Семашко Г.Л. и Благонравова О.В.

Б3-11-83-917.

ОБЪЕДИНЕННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИССЛЕДОВАНИЙ

Б3-11-83-917

ДЕПОНИРОВАННАЯ ПУБЛИКАЦИЯ

Дубна 1983

ОБ'ЕДИНЕННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИССЛЕДОВАНИЙ
ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И АВТОМАТИЗАЦИИ

Б3-11-83-917

Г.Л.СЕМАШКО, О.В.БЛАГОНРАВОВА

ЯЗЫК ПАСКАЛЬ ДЛЯ ЭВМ БЭСМ-6, CDC-6500 И ЕС ЭВМ.

ДУБНА, 1983 Г.

О Г Л А В Л Е Н И Е

		СТР.
	ВВЕДЕНИЕ	1
1.	ЗАДАНИЯ ДЛЯ ЭВМ С ПРОГРАММОЙ НА ПАСКАЛЕ	2
2.	АЛФАВИТ ЯЗЫКА ПАСКАЛЬ	4
3.	КОНСТАНТЫ	5
4.	ИДЕНТИФИКАТОРЫ	7
5.	О ТИПАХ ПЕРЕМЕННЫХ	7
6.	СКАЛЯРНЫЕ ТИПЫ	9
7.	СТРУКТУРА ПРОГРАММ	13
8.	ОПЕРАТОРЫ	16
9.	ПРОЦЕДУРА ВВОДА	22
10.	ПРОЦЕДУРА ВЫВОДА	23
11.	СЛОЖНЫЕ ТИПЫ	25
12.	ТИП POINTER	39
13.	ПРОЦЕДУРЫ	40
14.	ФУНКЦИИ	51
15.	РАБОТА С ВНЕШНИМИ МОДУЛЯМИ	52
16.	РЕЖИМЫ ТРАНСЛЯЦИИ (ПСЕВДОКОММЕНТАРИИ)	53
17.	СТАНДАРТНЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ	55
18.	КОДИРОВКА НЕКОТОРЫХ СИМВОЛОВ	58

ВВЕДЕНИЕ

ЯЗЫК ПАСКАЛЬ БЫЛ СОЗДАН Н.ВИРТСМ КАК СПЕЦИАЛЬНЫЙ ЯЗЫК ДЛЯ ОБУЧЕНИЯ СТУДЕНТОВ, НО ВСКОРЕ ПОЛУЧИЛ РАСПРОСТРАНЕНИЕ СРЕДИ ПРОГРАММИСТОВ. СЕЙЧАС ПАСКАЛЬ ШИРОКО ИСПОЛЬЗУЕТСЯ И ДЛЯ НАПИСАНИЯ ПРИКЛАДНЫХ ПРОГРАММ, И КАК ЯЗЫК СИСТЕМНОГО ПРОГРАММИРОВАНИЯ.

В ЧАСТНОСТИ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МНОГИХ МИНИ-МАШИН НАПИСАНО НА ЯЗЫКЕ ПАСКАЛЬ. ТРАНСЛЯТОРЫ С ЭТОГО ЯЗЫКА ИМЕЮТСЯ НА ВСЕХ НАИБОЛЕЕ РАСПРОСТРАНЕННЫХ ЭВМ МИРА.

ОСНОВНЫМИ ПРИЧИНАМИ ПОПУЛЯРНОСТИ ЯЗЫКА ПАСКАЛЬ ЯВЛЯЮТСЯ СЛЕДУЮЩИЕ:

1. ПРОСТОТА СТРУКТУРЫ ЯЗЫКА ПОЗВОЛЯЕТ БЫСТРО ЕГО ОСВОИТЬ И СОЗДАВАТЬ АЛГОРИТМИЧЕСКИ СЛОЖНЫЕ ПРОГРАММЫ.

2. РАЗВИТЫЕ СРЕДСТВА ПРЕДСТАВЛЕНИЯ СТРУКТУР ДАННЫХ ОБЕСПЕЧИВАЮТ УДОБСТВО РАБОТЫ КАК С ЧИСЛОВОЙ, ТАК С СИМВОЛЬНОЙ И БИТОВОЙ ИНФОРМАЦИЕЙ.

3. НАЛИЧИЕ СПЕЦИАЛЬНЫХ МЕТОДИК СОЗДАНИЯ ТРАНСЛЯТОРОВ С ПАСКАЛЯ УПРОСТИЛО ИХ НАПИСАНИЕ И СПОСОБСТВОВАЛО ШИРОКОМУ РАСПРОСТРАНЕНИЮ ЯЗЫКА.

4. ВЫСОКАЯ ЭФФЕКТИВНОСТЬ РАБОЧИХ ПРОГРАММ ОТЛИЧАЕТ ПАСКАЛЬ ОТ АЛГОЛА, ФОРТРАНА, КОБОЛА. ЭТО ЯВИЛОСЬ, В ЧАСТНОСТИ, ПРИЧИНОЙ ИСПОЛЬЗОВАНИЯ ПАСКАЛЯ В КАЧЕСТВЕ ЯЗЫКА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ.

5. В ЯЗЫКЕ ПАСКАЛЬ РЕАЛИЗОВАНЫ ИДЕИ СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ, ЧТО ДЕЛАЕТ ПРОГРАММУ НАГЛЯДНОЙ И ДАЕТ ХОРОШИЕ ВОЗМОЖНОСТИ ОТЛАДКИ.

ТРАНСЛЯТОРЫ С ЯЗЫКА ПАСКАЛЬ ИМЕЮТСЯ НА ЭВМ БЭСМ-6 [3],[4], CDC-6500 [1],[2] И ЕС ЭВМ[5].

В ДАННОЙ РАБОТЕ СОДЕРЖИТСЯ ИНФОРМАЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЕЙ, НАЧИНАЮЩИХ РАБОТАТЬ НА ЯЗЫКЕ ПАСКАЛЬ.
ОПИСЫВАЕМЫЕ ВОЗМОЖНОСТИ СТАНДАРТНОГО ПАСКАЛЯ РЕАЛИЗОВАНЫ НА ВСЕХ ТРЕХ ЭВМ, ЗА ИСКЛЮЧЕНИЕМ СЛУЧАЕВ, ОГОВОРЕННЫХ ОСОБО.

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ, ПРЕДССТАВЛЯЕМЫЕ ТРАНСЛЯТОРАМИ КОНКРЕТНЫХ ЭВМ, В РАБОТЕ НЕ ОПИСАНЫ. С НИМИ МОЖНО ОЗНАКОМИТЬСЯ В СООТВЕТСТВУЮЩЕЙ ЛИТЕРАТУРЕ: [1], [2], [3], [4], [5].

РЕЖИМЫ ТРАНСЛЯЦИИ ПРИВЕДЕНЫ ТОЛЬКО ТЕ, ЧТО ИМЕЮТСЯ НА ВСЕХ 3-Х ЭВМ.

1. ЗАДАНИЯ ДЛЯ ЭВМ С ПРОГРАММОЙ НА ПАСКАЛЕ

1.1. ЗАДАНИЕ ДЛЯ ЭВМ CDC-6500

```
XXXX,TXXXX. NAME
ACCOUNT(XXXX)
REDUCE.
PASCAL.
LGO.
7/8/9
```

```
--------------
I  ПРОГРАММА  I
I     НА      I
I   ПАСКАЛЕ   I
--------------
```

```
7/8/9
```

```
----------------------
I  ДАННЫЕ, ВВОДИМЫЕ  I
I  ОПЕРАТОРОМ  READ  I
----------------------
```

```
6/7/8/9
```

ЧЕРЕЗ XXXX ОБОЗНАЧЕНЫ СООТВЕТСТВЕННО ШИФР ЗАДАНИЯ, ЗАКАЗАННОЕ ВРЕМЯ И ШИФР ПОДРАЗДЕЛЕНИЯ.

1.2 ЗАДАНИЕ ДЛЯ ЕС-ЭВМ(1040, 1060)

```
//XXXXJOBXXXX,ФАМИЛИЯ,MSGLEVEL=(1,0)
// EXEC PASCLG, PARM, LKED='NOMAP, NOXREF,NOLIST'
// SYSIN DD *

     ------------------
     I  ПРОГРАММА НА  I
     I  ЯЗЫКЕ ПАСКАЛЬ I
     ------------------

//GO.SYSIN D *

     --------------------
     I  ДАННЫЕ ДЛЯ       I
     I  ОПЕРАТОРА READ   I
     --------------------

//
```

ЗДЕСЬ ЧЕРЕЗ XXXX ОБОЗНАЧЕНО СООТВЕТСТВЕННО ИМЯ ЗАДАНИЯ И ШИФР

1.3. ЗАДАНИЕ ДЛЯ ЭВМ БЭСМ-6 (ПАКЕТ ЗАДАЧИ).

```
   *NAME XXXX
   *PASS:XXXX
   *TIME:00.05
   *LIBRARY:11
   *CALL ALLMEMORY
   *PASCAL
------------------
I  ПРОГРАММА НА   I
I  ЯЗЫКЕ ПАСКАЛЬ  I
------------------

   *EXECUTE

--------------------
I  ДАННЫЕ   ДЛЯ    I
I  ОПЕРАТОРА READ  I
--------------------

   *END FILE

   ДИСПЕТЧЕРСКИЙ КОНЕЦ
```

В ЭТОМ ПРИМЕРЕ ЧЕРЕЗ XXXX ОБОЗНАЧЕНЫ СООТВЕТСТВЕННО ФАМИЛИЯ ПОЛЬЗОВАТЕЛЯ И ЕГО ШИФР.

2. АЛФАВИТ ЯЗЫКА ПАСКАЛЬ

ЯЗЫК ОПЕРИРУЕТ СО СЛЕДУЮЩИМ НАБОРОМ СИМВОЛОВ:

А) ВСЕХ ЛАТИНСКИХ БУКВ; НА БЭСМ-6 - И РУССКИХ БУКВ;
Б) ВСЕХ АРАБСКИХ ЦИФР;
В) ОГРАНИЧИТЕЛЕЙ И СПЕЦ. СИМВОЛОВ

+	:=	:	<	(	{
-	.	'	<=	)	}
*	,	=	>	[	↑
/	;	<>	>=	]	..

Г) КЛЮЧЕВЫХ СЛОВ

AND	END	NIL	SET
ARRAY	FILE	NOT	THEN
BEGIN	FOR	OF	TO
CASE	FUNCTION	OR	TYPE
CONST	GOTO	PACKED	UNTIL
DIV	IF	PROCEDURE	VAR
DO	IN	PROGRAM	WHILE
DOWNTO	LABEL	RECORD	WITH
ELSE	MOD	REPEAT	

В ПРОГРАММЕ НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ ИДЕНТИФИКАТОРЫ, СОВПАДАЮЩИЕ ПО НАПИСАНИЮ С ПРИВЕДЕННЫМИ ВЫШЕ КЛЮЧЕВЫМИ СЛОВАМИ.

ВМЕСТО СИМВОЛОВ {и} , ОТСУТСТВУЮЩИХ НА КЛАВИАТУРЕ, НАБИРАЮТСЯ СООТВЕТСТВЕННО ПАРЫ СИМВОЛОВ (* И *).

КОНСТРУКЦИЯ (*ТЕКСТ*) ВОСПРИНИМАЕТСЯ КАК КОММЕНТАРИЙ И МОЖЕТ БЫТЬ ПОМЕЩЕНА В ЛЮБОМ МЕСТЕ ПРОГРАММЫ.

НА ЕС ЭВМ (ПАСКАЛЬ 8000) ИСПОЛЬЗУЕТСЯ СЛЕДУЮЩЕЕ ПРЕДСТАВЛЕНИЕ СИМВОЛОВ:

СТАНДАРТНОЕ	[	]	{	}	AND	OR	NOT	< >	↑
ПАСКАЛЬ 8000	(.	.)	(*	*)	&	|	¬	¬=	@

3. КОНСТАНТЫ

ЧИСЛОВЫЕ КОНСТАНТЫ В ПАСКАЛЕ МОГУТ БЫТЬ ЦЕЛЫМИ И ВЕЩЕСТВЕННЫМИ.

3.1 ЦЕЛЫЕ КОНСТАНТЫ (INTEGER)

ЦЕЛЫЕ КОНСТАНТЫ В ПАСКАЛЕ НЕ ДОЛЖНЫ ПРЕВОСХОДИТЬ ПО АБСОЛЮТНОЙ ВЕЛИЧИНЕ

НА ЭВМ БЭСМ-6 $2^{40}-1$

CDC-6500 $2^{48}-1$

ЕС ЭВМ $2^{31}-1$

ЦЕЛЫЕ ВОСЬМЕРИЧНЫЕ КОНСТАНТЫ ПРЕДСТАВЛЯЮТ СОБОЙ ПОСЛЕДОВАТЕЛЬНОСТЬ ВОСЬМЕРИЧНЫХ ЦИФР, ЗАКАНЧИВАЮЩУЮСЯ СПРАВА БУКВОЙ B.

ПРИМЕР.

```
135B; 2B;
```

НА БЭСМ-6 ДОПУСТИМЫ И ВОСЬМЕРИЧНЫЕ КОНСТАНТЫ, ЗАКАНЧИВАЮЩИЕСЯ, ВМЕСТО B, БУКВАМИ C ЛИБО T.

БУКВА C ОЗНАЧАЕТ, ЧТО ПОСЛЕДОВАТЕЛЬНОСТЬ ВОСЬМЕРИЧНЫХ ЦИФР РАСПОЛОЖЕНА В МАШИННОМ СЛОВЕ СПРАВА, А СЛЕВА ДОПОЛНЯЕТСЯ НУЛЯМИ;

БУКВА T - ПОСЛЕДОВАТЕЛЬНОСТЬ ВОСЬМЕРИЧНЫХ ЦИФР РАСПОЛОЖЕНА В СЛОВЕ СЛЕВА, А СПРАВА ДОПОЛНЯЕТСЯ НУЛЯМИ.

ПРИМЕР.

```
0025T  ЕСТЬ КОНСТАНТА 0025 0000 0000 0000
146C        -"-       0000 0000 0000 0146
```

3.2. ВЕЩЕСТВЕННЫЕ КОНСТАНТЫ (REAL)

ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ КОНСТАНТ,КАК И НА ФОРТРАНЕ, ИМЕЕТ ДВЕ ФОРМЫ: В ВИДЕ ДЕСЯТИЧНОЙ ДРОБИ, ГДЕ ВМЕСТО ЗАПЯТОЙ ИСПОЛЬЗУЕТСЯ ТОЧКА (НАПРИМЕР, 3.2), И В ВИДЕ ЧИСЛА, СОДЕРЖАЩЕГО УКАЗАНИЕ НА СТЕПЕНЬ ДЕСЯТИ
(НАПРИМЕР, 2.E-5).

КОНСТАНТА, ЗАПИСАННАЯ В ВИДЕ ДЕСЯТИЧНОЙ ДРОБИ, В ОТЛИЧИЕ ОТ КОНСТАНТ ФОРТРАНА, ОБЯЗАТЕЛЬНО ДОЛЖНА СОДЕРЖАТЬ КАК ЦЕЛУЮ ЧАСТЬ, ТАК И ДРОБНУЮ, Т.Е. НЕЛЬЗЯ ЗАПИСАТЬ **2.** И **.5** А СЛЕДУЕТ ИСПОЛЬЗОВАТЬ **2.0** И **0.5**

ВЕЩЕСТВЕННЫЕ КОНСТАНТЫ НЕ ДОЛЖНЫ ПО АБСОЛЮТНОЙ ВЕЛИЧИНЕ ПРЕВОСХОДИТЬ

НА БЭСМ-6 10^{19}

НА CDC-6500 10^{322}

НА ЕС ЭВМ 10^{76}

3.3. СИМВОЛЬНЫЕ КОНСТАНТЫ (CHAR)

СИМВОЛ, ЗАКЛЮЧЕННЫЙ В АПОСТРОФЫ, ЕСТЬ СИМВОЛЬНАЯ КОНСТАНТА.

ПРИМЕР. 'A'; '1'; '='.

3.4. КОНСТАНТЫ-СТРОКИ (ALFA)

ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ, ЗАКЛЮЧЕННАЯ В АПОСТРОФЫ, ЕСТЬ СТРОКА.

ПРИМЕР. 'DUBNA'; '12345'; 'A*B'.

ДЛИНОЙ СТРОКИ К НАЗЫВАЕТСЯ ЧИСЛО СИМВОЛОВ В НЕЙ.
ВСЕГДА **1<K<=N**
ЗДЕСЬ N -НЕКОТОРАЯ КОНСТАНТА (<= ОЗНАЧАЕТ $\leq$).
ДЛЯ БЭСМ-6 N=6; ДЛЯ CDC-6500 N=10, ДЛЯ ЕС-ЭВМ N=8;
(N-ЧИСЛО СИМВОЛОВ В СЛОВЕ ЭВМ).

ЕСЛИ СРЕДИ СИМВОЛОВ КОНСТАНТЫ-СТРОКИ ДОЛЖЕН БЫТЬ АПОСТРОФ, ТО ОН ИЗОБРАЖАЕТСЯ ДВУМЯ АПОСТРОФАМИ.

ПРИМЕР.

A'B'C'D СЛЕДУЕТ ПРОБИТЬ КАК 'A''B''C''D'.

3.5. БУЛЕВСКИЕ КОНСТАНТЫ (BOOLEAN)

ИМЕЮТСЯ ДВЕ БУЛЕВСКИЕ КОНСТАНТЫ: TRUE И FALSE.

4. ИДЕНТИФИКАТОРЫ

ИДЕНТИФИКАТОРЫ ПАСКАЛЯ - ПОСЛЕДОВАТЕЛЬНОСТЬ БУКВ ИЛИ ЦИФР, НАЧИНАЮЩАЯСЯ С БУКВЫ.
ЗНАЧАЩИМИ ЯВЛЯЮТСЯ ПЕРВЫЕ 8 СИМВОЛОВ.

5. О ТИПАХ ПЕРЕМЕННЫХ

КАЖДАЯ ПЕРЕМЕННАЯ (НАПРИМЕР A,B,C) ИСПОЛЬЗУЕМАЯ В ПРОГРАММЕ, ДОЛЖНА БЫТЬ ОПИСАНА СЛЕДУЮЩИМ ОБРАЗОМ:

```
VAR A:TYPE1; C,B:TYPE2;...
```

ЗДЕСЬ VAR - КЛЮЧЕВОЕ СЛОВО;
A,B,C - ИДЕНТИФИКАТОРЫ ПЕРЕМЕННЫХ;

TYPE1,TYPE2 - ТИПЫ ПЕРЕМЕННЫХ.

ПРИМЕР:
```
VAR J,K:INTEGER; A,B,C:REAL; L:BOOLEAN;
```

В ПАСКАЛЕ ТИПЫ ПЕРЕМЕННЫХ ПОДРАЗДЕЛЯЮТСЯ НА ПРОСТЫЕ (СКАЛЯРНЫЕ), СЛОЖНЫЕ И ССЫЛКИ (POINTER).

СКАЛЯРНЫЕ - ЭТО ЛИБО НОВЫЕ ТИПЫ, ДЕКЛАРИРУЕМЫЕ В ДАННОЙ ПРОГРАММЕ, ЛИБО - ОДИН ИЗ СТАНДАРТНЫХ : ЦЕЛЫЙ (INTEGER), ВЕЩЕСТВЕННЫЙ (REAL), БУЛЕВСКИЙ (BOOLEAN), СИМВОЛЬНЫЙ (CHAR), СТРОКОВЫЙ (ALFA).

ЕСЛИ ПЕРЕМЕННАЯ МОЖЕТ ПРИНИМАТЬ ТОЛЬКО ЗНАЧЕНИЯ ИЗ НЕКОТОРОГО ПОДМНОЖЕСТВА, ОПРЕДЕЛЯЕМОГО КАКИМ-ЛИБО СКАЛЯРНЫМ ТИПОМ (КРОМЕ REAL) , ТО ЧАСТО ТИП ЭТОЙ ПЕРЕМЕННОЙ ЗАДАЮТ КАК ПОДМНОЖЕСТВО (ДИАПАЗОН, SUBRANGE).

К СЛОЖНЫМ ТИПАМ ОТНОСЯТСЯ СЛЕДУЮЩИЕ:

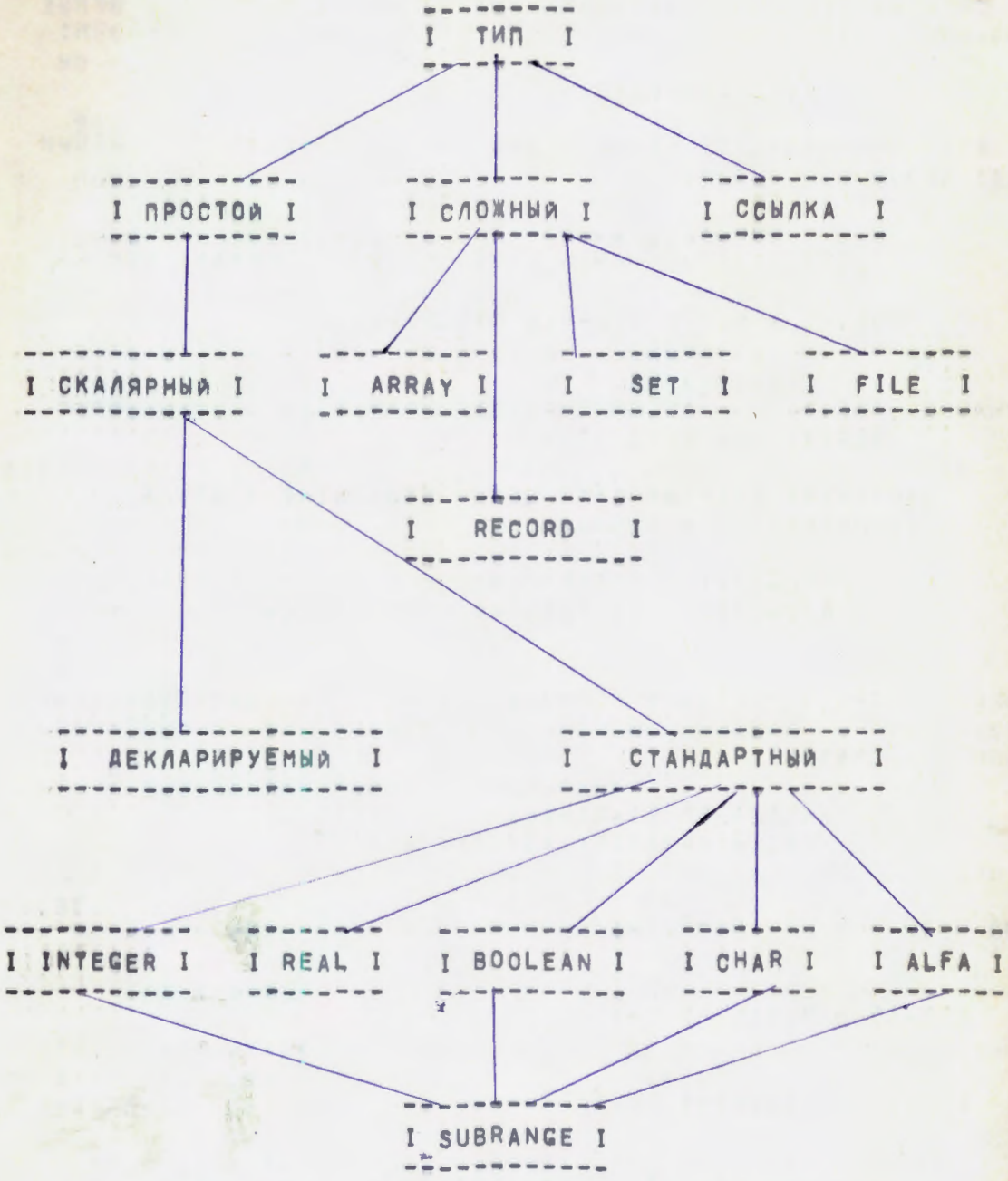
МАССИВ (ARRAY), ФАЙЛ (FILE), МНОЖЕСТВО (SET),ФАЙЛ (FILE), ЗАПИСЬ (RECORD)

ТИП POINTER ПОДРОБНО ОПИСАН В П.12.

В ЯЗЫКЕ ПАСКАЛЬ ИМЕЕТСЯ СЛЕДУЮЩИЙ НАБОР ТИПОВ ПЕРЕМЕННЫХ:

- ПРОСТЫЕ ТИПЫ
  - СКАЛЯРНЫЕ
    - ДЕКЛАРИРУЕМЫЕ
    - СТАНДАРТНЫЕ
      - ЦЕЛЫЙ (INTEGER)
      - ВЕЩЕСТВЕННЫЙ (REAL)
      - БУЛЕВСКИЙ (BOOLEAN)
      - СИМВОЛЬНЫЙ (CHAR)
      - СТРОКОВЫЙ (ALFA)
- СЛОЖНЫЕ ТИПЫ
  - МАССИВ (ARRAY)
  - МНОЖЕСТВО (SET)
  - ФАЙЛ (FILE)
  - ЗАПИСЬ (RECORD)
- ССЫЛКА (УКАЗАТЕЛЬ, POINTER)

НАБОР ТИПОВ ЯЗЫКА ПАСКАЛЬ ПРЕДСТАВЛЕН НА СЛЕДУЮЩЕЙ СХЕМЕ.



ТИП ПЕРЕМЕННОЙ ОПРЕДЕЛЯЕТ МНОЖЕСТВО ЗНАЧЕНИЙ ЭТОЙ ПЕРЕМЕННОЙ, НАБОР ОПЕРАЦИЙ, КОТОРЫЕ К НЕЙ МОГУТ БЫТЬ ПРИМЕНЕНЫ, И РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЭТИХ ОПЕРАЦИЙ.

6. СКАЛЯРНЫЕ ТИПЫ

КАЖДЫЙ СКАЛЯРНЫЙ ТИП ОПРЕДЕЛЯЕТ СООТВЕТСТВУЮЩЕЕ ЕМУ УПОРЯДОЧЕННОЕ МНОЖЕСТВО ЗНАЧЕНИЙ

6.1. ТИП ЦЕЛЫЙ (INTEGER)

ПЕРЕМЕННЫЕ ТИПА **INTEGER** МОГУТ ПРИНИМАТЬ ТОЛЬКО ЦЕЛЫЕ ЗНАЧЕНИЯ. ТАКИЕ ПЕРЕМЕННЫЕ ОПИСЫВАЮТСЯ СЛЕДУЮЩИМ ОБРАЗОМ:

```
A,B,C:INTEGER;
```

ЕСЛИ ИСПОЛЬЗУЮТСЯ ЦЕЛЫЕ ОПЕРАНДЫ, ТО СЛЕДУЮЩИЕ ОПЕРАЦИИ ДАЮТ ЦЕЛЫЙ РЕЗУЛЬТАТ:

```
        *      УМНОЖЕНИЕ.
        DIV    ДЕЛЕНИЕ БЕЗ ОКРУГЛЕНИЯ - ЦЕЛАЯ ЧАСТЬ
               ЧАСТНОГО
       MOD,    A MOD B = A-((A DIV B)*B)
        +      СЛОЖЕНИЕ.
        -      ВЫЧИТАНИЕ.
ФУНКЦИИ ABS(X)   - АБСОЛЮТНАЯ ВЕЛИЧИНА X
    И   SQR(X)   - КВАДРАТ X
```

СЛЕДУЮЩИЕ ФУНКЦИИ ДАЮТ ЦЕЛЫЙ РЕЗУЛЬТАТ И ДЛЯ X ВЕЩЕСТВЕННОГО ФУНКЦИИ:

```
        TRUNC(X) - (ОТБРАСЫВАНИЕ ДЕСЯТИЧНЫХ ЗНАКОВ)
    И   ROUND(X) - (ОКРУГЛЕНИЕ ДО ЦЕЛОГО)
```

ОПЕРАЦИЯ **OP** ВЫПОЛНЯЕТСЯ ПРАВИЛЬНО ТОЛЬКО ПРИ УСЛОВИИ, ЕСЛИ РЕЗУЛЬТАТ И КАЖДЫЙ ОПЕРАНД ПО МОДУЛЮ НЕ ПРЕВОСХОДЯТ КОНСТАНТЫ **MAXINT**.

```
ABS(A OP B)<MAXINT,
ABS(A)<=MAXINT,ABC(B)<=MAXINT.
```

ДЛЯ БЭСМ-6 $MAXINT=2^{40}-1$;

CDC-6500 $MAXINT=2^{48}-1$;

ЕС ЭВМ $MAXINT=2^{31}-1$;

6.2. ТИП ВЕЩЕСТВЕННЫЙ (REAL)

ВЕЛИЧИНЫ X ЭТОГО ТИПА МОГУТ ПРИНИМАТЬ ТОЛЬКО ВЕЩЕСТВЕННЫЕ ЗНАЧЕНИЯ.

