

Wstęp do informatyki

Wykład 12

Gramatyki, odwrotna notacja polska

Temat wykładu

- Gramatyki bezkontekstowe jako narzędzie do opisu składni języków programowania
- Odwrotna notacja polska (ONP):
 - Konwersja wyrażeń do postaci ONP.
 - Wyznaczanie wartości wyrażenia w postaci ONP.

Gramatyki bezkontekstowe: motywacja

Definicja języka programowania

Składnia – reguły definiujące formalnie poprawne programy i konstrukcje językowe (poprawne składniowo/syntaktycznie)

Semantyka – znaczenie konstrukcji językowych:

- semantyka operacyjna
- semantyka denotacyjna

Składnia języka

Przykład. Składnia wyrażeń arytmetycznych:

- argumenty: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
- operatory binarne (dwuargumentowe): +, *, —
- nawiasy: (,)

Poprawne:

$5 + 2 * 4 + 0 * 7$

$((5 + 2) * 4) + 0 * 7$

Niepoprawne:

$5 + 2 * 4 + 0 *$

$((5 + 2 * 4) + (0 * 7$

UWAGA: przyjmujemy dla uproszczenia, że argumenty to tylko **cyfry** (np. 723 nie może być argumentem)

Składnia wyrażeń arytmetycznych

Def. Poprawne wyrażenia

- Cyfra jest poprawnym wyrażeniem
- Jeśli $W1$ i $W2$ to poprawne wyrażenia, to poprawnymi wyrażeniami są również:
 - $W1 + W2$
 - $W1 * W2$
 - $(W1)$

Składnia wyrażeń arytmetycznych

Definicja za pomocą „reguł przepisywania”:

S – symbol startowy

$$S \rightarrow 0, S \rightarrow 1, \dots, S \rightarrow 9$$

$$S \rightarrow S + S$$

$$S \rightarrow S * S$$

$$S \rightarrow (S)$$

Składnia wyrażeń arytmetycznych

Stosowanie reguł przepisywania

(**wyprowadzenie**):

$$S \Rightarrow S * S \Rightarrow S + S * S \Rightarrow$$

$$(S) + S * S \Rightarrow$$

$$(S * S) + S * S \Rightarrow$$

$$(5 * S) + S * S \Rightarrow$$

$$(5 * 4) + S * S \Rightarrow$$

$$(5 * 4) + 1 * S \Rightarrow$$

$$(5 * 4) + 1 * 2$$

REGUŁY:

$$S \rightarrow 0, \dots, S \rightarrow 9$$

$$S \rightarrow S + S$$

$$S \rightarrow S * S$$

$$S \rightarrow (S)$$

Składnia wyrażeń arytmetycznych

Stosowanie **reguł przepisывania**

(**wyprowadzenie**):

- startujemy od symbolu startowego S
- kontynuujemy przepisывanie aż do uzyskania napisu składającego się wyłącznie z cyfr, operatorów i nawiasów

Składnia wyrażeń arytmetycznych

Napisy, których nie można uzyskać stosując reguły przepisывania:

- $(5 * 4))$

Narusza zasadę:

parzysta liczba

nawiasów (wstawiane parami)

- $)(5 + 4)($

Narusza zasadę: każdy nawias

zamykający jest wstawiany wraz z

poprzedzającym go nawiasem otwierającym „do pary”

REGUŁY:

$S \rightarrow 0, \dots, S \rightarrow 9$

$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Napisy, języki i gramatyki formalnie

Napisy i języki

Pojęcia:

- A – skończony zbiór (alfabet)
- Słowo/napis – skończony ciąg elementów z A (może być pusty)
- A^* - zbiór wszystkich słów
- L - język nad alfabetem A to podzbiór A^*

Przykład:

- $A = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *, (,) \}$
- A^* - zbiór wszystkich napisów ze znakami z A , np. $0*1+, ()$
- L – napisy odpowiadające poprawnym wyrażeniom

Napisy i języki

Pojęcia:

- A – skończony zbiór (alfabet)
- słowo – skończony ciąg elementów z A (może być pusty)
- A^* - zbiór wszystkich słów
- L - język nad alfabetem A to podzbiór A^*

Przykład:

- A – zbiór liter alfabetu polskiego i znaków przestankowych
- A^* - zbiór napisów ze znakami z A , np. $aa;.że$
- L – poprawne składniowo teksty w języku polskim

Napisy i języki

Oznaczenia:

- ε - słowo puste
- a^* - zbiór napisów złożonych z dowolnej liczby powtórzeń litery a czyli $\{\varepsilon, a, aa, aaa, aaaa, \dots\}$
- a^+ - zbiór napisów złożonych z dodatniej liczby powtórzeń litery a czyli $\{a, aa, aaa, aaaa, \dots\}$
- a^k – napis złożony z k powtórzeń litery a

Gramatyka bezkontekstowa

Gramatyka bezkontekstowa $G(N, T, P, S)$ jest zdefiniowana poprzez:

- N – zbiór nieterminali (symbole pomocnicze)
- T – zbiór terminali (symbole podstawowe)
- P – zbiór produkcji – podzbiór $N \times (N \cup T)^*$
- S – symbol startowy, element N

Gramatyka bezkontekstowa

Konwencje:

- **Duże** litery alfabetu – **nieterminale** (elementy zbioru **N**)
- **Małe** litery i inne znaki – **terminale** (zbiór **T**)

Gramatyka bezkontekstowa

Gramatyka bezkontekstowa $G(N,T,P,S)$ jest zdefiniowana poprzez:

- N – zbiór nieterminali (symbole pomocnicze)
- T – zbiór terminali (symbole podstawowe)
- P – zbiór produkcji – podzbiór $N \times (N \cup T)^*$
- S – symbol startowy, element N

Przykład:

- $N = \{ S \}$
- $T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *, (,) \}$
- $P = \{ S \rightarrow 0, \dots, S \rightarrow 9, S \rightarrow S + S, S \rightarrow S * S, S \rightarrow (S) \}$
- symbol startowy: S

Gramatyka bezkontekstowa

Wyprowadzenie (formalnie) w gramatyce

$G(N, T, P, S)$ to ciąg napisów

$$x_1 \Rightarrow x_2 \Rightarrow \dots \Rightarrow x_n$$

taki, że

- $x_1 = S$
- $x_n \in T^*$
- x_{i+1} powstaje przez zastosowanie produkcji w x_i (czyli zastąpienie lewej strony produkcji prawą)

Gramatyka - wyprowadzenie

Gramatyka:

- $N = \{ S \}$
- $T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *, (,) \}$
- $P = \{ S \rightarrow 0, \dots, S \rightarrow 9, S \rightarrow S + S, S \rightarrow S * S, S \rightarrow (S) \}$
- symbol startowy: S

Wyprowadzenie:

$$S \Rightarrow S * S \Rightarrow$$

$$\underline{S} + S * S \Rightarrow$$

$$(\underline{S}) + S * S \Rightarrow$$

$$(\underline{S * S}) + S * S \Rightarrow$$

$$(\underline{5} * S) + S * S \Rightarrow$$

$$(5 * \underline{4}) + S * S \Rightarrow$$

$$(5 * 4) + \underline{1} * S \Rightarrow$$

$$(5 * 4) + 1 * \underline{2}$$

Język a gramatyka bezkontekstowa

Gramatyka bezkontekstowa $G(N, T, P, S)$ definiuje język $L(G)$ złożony ze wszystkich słów nad alfabetem $w \in T^*$, które można uzyskać przy pomocy **wyprowadzeń**.

Przykład:

- $N = \{ S \}$
- $T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *, (,) \}$
- $P = \{ S \rightarrow 0, \dots, S \rightarrow 9, S \rightarrow S + S, S \rightarrow S * S, S \rightarrow (S) \}$
- symbol startowy: S
- $L(G)$ – zbiór poprawnych wyrażeń

Gramatyka bezkontekstowa - przykłady

Przykład. $G(N,T,P,S)$

- $N = \{ S \}$
- $T = \{ a, b \}$

$P = \{$

- $S \rightarrow aSb$

- $S \rightarrow \varepsilon$

$\}$

Przykłady napisów

$aaaaabb \notin L(G)$

$ab \in L(G)$

$ba \notin L(G)$

$aabb \in L(G)$

$L(G)$ – zbiór słów postaci $a^k b^k$,
gdzie $k \geq 0$

Gramatyka bezkontekstowa - przykłady

Przykład. Liczba ze znakiem

$G(N, T, P, S)$

- $N = \{ S, Z, C, X \}$
- $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

$P = \{$

- $S \rightarrow Z C X$
- $Z \rightarrow \varepsilon$
- $Z \rightarrow -$
- $C \rightarrow 0, C \rightarrow 1, \dots, C \rightarrow 9$
- $X \rightarrow \varepsilon$
- $X \rightarrow C X$

$\}$

$L(G)$ – liczby całkowite

Gramatyka bezkontekstowa - przykłady

Przykład. $G(N, T, P, S)$

- $N = \{ S, A, B, C \}$

- $T = \{ a, b \}$

$P = \{$

- $S \rightarrow A b B$

- $A \rightarrow aA$

- $A \rightarrow \varepsilon$ (słowo puste)

- $B \rightarrow bB$

- $B \rightarrow \varepsilon$ (słowo puste)

$\}$

$L(G)$ – zbiór słów

zaczynających się ciągiem liter a (być może pustym), za nim ciąg liter b (niepusty)

$L(G) = a^*b^+$

Przykłady słów

$aaaaabb \in L(G)$

$ab \in L(G)$

$b \in L(G)$

$ba \notin L(G)$

$aa \notin L(G)$

Dlaczego nazwa „**bezkontekstowa**”?

Bezkontekstowość:

- Użycie produkcji w wyprowadzeniu niezależne od **kontekstu** (symboli w otoczeniu zastępowanego nieterminala)

Inne typy gramatyk

- kontekstowe
- regularne
- ściśle kontekstowe, ...

Po co gramatyki?

1. **Precyzyjny** a zarazem intuicyjny opis, np. dla składni języków programowania (choć nie wszystkie elementy języka da się opisać!!!);
2. Istnieją efektywne narzędzia (algorytmy i programy) weryfikujące przynależność do języka definiowanego przez gramatykę – p. przedmioty JFiZO, AiSD, metody programowania, metody translacji
3. Twórca języka programowania i tłumacza:
 - definiuje jego **składnię** z użyciem gramatyk;
 - korzysta z narzędzi (punkt 2) do sprawdzania czy napisany program jest poprawny **składniowo**.

