O nawiasach i ich braku

czyli odwrotna notacja polska

Notacja prefiksowa Jana Łukasiewicza, zwana także notacją polską, oraz jej modyfikacja – odwrotna notacja polska (ONP) są sposobami zapisu wyrażeń matematycznych (arytmetycznych lub logicznych). Słyszeliby o nich tylko specjaliści od logiki matematycznej, gdyby nie rozwój informatyki.

Jarosław Deminet

informatyk od 1979 r., bywał nauczycielem akademickim, urzędnikiem, szefem działów produkujących oprogramowanie w korporacji, konsultantem biznesowym, publicystą. Członek założyciel PTI, obecnie pracownik Rządowego Centrum Legislacji i sekretarz Zarządu Oddziału Mazowieckiego PTI, rzeczoznawca PTI nr 13.

Znany nam dziś sposób zapisu wzorów matematycznych ma kilkaset lat. Znaki + i - pojawiły się po raz pierwszy w podręczniku „Szybki i piękny rachunek dla stanu kupieckiego" w Lipsku w 1489 r. (Johann Widman)[1], ale jeszcze w XVII w. François Viète równanie trzeciego stopnia zapisywał następująco: „A cubus minus Z quadrato ter in A aequatur Z cubo" (chodziło o x3-3r2x=r3). Trudno nam sobie dziś wyobrazić zapisywanie w ten sposób rozbudowanych wzorów, na szczęście teraz korzystamy z wygodniejszej notacji.

Pomysły na jednoznaczność zapisu

Dziś zapiszemy przykładowe wyrażenie matematyczne np. tak:

232/(7+6-9)×2

Podstawowymi elementami łączącymi symbole terminalne (stałe lub zmienne) są tu dwuargumentowe operatory: +, -, ×, / i operator potęgowania. Kolejność wykonania odpowiadających im działań (a właściwie ich hierarchię w wyrażeniu) określają nawiasy, a gdy ich nie ma – zasady łączności i przypisany priorytet. Przyjmujemy, że cztery podstawowe działania wyliczamy w kolejności od lewej do prawej, najwyższy priorytet ma potęgowanie, dalej mnożenie i dzielenie, a najniższy – dodawanie i odejmowanie, tzn.

a-b/c-d = (a-(b/c))-d

W przypadku potęgowania wygodnie jest przyjąć odwrotny kierunek:

abc = a(bc)

ponieważ w przeciwnym przypadku wzór można zapisać po prostu jako iloczyn wykładników:

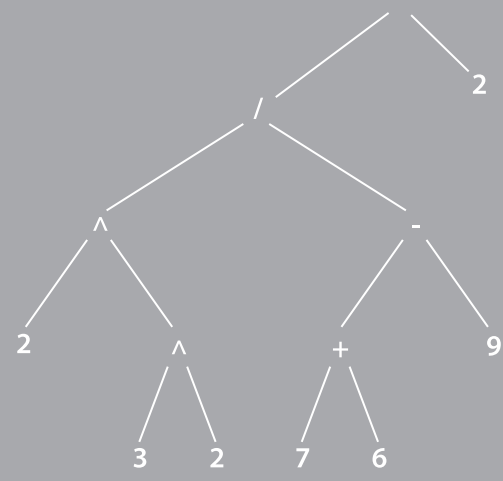
(ab)c = a(b×c)

Po wstawieniu nawiasów wynikających z zasad łączności możemy następująco przepisać powyższe wyrażenie i wyliczyć jego wartość:

((2(32))/((7+6)-9))×2 = ((29)/(13-9))×2 = (512/4)×2 = 128×2 = 256

1 A.P. Juszkiewicz (red.), *Historia matematyki.* PWN, Warszawa 1975.

Uniwersalną i jednoznaczną formą przedstawienia wyrażenia matematycznego jest drzewo, w którym każdym węzłem jest operator, a krawędziami (gałęziami) – jego argumenty. Drzewo naszego przykładowego wyrażenia wygląda następująco (zgodnie z informatyczną tradycją operator potęgowania jest oznaczony symbolem ^):



Można sobie zadać pytanie: czy da się jednoznacznie zapisać takie wyrażenie bez stosowania nawiasów i bez dodatkowych założeń dotyczących priorytetów oraz kierunku łączności operatorów? Odpowiedź jest prosta: tak, jeśli najpierw potraktujemy operatory jako (dwuargumentowe) funkcje, np. +(a, b) będzie oznaczać a+b, i zapiszemy wyrażenie następująco:

×(/(^(2, ^(3, 2)), -(+(7, 6), 9)), 2) = ×(/(^(2, 9), -(13, 9)), 2) = ×(/(512, 4), 2) = ×(128, 2) = 256

Taka notacja funkcyjna jest podstawą języka programowania Lisp.