ЕСЛИ ХОТЯ БЫ ОДИН ИЗ ОПЕРАНДОВ ВЕЩЕСТВЕННЫЙ, СЛЕДУЮЩИЕ ОПЕРАЦИИ ДАЮТ ВЕЩЕСТВЕННЫЙ РЕЗУЛЬТАТ: +,-,*.

ОПЕРАЦИЯ ДЕЛЕНИЯ / ДАЕТ ВЕЩЕСТВЕННЫЙ РЕЗУЛЬТАТ И В СЛУЧАЕ ДВУХ ЦЕЛЫХ ОПЕРАНДОВ!

ПРИМЕР. 6/2 ДАЕТ 3.0

СЛЕДУЕТ ОБРАТИТЬ ВНИМАНИЕ, ЧТО СТАНДАРТНАЯ ФУНКЦИЯ

ABS(X) - (МОДУЛЬ X) ОТ ЦЕЛОГО АРГУМЕНТА ДАЕТ ЦЕЛЫЙ РЕЗУЛЬТАТ, КАК И **SQR(X)** - КВАДРАТ X.

НО ФУНКЦИИ:

SIN(X) - СИНУС X,

COS(X) - КОСИНУС X,

LN(X) - НАТУРАЛЬНЫЙ ЛОГАРИФМ X,

EXP(X) - ЭКСПОНЕНТА X,

SQRT(X) - КОРЕНЬ КВАДРАТНЫЙ ИЗ X,

ARCTAN(X) - АРКТАНГЕНС X

ДАЮТ ВЕЩЕСТВЕННЫЙ РЕЗУЛЬТАТ КАК ДЛЯ ВЕЩЕСТВЕННОГО, ТАК И ДЛЯ ЦЕЛОГО АРГУМЕНТА

НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ ПЕРЕМЕННЫЕ ТИПА **REAL**

А) В ФУНКЦИЯХ **PRED(X),SUCC(X),ORD(X)**;
Б) В КАЧЕСТВЕ ИНДЕКСОВ МАССИВОВ;
В) В ОПЕРАТОРАХ ПЕРЕДАЧИ УПРАВЛЕНИЯ В КАЧЕСТВЕ МЕТОК;
Г) В ОПРЕДЕЛЕНИИ БАЗЫ ТИПА **SET** (СМ.НИЖЕ).

6.3. БУЛЕВСКИЙ ТИП (BOOLEAN)

ПЕРЕМЕННАЯ БУЛЕВСКОГО ТИПА ПРИНИМАЕТ ЗНАЧЕНИЯ **TRUE** ИЛИ **FALSE**.
ЭТИ ВЕЛИЧИНЫ УПОРЯДОЧЕНЫ СЛЕДУЮЩИМ ОБРАЗОМ:

FALSE<TRUE

ОПЕРАЦИИ **AND,OR,NOT** (ПРИМЕНЯЕМЫЕ К БУЛЕВСКИМ ОПЕРАНДАМ)ДАЮТ БУЛЕВСКИЕ ЗНАЧЕНИЯ.
СТАНДАРТНЫМИ БУЛЕВСКИМИ ФУНКЦИЯМИ ЯВЛЯЮТСЯ:

ODD (X) - TRUE, ЕСЛИ X НЕЧЕТНЫЙ (X ЦЕЛЫЙ);
EOLN(X) - TRUE, ЕСЛИ ВСТРЕТИЛСЯ КОНЕЦ СТРОКИ ФАЙЛА X;
EOF (X) - TRUE, ЕСЛИ ВСТРЕТИЛСЯ КОНЕЦ ФАЙЛА X.

В ОСТАЛЬНЫХ СЛУЧАЯХ ФУНКЦИИ ПРИНИМАЮТ ЗНАЧЕНИЕ **FALSE**.

6.4. ТИП СИМВОЛЬНЫЙ (CHAR)

ПЕРЕМЕННАЯ ТИПА **CHAR** МОЖЕТ ПРИНИМАТЬ ЗНАЧЕНИЯ ИЗ ОПРЕДЕЛЕННОЙ УПОРЯДОЧЕННОЙ ПОСЛЕДОВАТЕЛЬНОСТИ СИМВОЛОВ, РАЗРЕШЕННОЙ ТРАНСЛЯТОРОМ С ПАСКАЛЯ ДАННОЙ ЭВМ.

ДВЕ СТАНДАРТНЫЕ ФУНКЦИИ ПОЗВОЛЯЮТ СОПОСТАВИТЬ ДАННУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ С МНОЖЕСТВОМ ЦЕЛЫХ ПОЛОЖИТЕЛЬНЫХ ЧИСЕЛ - ПОРЯДКОВЫЕ НОМЕРА СИМВОЛОВ ПОСЛЕДОВАТЕЛЬНОСТИ.

ЭТИ ФУНКЦИИ НАЗЫВАЮТСЯ ФУНКЦИЯМИ ПРЕОБРАЗОВАНИЯ:

ORD(C) - ВЫДАЕТ НОМЕР СИМВОЛА **C**,
CHR(I) - ВЫДАЕТ **I** -Й СИМВОЛ ПОСЛЕДОВАТЕЛЬНОСТИ.

6.4. ТИП СТРОКОВЫЙ (ALFA)

ПЕРЕМЕННАЯ ЭТОГО ТИПА ПРЕДСТАВЛЯЕТ СОБОЙ МАШИННОЕ СЛОВО, СОДЕРЖАЩЕЕ СИМВОЛЬНУЮ ИНФОРМАЦИЮ.

СЛОВО БЭСМ-6 СОДЕРЖИТ 6 СИМВОЛОВ,
СЛОВО CDC-6500 СОДЕРЖИТ 10 СИМВОЛОВ,
СЛОВО ЕС-1060 СОДЕРЖИТ 8 СИМВОЛОВ.

ПЕРЕМЕННАЯ ТИПА **ALFA** ДОЛЖНА ЗАНИМАТЬ ОДНО ПОЛНОЕ МАШИННОЕ СЛОВО.

К ПЕРЕМЕННЫМ ЭТОГО ТИПА МОЖНО ПРИМЕНЯТЬ ОПЕРАЦИИ ОТНОШЕНИЯ (СМ.6.4)

6.4. ДЕКЛАРИРУЕМЫЙ ТИП

ТИП ТАКИХ ПЕРЕМЕННЫХ ДЕКЛАРИРУЕТСЯ ПРОГРАММИСТОМ СЛЕДУЮЩИМ ОБРАЗОМ:

```
TYPE NM=(WORD1,WORD2,...WORDN);
```

ЗДЕСЬ **NM** - ИДЕНТИФИКАТОР ТИПА (ПРОИЗВОЛЬНЫЙ ИДЕНТИФИКАТОР)
WORD1,WORD2... - КОНКРЕТНЫЕ ЗНАЧЕНИЯ, КОТОРЫЕ МОЖЕТ ПРИНИМАТЬ ПЕРЕМЕННАЯ ТИПА **NM**.
ЭТИ ЗНАЧЕНИЯ СЧИТАЮТСЯ УПОРЯДОЧЕННЫМИ, Т.Е.
ОПИСАНИЕ ТИПА ОДНОВРЕМЕННО ВВОДИТ УПОРЯДОЧЕНИЕ

```
WORD1<WORD2<....WORDN.
```

ПРИМЕР.

```
TYPE COLOR(RED,YELLOW,GREEN,BLUE);
```

ЗДЕСЬ `RED<YELLOW<GREEN<BLUE.`

КО ВСЕМ ПЕРЕМЕННЫМ СКАЛЯРНОГО ТИПА, КРОМЕ **REAL**, ПРИМЕНИМЫ СЛЕДУЮЩИЕ СТАНДАРТНЫЕ ФУНКЦИИ:

SUCC(X) - ВЫДАЕТ ЭЛЕМЕНТ, СЛЕДУЮЩИЙ ЗА **X**,
PRED(X) - ВЫДАЕТ ЭЛЕМЕНТ, ПРЕДШЕСТВУЮЩИЙ **X**,
ORD(X) - ПОРЯДКОВЫЙ НОМЕР ЭЛЕМЕНТА **X**.
ЭТА ФУНКЦИЯ НЕ ПРИМЕНИМА К ПЕРЕМЕННЫМ ТИПА **INTEGER**.

В ПРЕДЫДУЩЕМ ПРИМЕРЕ ПРИМЕНЕНИЕ ЭТИХ ФУНКЦИЙ ДАЕТ:

```
SUCC(GREEN)=BLUE
PRED(GREEN)=YELLOW
 ORD(GREEN)=2,
```

ТАК КАК НУМЕРАЦИЯ ЭЛЕМЕНТОВ НАЧИНАЕТСЯ С НУЛЯ.

КО ВСЕМ ПЕРЕМЕННЫМ ОДНОГО И ТОГО ЖЕ ТИПА ПРИМЕНИМЫ ОПЕРАЦИИ ОТНОШЕНИЯ:

=, <>("НЕ РАВНО"), <=(≤), >=(≥), >, <.

6.5. ТИП ОГРАНИЧЕННЫЙ (SUBRANGE).

ДЛЯ ПЕРЕМЕННОЙ МОЖНО УКАЗАТЬ НЕКОТОРОЕ ПОДМНОЖЕСТВО, ЗНАЧЕНИЕ ИЗ КОТОРОГО МОЖЕТ ПРИНИМАТЬ ДАННАЯ ПЕРЕМЕННАЯ.
ОБЩИЙ ВИД:

```
A:MIN..MAX;
```

ЗДЕСЬ
- A - ПЕРЕМЕННАЯ ТИПА **SUBRANGE**,
- MIN - ЛЕВАЯ ГРАНИЦА,
- MAX - ПРАВАЯ ГРАНИЦА ПОДМНОЖЕСТВА.

ТИП MIN И MAX ЗАДАЕТ МНОЖЕСТВО, ОПРЕДЕЛЯЮЩЕЕ ОСНОВНОЙ ТИП ПЕРЕМЕННОЙ A (БАЗОВЫЙ ТИП).

ПРИМЕР.

```
K:1..20;          - ОСНОВНЫМ ТИПОМ ПЕРЕМЕННОЙ K
                    ЯВЛЯЕТСЯ ТИП INTEGER.
B:RED..YELLOW;    - ОСНОВНЫМ ТИПОМ B - ТИП
                    COLOR, ОПИСАННЫЙ ВЫШЕ.
MIN ВСЕГДА ДОЛЖЕН БЫТЬ МЕНЬШЕ MAX.
```

7. СТРУКТУРА ПРОГРАММЫ

ПРОГРАММА СОСТОИТ ИЗ ЗАГОЛОВКА И БЛОКА (ДЕКЛАРАТИВНОЙ ЧАСТИ И ВЫПОЛНЯЕМЫХ ОПЕРАТОРОВ).

7.1. ЗАГОЛОВОК ПРОГРАММЫ

В ЗАГОЛОВКЕ УКАЗЫВАЕТСЯ ИМЯ ПРОГРАММЫ И СПИСОК ВНЕШНИХ ФАЙЛОВ.
ОБЩИЙ ВИД:

```
PROGRAM N(INPUT,OUTPUT,X,Y,...,);
```

ЗДЕСЬ
- N - ИМЯ ПРОГРАММЫ,
- INPUT - ФАЙЛ ВВОДА - УКАЗЫВАЕТСЯ, ЕСЛИ В ПРОГРАММЕ ЕСТЬ ОПЕРАТОР **READ**;
- OUTPUT - ФАЙЛ ВЫВОДА - УКАЗЫВАЕТСЯ ВСЕГДА;
- X,Y - ВНЕШНИЕ ФАЙЛЫ, ИСПОЛЬЗУЕМЫЕ В ПРОГРАММЕ (СМ.П.II.4.I).

7.2. ДЕКЛАРАТИВНАЯ ЧАСТЬ ПРОГРАММЫ

В ДЕКЛАРАТИВНОЙ ЧАСТИ ПОМЕЩАЕТСЯ ИНФОРМАЦИЯ О МЕТКАХ, КОНСТАНТАХ,ТИПАХ И ПЕРЕМЕННЫХ,ВСТРЕЧАЮЩИХСЯ В ПРОГРАММЕ.

7.2.1. РАЗДЕЛ МЕТОК (LABEL)

ЛЮБОЙ ВЫПОЛНЯЕМЫЙ ОПЕРАТОР МОЖЕТ БЫТЬ СНАБЖЕН МЕТКОЙ ЦЕЛОЙ ПОЛОЖИТЕЛЬНОЙ КОНСТАНТОЙ, СОДЕРЖАЩЕЙ НЕ БОЛЕЕ 4-Х ЦИФР. ВСЕ МЕТКИ,ВСТРЕЧАЮЩИЕСЯ В ПРОГРАММЕ,ДОЛЖНЫ БЫТЬ ОПИСАНЫ В РАЗДЕЛЕ LABEL.
ОБЩИЙ ВИД:

```
LABEL L1,L2,L3....;
```

ЗДЕСЬ L1,L2,L3,... - МЕТКИ.

ПРИМЕР.

```
LABEL 5,10,100;
```

МЕТКА ОТДЕЛЯЕТСЯ ОТ ОПЕРАТОРА ДВОЕТОЧИЕМ.

ПРИМЕР.

ПУСТЬ ОПЕРАТОР A:=B ИМЕЕТ МЕТКУ 20.
ТОГДА ЭТОТ ОПЕРАТОР ВЫГЛЯДИТ ТАК:

```
20:A:=B;
```

7.2.2. РАЗДЕЛ КОНСТАНТ (CONST).

ЕСЛИ В ПРОГРАММЕ ИСПОЛЬЗУЮТСЯ КОНСТАНТЫ, ИМЕЮЩИЕ ДОСТАТОЧНО ГРОМОЗДКУЮ ЗАПИСЬ (НАПРИМЕР, ЧИСЛО ПИ С 8-Ю ЗНАКАМИ), ЛИБО СМЕННЫЕ КОНСТАНТЫ (НАПРИМЕР, ДЛЯ ЗАДАНИЯ ВАРИАНТА ПРОГРАММЫ), ТО ТАКИЕ КОНСТАНТЫ ОБЫЧНО ОПИСЫВАЮТСЯ В РАЗДЕЛЕ CONST, А В ПРОГРАММЕ ИСПОЛЬЗУЮТСЯ ИМЕНА КОНСТАНТ. ЭТО ДЕЛАЕТ ПРОГРАММУ БОЛЕЕ НАГЛЯДНОЙ И УДОБНОЙ ПРИ ОТЛАДКЕ И ВНЕСЕНИИ ИЗМЕНЕНИЙ.

```
CONST A1=C1; A2=C2; ....
```

ЗДЕСЬ AI - ИМЯ КОНСТАНТЫ, CI - ЗНАЧЕНИЕ КОНСТАНТЫ.

ПРИМЕР. `CONST PI=3.14;`

7.2.3. РАЗДЕЛ ТИПОВ (TYPE)

ЕСЛИ В ПРОГРАММЕ ВВОДИТСЯ ТИП, ОТЛИЧНЫЙ ОТ СТАНДАРТНОГО (REAL,BOOLEAN И Т.Д.), ТО ЭТОТ ТИП ОПИСЫВАЕТСЯ В РАЗДЕЛЕ TYPE:

```
TYPE T1=(WORD1,WORD2,...);
     T2=..... ;
```

ГДЕ WORD1,WORD2,... - ИДЕНТИФИКАТОРЫ.

ПРИМЕР. `TYPE COLOR=(RED,YELLOW,GREEN,BLUE);`

7.2.4. РАЗДЕЛ ПЕРЕМЕННЫХ (VAR)

ЕСЛИ В ПРОГРАММЕ ВСТРЕЧАЮТСЯ ПЕРЕМЕННЫЕ V11,V12,..., ТО ВСЕ ОНИ ДОЛЖНЫ БЫТЬ ОПИСАНЫ СЛЕДУЮЩИМ ОБРАЗОМ:

```
VAR V11,V12,..:TYPE1; V21,V22,..:TYPE2;...
```

ЗДЕСЬ V11,V12,... - ИМЕНА ПЕРЕМЕННЫХ,
TYPE1 - ТИП ПЕРЕМЕННЫХ V11,V12,...

ПРИМЕР. `VAR I,J:INTEGER; A,B:REAL;`

КАЖДАЯ ПЕРЕМЕННАЯ ДОЛЖНА БЫТЬ ОПИСАНА ДО ЕЕ ИСПОЛЬЗОВАНИЯ В ПРОГРАММЕ И ОТНЕСЕНА К ОДНОМУ И ТОЛЬКО ОДНОМУ ТИПУ. НАЗВАНИЕ РАЗДЕЛОВ (CONST,TYPE,VAR...) УКАЗЫВАЮТСЯ ТОЛЬКО ОДИН РАЗ.

ПРИМЕР.
```
VAR A:REAL;
    B:TYPE1;
```

РАЗДЕЛЫ LABEL,CONST,TYPE МОГУТ ОТСУТСТВОВАТЬ; РАЗДЕЛ VAR В ПРОГРАММЕ ОБЯЗАТЕЛЕН.

7.3. ВЫПОЛНЯЕМАЯ ЧАСТЬ ПРОГРАММЫ
(ОПИСАНИЕ ДЕЙСТВИЙ)

ЭТА ЧАСТЬ ПРОГРАММЫ НАЧИНАЕТСЯ С BEGIN И ЗАКАНЧИВАЕТСЯ ОПЕРАТОРОМ END.
ОПЕРАТОРЫ ОТДЕЛЯЮТСЯ ДРУГ ОТ ДРУГА ТОЧКОЙ С ЗАПЯТОЙ.
ЕСЛИ ОПЕРАТОР СТОИТ ПЕРЕД END,UNTIL И ELSE ТО В ЭТОМ СЛУЧАЕ ТОЧКА С ЗАПЯТОЙ НЕ СТАВИТСЯ.

8. ОПЕРАТОРЫ

ПОД ОПЕРАТОРАМИ ЯЗЫК ПАСКАЛЬ ПОДРАЗУМЕВАЕТ ТОЛЬКО ОПИСАНИЕ ДЕЙСТВИЙ.

8.1. ОПЕРАТОР ПРИСВАИВАНИЯ

ОБЩИЙ ВИД: V:=A;

ЗДЕСЬ V - ПЕРЕМЕННАЯ,
A - ВЫРАЖЕНИЕ,
:= - ОПЕРАЦИЯ ПРИСВАИВАНИЯ.

ВЫРАЖЕНИЕ А МОЖЕТ СОДЕРЖАТЬ КОНСТАНТЫ, ПЕРЕМЕННЫЕ, НАЗВАНИЯ ФУНКЦИЙ, ПРОЦЕДУР, ЗНАКИ ОПЕРАЦИЙ И СКОБКИ.

ВИД ВЫРАЖЕНИЯ ОДНОЗНАЧНО ОПРЕДЕЛЯЕТ ПРАВИЛА ЕГО ВЫЧИСЛЕНИЯ: ДЕЙСТВИЯ ВЫПОЛНЯЮТСЯ СЛЕВА НАПРАВО С СОБЛЮДЕНИЕМ СЛЕДУЮЩЕГО СТАРШИНСТВА ОПЕРАЦИЙ(В ПОРЯДКЕ УБЫВАНИЯ)

1) NOT;
2) *,/, DIV,MOD,AND;
3) +,-, OR;
4) =,< >,<,>,<=,>=, IN.

ЛЮБОЕ ВЫРАЖЕНИЕ В СКОБКАХ ВЫЧИСЛЯЕТСЯ РАНЬШЕ, ЧЕМ ВЫПОЛНЯЕТСЯ ПРЕДШЕСТВУЮЩАЯ СКОБКАМ И ПОСЛЕДУЮЩАЯ ОПЕРАЦИЯ.

ПРИСВАИВАНИЕ ДОПУСКАЕТСЯ ДЛЯ ПЕРЕМЕННЫХ ВСЕХ ТИПОВ,ЗА ИСКЛЮЧЕНИЕМ ТИПА ФАЙЛ.

В ОПЕРАТОРЕ V:=A ПЕРЕМЕННАЯ V И ВЫРАЖЕНИЕ A ДОЛЖНЫ ИМЕТЬ ОДИН И ТОТ ЖЕ ТИП, А ДЛЯ ТИПА **SUBRANGE** - ОДНО И ТО ЖЕ ПОДМНОЖЕСТВО ЗНАЧЕНИЙ.

ИСКЛЮЧЕНИЕ. РАЗРЕШАЕТСЯ ПРИСВАИВАТЬ ПЕРЕМЕННОЙ ТИПА **REAL** ВЫРАЖЕНИЕ ТИПА **INTEGER** И ПЕРЕМЕННОЙ ТИПА **INTEGER** ВЫРАЖЕНИЕ ТИПА **REAL**. В ПОСЛЕДНЕМ СЛУЧАЕ ОТБРАСЫВАЮТСЯ ЦИФРЫ, СТОЯЩИЕ ПРАВЕЕ ДЕСЯТИЧНОЙ ТОЧКИ.

8.2. ОПЕРАТОР ВЫВОДА НА ПЕЧАТЬ

ОБЩИЙ ВИД WRITELN(P1,P2...PN);

ЗДЕСЬ P1,P2...PN - СПИСОК ВЫРАЖЕНИЙ,ВЫВОДИМЫХ НА ПЕЧАТЬ.

ПРИМЕР 1. ВЫЧИСЛИТЬ ДЛИНУ ОКРУЖНОСТИ РАДИУСА 5,782.

```
PROGRAM T10(OUTPUT); ЛИБО PROGRAM T11(OUTPUT);
CONST R=5;           I  VAR R:REAL;
VAR L: REAL;         I    BEGIN R:=5.782;
  BEGIN              I      WRITELN(' ' =',2*3.14*R)
   L:=2*3.1416*R;    I    END.
   WRITELN(' L=', L) I
  END.               I
```

ДЛЯ ОПЕРАТОРА **WRITELN** НЕ ТРЕБУЕТСЯ ОПЕРАТОР ТИПА ФОРТРАННОГО **FORMAT**, Т.К. ТИП ВЫРАЖЕНИЯ **(2*3.14*R)** ЛИБО ТИП ПЕРЕМЕННОЙ **(L)** ОПРЕДЕЛЯЮТ НЕКОТОРУЮ СПЕЦИФИКАЦИЮ, ВЫБРАННУЮ ПО УМОЛЧАНИЮ ДЛЯ ПЕРЕМЕННЫХ ДАННОГО ТИПА **(REAL)**. СМ 10.2

ПЕРВЫЙ СИМВОЛ СТРОКИ НА ПЕЧАТЬ НЕ ВЫВОДИТСЯ, А СЛУЖИТ ДЛЯ УПРАВЛЕНИЯ УСТРОЙСТВОМ ПЕЧАТИ: ЕСЛИ ЭТО 1, ТО УСТРОЙСТВО НАЧНЕТ ПЕЧАТАТЬ С НАЧАЛА НОВОЙ СТРАНИЦЫ, ЕСЛИ "ПРОБЕЛ" - СО СЛЕДУЮЩЕЙ СТРОКИ, ЕСЛИ "+"- БЕЗ ПЕРЕХОДА К НОВОЙ СТРОКЕ, Т.Е. С НАЛОЖЕНИЕМ СТРОК.

ЕСЛИ ОПЕРАТОР ВЫВОДА НА ПЕЧАТЬ СОСТАВЛЕН БЕЗ УЧЕТА РОЛИ ПЕРВОГО СИМВОЛА В СТРОКЕ, ТО МОГУТ БЫТЬ НЕПРЕДВИДЕННЫЕ РЕЖИМЫ ПЕЧАТИ: ПРОПУСК СТРАНИЦ, СТРОК, НАЛОЖЕНИЕ СТРОК.

8.4. ОПЕРАТОР БЕЗУСЛОВНОГО ПЕРЕХОДА GOTO

ОБЩИЙ ВИД: `GOTO N;`

МЕТКА N, НА КОТОРУЮ ПЕРЕДАЕТСЯ УПРАВЛЕНИЕ, ДОЛЖНА БЫТЬ ОПИСАНА В РАЗДЕЛЕ **LABEL**.

8.5. СОСТАВНОЙ ОПЕРАТОР

ЕСЛИ ПРИ НЕКОТОРОМ УСЛОВИИ НАДО ВЫПОЛНИТЬ ОПРЕДЕЛЕННУЮ ПОСЛЕДОВАТЕЛЬНОСТЬ ОПЕРАТОРОВ, ТО ИХ ОБ'ЕДИНЯЮТ В ОДИН СОСТАВНОЙ ОПЕРАТОР.

СОСТАВНОЙ ОПЕРАТОР НАЧИНАЕТСЯ СЛОВОМ **BEGIN** И ЗАКАНЧИВАЕТСЯ СЛОВОМ **END**
МЕЖДУ ЭТИМИ СЛОВАМИ ПОМЕЩАЮТСЯ СОСТАВЛЯЮЩИЕ ОПЕРАТОРЫ, КОТОРЫЕ ВЫПОЛНЯЮТСЯ В ПОРЯДКЕ ИХ СЛЕДОВАНИЯ.

ПРИМЕР.

```
BEGIN
  I:=2;
  K:=I*5
END;
```

СЛОВА **BEGIN** И **END** ИГРАЮТ РОЛЬ ОПЕРАТОРНЫХ СКОБОК. ТЕЛО САМОЙ ПРОГРАММЫ ТАКЖЕ ИМЕЕТ ВИД СОСТАВНОГО ОПЕРАТОРА. ПОСЛЕ ПОСЛЕДНЕГО **END** ПРОГРАММЫ СТАВИТСЯ ТОЧКА (**END.**).

8.6. ОПЕРАТОРЫ ЦИКЛА

В ЯЗЫКЕ ПАСКАЛЬ ИМЕЕТСЯ ТРИ ВИДА ОПЕРАТОРОВ ЦИКЛА:

WHILE, REPEAT И FOR.

8.6.1. ОПЕРАТОР ЦИКЛА WHILE

ОБЩИЙ ВИД: **WHILE A DO ST;**

ЗДЕСЬ **A** - БУЛЕВСКОЕ ВЫРАЖЕНИЕ.
ST - ОПЕРАТОР (ПРОСТОЙ ЛИБО СОСТАВНОЙ).

