МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Программной инженерии
Специальность 1-40 01 01 «Программное обеспечение информационных технологий»
Специализация 1-40 01 01 10 «Программное обеспечение информационных технологий (программирование интернет-приложений)»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:

«Разработка компилятора KVS-2021»

Выполнил студент Коржова Валерия Сергеевна
(Ф.И.О.)

Руководитель проекта ассистент Мущук Артур Николаевич
(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н. доц. Пацей Наталья Владимировна
(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты ассистент Мущук Артур Николаевич
(учен. степень, звание, должность, подпись, Ф.И.О.)

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер ассистент Мущук Артур Николаевич
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2021

Содержание

Введение

Задачей данного курсового проекта является разработка компилятора для языка программирования KVS-2021. Он предназначен для выполнения простейших операций над строками и числами.

Транслятор – это комплекс отдельных программ, позволяющих преобразовывать исходный код на одном языке программирования в исходный код на другом языке программирования. Транслироваться исходный код языка программирования KVS-2021 будет в исходный код на языке ассемблера. Язык ассемблера – это машинно-ориентированный язык, представляющий формат записи машинных команд, которые понятны для восприятия человеком.

Исходя из цели курсового проекта, были определены следующие задачи:

– разработка спецификации языка программирования;

– разработка структуры транслятора;

– разработка лексического анализатора;

– разработка синтаксического анализатора;

– разработка семантического анализатора;

– обработка выражений с помощью польской инверсии;

– генерация кода на язык ассемблера;

– тестирование транслятора.

Решения каждой из поставленных задач буду приведены в соответствующих главах курсового проекта:

В первой главе работы определена спецификация языка программирования, т.е. описан синтаксис и семантика языка.

Во второй главе работы представлена структура транслятора, т.е. перечислены компоненты транслятора, их назначение и принципы взаимодействия, перечень входных параметров, перечень протоколов, формируемых транслятором и их содержимое.

В третьей главе работы показана разработка лексического анализатора, порождающего таблицы лексем и идентификаторов.

В четвертой главе работы рассказывается о синтаксическом анализаторе, который выполняет синтаксический разбор текста с распечаткой протокола разбора и дерева разбора на основе таблицы лексем.

В пятой главе описан семантический анализатор, показана его работа (распечатка выданных сообщений в трёх примерах на разных этапах).

В шестой главе решены вопросы преобразования выражений, допускаемых языком и приведена часть протокола для контрольного примера, отображающая результаты преобразования выражений в польский формат.

В седьмой главе представлена генерация кода, где из промежуточного представления порождается код на целевом языке.

В восьмой главе описывается тестирование транслятора.

Глава 1. Спецификация языка программирования

1.1 Характеристика языка программирования

Язык программирования KVS-2021 является строго типизированным, не объектно-ориентированным, транслируемым, процедурным языком высокого уровня.

1.2 Алфавит языка

Алфавит языка программирования – набор символов, которые могут использоваться при написании исходного кода.

Символы, разрешенные к использованию при написании кода: кириллица и символы латинского алфавита верхнего и нижнего регистров, арабские цифры, знаки препинания, знаки арифметических и логических операций.

1.3 Применяемые сепараторы

Сепараторы необходимы для разделения операций языка. Сепараторы, используемые в языке программирования KVS-2021, приведены в таблице 1.1.

Таблица 1.1 – Сепараторы

Сепаратор	Назначение
; : « » (пробел) “ ,	Разделение конструкций
=	Оператор присваивания
+ - * /	Арифметические операции
A O ~	Побитовые операции
<>!&	Логические операции
{}	Программный блок инструкций
()	Параметры функций, изменение приоритетов в выражениях
#	Символ, отделяющий условную конструкцию или цикл

1.4 Применяемые кодировки

Для написания исходного кода на языке программирования KVS-2021 используется кодировка Windows-1251 – набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для русских версий Microsoft Windows до 10-й версии – представленная на рисунке 1.1.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
90	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A0	NBSP 00A0	Ў 040E	ў 045E	J 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B0	° 00B0	± 00B1	I 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	j 0458	S 0405	s 0455	ї 0457
C0	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D0	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E0	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F0	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

Рисунок 1.1 – Кодировка Windows-1251

1.5 Типы данных

В язык KVS-2021 предусмотрены два типа данных: целочисленный и строковый, представленные в таблице 1.2. Пользовательские типы данных не поддерживаются.

Таблица 1.2 – Типы данных

Тип данных	Краткое описание	Занимаемое место в памяти	Границы	Значение при инициализации:
number	Целое значение	4 байта	От 0 до 4294967294.	0
string	Строка	256 байт	От 0 до 255 символов	Пустая строка, длина строки 0

1.6 Преобразование типов данных

Преобразование не поддерживается, все типы данных определены однозначно и не могут быть преобразованы в другие, так как язык KVS-2021 является строго типизируемым.

1.7 Идентификаторы

Для именования функций, параметров и переменных используются Идентификаторы. Не предусмотрены зарезервированные идентификаторы, имена идентификаторов не должны совпадать с командами ассемблера. Имя идентификатора составляется по следующим образом:

- состоит из символов латинского алфавита [a..z].
- максимальная длина идентификатора равна 16 символов (8 отводится на префикс, 8 на имя идентификатора).

1.8 Литералы

В языке существует 2 вида литералов: литералы целого типа и строковые, описаны в таблице 1.3. Они осуществляют инициализацию переменных.

Таблица 1.3 – Литералы

Тип	Описание
Литералы целого типа	Интерпретируются как integer, являются rvalue. Задаются в двоичной форме с префиксом “00”, в восьмиричной форме с префиксом “0” и в десятичной форме без префиксов.
Строковые литералы	Интерпретируются как string, заключаются в двойные кавычки (“Hello”), являются rvalue.

1.9 Объявления данных и область видимости

В языке программирования KVS-2021 переменная должны быть объявлена до ее использования. Областью видимости переменной является блок функции, в которой она определена. Вне блока функции определении функции запрещено. Не допустимо объявление глобальных переменных. Область видимости схожа с областью видимости C++, то есть сверху вниз.

1.10 Инициализация данных

Инициализация на языке программирования KVS-2021 будет происходить при объявлении данных начинающихся с ключевого слова new, указывается тип данных и имя идентификатора, где потом инициализируется идентификатор, смотрите таблицу 1.4.

Таблица 1.4 – Инициализация

Инструкция	Форма записи
Инициализация переменной	new <тип данных> <идентификатор>; <идентификатор> = значение;
	new <тип данных> <идентификатор> = значение;

1.11 Инструкции языка

Инструкция для языка KVS-2021 представлена в таблице 1.5.

Таблица 1.5 – Инструкция языка

Инструкция	Запись на языке KVS-2021
Объявление переменной	new <тип данных> <идентификатор>;
Присваивание	<идентификатор> = <значение> | <идентификатор>;
Объявление внешней функции	<тип данных> function <идентификатор> (<тип данных> <идентификатор>, …) {…}
Объявление процедуры	procedure function <идентификатор> (<тип данных> <идентификатор>, …) {…}

Точка входа	main
	{
	…
	}

Возврат значения из подпрограммы	return <идентификатор> | <литерал>;
Вывод данных	write <идентификатор> | <литерал>;
Условный оператор	if <имя переменной, литерал><условный оператор><имя переменной, литерал># true {…} false{…}# Два блока сразу не обязательны, обязательными являются хотя бы один из них
Цикл	if <имя переменной, литерал><условный оператор><имя переменной, литерал># cycle {...}#

1.12 Операции языка

В языке KVS-2021 предусмотрены следующие операции с данными: арифметические, логические, побитовые. Приоритетность побитовых операций выше приоритета арифметических операций, приоритет операций умножения и деления выше приоритета операций сложения и вычитания. Для установки наивысшего приоритета используются круглые скобки. Операции языка представлены в таблице 1.6.

Таблица 1.6 — Операции языка

Тип операции	Приоритет операции	Оператор
Арифметические	+ – 1 */ – 2	+ – сложение * – умножение / – деление = – присваивание
Логические		> – больше < – меньше & – равно ! – не равно
Побитовые	A – 7 O – 8 ~ – 9	A – и O – или ~ – инверсия

1.13 Выражения и их вычисления

В выражении должны участвовать операторы и операнды одного типа, а также функции, возвращающие значения того же типа. Круглые скобки в выражении используются для изменения приоритетов операций. Не допускается запись двух подряд арифметических операций. Также круглые скобки могут использоваться для передачи параметров функций. Фигурные скобки содержат блоки кода функций и циклов.

1.14 Программные конструкции языка

Ключевые программные конструкции языка программирования KVS-2021 представлены в таблице 1.7

Таблица 1.7 – программные конструкции

Главная функция (точка входа в приложение)	main
	{
	…
	};

Функция	<тип> function <идентификатор>(<тип> <идентификатор>)
	{
	…
	return <выражение>;
	};

1.15 Область видимости идентификаторов

В языке KVS-2021 переменные обязаны находится внутри программного блока функций. Все объявления и операции с переменными происходят внутри какого-либо блока. Каждая переменная или параметр функции получают префикс –

название функции, внутри которой они находятся. Объявление глобальных переменных не предусмотрено. Объявление пользовательских областей видимости не предусмотрено.

1.16 Семантические проверки

Таблица с перечнем семантических проверок, предусмотренных языком KVS -2021, приведена в таблице 1.8.

Таблица 1.8 – Семантические проверки

Номер	Правило
1	Наличие функции **main** – точки входа в программу;
2	Единственность точки входа;
3	Переопределение идентификаторов;
4	Использование идентификаторов без их объявления;
5	Проверка соответствия типа функции и возвращаемого параметра;
6	Правильность передаваемых в функцию параметров: количество, типы;
7	Правильность строковых выражений;
8	Превышение размера строковых и числовых литералов;
9	Правильность составленного условия цикла/условного оператора.

1.17 Распределение оперативной памяти на этапе выполнения

Все переменные размещаются в стеке. Таблица лексем и таблица идентификаторов сохраняются в структуры с выделенной под них динамической памятью, которая очищается по окончанию работы транслятора.

1.18 Стандартная библиотека и её состав

Функции стандартной библиотеки с описанием представлены в таблице 1.9. Стандартная библиотека написана на языке программирования C++.

Таблица 1.9 – Состав стандартной библиотеки

Имя функции	Возвращаемое значение	Принимаемые параметры	Описание
strton	number	string x – строка	Функция перевода строки x в число
lenght	number	string x – строка	Функция вычисляет длину строки x
concat	string	string x – строка string y – строка	Функция конкатенации двух строк
outnum	0	number x - число	Функция выводит на консоль число x
outstr	0	string x - строка	Функция выводит на консоль строку x

1.19 Ввод и вывод данных

В языке программирования KVS-2021 ввод данных не поддерживается. Вывод данных происходит с помощью оператора write <идентификатор или литерал>;

1.20 Точка входа

Точкой входа является функция main.

1.21 Препроцессор

Препроцессор в языке программирования KVS-2021 не предусмотрен.

1.22 Соглашения о вызовах

В языке вызов функций происходит по соглашению о вызовах stdcall. Особенности stdcall:

– все параметры функции передаются через стек;
– память освобождает вызываемый код;
– занесение в стек параметров идёт справа налево.

1.23 Объектный код

KVS-2021 транслируется в язык ассемблера.

1.24 Классификация сообщений транслятора

В случае возникновения ошибки в коде программы на языке KVS-2021 и выявления её транслятором в текущий файл протокола выводится сообщение. Их классификация сообщений приведена в таблице 1.10.

Таблица 1.10 - Классификация сообщений транслятора

Номера ошибок	Характеристика
0 – 200	Системные ошибки
200 – 299	Ошибки лексического анализа
300 – 399	Ошибки семантического анализа
600 – 699	Ошибки синтаксического анализа
400-499, 700-999	Зарезервированные коды ошибок

1.25 Контрольный пример

Контрольный пример представлен в приложении А.

Глава 2. Структура транслятора

2.1 Компоненты транслятора, их назначение и принципы взаимодействия

Транслятор преобразует программу, написанную на языке программирования KVS-2021 в программу на языке ассемблера. Компонентами транслятора являются лексический, синтаксический и семантический анализаторы, а также генератор кода на языке ассемблера. Принцип взаимодействия представлен на рисунке 2.1.

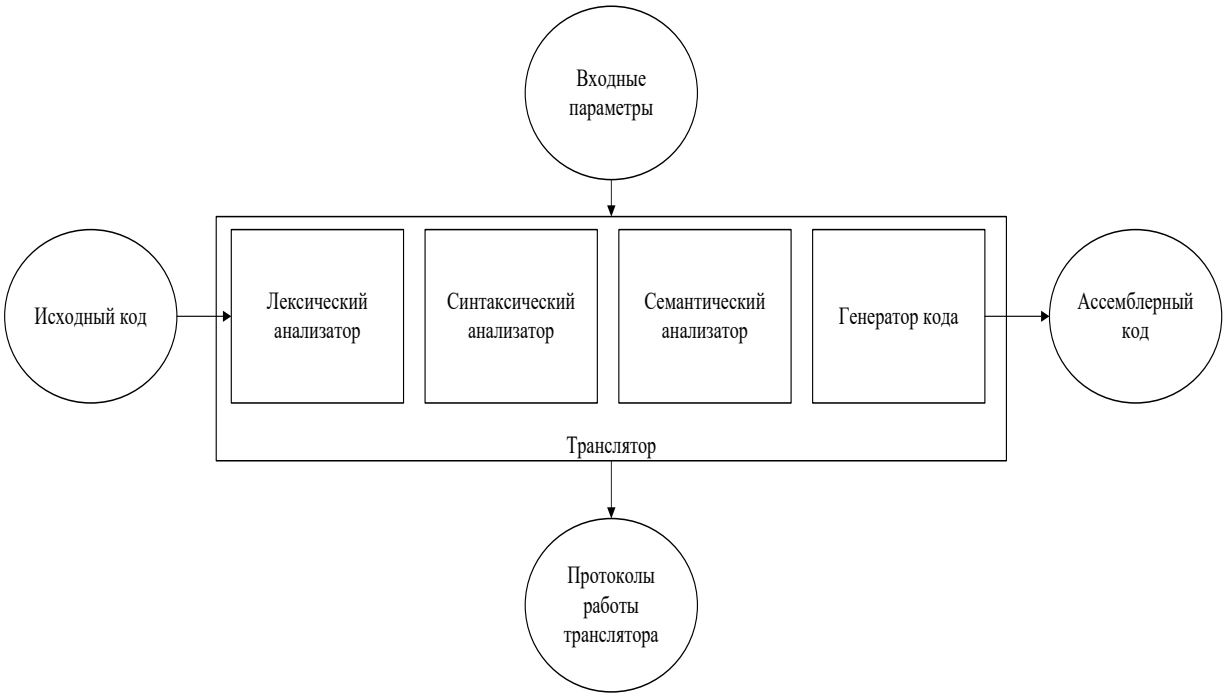

Рисунок 2.1 – Структура транслятора

Лексический анализ – первая фаза трансляции. На вход лексический анализатор получает исходный код на языке программирования KVS-2021, в котором сепараторами были разделены слова. Задачей лексического анализатора является нахождение лексических ошибок и формирование таблиц лексем и идентификаторов.

Синтаксический анализ – это основная часть транслятора, предназначенная для распознавания синтаксических конструкций. Входным параметров для синтаксического анализа является таблица лексем. Синтаксический анализ распознаёт синтаксические конструкции, выявляет синтаксические ошибки при их наличии и формирует дерево разбора.

Семантический анализ, в свою очередь, является проверкой исходного кода программы на семантическую согласованность с определением конструкций языка, то есть проверяет правильность текста исходной программы с точки зрения семантики.

Генератор кода – этап транслятора, выполняющий генерацию ассемблерного кода на основе полученных данных на предыдущих этапах трансляции. Генератор кода принимает на вход таблицы идентификаторов и лексем и транслирует кода на языке программирования KVS-2021 в код на языке Ассемблера.

2.2 Перечень входных параметров транслятора

Входные параметры представлены в таблице 2.1.

Таблица 2.1 - Входные параметры транслятора языка KVS-2021

Входной параметр	Описание	Значение по умолчанию
-in:<имя_файла>	Входной файл с расширением .txt, в котором содержится исходный код на KVS-2021	Не предусмотрено
-log:<имя_файла>	Файл для записи полного протокола работы транслятора	<имя_файла>.log
-out:<имя_файла>	Файл для записи результата работы транслятора	<имя_файла>.asm

2.3 Перечень протоколов, формируемых транслятором и их содержимое

Таблица с перечнем протоколов, формируемых транслятором языка KVS-2021 и их назначением представлена в таблице 2.2

Таблица 2.2 – Протоколы, формируемые транслятором языка KVS-2021

Формируемый протокол	Описание протокола
Файл журнала с параметром <log>	Содержит информацию о времени выполнения приложения; входных параметрах в приложение; код на языке KVS-2021 с сепараторами и без избыточных пробелов, табуляций и переходов на новую строку; таблицу идентификаторов; таблицу лексем; промежуточное представление кода; трассировку синтаксического анализа; дерево разбора, время выполнения разбора; промежуточное представление кода после приведения его к польской нотации.
Выходной файл с параметром <out>	Содержит сгенерированный код на языке Ассемблера.

Имена данных файлов формируются по умолчанию, если они не заданы в параметрах входной строки

Глава 3. Разработка лексического анализатора

3.1 Структура лексического анализатора

Лексический анализатор – часть транслятора, выполняющая лексический анализ. Лексический анализатор принимает обработанный и разбитый на отдельные компоненты исходный код на языке KVS-2021. На выходе формируется таблица лексем и таблица идентификаторов. Структура лексического анализатора представлена на рисунке 3.1.

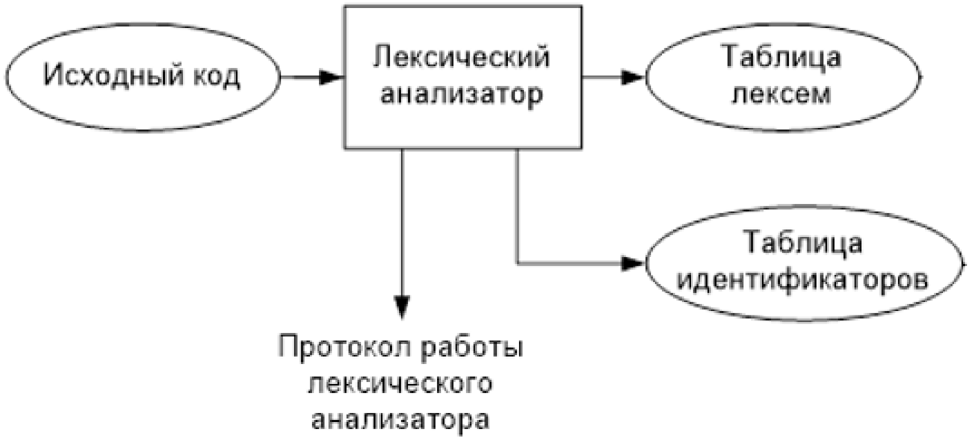

Рисунок 3.1 Структура лексического анализатора KVS-2021

3.2 Контроль входных символов

Таблица для контроля входных символов представлена на рисунке 3.2

```
#define IN_CODE_TABLE   {\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::P, IN::N, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::P, IN::S, IN::Q, IN::S,  IN::T, IN::T, IN::S, IN::F,  IN::S, IN::S, IN::S, IN::S,  IN::S, IN::S, IN::T, IN::S,\
        IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::S,  IN::S, IN::S, IN::S, IN::T,\
        IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,\
        IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::S,  IN::T, IN::S, IN::T, IN::T,\
        IN::F, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,\
        IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,  IN::T, IN::T, IN::T, IN::T,\
                                                                                                                       \
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,\
        IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F,  IN::F, IN::F, IN::F, IN::F \
}
```

Рисунок 3.2 - Таблица контроля входных символов

Принцип работы таблицы заключается в соответствии значения каждому элементу в шестнадцатеричной системе счисления значению в таблице ASCII.