Okazuje się, że możemy teraz opuścić wszystkie nawiasy, a zapis zachowa jednoznaczność (dla czytelności wprowadzono przecinki, ale można je pominąć):

×, /, ^, 2, ^, 3, 2, -, +, 7, 6, 9, 2

Taka technika jest znana jako prefiksowe przechodzenie (trawersowanie) drzewa – po dojściu do węzła zapisujemy najpierw jego symbol, a następnie po kolei rekurencyjnie przechodzimy przez jego poddrzewa, od lewego do prawego. Łatwo się przekonać, że na podstawie takiego ciągu symboli – znając liczbę argumentów każdej funkcji (w naszym przypadku wynosi ona zawsze dwa) – możemy jednoznacznie odtworzyć całe drzewo. Wystarczy wstawić nawias otwierający po każdym operatorze, a następnie nawias zamykający po zebraniu wymaganej liczby argumentów.

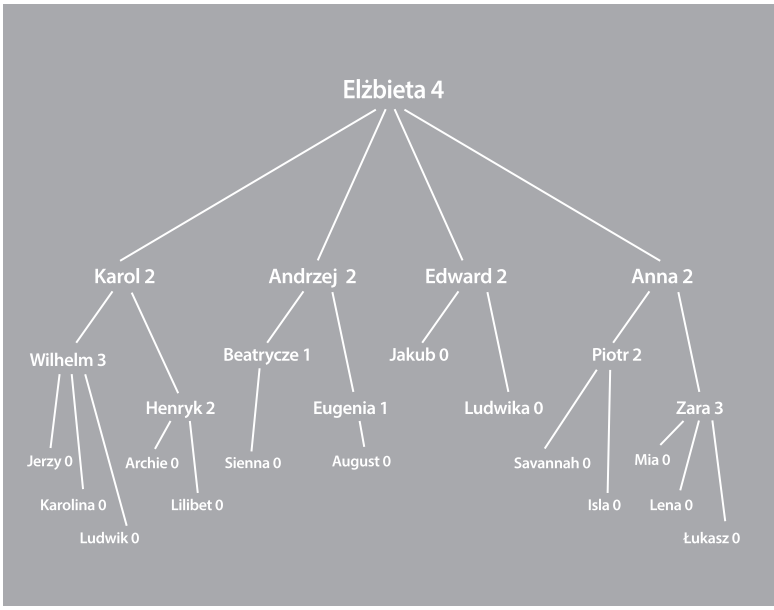
Taki właśnie beznawiasowy sposób zapisu wyrażeń zaproponował prof. Jan Łukasiewicz, polski logik, matematyk i filozof– stąd nazwa „notacja Łukasiewicza" albo „notacja polska" (ale także „notacja prefiksowa"). Dokładniej, Łukasiewicz sformułował notację dla wyrażeń logicznych ze standardowymi operatorami negacji (N), koniunkcji (K), alternatywy (A), implikacji (C) i równoważności (E), a także z modalnymi operatorami możliwości i konieczności oraz kwantyfikatorami:

CKCpqNqNp odpowiada wyrażeniu ((p⇒q)∧ ¬ q) ⇒ ¬ p

ENApqKNpNq odpowiada wyrażeniu ¬(p ∨ q)⇔(¬ p ∧ ¬ q)

Nie tylko w matematyce

Notacja prefiksowa występuje nie tylko w matematyce. Rozpatrzmy poniższy diagram.



Przedstawia on potomstwo królowej Elżbiety II. Aby zapewnić jednoznaczność interpretacji przy każdej osobie w górnym indeksie jest podana liczba poddrzew (czyli żyjących potomków).

Zapis funkcyjny jest następujący:

Elżbieta(Karol(Wilhelm(Jerzy, Karolina, Ludwik), Henryk (Archie, Lilibet)), Andrzej(Beatrycze(Sienna), Eugenia(August)), Edward(Jakub, Ludwika), Anna(Piotr(Savannah, Isla), Zara(Mia, Lena, Łukasz))).

Zapis w notacji polskiej po usunięciu nawiasów określa sukcesję do tronu brytyjskiego: po królowej Elżbiecie będzie panować książę Karol, potem jego najstarszy syn Wilhelm i kolejno jego dzieci. Jeśli Jerzy doczeka się potomstwa, to po nim będzie z kolei jego najstarsze dziecko.

