2. Лабораторная работа № 2.
Перевод исходной программы в обратную польскую запись

Оглавление

2. Лабораторная работа № 2. Перевод исходной программы в обратную польскую запись 1

2.1. Понятие обратной польской записи 2

2.2. Алгоритм Дейкстры 2

2.3. Перевод выражений, содержащих переменные с индексами, в ОПЗ 4

2.4. Перевод в ОПЗ выражений, содержащих указатели функций 7

2.5. Перевод оператора присваивания в ОПЗ 10

2.6. Перевод оператора безусловного перехода и меток в ОПЗ 11

2.7. Перевод условного оператора в ОПЗ 13

2.8. Перевод описаний переменных и процедурных блоков в ОПЗ 16

Таблица переходов МП-автомата 22

2.9. Комплексный пример перевода исходной программы в ОПЗ 25

2.10. Задание к лабораторной работе 28

2.1. Понятие обратной польской записи

Обратная польская запись (ОПЗ) применяется при трансляции как один из способов втрутеннего представления синтаксической структуры программы.ОПЗ представляет собой одну из форм записи выражений и операторов, отличительной особенностью которой является расположение аргументов (операндов) перед операцией (оператором).

Например, выражение, записанное в обычной (прямой) скобочной записи,

```
(a+d)/c+b*(e+d),
```

в ОПЗ имеет следующее представление:

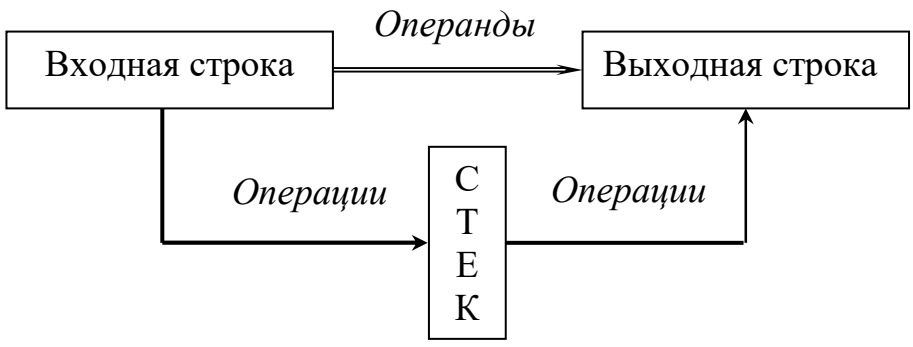
```
ad+c/bed+*+.
```

Обратная польская запись получила широкое распространение благодаря своему основному преимуществу: ОПЗ может быть вычислена за один просмотр цепочки слева направо, который часто называют проходом.

2.2. Алгоритм Дейкстры

Исследованию формальных способов преобразования арифметических и логических выражений в ОПЗ посвящены многочисленные исследования, однако в практике системного программирования наибольшее распространение получили способы преобразования на основе алгоритма Дейкстры.

В основе алгоритма лежит использование автомата с магазинной памятью для преобразования прямой записи в обратную. Суть алгоритма Дейкстры можно представить следующим рисунком:



Из этого рисунка следует, что на вход алгоритма по лексемам поступает исходное выражение. Операнды исходного выражения пропускаются на выход и формируют выходную строку. Операции обрабатываются по определенным правилам на основе стека.

Для реализации такой обработки известное в системном программировании понятие стека (магазин) используется также в алгоритме Дейкстры для размещения в нем операций. При этом предварительно каждой операции приписывается свой приоритет на основе таблицы приоритетов, которая приведена ниже (табл. 2.1).

Таблица 2.1

Входной элемент	Приоритет
(	0
)	1
V	2
&	3
¬	4
Отношения	5
+ -	6
* /	7
Возведение в степень	8

С учетом этой таблицы сформулируем правила работы со стеком.

Если рассматриваемый символ является операцией, то с ним производятся следующие действия путем сравнения его приоритета с приоритетом верхнего символа стека:

- Если стек пуст, операция заносится в стек.
- Если в стеке верхний элемент (операция) имеет меньший приоритет, то рассматриваемая операция проталкивается в стек.

- Если в стеке верхний элемент (операция) имеет больший или равный приоритет, то из стека в выходную строку выталкиваются все элементы (операции) до тех пор, пока не встретится операция с приоритетом меньше, чем у рассматриваемой операции, после чего операция из входной строки проталкивается в стек.
- Если входной символ "(", то он всегда проталкивается в стек. Если входной символ ")", то он выталкивает из стека все символы операций до ближайшей "(". Сами скобки взаимно уничтожаются и в выходную строку не попадают.

