ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель курсового проекта – разработка транслятора KPI-2016.

Для достижения цели сформулированы следующие задачи.

1. Создание лексического анализатора используя конечные автоматы (глава 3).
2. Создание синтаксического анализатора используя автомат с магазинной памятью (глава 4).
3. Создание программы по преобразованию выражения в обратную польскую нотацию (глава 6).
4. Создание генератора кода, генерирующего объектный код на языке Assembler (глава 7).

ГЛАВА 1. СПЕЦИФИКАЦИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ

1.1. Характеристика языка программирования

Язык программирования KPI-2016 является процедурным, строго типизированным.

1.2. Алфавит языка

Базовая таблица символов для языка KPI-2016, представлена на рис. 1.1 (ASCII, 7 bit)

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Рисунок 1.1 Базовые символы языка KPI-2016

Запрещенные символы представлены в табл. 1.1

Табл. 1.1 Запрещенные символы языка KPI-2016

Символ	Символ в 16 с/с
|	0x7C
#	0x23

Если был допущен запрещенный символ – транслятор игнорирует его, выдавая об этом сообщение в протокол. Ошибка не приводит к остановке транслятора.

1.3. Применяемые сепараторы

Язык KPI-2016 разрешает использовать сепараторы, для написания исходного кода, представленные в табл. 1.2

Табл. 1.2 Символы-сепараторы языка KPI-2016

Символ	Символ в 16 с/с	Описание
( )	0x28 ; 0x29	Приоритетность операций
( )	0x28 ; 0x29	Параметры
“ “	0x22	Строковый литерал
;	0x3B	Разделитель инструкций
\\	0x5B;0x5B	Однострочный комментарий, строка игнорируется транслятором
\ ... \	0x5B;0x5B	Многострочный комментарий, тело игнорируется транслятором

1.4. Применяемые кодировки

Язык KPI-2016 использует кодировку 7-битовая кодировка ASCII

1.5. Типы данных

В языке KPI-2016 разрешены типы данных, представленные в табл. 1.3.

Табл. 1.3 Типы данных языка KPI-2016

Тип	Описание
int	Целочисленный, беззнаковый тип данных, обязательная инициализация при объявлении
str	Строковый тип данных. Макс. 255 символа. Последний символ \0 – окончание строки. Обязательная инициализация строки при объявлении.

1.6. Преобразование типов данных

Язык KPI-2016 не поддерживает преобразование типов данных.

1.7. Идентификаторы

В языке KPI-2016 разрешается использовать идентификаторы. Идентификатор должен быть написан на допустимых символах языка (п.1.2). Запрещается использовать идентификаторы длиной больше 10 символов. Запрещается использовать цифры. Идентификатор не должен совпадать с ключевыми словами языка. Для стандартной библиотеки, при ее вызове, резервируются идентификаторы: sqrt, pow.

1.8. Литералы

В языке KPI-2016 разрешены 2 вида литералов:

1. Числа интерпретируются как int (п. 1.5) и могут быть только rvalue.
2. Строки интерпретируются как str (п. 1.5) и могут быть только rvalue. Текст строкового литерала должен быть заключен в кавычки.

1.9. Объявление данных и область видимости

В языке KPI-2016 область видимости сверху вниз, слева направо. Операторы могут быть объявлены только локально.

1.10. Инициализация данных

Инициализация данных по умолчанию, используемая в языке KPI-2016, описана в табл. 1.3.

1.11. Инструкции языка

Язык KPI-2016 предусматривает инструкции, описанные в табл. 1.4

Табл. 1.4 Инструкции языка KPI-2016

<тип данных> <идентификатор>:<значение>	Объявление переменных.
func < тип данных> <идентификатор> (<тип данных> < идентификатор >, …)	Объявление внешних функций.
:	Присвоение значений
sout <идентификатор/литерал>	Вывод в стандартный поток вывода.

1.12. Операции языка

В языке KPI-2016 предусмотрены операции, описанные в табл. 1.5.

Табл. 1.5 Операции языка KPI-2016

Операция	Описание
:	Бинарный, присвоение
+	Бинарный, суммирование
-	Бинарный, разность
*	Бинарный, умножение
/	Бинарный, деление
>	Бинарный, больше
<	Бинарный, меньше

1.13. Выражения и их вычисления

В языке KPI-2016 разрешается использование операции с выражениями, описанные в табл. 1.5

1.14. Программные конструкции языка

В языке KPI-2016 можно использовать конструкции языка, описанные в табл. 1.6.

Табл. 1.6 Конструкции программы языка KPI—2016

Конструкция	Описание
begin [<текст программы> **return** <идентификатор>;]	Главная функция (точка входа)
if(<условие>**)** **[** <текст программы>] **else** //если не выполняется **[** <текст программы>]	Оператор условия.
while (<условие>**)** **[** <текст программы> **]**	Цикл с предусловием
func <тип данных> <идентификатор> **(**<тип данных><идентификатор>, …**)** **[** <текст программы> **return** <идентификатор>**;** **]**	Тело функции

1.15. Область видимости идентификаторов

Область видимости идентификаторов в языке KPI-2016 – локальная внутри программных блоков функций.

1.16. Семантические проверки

В языке KPI -2016 предусмотрена следующие семантические проверки:

1) Проверка на наличие переобъявление идентификатора внутри функций.
2) Проверка на повторное наименование функции.
3) Проверка наличия точки входа begin.
4) Проверка деления на ноль.
5) Идентификатор не должен содержать цифру

6) Проверка на наличие в параметрах функции типа данных STR
7) Проверка на наличия типа данных STR в функции
8) Проверка на многократное подключение библиотеки
9) Проверка на использования типа данных STR в выражениях

1.17. Распределение оперативной памяти на этапе выполнения

В трансляторе KPI-2016 предусмотрено распределение оперативной памяти на этапе генерации: подсчет выражений ведется через вычислительный стек. Промежуточный код, 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 ее состав

В языке KPI-2016 разрешено использовать стандартную математическую библиотеку. Для возможности использования функций стандартной библиотеки необходимо, используя операцию препроцессора, дать команду на вставку кода: **include KPIlib;** Возможные функции стандартной библиотеки описаны в табл. 1.7.

Табл. 1.7 Разрешенные методы библиотеки KPIlib

Метод	Описание

xpow(<**int** идентификатор/ литерал>,<**int** идентификатор/ литерал>)	Данная функция возводит число в степень
sqrt(<**int** идентификатор/ литерал>)	Данная функция вычисляет корень из числа int

1.19. Ввод и вывод данных

В языке KPI-2016 не предусмотрен поток ввода, поток вывода описан в табл. 1.4.

1.20. Точка входа

В языке KPI-2016 может быть только одна точка входа и определяется наличием функции **begin.** При инициализации более одной или менее одной – выдаст ошибку лексического анализатора.

1.21. Препроцессор

В языке KPI-2016 предусмотрены операции препроцессора – вставка кода стандартной библиотеки (описана в п.1.18).

1.22. Соглашения о вызовах

В языке KPI-2016 применяется собственное соглашение о вызове.
Используется по умолчанию в программе. Порядок обработки аргументов – справа - налево, через стек.

1.23. Объектный код

В языке KPI-2016 предусмотрено построение объектного кода на основе промежуточного кода, с помощью Microsoft Micro Assembler

1.24. Классификация сообщений транслятора

В языке KPI-2016 существует префиксная таблица классификации сообщений транслятора, описанная в табл. 1.8 и таблица с критическими ошибками, описанная в табл. 1.9.

Табл. 1.8 Префиксная классификация сообщений транслятора KPI-2016

Префикс	Описание
[LA]	Префикс для ошибок лексического анализатора
[IN]	Префикс для фатальных ошибок
[SA]	Префикс для ошибок синтаксического анализатора
[MA]	Префикс для ошибок семантического анализатора

Табл. 1.9 Фатальные ошибки транслятора KPI-2016

Номер ошибки	Описание
Ошибка 001	Системный сбой
Ошибка 100	Параметр in должен быть задан
Ошибка 104	Превышена длина входного параметра
Ошибка 110	Ошибка при открытии файла с исходным кодом
Ошибка 111	Недопустимый символ в исходном файле
Ошибка 112	Ошибка при создании файла протокола
Ошибка 114	Многострочный комментарий должен быть закрытым
Ошибка 115	Неправильное объявление строчного литерала

1.25. Контрольный пример

Пример программы, реализованный на языке KPI-2016 представлен ниже

```
include KPIlib
func int fact(int n)
[
	sout `in function factorial` endl;
	int res:1;
	int count:25;
	n:n+1;
	while(count<n)
	[
		res:res*count;
		count:count+1;
	]
	return res;
```

```
]
begin
[
	str label: "KPI-2016 Translator";
	sout label endl;
	label: "new label";
	sout label endl;
	int fac: fact(3);
	sout `factorial: `;
	sout fac endl;
	int exp: xpow(2,5) + (12/12-(41*31-10)*((32-12)*20)/100) + xpow(2,5);
	sout `exp: `;
	sout exp endl;
	int expTwo: sqrt(25);
	sout `sqrt(25): `;
	sout expTwo endl;
	while(expTwo < 10)
	[
		if(expTwo>8)
		[
			sout expTwo;
			sout ` below 8` endl;
		]
		else
		[
			sout expTwo;
			sout ` abow 8` endl;
		]
		expTwo:expTwo+1;
	]
	return 0;
]
```

ГЛАВА 2. СТРУКТУРА ТРАНСЛЯТОРА

2.1. Компоненты транслятора их назначение и принципы взаимодействия

Схема, демонстрирующая работу транслятора представлена на рис. 2.1.