ВЫРАЖЕНИЕ A ВЫЧИСЛЯЕТСЯ ПЕРЕД КАЖДЫМ ВЫПОЛНЕНИЕМ ОПЕРАТОРА **ST**. ЕСЛИ A - **TRUE**, ТО **ST** ВЫПОЛНЯЕТСЯ; ЕСЛИ A - **FALSE**, ТО **ST** НЕ ВЫПОЛНЯЕТСЯ.

ЕСЛИ ПЕРВОНАЧАЛЬНОЕ ЗНАЧЕНИЕ A - **FALSE**, ТО **ST** НЕ БУДЕТ ВЫПОЛНЕН НИ РАЗУ.

ПРИМЕР.

```
WHILE X>0 DO
 BEGIN C:=C+1/X
       X:=X-1
 END;
```

ЗАДАЧА 1.

ВЫЧИСЛИТЬ СУММУ **S=1+1/2+1/3+...+1/50**

```
PROGRAM N1 (OUTPUT);
VAR S:REAL;N:INTEGER;
 BEGIN
   S:=0; N:=1;
   WHILE N<=50 DO
      BEGIN S:=S+1/N;
          N:=N+1
      END;
   WRITELN(' S=',S)
 END.
```

8.6.2. ОПЕРАТОР ЦИКЛА REPEAT

ОБЩИЙ ВИД: REPEAT STS UNTIL A;

ЗДЕСЬ STS - ГРУППА ВЫПОЛНЯЕМЫХ ОПЕРАТОРОВ;
A - БУЛЕВСКОЕ ВЫРАЖЕНИЕ.

РАБОТАЕТ ОПЕРАТОР ТАК:

ВЫПОЛНЯЮТСЯ ОПЕРАТОРЫ STS,
ВЫЧИСЛЯЕТСЯ ВЫРАЖЕНИЕ A ; ЕСЛИ ОНО "ЛОЖЬ" (FALSE),
ТО
ВЫПОЛНЯЮТСЯ ОПЕРАТОРЫ STS, ЕСЛИ "ИСТИНА" (TRUE) - НЕТ

ЕСЛИ A -"ИСТИНА" С САМОГО НАЧАЛА, ТО STS ВЫПОЛНЯЮТСЯ ОДИН РАЗ.
ЕСЛИ A НИКОГДА НЕ ПРИНИМАЕТ ЗНАЧЕНИЕ "ИСТИНА", ТО ГРУППА ОПЕРАТОРОВ STS ВЫПОЛНЯЕТСЯ БЕСКОНЕЧНОЕ ЧИСЛО РАЗ (ЗАЦИКЛИВАНИЕ!)

ЗАДАЧА 2.

РЕШИТЬ ЗАДАЧУ 1 С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА REPEAT

```
PROGRAM N2 (OUTPUT);
VAR S:REAL;N:INTEGER;
   BEGIN
     S:=0;N:=1;
     REPEAT S:=S+1/N;N:=N+1;
     UNTIL N>50;
     WRITELN(' S=',S)
   END.
```

8.6.3. ОПЕРАТОР ЦИКЛА FOR

ОБЩИЙ ВИД: FOR I:=N1 TO N2 DO ST;

ЗДЕСЬ I - ПАРАМЕТР ЦИКЛА,
N1 - НАЧАЛЬНОЕ ЗНАЧЕНИЕ ПАРАМЕТРА
N2 - КОНЕЧНОЕ ЗНАЧЕНИЕ,
ST - ОПЕРАТОР (ПРОСТОЙ ЛИБО СОСТАВНОЙ).

I,N1 И N2 ДОЛЖНЫ БЫТЬ ОДНОГО И ТОГО ЖЕ СКАЛЯРНОГО ТИПА, НО НЕ REAL.

I ПРИНИМАЕТ ПОСЛЕДОВАТЕЛЬНЫЕ ЗНАЧЕНИЯ ОТ N1 ДО N2.

ЕСЛИ N1 И N2 -ЦЕЛЫЕ ЧИСЛА, А I -ЦЕЛАЯ ПЕРЕМЕННАЯ, ТО ШАГ РАВЕН ЕДИНИЦЕ.

ПРИМЕР.

```
FOR I:=1 TO 20 DO A:=A+1;
```

ЕСЛИ N1 И N2 СИМВОЛЬНОГО ТИПА - НАПРИМЕР, СООТВЕТСТВЕННО ИМЕЮТ ЗНАЧЕНИЯ A И Z, ТО ПЕРЕМЕННАЯ I ПРИНИМАЕТ ПОСЛЕДОВАТЕЛЬНЫЕ ЗНАЧЕНИЯ В ПОРЯДКЕ АЛФАВИТА: A,B,C,...,Z.

ЕСЛИ N1 И N2 ТИПА COLOR (ТИП COLOR=(RED,YELLOW,GREEN,BLUE)), НАПРИМЕР, RED И GREEN, СООТВЕТСТВЕННО, ТО ПЕРЕМЕННАЯ I ПРИНИМАЕТ ЗНАЧЕНИЯ RED,YELLOW,GREEN.

ЦИКЛ ПО УБЫВАЮЩИМ ЗНАЧЕНИЯМ ПАРАМЕТРА ОТ N2 ДО N1 ИМЕЕТ ВИД:

```
FOR I:=N2 DOWNTO N1 DO ST;
```

В ЭТОМ СЛУЧАЕ ПАРАМЕТР I ПРИНИМАЕТ ПОСЛЕДОВАТЕЛЬНЫЕ УБЫВАЮЩИЕ ЗНАЧЕНИЯ ДАННОГО ТИПА ОТ N2 ДО N1.

ПРИМЕР.

```
FOR I:=20 DOWNTO 1 DO A:=A+1;
```

I ИЗМЕНЯЕТСЯ ОТ 20 ДО 1 С ШАГОМ 1.

8.7.ОПЕРАТОР УСЛОВНОГО ПЕРЕХОДА ИМЕЕТ ДВЕ РАЗНОВИДНОСТИ: IF И CASE

8.7.1.ОПЕРАТОР IF

ОБЩИЙ ВИД1: `IF A THEN ST;`

ЗДЕСЬ A - БУЛЕВСКОЕ ВЫРАЖЕНИЕ,
ST - ОПЕРАТОР (ПРОСТОЙ ЛИБО СОСТАВНОЙ),
ЕСЛИ A - "ИСТИНА", ВЫПОЛНЯЕТСЯ ОПЕРАТОР ST,
ЕСЛИ A - "ЛОЖЬ", ТО УПРАВЛЕНИЕ СРАЗУ ПЕРЕДАЕТСЯ СЛЕДУЮЩЕМУ ЗА ST ОПЕРАТОРУ.

ПРИМЕР.

```
IF A<>0 THEN B:=X/A;  (ПАРА СИМВОЛОВ <> ОЗНАЧАЕТ
                       " НЕ РАВНО").
```

ОПЕРАТОР IF МОЖЕТ ИМЕТЬ И ТАКОЙ ВИД (ВИД 2):

```
IF A THEN ST1 ELSE ST2;
```

ЗДЕСЬ A - БУЛЕВСКОЕ ВЫРАЖЕНИЕ,
ST1,ST2 - ОПЕРАТОРЫ.
ЕСЛИ A - "ИСТИНА", ВЫПОЛНЯЕТСЯ ОПЕРАТОР ST1
ЕСЛИ A - "ЛОЖЬ" -ST2.

ЗАМЕЧАНИЕ 1. ПЕРЕД **ELSE** НЕЛЬЗЯ СТАВИТЬ ТОЧКУ С ЗАПЯТОЙ.

ЗАМЕЧАНИЕ 2. СИНТАКСИЧЕСКАЯ НЕОДНОЗНАЧНОСТЬ ОПЕРАТОРА

```
IF A1 THEN IF A2 THEN ST1 ELSE ST2
```

ТРАКТУЕТСЯ ТАК

```
IF A1 THEN
   BEGIN IF A2 THEN ST1 ELSE ST2 END;
```

8.7.2. ОПЕРАТОР CASE

ОБЩИЙ ВИД:

```
CASE  N   OF
    M1,...,MN: ST1;
    K1,...,KL: ST2;
                .
                .
                .
                .
END;
```

ЗДЕСЬ N -ПЕРЕКЛЮЧАТЕЛЬ(СЕЛЕКТОР),
MI,KI - МЕТКИ (КОНСТАНТЫ).

ПЕРЕКЛЮЧАТЕЛЬ И МЕТКИ ДОЛЖНЫ БЫТЬ ОДНОГО И ТОГО ЖЕ СКАЛЯРНОГО ТИПА, НО НЕ **REAL**.

ОПЕРАТОР **CASE** ПЕРЕДАЕТ УПРАВЛЕНИЕ ТОМУ ОПЕРАТОРУ **STI**, С ОДНОЙ ИЗ МЕТОК КОТОРОГО СОВПАЛО ЗНАЧЕНИЕ ПЕРЕКЛЮЧАТЕЛЯ **N**, А ЗАТЕМ - НА СЛЕДУЮЩИЙ ЗА **CASE** ОПЕРАТОР.

ПРИМЕР 1.

```
CASE  I OF
   2: X:=0;
   3: X:=X*X;
 100: X:=SIN(X);
END;
```

ЕСЛИ ЗНАЧЕНИЕ I ЕСТЬ 3, ТО ВЫПОЛНИТСЯ X:=X*X;

ЗАМЕЧАНИЕ.

МЕТКИ ОПЕРАТОРА **CASE** НЕ ОПИСЫВАЮТСЯ В РАЗДЕЛЕ **LABEL** И НА НИХ НЕЛЬЗЯ ПЕРЕХОДИТЬ ОПЕРАТОРОМ **GOTO**.

9. ПРОЦЕДУРА ВВОДА

ОБЩИЙ ВИД:

```
READ(V1,V2,.... VN);
```

ЗДЕСЬ V1,V2,...VN - ИДЕНТИФИКАТОРЫ ПЕРЕМЕННЫХ. ЗНАЧЕНИЯ ПЕРЕМЕННЫХ ПРОБИВАЮТСЯ НА КАРТАХ (ЛИБО ВВОДЯТСЯ С ТЕРМИНАЛА) И ДОЛЖНЫ СООТВЕТСТВОВАТЬ ТИПАМ ПЕРЕМЕННЫХ.ПЕРЕМЕННЫЕ V1,V2..,VN МОГУТ БЫТЬ ТОЛЬКО ОДНОГО ИЗ 3-Х ТИПОВ: INTEGER,CHAR (ЛИБО SUBRANGE ЭТИХ ТИПОВ) И REAL.

ЕСЛИ ДАННЫЕ ВВОДЯТСЯ НЕ С ПЕРФОКАРТ, А С КАКОГО-ЛИБО ИНОГО ФАЙЛА ВВОДА F, ТО ИСПОЛЬЗУЕТСЯ ОПЕРАТОР

```
READ(F,V1,V2,...VN);.
```

ЧИСЛА, ПРОБИТЫЕ НА КАРТЕ, МОГУТ БЫТЬ РАЗДЕЛЕНЫ ОДНИМ ИЛИ НЕСКОЛЬКИМИ ПРОБЕЛАМИ, НО НЕЛЬЗЯ ОТДЕЛЯТЬ ПРОБЕЛОМ ЗНАК ЧИСЛА ЛИБО ОДНИ ЦИФРЫ ЧИСЛА ОТ ДРУГИХ.

ПРИМЕР.

```
PROGRAM N3(INPUT,OUTPUT);
  VAR A,B,C:REAL; I:INTEGER;
    BEGIN
      READ(A,B,C,I);
      WRITELN(' A=',A,' B=',B,' C=',C,' I=',I);
    END.
```

ДЛЯ ВВОДА ЧИСЕЛ 1.5, 2.15, -1.1, 25 ИХ МОЖНО ПРОБИТЬ ТАК:

```
1.5 2.15 -1.1 25.
```

ЕСЛИ ВВОДИТСЯ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ,ТО ПРОБЕЛ ВОСПРИНИМАЕТСЯ КАК СИМВОЛ.
В ЭТОМ СЛУЧАЕ И КОНЕЦ СТРОКИ (EOL) ТРАКТУЕТСЯ КАК СИМВОЛ "КОНЕЦ СТРОКИ",А СООТВЕТСТВУЮЩАЯ ПЕРЕМЕННАЯ ПОЛУЧАЕТ ЗНАЧЕНИЕ "ПРОБЕЛ".

ПРИМЕР.

```
ПУСТЬ R:REAL;I:INTEGER;C1,C2,C3:CHAR;

ПЕРЕМЕННЫМ R,C1,C2,C3,I НАДО ПРИСВОИТЬ
СООТВЕТСТВЕННО ЗНАЧЕНИЯ  1.5; 'A','B','C',25;
```

НА КАРТАХ ЭТИ ЗНАЧЕНИЯ МОЖНО РАСПОЛОЖИТЬ ОДНИМ ИЗ СПОСОБОВ:

```
А)      1.5ABC25
Б)      +1.5E-1ABC
        25
```

ОПЕРАТОР READ(R,C1,C2,C3,I); ВВЕДЕТ НЕОБХОДИМЫЕ ДАННЫЕ.

КОНЕЦ ВХОДНОГО ФАЙЛА И КОНЕЦ СТРОКИ (КАРТЫ) ВХОДНОГО ФАЙЛА МОЖНО ОПРЕДЕЛИТЬ С ПОМОЩЬЮ ФУНКЦИЙ **EOF** И **EOLN** СООТВЕТСТВЕННО.
ФУНКЦИЯ **EOF** ПРИНИМАЕТ ЗНАЧЕНИЕ **TRUE** В СЛУЧАЕ ИСЧЕРПАНИЯ ВСЕХ ВХОДНЫХ ДАННЫХ.
ЕСЛИ ИСЧЕРПАЛИСЬ ДАННЫЕ НА ОДНОЙ КАРТЕ (СТРОКЕ ВХОДНОГО ФАЙЛА), ТО ПРИНИМАЕТ ЗНАЧЕНИЕ **TRUE** ДРУГАЯ ФУНКЦИЯ - **EOLN**.

10. ПРОЦЕДУРЫ ВЫВОДА

10.1. ПРОЦЕДУРА **WRITE(P1,P2,...PN)** ДОПИСЫВАЕТ В ВЫХОДНОЙ ФАЙЛ **OUTPUT** ЗНАЧЕНИЯ ВЫРАЖЕНИЙ **P1,P2...PN**.

ПРОЦЕДУРА **WRITELN(P1,P2,...PN)** ДОПИСЫВАЕТ В **OUTPUT** ЗНАЧЕНИЯ **P1,P2,P3,...PN**, ЗАНОСИТ ПРИЗНАК КОНЦА СТРОКИ **EOLN** И ВЫДАЕТ СТРОКУ НА ПЕЧАТЬ.

ПРИМЕР.

ВЫДАЧА НА ПЕЧАТЬ ЗНАЧЕНИЙ A,B МОЖЕТ БЫТЬ ОСУЩЕСТВЛЕНА ПО-РАЗНОМУ:

```
А) WRITE(A);    Б) WRITE(A);     В) WRITELN(A,B);
   WRITE(B);       WRITELN(B);
   WRITELN.
```

10.2. ФОРМАТ ВЫВОДА НА ПЕЧАТЬ

ТРАНСЛЯТОР ОТВОДИТ, ПО УМОЛЧАНИЮ, ОПРЕДЕЛЕННОЕ ЧИСЛО ПОЗИЦИЙ ДЛЯ ВЕЛИЧИН КАЖДОГО СТАНДАРТНОГО ТИПА.

ТИП	CHAR	ALFA	BOOLEAN	INTEGER	REAL M	REAL N
ЧИСЛО ПОЗИЦИЙ НА БЭСМ-6	1	6	8	10	14	4
ЧИСЛО ПОЗИЦИЙ НА CDC-6500	1	10	10	10	22	1
ЧИСЛО ПОЗИЦИЙ НА ЕС ЭВМ	1	8	5(4)	12	24	16

ПРОГРАММИСТ ИМЕЕТ ВОЗМОЖНОСТЬ ЗАДАТЬ ШИРИНУ ПОЛЯ M ДЛЯ ВЫВОДИМОЙ ВЕЛИЧИНЫ P:

```
WRITE(P1:M1, P2:M2,...,PN:MN);
```

ЕСЛИ MI ИЗБЫТОЧНА, ТО ПОЛЕ СЛЕВА ДОПОЛНЯЕТСЯ ПРОБЕЛАМИ; ЕСЛИ MI НЕДОСТАТОЧНА ДЛЯ РАЗМЕЩЕНИЯ ЗНАЧЕНИЯ PI , ТО ТРАНСЛЯТОР САМ УВЕЛИЧИВАЕТ ШИРИНУ ПОЛЯ ТАК, ЧТОБЫ УМЕСТИЛОСЬ PI, А ДЛЯ БЭСМ-6 И CDC СЛЕВА ОСТАВЛЯЕТ ЕЩЕ ОДИН ПРОБЕЛ (КРОМЕ ВЕЛИЧИН ТИПА INTEGER, ГДЕ ПРОБЕЛ НЕ ПРЕДУСМОТРЕН).

ДЛЯ ВЕЩЕСТВЕННЫХ ЗНАЧЕНИЙ МОЖНО ЗАДАВАТЬ ПОЛЯ M И N, ГДЕ M -ОБЩЕЕ ЧИСЛО ПОЗИЦИЙ, ОТВОДИМЫХ ПОД ВСЕ PI ,N -ПОД ДРОБНУЮ ЕГО ЧАСТЬ.

ПРИМЕР:

```
WRITE(P:10:2);
```

ЗДЕСЬ ПОД P ОТВОДИТСЯ 10 ПОЗИЦИЙ, А 2 ПОЗИЦИИ - ПОД ДРОБНУЮ ЧАСТЬ.

ПРИМЕР. ОПЕРАТОР

```
( WRITELN(3.14159:0:2, 1=1, 3.14:7,56,'PAPA':5);
```

СОДЕРЖИТ ПЯТЬ ВЫРАЖЕНИЙ:

```
P1=3.14159
P2=TRUE
P3=3.14
P4=56
P5='PAPA'
```

ОНИ ИЗОБРАЗЯТСЯ НА ЛИСТИНГЕ СЛЕДУЮЩИМ ОБРАЗОМ:

А) НА БЭСМ-6

```
3.14E+00    TRUE 3.1400E+00        56 PAPA
```

Б) CDC-6500

```
3.14        TRUE 3.1E+0000         56 PAPA
```

В) НА ЕС ЭВМ НЕЛЬЗЯ УКАЗЫВАТЬ НУЛЕВОЕ ПОЛЕ И НЕОБХОДИМО, ЧТОБЫ БЫЛО M>N.

11. СЛОЖНЫЕ ТИПЫ

11.1. МАССИВ (ARRAY).

МАССИВ - СТРУКТУРА, СОСТОЯЩАЯ ИЗ ФИКСИРОВАННОГО ЧИСЛА КОМПОНЕНТ ОДНОГО ТИПА.

ОБЩИЙ ВИД ОПИСАНИЯ ПЕРЕМЕННОЙ:

```
A:ARRAY[TYPE1,TYPE2,...TYPEL] OF TYPEC;
```

ЗДЕСЬ A - ИМЯ ПЕРЕМЕННОЙ,
TYPE1,TYPE2,...TYPEL - ТИПЫ ИНДЕКСОВ,
TYPEC - ТИП КОМПОНЕНТ (БАЗОВЫЙ ТИП).

ИНДЕКСЫ МОГУТ ИМЕТЬ ЛЮБЫЕ СКАЛЯРНЫЕ ТИПЫ, КРОМЕ **REAL** И **INTEGER** (НЕ ПУТАТЬ С **SUBRANGE**). ИНДЕКСЫ РАЗДЕЛЯЮТСЯ ЗАПЯТЫМИ И ЗАКЛЮЧАЮТСЯ В КВАДРАТНЫЕ СКОБКИ.

КОМПОНЕНТЫ МАССИВА МОГУТ ИМЕТЬ ЛЮБОЙ ТИП.
НА ЕС ЭВМ СИМВОЛЫ [И] СООТВЕТСТВУЮТ ПАРАМ СИМВОЛОВ:
"(." И ".)"

ПРИМЕР.

```
A:ARRAY[1..5,M..N] OF INTEGER;

B:ARRAY[COLOR] OF BOOLEAN;
```

КОНКРЕТНАЯ КОМПОНЕНТА МАССИВА ОПРЕДЕЛЯЕТСЯ СООТВЕТСТВУЮЩИМИ ЗНАЧЕНИЯМИ ИНДЕКСОВ.

ПРИМЕР.

```
A[3,M]; B[RED];
```

НА РАЗМЕРНОСТЬ МАССИВА ЯЗЫК НЕ НАКЛАДЫВАЕТ ОГРАНИЧЕНИЙ.

ЕСЛИ КОМПОНЕНТЫ МАССИВА A САМИ ТИПА **ARRAY** ТО ГОВОРЯТ, ЧТО МАССИВ A ИМЕЕТ МУЛЬТИРАЗМЕРНОСТЬ.

ПРИМЕР.

```
A:ARRAY[M..N,K..L] OF ARRAY[C..D] OF REAL;
```

В ЭТОМ СЛУЧАЕ **J** -Я КОМПОНЕНТА ОТ **I** -Й КОМПОНЕНТЫ МУЛЬТИРАЗМЕРНОГО МАССИВА ЗАПИСЫВАЕТСЯ КАК **A[MI,KI] [J]**

11.1.1. УПАКОВАННЫЕ МАССИВЫ

ЕСЛИ ПЕРЕД НАЗВАНИЕМ ТИПА СТОИТ "PACKED" ТО ПРИ ТРАНСЛЯЦИИ ГЕНЕРИРУЕТСЯ ПРОГРАММА, ПЛОТНО УПАКОВЫВАЮЩАЯ ДАННЫЕ В ЯЧЕЙКИ ПАМЯТИ. ТАКАЯ ПРОГРАММА ЭКОНОМИТ ПАМЯТЬ, НО НЕ СЧЕТНОЕ ВРЕМЯ.

К УПАКОВАННОМУ МАССИВУ Z МОЖНО ПРИМЕНИТЬ СТАНДАРТНЫЕ ПРОЦЕДУРЫ PACK И UNPACK.
ПУСТЬ B - ИМЕЕТ ТИП ARRAY[M..N] OF T, A Z:PACKED ARRAY[U..V] OF T, ГДЕ N-M>=V-U;

ТОГДА

PACK(B,I,Z) ОЗНАЧАЕТ "УПАКОВКУ", НАЧИНАЯ С I - КОМПОНЕНТЫ МАССИВА B, В МАССИВ Z.

UNPACK(Z,B,I) ОЗНАЧАЕТ "РАСПАКОВКУ" МАССИВА Z В МАССИВ B, НАЧИНАЯ С I -ГО ЭЛЕМЕНТА МАССИВА B.

11.2. ЗАПИСЬ(RECORD)

ЗАПИСЬ - ЭТО СТРУКТУРА, СОСТОЯЩАЯ ИЗ ФИКСИРОВАННОГО ЧИСЛА КОМПОНЕНТ, НАЗЫВАЕМЫХ ПОЛЯМИ. В ОДНОМ ПОЛЕ ДАННЫЕ ИМЕЮТ ОДИН И ТОТ ЖЕ ТИП, А В РАЗНЫХ ПОЛЯХ МОГУТ ИМЕТЬ РАЗНЫЕ ТИПЫ.

ОБЩИЙ ВИД ОПИСАНИЯ ПЕРЕМЕННОЙ ТИПА RECORD:

```
A : RECORD

      ID11,ID12,...ID1N:TYPE1;

      ID21,ID22,...ID2L:TYPE2;
                  .
                  .
                  .

      IDK1,IDK2,...IDKM:TYPEK

    END;
```

ЗДЕСЬ IDIJ - ИДЕНТИФИКАТОРЫ ПОЛЕЙ;
TYPEI - ТИПЫ ПОЛЕЙ.

ПРИМЕР 1.

ДАННЫЕ КОМПЛЕКСНОГО ВИДА МОЖНО ОПИСАТЬ ПЕРЕМЕННОЙ ТИПА RECORD.

```
TYPE COMPLEX=RECORD
                RE,IM:REAL
             END;
VAR C:COMPLEX;
```

ПЕРЕМЕННАЯ С СОСТОИТ ИЗ ДВУХ ПОЛЕЙ: RE И IM.

ПРИМЕР 2.

ДАТЫ КАКИХ-ЛИБО СОБЫТИЙ МОЖНО ОПИСАТЬ СЛЕДУЮЩИМ ОБРАЗОМ:

```
TYPE DATE=RECORD
            MONTH:1..12;
            DAY  :1..31;
            YEAR : INTEGER
          END;
VAR   D : DATE;
```

ПОЛЕ P ПЕРЕМЕННОЙ A ЗАПИСЫВАЕТСЯ КАК A.P

ПРИМЕР 3.

ВЫЧИСЛИТЬ СУММУ S ДВУХ КОМПЛЕКСНЫХ ЧИСЕЛ X=2+7I И Y=6+3I, (T.E.X,Y,S:COMPLEX.)

ФРАГМЕНТ ПРОГРАММЫ ВЫГЛЯДИТ ТАК:

```
X.RE:=2.0;  X.IM:=7.0;
Y.RE:=6,0;  Y.IM:=3.0;
S.RE:=X,RE+Y.RE;
S.IM:=X,IM+Y.IM;
```

ЗАПИСИ, КАК И МАССИВЫ, МОГУТ БЫТЬ УПАКОВАННЫМИ.

ПРИМЕР 4.

```
X:PACKED  RECORD
            A:1..5;
            B:CHAR
          END;
```

ЗАПИСЬ МОЖЕТ БЫТЬ КОМПОНЕНТОЙ ДРУГИХ СТРУКТУР, НАПРИМЕР:

```
TYPE FAMILY=(FATHER,MOTHER,CHILD1,CHILD2);
VAR BIRTHDAY:ARRAY[FAMILY] OF DATE;
```

ГДЕ DATE - ОПИСАННЫЙ ВЫШЕ РЕКОРД.

ДЛЯ ЗАНЕСЕНИЯ ДАТЫ РОЖДЕНИЯ, НАПРИМЕР, МАТЕРИ (MOTHER), ДОСТАТОЧНО ВЫПОЛНИТЬ ОПЕРАТОРЫ:

```
BIRTHDAY[MOTHER].MONTH :=5;
BIRTHDAY[MOTHER].DAY   :=1;
BIRTHDAY[MOTHER].YEAR  :=1950;
```

ЗАНЕСЕТСЯ ДАТА РОЖДЕНИЯ МАТЕРИ - 1/5/1950Г.

11.2.1. ОПЕРАТОР WITH.

ОБЩИЙ ВИД:

```
WITH A DO ST;
```

ЗДЕСЬ A - ИМЯ ПЕРЕМЕННОЙ - ТИПА RECORD,
ST - ОПЕРАТОР.

В ОПЕРАТОРЕ ST ПРИ ССЫЛКАХ НА КОМПОНЕНТЫ ЗАПИСИ ИМЯ A МОЖНО ОПУСКАТЬ.