Приведем пример перевода выражения в ОПЗ:

Выходная строка	a		b	+		5	-	<	2		c	-	1		q	+	=	&
С Т Е К		+		<	-		&			-		=		+				
					<					&		&		=				
														&				
Входная строка	a	+	b	<	-	5	&		2	-	c	=	1	+	q	■		

2.3. Перевод выражений, содержащих переменные с индексами, в ОПЗ

В языках программирования используются массивы и их элементы, то есть переменные с индексами. Сами массивы могут быть как одномерными, так и многомерными. В действительности в памяти ЭВМ любые массивы размещаются только в виде одномерных массивов, поэтому при трансляции программ необходимо решать проблему отображения многомерного массива в одномерный и организации доступа к его элементам.

Для решения этой проблемы вводится операция *вычисления адреса элемента массива* (АЭМ).

Операция АЭМ для *k*-мерного массива имеет *k +1* операнд:

- имя массива;
- индексы массива,

и в ОПЗ представляется следующим образом:

<операнд1><операнд2> ... <операнд k+1> k+1 АЭМ,

где <операнд1> - имя массива, <операнд2>,..,<операндk+1> - индексы массива, k+1 – счетчик операндов.

Например, ОПЗ выражения

(a+b[i+20,j])*c+d

будет иметь вид

a b i 20 + j 3АЭМ + c * d +.

В этой ОПЗ операнды операции АЭМ подчеркнуты.

Для реализации алгоритма Дейкстры с учетом операции АЭМ, ее необходимо ввести в таблицу приоритетов, которая теперь выглядит следующим образом (табл. 2.2).

Таблица 2.2

Входной элемент	Приоритет
([АЭМ	0
,)]	1
V	2
&	3
¬	4
Отношения	5
+ -	6
* /	7
Возведение в степень	8

Правила работы со стеком при переводе выражений с индексами в ОПЗ принимают следующий вид:

- обычные знаки операций обрабатываются так же, как и в предыдущем случае;
- с открытием массива (идентификатор и «[») в стек заносится операция АЭМ со значением счетчика операндов равным 2;
- знак «,» выталкивает из стека все символы до ближайшего АЭМ и наращивает счетчик операндов на 1;
- закрывающая скобка «]» выталкивает из стека операцию АЭМ со значением счетчика операндов, но сама в выходную строку не попадает.

Перевод рассмотренного выше выражения в ОПЗ по алгоритму Дейкстры имеет следующий вид:

Выходная строка		a		b		i		20	+	j	3АЭМ	+		c	*	d	+
С Т Е К	(		+		2АЭМ		+		3АЭМ		+		*		+		
			(		+		2АЭМ		+		(						
						(	+		(								
							(										
Входная строка	(	a	+	b	[	i	+	20	,	j	]	)	*	c	+	d	■

Примечание. При трансляции исходной программы в объектный код на машинном языке адреса обращения к элементам массива должны быть правильно оформлены. Для этого необходимо вычислить так называемую функцию упорядочивания, которая устанавливает взаимно-однозначное соответствие между многомерным массивом и отображающим его одномерным массивом. Но поскольку в программах может использоваться множество массивов, то нужно еще и выбрать соответствующую данному массиву функцию упорядочивания. Обычно вся эта информация, касающаяся каждого массива, собирается транслятором в определяющий вектор массива, а сами определяющие вектора размещаются во временную таблицу транслятора. Таким образом, обращение к соответствующему элементу многомерного мас-

сива связано с обработкой определяющих векторов и вычислением функции упорядочивания, что является семантической процедурой, скрывающимся за символом операции АЭМ.

2.4. Перевод в ОПЗ выражений, содержащих указатели функций

В арифметических и логических выражениях могут использоваться функции.

Введем операцию *функция*, которая так же, как и АЭМ, имеет k операндов и записывается в ОПЗ в виде:

<операнд1><операнд2> ... <операндk> k Ф,

где <операнд1> - имя функции, а <операнд2>,..,<операндk> - операнды функции.

Пример. Обратная польская запись выражения

```
y-f(x,z,y+2)
```

имеет вид

y f x z y 2 + 4 Ф - .

Таблица приоритетов для алгоритма Дейкстры модифицируется следующим образом (табл. 2.3).