Описание значения символов: T – разрешённый символ, F – запрещённый символ, S – сепаратор, Q – символ ограничивающий литерал, P – пробел и табуляция, I – игнорируемый символ, N – новая строка.

3.3 Удаление избыточных символов

Избыточными символами являются символы табуляции и пробелы.

Избыточные символы удаляются на этапе разбиения исходного кода на лексемы.

Описание алгоритма удаления избыточных символов:

1. Посимвольно считываем исходный код, занесенный в структуру In.
2. Встреча пробела или знака табуляции вне пределов строкового литерала является своего рода встречей символа-сепаратора.
3. В отличие от других символов-сепараторов не записываем в таблицу лексем эти символы, т.е. игнорируем.

3.4 Перечень ключевых слов, сепараторов, символов операций соответствующих им лексем

Лексемы – это символы, соответствующие ключевым словам, символам операций и сепараторам, необходимые для упрощения дальнейшей обработки исходного кода программы. Данное соответствие описано в таблице 3.1.

Таблица 3.1 - Соответствие ключевых слов, символов операций и сепараторов с лексемами

Тип цепочки	Цепочка	Лексема
Ключевые слова	new	n
	number, string	t
	main	m
	function	f
	procedure	p
	return	e
	write	o
	newline	^
	if	?
	cycle	c
	true	r
	false	w
Иное	Идентификатор	i
	Литерал	l

Продолжение таблицы 3.1

Сепараторы	;	;
	,	,
	{	{
	}	}
	(	(
	)	)
	#	#
Операторы	Побитовые (A, O, ~)	A O ~
	Арифметические (+, *, /,)	+ * /
	Логические (& ! > <)	& ! > <
	Присваивание (=)	=

Каждому выражению соответствует детерминированный конечный автомат, то есть автомат с конечным состоянием, по которому происходит разбор данного выражения. На каждый автомат в массиве подаётся фраза и с помощью регулярного выражения, соответствующего данному графу переходов, происходит разбор. В случае успешного разбора выражения оно записывается в таблицу лексем. Если выражение является идентификатором или литералом, информация также заносится в таблицу идентификаторов. Пример реализации таблицы лексем представлен в приложении Б.

3.5 Основные структуры данных

Описание основных структур данных, используемых для хранения таблиц идентификаторов, представлено на рис. 3.3.

```
struct Entry // строка таблицы идентификаторов
{
    union
    {
        int vint;                       //значение integer
        struct
        {
            int len;                    //количество символов
            char str[STR_MAXSIZE - 1];//символы
        } vstr;                         //значение строки
        struct
        {
            int count;                  // количество параметров функции
            IDDATATYPE *types;          //типы параметров функции
        } params;
    } value;                          //значение идентификатора
    int         idxfirstLE;             //индекс в таблице лексем
    char        id[SCOPED_ID_MAXSIZE];  //идентификатор
    IDDATATYPE  iddatatype;             //тип данных
    IDTYPE      idtype;                 //тип идентификатора

    Entry()                           //конструктор без параметров
    {
        this->value.vint = NUM_DEFAULT;
        this->value.vstr.len = NULL;
        this->value.params.count = NULL;
    };
    Entry(char* id, int idxLT, IDDATATYPE datatype, IDTYPE idtype) //конструктор с параметрами
    {
        strncpy_s(this->id, id, SCOPED_ID_MAXSIZE - 1);
        this->idxfirstLE = idxLT;
        this->iddatatype = datatype;
        this->idtype = idtype;
    };
};
```

Рисунок 3.3 — Структуры таблиц идентификаторов KVS-2021

Описание основных структур данных, используемых для хранения таблиц лексем, представлено на рис. 3.4.

```
struct Entry
{
    char lexema;                    //лексема
    int sn;                         //номер строки в исходном тексте
    int idxTI;                      //индекс в ТИ

    Entry();
    Entry(char lexema, int snn, int idxti = NULLDX_TI);
};

struct LexTable                     //экземпляр таблицы лексем
{
    int maxsize;                    //ёмкость таблицы лексем
    int size;                       //текущий размер таблицы лексем
    Entry* table;                   //массив строк ТЛ
};
```

Рисунок 3.4 — Структуры таблиц лексем KVS-2021

3.6 Принцип обработки ошибок

При возникновении ошибки типа предупреждение транслятор продолжает свою работу, а предупреждения записываются в специальную структуру с номером ошибки и диагностическим сообщением.

Когда возникает критическая ошибка – работа транслятора прекращается.

3.7 Структура и перечень сообщений лексического анализатора

Перечень сообщений, генерируемых на этапе лексического анализа, представлен в таблице 3.2.

Таблица 3.2 - Сообщения лексического анализатора

Код	Сообщение
200	Ошибка лексики: Недопустимый символ в исходном файле(-in)
201	Ошибка лексики: Неизвестная последовательность символов
202	Ошибка лексики: Превышен размер таблицы лексем
203	Ошибка лексики: Превышен размер таблицы идентификаторов

3.8 Параметры лексического анализатора и режим его работы

Входным параметром лексического анализатора является исходный текст программы, написанный на языке KVS-2021, а также файл протокола.

3.9 Алгоритм лексического анализа

Лексический анализ выполняется программой (входящей в состав транслятора), называемой лексическим анализатором. Цель лексического анализа — выделение и классификация лексем в тексте исходной программы. Лексический анализатор производит распознаёт и разбирает цепочки исходного текста программы. Это основывается на работе конечных автоматов, которую можно представить в виде графов. Регулярные выражения — аналитический или формульный способ задания регулярных языков. Они состоят из констант и операторов, которые определяют множества строк и множество операций над ними. Любое регулярное выражение можно представить в виде графа.

3.10 Контрольный пример

Результат работы лексического анализатора – вывод в протокол таблицы лексем и идентификаторов – представлен в приложении Б.

Глава 4. Разработка синтаксического анализатора

4.1 Структура синтаксического анализатора.

Синтаксический анализ – это фаза трансляции, выполняемая после лексического анализа и предназначенная для распознавания синтаксических конструкций. Входом для синтаксического анализа является таблица лексем и таблица идентификаторов, полученные после фазы лексического анализа. Выходом – дерево разбора. Структура синтаксического анализатора представлена на рисунке 4.1.

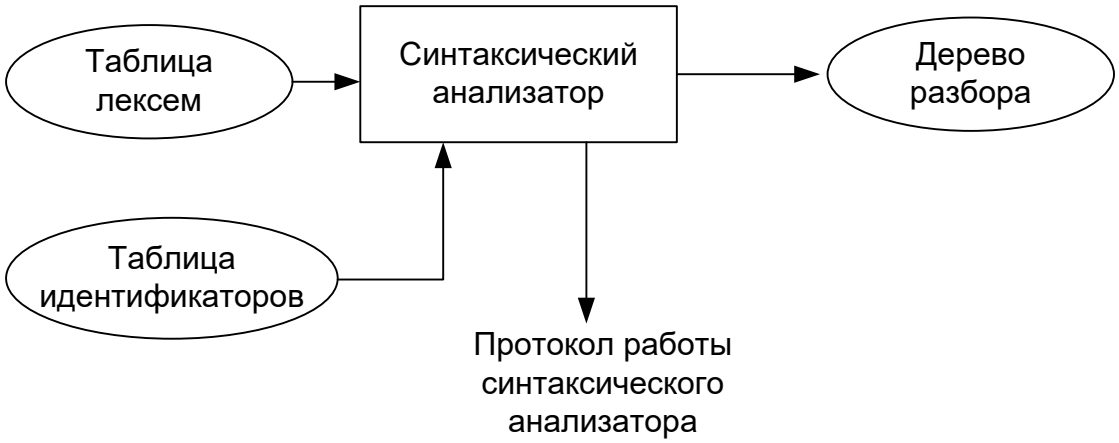

Рисунок 4.1 – Структура синтаксического анализатора

4.2 Контекстно-свободная грамматика, описывающая синтаксис языка

В синтаксическом анализаторе транслятора языка KVS-2021 используется контекстно-свободная грамматика $G = \langle T, N, P, S \rangle$, где

T – множество терминальных символов (было описано в разделе 1.2 данной пояснительной записки),

N – множество нетерминальных символов (первый столбец таблицы 4.1),

P – множество правил языка (второй столбец таблицы 4.1),

S – начальный символ грамматики, являющийся нетерминалом.

Эта грамматика имеет нормальную форму Грейбах, т.к. она не леворекурсивная (не содержит леворекурсивных правил) и правила P имеют вид:

1) $A \to a\alpha$, где $a \in T, \alpha \in (T \cup N) \cup \{\lambda\}$; (или $\alpha \in (T \cup N)^*$, или $\alpha \in V^*$)

2) $S \to \lambda$, где $S \in N$ — начальный символ, при этом если такое правило существует, то нетерминал S не встречается в правой части правил.

Правила языка KVS-2021:

TS – терминальные символы, которыми являются сепараторы, знаки арифметических операций и некоторые строчные буквы.

NS – нетерминальные символы, представленные несколькими заглавными буквами латинского алфавита.

Таблица 4.1 – Перечень правил, составляющих грамматику языка и описание нетерминальных символов KVS-2021

Символ	Правила	Какие правила порождает
S	S->tfiPTS S->pfiPGS S->m{K}	Стартовые правила, описывающее общую структуру программы
P	P->(E) P->()	Правила для параметров объявляемых функций
T	T->{eV;} T->[KeV;]	Правила для тела функций
G	G->{e;} G->{Ke;}	Правила для тела процедур
E	E->ti,E E->ti	Правила для списка параметров функции
F	F->(N) F->()	Правила для вывозов функций(в т.ч. и в выражениях)
N	N->i N>l N->i,N N->l,N	Правила для параметров вызываемых функций
R	R->rY# R>wY# R>cY# R->rYwY# R->wYrY#	Правила составления цикла/условного оператора

Продолжение таблицы 4.1

Z	Z->iLi Z->iLl Z->lLi	Правила для условия цикла/условного оператора
L	L->< L->> L->& L->!	Правила для логических операторов
Q	Q->+ Q->* Q->/ Q->A Q->O Q->~	Правила для арифметических операторов
V	V->l V->i	Правила для простых выражений
Y	Y->{X}	Правила для тела цикла/условного выражения
X	X->i=W;X X->oV;X X->^;X X->iF;X X->i=W; X->oV; X->^; X->iF;	Программные конструкции внутри цикла/условного оператора

Продолжение таблицы 4.1

W	W->l W->i W->(W) W->(W)QW W->~i W->~l W->iF W->iQW W->lQW W->iFQW	Правила для сложных выражений
K	K->nti=V;K K->nti;K K->i=W;K K->oV;K K->^;K K->?Z#RK K->iF;K K->nti=V; K->nti; K->i=W; K->oV; K->^; K->&Z#R K->iF;	Программные конструкции

4.3 Построение конченого магазинного автомата

Конечный автомат с магазинной памятью представляет собой семерку $M = \langle Q, V, Z, \delta, q_0, z_0, F \rangle$, описание которой представлено в таблице 4.2.

Таблица 4.2 – Описание компонентов магазинного автомата

Компонента	Определение	Описание
Q	Множество состояний автомата	Состояние автомата представляет из себя структуру, содержащую позицию на входной ленте, номера текущего правила и цепочки и стек автомата
V	Алфавит входных символов	Алфавит является множеством терминальных и нетерминальных символов, описание которых содержится в разделе 1.2 и в таблице 4.1.
Z	Алфавит специальных магазинных символов	Алфавит магазинных символов содержит стартовый символ и маркер дна стека
δ	Функция переходов автомата	Функция представляет из себя множество правил грамматики, описанных в таблице 4.1.
q_0	Начальное состояние автомата	Состояние, которое приобретает автомат в начале своей работы. Представляется в виде стартового правила грамматики (нетерминальный символ A)
z_0	Начальное состояние магазина автомата	Символ маркера дна стека ($)
F	Множество конечных состояний	Конечные состояние заставляют автомат прекратить свою работу. Конечным состоянием является пустой магазин автомата и совпадение позиции на входной ленте автомата с размером ленты

4.4 Основные структуры данных

Основные структуры данных синтаксического анализатора включают в себя структуру магазинного автомата и структуру грамматики Грейбах, описывающей правила языка KVS-2021.

4.5 Описание алгоритма синтаксического разбора

Принцип работы автомата следующий:

1. В магазин записывается стартовый символ;
2. На основе полученных ранее таблиц формируется входная лента;

3. Запускается автомат;
4. Выбирается цепочка, соответствующая нетерминальному символу, записывается в магазин в обратном порядке;
5. Если терминалы в стеке и в ленте совпадают, то данный терминал удаляется из ленты и стека. Иначе возвращаемся в предыдущее сохраненное состояние и выбираем другую цепочку нетерминала;
6. Если в магазине встретился нетерминал, переходим к пункту 4;
7. Если наш символ достиг дна стека, и лента в этот момент пуста, то синтаксический анализ выполнен успешно. Иначе генерируется исключение.

4.6 Структура и перечень сообщений синтаксического анализатора

Перечень сообщений синтаксического анализатора представлен на рисунке 4.2.

```
ERROR_ENTRY(600, "Ошибка синтаксиса: Неверная структура программы"),
ERROR_ENTRY(601, "Ошибка синтаксиса: Не найден список параметров функции"),
ERROR_ENTRY(602, "Ошибка синтаксиса: Ошибка в теле функции"),
ERROR_ENTRY(603, "Ошибка синтаксиса: Ошибка в теле процедуры"),
ERROR_ENTRY(604, "Ошибка синтаксиса: Ошибка в списке параметров функции"),
ERROR_ENTRY(605, "Ошибка синтаксиса: Ошибка в вызове функции/выражении"),
ERROR_ENTRY(606, "Ошибка синтаксиса: Ошибка в списке фактических параметров функции"),
ERROR_ENTRY(607, "Ошибка синтаксиса: Ошибка при констуировании цикла/условного выражения"),
ERROR_ENTRY(608, "Ошибка синтаксиса: Ошибка в теле цикла/условного выражения"),
ERROR_ENTRY(609, "Ошибка синтаксиса: Ошибка в условии цикла/условного выражения"),
ERROR_ENTRY(610, "Ошибка синтаксиса: Неверный условный оператор"),
ERROR_ENTRY(611, "Ошибка синтаксиса: Неверный арифметический оператор"),
ERROR_ENTRY(612, "Ошибка синтаксиса: Неверное выражение. Ожидаются только идентификаторы/литералы"),
ERROR_ENTRY(613, "Ошибка синтаксиса: Ошибка в арифметическом выражении"),
ERROR_ENTRY(614, "Ошибка синтаксиса: Недопустимая синтаксическая конструкция"),
ERROR_ENTRY(615, "Ошибка синтаксиса: Недопустимая синтаксическая конструкция в теле цикла/условного выражения"),
```

Рисунок 4.2 – Перечень сообщений синтаксического анализатора

4.7 Параметры синтаксического анализатора и режимы его работы

Входным параметром синтаксического анализатора является таблица лексем, полученная на этапе лексического анализа, поток вывода протокола, а также правила контекстно-свободной грамматики в форме Грейбах.

Выходными параметрами являются трассировка прохода таблицы лексем и правила разбора, которые записываются в файл протокола.

4.8 Принцип обработки ошибок

Обработка ошибок происходит следующим образом:

1. Синтаксический анализатор перебирает все правила и цепочки правила грамматики для нахождения подходящего соответствия с конструкцией, представленной в таблице лексем.
2. Если невозможно подобрать подходящую цепочку, то генерируется соответствующая ошибка.
3. Все ошибки записываются в общую структуру ошибок.

4. В случае нахождения ошибки, после всей процедуры трассировки в протокол будет выведено диагностическое сообщение.

4.9 Контрольный пример

Пример разбора синтаксическим анализатором исходного кода предоставлен в приложении В в виде фрагмента трассировки.

Глава 5. Разработка семантического анализатора

5.1 Структура семантического анализатора

Семантический анализ языка KVS-2021 выполняется после выполнения лексического и синтаксического анализа. Несмотря на это, некоторые семантические проверки выполняются на этапе лексического анализа. На вход семантического анализатора подаются таблица лексем и таблица идентификаторов.

5.2 Функции семантического анализатора

Семантический анализатор выполняет проверку на основные правила языка (семантики языка), которые описаны в разделе 1.16.

5.3 Структура и перечень сообщений семантического анализатора

Сообщения, формируемые семантическим анализатором, представлены на рисунке 5.1.

```
ERROR_ENTRY(300, "Ошибка семантики: Необъявленный идентификатор"),
ERROR_ENTRY(301, "Ошибка семантики: Отсутствует точка входа main"),
ERROR_ENTRY(302, "Ошибка семантики: Обнаружено несколько точек входа main"),
ERROR_ENTRY(303, "Ошибка семантики: В объявлении не указан тип идентификатора"),
ERROR_ENTRY(304, "Ошибка семантики: В объявлении отсутствует ключевое слово"),
ERROR_ENTRY(305, "Ошибка семантики: Попытка переопределения идентификатора"),
ERROR_ENTRY(306, "Ошибка семантики: Превышено максимальное количество параметров функции"),
ERROR_ENTRY(307, "Ошибка семантики: Слишком много параметров в вызове"),
ERROR_ENTRY(308, "Ошибка семантики: Кол-во ожидаемыех функцией и передаваемых параметров не совпадают"),
ERROR_ENTRY(309, "Ошибка семантики: Несовпадение типов передаваемых параметров"),
ERROR_ENTRY(310, "Ошибка семантики: Использование пустого строкового литерала недопустимо"),
ERROR_ENTRY(311, "Ошибка семантики: Обнаружен символ \'\"\'. Возможно, не закрыт строковый литерал"),
ERROR_ENTRY(312, "Ошибка семантики: Превышен размер строкового литерала"),
ERROR_ENTRY(313, "Ошибка семантики: Недопустимый целочисленный литерал"),
ERROR_ENTRY(314, "Ошибка семантики: Типы данных в выражении не совпадают"),
ERROR_ENTRY(315, "Ошибка семантики: Тип функции и возвращаемого значения не совпадают"),
ERROR_ENTRY(316, "Ошибка семантики: Недопустимое строковое выражение справа от знака \'=\'"),
ERROR_ENTRY(317, "Ошибка семантики: Неверное условное выражение"),
ERROR_ENTRY(318, "Ошибка семантики: Деление на ноль"),
```

Рисунок 5.1 – Перечень сообщений семантического анализатора

5.4 Принцип обработки ошибок

Принцип обработки ошибок идентичен принципу обработки ошибок на этапе лексического анализа (раздел 3.6).

5.5 Контрольный пример

Результат работы контрольного примера расположен в приложении Б, где показан результат лексического анализатора, т.к. представленные таблицы лексем и идентификаторов проходят лексическую и семантическую проверки одновременно.

Глава 6. Преобразование выражений

6.1 Выражения, допускаемые языком

В языке KVS-2021 допускаются выражения, применимые к целочисленным типам данных. В выражениях поддерживаются арифметические операции, такие как +, -, *, / и (), и вызовы функций как операнды арифметических выражений.

Приоритет операций представлен в таблице 6.1.

Таблица 6.1 – Приоритет операций в языке KVS-2021

Приоритет	Операция
1	+ -
2	* / %
3	()
7	A
8	O
9	~

6.2 Польская запись и принцип ее построения

Выражения в языке KVS-2021 преобразовываются к обратной польской записи.

Польская запись – это альтернативный способ записи арифметических выражений, преимущество которого состоит в отсутствии скобок.

Обратная польская запись — это форма записи математических выражений, в которой операторы расположены после своих операндов. Выражение в обратной польской нотации читается слева направо: операция выполняется над двумя операндами, непосредственно стоящими перед знаком этой операции.