Gramatyki: drzewo wyprowadzenia

Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

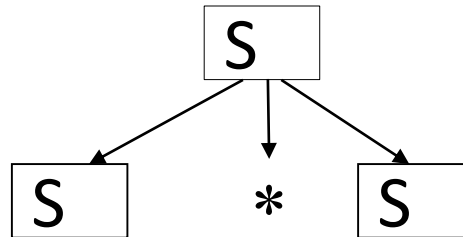
$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Wyprowadzenie:

$S \Rightarrow S * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

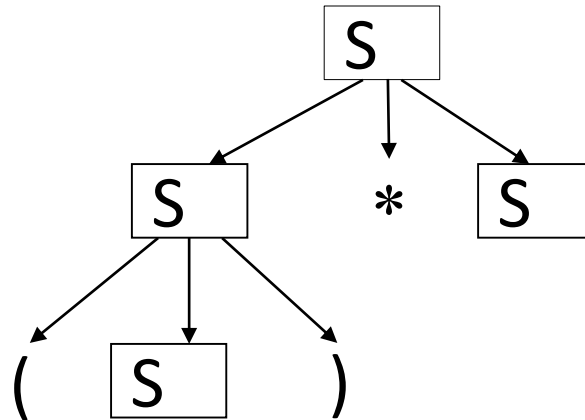
$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

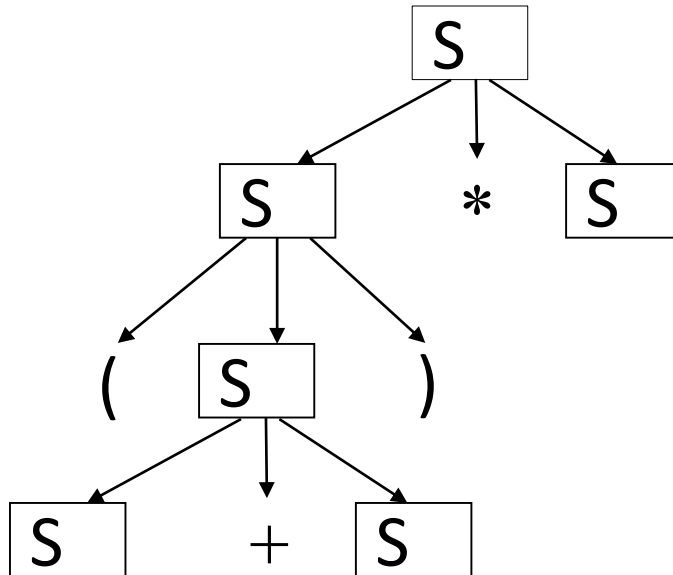
$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$
 $(S + S) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

$S \rightarrow S + S$

$S \rightarrow S * S$

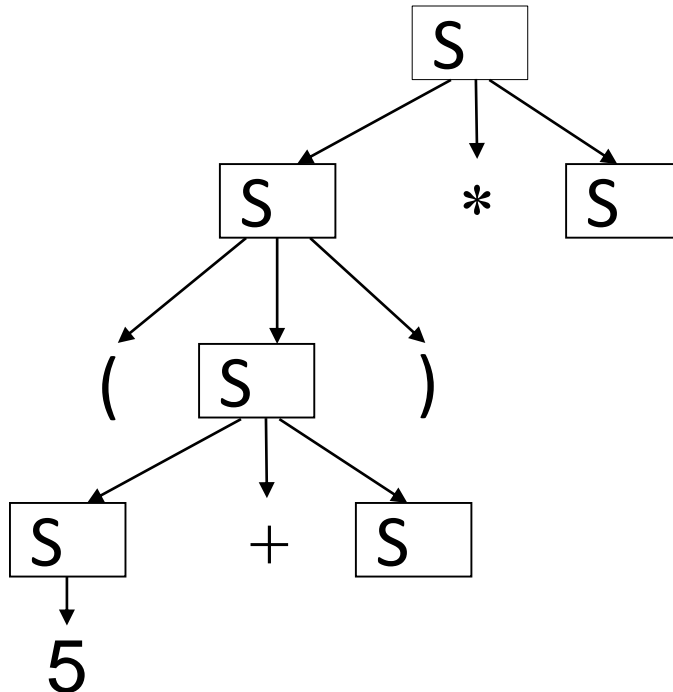
$S \rightarrow (S)$

Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

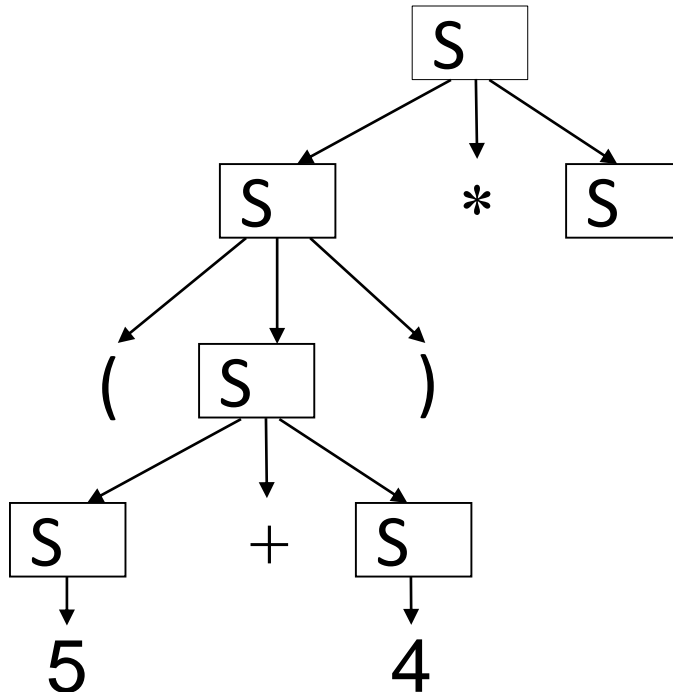
Wyprowadzenie:

$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Wyprowadzenie:

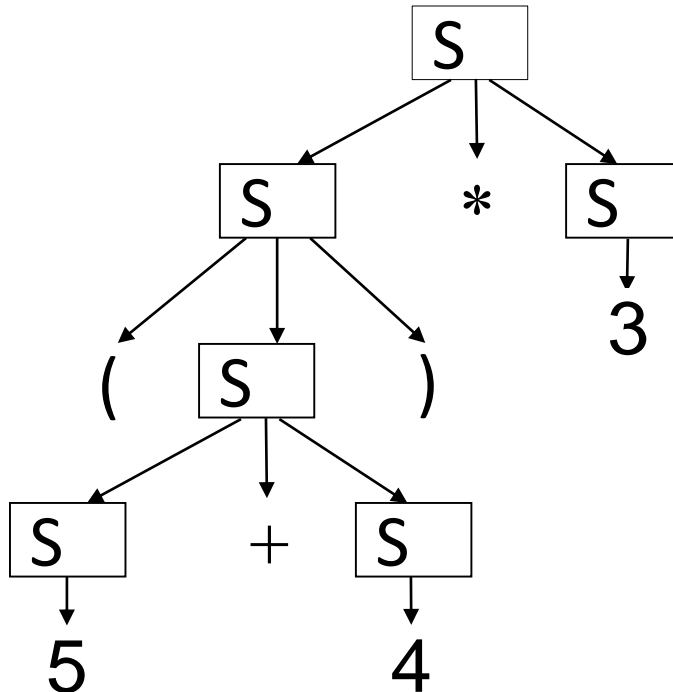
$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S \Rightarrow$

$(5 + 4) * 3$



Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Wyprowadzenie:

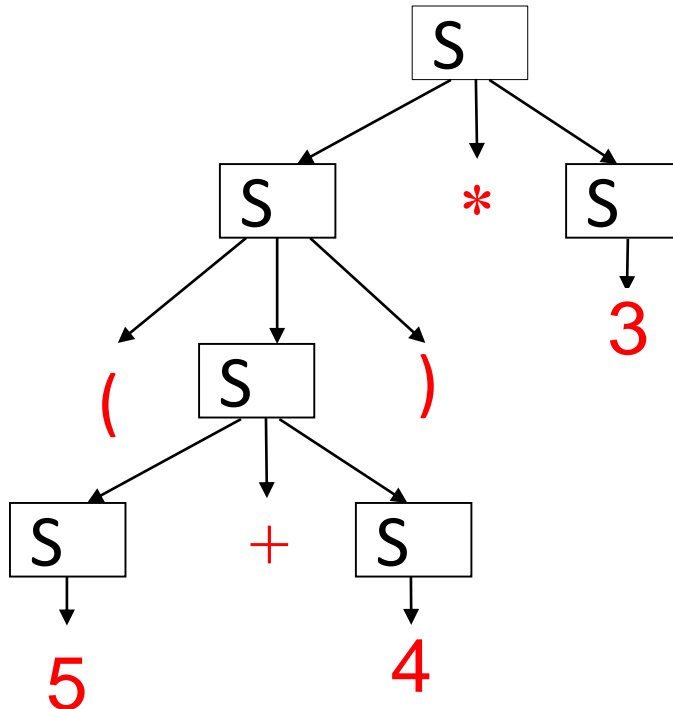
$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S \Rightarrow$

$(5 + 4) * 3$



Drzewo wyprowadzenia – formalnie

Drzewo (ukorzenione, z porządkiem):

Struktura złożona z powiązanych **węzłów**:

- Każdy węzeł może mieć dzieci uporządkowane od lewej do prawej.
- Każdy węzeł poza korzeniem ma dokładnie jednego **rodzica**
- **Korzeń** nie ma rodzica.
- **Dokładnie** jeden węzeł jest korzeniem.

Drzewo wyprowadzenia – formalnie

Drzewo wyprowadzenia w gramatyce

$G(N, T, P, S)$:

- Każdy węzeł ma etykietę ze zbioru $N \cup T$
- Etykieta korzenia to S (symbol startowy)
- Etykiety liści ze zbioru T
- Etykiety węzłów wewnętrznych (nie-liści) z N
- Jeśli v ma etykietę X , to jego dzieci mają etykiety $Y_1 \dots Y_n$ takie, że $X \rightarrow Y_1 \dots Y_n$ jest produkcją z P .

Drzewo wyprowadzenia – przykład

Produkcje:

$S \rightarrow 0, \dots, S \rightarrow 9$

$S \rightarrow S + S$

$S \rightarrow S * S$

$S \rightarrow (S)$

Wyprowadzenie:

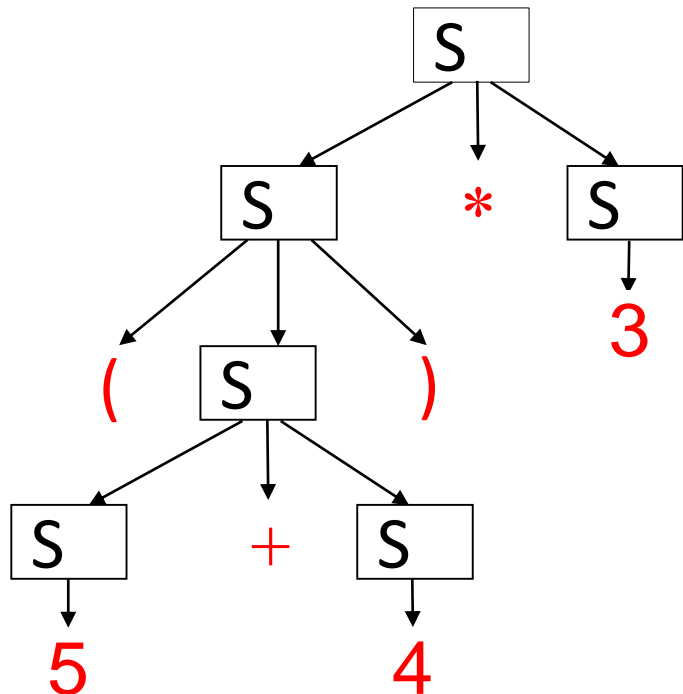
$S \Rightarrow S * S \Rightarrow (S) * S \Rightarrow$

$(S + S) * S \Rightarrow$

$(5 + S) * S \Rightarrow$

$(5 + 4) * S \Rightarrow$

$(5 + 4) * 3$



Wyprowadzony napis:

- Etykiety liści drzewa w porządku od lewej do prawej
- Przykład: $(5+4) * 3$

Notacja BNF (Backus-Naur)

Notacja BNF:

- Extended Backus-Naur Form: bardziej intuicyjna od formalnego opisu
- Używana głównie do definiowania składni języków programowania

Pojęcie	Gram. bezkontekst.	BNF
Nieterminale	A, B, C, D, ...	$\langle \text{napis} \rangle$
Terminale	Małe litery, inne znaki	Dowolne znaki poza \langle , \rangle
Produkcje	\rightarrow	$::=$
	$A \rightarrow w_1, A \rightarrow w_2$	$\langle \text{napis} \rangle ::= w_1 \mid w_2$

Gramatyka bezkontekstowa i BNF - przykład

Przykład. Liczba ze znakiem

$G(N, T, P, S)$

- $N = \{ S, C, X \}$
 - $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- $P = \{$
- $S \rightarrow - X$
 - $S \rightarrow X$
 - $C \rightarrow 0, C \rightarrow 1, \dots, C \rightarrow 9$
 - $X \rightarrow C$
 - $X \rightarrow C X$
- $\}$

Przykład. Liczba ze znakiem w BNF

$G(N, T, P, \langle \text{liczba_ze_zn} \rangle)$

- $N = \{ \langle \text{liczba_ze_zn} \rangle, \langle \text{cyfra} \rangle, \langle \text{liczba} \rangle \}$
 - $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- $P = \{$
- $\langle \text{liczba_ze_zn} \rangle ::= - \langle \text{liczba} \rangle \mid \langle \text{liczba} \rangle$
 - $\langle \text{cyfra} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
 - $\langle \text{liczba} \rangle ::= \langle \text{cyfra} \rangle \mid \langle \text{cyfra} \rangle \langle \text{liczba} \rangle$
- $\}$