Таблица 2.3

Входной элемент	Приоритет
([АЭМ Ф	0
,)]	1
V	2
&	3
¬	4
Отношения	5
+ -	6
* /	7
Возведение в степень	8

При обработке оператора "Функция" необходимо прежде всего различить роли открывающей круглой скобки. Она может:

- быть обычной группировочной скобкой в выражении;
- открывать список фактических параметров функции.

Для решения этой проблемы в тривиальный автомат с магазинной памятью, имеющий одно состояние 0, вводится дополнительной состояние 1. Алгоритм работы со стеком изменяется и дополняется следующим образом:

- при поступлении на вход идентификатора автомат переходит в состояние 1 и помещает идентификатор во входную сроку;
- в состоянии 1:
    - при поступлении на вход автомата символа '(' в стек помещается оператор Ф со значением счетчика, равным 1, и автомат остается в состоянии 1;
    - если в вершине стека находится не оператор Ф, то при поступлении на вход символа, отличного от '(', автомат переходит в состояние 0 и обрабатывает лексему по обычному алгоритму;
    - если в вершине стека находится оператор Ф, при поступлении на вход символа, отличного от ')', счетчик операндов увеличивается на 1, автомат переходит в состояние 0 и обрабатывает лексему по обычному алгоритму;
    - если в вершине стека находится оператор Ф, при поступлении на вход символа ')', из стека в выходную строку выталкивается оператор Ф со счетчиком, равным 1, автомат переходит в состояние 0 и получает следующий символ из входной строки.
- в состоянии 0:
    - по входному символу ',' из стека выталкивается все до последнего оператора Ф и значение счетчика наращивается на 1;

- по входному символу ')' из стека выталкивается все до последнего оператора Ф, оператор Ф выталкивается в выходную строку с текущим значением счетчика.

Сформулированные словесно правила работы со стеком для состояния 1 можно представить в виде таблицы переходов автомата с магазинной памятью, Таблица описывает действия в одном состоянии автомата. В ней каждой паре (лексема на входе, лексема в вершине стека) соответствует набор действий:

1. Изменение состояния;
2. Действие со стеком (Втолкнуть, Вытолкнуть, Заменить);
3. Действие с входной строкой (Держать, Сдвиг);
4. Действие с выходной строкой (Выдать).

Если какое-либо из действий отсутствует (например, не изменяется состояние, или ничего не выдается на выход), то его можно опускать.

Состояние 1		Входная лексема		
		(	)	Другая
Лексема в вершине стека	Ф 1	Состояние 0 Заменить (Ф 2) Держать	Вытолкнуть Сдвиг Выдать (Ф 1)	Состояние 0 Заменить (Ф 2) Держать
	Другая	Втолкнуть (Ф 1) Сдвиг	Состояние 0 Держать	Состояние 0 Держать

Например, для приведенного выражения процесс перевода в ОПЗ имеет вид:

Выходная строка	y		f		x		z		y		2	+	4Ф	-
С Т Е К		-		1Ф -	2Ф	3Ф -		4Ф -		+ 4Ф -		4Ф -	-	
Входная строка	y	-	f	(	x	,	z	,	y	+	2	)		■
Состояние	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Примечание. При трансляции исходной программы в объектный код на машинном языке функции должны быть оформлены как подпрограммы. Обращение к данным подпрограммам строится на основе обработки операции *функция* в ОПЗ. Поэтому операция *функция* является символом в ОПЗ, а за этим символом скрывается ряд семантических процедур, связанных с правильным обращением к соответствующим подпрограммам. Обычно поиск требуемой подпрограммы организовывается следующим образом: заводится таблица имен функций и в этой таблице указываются адреса, по которым располагаются эти подпрограммы. При распознавании оператора функции в ОПЗ из нее извлекается имя подпрограммы и ее операнды, а далее в объектном коде вставляются стандартные процедуры обращения к подпрограмме.

2.5. Перевод оператора присваивания в ОПЗ

Формат оператора присваивания имеет вид:

```
<переменная> := <выражение>;
```

а его ОПЗ представляется следующим образом:

```
<переменная> <выражение> :=
```

Для включения в ОПЗ оператора присваивания необходимо решить вопрос о месте этого оператора в таблице приоритетов. При выборе приоритета будем руководствоваться следующими соображениями:

- приоритет оператора присваивания должен быть выше, чем у любой арифметической или логической операции, чтобы знаки не могли вытолкнуть оператор присваивания из стека;
- приоритет должен быть меньше, чем у знака ';'.