Алгоритм построения:

– исходная строка: выражение;

– результирующая строка: польская запись;

– стек: пустой;

– результирующая строка: польская запись;

– исходная строка просматривается слева направо;

– операнды переносятся в результирующую строку в порядке их следования;

– операция записывается в стек, если стек пуст или в вершине стека лежит отрывающая скобка;

– операция выталкивает все операции с большим или равным приоритетом в результирующую строку;

– запятая не помещается в стек, если в стеке операции, то все выбираются в строку;

– отрывающая скобка помещается в стек;

– закрывающая скобка выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются;

– закрывающая скобка с приоритетом, равным 4, выталкивает все до открывающей с таким же приоритетом и генерирует @ – специальный символ, в

которого записывается информация о вызываемой функции, а в поле приоритета для данной лексемы записывается число параметров вызываемой функции;

– по концу разбора исходной строки все операции, оставшиеся в стеке, выталкиваются в результирующую строку.

Таблица 6.2 – Пример преобразования выражения в обратную польскую запись

Выражение	Стек	Результат
10+4*2/(1-5)		10
+4*2/(1-5)	+	10
4*2/(1-5)	+	104
*2/(1-5)	+*	104
2/(1-5)	+*	1042
/(1-5)	+/	1042*
(1-5)	+/(	1042*
1-5)	+/(	1042*1
-5)	+/(-	1042*1

6.3 Программная реализация обработки выражений

Программная реализация алгоритма преобразования выражений в обратный польский формат основана на функции setPolishNotation. Функция setPolishNotation принимает как параметр адрес таблицы лексем и содержит цикл, в ходе которого перебираются все лексемы исходного кода. Если последовательность лексем соответствует началу выражения, функция возвращает значение true.

6.4 Контрольный пример

Пример преобразования выражения к польской записи представлен в таблице 6.4. Преобразование выражений в формат польской записи необходимо для построения более простых алгоритмов их вычисления.

Глава 7. Генерация кода

7.1 Структура генератора кода

Генерация объектного кода — это перевод компилятором внутреннего представления исходной программы в цепочку символов выходного языка. На вход генератора подаются таблицы лексем и идентификаторов, на основе которых генерируется файл с ассемблерным кодом.

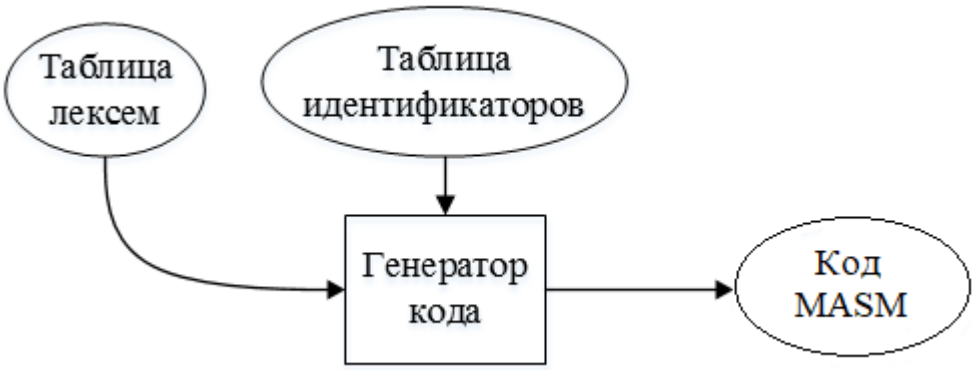

Рисунок 7.1 - Структура генератора кода

Генератор кода последовательно проходит таблицу лексем, при необходимости обращаясь к таблице идентификаторов. В зависимости от пройденных лексем выполняется генерация кода ассемблера.

Обобщенная блок-схема алгоритма генерации кода языка ассемблера изображена на рисунке 7.2.

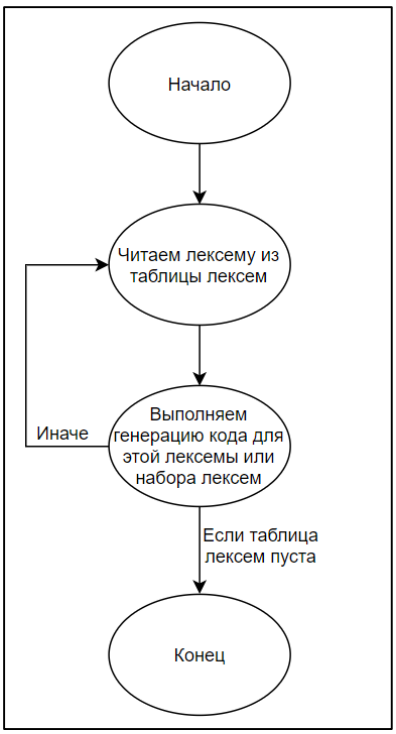

Рисунок 7.2 - Блок-схема алгоритма генерации кода языка ассемблер

7.2 Представление типов данных в памяти

Элементы таблицы идентификаторов расположены в разных сегментах языка ассемблера – .data и .const. Идентификаторы языка KVS-2021 размещены в сегменте данных(.data). Литералы – в сегменте констант (.const). Соответствия между типами данных идентификаторов на языке KVS-2021 и на языке ассемблера приведены в таблице 7.1.

Таблица 7.1 – Соответствия типов идентификаторов языка KVS-2021 и языка Ассемблера

Тип идентификатора на языке KVS-2021	Тип идентификатора на языке ассемблера	Пояснение
number	SDWORD	Хранит беззнаковый целочисленный тип данных.
string	DWORD	Каждый символ строки хранится размером в 1 байт.

7.3 Статическая библиотека

Функции из стандартной библиотеки содержатся в проекте StaticLib, в свойствах которого указан тип конфигурации «статическая библиотека». Подключение библиотеки происходит с помощью includelib на этапе генерации кода путем вывода в поток out. Таким же образом с помощью оператора EXTRN объявляются названия функций из библиотеки. Оператор EXTRN выполняет две функции. Во-первых, он сообщает ассемблеру, что указанное символическое имя является внешним для текущего ассемблирования. Вторая функция оператора EXTRN состоит в том, что он указывает ассемблеру тип соответствующего символического имени. Так как ассемблирование является очень формальной процедурой, то ассемблер должен знать, что представляет из себя каждый символ. Это позволяет ему генерировать правильные команды. Вышеописанное проиллюстрировано на рисунке 7.3.

```
#define BEGIN ".586\n.model flat, stdcall\n"\
"includelib libucrt.lib\n"\
"includelib kernel32.lib\n"\
"includelib \"D:\\KVS-2021\\Debug\\StaticLib.lib\n"\
"ExitProcess PROTO:DWORD \n"\
".stack 4096\n"
```

Рисунок 7.3 - Фрагмент функции генерации кода

Глава 8. Тестирование транслятора

8.1 Тестирование фазы проверки на допустимость символов

В языке KVS-2021 не разрешается использовать запрещённые входным алфавитом символы. Результат использования запрещённого символа показан в таблице 8.1.

Таблица 8.1 – Тестирование фазы проверки на допустимость символов

Исходный код	Диагностическое сообщение
new string s = "привет!";	Ошибка лексики: Недопустимый символ в исходном файле(-in)

8.2 Тестирование лексического анализатора

На этапе лексического анализа могут возникнуть ошибки, описанные в пункте 3.7. Результаты тестирования лексического анализатора показаны в таблице 8.2.

Таблица 8.2 – Тестирование лексического анализатора

Исходный код	Диагностическое сообщение
n = n % s	Ошибка лексики: Неизвестная последовательность символов

8.3 Тестирование синтаксического анализатора

На этапе синтаксического анализа могут возникнуть ошибки, описанные в пункте 4.6. Результаты тестирования синтаксического анализатора показаны в таблице 8.3.

Таблица 8.3 – Тестирование синтаксического анализатора

Исходный код	Диагностическое сообщение
n = write n;	Ошибка синтаксиса: Ошибка в арифметическом выражении

8.4 Тестирование семантического анализатора

Итоги тестирования семантического анализатора приведены в таблице 8.4

Таблица 8.4 – Тестирование семантического анализатора

Исходный код	Диагностическое сообщение
n = "jjjjj;	Ошибка семантики: Обнаружен символ "". Возможно, не закрыт строковый литерал

Приложения

Приложение А

```
number function sys(string str)
{
new number l;
l = lenght(str);
return l;
}

procedure function op(number a, number b)
{
write "a = ";
write a;
write ", b = ";
write b;
newline;
new number c;
c = ~ a;
write "inversion of a = ";
write c;
newline;
c = a O b;
write "logical or between a, b = ";
write c;
newline;
c = a A b;
write "logical and between a, b = ";
write c;
newline;
c = (b + a) * (b + a) / 2;
write "(b + a) * (b + a) / 2 = ";
write c;
newline;
return;
}
```

```
main
{
new number x = 6;
new number y = 10;
op(x, y);

new number z;
new string strx = "hello";
z = lenght(strx);
write "length of *hello* is ";
write z;
newline;

new string stry = " world!";
new string result;
result = concat(strx, stry);
write result;
newline;

if x > y#
true
{
write "x is bigger than y";
}
false
{
write "y is bigger than x";
}#
newline;
new number res;
new string s = "8";
res = strton (s);
write res;
newline;
if res > 1#
cycle
{
write res;
newline;
res = res / 2;
}#
}
```

Приложение Б

---------- ТАБЛИЦА ЛЕКСЕМ ----------

№	ЛЕКСЕМА	СТРОКА	ИНДЕКС В ТИ
0	t	1	
1	f	1	
2	i	1	0
3	(	1	
4	t	1	
5	i	1	1
6	)	1	
7	{	2	
8	n	3	
9	t	3	
10	i	3	2
11	;	3	
12	i	4	2
13	=	4	
14	i	4	3
15	(	4	
16	i	4	1
17	)	4	
18	;	4	
19	e	5	
20	i	5	2
21	;	5	
22	}	6	
23	p	8	
24	f	8	
25	i	8	4
26	(	8	
27	t	8	
28	i	8	5
29	,	8	
30	t	8	
31	i	8	6
32	)	8	
33	{	9	
34	o	10	
35	l	10	7
36	;	10	
37	o	11	
38	i	11	5
39	;	11	
40	o	12	
41	l	12	8
42	;	12	
43	o	13	
44	i	13	6
45	;	13	
46	^	14	
47	;	14	
48	n	15	
49	t	15	
50	i	15	9
51	;	15	
52	i	16	9
53	=	16	
54	~	16	
55	i	16	5
56	;	16	
57	o	17	
58	l	17	10
59	;	17	
60	o	18	
61	i	18	9
62	;	18	
63	^	19	
64	;	19	
65	i	20	9
66	=	20	
67	i	20	5
68	O	20	
69	i	20	6
70	;	20	
71	o	21	
72	l	21	11
73	;	21	
74	o	22	
75	i	22	9
76	;	22	
77	^	23	
78	;	23	
79	i	24	9
80	=	24	
81	i	24	5
82	A	24	
83	i	24	6
84	;	24	
85	o	25	
86	l	25	12
87	;	25	

88	o	26	
89	i	26	9
90	;	26	
91	^	27	
92	;	27	
93	i	28	9
94	=	28	
95	(	28	
96	i	28	6
97	+	28	
98	1	28	5
99	)	28	
100	*	28	
101	(	28	
102	i	28	6
103	+	28	
104	1	28	5
105	)	28	
106	/	28	
107	1	28	13
108	;	28	
109	o	29	
110	1	29	14
111	;	29	
112	o	30	
113	1	30	9
114	;	30	
115	^	31	
116	;	31	
117	e	32	
118	;	32	
119	}	33	
120	m	35	
121	{	36	
122	n	37	
123	t	37	
124	i	37	15
125	=	37	
126	1	37	16
127	;	37	
128	n	38	
129	t	38	
130	i	38	17
131	=	38	
132	1	38	18
133	;	38	
134	i	39	4
135	(	39	
136	i	39	15
137	,	39	
138	i	39	17
139	)	39	
140	;	39	
141	n	41	
142	t	41	
143	i	41	19
144	;	41	
145	n	42	
146	t	42	
147	i	42	20
148	=	42	
149	1	42	21
150	;	42	
151	i	43	19
152	=	43	
153	i	43	3
154	(	43	
155	i	43	20
156	)	43	
157	;	43	
158	o	44	
159	1	44	22
160	;	44	
161	o	45	
162	i	45	19
163	;	45	
164	^	46	
165	;	46	
166	n	48	
167	t	48	
168	i	48	23
169	=	48	
170	1	48	24
171	;	48	
172	n	49	
173	t	49	
174	i	49	25
175	;	49	
176	i	50	25
177	=	50	
178	i	50	26
179	(	50	

180	i	50	20
181	,	50	
182	i	50	23
183	)	50	
184	;	50	
185	o	51	
186	i	51	25
187	;	51	
188	^	52	
189	;	52	
190	?	54	
191	i	54	15
192	>	54	
193	i	54	17
194	#	54	
195	r	55	
196	{	56	
197	o	57	
198	l	57	27
199	;	57	
200	}	58	
201	w	59	
202	{	60	
203	o	61	
204	l	61	28
205	;	61	
206	}	62	
207	#	62	
208	^	63	
209	;	63	
210	n	64	
211	t	64	
212	i	64	29
213	;	64	
214	n	65	
215	t	65	
216	i	65	30
217	=	65	
218	l	65	31
219	;	65	
220	i	66	29
221	=	66	
222	i	66	32
223	(	66	
224	i	66	30
225	)	66	
226	;	66	
227	o	67	
228	i	67	29
229	;	67	
230	^	68	
231	;	68	
232	?	69	
233	i	69	29
234	>	69	
235	l	69	33
236	#	69	
237	c	70	
238	{	71	
239	o	72	
240	i	72	29
241	;	72	
242	^	73	
243	;	73	
244	i	74	29
245	=	74	
246	i	74	29
247	/	74	
248	l	74	13
249	;	74	
250	}	75	
251	#	75	
252	}	76	

---------- ТАБЛИЦА ИДЕНТИФИКАТОРОВ ----------

N	СТРОКА В ТЛ	ТИП ИДЕНТИФИКАТОРА	ИМЯ	ЗНАЧЕНИЕ (ПАРАМЕТРЫ)
0	2	number function	sys	P0:STRING |
1	5	string parameter	sysstr	
2	10	number variable	sysl	0
3	14	number LIB FUNC	lenght	P0:STRING |
4	25	proc function	op	P0:NUMBER | P1:NUMBER |
5	28	number parameter	opa	
6	31	number parameter	opb	
7	35	string literal	LTRL1	[4]a =
8	41	string literal	LTRL2	[6], b =
9	50	number variable	opc	0
10	58	string literal	LTRL3	[17]inversion of a =
11	72	string literal	LTRL4	[26]logical or between a, b =
12	86	string literal	LTRL5	[27]logical and between a, b =
13	107	number literal	LTRL6	2
14	110	string literal	LTRL7	[24](b + a) * (b + a) / 2 =
15	124	number variable	mainx	0
16	126	number literal	LTRL8	6
17	130	number variable	mainy	0
18	132	number literal	LTRL9	10
19	143	number variable	mainz	0
20	147	string variable	mainstrx	[0]
21	149	string literal	LTRL10	[5]hello
22	159	string literal	LTRL11	[21]length of *hello* is
23	168	string variable	mainstry	[0]
24	170	string literal	LTRL12	[7] world!
25	174	string variable	mainresult	[0]
26	178	string LIB FUNC	concat	P0:STRING | P1:STRING |
27	198	string literal	LTRL13	[18]x is bigger than y
28	204	string literal	LTRL14	[18]y is bigger than x
29	212	number variable	opres	0
30	216	string variable	ops	[0]
31	218	string literal	LTRL15	[1]8
32	222	number LIB FUNC	strton	P0:STRING |
33	235	number literal	LTRL16	1