Gramatyka bezkontekstowa i BNF - przykład

Przykład. Wyrażenie

$G(N, T, P, S)$

- $N = \{ S, C \}$
- $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

$P = \{$
 $C \rightarrow 0, \dots, C \rightarrow 9$
 $S \rightarrow C$
 $S \rightarrow S + S$
 $S \rightarrow S * S$
 $S \rightarrow (S)$
 $\}$

Przykład. Wyrażenie w BNF

- $N = \{ \langle \text{wyr} \rangle, \langle \text{arg} \rangle \}$
 - $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- $P = \{$
 $\langle \text{arg} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
 $\langle \text{wyr} \rangle ::= \langle \text{arg} \rangle \mid (\langle \text{wyr} \rangle) \mid$
 $\langle \text{wyr} \rangle + \langle \text{wyr} \rangle \mid \langle \text{wyr} \rangle * \langle \text{wyr} \rangle$
 $\}$

Notacja EBNF (Extended Backus-Naur)

Notacja EBNF:

- Extended Backus-Naur Form: rozszerzenie BNF
- Używana głównie do definiowania składni języków programowania

Pojęcie	BNF	EBNF
Nieterminale	$\langle \text{napis} \rangle$	napis
Terminale	Dowolne znaki poza \langle , \rangle	Dowolne znaki w cudzysłowie
Produkcje	$::=$	$=$
	$w_1 \mid w_2$	$w_1 \mid w_2$

Notacja EBNF (Extended Backus-Naur)

Notacja EBNF: dodatkowe konwencje

Opis	EBNF
0 lub 1 wystąpienie x	[x]
Dowolna dodatnia liczba wystąpień x	{ x }

BNF i EBNF - przykład

Przykład. Liczba ze znakiem w BNF

$G(N, T, P, \langle \text{liczba_ze_zn} \rangle)$

- $N = \{ \langle \text{liczba_ze_zn} \rangle, \langle \text{cyfra} \rangle, \langle \text{liczba} \rangle \}$
- $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- $P = \{$
 - $\langle \text{liczba_ze_zn} \rangle ::= - \langle \text{liczba} \rangle \mid \langle \text{liczba} \rangle$
 - $\langle \text{cyfra} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
 - $\langle \text{liczba} \rangle ::= \langle \text{cyfra} \rangle \mid \langle \text{cyfra} \rangle \langle \text{liczba} \rangle$ $\}$

Przykład. Liczba ze znakiem w EBNF

$G(N, T, P, \langle \text{liczba_ze_zn} \rangle)$

- $N = \{ \text{liczba_ze_zn}, \text{cyfra}, \text{liczba} \}$
- $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- $P = \{$
 - $\text{liczba_ze_zn} = ["-"] \text{liczba}$
 - $\text{cyfra} = "0" \mid "1" \mid "2" \mid "3" \mid "4" \mid "5" \mid "6" \mid "7" \mid "8" \mid "9"$
 - $\text{liczba} = \text{cyfra} \{ \text{cyfra} \}$ $\}$

BNF i EBNF - przykład

Przykład. Wyrażenie w BNF

$G(N, T, P, \langle \text{wyr} \rangle)$

- $N = \{ \langle \text{wyr} \rangle, \langle \text{arg} \rangle \}$
- $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- $P = \{$
 - $\langle \text{arg} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
 - $\langle \text{wyr} \rangle ::= \langle \text{arg} \rangle \mid (\langle \text{wyr} \rangle) \mid \langle \text{wyr} \rangle + \langle \text{wyr} \rangle \mid \langle \text{wyr} \rangle * \langle \text{wyr} \rangle$ $\}$

Przykład. Wyrażenie w EBNF

$G(N, T, P, \langle \text{wyr} \rangle)$

- $N = \{ \text{wyr}, \text{arg} \}$
- $T = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- $P = \{$
 - $\text{arg} = "0" \mid "1" \mid "2" \mid "3" \mid "4" \mid "5" \mid "6" \mid "7" \mid "8" \mid "9"$
 - $\text{wyr} = \text{arg} \mid "(" \text{wyr} ")" \mid \text{wyr} "+" \text{wyr} \mid \text{wyr} "*" \text{wyr}$ $\}$

Gramatyki – podsumowanie

Pojęcia:

- Alfabet, słowo, język
- Gramatyka bezkontekstowa, wyprowadzenie, drzewo wyprowadzenia
- Język definiowany przez gramatykę
- Notacja BNF
- Notacja EBNF

Korzyści z gramatyk:

- Precyzyjna definicja składni
- Narzędzia do sprawdzania przynależności do języka zdefiniowanego gramatyką

Odwrotna Notacja Polska (ONP) - motywacje

Wartościowanie wyrażeń

Dotychczas:

Składnia – czy wyrażenie jest poprawnie napisane

Teraz:

Semantyka (częściowa) – jak obliczyć wartość wyrażenia, czyli rozwiązać problem:

Wejście: poprawne wyrażenie w

Wyjście: wartość wyrażenia w

Wartościowanie wyrażeń

Wyrażenie:

$$w = w_1 \dots w_n$$

gdzie $w_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (,), +, *\}$

oraz w należy do języka $L(G)$ dla $G(N, T, P, S)$

gdzie

$$P = \{S \rightarrow 0 \dots S \rightarrow 9;$$

$$S \rightarrow (S); S \rightarrow S + S; S \rightarrow S * S\}$$

Przypomnienie:

Dla uproszczenia zakładamy, że argumenty to liczby **jednocyfrowe**.

Wartościowanie wyrażeń

Uproszczenie:

- wszystkie operatory mają taki sam priorytet (kolejność można wymusić nawiasami)
- operacje wykonujemy „od lewej do prawej”

Wartość $fi(w)$ wyrażenia $w = w_1 \dots w_n$:

- Jeśli wyrażenie $w \in \{0, 1, \dots, 9\}$, wówczas jego wartość jest równa liczbie w
- Jeśli $w = u \text{ op } v$ dla wyrażeń u i v i operatora op , to $fi(w) = fi(u) \text{ op } fi(v)$

Uwaga: podział dla najkrótszego v .

Wartościowanie wyrażeń

Wartość $fi(w)$ wyrażenia $w=w_1 \dots w_n$:

- Jeśli wyrażenie $w \in \{0, 1, \dots, 9\}$, wówczas jego wartość jest równa liczbie w
- Jeśli $w = u \text{ op } v$ dla wyrażeń u i v i operatora op , to $fi(w) = fi(u) \text{ op } fi(v)$

Uwaga: podział dla najkrótszego v , nie ma priorytetów.

Przykład

$$fi(7)=7$$

$$fi(3+7)=fi(3)+fi(7)=10$$

$$fi(3+(7*2))=fi(3)+fi(7*2)=3+14=17$$

$$fi(3+7*2)=fi(3+7)*fi(2)=10*2=20$$

gdyż nie uwzględniamy priorytetów, i „od lewej do prawej”!!!

Wartościowanie wyrażeń

Problem z algorytmem wartościującym wg rekurencyjnej definicji:

jak rozbić wyrażenie w na 3 człony: u , op , v ?

Przykład

Jak rozbić na u op v poniższe wyrażenie:

$$(5+7)-(2+3*4)*(2+3-5*(5-9))$$

Wartościowanie wyrażeń

Przykład

Jak rozbić na u op v poniższe wyrażenie:

$$(5+7) - (2+3*4) * (2+3-5*(5-9))$$

Gdy nie ma priorytetów – „jak najkrótsza prawa strona” („wykonuj od lewej”):

$$\left((5+7) - (2+3*4) \right) * \left((2+3-5*(5-9)) \right)$$

Z priorytetami:

$$\left((5+7) \right) - \left((2+3*4) * (2+3-5*(5-9)) \right)$$

Odwrotna notacja polska (ONP)

ONP – motywacja:

- zapis wyrażenia ułatwiający wartościowanie
- kolejność działań można ustalić bez nawiasów

Def. Postać ONP wyrażenia $w = w_1 \dots w_n$:

- Jeśli wyrażenie $w \in \{0, 1, \dots, 9\}$, wówczas $ONP(w)$ jest równa w
- Jeśli $w = u \text{ op } v$ dla wyrażień u , v i operatora op , to
$$ONP(w) = ONP(u) \text{ ONP}(v) \text{ op}$$

ONP – przykłady

$$\text{ONP}(3 + 4) = 3 \ 4 \ +$$

$$\text{ONP}((3+4)*5) = 3 \ 4 \ + \ 5 \ *$$

$$\text{ONP}(3 * (4+5)) = 3 \ 4 \ 5 \ + \ *$$

$$\text{ONP}(5 - (3*2)) = 5 \ 3 \ 2 \ * \ -$$

$$\text{ONP}((5 - (3*2)) + 7) = 5 \ 3 \ 2 \ * \ - \ 7 \ +$$

Przyjmując założenie o wykonywaniu operacji „od lewej do prawej:

$$\text{ONP}(5 - (3*2) * 7) =$$

$$\text{ONP}((5 - (3*2)) * 7) =$$

$$5 \ 3 \ 2 \ * \ - \ 7 \ *$$

ONP – kluczowa własność

- W wyrażeniach ONP nie potrzeba nawiasów
- Wyrażenia ONP jednoznacznie określają kolejność wykonywania operacji:

standardowo: $(a + b) - c$

ONP: $a b + c -$

standardowo: $a + (b - c)$

ONP: $a b c - +$

ONP – problemy algorytmiczne

ONP – problemy algorytmiczne

Problem obliczania wartości wyrażenia:

Wejście: postać ONP wyrażenia w

Wyjście: wartość wyrażenia w

Problem zamiany na ONP:

Wejście: wyrażenie w postaci „standardowej”

Wyjście: postać ONP wyrażenia w

ONP – algorytm obliczania wartości
wyrażenia

Problem obliczania wartości wyrażenia

Wartość $fi(w)$ wyrażenia $w = w_1 \dots w_n$ w ONP:

- Jeśli wyrażenie $w \in \{0, 1, \dots, 9\}$, wówczas jego wartość jest równa liczbie w
- Jeśli $w = u \ v \ op$ dla wyrażień u i v i operatora op , to $fi(w) = fi(u) \ op \ fi(v)$

Problem obliczania wartości wyrażenia:

Wejście: postać ONP wyrażenia w

Wyjście: wartość wyrażenia w

Problem obliczania wartości wyrażenia

Problem obliczania wartości wyrażenia:

Wejście: postać ONP wyrażenia w

Wyjście: wartość wyrażenia w

Przykład

Wejście: 5 3 + 7 *

Wyjście: 56

Narzędzie - stos

Używamy stosu jako abstrakcyjnej struktury danych:

- **StosPusty()** – tworzy pusty stos i zwraca go jako wynik
- **Wstaw(A,x)** – wstawia element x na szczyt stosu A
- **Zdejmij(A)** – zdejmuje i zwraca jako wynik element ze szczytu stosu A (o ile stos nie jest pusty)
- **CzyPusty(A)** – prawda gdy stos A jest pusty, fałsz w przeciwnym przypadku