ПРИМЕР.

```
WITH BIRTHDAY [MOTHER] DO
    BEGIN  MONTH:=5;
           DAY  :=1;
           YEAR := 1950
    END;
```

11.2.2. ЗАПИСИ С ВАРИАНТАМИ

ОБЩИЙ ВИД:

```
TYPE V = RECORD A:TYPE1;
                B:TYPE2;
                  .
                  .
                  .
                CASE N:TYPEN OF
                C1:(T11:TYPE11; T12:TYPE12;...)
                C2:(T21:TYPE21; T22:TYPE22;...)
                  .
                  .
                  .
                CK:(TK1:TYPEK1; TK2:TYPEK2;...)

           END
```

```
VAR  Z:V;
```

ЗДЕСЬ Z - ПЕРЕМЕННАЯ ТИПА RECORD
N - ПЕРЕКЛЮЧАТЕЛЬ ТИПА TYPEN. ЭТОМУ ЖЕ ТИПУ ДОЛЖНЫ ПРИНАДЛЕЖАТЬ ВСЕ МЕТКИ C1,C2,...CK. В ЗАВИСИМОСТИ ОТ ЗНАЧЕНИЯ N (ОНО МОЖЕТ БЫТЬ РАВНО ЗНАЧЕНИЮ C1,C2,...CK), НЕКОТОРАЯ ПЕРЕМЕННАЯ ТИПА V, ПОМИМО ПОЛЕЙ A,B, СОДЕРЖИТ ПОЛЯ, СООТВЕТСТВУЮЩИЕ ТОЙ МЕТКЕ C1,C2,...CK, С КОТОРОЙ СОВПАЛО ЗНАЧЕНИЕ ПЕРЕКЛЮЧАТЕЛЯ N.

ПОЛЯ T11,T12,... НАЗЫВАЮТСЯ КОМПОНЕНТАМИ ВАРИАНТА.

ПОЛЕ ПЕРЕКЛЮЧАТЕЛЯ N ТАКЖЕ НАЗЫВАЮТ ПОЛЕМ ТАГА, ПОЛЕМ ЯРЛЫКА, ПОЛЕМ ПРИЗНАКА, ДИСКРИМИНАНТОМ.

ЕСЛИ КАКОЙ-ЛИБО МЕТКЕ CI ВООБЩЕ НЕ СООТВЕТСТВУЕТ ПОЛЕЙ, ТО ПИШУТ

```
CI:(   );
```

ОБРАЩЕНИЕ К КОМПОНЕНТЕ TI ЗАПИСИ Z ПРОИСХОДИТ СЛЕДУЮЩИМ ОБРАЗОМ:

1) ПРИСВАИВАЕТСЯ СООТВЕТСТВУЮЩЕЕ ЗНАЧЕНИЕ (CI) ПЕРЕКЛЮЧАТЕЛЮ N;

2) ВЫПОЛНЯЕТСЯ ОПЕРАЦИЯ С КОМПОНЕНТОЙ Z.TI.

ПРИМЕР ЗАПИСИ С ВАРИАНТАМИ.

ПУСТЬ НЕОБХОДИМО СОБРАТЬ СЛЕДУЮЩИЕ СВЕДЕНИЯ О СОТРУДНИКАХ ИНСТИТУТА:ФАМИЛИЮ,ДАТУ РОЖДЕНИЯ И ,ЕСЛИ ЕСТЬ СЕМЬЯ, ТО ФАМИЛИЮ И ДАТУ РОЖДЕНИЯ СУПРУГИ(СУПРУГА).

ЭТА ИНФОРМАЦИЯ МОЖЕТ БЫТЬ ОПИСАНА,НАПРИМЕР,ЗАПИСЬЮ PERSON.

ПУСТЬ

```
KIND=(MARRIED,SINGLE)
  PERSON=RECORD
          NAME:ALFA;
          DATEBIRTH:DATE;
        CASE YESNO:KIND OF
          MARRIED:(NAME1:ALFA;DATE1:DATE);
          SINGLE:( )
          END;
```

ЗДЕСЬ NAME - СТРОКА СИМВОЛОВ(НАПРИМЕР, ROGOV);
DATEBIRTH - ЗАПИСЬ,ОПИСАННАЯ ВЫШЕ,СОДЕРЖИТ ДАТУ РОЖДЕНИЯ(НАПРИМЕР, 15/02/62);
YESNO - ПЕРЕКЛЮЧАТЕЛЬ ТИПА KIND ,КОТОРЫЙ МОЖЕТ ПРИНИМАТЬ ОДНО ИЗ ДВУХ ЗНАЧЕНИЙ: MARRIED ЛИБО SINGLE
NAME1 - СТРОКА СИМВОЛОВ,СОДЕРЖАЩАЯ ФАМИЛИЮ СУПРУГИ(СУПРУГА) (НАПРИМЕР, ROGOVA);
DATE1 - ЗАПИСЬ,СОДЕРЖАЩАЯ ДАТУ РОЖДЕНИЯ СУПРУГИ(СУПРУГА).
SINGLE - ПУСТОЕ ПОЛЕ.

ЕСЛИ ROGOV ЖЕНАТ, ТО ПРИСУТСТВУЕТ ПОЛЕ MARRIED, ЕСЛИ ХОЛОСТ - ПОЛЕ SINGLE , А ПОЛЕ MARRIED ОТСУТСТВУЕТ.

ПАСКАЛЬ ДОПУСКАЕТ ВЛОЖЕНИЕ ВАРИАНТОВ В ТИПЕ RECORD.

ПРИМЕР.

ПУСТЬ НЕОБХОДИМО, ПОМИМО ИНФОРМАЦИИ ПРЕДЫДУЩЕГО ПРИМЕРА, ИМЕТЬ О СОТРУДНИКАХ СЛЕДУЮЩИЕ СВЕДЕНИЯ: ЕСЛИ СОТРУДНИК ХОЛОСТ, НО СОСТОЯЛ В БРАКЕ РАНЬШЕ, ТО УКАЗАТЬ КОГДА РАЗВЕДЕН.

ОПИШЕМ ТИП KIND КАК

```
KIND=(MARRIED,SINGLE,DEVORCED,NO);
```

ТОГДА

```
PERSON=RECORD
    NAME:ALFA;
    DATEBIRTH:DATE;
   CASE YESNO:KIND OF
     MARRIED:(NAME1:ALFA;DATE1:DATE);
     SINGLE:(CASE YN:KIND OF
               DEVORCED:(DATEDV:DATE);
               NO:( ));
      END;
```

ЗДЕСЬ ДЛЯ ВАРИАНТА **SINGLE** ИМЕЕТСЯ ВЛОЖЕННАЯ ЗАПИСЬ С ВАРИАНТАМИ **DEVORCED** (РАЗВЕДЕН) И **NO.**

ЕСЛИ СОТРУДНИК СОСТОИТ В БРАКЕ, ТО В ЗАПИСЯХ ИНФОРМАЦИИ ОТСУТСТВУЕТ ПОЛЕ **SINGLE**; ЕСЛИ РАЗВЕДЕН, ТО ОТСУТСТВУЕТ **MARRIED**; ЕСЛИ В БРАКЕ НЕ СОСТОЯЛ, ТО ЗАПИСЬ СОДЕРЖИТ ЛИШЬ ПОЛЕ **NAME,DATEBIRTH** И ПУСТОЕ **NO.**

ПЕРЕД ЗАСЫЛКОЙ ИНФОРМАЦИИ В ЗАПИСЬ ПРОГРАММИСТ ДОЛЖЕН ПРИСВОИТЬ ПЕРЕКЛЮЧАТЕЛЮ СООТВЕТСТВУЮЩЕЕ ЗНАЧЕНИЕ. В ПРОТИВНОМ СЛУЧАЕ ИНФОРМАЦИЯ В ПОЛЕ (НАПРИМЕР, **MARRIED**) ЗАСЛАНА НЕ БУДЕТ, НО СИСТЕМА ДИАГНОСТИКИ НИКАКОЙ НЕ ВЫДАСТ!

ПРИМЕР ЗАСЫЛКИ ИНФОРМАЦИИ О СОТРУДНИКЕ РОГОВЕ, РОДИВШЕМСЯ 1/12/32,ЖЕНАТОМ НА РОГОВОЙ, РОДИВШЕЙСЯ 15/3/30. (ЭВМ БЭСМ-6)

```
PROGRAM L8T3(INPUT,OUTPUT);
TYPE KIND=(MARRIED,SINGLE);
     DATE=RECORD
            DAY:1..31;
          MONTH:1..12;
           YEAR:INTEGER;
         END;

     PERSON=RECORD
              NAME:ALFA;
          DATEBIRTH:DATE;
         CASE YESNO:KIND OF
           MARRIED:(NAME1:ALFA;DATE1:DATE);
           SINGLE:(NAMSING:ALFA);
          END;

VAR   P,S:PERSON; NAMEA,NAMEA1:ALFA;
BEGIN
    WITH P DO
      BEGIN
        YESNO:=MARRIED;
        NAME:='ROGOV ';
       WITH DATEBIRTH DO
         BEGIN
           DAY:=1;
           MONTH:=12;
            YEAR:=1932;
          END;

    CASE YESNO OF
      MARRIED: BEGIN
                 NAME1:='ROGOVA';
               WITH DATE1 DO
                 BEGIN
                     DAY:=15;
                     MONTH:=3;
                      YEAR:=1930;
                 END
               END;
      SINGLE:NAMESING:='SINGLE';
   END;
END;
```

```
    WITH P DO
         WRITE(' ',NAME);
   WITH P.DATEBIRTH DO
      WRITELN('  ',DAY,'/',MONTH,'/',YEAR);
   WITH P DO
      WRITE(' ',NAME1);
   WITH P.DATE1 DO
      WRITELN('  ',DAY,'/',MONTH,'/',YEAR)
   END.
```

11.3. МНОЖЕСТВО (SET)

Общий вид:

```
TYPE A=SET OF TYPEC
```

ЗДЕСЬ A - ИДЕНТИФИКАТОР ТИПА (ПРОИЗВОЛЬНЫЙ).
TYPEC - ТИП КОМПОНЕНТ МНОЖЕСТВА, НАЗЫВАЕМЫЙ БАЗОВЫМ ТИПОМ.

ПРИМЕР.

```
TYPE INT=(1,3,5);
     A=SET OF INT;
```

ПЕРЕМЕННАЯ ТИПА A В ЭТОМ СЛУЧАЕ МОЖЕТ ИМЕТЬ СЛЕДУЮЩИЕ ЗНАЧЕНИЯ:

```
[1],[3],[5],[1,3],[1,5],[3,5],[1,3,5],[  ],
```

ГДЕ [ ] ОЗНАЧАЕТ ПУСТОЕ МНОЖЕСТВО.

ПРИМЕР.

```
TYPE BIN=(BIT1,BIT2,BIT3);
     BTS=SET OF BIN;
```

ПЕРЕМЕННАЯ ТИПА BTS МОЖЕТ ПРИНИМАТЬ ЗНАЧЕНИЯ:

```
[BIT1],[BIT2],[BIT3],[BIT1,BIT2],[BIT1,BIT3],
[BIT2,BIT3],[BIT1,BIT2,BIT3],[ ].
```

ТАКИМ ОБРАЗОМ, ИСПОЛЬЗУЯ ПЕРЕМЕННЫЕ ТИПА **SET** , МОЖНО РАБОТАТЬ С БИТОВОЙ ИНФОРМАЦИЕЙ.

ПАСКАЛЬ ДОПУСКАЕТ МНОЖЕСТВА, СОСТОЯЩИЕ ИЗ ОГРАНИЧЕННОГО ЧИСЛА ЭЛЕМЕНТОВ N<=NMAX.

```
НА БЭСМ-6     NMAX=48
НА СДС-6500   NMAX=59
НА ЕС ЭВМ     NMAX=63
```

В КАЧЕСТВЕ БАЗОВОГО ТИПА МОЖНО ИСПОЛЬЗОВАТЬ ЛЮБОЙ СКАЛЯРНЫЙ ТИП, КРОМЕ **REAL**.

ЕСЛИ В КАЧЕСТВЕ БАЗОВОГО ТИПА ВЫБРАН ТИП **CHAR** , ТО ДОПУСКАЕТСЯ ИСПОЛЬЗОВАТЬ НЕ БОЛЕЕ NMAX ПЕРВЫХ СИМВОЛОВ, ИМЕЮЩИХСЯ В РАСПОРЯЖЕНИИ ТРАНСЛЯТОРА.

ДЛЯ ПОДМНОЖЕСТВА ТИПА INTEGER МОЖНО ИСПОЛЬЗОВАТЬ В КАЧЕСТВЕ КОМПОНЕНТ ЦЕЛЫЕ ЧИСЛА ОТ НУЛЯ ДО N<=NMAX-1.

11.3.1. ДАННЫЕ ТИПА SET

ОБЩИЙ ВИД:

```
[EXPR1,EXPR2,...EXPRN]
```

ЗДЕСЬ EXPRI - ВЫРАЖЕНИЯ БАЗОВОГО ТИПА.
НЕПУСТОЙ НАБОР МОЖЕТ БЫТЬ ВЫРАЖЕНИЕМ ВИДА:

```
[EXPR1]
[EXPR1..EXPRK]
[EXPR1,EXPRK..EXPRN]
```

НАБОР [J..I] , ГДЕ J<I ИНТЕПРЕТИРУЕТСЯ КАК ПУСТОЙ.

ПРИМЕР.

```
TYPE COLOR=(RED,YELLOW,GREEN,BLUE);
   VAR  MIX : SET OF COLOR;
                .
                .
                .
          MIX:=[RED,BLUE];
```

НА ЕС ЭВМ СИМВОЛАМ [И] СООТВЕТСТВУЮТ ПАРЫ СИМВОЛОВ "(." И ".)"

11.3.2. ОПЕРАЦИИ С ПЕРЕМЕННЫМИ ТИПА SET.

К ПЕРЕМЕННЫМ ТИПА SET ПРИМЕНИМЫ СЛЕДУЮЩИЕ ОПЕРАЦИИ:

```
=,<>,<=,>=,IN,+,-,*.
```

ОПЕРАЦИИ = И <> ИСПОЛЬЗУЮТСЯ ДЛЯ ПРОВЕРКИ ЭКВИВАЛЕНТНОСТИ: ДВА ЗНАЧЕНИЯ ПЕРЕМЕННОЙ ТИПА SET СЧИТАЮТСЯ РАВНЫМИ, ЕСЛИ ОНИ СОСТОЯТ ИЗ ОДНИХ И ТЕХ ЖЕ ЭЛЕМЕНТОВ.

ПРИМЕР.

```
    [1,3]=[3,1]          ДАЕТ TRUE,
  [1..3]=[1,2,3]         ДАЕТ TRUE,
    [1]<[2]              ДАЕТ TRUE.
  [1,2,3]=[1,4,3]        ДАЕТ FALSE,
[RED,BLUE]=[RED,YELLOW]  ДАЕТ FALSE,
   [RED]>[BLUE]          ДАЕТ FALSE,
```

ТАК КАК ПРИ ОПИСАНИИ ТИПА COLOR ЭЛЕМЕНТ RED ПРЕДШЕСТВУЕТ ЭЛЕМЕНТУ BLUE.
ОПЕРАЦИИ = И <> ИСПОЛЬЗУЮТСЯ ДЛЯ ПРОВЕРКИ ПРИНАДЛЕЖНОСТИ ОДНОГО МНОЖЕСТВА ДРУГОМУ: ТАК, ЕСЛИ МНОЖЕСТВО A СОДЕРЖИТСЯ В МНОЖЕСТВЕ B, ТО A<=B ДАЕТ TRUE.

ПРИМЕР.

[1,2]<=[1,2,3] ДАЕТ TRUE.

ПУСТОЕ МНОЖЕСТВО [ ] СОДЕРЖИТСЯ ВО ВСЕХ МНОЖЕСТВАХ, Т.Е. ВСЕГДА [ ]<=B ДАЕТ TRUE.

СПЕРАЦИЯ IN ИСПОЛЬЗУЕТСЯ ДЛЯ УСТАНОВЛЕНИЯ НАЛИЧИЯ ОПРЕДЕЛЕННОГО ЭЛЕМЕНТА В ВЕЛИЧИНЕ ТИПА SET. ТАК, ЕСЛИ X ЕСТЬ ЭЛЕМЕНТ МНОЖЕСТВА B, ТО (X IN B) ДАЕТ TRUE.

ОБЩИЙ ВИД:

X IN A

ЗДЕСЬ X - ВЕЛИЧИНА БАЗОВОГО ТИПА;
A - ВЕЛИЧИНА ТИПА SET.

ПРИМЕР.

RED IN [RED,YELLOW] ДАЕТ TRUE.
RED IN [BLUE] ДАЕТ FALSE.

НО, ЕСЛИ A:SET OF 1..50; X: INTEGER; И X:=65 ,Т.Е. БОЛЬШЕ 50, ТО РЕЗУЛЬТАТ (X IN A) МОЖЕТ БЫТЬ НЕОПРЕДЕЛЕННЫМ,Т.Е. НЕ ВСЕГДА FALSE.

К ПЕРЕМЕННЫМ ТИПА SET , ОТНОСЯЩИМСЯ К ОДНОМУ И ТОМУ ЖЕ КОНКРЕТНОМУ ТИПУ, ПРИМЕНИМЬ ОПЕРАЦИИ:

\+ - ОБ'ЕДИНЕНИЕ;
* - ПЕРЕСЕЧЕНИЕ;
\- - ДОПОЛНЕНИЕ.

ПУСТЬ A И B - ОПЕРАНДЫ, ИМЕЮЩИЕ ОДИН И ТОТ ЖЕ КОНКРЕТНЫЙ ТИП.
ТОГДА
A+B ПРЕДСТАВЛЯЕТ СОБОЙ ОБ'ЕДИНЕНИЕ МНОЖЕСТВ ЭЛЕМЕНТОВ,ВХОДЯЩИХ В A И B; ОДИНАКОВЫЕ ЭЛЕМЕНТЫ НЕ ПОВТОРЯЮТСЯ.
A*B - ПЕРЕСЕЧЕНИЕ МНОЖЕСТВ ЭЛЕМЕНТОВ A И B.
A-B - МНОЖЕСТВО ЭЛЕМЕНТОВ, КОТОРЫЕ ЕСТЬ В A , НО ОТСУТСТВУЮТ В B.

ПРИМЕР.

[1,3]+[1,4] ДАЕТ [1,3,4]

[1,3]*[1,4] ДАЕТ [1]

[1,3]-[1,4] ДАЕТ [3]

ОПЕРАЦИЯ A:=A+[X] ДОБАВЛЯЕТ X К МНОЖЕСТВУ A. ЕСЛИ X УЖЕ ИМЕЛСЯ В A , ТО МНОЖЕСТВО A НЕ МЕНЯЕТСЯ. A:=A-[X] ИСКЛЮЧАЕТ X ИЗ A. ЕСЛИ X ОТСУТСТВОВАЛ В A , ТО МНОЖЕСТВО A НЕ МЕНЯЕТСЯ.

11.4. ФАЙЛ(FILE)

ФАЙЛ ПРЕДСТАВЛЯЕТ СОБОЙ ПОСЛЕДСВАТЕЛЬНОСТЬ КОМПОНЕНТ ОДНОГО ТИПА. ЧИСЛО КОМПОНЕНТ НЕ ФИКСИРОВАНО. В КАЖДОЙ МОМЕНТ ДОСТУПНА ТОЛЬКО ОДНА КОМПОНЕНТА ДЛЯ ЧТЕНИЯ И ЗАПИСИ, НА КОТОРУЮ УСТАНОВЛЕН УКАЗАТЕЛЬ ФАЙЛА.

ФАЙЛ, НЕ СОДЕРЖАЩИЙ КОМПОНЕНТ, НАЗЫВАЕТСЯ ПУСТЫМ ФАЙЛОМ.

ОБЩИЙ ВИД:

```
TYPE R = FILE OF TYPEC;
```

ЗДЕСЬ R - ИДЕНТИФИКАТОР ТИПА;

TYPEC - ТИП КОМПОНЕНТ - МОЖЕТ БЫТЬ ЛЮБЫМ,КРОМЕ ТИПА FILE,

ЕСЛИ ПЕРЕМЕННАЯ F ИМЕЕТ ТИП FILE, ТО ТРАНСЛЯТОР АВТОМАТИЧЕСКИ ВВОДИТ ПЕРЕМЕННУЮ F↑ - ПЕРЕМЕННУЮ БУФЕРА, -"ОКНО", ЧЕРЕЗ КОТОРОЕ МОЖНО ПРОЧИТАТЬ ИЛИ ЗАПИСАТЬ КОМПОНЕНТУ.

ПЕРЕМЕННАЯ F↑ ИМЕЕТ ТОТ ЖЕ ТИП, ЧТС И КОМПОНЕНТЫ ФАЙЛА F.

ЕСЛИ ВЫПОЛНИЛАСЬ ОПЕРАЦИЯ ЗАПИСИ В N - Ю КОМПОНЕНТУ ФАЙЛА, ТО УКАЗАТЕЛЬ АВТОМАТИЧЕСКИ ПРОДВИГАЕТСЯ К (N+1) -Й КОМПОНЕНТЕ.

ЕСЛИ N -Я КОМПОНЕНТА БЫЛА ПОСЛЕДНЕЙ В ФАЙЛЕ, Т.Е. УКАЗАТЕЛЬ ПРОДВИНУЛСЯ ЗА ГРАНИЦЫ ФАЙЛА, ТО СТАНДАРТНАЯ ФУНКЦИЯ
EOF(F) (ТИПА BOOLEAN) ДАЕТ ЗНАЧЕНИЕ TRUE,
ВО ВСЕХ ОСТАЛЬНЫХ СЛУЧАЯХ ОНА - FALSE.

К ПЕРЕМЕННЫМ ТИПА FILE ПРИМЕНИМЫ СЛЕДУЮЩИЕ ОПЕРАЦИИ:

RESET(F), REWRITE(F), GET(F), PUT(F).

RESET(F) - УСТАНАВЛИВАЕТ УКАЗАТЕЛЬ ФАЙЛА НА ПЕРВУЮ КОМПОНЕНТУ; ФУНКЦИИ EOF(F) ПРИСВАИВАЕТСЯ ЗНАЧЕНИЕ FALSE, ЕСЛИ ФАЙЛ НЕ ПУСТ.
ИНАЧЕ - ЗНАЧЕНИЕ F↑ НЕОПРЕДЕЛЕННО, А ЗНАЧЕНИЕ EOF(F) ОСТАЕТСЯ TRUE.
ОПЕРАЦИЮ RESET(F) НЕОБХОДИМО ВЫПОЛНЯТЬ ПЕРЕД НАЧАЛОМ ЧТЕНИЯ ИНФОРМАЦИИ ИЗ ФАЙЛА F.

REWRITE(F) - ОЧИЩАЕТ ФАЙЛ F И ПРИСВАИВАЕТ ФУНКЦИИ EOF(F) ЗНАЧЕНИЕ TRUE. ЭТА ОПЕРАЦИЯ ДОЛЖНА ПРЕДШЕСТВОВАТЬ ЗАПИСИ В НАЧАЛО ФАЙЛА F.

GET(F) - ПРОДВИГАЕТ УКАЗАТЕЛЬ К СЛЕДУЮЩЕЙ КОМПОНЕНТЕ ФАЙЛА F И ПРИСВАИВАЕТ ПЕРЕМЕННОЙ F↑ ЗНАЧЕНИЕ ЭТОЙ КОМПОНЕНТЫ.
ЕСЛИ СООТВЕТСТВУЮЩЕЙ КОМПОНЕНТЫ В ФАЙЛЕ НЕТ(КОНЕЦ ФАЙЛА), ТС EOF(F) ПРИСВАИВАЕТСЯ TRUE ,А ЗНАЧЕНИЕ F↑ СТАНОВИТСЯ НЕОПРЕДЕЛЕННЫМ (ПОРТИТСЯ).

PUT(F) - ЗАНОСИТ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ F↑ В СООТВЕТСТВУЮЩУЮ КОМПОНЕНТУ ФАЙЛА F ,ЕСЛИ EOF(F) ИМЕЕТ ЗНАЧЕНИЕ TRUE. ПОСЛЕ ВЫПОЛНЕНИЯ PUT(F) ЗНАЧЕНИЕ F↑ СТАНОВИТСЯ НЕОПРЕДЕЛЕННЫМ.

ПРИМЕР. ВЫЧИСЛИТЬ СУММУ ПЯТИ ВЕЩЕСТВЕННЫХ ЧИСЕЛ.

```
PROGRAM KOD(INPUT,OUTPUT);
VAR S:REAL; F:FILE OF REAL; I:INTEGER;
    BEGIN REWRITE(F);
       FOR I:=1 TO 5 DO
        BEGIN READ(S); F   :=S; PUT(F) END;
      S:=0; RESET(F);
     WHILE NOT EOF(F) DO
      BEGIN S:=S+F    ; GET(F) END;
     WRITELN(' S=', S)
    END.
```

11.4.1. ВНЕШНИЕ ФАЙЛЫ

ФАЙЛЫ ПО ОТНОШЕНИЮ К ЗАДАЧЕ МОГУТ БЫТЬ ВНЕШНИМИ И ВНУТРЕННИМИ. СОДЕРЖИМОЕ ВНЕШНИХ ФАЙЛОВ, В ОТЛИЧИЕ ОТ ВНУТРЕННИХ, СОХРАНЯЕТСЯ И ПОСЛЕ ОКОНЧАНИЯ РАБОТЫ ПРОГРАММЫ. ВНЕШНИЕ ФАЙЛЫ ДОЛЖНЫ БЫТЬ УКАЗАНЫ В КАЧЕСТВЕ ПАРАМЕТРОВ ПРОГРАММЫ, НАРЯДУ С ФАЙЛАМИ INPUT И OUTPUT.

ПРИМЕР. `PROGRAM L(INPUT,OUTPUT,X);`

ЗДЕСЬ X -ВНЕШНИЙ ФАЙЛ, ЕГО ИМЯ И СОДЕРЖИМОЕ СИСТЕМА БУДЕТ ПОМНИТЬ ПОСЛЕ КОНЦА РАБОТЫ ПРОГРАММЫ L .

ПРИМЕР. ЗАПИСАТЬ ВО ВНЕШНИЙ ФАЙЛ X ПОСЛЕДОВАТЕЛЬНЫЕ ЗНАЧЕНИЯ ТИПА COLOR=(READ,YELLOW,GREEN,BLUE).

```
PROGRAM L10T3(OUTPUT,X);
TYPE COLOR = (RED,YELLOW,GREEN,BLUE);
VAR X : FILE OF COLOR; Z :COLOR;
      I : INTEGER;
BEGIN
      REWRITE(X);
      Z:=RED;
      FOR I:=1 TO 4 DO
         BEGIN
            WRITE(X,Z);
            Z:=SUCC(Z)
         END
END.
```

11.4.2. ТЕКСТОВЫЕ ФАЙЛЫ

ФАЙЛЫ, КОМПОНЕНТАМИ КОТОРЫХ ЯВЛЯЮТСЯ СИМВОЛЫ, НАЗЫВАЮТСЯ ТЕКСТОВЫМИ.

СТАНДАРТНЫЙ ТИП ТАКОГО ФАЙЛА TEXT В КАЖДОМ ТРАНСЛЯТОРЕ С ПАСКАЛЯ ПРЕДОПРЕДЕЛЕН КАК

```
TEXT=FILE OF CHAR
```

ТЕКСТОВЫЙ ФАЙЛ СОСТОИТ ИЗ ПОСЛЕДОВАТЕЛЬНОСТИ СТРОК, КАЖДАЯ ИЗ КОТОРЫХ СОДЕРЖИТ ВЕЛИЧИНЫ ТИПА CHAR И ЗАКАНЧИВАЕТСЯ СПЕЦИАЛЬНЫМ СИМВОЛОМ "КОНЦА СТРОКИ" (EOL).

СТАНДАРТНЫЕ ФАЙЛЫ INPUT И OUTPUT ЯВЛЯЮТСЯ ТЕКСТОВЫМИ.