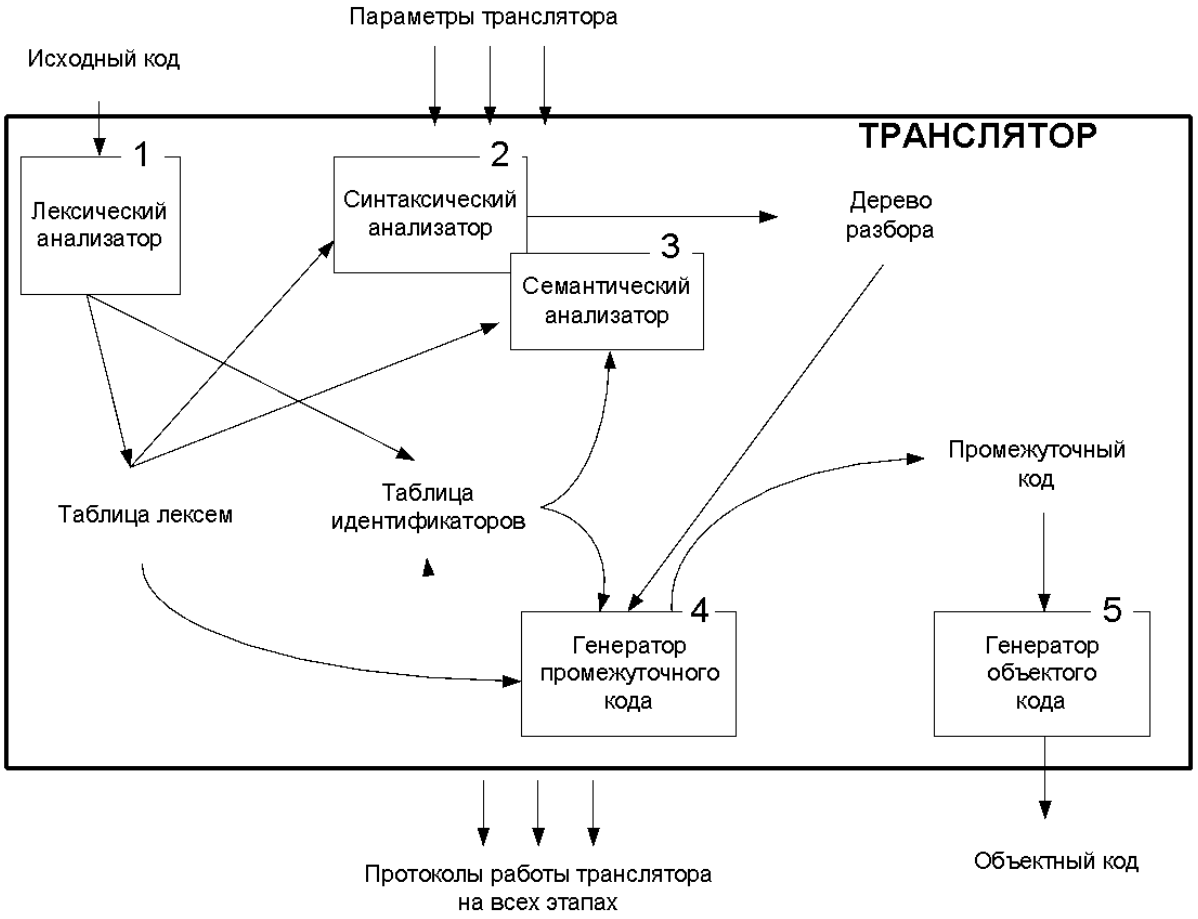

Рисунок 2.1 Схема работы транслятора KPI-2016

2.2. Перечень входных параметров транслятора

В трансляторе KPI-2016 предусмотрены входные параметры. Приоритет параметров повышается слева направо, если было введено более одного параметра, для одного этапа трансляции. Максимальная разрешенная длина параметра – 80 символов. Входные параметры описаны в табл. 2.1

Табл. 2.1 Входные параметры транслятора KPI-2016

Входной параметр	**Описание**
-in:	Указывает трансляторy в каком месте лежит исходный код. (Обязательный параметр)

-out:	Указывает транслятору в какой файл выводить объектный код после этапа трансляции. (При отсутствии за основу берется параметр -in)
-log:	Указывает транслятору в какой файл выводить протокол работы транслятора. (При отсутствии за основу берется параметр -in)
-DT	Вывод таблицы разделения слов. (При отсутствии параметра, таблица выводиться только в протокол)
-LT	Вывод таблицы лексем в консоль. (При отсутствии параметра, таблица выводиться только в протокол)
-IT	Вывод таблицы идентификаторов в консоль. (При отсутствии параметра, таблица выводиться только в протокол)
-SA	Вывод дерева разбора.(При отсутствии параметра, дерево выводиться только в протокол)
-R	Вывод правил разбора.(При отсутствии параметра, дерево выводиться только в протокол)
-NT	Вывод таблицы промежуточного кода. (При отсутствии параметра, таблица выводиться только в протокол)

2.3. Перечень протоколов, формируемых транслятором и их содержания

В языке KPI-2016, транслятор формирует протокол работы, описанный в таблице 2.2.

Табл. 2.2 Описание протоколов транслятора KPI-2016

Протокол	**Описание**
Лексического анализатора	Формирует в протокол таблицу лексем (ТЛ) и таблицу идентификаторов (ТИ). Каждый идентификатор ТЛ ссылается на ТИ. ТЛ формируется как входная таблица синтаксического анализатора, ТИ формируется для всего транслятора.
Синтаксического анализатора	Формирует в протокол пошаговую работу магазинного автомата с деревом разбора, для последующего разбора генератором кода.

Генератора кода	Формирует список структур промежуточного кода по таблице лексем, полученную из лексического анализатора. Из структур формируется объектный код ассемблера

ГЛАВА 3. РАЗРАБОТКА ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА

3.1. Структура лексического анализатора

Рисунок, описывающий работу лексического анализатора представлен на рис. 3.1

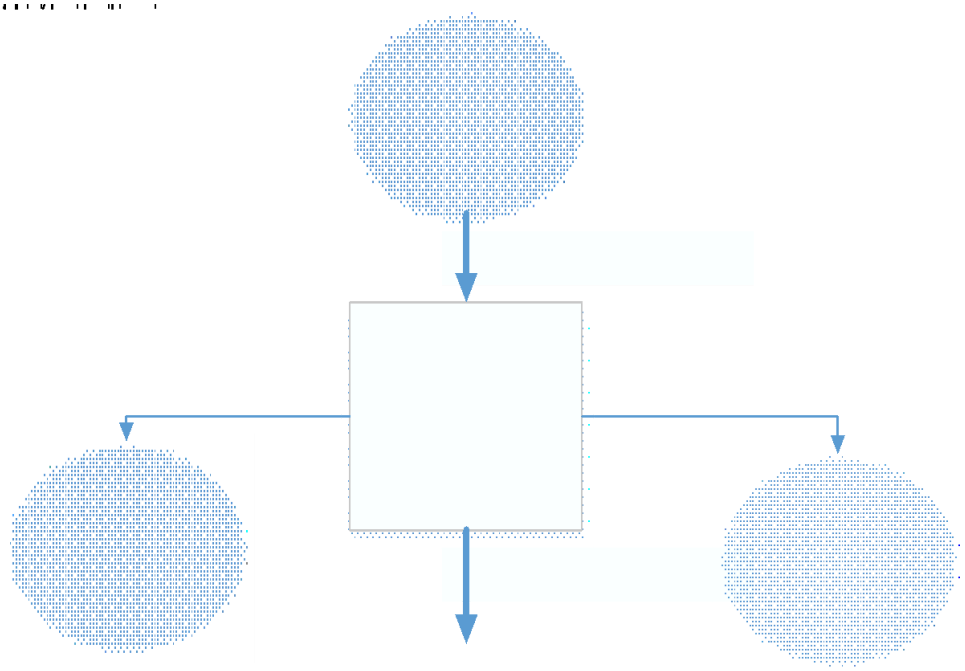

Рис.3.1. Структура лексического анализатора

Лексический анализатор представляет собой программу, которая транслирует исходную программу в набор символ, которая является таблицу лексем и формирует таблицу идентификаторов, записывая в нее полученную информацию об идентификаторах и литералах.

3.2 Контроль входных символов

Таблица контроля входных символов представлена на рис. 3.2.
Принцип работы таблицы состоит в следующем: каждому элементу в таблице соответствует значение в 16 с/с такой же, как и таблица ASCII.

Допустимые символы на рисунке обозначены буквой - T, запрещенные - F, игнорируемые - I, сепараторы - S, символ перехода на новую строку - N, символы пробела - P, кавычки - Q, комментарий - K.

```
IN::T, IN::T, IN::T, IN::T, IN::T, IN::T, IN::T, IN::T, IN::T, IN::P, IN::N, IN::T, IN::T, IN::I, 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::P, IN::T, IN::Q, IN::T, IN::T, IN::T, IN::T, IN::T, IN::Q, 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::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::T, IN::T, IN::T, IN::S, IN::K, IN::S, IN::T, IN::T
IN::Q, 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
```

Рисунок 3.2. Входная таблица символов KPI-2016.

3.3 Удаление избыточных символов

Алгоритм удаления избыточных символов (пробел, табуляция, переход на новую строку) основан на таблице входных символов (рис. 3.1)
Суть алгоритма:
1) Смотрим 1 байт.
1.1) Если он является пробелом (P), прибавляем один к переменной, которая отвечает за количество пробелов.
1.2) Если он является сепаратором (S), смотрим есть ли перед ним и после него существует один или несколько пробелов, если есть удаляем его.
1.3) Если он является переводом на новую строку (N), проверяем следующий байт, если он является пробелом, удаляем его.
1.4) Если он является кавычкой (Q), пропускаем все символы до тех пор, пока не встретим закрывающую кавычку.
1.5) Если он является разрешенным символом (T), зануляем переменную отвечающую за количество пробелов.

3.4. Перечень ключевых слов, сепараторов, символов операций соответствующих им лексем, конечных автоматов

Перечень ключевых слов, сепараторов, конечных автоматов транслятора KPI-2016 представлен в приложении А.

3.5. Основные структуры данных

Основные структуры данных лексического анализатора приведены в приложении В.

3.6 Принцип обработки ошибок

При обнаружении ошибки, работа транслятора KPI-2016 останавливается. Код ошибки и информация об ошибке выводиться на консоль и в протокол.

3.7 Структура и перечень сообщений лексического анализатора

Перечень сообщений лекс. анализатора представлен в табл. 3.1

Таблица 3.1. Перечень сообщений лексического анализатора

Номер сообщения	**Содержание сообщения**
Ошибка 200	Превышен максимальный размер таблицы лексем (4096)
Ошибка 201	Таблица лексем переполнена
Ошибка 202	Превышен максимальный размер таблицы идентификаторов (4096)
Ошибка 203	Таблица идентификаторов переполнена

3.8 Параметры лексического анализатора и режимы его работы

Транслятор KPI-2016 допускает использование параметров для управления работой лексического анализатора. Принцип их использования описан в табл. 2.1.

3.9. Алгоритм лексического анализа

Алгоритм работы лексического анализатора предоставлен ниже.
1) Выполняется предпоготовка исходного кода для анализатора.
1.1) Удаляются избыточные символы.
1.2) Происходит разделение лексем по цепочкам в отдельную структуру.
2) Из подготовленной структуры берем цепочку и выполняем проверку на алгоритме конечных автоматах.
2.1.1) Если строка совпала с автоматом – берем все данные из него.
2.1.1) Если символ-лексема – заполняем структуру лексем и заносим нужную информацию в стек (если необходимо).
2.1.2) Если идентификатор – заполняем структуру лексем и выполняем проверки.
2.1.3) Если литерал – заполняем таблицу лексем и идентификаторов.

