 4: Чтение инструкций интерпретатора
 x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
  z105_2 - компилируем инструкцию mov
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 64
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 х105_2 - является ли инструкцией интерпретатора mov - Да
 z105_2 - компилируем инструкцию mov
  z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 65
Состояние 4: Чтение инструкций интерпретатора
```

```
x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
  x105_2 - является ли инструкцией интерпретатора mov - Her
 x105_3 - является ли инструкцией интерпретатора put - Нет
  x105_4 - является ли инструкцией интерпретатора nop - Her
 х105_5 - является ли инструкцией интерпретатора skp - Нет
 x105_6 - является ли инструкцией интерпретатора bck - Нет
 x105_7 - является ли инструкцией интерпретатора jsn - Нет
 x105_8 - является ли инструкцией интерпретатора run - Да
  z105_8 - компилируем инструкцию run
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 66
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 67
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Нет
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 68
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
 z106 - увеличиваем номер текущей инструкции
  z102 - читаем следующую строчку программы, 69
Состояние 4: Чтение инструкций интерпретатора
  x105_0 - является ли инструкцией интерпретатора clr - Нет
 х105_1 - является ли инструкцией интерпретатора сру - Нет
 x105_2 - является ли инструкцией интерпретатора mov - Heт
 x105_3 - является ли инструкцией интерпретатора put - Да
  z105_3 - компилируем инструкцию put
```

```
z106 - увеличиваем номер текущей инструкции
    z102 - читаем следующую строчку программы, 70
  Состояние 4: Чтение инструкций интерпретатора
    x105_0 - является ли инструкцией интерпретатора clr - Нет
   х105_1 - является ли инструкцией интерпретатора сру - Нет
   x105_2 - является ли инструкцией интерпретатора mov - Her
   x105_3 - является ли инструкцией интерпретатора put - Нет
   х105_4 - является ли инструкцией интерпретатора пор - Нет
   х105_5 - является ли инструкцией интерпретатора skp - Нет
   х105_6 - является ли инструкцией интерпретатора bck - Нет
   x105_7 - является ли инструкцией интерпретатора jsn - Her
   x105_8 - является ли инструкцией интерпретатора run - Да
    z105_8 - компилируем инструкцию run
   z106 - увеличиваем номер текущей инструкции
    z102 - читаем следующую строчку программы, 71
 Состояние 4: Чтение инструкций интерпретатора
    x105_0 - является ли инструкцией интерпретатора clr - Нет
   х105_1 - является ли инструкцией интерпретатора сру - Нет
   x105_2 - является ли инструкцией интерпретатора mov - Heт
   x105_3 - является ли инструкцией интерпретатора put - Нет
    х105_4 - является ли инструкцией интерпретатора пор - Нет
    x105_5 - является ли инструкцией интерпретатора skp - Нет
   х105_6 - является ли инструкцией интерпретатора bck - Нет
   x105_7 - является ли инструкцией интерпретатора jsn - Her
   x105_8 - является ли инструкцией интерпретатора run - Нет
   x105_9 - является ли инструкцией интерпретатора off - Heт
   х105_В - является ли инструкцией интерпретатора prc - Нет
   x105_C - является ли инструкцией интерпретатора lbl - Heт
   х105_D - является ли инструкцией интерпретатора jmp - Нет
   x105_E - является ли инструкцией интерпретатора jss - Her
   x105_F - является ли инструкцией интерпретатора cll - Нет
   x105_G - является ли инструкцией интерпретатора end - Да
    z105_G - компилируем инструкцию end
    z106 - увеличиваем номер текущей инструкции
 Состояние 5: Нормальное завершение.
    z107 - завершаем запись
Завершен автомат: А1
```

Приложение 9. Протокол интерпретации примера Simple

Запущен автомат: A2 с программой simple

