Informatyka 1

Wykład III

Wyrażenia i instrukcje, złożoność obliczeniowa

Robert Muszyński
ZPCiR ICT PWr

Zagadnienia: składnia wyrażeń, drzewa rozbioru gramatycznego i wyliczenia wartości wyrażeń, operatory `DIV` i `MOD`, zagadnienia złożoności obliczeniowej, instrukcja warunkowa `IF-THEN`, forma zapisu programu, instrukcje pętli `WHILE-DO` i `FOR-TO-DO`.

– Skład FoilTEX –

Składnia wyrażeń

```
wyrazenie = proste-wyrazenie
        [operator-relacyjny proste-wyrazenie].

operator-relacyjny =
        "=" | "<>" | "<" | ">" | "<=" | ">=" | "IN".
```

Składnia wyrażeń

```
wyrazenie = proste-wyrazenie
        [operator-relacyjny proste-wyrazenie].

operator-relacyjny =
        "=" | "<>" | "<" | ">" | "<=" | ">=" | "IN".

proste-wyrazenie = [znak-liczby]
        skladnik {operator-addytywny skladnik}.
```

Składnia wyrażeń

```
wyrazenie = proste-wyrazenie
        [operator-relacyjny proste-wyrazenie].

operator-relacyjny =
        "=" | "<>" | "<" | ">" | "<=" | ">=" | "IN".

proste-wyrazenie = [znak-liczby]
        skladnik {operator-addytywny skladnik}.

operator-addytywny =  "+" | "-" |"><" | "OR" | "OR_ELSE".
```

Składnia wyrażeń

```
wyrazenie = proste-wyrazenie
        [operator-relacyjny proste-wyrazenie].
operator-relacyjny =
        "=" | "<>" | "<" | ">" | "<=" | ">=" | "IN".
proste-wyrazenie = [znak-liczby]
        skladnik {operator-addytywny skladnik}.
operator-addytywny =  "+" | "-" |"><" | "OR" | "OR_ELSE".
skladnik = czynnik {operator-multiplikatywny czynnik}.
```

Składnia wyrażeń

```
wyrazenie = proste-wyrazenie
        [operator-relacyjny proste-wyrazenie].

operator-relacyjny =
        "=" | "<>" | "<" | ">" | "<=" | ">=" | "IN".

proste-wyrazenie = [znak-liczby]
        skladnik {operator-addytywny skladnik}.

operator-addytywny =  "+" | "-" |"><" | "OR" | "OR_ELSE".

skladnik = czynnik {operator-multiplikatywny czynnik}.

operator-multiplikatywny = "*" | "/" |
                  "DIV" | "MOD" | "AND" | "AND_THEN".
```

Składnia wyrażeń

```
wyrazenie = proste-wyrazenie
        [operator-relacyjny proste-wyrazenie].

operator-relacyjny =
        "=" | "<>" | "<" | ">" | "<=" | ">=" | "IN".

proste-wyrazenie = [znak-liczby]
        skladnik {operator-addytywny skladnik}.

operator-addytywny =  "+" | "-" |"><" | "OR" | "OR_ELSE".

skladnik = czynnik {operator-multiplikatywny czynnik}.

operator-multiplikatywny = "*" | "/" |
                  "DIV" | "MOD" | "AND" | "AND_THEN".

czynnik = zmienna | stala-bez-znaku
        | wywolanie-funkcji | konstruktor-zbioru
        | "(" wyrazenie ")" | "NOT" czynnik
        | identyfikator-graniczny.
```

Składnia wyrażeń

```
wyrazenie = proste-wyrazenie
        [operator-relacyjny proste-wyrazenie].

operator-relacyjny =
        "=" | "<>" | "<" | ">" | "<=" | ">=" | "IN".

proste-wyrazenie = [znak-liczby]
        skladnik {operator-addytywny skladnik}.

operator-addytywny =  "+" | "-" |"><" | "OR" | "OR_ELSE".

skladnik = czynnik {operator-multiplikatywny czynnik}.

operator-multiplikatywny = "*" | "/" |
                  "DIV" | "MOD" | "AND" | "AND_THEN".

czynnik = zmienna | stala-bez-znaku
        | wywolanie-funkcji | konstruktor-zbioru
        | "(" wyrazenie ")" | "NOT" czynnik
        | identyfikator-graniczny.

stala-bez-znaku = liczba-bez-znaku | napis
                | identyfikator-stalej |  "NIL".
```

Przykładowe wyrażenia

$2.0 * 3.5$ — argumenty i wartość typu REAL

Przykładowe wyrażenia

$2.0 * 3.5$ — argumenty i wartość typu REAL

$(2 + 8) * 16$ — argumenty i wartość typu INTEGER

Przykładowe wyrażenia

$2.0 * 3.5$ — argumenty i wartość typu REAL

$(2 + 8) * 16$ — argumenty i wartość typu INTEGER

X>2.5 — lewy argument typu REAL lub INTEGER, prawy — REAL, wynik typu BOOLEAN

Przykładowe wyrażenia

$2.0 * 3.5$	—	argumenty i wartość typu REAL
$(2 + 8) * 16$	—	argumenty i wartość typu INTEGER
X>2.5	—	lewy argument typu REAL lub INTEGER, prawy — REAL, wynik typu BOOLEAN
$N * 25$	—	INTEGER lub REAL, zależnie od typu N

Przykładowe wyrażenia

$2.0 * 3.5$	–	argumenty i wartość typu REAL
$(2 + 8) * 16$	–	argumenty i wartość typu INTEGER
X>2.5	–	lewy argument typu REAL lub INTEGER, prawy — REAL, wynik typu BOOLEAN
$N * 25$	–	INTEGER lub REAL, zależnie od typu N
Z='T'	–	argumenty typu CHAR, wynik BOOLEAN

Przykładowe wyrażenia

$2.0 * 3.5$	–	argumenty i wartość typu REAL
$(2 + 8) * 16$	–	argumenty i wartość typu INTEGER
X>2.5	–	lewy argument typu REAL lub INTEGER, prawy — REAL, wynik typu BOOLEAN
$N * 25$	–	INTEGER lub REAL, zależnie od typu N
Z='T'	–	argumenty typu CHAR, wynik BOOLEAN
Z=15	–	jeśli Z jest CHAR, to wyrażenie jest niepoprawne, (mówimy: nielegalne), zawiera niezgodność typów

Przykładowe wyrażenia

$2.0 * 3.5$	–	argumenty i wartość typu REAL
$(2 + 8) * 16$	–	argumenty i wartość typu INTEGER
X>2.5	–	lewy argument typu REAL lub INTEGER, prawy — REAL, wynik typu BOOLEAN
$N * 25$	–	INTEGER lub REAL, zależnie od typu N
Z='T'	–	argumenty typu CHAR, wynik BOOLEAN
Z=15	–	jeśli Z jest CHAR, to wyrażenie jest niepoprawne, (mówimy: nielegalne), zawiera niezgodność typów
(X>0) AND (Z='T')	–	wyrażenie poprawne, wynik BOOLEAN

Wyrażenia — dalsze przykłady

```
i:=11; x:=-2.0; b:=FALSE; z:='a';
```

```
i * x >= ORD(z)
```

Wyrażenia — dalsze przykłady

```
i:=11; x:=-2.0; b:=FALSE; z:='a';
```

```
i * x >= ORD(z)
(z <= SUCC(z)) = 'TRUE'
```

Wyrażenia — dalsze przykłady

```
i:=11; x:=-2.0; b:=FALSE; z:='a';

i * x >= ORD(z)
(z <= SUCC(z)) = 'TRUE'
( i * x >= 0 ) < ( z = 'Z' ) AND b
```

Wyrażenia — dalsze przykłady

```
i:=11; x:=-2.0; b:=FALSE; z:='a';

i * x >= ORD(z)
(z <= SUCC(z)) = 'TRUE'
( i * x >= 0 ) < ( z = 'Z' ) AND b
i DIV i + x
```

Wyrażenia — dalsze przykłady

```
i:=11; x:=-2.0; b:=FALSE; z:='a';
```

```
i * x >= ORD(z)
(z <= SUCC(z)) = 'TRUE'
( i * x >= 0 ) < ( z = 'Z' ) AND b
i DIV i + x
i DIV (i + x)
```

Wyrażenia — dalsze przykłady

```
i:=11; x:=-2.0; b:=FALSE; z:='a';

i * x >= ORD(z)
(z <= SUCC(z)) = 'TRUE'
( i * x >= 0 ) < ( z = 'Z' ) AND b
i DIV i + x
i DIV (i + x)
PRED('b')=CHR(ORD('c')-1)
```

Wyrażenia — dalsze przykłady

```
i:=11; x:=-2.0; b:=FALSE; z:='a';

i * x >= ORD(z)
(z <= SUCC(z)) = 'TRUE'
( i * x >= 0 ) < ( z = 'Z' ) AND b
i DIV i + x
i DIV (i + x)
PRED('b')=CHR(ORD('c')-1)
SUCC(PRED('b'))=CHR(ORD('b'))
```

Wyrażenia — dalsze przykłady

```
x,y:INTEGER; V,W:BOOLEAN; a,b:CHAR;
```

```
NOT V AND W OR (x = y) OR (a <> b)
```

Wyrażenia — dalsze przykłady

```
x,y:INTEGER; V,W:BOOLEAN; a,b:CHAR;

NOT V AND W OR (x = y) OR (a <> b)
x MOD y = x DIV y
```

Wyrażenia — dalsze przykłady

```
x,y:INTEGER; V,W:BOOLEAN; a,b:CHAR;

NOT V AND W OR (x = y) OR (a <> b)
x MOD y = x DIV y
(((x=y)=V)=W)
```

Wyrażenia — dalsze przykłady

```
x,y:INTEGER; V,W:BOOLEAN; a,b:CHAR;

NOT V AND W OR (x = y) OR (a <> b)
x MOD y = x DIV y
(((x=y)=V)=W)
(NOT V = W) AND (NOT x = y) OR (NOT a <> b)
```

Wyrażenia — dalsze przykłady

```
x,y:INTEGER; V,W:BOOLEAN; a,b:CHAR;

NOT V AND W OR (x = y) OR (a <> b)
x MOD y = x DIV y
(((x=y)=V)=W)
(NOT V = W) AND (NOT x = y) OR (NOT a <> b)
(V = NOT W) AND NOT (x < y) AND (a > b)
```

Wyrażenia — dalsze przykłady

```
x,y:INTEGER; V,W:BOOLEAN; a,b:CHAR;

NOT V AND W OR (x = y) OR (a <> b)
x MOD y = x DIV y
(((x=y)=V)=W)
(NOT V = W) AND (NOT x = y) OR (NOT a <> b)
(V = NOT W) AND NOT (x < y) AND (a > b)
(x = x DIV y) AND (NOT y = x MOD y)
```

Wyrażenia — dalsze przykłady

Czy wyrażenie odpowiada formule matematycznej?

$$\frac{a+b}{ab}$$

```
(a + b) / a * b
```

Wyrażenia — dalsze przykłady

Czy wyrażenie odpowiada formule matematycznej?

$\frac{a+b}{ab}$ `(a + b) / a * b`

$e^{2x} \sin 5x$ `sin(5x) * exp(2x)`

Wyrażenia — dalsze przykłady

Czy wyrażenie odpowiada formule matematycznej?

$\frac{a+b}{ab}$	`(a + b) / a * b`
$e^{2x} \sin 5x$	`sin(5x) * exp(2x)`
$5 \sin \frac{a}{2} + \frac{1}{2} \cos(tx)$	`5*sin(a/2)+1/3*cos(t*x)`

Wyrażenia — dalsze przykłady

Czy wyrażenie odpowiada formule matematycznej?

$\frac{a+b}{ab}$	`(a + b) / a * b`
$e^{2x}\sin 5x$	`sin(5x) * exp(2x)`
$5\sin\frac{a}{2}+\frac{1}{2}\cos(tx)$	`5*sin(a/2)+1/3*cos(t*x)`
$\frac{a+b}{c}:\frac{def}{g}$	`(a+b)/c/d*e*f/g`

Wyrażenia — dalsze przykłady

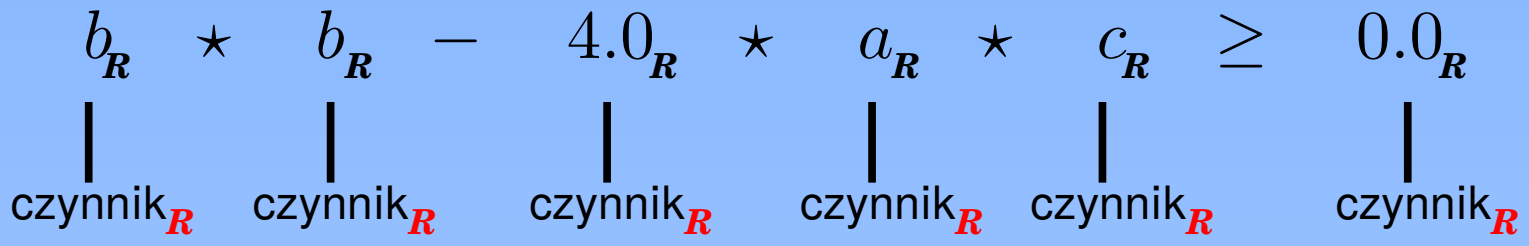
Czy wyrażenie odpowiada formule matematycznej?