Приложение В

```
Шаг :Правило              Входная лента                 Стек
0   :S->tfiPTS            tfi(ti){nti;i=i(i);ei;}pf     S$
1   : SAVESTATE:           1
1   :                     tfi(ti){nti;i=i(i);ei;}pf     tfiPTS$
2   :                     fi(ti){nti;i=i(i);ei;}pfi     fiPTS$
3   :                     i(ti){nti;i=i(i);ei;}pfi(     iPTS$
4   :                     (ti){nti;i=i(i);ei;}pfi(t     PTS$
5   :P->(E)               (ti){nti;i=i(i);ei;}pfi(t     PTS$
6   : SAVESTATE:           2
6   :                     (ti){nti;i=i(i);ei;}pfi(t     (E)TS$
7   :                     ti){nti;i=i(i);ei;}pfi(ti     E)TS$
8   :E->ti,E              ti){nti;i=i(i);ei;}pfi(ti     E)TS$
9   : SAVESTATE:           3
9   :                     ti){nti;i=i(i);ei;}pfi(ti     ti,E)TS$
10  :                     i){nti;i=i(i);ei;}pfi(ti,     i,E)TS$
11  :                     ){nti;i=i(i);ei;}pfi(ti,t     ,E)TS$
12  : TS_NOK/NS_NORULECHAIN
12  : RESSTATE
12  :                     ti){nti;i=i(i);ei;}pfi(ti     E)TS$
13  :E->ti                ti){nti;i=i(i);ei;}pfi(ti     E)TS$
14  : SAVESTATE:           3
14  :                     ti){nti;i=i(i);ei;}pfi(ti     ti)TS$
15  :                     i){nti;i=i(i);ei;}pfi(ti,     i)TS$
16  :                     ){nti;i=i(i);ei;}pfi(ti,t     )TS$
17  :                     {nti;i=i(i);ei;}pfi(ti,ti     TS$
18  :T->{eV;}             {nti;i=i(i);ei;}pfi(ti,ti     TS$
19  : SAVESTATE:           4
19  :                     {nti;i=i(i);ei;}pfi(ti,ti     {eV;}S$
20  :                     nti;i=i(i);ei;}pfi(ti,ti)     eV;}S$
21  : TS_NOK/NS_NORULECHAIN
21  : RESSTATE
21  :                     {nti;i=i(i);ei;}pfi(ti,ti     TS$
22  :T->{KeV;}            {nti;i=i(i);ei;}pfi(ti,ti     TS$
23  : SAVESTATE:           4
23  :                     {nti;i=i(i);ei;}pfi(ti,ti     {KeV;}S$
24  :                     nti;i=i(i);ei;}pfi(ti,ti)     KeV;}S$
25  :K->nti=V;K           nti;i=i(i);ei;}pfi(ti,ti)     KeV;}S$
26  : SAVESTATE:           5
26  :                     nti;i=i(i);ei;}pfi(ti,ti)     nti=V;KeV;}S$
27  :                     ti;i=i(i);ei;}pfi(ti,ti){     ti=V;KeV;}S$
28  :                     i;i=i(i);ei;}pfi(ti,ti){o     i=V;KeV;}S$
29  :                     ;i=i(i);ei;}pfi(ti,ti){ol     =V;KeV;}S$
30  : TS_NOK/NS_NORULECHAIN
30  : RESSTATE
30  :                     nti;i=i(i);ei;}pfi(ti,ti)     KeV;}S$
31  :K->nti;K             nti;i=i(i);ei;}pfi(ti,ti)     KeV;}S$
32  : SAVESTATE:           5
32  :                     nti;i=i(i);ei;}pfi(ti,ti)     nti;KeV;}S$
33  :                     ti;i=i(i);ei;}pfi(ti,ti){     ti;KeV;}S$
34  :                     i;i=i(i);ei;}pfi(ti,ti){o     i;KeV;}S$
35  :                     ;i=i(i);ei;}pfi(ti,ti){ol     ;KeV;}S$
36  :                     i=i(i);ei;}pfi(ti,ti){ol;     KeV;}S$
37  :K->i=W;K             i=i(i);ei;}pfi(ti,ti){ol;     KeV;}S$
38  : SAVESTATE:           6
38  :                     i=i(i);ei;}pfi(ti,ti){ol;     i=W;KeV;}S$
39  :                     =i(i);ei;}pfi(ti,ti){ol;o     =W;KeV;}S$
40  :                     i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
41  :W->i                 i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
42  : SAVESTATE:           7
42  :                     i(i);ei;}pfi(ti,ti){ol;oi     i;KeV;}S$
43  :                     (i);ei;}pfi(ti,ti){ol;oi;     ;KeV;}S$
44  : TS_NOK/NS_NORULECHAIN
44  : RESSTATE
44  :                     i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
45  :W->iF                i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
46  : SAVESTATE:           7
46  :                     i(i);ei;}pfi(ti,ti){ol;oi     iF;KeV;}S$
47  :                     (i);ei;}pfi(ti,ti){ol;oi;     F;KeV;}S$
48  :F->(N)               (i);ei;}pfi(ti,ti){ol;oi;     F;KeV;}S$
49  : SAVESTATE:           8
49  :                     (i);ei;}pfi(ti,ti){ol;oi;     (N);KeV;}S$
```

```
50  :                    i);ei;}pfi(ti,ti){ol;oi;o     N);KeV;}S$
51  :N->i                i);ei;}pfi(ti,ti){ol;oi;o     N);KeV;}S$
52  : SAVESTATE:          9
52  :                    i);ei;}pfi(ti,ti){ol;oi;o     i);KeV;}S$
53  :                    );ei;}pfi(ti,ti){ol;oi;ol     );KeV;}S$
54  :                    ;ei;}pfi(ti,ti){ol;oi;ol;     ;KeV;}S$
55  :                    ei;}pfi(ti,ti){ol;oi;ol;o     KeV;}S$
56  : TNS_NORULECHAIN/NS_NORULE
56  : RESSTATE
56  :                    i);ei;}pfi(ti,ti){ol;oi;o     N);KeV;}S$
57  :N->i,N              i);ei;}pfi(ti,ti){ol;oi;o     N);KeV;}S$
58  : SAVESTATE:          9
58  :                    i);ei;}pfi(ti,ti){ol;oi;o     i,N);KeV;}S$
59  :                    );ei;}pfi(ti,ti){ol;oi;ol     ,N);KeV;}S$
60  : TS_NOK/NS_NORULECHAIN
60  : RESSTATE
60  :                    i);ei;}pfi(ti,ti){ol;oi;o     N);KeV;}S$
61  : TNS_NORULECHAIN/NS_NORULE
61  : RESSTATE
61  :                    (i);ei;}pfi(ti,ti){ol;oi;     F;KeV;}S$
62  :F->()               (i);ei;}pfi(ti,ti){ol;oi;     F;KeV;}S$
63  : SAVESTATE:          8
63  :                    (i);ei;}pfi(ti,ti){ol;oi;     ();KeV;}S$
64  :                    i);ei;}pfi(ti,ti){ol;oi;o     );KeV;}S$
65  : TS_NOK/NS_NORULECHAIN
65  : RESSTATE
65  :                    (i);ei;}pfi(ti,ti){ol;oi;     F;KeV;}S$
66  : TNS_NORULECHAIN/NS_NORULE
66  : RESSTATE
66  :                    i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
67  :W->iQW              i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
68  : SAVESTATE:          7
68  :                    i(i);ei;}pfi(ti,ti){ol;oi     iQW;KeV;}S$
69  :                    (i);ei;}pfi(ti,ti){ol;oi;     QW;KeV;}S$
70  : TNS_NORULECHAIN/NS_NORULE
70  : RESSTATE
70  :                    i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
71  :W->iFQW             i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
72  : SAVESTATE:          7
72  :                    i(i);ei;}pfi(ti,ti){ol;oi     iFQW;KeV;}S$
73  :                    (i);ei;}pfi(ti,ti){ol;oi;     FQW;KeV;}S$
74  :F->(N)              (i);ei;}pfi(ti,ti){ol;oi;     FQW;KeV;}S$
75  : SAVESTATE:          8
75  :                    (i);ei;}pfi(ti,ti){ol;oi;     (N)QW;KeV;}S$
76  :                    i);ei;}pfi(ti,ti){ol;oi;o     N)QW;KeV;}S$
77  :N->i                i);ei;}pfi(ti,ti){ol;oi;o     N)QW;KeV;}S$
78  : SAVESTATE:          9
78  :                    i);ei;}pfi(ti,ti){ol;oi;o     i)QW;KeV;}S$
79  :                    );ei;}pfi(ti,ti){ol;oi;ol     )QW;KeV;}S$
80  :                    ;ei;}pfi(ti,ti){ol;oi;ol;     QW;KeV;}S$
81  : TNS_NORULECHAIN/NS_NORULE
81  : RESSTATE
81  :                    i);ei;}pfi(ti,ti){ol;oi;o     N)QW;KeV;}S$
82  :N->i,N              i);ei;}pfi(ti,ti){ol;oi;o     N)QW;KeV;}S$
83  : SAVESTATE:          9
83  :                    i);ei;}pfi(ti,ti){ol;oi;o     i,N)QW;KeV;}S$
84  :                    );ei;}pfi(ti,ti){ol;oi;ol     ,N)QW;KeV;}S$
85  : TS_NOK/NS_NORULECHAIN
85  : RESSTATE
85  :                    i);ei;}pfi(ti,ti){ol;oi;o     N)QW;KeV;}S$
86  : TNS_NORULECHAIN/NS_NORULE
86  : RESSTATE
86  :                    (i);ei;}pfi(ti,ti){ol;oi;     FQW;KeV;}S$
87  :F->()               (i);ei;}pfi(ti,ti){ol;oi;     FQW;KeV;}S$
88  : SAVESTATE:          8
88  :                    (i);ei;}pfi(ti,ti){ol;oi;     ()QW;KeV;}S$
89  :                    i);ei;}pfi(ti,ti){ol;oi;o     )QW;KeV;}S$
90  : TS_NOK/NS_NORULECHAIN
90  : RESSTATE
90  :                    (i);ei;}pfi(ti,ti){ol;oi;     FQW;KeV;}S$
91  : TNS_NORULECHAIN/NS_NORULE
```

```
91  : RESSTATE
91  :                    i(i);ei;}pfi(ti,ti){ol;oi     W;KeV;}S$
92  : TNS_NORULECHAIN/NS_NORULE
92  : RESSTATE
92  :                    i=i(i);ei;}pfi(ti,ti){ol;     KeV;}S$
93  :K->iF;K             i=i(i);ei;}pfi(ti,ti){ol;     KeV;}S$
94  : SAVESTATE:          6
94  :                    i=i(i);ei;}pfi(ti,ti){ol;     iF;KeV;}S$
95  :                    =i(i);ei;}pfi(ti,ti){ol;o     F;KeV;}S$
96  : TNS_NORULECHAIN/NS_NORULE
96  : RESSTATE
96  :                    i=i(i);ei;}pfi(ti,ti){ol;     KeV;}S$
97  :K->i=W;             i=i(i);ei;}pfi(ti,ti){ol;     KeV;}S$
98  : SAVESTATE:          6
98  :                    i=i(i);ei;}pfi(ti,ti){ol;     i=W;eV;}S$
99  :                    =i(i);ei;}pfi(ti,ti){ol;o     =W;eV;}S$
100 :                    i(i);ei;}pfi(ti,ti){ol;oi     W;eV;}S$
101 :W->i                i(i);ei;}pfi(ti,ti){ol;oi     W;eV;}S$
102 : SAVESTATE:          7
102 :                    i(i);ei;}pfi(ti,ti){ol;oi     i;eV;}S$
103 :                    (i);ei;}pfi(ti,ti){ol;oi;     ;eV;}S$
104 : TS_NOK/NS_NORULECHAIN
104 : RESSTATE
104 :                    i(i);ei;}pfi(ti,ti){ol;oi     W;eV;}S$
105 :W->iF               i(i);ei;}pfi(ti,ti){ol;oi     W;eV;}S$
106 : SAVESTATE:          7
106 :                    i(i);ei;}pfi(ti,ti){ol;oi     iF;eV;}S$
107 :                    (i);ei;}pfi(ti,ti){ol;oi;     F;eV;}S$
108 :F->(N)              (i);ei;}pfi(ti,ti){ol;oi;     F;eV;}S$
109 : SAVESTATE:          8
109 :                    (i);ei;}pfi(ti,ti){ol;oi;     (N);eV;}S$
110 :                    i);ei;}pfi(ti,ti){ol;oi;o     N);eV;}S$
111 :N->i                i);ei;}pfi(ti,ti){ol;oi;o     N);eV;}S$
112 : SAVESTATE:          9
112 :                    i);ei;}pfi(ti,ti){ol;oi;o     i);eV;}S$
113 :                    );ei;}pfi(ti,ti){ol;oi;ol     );eV;}S$
114 :                    ;ei;}pfi(ti,ti){ol;oi;ol;     ;eV;}S$
115 :                    ei;}pfi(ti,ti){ol;oi;ol;o     eV;}S$
116 :                    i;}pfi(ti,ti){ol;oi;ol;oi     V;}S$
117 :V->i                i;}pfi(ti,ti){ol;oi;ol;oi     V;}S$
118 : SAVESTATE:          10
118 :                    i;}pfi(ti,ti){ol;oi;ol;oi     i;}S$
119 :                    ;}pfi(ti,ti){ol;oi;ol;oi;     ;}S$
120 :                    }pfi(ti,ti){ol;oi;ol;oi;^     }S$
121 :                    pfi(ti,ti){ol;oi;ol;oi;^;     S$
122 :S->pfiPGS           pfi(ti,ti){ol;oi;ol;oi;^;     S$
123 : SAVESTATE:          11
123 :                    pfi(ti,ti){ol;oi;ol;oi;^;     pfiPGS$
124 :                    fi(ti,ti){ol;oi;ol;oi;^;n     fiPGS$
125 :                    i(ti,ti){ol;oi;ol;oi;^;nt     iPGS$
126 :                    (ti,ti){ol;oi;ol;oi;^;nti     PGS$
127 :P->(E)              (ti,ti){ol;oi;ol;oi;^;nti     PGS$
128 : SAVESTATE:          12
128 :                    (ti,ti){ol;oi;ol;oi;^;nti     (E)GS$
129 :                    ti,ti){ol;oi;ol;oi;^;nti;     E)GS$
130 :E->ti,E             ti,ti){ol;oi;ol;oi;^;nti;     E)GS$
131 : SAVESTATE:          13
131 :                    ti,ti){ol;oi;ol;oi;^;nti;     ti,E)GS$
132 :                    i,ti){ol;oi;ol;oi;^;nti;i     i,E)GS$
133 :                    ,ti){ol;oi;ol;oi;^;nti;i=     ,E)GS$
134 :                    ti){ol;oi;ol;oi;^;nti;i=~     E)GS$
135 :E->ti,E             ti){ol;oi;ol;oi;^;nti;i=~     E)GS$
136 : SAVESTATE:          14
136 :                    ti){ol;oi;ol;oi;^;nti;i=~     ti,E)GS$
137 :                    i){ol;oi;ol;oi;^;nti;i=~i     i,E)GS$
138 :                    ){ol;oi;ol;oi;^;nti;i=~i;     ,E)GS$
139 : TS_NOK/NS_NORULECHAIN
139 : RESSTATE
139 :                    ti){ol;oi;ol;oi;^;nti;i=~     E)GS$
140 :E->ti               ti){ol;oi;ol;oi;^;nti;i=~     E)GS$
141 : SAVESTATE:          14
```

```
141 :                    ti){ol;oi;ol;oi;^;nti;i=~     ti)GS$
142 :                    i){ol;oi;ol;oi;^;nti;i=~i     i)GS$
143 :                    ){ol;oi;ol;oi;^;nti;i=~i;     )GS$
144 :                    {ol;oi;ol;oi;^;nti;i=~i;o     GS$
145 :G->{e;}              {ol;oi;ol;oi;^;nti;i=~i;o     GS$
146 : SAVESTATE:           15
146 :                    {ol;oi;ol;oi;^;nti;i=~i;o     {e;}S$
147 :                    ol;oi;ol;oi;^;nti;i=~i;ol     e;}S$
148 : TS_NOK/NS_NORULECHAIN
148 : RESSTATE
148 :                    {ol;oi;ol;oi;^;nti;i=~i;o     GS$
149 :G->{Ke;}            {ol;oi;ol;oi;^;nti;i=~i;o     GS$
150 : SAVESTATE:           15
150 :                    {ol;oi;ol;oi;^;nti;i=~i;o     {Ke;}S$
151 :                    ol;oi;ol;oi;^;nti;i=~i;ol     Ke;}S$
152 :K->oV;K             ol;oi;ol;oi;^;nti;i=~i;ol     Ke;}S$
153 : SAVESTATE:           16
153 :                    ol;oi;ol;oi;^;nti;i=~i;ol     oV;Ke;}S$
154 :                    l;oi;ol;oi;^;nti;i=~i;ol;     V;Ke;}S$
155 :V->l                l;oi;ol;oi;^;nti;i=~i;ol;     V;Ke;}S$
156 : SAVESTATE:           17
156 :                    l;oi;ol;oi;^;nti;i=~i;ol;     l;Ke;}S$
157 :                    ;oi;ol;oi;^;nti;i=~i;ol;o     ;Ke;}S$
158 :                    oi;ol;oi;^;nti;i=~i;ol;oi     Ke;}S$
159 :K->oV;K             oi;ol;oi;^;nti;i=~i;ol;oi     Ke;}S$
160 : SAVESTATE:           18
160 :                    oi;ol;oi;^;nti;i=~i;ol;oi     oV;Ke;}S$
161 :                    i;ol;oi;^;nti;i=~i;ol;oi;     V;Ke;}S$
162 :V->i                i;ol;oi;^;nti;i=~i;ol;oi;     V;Ke;}S$
163 : SAVESTATE:           19
163 :                    i;ol;oi;^;nti;i=~i;ol;oi;     i;Ke;}S$
164 :                    ;ol;oi;^;nti;i=~i;ol;oi;^     ;Ke;}S$
165 :                    ol;oi;^;nti;i=~i;ol;oi;^;     Ke;}S$
166 :K->oV;K             ol;oi;^;nti;i=~i;ol;oi;^;     Ke;}S$
167 : SAVESTATE:           20
167 :                    ol;oi;^;nti;i=~i;ol;oi;^;     oV;Ke;}S$
168 :                    l;oi;^;nti;i=~i;ol;oi;^;i     V;Ke;}S$
169 :V->l                l;oi;^;nti;i=~i;ol;oi;^;i     V;Ke;}S$
170 : SAVESTATE:           21
170 :                    l;oi;^;nti;i=~i;ol;oi;^;i     l;Ke;}S$
171 :                    ;oi;^;nti;i=~i;ol;oi;^;i=     ;Ke;}S$
172 :                    oi;^;nti;i=~i;ol;oi;^;i=i     Ke;}S$
173 :K->oV;K             oi;^;nti;i=~i;ol;oi;^;i=i     Ke;}S$
174 : SAVESTATE:           22
174 :                    oi;^;nti;i=~i;ol;oi;^;i=i     oV;Ke;}S$
175 :                    i;^;nti;i=~i;ol;oi;^;i=iO     V;Ke;}S$
176 :V->i                i;^;nti;i=~i;ol;oi;^;i=iO     V;Ke;}S$
177 : SAVESTATE:           23
177 :                    i;^;nti;i=~i;ol;oi;^;i=iO     i;Ke;}S$
178 :                    ;^;nti;i=~i;ol;oi;^;i=iOi     ;Ke;}S$
179 :                    ^;nti;i=~i;ol;oi;^;i=iOi;     Ke;}S$
180 :K->^;K              ^;nti;i=~i;ol;oi;^;i=iOi;     Ke;}S$
181 : SAVESTATE:           24
181 :                    ^;nti;i=~i;ol;oi;^;i=iOi;     ^;Ke;}S$
182 :                    ;nti;i=~i;ol;oi;^;i=iOi;o     ;Ke;}S$
183 :                    nti;i=~i;ol;oi;^;i=iOi;ol     Ke;}S$
184 :K->nti=V;K          nti;i=~i;ol;oi;^;i=iOi;ol     Ke;}S$
185 : SAVESTATE:           25
185 :                    nti;i=~i;ol;oi;^;i=iOi;ol     nti=V;Ke;}S$
186 :                    ti;i=~i;ol;oi;^;i=iOi;ol;     ti=V;Ke;}S$
187 :                    i;i=~i;ol;oi;^;i=iOi;ol;o     i=V;Ke;}S$
188 :                    ;i=~i;ol;oi;^;i=iOi;ol;oi     =V;Ke;}S$
189 : TS_NOK/NS_NORULECHAIN
189 : RESSTATE
189 :                    nti;i=~i;ol;oi;^;i=iOi;ol     Ke;}S$
190 :K->nti;K            nti;i=~i;ol;oi;^;i=iOi;ol     Ke;}S$
191 : SAVESTATE:           25
191 :                    nti;i=~i;ol;oi;^;i=iOi;ol     nti;Ke;}S$
192 :                    ti;i=~i;ol;oi;^;i=iOi;ol;     ti;Ke;}S$
193 :                    i;i=~i;ol;oi;^;i=iOi;ol;o     i;Ke;}S$
194 :                    ;i=~i;ol;oi;^;i=iOi;ol;oi     ;Ke;}S$
```

```
195 :                    i=~i;ol;oi;^;i=iOi;ol;oi;     Ke;}S$
196 :K->i=W;K            i=~i;ol;oi;^;i=iOi;ol;oi;     Ke;}S$
197 : SAVESTATE:          26
197 :                    i=~i;ol;oi;^;i=iOi;ol;oi;     i=W;Ke;}S$
198 :                    =~i;ol;oi;^;i=iOi;ol;oi;^     =W;Ke;}S$
199 :                    ~i;ol;oi;^;i=iOi;ol;oi;^;     W;Ke;}S$
200 :W->~l               ~i;ol;oi;^;i=iOi;ol;oi;^;     W;Ke;}S$
201 : SAVESTATE:          27
201 :                    ~i;ol;oi;^;i=iOi;ol;oi;^;     ~l;Ke;}S$
202 :                    i;ol;oi;^;i=iOi;ol;oi;^;i     l;Ke;}S$
203 : TS_NOK/NS_NORULECHAIN
203 : RESSTATE
203 :                    ~i;ol;oi;^;i=iOi;ol;oi;^;     W;Ke;}S$
204 :W->~i               ~i;ol;oi;^;i=iOi;ol;oi;^;     W;Ke;}S$
205 : SAVESTATE:          27
205 :                    ~i;ol;oi;^;i=iOi;ol;oi;^;     ~i;Ke;}S$
206 :                    i;ol;oi;^;i=iOi;ol;oi;^;i     i;Ke;}S$
207 :                    ;ol;oi;^;i=iOi;ol;oi;^;i=     ;Ke;}S$
208 :                    ol;oi;^;i=iOi;ol;oi;^;i=i     Ke;}S$
209 :K->oV;K             ol;oi;^;i=iOi;ol;oi;^;i=i     Ke;}S$
210 : SAVESTATE:          28
210 :                    ol;oi;^;i=iOi;ol;oi;^;i=i     oV;Ke;}S$
211 :                    l;oi;^;i=iOi;ol;oi;^;i=iA     V;Ke;}S$
212 :V->l                l;oi;^;i=iOi;ol;oi;^;i=iA     V;Ke;}S$
213 : SAVESTATE:          29
213 :                    l;oi;^;i=iOi;ol;oi;^;i=iA     l;Ke;}S$
214 :                    ;oi;^;i=iOi;ol;oi;^;i=iAi     ;Ke;}S$
215 :                    oi;^;i=iOi;ol;oi;^;i=iAi;     Ke;}S$
216 :K->oV;K             oi;^;i=iOi;ol;oi;^;i=iAi;     Ke;}S$
217 : SAVESTATE:          30
217 :                    oi;^;i=iOi;ol;oi;^;i=iAi;     oV;Ke;}S$
218 :                    i;^;i=iOi;ol;oi;^;i=iAi;o     V;Ke;}S$
219 :V->i                i;^;i=iOi;ol;oi;^;i=iAi;o     V;Ke;}S$
220 : SAVESTATE:          31
220 :                    i;^;i=iOi;ol;oi;^;i=iAi;o     i;Ke;}S$
221 :                    ;^;i=iOi;ol;oi;^;i=iAi;ol     ;Ke;}S$
222 :                    ^;i=iOi;ol;oi;^;i=iAi;ol;     Ke;}S$
223 :K->^;K              ^;i=iOi;ol;oi;^;i=iAi;ol;     Ke;}S$
224 : SAVESTATE:          32
224 :                    ^;i=iOi;ol;oi;^;i=iAi;ol;     ^;Ke;}S$
225 :                    ;i=iOi;ol;oi;^;i=iAi;ol;o     ;Ke;}S$
226 :                    i=iOi;ol;oi;^;i=iAi;ol;oi     Ke;}S$
227 :K->i=W;K             i=iOi;ol;oi;^;i=iAi;ol;oi     Ke;}S$
228 : SAVESTATE:          33
228 :                    i=iOi;ol;oi;^;i=iAi;ol;oi     i=W;Ke;}S$
229 :                    =iOi;ol;oi;^;i=iAi;ol;oi;     =W;Ke;}S$
230 :                    iOi;ol;oi;^;i=iAi;ol;oi;^     W;Ke;}S$
231 :W->i                iOi;ol;oi;^;i=iAi;ol;oi;^     W;Ke;}S$
232 : SAVESTATE:          34
232 :                    iOi;ol;oi;^;i=iAi;ol;oi;^     i;Ke;}S$
233 :                    Oi;ol;oi;^;i=iAi;ol;oi;^;     ;Ke;}S$
234 : TS_NOK/NS_NORULECHAIN
234 : RESSTATE
234 :                    iOi;ol;oi;^;i=iAi;ol;oi;^     W;Ke;}S$
235 :W->iF               iOi;ol;oi;^;i=iAi;ol;oi;^     W;Ke;}S$
236 : SAVESTATE:          34
236 :                    iOi;ol;oi;^;i=iAi;ol;oi;^     iF;Ke;}S$
237 :                    Oi;ol;oi;^;i=iAi;ol;oi;^;     F;Ke;}S$
238 : TNS_NORULECHAIN/NS_NORULE
238 : RESSTATE
238 :                    iOi;ol;oi;^;i=iAi;ol;oi;^     W;Ke;}S$
239 :W->iQW              iOi;ol;oi;^;i=iAi;ol;oi;^     W;Ke;}S$
240 : SAVESTATE:          34
240 :                    iOi;ol;oi;^;i=iAi;ol;oi;^     iQW;Ke;}S$
241 :                    Oi;ol;oi;^;i=iAi;ol;oi;^;     QW;Ke;}S$
242 :Q->O                Oi;ol;oi;^;i=iAi;ol;oi;^;     QW;Ke;}S$
243 : SAVESTATE:          35
243 :                    Oi;ol;oi;^;i=iAi;ol;oi;^;     OW;Ke;}S$
244 :                    i;ol;oi;^;i=iAi;ol;oi;^;i     W;Ke;}S$
245 :W->i                i;ol;oi;^;i=iAi;ol;oi;^;i     W;Ke;}S$
246 : SAVESTATE:          36
```

```
247 :                    ;ol;oi;^;i=iAi;ol;oi;^;i=     ;Ke;}S$
248 :                    ol;oi;^;i=iAi;ol;oi;^;i=(     Ke;}S$
249 :K->oV;K             ol;oi;^;i=iAi;ol;oi;^;i=(     Ke;}S$
250 : SAVESTATE:          37
250 :                    ol;oi;^;i=iAi;ol;oi;^;i=(     oV;Ke;}S$
251 :                    l;oi;^;i=iAi;ol;oi;^;i=(i     V;Ke;}S$
252 :V->l                l;oi;^;i=iAi;ol;oi;^;i=(i     V;Ke;}S$
253 : SAVESTATE:          38
253 :                    l;oi;^;i=iAi;ol;oi;^;i=(i     l;Ke;}S$
254 :                    ;oi;^;i=iAi;ol;oi;^;i=(i+     ;Ke;}S$
255 :                    oi;^;i=iAi;ol;oi;^;i=(i+i     Ke;}S$
