21 Язык программирования PASCAL

21.1 Введение

PASCAL программа представляет из себя список PASCAL операторов.

Каждый оператор заканчивается двоеточием-разделителем (:). Имена, использующиеся в операторах (переменные, константы, ключевые слова), разделяются неактивными разделителями (символ пробела, EOL, табуляция), а также специальными символами (например, > "больше чем").

Комментарии могут быть включены в текст и заключены в звездочки. Структура программ языка PASCAL следующая.

- Программа должна начинаться с ключевого слова PROGRAM за которым следует имя программы, и заканчиваться ключевым словом End.
- Объявление функции должен начинаться с ключевого слова FUNCTION и заканчиваться ключевым словом END.
- Объявление процедуры должно начинаться с ключевого слова PROCEDURE и заканчиваться ключевым словом END.
- Группа операторов, начинающаяся с ключевого слова BEGIN и заканчивающаяся ключевым словом END рассматривается как один оператор.
- Блок объявления типов данных должен начинаться с ключевого слова VAR.

Пример.

```
program SumByCall;
function Summa(var N:integer):integer;
var
   I : integer;
begin
   if N=0 then
      Result:=0
   else begin
      I:=N-1;
      Result:=N+Summa(I);
      end;{else}
end;

procedure Main;
begin
   Res:=Summa(Inp)+Summa(Inp);
end;
;
```

21.2 Типы операторов

Имеются следующие типы PASCAL операторов:

- операторы воздействия на переменную (переменная:= выражение:);
- условные операторы (IF, THEN, ELSE);
- итерационные операторы (FOR, WHILE ...);
- управляющие операторы;
- специальные операторы для связи с другими языками.

21.3 Выражения

PASCAL выражения является комбинацией PASCAL операторов и переменных (или констант) операндов. Круглые скобки используют для выделения части выражения и задания определенного порядка выполнения операций.
Например:
`2+3*6` `дает значение 20`
`(2+3)*6` `дает значение 30`

Для каждого одиночного выражения (комбинации операндов с одним PASCAL оператором) все операнды должны иметь один тип.
Например:
`(boo_var1 AND boo_var2)` имеет тип `boolean`
`not(boo_var1)` имеет тип `boolean`
`(sin(3.14) + 0.72)` имеет тип `Float`
`(1s23 + 1.78)` является недействительным (ошибочным) выражением.

21.4 Список операций языка PASCAL в порядке убывания приоритета

№ Операция	Имя
1. Взятие адреса	@
2. Унарный минус	-
3. Унарный плюс	+
4. Порязрядное НЕ	**NOT**
5. Логическое НЕ	**NOT**
6.Умножение	*
7.Деление	/
8.Целочисленное деление	**DIV**
9.Остаток от деления	**MOD**
10.Логическое "И"	**AND**
11.Поразрядное "И"	**AND**
12.Цикл. сдвиг вправо	**SHR**
13.Цикл. сдвиг влево	**SHL**
14.Сложение	+
15.Вычитание	-
16.Логическое "ИЛИ"	**OR**
17.Поразрядное "ИЛИ"OR	
18.Логическое исключающее "ИЛИ"	**XOR**
19.Вхождение во множество	**IN**
20.Больше	>
21.Меньше	<
22.Равно	=
23.Не равно	<>
24.Больше или равно	>=
25.Меньше или равно	<=

21.5 Управляющие конструкции языка программирования PASCAL

`ASSIGN`	присвоить
`IF/THEN/ELSE`	если/тогда /иначе
`WHILE`	пока
`REPEAT`	повторить
`FOR`	до того как

CASE	поливариантное ветвление
EXIT	выход из процедуры
BREAK	выход их цикла
CONTINUE	начать новую итерацию цикла

21.5.1.1 Оператор ASSIGN

Имя: :=
Назначение: присвоить переменной значение выражения
Синтаксис: <переменная>:= <выражение>;

Операнды: Переменная и выражение должны иметь одинаковый тип.
Пример

```
(*переменная  -переменная *)
bo23 := bo10;
(*переменная -выражение *)
bo23 := NOT (bo10);
bo56 :=b34 OR alarm100 & (level >= over_value);
```