ONP – wartość wyrażenia

Wejście: $w=w_1 w_2 \dots w_n$ – wyrażenie w postaci ONP

Wyjście: wartość wyrażenia w

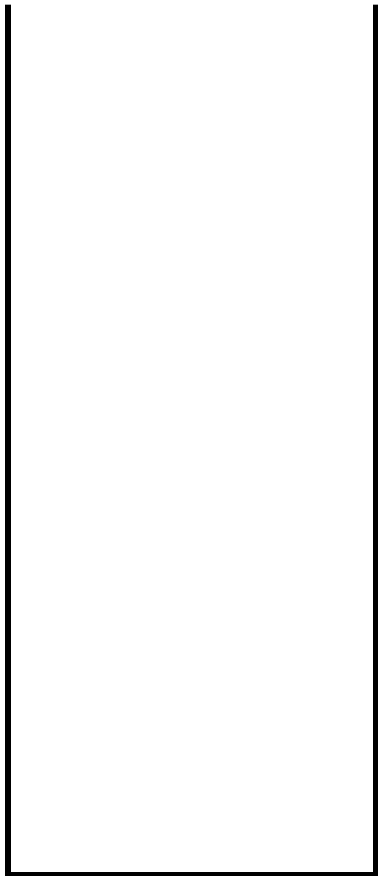
1. $A \leftarrow \text{StosPusty}()$
2. Dla $i=1,2,\dots,n$:
 - a) $x \leftarrow w_i$
 - b) Jeśli x to argument (liczba): $\text{Wstaw}(A,x)$
 - c) Jeśli x to operator:
 - i. $y_2 \leftarrow \text{Zdejmij}(A)$
 - ii. $y_1 \leftarrow \text{Zdejmij}(A)$
 - iii. Wykonaj działanie $y_1 x y_2$, wynik na stos:
 - $z \leftarrow y_1 x y_2$
 - $\text{Wstaw}(A,z)$
3. Wynik $\leftarrow \text{Zdejmij}(A)$

ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = \textcolor{red}{3} \ 4 \ 5 \ + \ 2 \ 7 \ + \ - \ *$$

stos



ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \text{ } 4 \text{ } 5 \text{ } + \text{ } 2 \text{ } 7 \text{ } + \text{ } - \text{ } *$$

stos



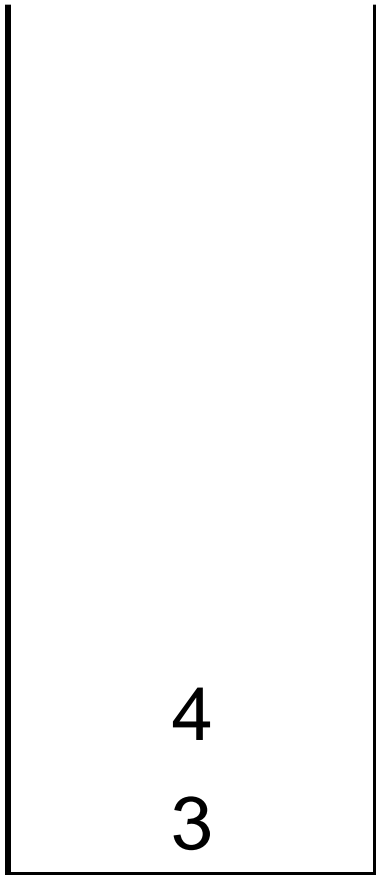
3

ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 + 2 \ 7 + - *$$

stos

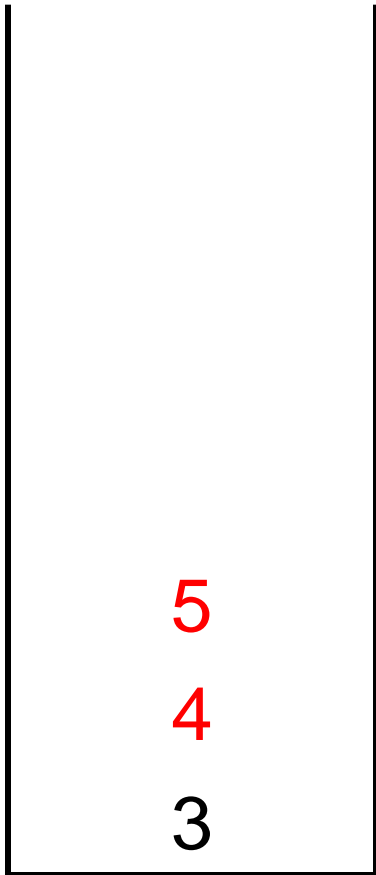


ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 \ + \ 2 \ 7 \ + \ - \ *$$

stos

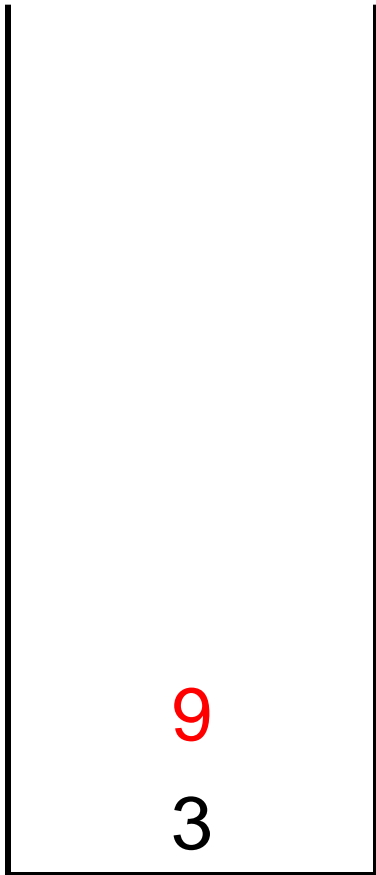


ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 + \textcolor{red}{2} \ 7 + - *$$

stos

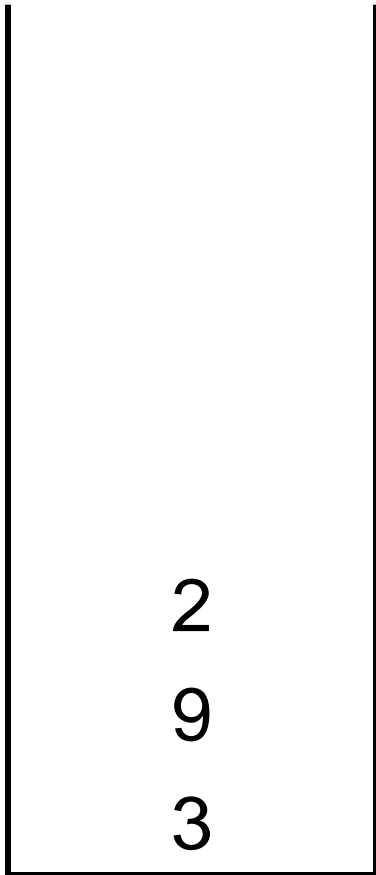


ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 \ + \ 2 \ 7 \ + \ - \ *$$

stos

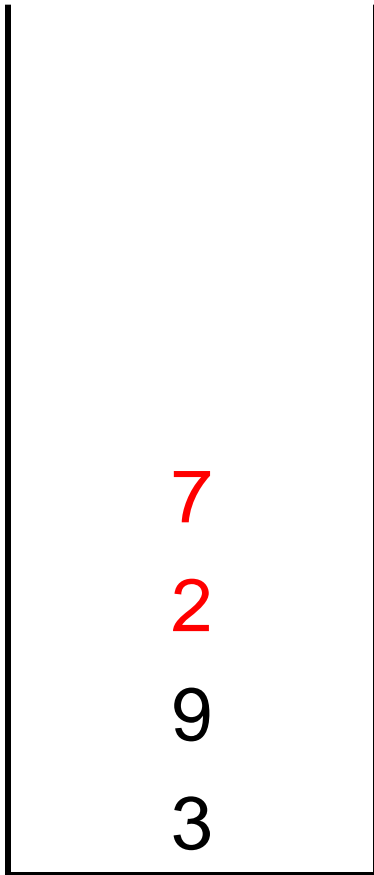


ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 \ + \ 2 \ 7 \ + \ - \ *$$

stos

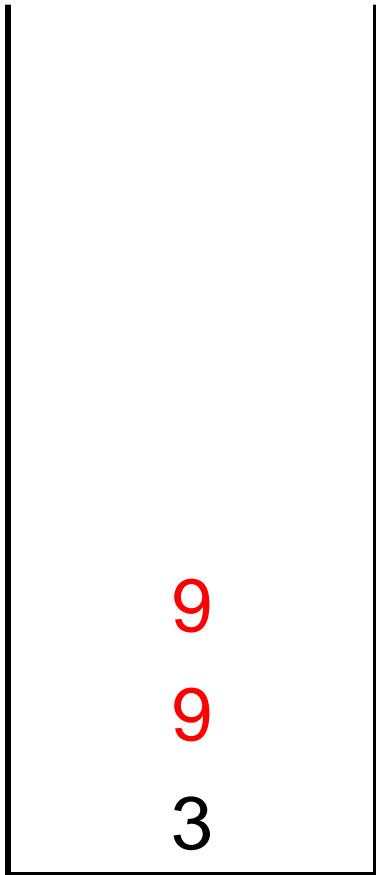


ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 \ + \ 2 \ 7 \ + \ - \ *$$

stos

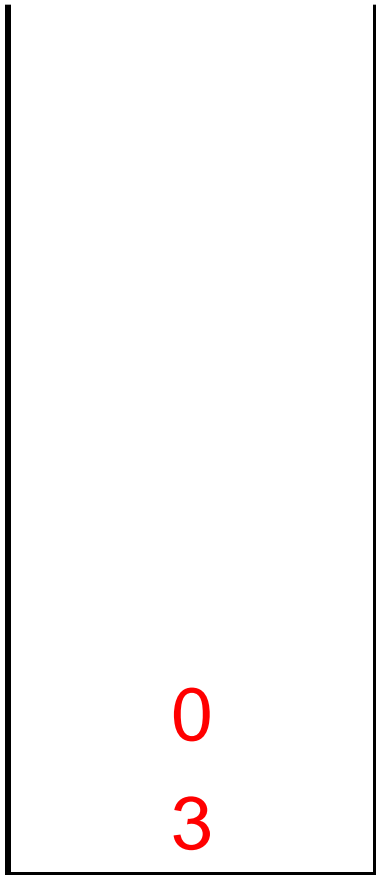


ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 \ + \ 2 \ 7 \ \div \ - \ *$$

stos



ONP wartościowanie – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3 \ 4 \ 5 \ + \ 2 \ 7 \ \- \ *$$

stos



0

ONP – wartość wyrażenia

Tw. (Poprawność)

Po krokach 1. i 2. algorytmu na stosie znajduje się wartość wyrażenia w .

Dowód (szkic)

Indukcja ze względu na długość w :

- w ma długość 1: w to liczba wstawiana na stos w 2.b)
- **Zał.:** tw. spełnione dla w o długości $<n$ oraz $n>1$

Teza: tw. spełnione dla w o długości n

Dowód kroku indukcyjnego: w ma postać $u v x$,

gdzie u , v to wyrażenia ONP krótsze niż n , x to operator oraz:

a) po przeczytaniu u na stosie wartość u (z zał. ind.)

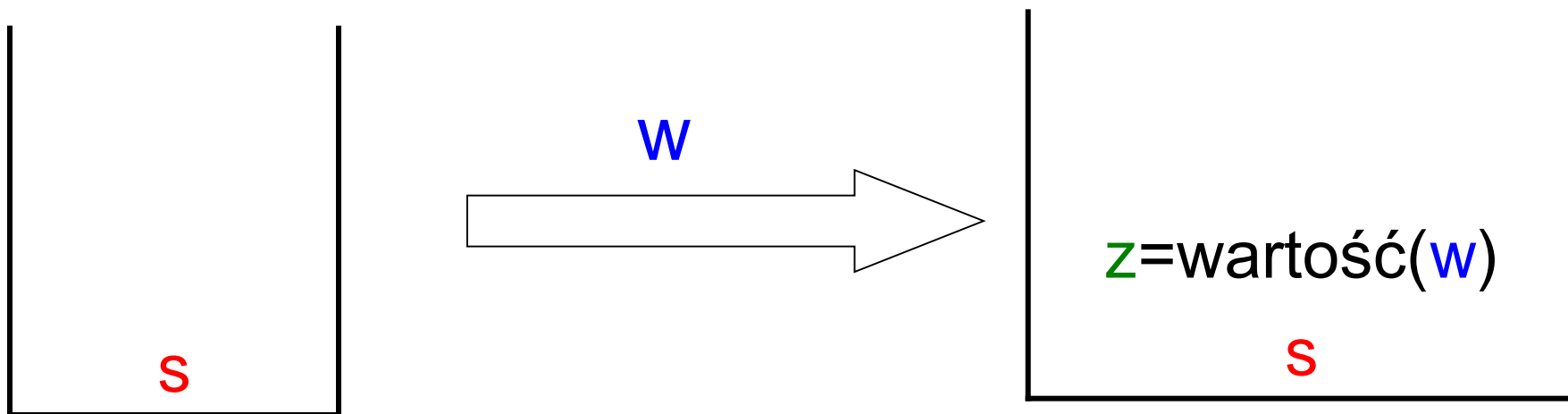
b) po przeczyt. v na stosie wartość u i wartość v

c) po przeczyt. x na stosie wartość w (2.c)

ONP – wartość wyrażenia

Uwaga. Dla pełnej poprawności dowodu (punkt b)) powinniśmy dowodzić silniejszej tezy:

Jeśli czytanie wyrażenia w zaczynamy z zawartością stosu s , to po przeczytaniu wyrażenia w zawartość stosu będzie równa $s \cdot z$, gdzie z to wartość wyrażenia w (z jest na szczycie).