256 :K->oV;K             oi;^;i=iAi;ol;oi;^;i=(i+i     Ke;}S$
257 : SAVESTATE:          39
257 :                    oi;^;i=iAi;ol;oi;^;i=(i+i     oV;Ke;}S$
258 :                    i;^;i=iAi;ol;oi;^;i=(i+i)     V;Ke;}S$
259 :V->i                i;^;i=iAi;ol;oi;^;i=(i+i)     V;Ke;}S$
260 : SAVESTATE:          40
260 :                    i;^;i=iAi;ol;oi;^;i=(i+i)     i;Ke;}S$
261 :                    ;^;i=iAi;ol;oi;^;i=(i+i)*     ;Ke;}S$
262 :                    ^;i=iAi;ol;oi;^;i=(i+i)*(     Ke;}S$
263 :K->^;K              ^;i=iAi;ol;oi;^;i=(i+i)*(     Ke;}S$
264 : SAVESTATE:          41
264 :                    ^;i=iAi;ol;oi;^;i=(i+i)*(     ^;Ke;}S$
265 :                    ;i=iAi;ol;oi;^;i=(i+i)*(i     ;Ke;}S$
266 :                    i=iAi;ol;oi;^;i=(i+i)*(i+     Ke;}S$
267 :K->i=W;K           i=iAi;ol;oi;^;i=(i+i)*(i+     Ke;}S$
268 : SAVESTATE:          42
268 :                    i=iAi;ol;oi;^;i=(i+i)*(i+     i=W;Ke;}S$
269 :                    =iAi;ol;oi;^;i=(i+i)*(i+i     =W;Ke;}S$
270 :                    iAi;ol;oi;^;i=(i+i)*(i+i)     W;Ke;}S$
271 :W->i               iAi;ol;oi;^;i=(i+i)*(i+i)     W;Ke;}S$
272 : SAVESTATE:          43
272 :                    iAi;ol;oi;^;i=(i+i)*(i+i)     i;Ke;}S$
273 :                    Ai;ol;oi;^;i=(i+i)*(i+i)/     ;Ke;}S$
274 : TS_NOK/NS_NORULECHAIN
274 : RESSTATE
274 :                    iAi;ol;oi;^;i=(i+i)*(i+i)     W;Ke;}S$
275 :W->iF               iAi;ol;oi;^;i=(i+i)*(i+i)     W;Ke;}S$
276 : SAVESTATE:          43
276 :                    iAi;ol;oi;^;i=(i+i)*(i+i)     iF;Ke;}S$
277 :                    Ai;ol;oi;^;i=(i+i)*(i+i)/     F;Ke;}S$
278 : TNS_NORULECHAIN/NS_NORULE
278 : RESSTATE
278 :                    iAi;ol;oi;^;i=(i+i)*(i+i)     W;Ke;}S$
279 :W->iQW              iAi;ol;oi;^;i=(i+i)*(i+i)     W;Ke;}S$
280 : SAVESTATE:          43
280 :                    iAi;ol;oi;^;i=(i+i)*(i+i)     iQW;Ke;}S$
281 :                    Ai;ol;oi;^;i=(i+i)*(i+i)/     QW;Ke;}S$
282 :Q->A                Ai;ol;oi;^;i=(i+i)*(i+i)/     QW;Ke;}S$
283 : SAVESTATE:          44
283 :                    Ai;ol;oi;^;i=(i+i)*(i+i)/     AW;Ke;}S$
284 :                    i;ol;oi;^;i=(i+i)*(i+i)/l     W;Ke;}S$
285 :W->i                i;ol;oi;^;i=(i+i)*(i+i)/l     W;Ke;}S$
286 : SAVESTATE:          45
286 :                    i;ol;oi;^;i=(i+i)*(i+i)/l     i;Ke;}S$
287 :                    ;ol;oi;^;i=(i+i)*(i+i)/l;     ;Ke;}S$
288 :                    ol;oi;^;i=(i+i)*(i+i)/l;o     Ke;}S$
289 :K->oV;K             ol;oi;^;i=(i+i)*(i+i)/l;o     Ke;}S$
290 : SAVESTATE:          46
290 :                    ol;oi;^;i=(i+i)*(i+i)/l;o     oV;Ke;}S$
291 :                    l;oi;^;i=(i+i)*(i+i)/l;ol     V;Ke;}S$
292 :V->l                l;oi;^;i=(i+i)*(i+i)/l;ol     V;Ke;}S$
293 : SAVESTATE:          47
293 :                    l;oi;^;i=(i+i)*(i+i)/l;ol     l;Ke;}S$
294 :                    ;oi;^;i=(i+i)*(i+i)/l;ol;     ;Ke;}S$
295 :                    oi;^;i=(i+i)*(i+i)/l;ol;o     Ke;}S$
296 :K->oV;K             oi;^;i=(i+i)*(i+i)/l;ol;o     Ke;}S$
297 : SAVESTATE:          48
297 :                    oi;^;i=(i+i)*(i+i)/l;ol;o     oV;Ke;}S$
298 :                    i;^;i=(i+i)*(i+i)/l;ol;oi     V;Ke;}S$
299 :V->i                i;^;i=(i+i)*(i+i)/l;ol;oi     V;Ke;}S$
```

```
300 : SAVESTATE:          49
300 :                    i;^;i=(i+i)*(i+i)/l;ol;oi     i;Ke;}S$
301 :                    ;^;i=(i+i)*(i+i)/l;ol;oi;     ;Ke;}S$
302 :                    ^;i=(i+i)*(i+i)/l;ol;oi;^     Ke;}S$
303 :K->^;K              ^;i=(i+i)*(i+i)/l;ol;oi;^     Ke;}S$
304 : SAVESTATE:          50
304 :                    ^;i=(i+i)*(i+i)/l;ol;oi;^     ^;Ke;}S$
305 :                    ;i=(i+i)*(i+i)/l;ol;oi;^;     ;Ke;}S$
306 :                    i=(i+i)*(i+i)/l;ol;oi;^;e     Ke;}S$
307 :K->i=W;K            i=(i+i)*(i+i)/l;ol;oi;^;e     Ke;}S$
308 : SAVESTATE:          51
308 :                    i=(i+i)*(i+i)/l;ol;oi;^;e     i=W;Ke;}S$
309 :                    =(i+i)*(i+i)/l;ol;oi;^;e;     =W;Ke;}S$
310 :                    (i+i)*(i+i)/l;ol;oi;^;e;}     W;Ke;}S$
311 :W->(W)              (i+i)*(i+i)/l;ol;oi;^;e;}     W;Ke;}S$
312 : SAVESTATE:          52
312 :                    (i+i)*(i+i)/l;ol;oi;^;e;}     (W);Ke;}S$
313 :                    i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
314 :W->i                i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
315 : SAVESTATE:          53
315 :                    i+i)*(i+i)/l;ol;oi;^;e;}m     i);Ke;}S$
316 :                    +i)*(i+i)/l;ol;oi;^;e;}m{     );Ke;}S$
317 : TS_NOK/NS_NORULECHAIN
317 : RESSTATE
317 :                    i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
318 :W->iF               i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
319 : SAVESTATE:          53
319 :                    i+i)*(i+i)/l;ol;oi;^;e;}m     iF);Ke;}S$
320 :                    +i)*(i+i)/l;ol;oi;^;e;}m{     F);Ke;}S$
321 : TNS_NORULECHAIN/NS_NORULE
321 : RESSTATE
321 :                    i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
322 :W->iQW              i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
323 : SAVESTATE:          53
323 :                    i+i)*(i+i)/l;ol;oi;^;e;}m     iQW);Ke;}S$
324 :                    +i)*(i+i)/l;ol;oi;^;e;}m{     QW);Ke;}S$
325 :Q->+                +i)*(i+i)/l;ol;oi;^;e;}m{     QW);Ke;}S$
326 : SAVESTATE:          54
326 :                    +i)*(i+i)/l;ol;oi;^;e;}m{     +W);Ke;}S$
327 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
328 :W->i                i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
329 : SAVESTATE:          55
329 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     i);Ke;}S$
330 :                    )*(i+i)/l;ol;oi;^;e;}m{nt     );Ke;}S$
331 :                    *(i+i)/l;ol;oi;^;e;}m{nti     ;Ke;}S$
332 : TS_NOK/NS_NORULECHAIN
332 : RESSTATE
332 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
333 :W->iF               i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
334 : SAVESTATE:          55
334 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     iF);Ke;}S$
335 :                    )*(i+i)/l;ol;oi;^;e;}m{nt     F);Ke;}S$
336 : TNS_NORULECHAIN/NS_NORULE
336 : RESSTATE
336 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
337 :W->iQW              i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
338 : SAVESTATE:          55
338 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     iQW);Ke;}S$
339 :                    )*(i+i)/l;ol;oi;^;e;}m{nt     QW);Ke;}S$
340 : TNS_NORULECHAIN/NS_NORULE
340 : RESSTATE
340 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
341 :W->iFQW             i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
342 : SAVESTATE:          55
342 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     iFQW);Ke;}S$
343 :                    )*(i+i)/l;ol;oi;^;e;}m{nt     FQW);Ke;}S$
344 : TNS_NORULECHAIN/NS_NORULE
344 : RESSTATE
344 :                    i)*(i+i)/l;ol;oi;^;e;}m{n     W);Ke;}S$
345 : TNS_NORULECHAIN/NS_NORULE
345 : RESSTATE
```

```
345 : RESSTATE
345 :                       +i)*(i+i)/l;ol;oi;^;e;}m{     QW);Ke;}S$
346 : TNS_NORULECHAIN/NS_NORULE
346 : RESSTATE
346 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
347 :W->iFQW                i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
348 : SAVESTATE:             53
348 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     iFQW);Ke;}S$
349 :                       +i)*(i+i)/l;ol;oi;^;e;}m{     FQW);Ke;}S$
350 : TNS_NORULECHAIN/NS_NORULE
350 : RESSTATE
350 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     W);Ke;}S$
351 : TNS_NORULECHAIN/NS_NORULE
351 : RESSTATE
351 :                       (i+i)*(i+i)/l;ol;oi;^;e;}     W;Ke;}S$
352 :W->(W)QW               (i+i)*(i+i)/l;ol;oi;^;e;}     W;Ke;}S$
353 : SAVESTATE:             52
353 :                       (i+i)*(i+i)/l;ol;oi;^;e;}     (W)QW;Ke;}S$
354 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     W)QW;Ke;}S$
355 :W->i                   i+i)*(i+i)/l;ol;oi;^;e;}m     W)QW;Ke;}S$
356 : SAVESTATE:             53
356 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     i)QW;Ke;}S$
357 :                       +i)*(i+i)/l;ol;oi;^;e;}m{     )QW;Ke;}S$
358 : TS_NOK/NS_NORULECHAIN
358 : RESSTATE
358 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     W)QW;Ke;}S$
359 :W->iF                  i+i)*(i+i)/l;ol;oi;^;e;}m     W)QW;Ke;}S$
360 : SAVESTATE:             53
360 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     iF)QW;Ke;}S$
361 :                       +i)*(i+i)/l;ol;oi;^;e;}m{     F)QW;Ke;}S$
362 : TNS_NORULECHAIN/NS_NORULE
362 : RESSTATE
362 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     W)QW;Ke;}S$
363 :W->iQW                 i+i)*(i+i)/l;ol;oi;^;e;}m     W)QW;Ke;}S$
364 : SAVESTATE:             53
364 :                       i+i)*(i+i)/l;ol;oi;^;e;}m     iQW)QW;Ke;}S$
365 :                       +i)*(i+i)/l;ol;oi;^;e;}m{     QW)QW;Ke;}S$
366 :Q->+                   +i)*(i+i)/l;ol;oi;^;e;}m{     QW)QW;Ke;}S$
367 : SAVESTATE:             54
367 :                       +i)*(i+i)/l;ol;oi;^;e;}m{     +W)QW;Ke;}S$
368 :                       i)*(i+i)/l;ol;oi;^;e;}m{n     W)QW;Ke;}S$
369 :W->i                   i)*(i+i)/l;ol;oi;^;e;}m{n     W)QW;Ke;}S$
370 : SAVESTATE:             55
370 :                       i)*(i+i)/l;ol;oi;^;e;}m{n     i)QW;Ke;}S$
371 :                       )*(i+i)/l;ol;oi;^;e;}m{nt     )QW;Ke;}S$
372 :                       *(i+i)/l;ol;oi;^;e;}m{nti     QW;Ke;}S$
373 :Q->*                   *(i+i)/l;ol;oi;^;e;}m{nti     QW;Ke;}S$
374 : SAVESTATE:             56
374 :                       *(i+i)/l;ol;oi;^;e;}m{nti     *W;Ke;}S$
375 :                       (i+i)/l;ol;oi;^;e;}m{nti=     W;Ke;}S$
376 :W->(W)                 (i+i)/l;ol;oi;^;e;}m{nti=     W;Ke;}S$
377 : SAVESTATE:             57
377 :                       (i+i)/l;ol;oi;^;e;}m{nti=     (W);Ke;}S$
378 :                       i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
379 :W->i                   i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
380 : SAVESTATE:             58
380 :                       i+i)/l;ol;oi;^;e;}m{nti=l     i);Ke;}S$
381 :                       +i)/l;ol;oi;^;e;}m{nti=l;     );Ke;}S$
382 : TS_NOK/NS_NORULECHAIN
382 : RESSTATE
382 :                       i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
383 :W->iF                  i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
384 : SAVESTATE:             58
384 :                       i+i)/l;ol;oi;^;e;}m{nti=l     iF);Ke;}S$
385 :                       +i)/l;ol;oi;^;e;}m{nti=l;     F);Ke;}S$
386 : TNS_NORULECHAIN/NS_NORULE
386 : RESSTATE
386 :                       i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
387 :W->iQW                 i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
388 : SAVESTATE:             58
388 :                       i+i)/l;ol;oi;^;e;}m{nti=l     iQW);Ke;}S$
389 :                       +i)/l;ol;oi;^;e;}m{nti=l;     QW);Ke;}S$
390 :Q->+                   +i)/l;ol;oi;^;e;}m{nti=l;     QW);Ke;}S$
391 : SAVESTATE:             59
391 :                       +i)/l;ol;oi;^;e;}m{nti=l;     +W);Ke;}S$
392 :                       i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
393 :W->i                   i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
394 : SAVESTATE:             60
394 :                       i)/l;ol;oi;^;e;}m{nti=l;n     i);Ke;}S$
395 :                       )/l;ol;oi;^;e;}m{nti=l;nt     );Ke;}S$
396 :                       /l;ol;oi;^;e;}m{nti=l;nti     ;Ke;}S$
397 : TS_NOK/NS_NORULECHAIN
397 : RESSTATE
397 :                       i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
398 :W->iF                  i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
399 : SAVESTATE:             60
399 :                       i)/l;ol;oi;^;e;}m{nti=l;n     iF);Ke;}S$
400 :                       )/l;ol;oi;^;e;}m{nti=l;nt     F);Ke;}S$
401 : TNS_NORULECHAIN/NS_NORULE
401 : RESSTATE
401 :                       i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
402 :W->iQW                 i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
403 : SAVESTATE:             60
```

```
403 :                    i)/l;ol;oi;^;e;}m{nti=l;n     iQW);Ke;}S$
404 :                    )/l;ol;oi;^;e;}m{nti=l;nt     QW);Ke;}S$
405 : TNS_NORULECHAIN/NS_NORULE
405 : RESSTATE
405 :                    i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
406 :W->iFQW             i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
407 : SAVESTATE:          60
407 :                    i)/l;ol;oi;^;e;}m{nti=l;n     iFQW);Ke;}S$
408 :                    )/l;ol;oi;^;e;}m{nti=l;nt     FQW);Ke;}S$
409 : TNS_NORULECHAIN/NS_NORULE
409 : RESSTATE
409 :                    i)/l;ol;oi;^;e;}m{nti=l;n     W);Ke;}S$
410 : TNS_NORULECHAIN/NS_NORULE
410 : RESSTATE
410 :                    +i)/l;ol;oi;^;e;}m{nti=l;     QW);Ke;}S$
411 : TNS_NORULECHAIN/NS_NORULE
411 : RESSTATE
411 :                    i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
412 :W->iFQW             i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
413 : SAVESTATE:          58
413 :                    i+i)/l;ol;oi;^;e;}m{nti=l     iFQW);Ke;}S$
414 :                    +i)/l;ol;oi;^;e;}m{nti=l;     FQW);Ke;}S$
415 : TNS_NORULECHAIN/NS_NORULE
415 : RESSTATE
415 :                    i+i)/l;ol;oi;^;e;}m{nti=l     W);Ke;}S$
416 : TNS_NORULECHAIN/NS_NORULE
416 : RESSTATE
416 :                    (i+i)/l;ol;oi;^;e;}m{nti=     W;Ke;}S$
417 :W->(W)QW            (i+i)/l;ol;oi;^;e;}m{nti=     W;Ke;}S$
418 : SAVESTATE:          57
418 :                    (i+i)/l;ol;oi;^;e;}m{nti=     (W)QW;Ke;}S$
419 :                    i+i)/l;ol;oi;^;e;}m{nti=l     W)QW;Ke;}S$
420 :W->i                i+i)/l;ol;oi;^;e;}m{nti=l     W)QW;Ke;}S$
421 : SAVESTATE:          58
421 :                    i+i)/l;ol;oi;^;e;}m{nti=l     i)QW;Ke;}S$
422 :                    +i)/l;ol;oi;^;e;}m{nti=l;     )QW;Ke;}S$
423 : TS_NOK/NS_NORULECHAIN
423 : RESSTATE
423 :                    i+i)/l;ol;oi;^;e;}m{nti=l     W)QW;Ke;}S$
424 :W->iF               i+i)/l;ol;oi;^;e;}m{nti=l     W)QW;Ke;}S$
425 : SAVESTATE:          58
425 :                    i+i)/l;ol;oi;^;e;}m{nti=l     iF)QW;Ke;}S$
426 :                    +i)/l;ol;oi;^;e;}m{nti=l;     F)QW;Ke;}S$
427 : TNS_NORULECHAIN/NS_NORULE
427 : RESSTATE
427 :                    i+i)/l;ol;oi;^;e;}m{nti=l     W)QW;Ke;}S$
428 :W->iQW              i+i)/l;ol;oi;^;e;}m{nti=l     W)QW;Ke;}S$
429 : SAVESTATE:          58
429 :                    i+i)/l;ol;oi;^;e;}m{nti=l     iQW)QW;Ke;}S$
430 :                    +i)/l;ol;oi;^;e;}m{nti=l;     QW)QW;Ke;}S$
431 :Q->+                +i)/l;ol;oi;^;e;}m{nti=l;     QW)QW;Ke;}S$
432 : SAVESTATE:          59
432 :                    +i)/l;ol;oi;^;e;}m{nti=l;     +W)QW;Ke;}S$
433 :                    i)/l;ol;oi;^;e;}m{nti=l;n     W)QW;Ke;}S$
434 :W->i                i)/l;ol;oi;^;e;}m{nti=l;n     W)QW;Ke;}S$
435 : SAVESTATE:          60
435 :                    i)/l;ol;oi;^;e;}m{nti=l;n     i)QW;Ke;}S$
436 :                    )/l;ol;oi;^;e;}m{nti=l;nt     )QW;Ke;}S$
437 :                    /l;ol;oi;^;e;}m{nti=l;nti     QW;Ke;}S$
438 :Q->/                /l;ol;oi;^;e;}m{nti=l;nti     QW;Ke;}S$
439 : SAVESTATE:          61
439 :                    /l;ol;oi;^;e;}m{nti=l;nti     /W;Ke;}S$
440 :                    l;ol;oi;^;e;}m{nti=l;nti=     W;Ke;}S$
441 :W->l                l;ol;oi;^;e;}m{nti=l;nti=     W;Ke;}S$
442 : SAVESTATE:          62
442 :                    l;ol;oi;^;e;}m{nti=l;nti=     l;Ke;}S$
443 :                    ;ol;oi;^;e;}m{nti=l;nti=l     ;Ke;}S$
444 :                    ol;oi;^;e;}m{nti=l;nti=l;     Ke;}S$
445 :K->oV;K              ol;oi;^;e;}m{nti=l;nti=l;     Ke;}S$
446 : SAVESTATE:          63
446 :                    ol;oi;^;e;}m{nti=l;nti=l;     oV;Ke;}S$
447 :                    l;oi;^;e;}m{nti=l;nti=l;i     V;Ke;}S$
448 :V->l                l;oi;^;e;}m{nti=l;nti=l;i     V;Ke;}S$
449 : SAVESTATE:          64
449 :                    l;oi;^;e;}m{nti=l;nti=l;i     l;Ke;}S$
450 :                    ;oi;^;e;}m{nti=l;nti=l;i(     ;Ke;}S$
451 :                    oi;^;e;}m{nti=l;nti=l;i(i     Ke;}S$
452 :K->oV;K              oi;^;e;}m{nti=l;nti=l;i(i     Ke;}S$
453 : SAVESTATE:          65
453 :                    oi;^;e;}m{nti=l;nti=l;i(i     oV;Ke;}S$
454 :                    i;^;e;}m{nti=l;nti=l;i(i,     V;Ke;}S$
455 :V->i                i;^;e;}m{nti=l;nti=l;i(i,     V;Ke;}S$
456 : SAVESTATE:          66
456 :                    i;^;e;}m{nti=l;nti=l;i(i,     i;Ke;}S$
457 :                    ;^;e;}m{nti=l;nti=l;i(i,i     ;Ke;}S$
458 :                    ^;e;}m{nti=l;nti=l;i(i,i)     Ke;}S$
459 :K->^;K              ^;e;}m{nti=l;nti=l;i(i,i)     Ke;}S$
460 : SAVESTATE:          67
460 :                    ^;e;}m{nti=l;nti=l;i(i,i)     ^;Ke;}S$
461 :                    ;e;}m{nti=l;nti=l;i(i,i);     ;Ke;}S$
462 :                    e;}m{nti=l;nti=l;i(i,i);n     Ke;}S$
463 : TNS_NORULECHAIN/NS_NORULE
```

```
463 : RESSTATE
463 :                       ^;e;}m{nti=l;nti=l;i(i,i)      Ke;}S$
464 :K->^;                  ^;e;}m{nti=l;nti=l;i(i,i)      Ke;}S$
465 : SAVESTATE:             67
465 :                       ^;e;}m{nti=l;nti=l;i(i,i)      ^;e;}S$
466 :                       ;e;}m{nti=l;nti=l;i(i,i);      ;e;}S$
467 :                       e;}m{nti=l;nti=l;i(i,i);n      e;}S$
468 :                       ;}m{nti=l;nti=l;i(i,i);nt      ;}S$
469 :                       }m{nti=l;nti=l;i(i,i);nti      }S$
470 :                       m{nti=l;nti=l;i(i,i);nti;      S$
471 :S->m{K}                m{nti=l;nti=l;i(i,i);nti;      S$
472 : SAVESTATE:             68
472 :                       m{nti=l;nti=l;i(i,i);nti;      m{K}$
473 :                       {nti=l;nti=l;i(i,i);nti;n      {K}$
474 :                       nti=l;nti=l;i(i,i);nti;nt      K}$
475 :K->nti=V;K             nti=l;nti=l;i(i,i);nti;nt      K}$
476 : SAVESTATE:             69
476 :                       nti=l;nti=l;i(i,i);nti;nt      nti=V;K}$
477 :                       ti=l;nti=l;i(i,i);nti;nti      ti=V;K}$
478 :                       i=l;nti=l;i(i,i);nti;nti=      i=V;K}$
479 :                       =l;nti=l;i(i,i);nti;nti=l      =V;K}$
480 :                       l;nti=l;i(i,i);nti;nti=l;      V;K}$
481 :V->l                   l;nti=l;i(i,i);nti;nti=l;      V;K}$
482 : SAVESTATE:             70
482 :                       l;nti=l;i(i,i);nti;nti=l;      l;K}$
483 :                       ;nti=l;i(i,i);nti;nti=l;i      ;K}$
484 :                       nti=l;i(i,i);nti;nti=l;i=      K}$
485 :K->nti=V;K             nti=l;i(i,i);nti;nti=l;i=      K}$
486 : SAVESTATE:             71
486 :                       nti=l;i(i,i);nti;nti=l;i=      nti=V;K}$
487 :                       ti=l;i(i,i);nti;nti=l;i=i      ti=V;K}$
488 :                       i=l;i(i,i);nti;nti=l;i=i(      i=V;K}$
489 :                       =l;i(i,i);nti;nti=l;i=i(i      =V;K}$
490 :                       l;i(i,i);nti;nti=l;i=i(i)      V;K}$
491 :V->l                   l;i(i,i);nti;nti=l;i=i(i)      V;K}$
492 : SAVESTATE:             72
492 :                       l;i(i,i);nti;nti=l;i=i(i)      l;K}$
493 :                       ;i(i,i);nti;nti=l;i=i(i);      ;K}$
494 :                       i(i,i);nti;nti=l;i=i(i);o      K}$
495 :K->i=W;K               i(i,i);nti;nti=l;i=i(i);o      K}$
496 : SAVESTATE:             73
496 :                       i(i,i);nti;nti=l;i=i(i);o      i=W;K}$