21.5.1.2 Оператор EXIT

Имя: EXIT
Назначение: безусловный выход из процедуры
Синтаксис: EXIT

21.5.1.3 Оператор BREAK

Имя: BREAK
Назначение: Позволяет досрочно закончить цикл
Синтаксис: BREAK

21.5.1.4 Оператор CONTINUE

Имя: CONTINUE
Назначение: Позволяет начать новую итерацию цикла, даже если предыдущая итерация не была закончена.
Синтаксис: CONTINUE

21.5.1.5 Оператор IF... THEN ... ELSE

Имя: IF...THEN...ELSE
Назначение: выполняет <оператор1>, если <двоичное_выражение> = TRUE, иначе <оператор2>. В качестве оператора может быть список операторов, начинающийся с ключевого слова BEGIN и заканчивающийся ключевым словом END.
Синтаксис:

```
IF <двоичное_выражение> THEN
          <оператор1>;
          ELSE
          <оператор2>;
```

Пример

```
     IF manual AND NOT(alarm) THEN
          Level := manual_level;
     ELSE
          Level := (lv100*100) / scale;
```

Примечание. Оператор ELSE является дополнительным. Если оператор ELSE отсутствует, тогда отсутствуют инструкции, соответствующие условию FALSE.

21.5.1.6 Оператор WHILE

Имя: WHILE...DO

Назначение: выполняет <оператор> если <двоичное_выражение> = TRUE. Условие продолжения проверяется до выполнения. В качестве оператора может быть список операторов, начинающийся с ключевого слова BEGIN и заканчивающийся ключевым словом END.

Синтаксис:

```
WHILE <двоичное_выражение> DO
          <оператор>;
```

Пример

```
J := 1;
WHILE J <= 100 & words[j] <> 'KEY' DO
  J := J +1;
```

Примечание. Не рекомендуется использовать в операторе WHILE входные переменные, т.к. программа считывает их один раз в цикле и не обновляет их в течении действия оператора WHILE. Это может привести к зависанию программы.

21.5.1.7 Оператор REPEAT

Имя: REAPEAT...UNTIL

Назначение: выполняет <оператор> пока <двоичное_выражение> = TRUE. Условие продолжения проверяется после выполнения. В качестве оператора может быть список операторов, начинающийся с ключевого слова BEGIN и заканчивающийся ключевым словом END.

Синтаксис:

```
REPEAT
<оператор>;
UNTIL<двоичное_выражение>;
```

Примечание. Не рекомендуется использовать в операторе REPEAT входные переменные, т.к. программа считывает их один раз в цикле и не обновляет их в течении действия оператора REPEAT. Это может привести к зависанию программы.

21.5.1.8 Оператор FOR

Имя: FOR...TO(DOWNTO)...DO

Назначение: выполняет ограниченное число итераций <оператора>, используя как индекс перечисляемую переменную. В качестве оператора может быть список операторов, начинающийся с ключевого слова BEGIN и заканчивающийся ключевым словом END.

Синтаксис:

```
FOR <index> := <mini> TO (DOWNTO) <maxi> DO
     <оператор>;
```

Операнды:

Index	<порядковая_переменная> переменная типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, BYTE, TIMER, увеличивающаяся (уменьшающаяся для DOWNTO) в каждой итерации.
Mini	начальное значение индекса перед первой итерацией
maxi	максимально (минимально для DOWNTO) допустимое значение индекса

Пример:
```
J := 101 ;
FOR I := 1 TO 100 DO
  IF WORDS[I] = 'KEY' THEN
    J := I ;
EXIT;
```

21.5.1.9 Оператор CASE…OF

Имя: CASE…OF

Назначение: поливариантное ветвление по многим направлениям. Выполняет <оператор_i> в зависимости от значения <порядковой_переменной>. В качестве оператора может быть список операторов, начинающийся с ключевого слова BEGIN и заканчивающийся ключевым словом END.

Синтаксис:
```
CASE <порядковая_переменная> OF
     Значение_1:
          <оператор_1>;
     Значение_2:
          <оператор_2>;
     Значение_N:
          <оператор_N>;
     ELSE
          <оператор_else>;
     END;
```

Операнды:
<порядковая_переменная> типа INTEGER, SHORTINT, LONGINT, WORD, DWORD, BYTE, TIMER.