3.10 Контрольный пример

Результат работы ЛА на основе контрольного примера из табл. 1.25 предоставлен на рис. 3.2 и рис. 3.3.

```
Таблица лексем:

01 %K
03 ftv{2}(tv{3})
04 [
05 pl{4}n;
06 tv{5}:l{6};
07 tv{7}:l{8};
08 v{9}:v{10}a{11}l{11};
09 c(v{12}b{13}v{13})
10 [
11 v{14}:v{15}a{16}v{16};
12 v{17}:v{18}a{19}l{19};
13 ]
14 rv{20};
15 ]
17 m
18 [
19 tv{21}:l{22};
20 pv{23}n;
21 tv{24}:v{25}(l{26});
22 pl{27};
23 pv{28}n;
24 tv{29}:v{30}(l{31})a{32}(l{32}a{33}l{33}a{34}(l{34}a{35}l{35}a{36}l{36})a{37}((l{37}a{38}l{38})
25 pl{44};
26 pv{45}n;
27 tv{46}:v{47}(l{48});
28 pl{49};
29 pv{50}n;
30 c(v{51}b{52}l{52})
31 [
32 i(v{53}b{54}l{54})
33 [
34 pv{55};
35 pl{56}n;
36 ]
37 e
38 [
39 pv{57};
40 pl{58}n;
41 ]
42 v{59}:v{60}a{61}l{61};
43 ]
44 rl{62};
45 ]
```

Рисунок 3.2. Результат работы лексического анализатора (таблица лексем)

Таблица идентификаторов:

IT	LT	УКАЗ.	ИМЯ	ТИП ИДЕНТИФИКАТОРА	ЗНАЧЕНИЕ
00	1		sqrt	функция	
01	1		xpow	функция	
02	6		factorial	int функция	
03	9		factorialn	int параметр функции	
04	15		L01	str литерал	'in function factorial' [23]
05	20		factorialres	int переменная	0
06	22		L02	int литерал	1
07	26		factorialcount	int переменная	0
08	28		L03	int литерал	2
09	9	*	factorialn	int переменная	0
10	9	*	factorialn	int переменная	0
11	35		L04	int литерал	1
12	26	*	factorialcount	int параметр в цикле	
13	9	*	factorialn	int параметр в цикле	
14	20	*	factorialres	int переменная	0
15	20	*	factorialres	int переменная	0
16	26	*	factorialcount	int переменная	0
17	26	*	factorialcount	int переменная	0
18	26	*	factorialcount	int переменная	0
19	58		L05	int литерал	1
20	20	*	factorialres	int переменная	0
21	75		beginlabel	str переменная	
22	77		L06	str литерал	'KPI-2016 Translator' [21]
23	75	*	beginlabel	str переменная	
24	86		beginfac	int переменная	0
25	6	*	factorial	int вызываемая функция	
26	90		L07	int параметр в вызываемой функции	
27	95		L08	str литерал	'factorial: ' [13]
28	86	*	beginfac	int переменная	0
29	104		beginexp	int переменная	0
30	6	*	factorial	int вызываемая функция	
31	108		L09	int параметр в вызываемой функции	
32	112		L10	int переменная	12
33	114		L11	int литерал	12
34	117		L12	int переменная	41
35	119		L13	int литерал	31
36	121		L14	int литерал	10
37	126		L15	int переменная	32
38	128		L16	int литерал	12
39	131		L17	int литерал	20
40	134		L18	int литерал	100
41	1	*	xpow	вызываемая функция	
42	139		L19	int параметр в вызываемой функции	
43	141		L20	int параметр в вызываемой функции	
44	146		L21	str литерал	'exp: ' [7]
45	104	*	beginexp	int переменная	0
46	155		beginexpTwo	int переменная	0
47	1	*	sqrt	вызываемая функция	
48	159		L22	int параметр в вызываемой функции	
49	164		L23	str литерал	'sqrt(25): ' [12]
50	155	*	beginexpTwo	int переменная	0
51	155	*	beginexpTwo	int параметр в цикле	
52	176		L24	int параметр в цикле	
53	155	*	beginexpTwo	int параметр в условии	
54	185		L25	int параметр в условии	
55	155	*	beginexpTwo	int переменная	0
56	195		L26	str литерал	' below 8' [10]
57	155	*	beginexpTwo	int переменная	0
58	210		L27	str литерал	' abow 8' [9]
59	155	*	beginexpTwo	int переменная	0
60	155	*	beginexpTwo	int переменная	0
61	220		L28	int литерал	1
62	226		L29	int литерал	0

Рисунок 3.3. Результат работы лексического анализатора (таблица идентификаторов)

ГЛАВА 4. РАЗРАБОТКА СИНТАКСИЧЕСКОГО АНАЛИЗАТОРА

4.1 Структура Синтаксического анализатора

Описание структуры синтаксического анализатора представлено на рис. 4.1.

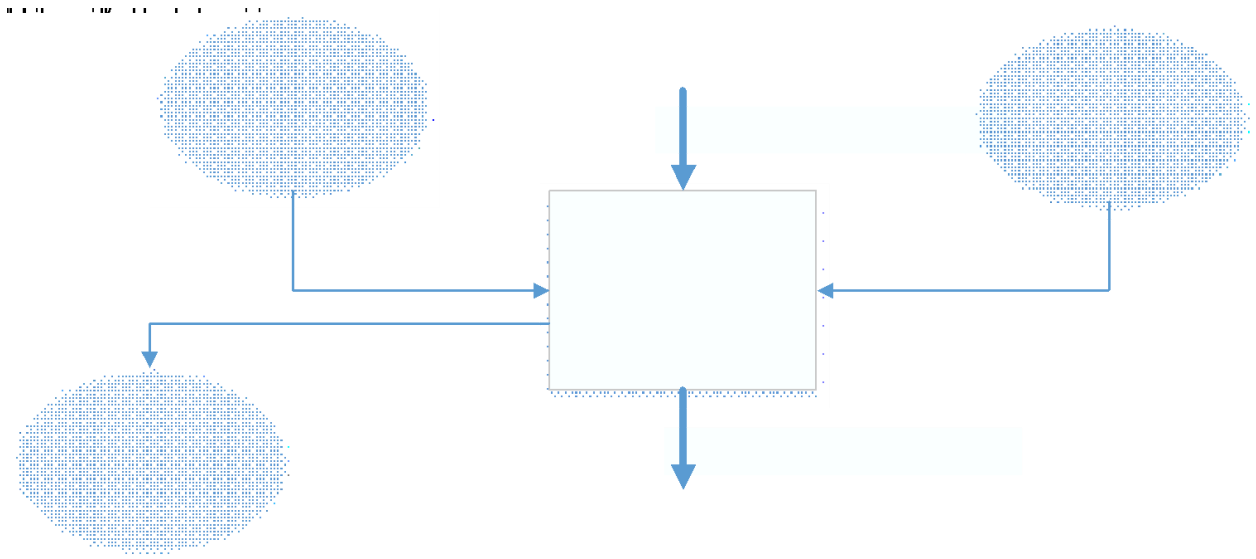

Рис.4.1. Структура синтаксического анализатора

Синтаксический анализатор представляет собой программу, проверяющую таблицу лексем по определенным правилам грамматики Грейбах. При удачной проверке таблицы лексем – формирует протокол работы, представляющий из себя работу автомата и дерева разбора.

4.2. Контекстно свободная грамматика, описывающая синтаксис языка

Грамматика для синтаксического разбора языка KPI-2016 представлена в табл 4.1.

Табл 4.1. Таблица правил перехода нетерминальных символов

Нетерминал	Нетерминал
S→	%KS ftv(F)[N]S m[N]
F→	tv tv,F
C→	v:E;C v:E;e pEn;C pEn; pE;C

	pE; sE;C sE;
N→	tv:E;N c(E)[C]N i(E)[C]N i(E)[C]e[C]N v:E;N pE;N pEn;N pEn;N v(W);N rE;
E→	V L (E) v(W) vM lM (E)M v(W)M
W→	v,W | l,W | l | v
M→	aE | bE | aEM | bEM

4.3. Построение конечного магазинного автомата

Формальное описание конечного магазинного автомата для Грейбах-грамматики представлено на рис 4.1.

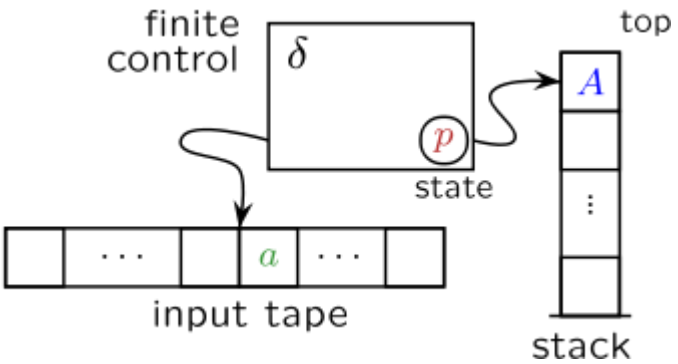

Рисунок 4.1. Формальное описание конечного магазинного автомата для Грейбах-грамматики

4.4. Основные структуры данных

Основные структуры данных описаны в приложении 4.

4.5. Описание алгоритма синтаксического разбора

Принцип работы синтаксического разбора транслятора KDS-2016 приведен ниже.

1. В магазин записывается стартовый символ.
2. На основе полученной таблицы лексем формируется входная лента.
3. Запускается автомат и выбирается цепочка, соответствующая нетерминальному символу, записывается в магазин в обратном порядке.
4. Если терминалы в стеке и в ленте совпадают, то данный терминал удаляется с ленты и стека. Иначе возвращаемся в предыдущее сохраненное состояние и выбираем другую цепочку нетерминала.
5. Если в магазине встретился нетерминал, переходим к пункту 3.
6. Если символ достиг символа дна стека, и лента в этот момент имеет символ дна стека, то синтаксический анализ выполнен успешно. Иначе генерируется ошибка.

4.6. Структура и перечень сообщений синтаксического анализатора

Перечень сообщений синтаксического анализатора представлен в табл. 4.2

Таблица 4.2. Перечень сообщений синтаксического анализатора

Номер сообщения	**Содержание сообщения**
Ошибка 600	Неверная структура программы
Ошибка 601	Параметры функции заданы неверно
Ошибка 602	Структура цикла или условия составлено неверно
Ошибка 603	Конструкция функции составлено неверно
Ошибка 604	Выражение составлено неверно

4.7. Параметры синтаксического анализатора и режимы его работы

Транслятор KPI-2016 допускает использование параметров для управления работой синтаксического анализатора. Принцип их использования описан в таблице 2.1.

4.8. Принцип обработки ошибок

При встрече нетерминального символа программа переходит к замене этого символа в соответствии с табл. 4.1. Если программа не находит подходящего правила, то генерируется ошибка из табл. 4.2.