С учетом этого, таблица приоритетов операций принимает следующий вид (табл. 2.4).

Таблица 2.4

Входной элемент	Приоритет
([АЭМ Ф	0
, ;)]	1
:=	2
V	3
&	4
¬	5
Отношения	6
+ -	7
* /	8
Возведение в степень	9

2.6. Перевод оператора безусловного перехода и меток в ОПЗ

Формат оператора безусловного перехода имеет вид:

```
GOTO <метка>;
```

а его ОПЗ представляется следующим образом:

```
<метка> БП
```

При установлении приоритета оператора безусловного перехода руководствуются такими соображениями, как и в случае оператора присваивания. Поэтому приоритеты этих операторов одинаковы.

Для перевода в ОПЗ меток из исходной программы вводится *операция метки*, которую для простоты обозначают символом ':'. Эта операция имеет наименьший приоритет.

Таблица приоритетов с учетом оператора безусловного перехода и операции метки принимает следующий вид (табл. 2.5):

Таблица 2.5

Входной элемент	Приоритет
([АЭМ Ф	0
, ;)]	1
:= GOTO	2
V	3
&	4
¬	5
Отношения	6
+ -	7
* /	8
Возведение в степень	9
:	10

Рассмотрим перевод в ОПЗ фрагмента программы, содержащего выше рассмотренные операторы и операции. Фрагмент исходной программы имеет следующий вид:

```
R1:=A+B;
GOTO M1;
R2:=C*D-A;
M1:
```

Перевод данного фрагмента программы в ОПЗ по алгоритму Дейкстры представлен ниже:

Выходная строка	R1		A		B	+	:=		M1	БП	R2		C		D	*	A	-	:=	M1		:
С Т Е К		:=		+		:=		GOTO					:=	*		-		:=			:	
				:=										:=		:=						
Входная строка	R1	:=	A	+	B	;		GOTO	M1	;	R2	:=	C	*	D	-	A	;		M1	:	■

2.7. Перевод условного оператора в ОПЗ

Существует два вида условного оператора:

- полный

```
IF условие THEN оператор1 ELSE оператор2 ;
```

- неполный

```
IF условие THEN оператор ;
```

В отличие от простых операторов и выражений, где порядок выполнения действий определяется старшинством операций или скобками и в процессе выполнения программы этот порядок не меняется, в условных выражениях он изменяется в зависимости от значения *условия*.

Таким образом, понятие условного оператора должно быть также введено в ОПЗ представления исходной программы. Для этого вводятся так называемые динамические деревья.

Рассмотрим ряд понятий и особенностей для динамических деревьев:

- узлы и ветви деревьев могут помечаться метками;
- пустой узел - это такой узел, в котором отсутствует операция. Из пустого узла может исходить любое количество ветвей. Пустой узел вводится для объединения нескольких последовательных действий или разветвления некоторого вычислительного процесса;
- условный оператор перехода по значению 'ложь'. Оператор имеет следующее представление:

<условие> <метка> УПЛ

- оператор безусловного перехода:

<метка> БП

С учетом введенных понятий ОПЗ оператора

```
IF a>b THEN z:=x ELSE z:=y;
```

принимает следующий вид:

```
ab>M1 УПЛ z x := M2 БП M1: z y := M2:
```

Для обобщения алгоритма Дейкстры на условный оператор требуется модифицировать таблицу приоритетов, исходя из следующих соображений. Нетрудно заметить, что IF играет роль открывающей скобки, а THEN и ELSE - запятых. Теперь с учетом данного обстоятельства модифицированная таблица приоритетов примет следующий вид (табл. 2.6):

Таблица 2.6

Входной элемент	Приоритет
IF ([АЭМ Ф	0
, ;)] THEN ELSE	1
:= GOTO	2
V	3
&	4
¬	5
Отношения	6
+ -	7
* /	8
Возведение в степень	9
:	10

Для трансляции условного оператора необходимо использовать рабочие метки (в отличие от «пользовательских», используемых программистом).

Формат таких меток примем «Mi», где i – номер метки. Для того, чтобы обеспечить уникальность номеров меток, введем счетчик меток *i*, который в начале трансляции программы инициализируем значением 1.