```
Состояние 0: Начальное состояние
  z200 - полключение к базе
 х204 - проверка наличия программы в базе - Да
Состояние 1: Загрузка программы
  z206 - загрузка данных из базы
 х200 - проверка отсутствия программы для возврата - Да
Состояние 2: Обработка директив загрузчика get
  x203 - является ли инструкцией загрузчика get - Нет
Состояние 3: Запуск процессора языка
       - подготовка к запуску автомата АЗ
Запущен автомат: АЗ с программой simple
  Состояние 0: Начальное состояние
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 1
    x301_0 - является ли инструкция инструкцией clr - Нет
    х301_1 - является ли инструкция инструкцией сру - Нет
    x301_2 - является ли инструкция инструкцией mov - Нет
    x301_3 - является ли инструкция инструкцией put - Да
    z301_3 - выполнение инструкции put
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 2
    x301_0 - является ли инструкция инструкцией clr - Нет
    х301_1 - является ли инструкция инструкцией сру - Нет
    х301_2 - является ли инструкция инструкцией mov - Нет
    x301_3 - является ли инструкция инструкцией put - Да
    z301_3 - выполнение инструкции put
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 3
    x301_0 - является ли инструкция инструкцией clr - Нет
    х301_1 - является ли инструкция инструкцией сру - Нет
    x301_2 - является ли инструкция инструкцией mov - Нет
    x301_3 - является ли инструкция инструкцией put - Да
    z301_3 - выполнение инструкции put
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 4
    x301_0 - является ли инструкция инструкцией clr - Нет
    х301_1 - является ли инструкция инструкцией сру - Нет
    x301_2 - является ли инструкция инструкцией mov - Нет
    x301_3 - является ли инструкция инструкцией put - Нет
    х301_4 - является ли инструкция инструкцией пор - Нет
    x301_5 - является ли инструкция инструкцией skp - Нет
```

```
x301_6 - является ли инструкция инструкцией bck - Нет
      x301_7 - является ли инструкция инструкцией jsn - Нет
      x301_8 - является ли инструкция инструкцией run - Да
      z301_8 - выполнение инструкции run
    Состояние 1: Обработка инструкций
      z300 - взятие следующей инструкции, 5
      x301_0 - является ли инструкция инструкцией clr - Нет
      х301_1 - является ли инструкция инструкцией сру - Нет
      х301_2 - является ли инструкция инструкцией mov - Нет
      x301_3 - является ли инструкция инструкцией put - Нет
      х301_4 - является ли инструкция инструкцией пор - Нет
      x301_5 - является ли инструкция инструкцией skp - Нет
      x301_6 - является ли инструкция инструкцией bck - Нет
      x301_7 - является ли инструкция инструкцией jsn - Нет
      x301_8 - является ли инструкция инструкцией run - Нет
      x301_9 - является ли инструкция инструкцией off - Heт
      х301_В - является ли инструкция инструкцией ргс - Нет
      x301_C - является ли инструкция инструкцией lbl - Нет
      x301_D - является ли инструкция инструкцией jmp - Нет
      x301_E - является ли инструкция инструкцией jss - Нет
      x301_F - является ли инструкция инструкцией cll - Нет
      x301_G - является ли инструкция инструкцией end - Да
    Состояние 3: Подготовка к завершению работы автомата
      х302 - требуется ли возврат во внешний автомат А2 - Нет
    Состояние 4: Нормальное завершение
  Завершен автомат: АЗ
  Состояние 5: Нормальное завершение
Завершен автомат: А2
```

Приложение 10. Структура базы данных

```
USE sil;

DROP TABLE wizard_info;

DROP TABLE program_texts;