4.9. Контрольный пример

Распечатка дерева разбора предоставлено в приложении D. Результат работы синтаксического анализатора приведен ниже:

```
Шаг : Правило          Входная лента              Стек
0   : S->%KS           %Kftv(tv)[pln;tv:l;tv:l;v     S$
0   : SAVESTATE:       1
0   :                  %Kftv(tv)[pln;tv:l;tv:l;v     %KS$
1   :                  Kftv(tv)[pln;tv:l;tv:l;v:     KS$
2   :                  ftv(tv)[pln;tv:l;tv:l;v:v     S$
3   : S->ftv(F)[N]S    ftv(tv)[pln;tv:l;tv:l;v:v     S$
3   : SAVESTATE:       2
3   :                  ftv(tv)[pln;tv:l;tv:l;v:v     ftv(F)[N]S$
4   :                  tv(tv)[pln;tv:l;tv:l;v:va     tv(F)[N]S$
5   :                  v(tv)[pln;tv:l;tv:l;v:val     v(F)[N]S$
6   :                  (tv)[pln;tv:l;tv:l;v:val;     (F)[N]S$
7   :                  tv)[pln;tv:l;tv:l;v:val;c     F)[N]S$
8   : F->tv            tv)[pln;tv:l;tv:l;v:val;c     F)[N]S$
8   : SAVESTATE:       3
8   :                  tv)[pln;tv:l;tv:l;v:val;c     tv)[N]S$
9   :                  v)[pln;tv:l;tv:l;v:val;c(     v)[N]S$
10  :                  )[pln;tv:l;tv:l;v:val;c(v     )[N]S$
11  :                  [pln;tv:l;tv:l;v:val;c(vb     [N]S$
12  :                  pln;tv:l;tv:l;v:val;c(vbv     N]S$
13  : N->pP;N          pln;tv:l;tv:l;v:val;c(vbv     N]S$
…
1314: SAVESTATE:       97
1314:                  l;]rl;]                  l;]N]$
1315:                  ;]rl;]                   ;]N]$
1316:                  ]rl;]                    ]N]$
1317:                  rl;]                     N]$
1318: N->rv;           rl;]                        N]$
1318: SAVESTATE:       98
1318:                  rl;]                     rv;]$
1319:                  l;]                      v;]$
1320: TS_NOK/NS_NORULECHAIN
1320: RESSTATE
1320:                  rl;]                     N]$
1321: N->rl;           rl;]                        N]$
1321: SAVESTATE:       98
1321:                  rl;]                     rl;]$
1322:                  l;]                      l;]$
1323:                  ;]                       ;]$
1324:                  ]                        ]$
```

```
1325:                                   $
1326: LENTA_END
1327: ------>LENTA_END
-----------------------------------------------------------------
0    : всего строк 196, синтаксический анализ выполнен без ошибок
```

ГЛАВА 5. РАЗРАБОТКА СЕМАНТИЧЕСКОГО АНАЛИЗАТОРА

5.1. Структура семантического анализатора

Семантический анализатор, используемый в языке KPI-2016, находится в лексическом анализаторе. Рисунок, поясняющий расположение компонентов семантического анализатора предоставлен на рис.2.1.

5.2. Функции семантического анализа

Все возможные проверки семантического анализатора описаны в п.1.16

5.3. Структура и перечень сообщений семантического анализатора

Так как семантический анализатор языка KPI-2016 объединен с лексическим и синтаксическим анализатором, следовательно, он не имеет своих генерируемых ошибок

5.4. Принцип обработки ошибок

Так как семантический анализатор языка KPI-2016 объединен с лексическим и синтаксическим анализатором следовательно, он не имеет отдельного принципа обработки ошибок.

ГЛАВА 6. ПРЕОБРАЗОВАНИЕ ВЫРАЖЕНИЙ

6.1. Допускаемые выражения

В языке KPI-2016 допускается вычисление выражений содержащие числовой литерал и функции, возвращаемым значением которого должен быть тип данных, соответствующий числовому литералу.

Работа над строками не допускается в выражениях.

Приоритетность операций, используемая в алгоритме при переводе в обратную польскую запись, описана в табл. 6.1.

Табл. 6.1 Приоритетность операций при переводе в ОПЗ

Операторы	Приоритетность
* /	Высокий
+ -	Средний
( )	Низкий

6.2. Польская запись и принцип ее построения

Принцип построения польской записи.
1. Рассматриваем последовательно каждый символ выражения:
2. Если этот символ - число или переменная, то помещаем его в выходную строку.
3. Если символ - знак операции (+, -, *, /), то проверяем приоритет данной операции.
Получив один из этих символов, мы должны проверить стек:
а) Если стек все еще пуст, или находящиеся в нем символы имеют меньший приоритет, чем приоритет текущего символа, то помещаем текущий символ в стек.
б) Если символ, находящийся на вершине стека имеет приоритет, больший или равный приоритету текущего символа,то извлекаем символы из стека в выходную строку до тех пор, пока выполняется это условие; затем переходим к пункту а).
3. Если текущий символ - открывающая скобка, то помещаем ее в стек.
4. Если текущий символ - закрывающая скобка, то извлекаем символы из стека в выходную строку до тех пор, пока не встретим в стеке открывающую скобку. Скобка извлекается из стека. Если вся входная строка разобрана, а в стеке еще остаются знаки операций, извлекаем их из стека в выходную строку.

6.3. Программная реализация обработки выражений

Алгоритм построения кода на языке C++ соответствует алгоритму, описанному в п. 6.2.

6.4. Контрольный пример

Результат работы преобразования выражений из контрольного примера п.1.25. представлен на рисунке 6.1.

```
%K
ftv{2}(tv{3})
[
pl{4}n;
tv{5}:l{6};
tv{7}:l{8};
v{9}:v{10}l{11}a;
c(v{12}bv{13})
[
v{14}:v{15}v{16}a;
v{17}:v{18}l{19}a;
]
rv{20};
]
m
[
tv{21}:l{22};
pv{23}n;
v{24}:l{25};
pv{26}n;
tv{27}:v{28}(l{29});
pl{30};
pv{31}n;
tv{32}:v{33}(l{34},l{35})l{36}l{37}al{38}l{39}al{40}a@{5}l{41}l{42}al{43}aal{44}aaav{45}(l{46},l{47})a;
pl{48};
pv{49}n;
tv{50}:v{51}(l{52});
pl{53};
pv{54}n;
c(v{55}bl{56})
[
i(v{57}bl{58})
[
pv{59};
pl{60}n;
]
e
[
pv{61};
pl{62}n;
]
v{63}:v{64}l{65}a;
]
rl{66};
]
```

Рис. 6.1. Результат работы преобразования выражений

ГЛАВА 7. ГЕНЕРАЦИЯ КОДА

7.1. Принцип построения промежуточного кода

После успешной проверки синтаксическим и лексическим анализом, транслятор KPI-2016 переходит к заполнению структур –промежуточный код с помощью результата лексического анализатора. Структура представляет из себя четверку: <тип структуры>(<тип данных>, <операнд 1>, <операнд 2>, <опреанд 3>). Типы структур представлены в табл 7.1.

Табл. 7.1 Возможные типы таблицы структур

Типы	Описание
PLUS	Сложение
MINUS	Вычитание
MUL	Умножение
DIV	Деление.
IF	Условие.
ENDIF	Завершение условия.
CIRCLE	Цикл.
MORE	Больше.
LESS	Меньше.
PRINT	Вывод.
INIT	Инициализация переменной.
LEX_INIT	Инициализация лексемы.
PUSH	Добавить переменную в стек.
POP	Извлечь из стека верхний элемент
RETURN	Вернуть переменную из функции
BEGIN	Главная функция

7.2. Принцип построения объектного кода

Генерация объектного кода начинается после этапа построения промежуточного кода. На основе полученных структур идет вставка инструкций для Microsoft Micro Полученный результат запускается автоматически с помощью bat-файла, в котором прописаны все инструкции cmd. Автоматический запуск будет невозможен при выведении объектного кода не в корне транслятора KPI-2016.

7.3. Структура генератора промежуточного кода

Структура генератора промежуточного кода транслятора KPI-2016 предоставлена на рис.7.1