Обработка символов операций будет дополнена правилами обработки ключевых слов условных выражений следующим образом:

- символ IF из входной строки заносится в стек;
- символ THEN выталкивает в выходную строку все операции из стека до ближайшего IF. В стеке к символу IF добавляется рабочая метка Mi и после этого в выходную строку записывается часть Mi УПЛ. Счетчик меток увеличивается на 2;
- ELSE выталкивает из стека все символы до ближайшего IF Mi. В выходную строку выталкивается часть Mi+1 БП Mi: . Метка при IF заменяется на новую метку Mi.
- Символ «;» указывает на конец условного оператора и выталкивает из стека все символы до ближайшего IF Mi, при этом сам IF в стеке уничтожается, а в выходную строку помещается метка Mi: .

Например, для приведенного выражения процесс перевода в ОПЗ имеет вид:

Выходная строка		a		b	>	M1 УПЛ	z		x	:=	M2 БП M1:	z		y	:=	M2:	
С Т Е К	IF		>			IF M1		:=		IF M1	IF M2		:=		IF M2		
			IF					IF M1					IF M2				
Входная строка	IF	a	>	b	THEN		z	:=	x	ELSE		z	:=	y	;		■

Примечание. Описанная выше обработка условного оператора связана с правильной организацией передачи управления внутри объектного кода, что и составляет внутреннюю семантику обработки условного оператора в ОПЗ.

Приведенный алгоритм правильно обрабатывает как полный, так и неполный условный оператор. В последнем случае будет пропущено действие, выполняемое при появлении лексемы ELSE.

Однако следует заметить, что в разных языках программирования принято различное употребление ограничителей ';' в условных операторах. Соответственно, функции ';' как концевого маркера оператора являются в разных языках различными. Например, в языке программирования Паскаль, как и в приведенном примере, лексема ';' всегда закрывает условный оператор. В этом случае этот знак должен вытолкнуть из стека IF и расставить соответствующие метки. В языке Си, напротив, лексема ';' завершает каждый оператор внутри условного оператора и, следовательно, не может выталкивать IF из стека. В этом случае вопрос о завершении IF решается с помощью дополнительного состояния, которое должно ответить на вопрос, появится ли лексема ELSE после лексемы ';', которая завершила оператор части THEN (аналогично обработке вызова функции).

2.8. Перевод описаний переменных и процедурных блоков в ОПЗ

В компиляторах обработка описаний переменных необходима для:

- выявления типов данных;
- подготовки информации для распределения памяти;
- выделения в объектной программе места для размещения данных.

Очевидно, что распределение памяти в объектной программе всегда должно предшествовать ее генерации. Поскольку исходная программа может содержать вложенные процедуры, то область действия идентификаторов, описанных во вложенных процедурах, является локализованной в пределах процедуры. Компилятор должен учитывать это обстоятельство и располагать информацией о начале и конце процедуры.

Приведем пример обработки описаний данных на базе описанного выше подмножества языка PL/1.

В этом языке программирования имеется несколько типов данных:

DEC FLOAT – десятичное число с плавающей точкой;

DEC FIXED – десятичное число с фиксированной точкой;

BIN FIXED – двоичное число с фиксированной точкой;

CHAR – символьная строка.

Сопоставим этим конструкциям языка в ОПЗ *операцию типа*, формат которой имеет вид:

Список переменных k ТИП,

где *k* – количество описываемых переменных;

$$\text{ТИП} = \begin{cases} \text{DFT} \\ \text{DFD} \\ \text{BFD} \\ \text{CHAR} \end{cases}$$

Поясним операцию типа, для чего предположим, что в программе описано несколько переменных:

DCL (A,B,C) DEC FLOAT

С учетом операции типа построенная ОПЗ этого описания примет следующий вид:

A B C 3 DFT

Рассмотрим далее следующие операции описания процедур:

- *начало процедуры* - имеет два операнда: номер процедуры, уровень процедуры

<номер><уровень> НП

- *конец процедуры* - не имеет операндов

КП

- *конец описания* - имеет два операнда: номер процедуры, уровень процедуры

<номер><уровень>КО

Операторы начала и конца процедуры и описания данных в исходной программе являются неисполняемыми. Им можно присвоить самый низкий

приоритет в таблице приоритетов, после чего она примет следующий вид (табл. 2.7).