С СИМВОЛОМ "КОНЕЦ СТРОКИ" ОПЕРИРУЮТ СЛЕДУЮЩИЕ ФУНКЦИИ:

WRITELN(X) - ЗАПИСЫВАЕТ СИМВОЛ "КОНЕЦ СТРОКИ (EOL) В КОМПОНЕНТУ ФАЙЛА X, НА КОТОРУЮ УСТАНОВЛЕН УКАЗАТЕЛЬ ФАЙЛА

READLN(X) - УКАЗАТЕЛЬ ФАЙЛА ПРОПУСКАЕТ ОСТАВШУЮСЯ ЧАСТЬ ТЕКУЩЕЙ СТРОКИ И УСТАНАВЛИВАЕТСЯ НА ПЕРВЫЙ СИМВОЛ НОВОЙ СТРОКИ; В "ОКНО" ФАЙЛА X↑ СЧИТЫВАЕТСЯ ЭТОТ СИМВОЛ.

EOLN(X) - ПРИНИМАЕТ ЗНАЧЕНИЕ TRUE, ЕСЛИ УКАЗАТЕЛЬ УСТАНОВЛЕН НА СИМВОЛ "КОНЕЦ СТРОКИ", И ЗАСЫЛАЕТ ПРОБЕЛ В X↑.

ЕСЛИ F - ТЕКСТОВЫЙ ФАЙЛ, А CH - СИМВОЛ, ТО МОЖНО ИСПОЛЬЗОВАТЬ СЛЕДУЮЩИЕ ФОРМЫ ПРОЦЕДУР ЗАПИСИ И ЧТЕНИЯ.

```
WRITE(F,CH)            - ВМЕСТО F.:=CH; PUT(F);
READ(F,CH)             - ВМЕСТО CH:=F. GET(F);
WRITELN(F,V1,V2,...)   - ВМЕСТО WRITE(F,V1);WRITE(F,V2);....
                                WRITELN(F);
READLN(F,V1,V2,...)    - ВМЕСТО READ(F,V1);READ(F,V2);....
                                READLN(F);
```

ПРИМЕР 1. ЗАПИСЬ ТЕКСТА В ФАЙЛ Y.

ПУСТЬ ПРОЦЕДУРА P(C) ВЫДАЕТ ПОСЛЕДОВАТЕЛЬНЫЕ СИМВОЛЫ ТЕКСТА И ПРИСВАИВАЕТ ОЧЕРЕДНОЙ СИМВОЛ ПАРАМЕТРУ C . ПРИ ЭТОМ КОНЕЦ СТРОКИ ОТМЕЧАЕТСЯ ПРИСВАИВАНИЕМ ПЕРЕМЕННОЙ B ЗНАЧЕНИЯ TRUE ; КОНЕЦ ТЕКСТА - ПРИСВАИВАНИЕМ ПЕРЕМЕННОЙ Q ЗНАЧЕНИЯ TRUE.

```
          .
          .
          .
    REWRITE(Y);
    REPEAT
        REPEAT P(C); WRITE(Y,C)
        UNTIL B;
        WRITELN (Y)
    UNTIL Q;
```

ПРИМЕР 2. ЧТЕНИЕ ТЕКСТА ИЗ ФАЙЛА X.

ПУСТЬ НАДО ПРОЧИТАТЬ СИМВОЛЫ ИЗ ФАЙЛА X ,ОБРАБОТАТЬ ИХ ПРОЦЕДУРОЙ Q(C) И ПОСЛЕ КАЖДОЙ СТРОКИ ТЕКСТА ВЫПОЛНИТЬ НЕКОТОРЫЕ ДЕЙСТВИЯ R .

```
        .
        .
        .
    RESET(X);
    WHILE NOT EOF(X) DO
        BEGIN
          WHILE NOT EOLN(X) DO
             BEGIN READ(X,C);
             Q(C)
             END;
          R;
          READLN(X)
        END;
```

ПРИМЕР 3. КОПИРОВАНИЕ ТЕКСТА ИЗ ФАЙЛА X В ФАЙЛ Y.

```
        .
        .
        .
    RESET(X);
    REWRITE(Y);
    WHILE NOT EOF(X) DO
       BEGIN
        WHILE NOT EOLN (X) DO
          BEGIN READ (X,C); WRITE (Y,C) END;
        READLN (X); WRITELN (Y)
       END;
```

11.4.3 СТАНДАРТНЫЕ ТЕКСТОВЫЕ ФАЙЛЫ

INPUT И OUTPUT

ПО УМОЛЧАНИЮ

WRITE(CH)	СООТВЕТСТВУЕТ	WRITE(OUTPUT,CH);
READ(CH)	СООТВЕТСТВУЕТ	READ(INPUT,CH);
WRITELN	СООТВЕТСТВУЕТ	WRITELN(OUTPUT);
EOF	СООТВЕТСТВУЕТ	EOF(INPUT);
EOLN	СООТВЕТСТВУЕТ	EOLN(INPUT);

К ФАЙЛАМ **INPUT** И **OUTPUT** НЕЛЬЗЯ ПРИМЕНЯТЬ **RESET** И **REWRITE**

ПЕРВЫЙ СИМВОЛ КАЖДОЙ СТРОКИ ФАЙЛА OUTPUT УПРАВЛЯЕТ УСТРОЙСТВОМ ПЕЧАТИ.

ЕСЛИ ЭТОТ СИМВОЛ "ПРОБЕЛ", ТО - ПЕРЕХОД К СЛЕДУЮЩЕЙ СТРОКЕ;
"НОЛЬ" - ПРОПУСК СТРОКИ;
"1" - ПЕРЕХОД К НАЧАЛУ СЛЕДУЮЩЕЙ СТРАНИЦЫ;
"+" - ПЕЧАТЬ БЕЗ ПЕРЕХОДА К НОВОЙ СТРОКЕ (ПЕЧАТЬ С НАЛОЖЕНИЕМ СТРОК).

ПРИМЕР 4. ПЕЧАТЬ СОДЕРЖИМОГО ВНЕШНЕГО ФАЙЛА X.

```
PROGRAM L1ØT2(OUTPUT,X);
VAR  CH:CHAR;
       X:FILE OF CHAR;
BEGIN
    RESET(X); WHILE NOT EOF(X) DO
      BEGIN WRITE (' ');
       WHILE NOT EOLN(X) DO
          BEGIN READ(X,CH); WRITE(CH) END;
       WRITELN; READLN(X)
          END
END.
```

12 ТИП ССЫЛКА (POINTER)

ЭТОТ ТИП СВЯЗАН С ДИНАМИЧЕСКИМИ ПЕРЕМЕННЫМИ, КОТОРЫЕ НЕ ОБОЗНАЧАЮТСЯ ИДЕНТИФИКАТОРАМИ; ОНИ СОЗДАЮТСЯ И СТИРАЮТСЯ ВО ВРЕМЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ СПЕЦИАЛЬНЫМИ ПРОЦЕДУРАМИ.

ДОСТУП К ДИНАМИЧЕСКИМ ПЕРЕМЕННЫМ ОСУЩЕСТВЛЯЕТСЯ ПОСРЕДСТВОМ ССЫЛОК (УКАЗАТЕЛЕЙ):

P⟶X (P "УКАЗЫВАЕТ" НА X)

ЗДЕСЬ P - ПЕРЕМЕННАЯ - УКАЗАТЕЛЬ (ССЫЛКА).
X - ДИНАМИЧЕСКАЯ ПЕРЕМЕННАЯ, НАПРИМЕР, ТИПА TYPEC. ИМЯ X В ПРОГРАММЕ НЕ ФИГУРИРУЕТ.

ОБЩИЙ ВИД ОПИСАНИЯ ТИПА POINTER:

```
TYPE PT = ↑TYPEC;
```

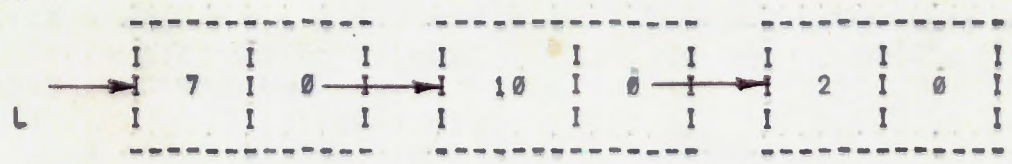
ЗДЕСЬ PT - ИДЕНТИФИКАТОР КОНКРЕТНОГО ТИПА;
TYPEC - ТИП ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ;
↑TYPEC - УКАЗАТЕЛЬ НА ПЕРЕМЕННЫЕ ТИПА TYPEC.

ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ СВЯЗАНЫ ДРУГ С ДРУГОМ В ЦЕПОЧКУ.

КАЖДАЯ ДИНАМИЧЕСКАЯ ПЕРЕМЕННАЯ СОДЕРЖИТ НЕ МЕНЕЕ ДВУХ ПОЛЕЙ: ПОЛЕ ЗНАЧЕНИЯ И ПОЛЕ УКАЗАТЕЛЯ НА СЛЕДУЮЩУЮ ПЕРЕМЕННУЮ.

АППАРАТ ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ ПОЗВОЛЯЕТ СОЗДАВАТЬ СТРУКТУРЫ, ЭКВИВАЛЕНТНЫЕ КОНЕЧНЫМ ГРАФАМ.
ВЕРШИНАМ СООТВЕТСТВУЮТ САМИ ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ (X), А РЕБРАМ - ССЫЛКИ (P⟶).

ПРИМЕР.



ЗДЕСЬ L - ПЕРЕМЕННАЯ - УКАЗАТЕЛЬ; ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ В ПОЛЕ ЗНАЧЕНИЯ СОДЕРЖАТ ВЕЛИЧИНЫ ТИПА **INTEGER**; А В ПОЛЕ УКАЗАТЕЛЯ - (0) ССЫЛКУ НА СЛЕДУЮЩУЮ ПЕРЕМЕННУЮ ЦЕПОЧКИ. ЭТОТ ПРИМЕР ОПИСЫВАЕТСЯ ТАК:

```
TYPE LISTP=↑ COMPON;
     COMPON= RECORD
             I: INTEGER;
             NEXT: LISTP
             END;
```

ЗДЕСЬ **LISTP** - ПРОИЗВОЛЬНЫЙ ИДЕНТИФИКАТОР.

В ПРИВЕДЕННОМ ПРИМЕРЕ ИСПОЛЬЗОВАН ИДЕНТИФИКАТОР **COMPON** ДО ЕГО ОПИСАНИЯ. ПАСКАЛЬ ТОЛЬКО ДЛЯ ТИПА **POINTER** ДОПУСКАЕТ ТАКОЕ ИСКЛЮЧЕНИЕ. ОБЩЕЕ ПРАВИЛО: КАЖДЫЙ ИДЕНТИФИКАТОР ДОЛЖЕН БЫТЬ ОПИСАН ДО ЕГО ИСПОЛЬЗОВАНИЯ В ПРОГРАММЕ

12.1. ПРОЦЕДУРЫ NEW И DISPOSE

ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ СОЗДАЮТСЯ СТАНДАРТНОЙ ПРОЦЕДУРОЙ **NEW**.

ОБЩИЙ ВИД:

```
NEW(B);
```

ЗДЕСЬ B - -ПЕРЕМЕННАЯ ТИПА **POINTER**, СВЯЗАННАЯ ПРИ ОПИСАНИИ ТИПА С ОПРЕДЕЛЕННЫМ ТИПОМ ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ. НАПРИМЕР, ЕСЛИ **B:LISTP**, ТО **NEW(B)** ОТВОДИТ МЕСТО В ПАМЯТИ И ЗАПОМИНАЕТ АДРЕС ЗАПИСИ ТИПА **COMPON**.

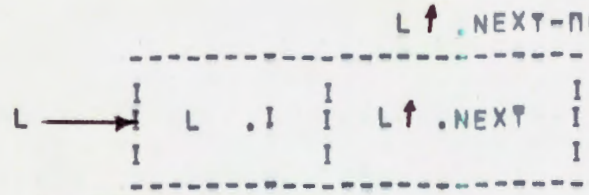
ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ НАЗЫВАЮТ ТАКЖЕ "УКАЗУЕМЫМИ", "УКАЗАННЫМИ" И "КОСВЕННЫМИ" ПЕРЕМЕННЫМИ.

ЕСЛИ P⟶R (ПЕРЕМЕННАЯ P УКАЗЫВАЕТ НА R), ТО R ЕСТЬ P↑

ЗДЕСЬ R - ДИНАМИЧЕСКАЯ (УКАЗУЕМАЯ) ПЕРЕМЕННАЯ, А
P - УКАЗАТЕЛЬ ЭТОЙ ПЕРЕМЕННОЙ.

ПРИМЕР. ПУСТЬ L:LISTP. ПОСЛЕ ОБРАЩЕНИЯ К NEW (L).

L↑ - ОБОЗНАЧАЕТ ВНОВЬ СОЗДАННУЮ ПЕРЕМЕННУЮ ТИПА COMMON;
L↑.I - ЦЕЛАЯ КОМПОНЕНТА ЭТОЙ ПЕРЕМЕННОЙ;
L↑.NEXT - УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ПЕРЕМЕННУЮ В ЦЕПОЧКЕ.

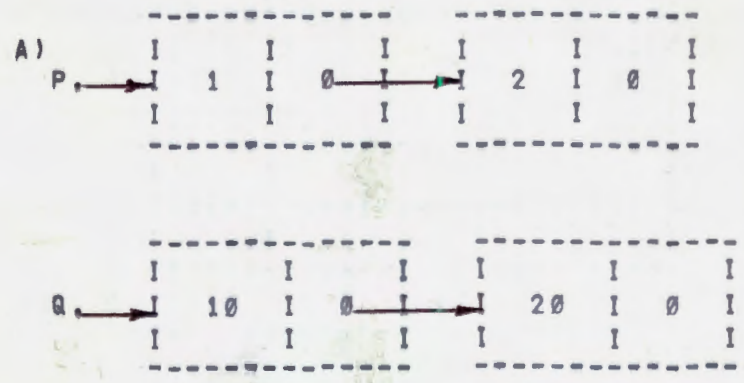


L .I- ПОЛЕ ЗНАЧЕНИЯ.

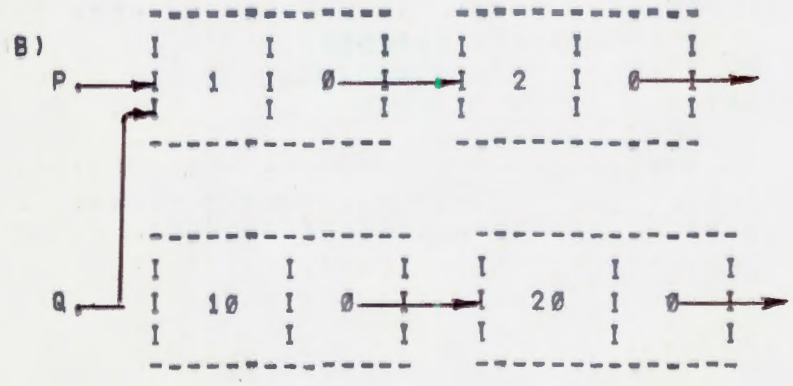
12.2. ОПЕРАЦИИ НАД ПЕРЕМЕННЫМИ ТИПА POINTER

УКАЗАТЕЛИ МОЖНО КОПИРОВАТЬ ОПЕРАТОРОМ ПРИСВАИВАНИЯ.

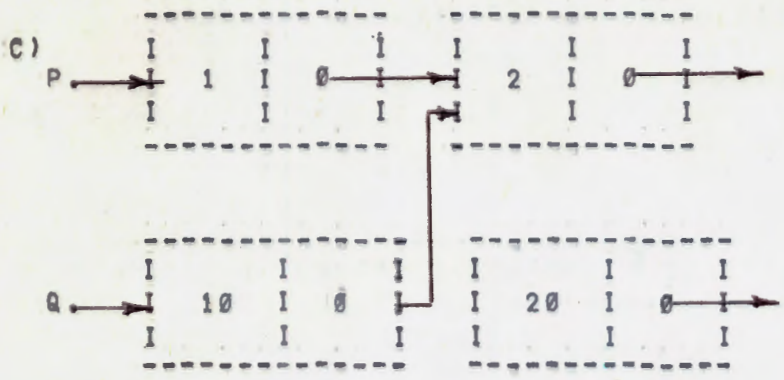
ПРИМЕР.
ПУСТЬ P И Q УКАЗЫВАЮТ НА РАЗНЫЕ ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ ТИПА COMPON:



ПОСЛЕ ВЫПОЛНЕНИЯ Q:=P ПОЛУЧИМ:



ЕСЛИ ПОСЛЕ А) ВЫПОЛНИТЬ Q↑:=P↑ ТО ПОЛУЧИМ



УКАЗАТЕЛИ МОЖНО СРАВНИВАТЬ, ИСПОЛЬЗУЯ ОПЕРАЦИИ = И < >.
В ПРИВЕДЕННОМ ВЫШЕ ПРИМЕРЕ

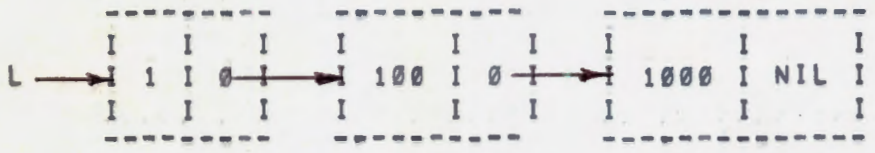
В СЛУЧАЕ B) P=Q ДАЕТ TRUE;

C) P=Q ДАЕТ FALSE.

ЕСЛИ УКАЗАТЕЛЬ НЕ УКАЗЫВАЕТ НИ НА ОДНУ ПЕРЕМЕННУЮ, ГОВОРЯТ, ЧТО ЕГО ЗНАЧЕНИЕ ЕСТЬ NIL.

P:NIL - ОЧИСТКА УКАЗАТЕЛЯ.

ПОСЛЕДНЯЯ КОМПОНЕНТА КАЖДОЙ ЦЕПОЧКИ ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ ИМЕЕТ ЗНАЧЕНИЕ NIL.



ЕСЛИ L=NIL, ТО ЦЕПОЧКА ПУСТА.

12.3. ПРОЦЕДУРА DISPOSE

ДИНАМИЧЕСКАЯ ПЕРЕМЕННАЯ, СОЗДАННАЯ ПРОЦЕДУРОЙ NEW, МОЖЕТ БЫТЬ СТЕРТА ТОЛЬКО ПРОЦЕДУРОЙ DISPOSE.

ОБЩИЙ ВИД: DISPOSE(P);

ЗДЕСЬ P УКАЗЫВАЕТ НА ТУ ДИНАМИЧЕСКУЮ ПЕРЕМЕННУЮ, КОТОРУЮ СЛЕДУЕТ СТЕРЕТЬ. ПОСЛЕ СТИРАНИЯ ПЕРЕМЕННОЙ НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ ЗНАЧЕНИЕ P↑: ТАКАЯ ОШИБКА МОЖЕТ ПРИВЕСТИ К ПОРЧЕ ПАМЯТИ И ДРУГИМ СЕРЬЕЗНЫМ ПОСЛЕДСТВИЯМ.

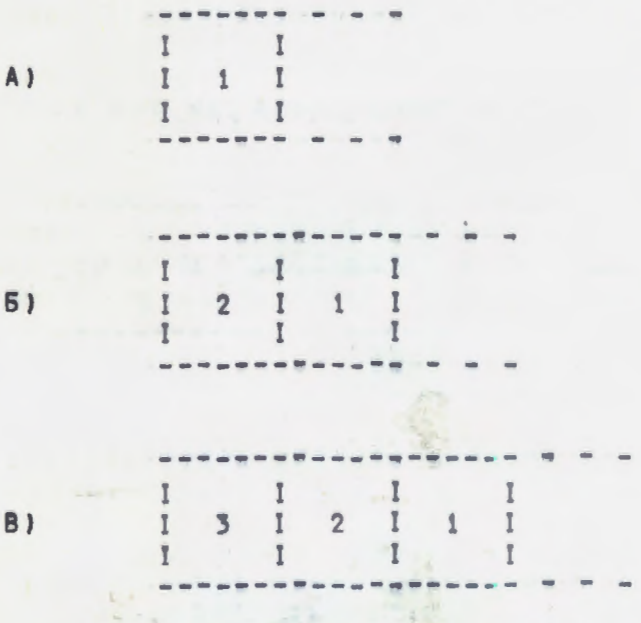
ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ, НЕ СТЕРТЫЕ ОПЕРАЦИЕЙ **DISPOSE**, ПРОДОЛЖАЮТ ЗАНИМАТЬ МЕСТО В ПАМЯТИ И ПОСЛЕ ОКОНЧАНИЯ РАБОТЫ СООТВЕТСТВУЮЩЕГО БЛОКА ПРОГРАММЫ. ПОЭТОМУ НЕОБХОДИМО ВСЕ ДИНАМИЧЕСКИЕ ПЕРЕМЕННЫЕ СТЕРЕТЬ ПЕРЕД КОНЦОМ РАБОТЫ БЛОКА.

12.4. СТЕК ("МАГАЗИН")

СТЕК - ТАКАЯ СТРУКТУРА ДАННЫХ, КОТОРАЯ СОСТОИТ ИЗ ПЕРЕМЕННОГО ЧИСЛА КОМПОНЕНТ ОДИНАКОВОГО ТИПА. КОМПОНЕНТЫ ИЗВЛЕКАЮТСЯ ИЗ СТЕКА ТАКИМ ОБРАЗОМ, ЧТО ПЕРВОЙ ВЫБИРАЕТСЯ ТА КОМПОНЕНТА, КОТОРАЯ ПОСЛЕДНЕЙ БЫЛА В СТЕК ЗАСЛАНА. ИЗВЛЕЧЕННАЯ КОМПОНЕНТА В СТЕКЕ НЕ СОХРАНЯЕТСЯ.

ПРИМЕР. РАССМОТРИМ ПОСЛЕДОВАТЕЛЬНЫЕ ЭТАПЫ ЗАСЫЛКИ В СТЕК ЧИСЕЛ 1;2;3.

СТЕК

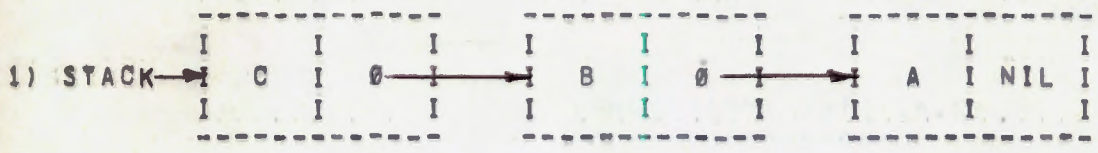


НА ЭТАПЕ Б) ОБРАЩЕНИЕ К ПРОЦЕДУРЕ ИЗВЛЕЧЕНИЯ ИЗ СТЕКА ДАЕТ ЧИСЛО 2, НА ЭТАПЕ В) - ЧИСЛО 3.

ОПИШЕМ СТЕК, В КОТОРЫЙ МОЖНО ПОМЕЩАТЬ ПОСЛЕДОВАТЕЛЬНО СИМВОЛЫ:

```
TYPE STACKP=↑STACKCOMP;
STACKCOMP=RECORD
            C:CHAR;
            NEXT:STACKP
          END;
VAR STACK:STACKP;
```

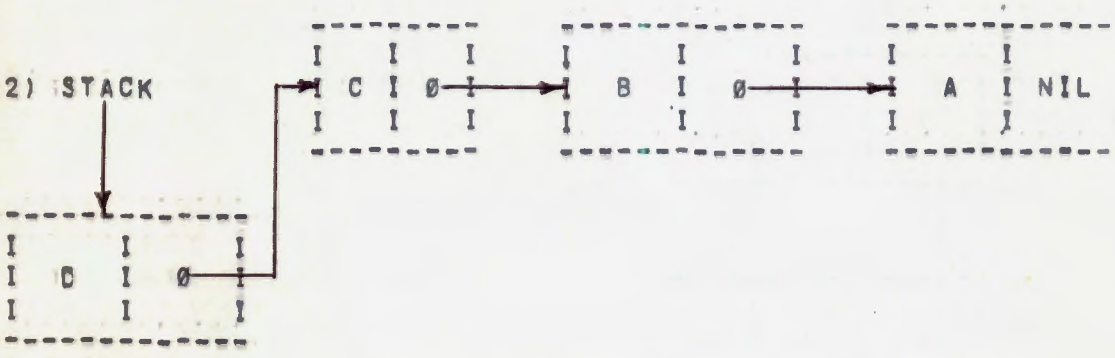
ЕСЛИ ПОМЕСТИТЬ В ЭТОТ СТЕК ПОСЛЕДОВАТЕЛЬНО СИМВОЛЫ A,B,C, ТО ПОЛУЧИТСЯ СЛЕДУЮЩАЯ КАРТИНКА



ПОМЕСТИТЬ ИНФОРМАЦИЮ В СТЕК МОЖНО ПРОЦЕДУРОЙ **PUSH**:

```
PROCEDURE PUSH(X:CHAR);
VAR NEWCOMP:STACKP;
                          (ТИП STACKP ДОЛЖЕН БЫТЬ
                          ОПИСАН ВЫШЕ В PROGRAM )
BEGIN
   NEW(NEWCOMP);
      WITH NEWCOMP  DO BEGIN
                         C:=X;
                         NEXT:=STACK
                       END;
    STACK:=NEWCOMP
END;
```

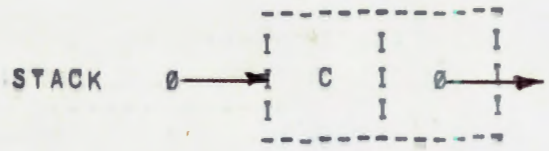
ЕСЛИ СО СТЕКОМ ВИДА 1) ОБРАТИТЬСЯ К ПРОЦЕДУРЕ **PUSH** ДЛЯ ЗАСЫЛКИ СИМВОЛА D, ТО ПОЛУЧИМ СТЕК



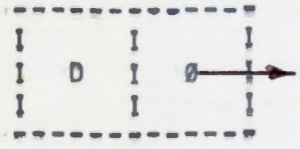
СПЕРАЦИЯ ИЗВЛЕЧЕНИЯ СИМВОЛА ИЗ СТЕКА МОЖЕТ БЫТЬ ОПИСАНА, НАПРИМЕР,ПРОЦЕДУРОЙ **OUT**. ПУСТЬ ПЕРЕМЕННАЯ **STACK** УКАЗЫВАЕТ НА ПОСЛЕДНИЙ ЭЛЕМЕНТ ЦЕПОЧКИ ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ.

```
PROCEDURE OUT(VAR X:CHAR);
VAR OLDCOMP:STACKP;
 BEGIN
      OLDCOMP:=STACK;
      X:=OLDCOMP↑.C;
      STACK:=OLDCOMP↑.NEXT;
      DISPOSE(OLDCOMP)
 END;
```

ЗДЕСЬ ОПЕРАТОР **STACK:=OLDCOMP↑.NEXT** ЗАСЫЛАЕТ В ПОЛЕ УКАЗАТЕЛЯ ПЕРЕМЕННОЙ **STACK** АДРЕС ПРЕДПОСЛЕДНЕГО ЭЛЕМЕНТА ЦЕПОЧКИ



ОПЕРАТОР DISPOSE(OLDCOMP) СТИРАЕТ ПОСЛЕДНИЙ ЭЛЕМЕНТ ЦЕПОЧКИ



ОБРАТИВШИСЬ СО СТЕКОМ ВИДА 2) К **OUT**, ПОЛУЧИМ СТЕК ВИДА 1).