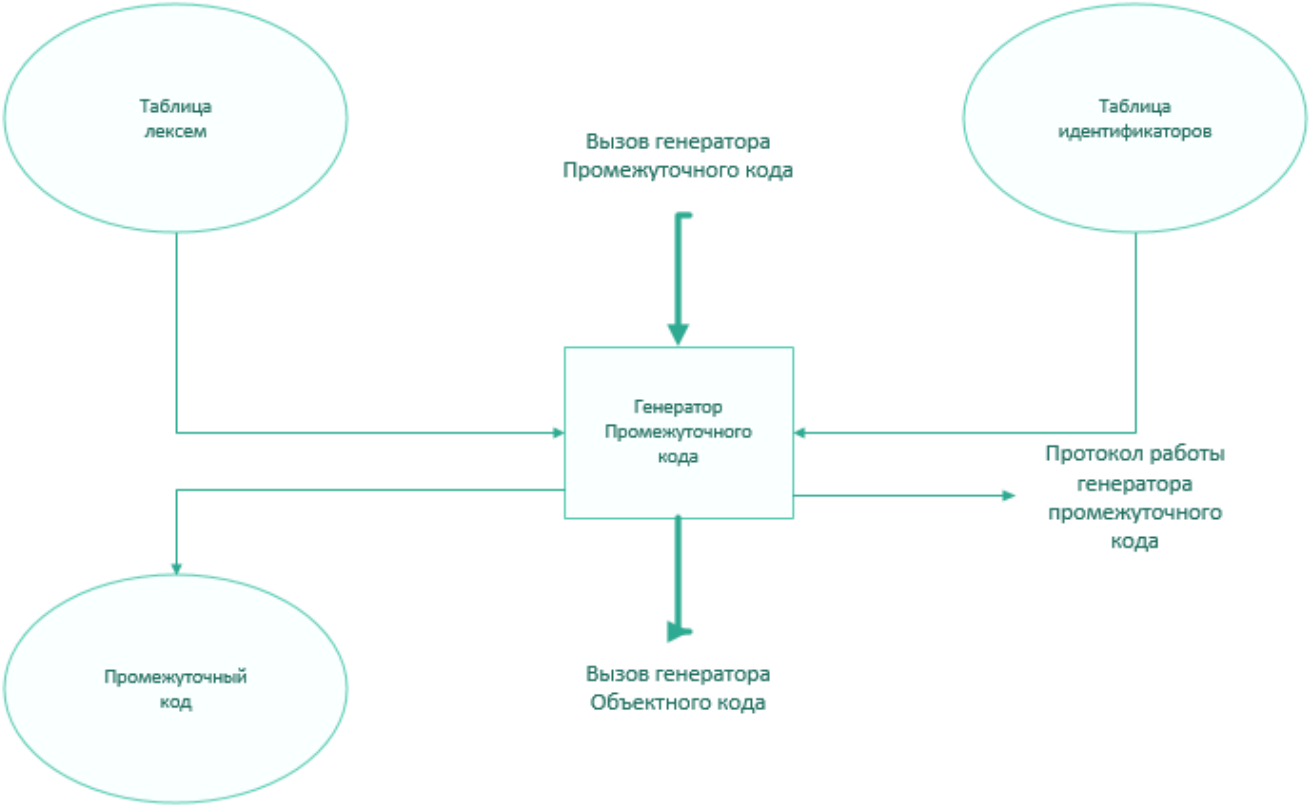

Рис.7.1. Структура генератора промежуточного кода

7.3. Структура генератора объектного кода

На рис.7.2 предоставлена генерация объектного кода транслятора KPI-2016

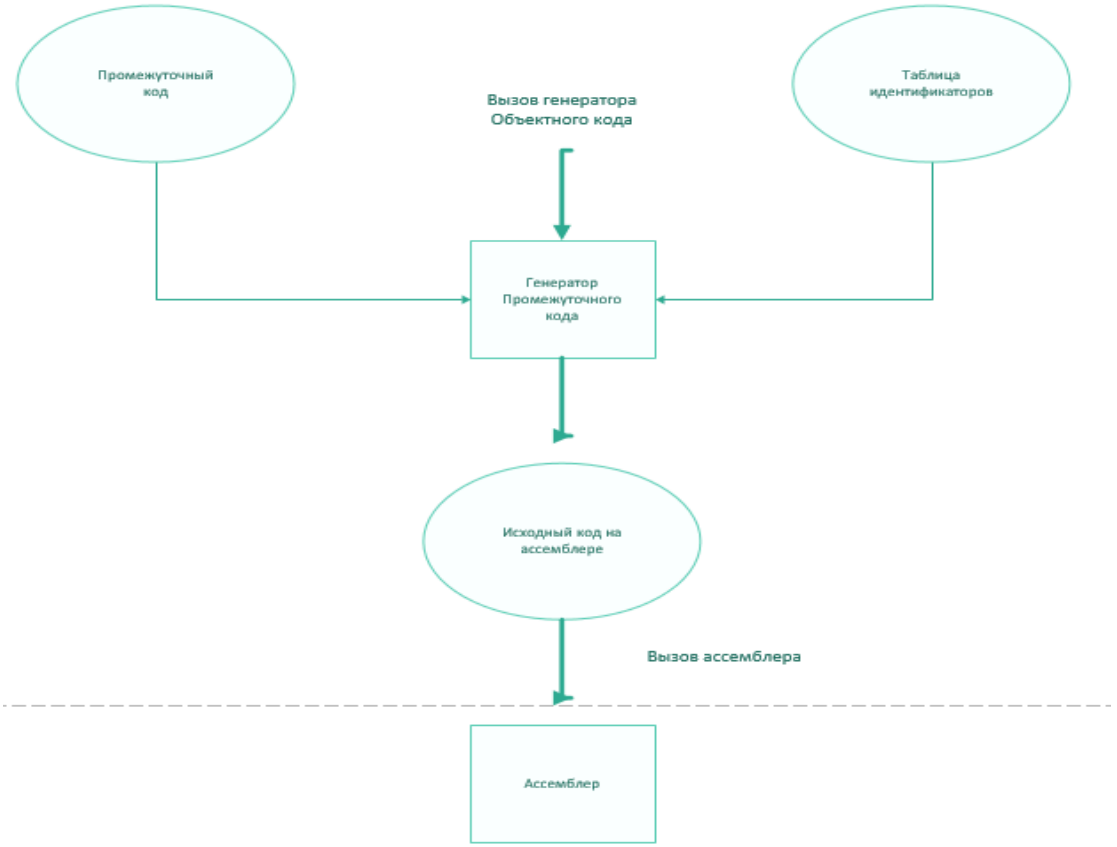

Рис.7.2. Генерация объектного кода

7.4. Принцип построения стандартной библиотеки

В трансляторе KPI-2016 предусмотрена явное подключение стандартной библиотеки. При встрече лексемы стандартной библиотеки, транслятор вставляет реализацию функций стандартной библиотеки. При повторном подключении стандартной библиотеки, работа транслятора завершается с ошибкой.

7.5. Основные структуры данных

Основной структурой данных, является структура NT, представленная на рис. 7.3.

```
namespace NT
{
    enum typeNible { PRINT, INIT, LEX_INIT, RETURN, PLUS, MINUS, MUL, PUSH, POP, DIV, IF, MORE, LESS, END_IF, ELSE, END_ELSE, CIRCLE,
    END_CIRCLE, FUNC, PARAM_INIT, PARAM_SECOND_INIT, START_FUNC, END_FUNC, END_MAIN_FUNC, BEGIN, FUNC_INVOKE, FANC_PARAM,PUSH_INVOKE,
    KPI_LIB, STR_PUSH};

    struct Entry
    {
        int TN;
        int TD;
        char* p2 = new char[50];
        char* p3 = new char[50];
        char* p4 = new char[50];
    };

    struct Nible
    {
        Nible() = default;
        Nible(int);
        int maxSize;
        int size;
        set<typeNible> component;
        Entry* table;
    };
```

Рис. 7.2. Структура данных генератора промежуточного кода.

7.6. Представление типов данных в памяти

Элементы таблицы идентификаторов расположены в разных сегментах языка ассемблера. Идентификаторы языка KPI-2016 размещены в сегменте данных. Литералы языка KPI-2016 размещены в сегменте констант. Соответствие между типами данных на языке KPI-2016 и на языке ассемблера приведены в табл. 7.1.

Табл. 7.1. Соответствие типов языка KPI-2016 и на языка ассемблера.

Тип на языке KPI-2016	Тип на языке ассемблера	Пояснение
int	dword	Беззнаковый, целочисленный тип данных
str	db	Массив из однобайтовых элементов

Под сегмент стека автоматически выделяется 4096 байта. Стек нужен для передачи параметров функции, а так же для вычисления выражений. Параметры типа Str передаются по адресу. Возврат значений через регистр eax.

7.7. Контрольный пример

В приложении С приведен полученные инструкции ассемблера для контрольного примера из п.1.25

ГЛАВА 8. ТЕСТИРОВАНИЕ ТРАНСЛЯТОРА

8.1. Контрольный пример

Демонстрация возможных ошибок транслятора KPI-2016 приведена в табл.8.2.

Табл. 8.1. Пример возможных ошибок транслятора KPI-2016

Ошибка	Пример	Описание
309	func str fact(int n) [sout `in function factorial` endl; int res:1;	В функции запрещено использовать тип данных STR
308	func str fact(int n) [sout `in function factorial` endl; int res:1/0;	Деление на ноль запрещено
304	str label: "KPI-2016 Translator"; sout label endl; str label: "new label"; sout label endl;	Нельзя переобъявлять переменные
315	int exp: xpow(2,5) + “fdfdfdfd”	Вычисления могут производиться над литералами/идентификаторами и типа INT
301	str 9label: "KPI-2016 Translator";	Идентификатор не должен содержать цифру

307	begin
	[….
	Return 0
	]
	Begin
	[
	….
	]
		Не должно быть более одной точки входа

ПРИЛОЖЕНИЕ А.
КЛЮЧЕВЫЕ СЛОВА И СООТВЕТСТВУЮЩИЕ ИМ КА

Ключевое слово	Конечный автомат
t	FST::FST fstTypeInteger("", 4, FST::NODE(1, FST::RELATION('i', 1)), FST::NODE(1, FST::RELATION('n', 2)), FST::NODE(1, FST::RELATION('t', 3)), FST::NODE());
t	FST::FST fstTypeString("", 4, FST::NODE(1, FST::RELATION('s', 1)), FST::NODE(1, FST::RELATION('t', 2)), FST::NODE(1, FST::RELATION('r', 3)), FST::NODE());
t	FST::FST fstTypeBool("", 5, FST::NODE(1, FST::RELATION('b', 1)), FST::NODE(1, FST::RELATION('o', 2)), FST::NODE(1, FST::RELATION('o', 3)), FST::NODE(1, FST::RELATION('l', 4)), FST::NODE());
l	FST::FST fstTrueLiteral("", 5, FST::NODE(1, FST::RELATION('t', 1)), FST::NODE(1, FST::RELATION('r', 2)), FST::NODE(1, FST::RELATION('u', 3)), FST::NODE(1, FST::RELATION('e', 4)), FST::NODE());
l	FST::FST fstFalseLiteral("", 6, FST::NODE(1, FST::RELATION('f', 1)), FST::NODE(1, FST::RELATION('a', 2)), FST::NODE(1, FST::RELATION('l', 3)), FST::NODE(1, FST::RELATION('s', 4)), FST::NODE(1, FST::RELATION('e', 5)), FST::NODE());
f	FST::FST fstKeyFunction("", 5, FST::NODE(1, FST::RELATION('f', 1)), FST::NODE(1, FST::RELATION('u', 2)), FST::NODE(1, FST::RELATION('n', 3)), FST::NODE(1, FST::RELATION('c', 4)), FST::NODE());
c	FST::FST fstCircleWhile("", 6, FST::NODE(1, FST::RELATION('w', 1)), FST::NODE(1, FST::RELATION('h', 2)), FST::NODE(1, FST::RELATION('i', 3)), FST::NODE(1, FST::RELATION('l', 4)), FST::NODE(1, FST::RELATION('e', 5)), FST::NODE());
i	FST::FST fstProvisionIf("", 3, FST::NODE(1, FST::RELATION('i', 1)), FST::NODE(1, FST::RELATION('f', 2)), FST::NODE());
m	FST::FST fstKeyBegin("", 6, FST::NODE(1, FST::RELATION('b', 1)), FST::NODE(1, FST::RELATION('e', 2)), FST::NODE(1, FST::RELATION('g', 3)), FST::NODE(1, FST::RELATION('i', 4)), FST::NODE(1, FST::RELATION('n', 5)), FST::NODE());
p	FST::FST fstKeySout("", 5, FST::NODE(1, FST::RELATION('s', 1)),

