BNF i „odchylenia” od normy

Stosowane oznaczenia:

- [] - oznacza “opcjonalnie” czyli “zero lub jeden raz”
- ()* - oznacza “zero lub więcej razy”
- | - oznacza “lub”
- znaki i wyrazy pisane na niebiesko są częścią języka: słowa kluczowe itp.

```
<litera>          -> a | b ... y | z | A | B ... | Y | Z
<cyfra>           -> 0 | 1 ... 8 | 9
<dow_znak>        -> dowolny drukowalny znak ASCII z wyjątkiem znaku nr 39 “'” (apostrof);
<podkr>           -> _
<separator>       -> ;
<oper_przypis>    -> :=
<oper_arytm>      -> + | - | * | /
<oper_bool>       -> or | and
<oper_not>        -> not
<oper_porow>      -> > | < | >= | <= | =
<ident>           -> <litera> (<podkr> | <litera> | <cyfra>)*
<typ_zmiennej>    -> integer | boolean | real | string
<liczba_int>      -> (-)* <cyfra>(<cyfra>)*
<liczba_real>     -> (-)* <cyfra>(<cyfra>)*.<cyfra>(<cyfra>)*
<liczba>          -> <liczba_int> | <liczba_real>
<ident_bool>      -> zmienna o typie boolean
<ident_int>       -> zmienna o typie integer
<ident_real>      -> zmienna o typie real
<ident_str>       -> zmienna o typie integer
<zmienna>         -> <ident_bool> | <ident_int> | <ident_real> | <ident_str>
<stala_bool>      -> true | false
<stala_string>    -> '<dow_znak>'
<wart_bool>       -> <stala_bool> | <ident_bool>
```

```
<program>          -> <naglowek_prog> [<deklar_zmienn>] <cialo_prog>
<naglowek_prog>    -> program <ident> <separator>
<deklar_zmienn>    -> var <ident> (, <ident>) : <typ_zmiennej> ; (<ident> (, <ident>) : <typ_zmiennej> ;)*
<cialo_prog>       -> begin (<instrukcja>;)* end.
<porownanie>       -> <liczba> <oper_porow> <liczba>
<instrukcja>       -> <instr_prosta> | <instr_if> | <instr_while>
<instr_blokowa>    -> begin (<instrukcja>;)* end
<instr_prosta>     -> <instr_przypis> | <instr_write> | <instr_read>
<instr_przypis>    -> <przypis_bool> | <przypis_int> | <przypis_real>
<przypis_bool>     -> <ident_bool> <oper_przypis> <dzialanie_bool>
<przypis_int>      -> <ident_int> <oper_przypis> <dzialanie_arytm_int>
<przypis_real>     -> <ident_real> <oper_przypis> <dzialanie_arytm_real>
<dzialanie_arytm_real>, <dzialanie_arytm_int> - wyglądają jak zwykłe działania arytmetyczne jakie można tworzyć przy pomocy
<oper_arytm> oraz nawiasów aby wymusić określoną kolejność działań; dozwolone jest używanie jednoargumetowego operatora "-", może
on wystąpić dowolną ilość razy, jednak parzyste wystąpienia np.: --<liczba> niczego nie zmieni; ...real różni się od ...int tylko
tym, że w "int" mogą wystąpić jedynie liczby całkowite, w przypadku "real" mogą to być dowolne liczby.
<dzialanie_bool>  -> opis na przykładach:
(<oper_not>)* <wart_bool> (<oper_bool> (<oper_not>)* <wart_bool>) * (<oper_bool> (<oper_not>)* <porownanie>)*
możemy uznać, że <porownanie> = <wart_bool> albo lepiej <stala_bool> bo porównanie taki właśnie zwróci nam wynik
uwagi: w tym przypadku liczba może mieć tylko jeden lub zero znaków negacji "-"!; stosowane są inne priorytety niż standardowo
przyjęte w Pascalu: porównania (najwyższy), operator not, operatory logiczne (najniższy).
<instr_if>         -> if <dzialanie_bool> then (<instr_blokowa> | <instr_prosta>) [else (<instr_blokowa> | <instr_prosta>)]
<instr_while>      -> while <dzialanie_bool> do (<instr_blokowa> | <instr_prosta>)
<instr_write>      -> (write | writeln) ((<stala_string>)* | (<zmienna>)* | ((, <zmienna>) | (, <stala_string>))*)
<instr_read>       -> (read | readln) (<zmienna>)
```

Dodatkowe uwagi, które mogły nie być uwzględnione w BNF.