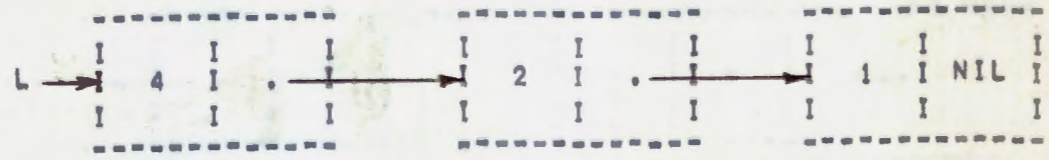
ВЫПОЛНЕНИЕ РАВНОГО КОЛИЧЕСТВА ОБРАЩЕНИЙ К **PUSN** И **OUT** ДАЕТ **STACK=NIL**, ЕСЛИ ПЕРВОНАЧАЛЬНО СТЕК БЫЛ ПУСТ.

12.3. ОЧЕРЕДЬ

ОЧЕРЕДЬ - ТАКАЯ СТРУКТУРА ДАННЫХ, ПРИ КОТОРОЙ ИЗЪЯТИЕ КОМПОНЕНТ ПРОИСХОДИТ ИЗ НАЧАЛА ЦЕПОЧКИ, А ЗАПИСЬ - В КОНЕЦ ЦЕПОЧКИ. В ЭТОМ СЛУЧАЕ ВВОДЯТ ДВА УКАЗАТЕЛЯ: ОДИН НА НАЧАЛО ОЧЕРЕДИ, ДРУГОЙ - НА ЕЕ КОНЕЦ.

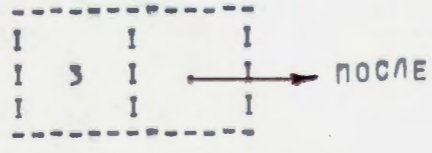
12.4. ДОЗАПИСЬ НОВЫХ КОМПОНЕНТ

ПРИМЕР. ПУСТЬ ИМЕЕТСЯ ЦЕПОЧКА ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ:

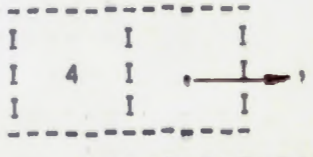


ПЕРЕМЕННЫЕ ИМЕЮТ ОПИСАННЫЙ ВЫШЕ ТИП **LISTP**.

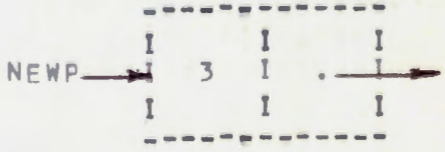
ТРЕБУЕТСЯ ВСТАВИТЬ КОМПОНЕНТУ



ПОСЛЕ



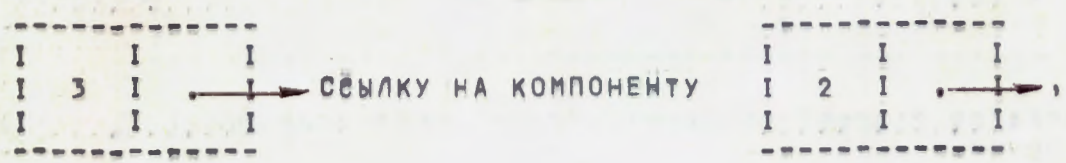
ЕСЛИ ИЗВЕСТЕН УКАЗАТЕЛЬ



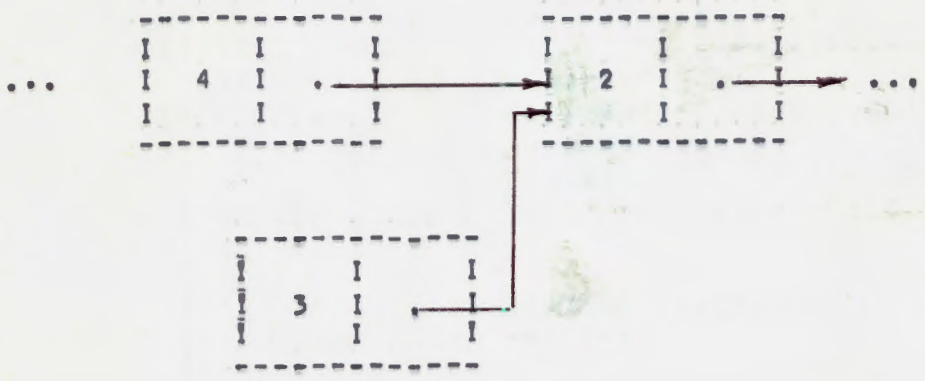
ДЛЯ ЗАПИСИ ЭТОЙ НОВОЙ КОМПОНЕНТЫ ДОСТАТОЧНО ВЫПОЛНИТЬ ОПЕРАТОРЫ:

```
NEWP↑.NEXT:=L↑.NEXT;
L↑.NEXT:=NEWP;
```

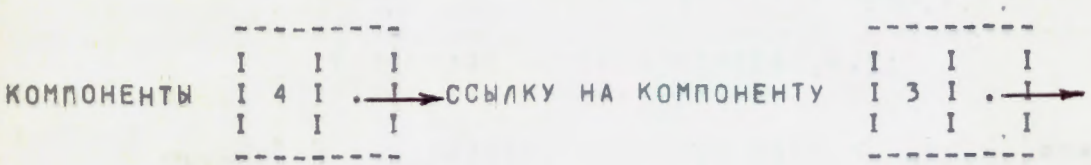
ПЕРВЫЙ ОПЕРАТОР ЗАСЫЛАЕТ В ПОЛЕ УКАЗАТЕЛЯ КОМПОНЕНТЫ

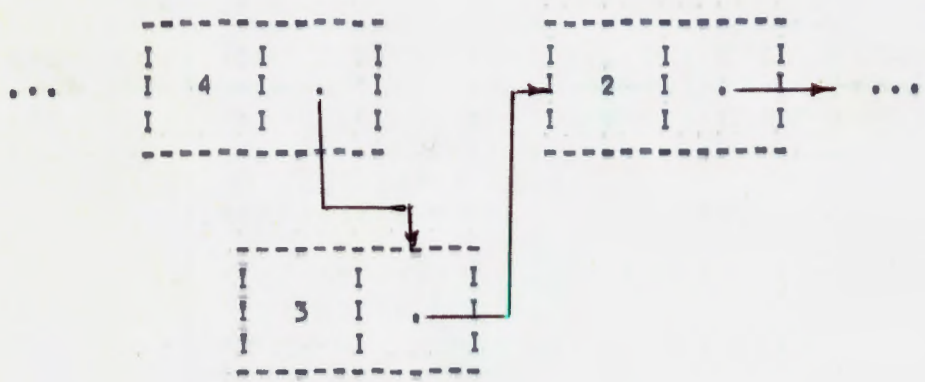


Т.Е. ПОЛУЧАЕТСЯ КАРТИНА:



ВТОРОЙ ОПЕРАТОР ПОМЕЩАЕТ В ПОЛЕ УКАЗАТЕЛЯ

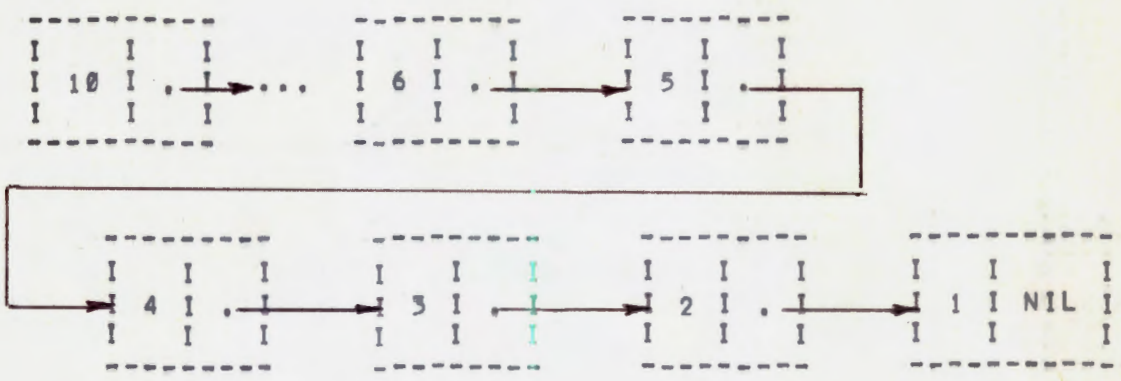




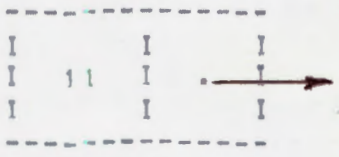
ПРИМЕР.

------- ПОСТРОИТЬ ЦЕПОЧКУ ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ, СОДЕРЖАЩИХ ЦЕЛЫЕ ЧИСЛА, А ЗАТЕМ МЕЖДУ 4-Й И 5-Й ПЕРЕМЕННОЙ ВСТАВИТЬ НОВУЮ ДИНАМИЧЕСКУЮ ПЕРЕМЕННУЮ.

ПУСТЬ ТРЕБУЕТСЯ ПОСТРОИТЬ ТАКУЮ ЦЕПОЧКУ:



И ВСТАВИТЬ ЭЛЕМЕНТ




```
PROGRAM POINT(INPUT,OUTPUT);
TYPE INTP=  INTREC;
     INTREC=RECORD
              I:INTEGER;
              NEXT:INTP
            END;
VAR IP,IR,INSERT1,INSERT4,INSERT5,WRTP:INTP;
    N,K,L,M:INTEGER;
BEGIN
    IR:=NIL; N:=10;
       FOR K:=1 TO N DO
        BEGIN READ(L); WRITE(' ',L);
          NEW(IP); IP↑.I:=L; IP↑.NEXT:=IR;
            IR:=IP;
    IF K=5 THEN BEGIN INSERT5:=IP;INSERT4:=IP↑.NEXT
                      END
        END;
 WRITELN;
    READ(L); WRITELN(' ',L);
 NEW((INSERT1));INSERT1↑.I:=L;INSERT1↑.NEXT:=INSERT4;
    INSERT5↑.NEXT:=INSERT1;
 FOR K:=1 TO N+1 DO
   BEGIN WRTP:=IR; M:=WRTP↑.I; WRITE(' ',M);
      IR:=WRTP↑.NEXT;
      DISPOSE(WRTP); WRITELN
   END
END.
```

РЕЗУЛЬТАТ:

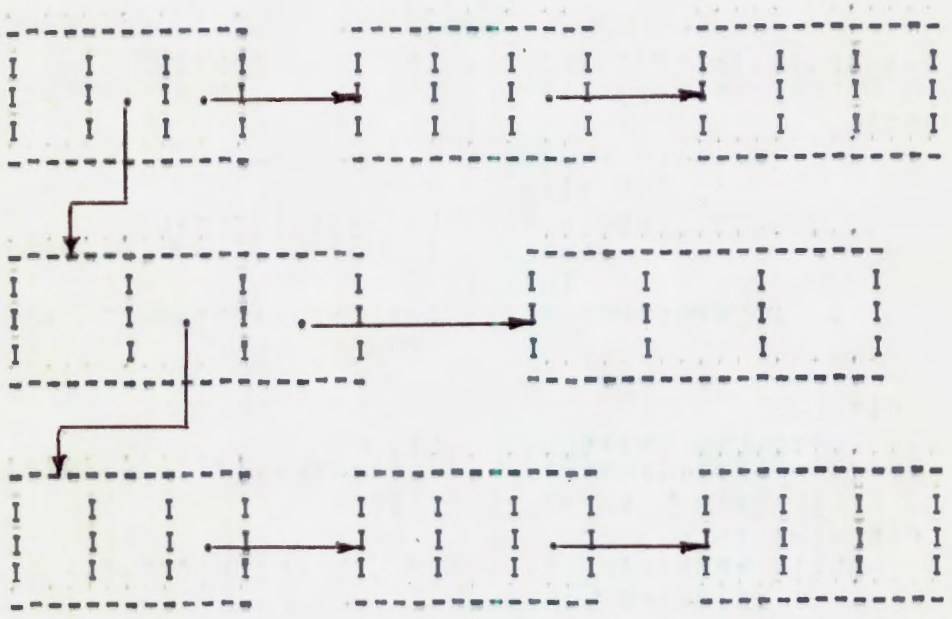
```
  1     2     3     4     5     6  . . .
 11
 10
  9
  8
  7
  6
  5
 11
  4
  3
  2
  1
```

ВВОДЯТСЯ ЧИСЛА: 1; 2; 3; 4; 5; 6; 7; 8; 9; 10 И 11.
МЕЖДУ 4 И 5 ВСТАВЛЯЕТСЯ ЧИСЛО 11.

12.5. НЕЛИНЕЙНЫЕ СТРУКТУРЫ

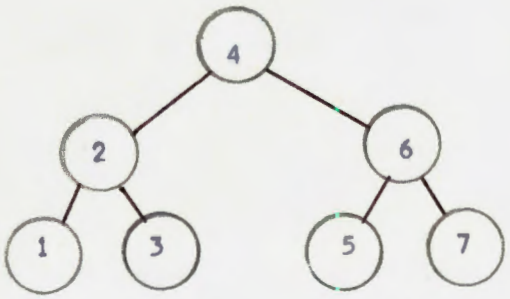
ВВЕДЕНИЕ В ДИНАМИЧЕСКУЮ ПЕРЕМЕННУЮ ДВУХ И БОЛЕЕ УКАЗАТЕЛЕЙ СОЗДАЕТ ВОЗМОЖНОСТЬ ПОЛУЧИТЬ НЕЛИНЕЙНЫЕ СТРУКТУРЫ.

ПРИМЕР.



В УЗЛОВЫХ РЕКОРДАХ СОДЕРЖИТСЯ ИНФОРМАЦИЯ О СЛЕДУЮЩЕМ ЭЛЕМЕНТЕ ДАННОЙ СТРОКИ И ПЕРВОМ ЭЛЕМЕНТЕ СЛЕДУЮЩЕЙ.

НЕЛИНЕЙНЫЕ СТРУКТУРЫ УДОБНЫ ДЛЯ ЗАДАЧ ПОИСКА.ПРЕДСТАВИМ СПИСОК УПОРЯДОЧЕННЫХ ЭЛЕМЕНТОВ В ВИДЕ ДВОИЧНОГО ДЕРЕВА:



УЗЕЛ (4) НАЗЫВАЕТСЯ КОРНЕМ ДЕРЕВА. ВХОД В ДЕРЕВО ПРОИСХОДИТ ТОЛЬКО ЧЕРЕЗ КОРЕНЬ.ДЛЯ КАЖДОГО УЗЛА РАЗЛИЧАЮТСЯ ЛЕВОЕ И ПРАВОЕ ПОДДЕРЕВЬЯ. УПОРЯДОЧЕННОСТЬ ЭЛЕМЕНТОВ СЛЕДУЮЩАЯ:ЭЛЕМЕНТЫ ЛЕВОГО ПОДДЕРЕВА < УЗЛА < ЭЛЕМЕНТОВ ПРАВОГО ПОДДЕРЕВА.

ВРЕМЯ ПОИСКА В ДВОИЧНОМ ДЕРЕВЕ СОКРАЩАЕТСЯ ПО СРАВНЕНИЮ С ЛИНЕЙНОЙ СТРУКТУРОЙ ПРИМЕРНО В LOG ПО ОСНОВАНИЮ 2 ОТ N, ГДЕ N - ЧИСЛО УЗЛОВ В ДЕРЕВЕ. КОМПОНЕНТУ ДВОИЧНОГО ДЕРЕВА МОЖНО ПРЕДСТАВИТЬ РЕКОРДОМ ВИДА:

```
BIREC = RECORD
        I:INTEGER;
        PRED,SUCC:INTP
        END;
```

13.ПРОЦЕДУРЫ

ПРОЦЕДУРА ПАСКАЛЯ - АНАЛОГ ФОРТРАННОЙ ПОДПРОГРАММЫ

ОБЩИЙ ВИД ЗАГОЛОВКА:

```
PROCEDURE N (P1:T1;P2:T2;VAR P3:T3;...);
```

ЗДЕСЬ N - ИМЯ ПРОЦЕДУРЫ.
PI - ФОРМАЛЬНЫЕ ПАРАМЕТРЫ.
TI - ИХ ТИПЫ.

ПРОЦЕДУРА ИМЕЕТ ТУ ЖЕ СТРУКТУРУ, ЧТО И ГЛАВНАЯ ПРОГРАММА (PROGRAM) :РАЗДЕЛЫ LABEL,CONST,TYPE,VAR И ВЫПОЛНЯЕМУЮ ЧАСТЬ ОТ BEGIN ДО END.

ПРОЦЕДУРА ПОМЕЩАЕТСЯ В ГЛАВНОЙ ПРОГРАММЕ ПОСЛЕ РАЗДЕЛА VAR И ПЕРЕД BEGIN ПРОГРАММЫ.

```
PROGRAM MA (INPUT,OUTPUT);
                .
                .
                .
VAR A:TYPE1; B:TYPE2;...
PROCEDURE N (P1:T1; P2:T2;...);
    .
    .
    .
END;
BEGIN
  .
  .
  .
END.
```

ПАРАМЕТРЫ ПРОЦЕДУР МОГУТ БЫТЬ ЧЕТЫРЕХ ВИДОВ:

ПАРАМЕТРЫ-ЗНАЧЕНИЯ,
ПАРАМЕТРЫ-ПЕРЕМЕННЫЕ,
ПАРАМЕТРЫ-ПРОЦЕДУРЫ,
ПАРАМЕТРЫ-ФУНКЦИИ.

ЕСЛИ ФОРМАЛЬНЫЙ ПАРАМЕТР ЕСТЬ ПАРАМЕТР-ЗНАЧЕНИЕ, ТО СООТВЕТСТВУЮЩИМ ФАКТИЧЕСКИМ ПАРАМЕТРОМ ДОЛЖНО БЫТЬ ВЫРАЖЕНИЕ ТОГО ЖЕ ТИПА, ЧТО И ФОРМАЛЬНЫЙ ПАРАМЕТР.

ЕСЛИ ФОРМАЛЬНЫЙ ПАРАМЕТР ЕСТЬ ПАРАМЕТР-ПЕРЕМЕННАЯ, ТО В ЗАГОЛОВКЕ ПРОЦЕДУРЫ ПЕРЕД ИДЕНТИФИКАТОРОМ ЭТОГО ПАРАМЕТРА ПИШУТ "СЛОВО" VAR.

ПРИМЕР. VAR P3:T3;

ПЕРЕМЕННЫЕ,ДЕКЛАРИРОВАННЫЕ В КАКОЙ-ЛИБО ПРОЦЕДУРЕ (ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ), МОГУТ ИСПОЛЬЗОВАТЬСЯ ТОЛЬКО В ДАННОЙ ПРОЦЕДУРЕ И В ПРОЦЕДУРАХ, ВЛОЖЕННЫХ В ДАННУЮ.
ПЕРЕМЕННЫЕ И ТИПЫ, ОПИСАННЫЕ В PROGRAM (ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ) МОЖНО ИСПОЛЬЗОВАТЬ ВО ВСЕХ ВЛОЖЕННЫХ В НЕЕ ПОЦЕДУРАХ.
ФАКТИЧЕСКИЕ ПАРАМЕТРЫ РАЗДЕЛЯЮТСЯ ЗАПЯТЫМИ, ФОРМАЛЬНЫЕ - ";".

ПОД ФОРМАЛЬНЫЕ И ФАКТИЧЕСКИЕ ПАРАМЕТРЫ ТРАНСЛЯТОР ОТВОДИТ РАЗНЫЕ ОБЛАСТИ ПАМЯТИ. ПОЭТОМУ РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОЦЕДУРЫ МОЖЕТ БЫТЬ ПЕРЕДАН ТОЛЬКО ЧЕРЕЗ ПАРАМЕТР -ПЕРЕМЕННУЮ.

ПРИМЕР.

```
PROGRAM L13T1(OUTPUT);
VAR A,B:INTEGER;
   PROCEDURE H(X:INTEGER; VAR Y:INTEGER);
        BEGIN
                X:=X+1; Y:=Y+1;
          WRITELN(X,Y)
       END;
```

```
BEGIN A:=0; B:=0;
H(A,B);
WRITELN(A,B)
END.
```

РЕЗУЛЬТАТЫ, ВЫДАВАЕМЫЕ ПРОЦЕДУРОЙ H : 1 И 1 ; PROGRAM ПЕЧАТАЕТ 0 И 1 , Т.К. ЯЧЕЙКА, СООТВЕТСТВУЮЩАЯ ФАКТИЧЕСКОМУ ПАРАМЕТРУ-ЗНАЧЕНИЮ A , СОДЕРЖИТ ПО-ПРЕЖНЕМУ ЗНАЧЕНИЕ 0 .

ПРОЦЕДУРЫ В ПАСКАЛЕ ДОПУСКАЮТ РЕКУРСИЮ, Т.Е. ПРОЦЕДУРА МОЖЕТ ВЫЗЫВАТЬ САМА СЕБЯ.

ЕСЛИ В ПРОЦЕДУРЕ P ЕСТЬ ОБРАЩЕНИЕ К ПРОЦЕДУРЕ Q , ОПИСАННОЙ НИЖЕ, ТО ПЕРЕД ОПИСАНИЕМ P ПРОЦЕДУРА Q ДЕКЛАРИРУЕТСЯ КАК FORWARD.

ПРИМЕР.

```
PROCEDURE Q(X:T1):FORWARD;
PROCEDURE P(X;Y);
    BEGIN
      Q(A);
    END;

PROCEDYPE Q;  (ПАРАМЕТРЫ ПРОЦЕДУРЫ НЕ ПОВТОРЯЮТСЯ)
    BEGIN
      P(B)
    END;
```

14. ФУНКЦИИ

АНАЛОГОМ ФОРТРАННОЙ ПОДПРОГРАММЫ-ФУНКЦИИ ЯВЛЯЕТСЯ ФУНКЦИЯ ПАСКАЛЯ.

ВИД ЗАГОЛОВКА:

```
FUNCTION F(P1:T1;...FUNCTION Q:TN):TYPEF;
```

ЗДЕСЬ F — ИМЯ ФУНКЦИИ,
TYPEF — ТИП РЕЗУЛЬТАТА,
P1,...Q — ФОРМАЛЬНЫЕ ПАРАМЕТРЫ,
T1,...TN,... — ИХ ТИПЫ.

ЕСЛИ ПРОЦЕДУРЫ ИЛИ ФУНКЦИИ ИСПОЛЬЗУЮТСЯ В КАЧЕСТВЕ ПАРАМЕТРОВ, ТО ОНИ ДОЛЖНЫ ИМЕТЬ СВОИМИ ФОРМАЛЬНЫМИ ПАРАМЕТРАМИ ТОЛЬКО ПАРАМЕТРЫ-ЗНАЧЕНИЯ.
ТЕКСТ FUNCTION , КАК И PROCEDURE , ПОМЕЩАЕТСЯ В PROGRAM ПОСЛЕ РАЗДЕЛА VAR И ПЕРЕД ПЕРВЫМ ОПЕРАТОРОМ BEGIN ПРОГРАММЫ.

АЛГОРИТМЫ, УПОТРЕБЛЯЕМЫЕ НАИБОЛЕЕ ЧАСТО РАЗЛИЧНЫМИ ПОЛЬЗОВАТЕЛЯМИ, ОФОРМЛЯЮТСЯ В ВИДЕ ПРОЦЕДУР И ФУНКЦИЙ, ПОМЕЩАЮТСЯ В ПАМЯТЬ МАШИНЫ И СОСТАВЛЯЮТ БИБЛИОТЕКУ СТАНДАРТНЫХ ПРОГРАММ (МОДУЛЕЙ).

15. РАБОТА С ВНЕШНИМИ МОДУЛЯМИ

ПАСКАЛЬ ПОЗВОЛЯЕТ РАБОТАТЬ С ВНЕШНИМИ ПРОЦЕДУРАМИ (ФУНКЦИЯМИ), КОТОРЫЕ СУЩЕСТВУЮТ ВНЕ ГЛАВНОЙ ПРОГРАММЫ (PROGRAM)

ЕСЛИ МОДУЛЬ ЯВЛЯЕТСЯ СТАНДАРТНЫМ (БИБЛИОТЕЧНЫМ), ТО НИКАКИХ ОПИСАНИЙ В ПРОГРАММЕ ЭТОТ МОДУЛЬ НЕ ТРЕБУЕТ.

В ОСТАЛЬНЫХ СЛУЧАЯХ ВНЕШНИЙ МОДУЛЬ ДОЛЖЕН БЫТЬ ОПИСАН В PROGRAM СЛЕДУЮЩИМ ОБРАЗОМ:

1). ПРОЦЕДУРА ПАСКАЛЯ:

```
PROCEDURE N(P1:T1;...); EXTERNAL;
```

ЗДЕСЬ N - ИМЯ ПРОЦЕДУРЫ, P1,... - ФОРМАЛЬНЫЕ ПАРАМЕТРЫ, T1,... ТИПЫ.

2) ФУНКЦИЯ ПАСКАЛЯ:

```
  FUNCTION FPSCL(X:TYPE1;...): TYPEF; EXTERNAL;
```

ЗДЕСЬ FPSCL - ИМЯ ФУНКЦИИ,
X - ФОРМАЛЬНЫЙ ПАРАМЕТР,
TYPE1 - ТИП ЭТОГО ПАРАМЕТРА,
TYPEF - ТИП РЕЗУЛЬТАТА,
EXTERNAL(EXTERN)- УКАЗАНИЕ НА ТО, ЧТО ЭТА

ФУНКЦИЯ ОПИСАНА ВНЕ ДАННОЙ ПРОГРАММЫ.
НА БЭСМ-6 СЛЕДУЕТ ПИСАТЬ EXTERNAL НА CDC - EXTERN. ЕС ЭВМ ДОПУСКАЕТ ОБА НАПИСАНИЯ.

3) ПОДПРОГРАММА ФОРТРАНА:

```
PROCEDURE SUB(X:TYPEX;...VAR Z:TYPEZ); FORTRAN;
```

ЗДЕСЬ X - ПАРАМЕТР,
TYPEX - ЕГО ТИП,
Z - ПАРАМЕТР-РЕЗУЛЬТАТ,
TYPEZ - ЕГО ТИП,
FORTRAN - УКАЗАНИЕ НА ТО, ЧТО ИСПОЛЬЗОВАН ФОРТРАННЫЙ МОДУЛЬ.