	FST::NODE(1, FST::RELATION('o', 2)), FST::NODE(1, FST::RELATION('u', 3)), FST::NODE(1, FST::RELATION('t', 4)), FST::NODE());
s	FST::FST fstKeySin("", 4, FST::NODE(1, FST::RELATION('s', 1)), FST::NODE(1, FST::RELATION('i', 2)), FST::NODE(1, FST::RELATION('n', 3)), FST::NODE());
r	FST::FST fstKeyReturn("", 7, FST::NODE(1, FST::RELATION('r', 1)), FST::NODE(1, FST::RELATION('e', 2)), FST::NODE(1, FST::RELATION('t', 3)), FST::NODE(1, FST::RELATION('u', 4)), FST::NODE(1, FST::RELATION('r', 5)), FST::NODE(1, FST::RELATION('n', 6)), FST::NODE());
a	FST::FST fstV("", 2, FST::NODE(5, FST::RELATION('-', 1), FST::RELATION('+', 1), FST::RELATION('/', 1), FST::RELATION('*', 1)), FST::NODE());
b	FST::FST fstMore("", 2, FST::NODE(2, FST::RELATION('>', 1), FST::RELATION('<', 1), FST::RELATION('=', 1)), FST::NODE());
=	FST::FST fstEqually("", 2, FST::NODE(1, FST::RELATION(':', 1)), FST::NODE());
(	FST::FST fstLeftThesis("", 2, FST::NODE(1, FST::RELATION('(', 1)), FST::NODE());
)	FST::FST fstRightThesis("", 2, FST::NODE(1, FST::RELATION(')', 1)), FST::NODE());
;	FST::FST fstSemiColon("", 2, FST::NODE(1, FST::RELATION(';', 1)), FST::NODE());
,	FST::FST fstComma("", 2, FST::NODE(1, FST::RELATION(',', 1)), FST::NODE());
[	FST::FST fstLeftBrace("", 2, FST::NODE(1, FST::RELATION('[', 1)), FST::NODE());
]	FST::FST fstRightBrace("", 2, FST::NODE(1, FST::RELATION(']', 1)), FST::NODE());
v	FST::FST fstIdentif("", 4, FST::NODE(78, FST::RELATION('a', 0), FST::RELATION('a', 1), FST::RELATION('a', 3), FST::RELATION('b', 0), FST::RELATION('b', 1), FST::RELATION('b', 3), FST::RELATION('c', 0), FST::RELATION('c', 1), FST::RELATION('c', 3), FST::RELATION('d', 0), FST::RELATION('d', 1), FST::RELATION('d', 3), FST::RELATION('e', 0), FST::RELATION('e', 1), FST::RELATION('e', 3), FST::RELATION('f', 0), FST::RELATION('f', 1), FST::RELATION('f', 3), FST::RELATION('g', 0), FST::RELATION('g', 1), FST::RELATION('g', 3), FST::RELATION('h', 0), FST::RELATION('h', 1), FST::RELATION('h', 3), FST::RELATION('i', 0), FST::RELATION('i', 1), FST::RELATION('i', 3), FST::RELATION('j', 0), FST::RELATION('j', 1), FST::RELATION('j', 3), FST::RELATION('k', 0), FST::RELATION('k', 1), FST::RELATION('k', 3), FST::RELATION('l', 0), FST::RELATION('l', 1), FST::RELATION('l', 3), FST::RELATION('m', 0), FST::RELATION('m', 1), FST::RELATION('m', 3), FST::RELATION('n', 0), FST::RELATION('n', 1), FST::RELATION('n', 3),

	FST::RELATION('o', 0), FST::RELATION('o', 1), FST::RELATION('o', 3), FST::RELATION('p', 0), FST::RELATION('p', 1), FST::RELATION('p', 3), FST::RELATION('q', 0), FST::RELATION('q', 1), FST::RELATION('q', 3), FST::RELATION('r', 0), FST::RELATION('r', 1), FST::RELATION('r', 3), FST::RELATION('s', 0), FST::RELATION('s', 1), FST::RELATION('s', 3), FST::RELATION('t', 0), FST::RELATION('t', 1), FST::RELATION('t', 3), FST::RELATION('u', 0), FST::RELATION('u', 1), FST::RELATION('u', 3), FST::RELATION('v', 0), FST::RELATION('v', 1), FST::RELATION('v', 3), FST::RELATION('w', 0), FST::RELATION('w', 1), FST::RELATION('w', 3), FST::RELATION('x', 0), FST::RELATION('x', 1), FST::RELATION('x', 3), FST::RELATION('y', 0), FST::RELATION('y', 1), FST::RELATION('y', 3), FST::RELATION('z', 0), FST::RELATION('z', 1), FST::RELATION('z', 3)), FST::NODE(30, FST::RELATION('0', 1), FST::RELATION('0', 2), FST::RELATION('0', 3), FST::RELATION('1', 1), FST::RELATION('1', 2), FST::RELATION('1', 3), FST::RELATION('2', 1), FST::RELATION('2', 2), FST::RELATION('2', 3), FST::RELATION('3', 1), FST::RELATION('3', 2), FST::RELATION('3', 3), FST::RELATION('4', 1), FST::RELATION('4', 2), FST::RELATION('4', 3), FST::RELATION('5', 1), FST::RELATION('5', 2), FST::RELATION('5', 3), FST::RELATION('6', 1), FST::RELATION('6', 2), FST::RELATION('6', 3), FST::RELATION('7', 1), FST::RELATION('7', 2), FST::RELATION('7', 3), FST::RELATION('8', 1), FST::RELATION('8', 2), FST::RELATION('8', 3), FST::RELATION('9', 1), FST::RELATION('9', 2), FST::RELATION('9', 3)), FST::NODE(52, FST::RELATION('a', 2), FST::RELATION('a', 3), FST::RELATION('b', 2), FST::RELATION('b', 3), FST::RELATION('c', 2), FST::RELATION('c', 3), FST::RELATION('d', 2), FST::RELATION('d', 3), FST::RELATION('e', 2), FST::RELATION('e', 3), FST::RELATION('f', 2), FST::RELATION('f', 3), FST::RELATION('g', 2), FST::RELATION('g', 3), FST::RELATION('h', 2), FST::RELATION('h', 3), FST::RELATION('i', 2), FST::RELATION('i', 3), FST::RELATION('j', 2), FST::RELATION('j', 3), FST::RELATION('k', 2), FST::RELATION('k', 3), FST::RELATION('l', 2), FST::RELATION('l', 3), FST::RELATION('m', 2), FST::RELATION('m', 3), FST::RELATION('n', 2), FST::RELATION('n', 3), FST::RELATION('o', 2), FST::RELATION('o', 3), FST::RELATION('p', 2), FST::RELATION('p', 3), FST::RELATION('q', 2), FST::RELATION('q', 3), FST::RELATION('r', 2), FST::RELATION('r', 3), FST::RELATION('s', 2), FST::RELATION('s', 3), FST::RELATION('t', 2), FST::RELATION('t', 3), FST::RELATION('u', 2), FST::RELATION('u', 3), FST::RELATION('v', 2), FST::RELATION('v', 3), FST::RELATION('w', 2), FST::RELATION('w', 3), FST::RELATION('x', 2), FST::RELATION('x', 3), FST::RELATION('y', 2), FST::RELATION('y', 3), FST::RELATION('z', 2), FST::RELATION('z', 3)), FST::NODE());
V	FST::FST fstFalseIdentif("", 3, FST::NODE(78, FST::RELATION('a', 0), FST::RELATION('A', 1), FST::RELATION('A', 2), FST::RELATION('b', 0), FST::RELATION('B', 1), FST::RELATION('B', 2), FST::RELATION('c', 0), FST::RELATION('C', 1), FST::RELATION('C', 2), FST::RELATION('d', 0), FST::RELATION('D', 1), FST::RELATION('D', 2), FST::RELATION('e', 0), FST::RELATION('E', 1), FST::RELATION('E', 2), FST::RELATION('f', 0), FST::RELATION('F', 1), FST::RELATION('F', 2), FST::RELATION('g', 0), FST::RELATION('G', 1), FST::RELATION('G', 2), FST::RELATION('h', 0), FST::RELATION('H', 1), FST::RELATION('H', 2), FST::RELATION('i', 0), FST::RELATION('I', 1), FST::RELATION('I', 2),

```
FST::RELATION('j', 0), FST::RELATION('J', 1), FST::RELATION('J', 2),
FST::RELATION('k', 0), FST::RELATION('K', 1), FST::RELATION('K', 2),
FST::RELATION('l', 0), FST::RELATION('L', 1), FST::RELATION('L', 2),
FST::RELATION('m', 0), FST::RELATION('M', 1), FST::RELATION('M', 2),
FST::RELATION('n', 0), FST::RELATION('N', 1), FST::RELATION('N', 2),
FST::RELATION('o', 0), FST::RELATION('O', 1), FST::RELATION('O', 2),
FST::RELATION('p', 0), FST::RELATION('P', 1), FST::RELATION('P', 2),
FST::RELATION('q', 0), FST::RELATION('Q', 1), FST::RELATION('Q', 2),
FST::RELATION('r', 0), FST::RELATION('R', 1), FST::RELATION('R', 2),
FST::RELATION('s', 0), FST::RELATION('S', 1), FST::RELATION('S', 2),
FST::RELATION('t', 0), FST::RELATION('T', 1), FST::RELATION('T', 2),
FST::RELATION('u', 0), FST::RELATION('U', 1), FST::RELATION('U', 2),
FST::RELATION('v', 0), FST::RELATION('V', 1), FST::RELATION('V', 2),
FST::RELATION('w', 0), FST::RELATION('W', 1), FST::RELATION('W', 2),
FST::RELATION('x', 0), FST::RELATION('X', 1), FST::RELATION('X', 2),
FST::RELATION('y', 0), FST::RELATION('Y', 1), FST::RELATION('Y', 2),
FST::RELATION('z', 0), FST::RELATION('Z', 1), FST::RELATION('Z', 2)),
FST::NODE(104,
FST::RELATION('A', 1), FST::RELATION('A', 2), FST::RELATION('a', 1),
FST::RELATION('a', 2),
FST::RELATION('B', 1), FST::RELATION('B', 2), FST::RELATION('b', 1),
FST::RELATION('b', 2),
FST::RELATION('C', 1), FST::RELATION('C', 2), FST::RELATION('c', 1),
FST::RELATION('c', 2),
FST::RELATION('D', 1), FST::RELATION('D', 2), FST::RELATION('d', 1),
FST::RELATION('d', 2),
FST::RELATION('E', 1), FST::RELATION('E', 2), FST::RELATION('e', 1),
FST::RELATION('e', 2),
FST::RELATION('F', 1), FST::RELATION('F', 2), FST::RELATION('f', 1),
FST::RELATION('f', 2),
FST::RELATION('G', 1), FST::RELATION('G', 2), FST::RELATION('g', 1),
FST::RELATION('g', 2),
FST::RELATION('H', 1), FST::RELATION('H', 2), FST::RELATION('h', 1),
FST::RELATION('h', 2),
FST::RELATION('I', 1), FST::RELATION('I', 2), FST::RELATION('i', 1),
FST::RELATION('i', 2),
FST::RELATION('J', 1), FST::RELATION('J', 2), FST::RELATION('j', 1),
FST::RELATION('j', 2),
FST::RELATION('K', 1), FST::RELATION('K', 2), FST::RELATION('k', 1),
FST::RELATION('k', 2),
FST::RELATION('L', 1), FST::RELATION('L', 2), FST::RELATION('l', 1),
FST::RELATION('l', 2),
FST::RELATION('M', 1), FST::RELATION('M', 2), FST::RELATION('m', 1),
FST::RELATION('m', 2),
FST::RELATION('N', 1), FST::RELATION('N', 2), FST::RELATION('n', 1),
FST::RELATION('n', 2),
FST::RELATION('O', 1), FST::RELATION('O', 2), FST::RELATION('o', 1),
FST::RELATION('o', 2),
FST::RELATION('P', 1), FST::RELATION('P', 2), FST::RELATION('p', 1),
FST::RELATION('p', 2),
FST::RELATION('Q', 1), FST::RELATION('Q', 2), FST::RELATION('q', 1),
FST::RELATION('q', 2),
FST::RELATION('R', 1), FST::RELATION('R', 2), FST::RELATION('r', 1),
FST::RELATION('r', 2),
FST::RELATION('S', 1), FST::RELATION('S', 2), FST::RELATION('s', 1),
FST::RELATION('s', 2),
FST::RELATION('T', 1), FST::RELATION('T', 2), FST::RELATION('t', 1),
FST::RELATION('t', 2),
FST::RELATION('U', 1), FST::RELATION('U', 2), FST::RELATION('u', 1),
FST::RELATION('u', 2),
FST::RELATION('V', 1), FST::RELATION('V', 2), FST::RELATION('v', 1),
FST::RELATION('v', 2),
```