DROP TABLE queries;
/*
   UNIT....: wizards.sql
```

PROJECT . . . : SIL AUTHOR....: M. Korotkov, A.Loukianova CREATED : 12.06.2003 PURPOSE....: Creates tables in SIL Создает таблицы языка SIL (MSL2.0) NOTES : HISTORY . . . : DATE COMMENT _____ 12.06.2003 Created Создан 22.11.2003 Moved into SIL project Перемещен в проект SIL */ /* TABLENAME: wizard_info FIELDS...: uid - unique id, primary уникальный номер программы, первичный ключ - unique own id id уникальный собственный номер программы program_name - program name имя программы - the first stack stack1 первый стек stack2 - the second stack второй стек cur_instruction - the number of current instruction in MSL program номер текущей инструкции - line of 0,1,2output_format defines output format номер стека для записи выходных данных created - time of creating the record время создания - caller id ret_id номер вызвавшей программы NOTES . . . : table for using MSL programs

```
таблица для программ MSL,
*/
CREATE TABLE wizard_info
    uid
                       INTEGER AUTO_INCREMENT,
    id
                       INTEGER UNSIGNED,
                       CHAR (255),
    program_name
    stack1
                       LONGBLOB,
    stack2
                      LONGBLOB,
    cur_instruction
                      INT,
    max_instruction
                       INT,
    output_type
                       INT,
    created
                       DATETIME,
    ret_id
                       INTEGER UNSIGNED DEFAULT O,
    PRIMARY KEY (uid)
);
CREATE UNIQUE INDEX wizard_id ON wizard_info(id);
/*
  TABLENAME: program_texts
  FIELDS...: name
                                  - unique id
                                    имя программы
                                  - the compiled MSL program
             program
                                    код программы
                                    (байт-код инструкций)
                                  - the instruction numbers
             program_num
                                    of compiled MSL program
                                    строка номеров инструкций
                                    для программы
                                  - the directive line in
             program_directives
                                    $***$value$ format
                                    строка директив программы
                                    в формате $***$value$
             program labels
                                  - the label line in
                                    $label_name$label_value$
                                    format
                                    строка меток программы
                                    в формате
```

```
$label_name$label_value$
 NOTES . . . :
             table for MSL program texts
             таблица для текстов программ
*/
CREATE TABLE program_texts
                              CHAR (255),
    program_name
    program
                              BLOB,
    program_num
                              BLOB,
    program_directives
                              BLOB,
    program_labels
                              BLOB
);
CREATE UNIQUE INDEX program_tname ON program_texts(program_name
  );
/*
  TABLENAME: queries
  FIELDS...: id - DBML query ID
                     идентификатор запроса
             query - query code
                     запрос
  NOTES....: dbml queries table
             Таблица dbml запросов
*/
CREATE TABLE queries
(
          CHAR (255),
    id
    query TEXT
);
CREATE UNIQUE INDEX queries_id ON queries(id);
```

Приложение 11. Листинг файла mslput.php

```
<?php
/**
```

```
UNIT..... mslput.php
  PROJECT...: SIL
  AUTHOR....: М. Коротков, А. Лукьянова
  CREATED . . . : 14.06.2003
  PURPOSE . . . . : Загрузчик в базу данных
  FUNCTIONS . . :
  PARAMS . . . . :
  NOTES . . . . . :
  HISTORY . . . : DATE
                           NOTES
               14.06.2003 Создан
               22.11.2003 Перемещен в
                            проект SIL, в
                            несены изменения
*/
/* Includes */
if (!function_exists("paths_escape"))
require_once("paths.php");
require_once(CGI_PATH.SN_DB);
require_once(CGI_PATH.SN_MSLUTILS);
require_once(CGI_PATH.SN_CONSTS);
echo "<head><link rel=\"stylesheet\" type=\"text/css\" href=\"
  main.css\"/></head><body>";
db_connect();
$f = fopen($scenario, "r") or
   exit("Cannot open file: ".$msl );
$pr_name = fgets($f);
echo("<h2>Loading programs from ".$scenario." file into
  database.</h2>");
while (!ereg("^0", $pr_name))
{
  $pr_name = substr($pr_name, 0, strlen($pr_name)-1);
  echo("<br><i>Adding program \"".$pr_name."\" into database.</
    i><br>");
```