$\frac{a+b}{ab}$	`(a + b) / a * b`
$e^{2x}\sin 5x$	`sin(5x) * exp(2x)`
$5\sin\frac{a}{2}+\frac{1}{2}\cos(tx)$	`5*sin(a/2)+1/3*cos(t*x)`
$\frac{a+b}{c}:\frac{def}{g}$	`(a+b)/c/d*e*f/g`
$\sin x + e^{2x\cos x}$	`sin(x) + exp(2*x*cos(x))`

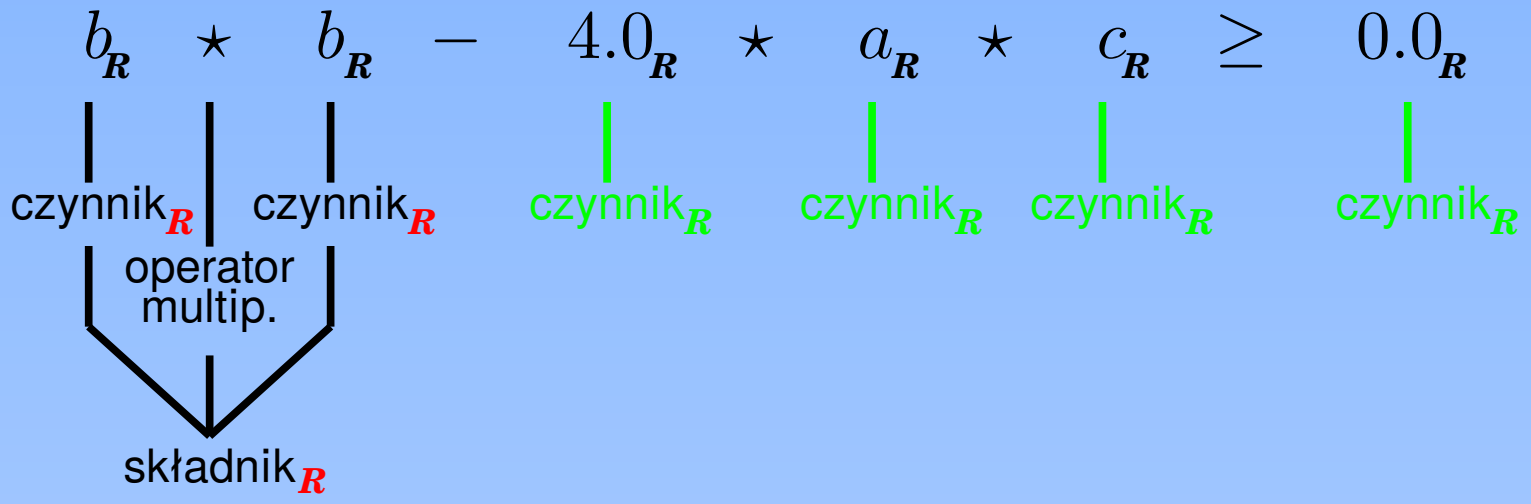
Drzewa rozbioru gramatycznego

$$b_R \quad \star \quad b_R \quad - \quad 4.0_R \quad \star \quad a_R \quad \star \quad c_R \quad \geq \quad 0.0_R$$