Po instrukcji *if* oraz *while* może wystąpić tylko pojedyncza instrukcja lub blok instrukcji (*begin ... end*), w przypadku gdy ktoś chce ponownie użyć instrukcji *if* lub *while* musi ją umieścić w bloku "*begin ... end*".

Brak operatora "<>" (różny), zamiast niego należy stosować konstrukcje zastępcze np.: `not <liczba> = <liczba>`.

Wszystkie słowa kluczowe piszemy małymi literami, wielkość liter jest również rozróżniana w przypadku identyfikatorów (zaleca się stosować małe).

Komunikat wyświetlany w przypadku znalezienia błędu składniowego nie zawsze określa dokładnie to co jest niepoprawne ale najczyściej oscyluje

wystarczająco blisko.

Nie można wykonywać żadnych operacji na stringach (zmiennych stringowych) z wyjątkiem ich wyświetlania w instrukcji *write/ln* oraz pobierania ze standardowego wejścia przy pomocy instrukcji *read/ln*.

Działania arytmetyczne na liczbach zachowują sie tak jak w Java: gdy dwa składniki są typu *integer* wynik będzie *integer*, gdy chociaż jeden argument jest *real* wynik będzie *real.* Działania logiczne *or* i *and* można przeprowadzać tylko na zmiennych boolowskich, dlatego operator porównania ma większy priorytet niż operatory logiczne – nie ma wtedy potrzeby umieszczania porównania w nawiasach.

Operator dzielenia "/" w przypadku dwóch liczb typu *integer* wykona dzielenie całkowite:

3 / 2 = 1

natomiast:

3.0 / 2 = 1.5

W miejscach gdzie jest działanie logiczne nie mogą wystąpić operacje arytmetyczne, dodatkowo może wystąpić maksymalnie jeden operator negacji liczby "-" (minus), przykłady:

```
false or (2 + 3) >= 3   BŁĄD
false or (5) >= 3       BŁĄD
false or (-5) >= 3      BŁĄD
false or (--5 >= 3)     BŁĄD
false or (-5 >= 3)      OK
false or -5 >= 3        OK
false or not -5 >= 3    OK
false or 5 >= 3         OK
```

Instrukcja *write/ln* podobnie do tej z Pascala, może drukować wiele elementów jednocześnie, warunkiem jest tylko to, że musi być to stała postaci stringowej: "apostrof" "dowolne znaki" "apostrof" lub identyfikator zmiennej oddzielone od siebie przecinkami ",". W przypadku zmiennej *boolean* wydrukowany zostanie napis: *true* lub *false*.

Wyrażenia arytmetyczne i logiczne muszą kończyć się zmienną, stałą (odpowiedniego typu) lub nawiasem zamykającym ")".

Nie występuje coś takiego jak instrukcja pusta, czyli sam średnik ";".

Działanie instrukcji *read* i *readln* jest identyczne i mogą być stosowane naprzemiennie.

Konstrukcja najprostszego programu:

```
program test;
begin
end.
```

Silnia

```
program silnia;
var
      silnia: real;
      b: integer;
begin
      silnia := 1;
      b := 1;
      while b <= 40 do
            begin
                  silnia := silnia * b;
                  writeln('Silnia dla b: ', b, ' wynosi: ', silnia);
                  b := b + 1;
            end;
end.
```

Drukowanie liczb z przedziału

Przedział: <25, 31) U (42, 48>

```
program przedzialy;
var
      b: integer;
begin
      b := 0;

      while b <= 100 do
            begin
                  if (b >= 25 and b < 31) or (b > 42 and b <= 48) then
                        writeln(b);

                  b := b + 1;

            end;
end.
```