Таблица 2.7

Входной элемент	Приоритет
IF ([АЭМ Ф	0
, ;)] THEN ELSE	1
:= GOTO	2
V	3
&	4
¬	5
Отношения	6
+ -	7
* /	8
Возведение в степень	9
: PROC END DCL	10

Сам алгоритм Дейкстры с учетом обработки описательных операторов модифицируется следующим образом:

- символ PROC из входной строки заносится в стек, при этом в стек с ним также заносятся номер и уровень процедуры;
- символа DCL из входной строки заносится в стек с номером и уровнем процедуры и начальным значением счетчика операндов, равным 1;
- символ описания типа помещается в вершину стека;
- символы (и) при DCL в вершине стека игнорируются;

- символ ',' из входной строки, в зависимости от верхнего элемента стека, выполняет различные функции:
    - если в вершине стека находится оператор DCL, значение счетчика операндов наращивается на 1 (запятая между операндами);
    - если в вершине стека находится символ описания типа, он выталкивает его из стека, помещает в выходную строку операцию ТИП со значением счетчика из оператора DCL и счетчик операндов принимает начальное значение 1 (запятая между символами описания типа);
- символ END из входной строки всегда заносится в стек;
- символ ';' из входной строки, в зависимости от верхнего элемента стека, выполняет различные функции:
    - если в вершине стека находится оператор PROC, то рассматриваемый символ выталкивает его из стека, занося в выходную строку операцию начала процедуры с ее номером и уровнем;
    - если в вершине стека находится оператор DCL, точка с запятой выталкивает его из стека, записывая в выходную строку операцию конца описания с номером и уровнем процедуры;
    - если в вершине стека находится символ описания типа, то рассматриваемый символ обрабатывает его аналогично запятой, а затем применяется к следующему в стеке оператору DCL;
    - если в вершине стека находится оператор END, точка с запятой выталкивает его из стека, записывая в выходную строку операцию конца процедуры;
- символ '.' выталкивает из стека оператор END, записывая в выходную строку операцию конца процедуры.

Например, алгоритм перевода программы

```
PROC F1;
DCL (A,B) DEC FIXED,
          D DEC FLOAT;
     PROC F2;
          DCL (C,D) BIN FIXED;
     END;
END.
```

выглядит следующим образом:

Выходная строка		F1	1 1 НП			A		B	
С Т Е К	PROC 1,1			DCL 1,1,1	DCL 1,1,1		DCL 1,1,2		DCL 1,1,2
Входная строка	PROC	F1	;	DCL	(	A	,	B	)

Выходная строка		2 DFD	D		DFT	1 1 КО		F2	2 2 НП
С Т Е К	DFD	DCL 1,1,1		DFT	DCL 1,1,1		PROC 2,2		
	DCL 1,1,2			DCL 1,1,1					
Входная строка	DEC FIXED	,	D	DEC FLOAT	;		PROC	F2	;

Выходная строка			C		D			2 BFD	2 2 КО
С Т Е К	DCL 2,2,1	DCL 2,2,1		DCL 2,2,2		DCL 2,2,2	BFD	DCL 2,2,1	
							DCL 2,2,2		
Входная строка	DCL	(	C	,	D	)	BIN FIXED	;	

Выходная строка		КП		КП
СТЕК	END		END	
Входная строка	END	;	END	.

Таким образом, ОПЗ рассмотренной программы имеет следующий вид:

F1 1 1 НП A B 2 DFD D DFT 1 1 КО F2 2 2 НП C D 2 BFD 2 2 КО КП КП.