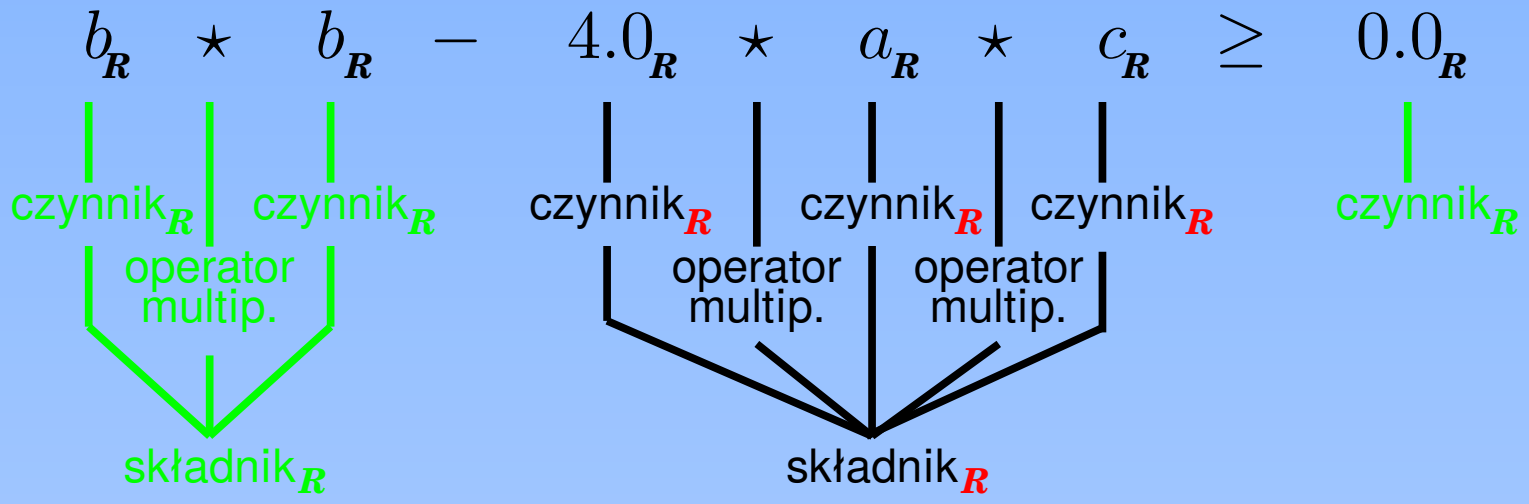
Drzewa rozbioru gramatycznego



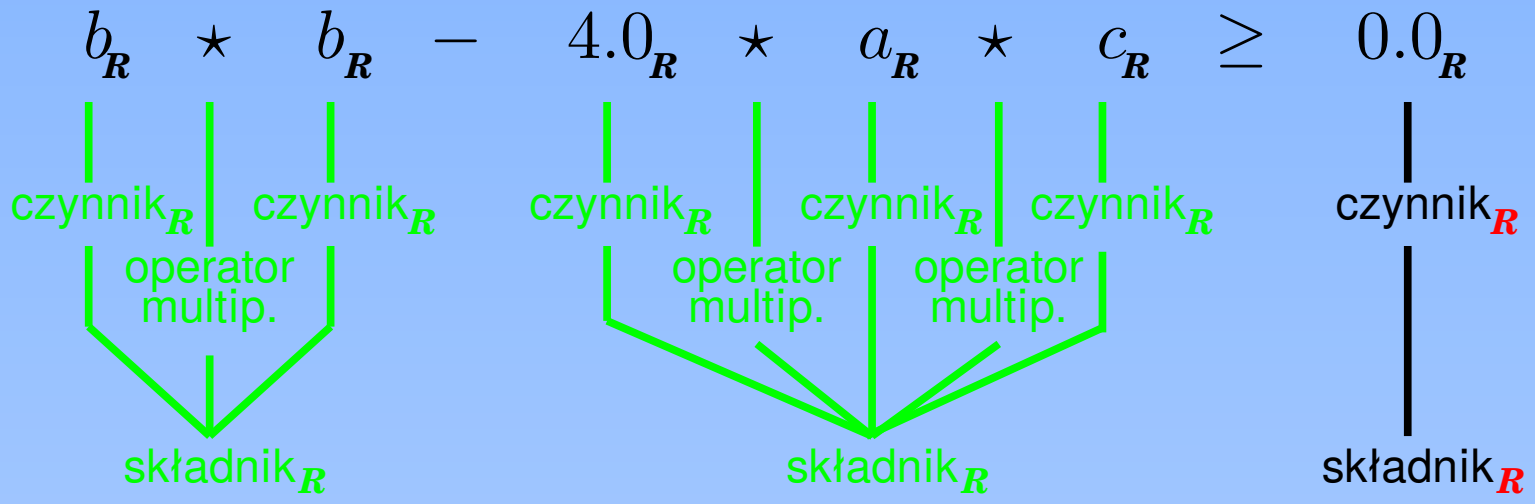
Drzewa rozbioru gramatycznego



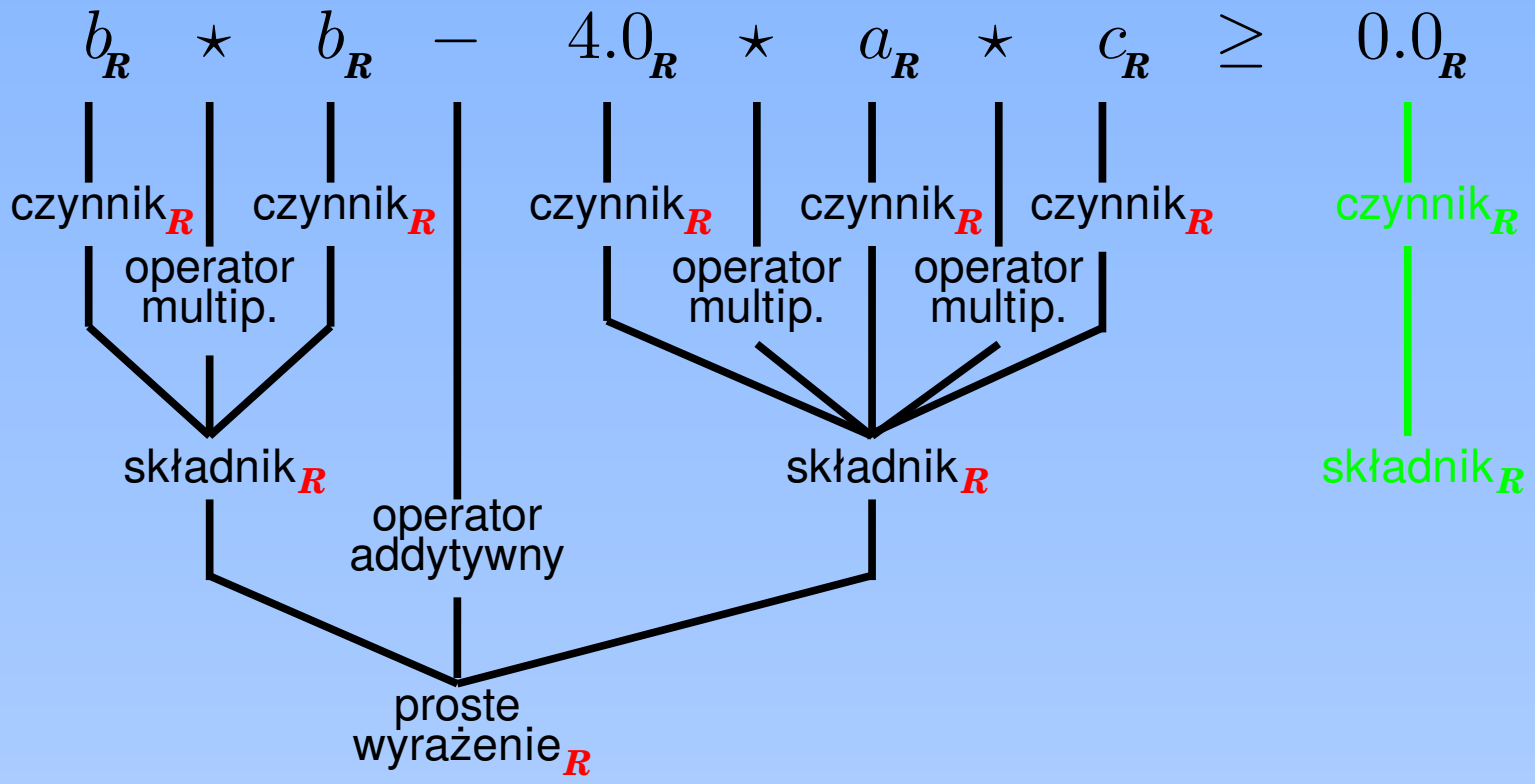
Drzewa rozbioru gramatycznego



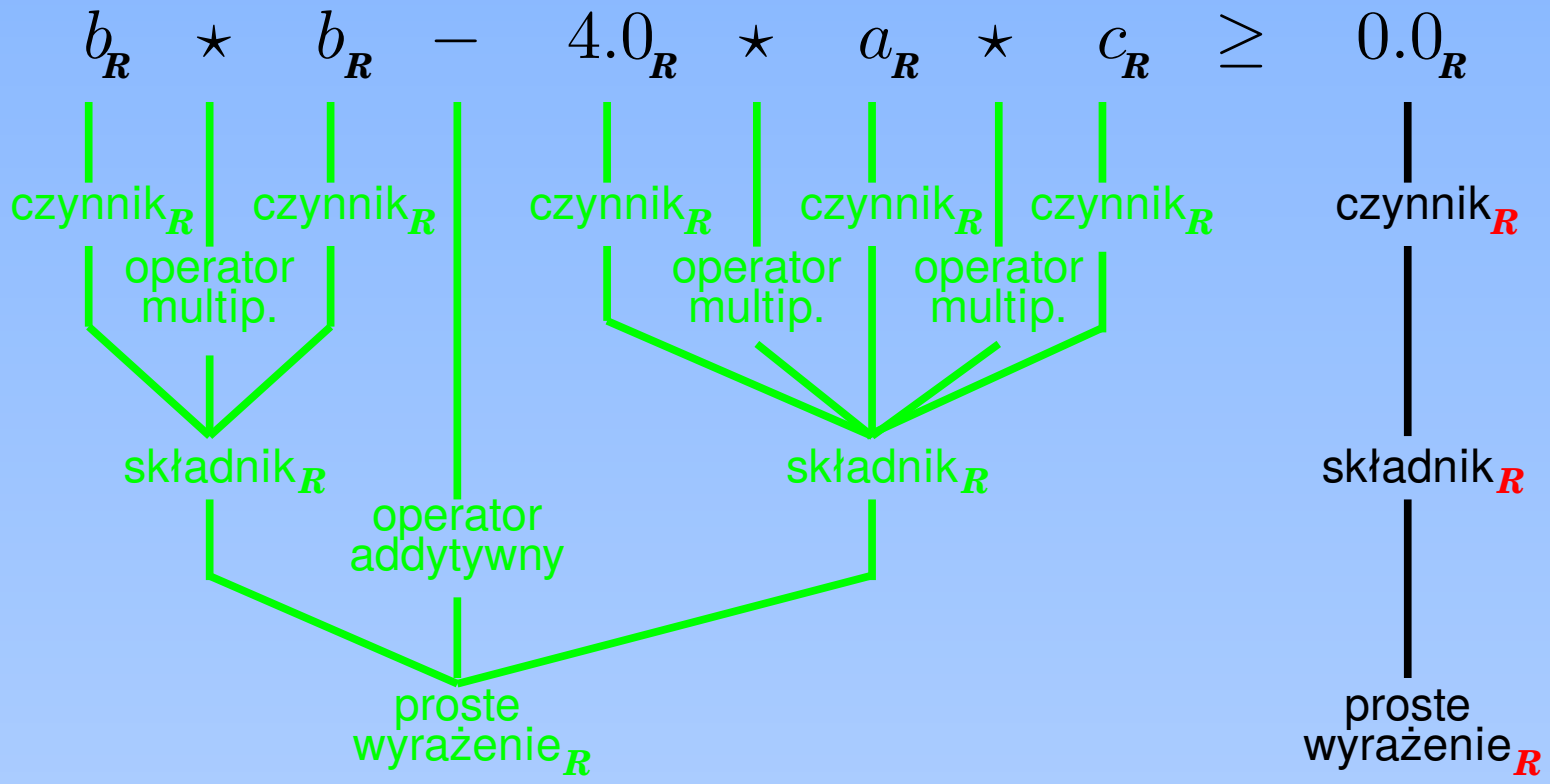
Drzewa rozbioru gramatycznego



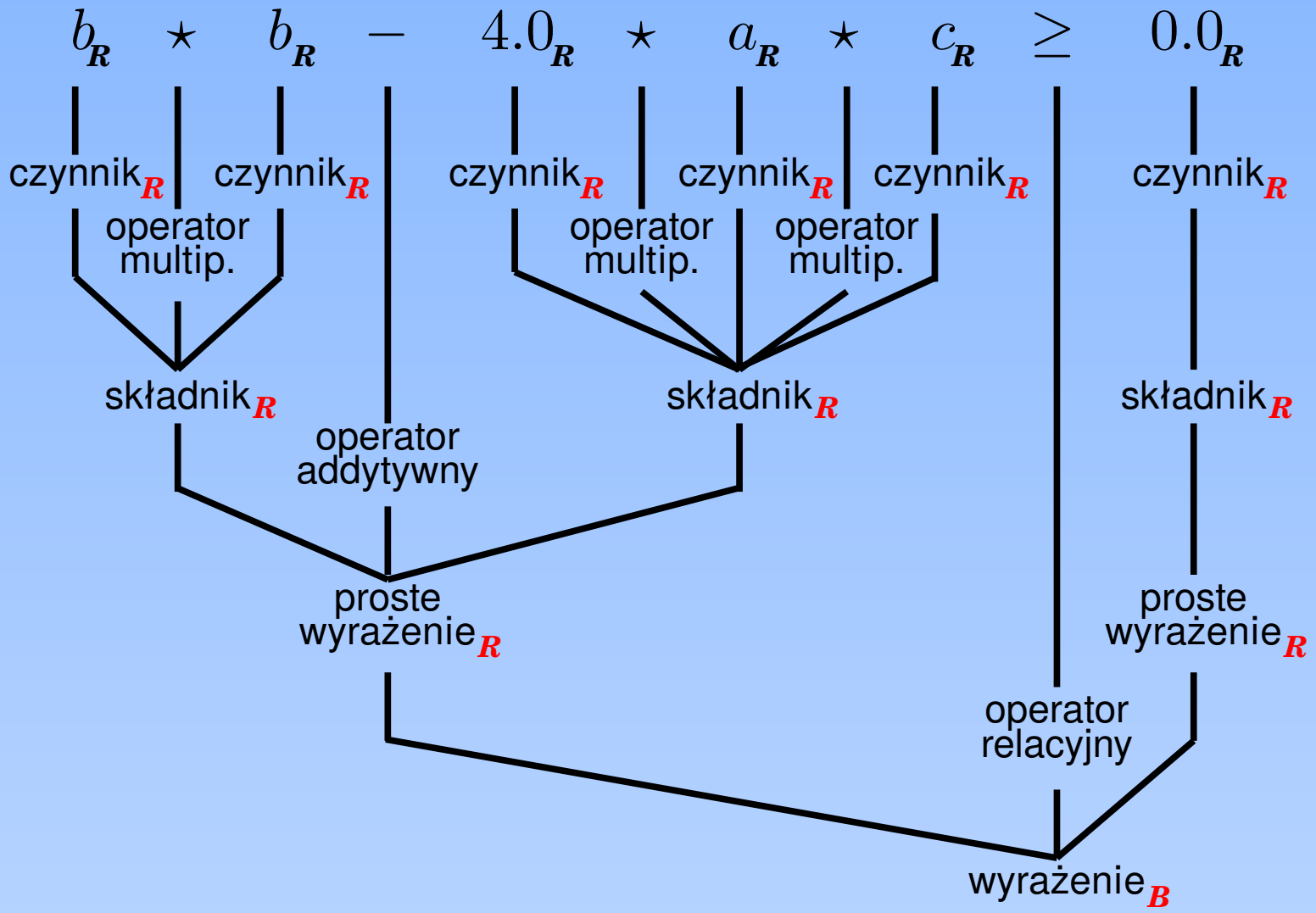
Drzewa rozbioru gramatycznego



Drzewa rozbioru gramatycznego



Drzewa rozbioru gramatycznego



Drzewa rozbioru gramatycznego

- Dowodzą zgodności wyrażenia z regułami języka.

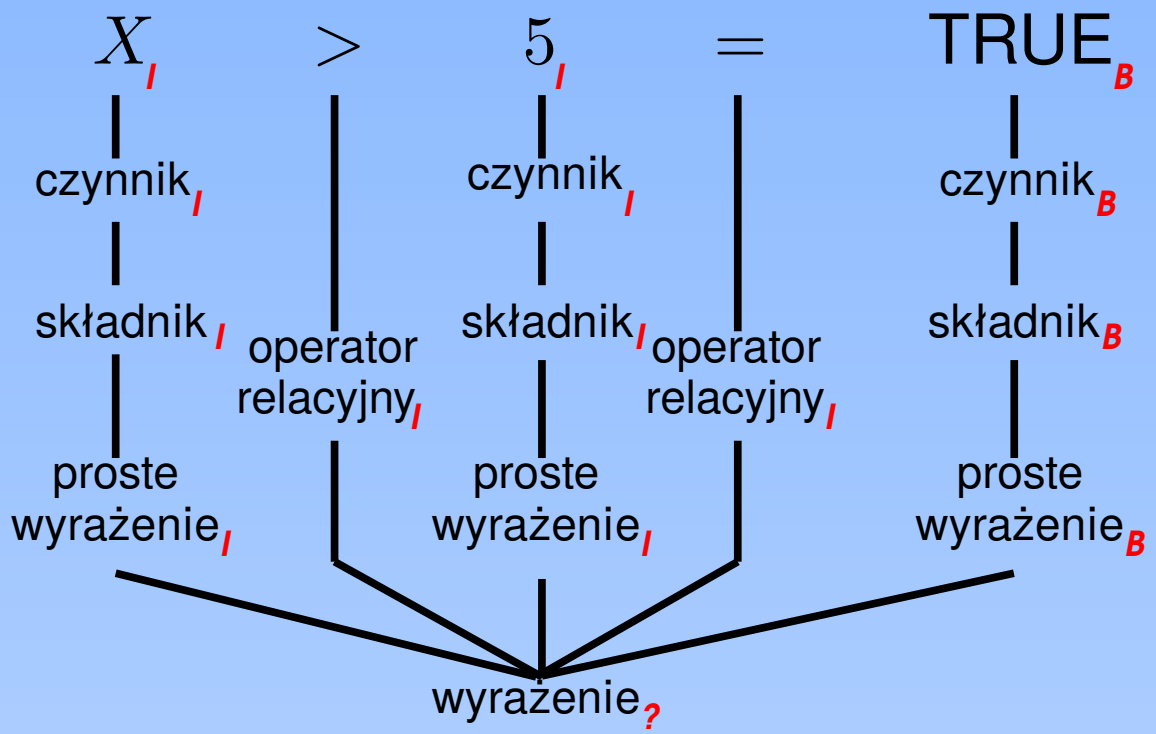
Drzewa rozbioru gramatycznego

- Dowodzą zgodności wyrażenia z regułami języka.
- Nie dowodzą zgodności typów.

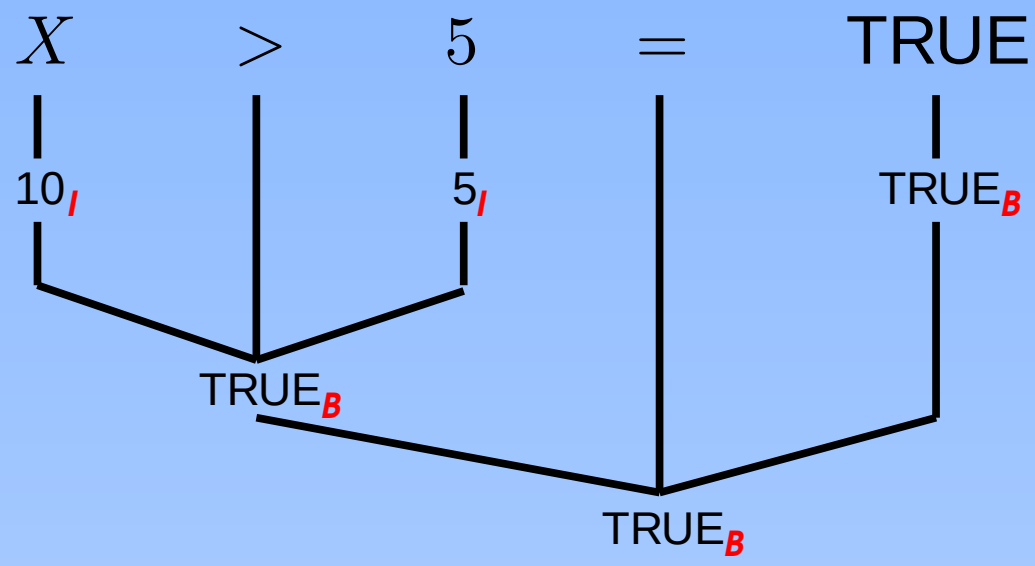
Drzewa rozbioru gramatycznego

- Dowodzą zgodności wyrażenia z regułami języka.
- Nie dowodzą zgodności typów.
- Nie w pełni odpowiadają porządkowi obliczeń.

Drzewa rozbioru gramatycznego



Drzewa wyliczania wartości wyrażeń

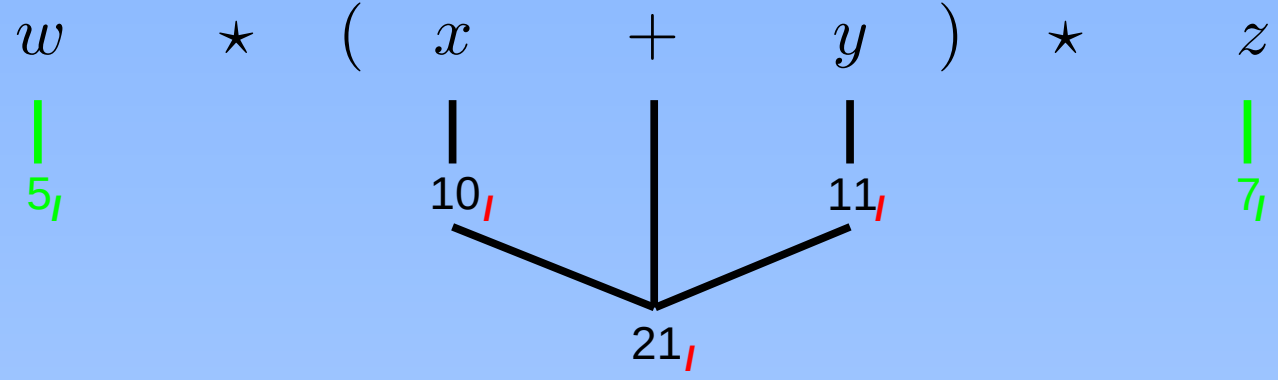


Drzewa wyliczania wartości wyrażeń

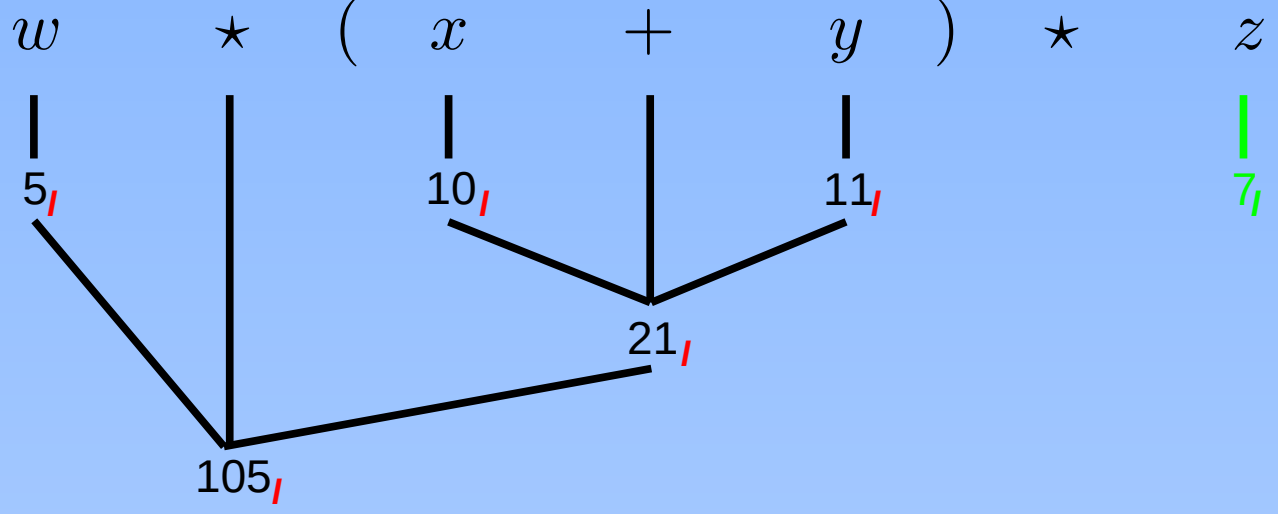
$$w \quad \star \quad (\quad x \quad + \quad y \quad) \quad \star \quad z$$

