|  |  |  |
| --- | --- | --- |
| **Politechnika Świętokrzyska w Kielcach** | | |
| **Algorytmy i Struktury Danych- Projekt** | | |
| Grupa:  **1IZ12B** | Temat:  **Ewaluowanie wartości wyrażeń z użyciem odwrotnej notacji polskiej.** | Skład:  **Aleksandra Więckowicz**  **Jakub Wojtaluk**  **Andrzej Zigliński**  **Karol Bernat** |

**Wybrane sposoby zapisu wyrażeń arytmetycznych**

1. **Notacja infiksowa** - konwencjonalny zapis arytmetyczny, to notacja, którą posługujemy się na co dzień. Składa się ona z dwóch operand (liczb, zmiennych) oraz operatora, który znajduje się między nimi. Kolejność wykonywania działań w standardowej notacji zależy od:
   1. *Priorytetów operatorów*. Operacje multiplikatywne (np. \*, /) wykonujemy przez addytywnymi (np. +, -). Potęgowanie (^) wykonujemy przed operacjami multiplikatywnymi.
   2. *Łączności operatorów*. Operacje tego samego typu wykonujemy zgodnie z ich łącznością. Operatory +, \* są obustronnie łączne. Operatory −, / są łączne lewostronnie, ale nie prawostronnie. Operacja potęgowania jest łączna prawostronnie, ale nie lewostronnie (a ^ b ^ c= a ^ (b ^ c)).
   3. *Nawiasów*.
2. **Notacja polska**- zapis prefiksowy. Sposób zapisu wyrażeń logicznych i arytmetycznych, podający najpierw operator, a potem operandy (argumenty), który został przedstawiony w [1920](https://pl.wikipedia.org/wiki/1920) przez Jana Łukasiewicza. Notacja ta pozwala na łatwiejsze przeprowadzanie operacji na formułach o znacznej długości. Obecnie [informatyka](https://pl.wikipedia.org/wiki/Informatyka) jest jedynym polem, gdzie notacja ta jest wciąż popularna. Wyrażenie w notacji polskiej nie wymaga nawiasów, ponieważ przypisanie argumentów do operatorów wynika wprost z ich kolejności w zapisie, o ile z góry znana jest liczba argumentów poszczególnych operatorów.
3. **Odwrotna notacja polska-** zapis postfiksowy. Sposób zapisu wyrażeń arytmetycznych, w którym znak wykonywanej operacji umieszczony jest po operandach. Zapis ten pozwala na całkowitą rezygnację z użycia nawiasów w wyrażeniach, jako że jednoznacznie określa kolejność wykonywanych działań. Odwrotna notacja polska została opracowana przez Charlesa Hamblina jako „odwrócenie” notacji polskiej na potrzeby zastosowań informatycznych. ONP ma sporo zalet nad zwykłą notacją: brak nawiasów, brak priorytetów operatorów (działania można wykonywać od lewej do prawej), łączność operacji jest nieistotna.

**Przykładowy zapis wyrażenia arytmetycznego przy użyciu różnych notacji:**

|  |  |  |
| --- | --- | --- |
| **Zapis infiksowy** | **Zapis prefiksowy** | **Zapis postfiksowy** |
| **7/(2+3)** | **/7+2 3** | **7 2 3 + /** |

**Zastosowanie ONP**

Na przestrzeni lat zastosowania ONP ewoluowało od kalkulatorów elektronicznych ku informatyce. Obecnie użytkownik komputera właściwie ma niewielkie szanse na bezpośredni kontakt z ONP, chyba że jest użytkownikiem specjalistycznych aplikacji.

W komputerach wszelkie wyrażenia w zrozumiałej dla człowieka postaci wrostkowej są konwertowane do bez nawiasowej postaci przyrostkowej. Konwersja do ONP jest przeprowadzana podczas wczesnej fazy kompilacji programów napisanych w popularnych językach wysokiego poziomu w trakcie analizy składni leksykalnej kodu źródłowego, tzw. parsery. Istnieją też takie języki, dla których ONP jest natywnym sposobem opisu wyrażeń. Przykładem takiego języka jest Forth, charakteryzujący się natychmiastową interpretacją wprowadzonych słów i wykonywaniem kodu w konwencji odwrotnej notacji polskiej właśnie z użyciem stosu.

ONP jest powszechnie stosowana w kompilatorach języków HLL do obliczania wartości wyrażeń arytmetycznych, co spowodowane jest prostotą obliczeń w porównaniu do zwykłej notacji. Algorytm obliczania wartości wyrażenia ONP wykorzystuje stos do składowania wyników pośrednich wyrażeń arytmetycznych zapisanej w zwykłej notacji.