497 :                       (i,i);nti;nti=l;i=i(i);ol      =W;K}$
498 : TS_NOK/NS_NORULECHAIN
498 : RESSTATE
498 :                       i(i,i);nti;nti=l;i=i(i);o      K}$
499 :K->iF;K               i(i,i);nti;nti=l;i=i(i);o      K}$
500 : SAVESTATE:             73
500 :                       i(i,i);nti;nti=l;i=i(i);o      iF;K}$
501 :                       (i,i);nti;nti=l;i=i(i);ol      F;K}$
502 :F->(N)                 (i,i);nti;nti=l;i=i(i);ol      F;K}$
503 : SAVESTATE:             74
503 :                       (i,i);nti;nti=l;i=i(i);ol      (N);K}$
504 :                       i,i);nti;nti=l;i=i(i);ol;      N);K}$
505 :N->i                   i,i);nti;nti=l;i=i(i);ol;      N);K}$
506 : SAVESTATE:             75
506 :                       i,i);nti;nti=l;i=i(i);ol;      i);K}$
507 :                       ,i);nti;nti=l;i=i(i);ol;o      );K}$
508 : TS_NOK/NS_NORULECHAIN
508 : RESSTATE
508 :                       i,i);nti;nti=l;i=i(i);ol;      N);K}$
509 :N->i,N                 i,i);nti;nti=l;i=i(i);ol;      N);K}$
510 : SAVESTATE:             75
510 :                       i,i);nti;nti=l;i=i(i);ol;      i,N);K}$
511 :                       ,i);nti;nti=l;i=i(i);ol;o      ,N);K}$
512 :                       i);nti;nti=l;i=i(i);ol;oi      N);K}$
513 :N->i                   i);nti;nti=l;i=i(i);ol;oi      N);K}$
514 : SAVESTATE:             76
514 :                       i);nti;nti=l;i=i(i);ol;oi      i);K}$
515 :                       );nti;nti=l;i=i(i);ol;oi;      );K}$
516 :                       ;nti;nti=l;i=i(i);ol;oi;^      ;K}$
517 :                       nti;nti=l;i=i(i);ol;oi;^;      K}$
518 :K->nti=V;K             nti;nti=l;i=i(i);ol;oi;^;      K}$
519 : SAVESTATE:             77
519 :                       nti;nti=l;i=i(i);ol;oi;^;      nti=V;K}$
520 :                       ti;nti=l;i=i(i);ol;oi;^;n      ti=V;K}$
521 :                       i;nti=l;i=i(i);ol;oi;^;nt      i=V;K}$
522 :                       ;nti=l;i=i(i);ol;oi;^;nti      =V;K}$
523 : TS_NOK/NS_NORULECHAIN
523 : RESSTATE
523 :                       nti;nti=l;i=i(i);ol;oi;^;      K}$
524 :K->nti;K               nti;nti=l;i=i(i);ol;oi;^;      K}$
525 : SAVESTATE:             77
525 :                       nti;nti=l;i=i(i);ol;oi;^;      nti;K}$
526 :                       ti;nti=l;i=i(i);ol;oi;^;n      ti;K}$
527 :                       i;nti=l;i=i(i);ol;oi;^;nt      i;K}$
528 :                       ;nti=l;i=i(i);ol;oi;^;nti      ;K}$
529 :                       nti=l;i=i(i);ol;oi;^;nti=      K}$
530 :K->nti=V;K             nti=l;i=i(i);ol;oi;^;nti=      K}$
531 : SAVESTATE:             78
531 :                       nti=l;i=i(i);ol;oi;^;nti=      nti=V;K}$
532 :                       ti=l;i=i(i);ol;oi;^;nti=l      ti=V;K}$
533 :                       i=l;i=i(i);ol;oi;^;nti=l;      i=V;K}$
```

```
532 :                    ti=l;i=i(i);ol;oi;^;nti=l     ti=V;K}$
533 :                    i=l;i=i(i);ol;oi;^;nti=l;     i=V;K}$
534 :                    =l;i=i(i);ol;oi;^;nti=l;n     =V;K}$
535 :                    l;i=i(i);ol;oi;^;nti=l;nt     V;K}$
536 :V->l                l;i=i(i);ol;oi;^;nti=l;nt     V;K}$
537 : SAVESTATE:          79
537 :                    l;i=i(i);ol;oi;^;nti=l;nt     l;K}$
538 :                    ;i=i(i);ol;oi;^;nti=l;nti     ;K}$
539 :                    i=i(i);ol;oi;^;nti=l;nti;     K}$
540 :K->i=W;K            i=i(i);ol;oi;^;nti=l;nti;     K}$
541 : SAVESTATE:          80
541 :                    i=i(i);ol;oi;^;nti=l;nti;     i=W;K}$
542 :                    =i(i);ol;oi;^;nti=l;nti;i     =W;K}$
543 :                    i(i);ol;oi;^;nti=l;nti;i=     W;K}$
544 :W->i                i(i);ol;oi;^;nti=l;nti;i=     W;K}$
545 : SAVESTATE:          81
545 :                    i(i);ol;oi;^;nti=l;nti;i=     i;K}$
546 :                    (i);ol;oi;^;nti=l;nti;i=i     ;K}$
547 : TS_NOK/NS_NORULECHAIN
547 : RESSTATE
547 :                    i(i);ol;oi;^;nti=l;nti;i=     W;K}$
548 :W->iF               i(i);ol;oi;^;nti=l;nti;i=     W;K}$
549 : SAVESTATE:          81
549 :                    i(i);ol;oi;^;nti=l;nti;i=     iF;K}$
550 :                    (i);ol;oi;^;nti=l;nti;i=i     F;K}$
551 :F->(N)              (i);ol;oi;^;nti=l;nti;i=i     F;K}$
552 : SAVESTATE:          82
552 :                    (i);ol;oi;^;nti=l;nti;i=i     (N);K}$
553 :                    i);ol;oi;^;nti=l;nti;i=i(     N);K}$
554 :N->i                i);ol;oi;^;nti=l;nti;i=i(     N);K}$
555 : SAVESTATE:          83
555 :                    i);ol;oi;^;nti=l;nti;i=i(     i);K}$
556 :                    );ol;oi;^;nti=l;nti;i=i(i     );K}$
557 :                    ;ol;oi;^;nti=l;nti;i=i(i,     ;K}$
558 :                    ol;oi;^;nti=l;nti;i=i(i,i     K}$
559 :K->oV;K            ol;oi;^;nti=l;nti;i=i(i,i     K}$
560 : SAVESTATE:          84
560 :                    ol;oi;^;nti=l;nti;i=i(i,i     oV;K}$
561 :                    l;oi;^;nti=l;nti;i=i(i,i)     V;K}$
562 :V->l                l;oi;^;nti=l;nti;i=i(i,i)     V;K}$
563 : SAVESTATE:          85
563 :                    l;oi;^;nti=l;nti;i=i(i,i)     l;K}$
564 :                    ;oi;^;nti=l;nti;i=i(i,i);     ;K}$
565 :                    oi;^;nti=l;nti;i=i(i,i);o     K}$
566 :K->oV;K            oi;^;nti=l;nti;i=i(i,i);o     K}$
567 : SAVESTATE:          86
567 :                    oi;^;nti=l;nti;i=i(i,i);o     oV;K}$
568 :                    i;^;nti=l;nti;i=i(i,i);oi     V;K}$
569 :V->i                i;^;nti=l;nti;i=i(i,i);oi     V;K}$
570 : SAVESTATE:          87
570 :                    i;^;nti=l;nti;i=i(i,i);oi     i;K}$
571 :                    ;^;nti=l;nti;i=i(i,i);oi;     ;K}$
572 :                    ^;nti=l;nti;i=i(i,i);oi;^     K}$
573 :K->^;K              ^;nti=l;nti;i=i(i,i);oi;^     K}$
574 : SAVESTATE:          88
574 :                    ^;nti=l;nti;i=i(i,i);oi;^     ^;K}$
575 :                    ;nti=l;nti;i=i(i,i);oi;^;     ;K}$
576 :                    nti=l;nti;i=i(i,i);oi;^;?     K}$
577 :K->nti=V;K          nti=l;nti;i=i(i,i);oi;^;?     K}$
578 : SAVESTATE:          89
578 :                    nti=l;nti;i=i(i,i);oi;^;?     nti=V;K}$
579 :                    ti=l;nti;i=i(i,i);oi;^;?i     ti=V;K}$
580 :                    i=l;nti;i=i(i,i);oi;^;?i>     i=V;K}$
581 :                    =l;nti;i=i(i,i);oi;^;?i>i     =V;K}$
582 :                    l;nti;i=i(i,i);oi;^;?i>i#     V;K}$
583 :V->l                l;nti;i=i(i,i);oi;^;?i>i#     V;K}$
584 : SAVESTATE:          90
584 :                    l;nti;i=i(i,i);oi;^;?i>i#     l;K}$
585 :                    ;nti;i=i(i,i);oi;^;?i>i#r     ;K}$
586 :                    nti;i=i(i,i);oi;^;?i>i#r{     K}$
587 :K->nti=V;K          nti;i=i(i,i);oi;^;?i>i#r{     K}$
588 : SAVESTATE:          91
588 :                    nti;i=i(i,i);oi;^;?i>i#r{     nti=V;K}$
589 :                    ti;i=i(i,i);oi;^;?i>i#r{o     ti=V;K}$
590 :                    i;i=i(i,i);oi;^;?i>i#r{ol     i=V;K}$
591 :                    ;i=i(i,i);oi;^;?i>i#r{ol;     =V;K}$
592 : TS_NOK/NS_NORULECHAIN
592 : RESSTATE
592 :                    nti;i=i(i,i);oi;^;?i>i#r{     K}$
593 :K->nti;K            nti;i=i(i,i);oi;^;?i>i#r{     K}$
594 : SAVESTATE:          91
594 :                    nti;i=i(i,i);oi;^;?i>i#r{     nti;K}$
595 :                    ti;i=i(i,i);oi;^;?i>i#r{o     ti;K}$
596 :                    i;i=i(i,i);oi;^;?i>i#r{ol     i;K}$
597 :                    ;i=i(i,i);oi;^;?i>i#r{ol;     ;K}$
598 :                    i=i(i,i);oi;^;?i>i#r{ol;}     K}$
599 :K->i=W;K            i=i(i,i);oi;^;?i>i#r{ol;}     K}$
600 : SAVESTATE:          92
600 :                    i=i(i,i);oi;^;?i>i#r{ol;}     i=W;K}$
601 :                    =i(i,i);oi;^;?i>i#r{ol;}w     =W;K}$
602 :                    i(i,i);oi;^;?i>i#r{ol;}w{     W;K}$
603 :W->i                i(i,i);oi;^;?i>i#r{ol;}w{     W;K}$
```

```
604 : SAVESTATE:           93
604 :                      i(i,i);oi;^;?i>i#r{ol;}w{      i;K}$
605 :                      (i,i);oi;^;?i>i#r{ol;}w{o      ;K}$
606 : TS_NOK/NS_NORULECHAIN
606 : RESSTATE
606 :                      i(i,i);oi;^;?i>i#r{ol;}w{      W;K}$
607 :W->iF                 i(i,i);oi;^;?i>i#r{ol;}w{      W;K}$
608 : SAVESTATE:           93
608 :                      i(i,i);oi;^;?i>i#r{ol;}w{      iF;K}$
609 :                      (i,i);oi;^;?i>i#r{ol;}w{o      F;K}$
610 :F->(N)                (i,i);oi;^;?i>i#r{ol;}w{o      F;K}$
611 : SAVESTATE:           94
611 :                      (i,i);oi;^;?i>i#r{ol;}w{o      (N);K}$
612 :                      i,i);oi;^;?i>i#r{ol;}w{ol      N);K}$
613 :N->i                  i,i);oi;^;?i>i#r{ol;}w{ol      N);K}$
614 : SAVESTATE:           95
614 :                      i,i);oi;^;?i>i#r{ol;}w{ol      i);K}$
615 :                      ,i);oi;^;?i>i#r{ol;}w{ol;      );K}$
616 : TS_NOK/NS_NORULECHAIN
616 : RESSTATE
616 :                      i,i);oi;^;?i>i#r{ol;}w{ol      N);K}$
617 :N->i,N                i,i);oi;^;?i>i#r{ol;}w{ol      N);K}$
618 : SAVESTATE:           95
618 :                      i,i);oi;^;?i>i#r{ol;}w{ol      i,N);K}$
619 :                      ,i);oi;^;?i>i#r{ol;}w{ol;      ,N);K}$
620 :                      i);oi;^;?i>i#r{ol;}w{ol;}      N);K}$
621 :N->i                  i);oi;^;?i>i#r{ol;}w{ol;}      N);K}$
622 : SAVESTATE:           96
622 :                      i);oi;^;?i>i#r{ol;}w{ol;}      i);K}$
623 :                      );oi;^;?i>i#r{ol;}w{ol;}#      );K}$
624 :                      ;oi;^;?i>i#r{ol;}w{ol;}#^      ;K}$
625 :                      oi;^;?i>i#r{ol;}w{ol;}#^;      K}$
626 :K->oV;K               oi;^;?i>i#r{ol;}w{ol;}#^;      K}$
627 : SAVESTATE:           97
627 :                      oi;^;?i>i#r{ol;}w{ol;}#^;      oV;K}$
628 :                      i;^;?i>i#r{ol;}w{ol;}#^;n      V;K}$
629 :V->i                  i;^;?i>i#r{ol;}w{ol;}#^;n      V;K}$
630 : SAVESTATE:           98
630 :                      i;^;?i>i#r{ol;}w{ol;}#^;n      i;K}$
631 :                      ;^;?i>i#r{ol;}w{ol;}#^;nt      ;K}$
632 :                      ^;?i>i#r{ol;}w{ol;}#^;nti      K}$
633 :K->^;K                ^;?i>i#r{ol;}w{ol;}#^;nti      K}$
634 : SAVESTATE:           99
634 :                      ^;?i>i#r{ol;}w{ol;}#^;nti      ^;K}$
635 :                      ;?i>i#r{ol;}w{ol;}#^;nti;      ;K}$
636 :                      ?i>i#r{ol;}w{ol;}#^;nti;n      K}$
637 :K->?Z#RK              ?i>i#r{ol;}w{ol;}#^;nti;n      K}$
638 : SAVESTATE:           100
638 :                      ?i>i#r{ol;}w{ol;}#^;nti;n      ?Z#RK}$
639 :                      i>i#r{ol;}w{ol;}#^;nti;nt      Z#RK}$
640 :Z->iLi                i>i#r{ol;}w{ol;}#^;nti;nt      Z#RK}$
641 : SAVESTATE:           101
641 :                      i>i#r{ol;}w{ol;}#^;nti;nt      iLi#RK}$
642 :                      >i#r{ol;}w{ol;}#^;nti;nti      Li#RK}$
643 :L->>                  >i#r{ol;}w{ol;}#^;nti;nti      Li#RK}$
644 : SAVESTATE:           102
644 :                      >i#r{ol;}w{ol;}#^;nti;nti      >i#RK}$
645 :                      i#r{ol;}w{ol;}#^;nti;nti=      i#RK}$
646 :                      #r{ol;}w{ol;}#^;nti;nti=l      #RK}$
647 :                      r{ol;}w{ol;}#^;nti;nti=l;      RK}$
648 :R->rY#                r{ol;}w{ol;}#^;nti;nti=l;      RK}$
649 : SAVESTATE:           103
649 :                      r{ol;}w{ol;}#^;nti;nti=l;      rY#K}$
650 :                      {ol;}w{ol;}#^;nti;nti=l;i      Y#K}$
651 :Y->{X}                {ol;}w{ol;}#^;nti;nti=l;i      Y#K}$
652 : SAVESTATE:           104
652 :                      {ol;}w{ol;}#^;nti;nti=l;i      {X}#K}$
653 :                      ol;}w{ol;}#^;nti;nti=l;i=      X}#K}$
654 :X->oV;X               ol;}w{ol;}#^;nti;nti=l;i=      X}#K}$
655 : SAVESTATE:           105
655 :                      ol;}w{ol;}#^;nti;nti=l;i=      oV;X}#K}$
656 :                      l;}w{ol;}#^;nti;nti=l;i=i      V;X}#K}$
657 :V->l                  l;}w{ol;}#^;nti;nti=l;i=i      V;X}#K}$
658 : SAVESTATE:           106
658 :                      l;}w{ol;}#^;nti;nti=l;i=i      l;X}#K}$
659 :                      ;}w{ol;}#^;nti;nti=l;i=i(      ;X}#K}$
660 :                      }w{ol;}#^;nti;nti=l;i=i(i      X}#K}$
661 : TNS_NORULECHAIN/NS_NORULE
661 : RESSTATE
661 :                      l;}w{ol;}#^;nti;nti=l;i=i      V;X}#K}$
662 : TNS_NORULECHAIN/NS_NORULE
662 : RESSTATE
662 :                      ol;}w{ol;}#^;nti;nti=l;i=      X}#K}$
663 :X->oV;                ol;}w{ol;}#^;nti;nti=l;i=      X}#K}$
664 : SAVESTATE:           105
664 :                      ol;}w{ol;}#^;nti;nti=l;i=      oV;}#K}$
665 :                      l;}w{ol;}#^;nti;nti=l;i=i      V;}#K}$
666 :V->l                  l;}w{ol;}#^;nti;nti=l;i=i      V;}#K}$
667 : SAVESTATE:           106
667 :                      l;}w{ol;}#^;nti;nti=l;i=i      l;}#K}$
668 :                      ;}w{ol;}#^;nti;nti=l;i=i(      ;}#K}$
669 :                      }w{ol;}#^;nti;nti=l;i=i(i      }#K}$
```

```
670 :                    w{ol;}#^;nti;nti=l;i=i(i)     #K}$
671 : TS_NOK/NS_NORULECHAIN
671 : RESSTATE
671 :                    l;}w{ol;}#^;nti;nti=l;i=i     V;}#K}$
672 : TNS_NORULECHAIN/NS_NORULE
672 : RESSTATE
672 :                    ol;}w{ol;}#^;nti;nti=l;i=     X}#K}$
673 : TNS_NORULECHAIN/NS_NORULE
673 : RESSTATE
673 :                    {ol;}w{ol;}#^;nti;nti=l;i     Y#K}$
674 : TNS_NORULECHAIN/NS_NORULE
674 : RESSTATE
674 :                    r{ol;}w{ol;}#^;nti;nti=l;     RK}$
675 :R->rYwY#            r{ol;}w{ol;}#^;nti;nti=l;     RK}$
676 : SAVESTATE:           103
676 :                    r{ol;}w{ol;}#^;nti;nti=l;     rYwY#K}$
677 :                    {ol;}w{ol;}#^;nti;nti=l;i     YwY#K}$
678 :Y->{X}              {ol;}w{ol;}#^;nti;nti=l;i     YwY#K}$
679 : SAVESTATE:           104
679 :                    {ol;}w{ol;}#^;nti;nti=l;i     {X}wY#K}$
680 :                    ol;}w{ol;}#^;nti;nti=l;i=     X}wY#K}$
681 :X->oV;X             ol;}w{ol;}#^;nti;nti=l;i=     X}wY#K}$
682 : SAVESTATE:           105
682 :                    ol;}w{ol;}#^;nti;nti=l;i=     oV;X}wY#K}$
683 :                    l;}w{ol;}#^;nti;nti=l;i=i     V;X}wY#K}$
684 :V->l                l;}w{ol;}#^;nti;nti=l;i=i     V;X}wY#K}$
685 : SAVESTATE:           106
685 :                    l;}w{ol;}#^;nti;nti=l;i=i     l;X}wY#K}$
686 :                    ;}w{ol;}#^;nti;nti=l;i=i(     ;X}wY#K}$
687 :                    }w{ol;}#^;nti;nti=l;i=i(i     X}wY#K}$
688 : TNS_NORULECHAIN/NS_NORULE
688 : RESSTATE
688 :                    l;}w{ol;}#^;nti;nti=l;i=i     V;X}wY#K}$
689 : TNS_NORULECHAIN/NS_NORULE
689 : RESSTATE
689 :                    ol;}w{ol;}#^;nti;nti=l;i=     X}wY#K}$
690 :X->oV;              ol;}w{ol;}#^;nti;nti=l;i=     X}wY#K}$
691 : SAVESTATE:           105
691 :                    ol;}w{ol;}#^;nti;nti=l;i=     oV;}wY#K}$
692 :                    l;}w{ol;}#^;nti;nti=l;i=i     V;}wY#K}$
693 :V->l                l;}w{ol;}#^;nti;nti=l;i=i     V;}wY#K}$
694 : SAVESTATE:           106
694 :                    l;}w{ol;}#^;nti;nti=l;i=i     l;}wY#K}$
695 :                    ;}w{ol;}#^;nti;nti=l;i=i(     ;}wY#K}$
696 :                    }w{ol;}#^;nti;nti=l;i=i(i     }wY#K}$
697 :                    w{ol;}#^;nti;nti=l;i=i(i)     wY#K}$
698 :                    {ol;}#^;nti;nti=l;i=i(i);     Y#K}$
699 :Y->{X}              {ol;}#^;nti;nti=l;i=i(i);     Y#K}$
700 : SAVESTATE:           107
700 :                    {ol;}#^;nti;nti=l;i=i(i);     {X}#K}$
701 :                    ol;}#^;nti;nti=l;i=i(i);o     X}#K}$
702 :X->oV;X             ol;}#^;nti;nti=l;i=i(i);o     X}#K}$
703 : SAVESTATE:           108
703 :                    ol;}#^;nti;nti=l;i=i(i);o     oV;X}#K}$
704 :                    l;}#^;nti;nti=l;i=i(i);oi     V;X}#K}$
705 :V->l                l;}#^;nti;nti=l;i=i(i);oi     V;X}#K}$
706 : SAVESTATE:           109
706 :                    l;}#^;nti;nti=l;i=i(i);oi     l;X}#K}$
707 :                    ;}#^;nti;nti=l;i=i(i);oi;     ;X}#K}$
708 :                    }#^;nti;nti=l;i=i(i);oi;^     X}#K}$
709 : TNS_NORULECHAIN/NS_NORULE
709 : RESSTATE
709 :                    l;}#^;nti;nti=l;i=i(i);oi     V;X}#K}$
710 : TNS_NORULECHAIN/NS_NORULE
710 : RESSTATE
710 :                    ol;}#^;nti;nti=l;i=i(i);o     X}#K}$
711 :X->oV;              ol;}#^;nti;nti=l;i=i(i);o     X}#K}$
712 : SAVESTATE:           108
712 :                    ol;}#^;nti;nti=l;i=i(i);o     oV;}#K}$
713 :                    l;}#^;nti;nti=l;i=i(i);oi     V;}#K}$
714 :V->l                l;}#^;nti;nti=l;i=i(i);oi     V;}#K}$
715 : SAVESTATE:           109
715 :                    l;}#^;nti;nti=l;i=i(i);oi     l;}#K}$
716 :                    ;}#^;nti;nti=l;i=i(i);oi;     ;}#K}$
717 :                    }#^;nti;nti=l;i=i(i);oi;^     }#K}$
718 :                    #^;nti;nti=l;i=i(i);oi;^;     #K}$
719 :                    ^;nti;nti=l;i=i(i);oi;^;?     K}$
720 :K->^;K             ^;nti;nti=l;i=i(i);oi;^;?     K}$
721 : SAVESTATE:           110
721 :                    ^;nti;nti=l;i=i(i);oi;^;?     ^;K}$
722 :                    ;nti;nti=l;i=i(i);oi;^;?i     ;K}$
723 :                    nti;nti=l;i=i(i);oi;^;?i>     K}$
724 :K->nti=V;K         nti;nti=l;i=i(i);oi;^;?i>     K}$
725 : SAVESTATE:           111
725 :                    nti;nti=l;i=i(i);oi;^;?i>     nti=V;K}$
726 :                    ti;nti=l;i=i(i);oi;^;?i>l     ti=V;K}$
727 :                    i;nti=l;i=i(i);oi;^;?i>l#     i=V;K}$
728 :                    ;nti=l;i=i(i);oi;^;?i>l#c     =V;K}$
729 : TS_NOK/NS_NORULECHAIN
729 : RESSTATE
729 :                    nti;nti=l;i=i(i);oi;^;?i>     K}$
730 :K->nti;K            nti;nti=l;i=i(i);oi;^;?i>     K}$
731 : SAVESTATE:           111
```

```
731 :                    nti;nti=l;i=i(i);oi;^;?i>     nti;K}$
732 :                    ti;nti=l;i=i(i);oi;^;?i>l     ti;K}$
733 :                    i;nti=l;i=i(i);oi;^;?i>l#     i;K}$
734 :                    ;nti=l;i=i(i);oi;^;?i>l#c     ;K}$
735 :                    nti=l;i=i(i);oi;^;?i>l#c{     K}$
736 :K->nti=V;K         nti=l;i=i(i);oi;^;?i>l#c{     K}$
737 : SAVESTATE:          112
737 :                    nti=l;i=i(i);oi;^;?i>l#c{     nti=V;K}$
738 :                    ti=l;i=i(i);oi;^;?i>l#c{o     ti=V;K}$
739 :                    i=l;i=i(i);oi;^;?i>l#c{oi     i=V;K}$
740 :                    =l;i=i(i);oi;^;?i>l#c{oi;     =V;K}$
741 :                    l;i=i(i);oi;^;?i>l#c{oi;^     V;K}$
742 :V->l               l;i=i(i);oi;^;?i>l#c{oi;^     V;K}$
743 : SAVESTATE:          113
743 :                    l;i=i(i);oi;^;?i>l#c{oi;^     l;K}$
744 :                    ;i=i(i);oi;^;?i>l#c{oi;^;     ;K}$
745 :                    i=i(i);oi;^;?i>l#c{oi;^;i     K}$
746 :K->i=W;K           i=i(i);oi;^;?i>l#c{oi;^;i     K}$
747 : SAVESTATE:          114
747 :                    i=i(i);oi;^;?i>l#c{oi;^;i     i=W;K}$
748 :                    =i(i);oi;^;?i>l#c{oi;^;i=     =W;K}$
749 :                    i(i);oi;^;?i>l#c{oi;^;i=i     W;K}$
750 :W->i               i(i);oi;^;?i>l#c{oi;^;i=i     W;K}$
751 : SAVESTATE:          115
751 :                    i(i);oi;^;?i>l#c{oi;^;i=i     i;K}$
752 :                    (i);oi;^;?i>l#c{oi;^;i=i/     ;K}$
753 : TS_NOK/NS_NORULECHAIN
753 : RESSTATE
753 :                    i(i);oi;^;?i>l#c{oi;^;i=i     W;K}$
754 :W->iF              i(i);oi;^;?i>l#c{oi;^;i=i     W;K}$
755 : SAVESTATE:          115
755 :                    i(i);oi;^;?i>l#c{oi;^;i=i     iF;K}$
756 :                    (i);oi;^;?i>l#c{oi;^;i=i/     F;K}$
757 :F->(N)              (i);oi;^;?i>l#c{oi;^;i=i/     F;K}$
758 : SAVESTATE:          116
758 :                    (i);oi;^;?i>l#c{oi;^;i=i/     (N);K}$
759 :                    i);oi;^;?i>l#c{oi;^;i=i/l     N);K}$