Drzewa wyliczania wartości wyrażeń

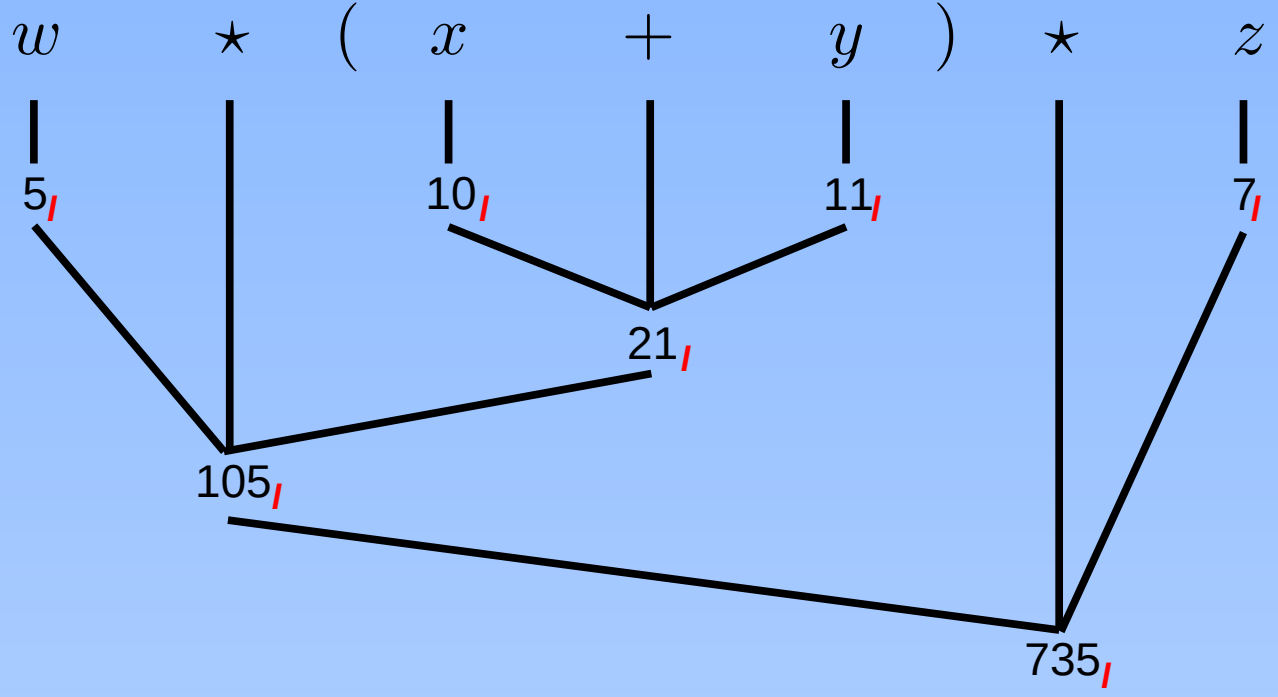
Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń

- Są podobne do drzew rozbioru gramatycznego.

Drzewa wyliczania wartości wyrażeń

- Są podobne do drzew rozbioru gramatycznego.
- Odzwierciedlają porządek obliczeń od lewej do prawej i priorytety operatorów.

Drzewa wyliczania wartości wyrażeń

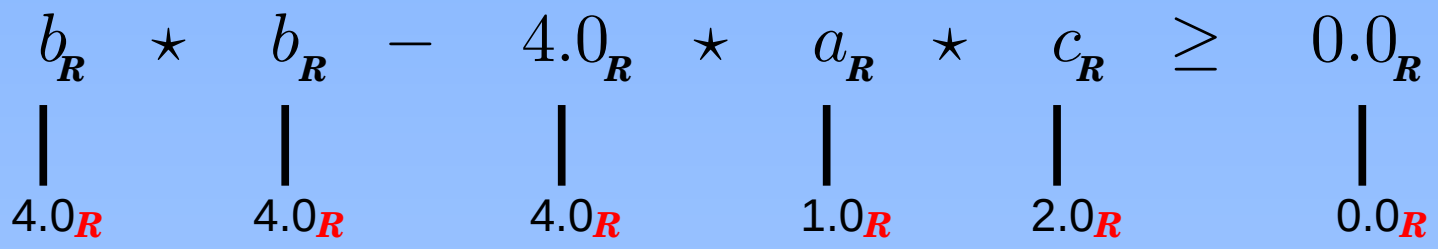
- Są podobne do drzew rozbioru gramatycznego.
- Odzwierciedlają porządek obliczeń od lewej do prawej i priorytety operatorów.
- Pozwalają sprawdzić zgodność typów oraz wyliczyć wartości wyrażeń (czasami).

Drzewa wyliczania wartości wyrażeń

$$b_R \star b_R - 4.0_R \star a_R \star c_R \geq 0.0_R$$

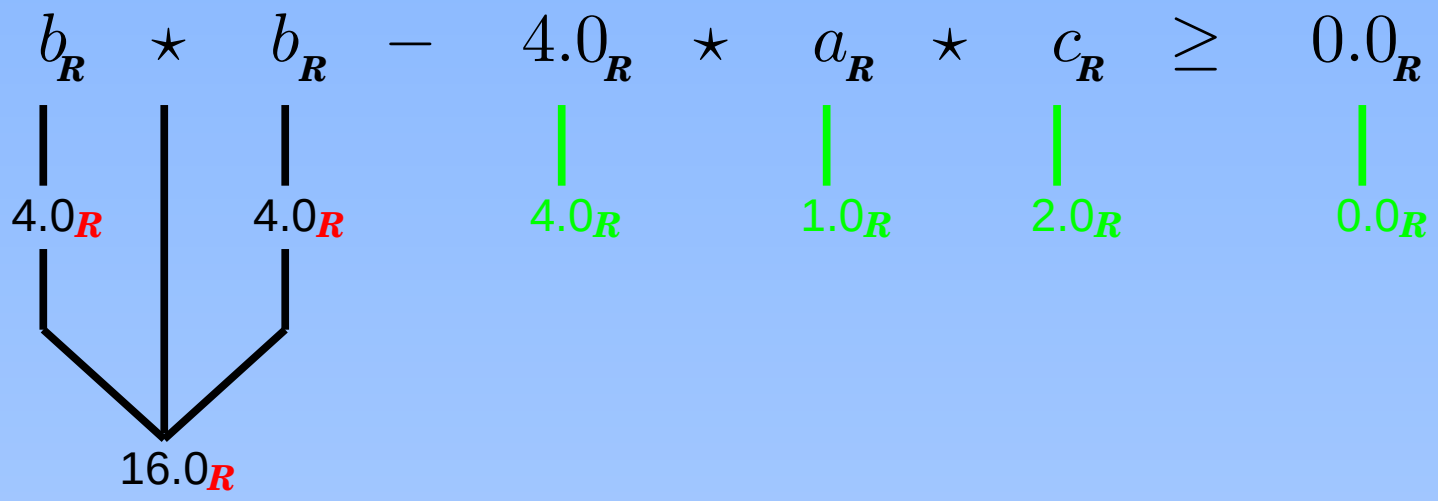
(a=1.0)
(b=4.0)
(c=2.0)

Drzewa wyliczania wartości wyrażeń



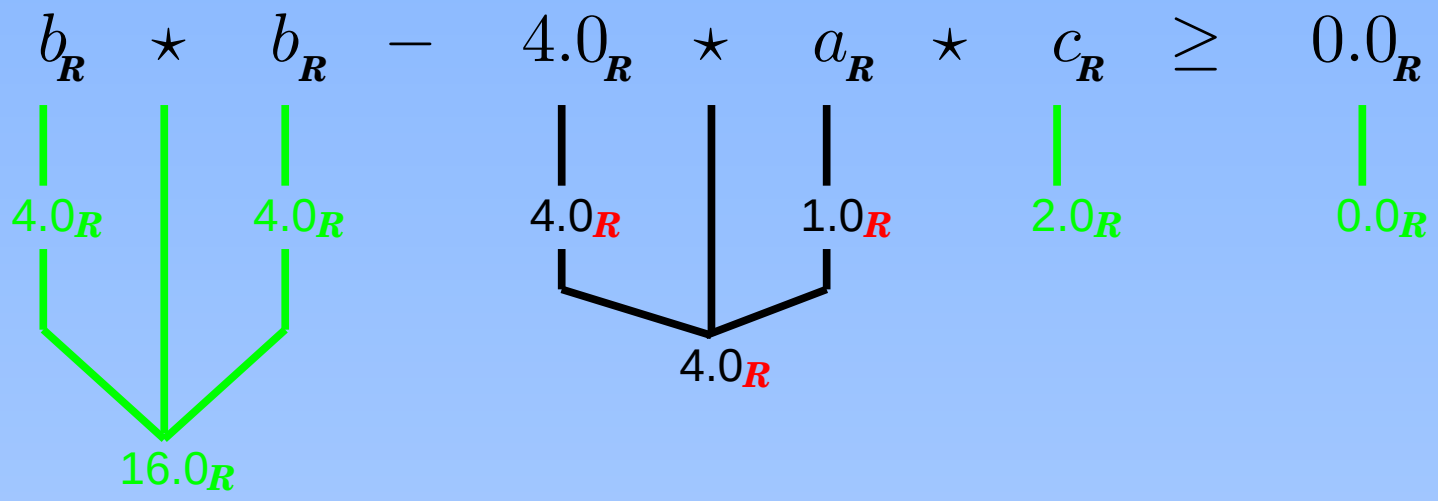
(a=1.0)
(b=4.0)
(c=2.0)

Drzewa wyliczania wartości wyrażeń



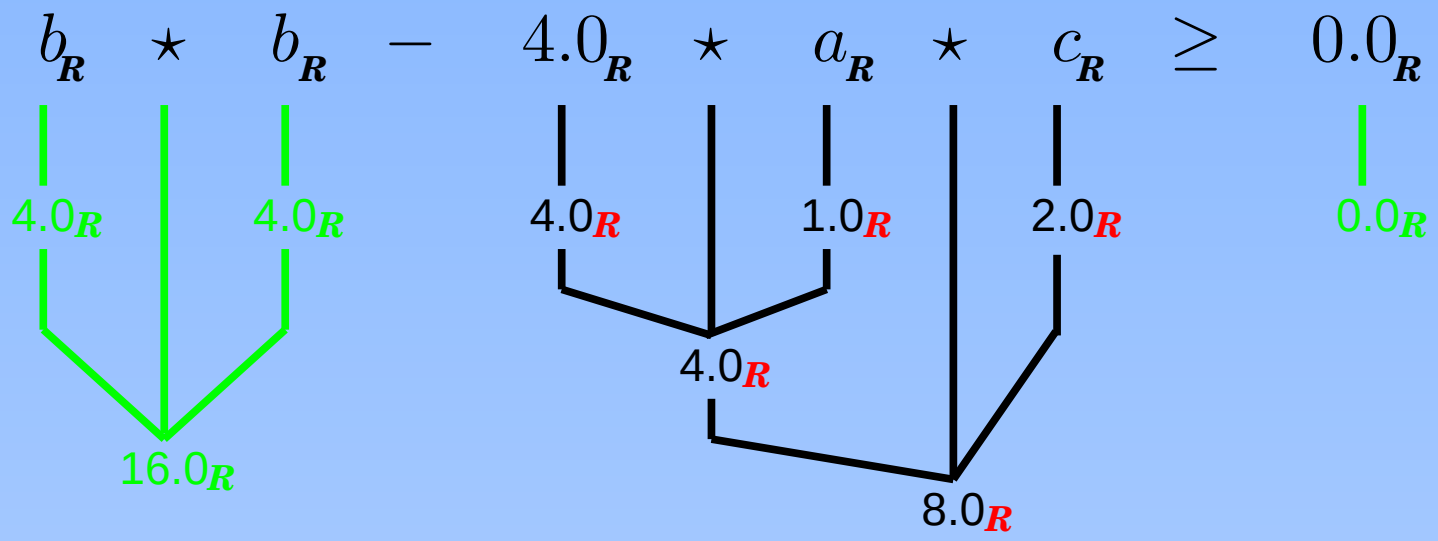
(a=1.0)
(b=4.0)
(c=2.0)