Odwrotna notacja polska znalazła także zastosowanie w projektowaniu komputerowych układów logicznych. We wczesnej fazie działania algorytmu wykorzystywane są właściwości ONP (pozbywanie się nawiasów i konwertowanie wejściowego wyrażenia logiczne do postaci przyrostkowej). Upraszcza to kolejne kroki poszukiwania minimalnej postaci opisu formalnego projektowanego układu, na podstawie którego tworzona jest sieć połączeń funktorów logicznych.

DZIAŁANIE ALGORYTMU ONP

Omówienie algorytmu konwersji wyrażeń algebraicznych przedstawionych w notacji infiksowej na postać Odwrotnej Notacji Polskiej:

1. Wczytujemy od użytkownika pełny zapis w notacji infiksowej. Zapisujemy go w tablicy zmiennych typu char. *char zapis[]*
2. Definiujemy kolejność wykonywania działań. Aby tego dokonać, tworzymy kolejną tablicę typu char zapisując w niej wykorzystywane operatory matematyczne zgodnie z kolejnością wykonywania działań. Umieszczamy w niej zatem operator mnożenia, dzielenia, dodawania oraz odejmowania (w tej kolejności). Tablica ta przedstawiona jest w tabeli poniżej:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | \* | / | + | - |
| Indeks | 0 | 1 | 2 | 3 |

*Tabela 1. char operatory[]*  
Działania będą zatem wykonywane w kolejności: mnożenie, dzielenie, dodawanie, odejmowanie.

1. Definiujemy tablicę zmiennych typu char, w której będziemy przechowywać wynik konwersji. Tablicę tę pozostawiamy póki co pustą.
2. Definiujemy tablicę operacji, również zmiennych typu char, również pozostawiamy ją pustą.
3. Tworzymy pętlę for, która ma wykonać się *n* razy, gdzie *n* to długość tablicy zapis. Iterujemy po kolejnych znakach z tablicy zapis.
4. Jeżeli znakiem z tablicy zapis jest cyfra (od 1 do 9), umieszczamy ją na końcu tablicy *wynik*.
5. Jeżeli znakiem z tablicy zapis jest otwarcie nawiasu ‘(’, umieszczamy go na końcu tablicy *operacje*.
6. Jeżeli znakiem z tablicy jest operator algebraiczny z tablicy *operatory*:  
   a) jeżeli tablica *operacje* nie zawiera żadnych elementów, zapisujemy w niej ten operator.  
   b) jeżeli tablica *operacje* zawiera już elementy, to sprawdzamy jej ostatni element.   
   c) Jeżeli jest to element z tablicy *operatory*, musimy sprawdzić kolejność wykonywania działań. Jeżeli analizowany z pętli operator ma wyższy priorytet (jego indeks w tablicy *operatory* jest niższy), do tablicy *wynik* dopisujemy ostatni element z tablicy *operacje*, a na jego miejsce wpisujemy analizowany aktualnie w pętli operator. Jeżeli natomiast analizowany aktualnie w pętli operator ma niższy (lub równy) priorytet od ostatniego operatora z tablicy *operacje*, dołączamy analizowany operator na koniec tablicy *operacje.*d)jeżeli ostatnim elementem tablicy *operacje* nie jest element z tablicy *operatory*, bierzący analizowany w pętli operator dodajemy na koniec tablicy *operacje*.
7. Jeżeli jednak nie jest to element z tablicy *operatory*, to pozostała nam tylko ostatnia możliwość, że jest to nawias domykający ‘)’. (w przeciwnym wypadku wprowadzone przez użytkownika wyrażenie infiksowe jest nieprawidłowe). Zatem w przypadku znalezienia nawiasu domykającego tworzymy pętlę *while* (warto zaznaczyć, iż wciąż znajdujemy się wewnątrz pętli *for*, zatem mamy tu do czynienia z zagnieżdżeniem pętli w pętli). Pętla *while* ma iterować po tablicy *operacje* od jej ostatniego elementu (od końca) aż do napotkania znaku nawiasu otwierającego, dodając do tablicy *wynik* wszystkie napotkane po drodze elementy i usuwając je. Gdy napotkamy znak ‘(’, nie wypisujemy go, ale usuwamy go z tablicy. Jeżeli wyczerpiemy wszystkie elementy z tablicy *operacje* nie napotkawszy znaku nawiasu otwierającego, wyrażenie infoksowe wprowadzone przez użytkownika również jest nieprawidłowe (oznacza to po prostu że użytkownik zamknął nawias, którego nigdy nie otworzył).
8. W ten sposób analizujemy każdy element z tablicy *zapis*.
9. Po zakończeniu wykonywania pętli *for* sprawdzamy, czy w tablicy *operacje* są jeszcze jakieś elementy. Jeżeli tak, dopisujemy je do tablicy *wynik* od końca i kolejno usuwamy. Robimy to dopóki tablica *operacje* nie pozostanie pusta (pętla *while*).

Wypisujemy na ekran tablicę *wynik*, tym razem już począwszy od indeksu 0. Jest to wyrażenie przekształcone do zapisu ONP.