760 :N->i               i);oi;^;?i>l#c{oi;^;i=i/l     N);K}$
761 : SAVESTATE:          117
761 :                    i);oi;^;?i>l#c{oi;^;i=i/l     i);K}$
762 :                    );oi;^;?i>l#c{oi;^;i=i/l;     );K}$
763 :                    ;oi;^;?i>l#c{oi;^;i=i/l;}     ;K}$
764 :                    oi;^;?i>l#c{oi;^;i=i/l;}#     K}$
765 :K->oV;K            oi;^;?i>l#c{oi;^;i=i/l;}#     K}$
766 : SAVESTATE:          118
766 :                    oi;^;?i>l#c{oi;^;i=i/l;}#     oV;K}$
767 :                    i;^;?i>l#c{oi;^;i=i/l;}#}     V;K}$
768 :V->i               i;^;?i>l#c{oi;^;i=i/l;}#}     V;K}$
769 : SAVESTATE:          119
769 :                    i;^;?i>l#c{oi;^;i=i/l;}#}     i;K}$
770 :                    ;^;?i>l#c{oi;^;i=i/l;}#}      ;K}$
771 :                    ^;?i>l#c{oi;^;i=i/l;}#}       K}$
772 :K->^;K             ^;?i>l#c{oi;^;i=i/l;}#}       K}$
773 : SAVESTATE:          120
773 :                    ^;?i>l#c{oi;^;i=i/l;}#}       ^;K}$
774 :                    ;?i>l#c{oi;^;i=i/l;}#}        ;K}$
775 :                    ?i>l#c{oi;^;i=i/l;}#}         K}$
776 :K->?Z#RK            ?i>l#c{oi;^;i=i/l;}#}         K}$
777 : SAVESTATE:          121
777 :                    ?i>l#c{oi;^;i=i/l;}#}         ?Z#RK}$
778 :                    i>l#c{oi;^;i=i/l;}#}          Z#RK}$
779 :Z->iLi              i>l#c{oi;^;i=i/l;}#}          Z#RK}$
780 : SAVESTATE:          122
780 :                    i>l#c{oi;^;i=i/l;}#}          iLi#RK}$
781 :                    >l#c{oi;^;i=i/l;}#}           Li#RK}$
782 :L->>                >l#c{oi;^;i=i/l;}#}           Li#RK}$
783 : SAVESTATE:          123
783 :                    >l#c{oi;^;i=i/l;}#}           >i#RK}$
784 :                    l#c{oi;^;i=i/l;}#}            i#RK}$
785 : TS_NOK/NS_NORULECHAIN
785 : RESSTATE
785 :                    >l#c{oi;^;i=i/l;}#}           Li#RK}$
786 : TNS_NORULECHAIN/NS_NORULE
786 : RESSTATE
786 :                    i>l#c{oi;^;i=i/l;}#}          Z#RK}$
787 :Z->iLl              i>l#c{oi;^;i=i/l;}#}          Z#RK}$
788 : SAVESTATE:          122
788 :                    i>l#c{oi;^;i=i/l;}#}          iLl#RK}$
789 :                    >l#c{oi;^;i=i/l;}#}           Ll#RK}$
790 :L->>                >l#c{oi;^;i=i/l;}#}           Ll#RK}$
791 : SAVESTATE:          123
791 :                    >l#c{oi;^;i=i/l;}#}           >l#RK}$
792 :                    l#c{oi;^;i=i/l;}#}            l#RK}$
793 :                    #c{oi;^;i=i/l;}#}             #RK}$
794 :                    c{oi;^;i=i/l;}#}              RK}$
795 :R->cY#              c{oi;^;i=i/l;}#}              RK}$
796 : SAVESTATE:          124
796 :                    c{oi;^;i=i/l;}#}              cY#K}$
797 :                    {oi;^;i=i/l;}#}               Y#K}$
798 :Y->{X}              {oi;^;i=i/l;}#}               Y#K}$
799 : SAVESTATE:          125
799 :                    {oi;^;i=i/l;}#}               {X}#K}$
```

```
801 :X->oV;X              oi;^;i=i/l;}#}              X}#K}$
802 : SAVESTATE:           126
802 :                     oi;^;i=i/l;}#}              oV;X}#K}$
803 :                     i;^;i=i/l;}#}               V;X}#K}$
804 :V->i                 i;^;i=i/l;}#}               V;X}#K}$
805 : SAVESTATE:           127
805 :                     i;^;i=i/l;}#}               i;X}#K}$
806 :                     ;^;i=i/l;}#}                ;X}#K}$
807 :                     ^;i=i/l;}#}                 X}#K}$
808 :X->^;X               ^;i=i/l;}#}                 X}#K}$
809 : SAVESTATE:           128
809 :                     ^;i=i/l;}#}                 ^;X}#K}$
810 :                     ;i=i/l;}#}                  ;X}#K}$
811 :                     i=i/l;}#}                   X}#K}$
812 :X->i=W;X             i=i/l;}#}                   X}#K}$
813 : SAVESTATE:           129
813 :                     i=i/l;}#}                   i=W;X}#K}$
814 :                     =i/l;}#}                    =W;X}#K}$
815 :                     i/l;}#}                     W;X}#K}$
816 :W->i                 i/l;}#}                     W;X}#K}$
817 : SAVESTATE:           130
817 :                     i/l;}#}                     i;X}#K}$
818 :                     /l;}#}                      ;X}#K}$
819 : TS_NOK/NS_NORULECHAIN
819 : RESSTATE
819 :                     i/l;}#}                     W;X}#K}$
820 :W->iF                i/l;}#}                     W;X}#K}$
821 : SAVESTATE:           130
821 :                     i/l;}#}                     iF;X}#K}$
822 :                     /l;}#}                      F;X}#K}$
823 : TNS_NORULECHAIN/NS_NORULE
823 : RESSTATE
823 :                     i/l;}#}                     W;X}#K}$
824 :W->iQW               i/l;}#}                     W;X}#K}$
825 : SAVESTATE:           130
825 :                     i/l;}#}                     iQW;X}#K}$
826 :                     /l;}#}                      QW;X}#K}$
827 :Q->/                 /l;}#}                      QW;X}#K}$
828 : SAVESTATE:           131
828 :                     /l;}#}                      /W;X}#K}$
829 :                     l;}#}                       W;X}#K}$
830 :W->l                 l;}#}                       W;X}#K}$
831 : SAVESTATE:           132
831 :                     l;}#}                       l;X}#K}$
832 :                     ;}#}                        ;X}#K}$
833 :                     }#}                         X}#K}$
834 : TNS_NORULECHAIN/NS_NORULE
834 : RESSTATE
834 :                     l;}#}                       W;X}#K}$
835 :W->lQW               l;}#}                       W;X}#K}$
836 : SAVESTATE:           132
836 :                     l;}#}                       lQW;X}#K}$
837 :                     ;}#}                        QW;X}#K}$
838 : TNS_NORULECHAIN/NS_NORULE
838 : RESSTATE
838 :                     l;}#}                       W;X}#K}$
839 : TNS_NORULECHAIN/NS_NORULE
839 : RESSTATE
839 :                     /l;}#}                      QW;X}#K}$
840 : TNS_NORULECHAIN/NS_NORULE
840 : RESSTATE
840 :                     i/l;}#}                     W;X}#K}$
841 :W->iFQW              i/l;}#}                     W;X}#K}$
842 : SAVESTATE:           130
842 :                     i/l;}#}                     iFQW;X}#K}$
843 :                     /l;}#}                      FQW;X}#K}$
844 : TNS_NORULECHAIN/NS_NORULE
844 : RESSTATE
844 :                     i/l;}#}                     W;X}#K}$
845 : TNS_NORULECHAIN/NS_NORULE
845 : RESSTATE
845 :                     i=i/l;}#}                   X}#K}$
846 :X->iF;X              i=i/l;}#}                   X}#K}$
847 : SAVESTATE:           129
847 :                     i=i/l;}#}                   iF;X}#K}$
848 :                     =i/l;}#}                    F;X}#K}$
849 : TNS_NORULECHAIN/NS_NORULE
849 : RESSTATE
849 :                     i=i/l;}#}                   X}#K}$
850 :X->i=W;              i=i/l;}#}                   X}#K}$
851 : SAVESTATE:           129
851 :                     i=i/l;}#}                   i=W;}#K}$
852 :                     =i/l;}#}                    =W;}#K}$
853 :                     i/l;}#}                     W;}#K}$
854 :W->i                 i/l;}#}                     W;}#K}$
855 : SAVESTATE:           130
855 :                     i/l;}#}                     i;}#K}$
856 :                     /l;}#}                      ;}#K}$
857 : TS_NOK/NS_NORULECHAIN
857 : RESSTATE
857 :                     i/l;}#}                     W;}#K}$
858 :W->iF                i/l;}#}                     W;}#K}$
859 : SAVESTATE:           130
```

```
859 : SAVESTATE:          130
859 :                     i/1;}#}                   iF;}#K}$
860 :                     /1;}#}                    F;}#K}$
861 : TNS_NORULECHAIN/NS_NORULE
861 : RESSTATE
861 :                     i/1;}#}                   W;}#K}$
862 :W->iQW               i/1;}#}                   W;}#K}$
863 : SAVESTATE:          130
863 :                     i/1;}#}                   iQW;}#K}$
864 :                     /1;}#}                    QW;}#K}$
865 :Q->/                 /1;}#}                    QW;}#K}$
866 : SAVESTATE:          131
866 :                     /1;}#}                    /W;}#K}$
867 :                     1;}#}                     W;}#K}$
868 :W->1                 1;}#}                     W;}#K}$
869 : SAVESTATE:          132
869 :                     1;}#}                     1;}#K}$
870 :                     ;}#}                      ;}#K}$
871 :                     }#}                       }#K}$
872 :                     #}                        #K}$
873 :                     }                         K}$
874 : TNS_NORULECHAIN/NS_NORULE
874 : RESSTATE
874 :                     1;}#}                     W;}#K}$
875 :W->1QW               1;}#}                     W;}#K}$
876 : SAVESTATE:          132
876 :                     1;}#}                     1QW;}#K}$
877 :                     ;}#}                      QW;}#K}$
878 : TNS_NORULECHAIN/NS_NORULE
878 : RESSTATE
878 :                     1;}#}                     W;}#K}$
879 : TNS_NORULECHAIN/NS_NORULE
879 : RESSTATE
879 :                     /1;}#}                    QW;}#K}$
880 : TNS_NORULECHAIN/NS_NORULE
880 : RESSTATE
880 :                     i/1;}#}                   W;}#K}$
881 :W->iFQW              i/1;}#}                   W;}#K}$
882 : SAVESTATE:          130
882 :                     i/1;}#}                   iFQW;}#K}$
883 :                     /1;}#}                    FQW;}#K}$
884 : TNS_NORULECHAIN/NS_NORULE
884 : RESSTATE
884 :                     i/1;}#}                   W;}#K}$
885 : TNS_NORULECHAIN/NS_NORULE
885 : RESSTATE
885 :                     i=i/1;}#}                 X}#K}$
886 :X->iF;               i=i/1;}#}                 X}#K}$
887 : SAVESTATE:          129
887 :                     i=i/1;}#}                 iF;}#K}$
888 :                     =i/1;}#}                  F;}#K}$
889 : TNS_NORULECHAIN/NS_NORULE
889 : RESSTATE
889 :                     i=i/1;}#}                 X}#K}$
890 : TNS_NORULECHAIN/NS_NORULE
890 : RESSTATE
890 :                     ^;i=i/1;}#}               X}#K}$
891 :X->^;                ^;i=i/1;}#}               X}#K}$
892 : SAVESTATE:          128
892 :                     ^;i=i/1;}#}               ^;}#K}$
893 :                     ;i=i/1;}#}                ;}#K}$
894 :                     i=i/1;}#}                 }#K}$
895 : TS_NOK/NS_NORULECHAIN
895 : RESSTATE
895 :                     ^;i=i/1;}#}               X}#K}$
896 : TNS_NORULECHAIN/NS_NORULE
896 : RESSTATE
896 :                     i;^;i=i/1;}#}             V;X}#K}$
897 : TNS_NORULECHAIN/NS_NORULE
897 : RESSTATE
897 :                     oi;^;i=i/1;}#}            X}#K}$
898 :X->oV;               oi;^;i=i/1;}#}            X}#K}$
899 : SAVESTATE:          126
899 :                     oi;^;i=i/1;}#}            oV;}#K}$
900 :                     i;^;i=i/1;}#}             V;}#K}$
901 :V->i                 i;^;i=i/1;}#}             V;}#K}$
902 : SAVESTATE:          127
902 :                     i;^;i=i/1;}#}             i;}#K}$
903 :                     ;^;i=i/1;}#}              ;}#K}$
904 :                     ^;i=i/1;}#}               }#K}$
905 : TS_NOK/NS_NORULECHAIN
905 : RESSTATE
905 :                     i;^;i=i/1;}#}             V;}#K}$
906 : TNS_NORULECHAIN/NS_NORULE
906 : RESSTATE
906 :                     oi;^;i=i/1;}#}            X}#K}$
907 : TNS_NORULECHAIN/NS_NORULE
907 : RESSTATE
907 :                     {oi;^;i=i/1;}#}           Y#K}$
908 : TNS_NORULECHAIN/NS_NORULE
908 : RESSTATE
908 :                     c{oi;^;i=i/1;}#}          RK}$
```

```
909 : TNS_NORULECHAIN/NS_NORULE
909 : RESSTATE
909 :                    >l#c{oi;^;i=i/l;}#}          Ll#RK}$
910 : TNS_NORULECHAIN/NS_NORULE
910 : RESSTATE
910 :                    i>l#c{oi;^;i=i/l;}#}         Z#RK}$
911 : TNS_NORULECHAIN/NS_NORULE
911 : RESSTATE
911 :                    ?i>l#c{oi;^;i=i/l;}#}        K}$
912 :K->?Z#R             ?i>l#c{oi;^;i=i/l;}#}        K}$
913 : SAVESTATE:          121
913 :                    ?i>l#c{oi;^;i=i/l;}#}        ?Z#R}$
914 :                    i>l#c{oi;^;i=i/l;}#}         Z#R}$
915 :Z->iLi              i>l#c{oi;^;i=i/l;}#}         Z#R}$
916 : SAVESTATE:          122
916 :                    i>l#c{oi;^;i=i/l;}#}         iLi#R}$
917 :                    >l#c{oi;^;i=i/l;}#}          Li#R}$
918 :L->>                >l#c{oi;^;i=i/l;}#}          Li#R}$
919 : SAVESTATE:          123
919 :                    >l#c{oi;^;i=i/l;}#}          >i#R}$
920 :                    l#c{oi;^;i=i/l;}#}           i#R}$
921 : TS_NOK/NS_NORULECHAIN
921 : RESSTATE
921 :                    >l#c{oi;^;i=i/l;}#}          Li#R}$
922 : TNS_NORULECHAIN/NS_NORULE
922 : RESSTATE
922 :                    i>l#c{oi;^;i=i/l;}#}         Z#R}$
923 :Z->iLl              i>l#c{oi;^;i=i/l;}#}         Z#R}$
924 : SAVESTATE:          122
924 :                    i>l#c{oi;^;i=i/l;}#}         iLl#R}$
925 :                    >l#c{oi;^;i=i/l;}#}          Ll#R}$
926 :L->>                >l#c{oi;^;i=i/l;}#}          Ll#R}$
927 : SAVESTATE:          123
927 :                    >l#c{oi;^;i=i/l;}#}          >l#R}$
928 :                    l#c{oi;^;i=i/l;}#}           l#R}$
929 :                    #c{oi;^;i=i/l;}#}            #R}$
930 :                    c{oi;^;i=i/l;}#}             R}$
931 :R->cY#              c{oi;^;i=i/l;}#}             R}$
932 : SAVESTATE:          124
932 :                    c{oi;^;i=i/l;}#}             cY#}$
933 :                    {oi;^;i=i/l;}#}              Y#}$
934 :Y->{X}              {oi;^;i=i/l;}#}              Y#}$
935 : SAVESTATE:          125
935 :                    {oi;^;i=i/l;}#}              {X}#}$
936 :                    oi;^;i=i/l;}#}               X}#}$
937 :X->oV;X             oi;^;i=i/l;}#}               X}#}$
938 : SAVESTATE:          126
938 :                    oi;^;i=i/l;}#}               oV;X}#}$
939 :                    i;^;i=i/l;}#}                V;X}#}$
940 :V->i                i;^;i=i/l;}#}                V;X}#}$
941 : SAVESTATE:          127
941 :                    i;^;i=i/l;}#}                i;X}#}$
942 :                    ;^;i=i/l;}#}                 ;X}#}$
943 :                    ^;i=i/l;}#}                  X}#}$
944 :X->^;X              ^;i=i/l;}#}                  X}#}$
945 : SAVESTATE:          128
945 :                    ^;i=i/l;}#}                  ^;X}#}$
946 :                    ;i=i/l;}#}                   ;X}#}$
947 :                    i=i/l;}#}                    X}#}$
948 :X->i=W;X            i=i/l;}#}                    X}#}$
949 : SAVESTATE:          129
949 :                    i=i/l;}#}                    i=W;X}#}$
950 :                    =i/l;}#}                     =W;X}#}$
951 :                    i/l;}#}                      W;X}#}$
952 :W->i                i/l;}#}                      W;X}#}$
953 : SAVESTATE:          130
953 :                    i/l;}#}                      i;X}#}$
954 :                    /l;}#}                       ;X}#}$
955 : TS_NOK/NS_NORULECHAIN
955 : RESSTATE
955 :                    i/l;}#}                      W;X}#}$
956 :W->iF               i/l;}#}                      W;X}#}$
957 : SAVESTATE:          130
957 :                    i/l;}#}                      iF;X}#}$
958 :                    /l;}#}                       F;X}#}$
959 : TNS_NORULECHAIN/NS_NORULE
959 : RESSTATE
959 :                    i/l;}#}                      W;X}#}$
960 :W->iQW              i/l;}#}                      W;X}#}$
961 : SAVESTATE:          130
961 :                    i/l;}#}                      iQW;X}#}$
962 :                    /l;}#}                       QW;X}#}$
963 :Q->/                /l;}#}                       QW;X}#}$
964 : SAVESTATE:          131
964 :                    /l;}#}                       /W;X}#}$
965 :                    l;}#}                        W;X}#}$
966 :W->l                l;}#}                        W;X}#}$
967 : SAVESTATE:          132
967 :                    l;}#}                        l;X}#}$
968 :                    ;}#}                         ;X}#}$
969 :                    }#}                          X}#}$
970 : TNS_NORULECHAIN/NS_NORULE
```

```
970 : TNS_NORULECHAIN/NS_NORULE
970 : RESSTATE
970 :                       1;}#}                       W;X}#}$
971 :W->1QW                 1;}#}                       W;X}#}$
972 : SAVESTATE:             132
972 :                       1;}#}                       1QW;X}#}$
973 :                       ;}#}                        QW;X}#}$
974 : TNS_NORULECHAIN/NS_NORULE
974 : RESSTATE
974 :                       1;}#}                       W;X}#}$
975 : TNS_NORULECHAIN/NS_NORULE
975 : RESSTATE
975 :                       /1;}#}                      QW;X}#}$
976 : TNS_NORULECHAIN/NS_NORULE
976 : RESSTATE
976 :                       i/1;}#}                     W;X}#}$
977 :W->iFQW                i/1;}#}                     W;X}#}$
978 : SAVESTATE:             130
978 :                       i/1;}#}                     iFQW;X}#}$
979 :                       /1;}#}                      FQW;X}#}$
980 : TNS_NORULECHAIN/NS_NORULE
980 : RESSTATE
980 :                       i/1;}#}                     W;X}#}$
981 : TNS_NORULECHAIN/NS_NORULE
981 : RESSTATE
981 :                       i=i/1;}#}                   X}#}$
982 :X->iF;X                i=i/1;}#}                   X}#}$
983 : SAVESTATE:             129
983 :                       i=i/1;}#}                   iF;X}#}$
984 :                       =i/1;}#}                    F;X}#}$
985 : TNS_NORULECHAIN/NS_NORULE
985 : RESSTATE
985 :                       i=i/1;}#}                   X}#}$
986 :X->i=W;                i=i/1;}#}                   X}#}$
987 : SAVESTATE:             129
987 :                       i=i/1;}#}                   i=W;}#}$
988 :                       =i/1;}#}                    =W;}#}$
989 :                       i/1;}#}                     W;}#}$
990 :W->i                   i/1;}#}                     W;}#}$
991 : SAVESTATE:             130
991 :                       i/1;}#}                     i;}#}$
992 :                       /1;}#}                      ;}#}$
993 : TS_NOK/NS_NORULECHAIN
993 : RESSTATE
993 :                       i/1;}#}                     W;}#}$
994 :W->iF                  i/1;}#}                     W;}#}$
995 : SAVESTATE:             130
995 :                       i/1;}#}                     iF;}#}$
996 :                       /1;}#}                      F;}#}$
997 : TNS_NORULECHAIN/NS_NORULE
997 : RESSTATE
997 :                       i/1;}#}                     W;}#}$
998 :W->iQW                 i/1;}#}                     W;}#}$
999 : SAVESTATE:             130
999 :                       i/1;}#}                     iQW;}#}$
1000:                       /1;}#}                      QW;}#}$
1001:Q->/                   /1;}#}                      QW;}#}$
1002: SAVESTATE:             131
1002:                       /1;}#}                      /W;}#}$
1003:                       1;}#}                       W;}#}$
1004:W->1                   1;}#}                       W;}#}$
1005: SAVESTATE:             132
1005:                       1;}#}                       1;}#}$
1006:                       ;}#}                        ;}#}$
1007:                       }#}                         }#}$
1008:                       #}                          #}$
1009:                       }                           }$
1010:                                                   $
1011: LENTA_END
1012: ------>LENTA_END
```