21.5.2 Управление таймером

TSTART	пуск таймера
TSTOP	останов таймера
TCONTINUE	продолжить счет таймера

21.5.2.1 Оператор TSTART

Имя: TSTART

Назначение: Очищает таймерную переменную (записывая в нее значение 0) и запускает счет времени

Синтаксис: TSTART (<таймерная переменная>);

Операнды: любая неактивная таймерная переменная

Пример
```
TSTART (T);
```

21.5.2.2 Оператор TSTOP

Имя: TSTOP

Назначение: Останавливает таймерную переменную. Значение переменной после останова не модифицируется.
Синтаксис: TSTOP (<таймерная переменная>);
Операнды: любая активная таймерная переменная

21.5.2.3 Оператор TCONTINUE

Имя: TCONTINUE
Назначение: Продолжает инкрементирование указанной таймерной переменной, счет которой был остановлен оператором TSTOP .
Синтаксис: TCONTINUE (<таймерная переменная>);
Операнды: любая неактивная таймерная переменная
Пример

```
TSTART (T);
TSTOP  (T);
TCONTINUE  (T);
```

21.5.3 Управление программами

GSTART включает программу в цикл контроллера
GSTOP исключает программу в цикл контроллера

21.5.3.1 Оператор GSTART

Имя: GSTART
Назначение: Включает указанную программу в цикл контроллера. Начиная с этого момента, программа исполняется в каждом цикле контроллера.
Синтаксис: GSTART (‘имя_программы’);
Операнды: любая неактивная программа

21.5.3.2 Оператор GSTOP

Имя: GSTOP
Назначение: Исключает указанную программу из цикла контроллера. Начиная с этого момента, программа не исполняется в цикле контроллера.
Синтаксис: GSTOP (‘имя_программы’);
Операнды: любая активная программа

Пример

```
GSTART (‘Sample‘);
GSTOP  (‘Sample‘);
```

21.5.4 Операторы форматного вывода

WRITE форматный вывод на терминал без перевода строки
WRITELN форматный вывод на терминал с переводом строки

Имя: WRITELN
Назначение: выполняет форматный вывод на терминал
Синтаксис:

```
WRITELN ('Элемент_1','Элемент_2','Элемент_N','Элемент_N+1 );
```

В качестве элементов могут быть:
- строка символов;
- переменная.

Формат вывода переменных:
<имя переменной>: число выводимых знаков
$ - вывод в шестнадцатеричном виде
Пример

```
writeln('B=$',B:h,' B1=',B1,' B2=',B2);          // boolean
writeln('SI=',SI,' SI:h=$',SI:h,' SI:2=',SI:2); // shortint
writeln('BY=',BY,' BY:h=$',BY:h,' BY:2=',BY:2); // byte
writeln('W=' ,W, ' W:h=$', W:h ,'  W:2=', W:2); // word
writeln('A=' ,A, ' A:h=$', A:h ,'  A:2=', A:2); // integer
writeln('DW=',DW,' DW:h=$',DW:h,' DW:2=',DW:2); // dword
writeln('LI=',LI,' LI:h=$',LI:h,' LI:2=',LI:2); // longint
```

```
Результат вывода
B=$00 B1=FALSE B2=FALSE
SI=0 SI:h=$00 SI:2=00
BY=0 BY:h=$00 BY:2=00
W=0 W:h=$0000  W:2=00
A=0 A:h=$0000  A:2=00
DW=0 DW:h=$00000000 DW:2=00
LI=0 LI:h=$00000000 LI:2=00
```

21.5.5 Математические функции

`Abs(x):` возвращает абсолютное значение числа X

`Cos(X):` возвращает косинус числа X, где X-угол в радианах

`Sin(X):` возвращает синус числа X, где X-угол в радианах

`Arctan(X):` возвращает арктангенс числа X, где X-угол в радианах

`Exp(X):` возвращает число, равное *e* в степени X

`Ln(X):` возвращает число, равное логарифму натуральному от числа X

`PI:` число ПИ

`Sqrt(X):` возвращает число, равное корню квадратному из X

`Trunc(X):` возвращает число, равное целой части числа X (округление происходит путем отбрасывания дробной части числа X, усеченное число имеет тип данных `LongInt`)

`Frac(X):` возвращает число, равное дробной части числа X

`Int(X):` возвращает число равное целой части числа X

`Round(X):` возвращает число равное целой части числа X (округление происходит по правилам математики, т.е. к ближайшему целому, округленное число имеет тип данных `LongInt`)