Algorytm zamiany wyrażenia na postać ONP – ver. 1 (bez priorytetów)

Zamiana na ONP – ver. 1 (bez priorytetów)

We: $w = w_1 w_2 \dots w_n$ – poprawne wyrażenie w postaci „tradycyjnej” (infiksowej), elementy w_i mogą być postaci:

- argument: $0, 1, 2, \dots, 9$
- operator: $+$, $*$
- nawias (lub)

Wy: ONP(w)

Przypomnienie: przyjmujemy, że wszystkie operatory mają takie same priorytety i wykonujemy „od lewej do prawej”!

ONP – bez priorytetów

Bez priorytetów (jak najkrótsza prawa strona):

$$\text{ONP}(3 * 4 + 5) = \text{ONP}((3 * 4) + 5) = 3 \ 4 \ * \ 5 \ +$$

$$\text{ONP}(3 + 4 * 5) = \text{ONP}(3 + (4 * 5)) = 3 \ 4 \ + \ 5 \ *$$

Gdy uwzględnimy priorytety:

$$\text{ONP}(3 * 4 + 5) = \text{ONP}((3 * 4) + 5) = 3 \ 4 \ * \ 5 \ +$$

$$\text{ONP}(3 + 4 * 5) = \text{ONP}(3 + (4 * 5)) = 3 \ 4 \ 5 \ * \ +$$

Zamiana na ONP – ver. 1 (bez priorytetów)

Algorytm 1

1. $A \leftarrow \text{StosPusty}()$

2. Dla $i=1,2,\dots,n$:

- $x \leftarrow w_i$
- Jeśli x to argument (liczba): wypisz x
- Jeśli x to nawias "(" : Wstaw(A, x)
- Jeśli x to operator:
 - Dopóki Szczyt(A) to operator: Wypisz Zdejmij(A)
 - Wstaw(A, x)
- Jeśli x to nawias ")" :
 - Dopóki Szczyt(A) \neq "(" :
 - Wypisz Zdejmij(A)
 - Zdejmij(A)

3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

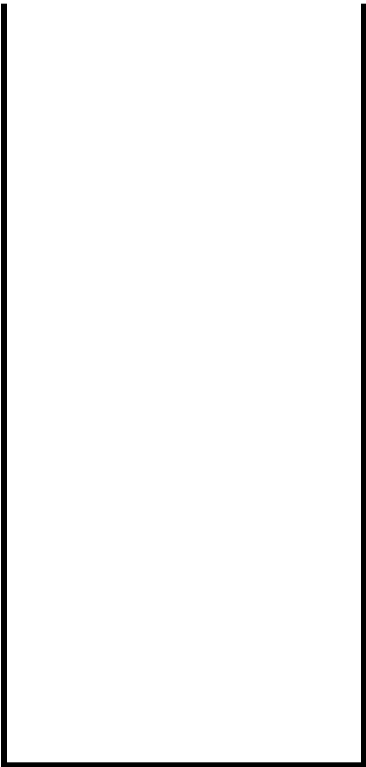
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

$$\text{ONP}(w) = 3\ 4\ 5\ +\ 2\ 7\ +\ -\ *$$

Wyjście:

Stos:



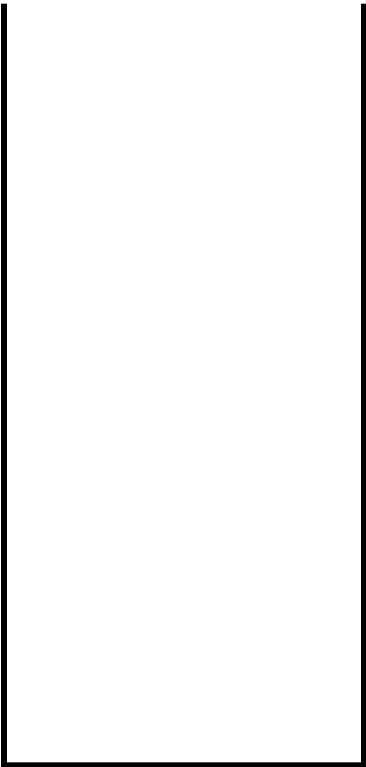
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3

Stos:



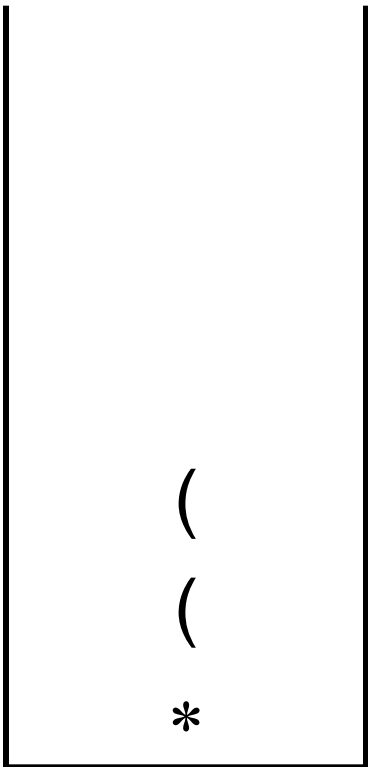
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3

Stos:



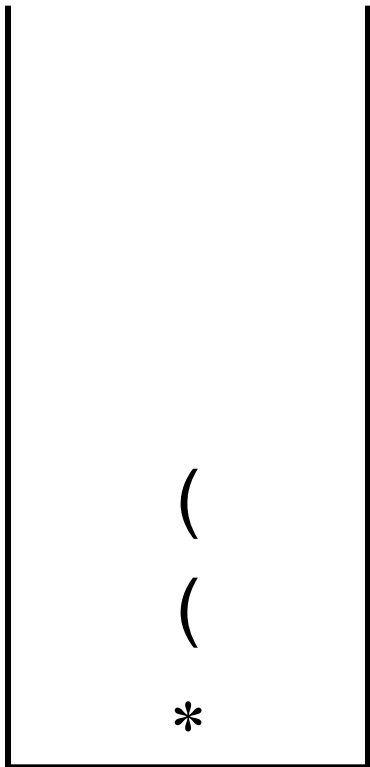
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4

Stos:



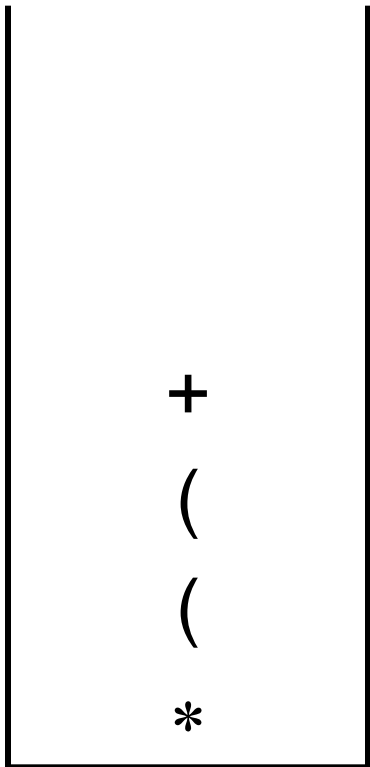
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4

Stos:



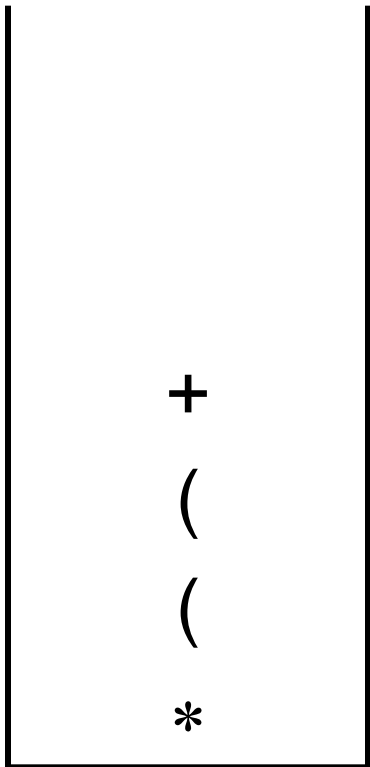
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5

Stos:



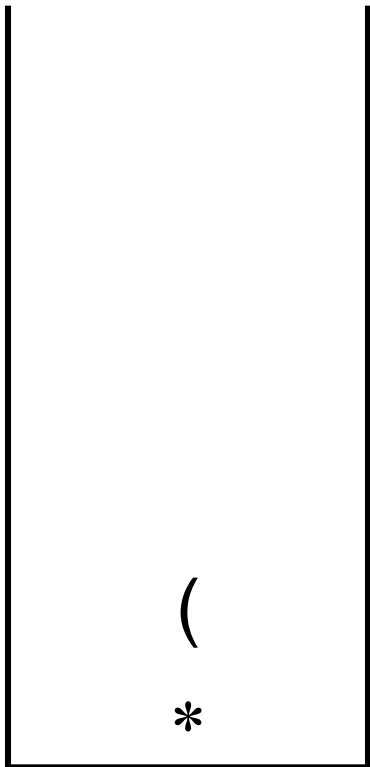
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 +

Stos:



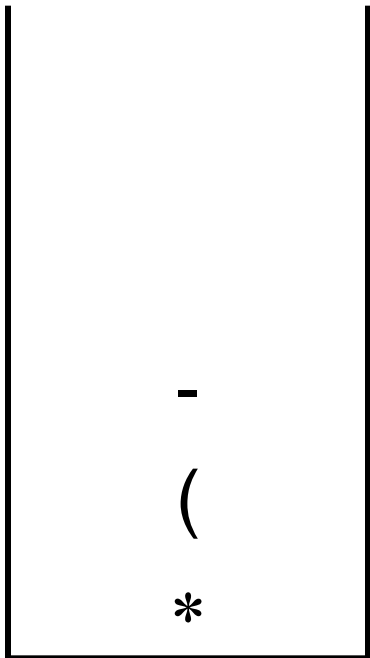
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 +

Stos:



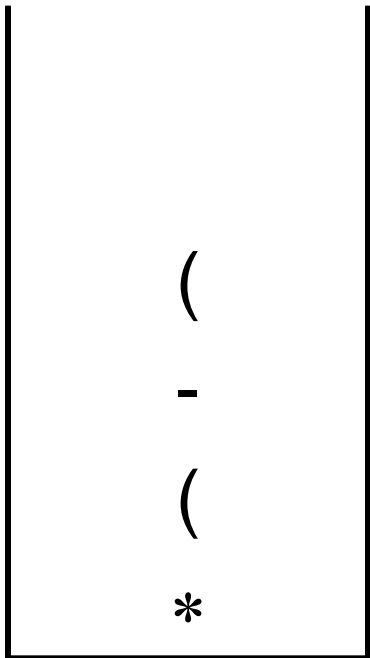
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 +

Stos:



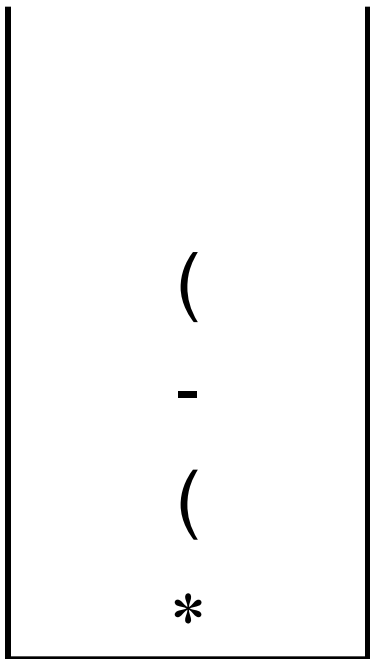
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 + 2

Stos:



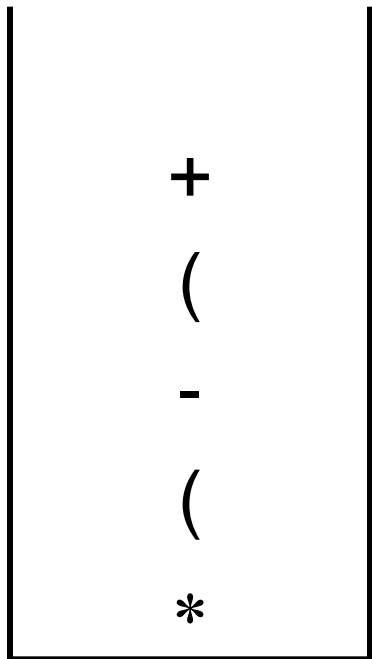
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 + 2

Stos:



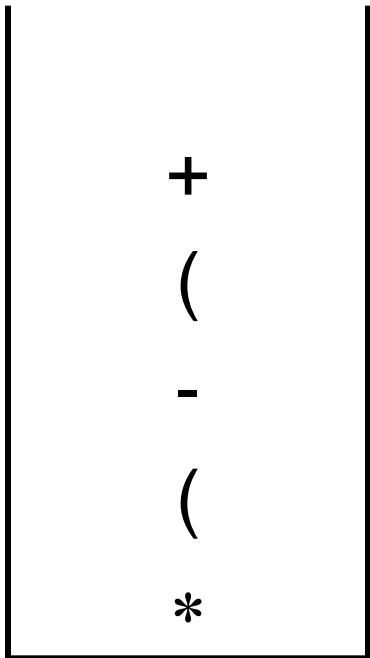
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 + 2 7

Stos:



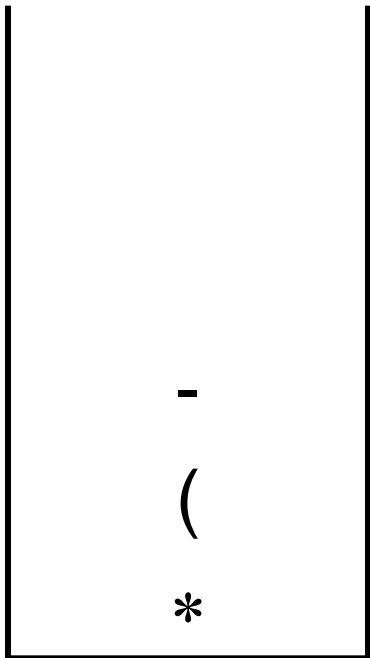
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 + 2 7 +

Stos:



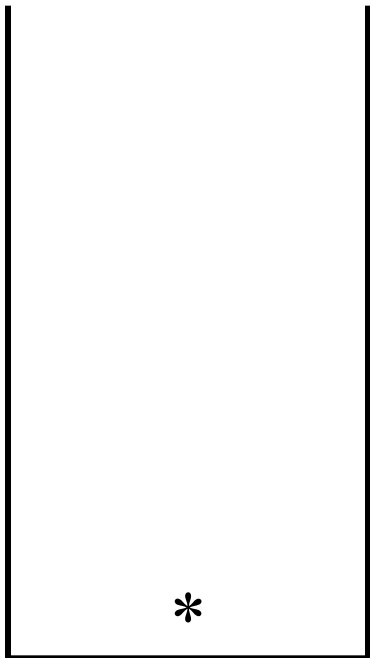
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 + 2 7 + -

Stos:



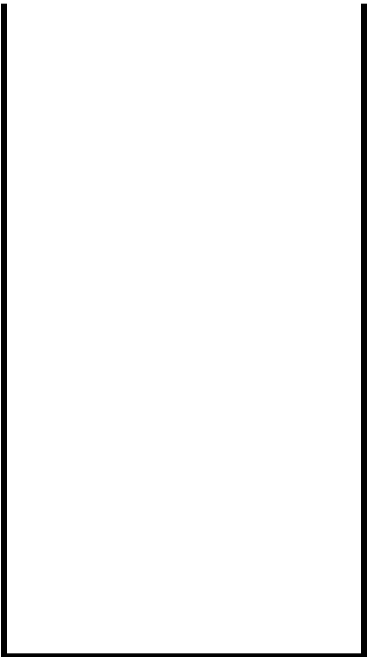
Zamiana na ONP – przykład

$$w = 3 * ((4 + 5) - (2 + 7))$$

ONP(w) = 3 4 5 + 2 7 + - *

Wyjście: 3 4 5 + 2 7 + - *

Stos:



O poprawności algorytmu zamiany
wyrażenia na postać ONP

Zamiana na ONP – poprawność

Intuicje:

- operatory w ONP są umieszczane **za** ich argumentami: dlatego operatory odkładamy na stos (do wypisania na wyjściu później), a argumenty od razu wypisujemy na wyjście;
- dlaczego używamy stosu (last in, first out): odpowiada realizacji „**rekurencyjnych**” zagłębień wynikających z definicji poprawnych wyrażeń

Zamiana na ONP – poprawność

Obserwacja o nawiasach

W każdym poprawnym wyrażeniu, przy odczytaniu $)$, Algorytm 1 zdejmuje ze stosu odpowiadający mu $($, nie zdejmuje niczego poniżej zdejmowanego $($.

Dowód:

- indukcja ze względu na długość wyrażenia pomiędzy $($ i $)$.
- z wykorzystaniem faktu, że tylko odczytanie $)$ powoduje zdjęcie ze stosu $($

Zamiana na ONP – poprawność

Lemat o nawiasach.

Założmy, że $w = x(v)w$, gdzie v to poprawne wyrażenie oraz zawartość stosu po przeczytaniu x jest równa s .

Wówczas, podczas czytania (v) :

- a) żaden element s nie będzie zdjęty
- b) na wyjściu zostanie wypisane słowo, które algorytm wypisuje dla wyrażenia v ,
- c) po przeczytaniu (v) zawartość stosu będzie równa s



Zamiana na ONP – ver. 1 (bez priorytetów)

Algorytm 1

1. $A \leftarrow \text{StosPusty}()$

2. Dla $i=1,2,\dots,n$:

- $x \leftarrow w_i$
- Jeśli x to argument (liczba): wypisz x
- Jeśli x to nawias "(": Wstaw(A, x)
- Jeśli x to operator:
 - Dopóki Szczyt(A) to operator: Wypisz Zdejmij(A)
 - Wstaw(A, x)
- Jeśli x to nawias ")":
 - Dopóki Szczyt(A) \neq "(":
 - Wypisz Zdejmij(A)
 - Zdejmij(A)

3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

Zamiana na ONP – poprawność

Lemat o nawiasach.

Założmy, że $w = x(v)w$, gdzie v to poprawne wyrażenie oraz zawartość stosu po przeczytaniu x jest równa s .

Wówczas, podczas czytania (v) :

- a) żaden element s nie będzie zdjęty
- b) na wyjściu zostanie wypisane słowo, które algorytm wypisuje dla wyrażenia v ,
- c) po przeczytaniu (v) zawartość stosu będzie równa s

Dowód (szkic).

Punkty a) i c): z Obserwacji o nawiasach

Punkt b):

z a) i c) wynika, że na dla wejścia $w = x(v)w$, algorytm zachowuje się na fragmencie v tak samo jakby miał na wejściu v :

- zaczyna ze stosem $s ($
- nie „widzi” s na stosie (patrz a)), a $($ „zobaczy” dopiero czytając $)$
- odczytanie $)$ ze słowa $x(v)w$ odpowiada krokowi 3 algorytmu uruchomionego na wejściu v

Zamiana na ONP – poprawność

Lemat o końcu obliczeń.

Założmy, że Algorytm 1 uruchamiamy na poprawnym wyrażeniu w . Niech s to zawartość stosu po zakończeniu kroku 2, przed krokiem 3. Wówczas s zawiera tylko operatory (nie ma nawiasów otwierających)

Zamiana na ONP – poprawność

Lemat o końcu obliczeń.

Założmy, że Algorytm 1 uruchamiamy na poprawnym wyrażeniu w . Niech s to zawartość stosu po zakończeniu kroku 2, przed krokiem 3. Wówczas s zawiera tylko operatory (nie ma nawiasów otwierających)

Dowód

- wprost z Obserwacji o nawiasach.

Zamiana na ONP – poprawność

Obserwacja.

Każde poprawne wyrażenie w ma postać:

w to liczba („argument”) LUB

$w = u \ p \ v$ LUB

$w = (\ u \)$

gdzie

u – poprawne wyrażenie

p – operator

v – to argument lub $v = (\ x \)$ dla poprawnego wyrażenia x .

Zamiana na ONP – poprawność

Tw. Algorytm 1 podaje na wyjściu poprawną postać ONP (bez uwzględnienia priorytetów).

Zamiana na ONP – poprawność

Tw. Algorytm 1 podaje na wyjściu poprawną postać ONP (bez uwzględnienia priorytetów).

Szkic dowodu indukcyjnego względem długości w :

1. Długość w równa 1: proste

2. Krok indukcyjny:

Zakładamy prawdziwość dla wyrażeń o długości $< n$

I. Weźmy $w = u p v$ o długości n (**inne przypadki podobne**), gdzie p to operator, oraz

a) u - wyrażenie

b) v postaci (x) dla wyrażenia x :

Wówczas

- $ONP(w) = ONP(u) ONP(v) p$
- Po przeczytaniu $u p$ na wyjściu $ONP(u)$, a stos zawiera tylko p – z zał. ind. i lematu o końcu obliczeń (czytając p zdejmujemy ze stosu operatory tak jak w kroku 3)
- Po przeczytaniu $v = (x)$ – na wyjściu $ONP(u) ONP(v) p$ (z lematu o nawiasach)

Zamiana na ONP – poprawność

Tw. Algorytm 1 podaje na wyjściu poprawną postać ONP (bez uwzględnienia priorytetów).

Szkic dowodu indukcyjnego względem długości w , **ciąg dalszy** kroku indukcyjnego:

II. Weźmy $w = (u)$: poprawność wynika z poprawności działania dla u (długość u mniejsza niż n , więc możemy skorzystać z założenia indukcyjnego).