	FST::RELATION('W', 1), FST::RELATION('W', 2), FST::RELATION('w', 1), FST::RELATION('w', 2), FST::RELATION('X', 1), FST::RELATION('X', 2), FST::RELATION('x', 1), FST::RELATION('x', 2), FST::RELATION('Y', 1), FST::RELATION('Y', 2), FST::RELATION('y', 1), FST::RELATION('y', 2), FST::RELATION('Z', 1), FST::RELATION('Z', 2), FST::RELATION('z', 1), FST::RELATION('z', 2)), FST::NODE());
l	FST::FST fstIntegerLiteral("", 2, FST::NODE(20, FST::RELATION('0', 0), FST::RELATION('0', 1), FST::RELATION('1', 0), FST::RELATION('1', 1), FST::RELATION('2', 0), FST::RELATION('2', 1), FST::RELATION('3', 0), FST::RELATION('3', 1), FST::RELATION('5', 0), FST::RELATION('4', 1), FST::RELATION('5', 0), FST::RELATION('5', 1), FST::RELATION('6', 0), FST::RELATION('6', 1), FST::RELATION('7', 0), FST::RELATION('7', 1), FST::RELATION('8', 0), FST::RELATION('8', 1), FST::RELATION('9', 0), FST::RELATION('9', 1)), FST::NODE());
l	FST::FST fstLiteral("", 4, FST::NODE(1, FST::RELATION('`', 1)), FST::NODE(266, FST::RELATION('0', 1), FST::RELATION('0', 2), FST::RELATION('1', 1), FST::RELATION('1', 2), FST::RELATION('2', 1), FST::RELATION('2', 2), FST::RELATION('3', 1), FST::RELATION('3', 2), FST::RELATION('4', 1), FST::RELATION('4', 2), FST::RELATION('5', 1), FST::RELATION('5', 2), FST::RELATION('6', 1), FST::RELATION('6', 2), FST::RELATION('7', 1), FST::RELATION('7', 2), FST::RELATION('8', 1), FST::RELATION('8', 2), FST::RELATION('9', 1), FST::RELATION('9', 2), FST::RELATION('a', 1), FST::RELATION('a', 2), FST::RELATION('A', 1), FST::RELATION('A', 2), FST::RELATION('b', 1), FST::RELATION('b', 2), FST::RELATION('B', 1), FST::RELATION('B', 2), FST::RELATION('c', 1), FST::RELATION('c', 2), FST::RELATION('C', 1), FST::RELATION('C', 2), FST::RELATION('d', 1), FST::RELATION('d', 2), FST::RELATION('D', 1), FST::RELATION('D', 2), FST::RELATION('e', 1), FST::RELATION('e', 2), FST::RELATION('E', 1), FST::RELATION('E', 2), FST::RELATION('f', 1), FST::RELATION('f', 2), FST::RELATION('F', 1), FST::RELATION('F', 2), FST::RELATION('g', 1), FST::RELATION('g', 2), FST::RELATION('G', 1), FST::RELATION('G', 2), FST::RELATION('h', 1), FST::RELATION('h', 2), FST::RELATION('H', 1), FST::RELATION('H', 2), FST::RELATION('i', 1), FST::RELATION('i', 2), FST::RELATION('I', 1), FST::RELATION('I', 2), FST::RELATION('j', 1), FST::RELATION('j', 2), FST::RELATION('J', 1), FST::RELATION('J', 2), FST::RELATION('k', 1), FST::RELATION('k', 2), FST::RELATION('K', 1), FST::RELATION('K', 2), FST::RELATION('l', 1), FST::RELATION('l', 2), FST::RELATION('L', 1), FST::RELATION('L', 2), FST::RELATION('m', 1), FST::RELATION('m', 2), FST::RELATION('M', 1), FST::RELATION('M', 2),

```
FST::RELATION('n', 1), FST::RELATION('n', 2), FST::RELATION('N', 1),
FST::RELATION('N', 2),
FST::RELATION('o', 1), FST::RELATION('o', 2), FST::RELATION('O', 1),
FST::RELATION('O', 2),
FST::RELATION('p', 1), FST::RELATION('p', 2), FST::RELATION('P', 1),
FST::RELATION('P', 2),
FST::RELATION('q', 1), FST::RELATION('q', 2), FST::RELATION('Q', 1),
FST::RELATION('Q', 2),
FST::RELATION('r', 1), FST::RELATION('r', 2), FST::RELATION('R', 1),
FST::RELATION('R', 2),
FST::RELATION('s', 1), FST::RELATION('s', 2), FST::RELATION('S', 1),
FST::RELATION('S', 2),
FST::RELATION('t', 1), FST::RELATION('t', 2), FST::RELATION('T', 1),
FST::RELATION('T', 2),
FST::RELATION('u', 1), FST::RELATION('u', 2), FST::RELATION('U', 1),
FST::RELATION('U', 2),
FST::RELATION('v', 1), FST::RELATION('v', 2), FST::RELATION('V', 1),
FST::RELATION('V', 2),
FST::RELATION('w', 1), FST::RELATION('w', 2), FST::RELATION('W', 1),
FST::RELATION('W', 2),
FST::RELATION('x', 1), FST::RELATION('x', 2), FST::RELATION('X', 1),
FST::RELATION('X', 2),
FST::RELATION('y', 1), FST::RELATION('y', 2), FST::RELATION('Y', 1),
FST::RELATION('Y', 2),
FST::RELATION('z', 1), FST::RELATION('z', 2), FST::RELATION('Z', 1),
FST::RELATION('Z', 2),
FST::RELATION('а', 1), FST::RELATION('а', 2), FST::RELATION('А', 1),
FST::RELATION('А', 2),
FST::RELATION('б', 1), FST::RELATION('б', 2), FST::RELATION('Б', 1),
FST::RELATION('Б', 2),
FST::RELATION('в', 1), FST::RELATION('в', 2), FST::RELATION('В', 1),
FST::RELATION('В', 2),
FST::RELATION('г', 1), FST::RELATION('г', 2), FST::RELATION('Г', 1),
FST::RELATION('Г', 2),
FST::RELATION('д', 1), FST::RELATION('д', 2), FST::RELATION('Д', 1),
FST::RELATION('Д', 2),
FST::RELATION('е', 1), FST::RELATION('е', 2), FST::RELATION('Е', 1),
FST::RELATION('Е', 2),
FST::RELATION('ё', 1), FST::RELATION('ё', 2), FST::RELATION('Ё', 1),
FST::RELATION('Ё', 2),
FST::RELATION('ж', 1), FST::RELATION('ж', 2), FST::RELATION('Ж', 1),
FST::RELATION('Ж', 2),
FST::RELATION('з', 1), FST::RELATION('з', 2), FST::RELATION('З', 1),
FST::RELATION('З', 2),
FST::RELATION('и', 1), FST::RELATION('и', 2), FST::RELATION('И', 1),
FST::RELATION('И', 2),
FST::RELATION('й', 1), FST::RELATION('й', 2), FST::RELATION('Й', 1),
FST::RELATION('Й', 2),
FST::RELATION('к', 1), FST::RELATION('к', 2), FST::RELATION('К', 1),
FST::RELATION('К', 2),
FST::RELATION('л', 1), FST::RELATION('л', 2), FST::RELATION('Л', 1),
FST::RELATION('Л', 2),
FST::RELATION('м', 1), FST::RELATION('м', 2), FST::RELATION('М', 1),
FST::RELATION('М', 2),
FST::RELATION('н', 1), FST::RELATION('н', 2), FST::RELATION('Н', 1),
FST::RELATION('Н', 2),
FST::RELATION('о', 1), FST::RELATION('о', 2), FST::RELATION('О', 1),
FST::RELATION('О', 2),
FST::RELATION('п', 1), FST::RELATION('п', 2), FST::RELATION('П', 1),
FST::RELATION('П', 2),
FST::RELATION('р', 1), FST::RELATION('р', 2), FST::RELATION('Р', 1),
FST::RELATION('Р', 2),
```

```
FST::RELATION('с', 1), FST::RELATION('с', 2), FST::RELATION('С', 1),
FST::RELATION('С', 2),
FST::RELATION('т', 1), FST::RELATION('т', 2), FST::RELATION('Т', 1),
FST::RELATION('Т', 2),
FST::RELATION('у', 1), FST::RELATION('у', 2), FST::RELATION('У', 1),
FST::RELATION('У', 2),
FST::RELATION('ф', 1), FST::RELATION('ф', 2), FST::RELATION('Ф', 1),
FST::RELATION('Ф', 2),
FST::RELATION('х', 1), FST::RELATION('х', 2), FST::RELATION('Х', 1),
FST::RELATION('Х', 2),
FST::RELATION('ц', 1), FST::RELATION('ц', 2), FST::RELATION('Ц', 1),
FST::RELATION('Ц', 2),
FST::RELATION('ч', 1), FST::RELATION('ч', 2), FST::RELATION('Ч', 1),
FST::RELATION('Ч', 2),
FST::RELATION('ш', 1), FST::RELATION('ш', 2), FST::RELATION('Ш', 1),
FST::RELATION('Ш', 2),
FST::RELATION('щ', 1), FST::RELATION('щ', 2), FST::RELATION('Щ', 1),
FST::RELATION('Щ', 2),
FST::RELATION('ъ', 1), FST::RELATION('ъ', 2), FST::RELATION('Ъ', 1),
FST::RELATION('Ъ', 2),
FST::RELATION('ы', 1), FST::RELATION('ы', 2), FST::RELATION('Ы', 1),
FST::RELATION('Ы', 2),
FST::RELATION('ь', 1), FST::RELATION('ь', 2), FST::RELATION('Ь', 1),
FST::RELATION('Ь', 2),
FST::RELATION('э', 1), FST::RELATION('э', 2), FST::RELATION('Э', 1),
FST::RELATION('Э', 2),
FST::RELATION('ю', 1), FST::RELATION('ю', 2), FST::RELATION('Ю', 1),
FST::RELATION('Ю', 2),
FST::RELATION('я', 1), FST::RELATION('я', 2), FST::RELATION('Я', 1),
FST::RELATION('Я', 2),
FST::RELATION('!', 1), FST::RELATION('!', 2),
FST::RELATION('?', 1), FST::RELATION('?', 2),
FST::RELATION(',', 1), FST::RELATION(',', 2),
FST::RELATION('.', 1), FST::RELATION('.', 2),
FST::RELATION(' ', 1), FST::RELATION(' ', 2)),
FST::NODE(1, FST::RELATION('`', 3)),
FST::NODE()
);
```