Заключение

В ходе выполнения курсовой работы был разработан транслятор для языка программирования KVS-2021. Таким образом, были выполнены основные задачи данной курсовой работы:

- Сформулирована спецификация языка KVS-2021;
- Разработаны конечные автоматы и алгоритмы для реализация лексического анализатора;
- Разработана контекстно-свободная, приведённая к нормальной форме Грейбах, грамматика для описания синтаксически верных конструкций языка;
- Разработан семантический анализатор, осуществляющий проверку смысла используемых инструкций;
- Разработан транслятор с языка программирования KVS-2021 на язык низкого уровня Assembler;
- Проведено тестирование всех вышеперечисленных компонентов.

Окончательная версия языка KVS-2021 включает:

1) 2 типа данных;
2) Поддержка операции вывода;
3) 3 библиотечные функции
4) Возможность вызова функций стандартной библиотеки;
5) Наличие 3 арифметических, 3 побитовых и 4 логических операторов для вычисления выражений;
6) Структурированная система для обработки ошибок пользователя.
7) Условный оператор.
8) Циклический оператор.

Список использованных источников

1. Ахо, А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – М.: Вильямс, 2003. – 768с.

2. Смелов, В.В. Курс лекций по предмету языки программирования – 2016

3. Прата, С. Язык программирования C++. Лекции и упражнения / С. Прата. – М., 2006 — 1104 с.

4. Страуструп, Б. Принципы и практика использования C++ / Б. Страуструп – 2009 – 1238 с.