Drzewa wyliczania wartości wyrażeń



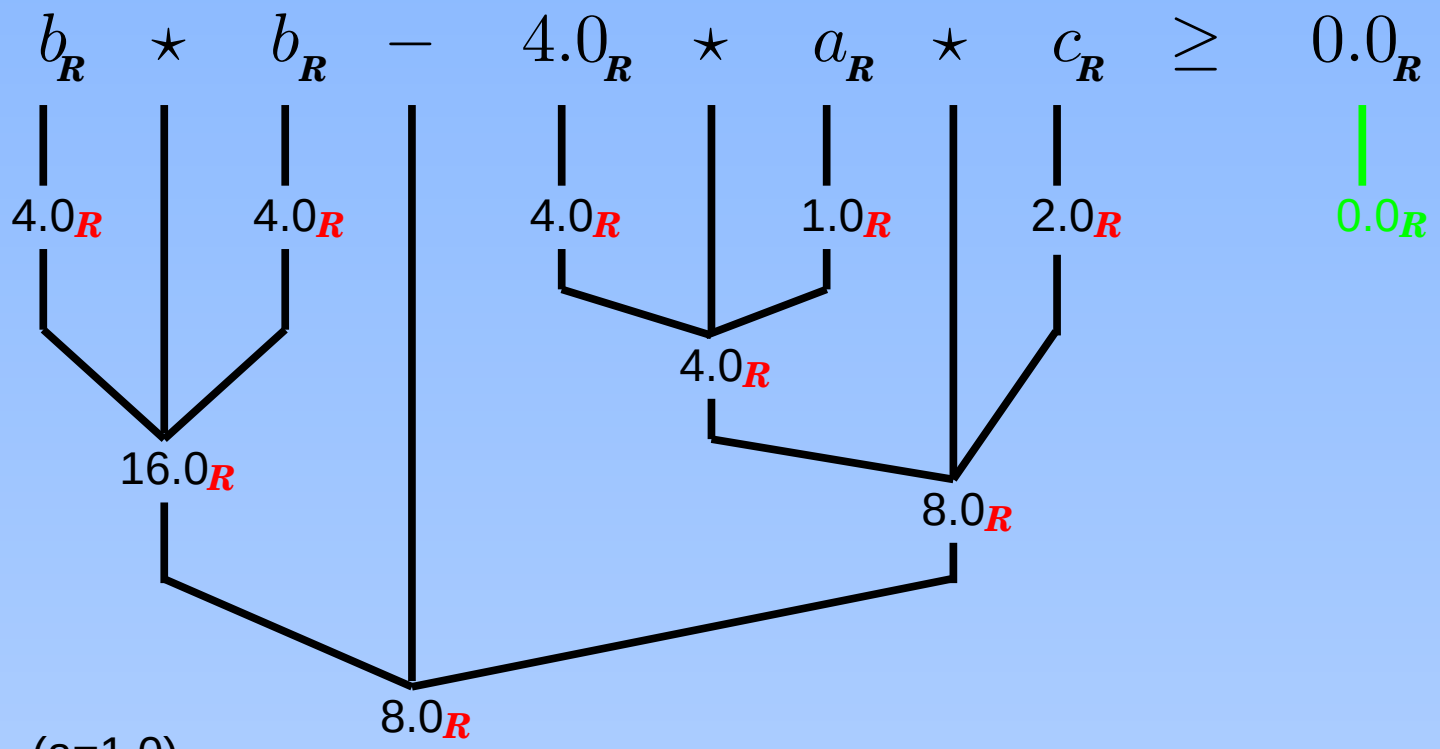
(a=1.0)
(b=4.0)
(c=2.0)

Drzewa wyliczania wartości wyrażeń



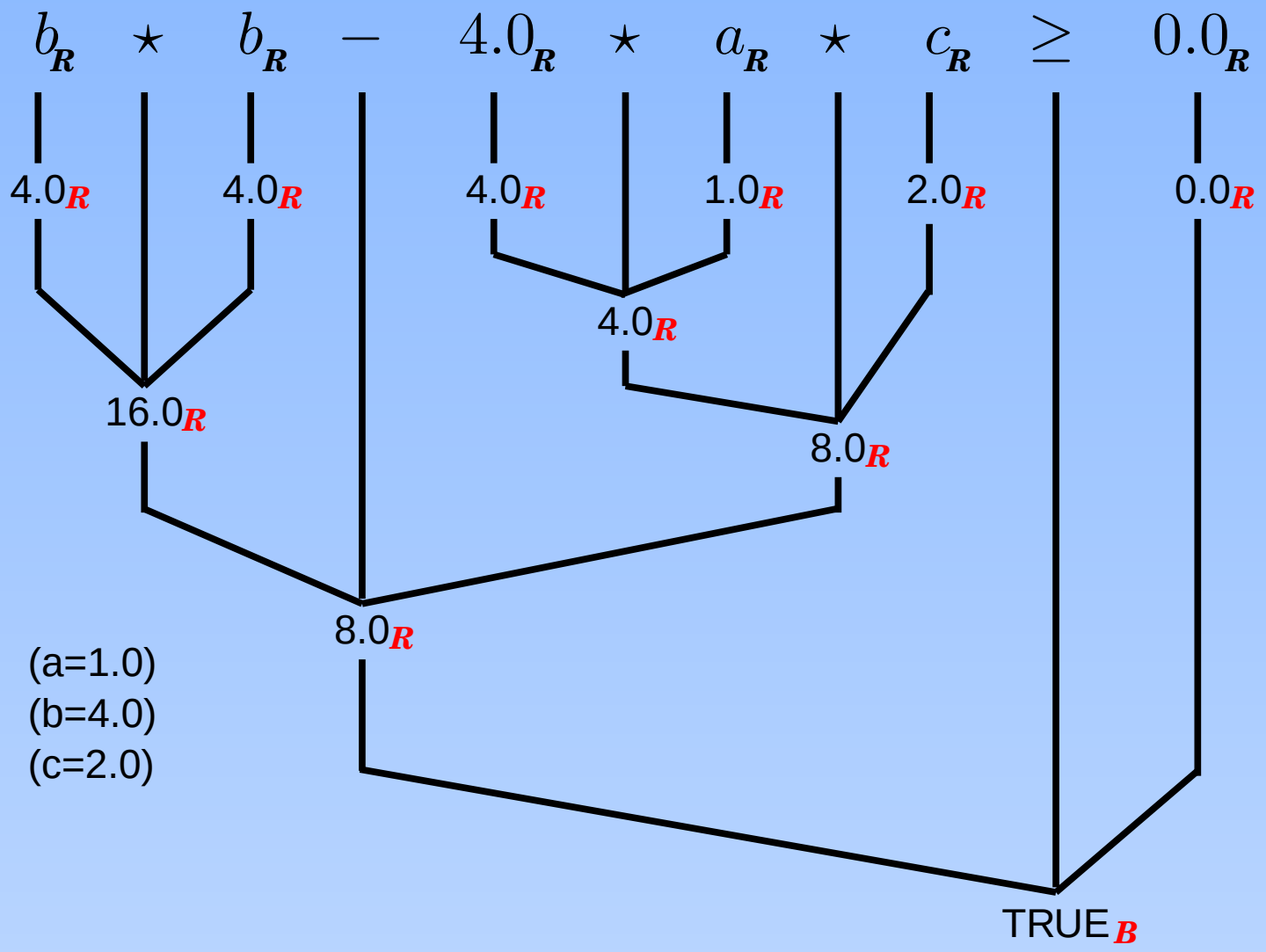
(a=1.0)
(b=4.0)
(c=2.0)

Drzewa wyliczania wartości wyrażeń



(a=1.0)
(b=4.0)
(c=2.0)

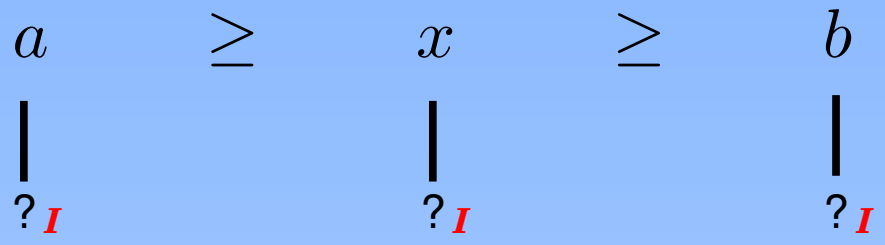
Drzewa wyliczania wartości wyrażeń



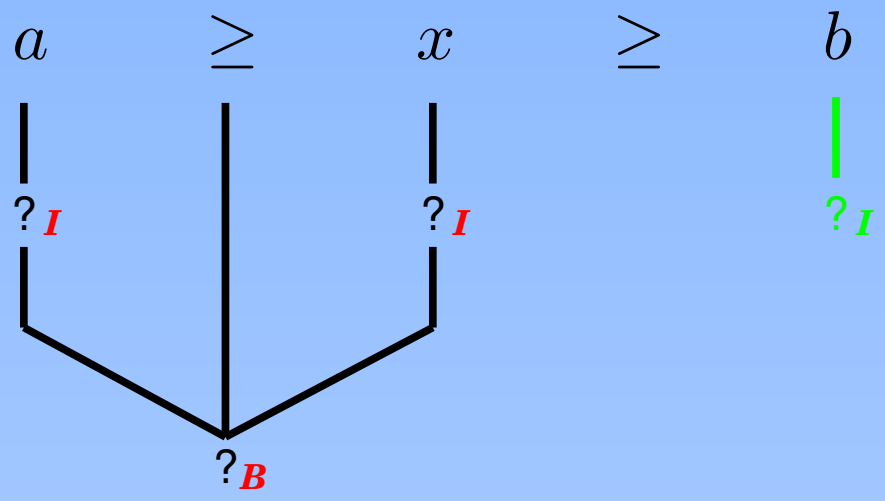
Drzewa wyliczania wartości wyrażeń

$$a \qquad \geq \qquad x \qquad \geq \qquad b$$

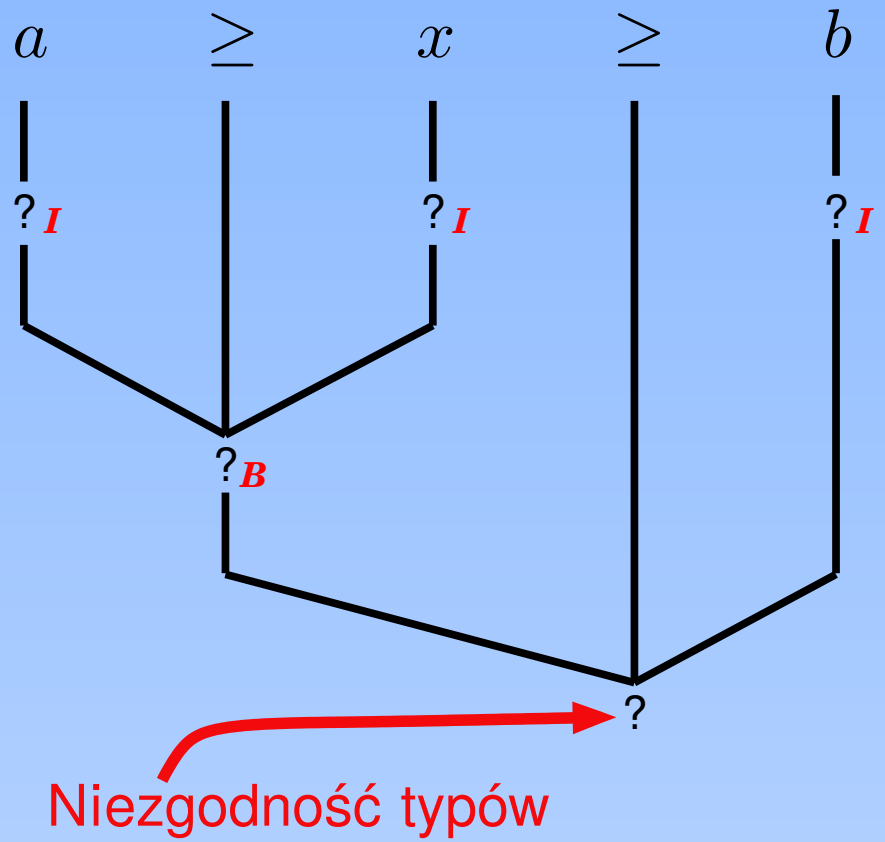
Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń

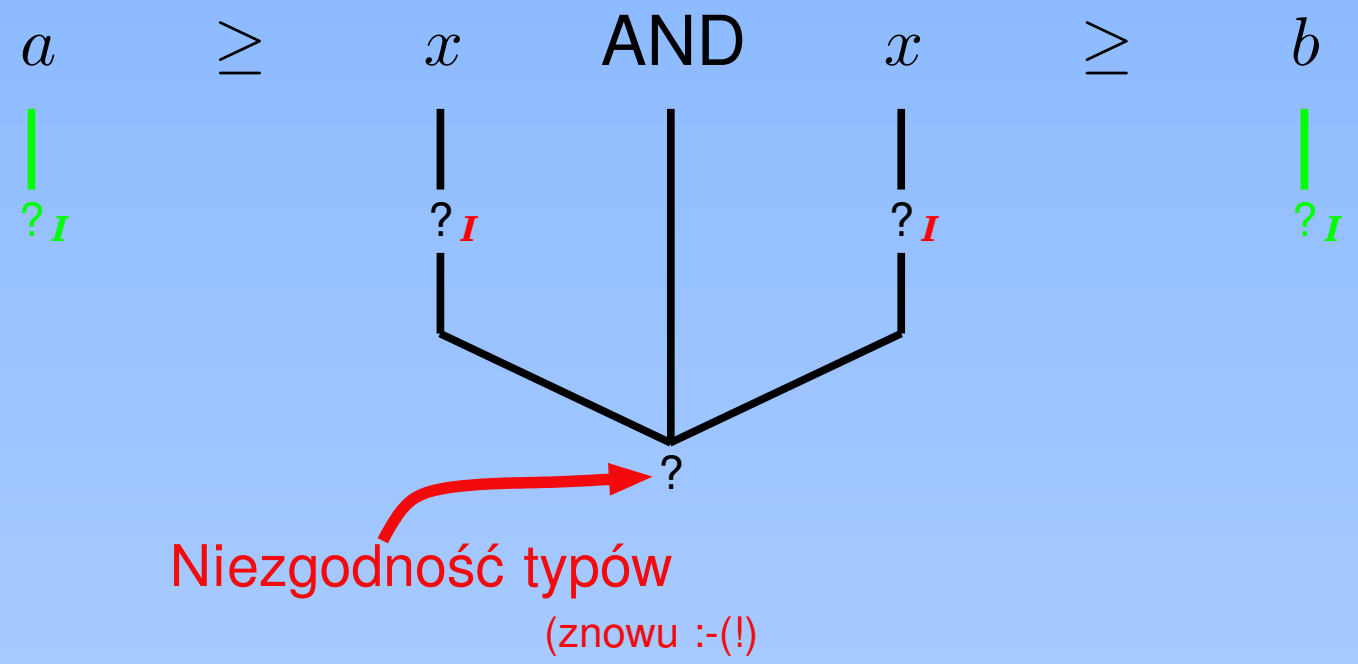


Drzewa wyliczania wartości wyrażeń

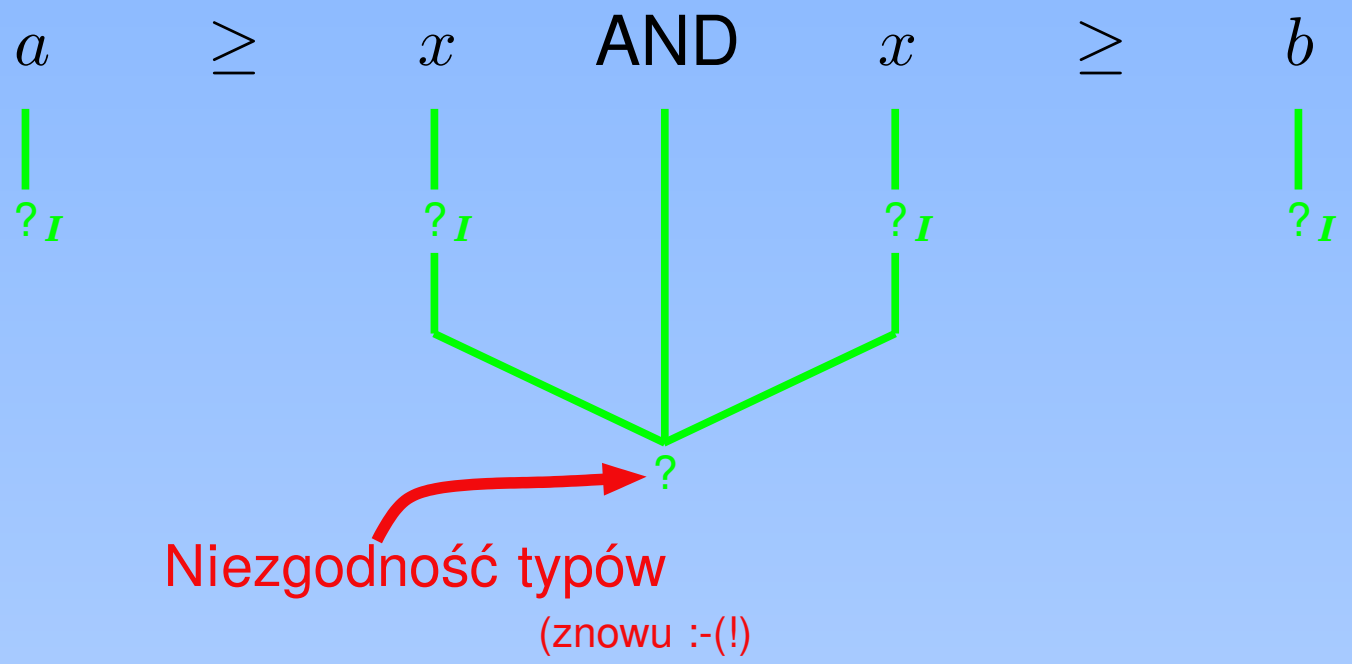
$$a \quad \geq \quad x \quad \text{AND} \quad x \quad \geq \quad b$$

Drzewa wyliczania wartości wyrażeń

Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń



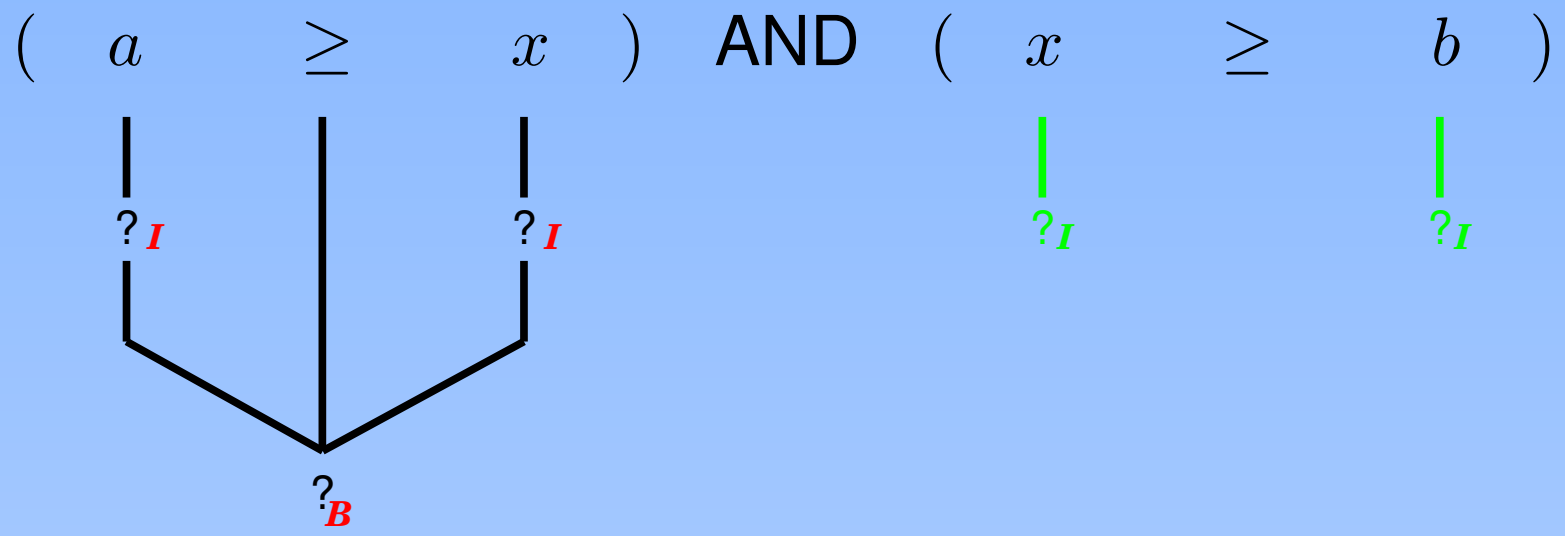
- Jak wyglądałoby wyliczanie powyższego wyrażenia w przypadku a, b i x typu BOOLEAN?

Drzewa wyliczania wartości wyrażeń

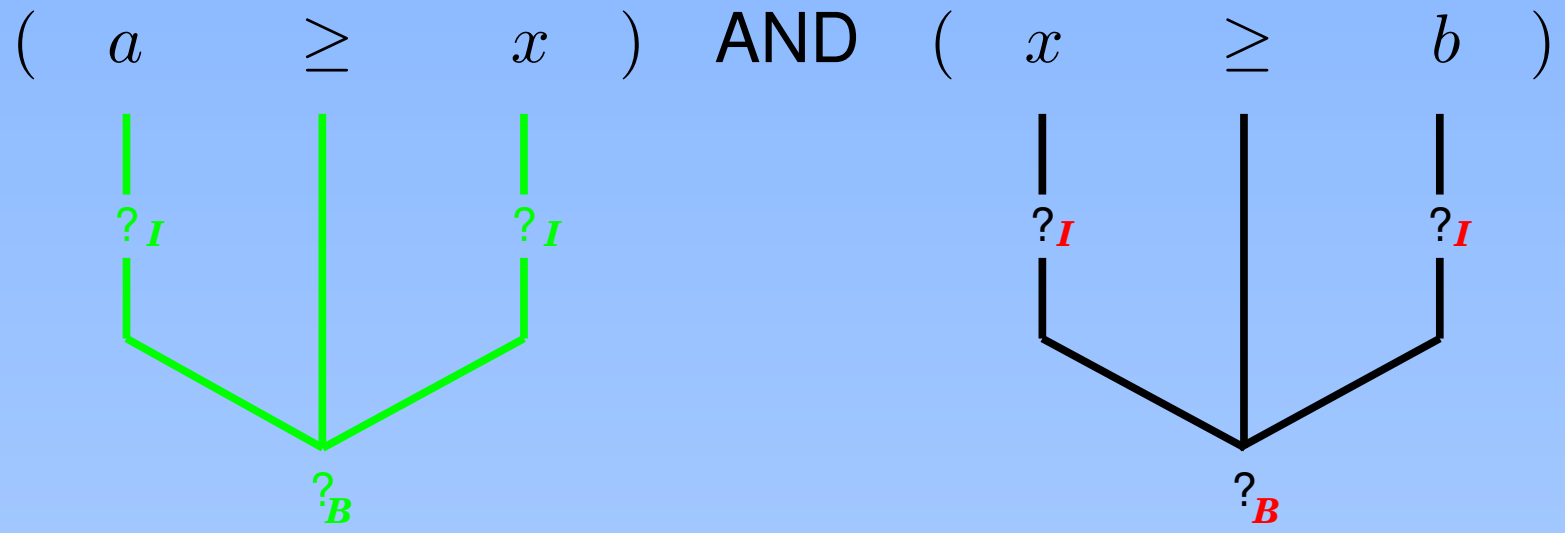
$$(\quad a \quad \geq \quad x \quad) \text{ AND } (\quad x \quad \geq \quad b \quad)$$

Drzewa wyliczania wartości wyrażeń

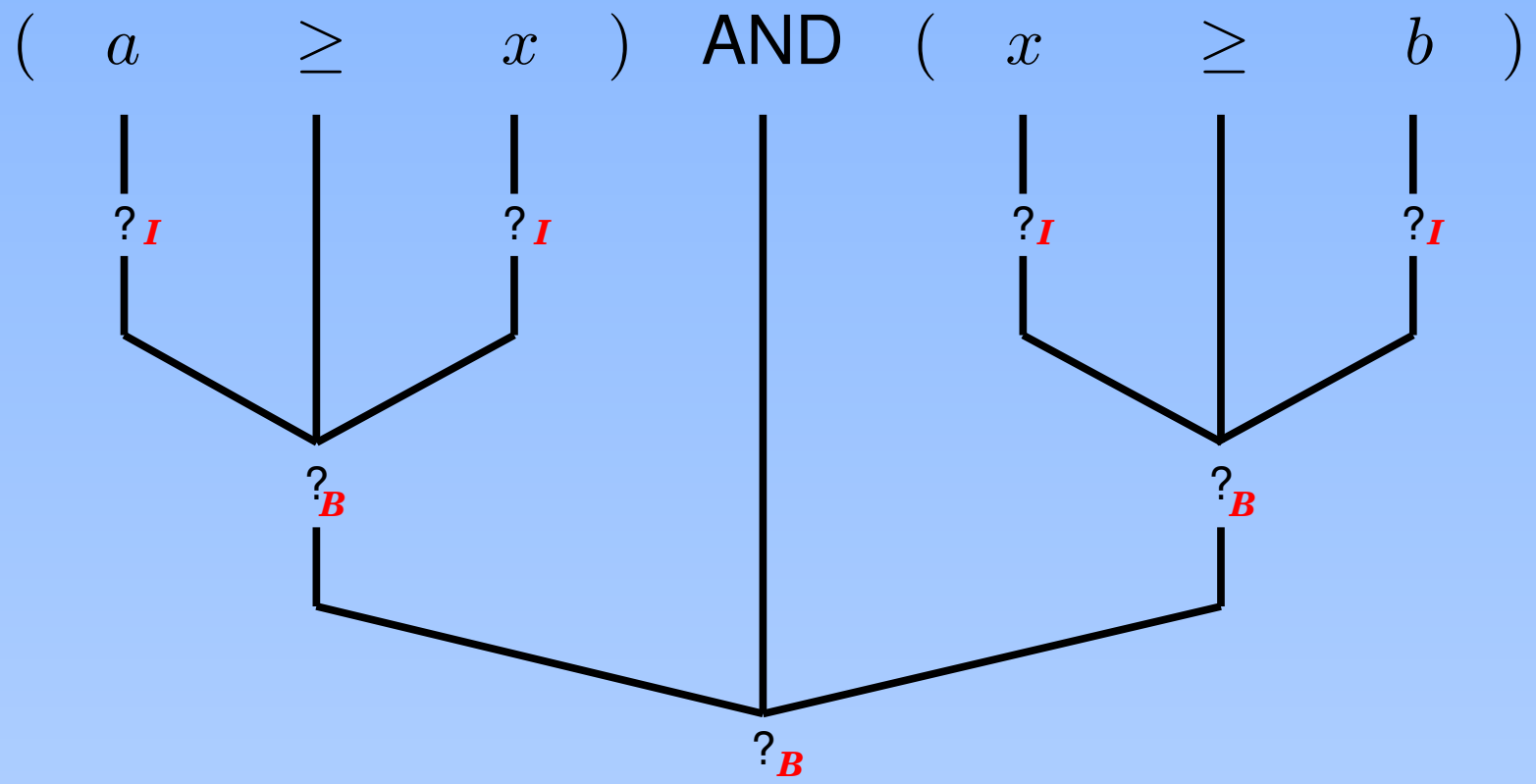
Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń



Drzewa wyliczania wartości wyrażeń



W końcu!