4) ПОДПРОГРАММА - ФУНКЦИЯ ФОРТРАНА:

```
  FUNCTION F(X:TYPEX;...):TYPEF; FORTRAN;
```

ПРИМЕР. (ЭВМ БЭСМ-6)

```
*NAME S
*PASS: ******
*TIME: 00.06
*LIBRARY: 11
      SUBROUTINE SUB(X)
      X=2
      RETURN
      END
      FUNCTION F(X)
      F=4*X
      RETURN
      END
```

```
*CALL ALLMEMORY
*PASCAL
PROGRAM OUT(OUTPUT)
(*=E+*)        (СМ. "РЕЖИМЫ ТРАНСЛЯЦИИ")
VAR B:REAL;
PROCEDURE PROC(VAR X:REAL);
  BEGIN X:=1.0 END;
FUNCTION FUNPSCL (X:REAL):REAL;
  BEGIN FUNPSCL:=3.0 END;
BEGIN B:=12.0 END.
PROGRAM GETHER (OUTPUT);
 VAR A,B:REAL;
  PROCEDURE SUB(VAR X:REAL);FORTRAN;
  FUNCTION F(X:REAL):REAL;FORTRAN;
  PROCEDURE PROC (VAR X:REAL);EXTERNAL;
  FUNCTION FUNPSCL(X:REAL):REAL;EXTERNAL;
     BEGIN A:=1.0;
      SUB(B);WRITE(' SUB(B):',B,'F(A)=',F(A));
      PROC(B);WRITE(' PROC(B):',B,'FUNPSCL(A)=',FUNPSCL(A)
     END.

*EXECUTE
```

В ЭТОМ ПРИМЕРЕ ВЫПОЛНЯЕТСЯ ПРОГРАММА **GETHER** ,ИСПОЛЬЗУЮЩАЯ ВНЕШНИЕ ФОРТРАННЫЕ МОДУЛИ **SUB** И **F** , А ТАКЖЕ МОДУЛИ ПАСКАЛЯ **PROC** И **FUNPSCL** , ОПИСАННЫЕ В ПРОГРАММЕ **OUT**.

16. РЕЖИМЫ ТРАНСЛЯЦИИ

РЕЖИМ ТРАНСЛЯЦИИ ЗАДАЕТСЯ КОММЕНТАРИЯМИ ОСОБОГО ВИДА (ПРЕВДОКОММЕНТАРИЯМИ):

(*СН R1,R2,...*)

ЗДЕСЬ (* И *) - ОГРАНИЧИТЕЛИ.
СН - СПЕЦИАЛЬНЫЙ СИМВОЛ,ПРЕВРАЩАЮЩИЙ КОММЕНТАРИЙ В УПРАВЛЯЮЩУЮ КАРТУ,
R1,R2 - ЗАДАВАЕМЫЕ КОДЫ РЕЖИМОВ ТРАНСЛЯЦИИ.

НА БЭСМ-6 СН ЕСТЬ " = ", НА CDC-6500 И НА ЕС ЭВМ ЕСТЬ "ДОЛЛАР"(ЛИБО СОВПАДАЮЩИЙ С НИМ ПО КОДИРОВКЕ ЗНАК $)
КАЖДЫЙ КОД РЕЖИМА СОСТОИТ ИЗ БУКВЫ И ЛИБО ЗНАКОВ " + " И " - ", ЛИБО ЦИФР.
ЗНАК " + " ОЗНАЧАЕТ ВКЛЮЧЕНИЕ ДАННОГО РЕЖИМА,
ЗНАК " - " ОТКАЗ ОТ НЕГО.

ПРИМЕР.

(*=T+,E*,P-*) ЛИБО (*$T+,E+,P-*)

ЧАЩЕ ВСЕГО ИСПОЛЬЗУЮТСЯ СЛЕДУЮЩИЕ РЕЖИМЫ:

1) T - ЭТОТ РЕЖИМ ОБЕСПЕЧИВАЕТ ДИНАМИЧЕСКИЕ ПРОВЕРКИ ВО ВРЕМЯ СЧЕТА:
   А) ВСЕХ ОПЕРАЦИЙ С ИНДЕКСНЫМИ ПЕРЕМЕННЫМИ НА ПРИНАДЛЕЖНОСТЬ ИНДЕКСА ДЕКЛАЛИРОВАННОМУ ИНТЕРВАЛУ ИНДЕКСОВ;
   Б) ВСЕХ ОПЕРАТОРОВ ПРИСВАИВАНИЯ НА ПРИНАДЛЕЖНОСТЬ ЗНАЧЕНИЙ ПЕРЕМЕННЫХ ТИПА **SUBRANGE** СООТВЕТСТВУЮЩЕМУ ПОДМНОЖЕСТВУ;
   В) ВСЕХ ДЕЛИТЕЛЕЙ В ОПЕРАЦИЯХ ДЕЛЕНИЯ (НА НОЛЬ);

   Г) ВСЕХ АВТОМАТИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

      INTEGER REAL НА ABS(I)<=MAXINT;

   Д) ВСЕХ ОПЕРАТОРОВ **CASE** НА СООТВЕТСТВИЕ ПЕРЕКЛЮЧАТЕЛЯ ОДНОЙ ИЗ МЕТОК **CASE**.

      ПО УМОЛЧАНИЮ T+.

   ДЛЯ ОТЛАЖЕННЫХ ПРОГРАММ РЕКОМЕНДУЕТСЯ ИСПОЛЬЗОВАТЬ **T-.**

2) P- ПОЗВОЛЯЕТ ВЫДАВАТЬ ПОДРОБНУЮ ИНФОРМАЦИЮ ПРИ АВОСТАХ. В РЕЖИМЕ **P+** ВЫДАЮТСЯ ЗНАЧЕНИЯ ЛОКАЛЬНЫХ ПЕРЕМЕННЫХ, ИДЕНТИФИКАТОРЫ ВЫЗВАННЫХ ПРОЦЕДУР (ФУНКЦИЙ) И НОМЕРА СТРОК ПРОГРАММЫ, В КОТОРЫХ НАЧИНАЮТСЯ СООТВЕТСТВУЮЩИЕ СОСТАВНЫЕ ОПЕРАТОРЫ.

   ПО УМОЛЧАНИЮ P+.

   ДЛЯ ОТЛАЖЕННЫХ ПРОГРАММ СЛЕДУЕТ УКАЗЫВАТЬ РЕЖИМ **P-**, ЧТО ЭКОНОМИТ ПАМЯТЬ И ВРЕМЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ.

3) E - ПОЗВОЛЯЕТ ТАК ТРАНСЛИРОВАТЬ ПРОЦЕДУРЫ И ФУНКЦИИ, ЧТО К НИМ МОЖНО ОБРАЩАТЬСЯ ИЗ ДРУГИХ ПРОГРАММ КАК К ВНЕШНИМ МОДУЛЯМ .
   ЕСЛИ ДАННАЯ ПРОЦЕДУРА ИСПОЛЬЗУЕТСЯ КАК **EXTERNAL**, ТО ЕЕ НЕОБХОДИМО ТРАНСЛИРОВАТЬ ТОЛЬКО В РЕЖИМЕ **E+.**

   ПО УМОЛЧАНИЮ E -.

4) U - ВСЕ СИМВОЛЫ ВХОДНОЙ СТРОКИ, НАЧИНАЯ С 73-ГО, СЧИТАЮТСЯ КОММЕНТАРИЯМИ. НА БЭСМ-6 РЕЖИМА **U** НЕТ. ЕСЛИ ИСПОЛЬЗУЕТСЯ **U-** , ТО ВСЕ СИМВОЛЫ, НАЧИНАЯ С 121-ГО, СЧИТАЮТСЯ КОММЕНТАРИЯМИ.

5) BN- ДЛЯ БЭСМ-6 И CDC.
   ПУСТЬ **S** - НИЖНЯЯ ГРАНИЦА РАЗМЕРА ПАМЯТИ, ВЫДЕЛЕННОЙ ПОД БУФЕР ФАЙЛОВ

НА	CDC-65000	S > 128 * N.
	БЭСМ-6	S > 256 * N,
ПО УМОЛЧАНИЮ	CDC-6500	N=4;
	БЭСМ-6	N=1:

B+ ДЛЯ ЕС ЭВМ - ЗАРЕЗЕРВИРОВАННЫЕ ПАСКАЛЕМ КЛЮЧЕВЫЕ СЛОВА (AND,ARRAY,...WITH) НА ЛИСТИНГЕ АЦПУ

ПЕЧАТАЮТСЯ ЖИРНО.

ПО УМОЛЧАНИЮ - B -.

6) L УПРАВЛЯЕТ ИНФОРМАЦИЕЙ ОБ ИСХОДНОЙ ПРОГРАММЕ, ВЫДАВАЕМОЙ НА АЦПУ.

ЕС ЭВМ И CDC-6500:

L+ - ПОДРОБНАЯ ВЫДАЧА,

L- - РЕЖИМ СЧЕТА, ПОДАВЛЕНИЕ ЛИСТИНГА ПРОГРАММЫ.

ПО УМОЛЧАНИЮ L+

БЭСМ-6:

L0 - ВЫДАЮТСЯ ТОЛЬКО СООБЩЕНИЯ ОБ ОШИБКАХ (*NO LIST);

L1 - ВЫДАЕТСЯ ТАБЛИЦА ЗАГРУЗКИ И ТЕКСТ ПРОГРАММЫ;

L2 - ДОПОЛНИТЕЛЬНО К ИНФОРМАЦИИ ПО L1 ВЫДАЮТСЯ КОДЫ СТАНДАРТНОГО МАССИВА.

ПО УМОЛЧАНИЮ L1.

17. СТАНДАРТНЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ

1) ORD(C) - ВЫДАЕТ ПОРЯДКОВЫЙ НОМЕР СИМВОЛА C В УПОРЯДОЧЕННОЙ ПОСЛЕДОВАТЕЛЬНОСТИ СИМВОЛОВ, ЗАДАВАЕМОЙ ОПЕРАЦИОННОЙ СИСТЕМОЙ. НУМЕРАЦИЯ НАЧИНАЕТСЯ С НУЛЯ.

2) CHR(I) - ВЫДАЕТ СИМВОЛ С ПОРЯДКОВЫМ НОМЕРОМ I. ФУНКЦИИ ORD(C) И CHR(I) ВЗАИМНООБРАТНЫЕ, Т.Е. CHR(ORD(C))=C И ORD(CHR(I))=I. В СИЛУ УПОРЯДОЧЕННОСТИ ПОСЛЕДОВАТЕЛЬНОСТИ СИМВОЛОВ, ORD(C_1)<ORD(C_2), ЕСЛИ $C_1<C_2$ И ВООБЩЕ, ЕСЛИ R - ЛОГИЧЕСКАЯ ОПЕРАЦИЯ ОТНОШЕНИЯ (=,<>,<,>,<=,>=), ТО ИЗ ORD(C_1) R ORD(C_2) СЛЕДУЕТ C_1 R C_2 И НАОБОРОТ.

3) PRED(C) - ВЫДАЕТ СИМВОЛ, ПРЕДШЕСТВУЮЩИЙ C В УПОРЯДОЧНОЙ ПОСЛЕДОВАТЕЛЬНОСТИ СИМВОЛОВ.

4) SUCC(C) - ВЫДАЕТ СИМВОЛ, СЛЕДУЮЩИЙ ЗА C. РЕЗУЛЬТАТ ЭТИХ ФУНКЦИИ БУДЕТ НЕОПРЕДЕЛЕННЫМ, ЕСЛИ ССОТВЕТСТВУЮЩИХ СИМВОЛОВ НЕ СУЩЕСТВУЕТ; В ЭТОМ СЛУЧАЕ НЕ ВСЕ ТРАНСЛЯТОРЫ ВЫДАЮТ ДИАГНОСТИКУ.

5) ODD(X) - ВЫДАЕТ TRUE, ЕСЛИ X НЕЧЕТНО.

6) ЭЛЕМЕНТАРНЫЕ МАТЕМАТИЧЕСКИЕ ФУНКЦИИ:

ABS(X) - МОДУЛЬ X
SQR(X) - КВАДРАТ ЧИСЛА X
(ОБРАТИТЬ ВНИМАНИЕ: НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ ВЫРАЖЕНИЕ X**2!)

TRUNC(X) - ЦЕЛАЯ ЧАСТЬ X, ПОЛУЧАЮЩАЯСЯ ПОСЛЕ ОТБРАСЫВАНИЯ ЗНАКОВ ПОСЛЕ ДЕСЯТИЧНОЙ ТОЧКИ.

ПРИМЕР.

TRUNC(3.9) ДАЕТ 3
TRUNC(-3.9) ДАЕТ - 3.

РЕЗУЛЬТАТ - ЦЕЛЫЙ, X - ВЕЩЕСТВЕННЫЙ.

ROUND(X) - X - ВЕЩЕСТВЕННЫЙ, ОКРУГЛЯЕТСЯ ДО ЦЕЛОГО, ВЫДАВАЕМОГО КАК РЕЗУЛЬТАТ.

ПРИМЕР.

ROUND(3.9) ДАЕТ 4
ROUND(-3.9) ДАЕТ - 4

АРГУМЕНТ СЛЕДУЮЩИХ ФУНКЦИЙ МОЖЕТ БЫТЬ КАК ВЕЩЕСТВЕННЫМ, ТАК И ЦЕЛЫМ. РЕЗУЛЬТАТ - ВСЕГДА ВЕЩЕСТВЕННЫЙ.

LN(X) - НАТУРАЛЬНЫЙ ЛОГАРИФМ

EXP(X) - E В СТЕПЕНИ X

SQRT(X) - КВАДРАТНЫЙ КОРЕНЬ ИЗ X

SIN(X) - СИНУС X

COS(X) - КОСИНУС X

ARCTAN(X) - АРКТАНГЕНС X.

7) PACK(A,I,Z) - УПАКОВКА МАССИВА A С I -й КОМПОНЕНТЫ В МАССИВ Z (С 1-й КОМПОНЕНТЫ). МАССИВ Z ТИПА PACKED

8) UNPACK(Z,A,I) - РАСПАКОВКА МАССИВА Z С 1-й КОМПОНЕНТЫ В МАССИВ A С I -й КОМПОНЕНТЫ. МАССИВ Z ТИПА PACKED

9) RESET(F) - УСТАНАВЛИВАЕТ УКАЗАТЕЛЬ ФАЙЛА НА ПЕРВУЮ КОМПОНЕНТУ ФАЙЛА F ;ФУНКЦИИ EOF(F) ПРИСВАИВАЕТСЯ ЗНАЧЕНИЕ FALSE ,ЕСЛИ ФАЙЛ НЕ ПУСТ; ИНАЧЕ - ЗНАЧЕНИЕ F. НЕОПРЕДЕЛЕННО И EOF(F) ОСТАЕТСЯ TRUE.

10) REWRITE(F) - ОЧИЩАЕТ ФАЙЛ(Т.Е. F СТАНОВИТСЯ ПУСТЫМ ФАЙЛОМ), ПРИСВАИВАЕТ ФУНКЦИИ EOF(F) ЗНАЧЕНИЕ TRUE; К ФУНКЦИИ REWRITE(F) НЕОБХОДИМО ОБРАТИТЬСЯ ПЕРЕД ЗАПИСЬЮ В ПЕРВУЮ КОМПОНЕНТУ ФАЙЛА F.

11) GET(F) - ПРОДВИГАЕТ УКАЗАТЕЛЬ ФАЙЛА К СЛЕДУЮЩЕЙ КОМПОНЕНТЕ И ПРИСВАИВАЕТ ЗНАЧЕНИЕ ЭТОЙ КОМПОНЕНТЫ "ОКНУ" F ↑; ЕСЛИ ЭТА КОМПОНЕНТА -"КОНЕЦ ФАЙЛА", ТО ЗНАЧЕНИЕ F ↑ СТАНОВИТСЯ НЕОПРЕДЕЛЕННЫМ А EOF(F) ПРИНИМАЕТ ЗНАЧЕНИЕ TRUE.

12) PUT(F) - ЗАНОСИТ ЗНАЧЕНИЕ F ↑ В ТУ КОМПОНЕНТУ ФАЙЛА, КУДА УСТАНОВЛЕН УКАЗАТЕЛЬ, ЕСЛИ ЗНАЧЕНИЕ EOF(F) ЕСТЬ TRUE К МОМЕНТУ ВЫПОЛНЕНИЯ PUT(F). ПОСЛЕ ВЫПОЛНЕНИЯ PUT(F) ЗНАЧЕНИЕ F ↑ СТАНОВИТСЯ НЕОПРЕДЕЛЕННЫМ.

ПРИМЕР:

```
        VAR DATA : FILE OF INTEGER;
                 A : INTEGER;
                   .
                   .
                   .
                 A :=SQR(DATA↑ );
                   GET(DATA);
                   .
                   .
                   .
```

ЗДЕСЬ ОПЕРАТОР A :=SQR(DATA↑) ПРИСВАИВАЕТ ПЕРЕМЕННОЙ A КВАДРАТ ТЕКУЩЕЙ КОМПОНЕНТЫ ФАЙЛА DATA; ОПЕРАТОР GET(DATA) ПРОДВИГАЕТ УКАЗАТЕЛЬ К СЛЕДУЮЩЕЙ КОМПОНЕНТЕ ФАЙЛА И ЧИТАЕТ ЕЕ В DATA↑.

13) WRITELN(X) - ЗАПИСЫВАЕТ СИМВОЛ КОНЦА СТРОКИ В ТЕКУЩУЮ КОМПОНЕНТУ ТЕКСТОВОГО ФАЙЛА X

14) READLN(X) - ПРОПУСКАЕТ ОСТАТОК ТЕКУЩЕЙ СТРОКИ, УСТАНАВЛИВАЕТ УКАЗАТЕЛЬ ФАЙЛА НА НАЧАЛО СЛЕДУЮЩЕЙ СТРОКИ ТЕКСТОВОГО ФАЙЛА X ;

15) EOLN(X) - ПРИНИМАЕТ ЗНАЧЕНИЕ TRUE, ЕСЛИ УКАЗАТЕЛЬ ФАЙЛА УСТАНОВЛЕН НА СИМВОЛ КОНЦА СТРОКИ (FALSE В ПРОТИВНОМ СЛУЧАЕ) И В X ↑ ЗАСЫЛАЕТ ПРОБЕЛ.

16) EOF(X) - ПРИНИМАЕТ ЗНАЧЕНИЕ TRUE, ЕСЛИ X↑ СООТВЕТСТВУЕТ КОНЦУ ФАЙЛА(FALSE -В ПРОТИВНОМ СЛУЧАЕ)

17) NEW(P) - ОТВОДИТ МЕСТО В ПАМЯТИ ДЛЯ НОВОЙ ДИНАМИЧЕСКОЙ ПЕРЕМЕННОЙ И ЗАПОМИНАЕТ ЕЕ АДРЕС.

18) NEW(P,P1) - ОТВОДИТ МЕСТО В ПАМЯТИ ДЛЯ ЗАПИСИ (RECORD) С ВАРИАНТОМ P1 .

19) DISPOSE(P) - СТИРАЕТ ДИНАМИЧЕСКУЮ ПЕРЕМЕННУЮ, НА КОТОРУЮ УКАЗЫВАЕТ ПЕРЕМЕННАЯ P .

20) DISPOSE(P,P1) - СТИРАЕТ ДИНАМИЧЕСКУЮ ПЕРЕМЕННУЮ, СОЗДАННУЮ ПРОЦЕДУРОЙ NEW(P,P1) И НА КОТОРУЮ УКАЗЫВАЕТ ПЕРЕМЕННАЯ P .

18. КОДИРОВКА СИМВОЛОВ, НЕ СОВПАДАЮЩИХ ЛИБО ОТСУТСТВУЮЩИХ НА КЛАВИАТУРЕ

1. ДЛЯ ЭВМ БЭСМ-6 ВСЕ СИМВОЛЫ, КРОМЕ ↑ , СООТВЕТСТВУЮТ СИМВОЛАМ КЛАВИАТУРЫ ДИСПЛЕЯ И ПЕРФОРАТОРА ЕС (КОДИРОВКА КПК-12-СССР).
   СИМВОЛ ↑ СООТВЕТСТВУЕТ СИМВОЛУ @ (КОММЕРЧЕСКОЕ А).
   ЕСЛИ КАРТЫ ПРОБИТЫ В КОДИРОВКЕ КПК-12-СССР, ТО ДЛЯ БЭСМ-6 С ОПЕРАЦИОННОЙ СИСТЕМОЙ "ДУБНА" В КАЧЕСТВЕ ПЕРВОЙ КАРТЫ НАДО ПОЛОЖИТЬ 7/9IBM (В РЕЖИМЕ MP ПРОБИТЬ 7 И 9 В ПЕРВОЙ ПОЗИЦИИ)

2. ДЛЯ ЭВМ CDC-6500 НИЖЕ ПРИВЕДЕНА ТАБЛИЦА СООТВЕТСТВИЯ НАБИРАЕМЫХ СИМВОЛОВ СИМВОЛАМ КЛАВИАТУРЫ.

НАБИРАЕМЫЙ СИМВОЛ	ДИСПЛЕЙ TEKTRONIX 4012	ПЕРФОРАТОР С КОДИРОВКОЙ CDC	ПЕРФОРАТОР ЕС С КОДИРОВКОЙ КПК-12-СССР
'(АПОСТРОФ)	#	0-6-8	#
:(ДВОЕТОЧИЕ)	:	2-8	:
;(ТОЧКА С ЗАП)	;	7-8-12	;
↑	АПОСТРОФ	5-8-11	◇
[	[	8-7	[
]	]	0-2-8	]
<	<	0-12	<
>	>	7-8-11	>

ЕСЛИ ЗАДАНИЕ ПОДГОТОВЛЕНО НА КАРТАХ В КОДИРОВКЕ КПК-12-СССР, ТО В 1-Й УПРАВЛЯЮЩЕЙ КАРТЕ (JOB-КАРТЕ) СЛЕДУЕТ ПРОБИТЬ В 78-Й, 79-Й И 80-Х ПОЗИЦИЯХ ЧИСЛО 029. ЕСЛИ В КОЛОДЕ НЕСКОЛЬКО РЕКОРДОВ КАРТ, ПРОБИТЫХ В КОДИРОВКЕ КПК-12-СССР, ТО В КАЖДОЙ КАРТЕ 7/8/9, ПРЕДШЕСТВУЮЩЕЙ ТАКОМУ РЕКОРДУ, ТАКЖЕ ПРОБИВАЕТСЯ ЧИСЛО 029 В ПОСЛЕДНИХ ПОЗИЦИЯХ.

3. НА ЕС ЭВМ ИСПОЛЬЗУЕТСЯ СЛЕДУЮЩЕЕ ПРЕДСТАВЛЕНИЕ СИМВОЛОВ:

СТАНДАРТНОЕ	[	]	{	}	AND	OR	NOT	<	>	↑
ПАСКАЛЬ 8000	(.	.)	(*	*)	&	|	¬	¬=		@

16. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О ПАСКАЛЕ ЕС ЭВМ(ПАСКАЛЬ 8000)

16.1 КАК ЧИТАТЬ ЛИСТИНГИ ПАСКАЛЬ-ЗАДАЧИ.

КАЖДАЯ СТРАНИЦА ЛИСТИНГА НАЧИНАЕТСЯ С ИНФОРМАЦИИ О ВЕРСИИ ТРАНСЛЯТОРА С ПАСКАЛЯ, ДАТЕ, ВРЕМЕНИ ЗАПУСКА ЗАДАЧИ.

ЗАТЕМ ВЫДАЕТСЯ ИНФОРМАЦИЯ О РЕЖИМЕ ТРАНСЛЯЦИИ. НАПРИМЕР, ЕСЛИ ЗАДАН B+ , ТО СЛЕДУЮЩАЯ СТРОКА БУДЕТ **INITIAL OPTIONS:B+** .

ДАЛЕЕ ИДЕТ ПРОГРАММА НА ПАСКАЛЕ.

ПЕРВАЯ ВЕРТИКАЛЬНАЯ КОЛОНКА ЧИСЕЛ СЛЕВА - НОМЕРА СООТВЕТСТВУЮЩИХ СТРОК ПРОГРАММЫ.

СЛЕДУЮЩАЯ ВЕРТИКАЛЬНАЯ КОЛОНКА ЧЕТЫРЕХЗНАЧНЫХ ЧИСЕЛ УКАЗЫВАЕТ НА ОТНОСИТЕЛЬНЫЕ АДРЕСА ПЕРЕМЕННЫХ, ДАННЫХ, ОБ'ЕКТНЫХ КОДОВ.

ДЛЯ ПЕРЕМЕННЫХ В РАЗДЕЛЕ **VAR** ЭТОТ АДРЕС УКАЗЫВАЕТ ОТНОСИТЕЛЬНОЕ СМЕЩЕНИЕ КАЖДОЙ ПЕРЕМЕННОЙ ОТ НАЧАЛА СТЕКА ТРАНСЛИРОВАННОЙ ПРОЦЕДУРЫ. В ТЕЛЕ ПРОЦЕДУРЫ (И ПРОГРАММЫ) АДРЕС - ЕСТЬ ОТНОСИТЕЛЬНЫЙ АДРЕС ОТ НАЧАЛА ПРОЦЕДУРЫ. ОТНОСИТЕЛЬНЫЕ АДРЕСА ИСПОЛЬЗУЮТСЯ В ССЫЛКАХ РАЗДЕЛА **ERROR MESSAGE**.

СЛЕДУЮЩАЯ ВЕРТИКАЛЬНАЯ КОЛОНКА СОСТОИТ ИЗ ДВУЗНАЧНЫХ ЧИСЕЛ - ИНДИКАТОРОВ ВЛОЖЕНИЙ ЦИКЛОВ И СОСТАВНЫХ ОПЕРАТОРОВ. КОГДА В ПРОГРАММЕ ВСТРЕЧАЕТСЯ ПЕРВЫЙ ОПЕРАТОР **BEGIN** , ТО В ЛЕВОМ РАЗДЕЛЕ СООТВЕТСТВУЮЩЕГО ДВУЗНАЧНОГО ЧИСЛА ЭТОЙ СТРОКИ ВЫДАЕТСЯ НОЛЬ.

КОГДА В КАКОЙ-ЛИБО СТРОКЕ ПОЯВЛЯЕТСЯ **END** , СООТВЕТСТВУЮЩИЙ ЭТОМУ **BEGIN** , ТО В ПРАВОМ РАЗРЯДЕ ДВУЗНАЧНОГО ЧИСЛА ПОЯВЛЯЕТСЯ НОЛЬ.

ЕСЛИ ВНУТРИ ОДНОЙ КОНСТРУКЦИИ ПОЯВЛЯЕТСЯ ВТОРАЯ (**BEGIN, CASE, REPEAT**) , ТО В ЛЕВОМ РАЗРЯДЕ ВЫДАЕТСЯ 1 , А ДЛЯ СООТВЕТСТВУЮЩЕГО **END** (**UNTIL**) В ПРАВОМ РАЗДЕЛЕ ПОЯВЛЯЕТСЯ 1 .

ПРАВИЛЬНО СОСТАВЛЕННАЯ ПРОГРАММА ДОЛЖНА НАЧИНАТЬСЯ НУЛЕМ В ЛЕВОМ РАЗРЯДЕ ИНДИКАТОРА ВЛОЖЕНИЯ У ПЕРВОГО ОПЕРАТОРА **BEGIN** ПРОГРАММЫ И КОНЧАТЬСЯ НУЛЕМ В ПРАВОМ РАЗРЯДЕ У ПОСЛЕДНЕГО ОПЕРАТОРА **END**.

ПРИМЕР.

```
1 0680  --  PROGRAM L10T(INPUT, OUTPUT);
2 06A8  --  VAR CH:CHAR;
3 0000  0-  BEGIN
4 0062  --    WHILE NOT EOF DO
5 008A  10      BEGIN WRITE(' ');
6 0002  --    WHILE NOT EOLN DO
7 00EE  22      BEGIN READ(CH); WRITE(CH) END;
8 016C  -1    WRITELN; READLN END
9 0184  -0  END.
```

16.2 ОШИБКИ ТРАНСЛЯЦИИ.

РАССМОТРИМ ПРИМЕР

```
1 0680  --  PROGRAM L6T1(INPUT, OUTPUT),
  ***ERROR***                                  ! 14,18
2 06A8  --  VAR A,B,R;REAL; I:INTEGER;
3 0000  0-    BEGIN
    .           .
    .           .
    .           .
11 0196 -0    END.
```

ОШИБКИ, ОБНАРУЖЕННЫЕ ПРИ ТРАНСЛЯЦИИ, ОТМЕЧАЮТСЯ В ТЕКСТЕ ПРОГРАММЫ СИМВОЛОМ " ! " И НОМЕРОМ, ВЫДАВАЕМЫМ В СЛЕДУЮЩЕЙ ЗА ОШИБОЧНОЙ СТРОКЕ.