Zapis wyrażenia w notacji polskiej pozwala łatwo zbudować jego drzewo, ale nie jest pomocny przy wyliczaniu wyrażenia. Musimy najpierw zapamiętać działanie do wykonania, a potem zająć się rekurencyjnie wyliczeniem jego argumentów – dopiero po ich wyliczeniu działanie może być wykonane. Lepiej byłoby odwrócić kolejność – najpierw zapisać argumenty, a dopiero po nich symbol działania. Przechodząc w ten sposób przez drzewo po dojściu do węzła najpierw przechodzimy przez jego poddrzewa, a dopiero potem zapisujemy symbol działania. Taka notacja nazywa się właśnie odwrotną notacją polską albo notacją postfiksową i również pozwala na jednoznaczny zapis bez użycia nawiasów. Nasze przykładowe wyrażenie w notacji postfiksowej wygląda następująco:

2, 3, 2, ^, ^, 7, 6, +, 9, -, /, 2, ×

Taki zapis wyrażenia jest od razu przepisem na wyliczenie jego wartości z wykorzystaniem prostej struktury danych – stosu, na który są odkładane częściowe wyniki obliczeń, a działania są zawsze wykonywane na argumentach ze szczytu stosu (w naszym przypadku – zawsze na dwóch, ale ten sam mechanizm można zastosować dla operatorów o dowolnej ustalonej liczbie argumentów). Algorytm obliczania polega na umieszczaniu napotkanych stałych na stosie i wykonywaniu działań określonych przez operatory. Poniższa tabela przedstawia wygląd stosu po przeczytaniu kolejnych symboli, przy czym pogrubione są argumenty na stosie, na których będzie wykonane następne działanie.

Pozostała część wyrażenia	Odczytany symbol	Zawartość stosu
2, 3, 2, ^, ^, 7, 6, +, 9, -, /, 2, x	2	2
3, 2, ^, ^, 7, 6, +, 9, -, /, 2, x	3	2, 3
2, ^, ^, 7, 6, +, 9, -, /, 2, x	2	2, **3, 2**
^, ^, 7, 6, +, 9, -, /, 2, x	^	**2, 9**
^, 7, 6, +, 9, -, /, 2, x	^	512
7, 6, +, 9, -, /, 2, x	7	512, 7
6, +, 9, -, /, 2, x	6	512, **7, 6**
+, 9, -, /, 2, x	+	512, 13
9, -, /, 2, x	9	512, **13, 9**
-, /, 2, x	-	**512, 4**
/, 2, x	/	128
2, x	2	**128, 2**
x	x	256

Pierwszym autorem tej koncepcji był Robert S. Barton, pracujący dla firmy Burroughs, który w 1985 r. zaproponował algorytm przetłumaczenia wyrażeń arytmetycznych zapisanych w klasycznej postaci nawiasowej na kod dla komputera ze stosem. Inspiracją była właśnie praca Łukasiewicza, na którą trafił przypadkowo, przygotowując się do konferencji naukowej[2]. Zaowocowało to wprowadzeniem do architektury komputera B 5000 sprzętowego stosu do obsługi wywołania procedur oraz wyliczania wartości wyrażeń. Bardzo uprościło to kompilowanie i wykonywanie programów napisanych w Algolu i strukturalnej wersji Cobolu. Warto przypomnieć, że w pierwotnym Cobolu nie było w ogóle wyrażeń arytmetycznych, w jednej instrukcji można było zapisać jedno działanie (ADD A TO B).

Jak to zwykle w przypadku takich genialnych pomysłów bywa, niezależnie od Bartona z zapisu postfiksowego na początku lat sześćdziesiątych XX w. korzystały inne osoby, kładące podwaliny pod budowę kompilatorów komputerowych, m.in. Charles L. Hamblin, Friedrich L. Bauer i Edsger W. Dijkstra. Dijkstra zaprojektował algorytm przetłumaczenia tradycyjnego zapisu na ONP, korzystający z tzw. gramatyki z pierwszeństwem, czyli specjalnej kategorii gramatyk bezkontekstowych.

Tłumaczenie wyrażeń na ONP jest do dzisiaj jednym z elementów działania kompilatorów, choć – ze względu na efektywność – po późniejszej optymalizacji częściowe wyniki obliczeń są w miarę możliwości przechowywane w rejestrach procesora, a nie na stosie w pamięci (w praktyce dość rzadko spotyka się wyrażenia, przy których wyliczaniu jest potrzebny bardzo wysoki stos). Stos jest także częścią maszyny wirtualnej wykonującej programy zapisane w Javie, a kompilator Javy tłumaczy wyrażenia matematyczne na ONP.

Zapewne notacja prefiksowa byłaby wymyślona bez wcześniejszych prac Łukasiewicza, bo taki sposób traktowania wyrażeń jest dość oczywisty, ale to właśnie Łukasiewicz jako pierwszy dostrzegł zalety tego typu rozwiązania i należy się cieszyć, że późniejsi autorzy to docenili. Jest to przede wszystkim zasługa Bartona – pozostali zapewne nawet nie wiedzieli o istnieniu Łukasiewicza i o jego pomyśle…

[2] Relacja w sprawozdaniu University of Minnesota Burroughs B 5000 Conference, 1985, Charles Babbage Institute https://conservancy.umn.edu/handle/11299/107105