Operatory DIV i MOD

Dla dowolnch x i y zachodzi

$$x = y * (x \texttt{ DIV } y) + (x \texttt{ MOD } y)$$

Operatory DIV i MOD

Dla dowolnch x i y zachodzi

$$x = y * (x \texttt{ DIV } y) + (x \texttt{ MOD } y)$$

Przykładowo procentowy wynik wyborów:

```
(IloscZa * 100) DIV IloscGlosow
```

Operatory DIV i MOD

Dla dowolnch x i y zachodzi

$$x = y * (x \texttt{ DIV } y) + (x \texttt{ MOD } y)$$

Przykładowo procentowy wynik wyborów:

```
(IloscZa * 100) DIV IloscGlosow
```

ale nie

```
IloscZa DIV IloscGlosow * 100
```

Operatory DIV i MOD

k-ta cyfra w rozwinięciu liczby n określonej w układzie pozycyjnym o podstawie p

$$n_k = \left(n \ \texttt{DIV} \ p^{(k-1)}\right) \ \texttt{MOD} \ p$$

Operatory DIV i MOD

k-ta cyfra w rozwinięciu liczby n określonej w układzie pozycyjnym o podstawie p

$$n_k = \left(n \ \texttt{DIV} \ p^{(k-1)}\right) \ \texttt{MOD} \ p$$

w szczególności w układzie dziesiętnym

$$n_k = \left(n \ \texttt{DIV} \ 10^{(k-1)}\right) \ \texttt{MOD} \ 10$$

Liczymy więc przykładowo

$$(853 \texttt{ DIV } 1) \texttt{ MOD } 10 = 3$$

Liczymy więc przykładowo

$$(853 \ \texttt{DIV} \ 1) \ \texttt{MOD} \ 10 = 3$$
$$(853 \ \texttt{DIV} \ 10) \ \texttt{MOD} \ 10 = 5$$

Liczymy więc przykładowo

$$\begin{aligned}
(853 \ \texttt{DIV} \ 1) \ \texttt{MOD} \ 10 &= \boxed{3} \\
(853 \ \texttt{DIV} \ 10) \ \texttt{MOD} \ 10 &= \boxed{5} \\
(853 \ \texttt{DIV} \ 100) \ \texttt{MOD} \ 10 &= \boxed{8}
\end{aligned}$$

Liczymy więc przykładowo

$$\begin{aligned}
(853 \ \texttt{DIV} \ \ 1) \ \texttt{MOD} \ 10 &= \boxed{3} \\
(853 \ \texttt{DIV} \ 10) \ \texttt{MOD} \ 10 &= 5 \\
(853 \ \texttt{DIV} \ 100) \ \texttt{MOD} \ 10 &= 8
\end{aligned}$$

i piszemy program

```
FUNCTION Potega(a, b : INTEGER) : INTEGER;
BEGIN
  m := 1;
  FOR i := 1 TO b DO
    m := m * a;
  Potega := m;
END;
                                    (* Algorytm 1 *)
```

Liczymy więc przykładowo

$$\begin{aligned}
(853 \ \texttt{DIV} \ 1) \ \texttt{MOD} \ 10 &= \boxed{3} \\
(853 \ \texttt{DIV} \ 10) \ \texttt{MOD} \ 10 &= \boxed{5} \\
(853 \ \texttt{DIV} \ 100) \ \texttt{MOD} \ 10 &= \boxed{8}
\end{aligned}$$

i piszemy program

```
FUNCTION Potega(a, b : INTEGER) : INTEGER;
BEGIN
  m := 1;
  FOR i := 1 TO b DO
    m := m * a;
  Potega := m;
END;
  ...                                    (* Algorytm 1 *)
  NrCyfry := 1;
  WHILE ??? DO
  BEGIN
    Cyfra := Liczba DIV Potega(10,NrCyfry-1) MOD 10;
    NrCyfry := NrCyfry + 1
  END;
```

cyfr	podstawień	mnożeń	dodawań	porównań
3	30	12	6	?

cyfr	podstawień	mnożeń	dodawań	porównań
3	30	12	6	?
6	69	33	12	?

cyfr	podstawień	mnożeń	dodawań	porównań
3	30	12	6	?
6	69	33	12	?
n	n^2+5n+3	$\frac{n^2}{2}+\frac{5n}{2}$	$2n$	?

Napisaliśmy:

```
FUNCTION Potega(a, b : INTEGER) : INTEGER;
BEGIN
  m := 1;
  FOR i := 1 TO b DO
    m := m * a;
  Potega := m;
END;
  ...                          (* Algorytm 1 *)
  NrCyfry := 1;
  WHILE ??? DO
  BEGIN
    Cyfra := Liczba DIV
               Potega(10,NrCyfry-1) MOD 10;
    NrCyfry := NrCyfry + 1
```

Napisaliśmy:

```
FUNCTION Potega(a, b : INTEGER) : INTEGER;
BEGIN
  m := 1;
  FOR i := 1 TO b DO
    m := m * a;
  Potega := m;
END;
  ...                          (* Algorytm 1 *)
  NrCyfry := 1;
  WHILE ??? DO
  BEGIN
    Cyfra := Liczba DIV
                Potega(10,NrCyfry-1) MOD 10;
    NrCyfry := NrCyfry + 1
  END;
```

Można prościej:

```
...                            (* Algorytm 2 *)
Dzielnik := 1;
WHILE Dzielnik < Liczba DO
BEGIN
Cyfra := Liczba DIV
             Dzielnik MOD 10;
Dzielnik := Dzielnik * 10;
END;
```

Napisaliśmy:

```
FUNCTION Potega(a, b : INTEGER) : INTEGER;
BEGIN
  m := 1;
  FOR i := 1 TO b DO
    m := m * a;
  Potega := m;
END;
  ...                          (* Algorytm 1 *)
  NrCyfry := 1;
  WHILE ??? DO
  BEGIN
    Cyfra := Liczba DIV
                Potega(10,NrCyfry-1) MOD 10;
    NrCyfry := NrCyfry + 1
  END;
```

Można prościej:

```
...                            (* Algorytm 2 *)
Dzielnik := 1;
WHILE Dzielnik < Liczba DO
BEGIN
Cyfra := Liczba DIV
             Dzielnik MOD 10;
Dzielnik := Dzielnik * 10;
END;
```

Napisaliśmy:

```
FUNCTION Potega(a, b : INTEGER) : INTEGER;
BEGIN
  m := 1;
  FOR i := 1 TO b DO
    m := m * a;
  Potega := m;
END;
  ...                          (* Algorytm 1 *)
  NrCyfry := 1;
  WHILE ??? DO
  BEGIN
    Cyfra := Liczba DIV
                Potega(10,NrCyfry-1) MOD 10;
    NrCyfry := NrCyfry + 1
  END;
```

Można prościej:

```
...                            (* Algorytm 2 *)
Dzielnik := 1;
WHILE Dzielnik < Liczba DO
BEGIN
Cyfra := Liczba DIV
             Dzielnik MOD 10;
Dzielnik := Dzielnik * 10;
END;
```

Napisaliśmy:

```
FUNCTION Potega(a, b : INTEGER) : INTEGER;
BEGIN
  m := 1;
  FOR i := 1 TO b DO
    m := m * a;
  Potega := m;
END;
  ...                          (* Algorytm 1 *)
  NrCyfry := 1;
  WHILE ??? DO
  BEGIN
    Cyfra := Liczba DIV
                Potega(10,NrCyfry-1) MOD 10;
    NrCyfry := NrCyfry + 1
  END;
```

Można prościej:

```
...                            (* Algorytm 2 *)
Dzielnik := 1;
WHILE Dzielnik < Liczba DO
BEGIN
Cyfra := Liczba DIV
              Dzielnik MOD 10;
Dzielnik := Dzielnik * 10;
END;
```

Obecnie:

cyfr	podstawień	mnożeń	dodawań	porównań
3	7	9	0	4

Obecnie:

cyfr	podstawień	mnożeń	dodawań	porównań
3	7	9	0	4
6	13	18	0	7

Obecnie:

cyfr	podstawień	mnożeń	dodawań	porównań
3	7	9	0	4
6	13	18	0	7
n	$2n+1$	$3n$	0	$n+1$

Obecnie:

cyfr	podstawień	mnożeń	dodawań	porównań
3	7	9	0	4
6	13	18	0	7
n	$2n+1$	$3n$	0	$n+1$

A mieliśmy:

cyfr	podstawień	mnożeń	dodawań	porównań
3	30	12	6	?
6	69	33	12	?
n	n^2+5n+3	$\frac{n^2}{2}+\frac{5n}{2}$	$2n$	?

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN              (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1		

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2		

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3		

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3	8	

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3	8	0

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3	8	0
4		

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3	8	0
4	STOP	

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3	8	0
4	STOP	

Teraz:

cyfr	podstawień	mnożeń	dodawań	porównań
3	6	6	0	4

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3	8	0
4	STOP	

Teraz:

cyfr	podstawień	mnożeń	dodawań	porównań
3	6	6	0	4
6	12	12	0	7

Lub jeszcze prościej:

```
WHILE Liczba > 0 DO
BEGIN                (* Algorytm 3 *)
  Cyfra := Liczba MOD 10;
  Liczba := Liczba DIV 10;
END;
```

co daje

krok	Cyfra	Liczba
0	—	853
1	3	85
2	5	8
3	8	0
4	STOP	

Teraz:

cyfr	podstawień	mnożeń	dodawań	porównań
3	6	6	0	4
6	12	12	0	7
n	$2n$	$2n$	0	$n+1$

Złożoność obliczeniowa - zarys

Podsumowując:

algorytm	podstawień	mnożeń	dodawań	porównań	złożoność
1	$n^2 + 5n + 3$	$\frac{n^2}{2} + \frac{5n}{2}$	$2n$	?	O(n^2)

Złożoność obliczeniowa - zarys

Podsumowując:

algorytm	podstawień	mnożeń	dodawań	porównań	złożoność
1	$n^2 + 5n + 3$	$\frac{n^2}{2} + \frac{5n}{2}$	$2n$	?	O(n^2)
2	$2n + 1$	$3n$	0	$n + 1$	O(n)

Złożoność obliczeniowa - zarys

Podsumowując:

algorytm	podstawień	mnożeń	dodawań	porównań	złożoność
1	n^2+5n+3	$\frac{n^2}{2}+\frac{5n}{2}$	$2n$	?	O(n^2)
2	$2n+1$	$3n$	0	$n+1$	O(n)
3	$2n$	$2n$	0	$n+1$	O(n)

Złożoność obliczeniowa - zarys

Podsumowując:

algorytm	podstawień	mnożeń	dodawań	porównań	złożoność
1	n^2+5n+3	$\frac{n^2}{2}+\frac{5n}{2}$	$2n$	?	O(n^2)
2	$2n+1$	$3n$	0	$n+1$	O(n)
3	$2n$	$2n$	0	$n+1$	O(n)

dla $n=100$

algorytm	podstawień	mnożeń	dodawań	porównań
1	10503	5250	200	?

Złożoność obliczeniowa - zarys

Podsumowując:

algorytm	podstawień	mnożeń	dodawań	porównań	złożoność
1	n^2+5n+3	$\frac{n^2}{2}+\frac{5n}{2}$	$2n$	?	O(n^2)
2	$2n+1$	$3n$	0	$n+1$	O(n)
3	$2n$	$2n$	0	$n+1$	O(n)

dla $n=100$

algorytm	podstawień	mnożeń	dodawań	porównań
1	10503	5250	200	?
2	201	300	0	101

Złożoność obliczeniowa - zarys

Podsumowując:

algorytm	podstawień	mnożeń	dodawań	porównań	złożoność
1	n^2+5n+3	$\frac{n^2}{2}+\frac{5n}{2}$	$2n$	?	O(n^2)
2	$2n+1$	$3n$	0	$n+1$	O(n)
3	$2n$	$2n$	0	$n+1$	O(n)

dla $n=100$

algorytm	podstawień	mnożeń	dodawań	porównań
1	10503	5250	200	?
2	201	300	0	101
3	200	200	0	101

Instrukcja warunkowa IF-THEN

```
instrukcja-IF = "IF" wyrazenie-logiczne "THEN" instrukcja
              | "IF" wyrazenie-logiczne "THEN" instrukcja
                "ELSE" instrukcja.
```

Instrukcja warunkowa IF-THEN

```
instrukcja-IF = "IF" wyrazenie-logiczne "THEN" instrukcja
              | "IF" wyrazenie-logiczne "THEN" instrukcja
                "ELSE" instrukcja.
```

Porządkowanie liczb

```
IF x > y THEN
BEGIN
  Wieksza := x;
  Mniejsza := y
END
ELSE
BEGIN
  Wieksza := y;
  Mniejsza := x
END;
```

Instrukcja warunkowa IF-THEN

```
instrukcja-IF = "IF" wyrazenie-logiczne "THEN" instrukcja
              | "IF" wyrazenie-logiczne "THEN" instrukcja
                "ELSE" instrukcja.
```

Porządkowanie liczb

```
IF x > y THEN
BEGIN
  Wieksza := x;
  Mniejsza := y
END
ELSE
BEGIN
  Wieksza := y;
  Mniejsza := x
END;
```

Można też tak ;-)

```
IF x>y THEN BEGIN Wieksza:=x;
Mniejsza:=y END ELSE BEGIN Wieksza:=y;
Mniejsza:=x END;
```