```
$dir_line = mslput_gets($f);
  $lbl_line = mslput_gets($f);
  $num_line = mslput_gets($f);
  $bin_line = mslput_gets($f);
  $result = db_query("max_get_program", $pr_name);
  if (mysql_num_rows(db_query("max_get_program", $pr_name)) == 0)
    echo("  Adding new program.<br>");
    db_query("max_add_program", $bin_line, $num_line, $pr_name
      , $dir_line, $lbl_line);
  } else
  {
    echo("  Modifying existent program.<br>");
    db_query("max_modify_program", $bin_line, $num_line,
      $dir_line, $lbl_line, $pr_name);
 }
  echo("<i>Program adding finished OK.</i><br>");
  $pr_name = fgets($f);
}
echo("<h2>Finished programs loading.</h2>");
echo "</body>";
/**
  FUNCTION ....: mslput_gets
 PARAMS..... $fl - file
 RETURNS....: очередную строку из файла
*/
function mslput_gets($f1)
  $tln = fgets($fl);
 return substr($tln,0,strlen($tln)-1);
}
?>
```

Приложение 12. Протокол интерпретации примера Cycle

```
Запущен автомат: A2 с программой cycle 
Состояние 0: Начальное состояние
```

```
z200 - подключение к базе
 х204 - проверка наличия программы в базе - Да
Состояние 1: Загрузка программы
  z206 - загрузка данных из базы
 х200 - проверка отсутствия программы для возврата - Да
Состояние 2: Обработка директив загрузчика get
 x203 - является ли инструкцией загрузчика get - Нет
Состояние 3: Запуск процессора языка
 z203
      - подготовка к запуску автомата АЗ
Запущен автомат: A3 с программой cycle
  Состояние 0: Начальное состояние
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 1
   x301_0 - является ли инструкция инструкцией clr - Нет
    х301_1 - является ли инструкция инструкцией сру - Нет
    х301_2 - является ли инструкция инструкцией mov - Нет
    x301_3 - является ли инструкция инструкцией put - Да
    z301_3 - выполнение инструкции put
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 2
    x301_0 - является ли инструкция инструкцией clr - Нет
    х301_1 - является ли инструкция инструкцией сру - Нет
    х301_2 - является ли инструкция инструкцией mov - Нет
    x301_3 - является ли инструкция инструкцией put - Нет
    х301_4 - является ли инструкция инструкцией пор - Нет
    x301_5 - является ли инструкция инструкцией skp - Нет
    x301_6 - является ли инструкция инструкцией bck - Нет
    x301_7 - является ли инструкция инструкцией jsn - Нет
    x301_8 - является ли инструкция инструкцией run - Нет
    x301_9 - является ли инструкция инструкцией off - Heт
    x301_B - является ли инструкция инструкцией prc - Нет
    x301_C - является ли инструкция инструкцией lbl - Да
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 3
    x301_0 - является ли инструкция инструкцией clr - Нет
    х301_1 - является ли инструкция инструкцией сру - Нет
    x301_2 - является ли инструкция инструкцией mov - Heт
    x301_3 - является ли инструкция инструкцией put - Да
    z301_3 - выполнение инструкции put
  Состояние 1: Обработка инструкций
    z300 - взятие следующей инструкции, 4
```

```
x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Нет
 x301_6 - является ли инструкция инструкцией bck - Нет
 x301_7 - является ли инструкция инструкцией jsn - Нет
 x301_8 - является ли инструкция инструкцией run - Да
 z301_8 - выполнение инструкции run
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 5
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Нет
 х301_6 - является ли инструкция инструкцией bck - Нет
 x301_7 - является ли инструкция инструкцией jsn - Нет
 x301_8 - является ли инструкция инструкцией run - Да
 z301_8 - выполнение инструкции run
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 6
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Да
 z301_1 - выполнение инструкции сру
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 7
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Да
 z301_5 - выполнение инструкции skp
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 8
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 х301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
```