Таблица переходов МП-автомата

Состояние 0

Стек	N,C	I	(	)	]	,	;	IF	THEN	ELSE	PROC	DCL	END	*mun*	:=	Оп3	Оп4	Оп5	Оп6	Оп7	Оп8	Оп9	.	:	[
⊨	Вых(X)	Сост(1) Вых(X)	Вт	Err	Err	Err	N	Вт(IF)	Err	Err	Вт(I,j Proc)	Вт(I,j,1 D)	Вт	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
(	Вых(X)	Сост(1) Вых(X)	Вт	Выт	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
i A	Вых(X)	Сост(1) Вых(X)	Вт	Err	Вых Выт	Зам(i+1 A)	Err	Err	Err	Err	Err	Err	Err	Err	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
i Ф	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт	Err	Зам(i+1 Ф)	Err	Err	Err	Err	Err	Err	Err	Err	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
IF	Вых(X)	Сост(1) Вых(X)	Вт	Err	Err	Err	Err	Err	Вых(Mi УПЛ) Зам(Mi IF)	Err	Err	Err	Err	Err	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
Mi IF	Вых(X)	Сост(1) Вых(X)	Вт	Err	Err	Err	Вых(Mi:) Выт	Вт(IF)	Err	Вых(Mi+1 БП Mi:) Зам(Mi+1 IF)	Err	Err	Err	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
Mi+1 IF	Вых(X)	Сост(1) Вых(X)	Вт	Err	Err	Err	Вых(Mi+1:) Выт	Вт(IF)	Err	Вых(Mi+1:) Выт, Држ	Err	Err	Err	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
i,j Proc	Err	Сост(1) Вых(X)	Вт	Err	Err	Err	Вых Выт	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err
I,j,k D	Err	Сост(1) Вых(X)	N	N	Err	Зам(I,j,k+1 D)	Вых(КО) Выт	Err	Err	Err	Err	Err	Err	Вт	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err
mun	Err	Err	Err	Err	Err	Вых(k тип) Выт Зам(I,j,1 D)	Вых(k тип) Выт Држ	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err
END	Err	Err	Err	Err	Err	Err	Вых(КП) Выт	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Err	Вых (КП) Выт	Err	Err
:=	Вых(X)	Сост(1) Вых(X)	Вт	Err	Err	Err	Вых, Выт Држ	Err	Err	Вых, Выт Држ	Err	Err	Err	Err	Err	Вт	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
Оп3	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт Држ	Вых Выт Држ	Вых Выт Држ	Выт Држ	Err	Err	Err	Err	Err	Err	Err	Err	Вых Выт Држ	Вт	Вт	Вт	Вт	Вт	Вт	Err	Err	Err

Стек	N,C	I	(	)	]	,	;	IF	THEN	ELSE	PROC	DCL	END	*mun*	:=	Оп3	Оп4	Оп5	Оп6	Оп7	Оп8	Оп9	.	:	[
Оп4	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Выт Держ	Err	Err	Err	Err	Err	Err	Err	Err	Вых Выт Держ	Вых Выт Держ	Вт	Вт	Вт	Вт	Вт	Err	Err	Err
Оп5	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Err	Err	Err	Err	Err	Err	Err	Err	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вт	Вт	Вт	Вт	Err	Err	Err
Оп6	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Err	Err	Err	Err	Err	Err	Err	Err	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вт	Вт	Вт	Err	Err	Err
Оп7	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Err	Err	Err	Err	Err	Err	Err	Err	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вт	Вт	Err	Err	Err
Оп8	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Err	Err	Err	Err	Err	Err	Err	Err	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вт	Err	Err	Err
Оп9	Вых(X)	Сост(1) Вых(X)	Вт	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Err	Err	Err	Err	Err	Err	Err	Err	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Вых Выт Держ	Err	Err	Err

Состояние 1

Стек	)	(	N,C	I	[	:	Другие лексемы
1 Ф	Выт Вых	Сост(0) Держ	Вых(X) Зам(2Ф) Сост(0)	Вых(X) Зам(2Ф) Сост(0)	Err		Err
Дру- гие эле- менты	Сост(0) Держ	Вт(1Ф)	Err	Err	Вт(2А) Сост(0)	Вых(:) Сост(0)	Сост(0) Держ

В приведенных таблицах вся основная работа автомата происходит в состоянии 0. При обработке идентификатора автомат переходит в состояние 1, где происходит начальная обработка функций, массивов, а также обработка меток. После этого автомат возвращается в состояние 0.

Для сокращения записи в таблице используются следующие сокращения:

Команды:

Err – ошибка

Выт – вытолкнуть элемент из вершины стека

Вт – поместить элемент в вершину стека. По умолчанию помещается элемент из входной строки. Если вталкиваемый элемент отличается от элемента из входной строки, то он указывается в скобках.

Вых – поместить элемент в выходную строку. По умолчанию помещается элемент из вершины стека. Если выдаваемый элемент отличается от элемента в вершине стека, то он указывается в скобках. X означает элемент из входной строки.

Сост(i) – перевести автомат в состояние i.

Зам(Y) – заменить элемент в вершине стека на элемент Y.

Држ – не переходить к следующему элементу входной строки (по умолчанию, если команда отсутствует, переход выполняется всегда).

N – отсутствие действий и переход к следующему символу входной строки.

Операции:

A – АЭМ.

D – DCL.