Instrukcja IF-THEN — przykłady

Porządkowanie liczb — inny przykład

Instrukcja IF-THEN — przykłady

Porządkowanie liczb — inny przykład

```
IF x > y THEN
BEGIN
  x := y;
  y := x
END;
```

Instrukcja IF-THEN — przykłady

Porządkowanie liczb — inny przykład

```
IF x > y THEN
BEGIN
  x := y;
  y := x
END;
```

```
IF x > y THEN
BEGIN
  tmp := x
  x := y;
  y := tmp
END;
```

Instrukcja IF-THEN — przykłady

Składanie instrukcji warunkowych

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1
  ELSE
    instrukcja2
```

Instrukcja IF-THEN — przykłady

Składanie instrukcji warunkowych

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1
  ELSE
    instrukcja2
```

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1;
ELSE
  instrukcja2
```

Instrukcja IF-THEN — przykłady

Składanie instrukcji warunkowych

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1
  ELSE
    instrukcja2
```

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1;
ELSE
  instrukcja2
```

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1
  ELSE
    instrukcja2
ELSE
  instrukcja3
```

Instrukcja IF-THEN — przykłady

Składanie instrukcji warunkowych

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1
  ELSE
    instrukcja2
```

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1;
ELSE
  instrukcja2
```

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1
  ELSE
    instrukcja2
ELSE
  instrukcja3
```

```
IF warunek1 THEN
  IF warunek2 THEN
    instrukcja1
  ELSE
    instrukcja2;
ELSE
  instrukcja3
```

Instrukcja IF-THEN — przykłady

```
IF TRUE THEN IF FALSE THEN WRITELN('false')
ELSE WRITELN('true');
```

Instrukcja IF-THEN — przykłady

```
IF TRUE THEN IF FALSE THEN WRITELN('false')
ELSE WRITELN('true');
IF FALSE THEN IF FALSE THEN WRITELN('false')
ELSE WRITELN('true');
```

Instrukcja IF-THEN — przykłady

```
IF TRUE THEN IF FALSE THEN WRITELN('false')
ELSE WRITELN('true');
IF FALSE THEN IF FALSE THEN WRITELN('false')
ELSE WRITELN('true');
IF FALSE THEN BEGIN IF FALSE THEN WRITELN('false')
END ELSE WRITELN('true');
```

Instrukcja IF-THEN — przykłady

```
IF TRUE THEN IF FALSE THEN WRITELN('false')
ELSE WRITELN('true');
IF FALSE THEN IF FALSE THEN WRITELN('false')
ELSE WRITELN('true');
IF FALSE THEN BEGIN IF FALSE THEN WRITELN('false')
END ELSE WRITELN('true');
IF FALSE THEN BEGIN IF TRUE THEN WRITELN('true')
END ELSE WRITELN('false');
```

Instrukcja IF-THEN — przykłady

```
BEGIN
  ...
  a:=10; b:=5; c:=20; d:='@'; e:=TRUE;

  IF NOT (a MOD b = c) AND e OR (d = '#') THEN
    a := 2 * a
  ELSE IF e OR (d = '&') THEN
    d := '#'
  ELSE
    b:= a + c;
  e := FALSE;

  WRITELN ((a = c) AND (e = FALSE));
  WRITELN ((a = 20) AND (d = '#'));...
END.
```

Instrukcja pętli WHILE-DO

```
instrukcja-WHILE = "WHILE" wyrazenie-logiczne
                   "DO" instrukcja.
```

Instrukcja pętli WHILE-DO

```
instrukcja-WHILE = "WHILE" wyrazenie-logiczne
                   "DO" instrukcja.
```

Sprawdzanie czy liczba jest pierwsza

```
Pierwsza := TRUE;
Dzielnik := 2;
WHILE Dzielnik < N DO
BEGIN
  IF (N MOD Dzielnik) = 0 THEN
    Pierwsza := FALSE;
  Dzielnik := Dzielnik + 1;
END;
```

Instrukcja pętli FOR-TO-DO

```
instrukcja-FOR = "FOR" zmienna ":="
       wartosc-poczatkowa ("TO" | "DOWNTO")
              wartosc-koncowa "DO" instrukcja.
wartosc-poczatkowa = wyrazenie.
wartosc-koncowa = wyrazenie.
```

Instrukcja pętli FOR-TO-DO

```
instrukcja-FOR = "FOR" zmienna ":="
       wartosc-poczatkowa ("TO" | "DOWNTO")
             wartosc-koncowa "DO" instrukcja.
wartosc-poczatkowa = wyrazenie.
wartosc-koncowa = wyrazenie.
```

Sprawdzanie czy liczba jest pierwsza

```
JestPierwsza := TRUE;
FOR Dzielnik := 2 TO N-1 DO
  IF (N MOD Dzielnik) = 0 THEN
    JestPierwsza := FALSE;
```

Zachowanie instrukcji FOR-TO-DO

- zmienna sterująca musi być zadeklarowana w programie,

Zachowanie instrukcji FOR-TO-DO

- zmienna sterująca musi być zadeklarowana w programie,
- wartości początkowa i końcowa są obliczane dokładnie raz, przed rozpoczęciem realizacji pętli,

Zachowanie instrukcji FOR-TO-DO

- zmienna sterująca musi być zadeklarowana w programie,
- wartości początkowa i końcowa są obliczane dokładnie raz, przed rozpoczęciem realizacji pętli,
- zmiennej sterującej przypisana zostaje wartość początkowa,

Zachowanie instrukcji FOR-TO-DO

- zmienna sterująca musi być zadeklarowana w programie,
- wartości początkowa i końcowa są obliczane dokładnie raz, przed rozpoczęciem realizacji pętli,
- zmiennej sterującej przypisana zostaje wartość początkowa,
- wartość zmiennej sterującej jest zwiększana (a w przypadku wystąpienia słowa DOWNTO zmniejszana), po każdym wykonaniu instrukcji wewnętrznej,

Zachowanie instrukcji FOR-TO-DO

- zmienna sterująca musi być zadeklarowana w programie,
- wartości początkowa i końcowa są obliczane dokładnie raz, przed rozpoczęciem realizacji pętli,
- zmiennej sterującej przypisana zostaje wartość początkowa,
- wartość zmiennej sterującej jest zwiększana (a w przypadku wystąpienia słowa DOWNTO zmniejszana), po każdym wykonaniu instrukcji wewnętrznej,
- wartość zmiennej sterującej nie może być zmieniana przez instrukcję wewnętrzną,

Zachowanie instrukcji FOR-TO-DO

- zmienna sterująca musi być zadeklarowana w programie,
- wartości początkowa i końcowa są obliczane dokładnie raz, przed rozpoczęciem realizacji pętli,
- zmiennej sterującej przypisana zostaje wartość początkowa,
- wartość zmiennej sterującej jest zwiększana (a w przypadku wystąpienia słowa DOWNTO zmniejszana), po każdym wykonaniu instrukcji wewnętrznej,
- wartość zmiennej sterującej nie może być zmieniana przez instrukcję wewnętrzną,
- jeżeli wartość zmiennej sterującej jest większa (mniejsza w przypadku DOWNTO) od wartości końcowej, to instrukcja wewnętrzna nie jest wykonywana,

Zachowanie instrukcji FOR-TO-DO

- zmienna sterująca musi być zadeklarowana w programie,
- wartości początkowa i końcowa są obliczane dokładnie raz, przed rozpoczęciem realizacji pętli,
- zmiennej sterującej przypisana zostaje wartość początkowa,
- wartość zmiennej sterującej jest zwiększana (a w przypadku wystąpienia słowa DOWNTO zmniejszana), po każdym wykonaniu instrukcji wewnętrznej,
- wartość zmiennej sterującej nie może być zmieniana przez instrukcję wewnętrzną,
- jeżeli wartość zmiennej sterującej jest większa (mniejsza w przypadku DOWNTO) od wartości końcowej, to instrukcja wewnętrzna nie jest wykonywana,
- po zakończeniu wykonywania instrukcji FOR-TO-DO wartość zmiennej sterującej jest nieokreślona.

Równoważność instrukcji WHILE-DO i FOR-TO-DO

```
{ FOR I := E0 TO Ek DO X; }
```

Równoważność instrukcji WHILE-DO i FOR-TO-DO

```
{ FOR I := E0 TO Ek DO X; }
```

```
I := E0;
```

Równoważność instrukcji WHILE-DO i FOR-TO-DO

```
{ FOR I := E0 TO Ek DO X; }

I := E0;
WHILE I <= Ek DO
```

Równoważność instrukcji WHILE-DO i FOR-TO-DO

```
{ FOR I := E0 TO Ek DO X; }

I := E0;
WHILE I <= Ek DO
BEGIN
  X;

END;
```

Równoważność instrukcji WHILE-DO i FOR-TO-DO

```
{ FOR I := E0 TO Ek DO X; }

I := E0;
WHILE I <= Ek DO
BEGIN
  X;
  I := I + 1;
END;
```

Instrukcja pętli FOR-TO-DO — inny przykład

```
PROGRAM Przyklad(INPUT,OUTPUT);
TYPE  Miesiace = (styczen, luty, marzec, kwiecien, maj,
                  czerwiec, lipiec, sierpien, wrzesien,
                  pazdziernik, listopad, grudzien);
      Dni = 1..31;
VAR   m : Miesiace;
      d : Dni;
BEGIN
  FOR m := styczen TO grudzien DO
    FOR d := 1 TO DniWMiesiacu(m) DO
      IF SwietoPanstwowe(m,d) THEN
        WRITELN ('Swieto dnia ',d,' miesiaca ',ORD(m));
END.
```

Instrukcja pętli FOR-TO-DO — inny przykład

```
PROGRAM Przyklad(INPUT,OUTPUT);
TYPE  Miesiace = (styczen, luty, marzec, kwiecien, maj,
                  czerwiec, lipiec, sierpien, wrzesien,
                  pazdziernik, listopad, grudzien);
      Dni = 1..31;
VAR  m : Miesiace;
     d : Dni;
BEGIN
  FOR m := styczen TO grudzien DO
    FOR d := 1 TO DniWMiesiacu(m) DO
      IF SwietoPanstwowe(m,d) THEN
        WRITELN ('Swieto dnia ',d,' miesiaca ',ORD(m));
END.
```

Instrukcja pętli FOR-TO-DO — inny przykład

```
PROGRAM Przyklad(INPUT,OUTPUT);
TYPE  Miesiace = (styczen, luty, marzec, kwiecien, maj,
                  czerwiec, lipiec, sierpien, wrzesien,
                  pazdziernik, listopad, grudzien);
      Dni = 1..31;
VAR   m : Miesiace;
      d : Dni;
BEGIN
  FOR m := styczen TO grudzien DO
    FOR d := 1 TO DniWMiesiacu(m) DO
      IF SwietoPanstwowe(m,d) THEN
        WRITELN ('Swieto dnia ',d,' miesiaca ',ORD(m));
END.
```

Instrukcja pętli FOR-TO-DO — inny przykład

```
PROGRAM Przyklad(INPUT,OUTPUT);
TYPE  Miesiace = (styczen, luty, marzec, kwiecien, maj,
                  czerwiec, lipiec, sierpien, wrzesien,
                  pazdziernik, listopad, grudzien);
      Dni = 1..31;
VAR   m : Miesiace;
      d : Dni;
BEGIN
  FOR m := styczen TO grudzien DO
    FOR d := 1 TO DniWMiesiacu(m) DO
      IF SwietoPanstwowe(m,d) THEN
        WRITELN ('Swieto dnia ',d,' miesiaca ',ORD(m));
END.
```

Instrukcja pętli FOR-TO-DO — inny przykład

```
PROGRAM Przyklad(INPUT,OUTPUT);
TYPE  Miesiace = (styczen, luty, marzec, kwiecien, maj,
                  czerwiec, lipiec, sierpien, wrzesien,
                  pazdziernik, listopad, grudzien);
      Dni = 1..31;
VAR   m : Miesiace;
      d : Dni;
BEGIN
  FOR m := styczen TO grudzien DO
    FOR d := 1 TO DniWMiesiacu(m) DO
      IF SwietoPanstwowe(m,d) THEN
        WRITELN ('Swieto dnia ',d,' miesiaca ',ORD(m));
END.
```

Instrukcja pętli FOR-TO-DO — inny przykład

```
PROGRAM Przyklad(INPUT,OUTPUT);
TYPE  Miesiace = (styczen, luty, marzec, kwiecien, maj,
                  czerwiec, lipiec, sierpien, wrzesien,
                  pazdziernik, listopad, grudzien);
      Dni = 1..31;
VAR   m : Miesiace;
      d : Dni;
BEGIN
  FOR m := styczen TO grudzien DO
    FOR d := 1 TO DniWMiesiacu(m) DO
      IF SwietoPanstwowe(m,d) THEN
        WRITELN ('Swieto dnia ',d,' miesiaca ',ORD(m));
END.
```

Instrukcja pętli FOR-TO-DO — inny przykład

```
PROGRAM Przyklad(INPUT,OUTPUT);
TYPE  Miesiace = (styczen, luty, marzec, kwiecien, maj,
                  czerwiec, lipiec, sierpien, wrzesien,
                  pazdziernik, listopad, grudzien);
      Dni = 1..31;
VAR   m : Miesiace;
      d : Dni;
BEGIN
  FOR m := styczen TO grudzien DO
    FOR d := 1 TO DniWMiesiacu(m) DO
      IF SwietoPanstwowe(m,d) THEN
        WRITELN ('Swieto dnia ',d,' miesiaca ',ORD(m));
END.
```

Indeks

- Inny przykład
- Zachowanie instrukcji

- Równoważność instrukcji `WHILE-DO` i `FOR-TO-DO`