```
х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Нет
 x301_6 - является ли инструкция инструкцией bck - Нет
 x301_7 - является ли инструкция инструкцией jsn - Нет
 x301_8 - является ли инструкция инструкцией run - Нет
 x301_9 - является ли инструкция инструкцией off - Нет
 x301_B - является ли инструкция инструкцией prc - Нет
 x301_C - является ли инструкция инструкцией lbl - Нет
 x301_D - является ли инструкция инструкцией jmp - Да
 z301_D - выполнение инструкции jmp
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 2
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Нет
 х301_6 - является ли инструкция инструкцией bck - Нет
 x301_7 - является ли инструкция инструкцией jsn - Нет
 x301_8 - является ли инструкция инструкцией run - Нет
 x301_9 - является ли инструкция инструкцией off - Heт
 x301_B - является ли инструкция инструкцией prc - Нет
 x301_C - является ли инструкция инструкцией lbl - Да
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 3
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Да
 z301_3 - выполнение инструкции put
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 4
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Нет
 x301_6 - является ли инструкция инструкцией bck - Нет
 x301_7 - является ли инструкция инструкцией jsn - Нет
 x301_8 - является ли инструкция инструкцией run - Да
```

```
z301_8 - выполнение инструкции run
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 5
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Нет
 x301_6 - является ли инструкция инструкцией bck - Нет
 x301_7 - является ли инструкция инструкцией jsn - Нет
 x301_8 - является ли инструкция инструкцией run - Да
 z301_8 - выполнение инструкции run
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 6
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Да
 z301_1 - выполнение инструкции сру
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 7
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Да
 z301_5 - выполнение инструкции skp
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 9
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Heт
 x301_3 - является ли инструкция инструкцией put - Да
 z301_3 - выполнение инструкции put
Состояние 1: Обработка инструкций
 z300 - взятие следующей инструкции, 10
 x301_0 - является ли инструкция инструкцией clr - Нет
 х301_1 - является ли инструкция инструкцией сру - Нет
 x301_2 - является ли инструкция инструкцией mov - Нет
 x301_3 - является ли инструкция инструкцией put - Нет
 х301_4 - является ли инструкция инструкцией пор - Нет
 x301_5 - является ли инструкция инструкцией skp - Нет
```

```
х301_6 - является ли инструкция инструкцией bck - Нет
     x301_7 - является ли инструкция инструкцией jsn - Нет
     x301_8 - является ли инструкция инструкцией run - Да
     z301_8 - выполнение инструкции run
    Состояние 1: Обработка инструкций
     z300 - взятие следующей инструкции, 11
     x301_0 - является ли инструкция инструкцией clr - Нет
     х301_1 - является ли инструкция инструкцией сру - Нет
     х301_2 - является ли инструкция инструкцией mov - Нет
     x301_3 - является ли инструкция инструкцией put - Нет
     х301_4 - является ли инструкция инструкцией пор - Нет
     x301_5 - является ли инструкция инструкцией skp - Нет
     x301_6 - является ли инструкция инструкцией bck - Нет
     x301_7 - является ли инструкция инструкцией jsn - Heт
     x301_8 - является ли инструкция инструкцией run - Нет
     x301_9 - является ли инструкция инструкцией off - Heт
     х301_В - является ли инструкция инструкцией ргс - Нет
     x301_C - является ли инструкция инструкцией lbl - Нет
     х301_D - является ли инструкция инструкцией jmp - Нет
     x301_E - является ли инструкция инструкцией jss - Нет
     x301_F - является ли инструкция инструкцией cll - Нет
     x301_G - является ли инструкция инструкцией end - Да
    Состояние 3: Подготовка к завершению работы автомата
     х302 - требуется ли возврат во внешний автомат А2 - Нет
    Состояние 4: Нормальное завершение
 Завершен автомат: АЗ
 Состояние 5: Нормальное завершение
Завершен автомат: А2
```