Лексемы:

Оп*i* – операции, имеющие приоритет *i.*

2.9. Комплексный пример перевода исходной программы в ОПЗ

Рассмотрим комплексный пример перевода исходной программы на языке, описанном в лабораторной работе № 1, в обратную польскую запись.

```
PROC MAIN;
     DCL (A1, A2) DEC FIXED;
     A1=378;

     A2=.73;

     PROC CALC;
          DCL (SUM, MULT) DEC FIXED;
          IF A1+A2<>3.2 THEN GOTO P;
          SUM=(A1+A2)*A2;
     P:   MULT=A1*A2;
     END;
END.
```

Прежде всего, составим таблицу приоритетов, включив в нее все операции и операторы, имеющиеся во входном языке (табл. 2.8). Для этого воспользуемся обозначениями операторов и операций, которые были введены в лабораторной работе № 1.

Таблица 2.8

Приоритет	Входной элемент	Обозначение
0	IF	W6
	(	R4
1	THEN	W8
	.	R6
	,	R2
	;	R3
	)	R5
2	GOTO	W5
	=	O5
3	<	O3
	>	O4
4	+	O1
5	*	O2
6	PROC	W7
	END	W3
	DCL	W2
	:	O6

Привычное представление процесса перевода исходной программы в ОПЗ на основе алгоритма Дейкстры имеет следующий вид:

Выходная строка		MAIN	1 1 НП			A1		A2	
С Т Е К	PROC 1,1			DCL 1,1,1	DCL 1,1,1		DCL 1,1,2		DCL 1,1,2
Входная строка	PROC	MAIN	;	DCL	(	A1	,	A2	)

Выходная строка		2 DFD	1 1 КО	A1		378	:=	A2		.73	:=		CALC
С Т Е К	DFD	DCL 1,1,1			:=				:=			PROC 2,2	
	DCL 1,1,2												
Входная строка	DEC FIXED	;		A1	:=	378	;	A2	:=	.73	;	PROC	CALC

Выходная строка	2 2 НП			SUM		MULT			2 BFD	2 2 КО
С Т Е К		DCL 2,2,1	DCL 2,2,1		DCL 2,2,2		DCL 2,2,2	BFD	DCL 2,2,1	
								DCL 2,2,2		
Входная строка	;	DCL	(	SUM	,	MULT	)	BIN FIXED	;	

Выходная строка		A1		A2	+	3.2	>	M1 УПЛ		P	БП	M1:	SUM		A2		A1	+	:=
С Т Е К	IF		+		>		IF	IF M1	GOTO		IF M1			:=		+		:=	
			IF		IF				IF M1							:=			
Входная строка	IF	A1	+	A2	>	3.2	THEN		GOTO	P	;		SUM	:=	A2	+	A1	;	

Выходная строка	P	:	MULT		A1			A2		SUM	+	*	:=		КП		КП
С Т Е К				:=		*	(		+		*	:=		END		END	
						:=	*		(		:=						
							:=		*								
									:=								
Входная строка	P	:	MULT	=	A1	*	(	A2	+	SUM	)	;		END	;	END	.

Полученная в результате перевода ОПЗ программы имеет вид:

MAIN 1 1 НП A1 A2 2 DFD 1 1 КО A1 378 := A2 .73 := CALC2 2 НП
SUM MULT 2 BFD 2 2 КО A1 A2 + 3.2 > M1 УПЛ P БП M1: SUM A2 A1 + :=
P : MULT A1 A2 SUM + * := КП КП .

Для наглядности в рассмотренном примере текст программы представлен на исходном языке, но в реальном трансляторе входной текст является результатом работы лексического анализатора. В нем уже выделены и классифицированы лексемы, но в остальном работа протекает в соответствии с описанным выше примером.

2.10. Задание к лабораторной работе

Написать на языке высокого уровня программу преобразования входной программы, представленной в виде последовательности токенов, во внутреннее представление – обратную польскую запись.

На вход программы подается файл (файлы), являющийся результатом работы лексического анализатора. Результатом работы программы должен быть файл, содержащий обратную польскую запись входной программы.

Отчет по работе должен содержать:

1. Таблицу приоритетов операция для алгоритма Дейкстры;
2. Словесное описание правил работы со стеком;
3. Таблицу(ы) переходов МП-автомата, выполняющего преобразование;
4. Описание программной реализации (структуры данных, алгоритмы);

5. Листинг программы;
6. Пример входного файла (в исходном виде и в виде токенов) и выходного файла (внутреннего представления).