Zamiana na ONP – poprawność

Wejście: u p v,

Czytamy	Stos po przeczytaniu	Wyjście
u p	p	ONP(u)
v		ONP(u) ONP(v) p

Zamiana na ONP z priorytetami operatorów

Wartościowanie i ONP dla wyrażeń z priorytetami

Uproszczenia:

- ~~wszystkie operatory mają taki sam priorytet
(kolejność można wymusić nawiasami)~~
- Kolejność od lewej do prawej, ale z uwzględnieniem priorytetów:

standardowo

$5 + 7 + 9$

$5 + 7 * 9$

ONP

$5\ 7\ +\ 9\ +$

$5\ 7\ 9\ *\ +$

~~$5\ 7\ 9\ +\ +$~~

Wartościowanie i ONP dla wyrażeń z priorytetami

Wartość $fi(w)$ wyrażenia $w=w_1...w_n$:

- Jeśli wyrażenie $w \in \{0,1,...,9\}$, wówczas $ONP(w)=w$
 - Jeśli $w=u \ p \ v$ dla wyrażeń u i v i operatora p , oraz
 - u nie można przedstawić jako $x \ q \ y$ dla wyrażeń x , y i operatora q o mniejszym priorytecie od priorytetu p
 - v nie można przedstawić jako $x \ q \ y$ dla wyrażeń x , y i operatora q o mniejszym lub równym priorytecie od priorytetu p
- to $fi(w) = fi(u) \ p \ fi(v)$ oraz $ONP(w)=ONP(u)ONP(v)p$

Wartościowanie i ONP - priorytety

$$\text{ONP}(3 + 4 * 5) = 3 \ 4 \ 5 \ * \ +$$

- poprawny podział: $3 \quad + \quad 4 * 5$
- niepoprawny podział: $3+4 \quad * \quad 5$

Uwaga: na lewo od $*$ mamy $3+4$ i $+$ ma mniejszy priorytet od $*$

$$\text{ONP}(3 * 4 + 5) = 3 \ 4 \ * \ 5 \ +$$

- poprawny podział: $3*4 \quad + \quad 5$
- niepoprawny podział: $3 \quad * \quad 4+5$

Uwaga: na lewo od $+$ nie ma operatora o mniejszym priorytecie

$$\text{ONP}(3 + 4 + 5) = 3 \ 4 \ * \ 5 \ +$$

- poprawny podział: $3+4 \quad + \quad 5$
- niepoprawny podział: $3 \quad + \quad 4+5$

Zamiana na ONP – ver. 2 (z priorytetami)

We: $w = w_1 w_2 \dots w_n$ – poprawne wyrażenie w postaci „tradycyjnej” (infiksowej)

Wy: ONP(w)

Zamiana na ONP – ver. 2 (z priorytetami)

Algorytm 2

1. $A \leftarrow \text{StosPusty}()$

2. Dla $i=1,2,\dots,n$:

- $x \leftarrow w_i$
- Jeśli x to argument: wypisz x
- Jeśli x to nawias „(„: Wstaw(A,x)
- Jeśli x to operator:
 - Dopóki Szczyt(A) to operator o większym priorytecie niż x bądź równym priorytetowi x :
 - Wypisz Zdejmij(A)
 - Wstaw(A,x)
- Jeśli x to nawias „)„:
 - Dopóki Szczyt(A) \neq „(„:
 - Wypisz Zdejmij(A)
 - Zdejmij(A)

3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

Dla porównania....

Zamiana na ONP – ver. 1 (bez priorytetów)

Algorytm 1

1. $A \leftarrow \text{StosPusty}()$
2. Dla $i=1, 2, \dots, n$:
 - $x \leftarrow w_i$
 - Jeśli x to argument: wypisz x
 - Jeśli x to nawias "(" : Wstaw(A, x)
 - Jeśli x to operator:
 - Dopóki Szczyt(A) to operator: Wypisz Zdejmij(A)
 - Wstaw(A, x)
 - Jeśli x to nawias ")" :
 - Dopóki Szczyt(A) \neq "(" :
 - Wypisz Zdejmij(A)
 - Zdejmij(A)
3. Dopóki A nie jest pusty: Wypisz Zdejmij(A)

Zamiana na ONP – ver. 2 (z priorytetami)

Intuicja

Gdy czytamy operator **p** to zamykamy „niewidoczne” nawiasy wynikające z wyższych priorytetów wcześniejszych operatorów na tym samym poziomie nawiasowania (w tym samym podwyrażeniu).

Przykład

$5 * 4 + 3$

$5 + 4 * 3$

$7 + (5 + 4 * 3)$

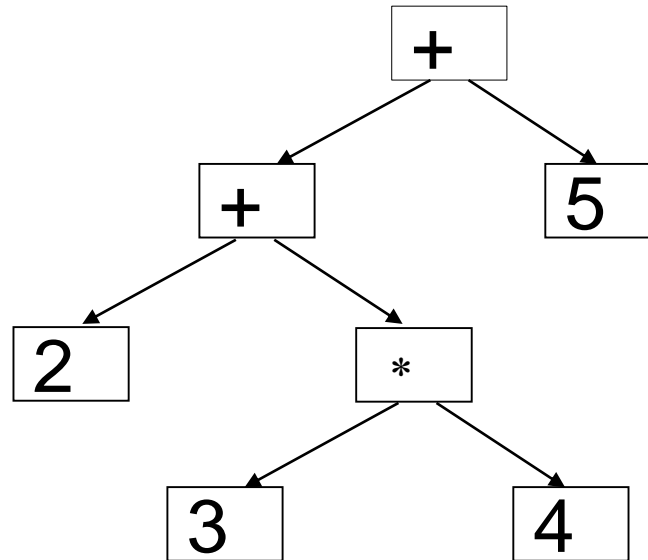
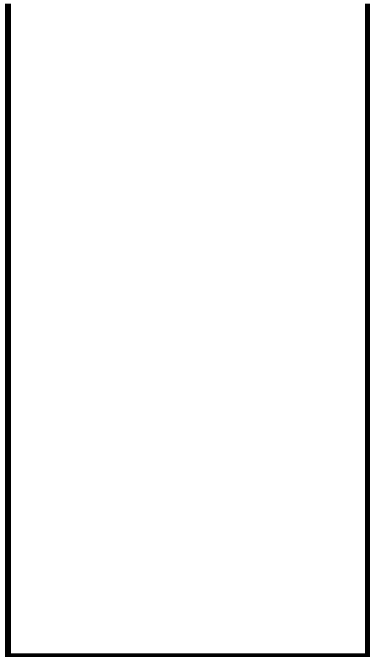
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * + 5 +

Wyjście:

Stos:



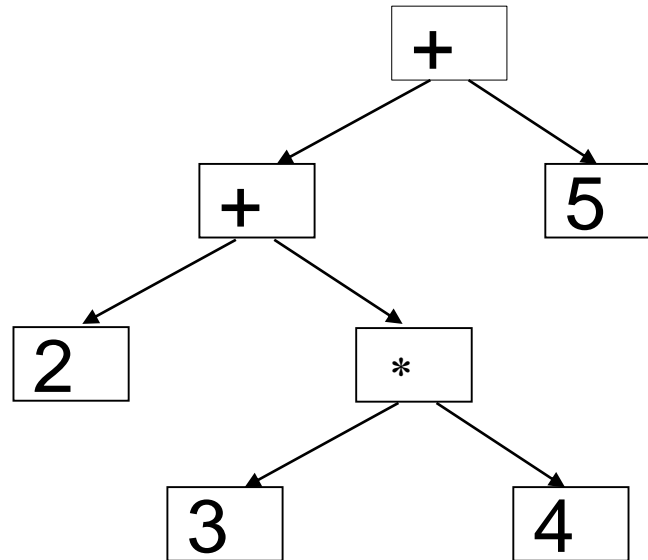
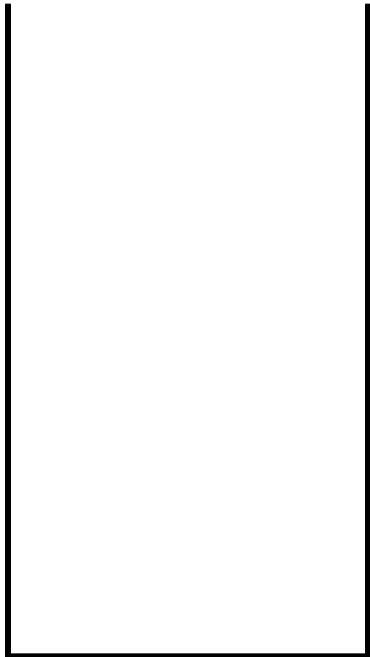
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2

Stos:



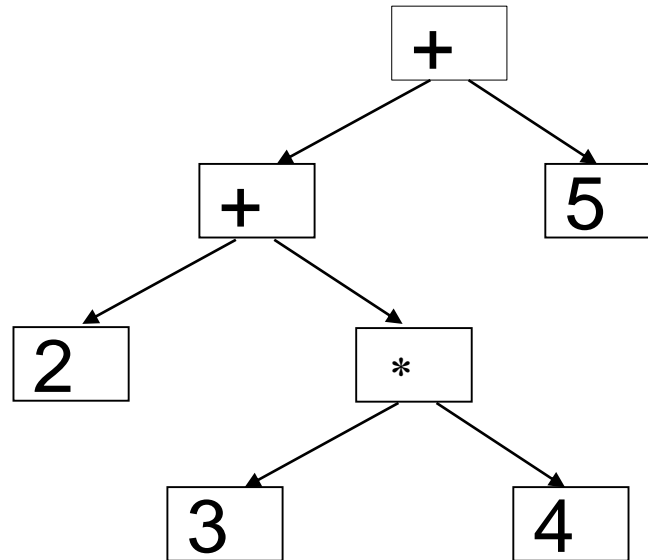
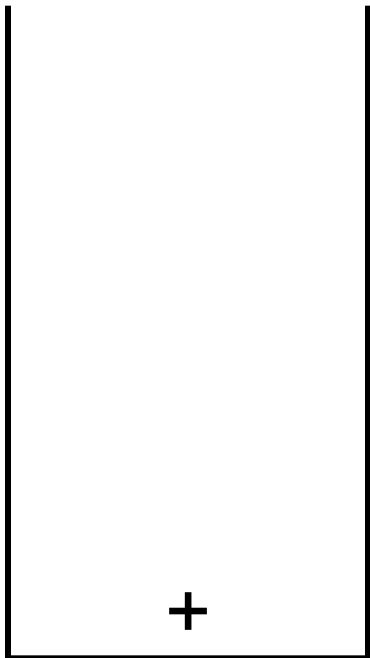
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * * 5 +

Wyjście: 2

Stos:



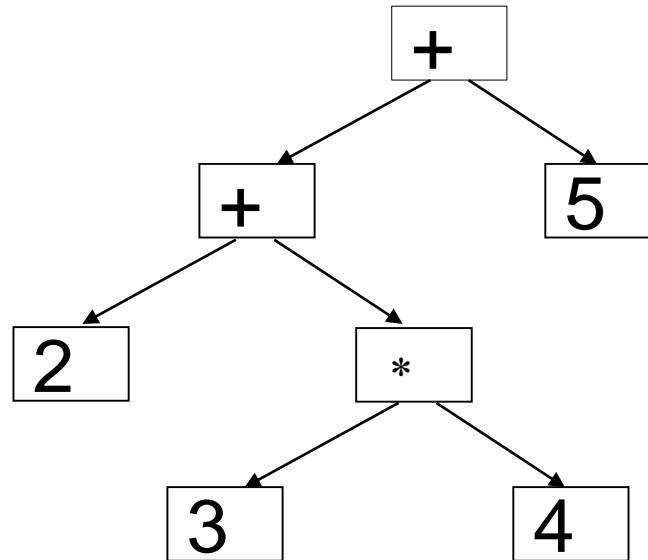
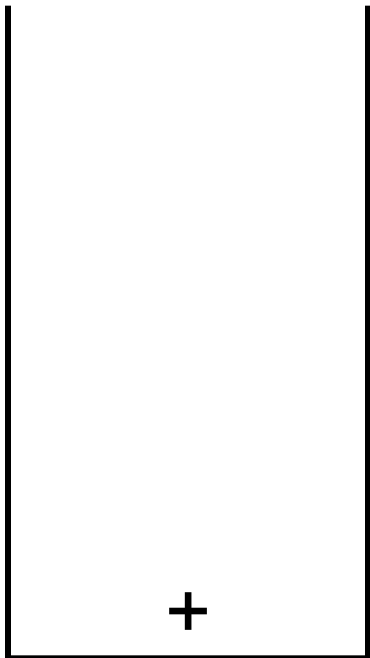
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * * 5 +

Wyjście: 2 3

Stos:



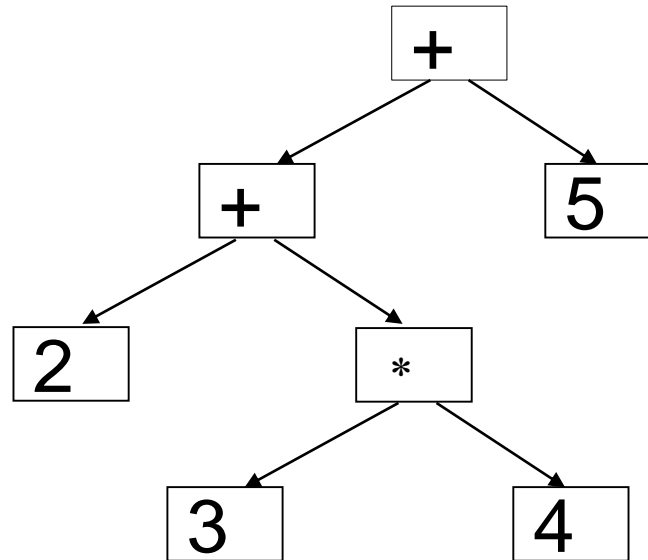
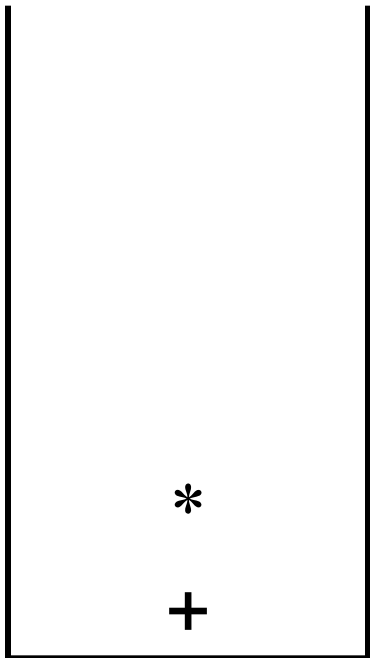
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * * 5 +

Wyjście: 2 3

Stos:



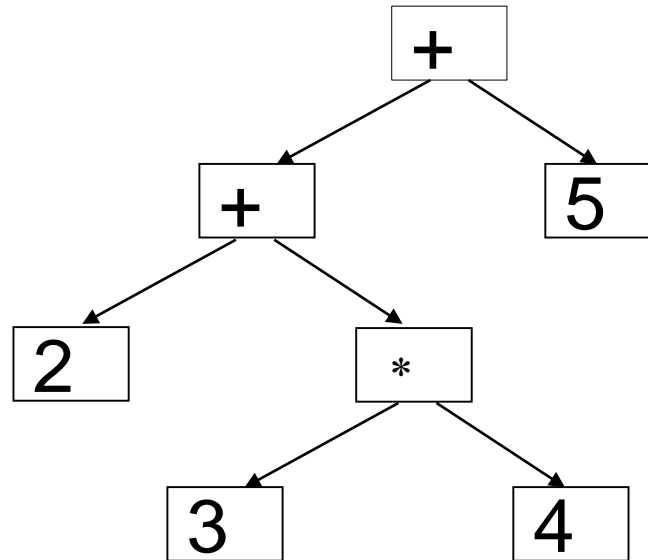
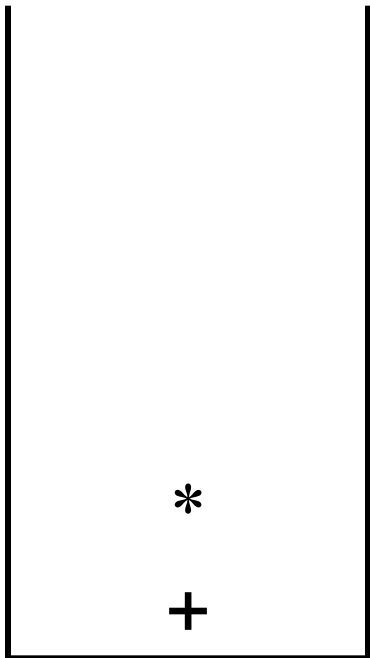
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4

Stos:



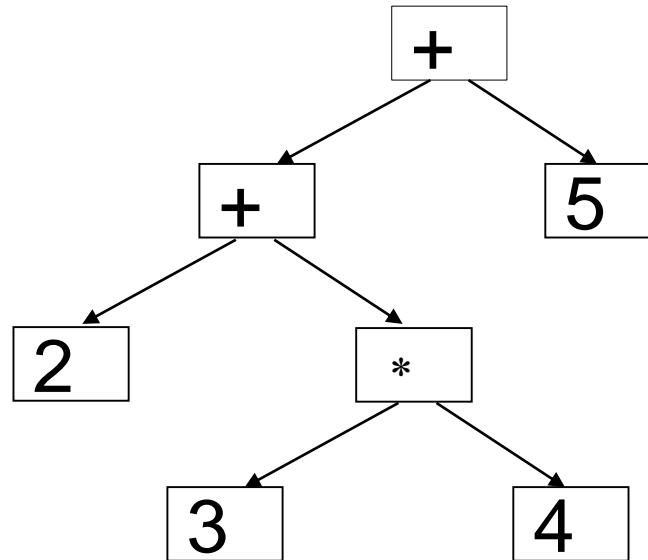
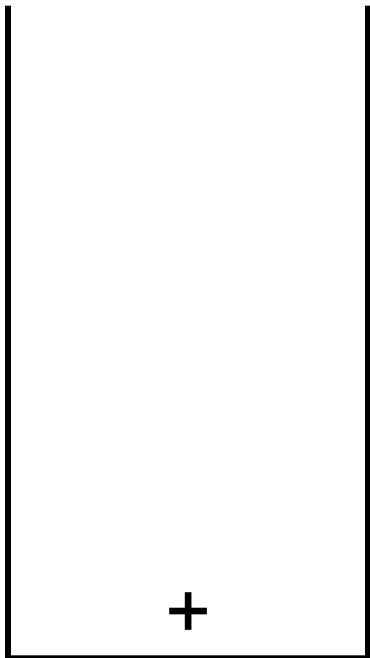
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4 * +

Stos:



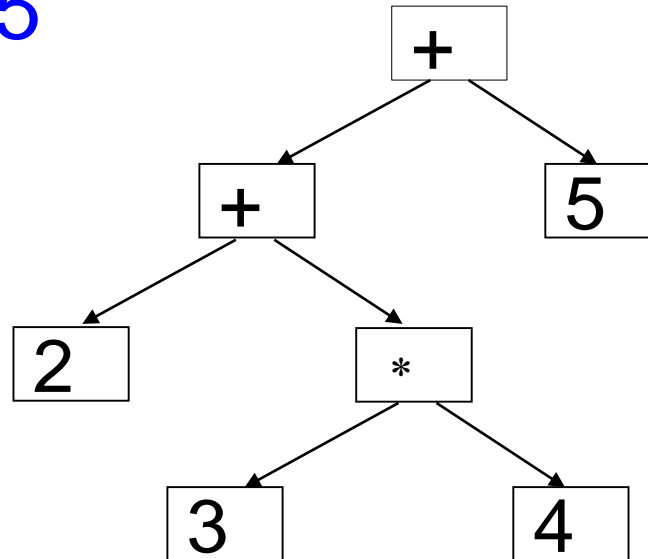
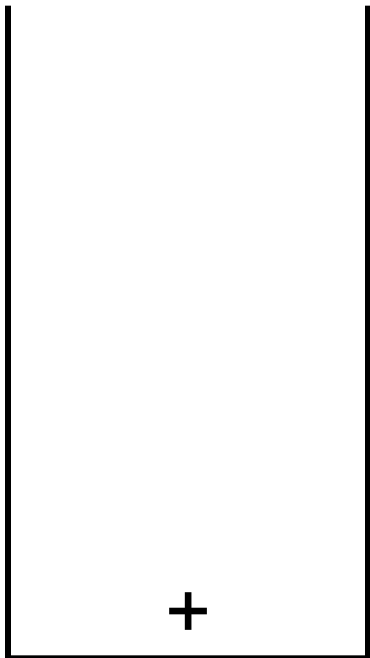
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4 * + 5

Stos:



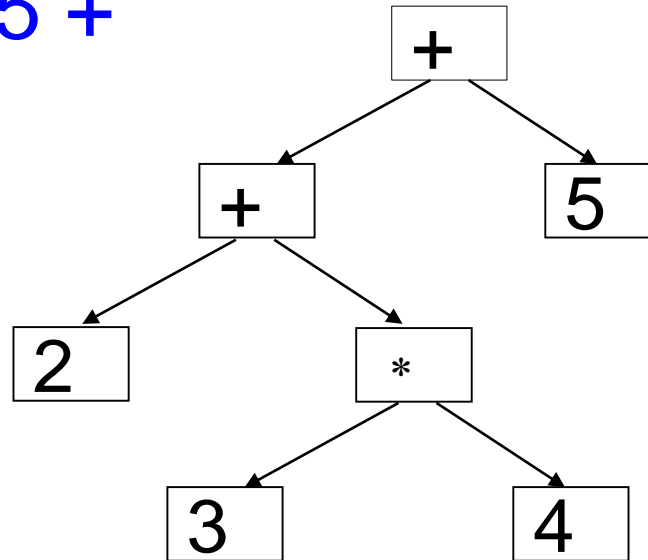
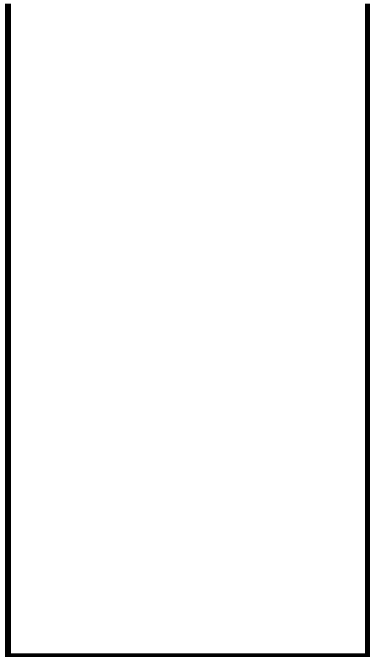
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5$$

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4 * + 5 +

Stos:



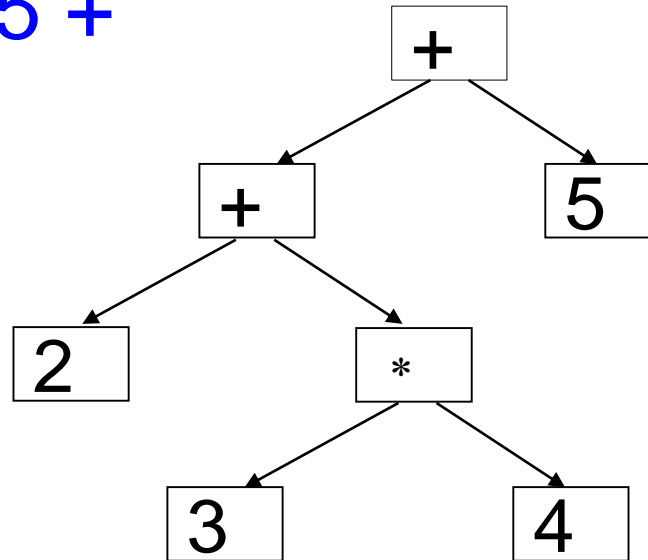
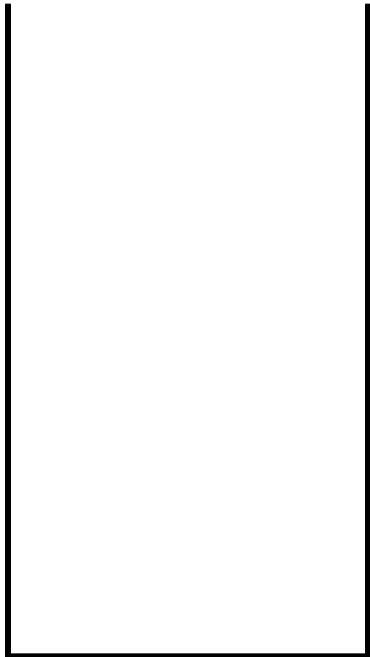
Zamiana na ONP – ver. 2 (z priorytetami)

$$w = 2 + 3 * 4 + 5 = (2 + (3 * 4)) + 5$$

ONP(w) = 2 3 4 * + 5 +

Wyjście: 2 3 4 * + 5 +

Stos:



Podsumowanie ONP

- Prosty algorytm wartościowania wyrażenia;
- Efektywna konwersja wyrażień w formie standardowej do postaci ONP
 - Konwersja bez priorytetów operatorów – ze szkicem dowodu poprawności na wykładzie
 - Konwersja z priorytetami operatorów – bez dowodu poprawności na wykładzie