ПРИЛОЖЕНИЕ В.
ОСНОВНЫЕ СТРУКТУРЫ ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА

```
namespace LT                                                    // таблица лексем
enum PN { PN_DEF, PN_MINUS = 2, PN_PLUS = 2, PN_STAR = 3, PN_DIRSLASH = 3, PN_LEFTTHESIS = 1 };
struct Entry                                                    // строка таблицы лексем
{
char lexema;                                        // лексема
int sn;                                                 // номер строки в исходном тексте
int idxTI;                                              // индекс в таблице идентификаторов иди
LT_TI_NULLIDX
int priority;                                       // приоритет
int braceType;
int automat;
};

struct LexTable                                                 // экземпляр таблиццы лексем
{
int maxsize;                                        // емкость таблицы лексем < LT_MAXSIZE
int size;                                               // текущий размер таблицы лексем <
maxsize
Entry* table;                                       // массив строк таблицы лексем
};

namespace IT                                                                    //
таблица идетификаторов
{
enum IDDATATYPE { DT_NO, DT_INT, DT_STR, DT_BOOL };                             //      типы
данных идентификатов: integer, stringяяяя
enum IDTYPE { T_NO, T_VAR, T_FUNC, T_FUNC_P, T_FUNC_I, T_FUNC_IP, T_LITERAL, T_CONDITION,
T_CONDITION_P, T_ELSE, T_CIRCLE, T_CIRCLE_P,
T_MAINFUNC};                        // типы идентификиторов: переменная, функция, параметр,
литерал, указатель

struct Entry                                                                    //      строка
таблицы идентификаторов
{
int                 idfirstLE;                                                  //      индекс
первой строки в таблице лексем
char          *id = new char[ID_MAXSIZE];                   //      идентификатор      (автомат.
усекаеться до ID_MAXSIZE)
char          *prefId = new char[ID_MAXSIZE];
bool          pointer = false;
int                 iddatatype;                                                 // тип данных
int                 idtype;                                                         // тип
идентификатора
int                 paramCount;
struct
{
char *val = new char[255];                          // значение integer
int len;
}value;                                                                         //
значение идентификатора
Entry();
};
```

ПРИЛОЖЕНИЕ С.
ИСХОДНЫЙ КОД ПОЛУЧЕННЫЙ ДЛЯ АССЕМБЛЕРА

```
.586
.model flat, stdcall
includelib kernel32.lib
ExitProcess PROTO,
x:DWORD
WaitMsg PROTO
Crlf PROTO
 WriteInt PROTO
Str_copy PROTO :DWORD, :DWORD
SetConsoleTitleA PROTO :DWORD
WriteString PROTO

.data
bufmath sdword ?
factres dword 0
factcount dword 0
beginlabel db 0,0
beginfac dword 0
beginexp dword 0
beginexpTwo dword 0

.const

cname db 'KPI-2016 Compiler RC1',0
L01 db 'in function factorial',0
L02 dword 1
L03 dword 25
L04 dword 1
L05 dword 1
L06 db 'KPI-2016 Translator',0
L07 db 'new label',0
L08 dword 3
L09 db 'factorial: ',0
L10 dword 2
L11 dword 5
L12 dword 12
L13 dword 12
L14 dword 41
L15 dword 31
L16 dword 10
L17 dword 32
```

```
L18 dword 12
L19 dword 20
L20 dword 100
L21 dword 2
L22 dword 5
L23 db 'exp: ',0
L24 dword 25
L25 db 'sqrt(25): ',0
L26 dword 10
L27 dword 8
L28 db ' below 8',0
L29 db ' abow 8',0
L30 dword 1
L31 dword 0

.code
kpixpow PROC USES ebx edx xpowx : SDWORD, xpowy : SDWORD
mov bufmath, 0
fild xpowy
fild xpowx
fyl2x
fld st(0)
frndint
fxch st(1)
fsub st(0), st(1)
f2xm1
fld1
faddp st(1), st
fscale
fistp bufmath
mov eax, bufmath
ret
kpixpow ENDP

kpisqrt PROC USES ebx edx sqrtx : SDWORD
mov bufmath, 0
finit
fild sqrtx
fsqrt
fist bufmath
mov EAX, bufmath
ret
kpisqrt ENDP
```

```
kpifact PROC uses ebx ebp esi edi factn:dword
mov EDX, offset L01
invoke WriteString
invoke Crlf
push L02
pop factres
push L03
pop factcount
push factn
push L04
pop eax
pop ebx
add eax, ebx
push eax
pop factn
@circle1:
mov eax, factn
cmp eax, factcount
jle @if1
push factres
push factcount
pop eax
pop ebx
imul eax, ebx
push eax
pop factres
push factcount
push L05
pop eax
pop ebx
add eax, ebx
push eax
pop factcount
jmp @circle1
@if1:
mov EAX, factres
ret
kpifact ENDP

main PROC

invoke SetConsoleTitleA, offset cname
invoke Str_copy, offset L06, offset beginlabel
```

```
mov EDX, offset beginlabel
invoke WriteString
invoke Crlf
invoke Str_copy, offset L07, offset beginlabel
mov EDX, offset beginlabel
invoke WriteString
invoke Crlf
invoke kpifact, L08
push eax
pop beginfac
mov EDX, offset L09
invoke WriteString
mov EAX, beginfac
invoke WriteInt
invoke Crlf
invoke kpixpow, L10, L11
push eax
push L12
push L13
pop ebx
pop eax
cdq
div ebx
push eax
push L14
push L15
pop eax
pop ebx
imul eax, ebx
push eax
push L16
pop ebx
pop eax
sub eax, ebx
push eax
push L17
push L18
pop ebx
pop eax
sub eax, ebx
push eax
push L19
pop eax
```

```
pop ebx
imul eax, ebx
push eax
pop eax
pop ebx
imul eax, ebx
push eax
push L20
pop ebx
pop eax
cdq
div ebx
push eax
pop ebx
pop eax
sub eax, ebx
push eax
pop eax
pop ebx
add eax, ebx
push eax
invoke kpixpow, L21, L22
push eax
pop eax
pop ebx
add eax, ebx
push eax
pop beginexp
mov EDX, offset L23
invoke WriteString
mov EAX, beginexp
invoke WriteInt
invoke Crlf
invoke kpisqrt, L24
push eax
pop beginexpTwo
mov EDX, offset L25
invoke WriteString
mov EAX, beginexpTwo
invoke WriteInt
invoke Crlf
@circle2:
mov eax, L26
```

```
cmp eax, beginexpTwo
jle @if2
mov eax, L27
cmp eax, beginexpTwo
jge @if3
mov EAX, beginexpTwo
invoke WriteInt
mov EDX, offset L28
invoke WriteString
invoke Crlf
jmp @endElse3
@if3:
mov EAX, beginexpTwo
invoke WriteInt
mov EDX, offset L29
invoke WriteString
invoke Crlf
@endElse3:
push beginexpTwo
push L30
pop eax
pop ebx
add eax, ebx
push eax
pop beginexpTwo
jmp @circle2
@if2:
mov EAX, L31
ret
main ENDP
END main
```

ПРИЛОЖЕНИЕ D.
РАСПЕЧАТКА ДЕРЕВА РАЗБОРА

```
0   : S->%KS
2   : S->ftv(F)[N]S
6   : F->tv
10  : N->pPn;N
11  : P->l
14  : N->tv:E;N
17  : E->l
19  : N->tv:E;N
22  : E->l
24  : N->v:E;N
26  : E->vM
27  : M->aE
28  : E->l
30  : N->c(K)[C]N
32  : K->vbv
37  : C->v:E;C
39  : E->vM
40  : M->aE
41  : E->v
43  : C->v:E;
45  : E->vM
46  : M->aE
47  : E->l
50  : N->rv;
54  : S->m[N]
56  : N->tv:E;N
59  : E->l
61  : N->pPn;N
62  : P->v
65  : N->v:E;N
67  : E->l
69  : N->pPn;N
70  : P->v
73  : N->tv:E;N
76  : E->v(W)
78  : W->l
81  : N->pP;N
82  : P->l
84  : N->pPn;N
85  : P->v
88  : N->tv:E;N
```

91 : E->v(W)M
93 : W->l,W
95 : W->l
97 : M->aE
98 : E->(E)M
99 : E->lM
100 : M->aE
101 : E->lM
102 : M->aE
103 : E->(E)M
104 : E->lM
105 : M->aE
106 : E->lM
107 : M->aE
108 : E->l
110 : M->aE
111 : E->(E)M
112 : E->(E)M
113 : E->lM
114 : M->aE
115 : E->l
117 : M->aE
118 : E->l
120 : M->aE
121 : E->l
123 : M->aE
124 : E->v(W)
126 : W->l,W
128 : W->l
131 : N->pP;N
132 : P->l
134 : N->pPn;N
135 : P->v
138 : N->tv:E;N
141 : E->v(W)
143 : W->l
146 : N->pP;N
147 : P->l
149 : N->pPn;N
150 : P->v
153 : N->c(K)[C]N
155 : K->vbl
160 : C->i(K)[C]e[C]C

162 : K->vbl
167 : C->pE;C
168 : E->v
170 : C->pEn;
171 : E->l
177 : C->pE;C
178 : E->v
180 : C->pEn;
181 : E->l
185 : C->v:E;
187 : E->vM
188 : M->aE
189 : E->l
192 : N->rl;

ЛИТЕРАТУРА

1. Assembly Language for x86 Processors, 7th edition, Kip Irvine, 2014
2. Assembly Language for Intel-Based Computers, 5th edition, Kip Irvine, 2007
3. ru.wikipedia.org/wiki/Обратная_польская_запись
4. ru.wikipedia.org/wiki/ASCII