НИЖЕ ТЕКСТА ПРОГРАММЫ ПЕЧАТАЕТСЯ РАСШИФРОВКА ВСЕХ НАЙДЕННЫХ ОШИБОК. (ERROR MESSAGE).
ТАК В ПРИВЕДЕННОМ ПРИМЕРЕ БУДЕТ НАПЕЧАТАНО.

14: ПРОПУЩЕНА " ; " (ВОЗМОЖНО, В ПРЕДЫДУЩЕЙ СТРОКЕ).

18: ОШИБКА В ДЕКЛАРАТИВНОЙ ЧАСТИ.

ПОСЛЕ ТОГО, КАК ТРАНСЛЯТОР ОБНАРУЖИЛ ОШИБКУ, ОН ПОПЫТАЕТСЯ ВОЗОБНОВИТЬ АНАЛИЗ ПРОГРАММЫ,ПРОПУСТИВ ЧАСТЬ ТЕКСТА ДО ОЖИДАЕМОГО СИМВОЛА. ЧАСТО УДАЕТСЯ УСПЕШНО ПРОДОЛЖИТЬ ТРАНСЛЯЦИЮ, А ИНОГДА ЭТО МОЖЕТ ПРИВЕСТИ К НАВЕДЕННЫМ ОШИБКАМ.
ПРОГРАММА, СОДЕРЖАЩАЯ ФАТАЛЬНЫЕ ОШИБКИ ТРАНСЛЯЦИИ, НА СЧЕТ НЕ ВЫХОДИТ.

16.3 КОДЫ ЗАВЕРШЕНИЯ ТРАНСЛЯЦИИ.

0 - ТРАНСЛЯЦИЯ ЗАКОНЧЕНА, ОШИБОК НЕТ.
4 - ВЫДАНА ПРЕДУПРЕДИТЕЛЬНАЯ ДИАГНОСТИКА, ФАТАЛЬНЫХ ОШИБОК НЕТ, ПРОГРАММА МОЖЕТ БЫТЬ ВЫПОЛНЕНА.
8 - БЫЛИ ФАТАЛЬНЫЕ ОШИБКИ, ПРОГРАММА НЕ ВЫПОЛНЯЕТСЯ.
12 - ОШИБКА СИСТЕМЫ.
16 - ОШИБКА ТРАНСЛЯТОРА.

ОШИБКИ 12 И 16 МОГУТ БЫТЬ СЛЕДСТВИЕМ ОШИБОК В УПРАВЛЯЮЩИХ КАРТАХ (НАПРИМЕР, МАЛО ЗАКАЗАНО ПАМЯТИ, НЕ ЗАКАЗАН СООТВЕТСТВУЮЩИЙ ТРАНСЛЯТОР И Т.Д.)

ЕСЛИ УПРАВЛЯЮЩИЕ КАРТЫ ПРАВИЛЬНЫЕ, ТО ПРИ КОДАХ 12 И 16 НЕОБХОДИМО ОБРАТИТЬСЯ К КОНСУЛЬТАНТУ.

16.4 ВНЕШНИЕ ПРОЦЕДУРЫ.

ПРОЦЕДУРЫ МОГУТ БЫТЬ УЖЕ В ВИДЕ МОДУЛЕЙ ЗАГРУЗКИ, СОЗДАННЫХ ТРАНСЛЯТОРАМИ КАК С **ПАСКАЛЯ**, ТАК И **ФОРТРАНА, И АССЕМБЛЕРА.**
ТАКИЕ ПРОЦЕДУРЫ ДОЛЖНЫ БЫТЬ ДЕКЛАРИРОВАНЫ В ВЫЗЫВАЮЩЕЙ ПАСКАЛЬ-ПРОГРАММЕ СПЕЦИАЛЬНЫМ ОБРАЗОМ.

А) ПАСКАЛЬ-ПРОЦЕДУРА (PROC)

```
PROCEDURE PROC( ... ); EXTERN;
```

ИМЯ СПИСОК ФОРМ.

Б) ФОРТРАННАЯ ПОДПРОГРАММА (SUB, НАПРИМЕР)

```
PROCEDURE SUB( ... ); FORTRAN;
```

ИМЯ СПИСОК ПАРАМ. ФОРМ.

В) ФОРТРАННАЯ ПОДПРОГРАММА-ФУНКЦИЯ (F)

```
FUNCTION F( ... ): TYPEF ; FORTRAN;
```

имя — список форм. парам. — тип результата

Г) ПАСКАЛЬ-ФУНКЦИЯ (FUNPSCL)

```
FUNCTION FUNPSCL(P1:T1,...): TYPEFUN; EXTERN
```

имя список форм. параметров.

ВНЕШНИЕ ПРОЦЕДУРЫ И ФУНКЦИИ НЕ МОГУТ ИМЕТЬ В КАЧЕСТВЕ ПАРАМЕТРОВ ПАРАМЕТРЫ-ФУНКЦИИ И САМИ НЕ МОГУТ БЫТЬ ПЕРЕДАНЫ В КАЧЕСТВЕ ПАРАМЕТРОВ.

ВНЕШНЯЯ ПРОЦЕДУРА МОЖЕТ ИСПОЛЬЗОВАТЬСЯ РЕКУРСИВНО И ПЕРЕДАНА КАК ПАРАМЕТР ТОЛЬКО ПРОЦЕДУРЕ, ДЕКЛАРИРОВАННОЙ ВНУТРИ НЕЕ САМОЙ.

16.5 ТРАНСЛЯЦИЯ ВНЕШНИХ ПАСКАЛЬ-ПРОЦЕДУР.

ВНЕШНИЕ ПРОЦЕДУРЫ И ФУНКЦИИ ТРАНСЛИРУЮТСЯ ТОЛЬКО В РЕЖИМЕ E+ . НЕОБХОДИМО СЛЕДОВАТЬ ТАКИМ ПРАВИЛАМ.

1. РЕЖИМ E+ ДОЛЖЕН БЫТЬ УСТАНОВЛЕН ДО ДЕКЛАРАЦИИ ВНЕШНЕЙ ПРОЦЕДУРЫ.
   ПОСЛЕ НЕЕ НЕ ДОЛЖНЫ ТРАНСЛИРОВАТЬСЯ ВНУТРЕННИЕ ПРОЦЕДУРЫ И ГЛАВНАЯ ПРОГРАММА.
2. ДОПУСКАЕТСЯ ТОЛЬКО ДВА УРОВНЯ ВЛОЖЕНИЯ ВНЕШНИХ ПРОЦЕДУР. ОДНАКО САМА ВНЕШНЯЯ ПРОЦЕДУРА МОЖЕТ ИМЕТЬ НЕСКОЛЬКО УРОВНЕЙ ВЛОЖЕНИЯ ВНУТРЕННИХ (ДЛЯ НЕЕ) ПРОЦЕДУР.
3. ЕСЛИ ВО ВНЕШНЕЙ ПРОЦЕДУРЕ ОПИСЫВАЮТСЯ ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ, КОНСТАНТЫ ИЛИ ТИПЫ, ТО ОНИ ДОЛЖНЫ В ТОЧНОСТИ СОВПАДАТЬ ПО ОПИСАНИЮ С СООТВЕТСТВУЮЩИМИ ПЕРЕМЕННЫМИ, КОНСТАНТАМИ, ТИПАМИ В **PROGRAM** .

ЕСЛИ ВНЕШНЯЯ ПРОЦЕДУРА СБРАЩАЕТСЯ К ГЛОБАЛЬНОЙ ПЕРЕМЕННОЙ, А ФАЙЛЫ **INPUT И OUTPUT** ЯВЛЯЮТСЯ ПАРАМЕТРАМИ ГЛАВНОЙ ПРОГРАММЫ, ТО ЭТИ ЖЕ ФАЙЛЫ И В ТОМ ЖЕ ПОРЯДКЕ, ЧТО В ГЛАВНОЙ, НАДО УКАЗАТЬ В ТОЙ **PROGRAM** , КОТОРАЯ ИСПОЛЬЗУЕТСЯ ДЛЯ ТРАНСЛЯЦИИ ДАННОЙ ВНЕШНЕЙ ПРОЦЕДУРЫ. ЕСЛИ НЕ ВЫПОЛНЯЕТСЯ ЭТО ПРАВИЛО, ТО ПОСЛЕДСТВИЯ НЕПРЕДСКАЗУЕМЫ.

4. ПРИ ТРАНСЛЯЦИИ ВНЕШНИХ ПРОЦЕДУР МОЖЕТ ОТСУТСТВОВАТЬ ГЛАВНАЯ ПРОГРАММА (PROGRAM)

ЕСЛИ ОДНАЖДЫ УСТАНОВЛЕН РЕЖИМ E+ , ТО НЕЛЬЗЯ ЕГО УСТАНАВЛИВАТЬ ПОВТОРНО.

ПРИМЕР.

КОГДА ПАСКАЛЬ-ПРОГРАММА ВЫЗЫВАЕТ ФОРТРАННУЮ ПОДПРОГРАММУ ИЛИ ФУНКЦИЮ, НАДО УЧИТЫВАТЬ СЛЕДУЮЩЕЕ.

1. ТИП **REAL** СООТВЕТСТВУЕТ ФОРТРАННОМУ **DOUBLE PRECISION** .
2. ТИП **BOOLEAN** СООТВЕТСТВУЕТ ФОРТРАННОМУ **LOGICAL** .
3. МАССИВЫ ПАСКАЛЯ РАСПОЛАГАЮТСЯ ПО СТРОКАМ, А МАССИВЫ ФОРТРАНА - ПО СТОЛБЦАМ (ЕСЛИ НЕ ИСПОЛЬЗОВАН НЕЯВНЫЙ ЦИКЛ **DO**).

ПРИ ПЕРЕДАЧЕ ДВУМЕРНОГО МАССИВА В КАЧЕСТВЕ ПАРАМЕТРА ИЗ ПАСКАЛЬ - ПРОЦЕДУРЫ В ФОРТРАННУЮ (И НАОБОРОТ) МАССИВ НЕОБХОДИМО ТРАНСПОНИРОВАТЬ.

16.6 ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ.

ТИП	ЧИСЛО БАЙТ	ВЕЛИЧИНА
INTEGER	4	32 БИТА
BOOLEAN	4	1=**TRUE**;0=**FALSE**.
CHAR	4	**EBCDIC** КОД В МЛАДШЕМ (ПРАВОМ) БАЙТЕ.
REAL	8	ЧИСЛО В ФОРМЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ
SET	8	ЭЛЕМЕНТЫ, ПРЕДСТАВЛЯЕМЫЕ ПОБИТНО, НАЧИНАЯ СЛЕВА.
SCALAR	4	ПОРЯДКОВЫЙ НОМЕР ВЕЛИЧИНЫ В ТИПЕ, НАЧИНАЯ С НУЛЯ.

В УПАКОВАННЫХ СТРУКТУРАХ СКАЛЯРНЫЕ ТИПЫ ЗАНИМАЮТ ЧАЩЕ ВСЕГО ОДИН **БАЙТ**, ЕСЛИ ПОРЯДКОВЫЙ НОМЕР УПАКОВАННЫХ ЭЛЕМЕНТОВ ЛЕЖИТ В 0..255.

ПРИМЕР.

```
X:INTEGER (ЗАНИМАЕТ 4 БАЙТА)
A:PACKED ARRAY [1..4] OF CHAR (ЗАНИМАЕТ 4 БАЙТА)
B:ARRAY[1..4] OF CHAR (ЗАНИМАЕТ 16 БАЙТ)
C:REAL (ЗАНИМАЕТ 8 БАЙТ)
```

17. ДИАГНОСТИКА ОШИБОК

```
1: ОШИБКА В ПРОСТОМ ТИПЕ
2: ПРОПУЩЕН ИДЕНТИФИКАТОР
3: ПРОПУЩЕН ОПЕРАТОР PROGRAM
4: ПРОПУЩЕНА ' ) '
5: ПРОПУЩЕНО ' : '
```

```
  6:  НЕЗАКОННЫЙ СИМВОЛ
  7:  ОШИБКА В ПАРАМЕТРАХ
  8:  ПРОПУЩЕНО OF
  9:  ПРОПУЩЕНА ' ( '
 10:  ОШИБКА В ТИПЕ
 11:  ПРОПУЩЕНА ' [ '
 12:  ПРОПУЩЕНА ' ] '
 13:  ПРОПУЩЕН END
 14:  ПРОПУЩЕНА ' ; ' (ВОЗМОЖНО,СТРОКОЙ ВЫШЕ).
 15:  ДОЛЖНО БЫТЬ INTEGER
 16:  ПРОПУЩЕНО ' =' 
 17:  ПРОПУЩЕН BEGIN
 18:  ОШИБКА В РАЗДЕЛЕ ОПИСАНИЙ
 19:  ОШИБКА В СПИСКЕ ПОЛЕЙ
 20:  ПРОПУЩЕНА ' , '
 21:  ПРОПУЩЕНА ' * '
 50:  ОШИБКА В КОНСТАНТЕ
 51:  ПРОПУЩЕН ЗНАК := '
 52:  ПРОПУЩЕНО THEN
 53:  ПРОПУЩЕНО UNTIL
 54:  ПРОПУЩЕНО DO
 55:  ПРОПУЩЕНО TO ИЛИ DOWNTO
 56:  ПРОПУЩЕНО IF
 57:  ПРОПУЩЕНО СЛОВО FILE
 58:  ОШИБКА В МНОЖИТЕЛЕ
 59:  ОШИБКА В ПЕРЕМЕННОЙ
101:  ДВАЖДЫ ОПИСАННЫЙ ИДЕНТИФИКАТОР
102:  НИЖНЯЯ ГРАНИЦА БОЛЬЦЕ ВЕРХНЕЙ
103:  ИДЕНТИФИКАТОР НЕ ТОГО КЛАССА
104:  НЕОПИСАННЫЙ ИДЕНТИФИКАТОР
105:  ЗДЕСЬ ЗНАК НЕ ДОПУСКАЕТСЯ
106:  ПРОПУЩЕНО ЧИСЛО
107:  НЕСОВМЕСТИМЫЕ SUBRANGE ТИПЫ.
108:  ЗДЕСЬ FILE НЕ ДОПУСКАЕТСЯ.
109:  ЗДЕСЬ ТИП НЕ МОЖЕТ БЫТЬ REAL
110:  ТИП ПЕРЕКЛЮЧАТЕЛЯ ДОЛЖЕН БЫТЬ СКАЛЯРНЫМ.
111:  ТИП НЕСОВМЕСТИМ С ТИПОМ ПЕРЕКЛЮЧАТЕЛЯ
112:  ТИП ИНДЕКСА НЕ МОЖЕТ БЫТЬ REAL
113:  ИНДЕКС ДОЛЖЕН ИМЕТЬ СКАЛЯРНЫЙ ТИП
114:  БАЗОВЫМ ТИПОМ НЕ МОЖЕТ БЫТЬ REAL
115:  БАЗОВЫМ ТИПОМ ДОЛЖЕН БЫТЬ СКАЛЯРНЫЙ ТИП.
116:  ОШИБКА В ТИПЕ ПАРАМЕТРОВ СТАНДАРТНОЙ ПРОЦЕДУРЫ.
117:  НЕДОПУСТИМАЯ ССЫЛКА НА ЕЩЕ НЕ ОПИСАННОЕ ПОНЯТИЕ.
118:  НЕОПИСАННЫЙ  ТИП ИСПОЛЬЗУЕТСЯ ПРИ ОПИСАНИИ ПЕРЕМЕННОЙ.
119:  ПОВТОРНОЕ  ОПИСАНИЕ СПИСКА ПАРАМЕТРОВ НЕ ДОПУСКАЕТСЯ.
120:  ТИП  РЕЗУЛЬТАТА  ФУНКЦИИ  МОЖЕТ БЫТЬ СКАЛЯРНЫМ ЛИБО POINTER.
121:  ПАРАМЕТР-ЗНАЧЕНИЕ НЕ МОЖЕТ БЫТЬ ФАЙЛОМ.
122:  ФУНКЦИЯ УЖЕ ДЕКЛАРИРОВАНА (FORWARD) ; НЕ ДОПУСКАЕТСЯ ПОВТОРНОЕ ОПИСАНИЕ ТИПА РЕЗУЛЬТАТА ФУНКЦИ.
123:  ПРОПУЩЕН ТИП РЕЗУЛЬТАТА В ОПИСАНИИ ФУНКЦИИ.
124:  ФОРМАТ F ДОПУСТИМ ТОЛЬКО ДЛЯ REAL.
125:  ОШИБКА В ТИПЕ ПАРАМЕТРА СТАНДАРТНОЙ ФУНКЦИИ.
126:  ЧИСЛО ПАРАМЕТРОВ ИНСЕ,ЧЕМ В ОПИСАНИИ ФУНКЦИИ.
127:  НЕДОПУСТИМЫЕ ФАКТИЧЕСКИЕ ПАРАМЕТРЫ.
128:  ТИП  РЕЗУЛЬТАТА  ПАРАМЕТРА-ФУНКЦИИ НЕ СООТВЕТСТВУЕТ ОПИСАНИЮ.
129:  НЕСООТВЕТСТВИЕ ТИПОВ ОПЕРАНДОВ.
```

130: ТИП ВЫРАЖЕНИЯ - НЕ **SET**.
131: ДОПУСКАЕТСЯ ПРОВЕРКА ТОЛЬКО НА РАВЕНСТВО.
132: НЕ ДОПУСКАЕТСЯ СТРОГОЕ ВКЛЮЧЕНИЕ.
133: НЕ ДОПУСКАЕТСЯ СРАВНЕНИЕ ФАЙЛОВ.
134: НЕЗАКОННЫЙ ТИП ОПЕРАНДА.
135: ОПЕРАНД ДОЛЖЕН ИМЕТЬ ТИП **BOOLEAN**.
136: ТИП ЭЛЕМЕНТА МНОЖЕСТВА ДОЛЖЕН БЫТЬ СКАЛЯРНЫМ.
137: НЕСОВМЕСТИМЫЕ ТИПЫ ЭЛЕМЕНТОВ МНОЖЕСТВА.
138: ТИП ПЕРЕМЕННОЙ - НЕ МАССИВ.
139: ТИП ИНДЕКСА НЕ СООТВЕТСТВУЕТ ОПИСАНИЮ.
140: ТИП ПЕРЕМЕННОЙ - НЕ **RECORD**.
141: ТИП ПЕРЕМЕННОЙ ДОЛЖЕН БЫТЬ **FILE** ЛИБО **POINTER**.
142: НЕДОПУСТИМЫЕ ФАКТИЧЕСКИЕ ПАРАМЕТРЫ.
143: НЕДОПУСТИМЫЙ ТИП ПАРАМЕТРА ЦИКЛА.
144: НЕДОПУСТИМЫЙ ТИП ВЫРАЖЕНИЯ.
145: НЕСООТВЕТСТВИЕ ТИПОВ.
146: НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ ФАЙЛЫ В ОПЕРАТОРЕ ПРИСВАИВАНИЯ
147: ТИП МЕТКИ НЕ СООТВЕТСТВУЕТ ТИПУ ЗНАЧЕНИЯ ПЕРЕКЛЮЧАТЕЛЯ.
148: ГРАНИЦЫ ДИАПАЗОНА ДОЛЖНЫ ИМЕТЬ СКАЛЯРНЫЙ ТИП.
149: ИНДЕКС МОЖЕТ ИМЕТЬ ТИП **SUBRANGE**, НО НЕ **INTEGER**.
150: НЕЛЬЗЯ ПРИСВАИВАТЬ ЗНАЧЕНИЯ СТАНДАРТНЫМ ФУНКЦИЯМ.
151: НЕЛЬЗЯ ПРИСВАИВАТЬ ЗНАЧЕНИЕ ФОРМАЛЬНОМУ ПАРАМЕТРУ - ФУНКЦИИ.
152: В ДАННОЙ ЗАПИСИ НЕТ ТАКОГО ПОЛЯ.
153: ОШИБКА В ТИПЕ ПАРАМЕТРА **READ**.
154: ФАКТИЧЕСКИЙ ПАРАМЕТР ДОЛЖЕН БЫТЬ ПЕРЕМЕННОЙ.
155: ПАРАМЕТР ЦИКЛА НЕ МОЖЕТ БЫТЬ ФОРМАЛЬНЫМ ПАРАМЕТРОМ.
156: ОДИНАКОВЫЕ МЕТКИ В **CASE**.
157: СЛИШКОМ МНОГО ОПЕРАТОРОВ **CASE** ВЛОЖЕНО В ДАННЫЙ.
158: ПРОПУЩЕНО ОПИСАНИЕ СООТВЕТСТВУЮЩЕГО ВАРИАНТА.
159: ТИПЫ **REAL** И **ALFA** НЕДОПУСТИМЫ ДЛЯ ПЕРЕКЛЮЧАТЕЛЯ.
160: ПРЕДЫДУЩЕЕ ОПИСАНИЕ НЕ БЫЛО **FORWARD**.
161: ОПЯТЬ ОПИСАНО **FORWARD**.
162: РАЗМЕР ПАМЯТИ,ЗАНИМАЕМЫЙ ПАРАМЕТРОМ,ДОЛЖЕН БЫТЬ КОНСТАНТОЙ.
163: ПРОПУЩЕН ВАРИАНТ В ОПИСАНИИ.
164: НЕ ДОПУСКАЕТСЯ ПОДСТАНОВКА СТАНДАРТНОЙ ПРОЦЕДУРЫ (ФУНКЦИИ).
165: МЕТКИ НЕ ДОЛЖНЫ ПОВТОРЯТСЯ.
166: ДВАЖДЫ ОПИСАННАЯ МЕТКА.
167: НЕОПИСАННАЯ МЕТКА.
168: НЕНАЙДЕННАЯ МЕТКА.
169: ОШИБКА В БАЗОВОМ ТИПЕ **SET**.
170: ДОЛЖЕН БЫТЬ ПАРАМЕТР - ЗНАЧЕНИЕ.
171: СТАНДАРТНЫЙ ФАЙЛ НЕ ТРЕБУЕТ ОПИСАНИЯ.
172: НЕОПИСАННЫЙ ВНЕШНИЙ ФАЙЛ.
173: ДОЛЖНА БЫТЬ ФОРТРАННАЯ ПРОЦЕДУРА ИЛИ ФУНКЦИЯ.
174: ДОЛЖНА БЫТЬ ПАСКАЛЬ - ПРОЦЕДУРА ИЛИ ФУНКЦИЯ.
175: В ЗАГОЛОВКЕ **PROGRAM** ПРОПУЩЕН ФАЙЛ **INPUT** .
176: В ЗАГОЛОВКЕ **PROGRAM** ПРОПУЩЕН ФАЙЛ **OUTPUT** .
201: ОТСУТСТВУЕТ ТОЧКА В ВЕЩЕСТВЕННОЙ КОНСТАНТЕ.
202: КОНСТАНТА ТИПА **ALFA** НЕ ДОЛЖНА ВЫХОДИТЬ ЗА ПРЕДЕЛЫ СТРОКИ.
203: СЛИШКОМ БОЛЬШАЯ ЦЕЛАЯ КОНСТАНТА.
204: В ВОСЬМЕРИЧНОМ ЧИСЛЕ НЕ МОЖЕТ БЫТЬ ЦИФР 8 И 9 .
250: СЛИШКОМ МНОГО УРОВНЕЙ ВЛОЖЕНИЯ ОБЛАСТЕЙ ДЕЙСТВИЙ ИДЕНТИФИКАТОРОВ.
251: СЛИШКОМ МНОГО ВЛОЖЕННЫХ ПРОЦЕДУР И (ИЛИ) ФУНКЦИЙ.
252: СЛИШКОМ МНОГО **FORWARD** .

253: СЛИШКОМ ДЛИННАЯ ПРОЦЕДУРА.
254: СЛИШКОМ МНОГО ДЛИННЫХ КОНСТАНТ В ЭТОЙ ПРОЦЕДУРЕ.
255: СЛИШКОМ МНОГО ОШИБОК В ЭТОЙ СТРОКЕ.
256: СЛИШКОМ МНОГО ВНЕШНИХ ССЫЛОК.
257: СЛИШКОМ МНОГО EXTERNAL .
258: СЛИШКОМ МНОГО ЛОКАЛЬНЫХ ФАЙЛОВ.
259: СЛИШКОМ СЛОЖНОЕ ВЫРАЖЕНИЕ.
300: ДЕЛЕНИЕ НА НОЛЬ.
301: НЕТ ПОЛЯ CASE ДЛЯ ДАННОГО ЗНАЧЕНИЯ.
302: ЗНАЧЕНИЕ ИНДЕКСА ВЫШЛО ИЗ ДИАПАЗОНА.
303: ПРИСВАИВАЕМОЕ ЗНАЧЕНИЕ ВЫШЛО ИЗ ДИАПАЗОНА.
304: ЗНАЧЕНИЕ ЭЛЕМЕНТА МНОЖЕСТВА ВЫШЛО ИЗ ДИАПАЗОНА.
398: ОГРАНИЧЕНИЕ, НАКЛАДЫВАЕМОЕ РЕАЛИЗАЦИЕЙ СИСТЕМЫ.
399: МАССИВЫ ПЕРЕМЕННОЙ РАЗМЕРНОСТИ НЕ РЕАЛИЗОВАНЫ СИСТЕМОЙ.

ЛИТЕРАТУРА

1. K.JENSEN, N.WIRTH. PASCAL. USER MANUAL AND REPORT, SPRINGER-VERLAG NEW-YORK HEIDELBERG BERLIN 1978.

2. К. ЙЕНСЕН, Н. ВИРТ. ПАСКАЛЬ М., "ФИНАНСЫ И СТАТИСТИКА". 1982Г.

3. ПИРИН С.И. ЯЗЫК ПАСКАЛЬ-МОНИТОР И ЕГО ИСПОЛЬЗОВАНИЕ. ИЗД-ВО ВЦ АН СССР, М., 1978

4. ПИРИН С.И. О РЕАЛИЗАЦИИ КОМПИЛЯТОРА ПАСКАЛЬ НА ЭВМ БЭСМ-6. В КН. "ОБРАБОТКА СИМВОЛЬНОЙ ИНФОРМАЦИИ." ИЗД-ВО ВЦ АН СССР, М ., 1978, ВЫП.4.

5. PASCAL 8000. REFERENCE MANUAL. VERSION 2.0

6. WELSH J., ELDER J. INTRODUCION TO PASCAL. QUEEN'S UNIVERSITY OF BELFAST, NORTHERN IRELAND, PRENTICE-HALL INTERNATICNAL SERIES IN COMPUTER SCIENCE, LONDON, 1979.

УВАЖАЕМЫЕ ТОВАРИЩИ ПОЛЬЗОВАТЕЛИ!

СО ВСЕМИ ПОЖЕЛАНИЯМИ И ЗАМЕЧАНИЯМИ ПО ДАННОМУ РУКОВОДСТВУ ОБРАЩАЙТЕСЬ ПО ТЕЛ. 63-746 Г.Л.СЕМАШКО.
В ДАЛЬНЕЙШИХ РЕДАКЦИЯХ ВАШИ ПРЕДЛОЖЕНИЯ БУДУТ УЧТЕНЫ.

СЕМАШКО Г.Л.